BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
Facultad de Ciencias de la Electrónica

TESIS

COMO REQUISITO PARCIAL PARA OBTENER EL TÍTULO DE:
LICENCIADO EN ELECTRÓNICA

TÍTULO DE LA TESIS:

SISTEMA DE CIFRADO DE DATOS USANDO CAOS COMO FUENTE DE ENTROPÍA

PRESENTA:
José Alberto Azamar Montero

Asesores:

Dr. Jesús Manuel Muñoz Pacheco
cDra. Lízbeth Vargas Cabrera

Puebla, Pue. México, 26 de Junio del 2025

Agradecimientos

Mi más profundo agradecimiento a la cDra. Lizbeth Vargas Cabrera y al Dr. Jesús Manuel Muñoz Pacheco, quienes me abrieron las puertas a la inmensidad de la teoría del caos. No solo me compartieron conocimiento, sino que también sembraron en mí la pasión por indagar más allá del orden aparente. Gracias por sus ánimos constantes, su paciencia sin medida y, sobre todo, por su amistad sincera. A través de ustedes comprendí que la ciencia también se cultiva con humanidad, ustedes creyeron en mí incluso cuando yo titubeaba, gracias por exigirme cuando más lo necesitaba y por compartirme su luz con humildad y generosidad.

Le agradezco a los miembros del jurado compuesto por la M. C. Selene Edith Maya Rueda, Mtra. Alinne Michelle Sánchez Tomay y el M.C. Rodrigo Lucio Maya Ramírez, por hacer de este un mejor trabajo.

Extiendo mi gratitud a todos mis maestros y maestras que me acompañaron en este largo viaje. Cada uno, con su estilo y vocación, dejó una huella indeleble en mi formación académica y personal.

Agradezco profundamente a la Benemérita Universidad Autónoma de Puebla, mi casa de estudios, por acogerme en su programa de excelencia en la Licenciatura en Electrónica. Aquí no solo adquirí conocimientos, sino también la confianza para aspirar a más y las oportunidades para impulsar mi formación profesional.

Dedicatoria

A mis padres, por darme la vida y enseñarme con el ejemplo a ser una persona de bien. Su amor incondicional y su esfuerzo diario son el verdadero motor de mis logros.

A mi hermano, por su compañía silenciosa y constante, por estar siempre ahí.

A mi abuela, por su ternura inagotable y su apoyo incondicional en cada paso que doy.

Al Dr. Alejandro Palma Almendra por sus innumerables enseñanzas, y por sembrar en mí las buenas virtudes y el compromiso con la diligencia profesional.

Este trabajo no es solo el resultado de largas horas de estudio y esfuerzo, sino también del conocimiento compartido, de los consejos sinceros, de las palabras que alientan y de las miradas que confían. A todos los que, de una forma u otra, me dieron impulso: este logro también es suyo.

Contenido

Resumen

Esta tesis se desarrolla en la intersección entre la teoría del caos y la ciberseguridad enfocada a sistemas del internet de las cosas (IoT), estudia cómo los sistemas caóticos pueden servir como fuente de entropía para conformar generadores de números y generar secuencias numéricas de alta calidad. Los sistemas generadores de números son esenciales en aplicaciones criptográficas pues permiten la creación de claves que garantizan la confidencialidad de la información cifrada que se almacena o transmite, así como ayudar a mantener la integridad de los mensajes y autenticidad del remitente. En este contexto el microcontrolador RP2040 (RaspberryPi Pico) destaca por su idoneidad para aplicaciones en IoT e integración a redes inalámbricas de baja potencia.

En el capítulo dos se expone el marco teórico del caos, iniciando con la revisión de las formas de expresión de sistemas dinámicos lineales y no lineales, y el estudio de su estabilidad local medite la obtención de sus puntos de equilibrio y de sus valores propios. Se muestran los 8 tipos de puntos de equilibrio hiperbólicos existentes para Ecuaciones diferenciales ordinarias (EDO) de 3 variables de estado. Se enuncian las características del caos y se mencionan algunas de sus características y condiciones necesarias para su existencia. Se explica el algoritmo de integración numérica de Euler con el que se desarrollarán los estados del sistema dinámico caótico además de que ayuda a comprobar el régimen caótico del sistema mediante series de tiempo o diagramas de fase y, finalmente se resuelven los puntos de equilibrio y eigenvalores de los cinco modelos dinámicos paradigmas de estudio para determinar su tipo de estabilidad.

El capítulo tres explica las principales características y aplicaciones del cifrado simétrico y asimétrico, también se define a los generadores de números aleatorios reales y pseudoaleatorios, lo que enmarca al sistema desarrollado como un sistema generador de números pseudoaleatorios para un cifrado simétrico de flujo. Para mejorar las cualidades estadísticas de las secuencias numéricas generadas por la fuente de entropía se introducen algoritmos de post-procesamiento, resultando de particular interés la combinación de los LFSR (por sus siglas en inglés: Linear Feedback Shift Register) con los sistemas caóticos en una configuración que se denomina *scrambler*. Se detalla el tipo de sistema de cifrado a desarrollar y se explica un método para mejorar el balance de bits de una secuencia numérica antes de pasar por una etapa de refinamiento que usa un *scrambler*, también se justifica la elección de uno de los cinco sistemas caóticos para su implementación en el microcontrolador. Se explica brevemente el objetivo y funcionamiento de las evaluaciones estadísticas del NIST (National Institute of Standards and Technology), las cuales verifican en las claves de cifrado la existencia de patrones que puedan comprometer la seguridad de

los sistemas criptográficos, así mismo se explica la manera de interpretar los resultados de las pruebas estadísticas y finalmente se exponen los resultados de las evaluaciones estadísticas para las secuencias numéricas generadas por cada uno de los cinco sistemas caóticos estudiados.

En el capítulo cuatro se definen los métodos y características del hardware utilizado para programar el sistema y se explica su funcionamiento usando diagramas de flujo. Por último, el capítulo cinco se dedica a las conclusiones de la tesis.

Como prueba de concepto, se transmite una señal analógica cifrada, modulada por un potenciómetro, entre un emisor y un receptor que descifra la información y la transmite por el puerto serial para ser graficada y visualizada en pantalla. Este experimento demuestra la capacidad del sistema de cifrado simétrico de flujo para mantener la integridad de los datos y sincronización del cifrado entre ambas máquinas, evidenciando una baja latencia y robustez en un entorno dinámico heterogéneo demostrando su viabilidad para aplicarse en comunicaciones de seguridad con una baja latencia.

1 Introducción

El Internet de las Cosas (IoT) está redefiniendo la interacción entre el entorno físico y digital al conectar a Internet a miles de millones de dispositivos inteligentes con capacidades de procesamiento, detección y actuación. Este paradigma abarca una amplia gama de objetos, desde automóviles y electrodomésticos hasta *wearables* y sensores ambientales, generando grandes volúmenes de datos heterogéneos que presentan desafíos en términos de gestión y seguridad. La falta de estándares específicos para dispositivos con recursos limitados y tecnologías diversas, sumada a las vulnerabilidades inherentes de estos sistemas, crean un entorno propenso a amenazas cibernéticas [1].

La interconexión masiva y la ubicuidad de los dispositivos IoT plantean desafíos críticos en términos de seguridad y privacidad. En el modelo del IoT, los sistemas suelen organizarse en capas: la capa de percepción que incluye sensores y actuadores que interactúan con el entorno físico, la capa de transporte encargada de transmitir todas la información recabada en la capa de percepción hacia cualquier sistema de procesamiento de la información o la nube y la capa de aplicación que ofrece servicios a un usuario o a un sistema autónomo, procesando y visualizando los datos capturados en la capa de percepción para tomar decisiones o desencadenar acciones. La capa de percepción es particularmente vulnerable a amenazas cibernéticas debido a su exposición directa al entorno físico y su heterogeneidad tecnológica, lo que dificulta la implementación de soluciones de seguridad uniformes [1]. Por ejemplo, un sensor en un espacio público podría ser físicamente accesible o susceptible a diversos ataques, lo que compromete la integridad del sistema.

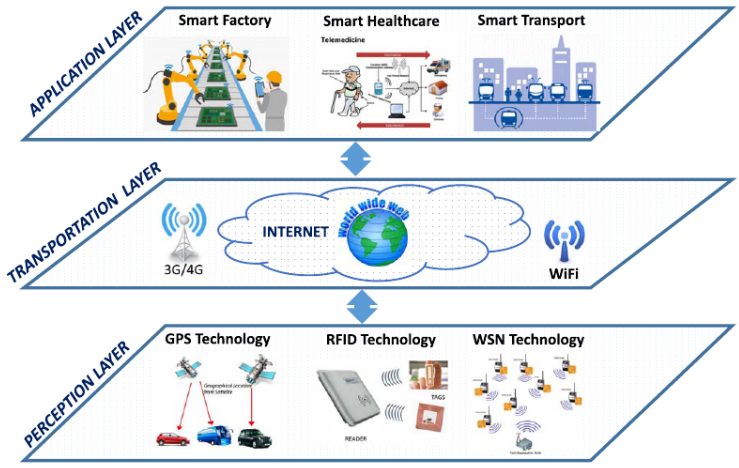


Fig. 1: Modelo de sistema IoT [1]

En este contexto, los cifrados se convierten en una herramienta esencial para garantizar la seguridad en el IoT, protegen la confidencialidad de la información;

asegurando que solo los destinatarios que cuentan con la llave de cifrado puedan leer los datos, ayudan a proteger la integridad; evitando que la información sea alterada durante la transmisión; sin estas medidas, datos sensibles como información personal o comandos críticos en sistemas industriales, podrían ser interceptados o manipulados por un tercero, erosionando la seguridad en el ecosistema [1].

La criptografía moderna depende en gran medida de la generación de números aleatorios para crear claves seguras. En este sentido, los sistemas caóticos ofrecen una solución prometedora debido a su sensibilidad a las condiciones iniciales y su comportamiento impredecible, que imita la aleatoriedad [2]. Estas propiedades del caos son ideales para diseñar generadores de números pseudoaleatorios (PRNGs); fundamentales en algoritmos de cifrado y autenticación. Sin embargo, en el IoT, los dispositivos de capacidades computacionales limitadas exigen que los algoritmos de cifrado, además de seguros, sean eficientes para no comprometer el rendimiento de los dispositivos y ayudar a reducir el gasto energético en redes con nodos autónomos de baja potencia.

Este trabajo propone implementar un sistema de cifrado simétrico de flujo con ayuda de la tarjeta Raspberry Pi Pico que contiene al microcontrolador RP2040; un dispositivo de bajo costo y consumo energético adecuado para IoT. El sistema de cifrado se probará en una red Ad Hoc creada con módulos de RF XBee3 transmitiendo de manera cifrada la información de una señal analógica. Las secuencias binarias generadas por los sistemas caóticos se evaluarán mediante el estándar de la NIST, lo que ayuda a determinar la calidad de los sistemas generadores así como el buen diseño de los mismos y, consecuentemente evalúan la calidad de las secuencias generadas, ayudando a validar su aplicación en sistemas criptográficos. Las pruebas de concepto para la recepción de información confirmarán las capacidades de la comunicación cifrada simétrica para mantener la sincronía en las claves de cifrado entre ambas máquinas así como la capacidad de operar con una baja latencia manteniendo la integridad y confidencialidad de la información.

1.1 Justificación

En el paradigma del Internet de las Cosas, la capa de percepción se destaca como un punto crítico de vulnerabilidad debido a la interacción directa de sensores y actuadores con el entorno físico, así como a la heterogeneidad tecnológica de los dispositivos que la componen. La disposición pública de los nodos demanda soluciones de seguridad robustas y computacionalmente eficientes capaces de garantizar la confidencialidad, integridad y autenticidad de los datos transmitidos. Para abordar esta necesidad, el presente trabajo propone un sistema de cifrado simétrico de flujo que emplea las propiedades de los

sistemas caóticos como base para la protección de la información en entornos IoT.

En este contexto, los sistemas caóticos se posicionan como fuentes de entropía con características útiles para aplicaciones criptográficas; por su sensibilidad extrema a las condiciones iniciales y su comportamiento impredecible a largo plazo. Estas propiedades facilitan la generación de secuencias pseudoaleatorias de alta calidad, esenciales para la creación de claves seguras.

Al integrar los generadores de números pseudoaleatorios basados en caos en un esquema de cifrado simétrico, el sistema logra un equilibrio entre seguridad y eficiencia, adaptándose a las limitaciones computacionales de los dispositivos IoT. La viabilidad práctica del sistema se demuestra mediante su implementación en el microcontrolador RP2040, un dispositivo caracterizado por su bajo costo y consumo energético. Los resultados obtenidos en pruebas de concepto confirman la capacidad del RP2040 y el módulo de RF XBee3 para operar con baja latencia, preservando la sincronización de las claves simétricas, posicionando al sistema como una solución viable para aplicaciones reales en entornos tecnológicos emergentes, como el SIoT (Social Internet of Things) o la industria 4.0. Al ofrecer una solución ligera, segura y adaptable, el sistema desarrollado se considera un buen modelo para proteger las comunicaciones en redes de baja potencia, con resultados que ayudan a fortalecer la confianza en un ecosistema donde la seguridad es un pilar fundamental para su expansión y adopción masiva.

Además de su funcionalidad inmediata, en esta tesis se prueba la hipótesis de que los sistemas caóticos muestran un sesgo característico en la función de distribución de cada una de sus variables de estado lo cual tiene un impacto prospectivo destacable al abrir camino para el mejoramiento de los generadores de números y la ciberseguridad en el IoT.

1.2 Objetivos

General: Implementar un generador de números aleatorios usando sistemas caóticos como fuente de entropía para cifrar información en un procesador ARM y transmitirla en una red Ad Hoc.

Específicos:

1) Revisar el estado del arte de cinco sistemas caóticos: Lorenz, Chua y tres hidden attractors.
2) Analizar la dinámica no lineal de los cinco sistemas caóticos mediante puntos de equilibrio, estabilidad, series de tiempo y diagramas de fase.
3) Diseñar el generador de números aleatorios usando los sistemas caóticos.
4) Evaluar la aleatoriedad de las señales caóticas por medio del estándar del NIST.

5) Implementar el sistema de cifrado usando las señales caóticas en una Raspberry Pi Pico.
6) Configurar una red Ad Hoc para el intercambio de información cifrada entre máquinas.

2 Marco Teórico del Caos

2.1 Introducción a los sistemas dinámicos

Se debe considerar que el caos es parte de la dinámica, que constituye un tema muy amplio, esta generalmente se utiliza para la descripción y estudio de sistemas que cambian y evolucionan con el tiempo. En la actualidad la dinámica se utiliza en campos como el cálculo diferencial, mecánica clásica, cinética química, biología de poblaciones y fue concebida a mediados del siglo XVII cuando Newton inventó las ecuaciones diferenciales y las combinó con las leyes de gravitación universal para explicar las leyes del movimiento planetario de Kepler.

Un gran avance en la teoría dinámica se presentó con el trabajo de Poincaré a finales del siglo XIX, él desarrolló un enfoque geométrico que es de amplio uso en el temas modernos de dinámica. Poincaré fue la primera persona en vislumbrar la posibilidad del caos, en el que un sistema determinista exhibe un comportamiento aperiódico que depende sensiblemente de las condiciones iniciales, lo que hace imposible las predicciones a largo plazo.

La invención de la computadora de alta velocidad en la década de 1950 extendió otro punto de inflexión para el estudio de sistemas dinámicos, las computadoras permitieron experimentar con las ecuaciones de una manera que antes era imposible, tales experimentos llevaron al descubrimiento de Edward Norton Lorenz en 1963 sobre el movimiento caótico en algo denominado *atractor extraño* (término acuñado por Ruelle y Takens 1971). Lorenz había estudiado un modelo simplificado de rollos de convección en la atmósfera para comprender la notoria impredecibilidad del clima, descubrió que las soluciones a sus ecuaciones nunca se estabilizaban en un equilibrio o en un estado periódico; en cambio, continuaban oscilando de manera irregular y aperiódica. Lorenz también mostró que había estructura en el caos: cuando se graficaban en tres dimensiones, las soluciones a sus ecuaciones caían en un conjunto de puntos en forma de mariposa, argumentó que este conjunto tenía que ser "un complejo infinito de superficies"; lo que hoy consideramos un ejemplo de fractal [3].

Con las nuevas ideas sobre el caos, a partir de 1970 los estudios en la teoría del caos cobraron interés teniendo diversos desarrollos como el inicio de la turbulencia en fluidos, existencia del caos en biología de poblaciones, el descubrimiento de leyes universales que gobiernan la transición de un comportamiento regular a uno caótico, oscilaciones biológicas especialmente en los ciclos circadianos y el ritmo cardíaco y diversos experimentos en dinámica de fluidos, reacciones químicas, circuitos electrónicos, oscilaciones mecánicas y semiconductores [3].

2.2 Modelado de sistemas dinámicos

Existen dos formas de modelar sistemas dinámicos: con ecuaciones diferenciales y con mapas iterados también llamados ecuaciones en diferencias. Mientras las ecuaciones diferenciales se utilizan para describir la evolución de un sistema en un tiempo continuo las ecuaciones en diferencias son útiles para problemas donde el tiempo es discreto. A su vez, las ecuaciones diferenciales se dividen en dos grupos; las ecuaciones diferenciales ordinarias (ODE) y las ecuaciones diferenciales parciales (PDE). Las ODEs involucran derivadas donde solo hay una variable independiente que, por lo general, es el tiempo y las PDEs se utilizan para modelar sistemas donde hay más de una variable independiente, como por ejemplo tiempo y espacio.

2.2.1 Sistema de primer orden de EDOs lineales: Para este trabajo únicamente son de interés las ODEs, consideremos de ejemplo la ecuación para un oscilador armónico amortiguado:

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0$$

Una forma general de representar un modelo dinámico de ecuaciones diferenciales ordinarias es con la siguiente estructura denominada sistema de primer orden autónomo donde la variable independiente t no se representa de forma explícita en el miembro de la derecha de cada ecuación diferencial [4]:

$$\begin{aligned}
\frac{dx_1}{dt} &= g_1(x_1, x_2, \ldots, x_n) \\
\frac{dx_2}{dt} &= g_2(x_1, x_2, \ldots, x_n) \\
\vdots &\qquad\qquad \vdots \\
\frac{dx_n}{dt} &= g_n(x_1, x_2, \ldots, x_n)
\end{aligned}$$

Esto convierte un sistema de ecuaciones diferenciales de segundo orden o superior en un sistema de primer orden, es decir donde el máximo orden de derivada es 1. Esta estructura permite la integración numérica y la representación en espacio de fases de una forma más directa que con su forma en ecuaciones diferenciales. Donde las funciones $g_1, ..., g_n$ son las reglas de evolución del sistema, así las variables del vector de estados $x_1, x_2 ..., x_n$ representan la posición, la velocidad y la aceleración.

Para el caso del oscilador armónico amortiguado se tiene $\frac{dx_i}{dt} \equiv \dot{x}_i$ donde un punto sobre x_i denota una diferenciación con respecto a la variable independiente t, de este modo se pueden introducir nuevas variables para manipular la ecuación diferencial ordinaria de segundo orden y convertirla en un sistema de

ecuaciones diferenciales ordinarias de primer orden:

$$x_1 = x, \qquad x_2 = \dot{x} \equiv \frac{dx}{dt}, \qquad x_3 = \ddot{x} \equiv \frac{d^2x}{dt^2} \quad \Longrightarrow \quad \dot{x_1} = x_2, \qquad \dot{x_2} = x_3$$

Donde la variable de estado $\dot{x_1}$ corresponde a la velocidad y, despejando para $\dot{x_2}$ correspondiente a la aceleración, el sistema puede ser reescrito como:

$$\dot{x_1} = x_2$$
$$\dot{x_2} = -\frac{b}{m}x_2 - \frac{k}{m}x_1$$

Este modelo que describe el comportamiento del oscilador armónico amortiguado se dice que es un sistema dinámico lineal porque todas las variables de estado x_i del lado derecho solo aparecen a la primera potencia. Si el modelo contara con términos no lineales como por ejemplo, $x_1x_2, (x_1)^3, cos(x_2)$ se trataría de un sistema dinámico no lineal. Las no linealidades producen comportamientos complejos en los sistemas dinámicos pudiendo resultar, entre otras complejidades, en múltiples puntos de equilibrio, bifurcaciones, o fenómenos como el caos.

2.2.2 Sistema de primer orden de EDOs no lineales: Un ejemplo de sistema dinámico no lineal es el circuito de Chua, propuesto por Leon Ong Chua en 1983, es aceptado como un paradigma para el estudio de características importantes de sistemas no lineales y se utiliza para múltiples propósitos como por ejemplo para generar números aleatorios, en sistemas de comunicación y en criptografía basada en caos [5]. A continuación se muestra el diagrama del circuito y las ecuaciones que modelan su comportamiento analógico.

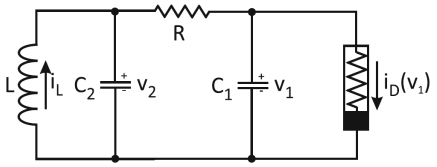


Fig. 2: Circuito de Chua [5]

El circuito de Chua es un sistema de tercer orden debido a que cuenta con 3 elementos activos que almacenan la energía de forma dinámica, estos son los capacitores C_1, C_2 y el inductor L, cada elemento almacenador de energía contribuye con una variable de estado independiente cuya evolución temporal describe la dinámica del sistema.

Con la ley de corrientes de Kirchhoff para el nodo donde se conecta C_1, R y el diodo de Chua se tiene que la corriente a través de C_1 es $C_1\frac{dV_{C_1}}{dt}$, la corriente a través de la resistencia R es $\frac{-V_{C_1}+V_{C_2}}{R}$ y la corriente que pasa por el diodo de Chua es $i_D(V_{C_1})$.

De manera análoga para el nodo donde se conecta C_2, R y L se aprecia que

la corriente del capacitor C_2 es $C_2\frac{dV_{C_2}}{dt}$, la corriente que atraviesa la resistencia R es $\frac{V_{C_1}-V_{C_2}}{R}$ y la corriente por el inductor es i_L.

Con la ley de voltajes de Kirchhoff donde la suma de voltajes de la malla es 0 se tiene la malla que incluye al inductor L y el capacitor C_2, donde el voltaje en el inductor es $L\frac{dI_L}{dt}$ y el voltaje en el capacitor C_2 es V_{C_2}.

De este modo se obtiene el siguiente conjunto de ecuaciones diferenciales ordinarias de primer orden que modelan el comportamiento analógico del circuito de Chua:

$$C_1\frac{dV_{C_1}}{dt} = \frac{-V_{C_1}+V_{C_2}}{R} - i_D(V_{C_1}) \tag{1}$$

$$C_2\frac{dV_{C_2}}{dt} = \frac{V_{C_1}-V_{C_2}}{R} + i_L \tag{2}$$

$$L\frac{di_L}{dt} + V_{C_2} = 0 \tag{3}$$

Estas ecuaciones, al ser linealmente independientes y describir todo el comportamiento del circuito, pueden combinarse algebraicamente en una única ecuación diferencial de tercer orden [4], no obstante, la formulación del sistema en términos de primer orden es preferible para los objetivos de análisis dinámico y de integración numérica.

