UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPTO. INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

"DISEÑO DE UN CODIFICADOR Y DECODIFICADOR DE UN CODIGO CICLICO PRODUCTO PARA LAS REDES DE AREAS INDUSTRIALES"

AUTORES: ALARCON BARRIENTOS, GISELA TERESA
SOTOMAYOR CHAVARRIA, LEONARDO DAVID

SEMINARIO PARA OPTAR AL TÍTULO DE
INGENIERO DE EJECUCIÓN EN ELECTRÓNICA

CONCEPCIÓN – CHILE
2016

UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPTO. INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

“DISEÑO DE UN CODIFICADOR Y DECODIFICADOR DE UN CODIGO CICLICO PRODUCTO PARA LAS REDES DE AREAS INDUSTRIALES”

AUTORES: ALARCON BARRIENTOS, GISELA TERESA
SOTOMAYOR CHAVARRIA, LEONARDO DAVID

WASHINGTON FERNANDEZ

PEDRO CRRASCO

“Diseño de un codificador y decodificador de un código cíclico producto para las redes de áreas industriales”

OBJETIVO GENERAL

Diseñar un codificador y decodificador de un código cíclico producto para las redes de áreas industriales.

Índice

RESUMEN

En el presente seminario se realiza una comparación de desempeño entre dos tipos de codificadores de diferentes características, estos codificadores están basados en la implementación de código cíclico normal y código producto cíclico, en ambas implementaciones se usa el código [7,4], el cual es puesto a prueba bajo condiciones de ruido clase A (modelo de Middleton) en canales de redes industriales simulados.

Para implementación del proceso planteado, se debe llevar a cabo ciertas condiciones, las cuales se describen a continuación:

Se debe diseñar un programa de acuerdo a las especificaciones de código cíclico normal y de código producto cíclico, para el código [7,4] utilizando decodificación por tabla de síndrome para ambos casos y considerando la utilización de un ruido clase A simulado que se agrega a la programación para la simulación de ruido de medios ambiente industrial.

De los resultados obtenidos en el proceso se puede concluir que:

El codificador y decodificador de código producto cíclico para el código [7,4] presenta un rendimiento significativamente mejor que el codificador y decodificador de código cíclico para el mismo código [7,4]

INTRODUCCIÓN

Las redes industriales se utilizan para el control, monitoreo, acciones y condiciones que se conectan a equipos físicos de alguna manera. Bajo estos parámetros se analiza y estudia las condiciones de ruido asociados a este medio industrial para la utilización de los sistemas de decodificación. Para esto se considera un código cíclico y código cíclico producto. Estos códigos son propuestos para la corrección de errores en canales, en los cuales los errores se producen de manera aleatoria o ruido impulsivo. Los códigos producto cíclico presentan un compromiso entre los códigos de corrección de errores aleatorios y errores impulsivos, lo que permite una mejora en el sistema de control de errores en el sistema de comunicación. Los códigos cíclicos son los más usados en los sistemas de control de errores construidos para los diferentes ambientes industriales.

El informe se compone de cinco capítulos, los cuales se describen a continuación:

En el capítulo I se describen de forma generalizada los medios de comunicación para el medio ambiente industrial, se detalla los tipos de medios que existen en el proceso de envío de información y como estos se pueden ver afectados por el ruido.

En el capítulo II se define la información necesaria para el desarrollo de un código cíclico normal, se describe sus propiedades para la correcta codificación y decodificación en una transmisión en presencia de ruido.

En el capítulo III se define las diferentes características y propiedades presentes en el código producto cíclico se describe sus propiedades para la correcta codificación y decodificación en una transmisión en presencia de ruido.

En el capítulo IV se describe la implementación del proceso de codificación y decodificación para los códigos cíclicos normales y códigos productos cíclicos, junto con su respectiva programación en Matlab.

En el capítulo V se especifican los resultados obtenidos en las respectivas programaciones entre el código cíclico y el código producto cíclico y como estos se diferencian, cuáles son sus ventajas y desventajas con respecto a su desempeño en base a un mismo código y se discuten estos resultados.

CAPÍTULO I

MEDIO AMBIENTE INDUSTRIAL COMO UN MEDIO DE TRANSMISIÓN DE DATOS EN PRESENCIA DE RUIDO.

1.1 MEDIOS DE TRANSMISIÓN PARA REDES INDUSTRIALES

1.1.1 CANAL DE COMUNICACIÓN.

El canal de comunicación de dato puede definirse en términos generales, como el conjunto de recursos en espectro, espacio, tiempo y equipos, necesarios para realizar una comunicación exitosa o libre de errores. Un esquema general, consiste en un transmisor, un receptor y un canal de trasmisión como se ilustra en la figura 1.1

Figura 1.1 Sistema de comunicaciones.

De acuerdo a la figura 1.1 la primera etapa presenta al transmisor, este tiene como función acondicionar las señales de información en ancho de banda y potencia para entregarlas al canal de transmisión. En la segunda etapa se muestra el canal de trasmisión, este puede ser el espacio libre, un cable, el agua u otro medio material. La mayoría de las comunicaciones eléctricas emplean como medio de transporte el aire, cables metálicos o fibras ópticas. La definición de canal de comunicaciones es muy amplia y en la práctica, con frecuencia se habla de “canal” para hacer referencia sólo a una parte de la totalidad del sistema. Como última etapa está el receptor, su función es capturar las señales en el medio de transporte, amplificarlas y acondicionarlas a fin de que resulten claros al usuario final.

1.2 RUIDO.

De acuerdo a [1], ruido puede ser cualquier señal no deseada que se presenta en un sistema de comunicación. El fenómeno de ruido es habitual e inevitable. Es decir, que el ruido está presente en todo sistema de comunicación y favorece, en mayor o menor medida, al deterioro de la señal transmitida a la entrada del receptor dificultando su detección, lo que degrada o limita el desempeño del sistema de comunicación. El uso de la expresión "señal" es discutible, esto debido a que la señal puede ser deseada o no deseada.

1.2.1 RUIDO NATURAL.

El ruido natural puede clasificarse en dos grandes grupos: el que se produce por los propios componentes electrónicos de un circuito o sistema y el que se produce por fuentes externas a él.

1.2.1.1 RUIDO TÉRMICO.

Se debe a la agitación térmica de los electrones asociados a los átomos del material del dispositivo o a la línea de transmisión. También llamado ruido de Johnson o de Nyquist presenta una componente de frecuencia aleatoria en todo lo ancho del espectro de frecuencias, cuya amplitud varía continuamente.

1.2.1.2 RUIDO DE DISPARO

El ruido de disparo (shot noise) se produce por las variaciones aleatorias en los tiempos de llegada de los portadores de carga (electrones o huecos) a los electrodos de salida en todos los dispositivos activos y aparece como una corriente variable de ruido, superpuesta a la corriente de señal de salida.

1.2.1.3 RUIDO DE PARTICIÓN.

Ocurre cuando los electrones de un haz inciden sobre dos o más electrodos, de modo que hay fluctuaciones aleatorias en el número de electrones que llegan a cada electrodo. Este tipo de ruido está presente por ejemplo en los transistores.

1.2.1.4 RUIDO FLICKER.

Este término se emplea para describir la fluctuación a baja frecuencia de los valores de resistencia que se traducen en tensión de ruido cuando por ellos circula una corriente continua. El espectro del ruido flicker se caracteriza por una disminución de densidad a medida que sube la frecuencia hasta llegar a ser despreciable respecto del ruido blanco. A tal tipo de ruido se les designa a veces como ruido de corriente, ruido en exceso o ruido de contacto

1.2.1.5 RUIDO ATMOSFÉRICO.

La atmósfera afecta a un receptor de dos formas: atenúa el ruido procedente del cosmos y, por otra parte, genera ruido propio. Las descargas eléctricas atmosféricas durante las tormentas producen ráfagas de ruido impulsivo, cuyas componentes en las bandas de frecuencias medias y altas se propagan a grandes distancias gracias a las formas de propagación ionosférica. De manera semejante a las ondas en esas bandas, este ruido depende del clima, hora del día, estación del año y ubicación del receptor con relación a las zonas de ocurrencia de tormentas.

1.2.1.6 RUIDO CÓSMICO.

El ruido cósmico se genera en el espacio exterior, fuera de la atmósfera terrestre. Las principales fuentes son el sol, la vía láctea y otras fuentes cósmicas discretas, que se definen como radioestrellas, entre las que se incluye una fuente particularmente intensa en la constelación de Casiopea.

1.2.1.7 RUIDO EN EL PLANO GALÁCTICO.

Es el ruido procedente del plano galáctico en dirección del centro de la galaxia (Vía Láctea) y es el de mayor nivel. El ruido procedente de otras zonas de la galaxia puede llegar a ser de 12 a 15 dB inferior al del plano galáctico.

1.2.1.8 RUIDO SOLAR.

En los sistemas de comunicación vía satélite, el sol constituye una fuente de ruido blanco muy importante, que puede causar severos problemas de interferencia, y aún, bloqueo total de las comunicaciones cuando hay alineamiento entre éste y la estación receptora terrestre.

1.2.2 RUIDO ARTIFICIAL.

El ruido artificial producido por la actividad humana se debe a la recepción de señales indeseables provenientes de otras fuentes tales como contactos defectuosos, radiación por ignición y alumbrado fluorescente. A menudo es el que predomina en algunas partes del espectro radioeléctrico y su intensidad puede cambiar al aumentar la densidad de utilización de dispositivos eléctricos y electrónicos y con la introducción de nuevos tipos de dispositivos.

1.2.2.1 RUIDO POR INTERFERENCIA.

Incluye la interferencia de un canal radioeléctrico sobre otro, como resultado de la interferencia que producen los transistores, variaciones en la frecuencia de la portadora en el transmisor, efectos debidos a dispersión troposférica o reflexión ionosférica en transmisiones de larga distancia, modulación cruzada entre canales en radioenlaces e interferencia debido a propagación multicamino. Estos tipos de ruidos son detectados por el receptor generando esta interferencia en la señal.

1.2.2.2 RUIDO POR LÍNEAS ELETRICAS DE ALTA POTENCIA.

Es un ruido periódico originado por las líneas de suministro eléctrico que transportan corriente alterna.

1.2.2.3 RUIDO IMPULSIVO.

Este término se emplea para designar una variedad de fenómenos, no todos de origen humano y puede modelarse como la superposición de un número reducido de impulsos de gran amplitud y corta duración que pueden ocurrir con cierta periodicidad, como el ruido por efecto corona en líneas de alta tensión, o bien producirse de forma aleatoria como el ruido producido por los equipos de conmutación telefónica. El ruido impulsivo tiende a tener una distribución no Gaussiana y suele ser no estacionario, por lo que resulta difícil su análisis matemático

1.3 RUIDO EN AMBIENTE INDUSTRIAL.

El ruido y las señales no deseadas se acoplan a todo equipo electrónico en una cadena de producción industrial, lo que provoca errores en el proceso de medida. La principal fuente de ruido en el ambiente industrial es la red eléctrica, ya que siempre está presente y por ella fluyen altos niveles de intensidades. Entre otras fuentes de ruido se pueden mencionar:

- Las variaciones de la temperatura en los sistemas electrónicos tienen una gran influencia sobre todos los dispositivos semiconductores.
- Los golpes y vibraciones mecánicas generan fallos e interrupciones en las conexiones y soldaduras deficientes.
- Los motores generan una señal interrumpida bruscamente de alta potencia y con un espectro frecuencial muy amplio.
- Los sistemas digitales se alimentan mediante intensidades que cambian de forma impulsiva durante los cambios de estado, a la frecuencia del reloj.
- Los conmutadores de potencia generan impulsos de gran amplitud que son fuente de intensas interferencias y los conmutadores electrónicos basados en tiristores, y dispositivos electrónicos de conmutación, que se utilizan en el control de motores y fuentes de potencia, son generadores de ruidos de amplio espectro, como consecuencia de la rapidez de sus cambios y del nivel de las intensidades que conmutan.
- Los interruptores mecánicos que operan a gran velocidad, y que generan un ruido con espectro entre 1 y 10 KHz.
- Las lámparas de descarga, como los tubos fluorescentes o de neón, generan un ruido de interferencia con espectro relevante por encima de 1 MHz.
- Los equipos que operan con espectros frecuenciales muy estrechos, como los generadores de RF magnetrones, equipos de soldadura y cualquier tipo de transmisor.

1.4 MODELO DE RUIDO MIDDLETON.

D. Middleton [2] propone en 1977 un modelo de ruido impulsivo, se compone por la suma de ruido blanco y ruido impulsivo. En este modelo, según el ancho de banda del ruido, se clasifica en tres clases generales: A, B y C.

-Ruido de Clase A: Interferencia cuyos espectros son iguales o más estrechos que el ancho de banda del receptor.

-Ruido de Clase B: Ruido cuyos espectros son más amplios que el ancho de banda del receptor.

-Ruido de Clase C: Interferencia que incluye las características de clase A y B.

De acuerdo a Middleton, el ruido presente en las líneas eléctricas es del tipo clase A.

La varianza σ^2 del ruido se representa por la suma de las varianzas de la componente Gaussiana y la componente impulsiva. Es decir:

$$\sigma^2 = \sigma_G^2 + \sigma_i^2 \tag{1.1}$$

Donde:

σ^2: Varianza del ruido de línea eléctrica de baja tensión.

σ_G^2: Varianza de la componente Gaussiana del ruido.

σ_i^2: Varianza de la componente impulsiva del ruido.

La función de densidad de probabilidad (FDP), del ruido de Middleton clase A, se representa por:

$$P_z(z) = \sum_{m=0}^{\infty} \frac{e^{-A} * A^m}{m!} * \frac{1}{\sqrt{2 * \pi * \sigma_m^2}} * e^{-\frac{|Z|^2}{2 * \sigma_m^2}} \tag{1.2}$$

Donde:

Z: Envolvente del ruido.

σ_m^2: Varianza de Middleton.

m: Número de subcanales de Middleton

A: Índice impulsivo

El parámetro A se define por el producto entre la duración promedio de los impulsos y la razón de generación media del ruido impulsivo (cantidad de eventos impulsivos por unidad de tiempo).

La varianza de Middleton se obtiene a partir de:

$$\sigma_m^2 = \sigma^2 \frac{\frac{m}{A+\Gamma}}{1+\Gamma} \tag{1.3}$$

Donde:

Γ: Relación entre las potencias Gaussiana e impulsiva, y se define por:

$$\Gamma = \frac{\sigma_G^2}{\sigma_i^2} \tag{1.4}$$

Esto significa que el modelo de Middleton se representa por el producto de una distribución Poisson, $P\ (m;\ A)$ y una distribución de Gauss $G(z, \sigma_m^2)$. Esto es:

$$P_z(z) = \sum_{m=0}^{\infty} P(m; A) G(z, \sigma_m^2) \tag{1.5}$$

Donde:

$$P(m; A) = \frac{e^{-A} * A^m}{m!} \tag{1.6}$$

y

$$G(z; \sigma_m^2) = \frac{1}{\sqrt{2 * \pi * \sigma_m^2}} * e^{-\frac{|Z|^2}{2 * \sigma_m^2}} \tag{1.7}$$

El modelo de Middleton clase A consiste de la sumatoria de infinitos ruidos Gaussianos, cada uno con distinta potencia de ruido. Aquellos que presentan mayor potencia se denominan impulsivos y los que tienen una potencia de ruido mucho menor que los impulsivos (al menos 10 veces) se definen como Gaussianos. La distribución de Poisson representa la probabilidad de pasar de un ruido más impulsivo a uno menos impulsivo y viceversa.

La potencia de ruido Gaussiano y su varianza se relacionan por:

$$N_G = 2\ \sigma_G^2 \tag{1.8}$$

Como el modelo de Middleton se compone de ruido gaussiano y ruido impulsivo, ambos con una distribución de Gauss, se cumple que:

$$N_0 = N_g + N_i = 2\ \sigma_G^2 + 2\sigma_i^2 = 2\sigma^2 \tag{1.9}$$

Donde:

N: Densidad de potencia de ruido de Middleton

N_g: Densidad de potencia de ruido Gaussiano

N_i: Densidad de potencia de ruido impulsivo

El modelo de ruido de Middleton se puede aplicar al ruido presente en el medio ambiente industrial y se utiliza en las simulaciones del ruido presente en el medio ambiente industrial.

1.5 RAZÓN SEÑAL A RUIDO.

Se define como relación entre la señal y el ruido (SNR o S/N), a la proporción existente entre la potencia de la señal que se transmite y la potencia del ruido, esta se expresa como:

$$SNR = \frac{Potencia\ de\ la\ señal\ en\ el\ receptor\ (S)}{Potencia\ de\ ruido\ (N)} \tag{1.10}$$

o expresada en dB:

$$SNR_{dB} = 10\ log_{10}\frac{S}{N} \tag{1.11}$$

La potencia de la señal desempeña un papel fundamental en la transmisión de información. Está relacionada con la calidad de la transmisión. Al incrementarse la potencia de la señal, se reduce el efecto del ruido de canal, y la información se recibe con mayor exactitud, o con menos errores. Una mayor relación de señal a ruido S/N permite también la transmisión a través de una distancia mayor. En cualquier caso, una cierta S/N mínima es necesaria para la comunicación.. En el diseño de este sistema, se desea que la relación señal a ruido tenga una baja tasa de error de bits con un muy pequeño SNR para un buen desempeño del sistema. A grandes rasgos esta medición expresa la razón en decibeles del aporte para el cual el nivel de la señal excede al ruido.

1.6 RAZÓN DE BIT ERRÓNEO.

El BER es un parámetro que cuantifica la fidelidad completa del sistema de comunicación bits que se transmiten y bits que se reciben, el concepto de BER se define como:

$$BER = \frac{Número\ de\ bits\ erroneos\ recibidos}{Número\ de\ bits\ transmitidos} \tag{1.12}$$

BER es la probabilidad estimada de que un bit transmitido por un dispositivo sea recibido de forma correcta o incorrecta. Por ejemplo si un 1 es transmitido y posteriormente se recibe un 0, entonces el resultado es un bit erróneo. Por lo tanto, es una medida en la cual se compara el número de bits incorrectos con el número total de bits transmitidos. Cuanto menor sea el valor de BER, mejor es el desempeño del sistema.

En la figura 1.2 se muestra una curva de la relación entre la probabilidad de error (P_B) y SNR ($^{E_b}/_{N_0}$).

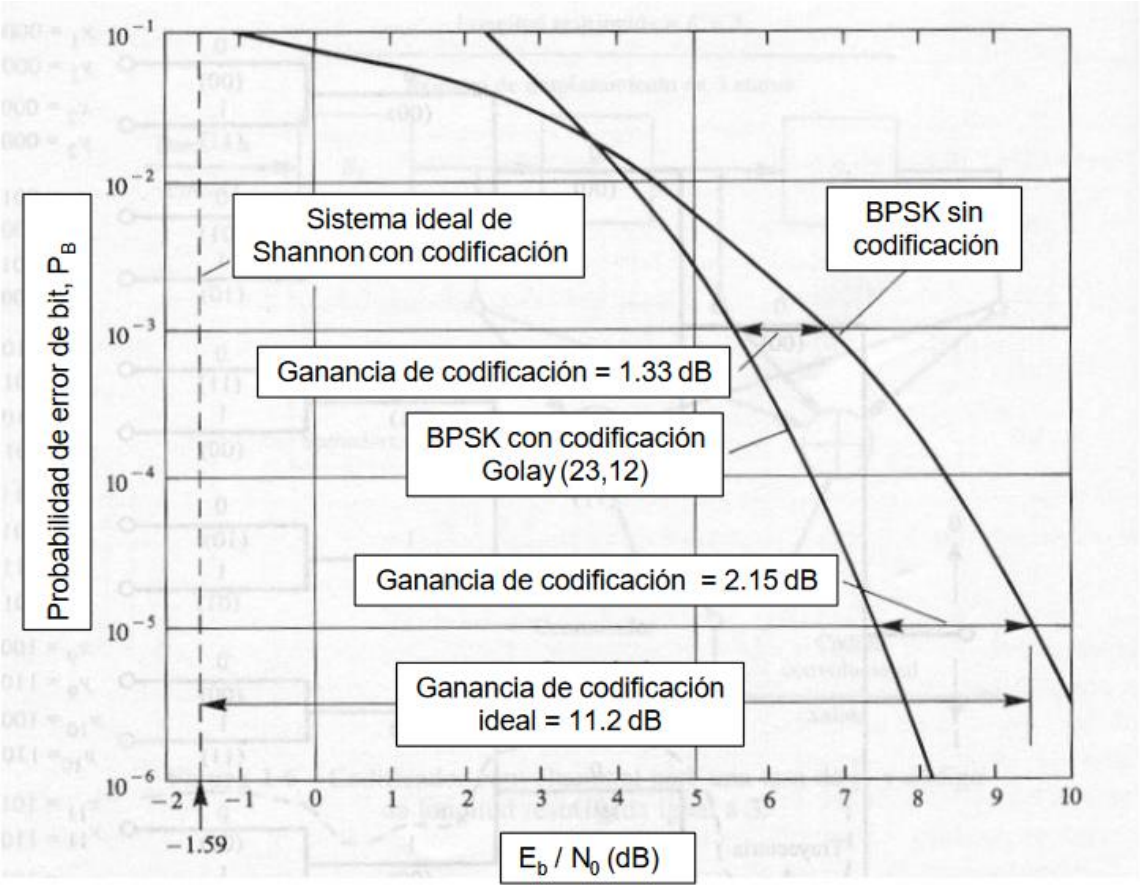

Figura 1.2 Gráfica ejemplo BER/SNR.

