Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

Ingeniería Mecatrónica

Estudio Experimental de Generación de Números Aleatorios Basado en Circuitos Caóticos

Tesis

presentada como requisito parcial para obtener el título de:

Ingeniero en Mecatrónica

Presenta:
Tonatiuh García Chávez

Asesores
Dr. Jesús Manuel Muñoz Pacheco (BUAP)
Dra. Luz del Carmen Gómez Pavón (BUAP)
Dr. Esteban Tlelo Cuautle (INAOE)

Diciembre de 2014

Agradecimientos

Agradezco a mis asesores de tesis, el Dr. Jesús Manuel Muñoz Pacheco, el Dr. Esteban Tlelo Cuautle y a la Dra. Luz del Carmen Gómez Pavón, su confianza y conocimientos han hecho posible la realizacion de este proyecto.

Al comité revisor, el Dr. Arnulfo Luis Ramos, el Dr. Victor González Díaz y el Dr. Héctor Santiago Ramírez, cuyas observaciones han enriquecido este trabajo de tesis.

Agradezco a los proyectos de los que ha formado parte esta investigación, así como las becas que han permitido que llegue a su fin, proyecto VIEP 2013, proyecto PROMEP BUAP-PTC-359 y proyecto CONACYT 131839-Y. Agradezco también al INAOE por las facilidades prestadas para la realización de esta tesis.

A la Facultad de Ciencias de la Electrónica de la BUAP por los años de formación brindados, a mis profesores por la excelencia y calidad humana, a mis compañeros y amigos.

Índice general

Agradecimientos I

Introducción V

1. Generadores de Números Aleatorios 1
- 1.1. Números Aleatorios . . . 1
- 1.2. Generador de Números Aleatorios . . . 2
- 1.3. Clasificación de Generadores de Números Aleatorios . . . 3
  - 1.3.1. Generador de Números Verdaderamente Aleatorios (TRNG) . . . 3
  - 1.3.2. Generador de Números Pseudo-Aleatorios (PRNG) . . . 4
- 1.4. Comparación Entre TRNG y PRNG . . . 6

2. Dinámica de Osciladores Caóticos 8
- 2.1. Conceptos Básicos de Caos . . . 8
  - 2.1.1. Sensibilidad a Condiciones Iniciales . . . 8
  - 2.1.2. Respuesta Temporal . . . 9
  - 2.1.3. Espectro de Potencia . . . 9
  - 2.1.4. Condiciones Necesarias para Generar Caos . . . 9
- 2.2. Exponente de Lyapunov . . . 11
- 2.3. Entropía . . . 13
- 2.4. Entropía de Shannon . . . 14
- 2.5. Entropía de Kolmogorov-Sinai: Complejidad de los Sistemas Caóticos . . . 15

3. Síntesis de Osciladores Caóticos a Nivel Experimental 18
- 3.1. Oscilador Caótico de Chua . . . 18
  - 3.1.1. Síntesis del Oscilador Caótico de Chua Usando OpAmps . . . 21
  - 3.1.2. Realización Experimental del Oscilador Caótico de Chua . . . 23
- 3.2. Oscilador Caótico de Funciones Saturadas . . . 24
  - 3.2.1. Síntesis del Oscilador Caótico de Funciones Saturadas Usando OpAmps . . . 29
  - 3.2.2. Realización Experimental del Oscilador Caótico de Funciones Saturadas . . . 32
- 3.3. Oscilador Caótico de Función Diente de Sierra . . . 32
  - 3.3.1. Síntesis del Oscilador Caótico de Funcion Diente de Sierra Usando OpAmps 36
  - 3.3.2. Realización Experimental del Oscilador Caótico de Diente de Sierra . . . 39
- 3.4. Escalamiento de la Frecuencia . . . 40

4. Diseño e Implementación de un TRNG basado en Caos **43**

5. Estudio sobre la Generación de Números Aleatorios **51**
5.1. Optimización del Máximo Exponente de Lyapunov . . . 51
5.2. Determinación de la Entropía de los Atractores Caóticos . . . 51
5.3. Test para Estimar la Aleatoriedad del TRNG . . . 54
5.3.1. Test de Frecuencia (Monobit) . . . 55
5.3.2. Test de Frecuencia Dentro de un Bloque . . . 55
5.3.3. Test de Ejecución . . . 57
5.3.4. Test de la Mayor Ejecución Dentro de un Bloque . . . 57
5.3.5. Test de la Transformada Discreta de Fourier . . . 57
5.3.6. Test Serial . . . 58
5.3.7. Test de Entropía Aproximada . . . 58
5.3.8. Test de Sumas Acumuladas . . . 58
5.4. Resultados de los Test de Aleatoriedad . . . 58
5.5. Discusión de Resultados . . . 64
5.5.1. Oscilador Caótico de Chua . . . 65
5.5.2. Oscilador Caótico de Función Saturada . . . 66
5.5.3. Oscilador Caótico de Diente de Sierra . . . 66

6. Conclusiones **68**
6.1. Conclusiones Generales . . . 68
6.2. Trabajo a Futuro . . . 69

A. Test de Aleatoriedad del NIST **70**
A.1. Frecuencia (Monobit) . . . 70
A.2. Frecuencia Dentro de un Bloque . . . 71
A.3. Corridas . . . 72
A.4. Mayor Corrida Dentro de un Bloque . . . 73
A.5. Rango de Matriz Binaria . . . 74
A.6. Transformada Discreta de Fourier . . . 75
A.7. Complejidad Lineal . . . 76
A.8. Serial . . . 77
A.9. Entropía Aproximada . . . 79
A.10. Sumas Acumuladas . . . 80

B. Evidencias de implementación **83**

Bibliografía **87**

A mis padres que me han apoyado durante todo este tiempo...

Introducción

El concepto de aleatoriedad se ha ocupado desde el pasado para fines específicamente militares donde se incorporaba como una herramienta crucial para la seguridad de datos encriptados [1]. Desde el punto de vista de la ingeniería, la aleatoriedad se aplica mediante generadores de bits aleatorios, los cuales representan el núcleo básico de las aplicaciones de criptografía debido a que se utilizan para la generación de las claves secretas para diferentes aplicaciones; tales como, aplicaciones para la generación de valores blindados y enmascarados en sistemas de comunicación, generación de semillas en algoritmos evolutivos, sistemas de simulación estadísticos, protocolos de identificación en radiofrecuencia y manipulación de robots autónomos [2]-[3].

La eficiencia de cualquier esquema de encriptación y su aplicación asociada, además de protocolos de comunicación robustos, requieren forzosamente de un generador de bits aleatorios completamente seguro [4].Un generador de bits aleatorios es un sistema capaz de procesar algún tipo de fuente de señal eléctrica para generar, a la salida, una palabra digital de n bits, los cuales deben ser aleatorios. En estadística, un número aleatorio es un resultado de una variable al azar donde todos sus componentes tienen la misma probabilidad de ser escogidos, por lo tanto el resultado no puede ser determinado antes de que éste se produzca. Para verificar que el resultado es aleatorio existen diversas pruebas estándares que deben ser satisfechas, una de las más conocidas en es la prueba denominada Monobit, la cual define la cantidad mínima de valores (unos y ceros) existentes en un tren de pulsos digitales mediantes una distribución estadística acotada [1],[5].

De lo anterior, se remarca que la generación de los bits aleatorios satisfactorios depende fuertemente de la fuente de señal eléctrica utilizada, debido a que la aleatoriedad de los bits de salida no es mayor que la aleatoriedad que tiene la señal de entrada [6]. Por tal motivo, la investigación actual respecto a la generación de números aleatorios se enfoca en estudiar en como mejorar las fuentes de señal eléctricas. En este contexto, los sistemas caóticos se han propuesto como una solución factible para generar la fuente de señal eléctrica a utilizarse en la generación de números aleatorios. Un sistema caótico es una sub-clasificación de los sistemas dinámicos no lineales en tiempo continuo, cuya evolución puede ser descrita en función del tiempo por

alguna variable continua. La teoría del caos demuestra que una relación determinística simple pero no lineal, puede generar una trayectoria dinámica extremadamente compleja que parece ser completamente aleatoria [7]. Los tres identificadores del caos son [8], [9]:

- Un comportamiento de un sistema determinista, esto es, la evolución actual es consecuencia del estado previo del sistema. Significando que, siempre se llega al mismo punto de equilibrio en el plano de fase a partir de una condición inicial definida, lo que significa que la dinámica de ese sistema es reproducible y por lo tanto controlable.
- Un sistema que es sensible a las condiciones iniciales. Esto es, una mínima variación de sus condiciones iniciales produce evoluciones temporales diferentes de las trayectorias.
- Comportamiento aperiódico a largo término; significa que las trayectorias no tienden a un punto fijo, órbitas periódicas u órbitas cuasi-periódicas conforme el tiempo tiende a infinito.

Durante la última década, la investigación de los sistemas caóticos se ha incrementado por el interés en novedosas y variadas aplicaciones potenciales; por ejemplo: sistemas biológicos (corazón, cerebro y técnicas de detección de cáncer), toma de decisiones críticas en eventos económicos, mezclado de líquidos, robótica, circuitos y sistemas electrónicos (moduladores sigma-delta, generadores de números aleatorios), y sistemas modernos de comunicaciones para el aumento del ancho de banda y de la seguridad en la encriptación de información. Un punto importante de la definición de caos es que la evolución temporal de un sistema caótico se vuelve impredecible a largo término sin importar que éste sea determinista.

Debido a este tipo de comportamiento, en esta tesis se propone utilizar un oscilador caótico como la fuente de señal eléctrica para el diseño de un generador de números aleatorios.

Existen dos tipos de generadores de números aleatorios; verdaderos generadores y pseudo-generadores, la diferencia depende del tipo de fuente de señal eléctrica utilizada. Los generadores de números verdaderamente aleatorios utilizan fuentes de señal reales de aleatoriedad, mientras que los pseudo-generadores emplean una fuente de señal (semilla) para inicializar y calcular los valores requeridos mediante un algoritmo preestablecido [6], [10]. Para este trabajo de tesis se propone diseñar un generador de números verdaderamente aleatorios. Dentro de las técnicas desarrolladas para la obtención de números aleatorios se han propuesto sistemas tanto de software como de hardware para producir verdaderos números aleatorios. Sin embargo, los sistemas de este tipo están limitados a la función o regla que los rige, resultando generalmente en un pseudo-generador de números aleatorios. Esta desventaja se reduce cuando el núcleo de

la arquitectura del generador de números aleatorios se substituye por un oscilador caótico, proporcionando así, la plataforma necesaria para desarrollar un verdadero generador de números aleatorios [6], [10], [11].

Además, las arquitecturas capaces de generar números verdaderamente aleatorios, por la necesidad de utilizar como fuente de aleatoriedad un sistema físico, realizan por separado la fuente de aleatoriedad. Esto complica el diseño electrónico respecto al factor de forma por lo que regularmente se prefiere la implementación de un pseudo-generador. Esto es una desventaja; por ejemplo, si la aplicación está enfocada en la seguridad de la información el utilizar un generador pseudo-aleatorio para generar la clave secreta resultará en un sistema con una pseudo-seguridad [2], [6], [11].

El estudio propuesto para esta tesis es diseñar un generador de números aleatorios del tipo verdadero considerando la frecuencia máxima para generación del tren de pulsos (bits) y la calibración de los parámetros del sistema caótico. Ambas consideraciones deber ser satisfechas en el diseño e implementación del oscilador caótico. La ventaja de utilizar un sistema caótico en tiempo continuo (analógico) para el diseño de números aleatorios, es que permite utilizar todo el rango dinámico de la señal caótica para generar las claves de cifrado [5], [6].

Aunque existen trabajos previos de generadores de bits aleatorios basados en la misma hipótesis, [10], [11], en estos trabajos no se ha considerado incrementar la cantidad de aleatoriedad que el sistema caótico puede proporcionar, la cual se define mediante el exponente máximo de Lyapunov (EML) y la entropía. En este trabajo de tesis se busca realizar un estudio del impacto al utilizar valores máximos de entropía y EML en el diseño de un generador de números aleatorios. La tarea de optimización se basa en un algoritmo evolutivo previamente desarrollado en [12].

El objetivo general de este trabajo de Tesis es:

Implementar experimentalmente un generador de números aleatorios utilizando osciladores caóticos teniendo exponentes de Lyapunov optimizados.

Teniendo como objetivos específicos:

1. Modelar y simular el oscilador caótico de Chua, el oscilador caótico de funciones saturadas y el oscilador caótico de diente de sierra.
2. Implementar los tres osciladores caóticos utilizando dispositivos electrónicos.
3. Partiendo de parámetros optimizados de los osciladores, seleccionar el que presente un exponente positivo de Lyapunov máximo y distribución uniforme de los enrollamientos en el plano de fase.

4. Evaluar la entropía de los tres osciladores caóticos optimizados del objetivo anterior.

5. Seleccionar las combinaciones donde se satisfaga un exponente de Lyapunov y una entropía de magnitudes grandes para implementar un generador de números aleatorios.

6. Implementar una plataforma de generación de números aleatorios para el estudio de los tres osciladores caóticos utilizados como fuente de señal de forma independiente.

7. Caracterizar los resultados del generador de números aleatorios para cada sistema caótico mediante la generación de trayectorias de movimiento de un robot móvil.

Este documento de Tesis se organiza de la siguiente manera: en el capítulo 1 se presenta la teoría relativa a los generadores de números aleatorios, tanto del tipo pseudo-generador como verdadero generador, además de una comparación de los mismos; en el capítulo 2 se presenta el modelo matemático del oscilador caótico de Chua, el oscilador de funciones saturadas y el oscilador de diente de sierra. Posteriormente, en el capítulo 3, se sintetizan utilizando amplificadores operacionales y se demuestra mediante realizaciones experimentales comparadas con las simulaciones en MATLAB y Spice. En el capítulo 4 se definen el exponente de Lyapunov y la entropía, además se presenta la arquitectura propuesta del generador de números aleatorios y su teoría de funcionamiento. El capítulo 5 presenta los resultados experimentales del estudio realizado considerado la generación de números aleatorios a partir de tres tipos de osciladores caóticos, los cuales están optimizados en exponente de Lyapunov y entropía. Además, de los resultados obtenidos de las pruebas estandar de validación de números aleatorios definidas por el Instituto Nacional de Estándares y Tecnología (NIST, National Institute of Standards and Technology, por sus siglas en inglés). Finalmente, en el capítulo 6 se dan las conclusiones generales del trabajo de Tesis y el trabajo a futuro.

Capítulo 1

Generadores de Números Aleatorios

1.1. Números Aleatorios

En el siglo XVII, el francés Anoite Gombauld, evidenció el fundamento matemático del éxito y fracaso en las mesas de juego mediante la formulación de la siguiente pregunta al matemático francés Blaise Pascal, "¿Cuáles son las posibilidades de que caigan dos seis por lo menos una vez en veinticuatro lanzamientos de un par de dados?". El problema fue resuelto por Pascal mediante la teoría de probabilidad. Ambos compartieron sus ideas con el matemático Pierre de Fermat, y las cartas escritas por los tres constituyeron la primera revista científica dedicada a la teoría de la probabilidad [6]. Referente a la historia de números aleatorios, esta tiene sus orígenes en la década de los cuarenta con el método de Montecarlo basado en los trabajos pioneros de Von Neumann y Lehmer. Antes del advenimiento de las computadoras, los números aleatorios eran generados por dispositivos físicos, por ejemplo en 1939, Kendall y Babington-Smith publicaron $100,000$ dígitos aleatorios obtenidos por un disco giratorio iluminado con una lámpara relámpago [6].

Los números aleatorios pueden ser generados a partir de fuentes de aleatoriedad, las cuales generalmente, son de naturaleza física (dados, ruletas, mecanismos electrónicos o mecánicos). Estos exhiben verdadera aleatoriedad en la realización de experimentos. Por su parte los números pseudo-aleatorios son aquellos que tienen un comportamiento similar a la naturaleza aleatoria, pero están acotados a un patrón, generalmente de naturaleza matemática, causando un comportamiento determinístico [8].

1.2. Generador de Números Aleatorios

Un número aleatorio, en realidad, por si solo no existe; por ejemplo: ¿Es 1 un número aleatorio? En lugar de eso, se pueden definir secuencias de números aleatorios independientes. La aleatoriedad no es una característica propia de un número, sino de éste en relación a otros, es decir, la aleatoriedad es una característica que posee una serie de números [9].

Una secuencia aleatoria puede ser interpretada como el resultado del lanzamiento de una moneda, sin ninguna tendencia, cuyos lados están etiquetados "0" y "1", donde cada lanzamiento tiene exactamente la probabilidad del 50 % de producir "0" o "1". Además, debido a que los lanzamientos son independientes entre sí, el resultado de cualquier lanzamiento previo no afectará a ningún intento posterior. Debe existir completa impredecibilidad tanto para intentos posteriores como intentos anteriores. Por lo tanto, los lanzamientos sin ninguna tendencia de una moneda definen un generador perfecto de números aleatorios debido a que sus valores estarán aleatoriamente distribuidos [7].

Aunque la utilización de una moneda para generar números aleatorios podría tener un resultado satisfactorio, desde el punto de vista de ingeniería es impráctico. Se utiliza simplemente para definir el concepto de un generador de números aleatorios [9]. Un generador de números aleatorios debe satisfacer las siguientes propiedades fundamentales:

- **Uniformidad:** Si el intervalo $(0, 1)$ se divide en n clases o subintervalos de igual longitud, el número esperado de observaciones en cada sub intervalo es N/n, donde N es el número total de observaciones.

- **Independencia:** La probabilidad de observar un valor en un determinado subintervalo es independiente de las anteriores, es decir, que la obtención de cierto valor no está condicionado por los valores obtenidos anteriormente.

Asociadas a estas, en [10] se definen cinco características sobre las que se debería valorar cualquier generador de números aleatorios, estas se listan a continuación:

- **Calidad:** Esta se refiere a que se deben cumplir con las pruebas estadísticas adecuadas.

- **Eficiencia:** El generador de números aleatorios debe ser rápido, es decir su frecuencia de muestreo debe ser alta, además de requerir la menor cantidad de almacenamiento de datos posible.

- **Transportable:** Se refiere a la reproducibilidad de los resultados en diferentes bancos de prueba (benchmark), es decir, si un método propuesto se implementa sobre sistemas diferentes estos deben de generar resultados similares.

- **Simplicidad:** El generador de números aleatorios debe ser relativamente simple de implementar y utilizar en relación con la aplicación.

- **Repetible:** El algoritmo para la generación de números aleatorios debe depender de una semilla para inicializarlo, la cual debe permitir realizar un experimento repetible. Sin embargo, esta característica únicamente es deseable para aplicaciones exclusivas de simulación y procesamiento de datos donde es necesaria la repetición de resultados para la evaluación.

1.3. Clasificación de Generadores de Números Aleatorios

En los últimos años, los generadores de números aleatorios son usados en muchas áreas de la ciencia y la ingeniería, las cuales incluyen simulaciones por computadora, análisis numérico, evaluación de desempeño de algoritmos computacionales, muestreo estadístico, métodos de optimización estocásticos, además de ser vital para la seguridad cifrada.

Por lo tanto, la investigación sobre las diferentes metodologías para la generación de números aleatorios seguros está en auge. En consecuencia, las aplicaciones de los generadores de números aleatorios han evolucionado desde su casi exclusiva utilización en sistemas militares de cifrado hacia aplicaciones cotidianas, las cuales utilizan herramientas de comunicaciones y firmas digitales secretas para proteger transacciones electrónicas encontradas en radiofrecuencia, internet, mensajes de telefonía celular, transferencias bancarias, etc.

Entonces, conforme la seguridad en la transmisión de información se incrementa, la necesidad de proponer y diseñar un generador de números aleatorios confiable es vital. Debido a que el resultado obtenido por el generador de números aleatorios es un punto crítico, se pueden clasificar en generadores de números verdaderamente aleatorios y pseudo-aleatorio [13], [14].

1.3.1. Generador de Números Verdaderamente Aleatorios (TRNG)

El generador de números verdaderamente aleatorios (True Random Number Generator, TRNG) generalmente utiliza una fuente de señal no determinística en conjunto con funciones o procesos para generar la aleatoriedad. El objetivo de la función o proceso es el reducir o cancelar ciertas

debilidades de la fuente de entropía tales como la generación de una secuencia no aleatoria de números; por ejemplo, conjuntos largos de bits de únicamente "1" o "0". Además, generalmente, el origen de la fuente de entropía consiste de algún fenómeno o cantidad física, tales como ruido de un circuito eléctrico, tiempo de algún proceso (movimientos de mouse, tecleo, tiempo de decaimiento en una señal radioactiva), la combinación de diferentes metodologías [7].

Por definición, el TRNG es completamente impredecible debido a que la fuente de entropía es una variable de un fenómeno físico. Por lo tanto, el resultado de este generador puede ser utilizado directamente como un número aleatorio o puede ser aplicado para ser la fuente de señal de entrada a un pseudo-generador de números aleatorios. Para que el TRNG pueda ser utilizado directamente, su secuencia aleatoria de bits debe satisfacer estrictos criterios de aleatoriedad como los establecidos por el NIST para evaluar si el fenómeno físico realmente es aleatorio [9]. Es importante remarcar que los TRNG son utilizados para la toma de decisiones críticas; por ejemplo, se aplican en sistemas de seguridad para cifrar claves de acceso en cuentas de banco y en sistemas de paridad (sincronización) para comunicaciones seguras, además de manipular los juegos en casinos, juegos de lotería, etc. En resumen, los TRNG son los generadores más complejos de diseñar e implementar, sin embargo son los que tienen mayor aplicación en la actualidad.

1.3.2. Generador de Números Pseudo-Aleatorios (PRNG)

El generador de números pseudo-aleatorio (Pseudorandom Number Generator, PRNG) no es estrictamente aleatorio debido a que la generación de las secuencias de bits no depende de una fuente de entropía proveniente de un fenómeno físico, contario a los TRNG. En cambio, los PRNG son sistemas deterministas (predecibles) desde del punto de vista que son calculados mediante la utilización de un algoritmo matemático. Además requiere de una semilla (secuencia de bits iniciales) para su operación, la cual por si misma debe de ser aleatoria e impredecible. Por lo tanto un PRNG debe obtener su semilla a partir de un TRNG. Consecuentemente, si el algoritmo y semilla usados son previamente conocidos, entonces el generador es completamente predecible e inservible para fines prácticos.

El objetivo de los PRNG es obtener secuencias que se comporten como si fueran aleatorias. Por definición, la máxima cantidad de secuencias de bits producidas por este tipo de algoritmos son finitas y completamente reproducibles, por lo tanto solo serán aleatorias en un sentido limitado [7], [9].

Un ejemplo de un PRNG es el Generador Lineal de Congruencias de Lehmer, el cual es el más común dentro de los PRNG, este se define por la siguiente ecuación de recurrencia.

$$X_{j+1} = (aX_j + c)mod(m), \tag{1.1}$$

donde para la primera iteración $j = 0$ y X_0 es el valor inicial o semilla, a es el multiplicador, c es el incremento, y m es el módulo. Si un PRNG se diseña para tener un periodo máximo posible se denomina PRNG de periodo completo. Sin embargo, no todos los PRNG de periodo completo son adecuados. Por lo tanto, los PRNG con menores valores de auto correlación entre números sucesivos son preferibles. Considerando el PRNG de Lehmer, si utilizamos la siguiente ecuación de recurrencia $X_n = (5X_{n-1}+1)mod(16)$ con condición inicial de $X_0 = 5$, los primeros $32(16 + 16)$ números generados son: 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5.

Se observa que los números generados son enteros entre 0 y 15. Este es un generador de periodo completo ya que en la secuencia generada transcurren dieciséis números hasta que la secuencia es repetida, lo cual es mayor o igual al valor del argumento del módulo. Transcurrida esta secuencia el PRNG es completamente predecible. Debido a que diversos PRNG no son de periodo completo se compromete la aleatoriedad de la secuencia generada como se observa en el ejemplo descrito previamente.

Regularmente, los PRNG de periodo completo son utilizados combinando diferentes tipos de generadores; por ejemplo, contrario a la operación de suma se cambia por operación de multiplicación, conocido como Generador de Método Multiplicativo de Congruencias. Actualmente, el generador de Lagged-Fibonacci es ampliamente utilizado, el cual está familiarizado con la serie de Fibonacci y se basa en un valor adicional para reducir las correlaciones de las secuencias generadas al incrementar el periodo y utilizar cualquier operación aritmética entre los valores secuenciales.

Este tipo de generadores se aplican exclusivamente en simulaciones numéricas, donde se requiere emular un comportamiento aleatorio con capacidad de reproducir su resultado en diferentes condiciones limitándolo a aplicaciones sin impacto.

Aunque la elección de los valores a usarse en un PRNG es un tema importante, ya que deben de satisfacer ciertos aspectos tales como ser números primos no relacionados, mantener en secreto el valor de la semilla (condición inicial) es un tema prioritario. Para este propósito, el esquema más ampliamente utilizado es generar la condición inicial a partir de un TRNG. Por lo tanto, el diseño de un TRNG adecuado resulta imprescindible aun cuando se requiera diseñar un PRNG.

1.4. Comparación Entre TRNG y PRNG

Debido a que generalmente los TRNG provienen de fuentes de entropía físicas, frecuentemente se combinan diversas fuentes como entrada al TRNG, sin embargo la secuencia binaria puede ser deficiente al ser evaluada por los estándares del NIST. El éxito en la generación de secuencias aleatorias no depende exclusivamente de la fuente de entropía física sino también del conjunto de reglas para la realización de la secuencia final. Es importante considerar que para asegurar el éxito en la secuencia binaria generada, se pueden emplear demasiadas reglas de pertenencia, esto también incide negativamente en el tiempo de procesamiento, requisito que debe ser minimizado para aplicaciones que requieren cantidades grandes de secuencias aleatorias [7].

En consecuencia, algunas veces, los PRNG parecen ser más aleatorios que aquellos creados a partir de fenómenos físicos, debido a que si una secuencia binaria pseudo-aleatoria es construida adecuadamente, cada valor en la secuencia se genera mediante una transformación agregando una aleatoriedad adicional. Sin embargo, se debe considerar que para los PRNG de la misma manera en que se pueden producir algoritmos para aumentar la aleatoriedad, también se pueden crear algoritmos capaces de descifrar su regla de operación, invalidando así el rendimiento estadístico satisfactorio debido a que sus secuencias binarias serían completamente predecibles.

Ambos, TRNG y PRNG, tienen ventajas y desventajas, donde generalmente la restricciones existentes en un tipo de generador son las cualidades del otro. Teniendo en cuenta la idea anterior, en el siguiente cuadro se muestran las ventajas de los TRNG.

Generador de Números Verdaderamente Aleatorios (TRNG)	
Ventajas	**Desventajas**
No periódicos	Procesamiento lento
No predecibles sin importar la cantidad de secuencias binarias precedentes	Inconvenientes para la instalacion y ejecucion del método
No existe ninguna dependencie en su salida, es decir la secuencia binaria es no correlacionada	Secuencias completamente irreproducibles invalidándolo para propósitos de simulación
Se incrementa substancialmente el nivel de seguridad	Costosos desde el punto de vista computacional
No se basan en algoritmos para la generación de las secuencias binarias	Posibilidad de manipular las secuencias binarias

Cuadro 1.1: Ventajas y Desventajas de los Generadores de Números Verdaderamente Aleatorios.

En base al cuadro 1.1 se puede realizar un análisis del generador de números aleatorios más adecuado en función de la aplicación, recordando que la desventaja de un tipo de generador es una ventaja para el otro. Sin embargo, se debe verificar el generador funcionando en la aplicación,

ya que aunque no satisfaga todas las pruebas estadísticas del NIST puede ser muy eficiente para la aplicación en particular. Una pregunta abierta radica en entender como diferenciar de forma práctica el resultado de un PRNG bien diseñado a aquel de un TRNG proveniente de una fuente perfecta de aleatoriedad sin conocer el estado interno del generador y, en consecuencia, cual de los dos es más apto para una aplicación específica[7].

Capítulo 2

Dinámica de Osciladores Caóticos

2.1. Conceptos Básicos de Caos

Históricamente los fenómenos complejos han sido notados a nivel experimental, pero a menudo se ha decidido despreciarlos debido a que estos no afectan considerablemente el comportamiento del sistema o los conceptos para explicar tales fenómenos no existen. Un ejemplo de lo anterior es el oscilador de Van der Pol publicado en la revista Nature en 1927, donde se menciona: algunas veces un sonido irregular se escucha del circuito. En este artículo no se consideró ni se explicó ese "ruido" como un fenómeno adicional. Hoy día son ampliamente conocidas y entendidas las condiciones bajo las cuales ese ruido es generado.

2.1.1. Sensibilidad a Condiciones Iniciales

Partiendo del análisis matemático de E. Lorenz, entendió que el comportamiento futuro de un sistema caótico puede ser determinado exactamente por sus condiciones iniciales, sin embargo, la complejidad con que el sistema puede ser determinado crece exponencialmente con la exactitud de la predicción. Es decir, un sistema caótico exhibe aleatoriedad en el dominio del tiempo debido a que las condiciones iniciales no pueden ser especificadas con la suficiente exactitud para hacer una predicción a largo término, conduciendo a la definición de que un sistema caótico está inmerso en un sistema dinámico no lineal determinista pero con dependencia sensitiva a sus condiciones iniciales. Este determinismo puede ser observado mediante la representación en fase de las trayectorias del sistema dinámico generando un atractor caótico.

En la Figura 2.1 se puede observar la respuesta de un oscilador caótico con condiciones iniciales que varían en 0.01 %. A pesar de esta mínima variación las trayectorias divergen en menos de 5s.

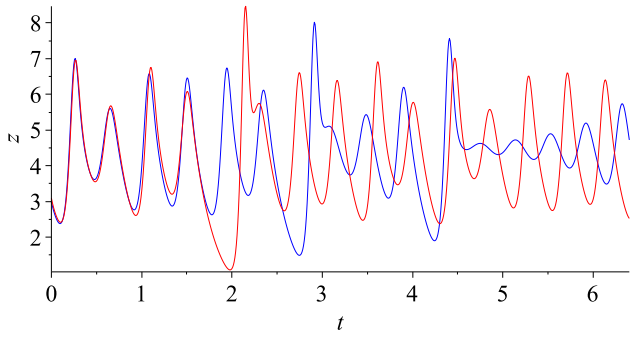

FIGURA 2.1: Dependencia sensitiva a condiciones iniciales.

2.1.2. Respuesta Temporal

La evolución temporal de las trayectorias de un sistema dinámico puede orientar la observación del fenómeno caótico. Esto significa que la evolución temporal de un oscilador caótico difiere de un oscilador típico donde su respuesta es periódica o cuasi-periódica. En un comportamiento caótico no es posible observar un periodo definido debido a que sus trayectorias parecen no repetirse durante una ventana de tiempo finito. Gráficamente, la Figura 2.2 muestra este fenómeno. En la Figura 2.2(superior) se observa la evolución temporal de un oscilador periódico mientras que en la Figura 2.2(central) se muestra la forma de onda de un oscilador cuasi-periódico, también conocido como oscilador de n-periodos. En contraste, la Figura 2.2(inferior) define una respuesta temporal de un oscilador caótico.

2.1.3. Espectro de Potencia

Análogamente, la Transformada Rápida de Fourier (FFT) de las señales temporales, se muestran en la Figura 2.3 en donde se puede apreciar, para el caso de un oscilador periódico, en la Figura 2.3(superior) un pico de potencia en la frecuencia fundamental y sus armónicos respectivos. En la Figura 2.3(central) el espectro de potencia se distribuye en más de un pico de potencia relativo a un intervalo de frecuencia más amplio. Finalmente, en la Figura 2.3(inferior) el espectro de potencia de un oscilador caótico está distribuido sobre un rango amplio de frecuencias. Por esto se define que su espectro es similar al generado por el ruido blanco pero con un cierto ancho de anda.