El análisis del circuto de Chua se optimiza mediante el reescalamiento de sus variables físicas $(V_{C_1}, V_{C_2}, I_L, C_1, C_2, R, L)$ en grupos adimensionales, lo cual elimina las unidades y reduce la dinámica del sistema a la descripción de los parámetros α, β [5]. Se introduce un nuevo tiempo adimensional τ consolidando un modelo equivalente permitiendo un estudio general del sistema en un marco abstracto sin cambiar las dinámicas cualitativas del circuito de Chua:

$$x = V_{C_1}, \quad y = V_{C_2}, \quad z = Ri_L, \quad \alpha = \frac{C_2}{C_1}, \quad \beta = \frac{R^2C_2}{L}, \quad \tau = \frac{t}{RC_2}$$

Sustituyendo para la ecuación 1 se obtiene:

$$C_1\frac{1}{RC_2}\frac{dx}{d\tau} = -\frac{x}{R} + \frac{y}{R} - i_D(x)$$

$$\frac{dx}{d\tau} = -\frac{\cancel{R}C_2}{C_1}\frac{x}{\cancel{R}} + \frac{\cancel{R}C_2}{C_1}\frac{y}{\cancel{R}} - \frac{C_2}{C_1}Ri_D(x)$$

El diodo de Chua es una reducción conceptual de un circuito que induce términos no lineales, este puede tener distintas configuraciones y ser modificado de muchas formas, en el sistema original de Leon O. Chua el diodo es un circuito que introduce una función piecewise linear, sin embargo, debido a que su implementación es compleja y requiere una considerable cantidad de componentes, algunos trabajos sugieren circuitos más veloces y con menos componentes que inducen funciones no lineales cúbicas suaves como la sugerida en [6]: $i_D(x) = aV_{C_1} + cV_{C_1}^3$ donde a, c son parámetros del diodo Chua e $i_D(x)$ es la

corriente del diodo Chua dependiente del voltaje en C_1, de este modo:

$$u(x) = Ri_D(x) = (Ra)V_{C_1} + (Rc)V_{C_1}^3 = a_1 x + a_3 x^3$$

Entonces a_1, a_3 se hacen parámetros adimensionales y la ecuación de primer orden para la variable de estado x se reduce a:

$$\frac{dx}{d\tau} = \alpha\left(-x + y - u(x)\right)$$

Sustituyendo para la ecuación 2 se obtiene:

$$\cancel{C_2}\frac{1}{R\cancel{C_2}}\frac{dy}{d\tau} = \frac{x}{R} - \frac{y}{R} + i_L$$

$$\frac{dy}{d\tau} = \cancel{R}\frac{x}{\cancel{R}} - \cancel{R}\frac{y}{\cancel{R}} + Ri_L = x - y + z$$

Sustituyendo para la ecuación 3:

$$L\frac{1}{RC_2}\frac{di_L}{d\tau} = -y$$

$$\frac{di_L}{d\tau}R = -\frac{RC_2}{L}yR$$

$$\frac{dz}{d\tau} = -\beta y$$

Así se obtiene el siguiente sistema de ecuaciones diferenciales de primer orden que describen el comportamiento del circuito de Chua, donde $u(x) = a_1 x + a_3 x^3$ es una función no lineal cúbica suave:

$$\frac{dx}{d\tau} = \alpha\left(-x + y - u(x)\right)$$

$$\frac{dy}{d\tau} = x - y + z$$

$$\frac{dz}{d\tau} = -\beta y$$

Las ecuaciones diferenciales no lineales de orden superior desafían virtualmente la solución de sus estados con métodos analíticos, aunque siempre es posible analizar de modo cualitativo y numérico una ED no lineal.

2.3 Puntos de equilibrio

Un punto de equilibrio, también denominado punto crítico, es un estado del sistema donde la evolución de todas las variables de estado permanece invariable. En otras palabras, los puntos de equilibrio se presentan cuando la razón de cambio de todas las variables de estado respecto al tiempo es cero. Considérese un sistema dinámico autónomo no lineal con tres variables de

estado (x,y,z), descrito por las ecuaciones:

$$\dot{x} = e(x,y,z),$$
$$\dot{y} = f(x,y,z),$$
$$\dot{z} = g(x,y,z).$$

Un punto de equilibrio (x^*,y^*,z^*) satisface:

$$\dot{x} = e(x^*,y^*,z^*) = 0, \quad \dot{y} = f(x^*,y^*,z^*) = 0, \quad \dot{z} = g(x^*,y^*,z^*) = 0.$$

Esto significa que, si el sistema comienza o alcanza el estado de equilibrio (x^*,y^*,z^*), se encontrará en un estado donde su razón de cambio es nula y permanecerá allí indefinidamente.

2.4 Linearización mediante series de Taylor

Dado que el análisis de estabilidad de un sistema no lineal rara vez tienen solución analítica, es útil estudiar el comportamiento del sistema cerca de un punto de equilibrio determinando su estabilidad local, donde es posible emplear una técnica de linealización que aproxima el sistema no lineal por un sistema lineal en las proximidades del estado de equilibrio (x^*,y^*,z^*). Esta aproximación se basa en la expansión en serie de Taylor, que permite expresar una función diferenciable como una suma de términos polinómicos alrededor de un punto dado.

Si se suponen estados (x,y,z) del sistema no lineal muy cercanos al estado de equilibrio (x^*,y^*,z^*), se pueden definir pequeñas perturbaciones desde el punto de equilibrio:

$$u = x - x^*, \qquad v = y - y^*, \qquad w = z - z^*$$

La dinámica de estas perturbaciones se obtiene al expandir las funciones en series de Taylor, por ejemplo, para $\dot{u} = e(x^*+u, y^*+v, z^*+w)$:

$$\dot{u} = e(x^*,y^*,z^*) + u\frac{\partial e}{\partial x} + v\frac{\partial e}{\partial y} + w\frac{\partial e}{\partial z} + \mathscr{O}(u^2, v^2, w^2, uv, uw, vw, \ldots),$$

Donde las derivadas parciales $\frac{\partial e}{\partial x}, \frac{\partial e}{\partial y}, \frac{\partial e}{\partial z}$ son términos lineales al ser evaluadas en el punto de equilibrio (x^*,y^*,z^*) estas denotan valores numéricos, no funciones. Y $\mathscr{O}(u^2, v^2, w^2, uv, uw, vw, \ldots)$ denota los términos cuádráticos y de oden superior de la expansión de la serie de Taylor.

Debido a que $e(x^*,y^*,z^*) = 0$, la serie se simplifica a términos lineales más términos de orden superior. De este modo, las ecuaciones para las tres

perturbaciones cercanas al equilibrio son:

$$\dot{u} = u\frac{\partial e}{\partial x} + v\frac{\partial e}{\partial y} + w\frac{\partial e}{\partial z} + \mathcal{O}(u^2, v^2, w^2, uv, uw, vw, \ldots),$$
$$\dot{v} = u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + w\frac{\partial f}{\partial z} + \mathcal{O}(u^2, v^2, w^2, uv, uw, vw, \ldots),$$
$$\dot{w} = u\frac{\partial g}{\partial x} + v\frac{\partial g}{\partial y} + w\frac{\partial g}{\partial z} + \mathcal{O}(u^2, v^2, w^2, uv, uw, vw, \ldots).$$

Estas ecuaciones que describen la evolución de las perturbaciones (u, v, w) pueden escribirse matricialmente como:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{\partial e}{\partial x} & \frac{\partial e}{\partial y} & \frac{\partial e}{\partial z} \\ \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \mathcal{O}(u^2, v^2, w^2, uv, \ldots) \\ \mathcal{O}(u^2, v^2, w^2, uv, \ldots) \\ \mathcal{O}(u^2, v^2, w^2, uv, \ldots) \end{bmatrix}$$

Dado que los términos cuadráticos y de orden superior de la expansión de la serie de Taylor son despreciables para perturbaciones (u, v, w) pequeñas, se obtiene un sistema linealizado con $\dot{u} = \dot{x}, \dot{v} = \dot{y}, \dot{w} = \dot{z}$ mientras x^*, y^*, z^* sean constantes, se obtiene:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{\partial e}{\partial x} & \frac{\partial e}{\partial y} & \frac{\partial e}{\partial z} \\ \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Así la matriz Jacobiana J en el estado de equilibrio (x^*, y^*, z^*) se define como:

$$J = \begin{bmatrix} \frac{\partial e}{\partial x} & \frac{\partial e}{\partial y} & \frac{\partial e}{\partial z} \\ \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} & \frac{\partial g}{\partial z} \end{bmatrix}_{(x^*, y^*, z^*)}$$

De este modo se dice que el Jacobiano linealiza al sistema al evaluar las derivadas parciales de las funciones para el punto de interés, en este caso un punto de equilibrio (x^*, y^*, z^*). Cada punto de equilibrio tiene su propio Jacobiano y es fundamental para estudiar la estabilidad local del sistema, ya que sus valores propios determinan las cualidades dinámicas en las proximidades del equilibrio [3].

2.5 Tipos de puntos de equilibrio hiperbólicos para EDO de 3 dimensiones

8 casos de estabilidad para sistemas dinámicos de 3 dimensiones.

Tipo	Num. Índice	Descripción	Eigenvalores λ
Tipo 1	Índice 0 Nodo	Este sistema tiene un solo punto de equilibrio con 3 eigenvalores reales, todos negativos. Es un nodo estable con índice 0.	$\lambda_{1,2,3} = -\alpha$
Tipo 2	Índice 1 Punto silla	Este sistema tiene tres eigenvalores reales, con dos negativos y uno positivo.	$\lambda_{1,2} = -\alpha$, $\lambda_3 = \alpha$
Tipo 3	Índice 2 Punto silla	Este sistema tiene tres eigenvalores, con dos positivos y uno negativo.	$\lambda_1 = -\alpha$, $\lambda_{2,3} = \alpha$
Tipo 4	Índice 3 Repulsor	Este sistema tiene tres eigenvalores reales, todos positivos, por lo tanto, es un nodo inestable, también llamado repulsor.	$\lambda_{1,2,3} = \alpha$
Tipo 5	Índice 0 Nodo espiral	Este sistema tiene un eigenvalor real negativo, y un par conjugado complejo con una parte real negativa.	$\lambda_1 = -\alpha$, $\lambda_{2,3} = -\alpha \pm j\beta$
Tipo 6	Índice 1 Silla espiral	Este sistema tiene un eigenvalor real positivo, y un par conjugado complejo con una parte real negativa.	$\lambda_1 = \alpha$, $\lambda_{2,3} = -\alpha \pm j\beta$
Tipo 7	Índice 2 Silla espiral	Este sistema tiene un eigenvalor real negativo, y un par conjugado complejo con una parte real positiva.	$\lambda_1 = -\alpha$, $\lambda_{2,3} = \alpha \pm j\beta$
Tipo 8	Índice 3 Rep. espiral	Este sistema tiene un eigenvalor real positivo, y un par conjugado complejo con una parte real positiva.	$\lambda_1 = \alpha$, $\lambda_{2,3} = \alpha \pm j\beta$

2.6 Características del caos

No existe un consenso generalizado sobre la definición de caos, sin embargo se pueden enmarcar tres principales características que lo definen como un comportamiento aperiódico a largo plazo en un sistema determinista que exhibe una dependencia sensible a las condiciones iniciales [3].

El comportamiento aperiódico del caos significa que los estados del sistema no se estabilizan en puntos fijos u órbitas periódicas además de que sus estados sean irrepetibles. De no ser por la limitada resolución para calcular los estados de un sistema, los conjuntos de distintas soluciones serían infinitos y nunca se repetirían. Esta característica del caos es vital para generar números aleatorios.

El caos emerge en sistemas dinámicos no lineales deterministas, como el circuito de Chua, donde se controlan todos los parámetros del sistema e idealmente no tiene entradas ni parámetros aleatorios o ruidosos. Las relaciones no lineales son necesarias en el sistema para que surja el caos además de que en sistemas descritos por ecuaciones diferenciales ordinarias de orden entero, el caos requiere al menos tres variables de estado debido al teorema de Poincaré-Bendixson [3], que excluye dinámicas caóticas en espacios de fases bidimensionales. Por otra parte, el cálculo fraccionario, al emplear derivadas de orden no entero, permite la aparición de comportamientos complejos y aperiódicos en sistemas no lineales con solo dos variables de estado; la caracterización precisa del caos usando cálculo fraccionario permanece como un tema de investigación activa [7], [8].

El caos es sensible a las condiciones iniciales, por ejemplo en un comportamiento caótico sin estado transitorio se puede apreciar que condiciones iniciales cercanas divergen de manera exponencial. Lo cuál resulta práctico

para generar números aleatorios con base en una semilla compuesta por los parámetros y condiciones iniciales de un sistema determinista.

La inestabilidad de los puntos de equilibrio, evidenciada por autovalores con parte real positiva, es una condición necesaria pero no suficiente para la emergencia de dinámicas caóticas. Al superar una bifurcación donde los puntos de equilibrio se hacen inestables y, careciendo de estabilidad marginal (sin autovalores puramente imaginarios), un sistema como el de Lorenz o Chua transita hacia un régimen donde el caos no transitorio puede emerger. De forma global las trayectorias en el espacio de fases son repelidas por objetos inestables y confinadas en un conjunto acotado describiendo un atractor caótico autosustentado; como una esfera en una máquina de pinball, en el sentido de que las dinámicas caóticas surgen y se mantienen sin requerir excitación externa. Sin embargo, estas condiciones no garantizan la persistencia del caos; por ejemplo, en el sistema de Lorenz, al variar el parámetro ρ, se observan ventanas de comportamiento periódico a pesar de la inestabilidad de sus puntos de equilibrio, e inclusive es posible apreciar la existencia de un ciclo límite global cuando $\rho > 313$ ($\sigma = 10, \beta = 8/3$) mientras los puntos de equilibrio permanecen inestables [3].

2.7 Método de Euler

Aún cuando se pueda demostrar que la solución de una ecuación diferencial exista, no siempre es posible expresarla en forma explícita o implícita. Para desarrollar los estados de un sistema, en muchos casos tenemos que conformarnos con una aproximación de la solución mediante integración numérica [4]. Las computadoras permiten aproximar soluciones a problemas intratables de forma analítica además de poder visualizar esas soluciones. Existen diversos métodos de integración numérica para resolver ecuaciones diferenciales, algunos con mejores aproximaciones que otros dado que estos algoritmos introducen errores de cálculo que desvían los estados de la solución real.

Se elige el método de Euler para desarrollar las variables de estado de un sistema de ecuaciones de primer orden debido a que presenta soluciones lo suficientemente aproximadas para los objetivos de esta tesis además de ser el esquema de integración numérica más simple que, al programarlo, resulta en un algoritmo ligero lo que se traduce en un menor uso de recursos computacionales para su aplicación.

Se considera el modelo del circuito de Chua con tres variables de estado x, y, z con condiciones iniciales $x(\tau_0) = x_0, y(\tau_0) = y_0, z(\tau_0) = z_0$ el método de Euler aproxima numéricamente las soluciones $x(\tau), y(\tau), z(\tau)$ en tiempos (τ) discretos,

el esquema iterativo tiene la forma:

$$x_{n+1} = x_n + he(x_n, y_n, z_n)$$
$$y_{n+1} = y_n + hf(x_n, y_n, z_n)$$
$$z_{n+1} = z_n + hg(x_n, y_n, z_n)$$

Se tiene una relación aproximada entre la integración numérica y las soluciones reales $x_n \approx x(\tau_n), y_n \approx y(\tau_n), z_n \approx z(\tau_n)$, donde las funciones e, f, g evalúan las tasas de cambio de los estados x_n, y_n, z_n y donde $h > 0$ es una constante y es el tamaño de paso entre x_n y $x_n + 1$ [4]. Este método linealiza el sistema en cada intervalo $[\tau_n, \tau_{n+1}]$ con un error dependiente del valor de h, mejorando su precisión al disminuir el valor de h [3].

Con el método de integración numérica de Euler se obtienen vectores solución para cada variable de estado, para apreciar fácilmente la transición dinámica del sistema es conveniente representar las soluciones de forma gráfica, principalmente de dos maneras; graficando los valores de cada variable de estado (x, y, z) respecto a su número de iteración n y graficando los valores de los estados en el espacio de fases también llamado espacio de estados asignando una dimensión del plano para cada variable de estado. Para el sistema de Chua con un tamaño de paso $h = 0.036$ se obtienen las siguientes gráficas para las primeras 1500 soluciones:

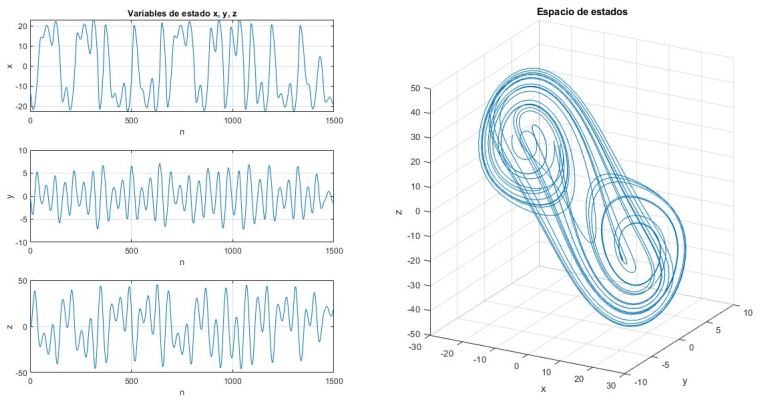


Fig. 3: Circuito de Chua con h=0.036, parámetros a=11; b=20; a1=-1.3; a3=0.001, y valores iniciales x(0) = -12.4844, y(0) = -0.04257, z(0) = -0.29804;

En la práctica, para el desarrollo de sistemas dinámicos no lineales, el tamaño de paso h debe ser lo suficientemente pequeño para que los conjuntos solución x_n, y_n, z_n sean valores que no introduzcan demasiado error como para que las soluciones diverjan de la dinámicas cualitativas del sistema, es decir que las soluciones se mantengan en la región del atractor caótico y no diverjan hacia el infinito como en el siguiente caso para el sistema de Chua con un tamaño de paso $h = 0.046$ que muestra las primeras 500 soluciones:

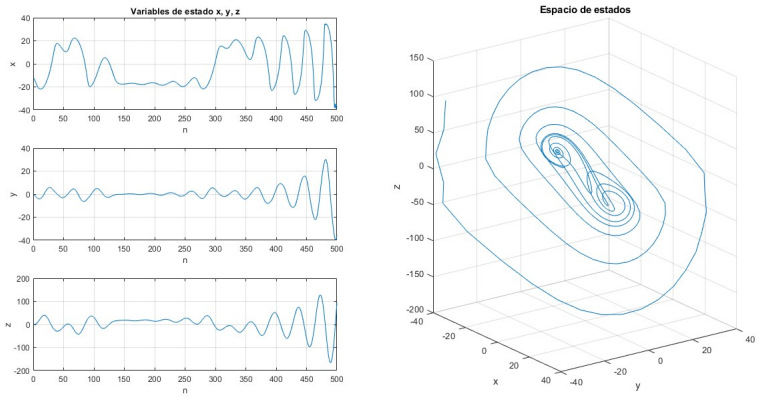


Fig. 4: Circuito de Chua con h=0.046, parámetros a=11; b=20; a1=-1.3; a3=0.001, y valores iniciales x(0) = -12.4844, y(0) = -0.04257, z(0) = -0.29804;

2.8 Sistemas caóticos paradigmas de estudio

2.8.1 Sistema de Lorenz:

$$\frac{dx}{dt} = \sigma(y-x) \tag{4}$$

$$\frac{dy}{dt} = \rho x - y - xz \tag{5}$$

$$\frac{dz}{dt} = xy - \beta z \tag{6}$$

Para hallar los puntos de equilibrio, se igualan las ecuaciones (4), (5) y (6) a cero y se despejan las variables de estado. De la ecuación (4) se obtiene:

$$\sigma y - \sigma x = 0 \implies \sigma y = \sigma x \implies y = x$$

Sustituyendo $y = x$ en la ecuación (6) se obtiene:

$$x(x) - \beta z = 0 \implies z = \frac{-x^2}{-\beta} = \frac{x^2}{\beta}$$

Sustituyendo $y = x$ junto con $z = x^2/\beta$ en la ecuación (5):

$$\rho x - (x) - x\frac{x^2}{\beta} = 0 \implies x\left(\rho - 1 - \frac{x^2}{\beta}\right) = 0$$

Con esto último, si se hace $x = 0$, se obtiene $y = 0$, de forma consecuente $z = 0$; obteniendo el punto de equilibrio trivial $(0,0,0)$. Continuando con el despeje para el miembro entre paréntesis:

$$\rho - 1 - \frac{x^2}{\beta} = 0 \implies x^2 = -\beta(1-\rho) \implies x = \pm\sqrt{\beta(\rho-1)} = y$$

Sustituyendo el resultado de x en $z = x^2/\beta$ se obtiene:

$$z = \frac{x^2}{\beta} = \frac{\left(\sqrt{\beta(\rho-1)}\right)^2}{\beta} = \frac{\beta(\rho-1)}{\beta} = \rho - 1$$

Por lo tanto los puntos de equilibrio del sistema de Lorenz son:

$$(0,0,0)$$
$$(\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho-1)$$
$$(-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho-1)$$

Estabilidad del punto de equilibrio (0,0,0)

Para determinar la estabilidad del punto de equilibrio, se calcula la matriz Jacobiana J del sistema evaluando las parciales de cada función para cada variable de estado, lo cual se dice que linealiza al sistema en el punto de equilibrio de estudio:

$$J = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - z & -1 & -x \\ y & x & -\beta \end{bmatrix}$$

Para el caso del sistema en estado de reposo $(0,0,0)$ los eigenvalores se estiman resolviendo para λ de $det(J - \lambda I) = 0$

$$J - \lambda I = \begin{bmatrix} -\sigma - \lambda & \sigma & 0 \\ \rho - z & -1 - \lambda & 0 \\ 0 & 0 & -\beta - \lambda \end{bmatrix}$$

$$\begin{aligned} det(J - \lambda I) &= (-\sigma - \lambda)(-1 - \lambda)(-\beta - \lambda) - (-\beta - \lambda)\sigma\rho \\ &= ((-\sigma - \lambda)(-1 - \lambda) - \sigma\rho)(-\beta - \lambda) \\ &(-\beta - \lambda) = 0 \implies \lambda_1 = -\beta \\ &(-\sigma - \lambda)(-1 - \lambda) - \sigma\rho = 0 \\ &\sigma + \lambda + \sigma\lambda + \lambda^2 - \sigma\rho = 0 \\ &\lambda^2 + \lambda(\sigma + 1) + \sigma - \sigma\rho = 0 \\ &a = 1 \quad , \quad b = \sigma + 1 \quad , \quad c = \sigma - \sigma\rho \end{aligned}$$

$$\lambda_{2,3} = \frac{(-\sigma - 1) \pm \sqrt{(\sigma+1)^2 - 4(1)(\sigma - \sigma\rho)}}{2(1)}$$

Para el conjunto $\sigma = 10, \rho = 28, \beta = \frac{8}{3}$ los autovalores pertenecen a $\mathbb{R}$, son diferentes y tienen signos opuestos, se dice que el punto de equilibrio $(0,0,0)$ es un punto de silla.

$$\lambda_1 = -\frac{8}{3} \quad , \quad \lambda_2 = 11.82 \quad , \quad \lambda_3 = -22.82$$

Para el caso del punto de equilibrio $(\sqrt{\beta(\rho-1)}, \sqrt{\beta(\rho-1)}, \rho-1)$ se tiene:

$$J-\lambda I = \begin{bmatrix} -\sigma-\lambda & \sigma & 0 \\ 1 & -1-\lambda & -\sqrt{\beta(\rho-1)} \\ \sqrt{\beta(\rho-1)} & \sqrt{\beta(\rho-1)} & -\beta-\lambda \end{bmatrix}$$

El polinomio característico de $det(J-\lambda I)$ es largo lo que complica la determinación analítica de los eigenvalores, la misma complejidad se presenta para los autovalores del punto de equilibrio simétrico $(-\sqrt{\beta(\rho-1)}, -\sqrt{\beta(\rho-1)}, \rho-1)$. Sin embargo, la documentación [3] indica que, estos puntos simétricos denominados como C^+ y C^-, son estables para un rango del parámetro ρ definido como.

$$1 < \rho < \rho_H = \frac{\sigma(\sigma+\beta+3)}{\sigma-\beta-1}$$

Donde ρ_H es la bifurcación de Hopf que define un límite donde el punto evaluado deja de ser linealmente estable. Esto se comprueba de manera numérica

ρ	λ_1	λ_2	λ_3
0.9999	-2.6	-11	$+1.8e-8$
1.00001	-2.6	-11	$-1.8e-6$
24.73684210	-13.6	$-1.6e^{-10}+9.6i$	$-1.6e^{-10}-9.6i$
24.73684212	-13.6	$+4.5e^{-10}+9.6i$	$+4.5e^{-10}-9.6i$

Se aprecia que en el rango definido para ρ todos los eigenvalores tienen parte real negativa lo cual indica que el punto de equilibrio es linealmente estable en ese rango. Cuando $\rho > \rho_H$ los puntos de equilibrio C^+ y C^- pierden su estabilidad lineal y el sistema puede exhibir comportamiento caótico.

