Diseño y Evaluación de un Compresor de Datos para Redes Inalámbricas de Sensores

Oscar Pablo Porto Solano

Trabajo de grado presentado como requisito parcial para optar al titulo de:
Magíster en Ingeniería con Énfasis en Eléctrica y Electrónica

Director:
Ph.D. Oscar Acevedo Patiño

Línea de investigación
Redes de Comunicaciones Inalámbricas
Grupo de investigación
Centro de Excelencia y Apropiación en Internet de las Cosas

Universidad Tecnológica de Bolívar
Facultad de Ingeniería
Cartagena de Indias, Colombia
2018

Agradecimientos

Primero agradecer a Dios Padre, Hijo y Espíritu Santo por mostrarme el camino de salvación y presentar en mi vida las oportunidades para alcanzarla, además de importantes bendiciones como lo es esta Maestría. Luego quiero agradecer a Jader Giraldo, mi hermano en el espíritu y la vida por haberme recomendado como aspirante a esta beca junto a la Dra. Sonia Contreras quien fue mi Directora de Tesis de pregrado.

Quiero agradecer de manera especial al Profesor Oscar Acevedo , PhD. quien es el Director de esta Tesis de Mestría por la gran acogida, amistad, paciencia y apoyo durante estos años de estudio e investigación. Siempre tuvo una palabra de aliento para animarme a continuar trabajando y poder culminar satisfactoriamente mis estudios. También quiero dar gracias al Profesor José Luis Villa, PhD. quien con su gran sabiduría y humildad dirigiendo las riendas de la Decanatura de la UTB, nos ha ayudado a involucrarnos en el maravilloso mundo de la investigación.

Muchas gracias al Profesor Julio Hurtado por su aporte en la parte estadística de este proyecto. A los profesores Javier Campillo, PhD. y Juan Carlos Martínez, PhD. quienes de con sus ayudas y correciones lograron aportar valiosas enseñanzas en mi proceso de formación académica. A todos los(as) profesores(as) y estudiantes que me encontré en el camino.

Gracias a mis compañeros del CEA - IoT: Leonardo, Luz, Iván, Maryoris y Oscar, con quienes he forjado una valiosa amistad. Al Centro de Excelencia y Apropiación en Internet de las Cosas por financiar mi Matrícula en la UTB y brindarme la oportunidad de hacer parte de este nuevo grupo de investigadores en Colombia. Muchas gracias a la Universidad Tecnológica de Bolívar por abrirme las puertas de sus campus, los cuales siento como mi segunda casa. Finalmente y no por ello siendo menos importantes, a todos los miembros de mi familia de sangre y corazón que de alguna u otra forma me ayudaron en estos dos años con su paciencia, amor, tiempo, dedicación, bienes materiales y espirituales.

Contenido

1. Descripción del proyecto

1.1. Introducción

El ahorro de energía en las redes inalámbricas de sensores (*WSN, Wireless Sensor Network*) constituye uno de los campos de investigación más importantes en el contexto del Internet de las Cosas (*IoT, Internet of Things*) y por fuera de ella también, debido a que conlleva a prolongar la vida útil de la *WSN*, reducir costos de producción al disminuir el tamaño de la fuente de alimentación y minimizar esfuerzos de todo tipo en su mantenimiento. Cabe destacar que las *WSN* están presentes en muchos campos de investigación y desarollo relacionados con la agricultura, la minería, el monitoreo de estructuras civiles como puentes y edificios, sistemas eléctricos y electrónicos, sistemas de control en la industria, puertos marítimos, etcétera.
La compresión de datos es una de las estratégias utilizadas para reducir el consumo de energía en una *WSN*, debido a que generalmente el sistema de transmisión/recepción inalámbrico (CDMA, WiFi, LoRa, etc) es el componente que mas consume potencia en un nodo sensor.

1.2. Contexto y descripción del problema

El ahorro en el consumo de energía es un requerimiento crítico para las *WSN*. El éxito en la reducción del consumo de energía determina el tiempo máximo de vida de la red. Existen una gran cantidad de artículos que estudian los patrones de consumo de energía en las *WSN*, en los cuales se evidencia, salvo algunos casos particulares, que el mayor consumidor de energía en los nodos sensores es el módulo de trasmisión/recepción inalámbrica. A partir de esto, se han realizado una gran cantidad de trabajos para adaptar y optimizar las diferentes capas que conforman el módulo de red inalámbrica.
Adicional a esto, se han adaptado técnicas para la compresión de datos, de tal forma que el tiempo de uso del radio sea lo más reducido posible. Una gran cantidad de aplicaciones utilizan técnicas de compresión donde los datos originales se pierden por completo. A esto se le conoce como agregación de datos. Ejemplos de esto son valores promedio, valor máximo o mínimo. Estos valores se calculan a lo largo de la red, reduciendo en gran medida la cantidad de datos a transmitir. Otras aplicaciones utilizan técnicas de compresión donde los datos se pierden parcialmente. Finalmente, existen otras aplicaciones donde se desea obtener todos los datos medidos, con el fin de modelar su comportamiento. En este grupo se ubican algunas

aplicaciones de *IoT* y minería de datos, donde la información se procesa con algoritmos de aprendizaje (*Machine Learning*) para generar predictores, clasificadores, etcétera.

Actualmente existen varios algoritmos para comprimir datos sin pérdida. Sin embargo, estos algoritmos tienen una fuerte dependencia en la correlación existente entre los datos o tienen requerimientos de memoria considerables. Cuando estos requerimientos no se cumplen, los algoritmos pueden fallar, originando datos de salida de mayor tamaño que el dato original.

Lograr que un nodo sensor reduzca su consumo de energía conlleva el beneficio inmediato en la extensión del tiempo de vida de la red, cuya implicación económica es considerable. También abre la posibilidad a la reducción en el hardware que trae beneficios como la miniaturización y su uso en otros campos, como es el caso de las redes alrededor de cuerpo (*Body Area Networks*), cuyo fin es monitorear variables fisiológicas de un ser vivo.
Además, reducir el consumo de energía en una red inalámbrica de sensores brinda la posibilidad de agregar más nodos sensores, lo que se traduce en un aumento del alcance de la red. Esto se debe que la cantidad de energía agregada por los nuevos nodos sensores es compensada por la reducción energética obtenida con el algoritmo de comprensión. Esta situación puede presentarse dependiendo de los requerimientos de diseño de la red inalámbrica de sensores o las características de una red ya instalada. En su defecto, aun si no se agregan nuevos nodos sensores, muchas *WSN* requieren aumentar la cantidad de datos que pueden enviar bien sea aumentando la frecuencia con la que se toman los datos o aumentando el tiempo con que se realiza una transmisión sin que esto conlleve a un aumento considerable en la cantidad de energía requerida, lo cual puede ser obtenido a través de la compensación energética generada por la compresión.
Naturalmente al realizar la compresión de los datos censados se requiere menos ancho de banda, lo que genera un impacto positivo directo en el envío de los datos a través de las redes comerciales de telecomunicaciones cuyo modelo de negocio muchas veces es cobrar por *kb* transmitido o establecen un valor máximo de datos por mes.

1.3. Aportes

A partir de lo expuesto anteriormente, se puede determinar la importancia de obtener algoritmos de compresión que reduzcan el consumo de energía y por lo tanto aumenten el tiempo de vida de una red inalámbrica de sensores, cuyo uso se ha masificado gracias a los avances tecnológicos y al impulso dado por el *IoT*.

Este trabajo propone evaluar el uso de la teoría de los Registros de Desplazamiento con Retroalimentación Lineal (*LFSR*, *Linear Feedback Shift Register*) para la implementación de un algoritmo de compresión sin pérdidas que permita reducir en general el consumo de energía en una red inalámbrica de sensores. El *LFSR* se utiliza normalmente como pseudo -

generador de números aleatorios y como generador de secuencias de vectores de prueba para identificar fallas en circuitos integrados.

1.4. Hipótesis de investigación

Mediante el uso de la teória del *LFSR* se puede diseñar un algoritmo de compresión de datos sin pérdida que puede ser implementado en una Red Inalámbrica de Sensores para reducir en general los requerimentos energéticos de la red.

1.5. Formulación de la pregunta de investigación

Según el planteamiento del problema, a continuación se muestra la formulación de la pregunta fundamental de esta investigación:

- ¿Es posible utilizar la teoría y algoritmos del *Linear Feedback Shift Registers* para implementar un algoritmo de compresión de datos sin pérdida con una eficiencia similar o mejor a otros algoritmos desarrollados para tal fin?

1.6. Objetivos

1.6.1. General

Diseñar, implementar y evaluar un algoritmo de compresión de datos sin pérdida basado en la teoría del *Linear Feedback Shift Registers* que permita comprimir los datos generados por una red inalámbrica de sensores.

1.6.2. Específicos

- Diseñar un algoritmo de compresión basado en la teoría del *LFSR* que permita comprimir datos provenientes de redes inalámbricas de sensores.
- Implementar el algoritmo de compresión diseñado con base en la teoría del *LFSR*.
- Evaluar la eficiencia del algoritmo implementado y su aplicabilidad en redes inalámbricas de sensores.

1.7. Metodología

La metodología propuesta en la presente investigación consta de las siguientes etapas:

1.7.1. Estudio general de las Redes Inalámbricas de Sensores

Esta etapa de la investigación tiene como objetivo principal realizar una descripción de la forma como están construidas las redes inalámbricas de sensores para resaltar características topológicas que contribuyan al diseño e implementación del algoritmo de compresión que finalmente aumenten el ahorro energético. Se incluirá también una descripción general de los componentes de un nodo sensor como por ejemplo: sensores, fuente de alimentación eléctrica, características de almacenamiento y procesamiento de datos, elementos de transmisión y recepción de señales de radio.

1.7.2. Estudio de la compresión de datos en las WSN

Una vez realizada la caracterización, es necesario que se identifiquen las variables que intervienen en las técnicas de compresión más importantes que se destacan a la fecha. De esta forma se conocerán más detalles de la arquitectura de diseño de estas técnicas lo cual permitirá reconocer patrones efectivos de compresión y de igual forma comparar estas características con las obtenidas de los diseños propuestos en esta investigación.

1.7.3. Diseño, implementación y prueba un algoritmo de compresión de datos con un LFSR

En esta etapa se busca determinar si un algoritmo basado en un *LFSR* puede reproducir los datos originales que hagan parte de una base datos de números predeterminados con una eficiencia de compresión adecuada. Esta es una etapa introductoria de diseño que tiene como objetivos secundarios familiarizarse con la arquitectura de diseño, los aspectos de su implementación y evaluación, y además, estudiar los aspectos principales del algoritmo que intervienen en la compresión de los datos.

1.7.4. Ajustes al algoritmo de compresión de datos

El algoritmo finalmente debe reajustado teniendo en cuenta los resultados encontrados en la etapa previa de esta metodología, con el fin de obtener los mayores beneficios que ayuden a alcanzar el objetivo de este trabajo. En esta etapa es válida la aplicación de teoremas de matemática booleana o cualquier herramienta lógico-matemática computacionalmente viable para una red inalámbrica de sensores. Aquí se establecerán los alcances del algoritmo y sus limitaciones.

1.7.5. Implementación y evaluación final del algoritmo de compresión de datos en un dispositivo IoT

Esta etapa busca la implementación y evaluación del algoritmo en algún dispositivo *IoT*. Se desea conocer características básicas como las siguientes:

- Evaluar la eficiencia del algoritmo mediante el tiempo de ejecución promedio con una base de datos predeterminada.
- Evaluar la eficiencia del algoritmo mediante el tiempo de ejecución promedio con valores aleatorios.
- Evaluar la eficacia promedio del algoritmo mediante la compresión de valores aleatorios extraidos de todo el rango de valores posibles.
- Evaluar la eficacia promedio del algoritmo mediante la compresión de valores aleatorios limitados a una base de datos que sea un subconjunto de todo el rango de valores posibles.

Algunas de las características mencionadas como el rendimiento del mismo puede variar dependiendo de las tareas en segundo plano que se ejecuten en el dispositivo *IoT*.

2. Compresión de datos en WSN: Estado del Arte

2.1. Introducción

Las redes inalámbricas de sensores empezaron a aparecer en la década de los 90's. Desde el principio fue evidente los retos a los que se enfrentaban los ingenieros: los nodos sensores se alimentan de baterías, lo cual implica que la energía disponible es limitada. Como consecuencia, fue requerido el uso de tecnología de bajo consumo de potencia, lo cual implicaba el uso de microprocesadores de baja capacidad, radios de baja potencia y el rediseño de los protocolos de comunicación, debido a que los protocolos de comunicación convencionales, como *WI-FI*, no están optimizados para minimizar el consumo de energía.

Adicional a los trabajos orientados hacia el hardware y la red de comunicación, se han desarrollado trabajos que buscan comprimir los datos a transmitir. La premisa es simple, a menor cantidad de bits por transmitir, menor es el consumo de energía del sistema de radio.

De acuerdo a la aplicación, se tienen tres tipos de compresión:

- Sin pérdida: los datos son completamente recuperables.
- Con pérdida: parte de la información desaparece en el proceso de compresión.
- Agregación de datos: ninguno de los datos originales se puede recuperar.

En general, las aplicaciones de monitoreo que utilizan redes de sensores, aplican los dos últimos tipos de compresión. Sin embargo, la compresión sin pérdida toma importancia en aplicaciones que buscan caracterizar el fenómeno bajo observación. Este es el caso de aplicaciones de Internet de las Cosas (IoT), donde la red de sensores genera información que se utiliza para construir modelos de predicción y/o acciones de control.
Teniendo en cuenta la situación expuesta anteriormente, se han desarrollado diferentes investigaciones relacionadas con la compresión de datos sin pérdida, cuyo objetivo principal es generar algoritmos que cumplan con las restricciones impuestas en los nodos sensores y logren una razón de compresión aceptable. A continuación, se muestran algunas descripciones de varias de las investigaciones realizadas en esta temática:

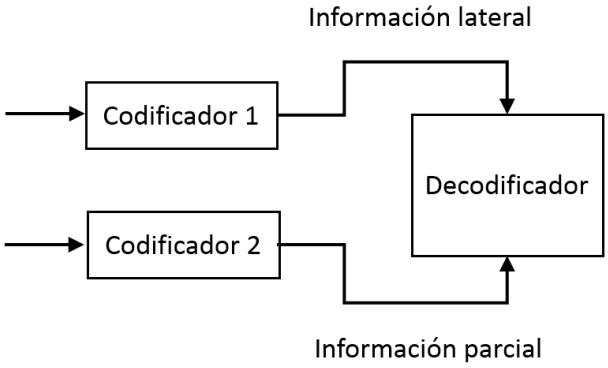


Figura 2-1.: Esquema de Compresión Distribuida.

2.2. Compresión de datos en WSN

La alta correlación espacial de los datos es aprovechada en redes de sensores altamente densa, es decir donde los sensores están relativamente cerca. La idea básica detrás del esquema de Compresión Distribuida [17, 12] es usar información lateral para codificar información de fuente. Por ejemplo, existen dos fuentes (X, Y) como se muestra en la figura **2-1**. Ellos deben estar correlacionados e idénticamente distribuidos según un alfabeto discreto. Ya que en una red de sensores, los nodos sensores son abundantes dentro del campo censado, esta condición de correlación es fácilmente satisfecha.
Entonces, Y puede ser comprimida a una tasa teórica de esta condición de entropía, $H(X|Y)$, en el Codificador 1 accediendo a Y [21]. La condición de entropía puede ser expresada como se muestra en la ecuaciín (2-1):

$$H(X|Y) = -\sum_{y}^{n} P_Y(y) \sum_{y}^{n} P_X(x|y) log_2 P_X(x|y) \tag{2-1}$$

El esquema general de la Compresión Distribuida consiste primero en crear un arreglo pre-configurado, de los codevectores de la fuente X. La distancia de alguno de los dos codevectores en el mismo arreglo pre-configurado tiene que ser suficientemente grande. Un valor de índice es asignado para cada arreglo pre-configurado. Cuando se transmiten los datos a un decodificador, la fuente X solo envía un índice del valor pre-configurado, para los cuales los codevectores encajan. Los decodificadores revisan los arreglos pre-configurados, los cuales tienen el mismo índice recibido de X. Entonces, los decodificadores seleccionan un codevector el cual tiene un valor cercano al codevector enviado por Y, en el arreglo pre-configurado. Sin embargo estos métodos no funcionan en redes inalámbricas de sensores donde los datos recolectados no tienen correlación. Más estudios relacionados pueden ser encontrados en [6, 8].

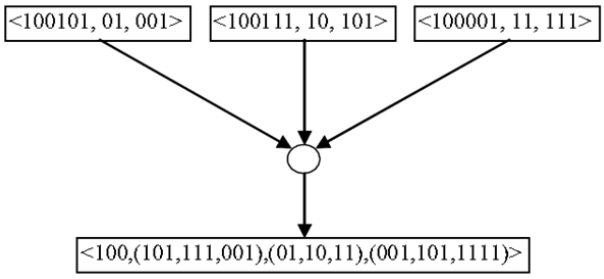


Figura 2-2.: Trama de datos en compresión *Pipelined In-Network Compression.* [2].

Otros trabajos se enfocan en agregación de datos y ruteo de la información, sin embargo estos métodos requieren correlación espacio-temporal para ser eficientes. Uno de estos es *Pipelined In-Network Compression* [2]. El algoritmo consiste en intercambiar alta latencia de datos por consumo de energía en la transmisión. Los datos censados son almacenados en un nodo búfer de agregación durante un tiempo en el cual la información es combinada para formar un sólo paquete y la redundancia en el paquete es removida para minimizar el total de datos transmitidos. Por ejemplo, cada paquete de datos tiene la forma:

$Trama =< valor\ medido, ID\ nodo, lista\ de\ marca\ de\ tiempos >$

Los "prefijos compartidos" son los de bits más significativos, que tienen en común todos los valores dentro del paquete de medidas. La longitud de los prefijos compartidos puede ser cambiada por un usuario basado en el conocimiento de correlación de los datos. La "lista de sufijos" son los bits menos significativos que quedan de cada valor medido al quitarle el prefijo compartido. La "lista de ID de nodo" es la lista de los identificadores de los nodos y la "lista de marcas de tiempo" es la lista que contiene información sobre fecha y hora en que el sensor tomó la medida. La figura **2-2** muestra medidas provenientes de tres sensores que tienen como prefijo 100_2 y la forma en que resulta la trama de datos agregados. La trama resultante tiene 6 bits menos que la trama que resultaría si no se aplica este procedimiento. La desventaja de este método es que depende de las similitudes de entre los datos medidos.

De esta misma categoría es el esquema de compresión de datos Codificación por Orden, el cual es introducido en [15] como parte de un Enrutamiento de Túnel de Datos (*Data Funneling Routing*). El esquema de compresión inicia pasando un dato de un nodo sensor en la región interesada a un nodo colector de la forma que se ve en la Figura **2-3**. En el Enrutamiento de Túnel de Datos, algunos de los nodos sensores trabajan como nodos de agregación de datos. Por ejemplo, los nodos A, B y D son nodos de agregación de datos. En este tipo de nodos los datos recolectados por otros nodos son combinados, y la agregación de datos es enviada a sus nodos padres.