2.1.4. Condiciones Necesarias para Generar Caos

No todos los sistemas dinámicos pueden exhibir comportamiento caótico. Por ejemplo, un sistema dinámico definido por dos ecuaciones diferenciales de primer orden únicamente puede tener como solución un estado estable o un ciclo límite. Por lo tanto, una condición que debe cumplirse

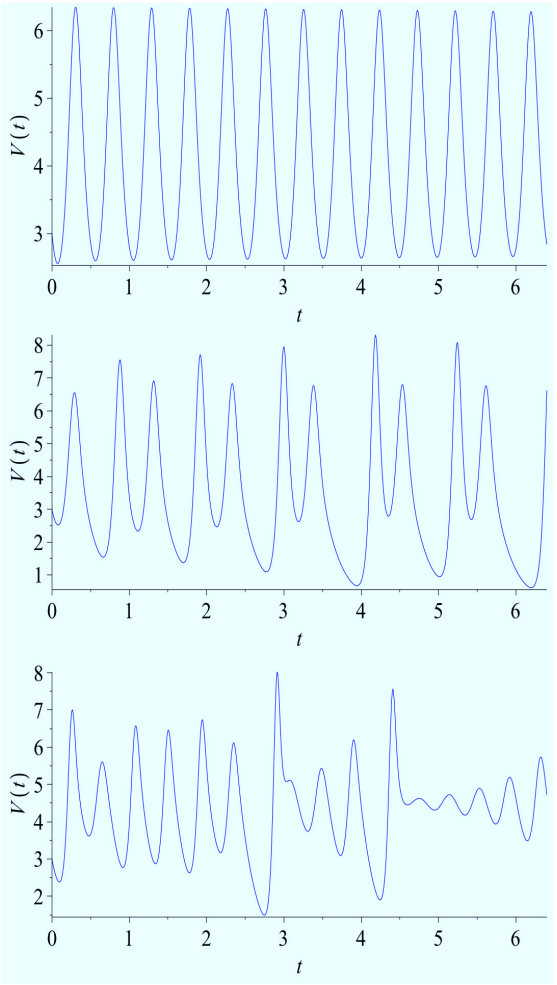

FIGURA 2.2: Comparación de formas de onda, $V(t)$, en osciladores.

para que exista caos es que el sistema dinámico esté definido por tres ecuaciones diferenciales de primer orden acopladas.

Sin embargo, no todos los sistemas con tres ecuaciones diferenciales de primer orden tienden a generar un comportamiento caótico. Entonces, otra condición exige la incorporación de un término no lineal al sistema de ecuaciones diferenciales.

Retomando los conceptos previamente mencionados, el caos generalmente se define como un sistema determinista con comportamiento aperiódico que exhibe una dependencia muy sensible a las condiciones iniciales, lo cual provoca un comportamiento aleatorio a largo plazo. Además para su aparición como fenómeno, la dinámica del sistema debe estar definida por al menos tres ecuaciones diferenciales de primer orden y una función no lineal.

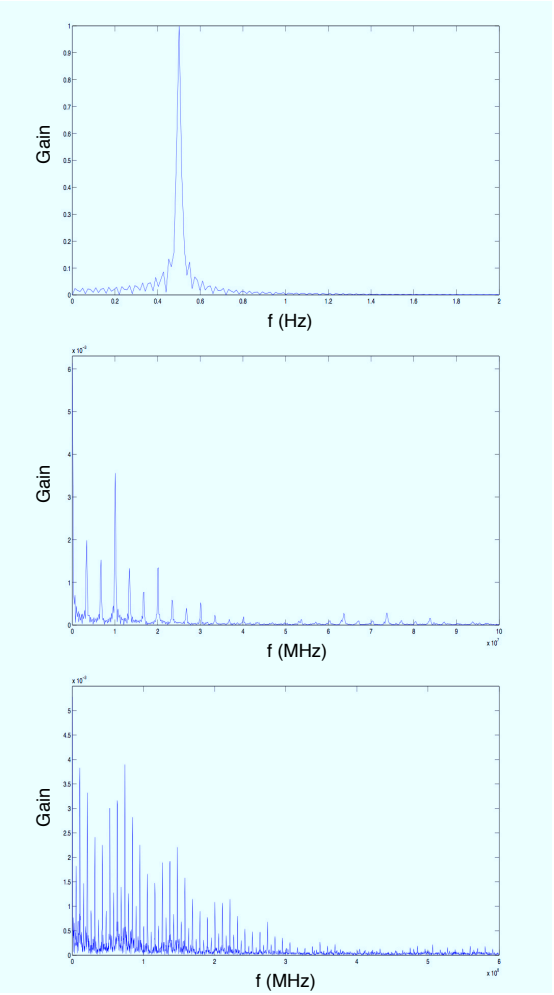

FIGURA 2.3: Comparación de espectro de potencia en osciladores.

2.2. Exponente de Lyapunov

Los exponentes de Lyapunov son mediciones asintóticas que caracterizan la divergencia o convergencia de puntos inicialmente cercanos en un sistema dinámico. Básicamente, es la razón de estiramiento o encogimiento de trayectorias, que comienzan desde puntos muy cercanos entre si, conforme el sistema evoluciona temporalmente[15]. Este concepto es de vital importancia para cuantificar el comportamiento caótico y evitar dejar a la apreciación del observador la existencia o no del régimen caótico.

El número de exponentes de Lyapunov en un sistema es igual al número de variables de estado del mismo. Además, la presencia de al menos un exponente positivo de Lyapunov dentro de un sistema dinámico no lineal se considera como una prueba definitoria del comportamiento caótico.

Formalmente el exponente de Lyapunov puede definirse de la siguiente manera: Considere dos puntos cercanos en el espacio (x_0) y $(x_0 + u_0)$, donde u_0 es una pequeña perturbación del punto inicial x_0. Después de un tiempo t, sus imágenes bajo el flujo serian $f^t(x_0)$ y $f^t(x_0 + u_0)$ y la perturbación u_t se convertiría en:

$$u_t = f^t(x_0 + u_0) - f^t(x_0) \tag{2.1}$$

Por lo tanto la tasa promedio exponencial de crecimiento o encogimiento de las dos trayectorias es definido por:

$$\lambda(x_0, u_0) = \lim_{t\to\infty} \frac{1}{t} \ln ||\frac{u_t}{u_0}|| \tag{2.2}$$

donde $||u||$ denota el tamaño del vector u.

Si $\lambda(x, u) > 0$, entonces se tiene divergencia exponencial de las órbitas que comienzan en puntos cercanos. La definición anterior se refiere al exponente de Lyapunov de primer orden. Para sistemas de orden superior, en los cuales se puede presentar comportamiento caótico como ha sido explicado anteriormente, es necesario un enfoque diferente. Considere un sistema dinámico n-dimensional:

$$\dot{x} = f(x), \qquad t > 0, \qquad x(0) = x_0 \in R^n \tag{2.3}$$

donde x y f son campos vectoriales de n-dimensión. Para determinar los n-exponentes de Lyapunov, se tiene que encontrar la evolución a largo término de las pequeñas perturbaciones a la trayectoria, las cuales están determinadas por la ecuación variacional de (2.3) dada por:

$$\dot{y} = \frac{\partial f}{\partial x}(x(t))y = J(x(t))y \tag{2.4}$$

donde J es la matriz Jacobiana de $n \times n$ de f.

Una solución de (2.4) con condiciones iniciales dadas $y(0)$ puede ser escrita como:

$$y(t) = Y(t)y(0) \tag{2.5}$$

Con $Y(t)$ como la solución fundamental que satisface

$$\dot{Y} = J(x(t))Y, \qquad Y(0) = I_n \tag{2.6}$$

Aquí I_n denota la matriz identidad de $n \times n$. Por lo tanto, el i-ésimo exponente de Lyapunov, el cual mide la sensibilidad de largo plazo del flujo de $x(t)$ con respecto a los datos iniciales $x(0)$ en la dirección $p_i(t)$, se define por la tasa de expansión de la longitud del i-ésimo eje $p_i(t)$ dada

por:

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \ln \frac{p_i(t)}{p_i(0)} \tag{2.7}$$

2.3. Entropía

La entropía fue originalmente formulada como un concepto útil en la termodinámica; no obstante, su importancia creció cuando se perfeccionó el campo de la mecánica estadística, ya que esta da un medio alternativo de interpretar la entropía y una importancia más general del concepto.

Para especificar el calor debemos emplear por lo menos dos números. Uno para indicar la cantidad de energía, el otro para medir la cantidad de desorden. La cantidad de energía se mide mediante una unidad práctica llamada caloría, la cantidad de desorden se mide por medio del concepto matemático que se denomina entropía.

Si hay una relación entre desorden y la entropía, entonces el desorden, lo mismo que la entropía, deben aumentar en los procesos naturales, se puede decir: Un proceso natural que comienza en un estado de equilibrio y termina en otro, se desarrollara en el sentido que haga que aumente la entropía del sistema más el medio ambiente.

En la mecánica estadística damos un significado preciso al desorden y expresamos su relación con la entropía mediante la ecuación:

$$S = k \ln w \tag{2.8}$$

En esta ecuación k es la constante de Boltzmann, S es la entropía del sistema y w, que podemos llamar parámetro de desorden, es la probabilidad de que el sistema exista en el estado en que se encuentra en relación con todos los estados posibles en que pudiera hallarse. Esta ecuación relaciona una cantidad termodinámica o macroscópica, la entropía, con una cantidad estadística o microscópica, la probabilidad. La dirección en que ocurren los procesos naturales (hacia una entropía mayor) queda determinada por las leyes de la probabilidad (hacia un estado más probable). El estado de equilibrio es el estado de máxima entropía termodinámicamente y el estado más probable estadísticamente.

En muchos artículos es ocupada la medición de la entropía, su uso generalmente es en investigaciones en campos como neurociencia, estimación de información mutua, asegurar compresión de información sin tener perdida de datos, secuencias de símbolos para poder descifrar información, entre otras cosas.

En la literaura se puede encontrar que la forma para medir la entropía de una señal es:

$$H_x = -\int_{-\infty}^{\infty} f_x(x) \ln f_x(x) dx \tag{2.9}$$

Donde H_x es la entropía de X,X es un vector con valores dentro de R^d y $f_x(x)$ es función de densidad de probabilidad.

La evaluación de la integral anterior a menudo necesita métodos numéricos y no es fácil de lograr. Sin embargo usando otro método puede ser fácilmente estimada, donde f_x es un histograma, esta aproximación es tomada generalmente para fines computacionales quedando de la siguiente manera:

$$H = -\Sigma_i p_i \ln p_i \tag{2.10}$$

Donde H es la etropía de un conjunto discreto de probabilidades $p_1, p_2, \ldots, p_n$.

2.4. Entropía de Shannon

La entropía de Shannon cuantifica la riqueza o "complejidad" de una fuente emitiendo secuencias, proporcionando una medida de la "sorpresa" que la fuente reserva para nosotros. Esto puede ser mejor expresado en términos de un teorema fundamental, primero demostrado por Shanon para las fuentes de Markov y entonces generalizado por McMillan para fuentes estacionarias ergódicas genéricas.

Considerando una fuente estacionaria y ergódica emitiendo símbolos provenientes de un alfabeto infinito de M letras, denotado con $s(t)$ el símbolo emitido en el tiempo t y con $P(W_N) = P(s(1), s(2), \ldots, s(N))$ la probabilidad de encontrar los N símbolos consecutivos (N-palabra) $W_N = s(1)s(2)\ldots s(N)$. Si N es lo suficientemente grande, el establecimiento de todas las posibles N-palabras, $\Omega(N) == W_N$ puede ser particionado dentro de dos clases $\Omega_1(N)$ y $\Omega_0(N)$, tal que si W_n pertenece a $\Omega_1(N)$ entonces, $P(W_N) \sim exp(-Nh_{Sh})$ y

$$\sum_{W_N \in \Omega_1(N)} P(W_N) \overrightarrow{N \to \infty} \quad 1,$$

mientras

$$\sum_{W_N \in \Omega_0(N)} P(W_N) \overrightarrow{N \to \infty} \quad 0.$$

En principio, por un alfabeto compuesto por M letras, existen M^N diferentes N-palabras, aunque algunas pueden ser prohibidas, tal que, en general, el número de posibles N-palabras es

$\mathcal{N}(N) \sim exp(Nh_T)$, donde

$$h_T = \lim_{N\to\infty} \frac{1}{N} \ln \mathcal{N}(N)$$

es nombrada *entropía topológica* y tiene como límite superior $h_T \leq \ln M$ (la igualdad es realizada si todas las palabras son permitidas). El significado del teorema Shannon-McMillan es que entre todas las N-palabras permitidas, $\mathcal{N}(N)$, el número de las palabras típicas ($W_N \in \Omega_1(N)$), que son efectivamente observadas, es:

$$\mathcal{N}_{eff}(N) \sim e^{Nh_{Sh}}.$$

Como $\mathcal{N}_{eff}(N) \leq \mathcal{N}(N)$ entonces:

$$h_{Sh} \leq h_T \leq \ln M.$$

Se concluye recalcando que la entropía de Shannon, h_{Sh}, es una propiedad inherente de la fuente y que, gracias a que la fuente es ergódica, puede ser derivada analizando únicamente una sola secuencia, suficientemente grande, en el conjunto de las palabras típicas. Por lo tanto, h_{Sh} también puede ser vista como una propiedad de las secuencias de palabras típicas [16].

2.5. Entropía de Kolmogorov-Sinai: Complejidad de los Sistemas Caóticos

Explotando el marco conceptual y técnico de la teoría de información es posible caracterizar sistemas dinámicos caóticos. Sin embargo, la mayoría de las herramientas introducidas se basan en secuencias símbolicas, por lo que es necesario que las trayectorías caóticas, existentes en el mundo de los números reales, puedan ser apropiadamente codificadas dentro de secuencias símbolicas discretas.

Para esto es necesario realizar una partición del espacio de fase, Ω, sin importar el tipo de sistema bajo estudio, es decir en tiempo discreto o continuo. En el último caso, una discretización temporal puede ser introducida por medio de un mapa de Poincaré o fijando un tiempo de muestreo τ y almacenando una trayectoría en $t_j = j\tau$. Por lo tanto, sin pérdida de generalidad, el análisis se puede límitar a mapas $\mathbf{x}(t+1) = \mathbf{F}(\mathbf{x}(t))$. Si se consideran particiones del tipo $A = \{A_0, ..., A_{M-1}\}$ de Ω hechas de elementos no adjuntos, entonces el conjunto $\mathcal{A} = \{0, 1, ..., M-1\}$ de $M \leq \infty$ símbolos constituye el alfabeto inducido por la partición, y cualquier trayectoria $\mathbf{X} = \{\mathbf{x}(0), \mathbf{x}(1)...\mathbf{x}(n), ...\}$ puede ser codificada en la secuencia símbolica $\mathcal{S} = \{s(1), s(2)...s(n)...\}$ con $s(j) = k$ si $\mathbf{x}(j) \in \mathbf{A}_k$. En principio, el número, tamaño y forma de los elementos de la partición puede ser seleccionado arbitrariamente. Estudios detallados sobre las técnicas de partición son ampliadas en [16].

En particular para la entropía Kolmogorov-Sinai, se considera la dinámica simbólica resultado de una partición A de el espacio fase Ω de un sistema dinámico ergódico discreto $\mathbf{x}(t+1) = \mathbf{F}(\mathbf{x}(t))$ con una medida invariante μ^{inv}. Es posible asociar una probabilidad $P(A_k) = \mu^{inv}(A_k)$ a cada elemento A_k de la partición. Tomando el $(N-1)$-refinamiento $A^{(N-1)} = \bigvee_{k=0}^{N-1} F^{-1}A$, $P(A_k^{(N-1)} = \mu^{inv}(A_k^{(N-1)})$ define la probabilidad de N-palabras $P(W_N(A))$ de la dinámica simbólica inducida por A, a partir de la cual se obtienen las entropías de N-bloques:

$$H_N(A) = H(\bigvee_{k=0}^{N-1} A) = -\sum_{W_N(A)} P(W_N(A)) \ln P(W_N(A)),$$

y las entropías de diferencia:

$$h_N(A) = H_N(A) - H_{N-1}(A).$$

La entropía de Shannon que caracteriza el sistema con respecto de la partición A,

$$h(A) = \lim_{N\to\infty} \frac{H(\bigvee_{k=0}^{N-1} A)}{N} = \lim_{N\to\infty} \frac{H_N(A)}{N} = \lim_{N\to\infty} h_N(A),$$

existe y depende tanto de la partición A como de las medidas invariantes. Ésta cuantifica la incertidumbre promedio por paso de tiempo en el elemento de la partición visitado por las trayectorias del sistema. Como el propósito es caracterizar la fuente y no una partición específica A, es deseable eliminar la dependencia de la entropía sobre A, esto puede ser llevado a cabo por considerar el supremo sobre todas las particiones posibles:

$$h_{KS} = \sup_A \{h(A)\}, \tag{2.11}$$

el cual define la entropía $Kolmogorov - Sinai$ (KS) de un sistema dinámico bajo estudio, y que únicamente depende en las medidas invariantes, por lo tanto otro nombre que recibe es *entropía métrica*. El supremo en la ecuación (2.11) es necesario por que particiones erróneas pueden eliminar la incertidumbre incluso si el sistema es caótico. Además, la propiedad del supremo hace la cantidad invariante con respecto a isomorfismos entre sistemas dinámicos.

Por lo tanto, la entropía-KS cuantifica no solo la riqueza de la dinámica del sistema si no también la dificultad de describir completamente el resultado de las secuencias simbólicas resultantes. Es importante notar que, análogamente a la dimensión de la información y a los exponentes de Lyapunov, la entropía Kolmogorov-Sinai provee una caracterización de trayectorias típicas, y no toma en cuenta fluctuaciones que pueden explicarse mediante la introducción de la entropía de Rényi [16]. Aunado a eso, la entropía métrica, como los exponentes de Luapunov, es una

cantidad característica invariante de un sistema dinámico, significando que los isomorfismos dejan a la entropía-KS sin cambio.

Por otro lado, la conexión entre la entropía-KS y los exponentes de Lyapunov se define, para sistemas génericos, por medio de la relación de Pesin,

$$h_{KS} \leq \sum_{\lambda_i > 0} \lambda_i.$$

Se observa que solo en sistemas de baja dimensión una determinación directa de h_{KS} es factible. Por lo tanto, el conocimiento del espectro de Lyapunov provee, a través de la relación de Pesin, la única posibilidad de estimar h_{KS} para sistemas de grandes dimensiones. En resumen, la relación de Pesin muestra que la imprevisibilidad de sistemas dinámicos caóticos, cuantificada por los exponentes de Lyapunov, tiene su contraparte en la teoría de la información. Caos determinísta genera mensajes que no pueden ser codificados de forma concisa, debido a la positividad de la entropía de Kolmogorov-Sinai, por lo que el caos puede ser interpretado como una fuente de información y trayectorias caóticas que son algorítmicamente complejas. Es decir, la entropía de Kolmogorov-Sinai es estrictamente positiva y finita, en particular $0 < h_{ks} \leq \sum_{(\lambda_i > 0)} \lambda_i < \infty$.

Se concluye entonces que tanto la entropía-KS como los exponentes de Lyapunov son caracterizaciones de los sistemas dinámicos, que desde un punto de vista de la teoría de la información, esto corresponde al requerimiento de la tasa de pérdida de recuperación de información producida por una fuente caótica.

Capítulo 3

Síntesis de Osciladores Caóticos a Nivel Experimental

Dentro de este capítulo se presentará el modelado matemático, la síntesis y la simulación de cada oscilador caótico a utilizar en el proyecto. El modelado matemático se basará en las ecuaciones respectivas a cada sistema caótico. La simulación numérica se empleará con la ayuda del programa Matlab para observar el comportamiento de cada sistema caótico. Mientras que la síntesis con dispositivos electrónicos se efectuará empleando programas como TopSpice y Proteus, para verificar que el comportamiento tanto de la función no lineal como la respuesta del sistema terminado sea el correcto.

3.1. Oscilador Caótico de Chua

El oscilador caótico de Chua es el circuito electrónico más ampliamente utilizado para generar el fenómeno de caos. Esto se debe a la variedad de comportamientos dinámicos tales como punto fijo, atractor de n-periodos, atractor caótico y ciclo límite, que pueden verificarse directamente mediante la implementación electrónica utilizando componentes electrónicos estándar[17], [11], [18], [3]. Una de las representaciones más conocidas del oscilador caótico de Chua es aquel que consiste en dos capacitores, un inductor, un resistor lineal y un resistor no lineal (Nr) llamado diodo de Chua. La desventaja de esta implementación radica en el uso de inductores.

El oscilador caótico de Chua se caracteriza principalmente por ser autónomo, es decir, no requiere de fuentes de señal externas dependientes del tiempo para oscilar; y estar compuesto por una parte lineal y una función no lineal. Esta función no lineal actúa como la fuente de energía del circuito mediante retroalimentación para sostener las oscilaciones [12]. Relativo a esto, el

conjunto de ecuaciones diferenciales acopladas que representan el comportamiento del oscilador caótico de Chua son las siguientes [19], [20]:

$$\begin{aligned} \frac{dx}{dt} &= \alpha(y - x - f(x)) \\ \frac{dy}{dt} &= x - y + z \\ \frac{dz}{dt} &= -\beta y - \gamma z \end{aligned} \tag{3.1}$$

Donde α, β y γ son los parámetros que definen la región donde el comportamiento puede estar presente y $f(x) = bx + 1/2(a-b)[|x+B_p| - |x-B_p|]$ es una función no lineal cuya representación gráfica se observa en la Figura 3.1. La función no lineal de la Figura 3.1 también se define como una función PWL compuesta por tres segmentos de línea recta, donde el valor de a es el valor de la pendiente central, b es el valor de las pendientes exteriores y $\pm B_P$ son los puntos de quiebre. Es importante mencionar que cada uno de los segmentos de la función no lineal se pueden aproximar por una relación constitutiva de rama relacionando voltajes y/o corrientes, siempre y cuando estas relaciones tengan una ganancia negativa.

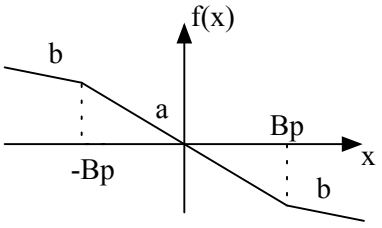

FIGURA 3.1: Segmentos característicos $V_R - I_R$ del resistor no lineal Nr.

Con el objetivo de simular numéricamente mediante MATLAB el oscilador caótico de Chua la ecuación (3.2) se adecúa a la forma matricial, obteniéndose la representación de espacio de estados de la siguiente manera.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -\alpha & \alpha & 0 \\ 1 & -1 & 1 \\ 0 & -\beta & \gamma \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\alpha f(x) \\ 0 \\ 0 \end{bmatrix} \tag{3.2}$$

Seleccionando valores para los parámetros del sistema y de la función no lineal de $\alpha = 10$, $\beta = 16$, $\gamma = 0.13$, $a = -1.22$, $b = -0.78$ y $B_p = \pm 0.5$ se obtiene el atractor caótico mostrado en la Figura 3.3 con el comportamiento de la función no lineal observado en la Figura 3.2.

Se observa de la Figura 3.1 el comportamiento de la función no lineal del diodo de Chua, el cual se caracteriza por dos pendientes negativas y el cual es análogo al comportamiento descrito

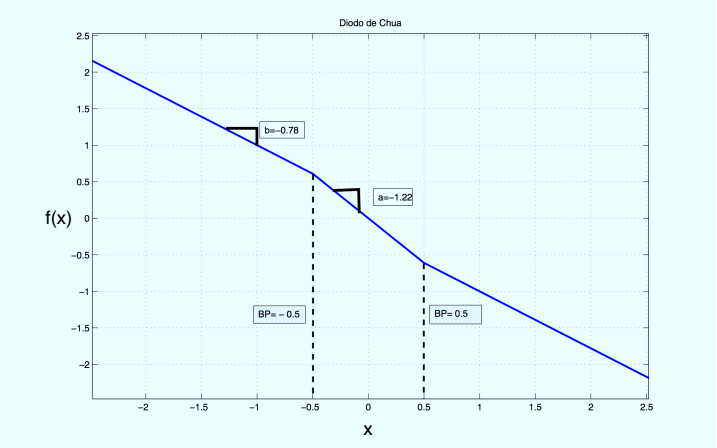

FIGURA 3.2: Curva característica de la función no lineal de oscilador caótico de Chua obtenida mediante MATLAB. x,f(x)

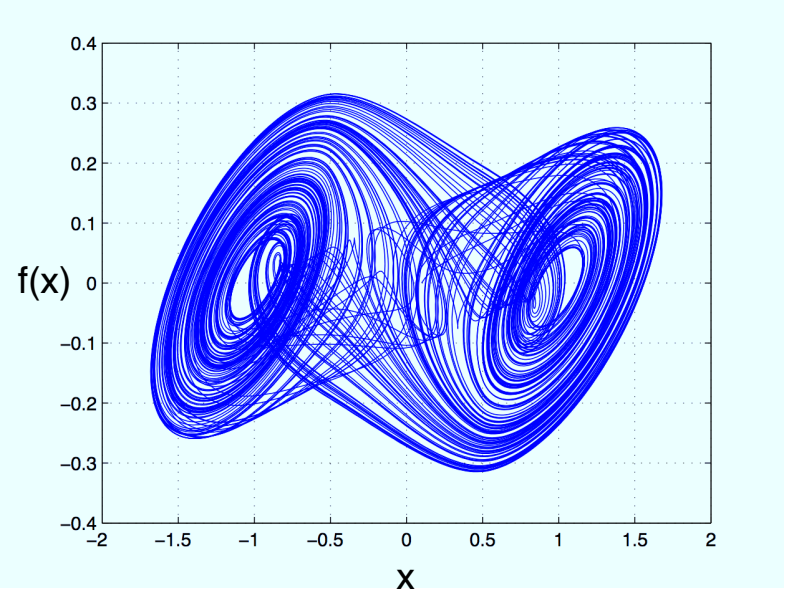

FIGURA 3.3: Resultado de la simulación numérica del sistema caótico de Chua usando MATLAB.

por la Figura 3.2. Por otro lado, en la Figura 3.3 se observa la dinámica del oscilador caótico de Chua, la cual está conformada por un atractor caótico de doble enrrollamiento teniendo 3 puntos de equilibrio ubicados a lo largo de la variable de estado $x1$. Dos puntos de equilibrio son responsables de generar los enrrollamientos, es decir, las órbitas de las trayectorias de solución son atraídas a esos puntos, mientras que el punto de equilibro restante se encarga de unir los enrrollamientos.

3.1.1. Síntesis del Oscilador Caótico de Chua Usando OpAmps

La síntesis de circuitos electrónicos es el proceso de mapear un modelo matemático a un circuito electrónico con cierta topología y elementos. El proceso de mapear se puede realizar, más frecuentemente, considerando una topología puramente pasiva, una topología activa eliminando el uso de inductores conocida como la técnica de OpAmp-RC, o una topología activa eliminando el uso tanto de inductores como de resistores conocida como la técnica de OTA-C. Para los propósitos de esta tesis se selecciona la técnica de OpAmp-RC, siendo esta la más adecuada y conocida para poder comparar los resultados obtenidos, además de que el uso de OpAmps diversifica la expansión del diseño por tratarse de circuitos electrónicos ampliamente conocidos y verificado en un gran número de aplicaciones a nivel de ingeniería.

Por lo tanto, el sistema de variables de estado en (3.2) puede ser sintetizado con OpAmps manteniendo las relaciones entre los parámetros del sistema y manteniendo inalterado el comportamiento de la función no lineal. Para este propósito, el primer paso es representar el oscilador caótico de Chua de (3.1) por medio de un diagrama a bloques mostrado en la Figura 3.4.

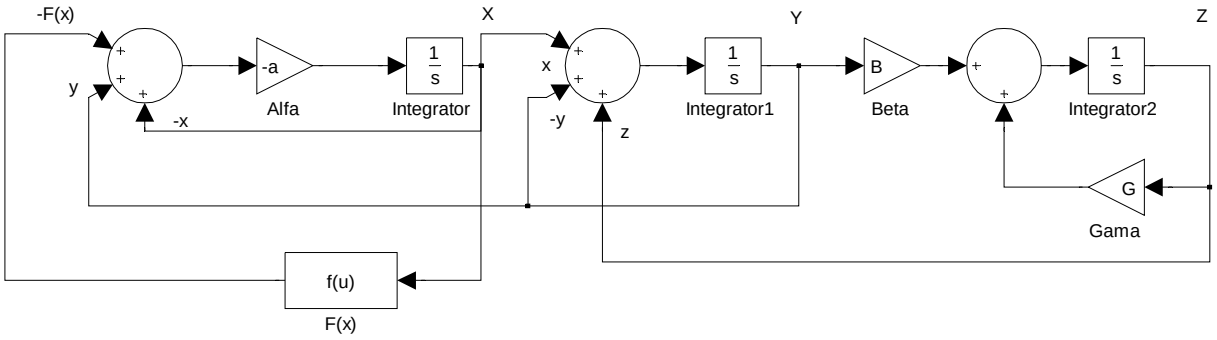

FIGURA 3.4: Diagrama a bloques del sistema de Chua basado en las ecuaciones (3.1) y (3.2)

Este paso es necesario para observar el flujo de señal y definir los elementos necesarios para la síntesis. Es decir, se observa que se requieren tres integradores, tres sumadores, dos bloques de ganancia y un bloque de función no lineal. Partiendo de este diagrama, los bloques son sintetizados por configuraciones electrónicas basadas en OpAmps. En la Figura 3.5 se muestra en resultado del proceso de síntesis para el oscilador caótico de Chua. Está diseñado con tres integradores de voltaje y tres sumadores de voltaje en configuración inversora, los bloques de ganancia para α, β, y γ están implícitamente diseñados mediante la ganancia de lazo cerrado de los sumadores de voltaje.

Además, el valor de las resistencias utilizadas y la ubicación de las variables del sistema se muestran en la Figura 3.5. Respecto a la función no lineal, se diseña el circuito de la Figura 3.6 para replicar el comportamiento mostrado en la Figura 3.2. Basándose en la idea de [17], donde la formulación de la función no lineal se realiza por medio de la suma de dos funciones lineales

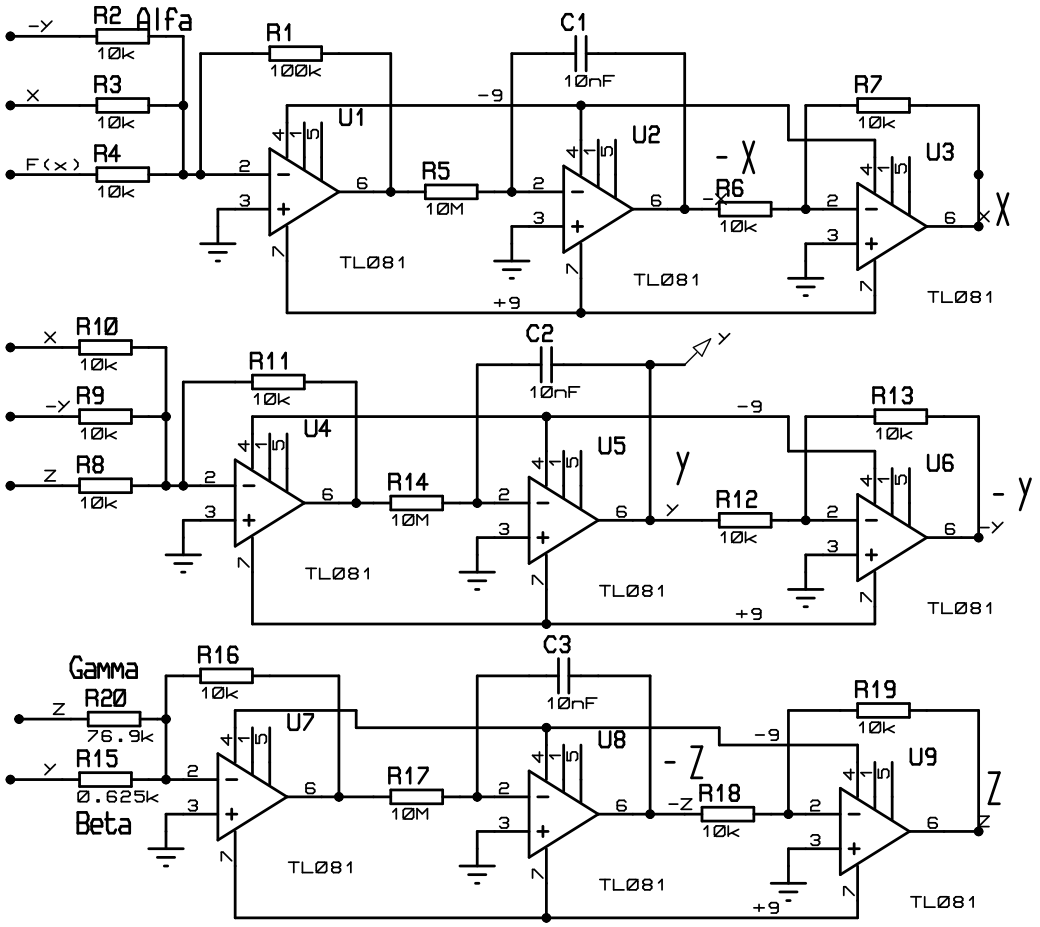

FIGURA 3.5: Circuito utilizado para la síntesis del sistema de Chua mediante TopSpice.

con pendiente negativa, se diseña el circuito de la Figura 3.6. La idea que se propone es obtener de forma independiente dos ganancias negativas (pendiente), obtenidas mediante dos OpAmps en configuración de ganancia inversora cada una y finalmente sumarlas, para obtener la característica de la función no lineal compuesta por la relación lineal por partes de tres segmentos. El resultado es una nueva propuesta para el diseño de la función no lineal del oscilador caótico de Chua.