Se ejecuta el siguiente código en Matlab para estimar los eigenvalores de los tres puntos de equilibrio con los parámetros estándar del sistema de Lorenz:

```
clearvars; s = 10; r = 28; b = 8 / 3;
v = [0, 0, 0;
     sqrt(b * (r - 1)),  sqrt(b * (r - 1)), r - 1;
    -sqrt(b * (r - 1)), -sqrt(b * (r - 1)), r - 1];
for i = 1:3
    x = v(i, 1); y = v(i, 2); z = v(i, 3);
    J = [-s, s, 0; r - z, -1, -x; y, x, -b];
    disp('Puntos de equilibrio:');
    fprintf('x = %f, y = %f, z = %f\n', x, y, z);
    Eigenvalores = eig(J)
end
```

El resultado de la ejecución brinda el valor de los puntos de equilibrio y los eigenvalores correspondientes:

```
Puntos de equilibrio:
x = 0.000000, y = 0.000000, z = 0.000000
Eigenvalores =
  -22.8277
   11.8277
   -2.6667
```

```
Puntos de equilibrio:
x = 8.485281, y = 8.485281, z = 27.000000
Eigenvalores =
 -13.8546 + 0.0000i
   0.0940 +10.1945i
   0.0940 -10.1945i

Puntos de equilibrio:
x = -8.485281, y = -8.485281, z = 27.000000
Eigenvalores =
 -13.8546 + 0.0000i
   0.0940 +10.1945i
   0.0940 -10.1945i
```

El tipo de estabilidad del punto de equilibrio $(0,0,0)$ es tipo 2 índice 1, es un punto de silla. El tipo de estabilidad de los puntos de equilibrio simétricos C^+ y C^- es tipo 7 índice 2, ambos son puntos de silla espiral. El índice indica el número de eigenvalores con parte real positiva. Se considera que el sistema de Lorenz es self-excited ya que sus tres puntos de equilibrio son hiperbólicos, es decir que todos los eigenvalores de sus tres puntos de equilibrio tienen parte real. Condiciones iniciales y parámetros convergentes a caos: x(1)=8.923421396589806, y(1)=-0.368700847771444, z(1)=36.308378745785262, g=10, p=28, B=8/3, h=0.0234815162342.

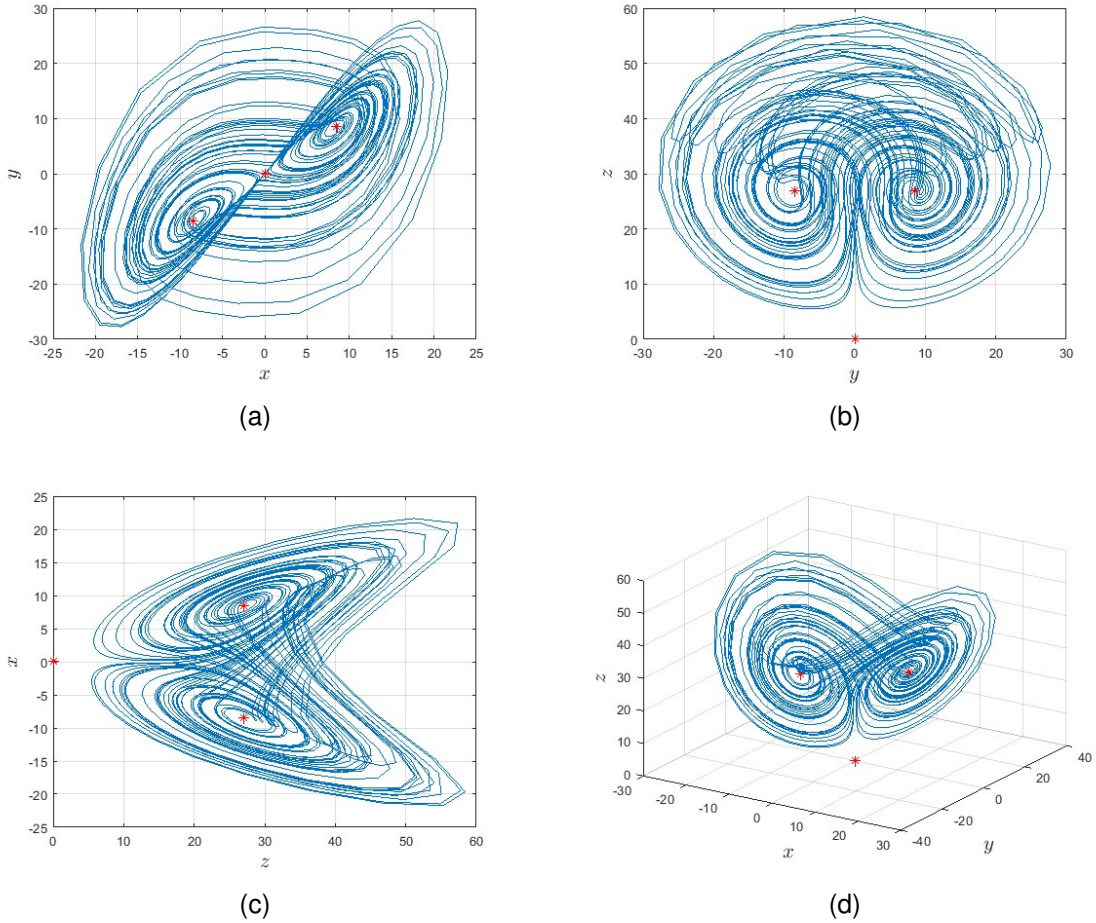


Fig. 5: Proyecciones bidimensionales y atractor tridimensional del sistema de Lorenz para las primeras 2500 iteraciones.

2.8.2 Sistema Hidden Attractor #1:

$$\dot{x} = y - a \tag{7}$$

$$\dot{y} = -bxz + c \tag{8}$$

$$\dot{z} = y - z + y^2 - yz \tag{9}$$

De la ecuación (7) se aprecia que $y = a$, sustituyendo esta igualdad en la ecuación (9) se obtiene:

$$a - z + a^2 - az = 0 = a + a^2 - z(1+a)$$

$$z = \frac{a+a^2}{1+a} = \frac{a(\cancel{1+a})}{\cancel{1+a}} = a$$

Con el resultado $z = a$ se resuelve para x de la ecuación (8):

$$-bxz + c = 0 = -bxa + c \implies x = \frac{c}{ab}$$

Con lo que se obtiene el único punto de equilibrio del sistema:

$$(x, y, z) = \left(\frac{c}{ab}, a, a\right)$$

Para determinar la estabilidad del punto de equilibrio, se calcula la matriz Jacobiana J del sistema evaluando las parciales de cada función para cada variable de estado:

$$J = \begin{bmatrix} 0 & 1 & 0 \\ -bz & 0 & -bx \\ 0 & 1+2y-z & -1-y \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -ab & 0 & -\frac{c}{a} \\ 0 & 1+a & -1-a \end{bmatrix}$$

Y se procede a despejar los eigenvalores del sistema igualando a cero el polinomio característico de $det|J-\lambda I|$:

$$J-\lambda I = \begin{bmatrix} -\lambda & 1 & 0 \\ -bz & -\lambda & -bx \\ 0 & 1+2y-z & -1-y-\lambda \end{bmatrix}$$

$$det|J-\lambda I| = -\lambda^2 - a\lambda^2 - \lambda^3 - \left(\frac{\lambda c}{a} + \lambda c + ab + a^2 b + ab\lambda\right)$$
$$-\lambda^3 - \lambda^2(1+a) - \lambda\left(\frac{c}{a} + c + ab\right) - ab - a^2 b = 0$$

Cuando se le asignan los valores $a = 0.95, b = 1.45, c = 0.01$ a los parámetros se encuentra el único punto de equilibrio estable, con sus correspondientes eigenvalores; se resuelve para λ con ayuda de la función $roots()$ en Matlab:

```
clearvars; a=0.95; b=1.45; c=0.01;
pc = [-1,-1-a,-c-a*b-c/a,-a*b-a*a*b];
Eigenvalores_PE = roots(pc)
```

Se obtienen los autovalores:

$$\lambda_1 = -1.9423, \qquad \lambda_{2,3} = -0.0039 \pm 1.1760i$$

El tipo de estabilidad del único punto de equilibrio es tipo 5 índice 0, es un nodo espiral. El índice indica el número de eigenvalores con parte real positiva. Se considera que el sistema es self-excited ya que su único punto de equilibrio es hiperbólico; todos sus eigenvalores tienen parte real. Condiciones iniciales y parámetros convergentes a caos: x(1)=0.505797298324602, y(1)=-1.383059557764777, z(1)=1.838875274462095, a=0.95, b=1.45, c=0.01, h=0.0134815162342.

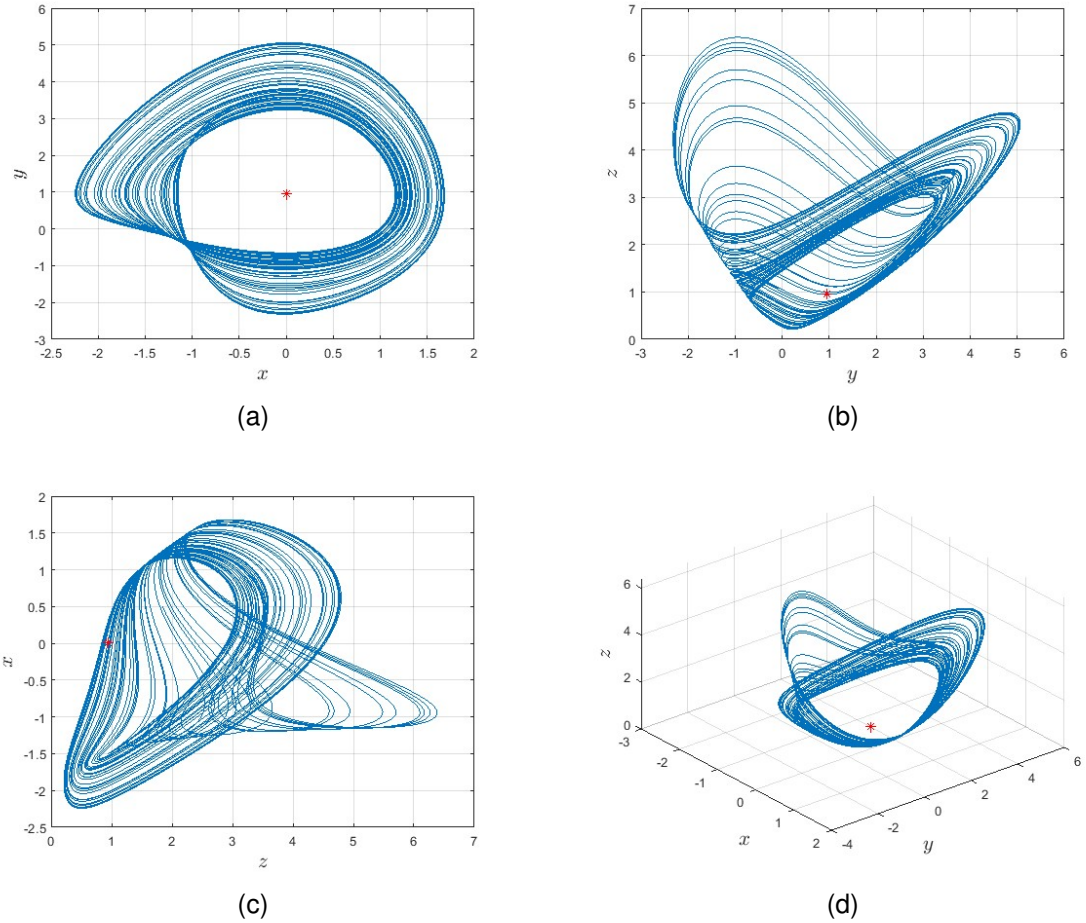


Fig. 6: Proyecciones bidimensionales y atractor tridimensional del sistema Hidden Attractor1 para las primeras 15 mil iteraciones.

2.8.3 Sistema Hidden Attractor #2:

$$D^{q1}x = yz + x(y-a) \tag{10}$$

$$D^{q2}y = 1 - |x| \tag{11}$$

$$D^{q3}z = -xy - z \tag{12}$$

De la ecuación (11) se obtienen los valores para el equilibrio de x:

$$1 - |x| = 0 \implies x = \pm 1$$

De la ecuación (12) se tiene que $z = -xy$, sustituyendo en la ecuación (10) se obtiene:

$$\begin{aligned} -yx(y) + x(y-a) &= 0 \\ -xy^2 + x(y-a) &= 0 \\ x(-y^2 + y - a) &= 0 \end{aligned}$$

Como $x \neq 0$ dado que $|x| = 1$ esto implica la siguiente igualdad:

$$-y^2 + y - a = 0$$

Al resolver la ecuación mediante la fórmula general para ecuaciones cuadráticas, se obtienen las siguientes soluciones para y:

$$y_1 = \frac{1+\sqrt{1-4a}}{2} \qquad y_2 = \frac{1-\sqrt{1-4a}}{2}$$

Para determinar las soluciones de la variable de estado z, se utiliza la relación $z = -xy$ de la ecuación (12), evaluada para cada combinación de $x = \pm 1$ con y_1, y_2. De este modo se obtienen los cuatro puntos de equilibrio del sistema:

$$(x,y,z) = \left(1, \frac{1+\sqrt{1-4a}}{2}, -\frac{1+\sqrt{1-4a}}{2}\right)$$
$$(x,y,z) = \left(1, \frac{1-\sqrt{1-4a}}{2}, -\frac{1-\sqrt{1-4a}}{2}\right)$$
$$(x,y,z) = \left(-1, \frac{1+\sqrt{1-4a}}{2}, \frac{1+\sqrt{1-4a}}{2}\right)$$
$$(x,y,z) = \left(-1, \frac{1-\sqrt{1-4a}}{2}, \frac{1-\sqrt{1-4a}}{2}\right)$$

Se determina la estabilidad para cada punto de equilibrio obteniendo los eigenvalores de $det|J - \lambda I|$. Para estimar el Jacobiano del sistema se tiene una derivada parcial particular:

$$\frac{\partial}{\partial x}(1-|x|) = -\frac{x}{|x|} = \begin{cases} 1 & \text{si} \quad -\infty < x < 0 \\ -1 & \text{si} \quad 0 > x > \infty \end{cases}$$

$$\frac{\partial}{\partial x}(1-|x|) = -sgn(x); \quad x \in \{-1, 1\}$$

$$J = \begin{bmatrix} y-a & z+x & y \\ -\text{sgn}(x) & 0 & 0 \\ -y & -x & -1 \end{bmatrix}$$

Se le asigna valor al único parámetro del sistema $a = -1$ y se calculan los eigenvalores con ayuda de Matlab usando:

```
clearvars; a = -1;
v=[1, (1 + sqrt(1 - 4*a))/2, -(1 + sqrt(1 - 4*a))/2;
   1, (1 - sqrt(1 - 4*a))/2, -(1 - sqrt(1 - 4*a))/2;
  -1, (1 + sqrt(1 - 4*a))/2,  (1 + sqrt(1 - 4*a))/2;
  -1, (1 - sqrt(1 - 4*a))/2,  (1 - sqrt(1 - 4*a))/2 ];
for i = 1:4
    x = v(i, 1); y = v(i, 2); z = v(i, 3);
    J = [y - a,       z + x,    y;
         -sign(x),    0,        0;
         -y,          -x,       -1];
    disp('Puntos de equilibrio:');
    fprintf('x = %f, y = %f, z = %f\n', x, y, z);
```

```
    Eigenvalores = eig(J)
end
```

El resultado del código es:

```
Puntos de equilibrio:
x = 1.000000, y = 1.618034, z = -1.618034
Eigenvalores=
   2.3064 + 0.0000i
  -0.3442 + 0.9225i
  -0.3442 - 0.9225i

Puntos de equilibrio:
x = 1.000000, y = -0.618034, z = 0.618034
Eigenvalores=
   0.2243 + 1.4304i
   0.2243 - 1.4304i
  -1.0666 + 0.0000i

Puntos de equilibrio:
x = -1.000000, y = 1.618034, z = 1.618034
Eigenvalores=
   2.3064 + 0.0000i
  -0.3442 + 0.9225i
  -0.3442 - 0.9225i

Puntos de equilibrio:
x = -1.000000, y = -0.618034, z = -0.618034
Eigenvalores=
   0.2243 + 1.4304i
   0.2243 - 1.4304i
  -1.0666 + 0.0000i
```

Analizando en orden el tipo de estabilidad para los puntos de equilibrio listados se tiene:
$PE(x=1, y=1.61, z=-1.61)$ es tipo 6 índice 1, es un punto de silla espiral.
$PE(x=1, y=-0.61, z=0.61)$ es tipo 7 índice 2, es un punto de silla espiral.
$PE(x=-1, y=1.61, z=1.61)$ es tipo 6 índice 1, es un punto de silla espiral.
$PE(x=-1, y=-0.61, z=-0.61)$ es tipo 7 índice 2, es un punto de silla espiral.
El índice indica el número de eigenvalores con parte real positiva. Se determina que el sistema es self-excited ya que todos sus puntos de equilibrio son hiperbólicos, es decir que cada uno tiene parte real en todos sus eigenvalores. Condiciones iniciales y parámetros convergentes a caos: x(1) = -0.777372627735120, y(1) = -1.096151842328545, z(1) = 0.142988298810338, a=-1, h=0.0004048151623427.

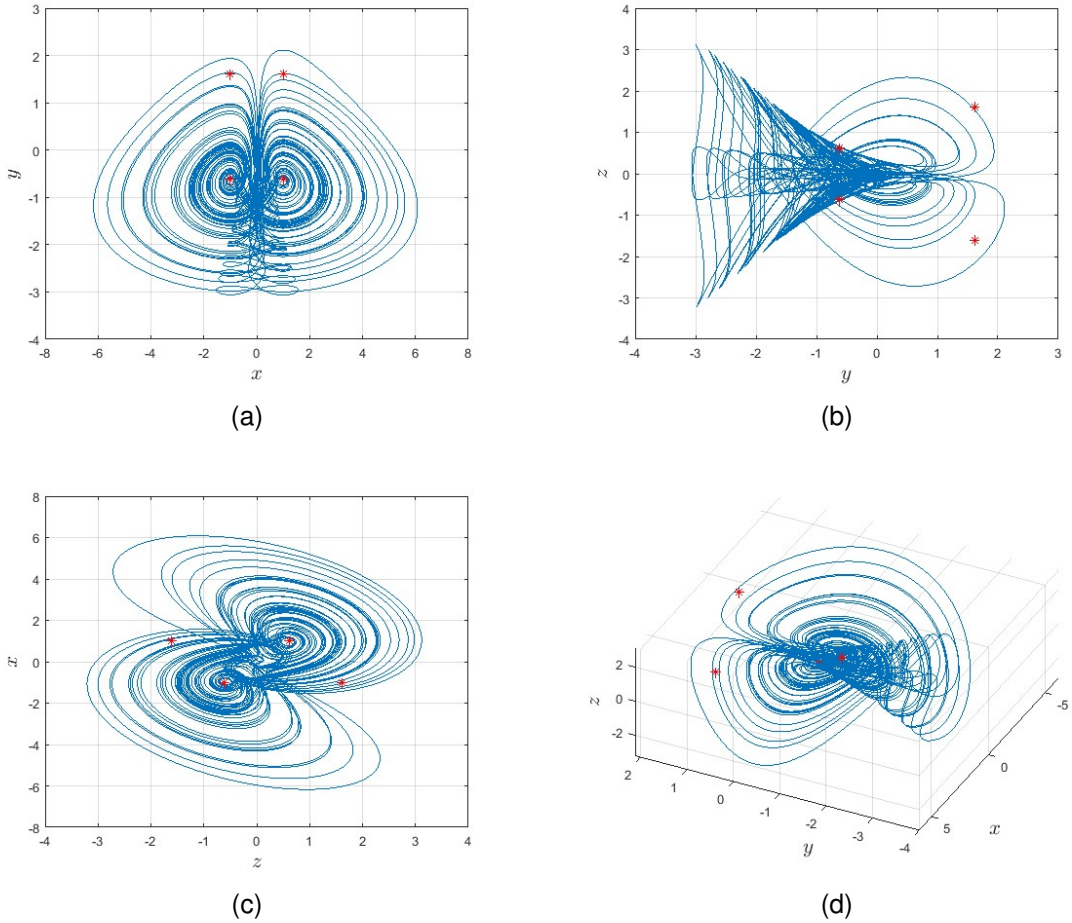


Fig. 7: Proyecciones bidimensionales y atractor tridimensional del sistema Hidden Attractor2 para las primeras 1.5 millones de iteraciones.