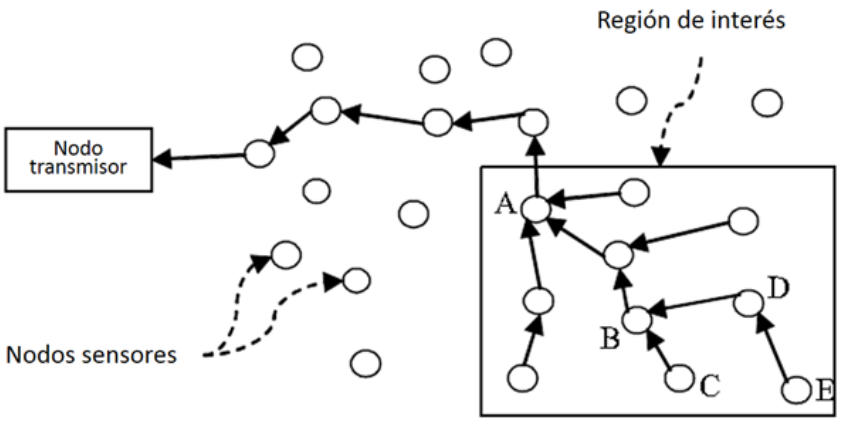


Figura 2-3.: Esquema de compresión de datos *Data Funneling Routing*.

En el algoritmo, cuando los datos son combinados en un nodo de agregación, algunos datos son excluidos. El orden de los datos del paquete es utilizado para incluir en los datos agregados la información excluida. Configurando algunos parámetros dentro del algoritmo se pueden obtener reducciones del 44 % en el tamaño de los datos. Una dificultad de utilizar este esquema es que ya que no hay un algoritmo eficiente de mapeo de permutación para el valor del dato, entonces se requiere una tabla de mapeo. A medida que el número de nodos sensores aumenta, el tamaño de la tabla incrementa exponencialmente. Otros trabajos similares pueden ser encontrados en [4, 10, 14, 20, 23].

A pesar que la correlación de los datos sea una limitante para ciertas aplicaciones, el concepto de compresión basada en imagen es muy interesante y útil. Una imagen es usualmente compuesta por muchos pequeños pixeles, y esta imagen puede ser modelada mediante una matriz cuyos elementos son los valores de estos pixeles. Al aplicar la *Transformada Wavelet* a la matriz, se logra extraer las características importantes de la imagen en el dominio de la frecuencia [18]. Entonces, el tamaño de la imagen puede ser significativamente reducido al almacenar sólo estas características importantes de la imagen.

La técnica de compresión basada en imagen adopta una idea similar. Se organiza una Red Inalámbrica de Sensores con una arquitectura jerárquica y se consideran los datos censados provenientes de todos los sensores como los múltiples pixeles que componen la imagen. Entonces, una Transformada *Wavelet* es aplicada para extraer un resumen espacial y temporal de esos datos censados. Explícitamente, cuando los datos censados poseen un alto grado de correlación espacial o temporal, los resultados obtenidos con la técnica de compresión basada en imagen mejoran significativamente. Dos técnicas de compresión desarrolladas con este concepto son *DIMENSIONS framework* documentada en 2005 en [7] y en 2009 *multiresolution compression & query (MRCQ) framework* encontrada en [24].

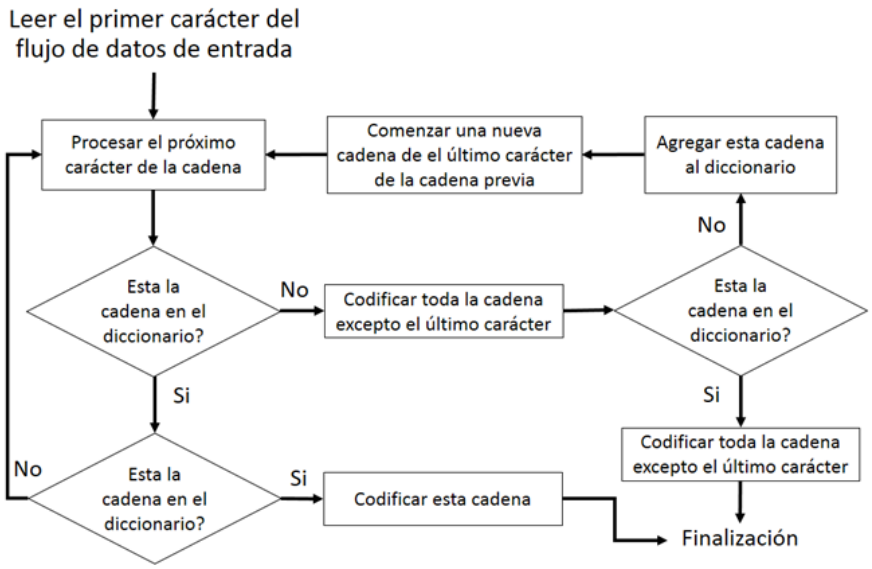


Figura 2-4.: Diagrama de flujo del algoritmo *LZW*.

Por otro lado, LZW (*Lempel-Ziv-Welch*) [27] es un algoritmo de compresión sin pérdida (*lossless compression*) en el cual se construye dinámicamente un diccionario para codificar nuevas cadenas de datos basados en las cadenas previamente encontradas. La figura **2-4** muestra un diagrama de flujo del algoritmo *LZW*, en el cual, un diccionario es inicializado para incluir la correspondiente cadena de un solo carácter para todos los posibles caracteres de entrada. Por ejemplo, al usar el Código Estándar Americano para Intercambio de Información (*American Standard Code for Information Interchange - ASCII*), el diccionario contendría 256 entradas iniciales.

Entonces, el algoritmo *LZW* escanea cada carácter del flujo de datos de entrada hasta que encuentra una sub-cadena que no está en el diccionario. Cuando una cadena es encontrada, se envía el índice de la cadena más larga encontrada al flujo de datos de salida, mientras que ésta nueva cadena es agregada dentro del diccionario con el próximo código disponible. Entonces, el algoritmo *LZW* continúa escaneando el flujo de datos de entrada, empezando desde el último carácter de la cadena anterior.

El algoritmo *LZW* es computacionalmente simple y no es de alta transmisión. Específicamente, debido a que el transmisor/receptor de entrada el mismo diccionario inicial y todas las entradas de diccionario nuevas pueden ser derivadas de las entradas del diccionario existente y el flujo de datos de entrada, el receptor puede así construir el diccionario completo en el momento en que reciba los datos comprimidos. Como consecuencia Sadler y Martonosi introducen *S-LZW* [19], una nueva variante del algoritmo *LZW* adaptada para soportar redes inalámbricas de sensores, que aprovecha los patrones característicos de los datos de los sensores para reducir el consumo de energía en un factor superior a $1{,}5X$, así como también una transformación adicional de los datos que pueden tomar ventaja de la estructura de los datos para reducir el promedio de consumo de energía en un factor superior a $2{,}5X$.

Los autores aseguran que los algoritmos son relevantes para una amplia variedad de aplicaciones e implementaciones de sensores. Ellos demuestran esto mediante la aplicación de sus algoritmos a datos provenientes de tres diferentes desarrollos reales y examinando sus efectos en tres radios de diferentes rangos y niveles de potencia. A medida que la energía del radio incrementa, lo hacen los beneficios de la compresión, ya que cada byte ahorrado lo incrementa significativamente. A través de estas evaluaciones, mostraron como la compresión acumula ahorros sustanciales de energía obtenidas en el nodo de compresión, y los nodos intermediarios entre el nodo fuente y el nodo final.

2.3. Compresión de datos mediante el uso del LFSR

Los Registros de Desplazamiento con Retroalimentación Lineal (*Linear Feedback Shift Register, LFSR*) son maquinas lógicas secuenciales y lineales construidas a partir de registros tipo D y compuertas XOR. Estas máquinas permiten la generación sucesiva de números pseudo-aleatorios. En la seccion 3.2.1 se ampliará en detalle sobre su funcionamiento, características y propiedades.

los LFSR aparecieron alrededor del 1969 de la mano del investigador James L. Massey, inicialmente se presentó como una herramienta para generar números pseudo - aleatórios muy útiles para muchas aplicaciones. Sin embargo Massey vislumbraba su uso como parte de un código mas grande, un compresor de datos, una fuente de datos con alta redundancia y desde entonces esta teoría ha sido objeto de uso en diferentes disciplinas, y principalmente adoptada por el campo de la criptografía. Por la amplia acogida que tuvo en este último campo, varios de los teoremas que definen su comportamiento y características se han enunciado con terminología criptográfica, como se verá más adelante en el marco teórico. En [5] realizan una descripción detallada de la teoría de un LFSR y el algoritmo de Berlemkamp - Massey. Este último es muy eficiente para determinar el LFSR mas pequeño (*Shorter Linear Feedback Shift Register, S - LFSR*) que genera una secuencia de datos cualquiera, utilizando solo la mitad de los datos de la secuencia. Este algoritmo es el que [Campbell, 1999] pone a prueba mediante varias posibes combinaciones que se pueden presentar en la solución.

La figura **2-5** es una gráfica que muestra que el LFSR no provee las mismas características que los algoritmos de la época y para los archivos estándares utilizados, que son los ampliamente conocidos archivos de Canterbury Corpus [3, 16]. Sin embargo la búsqueda de compresión que realizan es para archivos de cualquier denominación y no para Redes Inalámbricas de Sensores, en donde los datos pueden presentar características propias como alta redundancia. Además la investigación presentada en este documento hace uso de los LFSR sin considerar al Algoritmo de Berlemkamp - Massey.

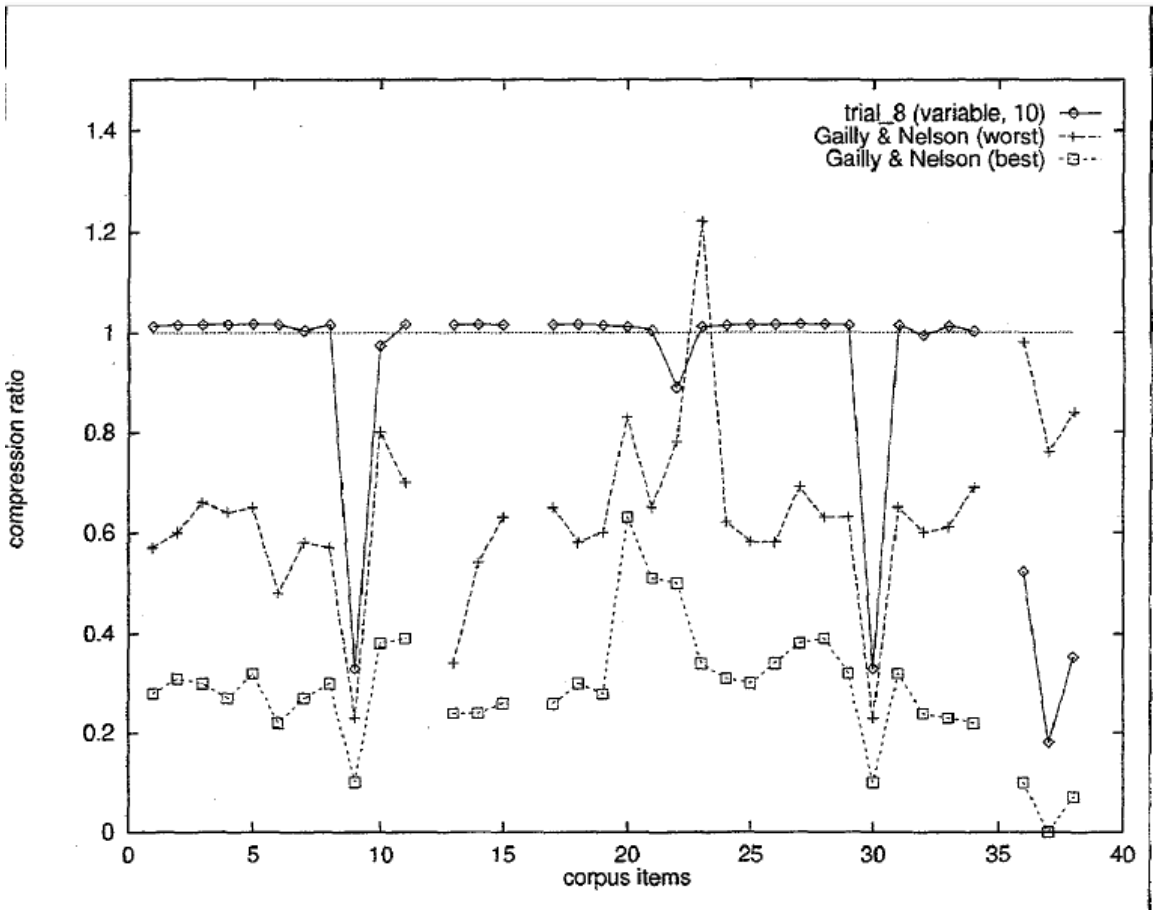


Figura 2-5.: Lo mejor el LFSR (Trial 8: variable 10); Lo peor y lo mejor de Gaily y Nelson. Los corpus item hacen parte de una Tabla 5 en el paper [5].

Por otro lado en [13] se presenta un algoritmo que utiliza LFSR que no está basado en el método de diccionario. En este algoritmo, usan la compresion de datos tipo *run length* y el algoritmo de Berlekamp - Massey. La secuencias continua de tramas binarias repetidas se codifican usando la técnica de *run length*. La otra secuencia, es decir, la secuencia discreta o secuencias codificadas de palabras son tomadas como una secuencia continua y se le entrega al algoritmo Berlekamp - Massey para calcular el polinomio y complejidad lineal de la secuencia. La complejidad lineal obtenida es el LFSR mas pequeño (*S-LFSR*) requerido para calcular o generar la secuencia recibida. Utilizando el polinomio y la complejidad lineal del S-LFSR se recupera la trama de entrada. De nuevo se observa que el algoritmo no ha sido diseñado para redes inalámbricas de sensores y además hace uso del algoritmo Berlekamp - Massey.

La principal desventaja de la compresión basada en *LFSR* es que debe combinarse generalmente con el restringido proceso de un *ATPG (Automatic test pattern generation)*, y, como resultado, no se puede aplicar efectivamente a núcleos IP de estructura desconocida [Koutsoupia, 2009]. En [11], una nuevo enfoque de compresión basado en *LFSR* que supera este problema se propone. El método propuesto permite que cada semilla del *LFSR* codifique tantos rebanadas como sea posible. Para lograr esto, una sección de propósito especial, llamado *stop-slice*, que indica que el final del uso de una semilla está codificado como la última rebanada de cada semilla. Por lo tanto, las semillas incluyen por construcción la información de dónde deberían parar y, para esa razón, los llamamos auto-stoppable. Una generación *stop-slice* se propone un procedimiento que explota las características inherentes del conjunto de prueba y genera cortes stop que imponen una compresión mínima gastos generales. Además, la arquitectura para implementar la técnica propuesta requiere hardware adicional insignificante sobrecarga en comparación con la arquitectura estándar basada en *LFSR*. La técnica propuesta también va acompañada de un cálculo inicial Algoritmo que intenta minimizar el número de semillas calculadas.

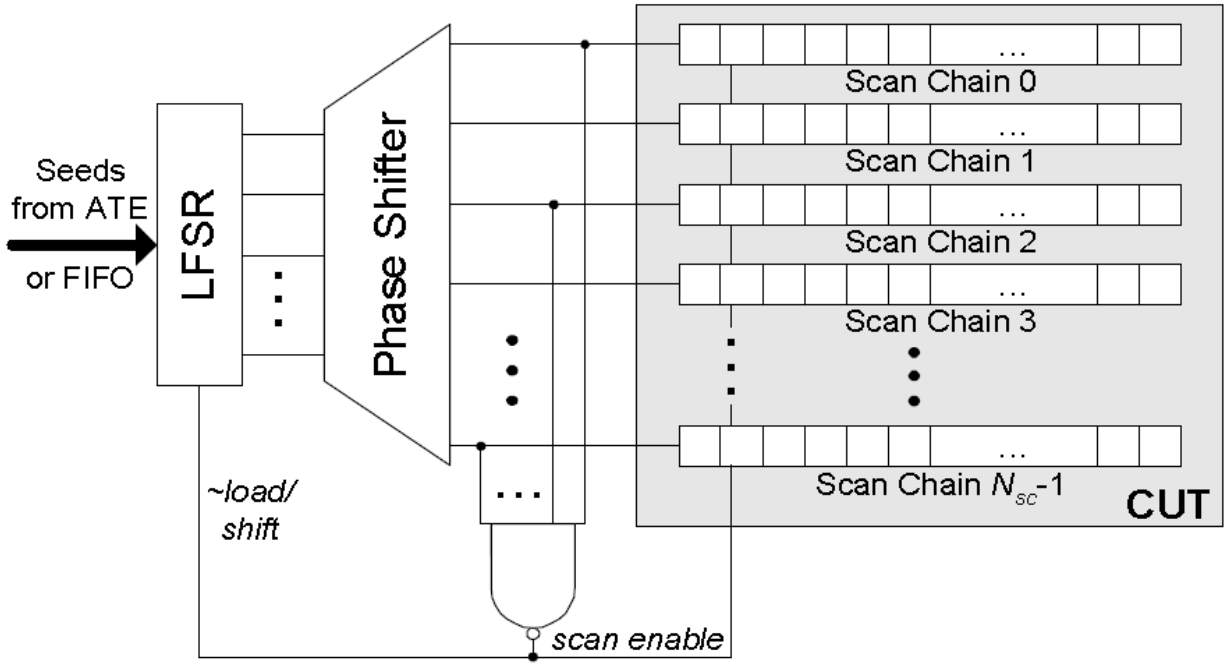


Figura 2-6.: Arquitectura propuesta en para generar patrones de pruebas comprimidos que se autogeneren [11].

3. Marco Referencial

3.1. Redes Inalámbricas de Sensores

Una Red Inalámbrica de Sensores o *WSN* consiste de una distribución de sensores autónomos para monitoreo de condiciones físicas o ambientales, como son por ejemplo: temperatura, humedad, polución (que generan bajo flujo de datos) y sonido, vibración, presión, movimiento (generan alto flujo de datos). Además, cooperativamente transmiten sus datos a través de la red que conforman hasta una central de datos principal conocida como *gateway*. En general los datos son recolectados en el nodo sensor inalámbrico individual, comprimido, y transmitido directamente al gateway o, si se requiere, se utilizan otros nodos sensores para dirigir los datos al gateway. Entonces son presentados ante el sistema final de control, almacenamiento y visualización mediante la conexión que tenga disponible el gateway.

Comparado con redes de sensores cableados, las *WSN* proveen monitoreo continuo de forma inalámbrica a un considerable bajo costo, simplifican el proceso de instalación al no requerir cables largos de alimentación y comunicaciones, en caso de daños en el sensor son fáciles de reemplazar, la reorganización de la topología de la red es fácil utilizando *Ad-Hoc* y *Multi-Hop*. Sin embargo, la transmisión de datos con tecnología *WSN* tiene serias limitaciones, incluyendo la distancia de la transmisión, ancho de banda y energía eléctrica disponible [28].

