Comunicaciones Aeronáuticas I

Sistemas de control de errores en comunicaciones digitales

Ben Thomas
Universidad Politécnica de Cataluña

Índice

1. Introducción

Los sistemas de comunicaciones digitales son ampliamente usados en la actualidad gracias a las numerosas ventajas que comportan, como su capacidad a la hora del almacenamiento, procesado, flexibilidad i velocidad. Tanto audio, como fotos i vídeos se digitalizan en una gran variedad de sistemas, con el fin de optimizar frente a la ideologia analógica.

Cualquier sistema de comunicaciones se puede modelar con 3 partes (Figura 1): El transmisor (aquel que desea mandar información), el canal (medio usado para transmitir) y el receptor (aquel que desea recibir la información). Uno de los problemas que surge es la distorsión de la señal que se produce mediante ruido generado en el canal, de manera que el mensaje final puede ser distinto al original, provocando lecturas erróneas.

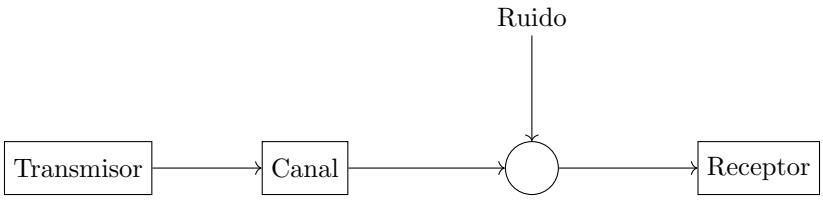

Figura 1: Esquema del Sistema de Comunicaciones

A propósito de mejorar la calidad de aquello digitalizado, se han desarrollado, al largo de los años, algoritmos que permiten detectar i corregir errores en la medida de lo posible.

2. Probabilidades de error

La probabilidad de error de un bit en un sistema de comunicaciones varia en función de la modulación, viene dado por diferentes combinaciones de funciones con la distribución normal. La característica que comparten todas es que dependen de la relación señal a ruido, determinada por la energía de cada bit E_b y la energía asociada al ruido N_0.

$$SNR = \left(\frac{E_b}{N_0}\right) \tag{1}$$

Teniendo en cuenta que $P\{A\} + P\{\bar{A}\} = 1$, se calcula la probabilidad de que ninguno de los n bits enviados en el paquete sea incorrecto y se obtiene que la probabilidad de error en un paquete de bits viene dada por:

$$P_p = 1 - (1 - P_e)^n \tag{2}$$

Utilizando valores para diferentes modulaciones y valores de $\frac{E_b}{N_0}$ se obtienen las probabilidades de error en el paquete resumidas en la Figura 2 [4]:

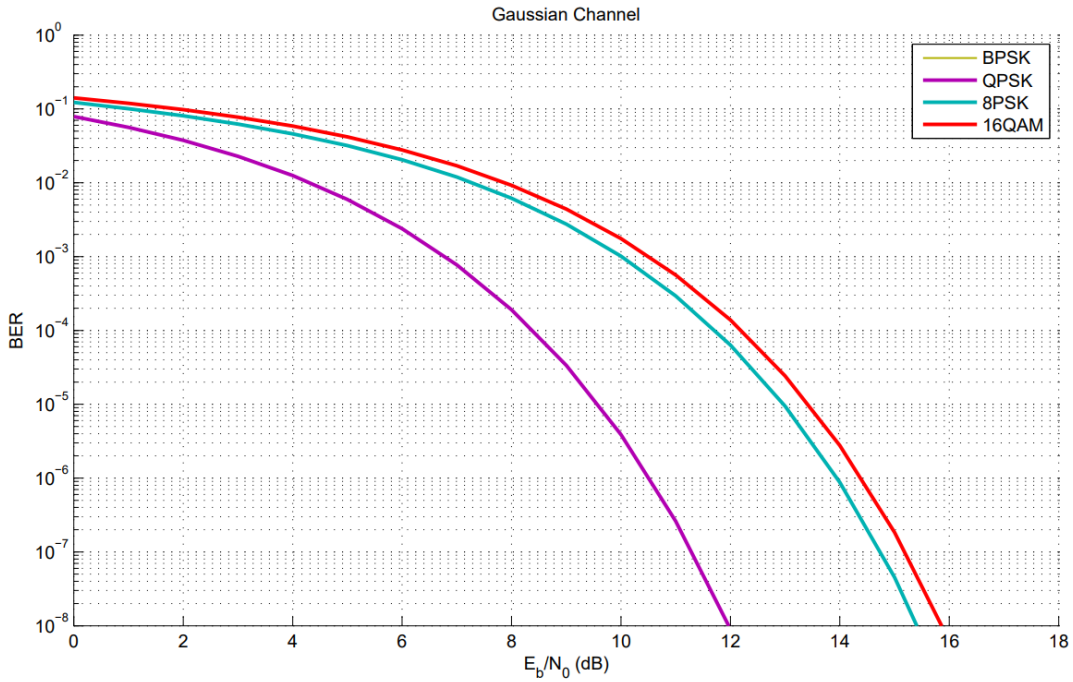

Figura 2: Bit Error Rate vs $\frac{E_b}{N_0}$

Al parecer, la probabilidad de error es muy pequeña y menospreciable, pero si se considera la cantidad de bits que se envían en las tecnologías modernas se puede ver la importancia de ellos. Usando la Ecuación 2, con diferentes valores de n y de P_e (teniendo en cuenta que un byte equivale a 8 bits) se puede cuantificar el caso:

- Información enviada: 1 Mega-byte $\implies$ $n = 8 \cdot 10^6$, $P_e = 10^{-6}$, $P_p = 0.99966$
- Información enviada 1 Giga-Byte $\implies$ $n = 8 \cdot 10^9$, $P_e = 10^{-9}$ $P_p = 0.99966$

Como se puede observar, aunque las probabilidades de error sean muy bajas, la probabilidad de que el paquete llegue sin ningún daño también lo es, de manera que no se pueden menospreciar los errores y es conveniente tener un método para corregirlos.

3. Ideas Iniciales

3.1. Definiciones

Un código $\mathcal{C}$ de longitud n sobre un alfabeto F es un subconjunto de F^n. Los code words son los elementos de $\mathcal{C}$. Se puede imaginar al código siendo todas las palabras válidas de un idioma (F), y al code word como una palabra en particular.

Sea un código $\mathcal{C} = \{1100, 1010, 1000\}$ en alfabeto binario $B = \{0, 1\}$ ($\mathcal{C}$ pertenece a B^4), los code words serian 1100, 1010, 1000. Supongamos que se recibe 1101: se define la distancia de Hamming como el número minimo de bits a cambiar hasta tener un code word valido, en éste caso 1.

3.2. Códigos de Repetición

Como primera solución al problema de los errores, se pensó en emplear un algoritmo basado en la repetición. En éste, cada bit del paquete se envía n veces, de manera que solo existen 2 code words (n zeros o n unos). Se asume que la probabilidad de error en un bit es menor a la probabilidad contraria, sino se introduciria más error del que había en primer lugar.

1	0	1	1	0

Información a enviar

111	000	111	111	000

Código a enviar (repetición con $n = 3$)

Dado un mensaje, el receptor calcula la distancia de Hamming mínima de la señal recibida y decide a qué codeword se corresponde (0 o 1). De manera que la capacidad de corrección es solamente de $\frac{n-1}{2}$ errores por code word.

101	100	100	111	100

Código recibido (con errores)

1	0	0	1	0

Información Recibida

Se podría pensar que la solución para mejorar el algoritmo repetitivo es simple, ya que los errores que se corrigen incrementan linealmente con n, de manera que si se escoge un valor alto se puede tener una alta calidad. La velocidad a la cual se reciben los bits no cambia, sin embargo la velocidad a la que se recibe la *información* sí (también linealmente con n). Se introduce aquí el concepto de redundancia, y se define al bit redundante como aquel que no aporta información nueva, cumple la función de control de errores. El propósito final consta de la optimización en los dos aspectos.

3.3. Bit de Paridad

Para la disminución de redundancia, se pensó en añadir un solo bit a cada código que indica la paridad o no paridad de la cantidad de unos que contiene (0 para número par y 1 para impar). Éste se suele añadir al principio o al final, de manera que si ha habido un error el bit de paridad no coincidirá con la información y el usuario sabrá que la información recibida no es correcta.

Surgen errores relevantes con este método. No solamente fallan al haber un numero par de errores sino que no pueden corregirlos. También se da el caso que el mismo bit de paridad puede fallar. Por esas razones, se suelen utilizar en códigos cortos, de 8 bits, donde la probabilidad de error es lo suficientemente baja para poder menospreciar la posibilidad de que más de un bit sea incorrecto.

