1. DEFINICIONES

1.1 Sistema:

Un *sistema* es una combinación de entes que interactúan conjuntamente para cumplir un determinado objetivo, y sobre el que actúan una o mas *entradas* provocando una o mas *salida*s.
De acuerdo a las características de las entradas y/o salidas se clasifican los sistemas, que para nuestro interés son los siguientes:

- **Sistemas Analógicos:** Son sistemas que manejan cantidades en forma analógica, es decir trabajan con variables continuas. El número de valores que pueden tomar las entradas/salidas son infinitos. Ejemplos de estos sistemas pueden ser los amplificadores de audio, los indicadores de aguja, etc.

- **Sistemas Digitales:** Son dispositivos diseñados para manejar cantidades en forma digital (discreta). En su mayoría son dispositivos electrónicos, pero también los hay eléctricos, mecánicos, neumáticos, magnéticos, etc. Las entradas/salidas sólo pueden tomar un cierto número finito de valores (valores discretos). Ejemplos de estos sistemas son las PC, los reproductores de DVD, etc.

Existen sistemas conversores de una forma a otra. Tenemos los conversores analógico a digital (A/D o ADC) que reciben señales analógicas o continuas, y entregan salidas digitales o discretas; y por otro lado tenemos los conversores digital a analógico (D/A o DAC) que reciben señales digitales y entregan señales analógicas.

1.2 Ruido:

Se denomina así a un disturbio (generalmente no buscado) que se superpone a una señal útil, tendiendo a degradar su contenido de información. Normalmente es aleatorio y esta formado por perturbaciones transitorias que se extienden sobre un espectro de frecuencias considerable. Suelen clasificarse según los orígenes en *naturales*, como los galácticos (radiaciones provenientes del espacio), y los *artificiales* o causados por el hombre, como los originados por los sistemas de encendido de automotores y los aparatos electrodomésticos.

1.3 BIT:

Se denomina así a cada dígito binario (**BI**nary digi**T**). Los *bits* se suelen agrupar en *palabras*, siendo las longitudes más usuales las de 4, 8, 16, 32 y 64 bits.

1.4 NIBBLE:

Se denomina de esta manera a la agrupación de 4 bits.

1.5 BYTE:

Se denomina así a la agrupación de 8 bits. También se le suele denominar *octeto*.

1.6 WORD:

Se denomina de esta manera a la agrupación de 16 bits (o de 2 bytes).

2. DIFERENCIAS ENTRE SISTEMAS ANALOGICOS Y DIGITALES

En general en los sistemas digitales se puede lograr mayor precisión que en los analógicos aumentando convenientemente el número de etapas o componentes; en los analógicos la precisión suele ir ligada mas bien a la calidad de los componentes y aumentarla suele ser más caro y dificultoso. Además los sistemas digitales son menos sensibles al ruido y pueden construirse mediante la interconexión de unos pocos circuitos básicos que se repiten varias veces. Resumiendo:

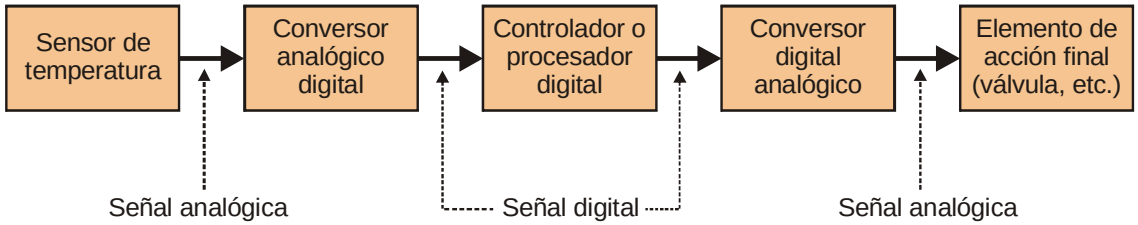
2.1 Ventajas de los sistemas digitales sobre los sistemas analógicos:

- **Son más fáciles de diseñar:** como trabajan con dispositivos de conmutación, los valores exactos de corriente o tensión no interesan, es suficiente con conocer un rango de sus valores extremos (alto, bajo).
- **Facilidad para almacenar información:** permiten almacenar y retener información en forma más confiable, utilizando memorias magnéticas o electrónicas.
- **Mayor inmunidad al ruido:** los sistemas digitales poseen intrínsecamente una mayor inmunidad al ruido eléctrico que los sistemas analógicos.