2.8.4 Sistema Hidden Attractor #3:

$$\dot{x} = y \tag{13}$$

$$\dot{y} = z \tag{14}$$

$$\dot{z} = -ax - by - cz + hf(x) \tag{15}$$

Donde $f(x)$ es una función piecewise linear basada en una función saturada que se define por:

$$f(x) = \begin{cases} k & \text{si } x > \beta \\ sx & \text{si } -\beta \leq x \leq \beta \\ -k & \text{si } x < -\beta \end{cases}$$

Donde x tiene pendiente s en el intervalo $x = [-\beta, \beta]$ y saturada en valores $-k, k$ para cualquier otro valor de x.

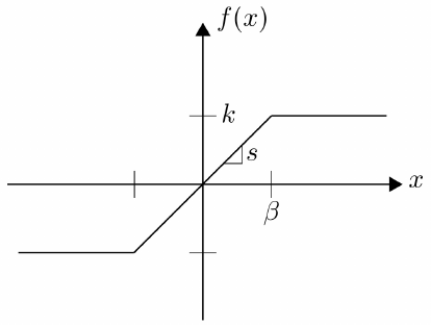


Fig. 8: Función saturada no lineal

Las ecuaciones (13) y (14) respaldan que $y,z=0$. Se determina x de la ecuación (15) evaluando $f(x)$ para los tres intervalos distintos.
Para el intervalo $x>\beta$:

$$-ax-0-0+hk=0 \implies x=\frac{hk}{a}$$

Para el intervalo $-\beta\leq x\leq\beta$:

$$-ax-0-0+hsx=0=x(-a+hs) \implies x=0$$

Para el intervalo $x<-\beta$:

$$-ax-0-0-hk=0 \implies x=-\frac{hk}{a}$$

El sistema cuenta con tres puntos de equilibrio:

$$\left(\frac{hk}{a},0,0\right);\quad (0,0,0);\quad \left(-\frac{hk}{a},0,0\right)$$

Para determinar el tipo de estabilidad de los puntos de equilibrio se linealiza el sistema con su Jacobiano y se resuelve para los eigenvalores λ con $det|J-\lambda I|=0$. Para el primer punto de equilibrio correspondiente al intervalo $x>\beta$:

$$J=\begin{bmatrix}0&1&0\\0&0&1\\-a&-b&-c\end{bmatrix} \therefore J-\lambda I=\begin{bmatrix}-\lambda&1&0\\0&-\lambda&1\\-a&-b&-c-\lambda\end{bmatrix}$$

$$det|J-\lambda I|=-c\lambda^2-\lambda^3-a-(b\lambda)=-\lambda^3-c\lambda^2-b\lambda-a$$

Para el segundo punto de equilibrio $(0,0,0)$ correspondiente al intervalo $-\beta\leq x\leq\beta$:

$$J-\lambda I=\begin{bmatrix}-\lambda&1&0\\0&-\lambda&1\\-a+hs&-b&-c-\lambda\end{bmatrix}$$

$$det|J-\lambda I|=-c\lambda^2-\lambda^3-a+hs-(b\lambda)=-\lambda^3-c\lambda^2-b\lambda-a+hs$$

Es notable que el polinomio característico para el tercer punto de equilibrio correspondiente al intervalo $x < -\beta$ es el mismo que para $x > \beta$:

$$det|J - \lambda I| = -\lambda^3 - c\lambda^2 - b\lambda - a = 0$$

Se asignan los valores a los parámetros: $a = 0.7, b = 0.7, c = 0.7, h = 0.7, s = 10$ y se obtienen los eigenvalores con ayuda de la función $roots()$ de Matlab:

```
clearvars; a=.7; b=.7; c=.7; h=.7; s=10;
pc1 = [-1, -c, -b, -a];
Eigenvalores_PE1 = roots(pc1)
pc2 = [-1, -c, -b, -a + h*s)];
Eigenvalores_PE2 = roots(pc2)
pc3 = [-1, -c, -b, -a];
Eigenvalores_PE3 = roots(pc3)
```

La salida del código es:

```
Eigenvalores_PE1 =
   0.0740 + 0.9055i
   0.0740 - 0.9055i
  -0.8480 + 0.0000i

Eigenvalores_PE2 =
  -1.1154 + 1.6944i
  -1.1154 - 1.6944i
   1.5309 + 0.0000i

Eigenvalores_PE3 =
   0.0740 + 0.9055i
   0.0740 - 0.9055i
  -0.8480 + 0.0000i
```

Se le asigna el valor $k = 1$, lo que sugiere $\beta = 0.1$ dado que s es la pendiente en el segmento $-\beta \leq x \leq \beta$, así mismo se obtienen los lugares de los puntos de equilibrio estimados:

$$(1,0,0); \quad (0,0,0); \quad (-1,0,0)$$

Analizando en orden el tipo de estabilidad para los puntos de equilibrio listados se tiene:
$PE(1,0,0)$ es tipo 7 índice 2, es un punto de silla espiral.
$PE(0,0,0)$ es tipo 6 índice 1, es un punto de silla espiral.
$PE(-1,0,0)$ es tipo 7 índice 2, es un punto de silla espiral.
El índice indica el número de eigenvalores con parte real positiva. Se determina que el sistema es self-excited ya que todos sus puntos de equilibrio son hiperbólicos, es decir que cada punto de equilibrio tiene parte real en todos sus eigenvalores. Condiciones iniciales y parámetros convergentes a caos: x(1) = -1.070152716191076, y(1)= -0.766479112437380, z(1) = -0.055488532023687, a=0.7, b=0.7, c=0.7, hh=0.7, s=10, beta=0.1, h=0.2248151623429557.

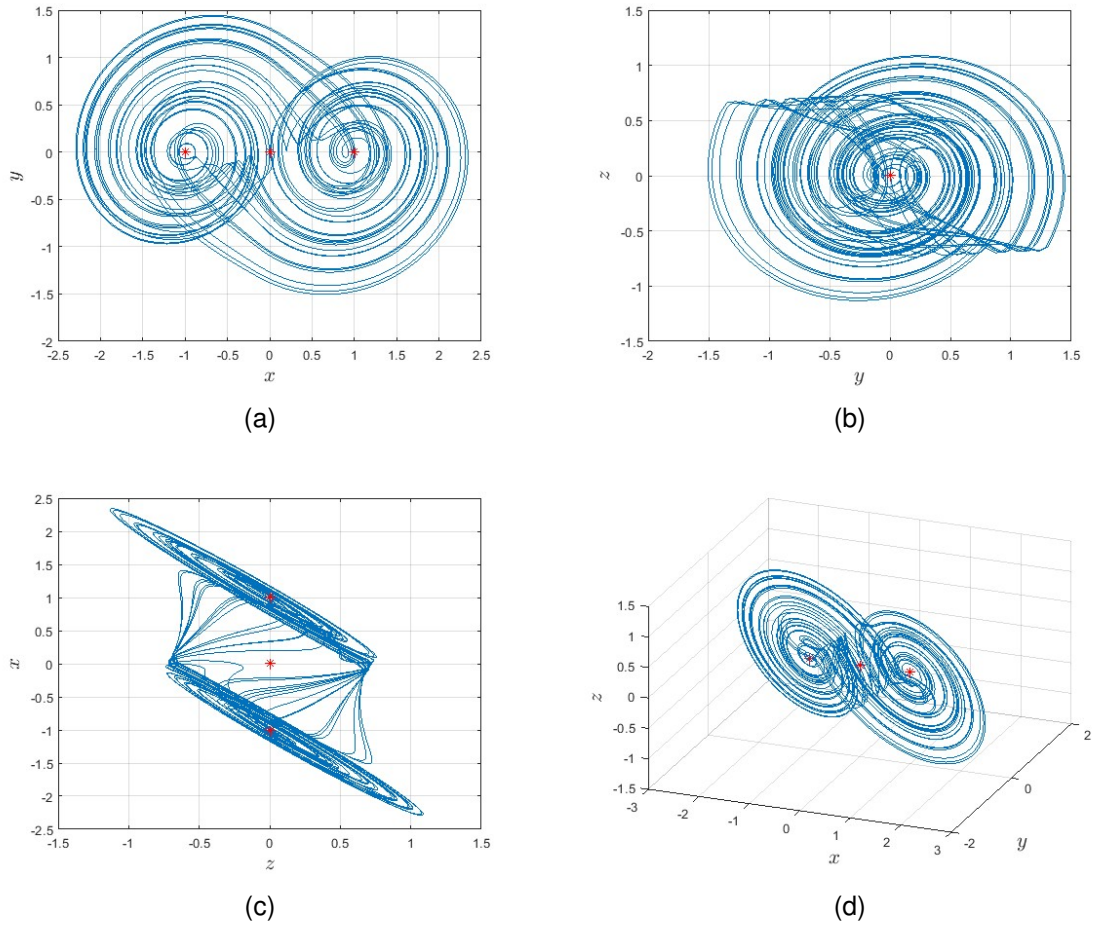


Fig. 9: Proyecciones bidimensionales y atractor tridimensional del sistema Hidden Attractor3 para las primeras 1500 soluciones.

2.8.5 Sistema de Chua:

$$\frac{dx}{d\tau} = \alpha\left(-x + y - u(x)\right) \tag{16}$$

$$\frac{dy}{d\tau} = x - y + z \tag{17}$$

$$\frac{dz}{d\tau} = -\beta y \tag{18}$$

Donde $u(x)$ es una función cúbica suave asignando $u(x) = a_1 x + a_3 x^3$.
Para hallar los puntos de equilibrio se igualan las ecuaciones a 0 y se despejan las variables de estado. De la ecuación (18) se obtiene:

$$-\beta y = 0 \implies y = \frac{0}{-\beta} = 0$$

Sustituyendo $y = 0$ en la ecuación (16) se tiene:

$$\alpha\left(-x + y - a_1 x - a_3 x^3\right) = 0$$

$$\alpha x\left(-1 - a_1 - a_3 x^2\right) = 0$$

Despejando el miembro fuera de los paréntesis:

$$\alpha x = 0 \implies x = 0$$

Sustituyendo $x, y = 0$ en la ecuación (17) se tiene:

$$x - y + z = 0 = 0 - 0 + z \implies z = 0$$

Entonces la primera solución de la variable de estado $x = 0$ implica un punto de equilibrio en $(x, y, z) = (0, 0, 0)$.
Continuando con la resolución de las demás raíces de la variable de estado x para el miembro que está entre paréntesis en $\alpha x\left(-1 - a_1 - a_3 x^2\right) = 0$ se obtiene:

$$-1 - a_1 - a_3 x^2 = 0 \implies x^2 = \frac{1 + a_1}{-a_3}$$

$$x = \pm\sqrt{-\frac{1 + a_1}{a_3}}$$

La ecuación (17) indica que $z = -x$, por lo tanto se tienen los dos puntos de equilibrio restantes y el trivial como:

$$(x, y, z) = \left(\sqrt{-\frac{1 + a_1}{a_3}}, 0, -\sqrt{-\frac{1 + a_1}{a_3}}\right)$$
$$(x, y, z) = \left(-\sqrt{-\frac{1 + a_1}{a_3}}, 0, \sqrt{-\frac{1 + a_1}{a_3}}\right)$$
$$(x, y, z) = (0, 0, 0)$$

Para determinar el tipo de estabilidad de los puntos de equilibrio primero se linealiza el sistema con su Jacobiano y se resuelve para los eigenvalores λ con $det|J - \lambda I| = 0$.

$$J = \begin{bmatrix} -3\alpha a_3 x^2 - \alpha a_1 - \alpha & \alpha & 0 \\ 1 & -1 & 1 \\ 0 & -\beta & 0 \end{bmatrix}$$

Se le asignan valores a los parámetros del sistema $\alpha = 11, \beta = 20, a_1 = -1.3, a_3 = 0.001$ y se calculan los eigenvalores con ayuda de Matlab usando el siguiente código:

```
clearvars; a=11; b=20; a1=-1.3; a3=0.001;
v=[sqrt(- (1 + a1) / a3), 0, -sqrt(- (1 + a1) / a3);
     -sqrt(- (1 + a1) / a3), 0, sqrt(- (1 + a1) / a3);
     0,0,0];
for i = 1:3
    x = v(i, 1); y = v(i, 2); z = v(i, 3);
    J = [-3*a*a3*x^2-a*a1-a,a, 0;1,-1,1;0,-b,0];
    disp('Puntos de equilibrio:');
    fprintf('x = %f, y = %f, z = %f\n', x, y, z);
    Eigenvalores = eig(J)
end
```

Se obtiene de resultado los valores de los puntos de equilibrio así como sus correspondientes eigenvalores:

```
Puntos de equilibrio:
x = 17.320508, y = 0.000000, z = -17.320508
Eigenvalores =
  -7.7767 + 0.0000i
   0.0883 + 4.1190i
   0.0883 - 4.1190i

Puntos de equilibrio:
x = -17.320508, y = 0.000000, z = 17.320508
Eigenvalores =
  -7.7767 + 0.0000i
   0.0883 + 4.1190i
   0.0883 - 4.1190i

Puntos de equilibrio:
x = 0.000000, y = 0.000000, z = 0.000000
Eigenvalores =
   4.4061 + 0.0000i
  -1.0530 + 3.7243i
  -1.0530 - 3.7243i
```

Analizando en orden el tipo de estabilidad para los puntos de equilibrio listados se tiene:
$PE(17.3, 0, -17.3)$ es tipo 7 índice 2, es un punto de silla espiral.
$PE(-17.3, 0, 17.3)$ es tipo 7 índice 2, es un punto de silla espiral.
$PE(0, 0, 0)$ es tipo 6 índice 1, es un punto de silla espiral.
El índice indica el número de eigenvalores con parte real positiva. Se determina que el sistema es self-excited ya que todos sus puntos de equilibrio son hiperbólicos, es decir que cada punto de equilibrio tiene parte real en todos sus eigenvalores. Condiciones iniciales y parámetros convergentes a caos: x(1) = -0.350426734, y(1) = 3.753408839, z(1) = -8.055190369, a=11, b=20, a1=-1.3, a3=0.001, h=0.03604815162342.

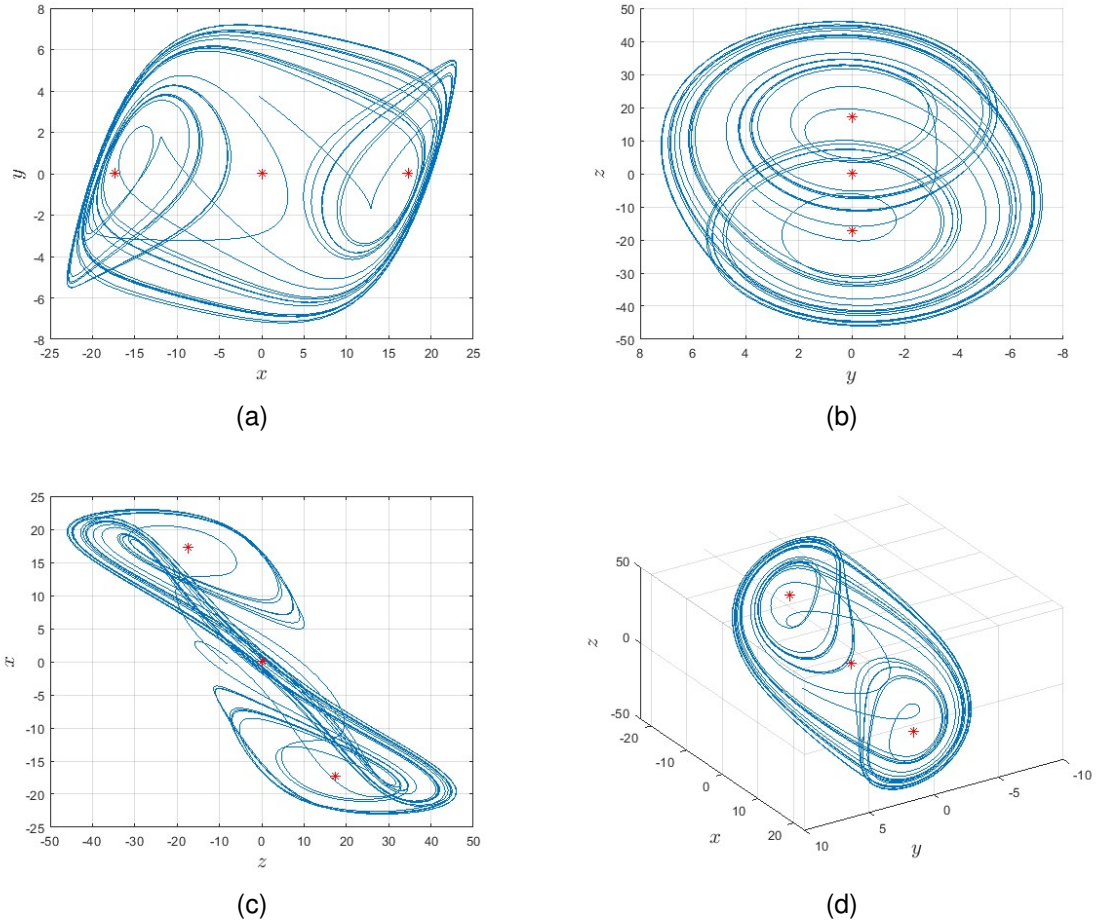


Fig. 10: Proyecciones bidimensionales y atractor tridimensional del sistema de Chua para las primeras 1500 iteraciones.

3 tipos de cifrado, generadores de números y evaluaciones estadísticas

En criptografía, la encriptación es el proceso de transformar un mensaje original (texto plano) en datos ininteligibles (texto cifrado) utilizando un algoritmo de encriptación. Un texto cifrado no debe brindar información alguna sobre el texto plano excepto a quienes cuentan con la llave de cifrado [9].

Las prácticas de la criptografía se remontan a civilizaciones antiguas como los egipcios, los griegos o los romanos que utilizaban métodos rudimentarios para ocultar la información que se enviaba entre ciudades, con el tiempo los métodos se fueron haciendo más sofisticados, no obstante el uso de la máquina Enigma durante la segunda guerra mundial, que era una máquina mecánica que generaba claves de cifrado aparentemente infinitas, marcó un punto de inflexión en la historia que condujo al desarrollo de las primeras computadoras eléctricas y con ellas métodos avanzados de criptoanálisis. A día de hoy, las matemáticas, la física y la electrónica constituyen las principales ciencias que brindan las bases para los métodos modernos de cifrado [10].

3.1 Tipos de cifrado punto a punto

Existen dos tipos de cifrado: el cifrado simétrico y el cifrado asimétrico.

El cifrado asimétrico, también conocido como cifrado de llave pública, utiliza un par de llaves: una pública y una privada. En el proceso para establecer la comunicación, cada receptor de mensajes crea su par de llaves y comparte su llave pública con el emisor sin considerar el aseguramiento del canal. Aunque se trata de llaves distintas, la relación entre ambas se basa en funciones matemáticas unidireccionales; fáciles de calcular en un sentido pero computacionalmente inviables de invertir, donde la llave privada es computacionalmente costosa y hasta imposible de encontrar mediante la llave pública. De este modo el remitente puede cifrar mensajes con la llave pública del receptor, quien los descifra con su llave privada; esto mantiene la confidencialidad del mensaje aunque sin verificar la identidad del remitente. En contraste, los mensajes cifrados con la llave privada se pueden descifrar con la llave pública y, al tratarse de un mensaje cifrado con una llave no compartida, esto certifica la autenticidad del remitente además de mantener la confidencialidad del mensaje. Este tipo de cifrado es común en autenticación y firmas digitales [9], sus principales usos son en protocolos de comunicaciones como: HTTPS (HyperText Transfer Protocol Secure), SSL (Secure Sockets Layer), SSH (Secure Shell), PGP (Pretty Good Privacy), IPsec (Internet Protocol Security), Blockchain, criptomonedas.

El cifrado simétrico emplea una única clave secreta compartida entre el emisor y el receptor para cifrar y descifrar la información. Su principal ventaja respecto

al cifrado asimétrico es la rapidez en el procesamiento de grandes volúmenes de datos, aunque su seguridad depende del resguardo y la confidencialidad de la clave compartida. Dentro de este esquema, los cifrados simétricos pueden ser usados como cifrados de bloques y cifrados de flujo. En el cifrado de bloques, la información es dividida en bloques de tamaño fijo, donde puede existir una relación y dependencia entre los bloques cifrados, por ejemplo la encriptación de un bloque puede depender de la permutación con bloques previos; esto le brinda al cifrado por bloques las propiedades de confusión y difusión que son características de una información cifrada segura. El cifrado por bloques ofrece una mejor seguridad que el cifrado de flujo frente a ciertos tipos de criptoanálisis. Los cifrados de flujo operan bit a bit, el cifrado entre bits es independiente y sólo dependen de secuencias de claves pseudoaleatorias generadas a partir de una llave, también llamada semilla. La seguridad del cifrado de flujo depende de la calidad e impredecibilidad de las claves pseudoaleatorias generadas. El operador utilizado para cifrar un texto plano con las claves generadas normalmente debe ser tan simple como sea posible, en la mayoría de casos es la operación XOR entre los bits del texto plano y las claves pseudoaleatorias generadas a partir de la llave, en términos de velocidad el cifrado de flujo es más rápido que el cifrado de bloques lo que brinda una baja latencia en las comunicaciones [9]. Los cifrados simétricos se emplean donde se requiere procesar grandes volúmenes de información de forma rápida y eficiente, sus principales aplicaciones incluyen el cifrado de discos duros, bases de datos, comunicaciones privadas en red como VPNs, servicios de mensajería instantánea y en comunicaciones de voz y datos en general.

3.2 Generador de números aleatorios

Los métodos de generación de números aleatorios son de particular interés para la criptografía, fundamentales en el proceso de cifrado para garantizar la confidencialidad de información digital almacenada y su transmisión por cables o medios inalámbricos, e igualmente en protocolos de autenticación en redes u otros sistemas criptográficos. En este marco, los sistemas caóticos emergen como una fuente de entropía prometedora debido a sus propiedades intrínsecas, tales como la no periodicidad, la impredecibilidad y su sensibilidad a las condiciones iniciales.

Existen dos tipos de sistemas generadores de secuencias numéricas aleatorias; generadores de números pseudoaleatorios (PRNGs, por sus siglas en inglés: Pseudo-Random Number Generators) y generadores de números aleatorios reales (TRNGs, por sus siglas en inglés: True Random Number Generators). Ambos se diferencían por la naturaleza de sus procesos generativos; mientras los PRNGs son sistemas definidos por algoritmos deterministas que producen

secuencias reproducibles a partir de una semilla inicial conocida, los TRNGs basan sus procesos generativos en sistemas analógicos con procesos físicos aleatorios, por ejemplo, el ruido introducido por el ambiente en un elemento resistivo [11].

3.2.1 TRNGs: Los TRNGs son sistemas analógicos (o híbridos) que producen secuencias numéricas a partir de fuentes de entropía sujetas a las condiciones del entorno, esto les confiere una aleatoriedad que no puede replicarse mediante algoritmos puramente matemáticos. Las fuentes de entropía más comunes de estos sistemas son el ruido en resistencias y capacitores de circuitos eléctricos, desfases de señales oscilantes denominadas jitter, y los estados de sistemas caóticos como el circuito de Chua.