1.7 CAPACIDAD DEL CANAL.

De acuerdo a [3] se define la capacidad de un canal a la velocidad, en bps (bits por segundo), a la que se pueden transmitir los datos en un canal o ruta de comunicación en presencia de ruido.

El ancho de banda de la señal transmitida se limita por la naturaleza del medio de transmisión (en hertzios). Para un ancho de banda determinado es aconsejable la mayor velocidad de transmisión posible pero de forma que no se supere la tasa de errores permitidos.

Para un ancho de banda dado W, la mayor velocidad de transmisión posible es 2W, pero si se permite (con señales digitales) codificar más de un bit en cada ciclo, es posible transmitir más cantidad de información.

La formulación de Nyquist dice que aumentado los niveles de voltaje diferenciables en la señal, es posible incrementar la cantidad de información transmitida. La cual se expresa de la siguiente manera:

$$C = 2wlog_2M \tag{1.13}$$

Donde:

C= capacidad del canal

M= niveles de la señal

Para el caso binario, M=2 entonces $log_2(2) = 1$, por lo tanto:

$$C = 2w \tag{1.14}$$

El problema de esta técnica es que el receptor debe de ser capaz de diferenciar más niveles de voltaje en la señal recibida, cosa que se dificulta por el ruido.

Shannon propuso la expresión que relaciona la potencia de la señal, la potencia del ruido, la capacidad del canal y el ancho de banda.

$$C = Wlog_2(1 + \frac{S}{N}) \tag{1.15}$$

Donde:

C: Capacidad teórica máxima en bps.

W: Ancho de banda del canal Hz.

S/N: Relación de señal a ruido, S y N dados en watts.

CAPITULO II

CÓDIGOS CÍCLICOS

2.1 INTRODUCCIÓN

Los códigos cíclicos son un subconjunto de los códigos lineales. Un código se denomina cíclico si existe un desplazamiento cíclico de derecha o izquierda se obtiene otro elemento del conjunto de código. La estructura matemática permite el diseño de un código que presenta un alto grado de corrección de error. Estos poseen dos características muy importantes, son fáciles de implementar usando una circuitería basada en registro de desplazamiento y presentan una estructura algebraica bien definida.

2.1.1 DESCRIPCIÓN DE LOS CÓDIGOS CÍCLICOS.

2.1.1.1 ROTACIÓN CÍCLICA.

Sea la n-tupla:

$$v = (v_0, v_1, \dots, v_{n-1}) \tag{2.1}$$

Si los componentes de n-tupla se desplazan cíclicamente un lugar a la derecha se obtiene la n-tupla:

$$v(1) = (v_{n-1}, v_0, \dots, v_{n-2}) \tag{2.2}$$

Denominada rotación cíclica de un elemento de v.

De una forma más general si los componentes de v se rotan cíclicamente i veces se obtiene:

$$v(i) = (v_{n-i}, v_{n-i+1}, \dots, v_{n-1}, v_0, v_1, \dots, v_{n-i-1}) \tag{2.3}$$

El código $C(n,k)$ es cíclico si y solo si cualquier rotación cíclica de un vector v perteneciente a C es también un vector del código C.

2.1.1.2 NOTACIÓN POLINOMIAL DE UN CODIGO CICLICO.

Para representar los vectores pertenecientes a los códigos cíclicos se usa la notación polinomial. Cada una de las componentes de un vector código:

$$v = (v_0, v_1, \dots, v_{n-1}) \tag{2.4}$$

Son los coeficientes del polinomio:

$$v(x) = (v_0 + v_1 x + v_2 x^2 \dots + v_{n-1} x^{n-1}) \tag{2.5}$$

A este polinomio se le denomina polinomio código. Por lo tanto a todo vector v le corresponde un polinomio código $v(x)$ de grado $(n-1)$ o menor.

$v^{(i)}$, en representación polinomial, es:

$$v^{(i)}(x) = (v_{n-i}, v_{n-i+1}x, \dots, v_{n-1}x^{i-1}, v_0 x^i, v_1 x^{i+1}, \dots, v_{n-i-1}x^{n-1}) \tag{2.6}$$

Se considera el vector código:

$$v = (v_0 + v_1, \dots, v_{n-1}) \tag{2.7}$$

y su polinomio asociado:

$$v(x) = v_0 + v_1 x + v_2 x^2 + \cdots + v_{n-1} x^{n-1} \tag{2.8}$$

Si se multiplica el polinomio por x, se tiene:

$$xv(x) = v_0 x + v_1 x^2 + v_2 x^3 + \cdots + v_{n-1} x^n \tag{2.9}$$

este polinomio módulo $(x^n - 1)$ de (2.9) da:

$$xv(x) = v_0 x + v_1 x^2 + v_2 x^3 + \cdots + v_{n-2} x^{n-1} + v_{n-1} \tag{2.10}$$

y en forma de vector:

$$(v_1, \dots, v_{n-1}, v_0) \tag{2.11}$$

Si se multiplica $x^{(i)}v(x)$, se tiene:

$$x^{(i)}v(x) = v_0 x^i + v_1 x^{i+1} + \cdots + v_{n-1} x^{n+i-1} \tag{2.12}$$

este proceso se puede expresar como:

$$(v_{n-1}, v_0, v_1, \dots, v_{n-2}) \tag{2.13}$$

que es una rotación cíclica del vector anterior.

De aquí se observa que $v^{(i)}(x)$ es $x^i v(x)$ módulo $(x^n - 1)$.

Así, se puede definir como código cíclico a un código C (n, k) en el que si se multiplica por x módulo $(x^n - 1)$ cualquier polinomio del código, se obtiene otro polinomio del código.

2.1.1.3 POLINOMIO DE GRADO MÍNIMO.

Proposición 1: C es un código cíclico si y solo si C es un ideal del anillo $\frac{F[x]}{x^n-1}$. Se llama $C(x)$ a este ideal.

Teorema 1: Polinomio de grado mínimo.

Sea $C(n, k)$ un código cíclico de longitud n y dimensión k y sea $g(x)$ un polinomio mónico[1] de $C(x)$ de grado r el menor posible. Se cumple que:

- $g(x)$ es el único polinomio mónico de grado r de $C(x)$ y genera el ideal principal.
- $g(x)$ divide $(x^n - 1)$.
- El conjunto de polinomios $g(x)$, $xg(x)$, $x^2 g(x)$,…,$x^{n-r-1} g(x)$ de grado r, $r + 1$, $r + 2$,…, $n - 1$ genera $C(x)$ como un subespacio vectorial. La dimensión del código C es $k = n - r$.

Por ser $C(x)$ un ideal principal generado por $g(x)$, se tiene:

Teorema 2:

Sea

$$g(x) = 1 + g_1 x + g_2 x^2 + \cdots + g_{r-1} x^{r-1} + x^r \tag{2.14}$$

El polinomio de grado mínimo, para todo $v(x)$ de grado menor o igual que $n - 1$, $v(x)$ pertenece al código si y solo si $v(x)$ es múltiplo de $g(x)$.

(1) Un polinomio mónico se define como aquel polinomio que sus coeficientes principales presentan valor unitario.

Teorema 3:

Si $g(x)$ tiene grado $(n-k)$ y es factor de $(x^n-1)\, entonces\, g(x)$ genera siempre un código cíclico $C(n,k)$.

2.1.2 MATRIZ GENERADORA.

Sea $C(n,k)$ un código cíclico y $g(x) = g_0 + g_1x + \cdots + g_{n-k}x^{n-k}$. Se sabe que el conjunto de k polinomios $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ forman una base del código $C(x)$. Si se distribuye en filas de una matriz $k * n$, las k n-tuplas correspondientes a esos k polinomios código se obtiene la matriz generadora.

Cabe mencionar que "$\oplus$" es el símbolo que indica la suma en $módulo\ 2$.

Para obtener la matriz G a partir de polinomio generador se procede de la siguiente manera:

- Se escribe en el renglón k-ésimo los coeficientes de $g(x)$, cualquiera que cumpla con los requisitos del código.

- A la fila k-1, corresponden los coeficientes de $g_2(x)$ donde $g_2 = x * g(x)$ si el coeficiente de x^{n-k-1} de $g(x)$ es cero, y $g_2(x) = x * g(x) \oplus g(x)$ si el coeficiente de x^{n-k-1} de $g(x)$ es uno.

- A la fila k-2, corresponden los coeficientes de $g_3(x)$ donde $g_3(x) = x * g_2(x)$ si el coeficiente de x^{n-k-1} de $g(x)$ es cero, y $g_3(x) = x * g_2(x) \oplus g(x)$ si el coeficiente de x^{n-k-1} de $g_2(x)$ es uno.

- Así sucesivamente. Por lo que la matriz generadora queda dada por (2.15)

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & . & . & . & . & . & g_{n-k} & 0 & 0 & 0 & . & . & 0 \\ 0 & g_0 & g_1 & g_2 & . & . & . & . & . & g_{n-k} & 0 & 0 & . & . & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & . & . & . & . & . & g_{n-k} & 0 & . & . & 0 \\ . & & & & & & & & & & & & & & . \\ . & & & & & & & & & & & & & & . \\ . & & & & & & & & & & & & & & . \\ 0 & 0 & . & . & . & 0 & g_0 & g_1 & g_2 & . & . & . & . & . & g_{n-k} \end{bmatrix} \qquad (2.15)$$

Ejemplo Se determina la matriz generadora de código cíclico

Para el cálculo de la matriz se utiliza el polinomio característico del código cíclico [7,4]. Si se presenta el factor $x^{n-k-1} = x^2$ el polinomio se multiplica por x y se suma en $mod\ 2$ con el polinomio característico $g(x)$, en el caso que no esté presente el factor, el polinomio se multiplica por x . Este proceso se repite k veces, donde k es la cantidad de bits de mensaje del código, en este caso $k = 4$.

A partir de lo anterior se inicia el proceso del cálculo de G. Se tiene:

$$g(x) = 1 + x + x^3 \tag{2.16}$$

Como $g(x)$ no presenta x^2, $g_2(x)$ se determina por:

$$g_2(x) = g(x) * x \tag{2.17}$$

$$g_2(x) = (1 + x + x^3) * x = x^4 + x^2 + x \tag{2.18}$$

Como $g_2(x)$ presenta x^2, $g_3(x)$ se determina por:

$$g_3(x) = g_2(x) * x \ \oplus g(x) \tag{2.19}$$

$$g_3(x) = (x^5 + x^3 + x^2) \ \oplus (1 + x + x^3) \tag{2.20}$$

$$g_3(x) = x^5 + x^2 + x + 1 \tag{2.21}$$

Como $g_3(x)$ presenta x^2, $g_4(x)$ se determina por:

$$g_4(x) = g_3(x) * x \ \oplus g(x) \tag{2.22}$$

$$g_4(x) = (x^6 + x^3 + x^2 + x) \oplus (1 + x + x^3) \tag{2.23}$$

$$g_4(x) = x^6 + x^2 + 1 \tag{2.24}$$

Los polinomios $g_1,\ g_2,\ g_3\ y\ g_4$ se cambian a su correspondiente vector y se ordenan en el orden inverso en los que se calculan creando la matriz G, la que queda de la siguiente manera:

$$G = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] \tag{2.25}$$

$$\overbrace{\text{Identidad}} \qquad \overbrace{\text{Paridad}}$$

Como se puede apreciar G se compone por la matriz identidad y la matriz de paridad, por lo que se encuentra en su forma estándar.

2.1.3 MATRIZ CHEQUEO DE PARIDAD.

De acuerdo a [5], como $g(x)$ es factor de $(x^n - 1)\ entonces\ (x^n - 1) = g(x)h(x)$, siendo $h(x) = h_o + h_1x + \cdots + h_kx^k$. Sea una palabra código $v = (v_0, v_1, \ldots, v_{n-1})$, por lo tanto $v(x)$, polinomio asociado a v, debe ser múltiplo del polinomio generador $g(x)$:

$$v(x) = a(x)g(x)\ v(x)h(x) = a(x)g(x)h(x) = a(x)(x^n - 1) = a(x)x^n - a(x) \tag{2.26}$$

El grado de $a(x)$ es menor o igual que $(k - 1)$, las potencias $x^k, x^{k+1}, \ldots, x^{n-1}$ no aparecen en $x^n a(x) - a(x)$. Si se expande el producto $v(x)h(x)$, los coeficientes de $x^{k+1}, \ldots, x^{n-1}$ deben ser iguales a cero. Por lo tanto, se obtiene $(n - k)$ ecuaciones.

El polinomio recíproco de $h(x)$ es:

$$x^k h(x^{-1}) = h_k + h_{k-1}x + h_{k-2}x^2 + \cdots + h_0x^k. \tag{2.27}$$

donde $x_k h(x^{-1})$ es también factor de $(x^n - 1)$, por lo tanto, genera un código cíclico $C(n, n-k)$.

La matriz generadora de ese código es:

$$H = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & . & . & . & . & . & h_0 & 0 & 0 & 0 & . & . & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & . & . & . & . & . & h_0 & 0 & 0 & . & . & 0 \\ 0 & 0 & h_k & h_{k-1} & h_{k-2} & . & . & . & . & . & h_0 & 0 & . & . & 0 \\ . & & & & & & & & & & & & & & . \\ . & & & & & & & & & & & & & & . \\ . & & & & & & & & & & & & & & . \\ 0 & 0 & . & . & . & 0 & h_k & h_{k-1} & h_{k-2} & . & . & . & . & . & h_0 \end{bmatrix} \quad (2.28)$$

Teorema 3:

Sea $C(n,k)$ un codigo cíclico con polinomio generador $g(x)$, el código dual de C es cíclico y se genera por $x^k h(x-1)$, con $h(x) = \frac{(x^n+1)}{g(x)}$.

Ejemplo matriz H

Si un código con una matriz de generación G en forma estándar, $G = (I_k | A)$, entonces $H = (A^t | I_{n-k})$ es una matriz de comprobación. El código que se genera a partir de H se llama código dual.

A continuación se muestra paso a paso el cálculo de la matriz H. Se tiene G

$$G = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right] \quad (2.29)$$

$$\left[\begin{array}{c|c} \mathbf{I^k} & \mathbf{A} \end{array}\right]$$

Se toma la matriz A, por lo que se tiene:

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \tag{2.30}$$

La matriz H_1se traspone, lo que da:

$$\mathbf{H}_1^{'} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{2.31}$$

A la matriz H_1' se le agrega la matriz identidad, con lo que se obtiene la matriz H:

$$\mathrm{H} = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{2.32}$$

$$\left[\mathbf{A}^{\mathrm{t}} \mid \mathbf{I}_{\mathrm{n-k}} \right]$$

Al igual que la matriz G, la matriz H se encuentra en su forma estándar.

2.1.4 CODIFICACIÓN SISTEMÁTICA.

De acuerdo a [4] la palabra de un código sistemático se forma por los bits de la palabra mensaje y una serie de bits de paridad. Para mejorar la eficacia de este tipo de código, se necesita un algoritmo para conseguir una implementación sistemática del código.

Sea:

$$u = (u_0, \dots, u_{k-1}) \tag{2.33}$$

$$u(x) = u_0 + \cdots + u_{k-1}x^{k-1} \tag{2.34}$$

el polinomio a codificar.

Pasos para la codificación sistemática de u:

- Multiplicar x^{n-k} por $u(x)$
- Calcular $b(x)$, que es el resto de dividir $x^{n-k}u(x)$ entre $g(x)$.
- Descomposición de $b(x)$ y $u(x)$.

Sea

$$u = (u_0, \dots, u_{k-1}) \tag{2.35}$$

$$u(x) = u_0 + \cdots + u_{k-1}x^{k-1} \tag{2.36}$$

$$x^{n-k}u(x) = u_0x^{n-k} + \cdots + u_{k-1}x^{n-1} \tag{2.37}$$

$$x^{n-k}u(x) = a(x)g(x) + b(x) \tag{2.38}$$

donde $b(x)$ es un polinomio de grado menor o igual que $(n-k-1)$.

$$b(x) = b_0 + \cdots + b_{n-k-1}x^{n-k-1} \tag{2.39}$$

$$b(x) + x^{n-k}u(x) = a(x)g(x) \tag{2.40}$$

pertenece a $C\ (n,k)$

Por lo tanto:

$$(b_o + b_1 x + \cdots + b_{n-k-1}x^{n-k-1} + u_0 x^{n-k} + \cdots + u_{k-1}x^{n-1}) \tag{2.41}$$

pertenece al código $C(n,k)$.

Por lo tanto se tiene:

$$\underbrace{\boxed{b_0,\ldots,b_{n-k-1}}}_{\text{REDUNDANCIA}}\underbrace{\boxed{u_0,\ldots,u_{k-1}}}_{\text{PALABRA FUENTE}} \tag{2.42}$$

Ejemplo codificación

A continuación se detalla el proceso de codificación del código [7,4] para código cíclico. En primer lugar se hace uso de la matriz generadora G, esta matriz está definida como

$$G=\left[\begin{array}{cccc:ccc} 1&0&0&0&1&0&1\\ 0&1&0&0&1&1&1\\ 0&0&1&0&1&1&0\\ 0&0&0&1&0&1&1 \end{array}\right]$$

$$\left[\begin{array}{cc:c} \text{I} & & \text{A} \end{array}\right] \tag{2.43}$$

Donde

I = Matriz identidad.

A= Matriz del polinomio característico.

Una vez dispuesto de esta matriz, se inicia el proceso de codificación para esto se emplea el método de enviar bloques de mensaje, se trabaja estos mensajes como una matriz de datos, un ejemplo de un bloque de mensaje se muestra a continuación

$$M=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.44}$$

Donde

Mensaje (1)= [0 0 0 1]

Mensaje (2)= [0 1 0 1]

Mensaje (3)= [1 1 0 0]

Mensaje (4)= [0 0 1 0]

La codificación del código cíclico [7,4] establece la multiplicación de la matriz mensaje y su respectiva matriz generadora G. De esto se tiene:

$$C_1 = M \text{ x } G \tag{2.45}$$

$$C_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{(4,4)} x \begin{bmatrix} 1\,0\,0\,0 \vdots 1\,0\,1 \\ 0\,1\,0\,0 \vdots 1\,1\,1 \\ 0\,0\,1\,0 \vdots 1\,1\,0 \\ 0\,0\,0\,1 \vdots 0\,1\,1 \end{bmatrix}_{(4,7)} \tag{2.46}$$

$$C_1 = \begin{bmatrix} 0\,0\,0\,1 \vdots 0\,1\,1 \\ 0\,1\,0\,1 \vdots 1\,0\,0 \\ 1\,1\,0\,0 \vdots 0\,1\,0 \\ 0\,0\,1\,0 \vdots 1\,1\,0 \end{bmatrix}_{(4,7)} \tag{2.47}$$

La matriz (2.47) está conformada por la matriz mensaje y la matriz de redundancia. El proceso de codificación se debe repetir para todos los bloques de mensajes que se deseen enviar por el canal.

2.1.5 EFICIENCIA DEL CODIGO CICLICO.

Si se considera un código cíclico de la forma (n, k) con un generador $g(x)$. Indudablemente este código tiene una eficiencia igual a:

$$e = \frac{k}{n} \tag{2.48}$$

Donde

e: Eficiencia

k: Número de bits de la palabra de datos

n: Número de bits de la palabra codificada

2.2 CODIFICACIÓN DE LOS CÓDIGOS CÍCLICOS.

En un código cíclico (n, k), cada polinomio de la palabra de código $v(x)$ en el subespacio S se puede expresar como $v(x) = m(x)g(x)$ donde $m(x)$ es el polinomio del mensaje y se escribe como:

$$m(x) = m_0 + m_1 x + m_2 x^2 + \cdots + m_{k-1} x^{k-1} \tag{2.49}$$

y $g(x)$ el polinomio generador y tiene la siguiente forma:

$$g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_{n-k} x^{n-k} \tag{2.50}$$

donde g_0 y g_{n-k} deben ser igual a 1.

A continuación se describe una forma sistemática para obtener a partir del vector de mensaje:

$$m = (m_0, m_1, \dots, m_{k-1}) \tag{2.51}$$

El vector del código

$$v = (r_0, r_1, \dots, r_{n-k-1}, m_0, m_1, \dots, m_{k-1}) \tag{2.52}$$

Donde al principio del vector se encuentran los $(n-k)$ bits de paridad y al final los k bits del mensaje. Al vector U le corresponde el polinomio de código $U(x)$ que se expresa como:

$$v(x) = r_0 + r_1 x + \cdots + r_{n-k-1}x^{n-k-1} + m_0 x^{n-k} + m_1 x^{n-k+1} + \cdots + m_{k-1}x^{n-1} \quad (2.53)$$

La (2.53) es equivalente a:

$$v(x) = r(x) + x^{n-k}m(x) \quad (2.54)$$

De (2.53) y (2.54) se ve que la codificación en forma sistemática usando código cíclico se logra colocando los dígitos del mensaje en la k etapas más hacia la derecha de registro de palabra de código y ubicando los dígitos de paridad en las $(n-k)$ etapas más hacia la izquierda. El polinomio $r(x)$ se obtiene de realizar la siguiente operación:

$$r(x) = x^{n-k}m(x)\ mod 2\ g(x) \quad (2.55)$$

2.2.1 REGISTROS DE DESPLAZAMIENTOS

Los registros de desplazamiento son un grupo de celdas de almacenamiento adecuado para mantener información binaria. Un grupo de flip-flop constituye un registro, ya que cada flip-flop es una celda binaria capaz de almacenar un bit de información. Un registro de n-bit tiene un grupo de n flip-flop y es capaz de almacenar cualquier información binaria que contengan bits. Además de los flip-flop, un registro puede tener compuertas lógicas combinacionales que realicen ciertas tareas de procesamiento de datos. En su definición más amplia, un registro consta de un grupo de flip-flop y compuertas que efectúan una transición. Los flip-flop mantienen la información binaria y las compuertas controlan cuando y como se transfiere información nueva al registro. Se utilizan para convertir la información de paralelo a serie (y vice-versa) para usar en las comunicaciones síncronas. Las comunicaciones a través de SPI, I2C, etc, se implementan con estos registros. También permiten realizar las operaciones de multiplicar por 2 y dividir entre 2 para número enteros.