Para tener control sobre los parámetros a y b de la función no lineal se tienen las ecuaciones (3.3) - (3.5). Estas ecuaciones de diseño sirven para modificar el valor de las pendientes exteriores, la pendiente central y el valor de lo puntos de ruptura.

$$R18/R17 = \frac{V_{sat}}{BreakPoint} \tag{3.3}$$

$$R20/R19 = \frac{BreakPoint \times (b-a)}{V_{sat}} \tag{3.4}$$

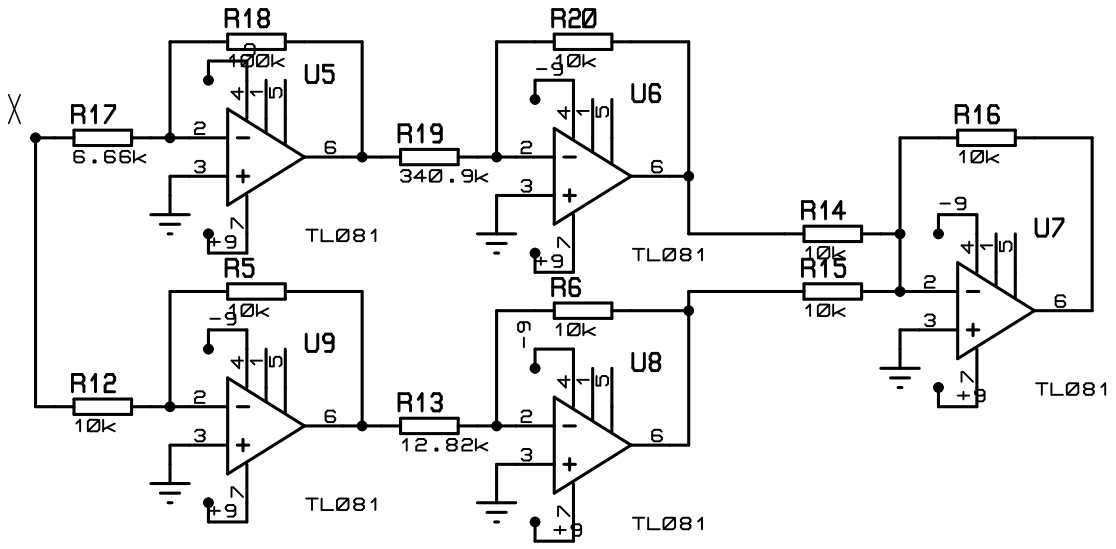

FIGURA 3.6: Circuito que permite obtener la función no lineal del diodo de Chua.

$$b = R6/R13 \tag{3.5}$$

La simulación a nivel circuito utilizando el software SPICE se muestran en la Figura 3.7 y 3.8, obteniéndose una dinámica análoga a la mostrada en las Figuras 3.2 y 3.3. En específico en la Figura 3.7 se observan las dos pendientes negativas y el resultado de la suma de las mismas, las cuales se comportan como una fuente de voltaje controlado por voltaje cuyo rango es alrededor de los $\pm 3V$. Por otra parte en la Figura 3.8 se muestra el atractor del oscilador caótico de Chua, observado en el plano de fase de voltaje utilizando la variables de estado x y y con rangos para la señal x rondando los $\pm 1.5V$ el cual es controlado por el rango de la función no lineal; y de alrededor de $\pm 300mV$ para la señal y.

3.1.2. Realización Experimental del Oscilador Caótico de Chua

Para el diseño de la arquitectura del generador de números aleatorios se requiere de la realización experimental del oscilador caótico de Chua. Por lo tanto, se implementan los circuitos de las Figura 3.5 y Figura 3.6 utilizando el OpAmp TL081 con fuentes de polarización de $\pm 9V$, resultando en un voltaje de saturación de $\pm 7.5V$. El resultado de la implementación de la función no lineal por partes se muestra en la Figura 3.9, donde se puede observar la medición de los puntos de ruptura correspondientes al diseño de la subsección anterior. El circuito completo se implementó con los valores de los elementos mostrados en las Figura 3.5 y Figura 3.6 para obtener valores de α, β, γ, a, b y $\pm Bp$ de 10, 16, 0.13, -1.22,-0.78 y ± 0.5, respectivamente.

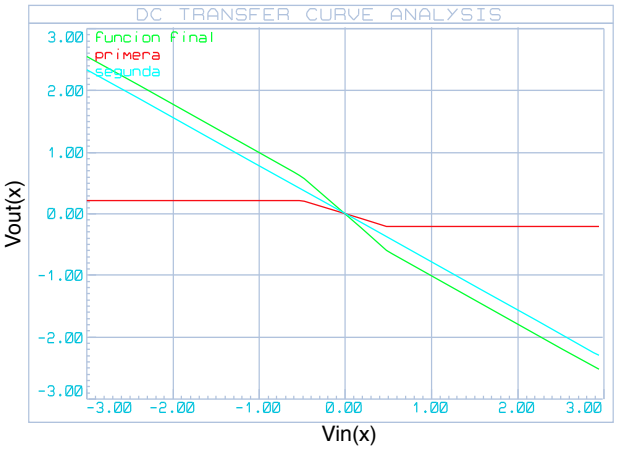

FIGURA 3.7: Función lineal por partes deseada. La función final, color verde; es la suma de las otras dos funciones.

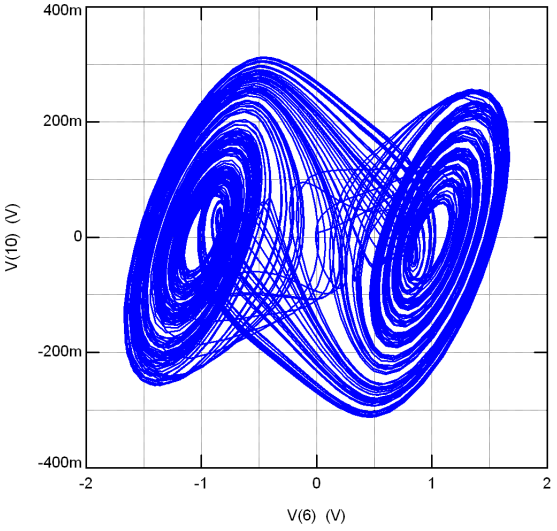

FIGURA 3.8: Simulación del comportamiento del circuito caótico de Chua mediante TopSpice.

Es importante mencionar que los valores resistivos no comerciales fueron obtenidos mediante el uso de trimpots (potenciómetros de precisión). El resultado de la implementación se muestra en la Figura 3.10, la cual valida a nivel experimental el diseño propuesto.

3.2. Oscilador Caótico de Funciones Saturadas

Debido a que recientemente el uso de señales caóticas dentro del campo de comunicaciones seguras ha atraído mucho interés, se ha aumentado la investigación y se han propuesto diferentes sistemas generadores de caos. Una de las áreas de mayor atención es la relacionada con la generación de más de dos enrollamientos, es decir, múltiples enrollamientos. La idea consiste en incrementar los puntos de ruptura de la función no lineal, lo cual trae como consecuencia

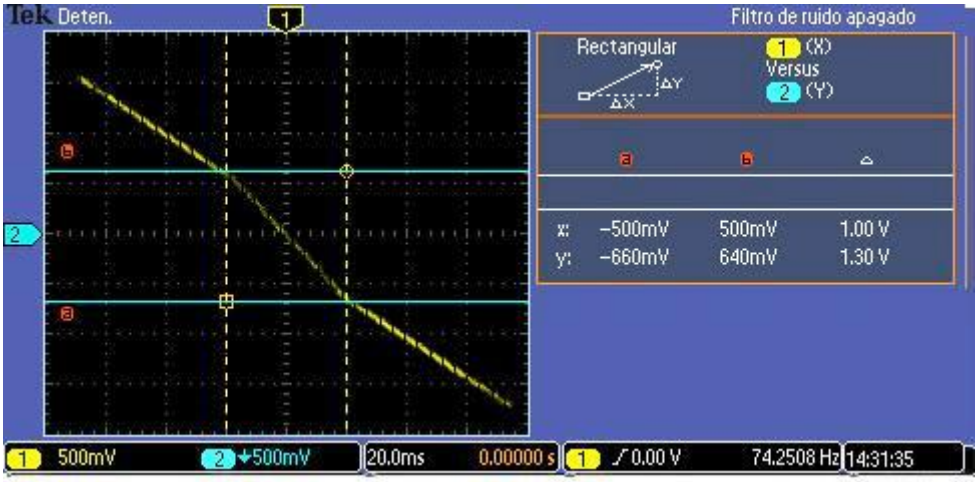

FIGURA 3.9: Resultado de la implementación de la función del diodo de Chua.

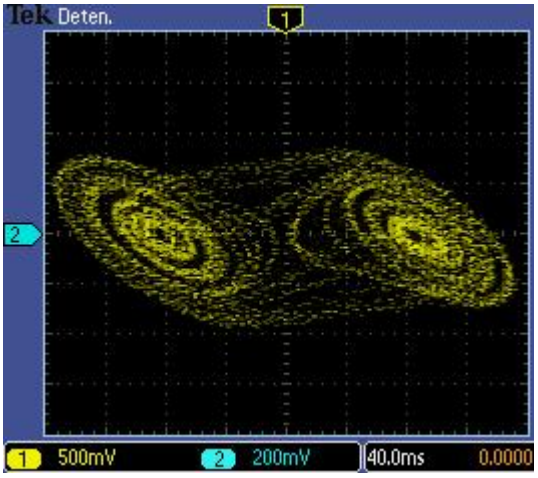

FIGURA 3.10: Resultado de la implementación circuito caótico de Chua.

que se formen nuevos puntos de equilibrio. Por lo tanto el comportamiento dinámico de la señal caótica tendrá dinámicas mas complejas en comparación con los sistemas caóticos de doble enrollamiento. Además, el número de enrollamientos se incrementa en relación al número de puntos de ruptura [21]. Uno de los generadores caóticos utilizados para este propósito es el generador caótico de funciones saturadas. Este generador se fundamenta en el uso de una serie de funciones saturas, la cual se conforma por múltiples puntos de ruptura y puede generar desde 2 hasta n-enrollamientos a lo largo de la línea descrita por una de las variables de estado. La descripción matemática del sistema caótico de funciones saturadas se representa por la ecuación siguiente:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -ax - by - cz + df(x;k,h,p,q) \end{aligned} \tag{3.6}$$

donde x, y, z son las variables de estado del sistema; a, b, c, d, son coeficientes reales positivos, que para cumplir las condiciones de caos que deben satisfacer $0 < a, b, c, d < 1$. Adicionalmente, la función no lineal $f(x)$ es definida por la ecuación (3.7) y cuyo comportamiento se muestra en la Figura 3.11 [22].

$$f(x;k,h,p,q) = \begin{cases} (2q+1)k, & \text{si} \quad x > qh+1 \\ k(x-ih)+2ik, & \text{si} \quad |x-ih| \leq 1, -p \leq i \leq q \\ (2i+1)k, & \text{si} \quad ih+1 < x < (i+1)h-1, \\ & \quad -p \leq i \leq q-1 \\ -(2p+1)k, & \text{si} \quad x < -ph-1. \end{cases} \tag{3.7}$$

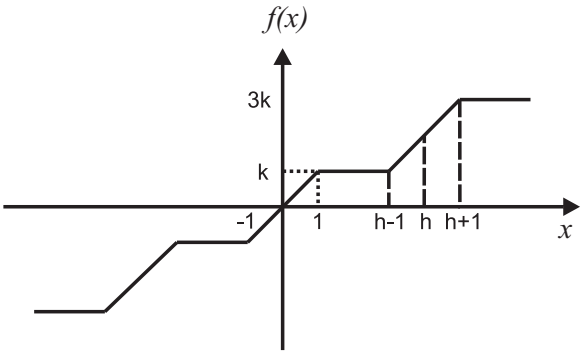

FIGURA 3.11: Gráfica de la función definida en la ecuación 2.5. Función lineal por partes para el oscilador caótico de funciones saturadas

De la ecuación (3.7), la variable $k \geq 2$ es la pendiente de la función saturada así como la constante de multiplicación para los próximos niveles de saturación satisfaciendo $\pm nk$, con n siendo un número entero par para crear un número impar de enrollamientos o n un número entero impar para crear un numero par de enrollamientos; h es el retraso las funciones saturadas anidadas en la serie de funciones saturadas. Este determina la distancia entre funciones saturadas, la cual para el caso típico debe ser función de k mediante $h = \pm mk$, donde m es un número par para enrollamientos pares y viceversa para el caso de enrollamiento impares; α define el valor del punto de ruptura; y p, q son constantes enteras positivas. En consecuencia el sistema de la ecuación (3.7) tiene la capacidad de generar $(p+q+2)$-enrollamientos [23].

Para propósitos de comparación entre los diferentes osciladores caóticos estudiados en esta Tesis, el sistema caótico de funciones saturadas se define por medio de una función para generar únicamente dos enrollamientos. Entonces, la función no lineal queda expresada de la siguiente manera:

$$f_0(x) = \begin{cases} k, & \text{si} \quad x > \propto \\ kx, & \text{si} \quad |x| \leq \propto \\ -k, & \text{si} \quad x < -\propto \end{cases} \tag{3.8}$$

representada gráficamente por la Figura 3.12

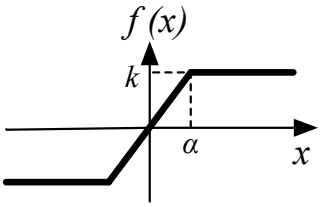

FIGURA 3.12: Representación de la función saturación a emplearse dentro del oscilador caótico de funciones saturadas para dos enrollamientos.

Partiendo de las ecuaciones (3.6) y (3.8) se adecua el oscilador caótico a la forma matricial de la ecuación (3.9) para obtener la respuesta dinámica mediante MATLAB.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a & -b & -c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f(x; \propto, k) \end{bmatrix} \tag{3.9}$$

Seleccionando a, b, c y d de 0.7; $k = 1$, y $\propto = 0.001$ se obtiene el comportamiento de la función saturada, mostrada en la Figura 3.13, para generar un atractor caótico de dos enrollamientos como se observa en la Figura 3.14.

Se observa de la Figura 3.13 el comportamiento de la función saturada que tiene límites de ± 1 con una pendiente de 89.42^o respecto al eje de las abscisas. Además, en la Figura 3.14 se observa la dinámica del oscilador caótico de funciones saturadas, la cual está conformada por un atractor caótico de doble enrollamiento, teniendo 3 puntos de equilibrio ubicados a lo largo de la variable de estado $X1$. Es importante mencionar que los puntos de equilibrio están directamente relacionados con la función saturada. Esto significa que el valor del nivel de saturación k indica la posición central de los enrollamientos, que para este caso es de ± 1. También, el punto central

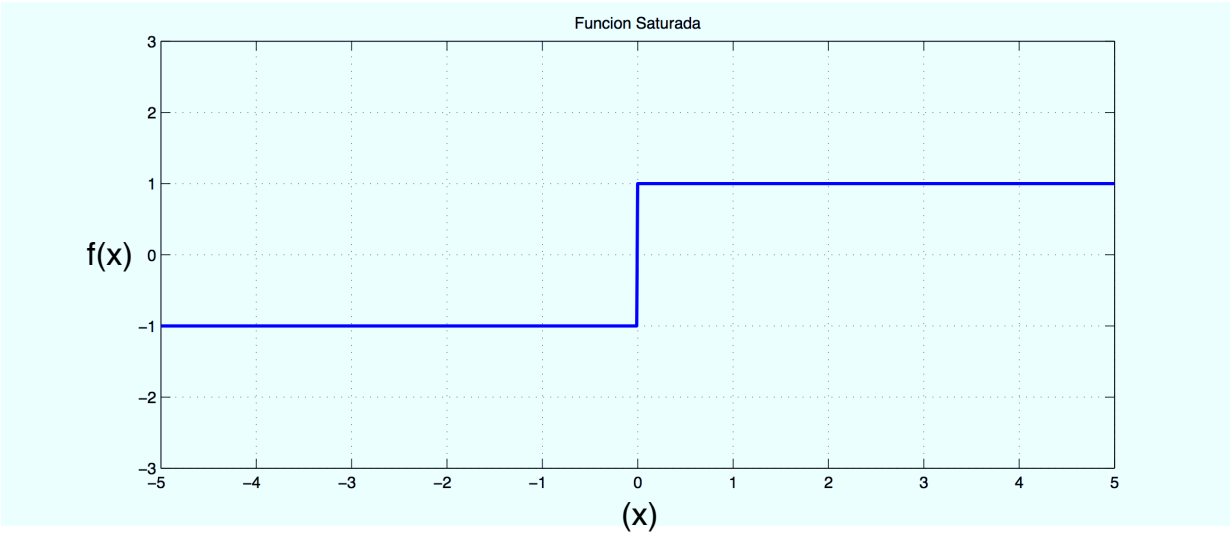

FIGURA 3.13: Función no lineal a utilizar en el oscilador caótico de función saturada.

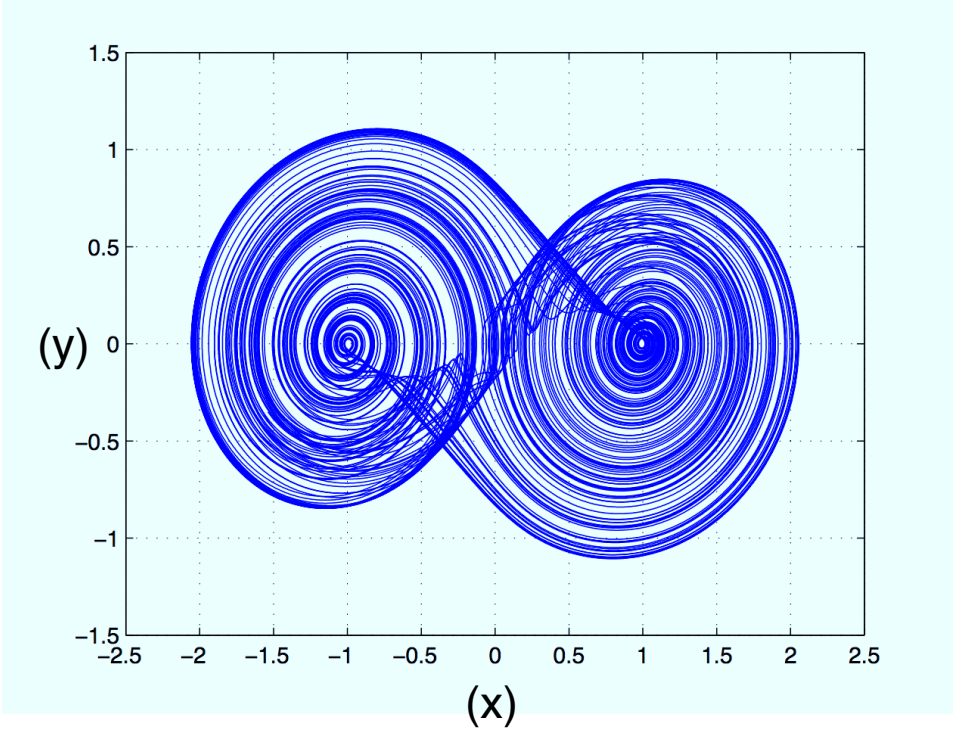

FIGURA 3.14: Resultado de la simulación para el sistema caótico de función saturada utilizando Matlab.

de la pendiente en la Figura 3.13, el cual es de cero indica el plano donde las trayectorias de los enrollamientos cambian del lado positivo al negativo y viceversa.

3.2.1. Síntesis del Oscilador Caótico de Funciones Saturadas Usando OpAmps

Análogamente al caso del oscilador caótico de Chua, se utiliza la técnica de OpAmp-RC para sintetizar con OpAmps el oscilador caótico de funciones saturadas y así comparar los resultados obtenidos con MATLAB. De esta manera, el sistema de variables de estado en (3.6) puede ser sintetizado con OpAmps manteniendo las relaciones entre los parámetros del sistema y el comportamiento de la función mediante la representación de un diagrama a bloques mostrado en la Figura 3.15.

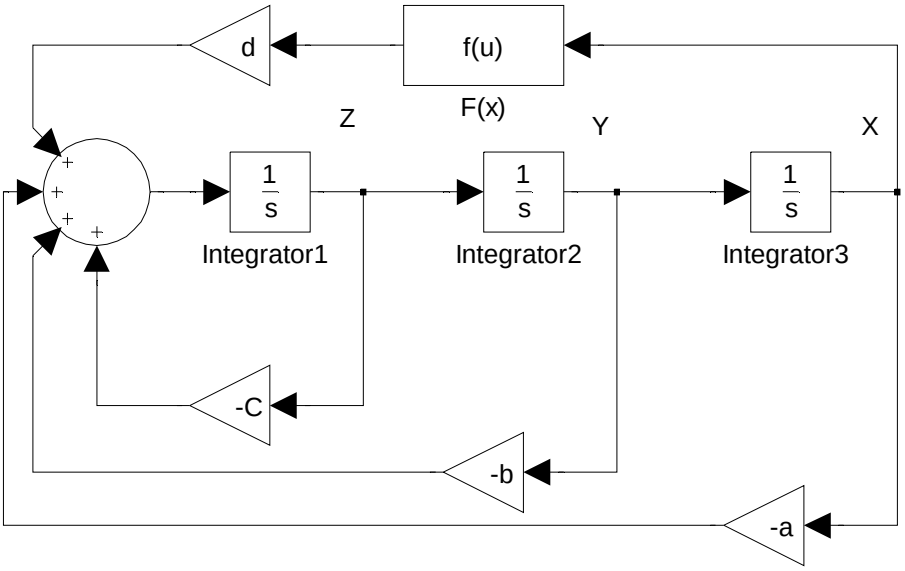

FIGURA 3.15: Diagrama a bloques utilizado para la implementación del oscilador caótico de funciones saturadas.

Los elementos electrónicos necesarios para la síntesis requiere de tres integradores, un sumador, cuatro bloques de ganancia y un bloque de función no lineal. Partiendo de este diagrama, los bloques son sintetizados por configuraciones electrónicas basadas en OpAmps. En la Figura 3.16 se muestra en resultado del proceso de síntesis para el oscilador caótico de funciones saturadas.

Está diseñado con tres integradores de voltaje en configuración inversora, un sumador de voltaje en configuración inversora, los bloques de ganancia para a, b, c, d están diseñados mediante cuatro OpAmps de ganancia en configuración inversora, dos opams inversores de voltaje y un bloque $F(x)$, el cual considera la función saturada.

Referente a la función no lineal, el diseño parte del modelo de ganancia finita del OpAmp para obtener los niveles de saturación de la función saturada. Primero se utiliza una configuración inversora de alta ganancia para obtener una salida saturada en los niveles del voltaje de saturación del OpAmp TL081. Posteriormente, estos niveles de saturación se atenuaron con un

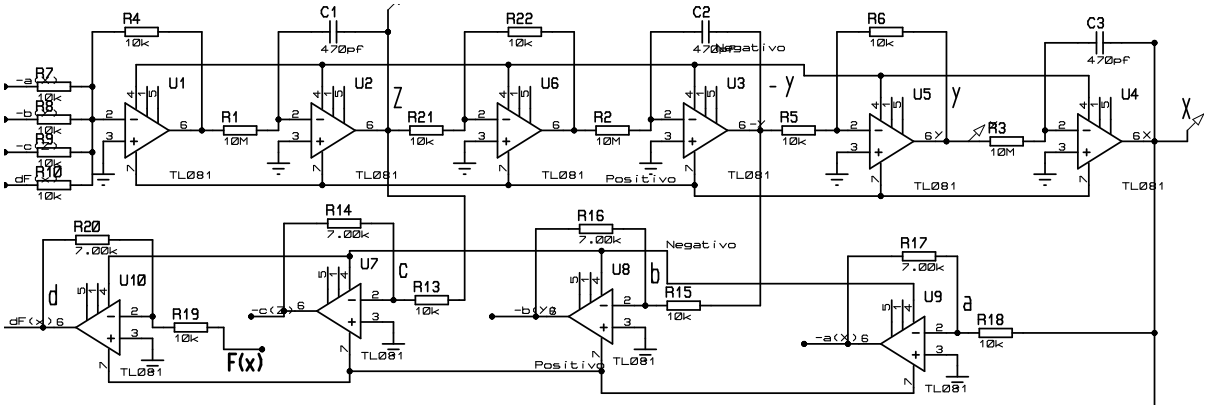

FIGURA 3.16: Esquemático del circuito caótico de funciones saturadas.

segundo OpAmp en su configuración de amplificador inversor con una ganancia siempre menor a 1, la cual debe estar relacionada con el nivel máximo y mínimo de los niveles saturados de la función saturada. El circuito resultante se muestra en la Figura 3.17.

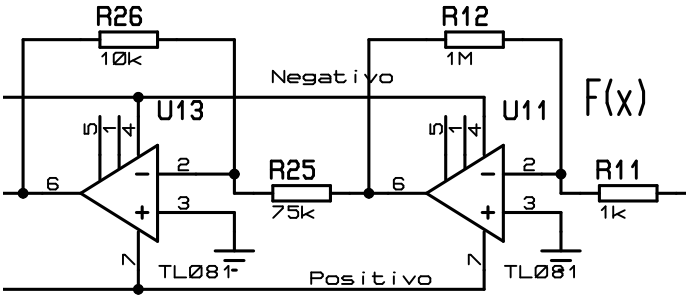

FIGURA 3.17: Esquemático del circuito para generar la función no lineal del oscilador caótico de función saturada.

La simulación por medio de SPICE del circuito de la Figura 3.17 se observa en la Figura 3.18. Este resultado es obtenido mediante proponer una ganancia de 0.13333 en la etapa que atenúa debido a que el voltaje de saturación es de $V_{sat} = \pm 7.5V$ cuando se polariza con $\pm 9V$. Lo anterior resulta en $\pm 1V$, lo cual es equivalente al nivel saturado k en la función saturada. En consecuencia, los parámetros k y $\propto$ se ajustan con las siguientes relaciones:

$$k = \frac{R26}{R25} V_{sat} \tag{3.10}$$

$$\propto = (\frac{R26}{R25} V_{sat})(\frac{R11}{R12}) \tag{3.11}$$

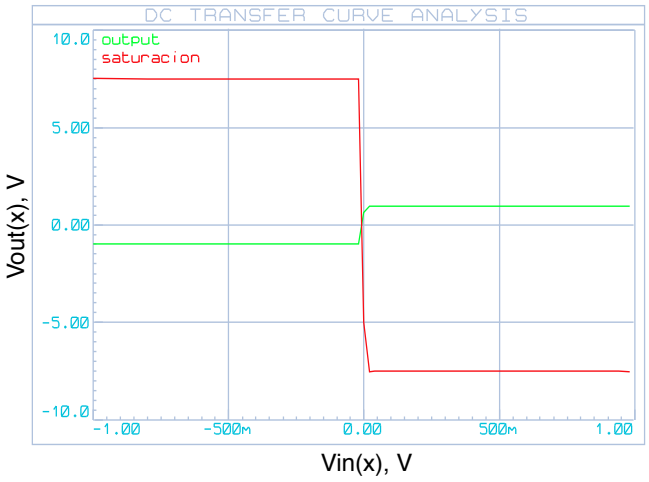

FIGURA 3.18: Función no lineal a utilizar en el oscilador caótico de función saturada. La función final ($\pm 1V$) es la atenuación invertida de la otra señal.

Por otra parte en la Figura 3.19 se muestra el atractor caótico del oscilador caótico de funciones saturadas observado en el plano de fase de voltaje utilizando la variables de estado x y y con rangos para la señal x de $\pm 2V$ el cual es controlado por el valor del parámetro k de la función no lineal; y de $\pm 1V$ para la señal y.

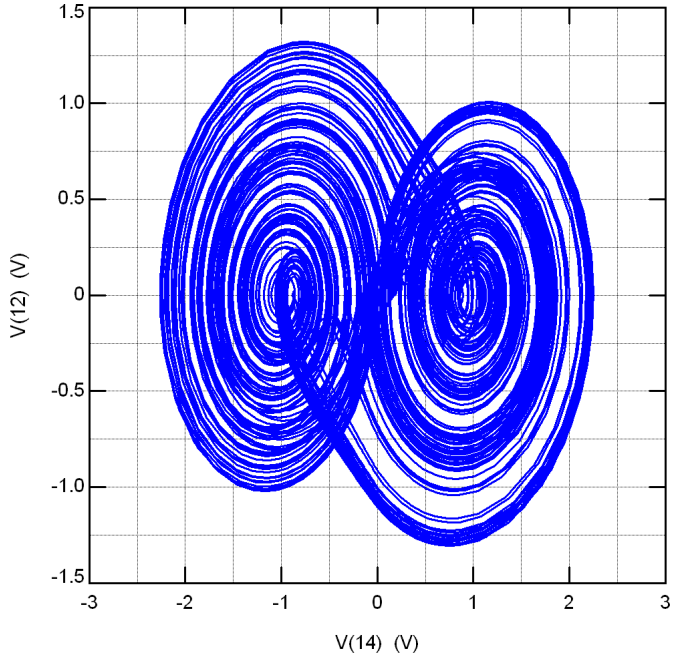

FIGURA 3.19: Resultado de la simulación del programa en TopSpice para el oscilador caótico de funciones saturadas.

3.2.2. Realización Experimental del Oscilador Caótico de Funciones Saturadas

Para el diseño de la arquitectura del generador de números aleatorios se requiere de la realización experimental del oscilador caótico de funciones saturadas. Por lo tanto, se implementan los circuitos de la Figura 3.16 y Figura 3.17 utilizando el OpAmp TL081 con fuentes de polarización de $\pm 9V$, resultando en un voltaje de saturación de $\pm 7.5V$. El resultado de la implementación de la función saturada se muestra en la Figura 3.20, donde se puede observar el valor equivalente del parámetro k correspondiente al diseño de la subsección anterior.

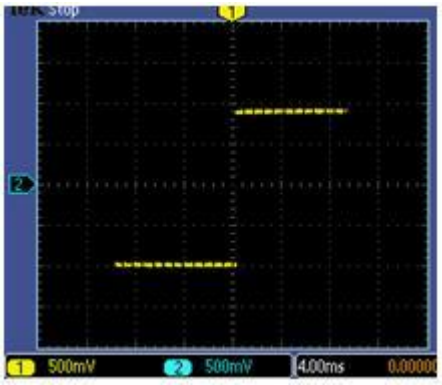

FIGURA 3.20: Resultado de la implementación de la función saturada.

El circuito completo se implementó con los valores de los elementos mostrados en la Figura 3.16 y Figura 3.17 para obtener valores de a, b, c, d, k y $\propto$ de 0.7, 0.7, 0.7, 0.7, 1 y ± 0.001, respectivamente. Es importante mencionar que los valores resistivos no comerciales fueron obtenidos mediante el uso de trimpots (potenciómetros de precisión). El resultado de la implementación se muestra en la Figura 3.21, el cual demuestra que la dinámica caótica es equivalente tanto con el modelo matemático simulado en MATLAB, como con la síntesis con OpAmps simulada en SPICE. Similar al caso anterior, se comprueba la correlación de los resultados obtenidos a diferentes niveles de jerarquía.