Para los sistemas criptográficos modernos es de interés el digitalizar las señales analógicas de estas fuentes de entropía lo cual lleva implícitas ciertas limitaciones que dificultan la uniformidad en la distribución estadística de las secuencias numéricas generadas, por ejemplo, en un sistema basado en el ruido de una resistencia eléctrica, las señales suelen amplificarse y compararse con un nivel de referencia para obtener señales digitales (bits), en este caso los amplificadores que tienen un ancho de banda limitado, aunado a que el ruido en la resistencia puede presentar periodicidad y el nivel de comparación puede estar desalineado, las semillas o secuencias que generan suelen propender a ciertos valores más que a otros, esto se traduce en claves de cifado no apropiadas para aplicaciones criptográficas de seguridad a causa de la mala calidad del generador debido a un sesgo en la función de distribución de los valores del elemento analógico.

En contraste con lo anterior, diversos trabajos e investigaciones evidencian una mejora notable en la calidad de los productos numéricos al usar sistemas que aprovechan las propiedades dinámicas del caos. Estos sistemas caóticos pueden clasificarse en: caóticos de tiempo continuo y caóticos de tiempo discreto. Los primeros, modelados por ecuaciones diferenciales dependientes del tiempo del tipo $\dot{x} = f(t,x)$, si bien pueden ser modelados matemáticamente, estos sistemas son no deterministas pues sus elementos activos permanecen influenciados por el ruido que el ambiente pueda generar en ellos, inclusive llegando a provocar que pierdan el régimen/comportamiento caótico [12] y puedan requerir un reinicio. El circuito de Chua o el sistema de Lorenz, sistema de jerk, oscilador caótico boleano, jitter, oscilador caótico acoplado o basados en FPGAs son los principales ejemplos de sistemas caóticos de tiempo continuo y con elementos activos sujetos al ruido lo cual los categoriza como TRNGs.

En la segunda categoría de TRNGs basados en caos, están los sistemas que se desarrollan en iteraciones de tiempo discreto modelados por ecuaciones del tipo $x_{k+1} = \tau(x_k)$. El funcionamiento de estos sistemas se basa en una semilla (condiciones iniciales) adquirida de un elemento activo para desarrollar sus

estados mediante un sistema determinista, normalmente son circuitos eléctricos que desarrollan sus procesos en un marco digital, en ellos es común encontrar señales de reloj que marcan el ritmo del procesamiento del sistema, por ejemplo, un mapa caótico de tiempo discreto es determinista; si se conoce su estado inicial, los estados que desarrolla pueden ser predecidos, sin embargo, debido a que el estado inicial se basa en el ruido térmico del circuito, resulta prácticamente imposible determinar su estado inicial y consecuentemente sus estados futuros. Bernooulli shift y tent maps son los dos mapas caóticos de tiempo discreto más usados [13]. Ejemplos de estos son el mapeo logístico,Tent mapping, Bernoulli mapping, PWAM, osciladores caóticos de tiempo discreto.

3.2.2 PRNGs: Un PRNG es un generador que simula un comportamiento aleatorio, se caracterizan por producir secuencias numéricas mediante algoritmos deterministas a partir de una semilla inicial conocida. Este proceso implica la extensión de una entrada corta (semilla) en una secuencia de salida más larga, lo que las hace reproducibles y predecibles si se conoce la semilla y el algoritmo utilizados. Debido a esta naturaleza determinista, los PRNGs son ampliamente utlizados en entornos donde la reproducibilidad es una ventaja, como por ejemplo en un sistema de cifrado simétrico de flujo. Su falta de verdadera aleatoriedad los hacen menos idóneos que los TRNGs para aplicaciones criptográficas de seguridad [11].

Algunos tipos de PRNGs están sujetos a ciertas debilidades como lo son el presentar ciclos y tener periodos reducidos de un comportamiento altamente complejo. Las secuencias numéricas generadas por un PRNG deben tener las cualidades de aleatoriedad e impredecibilidad. Aleatoriedad: cada bit debe tener una probablidad exacta de 0.5 de tener un valor de 0 o 1. Al evaluar las secuencias generadas los bits deben ser independientes entre sí y seguir una distribución uniforme, si bien existen pruebas estadísticas para evaluar una distribución uniforme, no existen pruebas estadísticas para determinar la independencia de los bits generados. Impredecibilidad: un tercero, mediante una conjetura no despreciable, no debería poder predecir o calcular el bit previo o siguiente del actual en la generación de claves [9].

Los modelos matemáticos en los que se basan los PRNGs también pueden ser los mismos que el de los sistemas de los TRNGs de tiempo continuo y los deterministas de tiempo discreto, de este modo los PRNGs basados en modelos caóticos están exentos de las usuales debilidades de presentar ciclos y tener periodos cortos de alta complejidad, con el añadido de ser sensibles a las condiciones iniciales, lo que ayuda a "ocultar" la semilla al eliminar un transiente o cierta cantidad de iteraciones.

3.2.3 Post-procesamiento: Para que un sistema generador de números sea apto para aplicaciones criptográficas es necesario que sus productos sean de alta calidad, es decir, que la distribución estadística de las secuencias numéricas

generadas sea uniforme con el objetivo de dificultar el criptoanálisis de la información que cifran. Esta uniformidad es esencial, ya que cualquier desviación o sesgo en la distribución reduciría la entropía efectiva del sistema y podría ser explotado por un atacante para inferir patrones, comprometiendo su seguridad.

A pesar de las importantes características de los sistemas caóticos y las mejoras en aleatoriedad que brindan respecto a otras fuentes de entropía, las secuencias numéricas generadas por un sistema caótico por si solo normalmente no cumplen con una distribución uniforme. Por este motivo es común encontrar etapas de post-procesamiento que mejoren la distribución estadística de las secuencias numéricas de un generador, a la par de someter los productos post-procesados a una serie de evaluaciones estadísticas que validen la distribución uniforme y aleatoriedad del sistema generador para aplicaciones de seguridad.

Para que el sistema sea seguro las claves de cifrado generadas deben ser impredecibles; sus estados futuros no deben poder predecirse a partir de los anteriores. Se asumen como necesarios los siguientes requisitos para garantizar la impredecibilidad de un sistema generador [14]:

1) Período largo: Un flujo de claves debe tener un período largo.
2) Alta complejidad lineal: Una complejidad alta dificulta un determinado criptoanálisis que podrían descifrar la llave y reconstruir sus secuencia si estas fueran linealmente simples.
3) Aleatoriedad: Una gran complejidad lineal no implica aleatoriedad. Las propiedades estadísticas de un bitstream deben ser las mismas que las de una fuente aleatoria ideal.
4) Inmunidad a correlación de orden adecuado: Esto asegura que ninguna combinación de un número limitado de bits de entrada (texto plano) esté correlacionada con la salida del generador (texto cifrado), protegiendo contra ataques de correlación.

Los métodos de post-procesamiento son variados y pueden ser desarrollados por hardware o por algoritmos definidos en software, estos no solo buscan mejorar la aleatoriedad y la función de distribución de las secuencias generadas, sino también garantizar que estas cumplan con las condiciones para su impredecibilidad.

La estructura de un sistema generador de números aleatorios con una fuente de entropía analógica comprende las siguientes características:

1) Obtención de una señal analógica aleatoria a partir de la fuente de entropía, mediante una sonda colocada en un elemento activo del circuito de entropía.
2) Muestreo y cuantificación de la señal mediante un bloque *sample and hold* (S/H), un dispositivo analógico que toma muestras de una señal variable y mantiene su valor constante durante un tiempo mínimo específico.
3) Conversión analógico-digital de la señal retenida para generar secuencias digitales aleatorias, utilizando un convertidor analógico-digital (ADC) que produce un valor digital, por ejemplo, en formato IEEE de precisión simple o doble.
4) Postprocesamiento de las secuencias para garantizar una distribución uniforme.
5) Validación de las secuencias numéricas mediante pruebas estadísticas de aleatoriedad.

Con una fuente de entropía definida en software, las estapas 1 a 3 difieren del siguiente modo:

1) Mediante integración numérica se desarrollan los estados de la fuente de entropía.
2) Producción de valores digitales de punto flotante (por ejemplo, 4.815162342) y almacenamiento de estos cada cierto número de iteraciones o con un tamaño de paso de integración determinado.
3) Conversión para generar secuencias digitales aleatorias, creando un bitstream a partir de los valores de punto flotante en formato IEEE, por ejemplo, mediante un mapeo que considera los valores máximo y mínimo de la variable de estado de interés, lo que ajusta la resolución del dato y el balance de sus bits.
4) Postprocesamiento para mejorar la distribución estadística de las secuencias.
5) Validación de las secuencias mediante pruebas estadísticas de aleatoriedad.

Una herramienta para mejorar la aleatoriedad y distribución estadística de una secuencia numérica generada con los estados de alguna fuente de entropía, son los registros de desplazamiento lineal retroalimentados (LFSR, por sus siglas en inglés: Linear Feedback Shift Register).

Estas estructuras de procesamiento, en primera instancia, son generadores independientes que crean patrones pseudoaleatorios con excelentes características en aleatoriedad y distribución estadística, y son ampliamente utilizados en sistemas de cifrado de flujo ya que su desempeño se ajusta a sistemas con características de altas velocidades de cifrado con reducido poder de cómputo. El comportamiento de los LFSR es matemáticamente bien comprendido; al escoger un vector de retroalimentación que corresponda a un polinomio primitivo, este generará un bitstream que iterará a través de todos los posibles estados lineales no nulos del registro con un periodo de repetición de $2^n - 1$ donde n es el número de celdas (bits) del registro [15].

El registro de desplazamiento lineal consiste en una serie de celdas que son inicializadas por un vector de bits que es la llave del sistema, también conocida como semilla; los estados de las celdas del registro cambian en cada iteración desplazando sus contenidos una posición a la derecha. En un LFSR el resultado de la operación XOR de un subconjunto de celdas es colocado en la celda más a la izquierda, durante este procedimiento de desplazamiento se suele derivar un bit de salida.

Los LFSR son sistemas independientes deterministas, sus secuencias de salida están determinadas por el estado del bit actual, su semilla y su polinomio característico de módulo 2, donde los coeficientes del polinomio solo pueden ser 0 o 1, lo que refleja la inclusión o exclusión de cada celda en una operación XOR de retroalimentación, operación que equivale a una sumatoria lo que es consistente en el caso de operaciones binarias [16]. Por ejemplo, en la Figura 11 se muestra el diagrama del LFSR definido por el polinomio primitivo $P'(x) = x^4 + x + 1$:

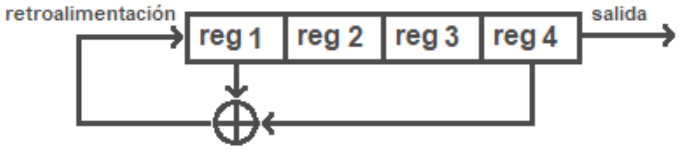


Fig. 11: Diagrama del LFSR para el polinomio primitivo $P'(x) = x^4 + x + 1$

El término $+1$ en el polinomio no implica la combinación de un registro adicional, sino que forma parte de la estructura del polinomio que garantiza su propiedad de ser irreducible. Con el polinomio primitivo para el LFSR de 4 bits, los estados de los registros se mantendrán en un ciclo con un periodo de repetición $2^n - 1$ como se muestra en la Figura 12:

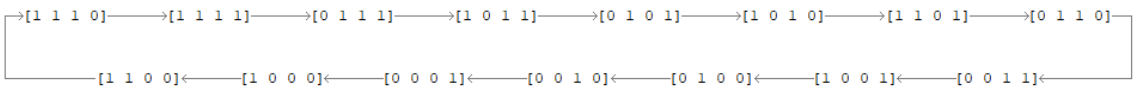


Fig. 12: Estados del registro con el polinomio primitivo $P'(x) = x^4 + x + 1$

El polinomio primitivo asegura que se genere una y otra vez el bistream 011110101100100, en esta secuencia se encuentran todos los estados del LFSR y se le conoce como secuencia de longitud máxima. Las secuencias de longitud máxima exhiben propiedades estadísticas notables: el balance entre ceros y unos es casi ideal, con 2^{n-1} unos y $2^{n-1} - 1$ ceros; la distribución rachas (secuencias consecutivas de ceros o unos) sigue un patrón exponencial decreciente, donde la mitad tiene longitud 1, un cuarto tiene longitud 2, y así sucesivamente; y la autocorrelación circular es máxima, igual a 1 cuando la secuencia está alineada consigo misma, pero cae abruptamente a un valor cercano a 0 para cualquier desplazamiento, lo que indica una baja similitud consigo misma en posiciones distintas [17].

Además de su uso como generadores autónomos, los LFSRs pueden combinarse con fuentes externas de bitstreams como los generadores caóticos, esta integración se realiza mediante la operación XOR entre el bitstream externo y los bits del registro de desplazamiento. Algunos autores, como en [18] y [19], han explorado configuraciones donde los registros carecen de taps internos, es decir, sin retroalimentación basada en un polinomio; en su lugar, para la entrada de retroalimentación se utilizan los bits de una fuente externa y el bit menos significativo del registro, también derivado como el bit de salida. A este sistema se le denomina scrambler.

A diferencia de los LFSR autónomos, que requieren una semilla no nula para evitar un estado trivial, los scramblers permiten inicializar todas las celdas del registro en cero sin afectar su funcionalidad. Mientras que los LFSR autónomos generan de forma cíclica secuencias binarias de longitud máxima mediante un polinomio primitivo, su integración con un sistema caótico como fuente externa

elimina esta periodicidad al incorporar la impredecibilidad inherente del caos al sistema. En consecuencia, un scrambler sin taps, al depender de la señal externa en lugar de un polinomio interno, pierde la garantía de generar secuencias de longitud máxima, pero puede exhibir mejores propiedades estadísticas y optimizar la eficiencia computacional al reducir la estructura de cálculo.

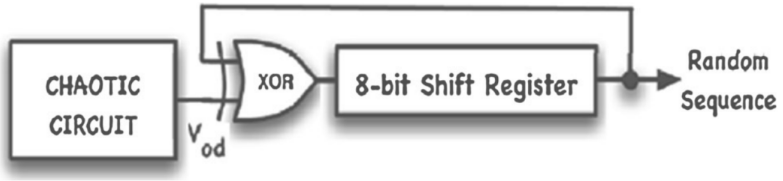


Fig. 13: Diagrama de retroalimentación del scrambler sin taps [18].

En su funcionamiento, los scramblers producen secuencias que preservan ciertas similitudes con la señal de entrada en términos de su distribución estadística. Por ejemplo, si un conjunto de valores con resolución de 32 bits provenientes de una fuente caótica se ordenan de menor a mayor, al graficar estos arreglos se puede apreciar la distribución frecuencial de la fuente caótica delimitada por una línea sesgada, su forma está relacionada con la densidad de los valores de la variable de estado; al procesar la señal de la fuente caótica con un scrambler y crear palabras de 32 bits, ordenándolas de menor a mayor se observa que la distribución resultante mejora su ajuste a una forma ideal o distribución uniforme pero exhibe una recurrencia periódica de la función de distribución del conjunto de datos de entrada a lo largo de su distribución de salida. Esta periodicidad está directamente relacionada con el número de bits del registro de desplazamiento, lo que sugiere que el scrambler introduce una clase de modulación dependiente de su tamaño.

Si la función de distribución de la señal de entrada se asemeja a una distribución ideal, las secuencias procesadas tienen una mayor probabilidad de satisfacer pruebas de aleatoriedad como las del NIST. Por ello, al diseñar un generador destinado a integrarse con un scrambler, resulta ventajoso optimizar la función de distribución de los datos de entrada para que se aproxime en lo más posible a una distribución uniforme. Para ilustrar este fenómeno, la Figura 14 muestra el efecto de procesar una señal caótica con una función de distribución cuadrática ascendente a través de scramblers con diferentes tamaños de registro:

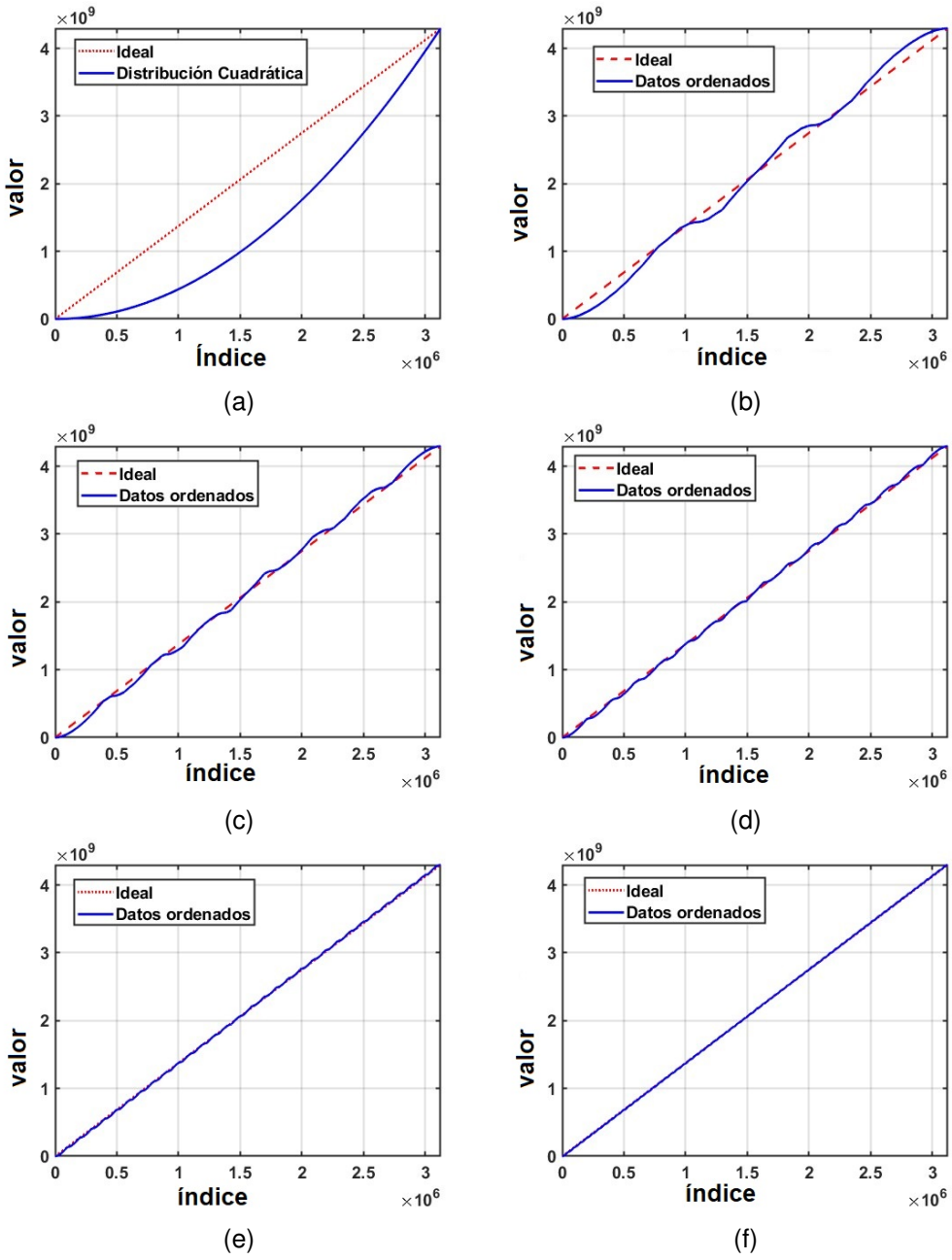


Fig. 14: a) Los datos ordenados de menor a mayor de la señal de entrada con índices de la fuente de entropía caótica; (b-f) muestran los datos ordenados tras pasar la señal caótica por scramblers con registros de tamaño incremental desde 2 a 6 bits, respectivamente.

Se aprecia cómo la distribución evoluciona hacia una forma más uniforme, con patrones recurrentes semejantes a la función de distribución de los datos de entrada y cuya estructura depende del tamaño del registro de desplazamiento. Sugiriendo que un registro de desplazamiento más grande mejora dicho ajuste, optimizando el balance de 1s y 0s, lo que incrementa la calidad estadística de la secuencia.

Estos resultados son consistentes con [19], donde se evaluaron las variables de estado del modelo de Chua, tanto matemático como físico, con y sin un

scrambler, ajustando la resolución de la fuente caótica entre 8 y 32 bits bajo un tamaño de registro fijo de 6 bits. Este tamaño de registro lo reportan como el mínimo necesario para superar las pruebas estadísticas. Los autores determinaron que para producir secuencias de buena calidad estadística, la resolución mínima de la fuente de entropía debe ser de 8bits, e incrementando esta resolución en la entrada se mejora la calidad estadística de la secuencia procesada. Así mismo, concluyeron que en implementaciones físicas, como el circuito de Chua, el uso de LFSRs permite generar secuencias numéricas de alta calidad estadística con una baja resolución de la fuente caótica lo que tiene implicaciones prácticas en la reducción de la actividad de conmutación en el ADC y el consumo energético.

En nuestro caso la resolución de entrada es de 32 bits, si esta se reduce a 8 bits o menos aparecen patrones en su función de distribución degradando la uniformidad en la salida, independientemente de la resolución de los valores en la salida:

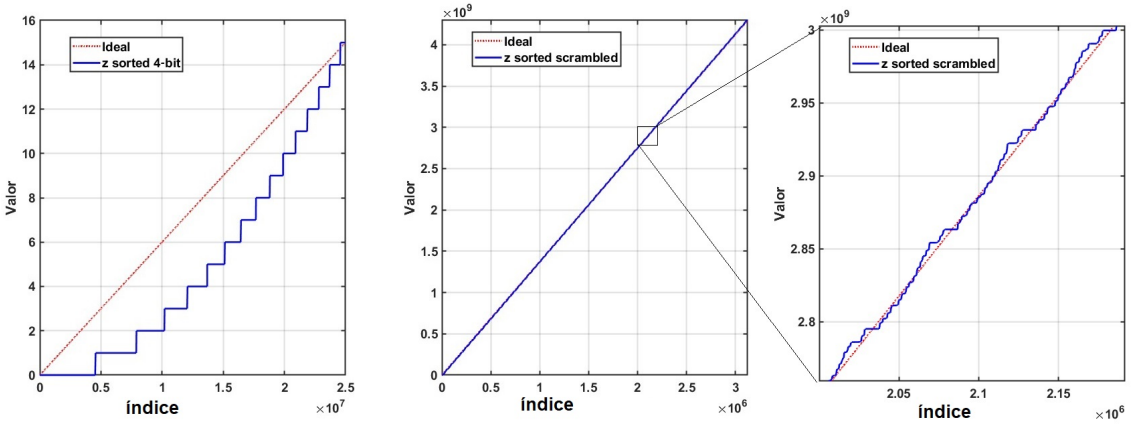


Fig. 15: Resolución de entrada de 4 bits, con registro de desplazamiento de 6 bits y palabras de 32bits de salida.