La salida del registro es de N bits. Tiene una entrada en paralelo de N bits, que permite cargar en el registro con un valor nuevo. La señal permite determinar el modo de funcionamiento, es decir, cuando está a 0 se realiza la carga de un valor nuevo al llegar un flanco de subida de reloj. Cuando está a 1 se realiza un desplazamiento hacia la izquierda en el flanco de subida del reloj. En este desplazamiento el bit más significativo se pierde, y el nuevo se lee de la entrada

2.2.2 CODIFICACION CON REGISTROS DE DESPLAZAMIENTO

Los códigos cíclicos binarios son fácilmente implementados mediante registros de desplazamientos realimentados. El cálculo del síndrome también se realiza de forma muy sencilla mediante registros de desplazamientos.

Los componentes del vector (2.1) son tratados como coeficientes de un polinomio (2.8) donde la presencia o ausencia de cada término en el polinomio indica la presencia de un 1 o 0 en la correspondiente localización de la n-upla.

De la ecuación (2.53) y (2.54) se ve que la codificación en forma sistemática usando códigos cíclicos se logra colocando los dígitos del mensaje en la k etapas más hacia la derecha del registro de palabra de código y ubicando los dígitos de paridad en las (n-k) etapas más hacia la izquierda.

En la figura 2.1 se presenta el esquema de codificación de desplazamiento cíclico que se realiza en el proceso de codificación.

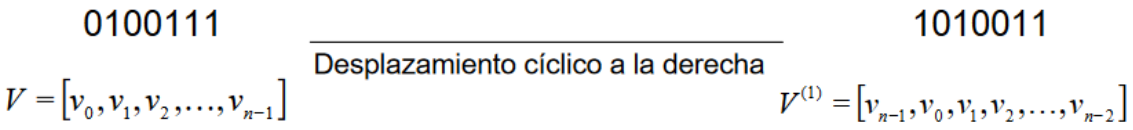

Figura 2.1 Ejemplo de desplazamiento cíclico a la derecha

En la figura 2.2 se ilustra un circuito que permite codificar un código cíclico en forma sistemática. El cálculo de los bits de paridad es el resultado de obtener $x^{n-k}m(x)$ módulo $g(x)$, en otras palabras, es la división del polinomio de mensaje desplazado hacia la derecha por el polinomio generador $g(x)$. La codificación comienza inicializando con ceros los registros r_0 a r_{n-k-1}, cerrando la llave de realimentación y colocando la llave de salida en la posición que permite que los bits del mensaje $m(x)$ pasen a la salida.

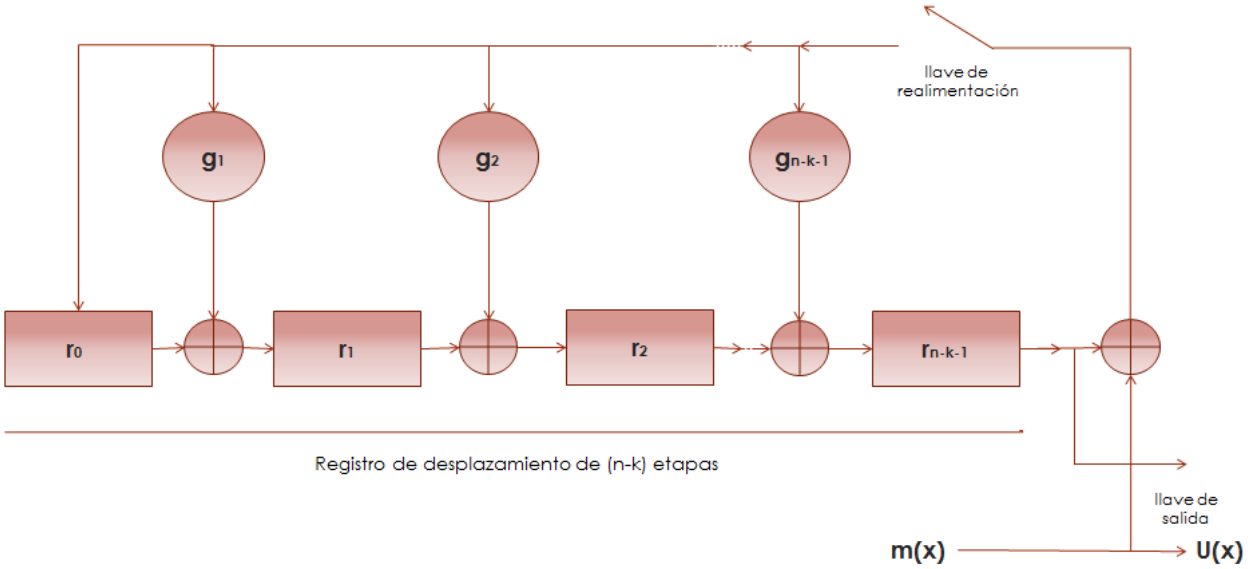

Figura 2.2 Codificador cíclico de $(n-k)$ etapas.

Los k bits del mensaje se desplazan dentro del registro r y simultáneamente se liberan hacia la salida. Luego de desplazar k veces, los registros r_0 a r_{n-k-1} contienen los bits de paridad. Entonces se abre la llave de realimentación, y la llave de salida se coloca en la posición que permite que los bits de paridad lleguen a la salida

2.3 DECODIFICACIÓN DE CÓDIGO CÍCLICO.

2.3.1 DETECCIÓN Y CORRECCIÓN DE ERRORES

De acuerdo a [6] el criterio para determinar si una palabra recibida es correcta o la transmisión sufrió interferencias es el siguiente: si la palabra recibida pertenece al conjunto de palabras código se supone que es la palabra enviada, en caso contrario se supone que produce errores durante la transmisión, sin saber exactamente cuántos. Si el error producido en la transmisión transforma una palabra código en otra palabra código, el decodificador supone que la palabra recibida es correcta.

Por otra parte, si el objetivo es corregir los errores de la transmisión, se debe establecer un criterio para sustituir los bits incorrectos, este proceso se denomina decodificación.

Sea C un codigo binario de longitud n cuya probabilidad de error es p, valor menor que 0,5, la probabilidad de que una palabra sea transmitida correctamente es:

$$(1-p) > 0.5 \tag{2.56}$$

La probabilidad de que una palabra se trasmite correctamente es

$$p(palabra\ correcta) = (1-p)^n \tag{2.57}$$

La probabilidad de que se produzcan l errores en l posiciones concretas es

$$P(error\ del\ tipo\ l\ concreto) = p^l(1-p)^{n-1} \tag{2.58}$$

Se denota por c una palabra código arbitraria y por r una palabra recibida, la probabilidad de ser enviada la palabra c y recibida como la palabra r es:

$$P(c\ enviada, r\ recivida) = p^{d(c,r)}(1-p)^{n-d(c,r)} = (\frac{p}{(1-p)})^{d(c,r)}(1-p)^n \tag{2.59}$$

Donde

$d(c,r)$: distancia de Hamming ente las palabras c y r.

Dado que $\left(\frac{p}{(1-p)}\right) < 1$, dicha probabilidad disminuye al aumentar $d(c,r)$, la distancia entre las palabras c y r.

Además de la longitud y el tamaño de un código bloque C hay un tercer parámetro de gran importancia, la distancia mínima, que permite conocer el número de errores que el código puede detectar o corregir al aplicar los procedimientos anteriores.

2.3.2 DISTANCIA MÍNIMA.

Para definir la distancia de Hamming de un código, primero es necesario introducir el concepto de distancia de Hamming entre dos palabras código.

La distancia de Hamming se define como el número de bits diferentes de cero que hay en la palabra.

Por ejemplo, el peso de Hamming de $v = (1001011)$ es igual a 4

La distancia mínima de un código, o distancia de Hamming de un código, se define como la menor distancia que hay entre dos palabras válidas del código.

Por ejemplo, la distancia Hamming entre dos palabras válidas, $v = (1001011)$ y $w = (0100011)$ es,

$$\begin{array}{l} 1001011 \\ \underline{0100011 + (\mathrm{mod}\, 2)} \\ 1101000 \end{array}$$

$$1+1+1=3$$

Por lo tanto la distancia de Hamming es: $d(v, w) = 3$

De esta definición se deduce que si la distancia de Hamming de un código vale d_h, cualquier combinación de n bits erróneos que cumpla $n < d_h$ se detecta con probabilidad 1.

2.3.3 CALCULO DEL SINDROME DE LA PALABRA RECIBIDA.

Si se considera que el vector código a transmitir, presenta perturbación por ruido al llegar al receptor, se asume que el vector que se recibe puede no coincidir con el vector original. Si se transmite la palabra de código $y(x)$ representada por:

$$y(x) = m(x)g(x) \qquad (2.60)$$

El polinomio $r(x)$ que se recibe se compone de:

$$r(x) = y(x) + e(x) \tag{2.61}$$

donde $e(x)$ representa el polinomio de error.

El síndrome $S(x)$ es el resto de dividir $r(x)$ por $g(x)$, es decir:

$$r(x) = q(x)g(x) + S(x) \tag{2.62}$$

Donde

$q(x)$: Cociente entre la palabra enviada y $g(x)$.

Combinando las ecuaciones anteriores se obtiene:

$$e(x) = [m(x) + q(x)]g(x) + S(x) \tag{2.63}$$

Observando (2.62) y (2.63) se ve que el síndrome $S(x)$ es el mismo polinomio que se obtiene del resto de dividir $e(x)$ por $g(x)$. Así el síndrome de la palabra de código recibida $r(x)$ contiene la información necesaria para la corrección de los errores producidos por el ruido.

Ejemplo cálculo de síndrome

La decodificación hace uso de una tabla de síndrome. Dicha tabla está compuesta por los síndromes de las diferentes posiciones de error del código, el síndrome está definido como el residuo de la división entre una posición de error y el polinomio característico de dicho código, en este caso es el código cíclico [7,4]. A partir de lo anterior se muestra un ejemplo para la posición de error x^6, la cual corresponde al bit más significativo de la palabra código.

$$S\big(x(6)\big) = rem\ \frac{x^6}{g(x)} = rem\ \frac{x^6}{x^3+x+1} = x^2 + 1 \tag{2.64}$$

Donde rem se define como el residuo.

Este residuo es cambiado de su forma polinómica a su correspondiente vector, utilizando $mod\ 2$, este proceso se repite desde la primera posición de error (bits más significativo) hasta la última (bits menos significativo). De esto se obtiene la siguiente tabla 2.1

Tabla 2.1 Tabla de síndrome para el código cíclico [7,4]

Error	Síndrome	Vector del síndrome
x^6	x^2+1	101
x^5	x^2+x+1	111
x^4	x^2+x	110
x^3	$x+1$	011
x^2	x^2	100
x	x	010
1	1	001

La tabla 2.1 es válida solo para el código [7,4], ya que se calcula a partir de su polinomio característico y como indica la distancia mínima de Hamming solo puede corregir un error, el cual puedes ser corregido en cualquier posición en la que se encuentre.

2.3.4 LIMITANCIAS DEL CÓDIGO.

Existe la posibilidad de que una combinación arbitraria de bits sea aceptada como palabra válida. Se ha visto que si el número de bits erróneos de una trama no excede la distancia de Hamming, la trama errónea se detecta con probabilidad 1. En caso contrario, hay dos posibilidades:

- La palabra código correspondiente a la trama errónea coincide con otra palabra código válida y, por lo tanto, no se detecta el error.
- La palabra código resultante es una palabra no válida y se detecta el error.

2.3.5 DECODIFICADOR MEGGITT.

Existen dos tipos de técnicas para la decodificación de código cíclico, estas son de forma general algébrica y no algebraica respectivamente. Se puede decir de las técnicas de forma algebraica que utilizan algebra lineal y campos aritméticos de Galois para resolver la problemática de la localización de los posibles errores. La técnica más simple para esto se encuentra en la forma no algebraica de decodificación, como lo es en el caso del decodificador Meggitt [7], donde este utiliza la determinación no algebraica del patrón de error.

Un parámetro necesario de la decodificación de códigos de control de error es el cálculo del síndrome. La palabra recibida en su forma polinomial es:

$$r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1} \tag{2.65}$$

El código polinomial $e(x)$ es definido por:

$$e(x) = e_0 + e_1 x + \cdots + e_{n-1} x^{n-1} \tag{2.66}$$

esta es la ecuación de error en su forma polinomial.

El síndrome en su forma polinomial es:

$$s(x) = s_0 + s_1 x + \cdots + s_{n-k-1} x^{n-k-1} \tag{2.67}$$

este se genera dividiendo $r(x)$ con $g(x)$, y todas las palabras de códigos son divisibles entre $g(x)$, por lo que se puede escribir:

$$\frac{r(x)}{g(x)} = a(x) + \frac{e(x)}{g(x)} \tag{2.68}$$

El síndrome requerido $s(x)$ es el residuo resultante de la división de:

$$s(x) = Rem\left[\frac{e(x)}{g(x)}\right] = e(x)\ mod 2\ g(x) \tag{2.69}$$

De acuerdo a 2.3 para el cálculo del síndrome se ingresa el polinomio $r(x)$, primeramente el símbolo r_{n-1}. Una vez es completamente ingresado el polinomio (bit por bit), el registro presenta el síndrome $s(x)$ de $r(x)$. La corrección de errores trabaja sobre un bit de la palabra recibida, al cual se asigna un $s(x)$ diferente de acuerdo a la posición de cada bit dentro de $r(x)$.

El decodificador de Meggitt para un código cíclico requiere un buffer de n etapas y un registro de síndrome de $n-k$ etapas. Este decodificador opera de la siguiente manera.

Dado un síndrome inicial $S(x)$ de $r(x)$, según la figura 2.3, este genera síndromes correspondientes a las cantidades de desplazamientos cíclicos que presente el polinomio $r(x)$. Entonces, si $r(x)$ se retiene en un buffer $n-símbolos$, se pueden decodificar estos síndromes correspondientes a un error en el final de este buffer y así realizar la decodificación bit por bit de acuerdo a la posición del error que presenta $S(x)$.

La figura 2.3 presenta el circuito para el cálculo del síndrome de código cíclico (n, k) para decodificador Meggitt.

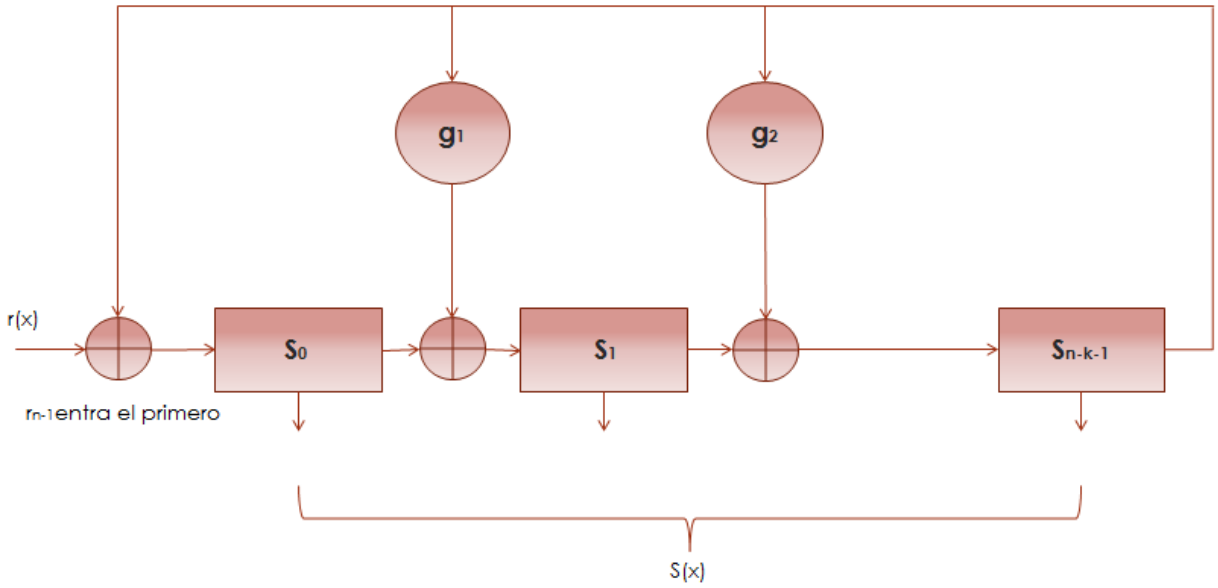

Figura 2.3 Circuito práctico para el cálculo del síndrome para decodificador Meggitt de códigos cíclicos

De acuerdo a la figura 2.4 el decodificador Meggitt opera de la siguiente manera:

Paso 1: Con la compuerta 1 abierta y la compuerta 2 cerrada $r(x)$ se almacena simultáneamente dentro del buffer y del registro de síndrome. Cuando la palabra recibida r_{n-1} ocupa la última etapa del buffer, el registro de síndrome contiene un $s(x)$ correspondiente a $r(x)$.

Paso 2: La corrección bit por bit se realiza con la compuerta 1 cerrada y la compuerta 2 abierta. Si r_{n-1} no es un bit erróneo, tanto el registro de síndrome como el buffer se desplaza simultáneamente, dando $s'(x)$ correspondiente a $r'(x)$. Si $s(x)$ corresponde a un error corregible en el símbolo r_{n-1} entonces el detector de error generara $\hat{e}_{n-1} = 1$ para completar el bit.

El polinomio corregido es:

$$r_1(x) = r_0 + r_1 x + \cdots + r_{n-2}x^{n-2} + (r_{n-1} + \hat{e}_{n-1})x^{n-1} \tag{2.70}$$

Un desplazamiento cíclico a la derecha es:

$$r_1(x) = (r_{n-1} + \hat{e}_{n-1}) + r_0 + \cdots + r_{n-2}x^{n-2} \tag{2.71}$$

Paso 3: Si r_{n-1} presenta un error, el síndrome correspondiente a $r'(x)$ se obtiene como $s_1'(x) = s'(x) + 1$, eso es, que se debe adicionar un 1 a la entrada del registro de síndrome mientras se desplaza. Se modifica el registro de síndrome y se corrige el efecto del error en r_{n-1} desde $s(x)$.

Paso 4: El símbolo r_{n-2} está ahora en el registro final del buffer y se corrige si es necesario utilizando el nuevo $s(x)$. La decodificación se detiene cuando el ultimo bit de $r(x)$ se corrige.

Los pasos que sigue el decodificador Meggitt son:

1 .Se corrige el bit más significativo

2. Se realiza cálculo del síndrome

3. Se realiza un nuevo desplazamiento

4. Se calcula un nuevo síndrome

5. Se ve el bit más significativo, se complementa el bit

6. Se repite todo los puntos anteriores

7. Último bit.

La figura 2.4 presenta el circuito del decodificador Meggitt para código cíclico.

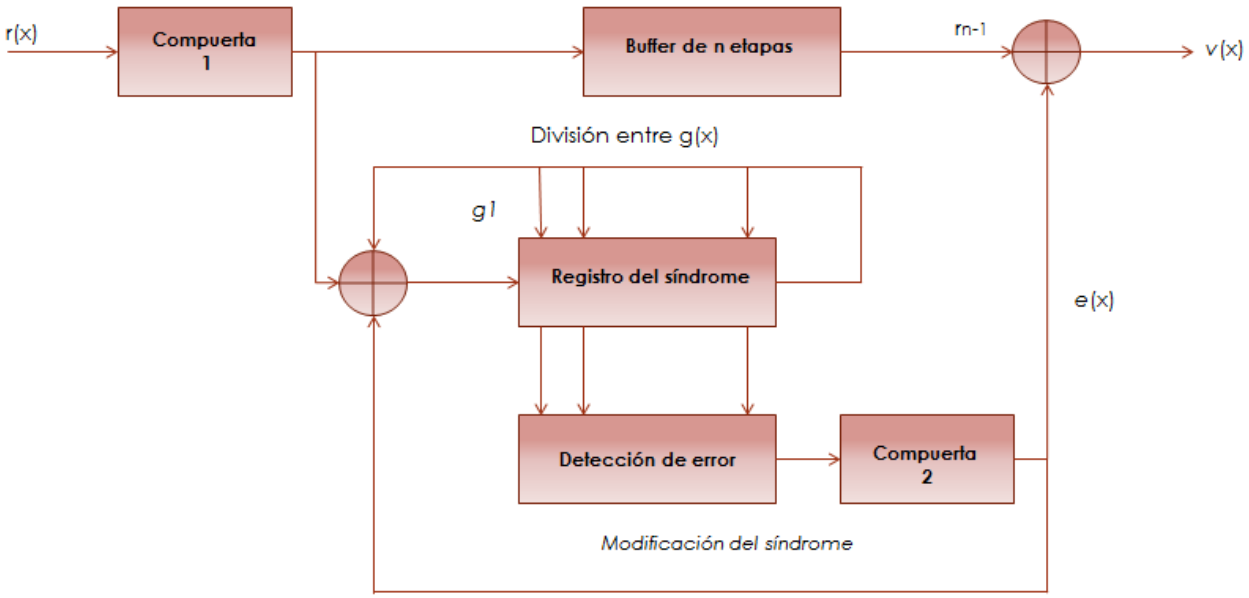

Figura 2.4 Circuito práctico del decodificador Meggitt de código cíclico.