3.3. Oscilador Caótico de Función Diente de Sierra

Actualmente, una de las áreas de investigación en sistemas caóticos se enfoca la exploración de nuevas opciones para obtener un comportamiento caótico mediante el estudio de osciladores caóticos basados en funcionciones llamadas 'jerk', las cuales definen la derivada de la aceleración con respecto del tiempo [24]. Dentro de este tipo de osciladores caóticos implementados, se muestran diversas opciones de funciones no lineales o PWL para ser implementadas dentro del oscilador caótico. Una de estas funciones es la función diente de sierra. Basados en [25] las

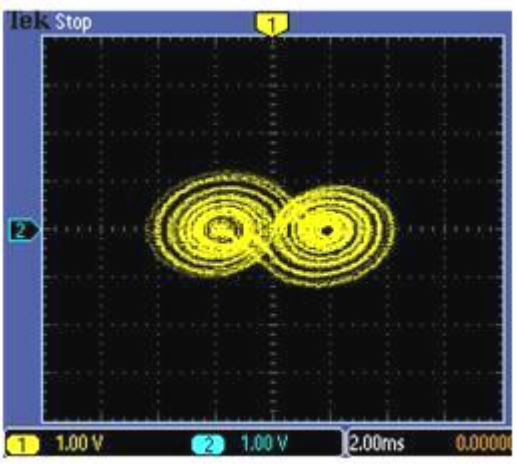

FIGURA 3.21: Resultado de la implementación del oscilador caótico de funciones saturadas.

ecuaciones del oscilador caótico de función diente de sierra se definen a continuación.

$$\begin{aligned} \dot{x} &= \alpha[y - f(x)] \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y \end{aligned} \tag{3.12}$$

Donde $\alpha > 0$, $\beta > 0$ son los parámetros del sistema y $f(x)$ es la función no lineal, cuyo comportamiento general se define la ecuación (3.13) y (3.14) con el objetivo de generar múltiples enrollamientos. La ec. (3.13) es para generar $2N$-enrollamientos, mientras que la ec. (3.14) es para generar $2N+1$-enrollamientos. La representación de esta funciones se muestra en la Figura 3.22 y Figura 3.23, respectivamente.

$$f(x) = \xi \left\{ x - A_1[-sgn(x) + \sum_{i=0}^{N-1} (sgn(x + 2iA_1) + sgn(x - 2iA_1))] \right\} \tag{3.13}$$

$$f(x) = \xi \left\{ x - A_1[\sum_{i=0}^{N-1} (sgn(x + (2i+1)A_1) + sgn(x - (2i+1)A_1))] \right\} \tag{3.14}$$

El valor de ξA_1 se refiere a la amplitud de la función diente de sierra, como se puede verificar en la Figura 3.22 y Figura 3.23. Nuevamente, se enfatiza el empleo de osciladores caóticos de dos enrollamientos únicamente con el propósito de comparar los resultados. En consecuencia, la función diente de sierra utilizada en esta Tesis es descrita por la ecuación (3.15) y su respuesta es mostrada en la Figura 3.24 [25].

$$f(x) = \xi(x - A_1 sgn(x)) \tag{3.15}$$

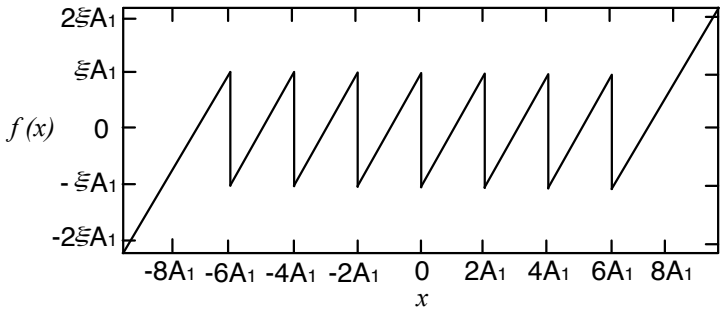

FIGURA 3.22: Representación gráfica de la función diente de sierra creada de la ecuación (3.13).

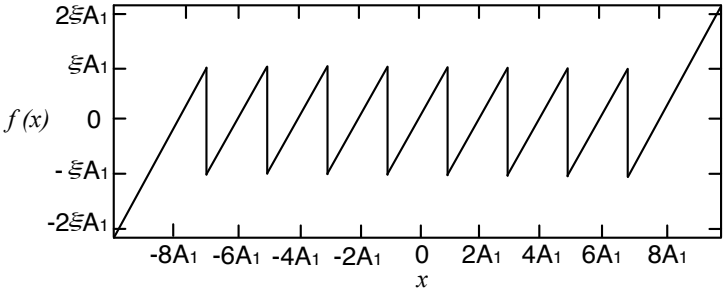

FIGURA 3.23: Representación gráfica de la función diente de sierra creada de la ecuación (3.14).

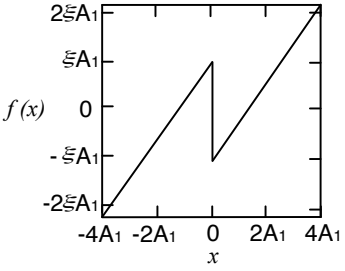

FIGURA 3.24: Función de diente de sierra para generar dos enrollamientos definida en la ecuación (3.15).

Siendo ξ y A_1 los parámetros de la función diente de sierra y $sgn(x)$ representa la función *signum*. Para realizar el modelado del sistema mediante MATLAB es necesario tener su representación de espacio de estados, la cual queda conformada de la siguiente manera:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & \alpha & 0 \\ 1 & -1 & 1 \\ 0 & -\beta & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\alpha f(x) \\ 0 \\ 0 \end{bmatrix} \tag{3.16}$$

Utilizando $\alpha = 10$, $\beta = 25$ para los parámetros del sistema y $\xi = 0.25$ y $A_1 = 0.5$ para los parámetros de la función diente de sierra, se obtienen los resultados numéricos (Figura 3.25 y Figura 3.26) del comportamiento de la función diente de sierra para generar dos enrollamientos y el atractor caótico, respectivamente.

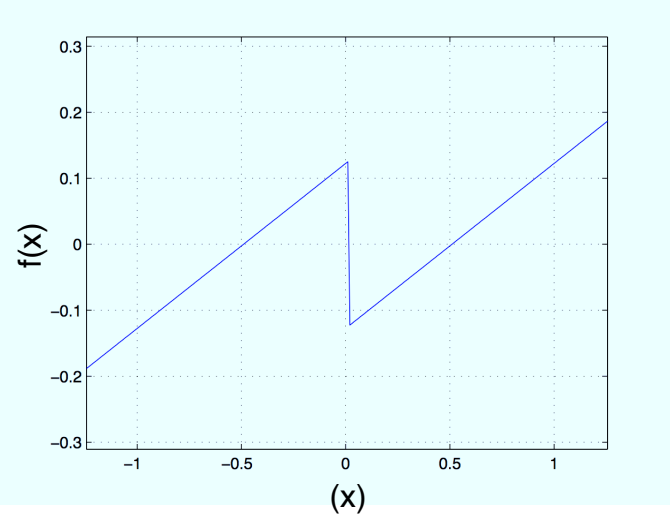

FIGURA 3.25: Función no lineal que se ocupa para el oscilador caótico de diente de sierra.

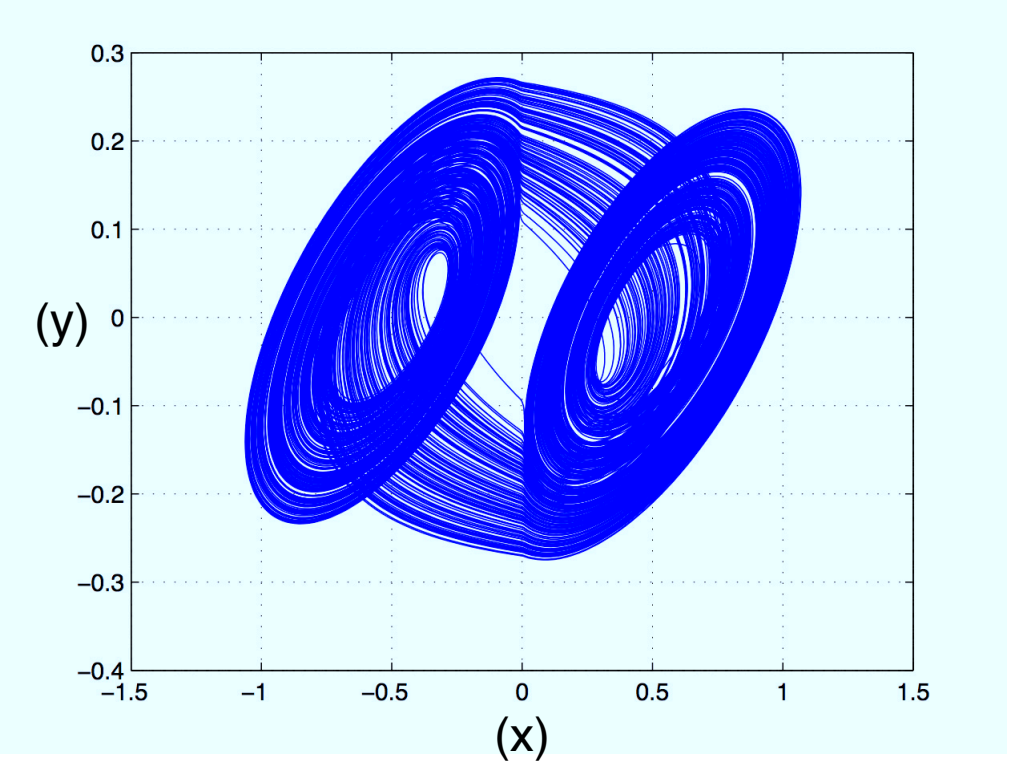

FIGURA 3.26: Resultado de la simulación del Oscilador caótico de diente de sierra mediante Matlab.

Se observa de la Figura 3.25 el comportamiento de la función saturada que tiene límites definidos por los valores de los parámetros ξ y A_1. Además, en la Figura 3.26 se observa la dinámica del oscilador caótico de función diente de sierra, la cual está conformada por un atractor caótico de doble enrollamiento. Es importante mencionar que los puntos de equilibrio están directamente relacionados con la función *signum*. Esto significa que el valor del parámetro A_1 indica la posición central de los enrollamientos, que para este caso es de ± 0.5.

3.3.1. Síntesis del Oscilador Caótico de Funcion Diente de Sierra Usando OpAmps

Al igual que los dos casos anteriores, se requiere utilizar la técnica de OpAmp-RC para sintetizar el oscilador caótico de función diente de sierra y así comparar los resultados obtenidos con MATLAB. En consecuencia, el sistema de variables de estado en (3.16) puede ser sintetizado con OpAmps manteniendo las relaciones entre los parámetros del sistema y el comportamiento de la función mediante la representación de un diagrama a bloques mostrado en la Figura 3.27.

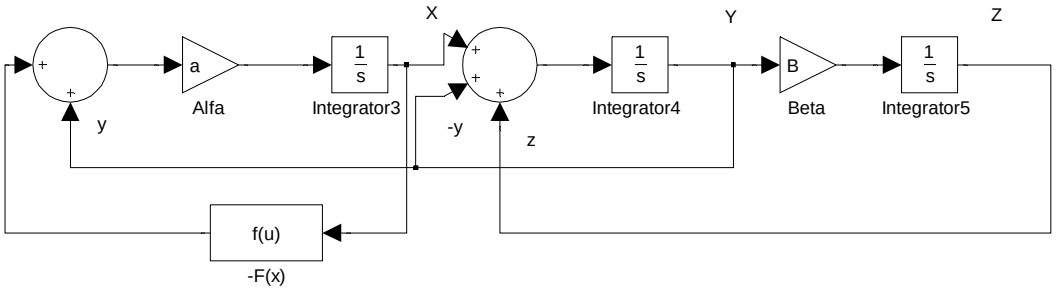

FIGURA 3.27: Diagrama a bloques usado para la implementación del oscilador caótico de función diente de sierra.

El diagrama a bloques obtenido es similar a aquella utilizada en el oscilador caótico de Chua. El flujo de señal es representado por las flechas y su dirección respectiva, lo cual es necesario para decidir el tipo de configuración electrónica a utilizar en el proceso de síntesis. Del diagrama de la Figura 3.27, se obtiene el circuito de la Figura 3.28.

El diseño electrónico para el oscilador caótico de función diente de sierra está formado por tres integradores de voltaje en configuración inversora, los bloques para las ganancias de $\alpha = 10$, $\beta = 25$ están diseñados implícitamente en el sumador de voltaje y explícitamente por un amplificador de ganancia inversora, respectivamente.

Para el diseño de la función diente de sierra se puede abordar el problema relativamente similar a lo mostrado en los casos previos. Primero se contempla la definición de la función diente de sierra donde está incrustada la función *signum* $sgn(x)$. Para este propósito esta función es

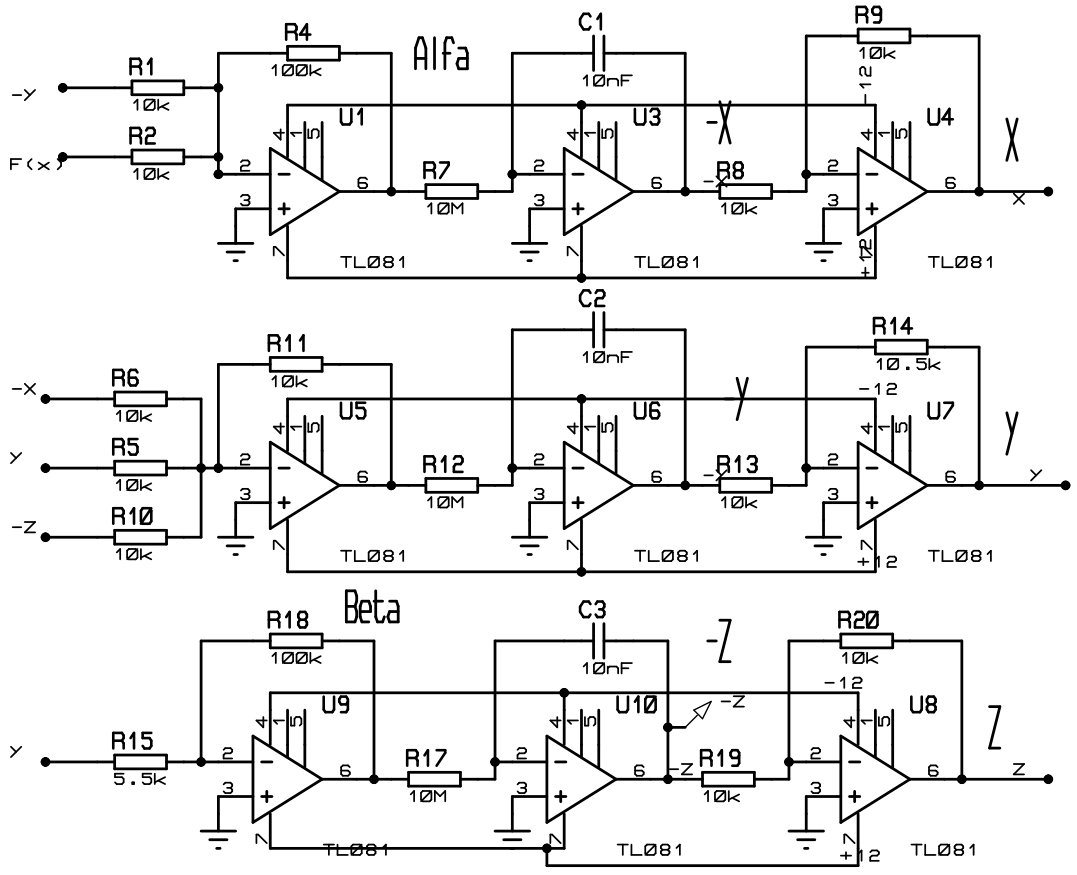

FIGURA 3.28: Diagrama de conexión empleado para la síntesis del oscilador caótico de función diente de sierra.

obtenida directamente con un OpAmp en su configuración de comparador de voltaje. Ahora, también derivada de la misma definición de la función diente de sierra, se requiere de la resta de dos valores $(x - A_1 sgn(x))$ lo cual motiva a la utilización de un OpAmp en su configuración como sumador. Por último, el factor multiplicador ξ se puede obtener implícitamente con la ganancia del mismo OpAmp usado como sumador o si es deseable puede ser explícitamente diseñada por un OpAmp independiente configurado con una ganancia del valor igual al factor multiplicativo. Recopilando los puntos anteriores, la función diente de sierra es diseñada por el circuito de la Figura 3.29.

La simulación por medio de SPICE del circuito de la Figura 3.29 se observa en la Figura 3.30. Para tener control sobre la forma de la función diente de sierra para cambiar o establecer valores requeridos se encuentran las ecuaciones (3.17) y (3.18).

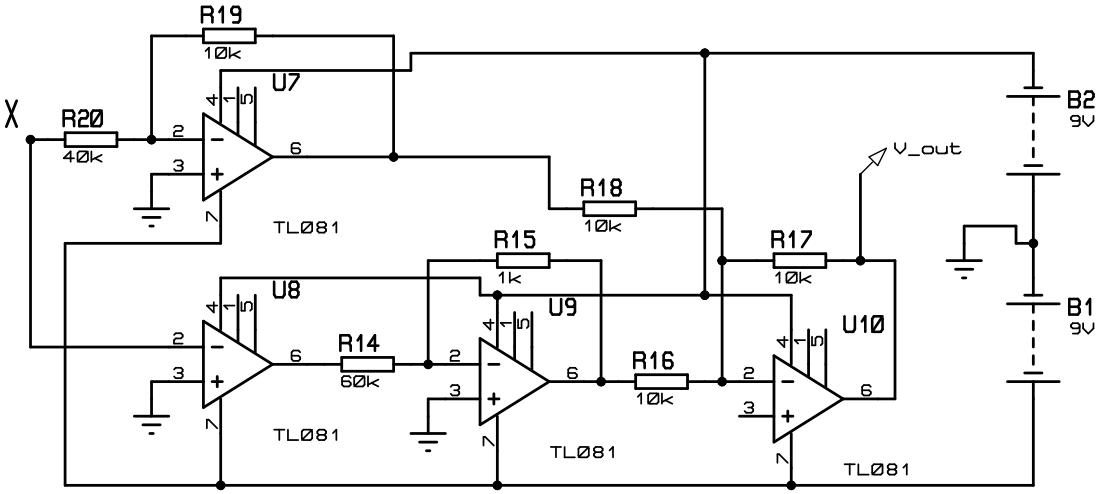

FIGURA 3.29: Circuito para implementar la función no lineal de diente de sierra.

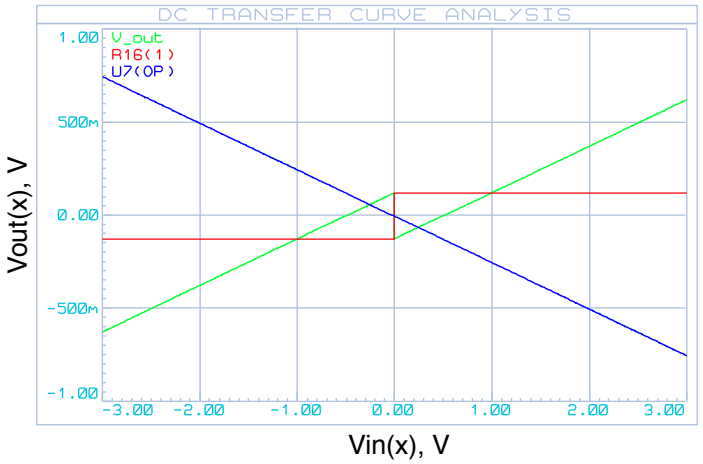

FIGURA 3.30: Respuesta del circuito para implementar la función no lineal de diente de sierra. La función final (línea verde) es la suma invertida de las otras dos funciones.

$$\xi A_1 = Amplitud = \frac{R_{15}}{R_{14}} V_{sat} \tag{3.17}$$

$$2A_1 = Periodo = 2 \cdot Amplitud / \frac{R_{19}}{R_{20}} \tag{3.18}$$

El periodo de la función diente de sierra se deduce con el valor de $2A_1$ y por otro lado el valor de ξA_1 representa la amplitud de la función, como se puede verificar en la Figura 3.24. Por lo que

al proponer valores para los resistores $R15$, $R14$, $R19$ y $R20$ con los mostrados en la Figura 3.29 es posible obtener un $A_1 = 0.5$ y un $\xi = 0.25$ con un $V_{sat} = 7.5V$.

Por otra parte en la Figura 3.31 se muestra el atractor del oscilador caótico de función diente de sierra, observado en el plano de fase de voltaje utilizando la variables de estado x y y con rangos para la señal x de $\pm 1V$, el cual es controlado por el valor de los parámetros ξ y A_1 de la función no lineal; y de $\pm 250mV$ para la señal y.

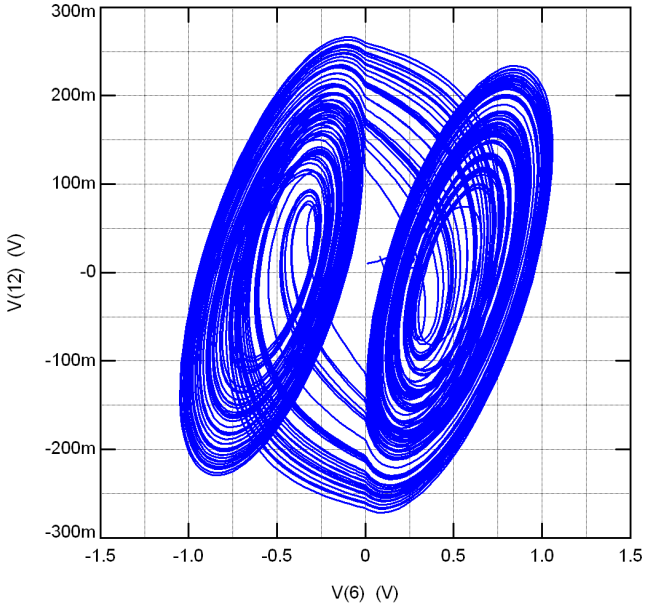

FIGURA 3.31: Respuesta de la síntesis del oscilador caótico de diente de sierra.

3.3.2. Realización Experimental del Oscilador Caótico de Diente de Sierra

Para el diseño de la arquitectura del generador de números aleatorios se requiere de la realización experimental del oscilador caótico de diente de sierra. Por lo tanto, se sigue la misma metodología que en la seccion 3.2.2. Utilizando el mismo tipo de OpAmp, voltajes de polarización y voltaje de saturación. El circuito completo se implementó con los valores de los elementos mostrados en la Figura 3.28 y Figura 3.29 para obtener valores de α, β, ξ, A_1 de 10, 25, 0.25, y 0.5, respectivamente.

El resultado de la implementación de la función saturada se muestra en la Figura 3.32, donde se puede observar el valor equivalente del parámetro ξ y A_1 correspondiente al diseño de la subsección anterior. El resultado de la implementación se muestra en la Figura 3.33, el cual demuestra que la dinámica caótica es equivalente tanto con el modelo matemático simulado en Matlab, como con la síntesis con OpAmps simulada en SPICE. Similar al caso anterior, se comprueba la correlación de los resultados obtenidos a diferentes niveles de jerarquía.

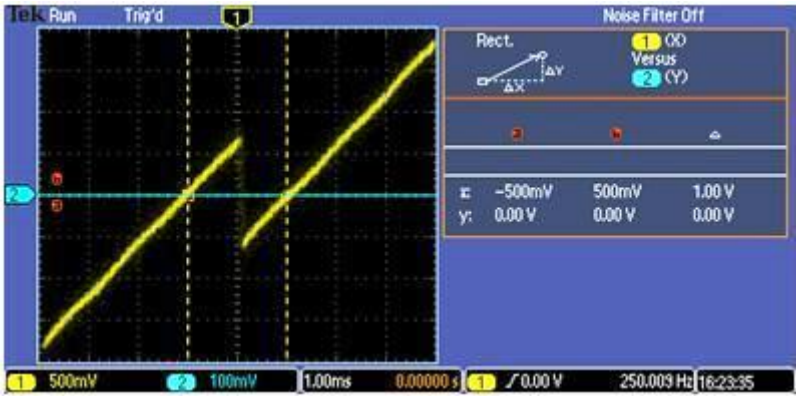

FIGURA 3.32: Resultado de la Implementación de la función diente de sierra.

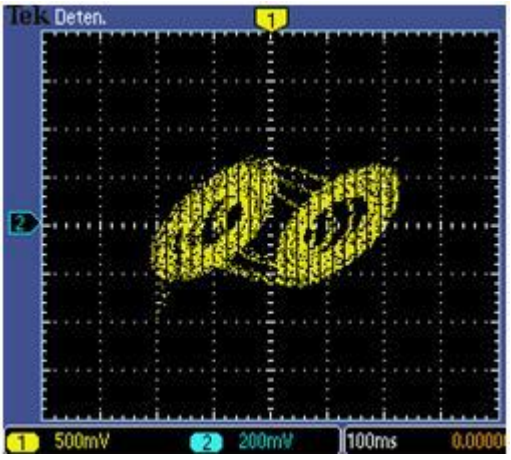

FIGURA 3.33: Resultado de la implementación del oscilador caótico de diente de sierra.

3.4. Escalamiento de la Frecuencia

Un punto importante a considerar en cada oscilador caótico es el tema de la frecuencia fundamental de oscilación para cada oscilador. En esta Tesis se toman como referencia los artículos en [17], [25], [26] para realizar el escalamiento en frecuencia. Similar al escalamiento en frecuencia realizado en las redes de dos puertos, donde el efecto de reescalar la frecuencia en un factor de FS es disminuir el valor de cada inductancia y capacitancia por el factor referido, mientras que las impedancias resistivas y conductivas así como las relaciones de ganancia de voltaje y de corriente permanecen inalteradas por el escalamiento.

En específico para los sistemas caóticos considerados en este trabajo de Tesis, el escalamiento de frecuencia consiste en multiplicar los sistemas de variables de estado por un factor requerido de escalamiento. En la implementación a nivel OpAmp, esto se realiza sobre los capacitores de cada integrador de voltaje de los diferente circuitos electrónicos para los osciladores caóticos mostrados en las Figs. 3.5, 3.16, 3.28 . Entonces suponiendo un valor de capacitancia C, la relación $C_{fs} = C/FS$, produce el escalamiento en frecuencia deseado, donde C_{fs} es la nueva

capacitancia requerida para cumplir con el factor de escalamiento FS. No obstante se debe recordar que el escalamiento de frecuencia está limitado por el ancho de banda finito de cada OpAmp, no pudiendo obtener señales con mayor frecuencia que este límite establecido [23].

El método propuesto es simple en funcionamiento, mientras menor sea el valor de la capacitancia más rápido será nuestro sistema. Cabe señalar que dentro del mismo oscilador caótico el valor de cada capacitor debe ser igual en todos los OpAmps configurados como integradores de voltaje, de lo contrario cada integrador trabaja a frecuencia diferente y el comportamiento total del sistema se degrada.

En las Figuras 3.34- 3.36, se muestra el espectro en frecuencia perteneciente a cada oscilador caótico. Se observa que cada uno está funcionando a diferente frecuencia.

Se opto la implementación a diferentes frecuencias debido a que cada oscilador caótico según los parámetros en los que se encuentre era capaz de tolerar una frecuencia máxima. Por lo tanto, se eligió una frecuencia donde los exponentes de Lyapunov se encuentren dentro de rangos similares para cada oscilador caótico.

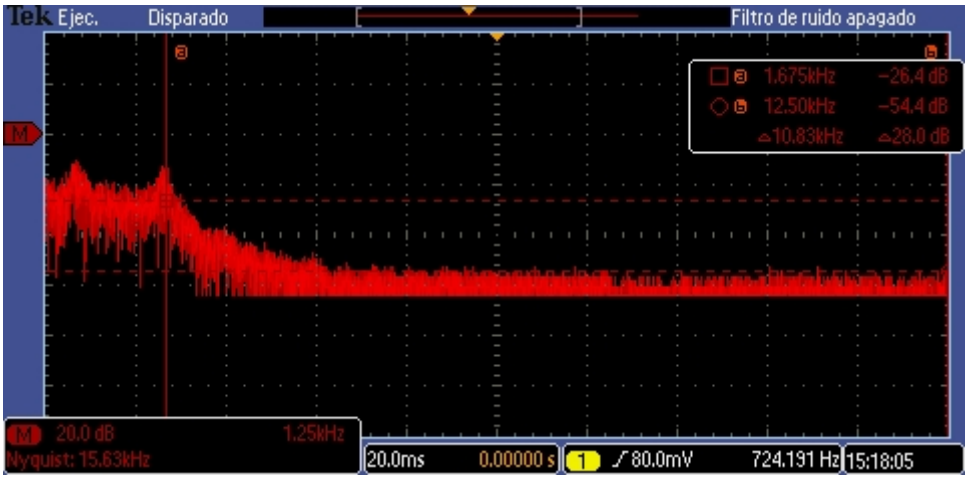

FIGURA 3.34: Espectro en frecuencia de la implementación del oscilador caótico de función saturada.

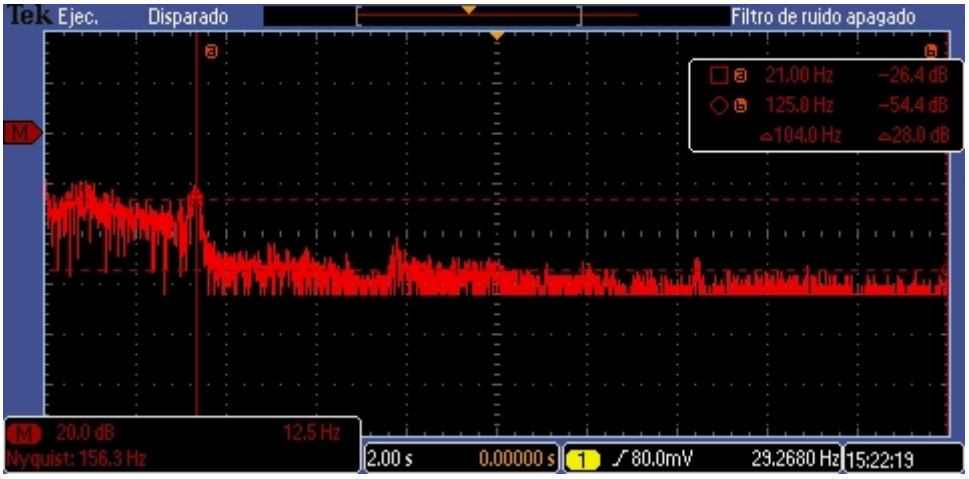

FIGURA 3.35: Espectro en frecuencia de la implementación del oscilador caótico de Chua.

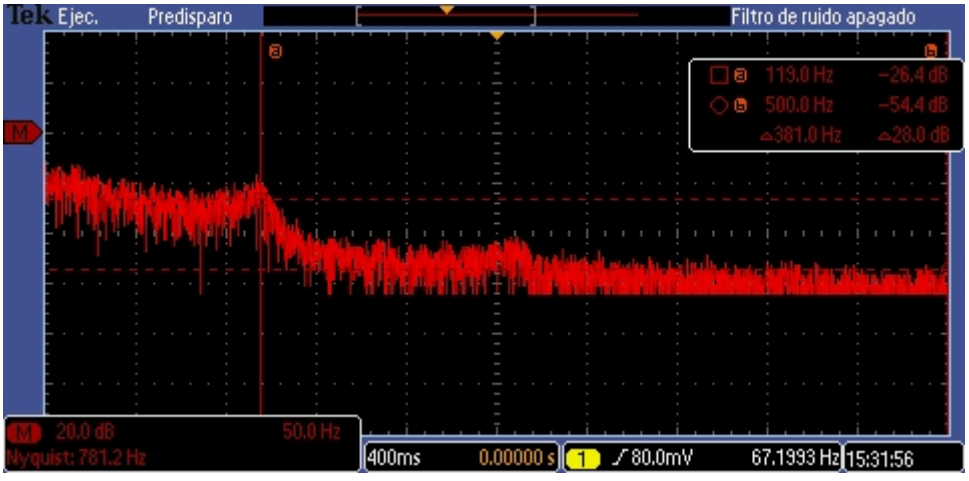

FIGURA 3.36: Espectro en frecuencia de la implementación del oscilador caótico de diente de sierra.