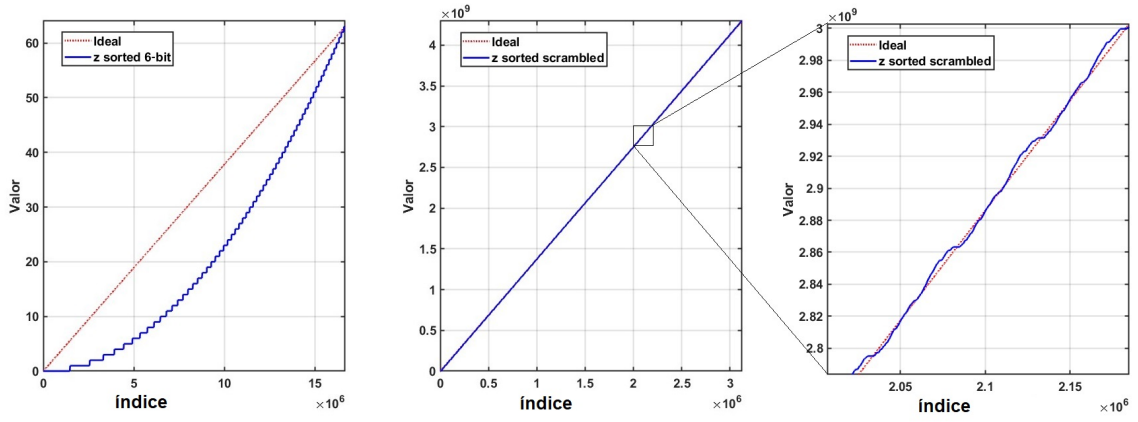


Fig. 16: Resolución de entrada de 6 bits, con registro de desplazamiento de 6 bits y palabras de 32bits de salida.

Se aprecia la gráfica de la función cuadrática con índices caóticos con una resolución de 4 y 6 bits, y su resultado procesado con el scrambler de 6 bits formando palabras de 32 bits. En el resultado procesado se puede apreciar que la función de distribución parece ajustarse mejor a una distribución uniforme, sin embargo los patrones de líneas rectas que aparecen en esta, indican que los los valores generados tienden a repetirse o a tener una alta similitud. Comparando los resultados de las secuencias generadas por fuentes con 4 y 6 bits, se observa que la suavidad de estas funciones de distribución dependen de la resolución en la entrada además del tamaño del registro.

De este modo, las secuencias numéricas procesadas por el scrambler muestran una mejoría en aleatoriedad con respecto a los estados caóticos de entrada como se aprecia en la Figura 17:

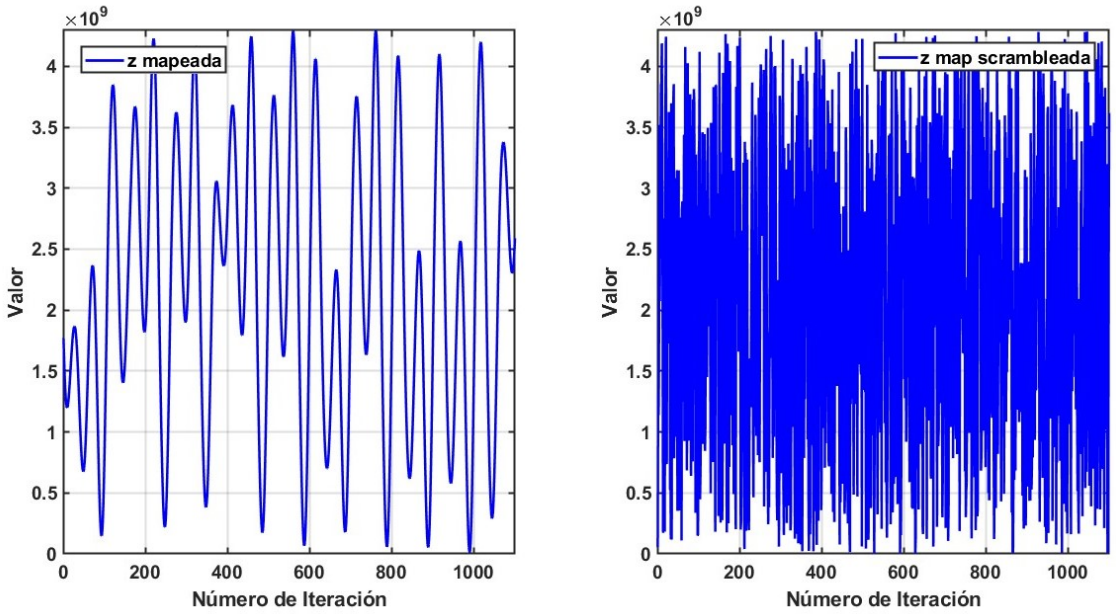


Fig. 17: Incremento en la aleatoriedad de los estados, relacionado con el balance mejorado de 1s y 0s de la secuencia numérica resultante.

3.3 Propuesta de diseño

En el contexto de un entorno de IoT, caracterizado por recursos de hardware limitados, y el objetivo de cifrar y transmitir información proveniente de señales analógicas, la elección de un cifrado simétrico de flujo resulta óptima debido a su eficiencia computacional y baja latencia.

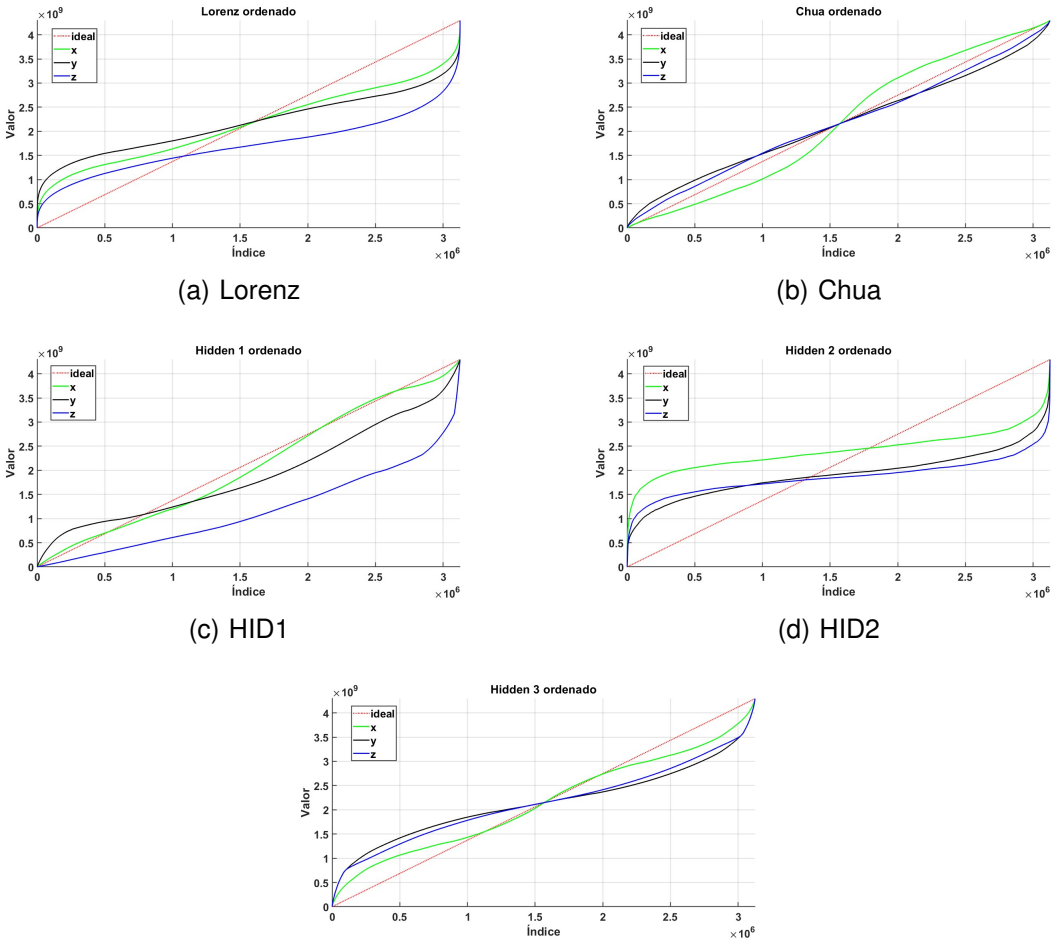
Para la generación del flujo de claves simétricas se define una fuente de entropía mediante algoritmos deterministas basados en caos. El comportamiento del sistema generador aprovecha las propiedades intrínsecas de los sistemas caóticos, tales como la sensibilidad extrema a las condiciones iniciales y su comportamiento aperiódico lo cual asegura que pequeños cambios en la llave

compartida generen secuencias numéricas radicalmente distintas e irrepetibles a largo plazo. Con el uso de algoritmos matemáticos deterministas, se asegura la reproducibilidad de las secuencias numéricas en ambos extremos de la comunicación para crear un cifrado simétrico. Además, la implementación y desarrollo de los estados de los modelos caóticos se facilita en software, especialmente en el microcontrolador RP2040, mediante el método de integración numérica de Euler.

Sin embargo, las secuencias numéricas generadas directamente por un sistema caótico, aunque impredecibles, no presentan un balance adecuado de bits debido a las propiedades del formato IEEE 754 empleado para representar los valores caóticos en punto flotante de tipo double. En la representación IEEE 754 para double, que consta de 1 bit de signo, 11 bits de exponente y 52 bits de mantisa, los valores de los estados caóticos que son del orden de 10^1 exhiben un exponente constante, lo que concentra la variabilidad en la mantisa. Para abordar esta limitación se incorpora una etapa de procesamiento que, en primera instancia, convierte los valores caóticos de formato double o float a enteros sin signo de 32 bits (o una menor resolucion) mediante un mapeo lineal basado en los valores máximo y mínimo de la variable de estado de interés, mejorando significativamente el balance de los bits generados. Los valores caóticos del tipo double, que tienen un orden de magnitud de 10^1, se normalizan y escalan al rango $[0, 2^{32} - 1]$ con la siguiente expresión para la variable de estado x:

$$x_{\text{mapped}} = \left\lfloor \frac{x - x_{\min}}{x_{\max} - x_{\min}} \times (2^{32} - 1) \right\rfloor_{\text{uint32}}$$

Para justificar la elección del modelo caótico del circuito de Chua se verifica la uniformidad en los valores de las variables de estado de cada atractor caótico utilizando 3.125 millones de iteraciones, mismas que también se utilizan para estimar los rangos máximo y mínimo de las variables. Este paso se realiza con una inspección gráfica de las iteraciones ordenadas de menor a mayor comparadas con una linea recta que, en este caso, es análoga a una función de distribución uniforme, también es posible obtener métricas objetivas calculando el área entre las funciones (y considerando la resolución de la fuente caótica).



(a) Lorenz

(b) Chua

(c) HID1

(d) HID2

(e) HID3

Fig. 18: (a-e) Funciones de distribución de los modelos caóticos con formato uint32, su forma sesgada puede relacionarse con la densidad de los valores de las variables de estado, así mismo como con la homogeneidad de los estados en la extensión del atractor caótico.

Se utiliza una segunda etapa de procesamiento mediante un scrambler con un tamaño de registro de 6 bits, lo que incrementa la aleatoriedad y hace más uniforme a la función de distribución de las secuencias binarias generadas. Este tamaño se seleccionó tras considerar los experimentos en [19] donde un registro de desplazamiento de 6 bits minimiza el consumo de recursos y la actividad de conmutación en comparación con configuraciones de mayor longitud. Este enfoque combina la impredecibilidad inherente de los sistemas caóticos con una etapa de refinamiento que asegura la uniformidad de las claves generadas.

3.4 Descripción de las pruebas NIST

El National Institute of Standards and Technology (NIST) desarrolla un set de 15 evaluaciones estadísticas que considera útiles para detectar desviaciones

de un comportamiento aleatorio en una secuencia binaria. En lo práctico, un evaluador podrá apreciar que las aparentes desviaciones de un comportamiento aleatorio pueden deberse a un generador de números mal diseñado o por anomalías en la secuencia binaria que es evaluada, por ejemplo, es de esperar cierto número de fallas en una secuencia binaria producida por un generador en particular.

Existe un número teóricamente infinito de posibles pruebas estadísticas, cada una enfocada en detectar patrones cuya presencia podría sugerir una falta de aleatoriedad, por lo que ningún conjunto de evaluaciones puede considerarse concluyente para certificar la aleatoriedad de una secuencia.

El NIST establece principios para comprender y evaluar la aleatoriedad. Las consideraciones de uniformidad, escalabilidad y consistencia ofrecen un estándar contra el cual medir el comportamiento de las secuencias binarias. Estos criterios proporcionan una base para garantizar que los generadores cumplan con un comportamiento aleatorio:

1) Uniformidad: Durante la generación de una secuencia de bits, ya sea aleatoria o pseudoaleatoria, la probabilidad de que ocurra un cero o un uno debe mantenerse constante e igual a $\frac{1}{2}$ en cada posición. En términos prácticos, esto implica que, para una secuencia de longitud n, el número esperado de ceros (o unos) será $\frac{n}{2}$, reflejando una distribución equilibrada y sin sesgos.
2) Escalabilidad: Las pruebas diseñadas para evaluar la aleatoriedad de una secuencia completa deben ser igualmente aplicables a cualquier subsecuencia extraída de manera aleatoria. Si la secuencia original es genuinamente aleatoria, dichas subsecuencias deberían preservar esta propiedad y superar con éxito las pruebas, validando su aleatoriedad y robustez a diferentes escalas.
3) Consistencia: El desempeño de un generador debe ser uniforme y reproducible a través de distintos valores iniciales o semillas. Evaluar un generador pseudoaleatorio (PRNG) basándose únicamente en la salida de una sola semilla, o un TRNG a partir de una única salida, resulta insuficiente; se requiere un análisis exhaustivo con múltiples inicializaciones para confirmar la fiabilidad y estabilidad del generador.

Las pruebas estadísticas del NIST evalúan la aleatoriedad de secuencias binarias utilizando distribuciones de referencia como la distribución normal estándar y la distribución chi-cuadrado (χ^2), si la secuencia evaluada no es aleatoria, su estadístico calculado se encontrará en las regiones extremas de la distribución de referencia.

En la distribución normal, se calcula un estadístico $z = \frac{x-\mu}{\sigma}$, donde x es el valor observado, y μ y σ son la media y desviación estándar esperadas bajo la hipótesis de un comportamiento aleatorio (H_0).

La distribución χ^2, usada en pruebas de bondad de ajuste (goodness-of-fit), calcula $\chi^2 = \sum \frac{(o_i - e_i)^2}{e_i}$, comparando frecuencias observadas (o_i) y esperadas (e_i).

El NIST destaca que la uniformidad en el número de bits, evaluada por la prueba de frecuencia, es la evidencia primordial de la existencia de aleatoriedad en una secuencia binaria; si esta prueba falla, la probabilidad de que las otras

pruebas fallen es alta. Para algunas de las 15 pruebas se asume que el largo de la secuencia (n) es significativamente grande, típicamente del orden de 10^3 a 10^7, rango para el cual las distribuciones asintóticas de referencia tienen resultados adecuados. Si se evalúan secuencias numéricas para valores más pequeños de n, las distribuciones asintóticas de referencia serían inadecuadas y necesitarían ser reemplazadas por distribuciones exactas cuyo cálculo es complejo y a menudo impracticable. Para secuencias de orden mayor no se menciona algún señalamiento limitante sobre las distribuciones de referencia o la precisión de los resultados; en diversas evaluaciones, para el cálculo de los P-values, el NIST emplea la biblioteca *cephes* que es una colección de funciones y distribuciones matemáticas en lenguaje C diseñada para procesar valores muy grandes con alta precisión [20].

3.4.1 Prueba de frecuencia (monobit): Se evalúa la cercanía/proximidad de la fracción de unos a $\frac{1}{2}$, es decir, el número de unos y ceros en una secuencia debe ser aproximadamente el mismo. La prueba de frecuencia evalúa si una secuencia binaria de longitud (n) tiene una proporción de ceros y unos cercana a $\frac{1}{2}$, como se espera en una secuencia aleatoria. El estadístico de prueba $s_{\text{obs}} = \left|\frac{\sum_{i=1}^{n}(2\varepsilon_i - 1)}{\sqrt{n}}\right|$, mide la desviación normalizada del balance entre ceros y unos, donde ε_i es el (i)-ésimo bit (0 o 1), transformado a ± 11. Para (n) grande, s_{obs} sigue una distribución medio normal (half normal), derivada de la magnitud de una normal estándar. Si la secuencia es aleatoria, los +1 y -1 tienden a cancelarse, dando un s_{obs} cercano a 0. Si hay demasiados ceros o unos, s_{obs} crece, cayendo en los extremos de la distribución, lo que sugiere no aleatoriedad mediante un P-Value bajo.

3.4.2 Prueba de frecuencia dentro de un bloque: La prueba se enfoca en determinar que proporción de 1s dentro de un bloque de bits de tamaño M sea de $\frac{M}{2}$ como sucedería en una secuencia aleatoria. Para bloques de tamaño $M = 1$ esta prueba emula el comportamiento de la prueba de frecuencia (monobit), el valor configurado por defecto es de $M = 128$. Se recomienda que cada secuencia (n) evaluada consista de al menos 100 bits, destacando que $n >= MN$, el tamaño del bloque M debe seleccionarse de tal modo que $M >= 20, M > 0.01n$ y $N < 100$. La distribución de referencia para la evaluación estadística es una distribución χ^2

3.4.3 Prueba de corridas: La prueba evalúa las corridas o rachas en una secuencia n, donde una corrida es una consecución ininterrumpida de bits idénticos. Se determinan series de longitud k de bits idénticos acotadas en los extremos por bits opuestos. En particular, esta prueba determina cuántos son los cambios entre rachas de 1s y rachas de 0s de diversas longitudes en toda la secuencia de n bits. La distribución de referencia para la evaluación estadística es una distribución χ^2

3.4.4 Prueba de corridas largas: El propósito de la prueba es determinar la frecuencia de las rachas más largas dentro de una secuencia binaria de largo n dividida en (N) bloques de largo M, se considera que una irregularidad en el largo esperado de las rachas de 1s implica una irregularidad en el largo esperado de las rachas de 0s por lo que sólo es necesaria la evaluación de los 1s. La evaluación ajusta el tamaño M de los bloques dependiendo del largo de la secuencia binaria n de acuerdo con la siguiente tabla:

Mínimo n	M
128	8
6272	128
$750,000$	10^4

El número de bloques N es dependiente del tamaño de la secuencia binaria n y de las longitudes M por defecto. Se crea un histograma (V_i) con las frecuencias de las rachas de 1s de distinta longitud de acuerdo a las siguientes categorías esperadas para un comportamiento aleatorio:

V_i	$M=8$	$M=128$	$M=10^4$
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

La distribución de referencia para la evaluación estadística es la distribución χ^2

3.4.5 Prueba de rango: La prueba de rango de matrices binarias evalúa el rango de submatrices disjuntas de la secuencia binaria de largo n. Se busca detectar dependencias lineales entre subcadenas de longitud fija en la secuencia original. La evaluación divide la secuencia binaria de largo n en bloques disjuntos de tamaño $M \cdot Q$; donde existirán $N = |\frac{n}{MQ}|$ bloques, cada bloque es una matriz de M filas por Q columnas con tamaño fijo de 32x32 bits, llenando cada fila con bloques sucesivos de Q bits. Los bits sobrantes se descartan y se reportan como no utilizados en el cálculo de cada bloque/matriz N. La prueba determina el rango de las N matrices y mide qué tan bien el número observado de rangos coincide con el esperado bajo la suposición de aleatoriedad mediante/usando la distribución de referencia χ^2.

El NIST ha calculado e insertado en el código de las evaluaciones las probabilidades para el caso $M = Q = 32$, y considera que al menos 38 matrices N

deben ser evaluadas por lo que se requiere una secuencia binaria $n \geq 38(MQ)$, es decir n≥38,912 bits.

3.4.6 Prueba de FFT: El propósito de esta prueba es analizar las alturas de los picos en la Transformada Discreta de Fourier de la secuencia binaria de largo n. Se busca detectar características periódicas (patrones repetitivos cercanos entre sí) que indiquen una desviación de la aleatoriedad. Se evalúa si el número de picos que superan el umbral del 95% es significativamente diferente del 5%. La distribución de referencia para la evaluación estadística es la distribución normal. Se recomienda que cada secuencia tenga un mínimo de 1000 bits ($n \geq 1000$).

3.4.7 Prueba de aproximación no superpuesta: La prueba de aproximación no superpuesta evalúa la frecuencia de aparición de patrones (plantillas binarias predefinidas) de longitud m en una secuencia binaria de largo n. El propósito es detectar si la frecuencia de aparición de estas plantillas se desvía de lo esperado bajo la suposición de aleatoriedad.

La evaluación divide la secuencia binaria de largo n en N bloques de longitud M, donde por defecto $N = 8$ y $M = \lfloor n/N \rfloor$ descartando los bits sobrantes. En cada bloque, se cuenta el número de veces que aparece una plantilla predefinida de m bits, sin superposición; si una plantilla no coincide con la ventana evaluada, la ventana de evaluación desplaza su posición un bit, si la plantilla es coincidente con la ventana evaluada esta desplaza su posición de evaluación m bits. Las frecuencias de coincidencias observadas se comparan con las esperadas usando la distribución de referencia χ^2.

El código en C está configurado con plantillas de tamaño $m = 2, 3, ..., 10$ ($m = 9$ por defecto). Con $N = 8$ por defecto, sujeto a ser alterado, donde se recomienda que el valor de N debe seleccionarse de modo que $N \leq 100$, adicionalmente asegurando que $M > 0.01 \cdot n$ y $N = \lfloor n/M \rfloor$.

3.4.8 Prueba de aproximación superpuesta: La prueba de aproximación superpuesta analiza la frecuencia de aparición de patrones binarios específicos (plantillas) de longitud m en una secuencia binaria de largo n, permitiendo solapamiento entre los segmentos analizados; al igual que en la prueba de aproximación no superpuesta se utiliza una ventana de longitud m para buscar coincidencias con plantillas de m bits. La diferencia entre esta prueba y la anterior, es que cuando se encuentran coincidencias en la ventana evaluada, esta se desplaza sólo un bit de posición antes de reanudar la búsqueda de coincidencias.

La evaluación divide la secuencia binaria de largo n en N bloques de longitud M, con $N = \lfloor n/M \rfloor$ donde por defecto $M = 1032$ (la documentación también indica N=968 por defecto, sin embargo esto no se refleja en el código donde N es dependiente del valor de n/M). Las coincidencias observadas se comparan con las esperadas usando la distribución de referencia χ^2.

Se recomienda que $n \geq 10^6$ (1 millón de bits), y seleccionar $m = 9$ o $m = 10$ para la evaluación.

3.4.9 Prueba de entropía aproximada: El test de entropía aproximada analiza la frecuencia de todos los posibles patrones superpuestos de tamaño m en una secuencia de n bits, comparándolas con las esperadas en una secuencia aleatoria. Compara la frecuencia de patrones de m bits superpuestos entre dos longitudes adyacentes (m y $m+1$) contra lo que se esperaría en una secuencia aleatoria. Utiliza la distribución χ^2 como referencia. Se sugiere seleccionar $m < \lfloor \log_2 n \rfloor - 5$. El código utiliza $m = 10$ por defecto, quedando sujeto a modificación por el evaluador, en función del tamaño n de la secuencia a evaluar.