4. Códigos de Hamming

A finales de los años 40, Richard Hamming sufre del problema de los errores en la transmisión de información de manera digital en su trabajo y decide implementar un código para arreglarlo. Quería saber dónde se producía el error (no solamente si había uno o no) y para ello usa el concepto del bit de paridad.

4.1. Funcionamiento

La primer idea surge de utilizar más de un bit de paridad. Hamming averigua que se podrían usar dos, cada uno dando información sobre un subconjunto, uno para los bits en posiciones pares y otro para los bits en posiciones impares. Si uno de los bits de paridad falla y el otro no, se puede descartar uno de los subconjuntos como correcto y buscar el error entre los bits del otro.

Dándose cuenta de que el método anterior ya es capaz de indicar en que zona está el error, Hamming supo que con más bits de paridad se podrían definir más subconjuntos (que podrían ser descartados en la búsqueda) e investigó cual era la redundancia mínima posible que podía tener el sistema para detectar la posición exacta de un bit erróneo. Dicho de otra manera, cual es el mínimo numero de preguntas dicotómicas que se pueden hacer para averiguar la posición de algo que se esconde en un mensaje de longitud l. Para resolver la pregunta, dibuja el mensaje como una tabla de $a \times b$ (considera longitudes $l = a \times b$, $\forall a, b \in \mathbb{N}$, con un matiz discutido en el Apartado 4.2).

Como ejemplo, se cogerá $a = b = 4$. La información de un mensaje de largo $l = 16$:

1 0	**0** 1	**1** 2	**1** 3	**1** 4	**1** 5	**0** 6	**0** 7	**0** 8	**0** 9	**1** 10	**0** 11	**0** 12	**1** 13	**1** 14	**1** 15

queda de la siguiente manera:

1 0	**0** 1	**1** 2	**1** 3
1 4	**1** 5	**0** 6	**0** 7
0 8	**0** 9	**1** 10	**0** 11
0 12	**1** 13	**1** 14	**1** 15

Primero, se averigua en qué columna se encuentra el posible error. Para ello, mediante un bit de paridad en la posición 1, se controla el subconjunto de bits en posiciones impares como se ve en la Figura 3 (columnas 2 y 4). Posteriormente, se controlan los de las columnas 3 y 4 con el bit redundante en la posición 2, (Figura 4). Un ejemplo de su funcionamiento es el siguiente: Si la paridad indicada por el bit de la posición 1 coincide con los datos recibidos, las columnas 2 y 4 no tienen error; si la paridad del de la posición 2 coincide también, las columnas 3 y 4 no lo tienen, con lo cual el posible error debe estar en la primera. Nótese que el bit de la posición 2 se deberá convertir en 0 por el número par de unos que contiene el subconjunto que controla.

1 0	**0** 1	**1** 2	**1** 3
1 4	**1** 5	**0** 6	**0** 7
0 8	**0** 9	**1** 10	**0** 11
0 12	**1** 13	**1** 14	**1** 15

Figura 3: Subconjunto impar

1 0	**0** 1	**0** 2	**1** 3
1 4	**1** 5	**0** 6	**0** 7
0 8	**0** 9	**1** 10	**0** 11
0 12	**1** 13	**1** 14	**1** 15

Figura 4: Subconjunto de columnas 3 y 4

El mismo razonamiento se puede utilizar para averiguar la fila en la que se encuentra el posible error, con bits de paridad en las posiciones 4 (Figura 5) y 8 (Figura 6). Nótese que se deberá cambiar

el 1 en la posición 4 por un 0, debido al número par de unos.

1 0	0 1	1 2	1 3
1 4	1 5	0 6	0 7
0 8	0 9	1 10	0 11
0 12	1 13	1 14	1 15

Figura 5: Subconjunto de fila par

1 0	0 1	0 2	1 3
1 4	1 5	0 6	0 7
0 8	0 9	1 10	0 11
0 12	1 13	1 14	1 15

Figura 6: Subconjunto de filas 3 y 4

El número total de preguntas será la suma de aquellas que se necesitan para averiguar en qué columna se encuentra más las que sirven para averiguar la fila.