Capítulo 4

Diseño e Implementación de un TRNG basado en Caos

Dentro de los trabajos donde se han propuesto generadores de números aleatorios basados en caos generalmente se componen de dos formas, la primera es usar un oscilador caótico junto con otro oscilador normal o generador de funciones empleado para la frecuencia de muestreo, además de incorporar una compuerta o arreglo de compuertas lógicas a la salida. La segunda es utilizando la señal en tiempo continuo, proveniente del oscilador caótico y mediante un establecimiento de comparadores de voltaje y reglas de asociación se obtiene el valor final de la cadena de bits [13], [27], [28] .

Este trabajo se basará en los generadores de números aleatorios de la segunda forma. En la Figura 4.1 se muestra un diagrama a bloques de la arquitectura del generador de números aleatorios propuesto y en seguida se explica el funcionamiento de cada bloque en particular [27].

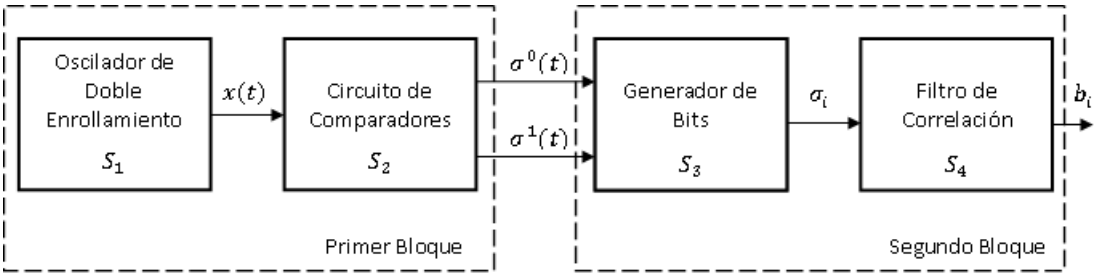

FIGURA 4.1: Generador de números verdaderamente aleatorios propuesto.

El primer bloque hace referencia a aquellos componentes implementados en hardware, estos componentes son completamente constituidos de circuitos electrónicos. Ahora, dentro de este bloque existe una división de dos bloques más, los cuales se explican a continuación. El primer cuadro representa al oscilador caótico de doble enrollamiento. Aunque la arquitectura del

generador de números aleatorios es la misma siempre, este bloque en particular va a ser el único que sufrirá algún cambio. Se propone la utilización de tres osciladores caóticos de doble enrollamiento, sin embargo eso no significa que estarán mezclados, el punto es utilizar cada oscilador caótico en forma independiente, uno a la vez, para así poder medir individualmente cada oscilador en su desempeño al momento de generar números aleatorios. Entonces, básicamente este bloque es el oscilador caótico que se elija, funcionando y entonado con los valores en sus parámetros que garanticen un exponente de Lyapunov alto, mientras se deja funcionar sin otra mayor modificación.

El segundo cuadro es un bloque con un circuito que a su respuesta nos emule el resultado de dos funciones de umbral. Generalmente una función de umbral al ser implementada en circuito (Threshold Circuit) es aquella que da como resultado un valor de 1 si una función especificada o elementos establecidos exceden un valor, y da un valor de 0 de otra forma [28].

Para poder obtener la respuesta deseada necesitamos de tres elementos, primero la señal que pueda exceder o no el valor de umbral, después es necesario establecer el valor de umbral y por último requerimos el uso de dispositivos para poder tener la respuesta de 1 y 0 lógico. Además, el circuito a ocuparse debe ser capaz de poder modificar sus parámetros para poder variar, establecer y controlar a que nivel deseémos que responda con un 1 lógico o con un 0 lógico.

Primero, la señal que se utilizará será aquella proveniente del oscilador caótico que se ocupe según sea cada caso, específicamente la señal de la variable de estado que representa a X. El segundo elemento, los valores de umbral c_1 y c_2, serán establecidos en base a los niveles de voltaje de la variable de estado de cada oscilador caótico en función de sus puntos de equilibrio, definiendo su respuesta en la ecuación (4.1) y ejemplificándolo en la Figura 4.2.

$$S_2 : \quad \begin{array}{ll} \sigma^1(x(t)) & \begin{cases} 0 & \text{si } x(t) < c_1 \\ 1 & \text{si } x(t) \geq c_1 \end{cases} \\ \sigma^0(x(t)) & \begin{cases} 0 & \text{si } \quad x(t) > c_2 \\ 1 & \text{si } \quad x(t) \leq c_2 \end{cases} \end{array} \tag{4.1}$$

Concretamente, los niveles de umbral serán establecidos por un aspecto en particular:

- Los osciladores caóticos utilizados son todos de doble enrollamiento y simétricos. Además de tener sus enrollamientos en puntos específicos definidos por los puntos de equilibrio del atractor.

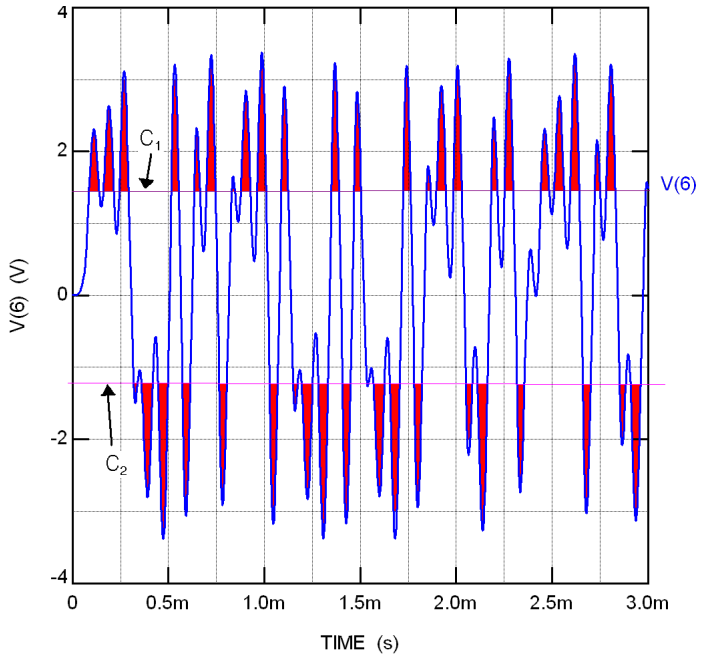

FIGURA 4.2: Proceso de muestreo en tiempo de la señal X del oscilador caótico usando los niveles de umbral c_1 y c_2.

Por lo anterior al ser simétricos permite asegurar teóricamente que el rango de voltaje positivo será igual al rango de voltaje negativo. Entonces al momento de establecer el voltaje de umbral positivo será el mismo valor pero de signo opuesto a aquel para el voltaje de umbral negativo. Para establecer el nivel de voltaje se ocuparán los puntos donde se efectúa el enrollamiento de cada oscilador. Esto último es mejor ejemplificado en la Figura 4.3.

En la Figura 4.3 (izquierda), que es la gráfica de fase de las señales X_1 y X_2 de un oscilador caótico, se puede observar que cada enrollamiento está formado alrededor del punto que cruza cada línea vertical. En la figura 4.3 (derecha), se presenta la gráfica con respecto al tiempo de las señales X_1 y X_2, los valores de voltaje de umbral están representados como las dos líneas horizontales que cruzan la imagen y se observa que en esos niveles es donde se mantiene la señal oscilando tanto para el valor de voltaje positivo como para el valor de voltaje negativo.

Para poder obtener la salida deseada, que es la función (4.1), se propone el circuito de la Figura 4.4. Este circuito prácticamente ocupa dos OpAmp configurados como comparadores de voltaje. Los potenciómetros son utilizados para poder modificar el nivel de voltaje al cual cada OpAmp dará como respuesta un valor de 1 o un valor de 0. La conexión en cada potenciómetro varía según el nivel que tenga que comparar, ya sea un voltaje con nivel positivo o un voltaje con nivel negativo. Así, el potenciómetro conectado al OpAmp encargado de comparar el nivel de umbral c_1, estará alimentado de $0V$ a $+9V$, mientras que el otro conectado al OpAmp que se encargue de la comparación para el nivel de umbral negativo c_2 estará alimentado de $0V$ a $-9V$.

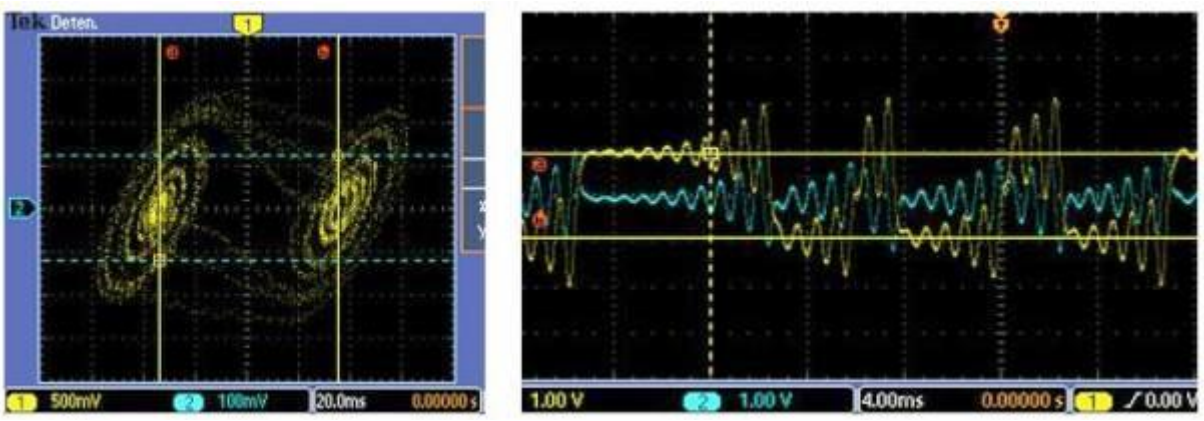

FIGURA 4.3: Izquierda, imagen de diagrama fase para las señales X_1 y X_2 de un oscilador caótico donde los valores de umbral son las dos líneas verticales que cruzan la imagen. Derecha, imagen de las señales X_1 y X_2 con respecto del tiempo en un oscilador caótico, los valores de umbral son las dos líneas horizontales que cruzan la imagen. Se puede notar que alrededor de los valores de umbral es donde de forman los enrollamientos del oscilador caótico.

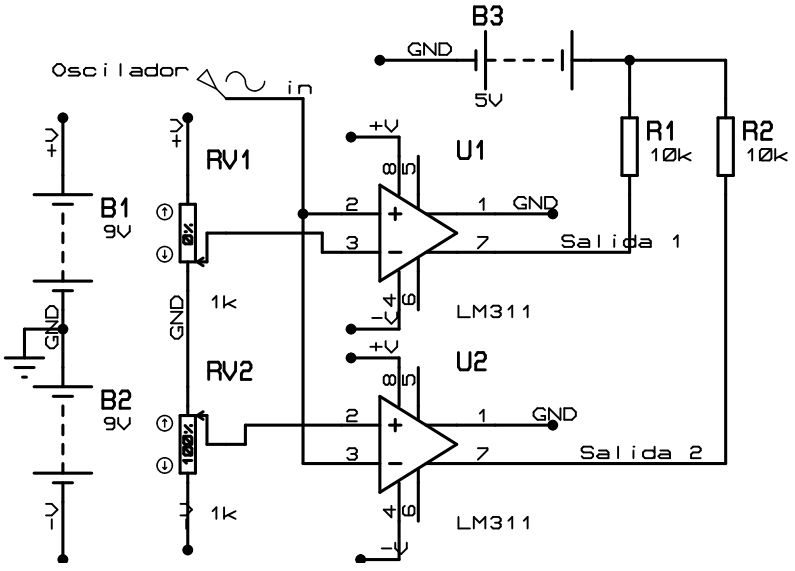

FIGURA 4.4: Implementación basada en comparadores usados para llevar a cabo la función 4.1.

En resumen, para este primer bloque se ocupará el oscilador caótico, en específico solamente la señal perteneciente a la variable X, después se pasará a través de comparadores, los cuales antes estarán entonados según sea el nivel de umbral para que a su respuesta otorguen un valor lógico de 1 o 0. Un ejemplo de la respuesta es mostrado en la Figura 4.5 y su realización experimental se muestra en las Figuras 4.6, 4.7 y 4.8.

En lo que respecta al segundo bloque de la imagen 4.1, representa la parte donde se incorpora tanto software como hardware al generador de números aleatorios y también está dividida en dos partes.

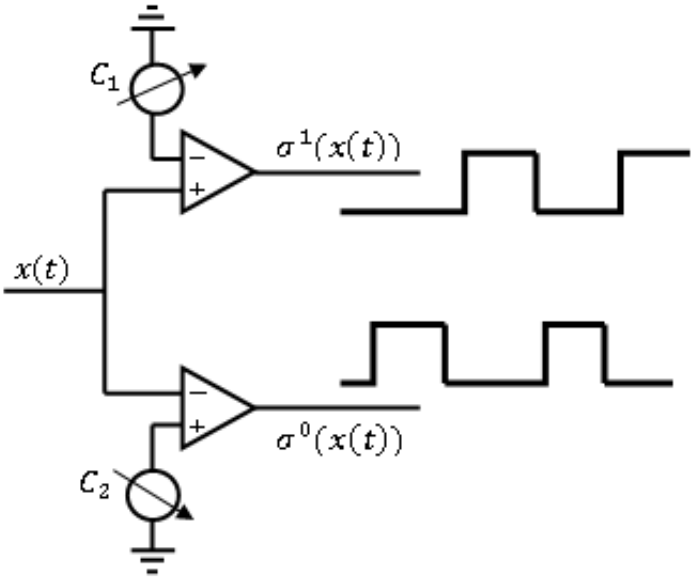

FIGURA 4.5: Respuesta del primer bloque del generador de bits aleatorios.

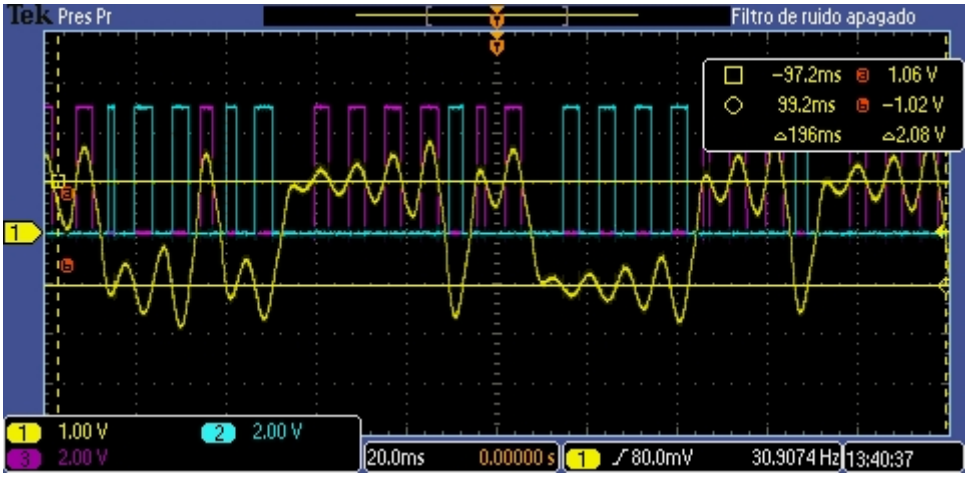

FIGURA 4.6: Resultado para el oscilador caótico de Chua. Se puede ver los niveles de umbral y pulsos positivos y negativos

El primero es un generador de bits simple el cual tiene dos entradas, una correspondiente al nivel de umbral positivo y otra correspondiente al nivel de umbral negativo. La tarea que lleva a cabo este bloque es establecer una regla de asociación entre estos dos valores obtenidos a la entrada. La regla de asociación es la siguiente:

$$S_3 : \sigma^i(\sigma^0, \sigma^1) \begin{cases} 0 & \text{si} \quad \sigma^0 = 0 \quad \text{y} \quad \sigma^1 = 1 \\ 1 & \text{si} \quad \sigma^1 = 0 \quad \text{y} \quad \sigma^0 = 1 \end{cases} \tag{4.2}$$

Nótese que esta función no toma en cuenta si es que los dos valores de entrada son 0, y por otro lado no existe valor si es que fueran las dos entradas iguales a 1, ya que es teóricamente

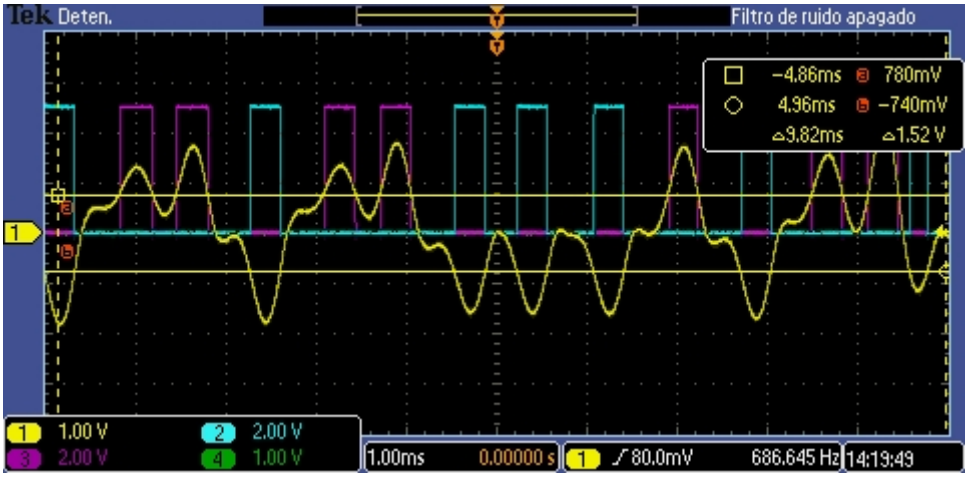

FIGURA 4.7: Resultado para el oscilador caótico función saturada. Se puede ver los niveles de umbral y pulsos positivos y negativos

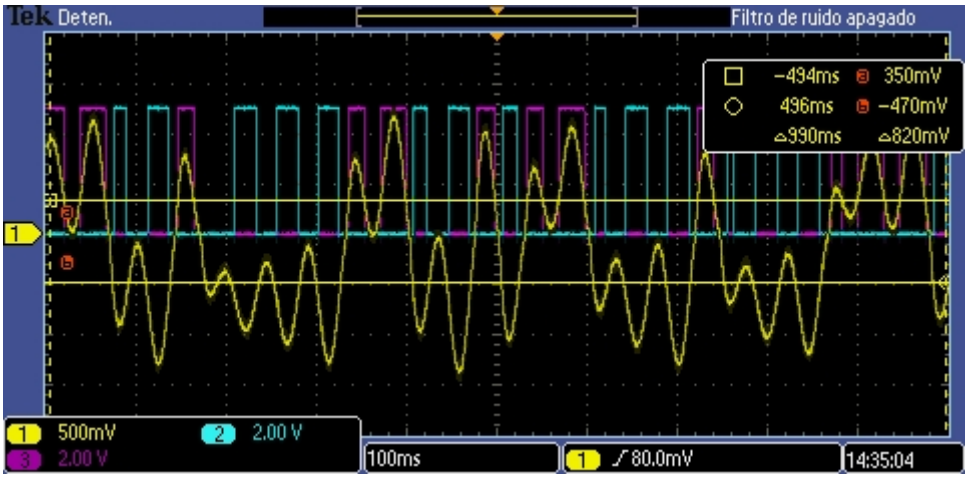

FIGURA 4.8: Resultado para el oscilador caótico de función diente de sierra. Se puede ver los niveles de umbral y pulsos positivos y negativos

imposible. Entonces está es la primera parte que se elabora en software, tomar las dos entradas y obtener un resultado como el definido por la función (4.2), guardar ese valor y pasarlo a la última parte de la función de este cuadro está representada en la figura 4.9.

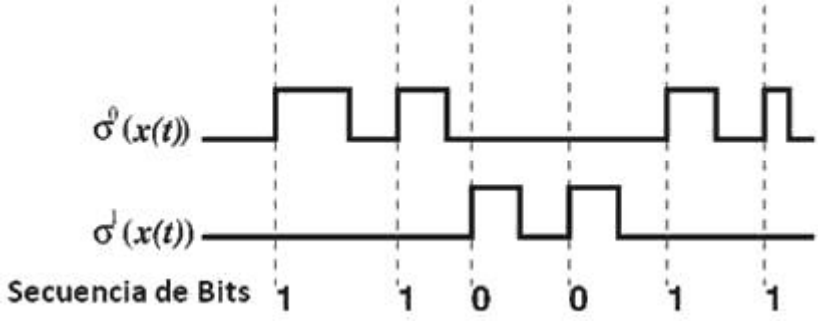

FIGURA 4.9: Respuesta del cuadro S_3 por la regla antes definida en 4.2.

Una fuente natural de aleatoriedad podría no tener una completa imparcialidad como respuesta directa. Por lo tanto existen técnicas para extraer esta predisposición que es un defecto dentro del generador de número aleatorios, sea o no sea conocido este patrón erróneo. Generalmente son llamados técnicas de corrección de preferencia (de-skewing techniques), y se usan para eliminar la correlación en la salida de fuentes naturales de aleatoriedad. Uno de los primeros autores en proponer teoría sobre números aleatorios fue Von Neumann, el propuso un post-procesamiento digital que balanceara la distribución de los bits. Su filtro fue sencillo, se toman parejas de bits y a la pareja con valor de 01 la convertía en un 0, la pareja con un 10 la convertía en un 1, mientras que las parejas de 00 y 11 eran descartadas. Expresado en otra forma está la ecuación (4.3), [13], [27].

$$S_4 : b^i(\sigma^{i-1}, \sigma^i) \begin{cases} 1, & \text{si} \quad \sigma^{i-1} = 0 \quad \wedge \quad \sigma^i = 1 \\ 0, & \text{si} \quad \sigma^{i-1} = 1 \quad \wedge \quad \sigma^i = 0 \end{cases} \tag{4.3}$$

Este último bloque es programado sobre una computadora para la recolección y almacenamiento de bits aleatorios, con el fin de su posterior utilización al ser probados en test de aleatoriedad o en un microcontrolador para su uso en la aplicación dentro del robot móvil. Un punto importante es establecer el tiempo de muestreo para cada oscilador caótico. En la Figura 4.10 se puede ver la señal X_1 con respecto del tiempo y un tren de pulsos que es el generado por el comparador σ^0 encargado del voltaje de umbral negativo. Se puede observar que los pulsos no tienen un valor constante de ancho de pulso. En este trabajo se propone usar dos valores para el tiempo de muestreo. El primer valor será igual al ancho de pulso más pequeño generado por el comparador y la señal, mientras que el segundo valor de muestreo será igual al mayor ancho de pulso obtenido en el tren de pulsos.

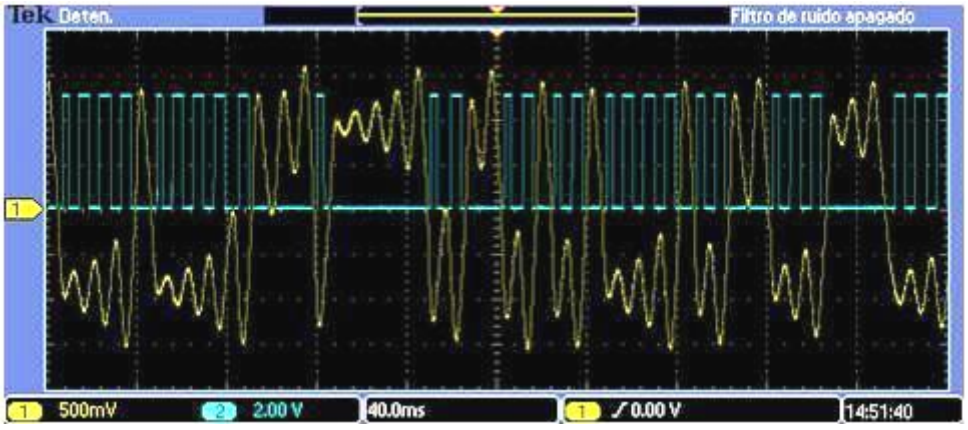

FIGURA 4.10: Tren de pulsos proveniente del comparador de umbral negativo σ^0.

Por último, respecto a la recolección de bits; se guardarán 4 tipos diferentes de datos por combinación de valores para cada oscilador caótico. Lo anterior es debido a que solo se ocupaba

la salida proveniente del primer bloque en la Figura 4.1, donde los valores de σ^1 y σ^0 en vez de pasar por la regla definida en (4.2) y por el filtro (4.3) simplemente pasaban por un sumador. Entonces, con el fin determinar que tan efectiva es la utilización de un filtro dentro de la arquitectura del generador de bits aleatorios y para medir ambos desempeños también se guardarán estos valores y se les aplicará los mismos test de aleatoriedad que a aquellos que siguen la regla (4.1) y el filtro (4.3).

Los cuatro tipos de datos que se guardaron por cada combinación fueron: datos ocupando el filtro y sin ocupar el filtro con un tiempo de muestreo menor así como datos ocupando el filtro y sin ocupar el filtro con tiempo de muestreo mayor. Se recolectaron 10 cadenas de 1000 bits filtrados para cada combinación de parámetros perteneciente a cada oscilador caótico. En lo que respecta a la recolección de bits que no pasan por el filtro no se tiene un número específico, definiéndose como la mínima cantidad necesaria para obtener los bits filtrados. Es decir, una vez que se juntaron los 1000 bits pasando con el filtro, la recolección de datos se detiene. No obstante, la recolección de bits sin filtrar están en un rango de 9000 – 11000 bits por cadena recolectada, variando según cada oscilador caótico.

Capítulo 5

Estudio sobre la Generación de Números Aleatorios

5.1. Optimización del Máximo Exponente de Lyapunov

Los resultados de la maximización del exponente de Lyapunov en cada oscilador caótico son mostrados en las tablas 5.1- 5.3. El uso del algoritmo de Evolución Diferencial se ocupó debido a la gran cantidad de procesamiento y operaciones necesarias para cumplir el primer objetivo de mejorar el exponente de Lyapunov. Un ejemplo es al maximizar el exponente de Lyapunov del oscilador caótico de función saturada, tomando en cuenta que los valores de a, b, c, d, van desde 0 a 1. Suponiendo que se ocupa una precisión de solo dos decimales, entonces al modificar solo un valor, a por ejemplo, y dejando sin variar b, c, d, se tendrían al final un número de 100 combinaciones diferentes. Ahora al combinar los cuatro parámetros serían alrededor de $100,000,000$ de combinaciones diferentes, lo cual llevaría un tiempo exagerado para procesar 10 soluciones para este oscilador caótico, más allá de eso hay que tener en cuenta que es solo el primero de tres osciladores, donde los otros dos osciladores caóticos no están limitados en sus parámetros. Desde este punto de vista se justifica la utilización del algoritmo evolutivo.

5.2. Determinación de la Entropía de los Atractores Caóticos

Se llevó a cabo la medición de la entropía a cada combinación mostrada en las tablas 5.1, 5.2 y 5.3 de los osciladores caóticos utilizados. Para medir la entropía se ocupó la señal de la variable de estado X perteneciente a cada combinación de cada oscilador. Para corroborar la medición de la entropía se comprobó sobre tres datos diferentes, datos del sistema simulado con Matlab,

Oscilador caótico de Chua			
Número de Combinación	**Alpha**	**Beta**	**Exponente de Lyapunov**
1	16.57	25.88	0.6868
2	17.96	31.66	0.6736
3	16	24.89	0.6658
4	17	27.12	0.6488
5	17.9	30.01	0.6281
6	16.65	26.48	0.6185
7	15	22.65	0.5992
8	14.94	22.73	0.5717
9	13.62	20.03	0.5662
10	13.53	21.76	0.5640

CUADRO 5.1: Resultado de la maximización del exponente de Lyapunov en el oscilador caótico de Chua.

Oscilador caótico de función saturada		
Número de Combinación	**Valores de a, b, c, d aplicando evolución diferencial**	**Exponente de Lyapunov**
1	$a = 1.000, b = 1.000, c = 0.499, d = 1.000$	0.3761
2	$a = 1.000, b = 0.788, c = 0.643, d = 0.666$	0.3713
3	$a = 0.866, b = 1.000, c = 0.393, d = 0.990$	0.3607
4	$a = 0.774, b = 0.658, c = 0.584, d = 0.493$	0.3460
5	$a = 1.000, b = 0.700, c = 0.700, d = 0.254$	0.3425
6	$a = 0.774, b = 0.671, c = 0.589, d = 0.846$	0.3391
7	$a = 0.924, b = 0.749, c = 0.668, d = 0.681$	0.3385
8	$a = 0.717, b = 0.659, c = 0.554, d = 0.224$	0.3376
9	$a = 0.706, b = 0.645, c = 0.552, d = 0.218$	0.3320
10	$a = 0.700, b = 0.700, c = 0.700, d = 0.700$	0.2658

CUADRO 5.2: Resultado de la maximización del exponente de Lyapunov en el oscilador caótico de función saturda.

datos del sistema en su síntesis con TopSpice y los datos experimentales obtenidos directamente del osciloscopio.

Debido al ruido implícito, tanto en la manipulación de circuitos electrónicos como en el uso de equipo de medición, para lograr una medición de la entropía lo más correcta posible es necesario tomar en cuenta un paso en específico, esto es tratar la señal antes de su procesamiento. Este paso se aplicó solamente a la señal obtenida gracias a la recolección de datos mediante el osciloscopio.

Parte de los datos obtenidos de la señal, sin ser tratada, son graficados en la figura 5.1. Aunque a simple vista no se observa alguna imperfección evidente en la señal, al aplicarle un acercamiento se observa que tiene ligeras partes donde parece un poco más ruidosa. A pesar de no parecer

Oscilador caótico de función diente de sierra			
Número de Combinación	**Alpha**	**Beta**	**Exponente de Lyapunov**
1	15	27.76	0.5618
2	14.5	27.04	0.5235
3	14	25.1	0.5458
4	13.5	24.08	0.5136
5	13	22.95	0.4994
6	12.44	21.33	0.4958
7	12	20.33	0.4805
8	11.5	19.09	0.4749
9	10.93	17.96	0.4431
10	10.32	16.7	0.4168

CUADRO 5.3: Resultado de la maximización del exponente de Lyapunov en el oscilador caótico de función diente de sierra.

mucho el ruido del sistema, al momento de ser graficado en su diagrama fase se hace mucho más evidente que en conjunto pierde el comportamiento real del sistema estudiado y por lo tanto es indispensable filtrar la señal para poder ser procesada correctamente. Al no filtrar la señal, este ruido provoca una medición errónea de la entropía que no concuerda en absoluto con resultados obtenidos por medio de la síntesis y de la simulación.

Para el tratamiento de la señal se aplicó a todos los datos obtenidos un filtro que proporciona MATLAB para el análisis de datos. El filtro que se ocupó es sgolayfilf. Esta función es para aplicar un filtro de suavizado de Savitzky-Golay, también conocidos como filtros de suavizado polinomial o mínimos cuadrados. Se suele utilizar para "suavizar" una señal ruidosa de frecuencia grande. Este tipo de filtros funcionan mucho mejor que el estándar promedio de filtros, que tienden a filtrar una parte significativa del contenido de alta frecuencia de la señal junto con el ruido.

Entonces al aplicar este filtro a cada señal de cada combinación, se obtuvo una mejor concordancia en la medición de la entropía respecto a la simulación numérica y síntesis con dispositivos electrónicos. Los resultados en la señal filtrada se muestran a continuación en la figura 5.2.

En las tablas 5.4, 5.5 y 5.6 se da el resultado de la medición de la entropía para cada oscilador caótico.