3.1.1. Arquitectura individual de un nodo sensor inalámbrico

Un diagrama de bloques funcional generalizado de un nodo sensor inalámbrico es mostrado en la figura **3-1**, el cual puede ser modificado según la aplicación. Además debido al avance de los circuitos integrados (*CI*) muchos de estos módulos vienen embebidos en un dispositivo disponible comercialmente. El microprocesador que poseen, tiene varias funciones entre las que se encuentran:

- Controlar la recolección de datos de los sensores
- Administrar el consumo de energía
- Interconectar los datos recolectados por el sensor con la capa física de radio
- Controlar el protocolo de red de la radio

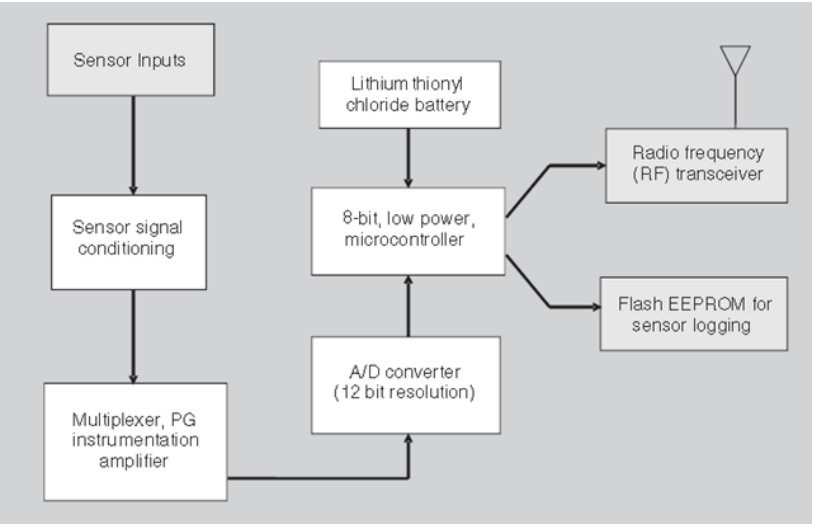


Figura 3-1.: Diagrama de bloques funcional de un nodo sensor típico.

Una característica clave en cualquier nodo de medición inalámbrico es minimizar el consumo de potencia del sistema. Generalmente los subsistemas de radio requieren de grandes cantidades de potencia. Por lo tanto es conveniente enviar datos sobre el sistema de radio solo cuando sea necesario. Entonces el modelo de recolección de datos requiere de un algoritmo que determine cuando los datos deben ser enviados basado en la sensibilidad de los eventos medidos. Adicional a esto es necesario reducir el consumo de energía del mismo sensor. En general el hardware debe ser diseñado para que el microcontrolador pueda ejercer control sobre el consumo de potencia del sistema de radio, el sensor, el acondicionador de la señal del sensor y la memoria flash.

3.1.2. Arquitectura de una Red Inalámbrica de Sensores

Las redes de comunicaciones por radio tienen distintos tipos de topologías. Las más relevantes para la aplicación desarrollada en esta investigación se muestran a continuación:

Topología en estrella (de un nodo a varios nodos)

La red en estrella mostrada en la figura **3-2** es una topología de comunicaciones en la cual una sola estación base (*gateway*) puede enviar y/o recibir mensajes a un número remoto de nodos. Los nodos remotos sólo pueden enviar o recibir mensajes de la estación principal y no entre ellos.
Las ventajas de esta topología para las *WSN* radican en su simplicidad y la habilidad para mantener al mínimo el consumo de potencia de los nodos remotos. Además esta topología ofrece una baja latencia de comunicación entre la estación principal y los nodos remotos. La desventaja está en que la estación base debe tener un amplio rango de transmisión para

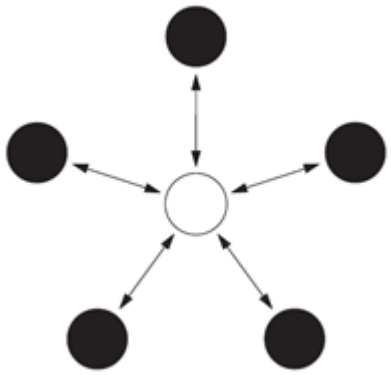


Figura 3-2.: Repesentación esquemática de la topología en estrella.

establecer comunicación con todos los nodos remotos y además no es tan robusta como otras redes debido a la dependencia de un solo nodo para el manejo de toda la red.

Topología tipo malla

Una red con topología tipo malla como la que se muestra en la figura **3-3**, permite a cualquier nodo de la red transmitir a cualquier otro nodo de la red que esté dentro su rango de transmisión de radio. Esto permite un método conocido como comunicación multi-salto (*multihop*); el cual consiste en que si un nodo quiere enviar un mensaje a otro nodo que esta fuera del rango de comunicación por radio, este puede usar un nodo intermedio para direccionar el mensaje hacia el nodo deseado.
Las ventajas de esta topología de red están en su redundancia y escalabilidad. Si un nodo individual falla, un nodo remoto puede aún mantener comunicación con otro nodo en su rango, el cual puede direccionar el mensaje la localización deseada. Además el alcance de la red puede ser ampliado mediante la adición de más nodos individuales al sistema. Las desventajas de este tipo de redes es que el consumo de potencia es superior al de redes que no implementan comunicación *multi-hop*, lo cual a menudo limita el tiempo de vida de la batería. Adicional a esto, a medida que la cantidad de saltos en la comunicación aumenta, el tiempo de entrega del mensaje incrementa también, especialmente si los nodos operan con requerimientos de ahorro de energía.

Topología híbrida estrella - malla

Un híbrido entre redes estrella y malla provee una comunicación robusta y versátil, mientras que posee la habilidad de mantener al mínimo el consumo de potencia de la *WSN*. En la figura **3-4**. En este tipo de topología, los nodos sensores con menor energía no se habilitan para enviar mensajes directamente. Esto permite que el menor consumo de potencia se mantenga. Por otro lado, otros nodos en la red si son habilitados con la capacidad de ser multihop, permitiendo el direccionamiento de los mensajes de los nodos con menos energías a otros

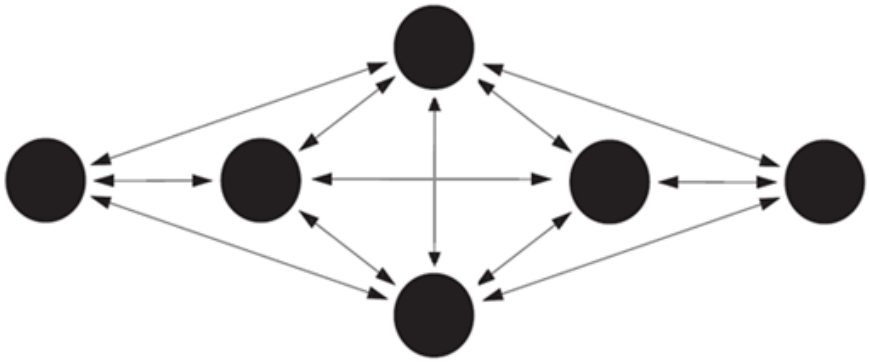


Figura 3-3.: Repesentación esquemática de la topología tipo malla.

Figura 3-4.: Repesentación esquemática de la topología híbrida estrella - malla.

nodos en la red. Generalmente, los nodos con capacidad multihop son de alta potencia, y si es posible, son conectados a las líneas de energía principal. Esta topología es implementada por la tecnología ZigBee.

3.1.3. Consideraciones de potencia en Redes Inalámbricas de Sensores

Uno de los aspectos más relevante en las Redes Inalámbricas de Sensores es el consumo de potencia. Avanzar en este aspecto permitirá la amplia difusión en la implementación de la tecnología de las *WSN*. La figura **3-5** muestra la contribución al consumo de potencia en una típica red inalámbrica versus la tasa de transmisión de datos en Hz (datos/segundo). Note que por una amplia diferencia el consumo de potencia está asociado con la conexión al sistema de radio.

Hay varias estrategias que pueden ser consideradas para disminuir el consumo de corriente del sistema de radio, que incluyen:

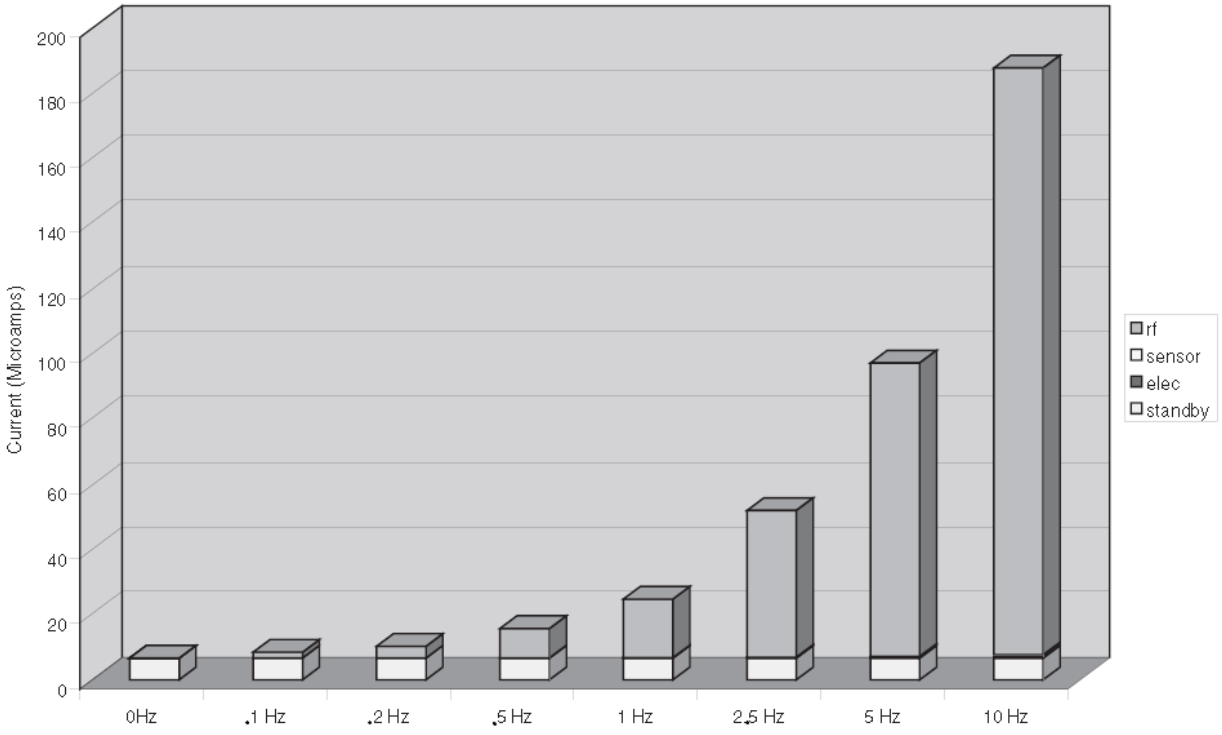


Figura 3-5.: Consumo de potencia de una Redes Inalámbricas de Sensores típica.

- Reducir la cantidad de datos transmitidos mediante compresión de datos.
- Menor ciclo de trabajo del transmisor y frecuencia de transmisión de datos.
- Implementar estrictos mecanismos en el manejo de la energía (modo dormido).
- Implementar estrategias de transmisión como por ejemplo: realizar la transmisión solo cuando ocurra un evento con el sensor.

Estrategias de reducción de potencia para el mismo sensor incluyen:

- Encender el sensor solo cuando se va a tomar una medición.
- Encender el acondicionador de señal solo cuando el sensor tome una medición.
- Configurar el sensor para realizar mediciones solo cuando un evento ocurra.
- Sensores con la menor tasa de muestreo según el requerimiento mínimo de la aplicación.

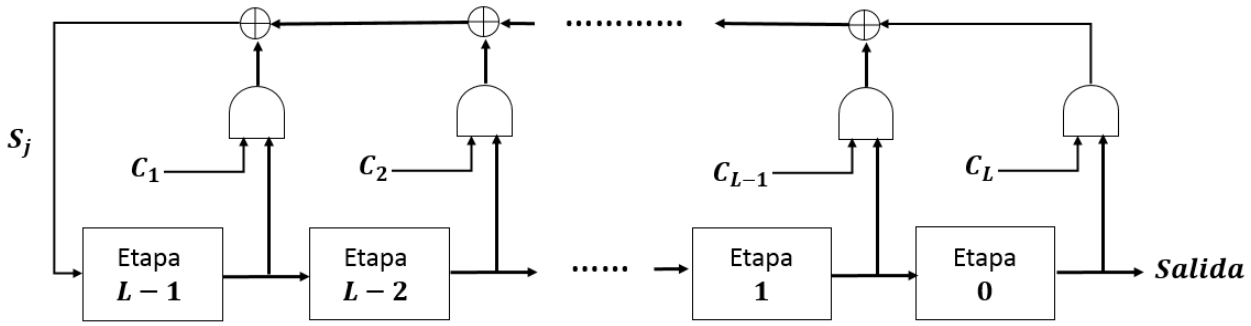


Figura 3-6.: Diagrama del circuito lógico de un *LFSR*.

3.2. Registros de Desplazamiento con Retroalimentación Lineal

3.2.1. Concepto general

Los Registros de Desplazamiento con Retroalimentación Lineal (*Linear Feedback Shift Register, LFSR*) son maquinas lógicas secuenciales y lineales construidas a partir de registros y compuertas XOR que permiten la generación sucesiva de números pseudo-aleatorios. Un *LFSR* de longitud L consiste en L registros numerados de 0 a $L-1$, cada uno capaz de almacenar un bit; tiene una entrada, una salida, una función de retroalimentación[1] y un reloj que sincroniza el flujo de datos entre los distintos registros.
Los primeros bits de la secuencia pseudo-aleatoria son los datos iniciales (semilla) de los L registros. Si la semilla es distinta de cero, se pueden obtener secuencias de números binarios con periodos largos, según la función de retroalimentación y la longitud del LFSR. La figura **3-6** muestra la estructura general de un *LFSR* de longitud L. Cada C_i es 0 o 1; el bit de retroalimentación S_j es una combinación lógica lineal del contenido de las L etapas, según sea el valor del vector C.

Un *LFSR* puede generar secuencias pseudo-aleatorias de tamaño máximo $2L-1$ bits. (No $2L$, porque un *LFSR* con semilla cero, genera como salida una secuencia infinita de ceros, entonces se debe restar esta opción). El periodo máximo $2L-1$ se obtiene cuando el polinomio formado por $\{C\}+1$ sea un polinomio primitivo módulo 2. A continuación se dan algunas definiciones para conocer las características que deben cumplir los polinomios primitivos.

3.2.2. Polinomios Primitivos e Irreducibles

Un polinomio primitivo es irreducible, aunque no todo polinomio irreducible es primitivo. Por otra parte, un polinomio binario primitivo $p(x)$ de grado L existe sobre todo el Campo

[1]También conocido como polinomio de retroalimentación o tap de retoalimentación

de Galois y todo Campo de Galois tiene al menos un elemento primitivo. Tener en cuenta las siguientes definiciones:

Polinomio Irreducible

Si hay un polinomio $p(x)$ tal que $r(x)p(x) = s(x)$, se dice que el polinomio $s(x)$ es divisible por el polinomio $r(x)$ o que $r(x)$ divide a $s(x)$. Un polinomio $p(x)$ diferente de cero, que sólo es divisible por $p(x)$ se dice irreducible ya que no puede ser escrito como el producto de dos polinomios, cada uno de grado positivo.

Polinomio Primitivo

Un polinomio irreducible $p(x)$ de grado L se dice que es primitivo si y solo sí:

- $p(x)$ divide a $X^n + 1$, cuando $n = 2L - 1$, y
- $p(x)$ no divide a $X^n + 1$, cuando $n < 2L - 1$.

Al utilizar un polinomio primitivo para construir la función de retroalimentación $\{C\}+1$ del *LFSR* se garantiza que la longitud de la secuencia generada no dependa de la semilla y que tenga un periodo máximo $2L - 1$. Ahora bien, considérese un *LFSR* definido por $[L, C(x)]$ siendo L la longitud del *LFSR* y $C(x)$ el polinomio que define su retroalimentación. Se dice que si $[L, C(x)]$ es singular, esto es, $C(x)$ tiene grado menor que L, entonces no todas las secuencias de salida generadas serán periódicas. Debido a esto se recomienda que el grado del polinomio sea L, es decir, el *LFSR* no debe ser singular.

3.2.3. Complejidad Lineal

La complejidad lineal revela características generales de los *LFSR* que pueden ser de mucha utilidad al momento de seleccionar las características del polinomio de retroalimentación, la longitud del *LFSR* y la longitud de la trama de salida.

Debido al amplio uso de esta teoría en criptografía, la complejidad Lineal suele definirse como la longitud L, del *LFSR* más pequeño o *SLFSR* (*Shortest Linear Feedback Shift Register*) que puede imitar la salida de un generador de secuencias binarias. A continuación se darán las definiciones formales relacionadas con la Complejidad Lineal:

Definición I

Sea $s = s_0, s_1, s_2, ...$ una secuencia infinita. La subsecuencia que consiste en los primeros n términos de s se denota por $s_n = s_0, s_1, s_2, ..., s_{n-1}$.

Definición II

Un *LFSR* genera una secuencia s, si hay un estado inicial para el cual la secuencia de salida del *LFSR* es s. De forma similar, un *LFSR* genera una secuencia finita s_n, si hay algún estado inicial para el cual la secuencia de salida del LFSR tiene s_n como sus primeros n términos.

Definición III

La Complejidad Lineal $L(s)$, de una secuencia binaria infinita s, se define como:

- Si s es la secuencia cero $s = 0, 0, 0, ..., 0$ entonces $L(s) = 0$.
- Si ningún *LFSR* genera s, entonces $L(s) = \emptyset$.
- De otra forma, $L(s)$ es la longitud del *LFSR* más pequeño que genera s.

3.2.4. Propiedades de la Complejidad Lineal

Sean s y t secuencias binarias, entonces las propiedades de la Complejidad Lineal son las que se muestran a continuación:

1. Para cualquier $n \geq 1$, la complejidad lineal de la subsecuencia s_n satisface la expresión $0 \leq L(s_n) \leq n$.
2. $L(s_n) = 0$ si y solo sí s_n es la secuencia cero de longitud n.
3. $L(s_n) = n$ si y solo sí $s_n = 0, 0, 0, ..., 0, 1$.
4. Si s es periódica con periodo N, entonces $L(s_n) \leq n$.
5. $L(s \oplus t) \leq L(s) + L(t)$, donde $s \oplus t$ denota la operación XOR entre los bits de s y t.

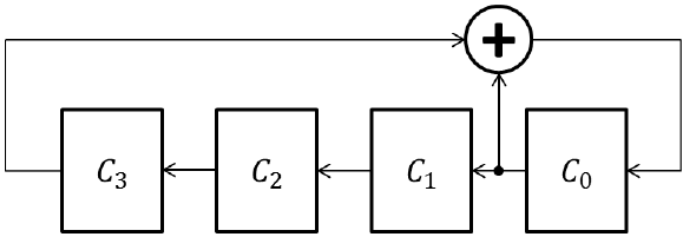
4. Diseño de un algoritmo de compresión de datos mediante el uso de LFSR

Como se mencionó en el Estado del Arte de esta investigación, varios algoritmos de compresión de datos han sido diseñados mediante el uso de un *LFSR*. Sin embargo se considerará un enfoque distinto en el uso de éstas máquinas lógicas, el cual, puede conducir a resultados que podrían ser la base de nuevas investigaciones y desarollos en el campo de la compresión de datos. Este capítulo inicia describiendo la forma como un *LFSR* genera secuencias, luego como se puede utilizar el principio de generación de secuencias para comprimir datos, y luego consideraciones importantes para el diseño del algoritmo.

4.1. Principio de generación de secuencias de un LFSR

La figura **4-1** muestra el diagrama de bloques de un *LFSR* constituido por 4 Flip-Flops tipo D (FFD o etapas) y una compuerta XOR. El *LFSR* empieza a generar secuencias no nulas cuando se proporcione una semilla distinta de cero en su entrada es decir por lo menos un bit en alto en la entrada Q de cualquiera de los registros C_n. La salida en serie del LFSR son los bits en el puerto D de C_3, mientras que la retroalimentación se da mediante la conexión de la salida de la compuerta XOR a la entrada Q de C_0.
El LFSR de la figura **4-1** puede ser descrito mediante la expresión $P(x) = x^4 + x^3 + 1$ conocida como polinomio característico y su funcionamiento se emula matemáticamente mediante el uso de la matriz de transición que utilizando unos y ceros representa las conexiones entre cada etapa. La última fila de la matriz de transición es la representación binaria del polinomio característico (polinomio de retroalimentación).