Para ver el funcionamiento ante cualquier caso, supongamos que se envía un mensaje (con los bits de paridad en su numero correspondiente), en el cual se produce un error y un bit aleatorio es cambiado:

0 0	0 1	0 2	0 3
1 4	0 5	0 6	1 7
0 8	0 9	1 10	1 11
0 12	1 13	0 14	1 15

Mensaje enviado

0 0	0 1	0 2	0 3
1 4	0 5	0 6	1 7
0 8	0 9	1 10	0 11
0 12	1 13	0 14	1 15

Mensaje con error

El receptor, al analizar el bit de paridad de la posición 1, detecta que es incorrecto, de manera que el error debe estar en las columnas 2 o 4. Cuando procede con el bit de la posición 2, encuentra que éste tampoco es correcto y por lo tanto sabe que hay un error en la columna 3 o 4. Con esta información puede deducir que la columna 4 lo contiene. El siguiente paso es observar que el bit en la posición 4 es correcto (descarta la fila 2 y 4), mientras el de la posición 8 no (descarta la fila 1 y 2), con lo cual ya sabe que debe cambiar el 0 que se encuentra en la posición 11 por un 1, ha corregido el incorrecto.

4.2. Redundancia

El proceso funciona para un error en cualquier posición, incluso para la de los bits de paridad mismos. Una corta demostración de ello es considerando que hay 4 preguntas dicotómicas que se han realizado, de manera que hay 2^4 posibles estados de respuesta (dos estados por pregunta), de manera que en principio se abarca todo el conjunto de 16 bits. Lo anterior es falso, debido a que hay que tener en cuenta que existe un estado numero 17, el de no error. Para arreglar el problema, el bit en la posición 1 se elimina, teniendo así 15 bits y un estado de no error, 16 en total. Otra alternativa (la más comúnmente usada, denominada código de Hamming extendido) es utilizar ese bit extra para comprobar la paridad del conjunto total, de manera que se pueden detectar 2 errores y corregir uno de ellos.

La pregunta natural que debería surgir es: ¿cuánta redundancia debo usar si quiero enviar una

cantidad de información i? Sabiendo

$$i = l - p \tag{3}$$

(donde l es el largo del mensaje y p la cantidad de bits de paridad a usar) y dejando todo en función de p, se podría calcular cuánta redundancia se necesita para esa cantidad de información y se podría definir un factor de redundancia R para estudiar y determinar las mejores condiciones en las que usar el código.

Para un cierto l, la cantidad de preguntas dicotómicas a realizar sin tener en cuenta el bit de paridad global deben abarcar todo el conjunto del mensaje, de manera que el largo del mensaje l debe cumplir $2^{p-1} = l$. Si se deja en p en función de l se obtiene el número de bits de paridad que debe contener el mensaje:

$$p = \frac{\ln l}{\ln 2} + 1 \tag{4}$$

Como se puede observar de la Ecuación 4, $p \in \mathbb{R}$ para valores $l \in \mathbb{N}$. El problema es que en la realidad solo tiene sentido usar aquellos $p \in \mathbb{N}$, de manera que, por fuerza, el largo del mensaje debe cumplir $l = 2^{p-1}$, $\forall p \in \mathbb{N}$. Como se ha visto en el Apartado 4.1, para usar un argumento geométrico parecido al anterior, la longitud del mensaje también debe cumplir $l = a \times b$. De esta manera $a = 2^n$ y $b = 2^{p-1-n}$, $n < p$, $\forall n, p \in \mathbb{N}$.

$$l = 2^{p-1}, \quad a = 2^n, \quad b = 2^{p-1-n}, \qquad n < p, \quad \forall n, p \in \mathbb{N} \tag{5}$$

El factor R a definir, describe la relación entre la cantidad de información útil que se desea enviar y la cantidad de bits de paridad que se deben usar para controlar los errores (usando el código extendido de Hamming).

$$R = \frac{i}{p} = \frac{l-p}{p} = \frac{2^{p-1} - p}{p} \tag{6}$$

Analizando el valor de la derivada de R con respeto a p:

$$\frac{d}{dp}(R_r) = \frac{d}{dp}\left(\frac{2^{p-1} - p}{p}\right) = \frac{1}{p}\frac{d}{dp}\left(2^{p-1} - p\right) + \left(2^{p-1} - p\right)\frac{d}{dp}\left(\frac{1}{p}\right) = \frac{2^{p-1}\ln(2) - 1}{p} - \frac{2^{p-1} - p}{p^2}$$

$$\frac{d}{dp}(R_r) = \frac{1}{p}\left(2^{p-1}\ln(2) - 1 - \frac{2^{p-1} - p}{p}\right) \tag{7}$$