Se pueden observar dos puntos importantes como resultado en la medición de la entropía. El primero es que existe una buena correlación de valores para las diferentes mediciones de entropía pertenecientes a cada oscilador caótico y a cada combinación utilizada. El punto anterior tiene una ligera excepción perteneciente al oscilador caótico de función diente de sierra, donde los valores en cada medición son los más desiguales de todos. Esto se puede entender desde el

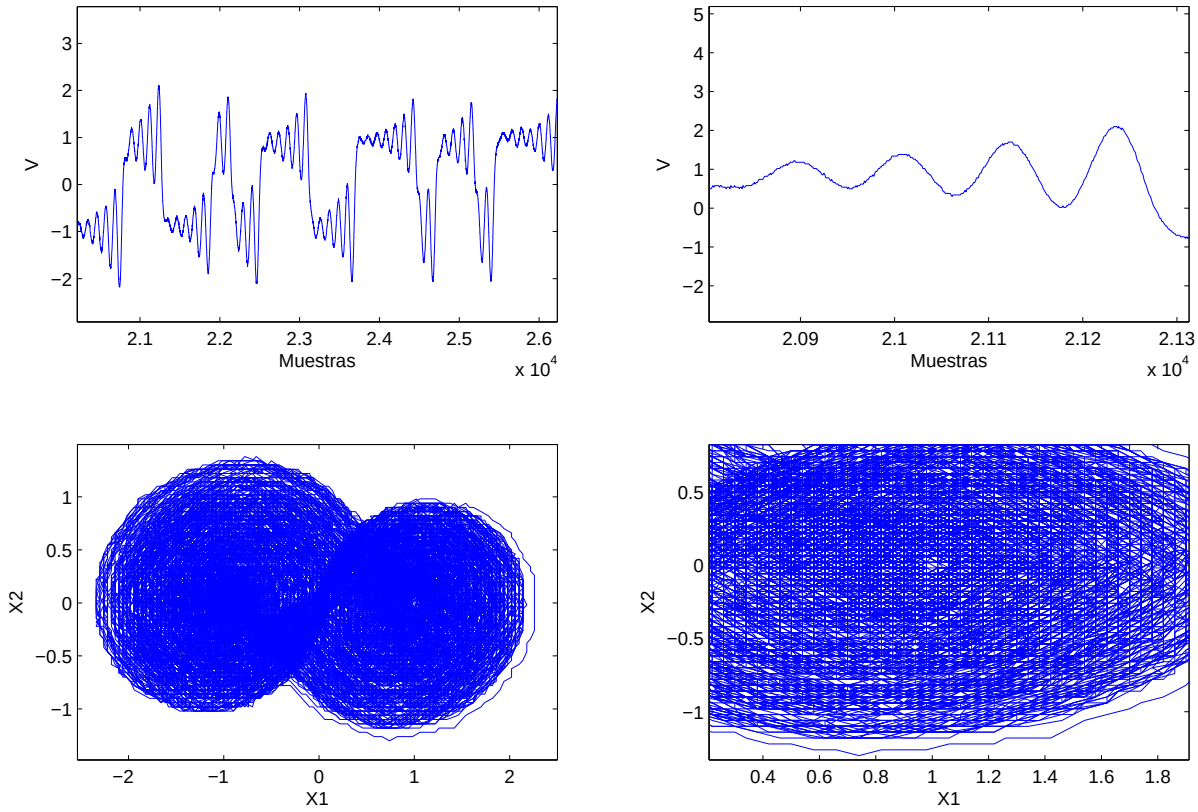

FIGURA 5.1: (Izquierda superior) Señal con respecto del tiempo, aparentemente sin inconvenientes graves. (Derecha Arriba) Acercamiento de la señal con respecto del tiempo. (Izquierda abajo) Gráfica de las señales en su plano fase. (Derecha inferior) Acercamiento a la gráfica del diagrama a fase.

punto de vista de los valores utilizados en este oscilador caótico, donde necesitaba establecer parámetros de $\xi = 0.25$ y $A_1 = 0.5$, lo cual provoca que un pequeño error en la medición provoque cambios mayores en la dinámica del sistema. El segundo punto a notar es que el valor de la entropía no es obligatoriamente proporcional al valor del exponente positivo de Lyapunov. Este resultado también se puede explicar retomando la teoría de la entropía donde se menciona que cada proceso tiende a desarrollarse en el sentido que aumente su entropía, sin embrago, a pesar de ser la opción más probable no es el único camino posible. No obstante se puede observar que generalmente las combinaciones cuyo exponente positivo de Lyapunov es el más alto también forman parte de aquellas con mayor valor en la medición de la entropía.

5.3. Test para Estimar la Aleatoriedad del TRNG

Los test de aleatoriedad que se llevaron a cabo para examinar cada cadena de bits, obtenida por combinacion de parámetros, según el oscilador caótico utilizado, son tomados de la NIST (National Institute of Standards and Technology) [7]. Donde los parámetros a utilizar para cada

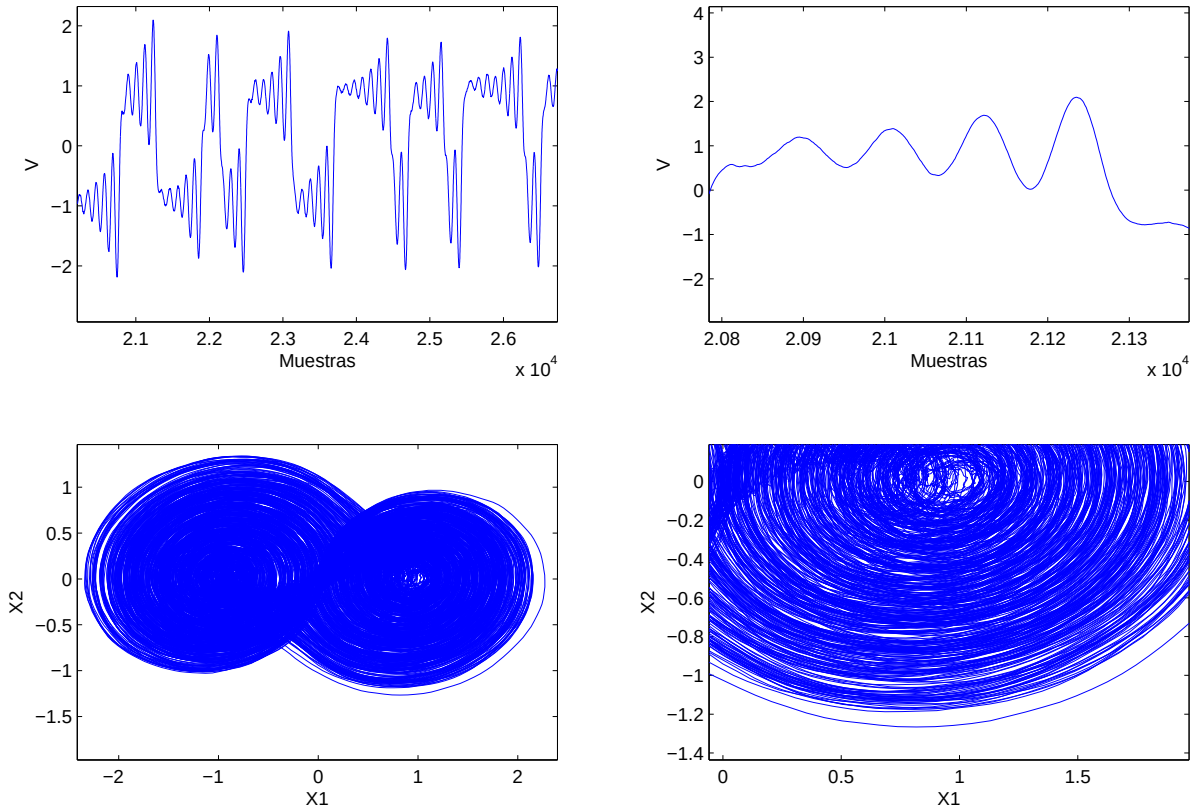

FIGURA 5.2: (Izquierda superior) Señal con respecto del tiempo. (Derecha Arriba) Acercamiento de la señal con respecto del tiempo. (Izquierda abajo) Gráfica de las señales en su plano fase. (Derecha inferior) Acercamiento a la gráfica del diagrama a fase.

test son aquellos que se recomiendan en el documento antes citado. A continuación se describen los test utilizados para este trabajo.

5.3.1. Test de Frecuencia (Monobit)

Este test se enfoca en la proporción de 0's y 1's dentro de la secuencia entera. El propósito de este test es determinar donde el número de 1's y 0's es aproximadamente el mismo, como debería de esperarse para una verdadera secuencia aleatoria. El test evalúa la cercanía de la fracción de $1's$ al valor de $1/2$, esto es, el número de uno y ceros en una secuencia debe ser casi el mismo. Es el test principal y los demás dependen de que se pase o no este test.

5.3.2. Test de Frecuencia Dentro de un Bloque

Este test se centra en la proporción de 1's dentro de bloques de M-bit. Su propósito es determinar donde la frecuencia de 1's dentro de un bloque de M-bits es aproximadamente $M/2$, como debería

Oscilador caótico de Chua						
Combinación	**Alpha**	**Beta**	**Entropía Experimental sin filtro**	**Entropía Experimental con filtro**	**Entropía TopSpice**	**Entropía Matlab**
1	16.57	25.88	0.1572	1.1406	1.0952	0.9908
2	17.96	31.66	0.1882	1.1697	1.1472	1.0108
3	16.00	24.89	0.3166	1.1450	1.1638	0.9704
4	17.00	27.12	0.1802	1.1379	1.1442	0.9738
5	17.90	30.01	0.1540	1.1360	1.1364	0.9667
6	16.65	26.48	0.1583	1.1320	1.1391	0.9604
7	15.00	22.65	0.3000	1.1035	1.1385	0.9550
8	14.94	22.73	0.2234	1.1032	1.1499	0.9426
9	13.62	20.03	0.1561	1.1608	1.1270	0.9273
10	13.53	21.76	0.1049	1.1393	1.0769	0.9011

Cuadro 5.4: Resultados en la medición de la entropía para las combinaciones obtenidas del oscilador caótico de Chua.

Oscilador caótico de función saturada					
N	**Valores de a,b,c,d aplicando evolución diferencial**	**Entropía Experimental sin filtro**	**Entropía Experimental con filtro**	**Entropía TopSpice**	**Entropía Matlab**
1	$a = 1.000, b = 1.000, c = 0.499, d = 1.000$	0.3409	1.372	1.456	1.3976
2	$a = 1.000, b = 0.788, c = 0.643, d = 0.666$	-0.4485	0.9819	1.0383	0.986
3	$a = 0.866, b = 1.000, c = 0.393, d = 0.990$	0.5715	1.4341	1.4693	1.5067
4	$a = 0.774, b = 0.658, c = 0.584, d = 0.493$	-0.0969	0.9459	0.9712	1.0035
5	$a = 1.000, b = 0.700, c = 0.700, d = 0.254$	-1.6962	0.044	0.0686	0.0867
6	$a = 0.774, b = 0.671, c = 0.589, d = 0.846$	0.4428	1.499	1.500	1.4736
7	$a = 0.924, b = 0.749, c = 0.668, d = 0.681$	-0.3416	1.0894	1.0891	1.0755
8	$a = 0.717, b = 0.659, c = 0.554, d = 0.224$	-1.3544	0.2157	0.2269	0.2199
9	$a = 0.706, b = 0.645, c = 0.552, d = 0.218$	-1.4076	0.2008	0.2322	0.2182
10	$a = 0.700, b = 0.700, c = 0.700, d = 0.700$	-0.0294	1.2723	1.3506	1.3067

Cuadro 5.5: Resultados en la medición de la entropía para las combinaciones obtenidas del oscilador caótico de función saturada.

de esperarse bajo el supuesto de aleatoriedad. Para un bloque de tamaño $M = 1$, este test se convertiría en el primer test de Monobit.

Oscilador caótico de diente de sierra						
Combinación	**Alpha**	**Beta**	**Entropía Experimental sin filtro**	**Entropía Experimental con filtro**	**Entropía TopSpice**	**Entropía Matlab**
1	15.00	27.76	−0.1486	0.6520	0.7251	0.8243
2	14.50	27.04	−0.0417	0.5821	0.7244	0.8234
3	14.00	25.10	−0.1686	0.5820	0.7180	0.8223
4	13.50	24.08	−0.1253	0.5830	0.6983	0.8251
5	13.00	22.95	−0.1273	0.5966	0.6942	0.8195
6	12.44	21.33	−0.1510	0.5907	0.6754	0.8057
7	12.00	20.33	−0.1732	0.6075	0.6699	0.7671
8	11.50	19.09	−0.1558	0.6137	0.6735	0.8250
9	10.93	17.96	−0.1496	0.6214	0.6921	0.8063
10	10.32	16.70	−0.1858	0.5516	0.6717	0.8026

Cuadro 5.6: Resultados en la medición de la entropía para las combinaciones obtenidas del oscilador caótico de diente de sierra.

5.3.3. Test de Ejecución

El test se orienta en el número total de corridas en la secuencia, donde una corrida es una secuencia ininterrumpida de bits idénticos. Una corrida de extensión k consiste en exactamente k bits idénticos limitado antes y después con un bit de valor positivo. El propósito de este test es determinar si el número de corridas de 1's y 0's de diferentes medidas es como se espera para una secuencia aleatoria. En particular, este test determina si la oscilación entre 0's y 1's es demasiado rápida o demasiado lenta.

5.3.4. Test de la Mayor Ejecución Dentro de un Bloque

Como su nombre lo dice, este test de aleatoriedad verifica la mayor corrida de 1's dentro de bloques de m-bits. Su propósito es determinar si la longitud de la mayor corrida de 1's es como se esperaría para una secuencia aleatoria. Tomando en cuenta que una irregularidad en la medida esperada de la mayor corrida de 1's implica que existe también una irregularidad en la medida esperada para la mayor corrida de 0's. Por lo tanto, solo un test para 1's es necesario.

5.3.5. Test de la Transformada Discreta de Fourier

Este test enfoca en el mayor pico en la Transformada Discreta de Fourier de una secuencia. Es propuesto para detectar características periódicas en la secuencia (por ejemplo, patrones repetitivos que están cercanos unos de otros) que indican una desviación del supuesto de aleatoriedad.

La intención es detectar si el número de picos excediendo el 95 % de umbral es significativamente diferente a 5 %.

5.3.6. Test Serial

El test se enfoca en la frecuencia de todas las posibles superposiciones de patrones de m-bit a través de la secuencia entera. El propósito es determinar si el número de ocurrencias de los 2^m $m-bit$ patrones de superposición es aproximadamente la misma, como debería de serlo para una secuencia aleatoria. Las secuencias aleatorias tienen uniformidad, esto es, cada patrón de m-bits tiene la misma posibilidad de aparecer como lo tiene otro patrón de m-bits.

5.3.7. Test de Entropía Aproximada

De la misma manera que el test anterior, el enfoque de este test es la frecuencia de todas las posibles superposiciones de patrones de m-bits a través de la secuencia entera. El propósito en este test es comparar la frecuencia de bloques superpuestos en dos consecutivas/adyacentes medidas (m y $m+1$) contra el resultado esperado para una secuencia aleatoria.

5.3.8. Test de Sumas Acumuladas

El test se enfoca en la máxima excursión de un camino aleatorio definido por la suma acumulativa de dígitos ajustados en la secuencia $(+1, -1)$. El propósito del test es determinar si la suma acumulativa de secuencias parciales ocurridas en el testeo es demasiado larga o demasiado pequeña relacionado al valor de comportamiento esperado en aquellas sumas para secuencias aleatorias. Estas sumas acumulativas deben ser consideradas como un camino aleatorio. Para una secuencia aleatoria, las excursiones del camino aleatorio deben estar cerca de cero. Para ciertos tipos de secuencias no-aleatorias, las excursiones de cero en este camino aleatorio son mayores.

5.4. Resultados de los Test de Aleatoriedad

Los resultados obtenidos al realizar los test de aleatoriedad antes mencionados están representados en las tablas 5.7, 5.8 y 5.9. Cada tabla representa el resultado de un oscilador caótico, así por ejemplo, la tabla 5.7 es el resultado de los test aplicados para los bits aleatorios provenientes del oscilador caótico de Chua. También debe recordarse que por cada oscilador caótico se obtuvieron 10 combinaciones de parámetros, y por cada combinación de parámetros se recolectaron

4 tipos diferentes de cadenas de bits (una con tiempo de muestreo menor y con filtro, otra con tiempo de muestreo menor y sin filtro, otra con tiempo de muestreo mayor y con filtro y por ultimo con muestreo mayor y sin filtro). Por lo tanto, por cada oscilador caótico tenemos 40 diferentes resultados mostrados en las tablas antes mencionadas.

Por el momento se puede diferenciar en las tablas los valores que pasaron el test como aquellos dentro de celdas azules, para los que están con un relleno más claro son combinaciones que se quedaron a pocas cadenas de pasar el test y, por último, aquellos valores donde están en blanco significan que no pasaron el test. Posteriormente se explicara cómo medir si una cadena de bits paso o no el test según su relación.

Tabla 5.7 Resultados de los Test Aleatorios Aplicados al Oscilador Caótico de Chua

Número de Combinación	Nombre del Test Aplicado									
	Monobit	Frequency	Cumsum	Runs	Longest Runs	FFT	Entropy	Serial	Rank	Linear
	100	100	100	100	128	100	100	100	38912	100000
1RB_min	99/100	100/100	100/100 100/100	87/100	78/78	99/100	97/100	95/100 98/100		
1RB_max	100/100	100/100	100/100 100/100	93/100	78/78	100/100	99/100	99/100 99/100		
1suma_min	958/1000	999/1000	990/1000 991/1000	473/1000	38/500	929/1000	12/1000	103/1000 851/1000	3/3	1/1
1suma_max	944/1000	1000/1000	980/1000 980/1000	1/1000	1/500	876/1000	0/1000	8/1000 256/1000	3/3	1/1
2RB_min	100/100	100/100	100/100 100/100	21/100	75/78	99/100	48/100	65/100 91/100		
2RB_max	100/100	100/100	100/100 100/100	8/100	72/78	96/100	26/100	49/100 96/100		
2suma_min	806/1000	1000/1000	936/1000 940/1000	36/1000	1/500	885/1000	5/1000	69/1000 769/1000	3/3	1/1
2suma_max	889/1000	999/1000	962/1000 963/1000	0/1000	0/500	825/1000	0/1000	1/1000 95/1000	3/3	1/1
3RB_min	98/100	97/100	99/100 98/100	96/100	76/78	100/100	94/100	92/100 99/100		
3RB_max	100/100	100/100	100/100 100/100	85/100	78/78	99/100	94/100	97/100 100/100		
3suma_min	942/1000	1000/1000	987/1000 988/1000	682/1000	5/500	894/1000	5/1000	39/1000 808/1000	3/3	1/1
3suma_max	892/1000	993/1000	945/1000 946/1000	7/1000	5/500	868/1000	0/1000	5/1000 316/1000	3/3	1/1
4RB_min	100/100	100/100	100/100 100/100	83/100	75/78	98/100	91/100	97/100 100/100		
4RB_max	100/100	100/100	100/100 100/100	88/100	78/78	100/100	95/100	97/100 98/100		
4suma_min	991/1000	999/100	998/1000 997/1000	360/1000	31/500	936/1000	12/1000	97/1000 843/1000	3/3	1/1
4suma_max	983/1000	1000/1000	995/1000 994/1000	0/1000	0/500	876/000	0/1000	1/1000 209/1000	3/3	1/1
5RB_min	100/100	100/100	100/100 100/100	77/100	78/78	100/100	89/100	92/100 98/100		
5RB_max	100/100	100/100	100/100 100/100	39/100	77/78	100/100	70/100	83/100 97/100		
5suma_min	850/1000	996/1000	927/1000 922/1000	121/1000	31/500	925/1000	24/100	57/1000 651/1000	3/3	1/1
5suma_max	976/1000	1000/1000	995/1000 998/1000	0/1000	0/500	869/1000	0/1000	0/1000 79/1000	3/3	1/1
6RB_min	100/100	100/100	100/100 100/100	58/100	76/78	99/100	80/100	91/100 98/100		
6RB_max	99/100	100/100	99/100 100/100	65/100	77/78	98/100	87/100	93/100 99/100		
6suma_min	950/1000	1000/1000	982/1000 986/986	201/1000	37/500	928/1000	18/1000	94/1000 716/1000	3/3	1/1
6suma_max	963/1000	1000/1000	994/1000 993/1000	0/1000	0/500	870/1000	0/1000	9/1000 157/1000	3/3	1/1
7RB_min	99/100	100/100	100/100 100/100	93/100	78/78	99/100	99/100	97/100 100/100		
7RB_max	100/100	100/100	100/100 100/100	92/100	78/78	100/100	95/100	98/100 100/100		
7suma_min	892/1000	999/1000	960/1000 964/1000	231/1000	31/500	919/1000	18/1000	83/1000 730/1000	3/3	1/1
7suma_max	870/1000	997/1000	936/1000 938/1000	7/1000	1/500	890/1000	0/1000	6/1000 224/1000	3/3	1/1
8RB_min	100/100	100/100	100/100 100/100	81/100	77/78	99/100	95/100	96/100 99/100		
8RB_max	100/100	100/100	100/100 100/100	73/100	78/78	100/100	89/100	96/100 99/100		
8suma_min	944/1000	1000/1000	978/1000 980/1000	65/1000	52/500	890/1000	16/1000	24/1000 447/1000	3/3	1/1
8suma_max	958/1000	994/1000	975/1000 974/1000	0/1000	2/500	895/1000	0/1000	1/1000 91/1000	3/3	1/1
9RB_min	99/100	100/100	100/100 100/100	96/100	77/78	100/100	96/100	98/100 100/100		
9RB_max	100/100	100/100	100/100 100/100	84/100	78/78	100/100	95/100	98/100 100/100		
9suma_min	938/1000	998/999	967/1000 967/1000	109/1000	81/500	906/1000	27/1000	30/1000 478/1000	3/3	1/1
9suma_max	926/1000	1000/1000	972/1000 971/1000	3/1000	1/500	902/1000	0/1000	2/1000 130/1000	3/3	1/1
10RB_min	99/100	100/100	99/100 99/100	93/100	77/78	100/100	95/100	98/100 100/100		
10RB_max	99/100	100/100	100/100 100/100	86/100	78/78	100/100	92/100	98/100 98/100		
10suma_min	900/1000	998/1000	961/1000 960/1000	33/1000	84/500	899/1000	46/1000	96/1000 807/1000	3/3	1/1
10suma_max	990/1000	1000/1000	999/1000 999/1000	8/1000	5/500	882/1000	9/1000	12/1000 173/1000	3/3	1/1

Tabla 5.8 Resultados de los Test Aleatorios Aplicados al Oscilador Caótico de Función Saturada

Número de Combinación	Nombre del Test Aplicado									
	Monobit	Frequency	Cumsum	Runs	Longest Runs	FFT	Entropy	Serial	Rank	Linear
	100	100	100	100	128	100	100	100	38912	100000
1RB_min	99/100	100/100	100/100 100/100	5/100	64/78	94/100	28/100	44/100 87/100		
1RB_max	100/100	100/100	100/100 100/100	8/100	68/78	97/100	44/100	61/100 92/100		
1suma_min	869/900	900/900	899/900 898/900	0/900	19/500	737/1000	0/1000	0/1000 130/1000	2/2	
1suma_max	720/800	800/800	785/800 778/800	0/800	0/500	663/800	0/1000	13/800 444/800	2/2	
2RB_min	100/100	100/100	100/100 100/100	76/100	76/78	99/100	92/100	95/100 97/100		
2RB_max	100/100	100/100	100/100 100/100	66/100	78/78	100/100	89/100	95/100 96/100		
2suma_min	1000/1000	1000/1000	1000/1000 1000/1000	16/1000	418/500	930/1000	48/1000	238/1000 812/1000	2/2	1/1
2suma_max	799/800	800/800	799/800 799/800	36/800	361/500	775/800	123/800	258/800 654/800	2/2	
3RB_min	100/100	100/100	100/100 100/100	82/100	77/78	100/100	91/100	96/100 99/100		
3RB_max	99/100	100/100	99/100 99/100	100/100	78/78	100/100	97/100	100/100 100/100		
3suma_min	505/100	1000/1000	750/1000 742/1000	731/1000	204/500	960/1000	14/1000	76/1000 529/1000	2/2	1/1
3suma_max	446/900	900/900	650/900 656/900	46/900	29/500	852/900	16/900	30/900 243/900	2/2	
4RB_min	100/100	100/100	100/100 100/100	88/100	77/78	99/100	98/100	97/100 100/100		
4RB_max	100/100	100/100	100/100 100/100	62/100	77/78	100/100	87/100	92/100 98/100		
4suma_min	1000/1000	1000/1000	1000/1000 1000/1000	11/1000	319/500	931/1000	21/1000	202/1000 716/1000	2/2	1/1
4suma_max	900/900	900/900	900/900 900/900	0/900	10/500	769/900	0/900	1/900 457/900	2/2	
5RB_min	99/100	100/100	99/100 99/100	62/100	76/78	100/100	83/100	89/100 100/100		
5RB_max	100/100	100/100	100/100 100/100	64/100	78/78	100/100	89/100	92/100 99/100		
5suma_min	900/900	900/900	900/900 900/900	0/900	298/500	835/900	9/900	83/900 589/900	2/2	
5suma_max	800/800	800/800	800/800 800/800	2/800	204/500	737/800	22/800	125/800 959/800	2/2	
6RB_min	90/90	90/90	90/90 90/90	49/90	76/78	89/90	77/90	88/90 96/100		
6RB_max	100/100	100/100	100/100 100/100	55/100	74/78	99/100	86/100	87/100 97/100		
6suma_min	1000/1000	1000/1000	1000/1000 1000/1000	0/1000	241/500	925/1000	12/1000	143/1000 734/1000	2/2	1/1
6suma_max	800/800	800/800	800/800 800/800	0/800	30/500	691/800	0/800	12/800 404/800	2/2	
7RB_min	100/100	100/100	100/100 100/100	44/100	75/78	98/100	79/100	83/100 96/100		
7RB_max	100/100	100/100	100/100 100/100	38/100	75/78	98/100	82/100	87/100 95/100		
7suma_min	1000/1000	1000/1000	1000/1000 1000/1000	1/1000	303/500	919/1000	26/1000	149/1000 618/1000	2/2	1/1
7suma_max	800/800	800/800	800/800 800/800	9/800	361/500	754/800	95/800	295/800 726/800	2/2	
8RB_min	100/100	100/1000	100/100 100/100	63/100	77/78	100/100	90/100	92/100 98/100		
8RB_max	100/100	100/100	100/100 100/100	75/100	77/78	100/100	86/100	93/100 97/100		
8suma_min	1000/1000	1000/1000	1000/1000 1000/1000	113/1000	447/500	955/1000	170/1000	300/1000 871/1000	2/2	1/1
8suma_max	800/800	800/800	800/800 800/800	0/800	359/500	736/800	18/800	128/800 555/800	2/2	
9RB_min	99/100	100/100	99/100 100/100	26/100	73/78	99/100	80/100	87/100 96/100		
9RB_max	100/100	100/100	100/100 100/100	89/100	78/78	99/100	96/100	99/100 98/100		
9suma_min	920/920	920/920	920/920 920/920	395/920	383/500	881/920	175/920	447/920 856/920	2/2	1/1
9suma_max	836/836	836/836	836/836 836/836	0/836	316/500	777/836	7/836	48/836 406/836	2/2	
10RB_min	100/100	100/100	100/100 100/100	2/100	66/78	92/100	11/100	21/100 79/100		
10RB_max	100/100	100/100	100/100 100/100	100/100	77/78	95/100	58/100	65/100 94/100		

Tabla 5.9 Resultados de los Test Aleatorios Aplicados al Oscilador Caótico de Diente de Sierra

Número de Combinación	Nombre del Test Aplicado									
	Monobit	Frequency	Cumsum	Runs	Longest Runs	FFT	Entropy	Serial	Rank	Linear
	100	100	100	100	128	100	100	100	38912	100000
1RB_min	99/100	100/100	100/100 100/100	85/100	78/78	98/100	96/100	96/100 97/100		
1RB_max	99/100	100/100	100/100 100/100	46/60	78/78	100/100	95/100	98/100 99/100		
1suma_min	800/800	800/800	800/800 800/800	11/800	477/500	777/800	106/800	189/800 612/800	2/2	
1suma_max	450/450	450/450	450/450 450/450	115/450	496/500	437/450	260/437	264/450 433/450	2/2	
2RB_min	100/100	100/100	100/100 100/100	48/100	76/78	97/100	71/100	84/100 98/100		
2RB_max	100/100	100/100	100/100 100/100	26/100	75/78	100/100	60/100	85/100 96/100		
2suma_min	800/800	800/800	800/800 800/800	4/800	485/500	762/800	44/800	157/800 604/800	2/2	
2suma_max	700/700	700/700	700/700 700/700	372/700	486/500	688/700	493/700	584/700 694/700	2/2	
3RB_min	100/100	100/100	100/100 100/100	65/100	78/78	100/100	87/100	91/100 98/100		
3RB_max	100/100	100/100	100/100 100/100	40/100	76/78	98/100	65/100	78/100 100/100		
3suma_min	800/800	800/800	800/800 800/800	4/800	477/500	786/800	79/800	130/800 556/800	2/2	
3suma_max	699/700	700/700	699/700 700/700	346/700	484/500	683/700	424/700	539/700 693/700	2/2	
4RB_min	100/100	100/100	100/100 100/100	24/100	75/78	98/100	54/100	74/100 99/100		
4RB_max	100/100	100/100	100/100 100/100	22/100	69/78	96/100	58/100	79/100 96/100		
4suma_min	800/800	800/800	800/800 800/800	7/800	468/500	764/800	23/800	100/800 561/800	2/2	
4suma_max	700/700	700/700	700/700 700/700	361/700	482/500	679/700	475/700	545/700 681/700	2/2	
5RB_min	98/100	100/100	98/100 99/100	95/100	76/78	100/100	97/100	97/100 98/100		
5RB_max	100/100	100/100	100/100 100/100	13/100	74/78	95/100	30/100	54/100 93/100		
5suma_min	986/1000	985/1000	986/1000 985/1000	42/1000	467/500	970/1000	181/1000	376/1000 939/1000	2/2	1/1
5suma_max	800/800	800/800	800/800 800/800	25/800	484/500	773/800	182/1000	351/1000 766/1000	2/2	
6RB_min	99/100	100/100	100/100 100/100	83/100	77/78	100/100	93/100	95/100 98/100		
6RB_max	100/100	100/100	100/100 100/100	23/100	76/78	100/100	63/100	68/100 96/100		
6suma_min	999/1000	1000/1000	1000/1000 1000/1000	23/1000	475/500	968/1000	139/1000	241/900 866/1000	2/2	1/1
6suma_max	799/800	800/800	800/800 800/800	21/800	482/500	774/800	110/800	185/800 693/800	2/2	
7RB_min	100/100	100/100	100/100 100/100	81/100	78/78	99/100	96/100	97/100 99/100		
7RB_max	100/100	100/100	100/100 100/100	16/100	73/78	98/100	39/100	59/100 90/100		
7suma_min	999/1000	1000/1000	1000/1000 1000/1000	8/1000	485/500	967/1000	87/1000	359/1000 915/1000	2/2	1/1
7suma_max	799/800	800/800	800/800 800/800	195/800	484/500	773/800	350/800	513/800 782/800	2/2	
8RB_min	100/100	100/100	100/100 100/100	93/100	77/78	100/100	97/100	97/100 98/100		
8RB_max	100/100	100/100	100/100 100/100	31/100	76/78	99/100	57/100	67/100 97/100		
8suma_min	999/1000	1000/1000	1000/1000 1000/1000	4/1000	471/500	969/1000	83/1000	109/1000 739/1000		
8suma_max	800/800	800/800	800/800 800/800	22/800	492/500	770/800	127/800	340/800 757/800		
9RB_min	100/100	100/100	100/100 100/100	21/100	72/78	95/100	37/100	48/100 85/100		
9RB_max	99/100	100/100	99/100 99/100	34/100	74/78	100/100	52/100	69/100 95/100		
9suma_min	923/1000	994/1000	955/1000 958/1000	9/1000	475/500	929/1000	5/1000	34/1000 370/1000	2/2	1/1
9suma_max	776/800	800/800	790/800 791/800	81/800	481/500	756/800	159/800	381/800 775/800	2/2	
10RB_min	100/100	100/100	100/100 100/100	71/100	77/78	100/100	83/100	91/100 97/100		
10RB_max	100/100	100/100	100/100 100/100	55/100	77/78	99/100	76/100	82/100 98/100		
10suma_min	998/1000	1000/1000	1000/1000 999/1000	2/1000	422/500	953/1000	31/1000	134/1000 831/1000	2/2	1/1
10suma_max	799/800	800/800	800/800 800/800	0/800	489/500	758/800	11/800	157/800 659/800	2/2	

La información contenida en las tablas 5.7, 5.8 y 5.9 se explica a continuación:

Primero, debajo del nombre de cada tabla se encuentra el nombre de cada test que se realizó e inmediatamente debajo del nombre del test se encuentra un número encerrado en su respectiva celda. Este número indica la cantidad de bits que fueron puestos a prueba en cada test correspondientemente.