La figura **4-2** muestra la matriz de transición con los nombres de las etapas asignadas, de esta forma la fila 1 indica que la entrada Q de la etapa C_3 es la salida de D de la etapa C_2, la fila 2 indica que la entrada Q de la etapa C_2 es la salida de D de la etapa C_1, la fila 3 indica que la entrada Q de la etapa C_1 es la salida de D de la etapa C_0 y finalmente la fila 4 indica que la entrada Q de la etapa C_0 es la combinación (XOR) de la salida de D de las etapas C_3 y C_0. El algoritmo para construir la matriz de transición se describe el mediante pseudo - código mostrado en la figura **4-3**.



$$P(x) = x^4 + x^3 + 1$$

$$M=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

C_3	C_2	C_1	C_0
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1
1	1	1	0
1	1	0	1
1	0	1	0
0	1	0	1
1	0	1	1
0	1	1	0
1	1	0	0
1	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Figura 4-1.: Representación esquemática de un *LFSR* con polinomio de retroalomentación $P(x) = x^4 + x^3 + 1$. También se muestra la Matriz de Transición y la secuencia generada con la semilla $C = (0, 0, 0, 1)$ [1] .

$$M = \begin{array}{c} \begin{matrix} C_3 & C_2 & C_1 & C_0 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{array}{l} \rightarrow C_3 \\ \rightarrow C_2 \\ \rightarrow C_1 \\ \rightarrow C_0 \end{array}$$

Figura 4-2.: Matriz de transición del LFSR con polinomio de retroalimentación $P(x) = x^4 + x^3 + 1$.

Procedure 1 Calculate of matrix transition

Input: Polinomial $p(x)$

Output: Matrix transition M_t

```
1 : le ← long(p)
2 : M_t ← [∅]
3 : for i = 0 to le − 1 do
4 :         insert a bit = 0 to M_t vector
5 : end for
6 : tmp ← M_t
7 : M_t ← [∅]
8 : for i = 1 to le − 2 do
9 :         insert tmp in row i of M_t
10:         M_t[i, i + 1] ← 1
11: end for
12: insert tmp in last row of M_t
13: return M_t
```

Figura 4-3.: Pseudo - código para construir la Matriz de Transición M_t a partir del Polinomio de retroalimentación $P(x)$

Para iniciar la generación de una secuencia se debe multiplicar la matriz de transición M_t por un vector E_n cuya cantidad de elementos sea igual al número de etapas que tenga el *LFSR*; el vector E_n debe contener por lo menos un elemento no nulo. El vector inicial E_0 utilizado para generar la secuencia se conoce como *semilla*. Al efectuar la multiplicación $M_t \times E_n$ el vector resultante E_{n+1} es el próximo estado, el cual, se utiliza para calcular el estado subsiguiente $E_{(n+1)+1}$. Este procedimiento se puede efectuar de forma consecutiva e infinita para calcular los estados que continuan, sin embargo tarde o temprano la sucesión de números generados se repetirá. El primer elemento de los vectores de estado E_n constituyen la secuencia de salida del *LFSR*. La figura **4-4** ilustra gráficamente el proceso generación de la secuencia.

Si se ordenada secuencialmente el primer bit de cada uno de los vectores de estado E_n del estado 0 al estado $N-1$ se obtiene la secuencia S con periodo máximo N. A partir del estado $n = N$ la secuencia se repite. El primer bit de cada uno de los vectores de estado E_n es corresponde a los bits en la salida D del último flip flop del circuito lógico del *LFSR*, en la figura **4-1** la secuencia serían todos los bits en la D del flip flop o estado C_3.

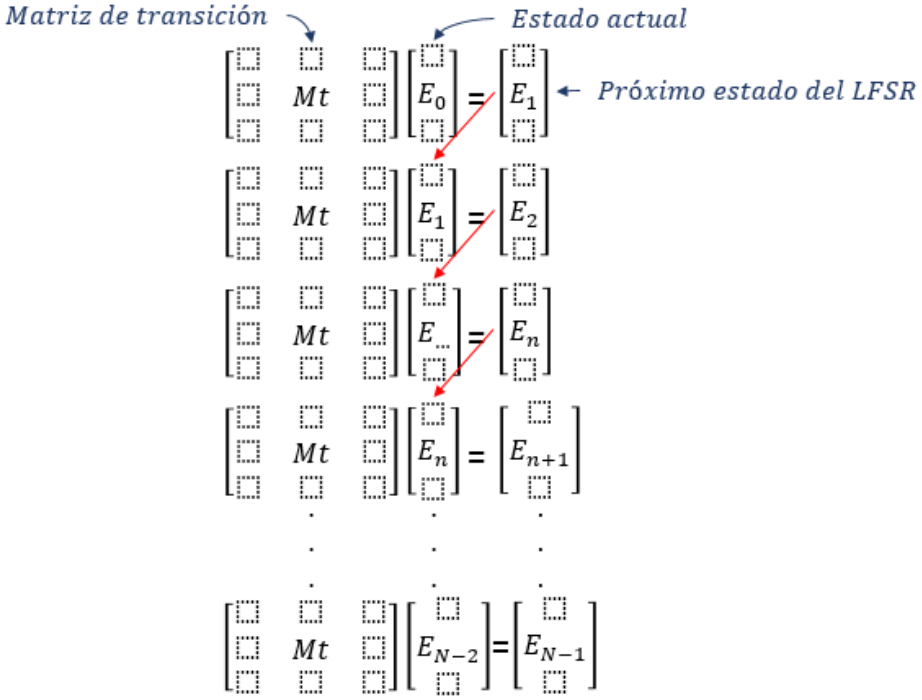


Figura 4-4.: Generación de la secuencia de salida de un *LFSR* a partir del producto sucesivo $M_t \times E_n$. La secuencia sería $s = E_0(0), E_1(0), E_2(0), ..., E_{N-2}(0), E_{N-1}(0)$.

4.2. Compresión de datos mediante el uso de la secuencia generada por un LFSR

En general la compresión de datos consiste en aplicar a una trama de bits B un algoritmo de compresión que arroje una trama B_c con menor cantidad de bits los cuales permitan mediante un algoritmo de descompresión recuperar la trama B, tal como lo ilustra la figura **4-5**. La trama B_c es la que se transmitiría desde el nodo fuente a través de todos los nodos intermedios de la *WSN* hasta llegar al nodo o dispositivo de destino en donde finalmente se aplicaría la descompresión. A continuación se expone la forma en que se utiliza el *LFSR* en esta investigación, además de consideraciones especiales para su uso.

4.2.1. Fundamento de la compresión de datos con un LFSR

Para esta investigación utilizando un *LFSR* lo que se busca es que la trama B sea una subsecuencia s_n de la secuencia mayor s de periodo N, y de esa forma aprovechar las características y propiedades de la Complejidad Lineal de los *LFSR* descritas en la sección 3.2.3 y 3.2.4. Entonces se tomaría la semilla E_0 como la trama comprimida B_c debido a que son estos valores los que permitirían recuperar la trama original B.

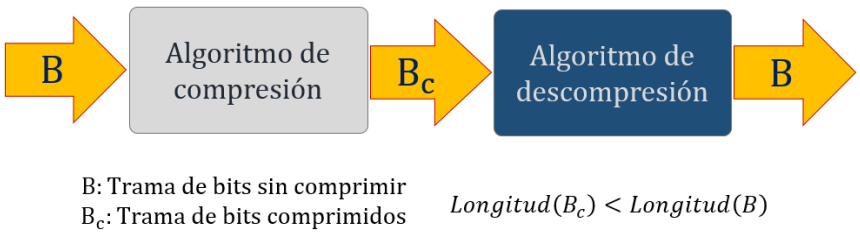


Figura 4-5.: Proceso general idealizado de la compresión de datos

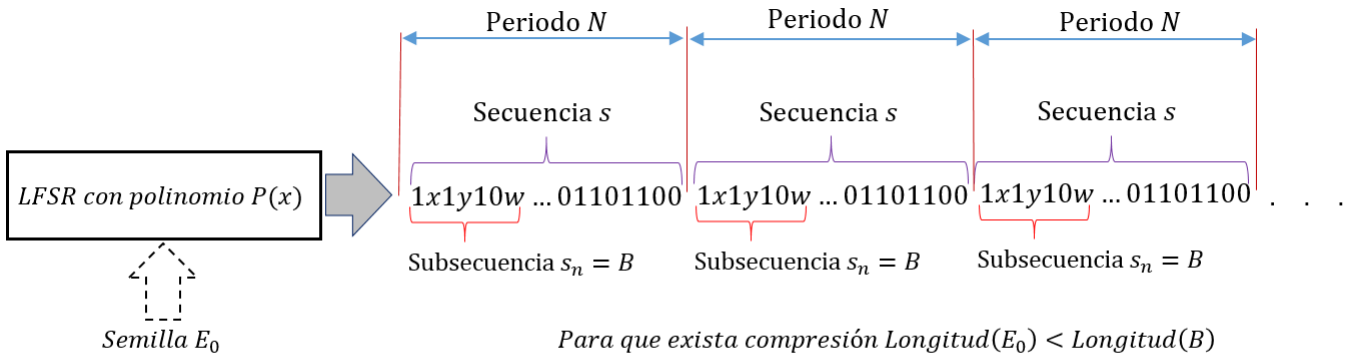


Figura 4-6.: Uso de un *LFSR* para compresión de datos

Para lograr esto, se requiere diseñar un algoritmo que encuentre la semilla E_0, de esta forma utilizando el polinomio $P(x)$ en el descompresor se obtiene la trama de entrada al algoritmo $B = s_n$, siendo $E_0 < B$. Es decir que para que exista compresión la longitud en bits de la subsecuencia $s_n = B$ debe ser mayor que la longitud en bits de la semilla E_0. la figura **4-6** ilustra la ubicación de estas variables mencionadas.

4.2.2. Planteamiento de ecuaciones para la obtención de la trama comprimida

En la figura **4-4** se puede observar que la secuencia s es el producto de la Matriz de Transición M_t por los estados E_n del *LFSR*. Debido a que la trama original B es conocida y la semilla E_0 desconocida, el problema puede plantearse como un sistema de ecuaciones lineales tomando como incógnitas los elementos que conforman la semilla, los coeficientes de las incógnitas los elementos de M_t y el valor constante de igulación los elementos de la trama que se desea comprimir $B = [s_0, s_1, s_2, ..., s_{k-1}, s_k]$, tal como se muestra en la figura **4-7**.

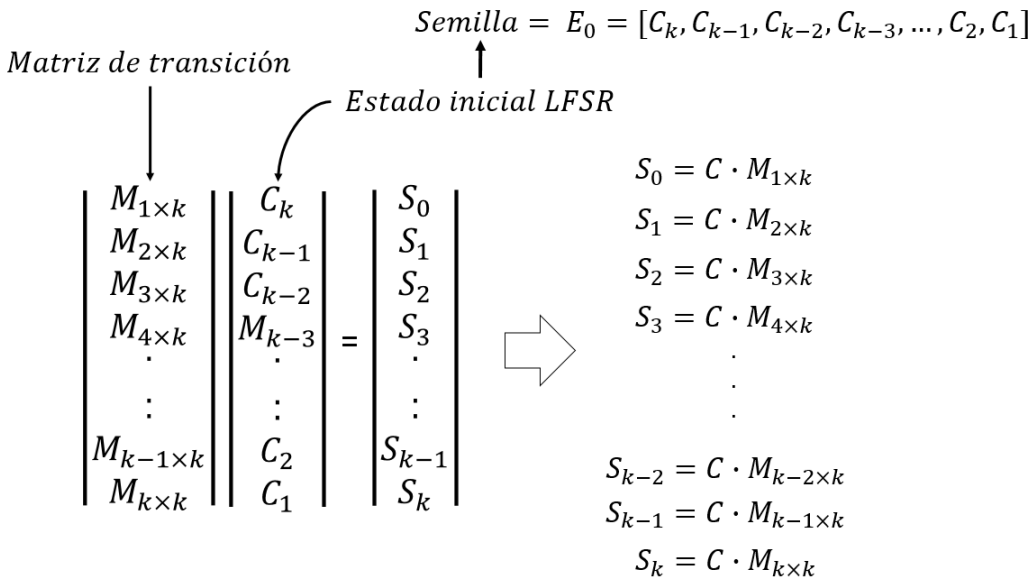


Figura 4-7.: Planteamiento de las ecuaciones necesarias para obtener la semilla E_0 que permita generar la secuencia $B = [s_0, s_1, s_2, ..., s_{k-1}, s_k]$.

Los primeros k bits de la secuencia s corresponden a la semilla E_0 debido a que circuitalmente los valores de E_0 entran al *LFSR* en forma paralela y salen en forma secuencial inmediatamente despúes de haber ingresado. Esto se ve reflejado matemáticamente en la matriz de transiciónes M_t por el hecho que sus primeras $k-1$ filas estén conformadas por un solo elemento no nulo, y que la última fila contenga al polinomio en formato binario. La salida de los bits del *LFSR* se desarrolla a un ritmo controlado por los ciclos de la señal de reloj conectado a cada flip flop D, que corresponde a cada iteración de la expresión $M_t \times E_n$.

Como se mencionó en la sección 4.2.1 para que exista compresión, la subsecuencia $s_n = B$ debe ser mayor que la semilla $C = E_0$. Por lo tanto, ubicar la subsecuencia s_n en los primeros k bits de la secuencia s generaría ecuaciones triviales según el planteamiento mostrado en la figura **4-7**. Esta situación no permitiría encontrar todos los bits requeridos de la semilla E_0. Como solución se plantea que cada bit de la trama B sea una combinación lineal de al menos dos bits de la semilla, lo cual se logra planteando el sistema de ecuaciones a partir del estado $n = k$. Finalmente para obtener la semilla se debe resolver el sistema de ecuaciones módulo dos mediante alguno de los muchos métodos disponibles en la bibliografía matemática.

4.2.3. Simplificación del planteamiento de las ecuaciones

Según lo expuesto en las secciones previas para obtener un estado cualquiera E_n es necesario calcular todos los estados anteriores $E_{n-1}, E_{n-2}, E_{n-3}, ..., E_1, E_0$, lo que supone gran canti-

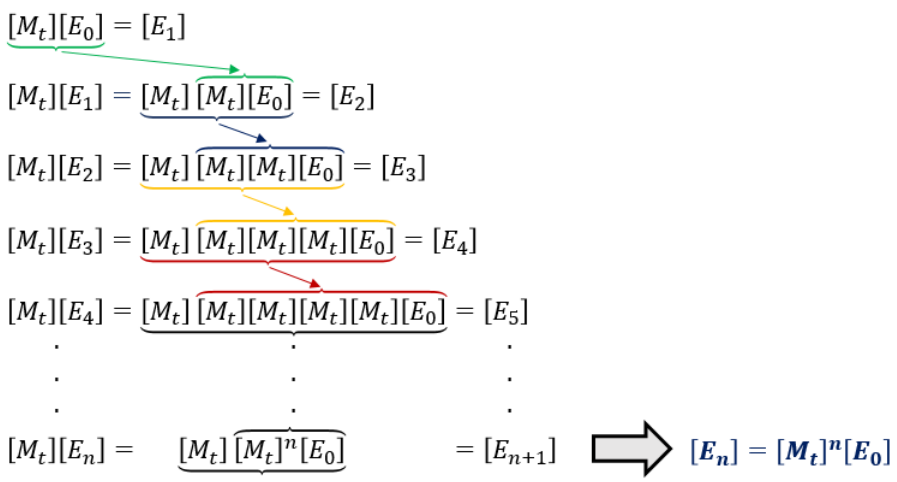


Figura 4-8.: Obtención del estado E_n mediante las potencias de la Matriz de Transición.

dad de cálculos en comparación a los deseados en un algoritmo de compresión para *WSN*. La figura **4-8** muestra el cálculo del estado E_n del *LFSR* mediante la utilización del estado inicial o semilla E_0 y la enésima potencia de la matriz de transición $[M_t]^n$, lo que finalmente reduciría la cantidad de cálculos que se deben efectuar.

M_t es una matriz cuadrada de tamaño $k \times k$, siendo k el tamaño del polinomio de retroalimentación del *LFSR*. Una característica útil de este tipo de matrices es que las filas de las potencias calculadas se desplazan en forma vertical ascendente (desde la fila k hasta la fila 1) con cada potencia iterada. Es decir que los valores de la fila k de $[M_t]^n$ serán los valores de la fila $k-1$ de $[M_t]^{n+1}$, $k-2$ de $[M_t]^{n+2}$, $k-3$ de $[M_t]^{n+3}$, $k-4$ de $[M_t]^{n+4}$, $k-5$ de $[M_t]^{n+5}$,..., $k-i$ de $[M_t]^{n+i}$. Si $k-i \leq 0$ entonces los valores de la fila k de la matriz de transición $[M_t]^n$ ya habrán salido de la matriz, es decir ya no pertenecerán a la matriz $[M_t]^{n+i}$. La ecuación 4-1 lo muestra en forma matemática.

$$\forall\ k-i \leq 0 \longrightarrow [M_t]^n_{k\times[1,\ldots,k]} \notin [M_t]^{n+i}_{k\times k} \tag{4-1}$$

Debido a que el primer bit del vector de estado $E_n[0]$ del *LFSR* es el que corresponde a la secuencia y la matriz de transición sufre un desplazamiento de filas hacia arriba [1], entonces se puede concluir que con una sola potencia de la matriz de transición $[M_t]^n$ se tienen k filas que al multiplicarlas por la semilla E_0 generarán los bits de la secuencia de salida con subindices desde el $k(n-1)$ hasta el $k(n-1)+(k-1)$ donde n es la potencia de la matriz de transición, como se muestra en el conjunto de ecuaciones 4-2.

[1] En forma vertical ascendente (desde la fila k hasta la fila 1) de la matriz

$$
\begin{aligned}
M^n_{1,k} \times E_0 &= S_{k(n-1)} \\
M^n_{2,k} \times E_0 &= S_{k(n-1)+2-1} \\
M^n_{3,k} \times E_0 &= S_{k(n-1)+3-1} \\
M^n_{4,k} \times E_0 &= S_{k(n-1)+4-1} \\
&\vdots \\
M^n_{j,k} \times E_0 &= S_{k(n-1)+j-1} \\
M^n_{j+1,k} \times E_0 &= S_{k(n-1)+(j+1)-1} \\
&\vdots \\
M^n_{k-2,k} \times E_0 &= S_{k(n-1)+(k-2-1)} \\
M^n_{k-1,k} \times E_0 &= S_{k(n-1)+(k-1-1)} \\
M^n_{k,k} \times E_0 &= S_{k(n-1)+(k-1)}
\end{aligned}
\tag{4-2}
$$

En la **4-9** se muestra un algoritmo optimizado para el cálculo de la potencia n de una matriz cuadrada.

4.2.4. Método de solución de ecuaciones de Gauss - Jordan para encontrar la trama comprimida B_c

El método de Gauss - Jordan es ampliamente utilizado para encontrar la solución de sistema de ecuaciones lineales debido a su potencia, efectividad y simplicidad respecto a otros métodos al momento de su implementación en software [25]. Para utilizarlo como método de solución del sistema de ecuaciones presentado en la sección 4.2.2 y 4.2.3 se deben realizar ciertos ajustes y consideraciones en la construcción de la matriz expandida y la forma como se ejecuta su solución.