Ejemplo codificador y decodificador código cíclico

A continuación se detalla el proceso de codificación y decodificación del código cíclico [7,4]. En primer lugar se usa el polinomio característico, el cual se define como:

$$g(x) = x^3 + x + 1 \tag{2.72}$$

en base a este polinomio se calcula la matriz generadora. Esta matriz se define como:

$$G=\left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$\left[\begin{array}{ccc} I & \vdots & A \end{array}\right] \tag{2.73}$$

Donde

I = matriz identidad.

A= matriz de bit de paridad.

Una vez se obtiene la matriz G, ahora debemos comprobar que el código que hemos generado es capaz de corregir los errores producidos por el canal, para ellos hallamos la distancia entre todas las palabras del código. Obtenemos que d = 3 por lo que sabemos que es capaz de detectar dos errores y corregir al menos uno, la tabla 2.1 muestra los síndromes para las posiciones de error.

Tabla 2.1 Tabla de síndrome

Tabla del síndrome	
Síndrome	Posición del error
101	1000000
111	0100000
011	0010000
110	0001000
001	0000100
010	0000010
100	0000001

La tabla 2.1 permite la corrección de un error en cualquier posición posible. El bloque de mensajes a transmitir es:

$$M=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.74}$$

Donde

Mensaje (1)= [0 0 0 1]

Mensaje (2)= [0 1 0 1]

Mensaje (3)= [1 1 0 0]

Mensaje (4)= [0 0 1 0]

Se realiza el proceso de codificación, en el cual se multiplica la matriz mensaje con la matriz G MOD 2, se obtiene las siguientes palabras codificadas:

$$C_1 = \left[\begin{array}{cccc:cccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right] \tag{2.75}$$

La matriz C_1 se compone por la matriz mensaje y la matriz de redundancia (bits de paridad). Este mensaje se envía a través del canal en presencia de ruido, el cual se simula por el modelo de Middleton y se envía a un detector umbral, donde se recibe las siguientes palabras:

$$R = \left[\begin{array}{cccc:cccc} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}\right] \tag{2.76}$$

En la tabla 3.2 se muestra la corrección de la primera fila del mensaje, se observa que el valor del síndrome en el primer vector es [111] lo que permite establecer que el error se encuentra en la segunda posición, por lo que se procede al arreglo de este bit.

Tabla 3.2 Calculo de síndrome y desplazamiento de la primera fila de mensaje recibido

$m=[0101011]$
$S(0101011)=[111]$
$S(1000101)=[000]$
$S(1100010)=[000]$
$S(0110001)=[000]$
$S(1011000)=[000]$
$S(0101100)=[000]$
$S(0010110)=[000]$

Se repite el proceso hasta completar la última fila y se obtienen las palabras códigos corregida:

$$R'=\left[\begin{array}{cccc:ccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right]$$

$\overbrace{\text{mensaje corregido}}$ $\overbrace{\text{matriz de redundancia}}$

(2.77)

Quitando la matriz de redundancia se obtiene las palabras mensajes

$$M_5=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (2.78)$$

CAPÍTULO III

CÓDIGO PRODUCTO CÍCLICO

3.1 INTRODUCCION

Según H. BURTON y E. WELDON [8], debido a la facilidad con que se implementan, los códigos cíclicos, tienen una mayor utilidad en los sistemas de control de errores diseñados hasta la fecha. Estos sistemas son fundamentalmente de dos tipos. Primero están los que corrigen errores por detección y retransmisión, en segundo lugar están los códigos que corrigen errores usando "forward-acting". Los códigos cíclicos poseen significativas ventajas en ambas situaciones.

En este trabajo se considera una nueva clase de código cíclico, el código cíclico producto. Estos códigos se proponen para la corrección de errores en canales tales como los que se encuentran en la red telefónica conmutada, en las cuales los errores no se producen de manera aleatoria o en interrupciones bien definidas. En tales canales, las clases que se conocen de código cíclico no parecen estar muy bien adaptadas a la corrección de errores. En cambio los códigos Bose-Chaudhuri-Hocquenghem (BCH), por ejemplo, están muy cercanos a lo óptimo en su capacidad de corrección de errores en canales donde este tipo de errores ocurren aleatoria e independientemente, pero esta clase de código no es útil siendo un corrector de interrupciones. Por otro lado, para errores de interrupciones bien definidos, los códigos "Fire" son efectivos pero tienen una capacidad de corrección regular para errores aleatorios. El código cíclico producto presentan un compromiso entre los códigos de corrección de errores aleatorios y errores de interrupción, el cual se debe, por lo tanto, permitir algo más de libertad en el diseño del sistema de control de errores en la red de comunicación.

Estos códigos son al mismo tiempo tanto códigos cíclicos como códigos producto. Un código producto bidimensional se representan usualmente por un arreglo rectangular en el cual cada columna es una palabra de código en un sub-código y cada fila es una palabra de código en otro (no necesariamente el mismo código). Esta idea se generaliza de una manera directa a un número arbitrario de dimensiones.

3.2 ESTRUCTURA Y PROPIEDADES DE LOS CODIGOS PRODUCTO.

Peterson [9] señala que el producto de dos códigos cíclicos con generador de polinomios $x+1$ es cíclico, siempre que la longitud del sub-código difiera en uno. Él muestra que el generador $g(x)$ del código producto es:

$$g(x) = \frac{(x^{n_1}+1)(x^{n_2}+1)}{x+1} \tag{3.1}$$

donde n_1 y n_2 denotan la longitud de los sub-códigos.

En este trabajo el resultado se generaliza a los códigos cíclicos producto cuyos sub-códigos tienen un generador de polinomios arbitrario, y esto condiciona la restricción de la longitud de estos sub-códigos.

Si se considera un código producto arbitrario de dos dimensiones cuyos sub-códigos son códigos cíclicos (n_1, k_1) y (n_2, k_2) con un generador $g_1(x)$ y $g_2(x)$ respectivamente. Esto implica que el código presenta una longitud $n = n_1 n_2$, eficiencia $k/n = k_1 k_2 / n_1 n_2$ y una distancia mínima $d = d_1 d_2$.

En el teorema 4 se demuestra que bajo ciertas condiciones este código producto es equivalente a un código cíclico. El teorema 6 da una regla mediante la cual el generador de polinomios de este código cíclico se determina desde el generador de polinomios de los sub-códigos. Antes de afirmar y demostrar estos teoremas es necesario introducir algunas notaciones previas.

Una palabra de código de longitud $n_1 n_2$ es normalmente representada por un polinomio de grado $n_1 n_2 - 1$, es así como

$$p(x) = p_0 x^0 + p_1 x^1 + \ldots + p_{n_1 n_2 - 1} x^{n_1 n_2 - 1} \tag{3.2}$$

El coeficiente p_i es en general, la combinación lineal de los k símbolos de información.

Ahora si se considera un código producto cuyo arreglo tiene n_1 columnas y n_2 filas, sin perder la generalidad se puede suponer que $n_1 \geq n_2$. Cada posición del arreglo se señala por un par ordenado (i, j), donde i señala las filas y j señala las columnas.

Un mapeo $m(i, j)$ se define relacionando los (i, j) elementos del arreglo con el coeficiente de los $(l+1)$ términos del polinomio. Esto es:

$$p_l = a_{ij} \tag{3.3}$$

Donde $l = m(i, j)$ y a_{ij} es el elemento (i, j) del arreglo.

En otras palabras, $m(i, j)$ define el orden en el cual los dígitos del arreglo se transmiten. Este orden no es fila por fila o columna por columna como es normalmente en el caso de los códigos producto. Entonces, el orden se define y se asume como $m(i, j)$, con este orden, el producto de dos códigos cíclicos es un código cíclico.

Las condiciones en el arreglo y la definición de $m(i, j)$ en la cual resulta un código producto a partir de códigos cíclicos se define en el teorema 4:

Teorema 4

El producto de dos códigos cíclicos es por sí mismo un código cíclico siempre que:

1) Los valores de las longitudes de los sub-códigos sean relativamente primos, es decir, $an_1 + bn_2 = 1$ donde a y b son enteros

2) El mapeo $m(i, j)$ se define como:

$$m(i, j) \equiv (j - i)bn_2 + i\,, \qquad m(i, j) = 0,\ 1,\ ...,\ n_1n_2 - 1 \tag{3.4}$$

Demostración

Antes de proceder con la demostración es conveniente definir el lema 1:

Lema 1:

$$m(i, j) = m(i', j') \text{ Si y solo si } i \equiv i'(n_2) \text{ y } j \equiv j'(n_1)\,. \tag{3.5}$$

Demostración del lema

El siguiente enunciado muestra que son equivalentes

$$m(i, j) = m(i', j') \tag{3.6}$$

$$(j - i)bn_2 + i \equiv (j' - i')bn_2 + i' \tag{3.7}$$

$$(j - j')bn_2 + (bn_2 - 1)(i - i') \equiv 0 \tag{3,8}$$

$$(j - j')bn_2 + (i - i')an_1 \equiv 0 \tag{3.9}$$

La (3.9) es cierta cuando se cumple que $i \equiv i'(n_2)$ y $j \equiv j'(n_1)$, lo que demuestra la equivalencia del enunciado.

Se presume que (3.6) se mantiene; reordenando la (3.8) queda de la siguiente manera

$$[(j - j') + (i' - i)]bn_2 + (i - i') \equiv 0 \tag{3.10}$$

Lo que implica que $(i - i')$ es divisible por n_2, o $(i - i') \equiv 0(n_2)$. Una operación similar en (3.9) después de reemplazar bn_2 por $1 - an_1$, lleva a la conclusión de que $(j - j') \equiv 0(n_1)$. Esto completa la demostración del lema.

Con el fin de demostrar que el código es cíclico es suficiente por ahora mostrar que el desplazamiento cíclico de una palabra de código es una palabra de código. Esto es verdadero si cada fila y columna es un desplazamiento cíclico de una fila y columna del arreglo original.

Particularmente una condición suficiente es:

$$m(i, j) + 1 \equiv m(i + 1, j + 1) \tag{3.11}$$

donde $i + 1$ y $j + 1$ deben interpretarse como $(i + 1) \mod n_2$ y $(j + 1) \mod n_1$,

$$m(i + 1, j + 1) \equiv (j - i)bn_2 + i + 1 \mod n_1 n_2 \tag{3.12}$$

$$\equiv \{[(j - i)bn_2 + i] \mod n_1 n_2 + 1\} \mod n_1 n_2 \tag{3.13}$$

$$\equiv m(i, j) + 1 \mod n_1 n_2 \tag{3.14}$$

queda demostrado que

Esto muestra que bajo ciertas condiciones un código cíclico es un código producto.

Esto se expresa de la siguiente manera en el corolario 1.

Corolario 1:

La mínima distancia d entre la palabra de código en un código cíclico producto es igual al producto de la mínima distancia de los sub-códigos, esto es:

$$d = d_1 d_2 \tag{3.15}$$

Claramente el código es capaz es corregir t errores, donde:

$$t = \left\lfloor \frac{d-1}{2} \right\rfloor \tag{3.16}$$

donde el símbolo $\lfloor x \rfloor$ denota el entero más grande que contiene en x.

Se considera un código cíclico producto cuyas filas son un sub-código que tienen una longitud n_1, este igualmente tiene una capacidad para corregir t_1 errores aleatorios y una capacidad para corregir B_1 interrupciones impulsivas. Las columnas correspondientes a este código cíclico son un sub-código que tienen una longitud n_2 e igualmente tienen una capacidad para corregir t_2 errores aleatorios y una capacidad para corregir B_2 interrupciones. Entonces B denota la capacidad de corregir errores de interrupción del código cíclico producto. Esto se presenta de la siguiente manera:

$$B \geq n_1 t_2 + B_1 \tag{3.17}$$

y

$$B \geq n_2 t_1 + B_2 \tag{3.18}$$

La primera de esas aseveraciones puede ser verdadera de la siguiente manera. Si se considera una interrupción impulsiva de longitud $n_1 t_2 + B_1$, la capacidad B_1 de corregir columnas cíclicamente adyacentes que tiene t_2 bits de estas interrupciones. Cada capacidad B_1 presente en cada una de las columnas cíclicamente adyacentes que tiene en $t_2 + 1$ de esos bits. Si las

columnas se codifican primero, claramente, a lo sumo, B_1 tiene errores después de decodificar las columnas. Ya que el código de fila es capaz de corregir cualquier interrupción de longitud no mayor que B_1, todos los errores restantes se corrigen. Claramente las interrupciones más cortas se corrigen de una manera similar.

La segunda aseveración se demuestra con un argumento similar en el que las filas se decodifican antes que las columnas. Si los sub-códigos son de tal manera que $n_1 = bn_2 \pm 1$, entonces el resultado que más sentido tiene es el siguiente

$$B \geq n_1 B_2 + B_1 \tag{3.19}$$

Esto es por lo que, en este caso, una interrupción de longitud $n_1 B_2 + B_1$ afecta los B_2 bits adyacentes en $n_1 - B_1$ columnas y en $B_2 + 1$ bits adyacentes en las restantes B_1 columnas adyacentes. Si las columnas se decodifican primero, solo B_1 columnas adyacentes una vez decodificadas contienen errores. Estos se corrigen con el código fila.

Si uno de los sub-códigos en el código producto es en sí mismo un producto de $r-1$ códigos cíclicos, el código producto se considera como un código cíclico producto compuesto de r códigos cíclicos. En consecuencia, el teorema I se extiende inductivamente para dar el corolario 2.

Corolario 2

El producto de un número arbitrario de códigos cíclicos es por sí mismo un código cíclico siempre que:

1) La longitud de los sub-códigos sean relativamente números primos.

2) El mapeo entre las posiciones en el arreglo y los términos en la representación polinómica del código cíclico está dada por la aplicación sucesiva de la función $m(i, j)$ definida en el teorema 4.

Los resultados anteriores caracterizan la capacidad que presenta el código cíclico producto de corregir errores aleatorios y errores de interrupción impulsiva. Es importante darse cuenta, que no obstante, la expresión para t y los límites en B no necesariamente se mantienen al mismo tiempo. Las conclusiones a considerar para estos puntos son:

1) Si se utiliza para corrección de errores aleatorios, a lo más, t errores se corrigen, donde t se define en el corolario 1.
2) Si se usa para corrección de errores de interrupción, las interrupciones de longitud B o menores, se corrigen donde B es el límite inferior.

En el teorema 5 se muestran las condiciones que ocurren cuando se producen simultáneamente errores aleatorios y errores de interrupción impulsiva.

Teorema 5

El sub-conjunto contiene patrones de errores aleatorios de valor:

$$\left\lfloor \frac{d_1 d_2 - 1}{2} \right\rfloor \tag{3.20}$$

o menor los cuales están separados del sub-conjunto que contiene las interrupciones de longitud máxima $(n_1 t_2, n_2 t_1)$ o menor (exceptuando, por supuesto, los patrones de errores que satisfacen ambas condiciones).

Demostración del teorema 5:

Se supone que $n_1 t_2 \geq n_2 t_1$. Se considera una interrupción impulsiva de longitud $n_1 t_2$ o menor en una única palabra de código. Cada columna del arreglo contiene al menos t_2 errores.

Se supone que este arreglo de interrupciones y algunos de los arreglos de errores aleatorios corregibles son miembros del mismo sub-conjunto relativo del código cíclico producto. Entonces la suma de los patrones de dos errores (arreglo de errores) es un arreglo de código. Para que esto se cumpla, la suma de cada columna del arreglo debe tener valor cero o un valor de al menos d_2. Cada suma de la columna del arreglo con términos distintos de cero se compone de al menos $d_2 - t_2$ errores del arreglo de errores aleatorios y a lo sumo de t_2 errores del arreglo de errores de interrupción.

Ya que hay como máximo t errores aleatorios, estos errores se distribuyen entre un máximo de:

$$\left\lceil \frac{t}{d_2 - t_2} \right\rceil \text{ Columnas} \qquad (3.21)$$

.

Así la suma del arreglo contiene a los más:

$$\left\lceil \frac{t}{d_2 - t_2} \right\rceil t_2 + t \qquad (3.22)$$

elementos distintos de cero.

Sin embargo:

$$\left\lceil \frac{t}{d_2 - t_2} \right\rceil t_2 + t \le t\left\{ \frac{t_2}{d_2 - t_2} + 1 \right\} = t\left\{ \frac{d_2}{d_2 - t_2} \right\} < 2t \qquad (3.23)$$

Por lo tanto, la suma del arreglo contiene (como máximo) menos que $2t < d$ elementos distintos de cero y no puede ser un arreglo de código. Esta contradicción implica que los patrones de error están en un subconjunto distinto. Si $n_2 t_1 > n_1 t_2$ el mismo argumento se aplica a las filas en vez de a las columnas. Queda demostrado que.

En el caso de $n_1 = bn_2 \pm 1$ el resultado que se obtiene es más categórico.

Para esto se tiene el corolario 3.

Corolario 3

Se considera un código cíclico producto con $n_1 = bn_2 \pm 1$. Los subconjuntos contienen los patrones de errores aleatorios de valor r o menor que están separados de los subconjuntos en los cuales tienen las interrupciones impulsivas de longitud $n_1 s$ o menor (nuevamente exceptuando los patrones que satisfacen ambas condiciones), siempre que:

$$s \le \min(B_2, d_2), \qquad \mathrm{r} \le \left\lfloor \frac{d-1}{2} \right\rfloor \tag{3.24}$$

y

$$\frac{r}{d} < 1 - \frac{s}{d_2} \tag{3.25}$$

Demostración del corolario 3:

Esta demostración sigue la misma línea que la del teorema 5. Para ver esto se reemplaza t_2 por s y t por r en la inecuación de la demostración del teorema 5.

Cuando el polinomio $x+1$ es un factor del generador polinomial de un corrector de t errores de un código cíclico binario, muestra que el código es capaz de corregir todas las interrupciones impulsivas de longitud no superior que $t+1$. Si tal código se usa como el código columna de un código cíclico producto y $n_1 = bn_2 \pm 1$ entonces, el corolario 3 mantiene que $s = t_2 + 1$ y

$$r = t = \left\lfloor \frac{d-1}{2} \right\rfloor \tag{3.26}$$

Dado que un código es cíclico la existencia de un único generador de polinomios se garantiza. Debido a que esto sucede, el generador de polinomios de un código cíclico producto es una simple función del generador de polinomios de dichos sub-códigos.

Con relación a esta se tiene el teorema 6.

Teorema 6

Si $g_1(x)$ y $g_2(x)$ son los respectivos generadores de polinomios de los sub-códigos de la fila y columna de un código cíclico producto, el polinomio generador de un código producto es el mayor común divisor (GCD) de:

$$g_1(x^{bn_2})g_2(x^{an_1}) \tag{3.27}$$

y

$$x^{n_1n_2}-1. \tag{3.28}$$

Demostración del teorema 6:

Se tiene el polinomio:

$$r_i(x)=\sum_{j=0}^{n_1-1}a_{ij}x^j \tag{3.29}$$

y

$$c_j(x)=\sum_{i=0}^{n_2-1}a_{ij}x^i \tag{3.30}$$

indica la i-enésima fila y la j-enésima columna del arreglo $\left[a_{ij}\right]$.

$$p(x)=\sum_{i,j}a_{ij}x^{m(i,j)}\equiv\left[\sum_i\sum_j a_{ij}x^{(j-i)bn_2+i}\right]\bmod(x^{n_1n_2}-1) \tag{3.31}$$

ya que

$$\sum_i\sum_j a_{ij}x^{(j-i)bn_2+i}=\sum_i x^{i(1-bn_2)}\sum_j a_{ij}x^{jbn_2}=\sum_i x^{ian_1}r_i(x^{bn_2}), \tag{3.32}$$

$$p(x)=\sum_{i,j}a_{ij}x^{m(i,j)}\equiv q(x)(x^{n_1n_2}-1)+\sum_i x^{ian_1}r_i(x^{bn_2}) \tag{3.33}$$

Para un $q(x)$. Ya que $g_1(x^{bn_2}) \mid r_i(x^{bn_2})$, es el GCD de $g_1(x^{bn_2})$ y $x^{n_1 n_2} - 1$ pueden dividir a $\sum_{i,j} a_{ij} x^{m(i,j)}$ para todos los arreglos $\left[a_{ij}\right]$. Así, $GCD\left[g_1(x^{bn_2}), x^{n_1 n_2} - 1\right]$ es un factor de $g(x)$, el cual es el generador del código producto. Análogamente

$$p(x) = \sum_{i,j} a_{ij} x^{m(i,j)} \equiv \left[\sum_i x^{jbn_2} c_j(x^{an_1})\right] \bmod(x^{n_1 n_2} - 1) \tag{3.34}$$

y

$$GCD\left[g_2(x^{an_1}), x^{n_1 n_2} - 1\right] \mid g(x). \tag{3.35}$$

Por lo tanto, el mínimo común divisor (LCD) de $GCD\left[g_1(x^{bn_2}), x^{n_1 n_2} - 1\right]$ y $GCD\left[g_2(x^{an_1}), x^{n_1 n_2} - 1\right]$ dividen $g(x)$.