Ahora, en la primera columna, de izquierda a derecha, está el nombre de la combinación a la que pertenecen los resultados de cada renglón. Se encuentran distinguidos primeramente con números, del 1 al 10, que son las combinaciones mostradas en las tablas 5.1 5.3, según sea el caso del oscilador caótico. Después están divididas en RB y Suma, donde RB representa los resultados referentes a aquellos que ocupan el filtrado para la recolección de la cadena de bits, mientras que aquellos que contienen Suma son el resultado de la recolección de bits proveniente de la suma de los comparadores de nivel de umbral, mostrados en la figura 4.5. Por último el término min y max, hace referencia al tiempo de muestreo donde min es el tiempo de muestreo menor y max es el tiempo de muestreo mayor.

En lo que respecta al resultado concreto de los test de aleatoriedad, se puede observar que están mostrados como números separados por diagonales; donde el primer número es la cantidad de cadenas de bits aleatorios que pasaron el test y el número a la derecha de la diagonal es la cantidad de cadenas que fueron testeadas en cada prueba. Un ejemplo a lo anterior, tomando los cuatro primeros renglones de la tabla 5.7 perteneciente a los resultados del oscilador caótico de Chua se muestran a continuación:

Resultados de los Test Aleatorios Aplicados al Oscilador Caótico de Chua

	Nombre del Test Aplicado									
	Monobit	Frequency	Cumsum	Runs	Longest Runs	FFT	Entropy	Serial	Rank	Linear
	100	100	100	100	128	100	100	100	38912	100000
1RB_min	99/100	100/100	100/100 100/100	87/100	78/78	99/100	97/100	95/100 98/100		
1RB_max	100/100	100/100	100/100 100/100	93/100	78/78	100/100	99/100	99/100 99/100		
1suma_min	958/1000	999/1000	990/1000 991/1000	473/1000	38/500	929/1000	12/1000	103/1000 851/1000	3/3	1/1
1suma_max	944/1000	1000/1000	980/1000 980/1000	1/1000	1/500	876/1000	0/1000	8/1000 256/1000	3/3	1/1

CUADRO 5.7: Ejemplo de la tabla de resultados para los test de aleatoriedad ejecutados con los bits aleatorios del oscilador caótico de Chua.

En la fracción de la tabla anterior, se puede ver primeramente que son los 4 resultados pertenecientes a la primera combinación del oscilador caótico de Chua. La primera combinación para este oscilador está en la tabla 5.1, (con valores de $\alpha = 16.57$ y $\beta = 25.88$). En el primer renglón están los resultados pertenecientes a los bits recolectados utilizando el filtro y con un tiempo de muestreo menor, el segundo renglón son los resultados referentes a los bits recolectados usando también el filtro pero con un tiempo de muestreo mayor. Los renglones tercero y cuarto muestran los resultados de los bits obtenidos sin ocupar ningún filtro, donde el tercer renglón indica aquellos con un tiempo de muestreo menor y el cuarto renglón los que provienen de un tiempo

de muestreo mayor. Si deseamos saber cómo fue el desempeño en el test del monobit para la primera combinación perteneciente al oscilador caótico de Chua podríamos observar que para los casos en que se ocupó el filtro y con un tiempo de muestreo mayor y menor se aprobó satisfactoriamente un total de 100 y 99 veces, respectivamente, de un total de 100 ocasiones en que fue aplicado el test. Mientras que en los casos donde no se utilizó el filtro con un tiempo de muestreo menor y mayor el resultado fue de 958 y 944 veces que se aprobó el test, respectivamente, de un total de 1000.

Por otro lado, se puede ver que algunos espacios pertenecientes a los resultados de algunas de las pruebas realizadas, principalmente los test Rank y Serial, se han dejado en blanco. Cuando existan espacios en blanco es simplemente indicador de que la prueba no se realizó. La principal razón para omitir la prueba es que no se contaba con la mínima cantidad de bits necesarios para poder ejecutar el test de manera correcta.

5.5. Discusión de Resultados

En esta sección primero se expondrá como poder considerar que una prueba resultó aprobatoria o no sin importar que tipo de oscilador sea, después se hará una comparación de los resultados obtenidos según el tipo de oscilador caótico y de ahí se dará paso a casos más particulares. En específico, se tomarán en cuenta divisiones según el método que se utilizó al obtener cada cadena de bits, más concretamente, que tan efectivo es el desempeño con o sin la utilización de un filtro y las diferencias en los resultados según sea el tiempo de muestreo para la recolección de los bits.

Primeramente, lo que respecta al criterio para determinar que una cadena de bits es aleatoria según los resultados obtenidos provenientes de la test suite, basados en el documento [39], establece que hay dos parámetros para poder concluir que es satisfactorio o no, el primero es el nivel de importancia $\propto$ y el segundo es el intervalo de seguridad. Para un nivel de importancia establecido un cierto porcentaje de $P - values$ se prevé para indicar una falla. Por ejemplo, si el nivel de importancia escogido es de 0.01 ($i.e., \propto = 0.01$), entonces alrededor de 1 % de las secuencias se pronostica que pueden fallar. Una secuencia pasa el test estadístico de aleatoriedad siempre y cuando el valor del $P - value \geq \propto$ y lo falla de otra forma.

Un ejemplo para determinar si una secuencia es aleatoria o no es el siguiente. Si se pusieran a prueba 1000 secuencias ($i.e., m = 1000$), $\propto = 0.01$ (el valor de importancia), y 996 secuencias binarias obtuvieron un $P - value \geq 0.01$, entonces la proporción es $996/1000 = 0.9960$. El rango de proporción aceptable es determinado según el intervalo de seguridad definido como $\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$, donde $\hat{p} = 1 - \propto$, y m es el tamaño de la muestra. Si la proporción cae fuera de

este intervalo entonces existe evidencia para indicar que la secuencia es no aleatoria. Regresando al ejemplo, el intervalo de confianza es $0.99 \pm 3\sqrt{((0.99(0.01))/1000)} = .99 \pm 0.0094392$ entonces la proporción debe caer arriba de 0.9805607 lo cual es igual a decir que 980 secuencias de bits deberían pasar el test estadístico para poder decir que la secuencia es aleatoria.

Con base en lo anterior, utilizando los valores recomendados de la test y poniendo a prueba 100 secuencias se tendría lo siguiente:

$m = 100$; Cantidad de secuencias.
$\propto = 0.01$; Nivel de importancia.
$\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$; Rango de proporción aceptable.
Si tenemos: $\hat{p} = 1 - \propto = 0.99$, entonces nuestra proporción aceptable sería:

$$0.99 \pm 3\sqrt{((0.99(0.01))/100)} = .99 \pm 0.0298494 \Rightarrow 0.960151$$

La proporción para los resultados se obtiene como:

$$\frac{\text{Número de secuencias con} \quad P - value \geq 0.01}{\text{cantidad de secuencias puestas a prueba}}$$

Entonces es necesario que la proporción de los resultados sea mayor a 0.960151, expresado de otra forma:

$$\frac{\text{Número de secuencias con} \quad P - value \geq 0.01}{\text{cantidad de secuencias puestas a prueba}} \geq 0.960151$$

Por lo tanto, la mínima proporción para decir que se ha pasado el test es 96/100.

5.5.1. Oscilador Caótico de Chua

Los resultados obtenidos por este oscilador en su mayoría son aprobatorios, pasando la mayoría de los test a los que fue sometido. Aquellos test donde el generador de bits tuvo un menor rendimiento fue en el de Runs y el de Entropy pasando solo 2 y 4 veces, respectivamente, en total cada test. Aunque muchas veces se quedó en valores muy cercanos de poder aprobar el test (faltando $1-3$ cadenas correctas) muestra una debilidad para estas pruebas. También falló en el test Serial en solo 3 ocasiones.

Lo que respecta al tipo de cadena de bits que se puso a prueba, el rendimiento por parte de aquellos bits donde se ocupó el filtro fue completamente mejor al de aquellos sin el filtro. Se puede ver como algunas veces, test básicos como lo es el caso del monobit, resultan fallar para las muestras de bits que no pasaron por el filtro. Mientras que aquellos bits que fueron filtrados

pasan en su totalidad las pruebas, exceptuando las dos antes indicadas y mencionando que las veces en que se aprobaron estos test fueron todos por cadenas de bits filtradas.

Por el tiempo de muestreo se puede decir que este generador tuvo un rendimiento mejor con aquellas cadenas de bits provenientes del tiempo de muestreo menor. Ya que solo 6 veces de 20, dentro de los test que no fueron aprobados, los resultados del tiempo de muestreo mayor lograron mejores resultados. No se toman en cuenta comparaciones con resultados de test que fueron aprobatorios ya que a fin de cuentas el resultado se cumplió, ser capaces de pasar el test.

5.5.2. Oscilador Caótico de Función Saturada

El resultado de este oscilador caótico fue similar al anterior, en general pasa la mayoría de las pruebas, sin embargo también muestra poco rendimiento en los test Runs y Entropy, pasando solo en 2 y 3 ocasiones cada uno de los test, respectivamente, de la misma manera tiene resultados muy cerca de lo necesario para aprobar el test. En cuanto a la prueba de transformada de Fourier los bits obtenidos no pasaron la prueba en 3 ocasiones, mientras que el test Serial fue reprobado en 4 ocasiones señalando que tiene también una cantidad considerable de resultados en los que casi no pasa la prueba.

El tipo de cadena de bits que obtuvo mejores resultados fue nuevamente aquellas cadenas provenientes de la configuración con filtro, todos los test que se pasaron en las pruebas que presenta menor rendimiento fueron con cadenas de bits filtradas. Por otro lado, las cadenas de bits provenientes de la configuración sin filtro pasaron también muchas de las pruebas de la test suite. De hecho teniendo un muy buen rendimiento incluso en pruebas con mayor nivel de dificultad como lo es el test de Fourier. Sin embargo, pruebas con un nivel alto de exigencia fueron completamente fracasadas, por ejemplo Runs.

Para los resultados según el tiempo de muestreo se observa un mejor desempeño para aquellos bits recolectados con un tiempo de muestreo mayor, ya que 14 de 20 combinaciones mostraron mejores resultados a aquellos provenientes de un tiempo de muestreo mayor.

5.5.3. Oscilador Caótico de Diente de Sierra

El desempeño para el oscilador caótico de diente de sierra una vez más fue muy similar a los anteriores teniendo un bajo rendimiento en los test de Runs y Entropy, donde al primero falló en todas las ocasiones mientras que el segundo solo lo aprobó satisfactoriamente en 4 combinaciones. Lo que respecta a la prueba de Fourier solo falló en 2 ocasiones. Por último la prueba serial se reprobó en 6 ocasiones. Todos los demás test fueron aprobatorios.

En lo que respecta a los resultados según la forma en la recolección de los bits, con o sin el uso de un filtro, una vez más aquellos provenientes del filtrado mostraron un resultado mejor, donde las pruebas de mayor grado de dificultad fueron aprobadas solo por cadenas de bits filtrados. Las cadenas de bits que no fueron filtrados pasan las pruebas básicas de aleatoriedad sin ningún problema, pero para test más difíciles estas cadenas fallan las pruebas.

Por último, el desempeño según el tiempo de muestreo, generalmente aquellos provenientes de un tiempo de muestreo menor tuvieron un mejor resultado, donde solo dos veces los bits recolectados con un tiempo de muestreo mayor mostraron mejor desempeño.

Capítulo 6

Conclusiones

6.1. Conclusiones Generales

En este trabajo de Tesis ha sido estudiado la generación de números aleaotorios considerando una fuente física basada en osciladores caóticos de doble enrrollamiento, tales como Circuito de Chua, Función Saturada y Diente de Sierra. Dos casos fueron considerados, el primero generando los números aleatorios partiendo de una arquitectura de comparación de niveles de referencia y la otra, considerando la técnica de Von Neumann para despreciar cadenas de bits repetidas. Es importante mencionar que antes de generar los números aleatorios, se realizó la optimización del exponente positivo de Lyapunov tomando en cuenta un algoritmo evolutivo del tipo diferencial. Posteriormente, se midio la entropía Kolmogorov-Sinai de la señal caótica y fue necesario utilizar un filtro digital para tener resultados correlacionados entre los datos númericos de MATLAB y SPICE, con los obtenidos de la serie de tiempo.

De esta manera, se implementó el generador de números aleatorios utilizando como fuente de aleatoridad las señales caóticas de los tres osciladores pero considerando los valores más altos de exponente de Lyapunov y entropía. De la implementación se almacenaron cadenas de bits para la caracterización por medio del estándar del NIST. De los resultados obtenidos, existen ciertos puntos que pueden ser vistos en general al haber realizado las pruebas de aleatoriedad. Primeramente las cadenas de bits que no son filtradas tienen la capacidad de aprobar los test básicos generalmente sin problema alguno, sin embargo con tests de mayor exigencia tienen muy poco rendimiento. Las pruebas en las que se mostró un menor rendimiento fueron Runs y Entropy. Otras pruebas realizadas como Rank y Linear, debido a la cantidad de bits necesarios para ejecutar estas pruebas era mayor a lo recolectado, solo fueron hechas con aquellas combinaciones que superaban este valor mínimo. Todas las pruebas que se hicieron para estos dos test fueron con cadenas de bits sin ocupar el filtro, sin embargo todas fueron aprobatorias.

Por lo anterior, aunque fueron pocas pruebas, podríamos asegurar que estos test también serían aprobados por cadenas de bits filtradas.

La utilización de un filtro mejora, sin lugar a duda, los resultados al ponerse a prueba las cadenas de bits recolectadas, por ejemplo en la combinación 1 para el oscilador caótico de Chua, la prueba de Entropy, los bits provenientes de un tiempo de muestreo mayor y sin filtro tienen un resultado de 0/1000 y la misma cadena filtrada tiene un resultado de 99/100. Por otra parte, aunque no se cumple en todas las combinaciones, en la mayoría, se puede observar que los resultados con un exponente positivo de Lyapunov mayor y con una entropía Kolmogorov-Sinai similar están dentro de aquellas combinaciones que lograron un mejor desempeño al momento de poner a prueba el generador de bits aleatorios. En general el uso de un oscilador caótico para generar números verdaderamente aleatorios muestra un buen rendimiento en cuanto a los resultados obtenidos además de tener la ventaja de ser fácilmente entonado, construido y modificado.

6.2. Trabajo a Futuro

- El filtro, aún cuando mostró un buen resultado para obtener los bits finales, su diseño específico es requerido. Entonces se espera que un filtro diseñado a la medida con un enfoque dirigido para pruebas del NIST tenga mejores resultados.
- Aunque en general el tiempo de muestreo no afecta en gran manera los resultados, se observó en algunas ocasiones que si fue un factor que provocaba pasar o no un determinado test. Entonces una mayor investigación sobre cuál podría ser un tiempo de muestreo óptimo que otorgue mejores resultados podría llevarse a cabo.
- Los niveles de umbral, estos se escogieron por practicidad y por conocimiento empírico al haber trabajado de esta forma anteriormente. Sin embargo pequeños cambios en los valores de umbral que quizá no eran apreciables a simple vista, provocaban trenes de pulsos muy diferentes. Entonces analizar los niveles de umbral para saber a qué valores de voltaje deben de establecerse y como afecta la exactitud al momento de ser entonados puede brindarnos un mejor diseño del generador de bits aleatorios.
- Comparar el desempeño y características con otros TRNG.

Apéndice A

Test de Aleatoriedad del NIST

En este apéndice se presenta una explicacion detallada sobre los test de aleatoriedad aplicadas en este trabajo de Tesis. Todos los test aplicados pertenecen a la suit definida por el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) del Departamento Comercio de los Estados Unidos de América.

A.1. Frecuencia (Monobit)

Descripción del Test:

- Conversión a ± 1: Los ceros y unos en la secuencia de entrada (ϵ) son convertidos a valores de -1 y $+1$ respectivamente, son sumados juntos para producir $S_n = X_1 + X_2 + \ldots + X_n$, donde $X_i = 2\varepsilon_{i-1}$.

  Por ejemplo si $\varepsilon = 1011010101$ entonces $n = 10$ y $S_n = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 = 2$.

- Computando el test estadístico $S_{obs} = \frac{|S_n|}{\sqrt{n}}$

  Por ejemplo en esta sección, $S_{obs} = \frac{|2|}{sqrt10} = 0.632455532$.

- Computando el $P - value = erfc(\frac{|S_{obs}|}{\sqrt{2}})$, donde $erfc$ es función de error complementario como es definida en la sección 5.5.3 del documento [7].

Por ejemplo en esta sección, $P - value = erfc((.0632455532)/\sqrt{2}) = 0.527089$.

Conclusión e Interpretación de Resultados:

Debido a que el P-value obtenido es ≥ 0.01 (i.e., $P-value = 0.527089$), la conclusión es que la secuencia es aleatoria.

Note que si el $P-Value$ fuera pequeño ($< 0-.01$), entonces esto podría ser causado debido a que $|S_n|$ o $|S_{obs}|$ es grande. Valores grandes positivos de S_n indican demasiados unos, y grandes valores negativos de S_n son indicadores de demasiados ceros.

A.2. Frecuencia Dentro de un Bloque

Descripción del Test:

- Se hace una partición a la secuencia como $N = |n/M|$ sin bloques encimados y descartando cualquier bit que no se ocupe.

  Por ejemplo, si $n = 10$,$M = 3$ y $\varepsilon = 0110011010$,3 bloques ($N = 3$) deberían ser creados, consistiendo de 011, 001 y 101. El '0' final es descartado.

- Determinar la proporción de π_i de unos en cada bloque de $M-bit$ usado en la ecuación $\pi_i = \frac{\sum_{j=1}^{M} \varepsilon_{(i-1)M+j}}{M}$, para $1 \leq i \leq N$.

  Por ejemplo en esta sección $\pi_1 = 2/3$, $\pi_2 = 1/3$ y $\pi_3 = 2/3$.

- Computar la X^2 estadística: $X^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - 1/2)^2$.

  Como ejemplo en esta sección:

  $X^2(obs) = 4x3x((2/3 - 1/2)^2 + (1/3 - 1/2)^2 + (1/3 - 1/2)^2) = 1.$

- Computar $P-Value = igamc(N/2, X^2(obs)/2)$, donde $igamc$ es la función gama incompleta para $Q(a, x)$ como es definida en sección 5.5.53 de la referencia [7].

Por ejemplo en esta sección, $P-Value = igamc(3/2, 1/2) = 0.801252$.

Interpretación de Resultados:

Debido a que el $P-value$ obtenido es ≥ 0.01 (i.e., $P-value = 0.801252$), la conclusión es que la secuencia es aleatoria.

Note que pequeños $P-values(< 0.01)$ deben indicar una larga desviación de una proporción ecuánime de unos y ceros dentro de al menos uno de los bloques.

A.3. Corridas

Descripción del test:

- Computa el pre-test proporción π de unos en la secuencia de entrada: $\pi = \frac{\sum_j \varepsilon_j}{n}$.

  Por ejemplo, si $\varepsilon = 1001101011$, entonces $n = 10$ y $\pi = 6/10 = 3/5$.

- Determinar cómo prerrequisito el test de Frequency, si puede ser mostrado que $|\pi - 1/2| \geq \tau$, entonces el test Runs no necesita ser llevado a cabo (i.e., el test no debería de tener Run's debido a una falla al pasar el test de Frequency). Si el test es no aplicable, entonces el $P - value$ se establecerá a 0.0000. Note que para este test,$\tau = 2/\sqrt{n}$ a sido predefinido dentro del código del test.

  Por ejemplo, $\tau = 2/\sqrt{10} = 0.63246$, por lo tanto $|\pi - 1/2| = |3/5 - 1/2| = 0.1 < \tau$, y el test no correrá. Debido a que el valor observado π esta dentro de los límites seleccionados, el test es aplicable.

- Computando el test estadístico $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$, donde $r(k) = 0$ si $\varepsilon_k = \varepsilon_{k+1}$, y $r(k) = 1$ de otra manera.

  Si tenemos $\varepsilon = 1001101011$, entonces $V_{10}(obs) = (1+0+1+0+1+1+1+1+1+0)+1 = 7$

- Computando $P - value = erfc\left(\frac{V_n(obs) - 2n\pi(1-\pi)}{2\sqrt{2n\pi(1-\pi)}}\right)$

Por ejemplo $P - valua = erfc\left(\frac{7-(2\cdot 10\cdot \frac{3}{5}(1-\frac{3}{5}))}{2\cdot\sqrt{(2\cdot 10)\cdot\frac{3}{5}\cdot(1-\frac{3}{5})}}\right) = 0.147232$

Conclusión e interpretación de resultados:

Debido a que $P - value$ obtenido es ≥ 0.01 (i.e.,$P - value = 0.147232$), la conclusión es que la secuencia es aleatoria.

Nota que un valor grande de $V_n(obs)$ debe haber indicado una oscilación en la cadena que es demasiado rápida; un valor pequeño ha indicado que la oscilación es demasiado lenta. (Una oscilación consiste en un cambio de uno a cero o viceversa.) Una oscilación rápida ocurre cuando existen muchos cambios, e.g., 010101010 oscila cada bit. Un flujo con una oscilación lenta tiene menos corridas de las que se esperan para una secuencia aleatoria, e.g., una secuencia que contiene 100 unos, seguida de 73 ceros, seguida por 127 unos (300 bits en total) solamente tendría 3 corridas, donde 150 corridas son esperadas.

A.4. Mayor Corrida Dentro de un Bloque

Descripción del test:

- Dividir la secuencia en bloques de $M - bits$.
- Tabular las frecuencia v_i de la mayor cantidad de runs de unos en cada bloque dentro de categorías, donde cada división contiene el numero de runs de unos para una cantidad especificada.

  Para los valores de M soportados por la test suit, la división v_i quedara de la manera como lo ejemplifica la tabla A.1:

v_i	M=8	M=128	M=10,000
v_0	≤ 1	≤ 4	≤ 10
v_1	2	5	11
v_2	3	6	12
v_3	≥ 4	7	13
v_4		8	14
v_5		≥ 9	15
v_6			≥ 16

Cuadro A.1: Valores de M soportados y diviciones v_i.

- Computando $X^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, donde los valores para π_i son provistos en la sección 3.4.Los valores d, K y N son determinados por el valor de M de acuerdo con la tabla A.2:

M	K	N
8	3	16
128	5	49
10,000	6	75

Cuadro A.2: Valores de K, N, segun el M escogido.

  Por ejemplo:

  $X^2(obs) = \frac{(4-16(.2148))^2}{16(.2148)} + \frac{(9-16(.3672))^2}{16(.3672)} + \frac{(3-16(.2305))^2}{16(.2305)} + \frac{(0-16(.1875))^2}{16(.1875)} = 4.882605$

- Computar el $P - value = igamc(\frac{K}{2}, \frac{X^2(obs)}{2})$.

Por ejemplo, $P - value = igamc(\frac{3}{2}, \frac{4.882605}{2}) = 0.180598$

Conclusión e interpretación de resultados:

Debido a que el $P - value \geq 0.01$,($P - value = 0.180609$), la conclusión es que la secuencia es aleatoria. Note que valores grandes de $X^2(obs)$ indican que la secuencia testeada contiene agrupaciones de unos.

A.5. Rango de Matriz Binaria

Descripción del test:

- Secuencialmente divide la secuencia en $M \cdot Q - bit$ bloques divididos, donde la cantidad de bloques será $N = \left|\frac{n}{MQ}\right|$. Descartando bits que no son usados en la computación dentro de cada bloque. Colectando los $M \cdot Q$ segmentos dentro de matrices de MxQ. Cada renglón de la matriz es llenado con bloques de Q bit's provenientes de la secuencia original ε.

  Por ejemplo, si $n = 20$, $M = Q = 3$ y $\varepsilon = 01011001001010101101$, entonces la partición de la cadena dentro de $N = \left|\frac{n}{3 \cdot 3}\right| = 2$ matrices de $M \cdot Q(3 \cdot 3 = 9)$. Note que al menos dos bits (0 y 1) serán descartados. Donde las matrices formadas son $\begin{vmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{vmatrix}$ y $\begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}$.
  Note que la primera matriz consiste de los primeros 3 bits dentro del renglón 1, el segundo conjunto de 3 bits en el renglón 2, el tercer juego de bits en el tercer renglón. La segunda matriz es construida similarmente usando los siguientes 9 bits en la secuencia.

- Determine el rango binario de la matriz (R_l) de cada matriz, donde $l = 1, \cdots, N$. El método para determinar el rango es descrito en el apéndice A de la referencia [7].

  Por ejemplo en esta sección, el rango de la primera matriz es $2(R_1 = 2)$ y el sango para la segunda matriz es $3(R_2 = 3)$.

- Pongamos F_M= el número de matrices con $R_l = M$ (rango completo).

  F_{M-1} = el numero de matrizes con $R_l = M - 1$(rango completo -1).

  $N - F_M - F_{M-1}$ = el numero de matrices restantes.

  Por ejemplo, $F_M = F_3 = 1$ (R_2 tiene un rango completo de 3), $F_{M-1} = F_2 = 1$ (R_1 tiene rango 2), y no hay matrices con un rango menor.

  Computando:

  $X^2(obs) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N}$

  En el ejemplo de esta sección:

  $X^2(obs) = \frac{(1 - 0.2888 \cdot 2)^2}{0.2888 \cdot 2} + \frac{(1 - 0.5776 \cdot 2)^2}{0.5776 \cdot 2} + \frac{(2 - 1 - 1 - 0.1336 \cdot 2)^2}{0.1336 \cdot 2} = 0.596953$

- Computar los P-value= $e^{-X^2(obs)/2}$. Como hay 3 clases en este ejemplo, el $P-value$ es igual a $igamc(1, \frac{X^2(obs)}{2})$.

  Para este ejemplo, $P-value = e^{0.596953/2} = 0.741948$.

Conclusión e interpretación de resultados:

Debido a que el P-value obtenido es ≥ 0.01 ($P-value = 0.741948$) la conclusión es que la secuencia de bits es aleatoria. Note que grandes valores de $X^2(obs)$ (y por lo tanto, pequeños $P-values$) indicarían una desviación en la distribución de rango de aquella correspondiente a una secuencia aleatoria.

A.6. Transformada Discreta de Fourier

Descripción del test:

- Los ceros y unos de la secuencia de entrada (ε) son convertidos en valores de -1 y $+1$ para crear la secuencia $X = x_1, x_2, \cdots, x_n$ donde $x_i = 2\varepsilon_i - 1$.

  Por ejemplo si $n = 10$ y $\varepsilon = 1001010011$, entonces $X = 1, -1, -1, 1, -1, 1, -1, -1, 1, 1$.

- Aplicando la transformada discreta de Fourier sobre X para producir $S = DFT(X)$. Una secuencia de variables complejas es producida que representa que representa componentes periódicas de la secuencia de bits a diferentes frecuencias.

- Calculando $M = modulus(S') = |S|$, donde S' es la sub cadena que consiste de los primeros $n/2$ elementos dentro de S, y la función $modulus$ produce una secuencia de alturas de picos.

- Computado $T = \sqrt{\log \frac{1}{0.05} n}$ = al 95 % del valor máximo de altura de umbral. Bajo una suposición de aleatoriedad, 95 % de los valores obtenidos del test no deberían exceder T.

- Computando $N_0 = .95n/2$. N_0 es el numero teórico esperado de picos (bajo la suposición de aleatoriedad) que es menos que T.

  Por ejemplo en esta sección, $N_0 = 4.75$.

- Computando $N_1 =$ es actual numero observado de picos dentro de M que son menores a T.

  Por ejemplo, en esta sección $N_1 = 4$.

- Computando $d = \frac{N_1 - N_0}{\sqrt{n(.95)(.05)/4}}$

  En el ejemplo $d = \frac{4-4.75}{\sqrt{10(.95)(.05)/4}} = -2.176429$.

- Computando $P - value = erfc(\frac{|d|}{\sqrt{2}})$.

  Para el ejemplo $P - value = erfc(\frac{|2.176492|}{\sqrt{2}} = 0.029523$.

Conclusión e interpretación de resultados: Como el P-value obtenido es ≥ 0.01 ($P - value = 0.029523$), la conclusión es que la secuencia es aleatoria. Un d valor que es demasiado bajo es indicador de que hubieron muy pocos picos ($< 95\,\%$) abajo de T, y demasiados picos (más del $5\,\%$) arriba de T.

A.7. Complejidad Lineal

Descripción del test:

- Partición de la secuencia de $n-bit$ en N independientes bloques de M bits, donde $n = MN$.
- Usando el algoritmo de Berlekamp-Massey, determinamos la complejidad lineal L_i de cada N bloque ($i = 1, \cdots, N$). L_i es la longitud más pequeña del registro de cambio de retroalimentación lineal que genera todos los bits en el bloque i. Dentro de cualquier secuencia de $L_i - bit$, alguna combinación de bits, cuando se le suma junto con el modulo 2, produce el siguiente bit en la secuencia (bit $L_i + 1$).

  Por ejemplo, si $M = 13$ y el bloque a ser testeado es 1101011110001, entonces $L_i = 4$, y la secuencia es producida por sumar el 1º y 2º bit dentro de una sub cadena de 4 bits para producir el siguiente bit (el 5º bit), la exanimación procede como sigue:

	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
Los primeros 4 bits y resultando el 5º bit	1	1	0	1	0
Bits 2-5 y resultando el 6º bit	1	0	1	0	1
Bits 3-6 u resultando el 7º bit	0	1	0	1	1
Bits 4-7 y resultando el 8º bit	1	0	1	1	1
Bits 5-8 y resultando el 9º bit	0	1	1	1	1
Bits 6-9 y resultando el 10º bit	1	1	1	1	0
Bits 7-10 y resultando el 11º bit	1	1	1	0	0
Bits 8-11 y resultando el 12º bit	1	1	0	0	0
Bits 9-12 y resultando el 13º bit	1	0	0	0	1

  Para este bloque, el algoritmo retroalimentado de prueba funciona. Si no fuera el caso, otra retroalimentación podría ser intentada por el bloque (ejemplo, sumar bit 1 y 3 para producir el 5º bit, o sumar bits 1, 2 y 3 para producir el bit 6, etc.)

- Bajo el supuesto de aleatoriedad, calcular el teórico principal μ:

$\mu = \frac{M}{2} + \frac{9+(-1)^{M+1}}{36} - \frac{\frac{M}{3}+\frac{2}{9}}{2^M}$.

Por ejemplo en esta sección, $\mu = \frac{13}{2} + \frac{9+(-1)^{13+1}}{36} - \frac{\frac{13}{3}+\frac{2}{9}}{2^M} = 6.777222$.

- Para cada sub cadena, calcular un valor de T_i donde $T_i = (-1)^M \cdot (L_i - \mu) + \frac{2}{9}$.

  Por ejemplo, $T_i = (-1)^{13}(4 - 6.777222) + \frac{2}{9} = 2.999444$.