La matriz expandida de Gauss - Jordan se construye a partir de las ecuaciones del producto $[M_t]^n \times E_0$ igualada al vector de bits $B = [S_0, S_1, S_2, ..., S_{k-1}]$ que se quiere calcular como se muestra en el conjunto de ecuaciones 4-3.

Procedure 2 Power of a matrix in a optimized way

Input: Square boolean matrix M and power n in base 10

Output: M^n

```
1 : b ← n in base 2        Ma = M
2 : lb ← long(b)
3 : if b_{lb-1} = 1 then
4 :       M_tmp = Ma
5 : else
6 :       M_tmp = [∅]
7 : end if
8 : for i = 1 to lb do
9 :       Ma = Ma × Ma
10:       if M_tmp = [∅] and b_{lb-1-i} = 1 then
11:         M_tmp = Ma
12:       else b_{lb-1-i} = 1
13:         M_tmp = M_tmp × Ma
14:       end if
15:  end for
16: return   M_tmp
```

Figura 4-9.: Pseudo - código para calcular la potencia de una matriz cuadrada con pasos reducidos.

$$
\begin{aligned}
[M^n_{1,k}] \times [E_0] &= S_0 \\
[M^n_{2,k}] \times [E_0] &= S_1 \\
[M^n_{3,k}] \times [E_0] &= S_2 \\
[M^n_{4,k}] \times [E_0] &= S_3 \\
&\vdots \\
M^n_{k-3,k} \times [E_0] &= S_{k-4} \\
M^n_{k-2,k} \times [E_0] &= S_{k-3} \\
M^n_{k-1,k} \times [E_0] &= S_{k-2} \\
M^n_{k,k} \times [E_0] &= S_{k-1}
\end{aligned} \tag{4-3}
$$

Debido a que la matriz es cuadrada con tamaño $k \times k$ entonces la trama original B debe ser fragmentada en subtramas de tamaño k. Si una de estas tramas queda incompleta se debe rellenar con bits *"don' t care"*. La trama B quedaría fragmentada idealmente como en la ecuacion 4-4.

$$
\begin{aligned}
B = [b_1 \cup b_2 \cup b_3 ... \cup b_j] \; con : & \\
B > b_j & \\
b_1, b_2, b_3, ..., b_j \subset B &
\end{aligned} \tag{4-4}
$$

Con el fin de simplificar el análisis se toma en consideración un solo bit de toda la trama B como se muestra en la ecuación 4-4.

$$
\begin{aligned}
b_{k(n-1)+i} &= E_0 \cdot M^n_{i,k} = [C_k, C_{k-1}, C_{k-2}, C_{k-3}, ..., C_2, C_1] \cdot [M_{i,1}, M_{i,2}, M_{i,3}, ..., M_{i,k-1}, M_{i,k}] \\
&= C_k \cdot M_{i,1} \oplus C_{k-1} \cdot M_{i,2} \oplus C_{k-2} \cdot M_{i,3}, ..., C_2 \cdot M_{i,k-1} \oplus C_1 \cdot M_{i,k}
\end{aligned} \tag{4-5}
$$

Es decir que cada bit de la secuencia B se puede calcular con una expresión semejante a la ecuación 4-5 compuesta por las operaciones lógicas AND ($\cdot$) y XOR ($\oplus$). El número de ecuaciones análogas a la 4-5 debe ser mayor o igual que el número de bits k de la semilla para aspirar a obtener una solución numérica del sistema.
Se observa en cada producto $M_{i,j} \cdot C_j$ que el coeficiente que hará parte de la matriz expandida es el término $M_{i,j}$. Por ejemplo, seleccionemos el polinomio primitivo e irreductible $p(x) = x^4 + x^3 + 1$ con el fin de realizar la compresión de la trama $B = [11110101]$. La semilla tiene la forma $E_0 = [C_4 C_3 C_2 C_1]$ y el polinomio en formato binario sería $p(x) = [1, 0, 0, 1]$.

$$(M_t)^4 \cdot E_0 = \begin{vmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix} \cdot \begin{vmatrix} C_4 \\ C_3 \\ C_2 \\ C_1 \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \end{vmatrix} \Longrightarrow \begin{array}{c} C_4 + C_1 = 1 \\ C_4 + C_3 + C_1 = 1 \\ C_4 + C_3 + C_2 + C_1 = 1 \\ C_4 + C_3 + C_2 = 1 \end{array}$$

$$(M_t)^4 \cdot E_0 = b1$$

$$B = b1 \cup b2$$

$$(M_t)^8 \cdot E_0 = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} C_4 \\ C_3 \\ C_2 \\ C_1 \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 0 \\ 1 \end{vmatrix} \Longrightarrow \begin{array}{c} C_3 + C_2 + C_1 = 0 \\ C_4 + C_2 = 1 \\ C_3 + C_1 = 0 \\ C_4 + C_2 + C_1 = 0 \end{array}$$

$$(M_t)^8 \cdot E_0 = b2$$

Figura 4-10.: Planteamiento del sistemas de ecuaciones a partir de un producto matricial.

La Matriz de Transición es como se muestra a continuación:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Para evitar las ecuaciones triviales se empieza el cálculo desde la potencia $n = k = 4$ de la matriz de transición, siendo k el tamaño o grado del polinomio. Con el signo + representando la operación XOR, quedaría como se muestra en la **4-10**.

Del sistema de ecuaciones que resulta se contruye la matriz expandida. Dicha matriz se compone de los coeficientes de las variables debidamente organizadas y de los valores constantes a los que están igualdos cada ecuación. Estos coeficientes son ceros y unos debido a que las ecuaciones son módulo 2. La figura **4-11** muestra la correspondencia que existe entre el sistemas de ecuaciones y los valores que finalmente componen la matriz expandida. Observe que los coeficientes de las ecuaciones se obtienen tan sólo calculando las potencias requeridas de M_t, lo que simplifica su implementación en software. La figura **4-12** muestra el algortimo para obtener la matriz expandida.

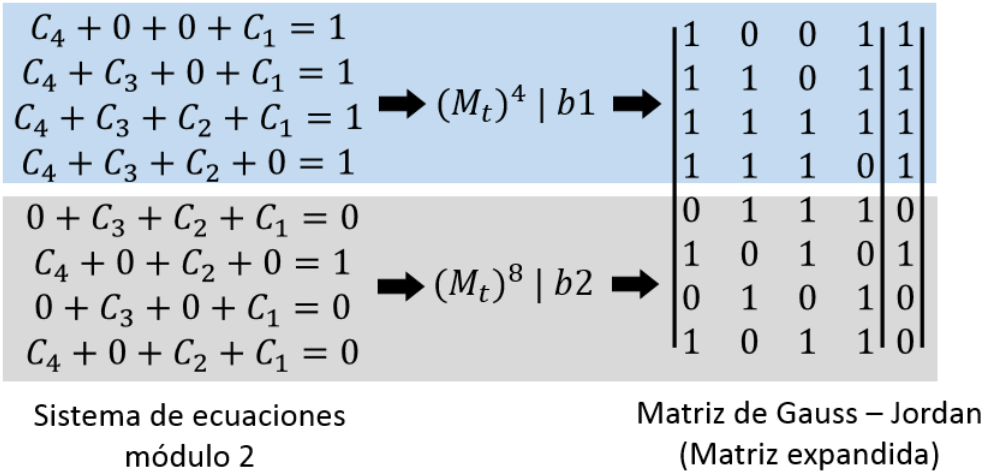


Figura 4-11.: Los elementos que componen la matriz expandida finalmente son la unión de los elementos de $(M_t)^8$, $(M_t)^4$ y B.

Procedure 3 Calculate of expand matrix

Input: Matrix transition M_t and secuence B

Output: Expand matrix

```
1 : lp ← long(M_t)      ls ← long(B)      z ← [∅]
2 : if residue(ls/lp) > 0 then
3 :       t ← entire(ls/lp) + 1
4 : else
5 :       t ← entire(ls/lp)
6 : end if
7 : for i = 1 to t do
8 :       insert all row of (M_t)^(i×lp) in z
9 : end for
10: for i = 1 to ls − 1 do
11:       insert B_i next to z_(i,lp)
12: end for
13: return z[1 to ls, 1 to lp + 1]
```

Figura 4-12.: Pseudo-código para obtener la matriz expandida a partir de M_t y B.

4.2.5. Ajustes al método de solución de ecuaciones de Gauss - Jordan para sistemas de ecuaciones módulo 2

Las tres operaciones elementales por renglones que pueden aplicar a una matriz aumentada son:

1. Multiplicar o dividir una fila por un número diferente de cero.

2. Sumar un múltiplo de una fila a otra fila.

3. Intercambiar dos filas.

Teniendo en cuenta las características y propiedades de los números binarios y sus consecuentes ecuaciones booleanas, las tres operaciones anteriores sufren las siguientes modificaciones:

1. La primera operación elemental queda sin efecto debido a que el único número válido por el que se podría multiplicar o dividir es 1, por lo que se obtendrían los mismos valores en la fila.

2. Esta segunda operación se reduce solo a sumar un renglón a otro, sin buscar múltiplos de otras filas debido al cambio en la primera operación. Además la suma se convierte en una operación de *Or Exclusiva* ($\oplus$).

3. La tercera operación no sufre cambios.

Entonces las operaciones elementales para matrices booleanas quedan de la siguiente forma:

1. Realizar la operación Or Exclusiva ($\oplus$) entre dos renglones.

2. Intercambiar dos renglones.

Al aplicar el método de Gauss - Jordan nos interesa la solución única del sistema, que de llegarse a encontrar concluiría en una compresión exitosa, de lo contrario no habría compresión. Es por eso que el algoritmo de Gauss - Jordan debe ser capaz de arrojar un indicador cuando no se encuentre una solución, casos que se presentan si en la diagonal principal no se puede ubicar un pivote igual a 1 o si al convertir la triangular inferior en ceros las filas inferiores a la triangular inferior no cumplen con la ecuación $0x = 0$ [9]. En la figura **4-12** se muestra el algoritmo del método de solución de ecuaciones Gauss - Jordan para ecuaciones booleanas.

Procedure 4 Gauss − Jordan to boolean matrix

Input: Expand matrix M
Output: Solutions of expand matrix

```
1 : f, c ← size(M)
2 : for i = 1 to c − 1 then
3 :     if pivote M_{i,i} ≠ 1 then
4 :         search pivote in all row, begin in i + 1 and use change of row
5 :         if not match pivote then BREAK ALGORITHM end if
6 :     end if
7 :     for k = 1 + i to f do
8 :         if M_{k,i} = 1 then M_{k,i...c} ← M_{k,i...c} ⊕ M_{i,i...c} end if
9 :     end for
10: end for
11: if M_{c+1...f,1 to c−1} satisface 0x = 1 then BREAK ALGORITHM end if
12: for k = c − 1 to 1 do
13:     for j = k − 1 to 0 do
14:         if M_{j,k} = 1 then M_{j,k} ← 0,    M_{j,c} ← M_{j,c} ⊕ M_{k,c} end if
15:     end for
16: end for
17: return M
```

Figura 4-13.: Pseudo-código para obtener la semilla E_0.

4.3. Pruebas para la evaluación del algoritmo inicial

Para verificar que efectivamente el algoritmo funciona correctamente se ha diseñado una sencilla prueba que consiste en comprimir tramas de las cuales se conoce su versión comprimida. Estas tramas fueron extraídas de la secuencia generada por el Polinomio Primitivo $p(x) = x^4 + x^3 + 1$ utilizando como semilla $C = 0001$ [26]. Por lo tanto según la teoría y análisis expuestos en las secciones y capítulos anteriores, dichas tramas de prueba deben provenir de alguna semilla correspondiente para dicho polinomio. La tabla **4-1** muestra dichas tramas de prueba y la salida del algoritmo.

B: **entrada de 5 bits**	B_c: **salida de 4 bits**
00000	0000
00100	1011
01000	0110
01100	1101
10001	1100
10101	0111
11001	1010
11101	0001
00011	1001
00111	0010
01011	1111
01111	0100
10010	0101
10110	1110
11010	0011
11110	1000

Tabla 4-1.: Tramas de prueba de 5 bits y salida del algoritmo de compresión.

Los resultados de la tabla **4-1** significan que si se construye un *LFSR* con polinomio $p(x) = x^4 + x^3 + 1$ entonces se obtendrá la trama B_j utilizando como semilla la correspondiente trama B_{cj}. Hay que tener en cuenta que el *LFSR* generá la trama B_j una vez la semilla B_{cj} haya salido del *LFSR*. Note además que la salida es un 20 % menor que la entrada, por lo tanto hubo una compresión efectiva.

5. Ajustes, implementación y pruebas

Para realizar la implementación del algoritmo y hacerlo realmente funcional es necesario llevar a cabo algunas modificaciones a la trama de salida. Además se debe establecer las condiciones de trabajo del algoritmo como el tamaño de la trama de entrada, el polinomio que se va a utilizar, entre otros detalles. Las siguientes secciones trata sobre cada uno de estos aspectos.

5.1. Criterio de selección del tamaño del polinomio $p(x)$ y la secuencia de entrada B

El tamaño de la semilla E_0 es igual al tamaño L del polinomio que junto al tamaño N de la secuencia de entrada B, determinan el porcentaje de compresión y la probabilidad de acierto del algoritmo. Así por ejemplo, sí a un compresor de datos entran 100 bits y salen 90 bits, se dice que hubo una compresión del 10 %, aunque algunos autores prefieren expresar la compresión como un factor que resulta de la razón entre los bits de entrada y los de salida, para este caso sería $100/90 = 1{,}11X$. Para este trabajo se utiliza el porcentaje, disponiendo de la ecuación normalizada 5-1, pudiendo convertir esta magnitud a porcentaje multiplicando por 100 si fuera necesario. L es el tamaño del polinomio de retroalimentación $p(x)$ del $LFSR$ y N el tamaño de la secuencia de entrada B.

$$C(N, L) = 1 - \frac{L}{N} \tag{5-1}$$

De esta forma para el ejemplo mencionado en el primer párrafo se tendría:

$$C(N, L) = 1 - \frac{90}{100} = 1 - 0{,}9 = 0{,}1 \longrightarrow 0{,}1 \times 100 = 10\,\%$$

Por otro lado asumiendo inicialmente que la salida del algoritmo no requiere modificación ni tratamiento alguno se puede calcular la probabilidad de acierto como la razón entre todos los

casos exitosos y todos los casos posibles, es decir todas las posibles secuencia que un *LFSR* puede generar dividido entre todas las posibles tramas de entrada al compresor. Es aquí donde se destaca la importancia de seleccionar un polinomio primitivo e irreductible para la retroalimentación del *LFSR*, pues de esta forma se obtendrá un periodo máximo en la salida, es decir mayor cantidad de casos exitosos que contribuirán a aumentar la probabilidad de acierto. La ecuación 5-2 muestra dicha probabilidad.

$$P(N, L) = \frac{2^N - 1}{2^L - 1} \tag{5-2}$$

En el denominador de la ecuación 5-2 se resta uno descartando tramas compuestas sólo de ceros, debido a que este es sería un caso de compresión trivial en el que sí todos los bits de entrada son ceros entonces en la salida también habrá ceros. Entonces, a partir de lo expuesto anteriormente en relación al grado de compresión y la probabilidad de acierto, la pregunta que surge es:

- ¿Cuáles son los valores de N y L que maximizan $P(N, L)$ y $C(N, L)$?

Las soluciones analíticas se descartan debido a que hay 4 variables y sólo 2 ecuaciones. Debido a esto es conveniente establecer alguna relación válida entre N y L, la cual se propone en la ecuación 5-3.

$$N = L + d \tag{5-3}$$

Siendo d la cantidad de bits que diferencian la longitud L del polinomio $p(x)$ y la longitud N de la secuencia de entrada B. De esta forma según la ecuación 5-1 entre mayor sea d mayor será la compresión, sin embargo menor será la probabilidad de acierto según la ecuación 5-2. Además estas relaciones dependen también del tamaño del polinomio. Algunos valores propuestos se muestran en la tabla **5-1**.
La variable d también puede interpretarse como la cantidad de bits que eliminó el compresor de la secuencia original B. En la tabla **5-1** se observa que a medida que la compresión disminuye la probabilidad de acierto aumenta. Las tres primeras filas de la tabla muestran que con los parámetros escogidos el algoritmo está muy lejos de realizar una compresión efectiva debido a la extremadamente baja probabilidad de acierto. Por otro lado la cuarta fila muestra una probabilidad aceptable con la desventaja que es para comprimir sólo 1 bit que representa el 1,7 % de la trama de entrada. Ahora bien, según la literatura consultada con sólo 1 bit que se comprima en un nodo sensor se obtiene ganancia a nivel global en la

B **(bits)**	$p(x)$ **(bits)**	E_0 **(bits)**	d **(bits)**	$C(N,L)$ **(%)**	$P(N,L)$ **(%)**
60	30	30	30	50	0,0000001
60	40	40	20	33	0,0001
60	50	50	10	17	0,1
60	59	59	1	1,7	50

Tabla 5-1.: Compresión y probabilidad de acierto con algunos valores de entrada al algoritmo.