(la expresión 7 representa una función monótona creciente a partir de $p > 0$)

$$\lim_{p\to\infty} \frac{d}{dp}(R) = \lim_{p\to\infty} \frac{1}{p}\left(2^{p-1}\ln(2) - 1 - \frac{2^{p-1} - p}{p}\right) = \infty$$

se deduce que R siempre crece con p dentro del rango de uso ($p > 0$), de modo que a primera vista uno puede pensar que siempre será más eficaz el sistema en cuanto a bits redundantes cuanta más información se quiera enviar, lo cual parece absolutamente práctico. Sin embargo, hay que recordar que el sistema no es capaz de detectar i corregir múltiples errores, los cuales pueden surgir al incrementar la longitud del mensaje; de modo que para cada sistema con su propia probabilidad de error por bit, hay que buscar la manera más óptima de agruparlos en paquetes a ser sometidos al código por separado.

La anterior técnica es la que se utiliza en los sistemas de comunicación. Los valores cantidad de bits en paquetes se pueden obtener con la Ecuación 6, dándole valores a p. Se suelen escribir éstos casos distintos en el formato (l, i). Utilizando las expresiones 3 y 5, se obtienen valores típicos para códigos de Hamming extendidos, algunos típicos son:

- $p = 4 \implies (2^{4-1}, 2^{4-1} - 4) \implies$ **Extended Hamming** $(\mathbf{8}, \mathbf{4})$
- $p = 5 \implies (2^{5-1}, 2^{5-1} - 5) \implies$ **Extended Hamming** $(\mathbf{16}, \mathbf{11})$
- $p = 6 \implies (2^{6-1}, 2^{6-1} - 6) \implies$ **Extended Hamming** $(\mathbf{32}, \mathbf{26})$

5. Códigos de Reed-Solomon

La metodología de control de errores de Reed-Solomon es ampliamente usada en el mundo de las comunicaciones modernas. Su uso puede ser observado en comunicaciones móviles, de satélites, en la televisión digital, los CD's, etc. El método se basa en el uso de las funciones matemáticas: de la misma manera que dada una función se puede calcular su valor en cualquier punto; dada suficiente información, el receptor puede conocer una función especifica y única para esa información y usarla para averiguar qué se ha mandado en puntos donde la información ha sido dañada o directamente no haya llegado.

5.1. Polinomio interpolador de Lagrange

El polinomio interpolador de Lagrange es una función polinómica **única** de grado $deg\,(P) \leq (n-1)$ que pasa por n puntos. Para conseguirlo, se suman polinomios que pasan por 0 en todos los puntos menos en (x_i, y_i), donde se aplica un factor escalador para hacerlos pasar por $(x_i, 1)$. De esta manera se fuerza a la suma de todos éstos a no afectarse entre sí en cuanto a pasar por las coordenadas en cuestión, de modo que todos los coeficientes se suman una vez multiplicados por el valor de la coordenada y_i.

Para $n = 3$ (por ejemplo), con $p_1 = (0, 3)$, $p_2 = (1, 2)$, $p_3 = (3, 1)$, el polinomio de grado 2 en cuestión se calcularía de la siguiente manera:

$$P_1(x) = \frac{(x-1)\,(x-2)}{(0-1)\,(0-3)} \qquad P_2(x) = \frac{(x-0)\,(x-3)}{(1-0)\,(1-3)} \qquad P_3(x) = \frac{(x-0)\,(x-1)}{(3-0)\,(3-1)}$$

$$P(x) = 3P_1(x) + 2P_2(x) + 1P_3(x)$$

Para obtener la expresión general, se puede partir del ejemplo anterior y, mediante el argumento inductivo, se llega a:

$$P(x) = \sum_{i=1}^{n} P_i(x) \qquad P_i(x) = y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

De manera que la expresión analítica del polinomio interpolador de n puntos queda como se observa en la Ecuación 8.

$$P(x) = \sum_{i=1}^{n} \left(y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \right) \tag{8}$$

5.2. Funcionamiento

Dado un mensaje que contiene n valores, cada uno siendo m_1, m_2, ..., m_n se añaden un número k de valores que representa el número máximo de información se desea poder perder. Se procede a asignar el valor de éstos a puntos distintos a los del mensaje que pertenecen al polinomio único interpolador de Lagrange, de manera que si en la transmisión se pierden k valores, el receptor puede obtener cada uno usando el polinomio.

Supongamos que se quiere enviar la siguiente información, y que como mucho se admite perder 2 valores.

3 m_0	**0** m_1	**4** m_2	**0** m_3	**1** m_4	**2** m_5

Se deben añadir 2 paquetes de información para cumplir con los requisitos. Supongamos que se pierden m_2 y m_3.