  Almacenando los T_i valores dentro $V_0, \cdots, V_6$ como sigue:

$$
\begin{array}{rrlll}
Si: & & & & \\
 & T_i \leq -2.5 & Incrementa & V_0 & por\ uno \\
-2.5 < & T_i \leq -1.5 & Incrementa & V_1 & por\ uno \\
-1.5 < & T_i \leq -0.5 & Incrementa & V_2 & por\ uno \\
-0.5 < & T_i \leq 0.5 & Incrementa & V_3 & por\ uno \\
0.5 < & T_i \leq 1.5 & Incrementa & V_4 & por\ uno \\
1.5 < & T_i \leq 2.5 & Incrementa & V_5 & por\ uno \\
 & T_i > 2.5 & Incrementa & V_6 & por\ uno.
\end{array}
$$

- Computado $X^2(obs) = \sum_{1=0}^{K} \frac{V_i - N\pi_i}{(N\pi_i)^2}$, donde $\pi_0 = 0.010417, \pi_1 = 0.03125, \pi_2 = 0.125, \pi_3 = 0.5, \pi_4 = 0.25, \pi_5 = 0.0625, \pi_6 = 0.020833$ son las probabilidades computadas por la ecuación.
- Computando el $P - value = igamc(\frac{K}{2}, \frac{X^2(obs)}{2})$.

Conclusión e interpretación de resultados:

Si el $P - value < 0.01$ se toma como secuencia no aleatoria. Esto debería indicar que la cuenta de la frecuencia observada de T_i almacenada dentro de V_I varia para valores esperados; es esperado que la distribución de la frecuencia de T_i ser proporcional al π_i computado.

A.8. Serial

Descripción del Test:

- Para una secuencia aumentada ε': Extiende la secuencia mediante la adición de los primeros $m - 1$ bits al final de la secuencia para distintos valores de n.

Por ejemplo, dando $n = 10$ y $\varepsilon = 0011011101$. Si $m = 3$, entonces $\varepsilon' = 001101110100$. Si $M = 2$, entonces $\varepsilon' = 00110111010$. Si $M = 1$, entonces $\varepsilon' =$ la secuencia original 0011011101.

- Determinando la frecuencia para todos los bloques de $m - bit$ que pudieran sobre encimarse, todas las posibles superposiciones de bloques de $(m-1)$ bits y todas las posibles superposiciones de bloques de $(m-2)$bits. Dejando $V_{i1,\cdots,im}$ denotando la frecunacia del patrón $m-bit$ $i_1 \cdots i_m$; dejando $V_{i,\cdots,i_{m-1}}$ denotando la frecuencia del patrón de bit $(m-1)$ como $i_1, \cdots, i_{m-1}$; y dejando $V_{i,\cdots,i_{m-2}}$ denotando la frecuencia del patrón $(m-2)$bit como $i_1, \cdots, i_{m-2}$.

  Por ejemplo en esta sección, cuando $m = 3$, entonces $(m-1) = 2$, y $(m-2) = 1$. La frecuencia de todos los bloques de $3 - bits$ son: $v_{000} = 0$, $v_{001} = 1$, $v_{010} = 1$, $v_{011} = 1$, $v_{100} = 1$, $v_{101} = 2$, $v_{110} = 2$, $v_{111=0}$. La frecuencia de todos los posibles bloques $(m-1)$bit es: $v_{00} = 1$, $v_{01} = 3$, $v_{10} = 3$, $v_{11} = 3$. La frecuencia para todos los bloques $(m-2)$bit es: $v_0 = 4$, $v_1 = 6$.

  Computando:

$$\begin{aligned}
\psi_m^2 = & \quad \tfrac{2^m}{n}\textstyle\sum_{i_1\cdots i_m}(v_{i_1\cdots i_m} - \tfrac{n}{2^m})^2 & = \frac{2^m}{n}\sum_{i_1\cdots i_m} v_{i_1\cdots i_m}^2 - n \\
\psi_{m-1}^2 = & \quad \tfrac{2^{m-1}}{n}\textstyle\sum_{i_1\cdots i_{m-1}}(v_{i_1\cdots i_{m-1}} - \tfrac{n}{2^{m-1}})^2 & = \frac{2^{m-1}}{n}\sum_{i_1\cdots i_{m-1}} v_{i_1\cdots i_{m-1}}^2 - n \\
\psi_{m-2}^2 = & \quad \tfrac{2^{m-2}}{n}\textstyle\sum_{i_1\cdots i_{m-2}}(v_{i_1\cdots i_{m-2}} - \tfrac{n}{2^{m-2}})^2 & = \frac{2^{m-2}}{n}\sum_{i_1\cdots i_{m-2}} v_{i_1\cdots i_{m-2}}^2 - n
\end{aligned}$$

  Para esta sección:

  $\psi_3^2 = \frac{2^3}{10}(0+1+1+4+1+4+4+1) - 10 = 12.8 - 10 = 2.8$

  $\psi_2^2 = \frac{2^2}{10}(1+9+9+9) - 10 = 11.2 - 10 = 1.2$

  $\psi_1^2 = \frac{2}{10}(16+36) - 10 = 10.4 - 10 = 0.4$

  Computando $\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2$, y $\nabla^2\psi_m^2 = \psi_m^2 - \psi_{m-1)}^2 + \psi_{m-2}^2$.

  Para esta sección:

  $\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2 = \psi_3^2 - \psi_2^2 = 2.8 - 1.2 = 1.6$

  $\nabla^2\psi_m^2 = \psi_m^2 - \psi_{m-1)}^2 + \psi_{m-2}^2 = \psi_3^2 - 2\psi_2^2 + \psi_1^2 = 2.8 - 2(1.2) + 0.4 = 0.8$

- Computando el:

  $P - value1 = igamc(2^{m-2}, \nabla\psi_m^2)$

  $P - value2 = igamc(2^{m-3}, \nabla^2\psi_m^2)$

Por ejemplo en esta sección

$P - value1 = igamc(2, \frac{1.6}{2}) = 0.9057$

$P - value1 = igamc(1, \frac{0.8}{2}) = 0.8805$

Conclusión e interpretación de resultados: Debido a que el $P - value1 \geq 0.01$ ($P - value1 = 0.808792$ y $P - value2 = 0.670320$) se concluye que la secuencia es aleatoria. Note que si $\nabla^2\psi_m^2$ o $\nabla\psi_m^2$ hubieran sido grandes implica una no uniformidad de bloques de $m - bit$.

A.9. Entropía Aproximada

Descripción del Test:

- Aumentando la secuencia de $n - bit$ para crear n secuencias de $m - bit$ superpuestas mediante la adición de $m - 1$ bits del inicio de la secuencia hasta el final de la secuencia.

  Por ejemplo, si $\varepsilon = 0100110101$ y $m = 3$, entonces $n = 10$. Añadiendo el 0 y 1 en el principio de la secuencia al final de la secuencia. La secuencia a ser testeada se vuelve 010011010101. Esto es aplicado para cada valor de m.

- Una cuenta de frecuencia es hecha de los n bloques superpuestos (por ejemplo, si un bloque contienen ε_j a ε_{j+m-1} es examinada en el tiempo j, entonces el bloque contiene ε_{j+1} a ε_{j+m} es examinada en el tiempo $j+1$. Dejando la cuenta de el posible $m-bit((m+1)-bit)$ valores ser representada como C_i^m, donde i es el valor del $m - bit$.

  Por ejemplo en esta sección, la superposición de bloques de $m - bit$ (donde $m = 3$), se convierte 010, 100, 001, 011, 110, 101, 010, 101, 010 y 101.

  Las cuentas calculadas para los $2^m = 2^3 = 8$ posibles $m - bits$ son:

  $\sharp 000 = 0$, $\sharp 001 = 1$, $\sharp 010 = 3$, $\sharp 011 = 1$, $\sharp 100 = 1$, $\sharp 101 = 3$, $\sharp 110 = 1$, $\sharp 111 = 0$.

- Computando $C_i^m = \frac{\sharp i}{n}$ para cada valor de i.

  Por ejemplo en esta sección:

  C_{000}^3=0, $C_{001}^3 = 0.1$, $C_{010}^3 = 0.3$, $C_{011}^3 = 0.1$, $C_{100}^3 = 0.1$, $C_{101}^3 = 0.3$, $C_{110}^3 = 0.1$, $C_{111}^3 = 0$.

- Computando $\varphi^m = \sum_{i=0}^{2^{m-1}} \pi_i \log \pi_i$ donde $\pi_i = C_j^3$ y $j = \log_2 i$.

  Para el ejemplo de esta sección:

  $\varphi^3 = 0(\log 0) + 0.1(\log(0.1) + 0.3(\log 0.3) + 0.1(\log 0.1) + 0.1(\log 0.1) + 0.3(\log 0.3) + 0.1(\log 0.1) + 0(\log 0) = -1.64341772$.

- Se repiten los pasos anteriores remplazando m por $m+1$.

  Por ejemplo, primero m es ahora 4, la secuencia a ser testeada se convierte en 0100110101010.

  Los bloques de superposición ahora son 0100, 1001, 0011, 0110, 1101, 1010, 0101, 1010, 0101, 1010. Los valores calculados son: $\sharp 0011 = 1$, $\sharp 0100 = 1$, $\sharp 0101 = 2$, $\sharp 0110 = 1$, $\sharp 1001 = 1$, $\sharp 1010 = 3$, $\sharp 1101 = 1$, y todos los demás patrones son 0.

  Después: $C^4_{0011} = C^4_{0100} = C^4_{0110} = C^4_{1001} = C^4_{1101} = 0.1, C^4_{0101} = 0.2, C^4_{1010} = 0.3$, y todos los demás valores son cero.

  Por último: $\varphi^4 = 0+0+0+0.1(\log 0.01)+0.1(\log 0.01)+0.2(\log 0.02)+0.1(\log 0.01)+1+1+0.1(\log 0.01)+0.3(\log 0.03)+0+0+0.1(\log 0.01)+0+0) = -1.83437197$.

- Computando el test estadístico: $X^2 = 2n[\log 2 - ApEn(m)]$; donde $ApEn(m) = \varphi^m - \varphi^{m+1}$.

  Del ejemplo de esta sección:

  $ApEn(3) = -1.643418 - (-1.834372) = 0.190954$.

  $X^2 = 2 \cdot 10(0.693147 - 0.190954) = 0.502193$.

- Computando el $P - value = igamc(2^{m-1}, \frac{X^2}{2})$.

Del ejemplo de esta sección, $P - value = igamc(2^2, \frac{0.502193}{2}) = 0.261961$.

Conclusión e interpretación de resultados: Ya que el $P - value$ obtenido es ≥ 0.01 ($P - value = 0.261961$), la conclusión es que la secuencia es aleatoria. Note que pequeños valores de $ApEn(m)$ debe implicar una fuerte regularidad. Grandes valores deben implicar fluctuaciones substanciales o irregularidad.

A.10. Sumas Acumuladas

Descripción del test:

- De una secuencia normalizada: Los ceros y unos de la secuencia de entrada ε son convertidos a valores X_i de -1 y $+1$ usando $X_i = 2\varepsilon_i - 1$.

  Por ejemplo, si $\varepsilon = 1011010111$, entonces $X = 1, -1, 1, 1, -1, 1, -1, 1, 1, 1$.

- Se computa la suma parcial S_i en orden de las sub secuencias mas grades, cada empiezo con X_1 (si $mode = 0$) o X_2 (si $mode = 1$).

  Esto es $S_k = S_{k-1} + X_k$ para el mode 0, y $S_k = S_{k-1} + X_{n-k+1}$ para $mode1$.

Mode = 0 (forward)	Mode = 1 (backward)
$S_1 = X_1$	$S_1 = X_n$
$S_2 = X_1 + X_2$	$S_2 = X_n + X_{n-1}$
$S_3 = X_1 + X_2 + X_3$	$S_3 = X_n + X_{n-1} + X_{n-2}$
.	.
.	.
$S_k = X_1 + X_2 + X_3 + \cdots + X_k$	$S_2 = X_n + X_{n-1} + X_{n-2} + \cdots + X_{n-k+1}$
.	.
.	.
$S_k = X_1 + X_2 + X_3 + \cdots + X_k + \cdots + X_n$	$S_2 = X_n + X_{n-1} + X_{n-2} + \cdots + X_{k-1} + \cdots + X_1$

Por ejemplo en esta sección, cuando $mode = 0$ y $X = 1$, (-1), 1, 1, (-1), 1, (-1), 1, 1, 1, entonces:

$$\begin{aligned}
S_1 &= 1 \\
S_2 &= 1 + (-1) = 0 \\
S_3 &= 1 + (-1) + 1 = 1 \\
S_4 &= 1 + (-1) + 1 + 1 = 2 \\
S_5 &= 1 + (-1) + 1 + 1 + (-1) = 1 \\
S_6 &= 1 + (-1) + 1 + 1 + (-1) + 1 = 2 \\
S_7 &= 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) = 1 \\
S_8 &= 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 = 2 \\
S_9 &= 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + 1 = 3 \\
S_{10} &= 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + 1 + 1 = 4
\end{aligned}$$

- Computar el test estadístico $z = max_{1 \leq k \leq n} |s_k|$, donde $max_{1 \leq k \leq n} |s_k|$ el mayor de los valores absolutos de la suma parcial S_k.

  Para esta sección, S_k es 4, por lo tanto $z = 4$.

- Computar el $P - value =$
  $1 - \sum_{k=(-\frac{n}{z}+1)/4}^{(\frac{n}{z}-1)/4} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] + \sum_{k=(\frac{-n}{z-3})4}^{(\frac{n}{z}-1)/4} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right]$

Donde Φ es la Función Estándar Normal de distribución de Probabilidad Cumulativa. Por el ejemplo de esta sección es $P - value = 0.4116588$.

Conclusión e interpretación de resultados:

Debido al $P - valor$ obtenido ($P - value = 0.411658$), la conclusión es que la secuencia es aleatoria.

Note que en el $mode = 0$, largos valores indican que existe alguno demasiados unos o demasiados ceros en las primeras fases de la secuencia. Cuando está en el $mode = 1$, grandes valores de esto indica que existen ya sea demasiados unos o demasiados ceros en las últimas fases de la secuencia. Pequeños valores indican que unos y ceros están entremezclados balanceadamente.

Apéndice B

Evidencias de implementación

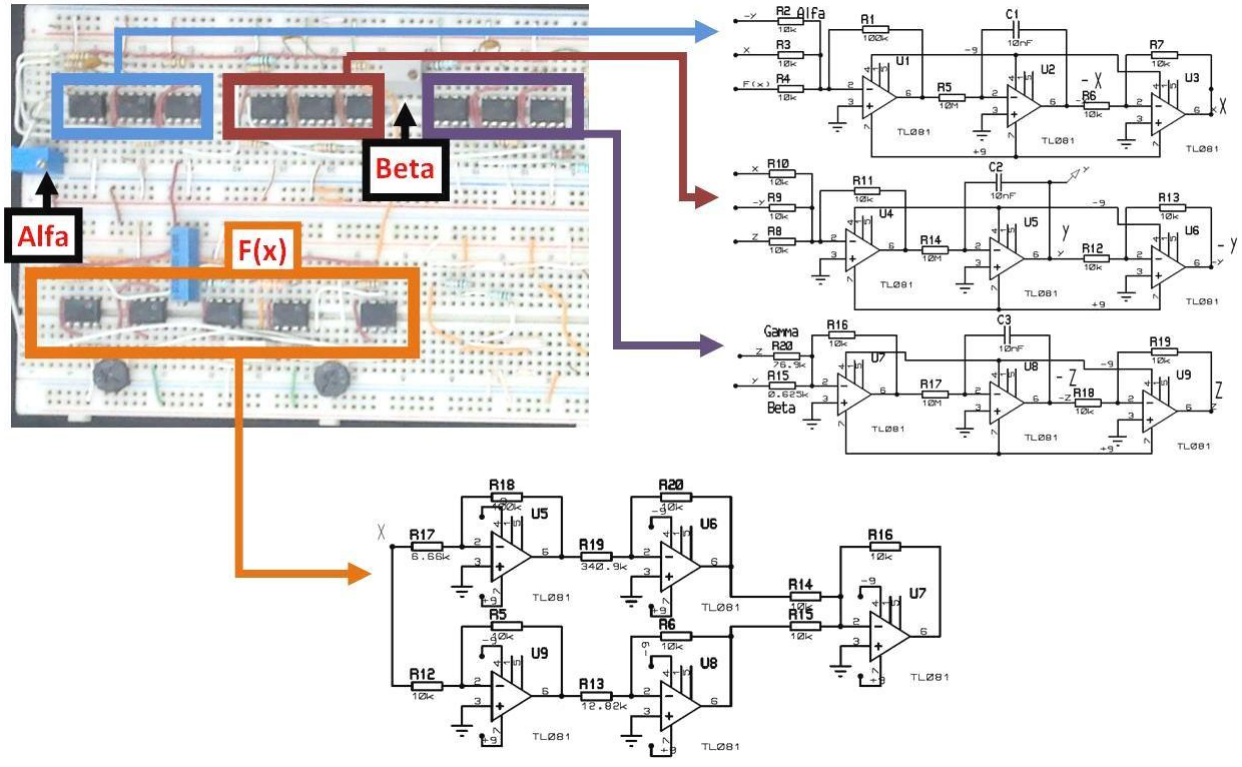

Figura B.1: Implementación del oscilador caótico de Chua

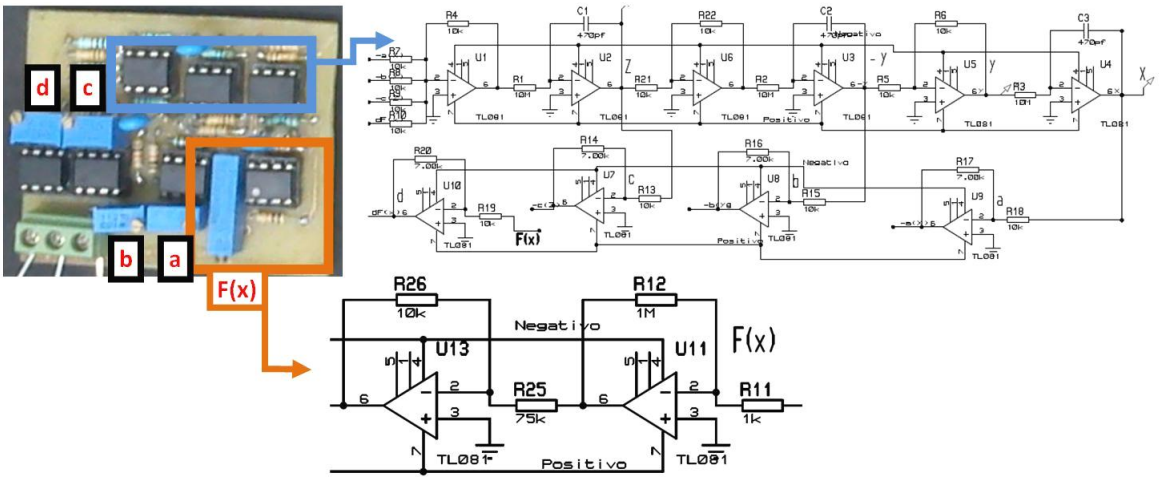

FIGURA B.2: Implementación del oscilador caótico de función saturada

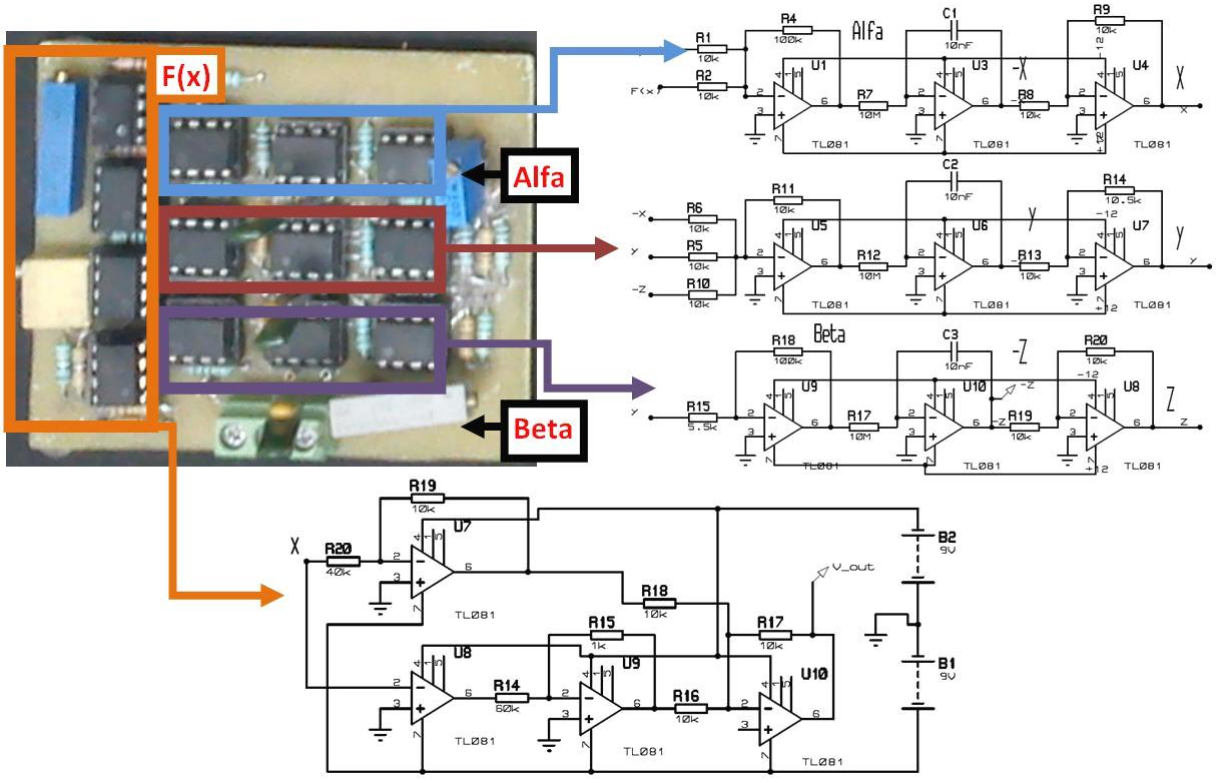

FIGURA B.3: Implementación del oscilador caótico de función diente de sierra

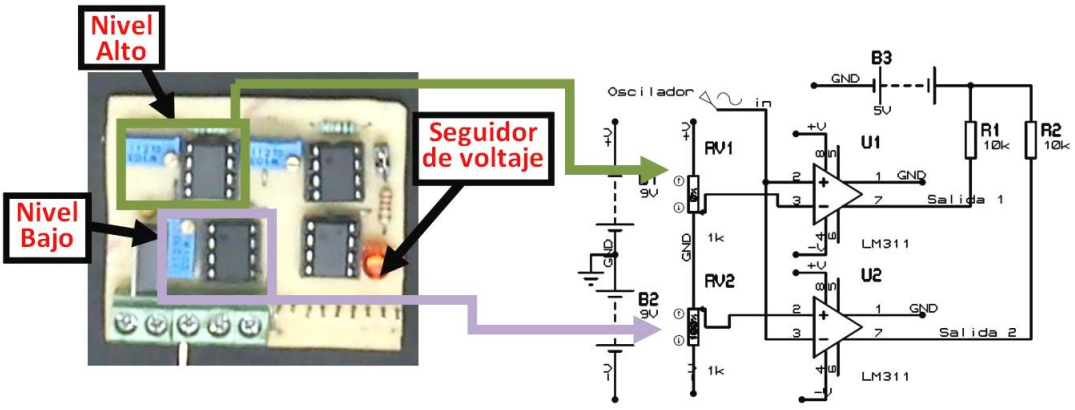

FIGURA B.4: Implementación de los comparadores de nivel.

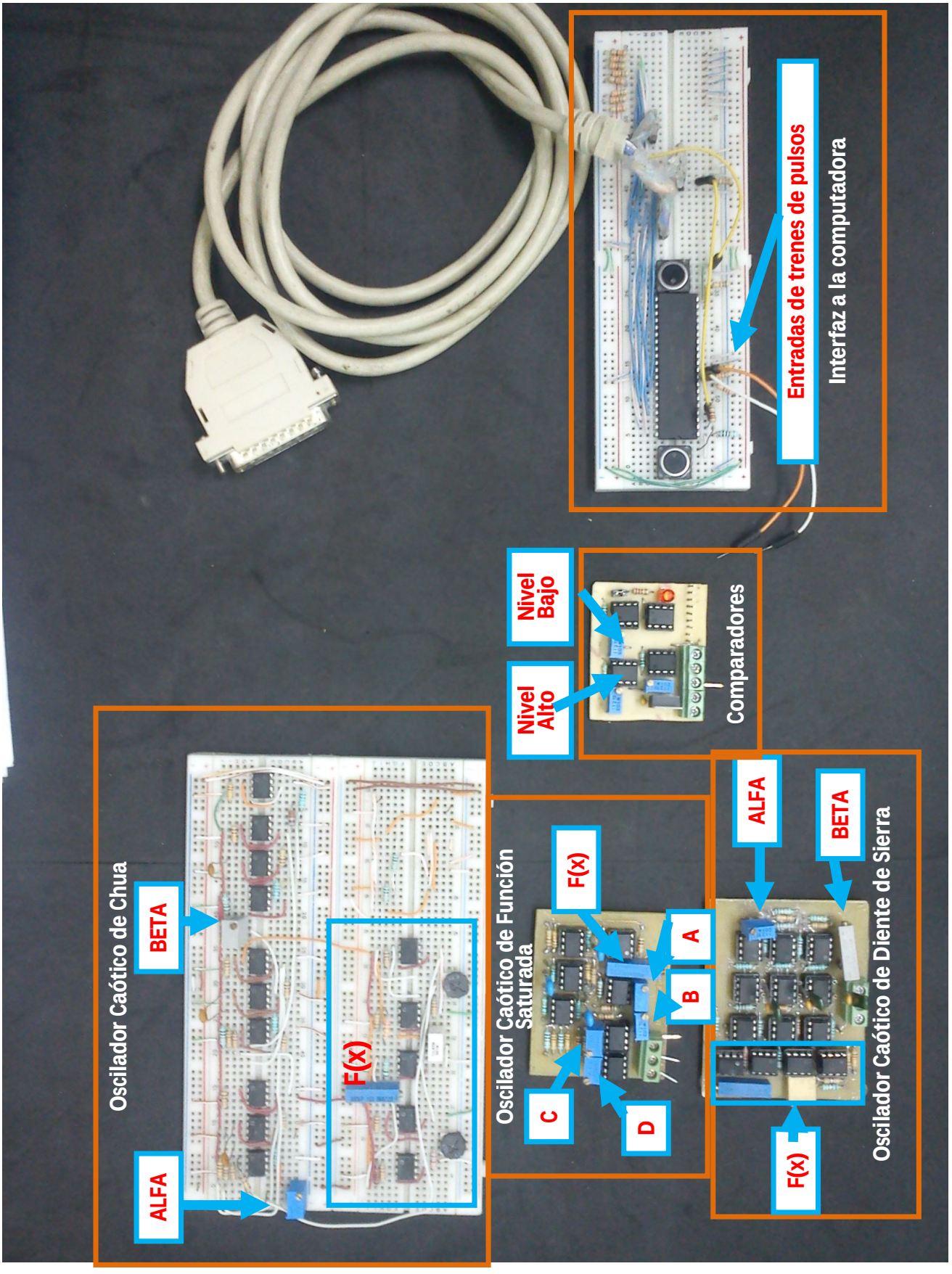

FIGURA B.5: Implementación de los circuitos usados

Bibliografía

[1] Michael Peter Kennedy, Three Steps to Chaos Part I: Evolution, IEEE Transactions on Circuit and Systems, vol. 40, no. 10, 1993.

[2] Charmaine Kenny, Random Number Generators An Evaluation and Comparison of Random.org and Some Commonly Used Generators, Trinity College of Dublin, Project Report 2005.

[3] Victor Hugo Carbajal Gómez, Diseño de osciladores caóticos para comunicaciones seguras, Benemerita Universidad Autónoma de Puebla, Puebla, Manuscrito de Tesis, 2009.

[4] Michael Peter Kennedy, Experimental Chaos from Autonomous Electronics Circuits, The Royal Society TEX Paper, pp. 13-32, 1995.

[5] Michael Peter Kennedy, Three Steps to Chaos Part II: A Chua's Circuit Primer, IEEE Transactions on Circuits and Systems, vol. 40, no. 10, 1993.

[6] Alfonso Manuel Mancilla Herrera, Números Aleatorios Historia, Teoría y Aplicaciones, Ingenieria Desarrollo, pp. 49-69, 2000.

[7] JuanSoto, et al., A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST, National Institute of Standars and Technology, Reports on Computer Systems Technology, 2010.

[8] Alfonso Manuel Mancilla Herrera, Números Aleatorios. Historia teoría y aplicaciones, Ingenieria y desarrollo, no. 8, 2000.

[9] Charmaine Kenny, Random Number Generators: An Evaluation and Comparison of Random.org and Some Commonly Used Generators, Trinity College of Dublin, Project Report 2005.

[10] D. Kahaner, Numerical Methods and Software.: Prentice Hall, 1977

[11] Leon O. Chua, The genesis of Chua's Circuit, Hirzel-Verlag, vol. 46, no. 4, pp. 250-257, 1992.

[12] G. M. Ramírez A. G. Conde S, Estudio de dos circuitos caóticos, Revista Boliviana de Física, pp. 58-74, 2007.

[13] Johan A. K. Sukens, Joos Vandewalle Müstak E. Yalcin, True Random Bit Generation From a Double-Scroll Attractor, IEEE Transactions on Circuits and Systems, vol. 51, no. 7, 2004

[14] Serdar Özoguz Salih Ergün, A Chaos-Modulated Dual Oscillator-Based Truly Random Number Generator, IEEE Transactions on Circuits and Systems, 2007.

[15] Ch. Skokos, The Lyapunov Characteristics Exponents and Their Computation, Spinger-Verlag Berlin Heidelberg, 2010.

[16] M. Cencini, F. Cecconi and A. Vulpiani, Chaos: From Simple Models to Complex Systems, Series on Advances in Statistical Mechanics: Volume 17, 480pp, World Scientific Publishing, Singapore, 2009.

[17] Michael Peter Kennedy, Robust Op Amp realization of Chua's circuit, Frequenz, vol. 46, no. 3-4, pp. 66-80, 1992.

[18] Gui-Nian Lin Leon O. Chua, Canonical Realization of Chua's Circuit Family, IEEE Transaction on circuits and systems, vol. 37, no. 7, July 1990.

[19] Chai Wah Wu, Anshan Huang, and Guo-Qun Zhong Leon O. Chua, A Universal Circuit for Studying and Generating Chaos-PartII: Strange Attractors, IEEE transactions on circuits and systems, vol. 40, no. 10, October 1993.

[20] Leon O. Chua, Chai Wah Wu, Anshang Huang, and Guo-Qun Zhong, A Universal Circuit for Studying and Generating Chaos - Part I: Routes to Chaos, IEEE transactions on circuits and systems-Part I: Fundamental theory and applications, vol. 40, no. 10, 1993.

[21] S. Ozoguz, J.A.K. Suykens and J. Vandewalle M.E. Yalqin, N-scroll chaos generators: a simple circuit, Electronics Letters, vol. 37, no. 3, Noviembre 2000.

[22] E. Tlelo-Cuautle J. M. Muñoz-Pacheco, Automatic synthesis of 2D-n-scrolls chaotic systems by behavioral modeling, Journal of Applied Research and Technology, vol. 7, no. 1, 2009.

[23] Jesús-Manuel Muñoz-Pacheco Esteban Tlelo-Cuautle, Automatic Simulation of 1D and 2D Chaotic Oscillators, Journal of Physics: Conference Series, International Symposium on Nonlinear Dynamics, 2007.

[24] J. C. Sprott, A new class of chaotic circuit, Physics Letters A, 2000.

[25] Wallace K. S. Tang and G. Chen, Simin Yu, Generation of nxm-scroll attractors under a Chua-circuit framework, International Journal of Bifurcation and Chaos, vol. 7, no. 11, 2007.

[26] Esteban Tlelo Cuautle Jesús Manuel Muñoz Pacheco, Synthesis of n-Scroll Attractors using Saturated Functions from High Level Simulation, 2007 International Symposium on Nonlinear Dynamics Journal of Physics: Conference Series , 2008.

[27] I.M. Kyprianidis, I.N. Stouboulos Ch.K. Volos, A chaotic path planning generator for autonomous mobile robots, Robotics and Autonomous Systems, 2012.

[28] A. S. Demirkol, S. Ozoguz, S. Kilinc, A. Toker, A. Zeki V. Tavas, An IC Random Number Generator Based on Chaos, International Conference on Applied Electronics, 2010.