Se escribe

$$g_1(x) = \sum_j \varepsilon_j x^i \quad \text{y} \quad g_2(x) = \sum_i \delta_i x^i \tag{3.36}$$

Se considera este arreglo de filas

$$r_i(x) = \delta_i g_1(x) \tag{3.37}$$

Y de columnas

$$c_j(x) = \varepsilon_j g_2(x) \tag{3.38}$$

El correspondiente polinomio del código cíclico producto $p_s(x)$ es:

$$p_s(x) \equiv \left[\sum_i x^{ian_1} r_i(x^{bn_2})\right] \bmod(x^{n_1 n_2} - 1) \tag{3.39}$$

$$\sum_i x^{ian_1} r_i(x^{bn_2}) = \sum_i x^{ian_1} \delta_i g_1(x^{bn_2}) = g_1(x^{bn_2}) g_2(x^{an_1}) \tag{3.40}$$

Por lo tanto, $g_1(x^{bn_2})g_2(x^{an_1}) = p_s(x) + q(x)(x^{n_1n_2} - 1)$ para un $q(x)$. Porque $g(x) \mid p_s(x)$ y $g(x) \mid x^{n_1n_2} - 1$, $g(x) \mid g_1(x^{bn_2})g_2(x^{an_1})$. Entonces $g(x) \mid GCD\left[g_1(x^{bn_2})g_2(x^{an_1}), x^{n_1n_2} - 1 \right]$.

se tiene

$$L(x) = LCM\left\{ GCD\left[g_1(x^{bn_2}), x^{n_1n_2} - 1 \right], GCD\left[g_2(x^{an_1}), x^{n_1n_2} - 1 \right] \right\} \tag{3.41}$$

y

$$G(x) = GCD\left[g_1(x^{bn_2})g_2(x^{an_1}), x^{n_1n_2} - 1 \right] \tag{3.42}$$

Se demuestra que

$$L(x) \mid g(x) \tag{3.43}$$

y

$$g(x) \mid G(x) \tag{3.44}$$

Por consiguiente, $L(x) \mid G(x)$. Esto efectivamente cumple que $L(x) = G(x)$.

La única posibilidad para una inigualdad es que un factor de $x^{n_1n_2} - 1$ sea también un factor de $g_1(x^{bn_2})$ y $g_2(x^{an_1})$ que tengan una gran multiplicidad en $G(x)$ que en $L(x)$.

Se supone que las palabras de código son polinomios con coeficientes en el conjunto de los números primos con p elementos y se supone $n_1n_2 = p^s n$ donde n y p son primos relativos y s es un entero positivo o cero. Entonces

$$x^{n_1n_2} - 1 = (x^n - 1)^{p^s} \tag{3.45}$$

$x^n - 1$ no presenta factores repetidos, así cada factor irreductible de $x^{n_1n_2} - 1$ se repite exactamente p^s veces. Ya que n_1 y n_2 son relativamente primos entre sí, existe la posibilidad que $n_1 = p^s n'_1$ y n_2 sea primo en p o viceversa. En el caso $n_1 = p^s n'_1$

$$g_2(x^{an_1}) = \left[g_2(x^{an'1}) \right]^{p^s} \tag{3.46}$$

Por lo tanto, cada factor común de $x^{n_1 n_2} - 1$, $g_2(x^{an_1})$ y $g_1(x^{bn_2})$ tiene multiplicidad p^s en ambos $G(x)$ y $L(x)$. El mismo argumento se usa si $n_2 = p^s n'_2$ y n_1 es primo en p. Por lo tanto:

$$g(x) = L(x) = G(x) \tag{3.47}$$

Corolario 4

$$g(x) = LCM\left\{ GCD\left[g_1(x^{bn_2}), x^{n_1 n_2} - 1 \right], GCD\left[g_2(x^{an_1}), x^{n_1 n_2} - 1 \right] \right\} \tag{3.48}$$

Corolario 5

El generador polinomial de un código cíclico producto compuesto de un número arbitrario de productos se encuentra por sucesivas aplicaciones del teorema 6.

La estructura de un código cíclico producto al terminar el proceso de codificación tiene dimensión (n_1, n_2), la cual está compuesta por la matriz mensaje (k_1, k_2) y los bits de redundancia r_1 y c_2. En la figura 3.1 se muestra un ejemplo de la estructura de un código cíclico producto.

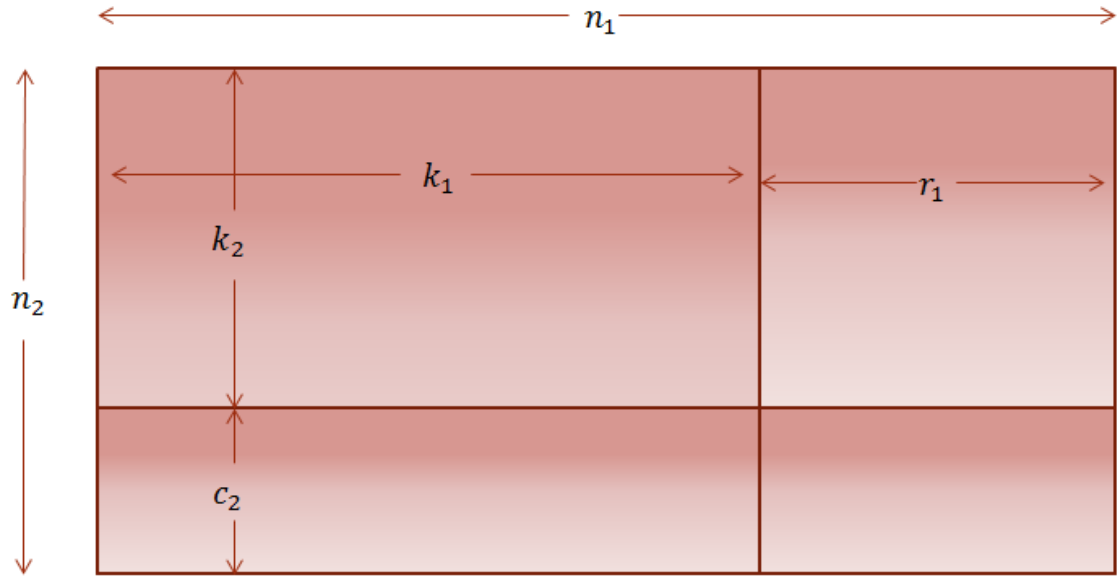

Figura 3.1 Estructura de un código cíclico producto.

A continuación se muestra un ejemplo de la estructura del mensaje M el cual es codificado a través de código cíclico producto M'.

$$M=\begin{bmatrix} k_{11} & k_{12} & k_{13}....k_{1n} \\ k_{21} & k_{22} & k_{23}....k_{2n} \\ k_{31} & k_{32} & k_{33}....k_{3n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ k_{n1} & k_{n2} & k_{n3}....k_{nn} \end{bmatrix} \tag{3.49}$$

$$M'=\left[\begin{array}{cccc:cccc} \overbrace{k_{11} & k_{12} & k_{13}....k_{1n}}^{\text{mensaje}} & & \overbrace{r_{11} & r_{12} & r_{13}...r_{1n}}^{\text{redundancia horizontal}} \\ k_{21} & k_{22} & k_{23}....k_{2n} & & r_{21} & r_{22} & r_{23}...r_{2n} \\ k_{31} & k_{32} & k_{33}....k_{3n} & & r_{31} & r_{32} & r_{33}...r_{3n} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ k_{n1} & k_{n2} & k_{n3}....k_{nn} & & r_{n1} & r_{n2} & r_{n3}...r_{nn} \\ \hdashline c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16}.....c_{1n} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26}.....c_{2n} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ c_{n1} & c_{n2} & c_{n3} & c_{n4} & c_{n5} & c_{n6}.....c_{nn} \end{array}\right] \tag{3.50}$$

redundancia vertical

Ejemplo codificador y decodificador código cíclico producto

A continuación se detalla el proceso de codificación y decodificación del código [7,4] para código cíclico producto. En primer lugar se usa el polinomio característico, el cual es definido como

$$g(x) = x^3 + x + 1 \tag{3.51}$$

En base a este polinomio se calcula la matriz generadora G. Esta matriz se define como:

$$G=\left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

$$\left[\begin{array}{c:c:c} \quad I \quad & \quad \vdots \quad & \quad A \quad \end{array}\right] \tag{3.52}$$

Donde

I = Matriz identidad.

A= Matriz del polinomio característico.

Una vez se obtiene la matriz G y ya que el código cíclico [7,4] presenta una distancia mínima de Hamming de valor 3, se estima la siguiente tabla de síndrome (2.3.3):

Tabla 3.1 Tabla de síndrome

Tabla del síndrome	
101	1000000
111	0100000
011	0010000
110	0001000
001	0000100
010	0000010
100	0000001

Esta tabla permite la corrección de un error en cualquier posición. Una vez dispuesto todo lo anterior se envía el siguiente mensaje:

$$M=\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.53}$$

Donde

Mensaje (1)= [1 1 1 0]

Mensaje (2)= [0 0 1 1]

Mensaje (3)= [1 0 1 1]

Mensaje (4)= [1 1 1 1]

Se procede a realizar la codificación del código cíclico producto, multiplicando la matriz G con la matriz mensaje, la matriz mensaje queda de la siguiente forma:

$$M=\left[\begin{array}{cccc:cccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \tag{3.54}$$

La matriz M se compone de la matriz mensaje original y la matriz de redundancia de las filas, luego se multiplica nuevamente por la matriz G, por lo que el proceso el mensaje codificado queda de la siguiente forma:

$$M = \left[\begin{array}{cccc:cccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hdashline 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right] \tag{3.55}$$

La matriz M se compone por cuatro cuadrantes, estos cuadrantes se constituyen por la matriz mensaje, la matriz de redundancia de filas, la matriz de redundancia de columnas y la matriz de redundancia de la redundancia de las columnas.

Con esto se termina el proceso de codificación y se envía el mensaje codificado a través del canal en presencia del ruido, el cual es simulado por el modelo de Middleton, donde se recibe de la siguiente manera

$$M = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ \hdashline 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right] \tag{3.56}$$

Se decodifican las filas de la matriz recibida

En la tabla 3.2 muestra el desplazamiento de la fila, a cada desplazamiento le corresponde un cálculo del síndrome.

Tabla 3.2 cálculo del síndrome y desplazamiento de la primera fila de mensaje enviado

$m=[1110000]$
$S(1110000)=[001]$
$S(0111010)=[000]$
$S(0011101)=[000]$
$S(1001110)=[000]$
$S(0100111)=[000]$
$S(1010011)=[000]$
$S(1101001)=[000]$

El valor del síndrome en el primer vector es [001], la tabla permite establecer que el error se encuentra en la quinta posición, por lo que procede al arreglo de este bit en particular. Una vez se cumple este arreglo este se repite para todas filas restantes

Al finalizar estos procesos la matriz obtenida es la siguiente

$$M=\left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hdashline 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right] \quad (3.57)$$

Se quita la matriz redundancia, y se obtiene

$$M=\left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \tag{3.58}$$

Se decodifican las filas de la matriz sin redundancia. En la tabla 3.3 se ejemplifica como es usada la primera fila de mensaje enviado:

Tabla 3.3 cálculo del síndrome y desplazamiento de la primera fila de mensaje enviado

$m=[1110100]$
$S(1110100)=[000]$
$S(0111010)=[000]$
$S(0011101)=[000]$
$S(1001110)=[000]$
$S(0100111)=[000]$
$S(1010011)=[000]$
$S(1101001)=[000]$

La tabla anterior muestra el desplazamiento de cada bits de la fila, a cada desplazamiento le corresponde un cálculo de síndrome, se observa que el valor del síndrome en el primer vector es [000]. Esto indica que el vector no presenta error por lo cual no modifica el vector mensaje y sigue su curso de desplazamiento. Este proceso se repite para todas las filas restantes del mensaje.

Al finalizar estos procesos la matriz obtenida es la siguiente

$$M=\left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \tag{3.59}$$

Se quita la matriz de redundancia de las filas de, por lo que se obtiene:

$$M=\left[\begin{array}{cccc} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right] \tag{3.60}$$

CAPITULO 4

IMPLEMENTACIÓN DE UN CODIFICADOR Y DECODIFICADOR DE UN CODIGO PRODUCTO CICLICO Y CODIGO CICLICO, PARA UN CANAL CON MEDIO AMBIENTE INDUSTRIAL.

4.1 SISTEMAS DE COMUNICACIONES A SIMULAR.

La implementación de este sistema está compuesta de tres etapas, la primera es la de un codificador cíclico, donde se modifica el mensaje estableciendo las características propias de cada tipo de codificación. Posteriormente se encuentra la etapa del ruido, en la cual se usa el modelo de Middelton. La última etapa que se presenta es la del decodificador, en esta etapa se usa el decodificador a través de tabla de síndrome debido al alto rendimiento que este presenta. La figura 4.1 muestra el esquema general del sistema de comunicación.

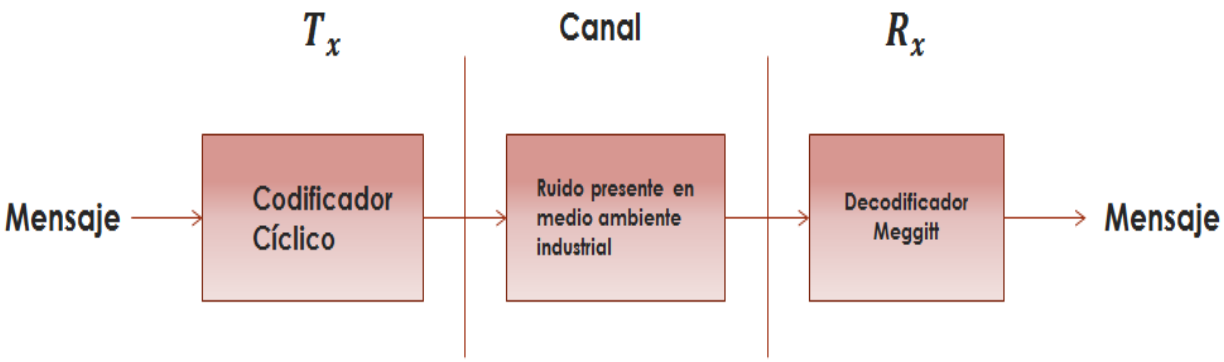

Figura 4.1, Canal de comunicaciones

Para la implementación de estas tres etapas se usan el código [7,4], este código es elegido para contrastar la eficiencia que presentan los dos tipos de codificadores. Para estos códigos se envían bloques de 100, 1000 y 10000 bits con una variación del ruido entre 1 y 20 dB.

4.2 IMPLEMENTACION DE CODIGO CICLICO Y CODIGO PRODUCTO CICLICO.

4.2.1 IMPLEMENTACION DE UN CODIGO CICLICO [7,4]

El codificador que se implementa se basa en el código cíclico [7,4], el cual presenta 7 bits de transmisión de los cuales cuatro son de mensaje y tres de redundancia. En esta codificación se envían bloques de 4 mensajes, por lo que la matriz resultante posee 16 bits de mensaje y 12 de paridad. Las figuras 4.2 y 4.3 muestran un ejemplo del proceso de codificación cíclica del código cíclico [7,4].

$$M = \begin{bmatrix} \overbrace{1 \quad 1 \quad 1 \quad 0}^{\text{mensaje}} \\ 0 \quad 0 \quad 1 \quad 1 \\ 1 \quad 0 \quad 1 \quad 1 \\ 1 \quad 1 \quad 1 \quad 1 \end{bmatrix}$$

Figura 4.2, Matriz mensaje.

$$M' = \begin{bmatrix} \overbrace{1 \quad 1 \quad 1 \quad 0}^{\text{mensaje}} \vdots \overbrace{1 \quad 0 \quad 0}^{\text{redundancia horizontal}} \\ 0 \quad 0 \quad 1 \quad 1 \vdots \quad 1 \quad 0 \quad 1 \\ 1 \quad 0 \quad 1 \quad 1 \vdots \quad 0 \quad 0 \quad 0 \\ 1 \quad 1 \quad 1 \quad 1 \vdots \quad 1 \quad 1 \quad 1 \end{bmatrix}$$

Figura 4.3, Matriz codificada verticalmente (matriz codificada con el proceso del codificador cíclico).

Posteriormente, estos bloques se envían por el canal en forma serial donde se le suma ruido y luego se decodifica utilizando el decodificador a través de tabla de sindrome, para finalizar se le quita la redundancia obteniéndose la matriz mensaje original.

En la figura 4.4 se muestra el proceso general del código [7,4]

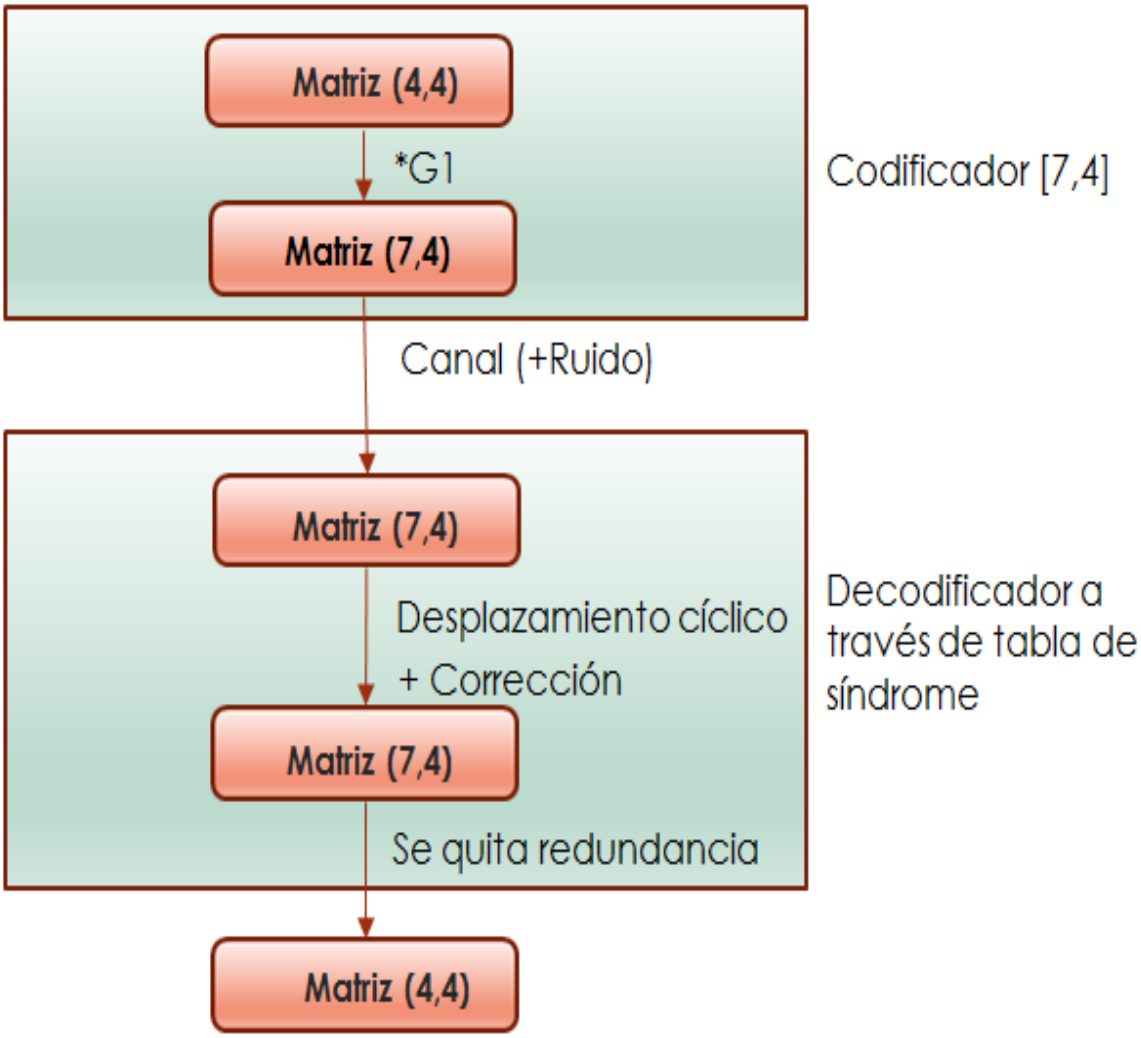

Figura 4.4, Diagrama general de implementación de código cíclico [7,4]

Este proceso tiene tres pasos a seguir, el primero de estos es la codificación donde una fila del bloque de mensajes es multiplicado por la matriz generadora G. La segunda etapa es donde se le suma el ruido clase A (modelo de Middleton) a cada bit para simular el medio ambiente industrial, la tercera y última etapa es la decodificación utilizando el algoritmo a través de tabla de síndrome donde se verifica si las palabras recibidas son palabras validas del código cíclico [7,4].