3 m_0	**0** m_1	**4** m_2	**0** m_3	**1** m_4	**2** m_5	m_6	m_7

Para encontrar el valor de m_6 y m_7 antes de enviar el mensaje, se debe calcular el polinomio interpolador de Lagrange asociado a los datos que se quieren enviar (visualizado en Desmos https://www.desmos.com/calculator/m9b4h8f0vx?lang=es):

$$P_1(x) = \frac{(x-2)(x-3)(x-4)(x-5)(x-6)}{(1-2)(1-3)(1-4)(x-5)(1-6)} = \frac{1}{-120}\left(x^5 - 20x^4 + 155x^3 - 580x^2 + 1044x - 720\right)$$

$$P_3(x) = \frac{(x-1)(x-2)(x-4)(x-5)(x-6)}{(3-1)(3-2)(3-4)(3-5)(3-6)} = \frac{1}{-12}\left(x^5 - 18x^4 + 121x^3 - 372x^2 + 508x - 240\right)$$

$$P_5x) = \frac{(x-1)(x-2)(x-3)(x-4)(x-6)}{(5-1)(5-2)(5-3)(5-4)(5-6)} = \frac{1}{-24}\left(x^5 - 16x^4 + 95x^3 - 260x^2 + 324x - 144\right)$$

$$P_6(x) = \frac{(x-1)(x-2)(x-3)(x-4)(x-5)}{(6-1)(6-2)(6-3)(6-4)(6-5)} = \frac{1}{120}\left(x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120\right)$$

$$P(x) = 3P_1(x) + 4P_3(x) + P_5(x) + 3P_6(x)$$

Los valores de paquetes de control a enviar deben ser los valores de la función de Lagrange, para que el receptor sea capaz de generar el polinomio y obtener aquellos perdidos. Usando lo calculado, $m_6 = P(7) = -66$ y $m_7 = P(8) = -382$, de manera que la información recibida sería:

3 m_0	**0** m_1	m_2	m_3	**1** m_4	**2** m_5	**-66.0** m_6	**-382.0** m_7

El receptor generaría el polinomio, y mediante $P(3) = 4$ y $P(4) = 0$, rellena la información que falta. Dado el caso de que uno de los errores estuviera en los bits redundantes, el receptor sigue teniendo la suficiente información como para generar el polinomio y encontrar el valor de aquello que falta,

Se observa que los valores a enviar de los paquetes de control son negativos y de valores muy distintos a los de información. Es cierto que lo anterior es un problema importante del sistema, la manera de corregirlo es usando la aritmética modular con módulo p, de manera que el polinomio de Lagrange en sí (definido como $f : \mathbb{Z}_p \to \mathbb{Z}_p$) es totalmente diferente al calculado (definido como $f : \mathbb{R} \to \mathbb{R}$) ya que existe en un espacio numérico distinto. De esta manera se controla el rango de valores recibidos en función del receptor en cuestión, facilitando y haciendo más eficaz la velocidad de transmisión. Para sistemas digitales (por ejemplo), se cogería $p = 1$, de manera que se limitan los valores recibidos a 0 y 1.

Referencias

[1] Nuh Aydin. "An Introduction to Coding Theory via Hamming Codes: A Computational Science Model". En: *Faculty Publications* (2007). URL: https://digital.kenyon.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1000&context=math_pubs.

[2] 3B1B Grant Sanderson. *But what are Hamming codes? The origin of error correction.* 2020. URL: https://youtu.be/X8jsijhllIA?si=YP-iCKWc0OCBuXY5.

[3] Henry D. Pfister Jack Keihl Wolf Paula Evans. "An Introduction to Reed-Solomon Codes". En: (2021). URL: http://pfister.ee.duke.edu/courses/ecen604/rspoly.pdf.

[4] Vahid Meghdadi. "Wireless Communications by Andrea Goldsmith". En: (2008). URL: https://www.unilim.fr/pages_perso/vahid/notes/ber_awgn.pdf.

[5] "Repetition Code". En: (2023). URL: https://en.wikipedia.org/wiki/Repetition_code#:~:text=In%20coding%20theory%2C%20the%20repetition,repeat%20the%20message%20several%20times..

[6] vcubingx Vivek Verma. *What are Reed-Solomon Codes?* 2022. URL: https://youtu.be/1pQJkt7-R4Q?si=qWnqXVVAdU6dJ-9h.