3.4.10 Prueba de excursión aleatoria: La prueba de excursiones aleatorias analiza las sumas acumuladas parciales de una secuencia binaria ajustada a valores -1 y $+1$, evalúa la frecuencia en la que las sumas parciales toman ciertos valores ($-4, -3, -2, -1, +1, +2, +3, +4$) dentro de ciclos definidos por un cruce por 0. La prueba compara las frecuencias observadas con las esperadas (valores diseñados por el NIST), generando un p-valor para cada uno de los 8 estados. La prueba determina la secuencia evaluada de largo n cuente con al menos 500 cruces por cero para aplicar el test, de otro modo asigna un p-valor de 0 y no aplica el test, discriminando de la evaluación la el bloque que no cumple este requisito. La distribución de referencia para la prueba estadística es la distribución χ^2. Se recomienda que cada secuencia/bloque a ser evaluada consista de al menos 1 millón de bits ($n \geq 10^6$).

3.4.11 Prueba de variantes de excursión aleatoria: La prueba de variantes de excursiones aleatorias evalúa el número total de visitas a cada estado en un camino aleatorio de sumas acumulativas. Su objetivo es detectar desviaciones del número esperado de visitas a estados con valores de -9 a $+9$ exceptuando el estado 0. La prueba determina dieciocho p-valores, uno para cada estado, discriminando de la evaluación el bloque que tenga menos de 500 cruces por cero. La distribución de referencia para la prueba estadística es la distribución medio normal. Se recomienda que cada bloque a ser evaluado consista de al menos 1 millón de bits ($n \geq 10^6$).

3.4.12 Prueba de suma acumulada: La prueba de sumas acumuladas evalúa la máxima desviación (z) desde 0 de la suma acumulada de valores ajustados a $-1, +1$ en la secuencia evaluada de largo n. Se realiza la suma parcial de subsecuencias de distintos largos desde 1 hasta n y se determina la suma parcial (S_k) con el valor absoluto más alto para estimar un valor estadístico ($z = max_{1 \leq k \leq n} |S_k|$), en secuencias aleatorias, el valor máximo absoluto de sumas parciales (z) deberían ser cercano a cero, mientras que en secuencias no aleatorias pueden ser grandes; divergen del balance 0. La prueba estima de dos modos las sumas acumuladas de la secuencia de largo n; iniciando la

sumatoria desde el bit más significatiivo hasta el menos significativo (forward), e iniciando desde el bit menos significativo hasta el más significativo (backward), obteniendo resultados distintos para ambos casos, mismos que se reflejan en los resultados del test. La distribución de referencia para la evaluación estadística es la distribución normal. Se recomienda que cada secuencia (de largo n) a ser evaluada tenga al menos 100 bits.

3.4.13 Prueba Serial: La prueba serial evalúa la frecuencia de todos los posibles patrones superpuestos de m bits en una secuencia de longitud n igual que en la prueba de entropía aproximada. Se determina si el número de ocurrencias de patrones superpuestos de 2^m m-bits es el esperado para una secuencia aleatoria, donde una secuencia aleatoria tiene uniformidad y cualquier patrón de largo m tiene la misma probabilidad de coincidir con el bloque de largo m evaluado. Para el caso $m = 1$, esta prueba tiene el mismo comportamiento que la prueba de frecuencia. Se utiliza la distribución χ^2 como referencia. Se recomienda elegir $m < \lfloor \log_2 n \rfloor - 2$. El código de la evaluación utiliza $m = 16$ por defecto quedando sujeto a modificarse por el usuario en función del largo n de la secuencia binaria evaluada.

3.4.14 Prueba de complejidad lineal: La prueba de complejidad lineal evalúa si una secuencia binaria de longitud n puede ser generada por un registro de desplazamiento lineal (LFSR) corto, lo que sugeriría no aleatoriedad. La secuencia se divide en $N = \lfloor n/M \rfloor$ bloques de tamaño M, descartando los bits sobrantes. Se calcula la complejidad líneal L para cada bloque N mediante el algoritmo de Berlekamp-Massey, que determina la longitud del LFSR más corto capaz de producir la secuencia del bloque N. El estadístico resultante sigue una distribución χ^2.

Se recomienda utilizar secuencias binarias $n \geq 10^6$. El valor de M debe estar en el rango de $500 \leq M \leq 5000$, y con $N \geq 200$, para garantizar robustez estadística. La prueba asigna $M = 500$ por defecto.

3.4.15 Prueba universal de Maurer: La prueba Universal de Maurer evalúa si una secuencia binaria de largo n puede ser comprimida significativamente sin pérdida de información, donde se le considera no aleatoria si es significativamente compresible, se analiza la distancia entre repeticiones de patrones de L bits. La secuencia se divide en dos segmentos: uno de inicialización con Q bloques de L bits y otro de prueba con K bloques de L bits, descartando los bits sobrantes.

Se crea un arreglo para cada posible permutación de L bits y se acumula el número de coincidencias de los Q bloques de largo L del segmento de inicialización ($Q \cdot L$) en su correspondiente celda del arreglo. En el segmento de prueba ($K \cdot L$), para cada bloque K se calcula la distancia en bloques desde su última aparición, acumulando la suma de $\log_2$ de estas distancias para obtener

un estadístico que mide la compresibilidad, el valor estadístico se compara con la distribución de referencia semi-normal.

Esta evaluación requiere una secuencia de n bits significativamente larga donde $n \geq (Q+K)L$, se recomienda que L esté entre 6 y 16, con $Q = 10 \cdot 2^L$ y $K \approx 1000 \cdot 2^L$, es decir $n \geq (1010 \cdot 2^L)L$. El código redefine el valor de L según la longitud n de la secuencia evaluada como se muestra en la tabla:

n mínimo	L	$Q = 10 \cdot 2^L$
$387,840$	6	640
$904,960$	7	$1,280$
$2,068,480$	8	$2,560$
$4,654,080$	9	$5,120$
$10,342,400$	10	$10,240$
$22,753,280$	11	$20,480$
$49,643,520$	12	$40,960$
$107,560,960$	13	$81,920$
$231,669,760$	14	$163,840$
$496,435,200$	15	$327,680$
$1,059,061,760$	16	$655,360$

3.5 Interpretación de resultados de las pruebas

El NIST ha adoptado dos procedimientos para la interpretación de los resultados empíricos, el primero es examinando las proporciones de bloques que aprueban una evaluación estadística, y el segundo método es examinando la distribución de los p-valores de los bloques para revisar la uniformidad de estos.

En el análisis de proporciones aprobadas se estima un intervalo de confianza que se calcula como $\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$ donde $\hat{p} = 1-\alpha$ y m es el tamaño del bloque evaluado; el límite inferior de este intervalo indica el mínimo de proporciones que deben tener un p-valor mayor al nivel de sinificancia α (0.01) para que la secuencia evaluada pueda considerarse aleatoria. Sobrepasar el intervalo de confianza acercándose a una proporción de 1 sólo indica una alta aleatoriedad y no sugiere anomalías en la aleatoriedad. Por ejemplo, al evaluar mil bloques $m = 1000$, el nivel de significancia es $0.99 \pm 3\sqrt{\frac{0.99(0.01)}{1000}} = 0.99 \pm 0.0094392$, indicando un límite inferior de 0.9805607 por lo que al menos 981 bloques evaluados deben tener p-valores mayores a α para considerar que no existen anomalías en la secuencia binaria evaluada.

La uniformidad de los p-valores se examina dividiendo el intervalo $[0,1]$ de los p-valores de cada bloque en 10 subintervalos donde se cuentan los p-valores que caen en cada intervalo, al igual que en un histograma. Los p-valores con valor 0 se asignan en el primer intervalo $[0,0.1]$, con esto se obtiene un p-valor total de todos los p-valores de cada bloque evaluado. Para evaluar la uniformidad

de los p-valores se aplica una prueba χ^2 con el estadístico $\chi^2 = \sum_{i=1}^{10} \frac{\left(F_i - \frac{s}{10}\right)^2}{\frac{s}{10}}$, donde F_i es el conteo en el subintervalo i y s es el número de bloques. El P-value$_T$ = igamc $\left(\frac{9}{2}, \frac{\chi^2}{2}\right)$ debe ser ≥ 0.0001 para confirmar la uniformidad. Se requieren al menos 55 bloques para resultados estadísticamente significativos.

Para la prueba de entropía aproximada, para una secuencia (n) de 1millón de bits, utilizar la relación $m = log_2 n$ resulta en m=19, lo cual no es un valor de parámetro adecuado porque evidencia empírica sugiere que más allá de $m = 14$ los resultados estadísticos no concuerdan con los valores esperados, particularmente para buenos generadores de números. El NIST sugiere evaluar bloques de tamaño pequeño para esta prueba [20].

El NIST realiza evaluaciones de ejemplo de 1 bloque de 1millón de bits para las expansiones de π, $\sqrt{2}$,e, etc.

3.6 Resultados de las evaluaciones estadísticas

Prueba	Chua						Lorenz					
	x		y		z		x		y		z	
	p-val	prop	p-val	prop	p-val	prop	p-val	prop	p-val	prop	p-val	prop
Frequency	0.9781	99/100	0.7197	100/100	0.4190	98/100	0.2493	99/100	0.8165	99/100	0.4944	99/100
BlockFrequency	0.1917	98/100	0.2133	99/100	0.5341	99/100	0.0083	99/100	0.3345	99/100	0.2133	97/100
CumulativeSums 1	0.9558	99/100	0.3669	100/100	0.5544	99/100	0.6993	98/100	0.2757	100/100	0.0288	98/100
CumulativeSums 2	0.7197	99/100	0.4012	99/100	0.4559	98/100	0.0376	99/100	0.2368	97/100	0.7981	98/100
Runs	0.8165	100/100	0.4750	100/100	0.5341	100/100	0.2897	100/100	0.0179	99/100	0.9943	100/100
LongestRun	0.3041	99/100	0.1372	100/100	0.1154	98/100	0.0054	95/100*	0.0269	98/100	0.1719	97/100
Rank	0.0000*	93/100*	0.3504	98/100	0.6163	98/100	0.0000*	1/100*	0.0000*	0/100*	0.0000*	93/100*
FFT	0.6163	99/100	0.3504	99/100	0.5956	97/100	0.4559	98/100	0.6579	100/100	0.2757	99/100
NonOverlpTemplate	0.3505	99/100	0.2757	99/100	0.0590	100/100	0.5141	98/100	0.7197	99/100	0.8832	98/100
OverlappingTemplat	0.2757	99/100	0.0102	97/100	0.2757	99/100	0.0000*	92/100*	0.0000*	97/100	0.0805	97/100
Universal	0.2023	100/100	0.6993	99/100	0.5955	100/100	0.4373	98/100	0.1025	98/100	0.9463	99/100
ApproximateEntrop	0.0000*	70/100*	0.0000*	0/100*	0.0000*	56/100*	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*
RandomExcursions	0.0628	56/57	0.6890	60/61	0.7598	57/57	0.6371	60/60	0.1347	64/64	0.8677	56/57
RExcursionsVariant	0.0167	55/57	0.7231	61/61	0.8977	56/57	0.2993	58/60	0.4373	64/64	0.0712	57/57
Serial 1	0.0009*	99/100	0.0000*	89/100*	0.0352	95/100*	0.0000*	4/100*	0.0000*	19/100*	0.0000*	39/100*
Serial 2	0.9879	100/100	0.4012	99/100	0.9114	99/100	0.8514	100/100	0.6163	100/100	0.3838	99/100
LinearComplexity	0.1538	99/100	0.9942	100/100	0.2368	100/100	0.7981	100/100	0.9558	98/100	0.5544	100/100

Prueba	Hid1						Hid2					
	x		y		z		x		y		z	
	p-val	prop	p-val	prop	p-val	prop	p-val	prop	p-val	prop	p-val	prop
Frequency	0.9642	100/100	0.8165	99/100	0.2368	98/100	0.0077	85/100*	0.0000*	63/100*	0.0000*	68/100*
BlockFrequency	0.0029	99/100	0.2248	99/100	0.0000*	2/100*	0.0000*	83/100*	0.0000*	62/100*	0.0000*	69/100*
CumulativeSums 1	0.5749	99/100	0.4943	98/100	0.3838	100/100	0.0628	86/100*	0.0000*	1/100*	0.0000*	70/100*
CumulativeSums 2	0.2757	100/100	0.5955	99/100	0.1816	99/100	0.0006	86/100*	0.0000*	62/100*	0.0000*	67/100*
Runs	0.3504	98/100	0.9114	100/100	0.9558	99/100	0.0000*	76/100*	0.0000*	53/100*	0.0000*	51/100*
LongestRun	0.2492	99/100	0.4749	98/100	0.0000*	88/100*	0.0000*	81/100*	0.0000*	56/100*	0.0000*	76/100*
Rank	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*
FFT	0.2368	99/100	0.7597	99/100	0.0000*	92/100*	0.0000*	72/100*	0.0000*	49/100*	0.0000*	36/100*
NonOverlpTemplate	0.5955	98/100	0.5341	99/100	0.0000*	92/100*	0.0000*	52/100*	0.0000*	42/100*	0.0000*	36/100*
OverlappingTemplat	0.7791	99/100	0.0805	100/100	0.0000*	66/100*	0.0000*	77/100*	0.0000*	51/100*	0.0000*	69/100*
Universal	0.1537	98/100	0.8676	100/100	0.0000*	96/100	0.0000*	72/100*	0.0000*	47/100*	0.0000*	37/100*
ApproximateEntrop	0.0000*	71/100*	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*	0.0000*	1/100*	0.0000*	0/100*
RandomExcursions	0.3669	54/55	0.3781	61/64	0.0956	65/65	0.4221	60/63	0.7887	39/40	0.6371	54/55
RExcursionsVariant	0.0711	53/55	0.1109	63/64	0.0956	64/65	0.8755	60/63	0.2932	40/41	0.2133	54/54
Serial 1	0.2022	99/100	0.0000*	84/100*	0.0000*	0/100*	0.0000*	4/100*	0.0000*	8/100*	0.0000*	3/100*
Serial 2	0.9642	98/100	0.3504	99/100	0.7981	99/100	0.0000*	52/100*	0.0000*	47/100*	0.0000*	37/100*
LinearComplexity	0.2896	99/100	0.1373	99/100	0.8978	100/100	0.0000*	74/100*	0.0000*	50/100*	0.0000*	37/100*

Prueba	Hid3					
	x		y		z	
	p-val	prop	p-val	prop	p-val	prop
Frequency	0.9879	98/100	0.7399	97/100	0.0628	98/100
BlockFrequency	0.0669	97/100	0.2023	100/100	0.7981	100/100
CumulativeSums 1	0.9463	98/100	0.2493	97/100	0.1626	97/100
CumulativeSums 2	0.9963	98/100	0.4012	97/100	0.0909	98/100
Runs	0.5341	99/100	0.8343	98/100	0.3838	100/100
LongestRun	0.8677	100/100	0.0428	99/100	0.1296	99/100
Rank	0.7792	97/100	0.0000*	72/100*	0.0401	99/100
FFT	0.3191	98/100	0.7792	99/100	0.7197	98/100
NonOverlpTemplate	0.3505	99/100	0.6787	99/100	0.2023	98/100
OverlappingTemplate	0.0001	95/100	0.0004	96/100	0.0179	97/100
Universal	0.4559	100/100	0.7197	99/100	0.7981	99/100
ApproximateEntropy	0.0000*	0/100*	0.0000*	0/100*	0.0000*	0/100*
RandomExcursions	0.2041	65/67	0.1866	67/67	0.2133	69/70
RExcursionsVariant	0.2645	66/67	0.4220	67/67	0.4242	70/70
Serial 1	0.0000*	65/100*	0.0000*	80/100*	0.0000*	78/100*
Serial 2	0.8514	100/100	0.0252	100/100	0.8165	99/100
LinearComplexity	0.7981	99/100	0.4559	99/100	0.7197	99/100

4 Implementación en hardware

4.1 Características del RP2040 utilizadas para el sistema

Multicore: el RP2040 integra dos núcleos ARM Cortex-M0+ que operan a una frecuencia máxima de 133 MHz. Estos núcleos están conectados a un bus AHB-Lite completamente cruzado, lo que permite accesos simultáneos a la memoria y periférico. Esta arquitectura multicore habilita la ejecución paralela de tareas, como la generación de claves en un núcleo y la gestión de comunicación y cifrado en el otro.

UART: el microcontrolador cuenta con dos controladores UART. Cada UART incluye FIFOs separados de 32×8 para transmisión (Tx) y 32×12 para recepción (Rx), su tasa de baudios es configurable y puede usar control de flujo por hardware que proporciona una interfaz para conectarse al controlador DMA. Estas características facilitan la transferencia eficiente de datos en aplicaciones que requieren comunicación serial, como la conexión con módulos externos.

ADC: el RP2040 dispone de un ADC de 12 bits con cuatro canales analógicos multiplexados. Este ADC tiene una tasa de muestreo máxima de 500 ksps. Admite solicitudes de DMA (DREQ) para transferir datos directamente a la memoria, reduciendo la carga en los núcleos. Por defecto (DIV = 0), las nuevas conversiones comienzan inmediatamente después de que finaliza la conversión anterior, produciendo una nueva muestra cada 96 ciclos. A una frecuencia de reloj de 48 MHz, esto genera 500 ksps.

Canales DMA: El canal de Acceso Directo a Memoria del RP2040 realiza transferencias masivas de datos en nombre del procesador. Esto libera a los procesadores para realizar otras tareas. La salida de datos de un canal DMA es significativamente mayor que el de los procesadores del RP2040. El DMA puede realizar un acceso de lectura y uno de escritura, de hasta 32 bits, en cada ciclo de reloj. Hay 12 canales independientes, cada uno supervisando una secuencia de transferencias de bus, generalmente en uno de los siguientes escenarios:

1) Memoria a periférico: un periférico señala al DMA cuando necesita más datos para transmitir. El DMA lee datos de un arreglo en RAM o flash y escribe en el FIFO de datos del periférico.
2) Periférico a memoria: un periférico señala al DMA cuando ha recibido datos. El DMA lee estos datos del FIFO del periférico y los escribe en un arreglo en RAM.
3) Memoria a memoria: el DMA transfiere datos entre dos búferes en RAM lo más rápido posible.

El tamaño de transferencia puede ser de 32, 16 u 8 bits. Esto se configura una vez por canal cuando el tamaño de transferencia de origen y destino es el mismo en cada transferencia.

Los canales DMA pueden combinarse de varias maneras para un comportamiento más sofisticado y mayor autonomía. Por ejemplo, un canal DMA puede reconfigurar a otro canal DMA, cargando datos de configuración desde una secuencia de bloques de control en memoria, y el segundo puede volver a llamar al primero mediante la opción CHAIN_TO cuando necesita reconfigurarse. Hacer que el DMA sea más autónomo implica que se requiere menos supervisión del procesador: en general, esto permite que el sistema haga más a la vez o consuma menos energía. Cada canal DMA puede generar una interrupción al completar una transferencia, estas interrupciones permiten al software (intervención del core) responder rápidamente a la finalización de una transferencia de datos. Al configurar el bit correcto en INTE0 o INTE1, podemos indicar al DMA que genere una de sus dos líneas de solicitud de interrupción cuando un canal determinado finaliza.

Cálculo de CRC: El DMA puede observar los datos de un canal que pasan por el FIFO de datos y calcular sumas de verificación basadas en estos datos. Esto es completamente pasivo: el hardware no altera los datos, solo los observa. Las sumas de verificación soportadas son:

1) CRC-32, MSB primero y LSB primero.
2) CRC-16-CCITT, MSB primero y LSB primero.
3) Suma simple (adición a un acumulador de 32 bits).
4) Paridad par.

Atomic XOR: El RP2040 ofrece soporte para operaciones atómicas, incluyendo XOR, en ciertos registros de hardware. Estas operaciones permiten realizar manipulaciones bit a bit rápidas y confiables, ideales para procesos como el cifrado de datos. Cada bloque de registros de periféricos tiene asignado 4 kB de espacio de direcciones, con los registros accedidos mediante uno de cuatro métodos, seleccionados por decodificación de direcciones:

1) Dirección + 0x0000: acceso normal de lectura y escritura.
2) Dirección + 0x1000: XOR atómico en escritura.
3) Dirección + 0x2000: establecimiento de máscara de bits atómica en escritura.
4) Dirección + 0x3000: borrado de máscara de bits atómica en escritura.

Esto permite modificar campos individuales de un registro de control sin realizar una secuencia de lectura-modificación-escritura en software: en su lugar, los cambios se envían al periférico y se realizan in situ [21].

4.2 Caracteristicas de los módulos de RF Xbee

Los módulos de radio frecuencia (RF) XBee implementan comunicaciones inalámbricas en la frecuencia de 2.54GHz, son configurables y soportan múltiples protocolos de comunicación. Ideales para transmitir información entre dos

máquinas o diseñar una red de comunicación de datos con múltiples dispositivos entre coordinadores, routers y end devices.Zigbee que es un estándar de red, gracias a sus capacidades de ruteo, los nodos de RF permiten la transmisión de datos en largas distancias mediante el intercambio de frames en una red con nodos intermediarios (routers) para alcanzar nodos distantes. Los rangos de distancia de transmisión van desde 1200 a 3200 metros bajo la línea de visión (sin obstáculos) y las redes con direccionamiento pueden contener hasta 65mil (65535 distintas direcciones físicas) nodos por red [22]. El modo API de los XBee provee una interfaz etructurada donde la información es transmitida a través de la interfaz serial en frames organizados y en un determinado orden.

Delimitador de Inicio	Longitud		Tipo de frame	Datos							Suma de Verificación
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	Tipo de API frame	Datos del frame							Byte único

El delimitador de inicio es el primer byte en un API frame que consiste de una secuencia especial de bits que indican el inicio de un frame, su valor es siempre 0x7E, esto permite una fácil detección de un nuevo frame entrante. Los bytes 2 y 3 del API frame indican el número de bytes en un frame excluyendo 4bytes correspondientes al delimitador de inicio, 2 bytes de longitud y 1 byte de checksum. El tipo de API frame determina cómo está organizada la información en los bytes de datos. Los bytes de datos contienen la información recibida o a ser transmitida, estos bytes incluyen opciones de la solicitud, la dirección física del destinatario o remitente, radio de difusión entre nodos, opciones de transmisión, así como los bytes de información a transmitir. El checksum es el último byte del frame y ayuda a evaluar la integridad de los datos. Es una suma de verificación entre todos los bytes que le anteceden excluyendo los primeros 3 bytes del API frame. Los frames enviados a través de la interfaz serial con sumas de comprobación incorrectas son ignorados.