WSN debido al ahorro energético que habría en los nodos que retransmiten y no comprimen. Observe que este último resultado favorable se obtiene a costo de un algoritmo que tiene trabajar con matrices de tamaño 59×59, lo que podría ralentizar el proceso de compresión y generar mayor consumo de energía debido a la complejidad de los cálculos en comparación con otros métodos. Con el fin de encontrar los valores más óptimos para N y L en la figura **5-1** se grafican las ecuaciones 5-4 y 5-5 que son la combinación de las ecuaciones 5-3, 5-2, 5-1.

$$C(L,d) = 1 - \frac{L}{L+d} \tag{5-4}$$

$$P(L,d) = \frac{2^L - 1}{2^{L+d} - 1} \tag{5-5}$$

La figura **5-1** es reveladora pues muestra claramente las tendencias y comportamientos que tienen el grado de compresión y la probabilidad de acierto. Las líneas continuas [1] de la gráfica corresponden a la compresión y las líneas discontinuas [2] corresponden a la probabilidad de acierto, ambas en funciones del tamaño L del polinomio $p(x)$ para distintos valores de la variable d. El número que acompaña a las letras C y P que dan nombre a cada curva, es el valor de la variable d que se usó para graficar la correspondiente línea. Así por ejemplo la línea continua C6 y discontinua P6 de color verde corresponden a la compresión y probabilidad de acierto respectivamente para una diferencia de 6 bits entre N y L.
En general la gráfica muestra que para todos los valores de d utilizados, la probabilidad de acierto tiende a estabilizarse rápidamente al aumentar L, es decir que la familia de ecuaciones 5-5 se acercan de forma considerable a su asíntota horizontal. Tanto así, que por ejemplo, para valores pequeños de estas variable como: $L = 4$ y $d = 1$, la función de probabilidad alcanza el 96 % del valor de su asíntota horizontal, es decir un valor de $0,48$ siendo la asíntota horizontal $P = 0,5$. Por otro lado el nivel de compresión desciende dramáticamente a medida que aumenta el tamaño L del polinomio. De hecho en la función 5-4, la variable d se vuelve

[1] Las curvas que inician en la coordenada $(0,1)$
[2] Las curvas que inician en la coordenada $(0,0)$

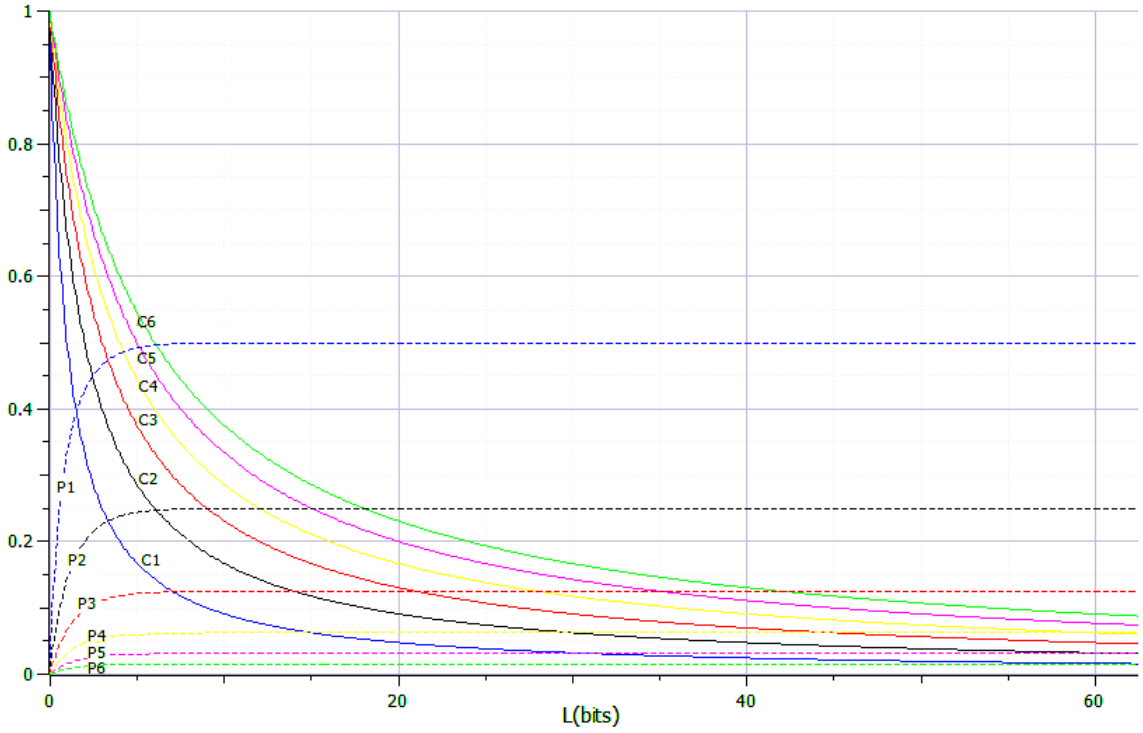


Figura 5-1.: Probabilidad de acierto y grado de compresión *vs* tamaño del polinomio. Ambas funciones se estabilizan a medida que el tamaño L del polinomio aumenta.

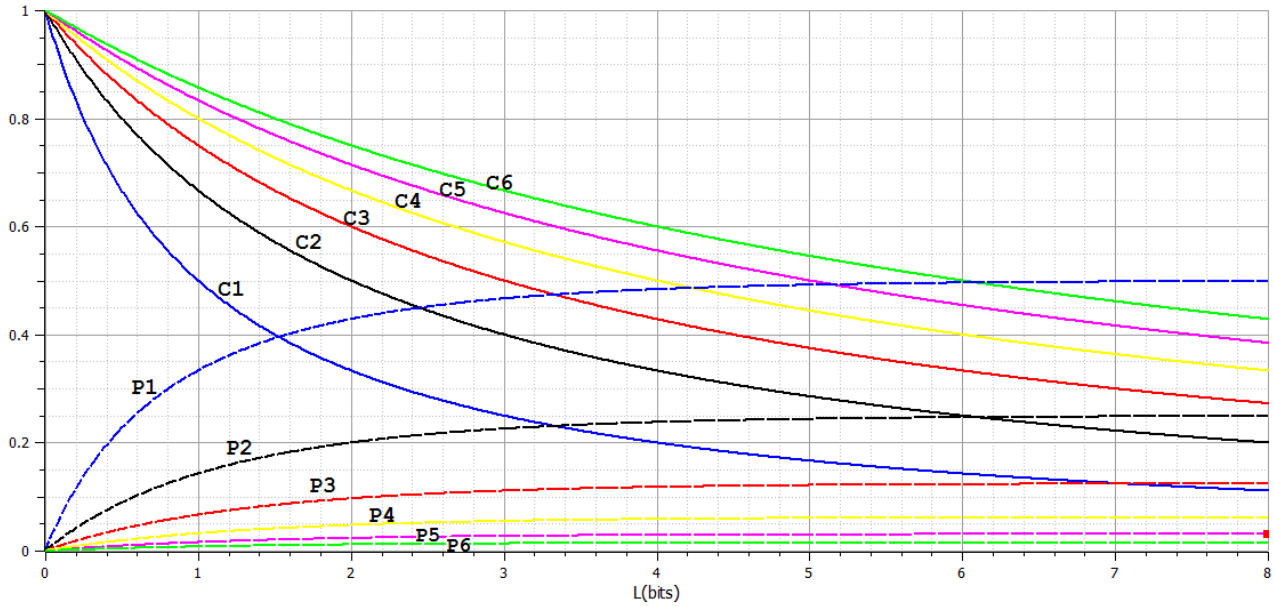


Figura 5-2.: Zona recomendada de operación para el *LFSR* del algoritmo presentado en este trabajo.

menos significativa para valores grandes de L, y la función queda $C = 1 - L/L = 0$.
Siendo métricas fundamentales de diseño P y C, y según las observaciones anteriormente expuestas se determina que la zona óptima de trabajo del algoritmo se da con polinomios de grados menores a 10. La figura **5-2** es un acercamiento de la figura **5-1** para $L < 8$. Se seleccionan entonces los siguientes valores $L = 4$ y $d = 1$, por lo tanto $N = 5$. De esta forma se obtiene:

$$C(L,N) = 1 - \frac{4}{5} = 0{,}2 \rightarrow 20\,\% \qquad P(L,N) = \frac{2^4 - 1}{2^5 - 1} = \frac{15}{31} = 0{,}48 \rightarrow 48\,\%$$

Comparando estos resultados con los de la tabla **5-1** observe que se ha mantenido aproximadamente igual la probabilidad de acierto $P(L,d)$, aumentando relativamente el porcentaje de compresión de datos $C(L,d)$ casi 12 veces. Además debido a los valores de L, d y N, el tamaño de las matrices que tienen que ser resueltas por el algoritmo son 10 veces menor.

5.2. Construcción final de la trama de salida B_c del algoritmo

En la sección anterior se determinó el tamaño del polinomio $p(x)$ y la secuencia de entrada B con que debería operar el algoritmo para aumentar los beneficios en cuanto a nivel de compresión, probabilidad de acierto y tamaño de las matrices que intervienen en los cálculos. Sin embargo debido a que la probabilidad de acierto obtenido es del 48 % y no del 100 %, se prevé que pueda fallar en algún momento con una certeza del 52 %. Por esto se debe agregar a la trama de salida B_c una marca que indique el éxito o falla del proceso de compresión, como se muestra en la figura **5-3**. Este bit B_{sf} debe ser el primero de la trama para indicar al descompresor cual procedimiento llevar a cabo los bits subsiguientes y obtener el dato original.

Si se omite el envío del bit éxito/falla B_{sf} y se transmite una trama con compresión fallida, el algoritmo de descompresión probablemente generará datos erróneos que distorsionarán la información original, dando lugar a traumatismos indeseados. El bit que se agrega a la trama de salida suprime el efecto de compresión que tiene el algoritmo inicial, debido a que se está quitando y luego agregando un bit a B_c. Para mitigar este efecto se toma la trama de entrada con longitud de 10 bits y se divide en dos subtramas de igual tamaño que a las que se les aplica el algoritmo de compresión de forma individual. Entonces las métricas globales cambian para esta nueva trama de entrada, como se muestra a continuación:

$$C(L,N) = 1 - \frac{4+4+1}{10} = 0{,}1 \rightarrow 10\,\% \qquad P(L,N) = \frac{15}{31} \times \frac{15}{31} = 0{,}234 \rightarrow 23{,}4\,\%$$

Figura 5-3.: Posibles salidas del algoritmo de compresión teniendo en cuenta el bit de éxito/falla B_{sf}.

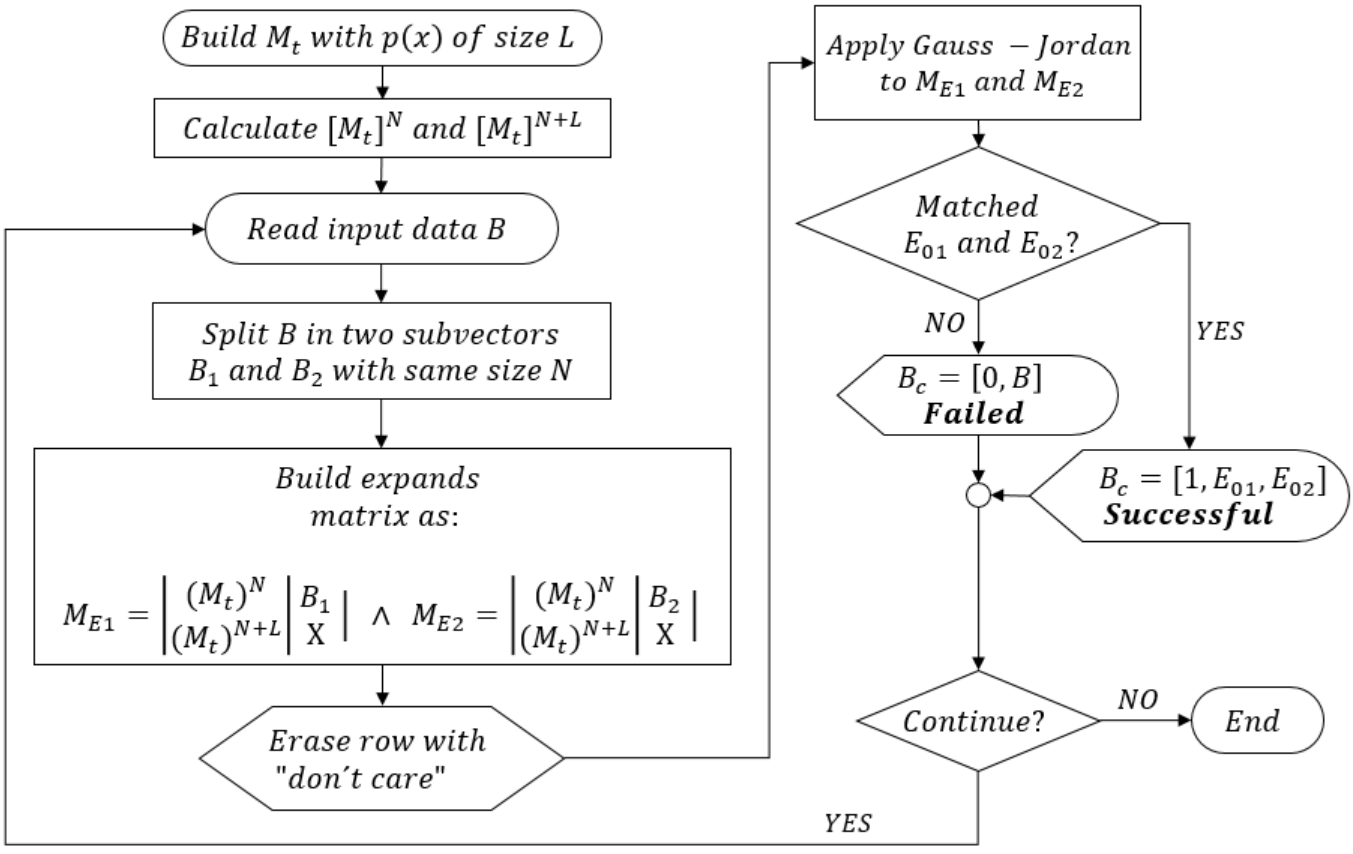


Figura 5-4.: Diagrama de flujo del algoritmo de compresión ***LFSR Algorithm v1.0.*** propuesto en este documento.

Es decir que en caso de éxito la compresión sería del 10 %, debido a que de cada una de las subtramas de entrada de 5 bits B_1 y B_2 generarían tramas de salida de 4 bits B_{c1} y B_{c2} (también pueden ser llamadas E_{01} y E_{02} respectivamente, por su correspondencia con la semilla del *LFSR*), a las que se le agrega el bit de éxito/falla B_{sf}. Por otro lado, La probabilidad total se calcula como la probabilidad de que las subtramas B_1 y B_2 sea comprimida exitosamente de forma individual. Un esquema general del funcionamiento del algoritmo puede ser visto en la figura **5-4**. Este diagrama de flujo muestra el proceso de compresión, el cual se repite en caso que el dispositivo que lo ejecuta así lo determine según el tamaño de los datos que deba comprimir.

5.3. Implementación en software y pruebas de concepto en dispositivo IoT

El algoritmo de compresión presentado en esta investigación se ha puesto a prueba con el lenguaje de programación *Python 3* compilado con el *IDE Python Community Edition 2017.2.1*. *Python* permite una implementación rápida y la integración efectiva con sistemas *IoT* como la tarjeta de desarrollo académico *Raspberry Pi 2 Model B*, también aquí utilizada.

Las pruebas se llevaron a cabo con secuencias aleatorizadas de *1 Mbits (125 kBytes)* de longitud, extraídas de una base de datos de tramas de 10 bits. Cada prueba está identificada con el nombre dB_j, en donde j indica el número de tramas *incompresibles* por el algoritmo. De este modo por ejemplo dB_2 corresponde a la prueba en la cual el algoritmo procesó una secuencia de *1 Mbits* de longitud, construida a partir de la selección aleatoria[3] de tramas de 10 bits extraídas de una base de datos constituida por las 16 tramas compresibles y 2 tramas incompresibles. La base de datos está disponibles en la tabla **A-1** y **A-2** del Anexo **A** de este documento. El LFSR utilizado tiene como polinomio de retroalimentación a $p(x) = x^4+x^3+1$.

5.3.1. Implementación en Laptop

Inicialmente se probró el algoritmo en un Laptop, el cual es un dispositivo *IoT* por excelencia debido a su capacidad de procesamiento, almacenamiento, interfaz de usuario y conexión a internet. Este dispositivo posee un procesador *Intel Core i5-5200U* a *2.20GHz x64* con *6GB RAM DDR3* y sistema operativo *Windows 8.1 pro 64bits*.
La tabla **5-2** muestra los resultados para la prueba del algoritmo de la figura **5-4** compilado en el laptop. Esta es la prueba dB_0, lo que quiere decir que la base de datos utilizada para construir la secuencia de entrada no contenía tramas incompresibles, en otras palabas todas las tramas de 10 bits de la secuencia de entrada eran 100 % compresibles, como lo evidencia

[3]Todas las tramas tienen la misma probabilidad de ser escogidas.

la cantidad de éxitos general gC. Por esta razón la compresión general gC fue del 10 %, es decir del máximo previsto en la sección 5.2. El tiempo promedio aT para la compresión de una trama de 10 bits fue de 327 ± 58 μs lo que corresponde a una tasa de compresión cR de $30,58 \pm 5,58$ $kbps$. La cantidad de bits que el algoritmo ahorra al sistema de transmisión/recepción son $sB = 100$ kb.

Test dB_0		
Average Compression	aC (%)	10 ± 0
Average Success	aS (%)	100 ± 0
General Compression	gC (%)	10
General Success	gS (%)	100
Average Time per 10 bits frame	aT (μs)	327 ± 58
Compression Rate	cR ($kbps$)	$30,58 \pm 5,58$
Saved bits	sB (kb)	100

Tabla 5-2.: Características del algoritmo propuesto utilizando como entrada una secuencia de datos 100 % compresible con longitud 1000 kbits.

La tabla **5-3** muestra el comportamiento del algoritmo en la implentación en Laptop a medida que se agregan tramas incompresibles a la secuencia de entrada. La base de datos está compuesta de 16 tramas compresibles y de j tramas incompresibles. La primera fila de la tabla **5-3** es el caso ideal en donde toda la secuencia se puede comprimir, los valores de las columnas aC y aS son bastante parecidos a las columnas gC y gS por lo tanto sólo se tendrán en cuenta estas dos últimas columas mencionadas para las próximas gráficas y tablas. En las columnas aC y gC el valor negativo significa que ha habido una compresión fallida a un porcentaje allí indicado, lo que se traduce en un incremento en la trama de salida que se puede observar en la columna sB. Las columnas aT y cR muestran el tiempo medio de compresión por trama y la cantidad de *kilobits* que el algoritmo puede procesar en un segundo.

La figura **5-5** es una gráfica normalizada de las columnas gC, gS, aT, cR vs j de la tabla **5-3**. Allí, se ve claramente la corelación inversa entre la cantidad de tramas incompresibles j en la secuencia de entrada y el porcentaje medio de compresión gC del algoritmo. Esta misma correlación inversa también existe con la tasa media de éxitos gS, la cual para un valor de $j = 16$ llega finalmente al valor previsto en la sección 5.2 para la probabilidad de aciertos $P(N, L)$, es decir alrededor del 25 %. El tiempo medio aT empleado para comprimir una trama de 10 bits no parece verse afectado por el valor de j, lo mismo sucede con la tasa de compresión cR.

j	**aC** (%)	**aS** (%)	**gC** (%)	**gS** (%)	**aT** (μS)	**cR** ($kbps$)	**sB** (kb)
0	10,0	100,0	10,0	100,0	327,0	30,6	100
1	7,8	88,8	7,7	88,7	310,8	32,2	77,4
2	5,8	79,1	5,8	79,1	310,2	32,2	58,1
3	4,2	71,1	4,2	71,0	308,3	32,4	42,0
4	2,8	63,8	2,8	64,0	313,1	31,9	27,9
5	1,7	58,4	1,6	58,1	309,1	32,4	16,2
6	0,6	52,8	0,6	53,1	305,6	32,7	6,1
7	-0,4	48,2	-0,3	48,5	309,4	32,3	-3,1
8	-1,1	44,6	-1,1	44,4	313,1	31,9	-11,2
9	-1,8	41,2	-1,8	41,1	310,9	32,2	-17,7
10	-2,4	37,9	-2,4	38,1	316,6	31,6	-23,9
11	-3,0	35,2	-3,0	35,2	312,0	32,0	-29,5
12	-3,4	33,1	-3,4	32,9	324,4	30,8	-34,1
13	-3,9	30,4	-3,9	30,5	313,6	31,9	-39,0
14	-4,3	28,6	-4,3	28,6	322,5	31,0	-42,8
15	-4,6	27,0	-4,7	26,5	309,1	32,4	-47,0
16	-5,0	25,2	-5,0	25,2	306,1	32,7	-49,6

Tabla 5-3.: Características del algoritmo propuesto utilizando como entrada una secuencia de 1000 $kbits$ construida con tramas aleatorias de una base de datos dB_j. Siendo j el número de tramas incompresibles por el algoritmo y $j + 16$ el tamaño total de la base de datos. Software probado en un laptop.