4.2.2 IMPLEMENTACIÓN DE UN CÓDIGO CÍCLICO PRODUCTO [7,4]

Este codificador se basa en dos códigos cíclicos [7,4], al igual que en el caso anterior se envían bloques de 4 mensajes, por lo que este bloque presenta, al igual que en el caso anterior, un mensaje de 16 bits pero como su proceso indica la multiplicación de dos códigos asociados, la redundancia de dicha matriz aumenta a 33. Las figuras 4.5, 4.6 y 4.7 muestran un ejemplo del proceso de codificación del código cíclico producto [7,4].

$$M = \begin{bmatrix} \overbrace{1 \;\; 1 \;\; 1 \;\; 0}^{\text{mensaje}} \\ 0 \;\; 0 \;\; 1 \;\; 1 \\ 1 \;\; 0 \;\; 1 \;\; 1 \\ 1 \;\; 1 \;\; 1 \;\; 1 \end{bmatrix}$$

Figura 4.5, Matriz mensaje.

$$M' = \left[\begin{array}{cccc:ccc} \overbrace{1 & 1 & 1 & 0}^{\text{mensaje}} & \vdots & \overbrace{1 & 0 & 0}^{\text{redundancia horizontal}} \\ 0 & 0 & 1 & 1 & \vdots & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & \vdots & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & \vdots & 1 & 1 & 1 \end{array}\right]$$

Figura 4.6, Matriz codificada horizontalmente.

$$M = \left[\begin{array}{cccc:ccc} \overbrace{1 & 1 & 1 & 0}^{\text{mensaje}} & \vdots & \overbrace{1 & 0 & 0}^{\text{redundancia horizontal}} \\ 0 & 0 & 1 & 1 & \vdots & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & \vdots & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & \vdots & 1 & 1 & 1 \\ \hdashline 0 & 1 & 1 & 0 & \vdots & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & \vdots & 0 & 1 & 0 \\ \underbrace{0 & 0 & 1 & 0}_{\text{redundancia vertical}} & \vdots & 1 & 1 & 0 \end{array}\right]$$

Figura 4.7, Matriz codificada verticalmente (matriz codificada con el proceso del codificador producto cíclico).

Estos bloques de matrices se envían por el canal sumándole el ruido, luego cada bloque se reagrupa en columnas y se analizan por el decodificador a través de tabla de síndrome, cuando este proceso termina se eliminan los bits de redundancia, que corresponden a las columnas de la matriz obtenida. Posteriormente se analizan las filas por el decodificador a través de tabla de síndrome, a la matriz obtenida de dicha operación se le quitan los bits de redundancia que esta vez corresponden a las filas. Para finalizar la matriz se traspone obteniéndose la matriz mensaje original. En la figura 4.8 se muestra el proceso del código producto [7,4].

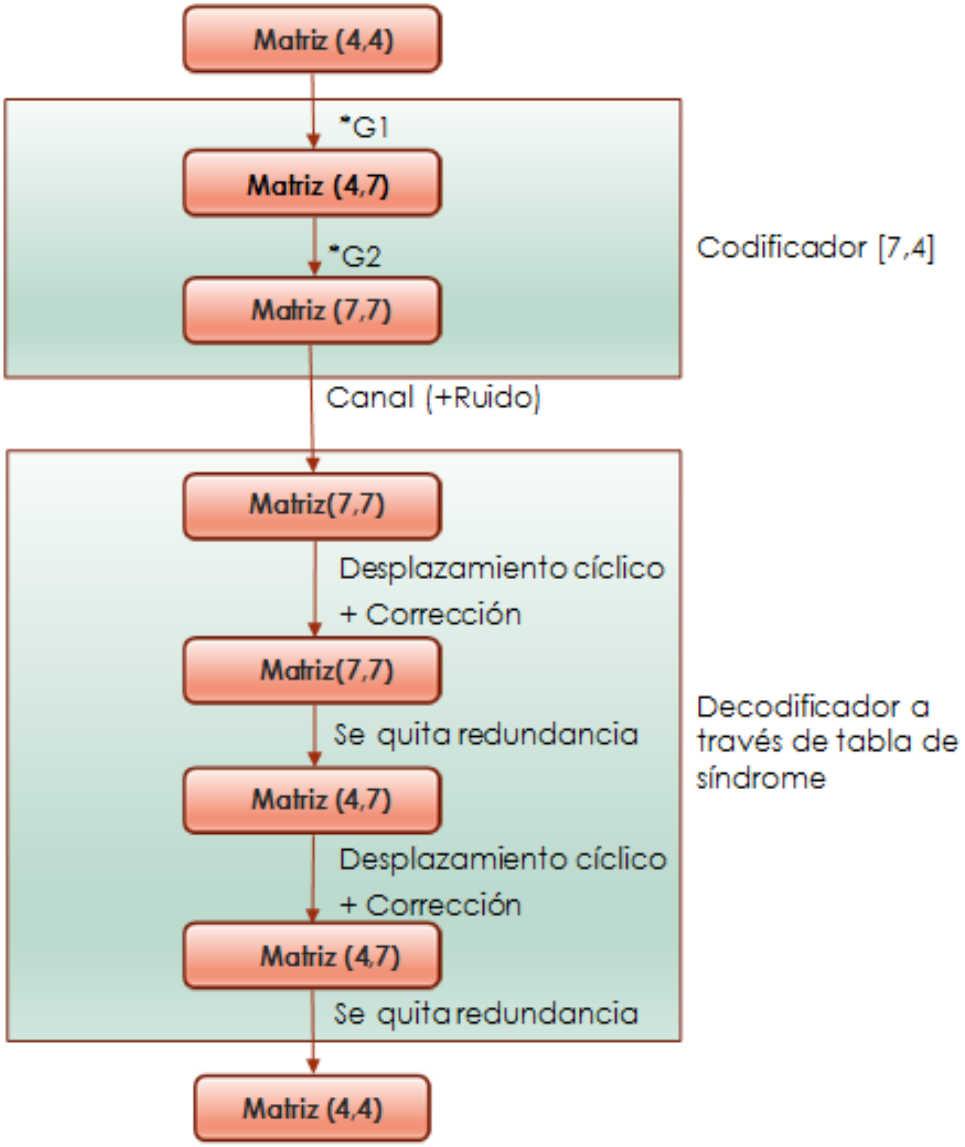

Figura 4.8, Diagrama general de implementación de código producto cíclico [7,4]

Este posee los mismo tres pasos que el código cíclico, en el proceso de la codificación el bloque de mensajes es multiplicado por la matriz generadora G, donde se obtienen la redundancia de las filas, luego el bloque obtenido es nuevamente multiplicado por G, donde esta vez se obtiene la redundancia de las columnas. Con esto se obtiene un bloque de mayor tamaño que el anterior, el cual posee una mayor cantidad de bits de redundancia. El segundo paso al igual que en el código anterior es la suma del ruido simulado a través de Middleton con el bloque de mensajes obtenido. Para finalizar este proceso se revisa si las columnas son palabras válidas del el código, luego se quitan los bits de redundancia de dichas columnas. Ahora se revisan si las filas son palabras validas del código y para finalizar se quita la redundancia de dichas filas, obteniéndose el mensaje final recibido.

4.3 ETAPA DE CODIFICACION

Como se describe anteriormente la primera etapa en esta implementación es la codificación, esta consiste principalmente en agregar a los bloques de mensajes originales a enviar los bits de redundancia para cada tipo de codificación, es decir, se convierten para adecuar el sistema de datos de destino. En esta tesis se utilizan dos tipos, la codificación cíclica y la codificación producto cíclica, la codificación cíclica se caracteriza por agregar bits de redundancia a las filas del bloque mensaje a enviar, en cambio la codificación producto cíclica está caracterizada por agregar bits de redundancia a las filas y a las columnas del bloque de mensaje a enviar.

La codificación cíclica está basada en agregar bits de redundancia ya sea solo en fila o en columnas por lo que basta solo multiplicar el bloque de mensaje por la matriz generadora G, en cambio la codificación de un código producto cíclico está basada en agregar bits de redundancia en filas y columnas, en este documento se decide que para que el bloque de mensaje sea cuadrado, este bloque sea multiplicado dos veces por la matriz generadora G.

4.4 ETAPA DE RUIDO

En esta etapa se implementa el ruido de clase A (modelo de Middleton), este ruido se suma a cada bit del mensaje que se asemeja al ruido del medio ambiente industrial.

Un ejemplo de este ruido se muestra en la figura 4.9, el cual tiene un $SNR = 10\ dB$ y se forman 1000 muestras.

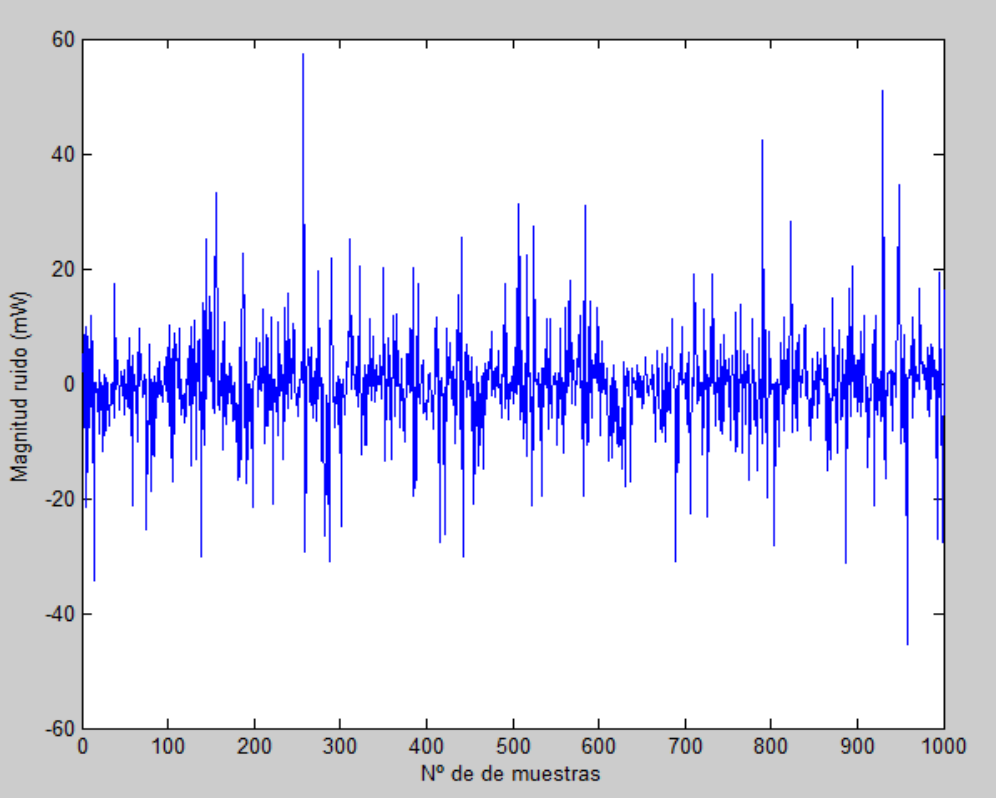

Figura 4.9, Ruido

De acuerdo a la gráfica se puede apreciar las características del ruido clase A, el cual posee un ruido impulsivo y un ruido Gaussiano, donde los peak que se producen en un corto periodo de tiempo es el ruido impulsivo y el ruido de fondo, el ruido Gaussiano, posee un valor mucho menor y más constante.

4.5 ETAPA DE DECODIFICACION

En esta etapa se verifica si los bloques de mensajes enviados a través del canal son palabras válidas para los diferentes tipos de códigos cíclicos. En el caso del código cíclico solo consta de una etapa en la cual se verifican los vectores filas de los bloques de mensajes a través del decodificador a través de tabla de síndrome, en cambio, para el código producto cíclico este proceso consta de dos etapas, en la primera se analiza el mensaje en vectores columnas, enviando cada columna al decodificador a través de tabla de síndrome, luego de pasar por este, se reagrupan los vectores recibidos. En la segunda etapa es donde se produce la decodificación a través de tabla de síndrome, por lo que en esta etapa se analiza si cada vector columna es palabra valida.

4.5.1 DECODIFICACION DE UN CODIGO CICLICO

En esta decodificación la matriz mensaje con ruido agregado se separa en vectores a través de sus filas, a cada una de estas filas se le calcula su respectivo síndrome, si este síndrome no coincide con alguno presente en la tabla de síndrome indica que el vector recibido no presenta errores, si coincide al vector se le suma un 1 en mod 2 en la posición donde la tabla indica está el error. Un ejemplo de la decodificación cíclica usando el método de decodificación por a través de tabla de síndrome se muestra en las figuras 4.10, 4.11 y 4.12.

$$\mathbf{M}=\begin{bmatrix} \overbrace{\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0}}^{mensaje} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

Figura 4.10, Matriz mensaje.

$$\mathbf{M}'=\left[\begin{array}{cccc:ccc} \overbrace{1 & 1 & 1 & 0}^{\text{mensaje}} & \vdots & \overbrace{1 & 0 & 0}^{\text{redundancia horizontal}} \\ 0 & 0 & 1 & 1 & \vdots & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & \vdots & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & \vdots & 1 & 1 & 1 \end{array}\right]$$

Figura 4.11, Matriz mensaje con ruido agregado.

$$M' = \begin{bmatrix} \overbrace{1 \quad 1 \quad 1 \quad 0}^{\text{mensaje}} \\ 0 \quad 0 \quad 1 \quad 1 \\ 1 \quad 0 \quad 1 \quad 1 \\ 1 \quad 1 \quad 1 \quad 1 \end{bmatrix}$$

Figura 4.12, Matriz decodificada mediante un decodificador cíclico.

En la figura 4.13 se muestra los pasos del decodificador a través de tabla de síndrome para un código cíclico [7,4].

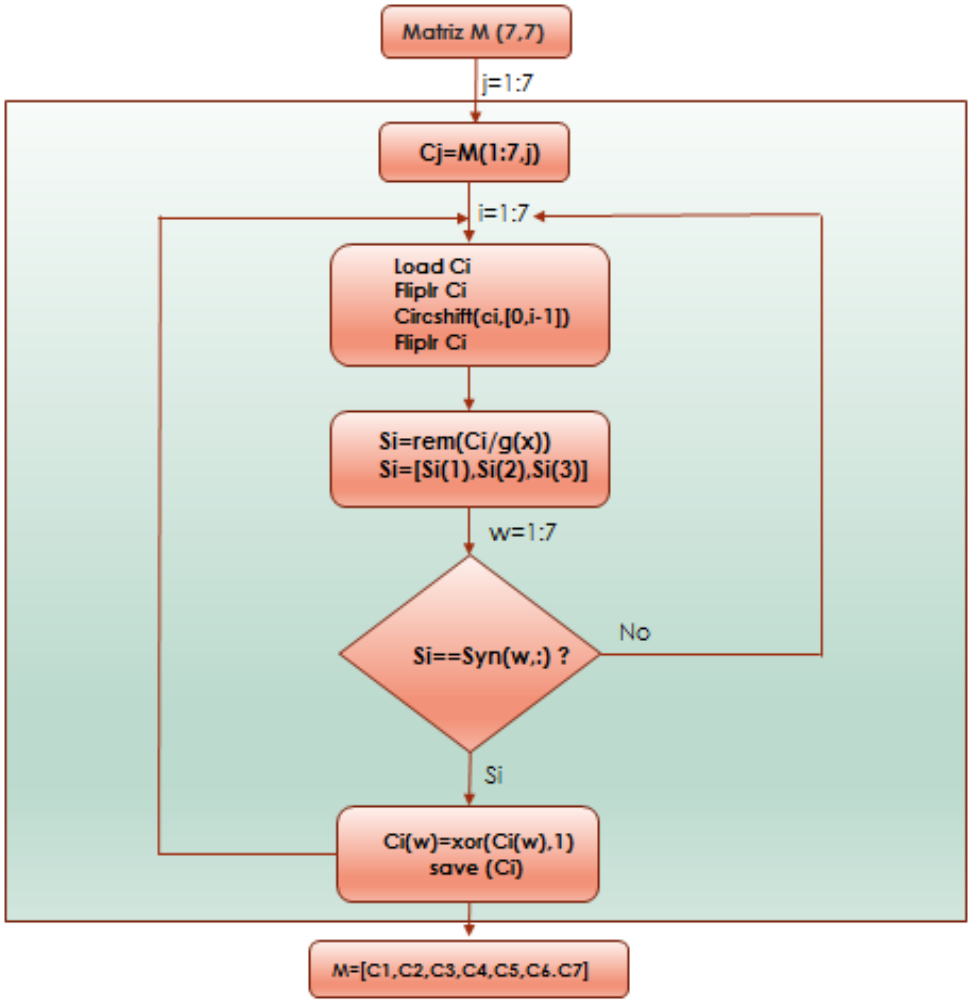

Figura 4.13, Esquema de decodificador a través de tabla de síndrome para código cíclico [7,4]

4.5.2 DECODIFICACION DE UN CODIGO PRODUCTO CICLICO

Este tipo decodificación, presenta cuatro etapas. La primera de estas es donde se reagrupan los vectores columnas de la matriz proveniente del canal y es enviada al decodificador a través de tabla de síndrome para verificar si es palabra valida. La segunda etapa es donde los vectores se analizan de la misma manera que en el código cíclico.

Al terminar este proceso, se quitan los bits de redundancia de los vectores columnas. Durante la tercera etapa estos vectores se reorganizan en filas y se envían para su análisis a través de tabla de síndrome, estos vectores se analizan de igual manera que las columnas. Para finalizar, quitando los bits de redundancias de dichas filas con lo que se obtiene la matriz mensaje original. Un ejemplo de la decodificación cíclica usando el método de decodificación por a través de tabla de síndrome se muestra en las figuras 4.14, 4.15, 4.16 y 4.17.

$$M=\begin{bmatrix} \overbrace{1 & 1 & 1 & 0}^{\text{mensaje}} \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Figura 4.14, Matriz mensaje.

$$M=\left[\begin{array}{cccc:ccc} \overbrace{1 & 1 & 1 & 0}^{\text{mensaje}} & & \overbrace{1 & 0 & 0}^{\text{redundancia horizontal}} \\ 0 & 0 & 1 & 1 & & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & & 1 & 1 & 1 \\ \hdashline 0 & 1 & 1 & 0 & & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & & 0 & 1 & 0 \\ \underbrace{0 & 0 & 1 & 0}_{\text{redundancia vertical}} & & 1 & 1 & 0 \end{array}\right]$$

Figura 4.15, Matriz mensaje con ruido agregado..

$$M' = \left[\begin{array}{\overbrace{cccc}^{\text{mensaje}} \vdots \overbrace{ccc}^{\text{redundancia horizontal}}} 1&1&1&0 \vdots 1&0&0 \\ 0&0&1&1 \vdots 1&0&1 \\ 1&0&1&1 \vdots 0&0&0 \\ 1&1&1&1 \vdots 1&1&1 \end{array}\right]$$

Figura 4.16, Matriz decodificada horizontalmente.

$$M' = \left[\begin{array}{\overbrace{cccc}^{\text{mensaje}}} 1&1&1&0 \\ 0&0&1&1 \\ 1&0&1&1 \\ 1&1&1&1 \end{array}\right]$$

Figura 4.17, Matriz decodificada mediante un decodificador producto cíclico.

En la figura 4.18 y 4.19 se muestran los pasos de este tipo de decodificación.

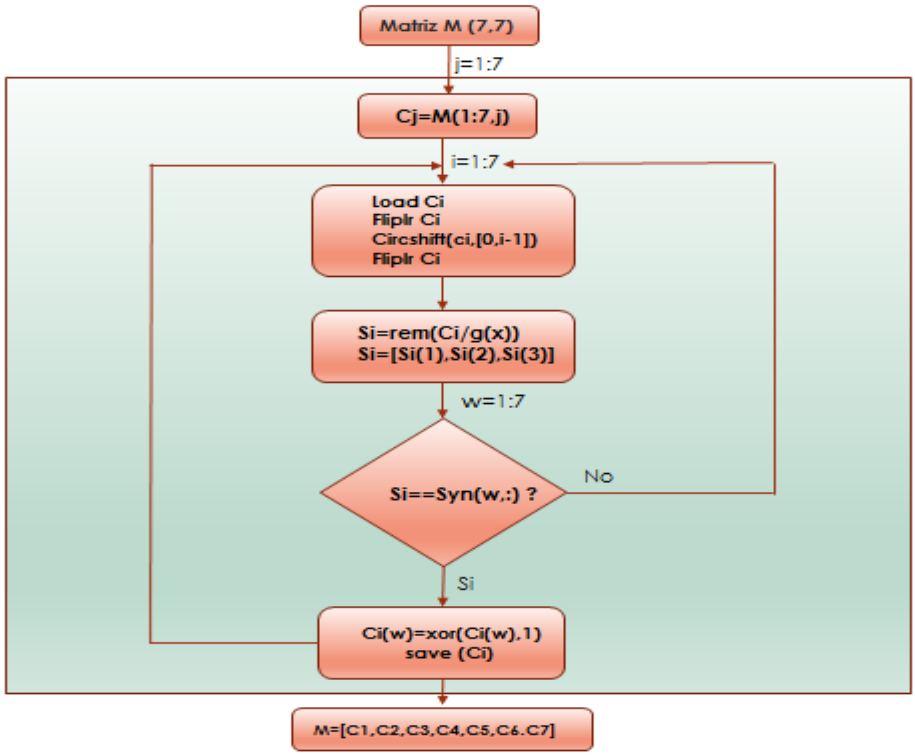

Figura 4.9, Esquema de la primera etapa de un decodificador a través de tabla de síndrome para el código producto cíclico [7,4].