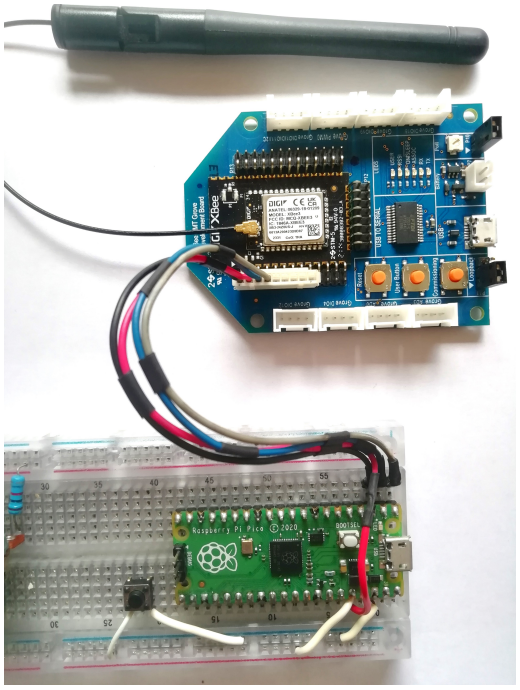
4.3 Sistema de cifrado simétrico implementado

El sistema de cifrado simétrico a implementar se basa en el circuito de Chua en conjunto con un scrambler para producir secuencias numéricas pseudoaleatorias que cifren la información capturada por el conversor analógico digital (ADC) del microcontrolador RP2040. En el dispositivo emisor, un núcleo del microcontrolador generará claves pseudoaleatorias mientras el otro núcleo manejará las interrupciones de los canales de acceso directo a memoria (DMA) que capturan, cifran, generan API frames y transmiten la información hacia módulos de RF XBee3 configurados en una red Ad Hoc. En el lado del receptor, el uso del doble núcleo y canales DMA harán posible que la adquisición,

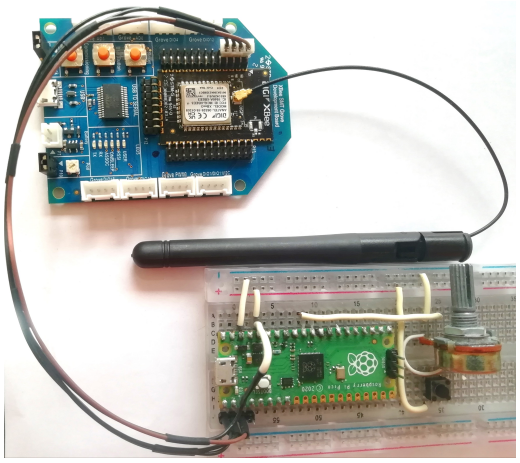
validación y descifrado de los API frames se realice sin inconvenientes (procesos paralelos con tasa de descifrado> tasa de adquisición). El sistema propuesto resguarda la integridad de la información y mantiene la sincronización de claves de cifado entre ambos nodos. Las capacidades de doble núcleo permiten el manejo de procesos paralelos mientras los canales DMA crean un flujo continuo de datos sin intervención constante de los procesadores ayudando a mejorar la tasa de cifrado-descifrado y transmisión-recepción.

4.4 Montaje del Hardware y pruebas de recepción

En las siguientes imágenes se aprecia el hardware del sistema. En el lado receptor tenemos al módulo XBee con su correspondiente antena, se conecta a la tarjeta Raspberry Pi Pico con 4 cables: 2 para la alimentación y 2 para RX y TX. En el lado emisor se tiene la misma configuración pero que añade un potenciómetro el cual nos brinda la señal analógica que se cifra.



(a) Lado receptor.



(b) Lado emisor.

En la figura a continuación tenemos una pantalla del entorno de desarrollo de los módulos XBee llamado XCTU, en esta interfaz es posible configurar los modos de operación de los módulos de radio frecuencia así como realizar pruebas de envío o, como en este caso, una prueba de recepción. Donde el XBee del lado receptor está mostrando en pantalla los paquetes de datos recibidos. Del lado izquierdo en color rojo se aprecian numerados los paquetes que el sistema cuenta como recibidos, estos tienen una estampa de tiempo donde se aprecia la baja latencia del sistema implementado. Del lado derecho en color negro se aprecia a detalle el contenido del paquete #1220, el paquete

cuenta con la dirección física del remitente. Un punto importante a destacar es el contenido de datos del paquete en el campo ”RF data” encontramos un total de 13 bytes, el primero corresponde a un byte no cifrado que enumera al paquete para mantener sincronía en el proceso de descifrado, y los 12 bytes restantes corresponden a 12 bytes de información que está cifrada.

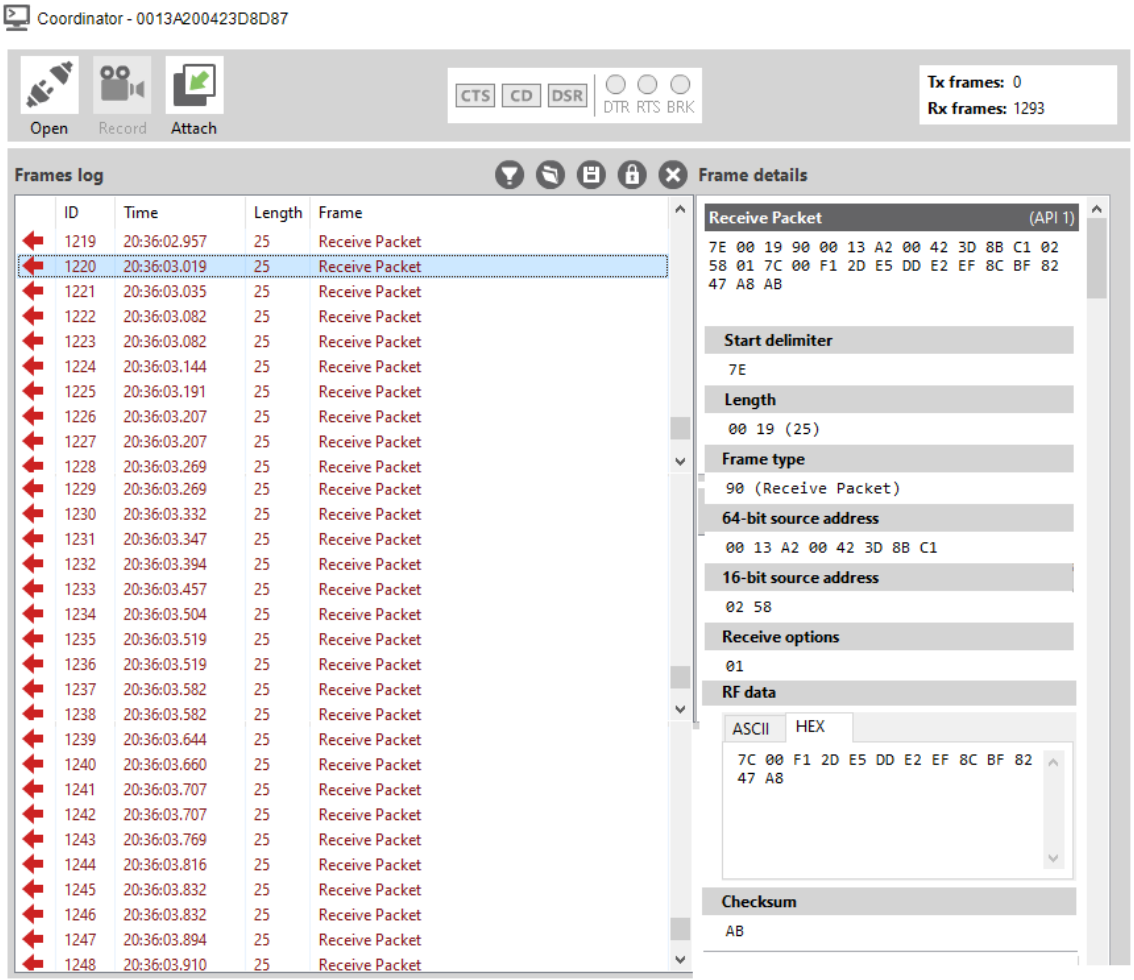


Fig. 20: Pruebas de recepción en XCTU.

Estos paquetes recibidos con la información cifrada, el XBee receptor se los entrega a la Raspberry Pi Pico por uno de sus puertos de comunicación serial para que descifre la información, y finalmente la Raspberry Pi Pico envía los bytes descifrados por un puerto serial para su visualización en pantalla. La máxima tasa de transferencia de datos que se alcanzó fue de 1KB por segundo; limitado por los módulos de RF.

4.5 Diagramas de flujo

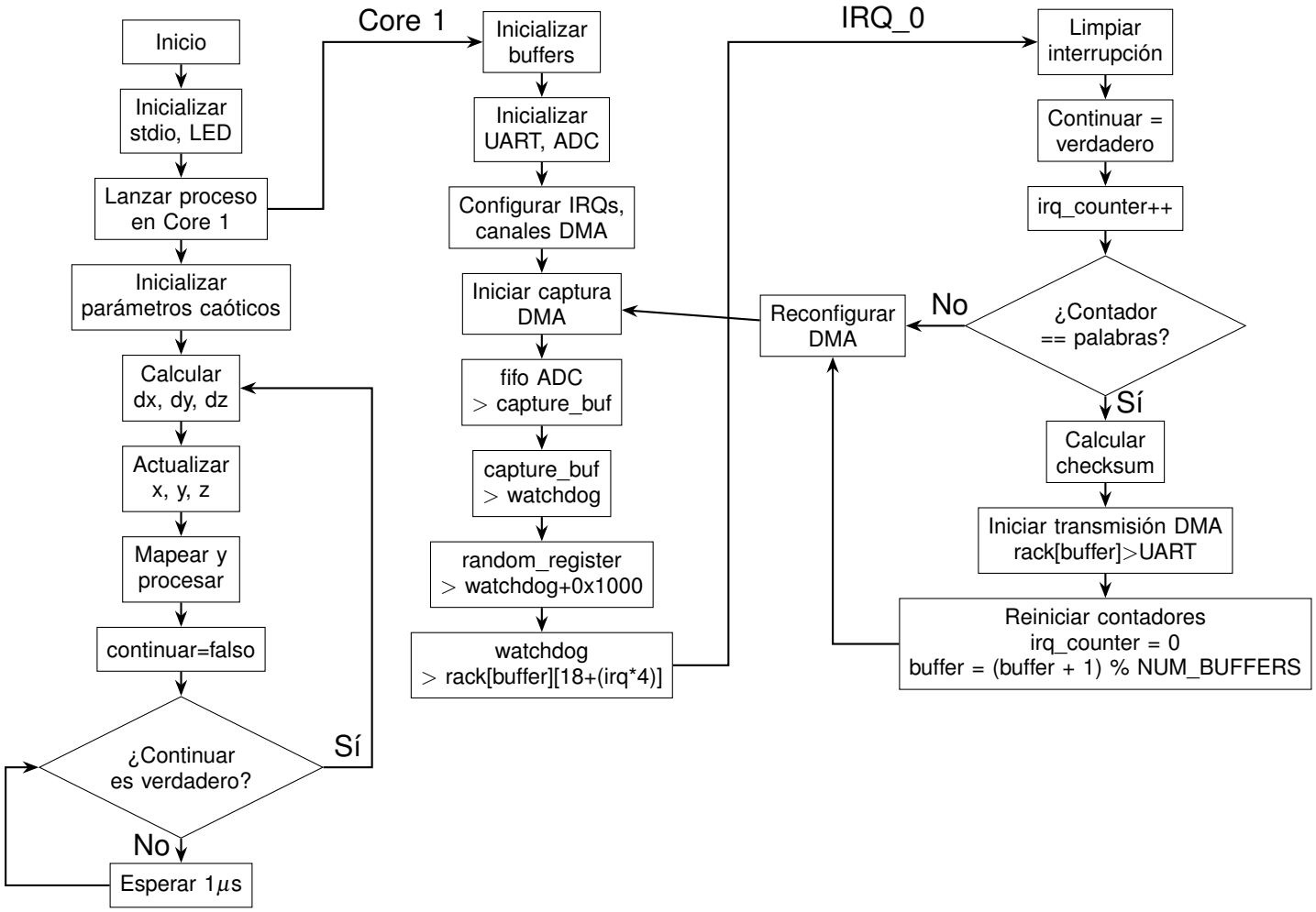


Fig. 21: Diagrama de flujo del sistema emisor, mostrando las operaciones en Core0, Core1, canales DMA para cifrado mediante XOR atómico y el manejador de interrupciones DMA.

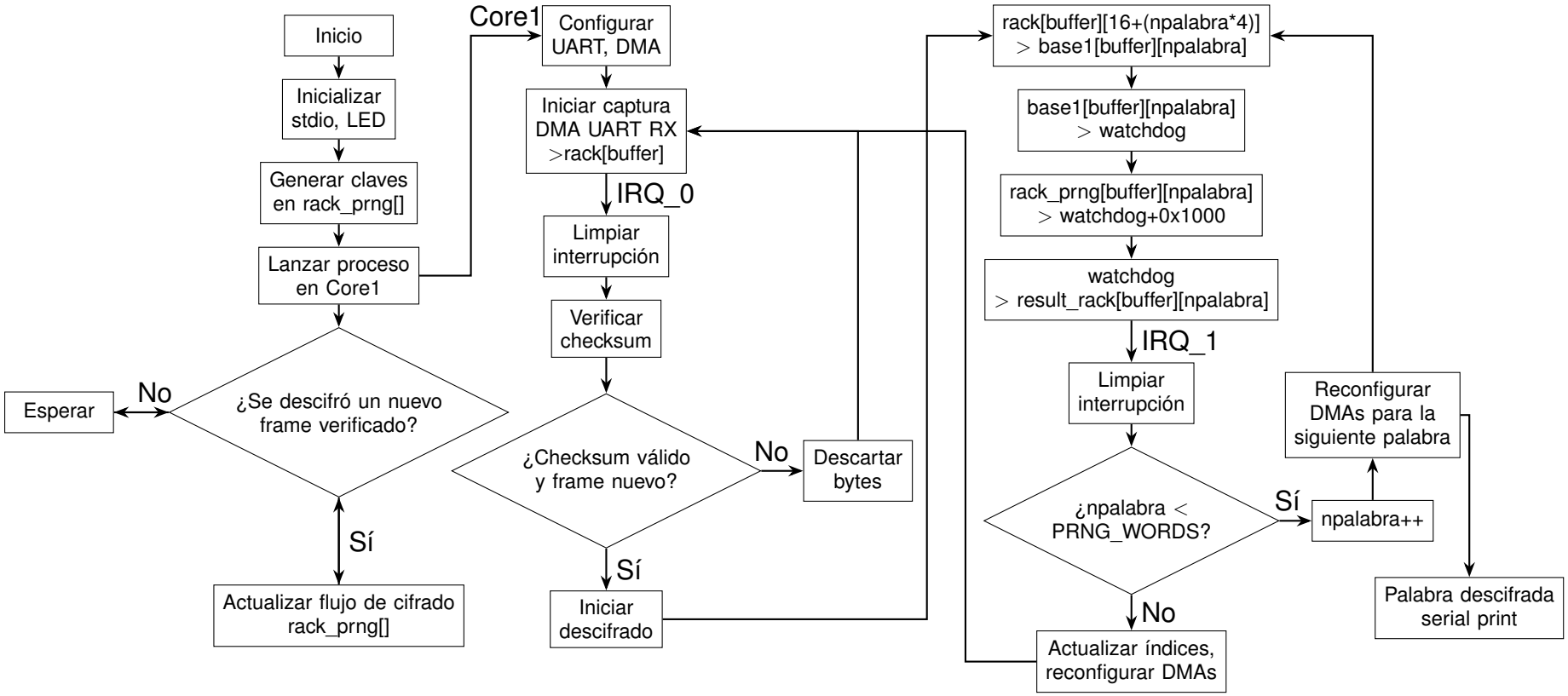


Fig. 22: Diagrama de flujo del sistema receptor, mostrando las operaciones en Core0, Core1 con canales DMA para descifrado mediante XOR atómico y los manejadores de interrupciones DMA.

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Este trabajo no solo valida la utilidad práctica de la teoría del caos en la generación de números pseudoaleatorios, sino que también explora cómo la aleatoriedad inherente de los sistemas caóticos puede robustecer la seguridad en la transmisión de datos. Un aporte distintivo radica en el estudio de las secuencias numéricas ordenadas de los sistemas caóticos y el efecto de refinamiento producido por etapas de postprocesamiento. Este proceso ha culminado en la formulación de un método para producir secuencias numéricas seguras, diseñadas para aplicaciones criptográficas en sistemas IoT.

El algoritmo generador de números implementado en esta tesis, ha demostrado tener buenos resultados en las evaluaciones estadísticas del NIST lo que valida la aplicación de las claves generadas en sistemas de seguridad. Las evaluaciones estadísticas demuestran que los algoritmos de cifrado basados en caos ofrecen un nivel de seguridad adecuado para sistemas IoT al aprobar prácticamente las 15 evaluaciones estadísticas, confirmando que los números generados por nuestro sistema tienen una función de distribución que se ajusta a la uniforme.

La naturaleza determinista, pero impredecible a largo plazo, de nuestro generador de números ha demostrado una reproducibilidad de las claves de cifrado en ambos lados de la comunicación para mantener una sincronía adecuada. En conjunto con el algoritmo de intercambio de paquetes desarrollado en código C, propician que el sistema puede mantener la sincronía en el descifrado de paquetes. La enumeración de los paquetes ha demostrado ser un método adecuado para mantener una secuencia de descifrado ordenada en los paquetes que se reciben y descifran, debido a que en ocasiones los paquetes llegan al destinatario con un retraso, es decir; paquetes posteriores llegan antes que los primeros paquetes transmitidos, o no llegan; se pierden.

La combinación de robustez criptográfica, eficiencia energética y hardware limitado, posiciona a este tipo de soluciones como un avance favorable en el fértil campo de la ciberseguridad para tecnologías IoT.

El empleo de algoritmos caóticos en el cifrado de datos abre una prometedora vía de investigación con potencial de transformar la seguridad que pueden ofrecer los generadores de números pseudoaleatorios, al considerar que las fuentes de entropía, definidas por sistemas dinámicos no lineales con parámetros convergentes a caos presentan una función de distribución sesgada en cada una de sus variables de estado. Lo que puede conducir a definir nuevos sistemas dinámicos convergentes a caos que cuenten con alguna variable de estado con una función de distribución muy cercana a la uniforme.

En vista de que la mejoría de las características estadísticas del sistema dependen mucho del número de celdas del registro de desplazamiento y de la resolución de la fuente de entropía, es posible determinar el mínimo de estos parámetros (tamaño de registro y resolución de la fuente) para hacer que una secuencia generada que presenta cierto sesgo en su función de distribución mejore sus cualidades estadísticas hasta cierto nivel.

5.2 Trabajo Futuro

- Buscar incrementar la velocidad de transmisión de los módulos de radio frecuencia XBee3.
- Realizar pruebas de transmisión-recepción con datos reales en un ambiente controlado a distintas distancias.

References

[1] M. Frustaci, P. Pace, G. Aloi, G. Fortino, Evaluating critical security issues of the IoT world: Present and future challenges, IEEE Internet of Things Journal, 5 (4) (2018) 2483–2495.

[2] D. Clemente-López, J.M. Munoz-Pacheco, J.J. Rangel-Magdaleno, Experimental validation of IoT image encryption scheme based on a 5-D fractional hyperchaotic system and Numba JIT compiler, Internet of Things, Available at: www.elsevier.com/locate/iot.

[3] S. H. Strogatz, Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering, CRC Press, Boca Raton, FL, 2018.

[4] D. G. Zill, W. S. Wright, Ecuaciones diferenciales con problemas con valores en la frontera, 8th ed., Cengage Learning, México, D.F., 2015.

[5] R. Rocha, R. O. Medrano-T., Stability analysis for the Chua circuit with cubic polynomial nonlinearity based on root locus technique and describing function method, Nonlinear Dyn. 102 (2020) 2859–2874.

[6] G.-Q. Zhong, Implementation of Chua's circuit with a cubic nonlinearity, IEEE Trans. Circuits Syst. I, Fundam. Theory Appl. 41 (12) (1994).

[7] K. M. Owolabi, A. Atangana, Chaotic behaviour in system of noninteger-order ordinary differential equations, Chaos Solitons Fractals 115 (2018) 362–370. https://doi.org/10.1016/j.chaos.2018.07.034

[8] A. Al-Husban et al., Chaos in a two dimensional fractional discrete Hopfield neural network and its control, Alexandria Eng. J. 75 (2023) 627–638.

[9] O. Salhab et al., Survey paper: Pseudo random number generators and security tests, J. Theor. Appl. Inf. Technol. 96 (7) (2018) 1859–1872.

[10] https://www.nationalgeographic.com.es/ciencia/criptografia-ciencia-que-protege-era-digital_21781

[11] F. Yu et al., A survey on true random number generators based on chaos, Discrete Dyn. Nat. Soc. 2019 (2019) 2545123, 10 pages. https://doi.org/10.1155/2019/2545123

[12] R. Trejo-Guerra et al., Integrated circuit generating 3- and 5-scroll attractors, Commun. Nonlinear Sci. Numer. Simul. 17 (11) (2012) 4328–4335.

[13] H. Nejati, A. Beirami, A. G. Sahebi and W. H. Ali, "Variability analysis of tent map-based chaotic-map truly random number generators," 2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS), Columbus, OH, USA, 2013, pp. 157-160, doi: 10.1109/MWSCAS.2013.6674609.

[14] Lee, H., & Moon, S. (2002). Parallel stream cipher for secure high-speed communications. Signal Processing, 82, 259–265.

[15] A. Mitra, On the construction of m-sequences via primitive polynomials with a fast identification method, Int. J. Electron. Commun. Eng., 2 (9) (2008) 1991–1996.

[16] Zarei Moghadam, I., Rostami, A.S., and Tanhatalab, M.R. (2010). Designing a random number generator with novel parallel LFSR substructure for key stream ciphers. In International Conference on Computer Design and Applications (ICCDA). (pp. V5-598–V5-601).

[17] Haridas, Nisha & Devi, Nirmala. (2011). Efficient Linear Feedback Shift Register design for Pseudo Exhaustive Test Generation in BIST. 1. 10.1109/ICECTECH.2011.5941621.

[18] Rosselló, J. L., Canals, V., Paúl, I. D., Bota, S., & Morro, A. (2008). A simple CMOS chaotic integrated circuit. IEICE Electronics Express, 5, 1042–1048.

[19] H. Moqadasi and M. B. Ghaznavi-Ghoushchi, "A new Chua's circuit with monolithic Chua's diode and its use for efficient true random number generation in CMOS 180nm", Analog Integrated Circuits and Signal Processing, vol. 82, no. 3, pp. 719–731, 2015.

[20] National Institute of Standards and Technology, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Natl. Inst. Stand. Technol. Spec. Publ. 800-22rev1a, 131 pages, April 2010.

[21] Raspberry Pi Ltd., RP2040 Datasheet: A microcontroller by Raspberry Pi, Raspberry Pi Ltd., Cambridge, UK, 2024. [Online]. Available: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf

[22] Digi International Inc., XBee Zigbee Mesh Kit: Radio Frequency (RF) Module User Guide, Digi International Inc., Hopkins, MN, USA, 2022. [Online]. Available: https://www.digi.com/resources/documentation/digidocs/pdfs/90001942-13.pdf