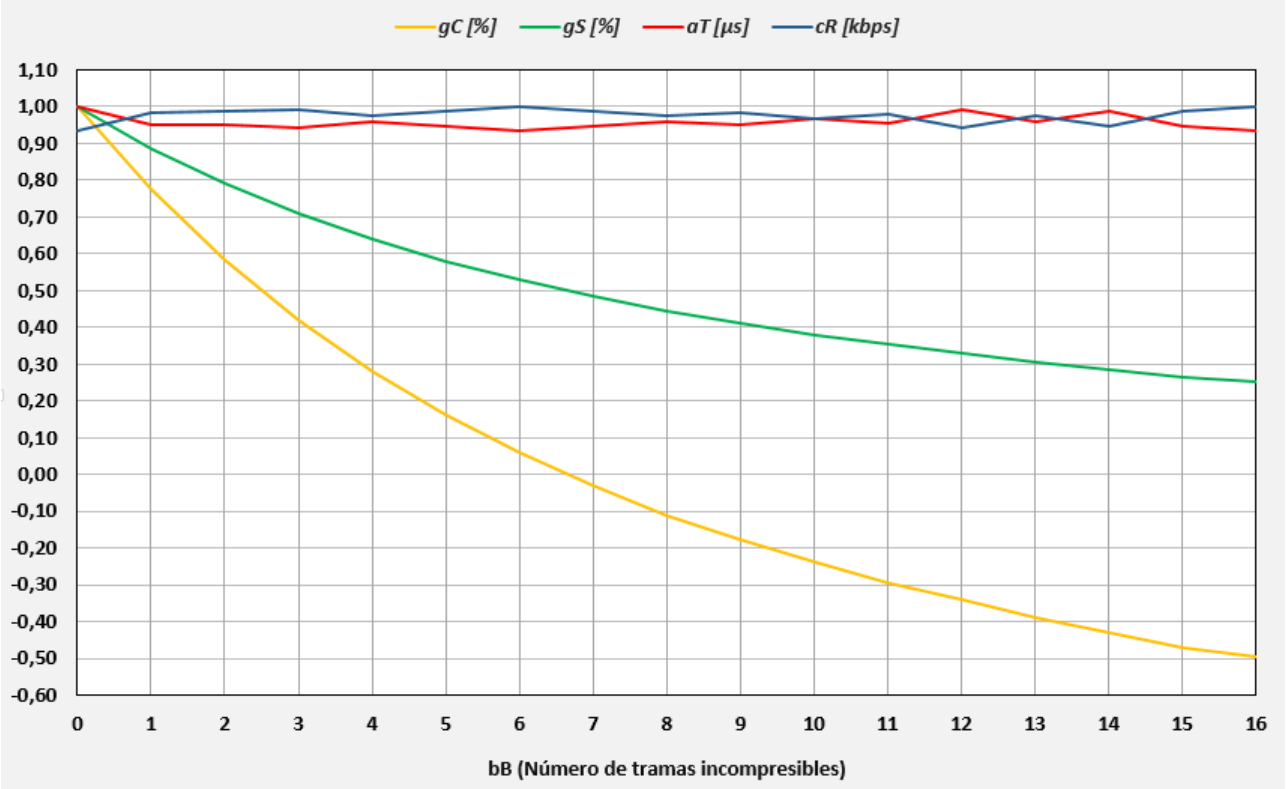


Figura 5-5.: Gráfica normalizada de las columnas gC, gS, aT, cR vs j de la tabla **5-3**. Software probado en un laptop. Las unidades corresponden a la variable sin normalizar.

De la información que proveen estas pruebas se puede determinar que el algorimo tiene efectividad de comprimir sólo si el número de tramas incompresibles j utilizados para construir la secuencia de entrada es menor o igual a 6. Esta 6 tramas incompresibles sumadas a las tramas compresibles daría un total de 22 tramas de 5 bits, lo que representa el 68,68 % de todos los valores posibles en la entrada. Hay que destacar que debido a que todas las tramas j fueron seleccionadas al azar no importa cuales 6 de todas las 16 tramas incompresibles sean seleccionadas.

Bajo estas condiciones la tasa de compresión es variable y puede alcanzar valores negativos lo que significa que la salida tiene más bits que la entrada. Note que para una tasa media de éxitos gS del 50 % el porcentaje de compresión gC ronda el 0 %, lo cual se entiende que sucede porque si se quitan bits de la secuencia de entrada con las compresiones existosas y se agrega esta misma cantidad de bits con las compresiones fallidas entonces no habrá una compresión general efectiva.

5.3.2. Implementación en sistema embebido

La misma prueba realizada en el Laptop, fue llevada a cabo en una tarjeta de desarrollo acádemico Raspberry Pi Modelo B cuyas características son: bajo costo, single-board computer, velocidad procesador 900 MHz, 1 GB RAM, 40 pines de extensión GPIO, puerto Ethernet, 4 puertos USB, puerto de cámara CSI, puerto display DSI y un puerto micro SD, la tarjeta utilizó el Raspbian OS. Muchas de estas tarjetas están siendo utilizadas para desarrollar prototipos de sistemas de monitoreo eléctricos [22], medición de temperatura, y un sin número de variables ambientales.
De los resultados se observa que el Laptop ha sido más rápido que la Raspberry debido a que los tiempos de ejecución aT de la tabla **5-3** son 10 veces menores que los valores de aT en la tabla **5-4**. Esto se ve más claramente al comparar entre ambas tablas los valores registrados para la tasa de compresión cR. Esta situación puede presentarse debido a las características de ambos dispositivos, tales como frecuencia del reloj de la CPU, memoria RAM, gestión del sistema operativo son distintos y con una marcada diferencia.

dB	**gC** (%)	**gS** (%)	**aT** (mS)	**cR** ($kbps$)	**sB** (kb)
0	10	100	2,88	3,47	100
1	7,7	88,5	2,87	3,48	77,1
2	5,8	78,8	2,88	3,48	57,6
3	4,2	70,9	2,86	3,49	41,8
4	2,8	64,1	2,87	3,48	28,1
5	1,6	58,2	2,87	3,49	16,5
6	0,6	52,9	2,86	3,5	5,7
7	-0,3	48,4	2,86	3,5	-3,1
8	-1,1	44,5	2,86	3,5	-10,9
9	-1,8	41	2,86	3,5	-18,1
10	-2,4	37,9	2,85	3,51	-24,2
11	-3	35,2	2,86	3,5	-29,5
12	-3,4	32,9	2,86	3,5	-34,3
13	-3,9	30,5	2,84	3,52	-38,9
14	-4,3	28,6	2,85	3,51	-42,9
15	-4,7	26,7	2,85	3,51	-46,6
16	-5	25,1	2,85	3,51	-49,8

Tabla 5-4.: Características del algoritmo propuesto utilizando como entrada una secuencia de 1000 $kbits$ construida con tramas aleatorias de una base de datos dB_j. Siendo j el número de tramas incompresibles por el algoritmo y $j + 16$ el tamaño total de la base de datos. Software probado en un Raspberry Pi Model B.

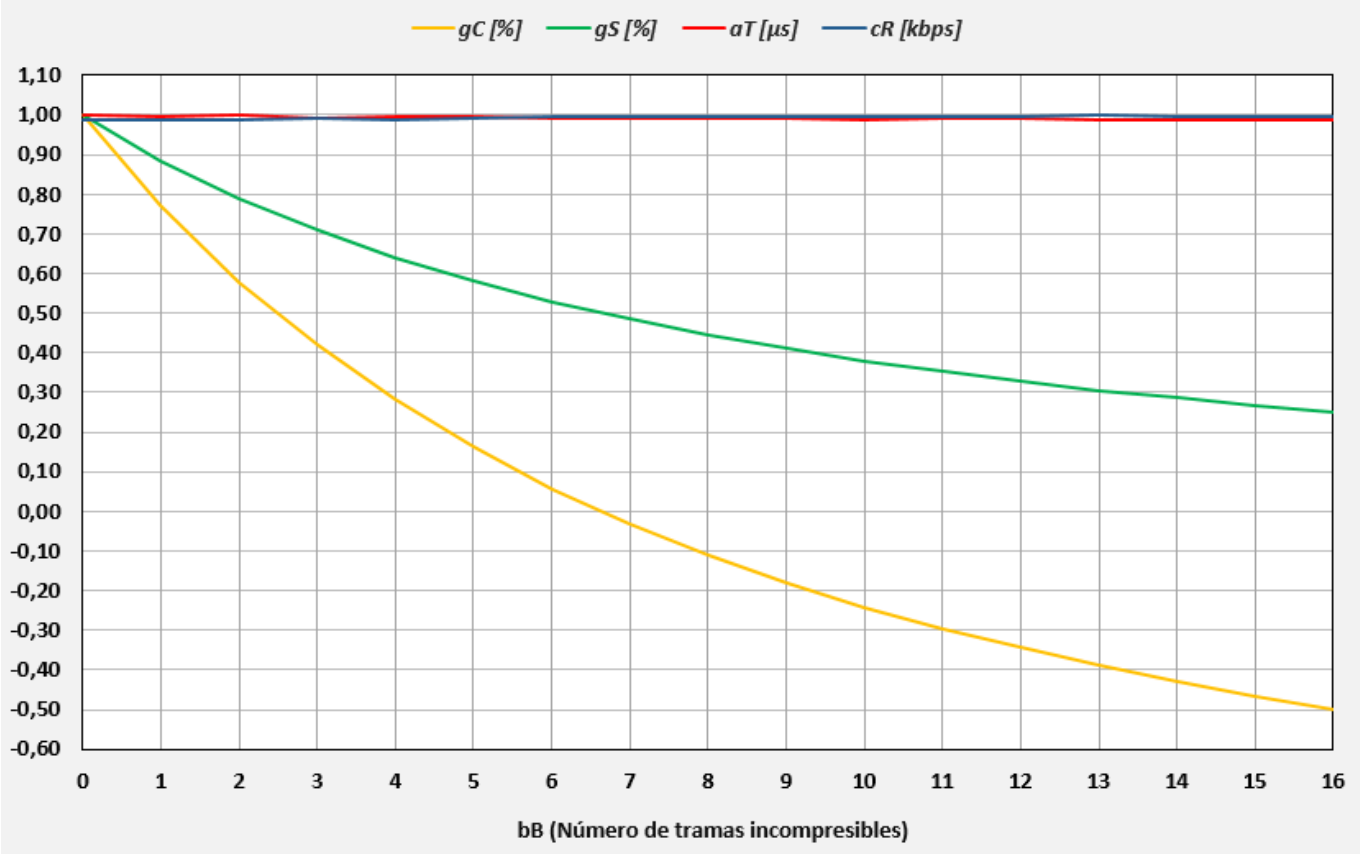


Figura 5-6.: Gráfica normalizada de las columnas gC, gS, aT, cR vs j de la tabla **5-4**. Software probado en un Raspberry Pi Model B. Las unidades corresponden a la variable sin normalizar.

Continuando con las demás columnas de la tabla hay que destacar que gC, gS y aT son casi iguales para aambos dispositivos. La figura **5-6** es una gráfica normalizada de las columnas gC, gS, aT, cR vs j de la tabla **5-4**. Se observa que el algoritmo tiene los mismos patrones de comportamiento que en el Laptop, es decir que será efectivo siempre y cuando se cumpla que $j \leq 6$ y que todas las $16 + j$ tramas tengan la misma probabilidad de ser seleccionadas para hacer parte de la secuencia de entrada.

5.4. Consideraciones energéticas para un sólo evento de sensado, compresión y transmisión

Un análisis energético del algoritmo está propuesto como trabajo futuro. Sin embargo, sabiendo que la reducción de la cantidad de bits a transmitir conlleva a una disminución en la potencia requerida por el módulo de transmisión (ver figura **3-5**) se considera pertinente el planteamiento de la siguiente pregunta:

- **¿Cuantos nodos deben retransmitir la secuencia comprimida para que exista ahorro en el consumo de energía eléctrica a nivel global en la WSN?**

El caso del consumo de energía cuando no se ejecuta el proceso de compresión se puede expresar simplificadamente mediante la ecuación 5-6:

$$E_F = n \times E_{TB} \tag{5-6}$$

Siendo E_F la energía total requerida por la *WSN* para transmitir la secuencia original de datos desde el nodo de origen hasta el nodo de destino, E_{TB} la energía necesaría para que un nodo sensor retransmita la secuencia original a otro nodo y n es el número de nodos de la *WSN* que transmitieron la secuencia original. Para el caso en que se aplica el proceso de compresión exitosamente el consumo de energía de la *WSN* puede ser descrito mediante la ecuación 5-7:

$$E_{FC} = E_C + E_D + n \times E_{TB_c} \tag{5-7}$$

Con E_{FC} como la energía total requerida por la *WSN* para transmitir la secuencia de datos comprimida B_c desde el nodo de origen hasta el nodo de destino, E_{TB_c} la energía necesaría para que un nodo sensor retransmita la secuencia de datos comprimida B_c a otro nodo, E_c la energía utilizada para comprimir la secuencia de datos original, E_D la energía utilizada en descomprimir B_c para obtener la secuencia de datos original y n es el número de nodos de la *WSN* que transmitieron la secuencia B_c. Se plantea ahora la desigualdad 5-8 entre la ecuación 5-7 y la 5-6:

$$E_{FC} < E_F \tag{5-8}$$

Reemplazando sus equivalentes correspondientes y despejando se obtiene:

$$E_C + E_D < n(E_{TB} - E_{TB_c}) \Longrightarrow con\ E_{TB} - E_{TB_c} = E_{Saved} = E_S$$
$$E_C + E_D < n \times E_S$$

Nuevamente despejando queda la inecuación 5-9, la cual, justifica todas optimizaciones al proceso de compresión y descompresión tanto para simplificar el procedimiento como para incrementar sus beneficios en términos cantidad de bits ahorrados. Entonces si $E_C+E_D \leq E_S$ habría ahorro desde la transmisión del nodo de origen.

$$n > \frac{E_C + E_D}{E_S} \tag{5-9}$$

Ahora, si la compresión se lleva a cabo y no es exitosa, dado el diseño actual del algoritmo la trama de salida ($B_{SALIDA} > B_{ENTRADA}$) se trasmitiría bajo un esquema de consumo energético descrito en la ecuación 5-10, en donde se espera con mucha certeza que $E_{TB_{cf}} > E_{TB_c}$, siendo $E_{TB_{cf}}$ la energía necesaría para que un nodo sensor retransmita la secuencia de datos comprimida fallidamente B_{cf} a otro nodo y E_{FC_f} la energía total requerida por la *WSN* para transmitir la secuencia de datos comprimida fallidamente B_{cf} desde el nodo de origen hasta el nodo de destino.

$$E_{FC_f} = E_C + E_D + n \times E_{TB_{cf}} \tag{5-10}$$

5.5. Ajustes finales a la secuencia de salida según las restricciones encontradas

Con base en los resultados obtenidos y el análisis descriptivo de la sección 5.3 se puede concluir que el algoritmo funciona si la secuencia de entrada está conformada por subtramas de 5 bits que hagan parte de una base de datos de máximo 22 valores, de los cuales 16 sean 100 % compresibles. Además todos los valores dentro de la base de datos deben tener una distribución de probabilidad uniforme. Si estas condiciones no se cumplen, el algoritmo está expuesto a fallar, lo cual significa que va a tener como salida una secuencia de mayor longitud que la secuencia de entrada, y teniendo en cuenta las consideraciones energéticas de la seccion 5.4, se hace prudente realizar alguna modificación a la secuencia de salida que permita mitigar el impacto energético negativo decrito por la ecuación 5-10, la cual nos muestra que la energía consumida por la *WSN* cuando se transmite una trama comprimida fallidamente es mayor que transmitir la trama sin la aplicación del algortimo.

Debido a esto se propone agregar a la secuencia transmitida un bit que indique que la compresión a nivel general fue exitosa o no. La figura **5-7** muestra el diagrama de flujo de un algoritmo que ayuda mitigar los efectos adversos de una compresión no exitosa de parte del ***LFSR Algorithm v1.0***. Note que el algoritmo de la figura **5-7** ejecuta el *LFSR Algorithm v1.0* por lo tanto la energía E_C de la ecuación 5-10 se mantiene sin importar que el resultado de la compresión sea exitoso o fallido. Por otro lado la energía E_D se reduce sólo a la cantidad de energía que el descompresor del nodo de destino necesita para identificar si el primer bit de la secuencia que llega es 1 o 0. Por ejemplo procesar 1 bit en comparación con el millón de bits que se procesaron en cada prueba de la sección 5.3 sólo representa el 0,0001 %.
Aún mejor es lo que sucede con la longitud de salida en caso de una falla en la compresión pues incrementa en tan sólo 1 bit. Compare este incremento con los 3100 bits que aumentaría si no se aplicase este segunda parte del algoritmo a la secuencia de *1Mbits* para dB_7 en la tabla **5-4**. Al aplicar de forma teórica el ***LFSR Algorithm v1.1*** a la tabla **5-4** se obtiene la tabla **5-5** y la gráfica **5-9**.

dB	**gC**	**gS**	**aT**	**cR**	**sB - v1.0**	**sB - v1.1**
-	(%)	(%)	(mS)	($kbps$)	(kb)	(kb)
0	10	100	2,88	3,47	100	100
1	7,7	88,5	2,87	3,48	77,1	77,1
2	5,8	78,8	2,88	3,48	57,6	57,6
3	4,2	70,9	2,86	3,49	41,8	41,8
4	2,8	64,1	2,87	3,48	28,1	28,1
5	1,6	58,2	2,87	3,49	16,5	16,5
6	0,6	52,9	2,86	3,5	5,7	5,7
7	-0,3	48,4	-	-	-3,1	-0,001
8	-1,1	44,5	-	-	-10,9	-0,001
9	-1,8	41	-	-	-18,1	-0,001
10	-2,4	37,9	-	-	-24,2	-0,001
11	-3	35,2	-	-	-29,5	-0,001
12	-3,4	32,9	-	-	-34,3	-0,001
13	-3,9	30,5	-	-	-38,9	-0,001
14	-4,3	28,6	-	-	-42,9	-0,001
15	-4,7	26,7	-	-	-46,6	-0,001
16	-5	25,1	-	-	-49,8	-0,001

Tabla 5-5.: Al aplicar la versión v1.1 del ***LFSR Algorithm*** a una secuencia de 1Mbits se reduce en promedio un 99,968 % el aumento de bits a la trama de salida en caso de una compresión fallida.