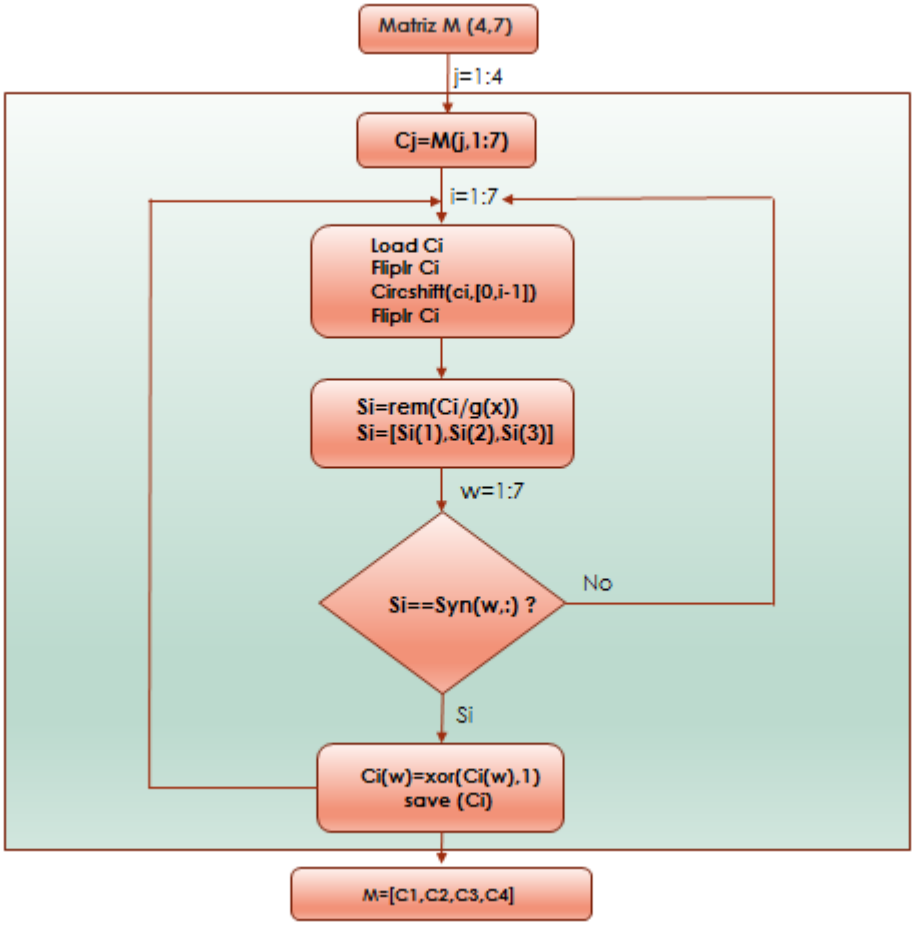

Figura 4.10, Esquema de la segunda etapa de un decodificador a través de tabla de síndrome para el código producto cíclico [7,4].

CAPÍTULO V

RESULTADOS DEL PROCESO DE CODIFICACION Y DECODIFICACION DE CODIGOS CICLICOS EN MEDIO AMBIENTE INDUSTRIAL.

5.1 CRITERIO PARA EL RUIDO DEL CODIGO CICLICO [7,4]

Como el ruido en el canal tiene un parámetro normal de 10 mW, se decide aplicar el criterio de que si el elemento de la matriz resultante posee un valor mayor a 10 mW es un 1 lógico, en el caso contrario es un 0 lógico. Para el código cíclico [7,4] y código producto cíclico [7,4] la cantidad de muestras tomadas es de 100. En la figura 5.1 se muestra un diagrama del criterio usado para el código cíclico [7,4]

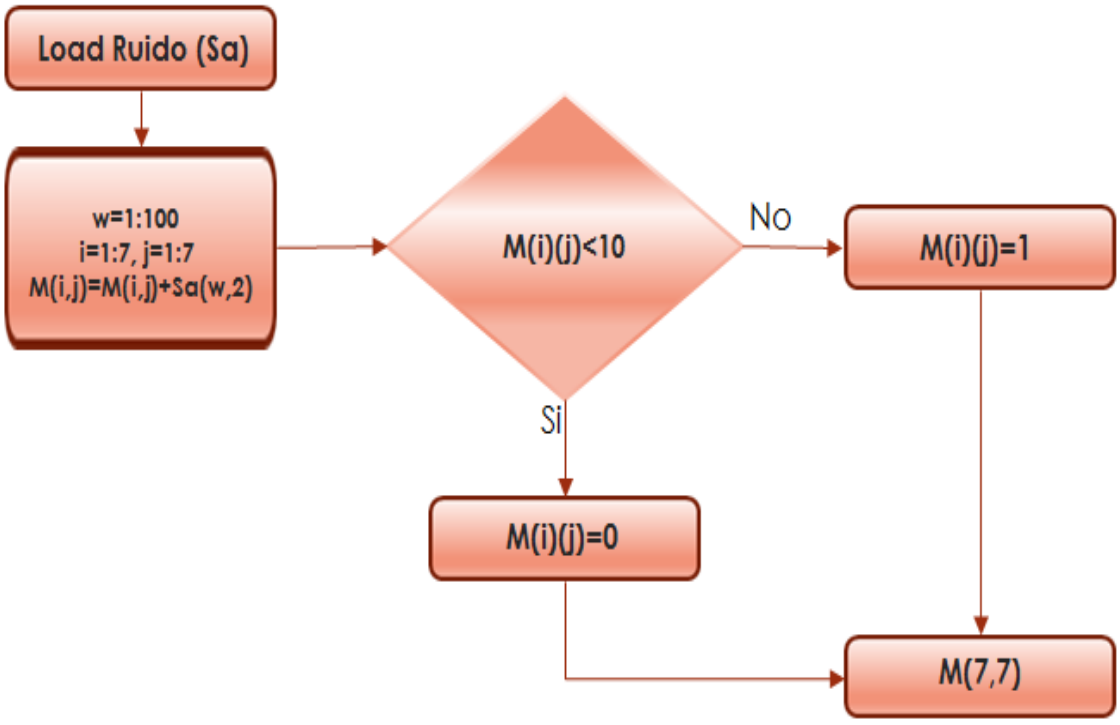

Figura 5.1, Esquema del criterio del ruido para canal de transmisión en código cíclico [7,4]

5.2 RESULTADOS DEL DESEMPEÑO DE CODIGOS CICLICO Y CODIGO CICLICO PRODUCTO EN MEDIO AMBIENTE INDUSTRIAL.

Para comparar el desempeño de los diferentes tipos de códigos cíclicos al introducirles ruido clase A (modelo de Middleton), con el cual se simula un medio ambiente industrial, y mediante la decodificación a través de tabla de síndrome, se envían bloques de 100, 1.000, 10.000 y 100.000 bits [esta variación de bits enviados se debe a que los gráficos se muestran en escala logarítmica].

Se analiza el desempeño del código cíclico [7,4]. La figura 5.2 muestra la comparación del desempeño del código cíclico con relación al tamaño de bloque de 100, 1000, 10000 y 100.000 bits.

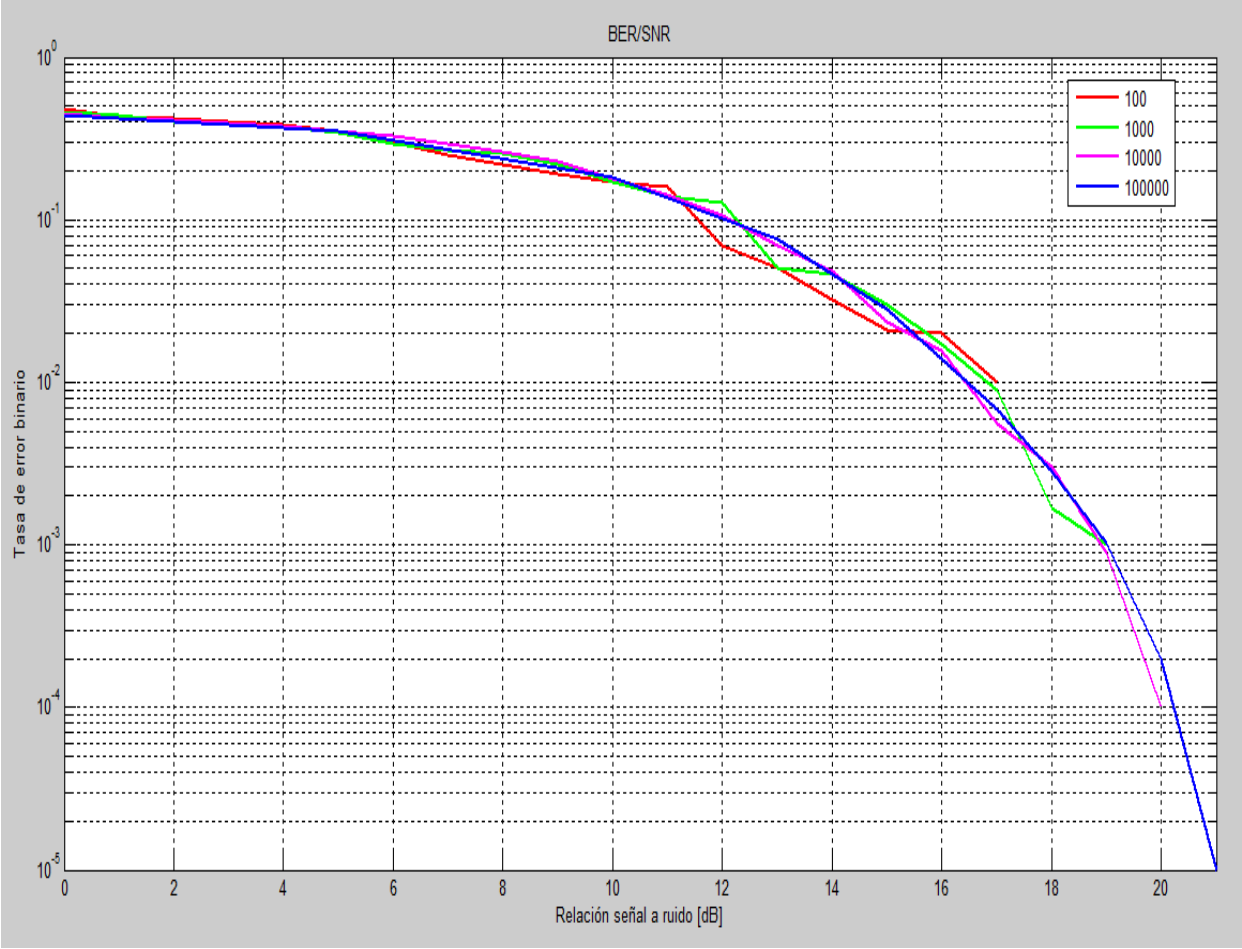

Figura 5.2, Comparación del desempeño de un sistema de comunicación enviando 100, 1.000, 10.000 y 100.000 bits que utiliza el formato del código cíclico [7,4].

La figura 5.3 muestra la comparación del desempeño del código cíclico producto con relación al tamaño de bloque de 100, 1000, 10000 y 100.000 bits.

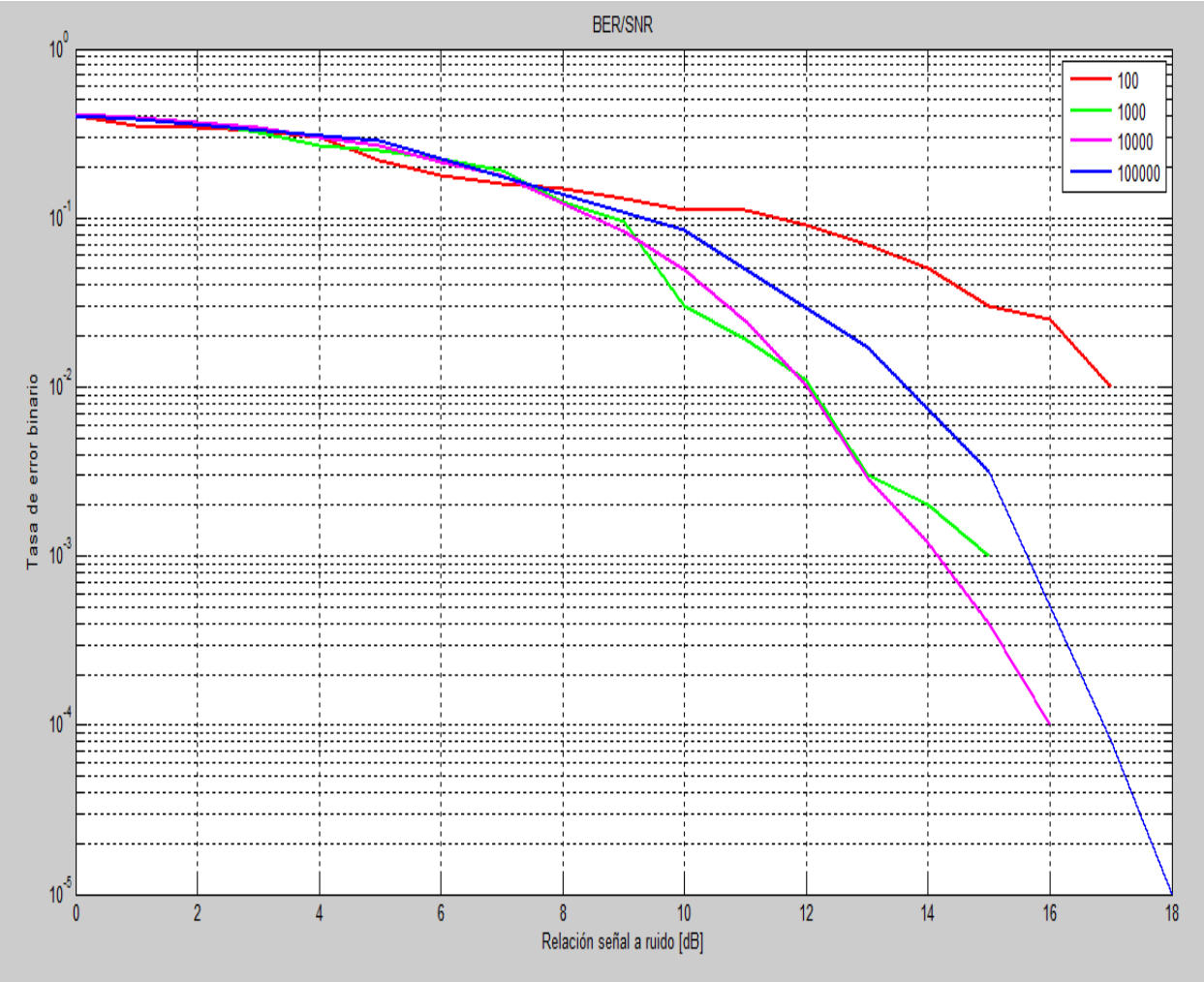

Figura 5.3, Comparación del desempeño de un sistema de comunicación enviando 100, 1.000, 10.000 y 100.000 bits que utiliza el formato del código producto cíclico [7,4].

La figura 5.4 muestra una comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 100 bits.

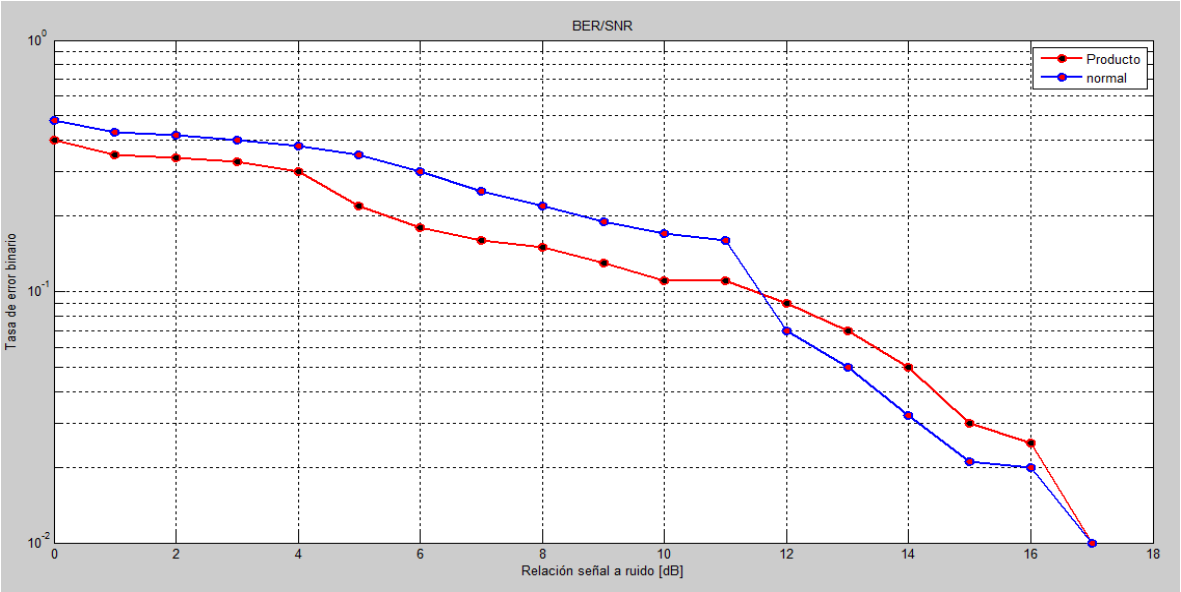

Figura 5.4, Comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 100 bits.

La figura 5.5 muestra una comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 1000 bits.

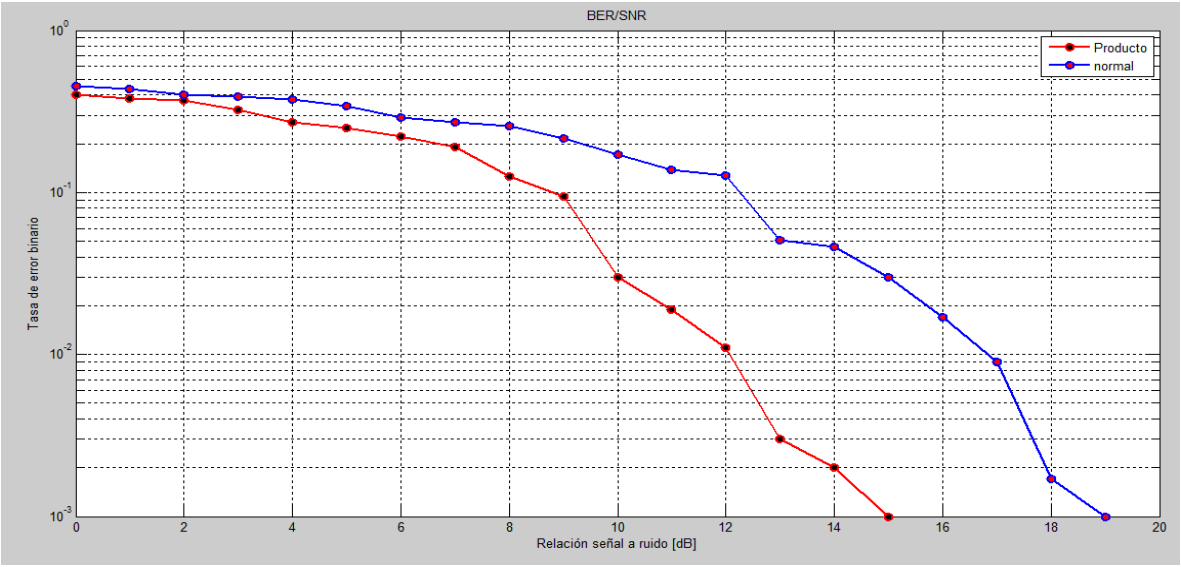

Figura 5.5, Comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 1000 bits.

La figura 5.6 muestra una comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 10.000 bits.

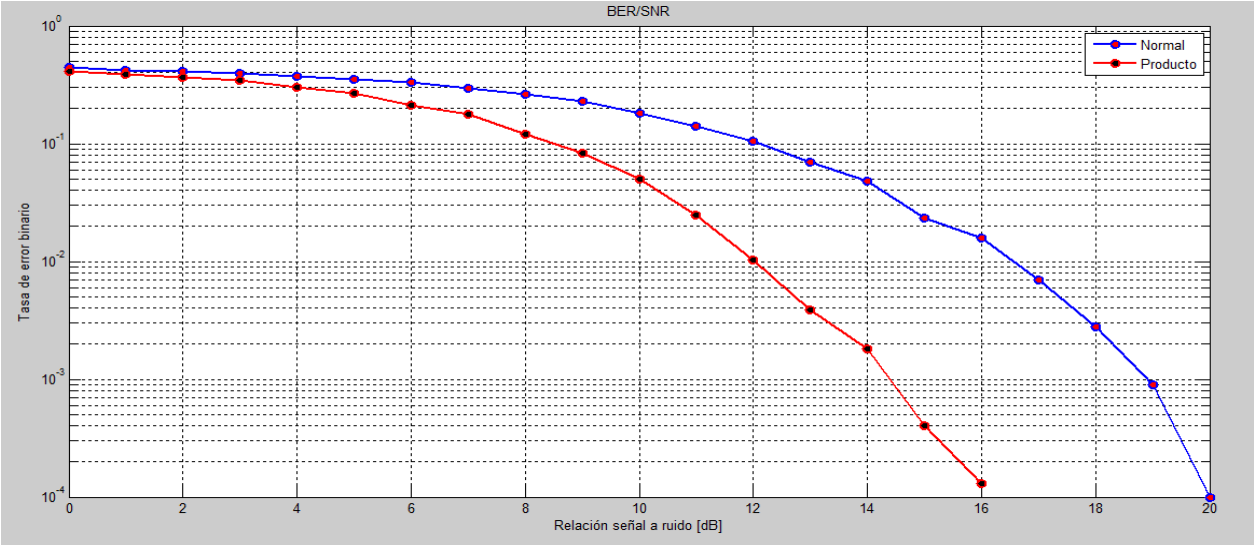

Figura 5.6, Comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 10.000 bits.