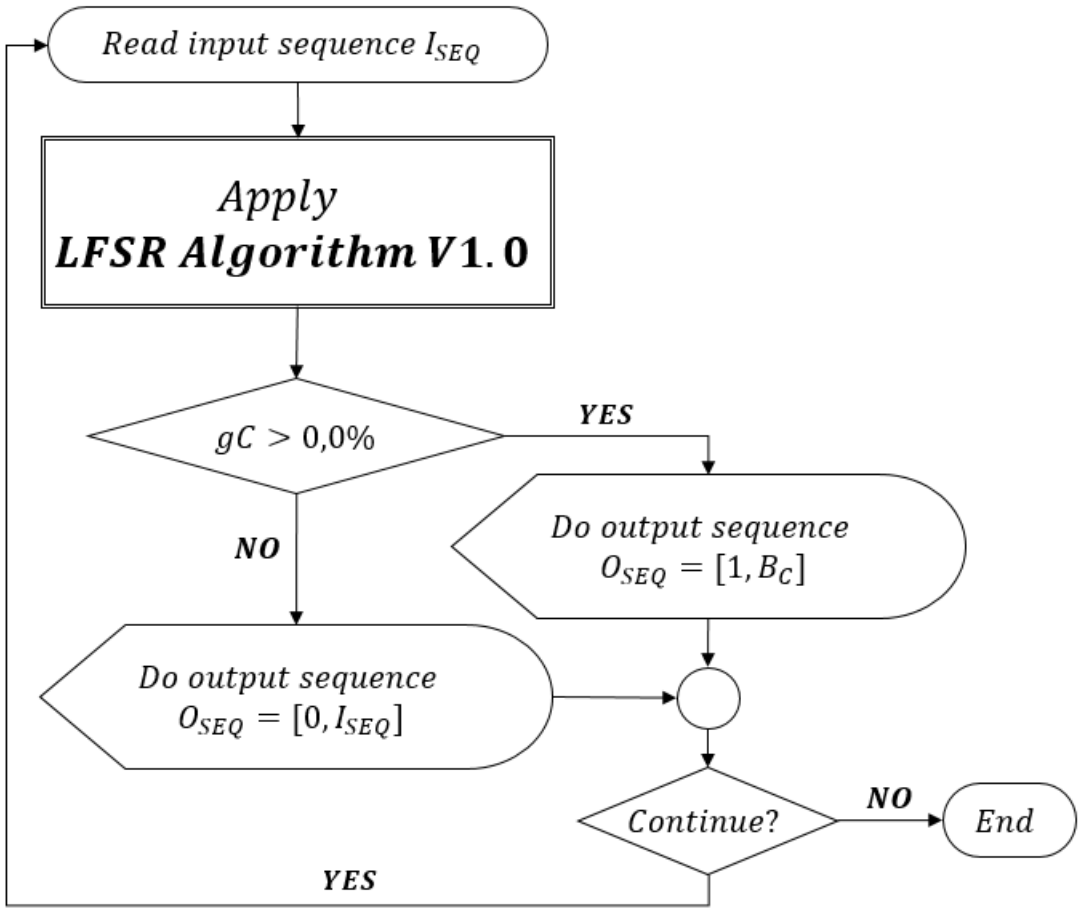


Figura 5-7.: Diagrama de flujo de algoritmo ***LFSR Algorithm v1.1*** que encapsúla al ***LFSR Algorithm v1.0*** mostrado en la figura **5-4**, con el fin de evitar la propagación de excesos de energía através de la *WSN* cuando hay fallas en la compresión.

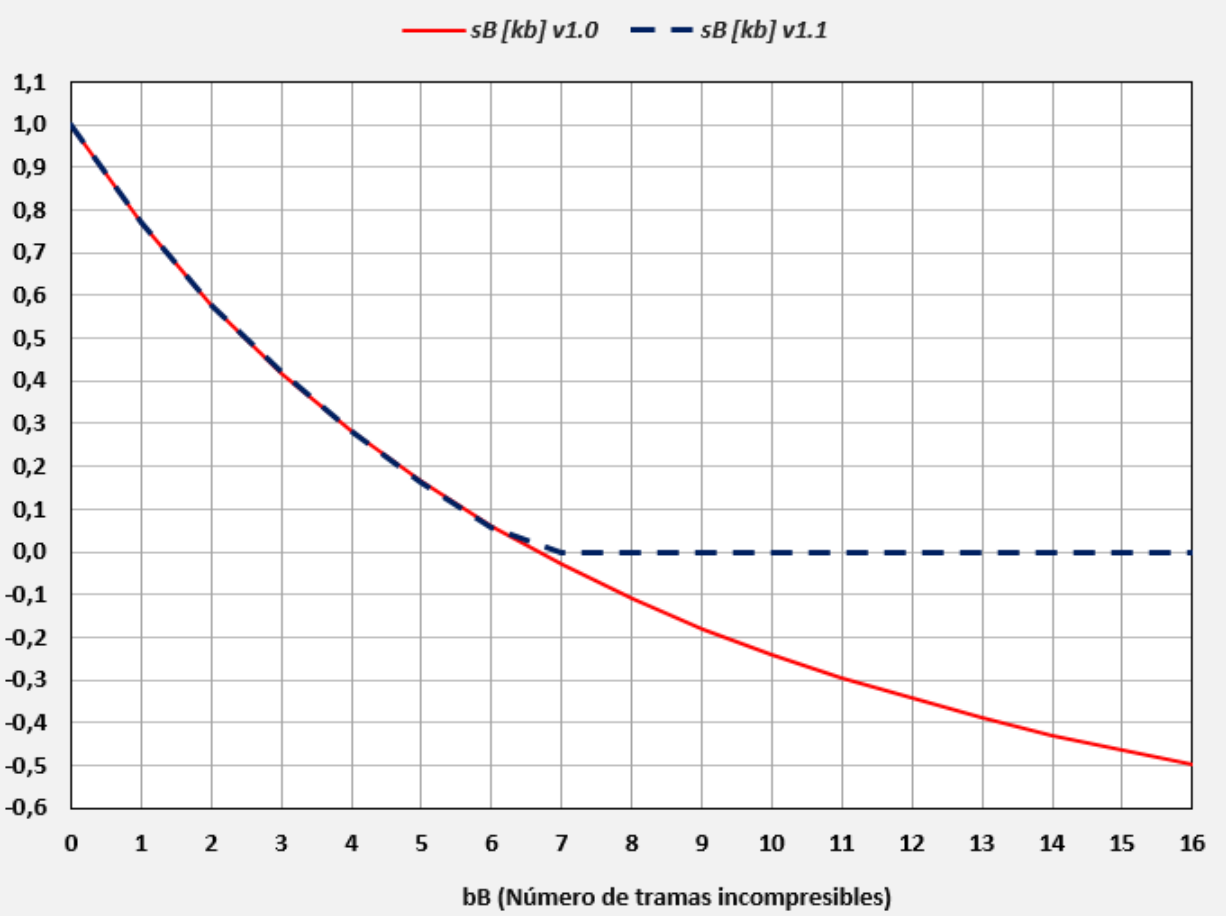


Figura 5-8.: Curvas normalizadas de la cantidad de bits comprimidos de la secuencia de entrada utilizando el ***LFSR Algorithm v1.0 y el v1.1***. Las unidades corresponden a la variable sin normalizar.

Entonces mediante el ***LFSR Algorithm v1.1*** la ecuación de energía 5-10 se transforma en la ecuación 5-11, en donde E_d es la energía utilizada para identificar el bits indicador de compresión general efectiva (es decir que indica si $gG > 0{,}0\,\%$) y se prevee que con mucha certeza $E_d \ll E_D$. Del mismo modo $E_{T(B+1)}$ significa la energía necesaria para transmitir la secuencia original adicionándole el bits indicador, también se prevé que $E_{T(B+1)} \ll E_{TB_{cf}}$.

$$E_{FC_f} = E_C + E_d + n \times (E_{T(B+1)}) \tag{5-11}$$

5.6. Algoritmo de descompresión

El algoritmo de descompresión consiste en aplicar el producto punto entre la trama comprimida y la matriz de transición.

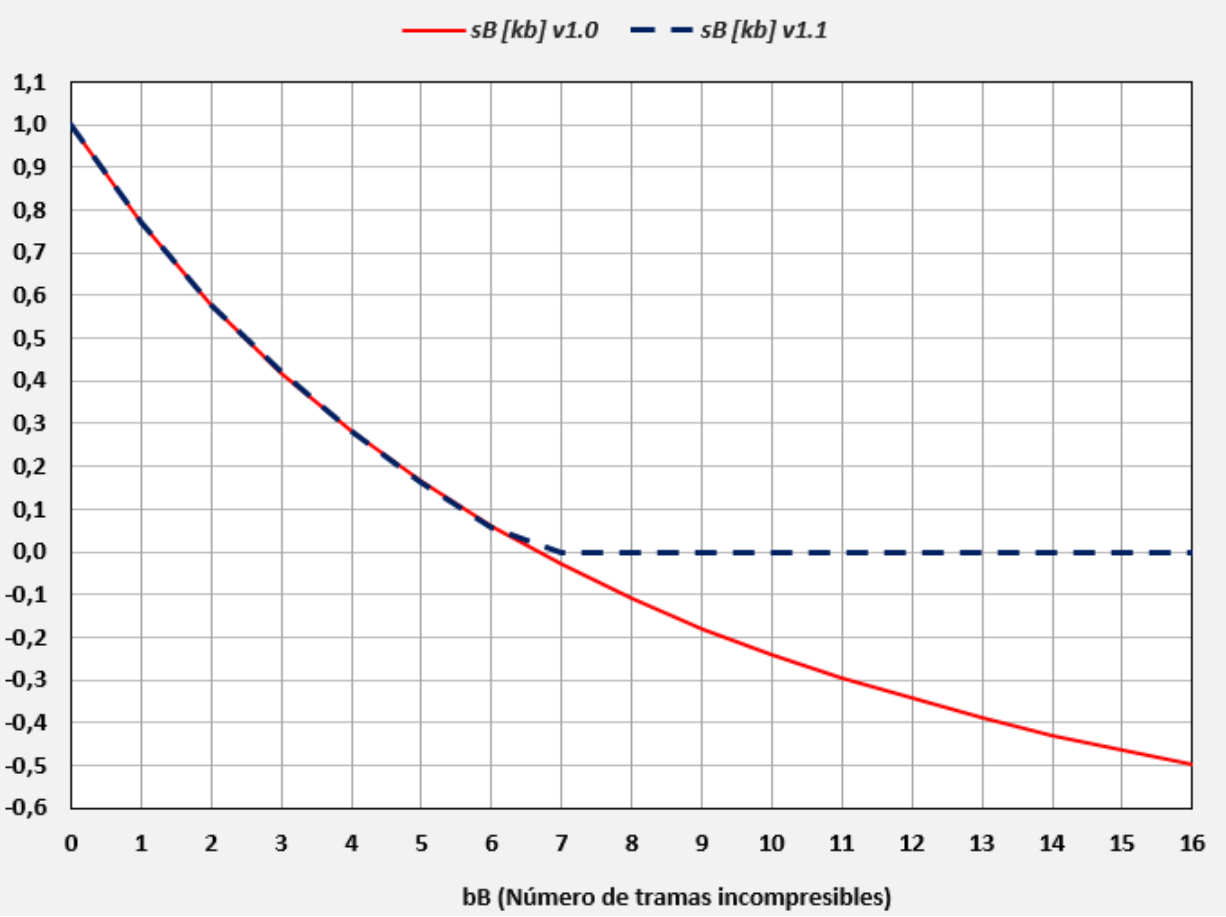


Figura 5-9.: Curvas normalizadas de la cantidad de bits comprimidos de la secuencia de entrada utilizando el ***LFSR Algorithm v1.0 y el v1.1***. Las unidades corresponden a la variable sin normalizar.

6. Conclusiones y recomendaciones

6.1. Conclusiones

- Si es posible diseñar un algoritmo de compresión de datos para redes inalámbricas de sensores utilizando la teoría del *LFSR*. Y, contario a lo encontrado en la literatura, en esta investigación se ha propuesto una forma de utilizar el LFSR dentro del algoritmo de tal forma que no se haga uso del Algoritmo de Berlekamp - Massey.

- Se encontró que la tasa de compresión del algoritmo en *kbps* depende del dispositivo en donde se esté ejecutando. Además se logró optimizar el algoritmo al punto que el tiempo de compresión por trama de 10 bits está en el orden de los μs y ms dependiendo del hardware lo utilice. Las funciones más complejas que utiliza el algoritmo son operaciones matriciales las cuales están disponibles en la gran mayoría de los dispositivos embebidos, IoT y microcontroladores. Todo significa que el algoritmo si se puede implementar en *WSN's* reales sin experimentar mayores problemas. El algoritmo final fue diseñado principalmente para redes con topología multihop. Y aplica también para redes híbridas estrella - malla si la información debe llegar a un nodo de destino atravesando un número n de nodos según las consideraciones energéticas expuestas en la sección 5.4.

- El algoritmo diseñado comprime entre un 0,5 % y un 10 % si la secuencia de entrada está conformada por subtramas de 5 bits que hagan parte de una base de datos de máximo 22 valores, de los cuales 16 sean 100 % compresibles. Además todos los valores dentro de la base de datos deben tener una distribución de probabilidad uniforme. La secuencia de entrada puede tener longitudes en kB, MB, GB, étc. Siempre y cuando se cumpla con la especificación de la base de datos descrita. Si estas condiciones no se cumplen, la version 1.1 del algoritmo es capaz de mitigar el impacto negativo que pueda tener una compresión fallida, reduciendo el incremento de la secuencia de salida en un 99,97 %.

6.2. Recomendaciones y trabajos futuros

- Se sugiere realizar un estudio del consumo de energía del hardware que ejecuta el algoritmo, utilizando para ello por lo menos tres plataformas IoT embedidas y tres

lenguajes de programación.

- Se sugiere llevar a cabo la simulación de una *WSN* y estudiar los resultados del algoritmo bajo condiciones controladas . Y el mismo estudio se debería repetir con la implementación del algoritmo en una WSN instalada fisicamente.
- Se sugiere contemplar el uso de la teoría del resembrado del LFSR (cambiar la semilla del LFSR durante la generación de la secuencia) para determinar su aplicabilidad en el diseño de un algoritmo de compresión.

A. Anexo: Bases de datos

B_{10}	B_2	Compressible (Yes/No)
1	00001	No
2	00010	No
5	00101	No
6	00110	No
9	01001	No
10	01010	No
13	01101	No
14	01110	No
16	10000	No
19	10011	No
20	10100	No
23	10111	No
24	11000	No
27	11011	No
28	11100	No
31	11111	No

Tabla A-1.: Base de datos para la construcción de las secuencias de prueba del algoritmo

B_{10}	B_2	Compressible (Yes/No)
0	00000	Yes
3	00011	Yes
4	00100	Yes
7	00111	Yes
8	01000	Yes
11	01011	Yes
12	01100	Yes
15	01111	Yes
17	10001	Yes
18	10010	Yes
21	10101	Yes
22	10110	Yes
25	11001	Yes
26	11010	Yes
29	11101	Yes
30	11110	Yes

Tabla A-2.: Base de datos para la construcción de las secuencias de prueba del algoritmo

Bibliografía

[1] Acevedo, Oscar: On the Computation of LFSR Characteristic Polynomials for One-Dimensional and Two-Dimensional Test Pattern Generation. (2014)

[2] Arici, Tarik ; Gedik, Bugra ; Altunbasak, Yucel ; Liu, Ling: PINCO: A pipelined in-network compression scheme for data collection in wireless sensor networks. En: *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on* IEEE, 2003, p. 539–544

[3] Arnold, Ross ; Bell, Tim: A corpus for the evaluation of lossless compression algorithms. En: *Data Compression Conference, 1997. DCC'97. Proceedings* IEEE, 1997, p. 201–210

[4] Baek, Seung J. ; De Veciana, Gustavo ; Su, Xun: Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation. En: *IEEE Journal on Selected Areas in Communications* 22 (2004), Nr. 6, p. 1130–1140

[5] Campbell, Philip L. ; Pierson, Lyndon G.: LFSRs Do Not Provide Compression / Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US). 1999. – Informe de Investigación

[6] Chou, Jim ; Petrovic, Dragan ; Ramachandran, Kannan: A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks. En: *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* Vol. 2 IEEE, 2003, p. 1054–1062

[7] Ganesan, Deepak ; Greenstein, Ben ; Estrin, Deborah ; Heidemann, John ; Govindan, Ramesh: Multiresolution storage and search in sensor networks. En: *ACM Transactions on Storage (TOS)* 1 (2005), Nr. 3, p. 277–315

[8] Ganesan, Deepak ; Greenstein, Ben ; Perelyubskiy, Denis ; Estrin, Deborah ; Heidemann, John: An evaluation of multi-resolution storage for sensor networks. En: *Proceedings of the 1st international conference on Embedded networked sensor systems* ACM, 2003, p. 89–102

[9] Herrero, Mariano. *Sistemas lineales con más ecuaciones que incógnitas.* goo.gl/2HS2YF. 2012

[10] Intanagonwiwat, Chalermek ; Govindan, Ramesh ; Estrin, Deborah: Directed diffusion: A scalable and robust communication paradigm for sensor networks. En: *Proceedings of the 6th annual international conference on Mobile computing and networking* ACM, 2000, p. 56–67

[11] Koutsoupia, M ; Kalligeros, Emmanouil ; Kavousianos, Xrysovalantis ; Nikolos, Dimitris: LFSR-based test-data compression with self-stoppable seeds. En: *Proceedings of the Conference on Design, Automation and Test in Europe* European Design and Automation Association, 2009, p. 1482–1487

[12] Kusuma, Julius ; Doherty, Lance ; Ramchandran, Kannan: Distributed compression for sensor networks. En: *Image Processing, 2001. Proceedings. 2001 International Conference on* Vol. 1 IEEE, 2001, p. 82–85

[13] Muthiah, R ; Neelakantan, K ; Sebastian, Somi ; TIFAC-CORE, SASTRA ; Thirumalaisamudram, Thanjavur: Lossless Compression Using LFSRs. En: *Journal of Computer Science* 3 (2007), Nr. 1, p. 25–27

[14] Pattem, Sundeep ; Krishnamachari, Bhaskar ; Govindan, Ramesh: The impact of spatial correlation on routing with compression in wireless sensor networks. En: *ACM Transactions on Sensor Networks (TOSN)* 4 (2008), Nr. 4, p. 24

[15] Petrovic, Dragan ; Shah, Rahul C. ; Ramchandran, Kannan ; Rabaey, Jan: Data funneling: Routing with aggregation and compression for wireless sensor networks. En: *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on* IEEE, 2003, p. 156–162

[16] Powell, Matt. *the Canterbury Corpus*. http://corpus.canterbury.ac.nz/descriptions/. 1997

[17] Pradhan, S S. ; Kusuma, Julius ; Ramchandran, Kannan: Distributed compression in a dense microsensor network. En: *IEEE Signal Processing Magazine* 19 (2002), Nr. 2, p. 51–60

[18] Rao, Raghuveer M.: *Wavelet transforms: Introduction to theory and applications*. Pearson Education India, 1998

[19] Sadler, Christopher M. ; Martonosi, Margaret: Data compression algorithms for energy-constrained devices in delay tolerant networks. En: *Proceedings of the 4th international conference on Embedded networked sensor systems* ACM, 2006, p. 265–278

[20] Scaglione, Anna ; Servetto, Sergio: On the interdependence of routing and data compression in multi-hop sensor networks. En: *Wireless Networks* 11 (2005), Nr. 1-2, p. 149–160

[21] Slepian, David ; Wolf, Jack: Noiseless coding of correlated information sources. En: *IEEE Transactions on information Theory* 19 (1973), Nr. 4, p. 471–480

[22] Solano, Oscar P. ; Acuña, Leonardo C. ; VillaRamírez, Jose L.: Power Monitoring Based on Industrial Internet of Things. En: *Workshop on Engineering Applications* Springer, 2017, p. 324–330

[23] Tang, Caimu ; Raghavendra, Cauligi S. ; Prasanna, Viktor K.: Power aware coding for spatio-temporally correlated wireless sensor data. En: *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on* IEEE, 2004, p. 134–143

[24] Wang, You-Chiun ; Hsieh, Yao-Yu ; Tseng, Yu-Chee: Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications. En: *IEEE Transactions on Computers* 58 (2009), Nr. 6, p. 827–838

[25] Weisstein, Eric W. *"Gauss-Jordan Elimination."From MathWorld–A Wolfram Web Resource*. http://mathworld.wolfram.com/Gauss-JordanElimination.html. 2018

[26] Weisstein, Eric W. *"Primitive Polynomial."From MathWorld–A Wolfram Web Resource*. http://mathworld.wolfram.com/PrimitivePolynomial.html. 2018

[27] Welch, Terry A.: Technique for high-performance data compression. En: *Computer* (1984), Nr. 52

[28] Wilson, Jon S.: Wireless sensor networks: Principles and applications. En: *Sensor Technology Handbook* (2004), p. 704