La figura 5.7 muestra una comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 100.000 bits.

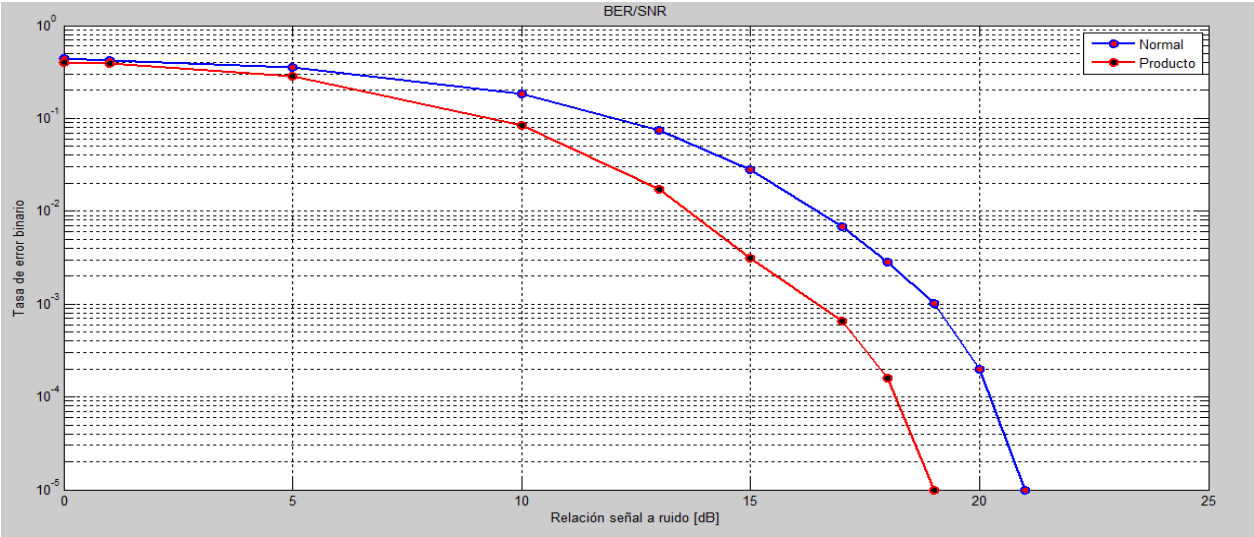

Figura 5.7, Comparación de desempeño entre el código cíclico y el código cíclico producto al enviar un bloque de código de 100.000 bits.

5.3 DISCUSIÓN DE RESULTADOS

De acuerdo a la figura 5.2 se puede apreciar un comportamiento uniforme y una gran similitud entre las curvas obtenidas a los diferentes valores de SNR para el código cíclico [7.4], esto se ve claramente para un SNR de 17 dB donde se intersectan las curvas de tamaño de bloque 1.000, 10.000 y 100.000.

La figura 5.3 muestra una gran similitud en los SNR bajo los 10 dB luego, para SNR mayores, aunque las pendientes de las curvas varían se puede apreciar una gran linealidad de las tasas de error binario obtenidas, cabe mencionar que las grandes diferencias se deben igualmente a la irregularidad que presenta el ruido impulsivo y que la curva más representativa es la de bloque de 100.000 bits ya que presenta la mayor cantidad de bits enviados.

Las figuras comparativas 5.4, 5.5, 5.6 y 5.7 muestran que la codificación cíclica producto presenta una notoria mayor eficiencia que la codificación cíclica, donde desde los SNR más bajos se aprecia una gran diferencia entre estas formas de codificación, esta diferencia se va incrementando a mayores SNR, hasta donde se puede apreciar una diferencia de 2 SNR, lo que implica que la potencia necesaria para la codificación cíclica producto es la cuarta parte de la potencia que necesita la codificación cíclica. De las gráficas también se puede extender es que al enviar 1.000.000 bits la eficiencia que presenta el código cíclico producto será mayor que la presente en el código cíclico.

CONCLUSIONES

Un codificador y decodificador de un código cíclico producto presenta un diseño muy similar al de un codificador y decodificador de un código cíclico, es decir, la lógica que ambos utilizan es muy similar por lo que al implementar un codificador y decodificador de un código cíclico se puede extender este de manera muy sencilla al de un codificador y decodificador de un código cíclico producto.

La eficiencia que se obtiene de los datos indica que el proceso de codificación y decodificación del código cíclico producto es más elevada que la presente en el código cíclico, esto se muestra en los valores que presentan cada uno para cada caso, el código [7,4] para código cíclico, tiene un valor de 0.5714, mientras que el mismo código [7,4] para código producto cíclico, tiene un valor de 0.3265, lo que deja en evidencia lo señalado con respecto al mejor comportamiento que presenta el código producto cíclico, ya que al ser menor el valor de su eficiencia mejor es su respuesta con respecto a la corrección de errores al momento de realizar la transmisión y posterior recuperación de datos a través del medio ambiente industrial que presenta ruido de clase A. Por lo que, los resultados que se obtienen concuerdan con la teoría referida a la eficiencia que presentan ambos códigos cíclicos.

Se concluye que el código producto cíclico es mejor tanto en la cantidad de errores que se transmiten y en la potencia necesaria para enviar la misma cantidad de bits, de lo primero se puede señalar que la curva de rendimiento (SNR/BER) del código producto cíclico está siempre por debajo de la curva del código cíclico lo que indica que a cualquier SNR este presenta un mejor desempeño y de lo segundo se puede decir que la potencia necesaria para enviar bits será menor para el código producto cíclico.

BIBLIORAFÍA

[1] PÉREZ CONSTANTINO, "sistemas de telecomunicación", editorial: Universidad de Cantabria, año 2007.

[2] D. MIDDLETON, "Canonical non gaussian models: their implications for measurements and prediction receiver performance". Editorial: IEEE transactions of comunications. Año: August 1979 Vol. COM-21 Nº3, pág.209-220.

[3] OLIVA NURIA, "Redes de comunicaciones industriales", Editorial: Universidad educacional de educación a distancia Madrid. Año: 2013.

[4] CUBERO MANUEL, "la televisión digital: fundamentos y teoría", editorial: S.A. marcombo, año 2009.

[5] ROBERT H. MORELOS-ZARAGOZA, "The Art of Error Correcting Coding", editorial: Wiley, año: 2002, pág.39-52

[6] IÑIGO JORDI, BARCELO JOSÉ, CERDA LLORENÇ, "estructura de redes de computadores", editorial: UOC (universitat oberta de Catalunya), año 2012.

[7] J. E. MEGGITT, "error correcting codes for correcting burst error" editorial: IBM J. res. develop. 4, año: 1960, pág. 329-343

[8] H. BURTON, E. WELDON, "cyclic product codes" editorial: IEE transacción on information theory, año: 1965, pág. 433-439

[9] PETERSON, W. W., "Error-Correcting Codes", editorial: Cambridge, Mass.: M.I.T. Press, año: 1961.

ANEXO A

Codificación de código cíclico [7,4]

% Se crea una matriz de (10000, 10000) compuesta de 1 y 0, luego esta matriz es reducida a (10000,4).

```
m=randi(10000,10000);

for i=1: size(m,1)

[fila,columna]=find(rem(m(i,:),2)==0);

m(i,columna)=0;

end

for i=1: size(m,1)

[fila,columna]=find(rem(m(i,:),2)==1);

m(i,columna)=1;

end

m=m(1:10000,1:4);
```

% Se crea la matriz G, y se crean los bloques de mensajes de (4,4) a partir de la matriz anterior.

```
a=0;

b=1;

G=[1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 1 1 0; 0 0 0 1 0 1 1];

for i=0:4:2496

a=a+4;

R=m(b+i:a,1:4);

save ('matrizMinicial.mat','R')
```

% Se inicia el proceso de codificación, donde se multiplica el bloque de mensaje por G, y luego la matriz resultante es convertida en mod 2.

```
K=R*G;

K=K';

for i=1: size(K,1)

[fila,columna]=find(rem(K(i,:),2)==0);

K(i,columna)=0;

end

for i=1: size(K,1)

[fila,columna]=find(rem(K(i,:),2)==1);

K(i,columna)=1;

end
```

ANEXO B

Codificación de código cíclico producto [7,4]

% Se crea una matriz de (10000, 10000) compuesta de 1 y 0, luego esta matriz es reducida a (10000,4).

```
m=randi(10000,10000);

for i=1: size(m,1)

[fila,columna]=find(rem(m(i,:),2)==0);

m(i,columna)=0;

end

for i=1: size(m,1)

[fila,columna]=find(rem(m(i,:),2)==1);

m(i,columna)=1;

end

m=m(1:10000,1:4);
```

% Se crea la matriz G, y se crean los bloques de mensajes de (4,4) a partir de la matriz anterior.

```
a=0;

b=1;

G=[1 0 0 0 1 0 1; 0 1 0 0 1 1 1; 0 0 1 0 1 1 0; 0 0 0 1 0 1 1];

for i=0:4:2496

a=a+4;

R=m(b+i:a,1:4);

save ('matrizMinicial.mat','R')
```

% Se inicia el proceso de codificación, donde se multiplica el bloque de mensaje por G, y luego la matriz resultante es convertida en mod 2, ya con la matriz convertida se procede a multiplicar nuevamente por G, entonces es convertida en mod 2.

```
K=R*G;
for i=1: size(K,1)
[fila,columna]=find(rem(K(i,:),2)==0);
K(i,columna)=0;
end
for i=1: size(K,1)
[fila,columna]=find(rem(K(i,:),2)==1);
K(i,columna)=1;
end
K=K';
M=K*G;
for i=1: size(M,1)
[fila,columna]=find(rem(M(i,:),2)==0);
M(i,columna)=0;
end
for i=1: size(M,1)
[fila,columna]=find(rem(M(i,:),2)==1);
M(i,columna)=1;
end
```

ANEXO C

Ruido código cíclico [7,4]

% Se estandarizara la potencia del canal de transmisión en 10 mW y -10 mW.

```
M=K*10;
for i=1: size(M,1)
for j=1:size(M,1)
if M(i,j)==0
M(i,j)=-10;
end
end
end
load 'Ruido0SNR.mat'
Matrizruidonormal(Sa,K);
load 'Matrizruido.mat'
```

% Se analiza la matriz V3, si el valor de la componente (i, j) es mayor a 10, la componente es cambiada por un 1, en el caso de que la componente sea menor es cambiada por un 0.

```
for i=1:7
for j=1:4
if V3(i,j)>10
V3(i,j)=1;
else  V3(i,j)=0;
```

```
end

end

end


% Se Inicia la etapa de decodificación


decomeggitnormal(V3);

% Se crea y se ajustan los parámetros del Modelo de ruido de Middleton el cual simula el ruido impulsivo y el gaussiano (Ruido0SNR)


n = 1000;

A = 0.01;

T = 0.0001;

x = zeros(n,1);

y = zeros (n,1);

SNR= 10;

save('SNRdato.mat','SNR')

snr = 10^(SNR/10);

n0 = sqrt(1/snr);

n1= (((1/A) + T)/ (1+T))*n0;

for i= 1:1000;

x1 = rand;

x2 = rand;

ng = n0*sqrt (-2*log10(x1)*cos(2*pi*x2));

ni = n1*sqrt (-2*log10(x1)*cos(2*pi*x2));

x(i,1) = ng*ni;
```

```
y(i,1) = i;

Sa=[y,x];

end

save ('Ruido0SNR.mat','Sa')

plot(y(:,1),x(:,1))

ylabel('Magnitud ruido (mW)')

xlabel('Nº de de muestras')

end
```

% A la matriz que es enviada por el canal se le suma el ruido, esto se hace sumando a algunas filas distintas muestras y luego se procede a hacer lo mismo con las columnas (Matrizruidonormal(Sa,K))

```
i=1;

for w=1:100

for j=1:4

V3(i,j)=M(i,j)+Sa(w,2);

end

end

i=2;

for w=1:90

for j=1:4

V3(i,j)=M(i,j)+Sa(w,2);

end

end

i=3;
```

```
for w=1:50
for j=1:4
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=4;
for w=1:30
for j=1:4
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=5;
for w=1:57
for j=1:4
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=6;
for w=1:53
for j=1:4
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=7;
for w=1:20
```

```
for j=1:4
V3(i,j)=M(i,j)+Sa(w,2);
end
end
j=2;
for w=1:26
for i=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
j=3;
for w=1:27
for i=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
save ('Matrizruido.mat','V3')
```

ANEXO D

Ruido código cíclico producto [7,4]

% Se estandarizara la potencia del canal de transmisión en 10 mW y -10 mW.

```
M=10*M;
for i=1: size(M,1)
for j=1:size(M,1)
if M(i,j)==0
M(i,j)=-10;
end
end
end
load 'Ruido0SNR.mat'
Matrizruido(Sa,M);
load 'Matrizruido.mat'
```

% Se analiza la matriz V3, si el valor de la componente (i, j) es mayor a 10, la componente es cambiada por un 1, en el caso de que la componente sea menor es cambiada por un 0.

```
for i=1:7
for j=1:7
if V3(i,j)>0
V4(i,j)=1;
else  V4(i,j)=0;
```

```
end
end
end
```

% Se Inicia la etapa de decodificación

```
decomeggit1(V4);
```

% A la matriz que es enviada por el canal se le suma el ruido, esto se hace sumando a algunas filas distintas muestras y luego se procede a hacer lo mismo con las columnas

```
i=1;
for w=1:100
for j=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=2;
for w=1:90
for j=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=3;
for w=1:50
for j=1:7
```

```
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=4;
for w=1:30
for j=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=5;
for w=1:57
for j=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=6;
for w=1:53
for j=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
i=7;
for w=1:20
for j=1:7
V3(i,j)=M(i,j)+Sa(w,2);
```

```
end
end
j=2;
for w=1:26
for i=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
j=3;
for w=1:27
for i=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
j=6;
for w=1:6
for i=1:7
V3(i,j)=M(i,j)+Sa(w,2);
end
end
save ('Matrizruido.mat','V3')
end
```

ANEXO E

Primera etapa de decodificación del código cíclico

```
% Se recibe la matriz enviada por el canal, y luego esta es definida a través de sus columnas.
C1=M(1:7,1);
C2=M(1:7,2);
C3=M(1:7,3);
C4=M(1:7,4);
save ('matrizF1.mat','C1')
save ('matrizF2.mat','C2')
save ('matrizF3.mat','C3')
save ('matrizF4.mat','C4')
% Se envían las columnas al decodificador a través de tabla de síndrome
for i=1:4
a=i;
M(1:7,i:i)=M(1:7,i:i);
if a==1
deco1(M(1:7,i:i),a,H,C1);
elseif a==2
deco2(M(1:7,i:i),a,H,C2);
elseif a==3
deco3(M(1:7,i:i),a,H,C3);
else
deco4(M(1:7,i:i),a,H,C4);
end
```

```
end

% Se recibe el bloque de mensaje decodificado a través de a través de tabla de síndrome y se le quitan los bits de redundancia

matrizdecomeggitt1normal( M );

load 'matrizdecomeggittfinal';

Q=A;

Q=Q(1:4,1:4);

Q=Q';

% Se compara el bloque de mensaje original con el bloque de mensajes decodificado

load 'matrizMinicial.mat'

error=0;

for i=1:4

for j=1:4

if R(i,j)~=Q(i,j)

error=error+1;

end

end

end

save ('error.mat','error')

 % Se reagrupa cada columna verificada por el decodificador Meggitt

load 'Vectordeco1.mat';

load 'Vectordeco2.mat';

load 'Vectordeco3.mat';

load 'Vectordeco4.mat';

M=[C1,C2,C3,C4];

A=M;
```

```
save ('matrizdecomeggittfinal.mat','A')
```

ANEXO F

Primera etapa de decodificación del código cíclico producto

```
% Se recibe la matriz enviada por el canal, y luego esta es definida a través de sus columnas.

C1=M(1:7,1);
C2=M(1:7,2);
C3=M(1:7,3);
C4=M(1:7,4);
C5=M(1:7,5);
C6=M(1:7,6);
C7=M(1:7,7);
save ('matrizF1.mat','C1')
save ('matrizF2.mat','C2')
save ('matrizF3.mat','C3')
save ('matrizF4.mat','C4')
save ('matrizF5.mat','C5')
save ('matrizF6.mat','C6')
save ('matrizF7.mat','C7')
```

```
% Se envían las columnas al decodificador a través de tabla de síndrome


for i=1:7
a=i;
M(1:7,i:i)=M(1:7,i:i);
if a==1
deco1(M(1:7,i:i),a,H,C1);
elseif a==2
deco2(M(1:7,i:i),a,H,C2);
elseif a==3
deco3(M(1:7,i:i),a,H,C3);
elseif a==4
deco4(M(1:7,i:i),a,H,C4);
elseif a==5
deco5(M(1:7,i:i),a,H,C5);
elseif a==6
deco6(M(1:7,i:i),a,H,C6);
else
deco7(M(1:7,i:i),a,H,C7);
end
end
```

% Se recibe el bloque de mensaje decodificado a través de tabla de sindrome y se le quitan los bits de redundancia. Este bloque de mensajes es descompuesto en filas para su posterior revisión.

```
matrizdecomeggitt1( M );
load 'matrizdecomeggittfinal';
A=A(1:4,1:7);
A1=A(1,1:7);
A2=A(2,1:7);
A3=A(3,1:7);
A4=A(4,1:7);
 % Se envían las filas al decodificador a través de tabla de síndrome

for i=1:4
a=i;
M(1:7,i:i)=M(1:7,i:i);
if a==1
deco78(A(i:i,1:7),a,H,A1);
elseif a==2
deco22(A(i:i,1:7),a,H,A2);
elseif a==3
deco23(A(i:i,1:7),a,H,A3);
else
deco24(A(i:i,1:7),a,H,A4);
end
end
```

% Se recibe el bloque de mensaje decodificado a través de tabla de síndrome y se le quitan los bits de redundancia.

```
matrizdecomeggitt2(A);

load 'matrizdecomeggittfinal2';

Q=Q(1:4,1:4);

Q=Q';
```

% Se compara el bloque de mensaje original con el bloque de mensajes decodificado

```
load 'matrizMinicial.mat'

error=0;

for i=1:4

for j=1:4

if R(i,j)~=Q(i,j)

error=error+1;

save ('error.mat','error')

end

end

end

save ('error.mat','error')
```

% Se reagrupa cada columna verificada por el decodificador de tabla de sindrome (matrizdecomeggitt1)

```
load 'Vectordeco1.mat';
load 'Vectordeco2.mat';
load 'Vectordeco3.mat';
load 'Vectordeco4.mat';
load 'Vectordeco5.mat';
load 'Vectordeco6.mat';
load 'Vectordeco7.mat';
M=[C1,C2,C3,C4,C5,C6,C7];
A=M;
save ('matrizdecomeggittfinal.mat','A')

% Se reagrupa cada fila verificada por el decodificador Meggitt  (matrizdecomeggitt2)

load 'Vectordeco21.mat';
load 'Vectordeco22.mat';
load 'Vectordeco23.mat';
load 'Vectordeco24.mat';
A=[A1,A2,A3,A4];
Q=A';
save ('matrizdecomeggittfinal2.mat','Q')
```

ANEXO G

Decodificador a través de tabla de síndrome para códigos cíclicos [7,4]

% Se ingresa cada vector y es transformada a su respectiva forma de polinomio, también se crea el polinomio característico

```
syms x;

g=[1 0 1 1];

poly2sym(g);

save ('matrizmeggitt.mat','M');
```

% Se inicia el proceso de desplazamiento de cada bit del vector recibido, y se calcula el síndrome de dicho vector

```
load 'matrizmeggitt.mat';

M = fliplr(M);

m11 = circshift(M,T-1);

m11 = fliplr(m11);

poly2sym(m11);

S1=rem(poly2sym(m11),poly2sym(g));

E=2*x^2+2*x+2;

Q=[0 0 0];

B=[0 0 0];

S1=S1+E;

S1 = sym2poly(S1);

S1=[S1 Q];

for j=1:3
```

```
if rem(S1(j),2)==0
S1(j)=0;
save ('matrizmeggittsyndrome.mat','S1');
end
end
for i=1:3
if rem(S1(i),2)==1
S1(i)=1;
save ('matrizmeggittsyndrome.mat','S1');
end
end
S1 = abs(S1);
D=[S1 B];
S1(1)=xor(B(1),D(1));
S1(2)=xor(B(2),D(2));
S1(3)=xor(B(3),D(3));
S1=[S1(1),S1(2),S1(3)];
S1=fliplr(S1);

% Se Compara el síndrome del vector recibido con la tabla del síndrome.
for w=1:7
if S1==Syn(w,:)
m11(w)=xor(m11(w),1);
M=m11;
M = circshift(M,-T+1);
```

```
save ('matrizmeggitt.mat','M');
break
end
end

% Tabla del síndrome
 Syn=[1 0 1
      1 1 1
      0 1 1
      1 1 0
      0 0 1
      0 1 0
      1 0 0];
```