2.2 Limitaciones de los sistemas digitales:
La principal desventaja se encuentra en el hecho de que cualquier variable a procesar por un sistema digital es, en la gran mayoría de los casos, una variable analógica (por ejemplo una temperatura, o una velocidad, un nivel, etc.). Las salidas de un sistema digital también suelen ser variables analógicas (por ejemplo, la posición de una válvula). Es decir, las variables que manipula un sistema digital, suelen ser variables analógicas. Por lo tanto, la limitación del sistema digital radica en el hecho de que cualquier variable analógica de entrada o salida, debe convertirse en su versión digital.
Por ejemplo, veamos como sería el caso de un control de temperatura digital. La entrada del sistema es una variable analógica (la temperatura), y la salida es otra variable analógica (la posición de una válvula de gas). Para procesar esta información, el sistema digital debe realizar los siguientes pasos:

1. Convertir la entrada analógica del "mundo real" en una magnitud digital.
2. Procesar en forma digital esta magnitud para obtener la salida digital.
3. Convertir la salida digital a su forma analógica "del mundo real" para poder aplicarla o interpretarla.

Esto en un diagrama de bloques se vería de la siguiente manera:



3. REPRESENTACIONES NUMERICAS

Cotidianamente acostumbramos a trabajar con "cantidades". Estas cantidades se miden, se monitorean, se registran, se manipulan aritméticamente, se aplican en procesos físicos etc. A estas "cantidades" es necesario poder representarlas de alguna manera, para ello existen básicamente dos formas de representación: las denominadas representación analógica y representación digital.

3.1 Representación Analógica:
La característica principal de esta representación, es que dado un intervalo de valores, las "cantidades" pueden tomar infinitos valores dentro de este intervalo. Dicho de otra forma, si la "cantidad" esta representada por una variable analógica, dicha variable analógica puede tomar infinitos valores entre dos valores dados. Por ejemplo, si representamos una temperatura con una variable analógica, entre dos valores cualesquiera de temperatura (digamos entre 35 y 36 °C), la variable podrá tomar infinitos valores (35,1°C, 35,11°C, 35,268°C, etc.). Otro ejemplo: los números reales.

3.2 Representación Digital:
En este tipo de representación, dado un intervalo de valores, las "cantidades" no toman infinitos valores dentro de ese intervalo, sino que solamente pueden tomar una cantidad finita. Para representar una variable digital se utilizan símbolos denominados "dígitos"; ejemplo de estos son el sistema decimal (compuesto de 10 símbolos o dígitos distintos), o también el sistema binario (compuesto por 2 dígitos). Ejemplo de una variable digital es la cantidad de lados que posee un dado, otro ejemplo son los números enteros.
Los circuitos electrónicos digitales trabajan con el sistema de representación binario (dos símbolos), y para representar cada símbolo lo hacen utilizando dos niveles de tensión eléctrica diferenciados.

3.3 Sistemas de representación de cantidades numéricas

3.3.1 Sistema decimal:
Es el sistema que utilizamos cotidianamente. El mismo se compone por diez símbolos distintos (del 0 al 9). También es un sistema de valor posicional, en el que cualquier número se representa por una serie de potencias de base diez. Por ejemplo:

$874_{10} = 8x10^2 + 7x10^1 + 4x10^0$

$54{,}48_{10} = 5x10^1 + 4x10^0 + 4x10^{-1} + 8x10^{-2}$

3.3.2 Sistema binario natural:

Este utiliza dos símbolos (0 y 1). También es un sistema de valor posicional, pero en este caso cualquier número se representa en potencias en base 2. Ejemplo:

$11011_2 = 1x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 27_{10}$

3.3.3 Sistema octal:

Se utilizaba como medio taquigráfico para simplificar la representación de números binarios, en la actualidad prácticamente no se utiliza. Consta de 8 símbolos distintos (del 0 al 7). Un número octal se representa por potencias en base 8. Ejemplo:

$152_8 = 1x8^2 + 5x8^1 + 2x8^0 = 106_{10}$

3.3.4 Sistema hexadecimal:

Su aplicación es similar al del sistema octal, es decir se utiliza para representar en forma mas simplificada un número binario. Por ejemplo, un número binario de 16 dígitos queda reducido a un número hexadecimal de 4 dígitos.
El sistema hexadecimal consta de 16 símbolos alfanuméricos: del 0 al 9 y de la A hasta la F, es decir:

0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - A - B - C - D - E - F

En forma análoga a los sistemas anteriores, un número hexadecimal se expresa como potencias en base 16. Ejemplo:

$3E1A_{16} = 3x16^3 + 14x16^2 + 1x16^1 + 10x16^0 = 15898_{10}$

3.4 Conversión entre distintos sistemas

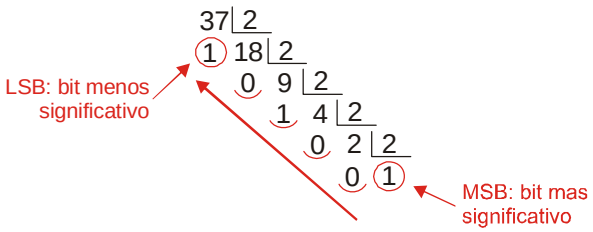
3.4.1 Conversión de un número binario natural a decimal:

Simplemente se suman las distintas potencias (base 2) del número binario, multiplicando cada potencia por su dígito correspondiente. Ejemplo, para convertir a decimal el número 1101011_2:

$1101011_2 = 1x2^6 + 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 107_{10}$

3.4.2 Conversión de un número decimal a binario natural:

El número decimal se divide por 2, el resto de esta división es el bit menos significativo del número binario. El resultado se vuelve a dividir por 2, el resto de esta división es el bit más significativo que el anterior; y así hasta finalizar la división. Los restos de la división forman el número binario. Ejemplo, para convertir a binario natural el número decimal 37_{10}:



$37_{10} = 100101_2$

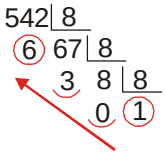
3.4.3. Conversión de un número octal a decimal:

Se realiza mediante la suma de potencias en base 8. Ejemplo, para convertir 552_8 a decimal:

$552_8 = 5x8^2 + 5x8^1 + 2x8^0 = 362_{10}$

3.4.4. Conversión de decimal a octal:

Se utiliza la división repetida por 8. Ejemplo, para pasar el decimal 542_{10} a octal:



$542_{10} = 1036_8$

3.4.5. Conversión de octal a binario:

Cada dígito del número se reemplaza por su equivalente binario de 3 bits, según la siguiente tabla:

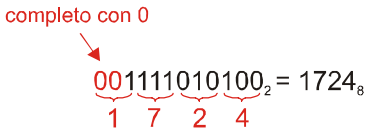
Dígito Octal	Equivalente Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Por ejemplo, el octal 542_8 en binario natural sería:

$542_8 = 101100010_2$
5 4 2

3.4.6. Conversión de binario a octal:

Es la operación inversa a la anterior. Los dígitos binarios se agrupan de a tres (de derecha a izquierda; y en caso de que la última agrupación de mas a la izquierda no llegue a 3 bits, se completa con ceros), y cada agrupación se reemplaza por su equivalente octal. Por ejemplo, el binario natural 1111010100_2 sería:



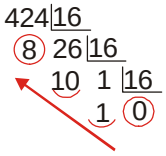
3.4.7. Conversión de hexadecimal a decimal:

Análogo a los casos anteriores, se suman las potencias en base 16. Por ejemplo, el hexadecimal $FE15_{16}$ en decimal sería:

$FE15_{16} = 15x16^3 + 14x16^2 + 1x16^1 + 5x16^0 = 65045_{10}$

3.4.8. Conversión de decimal a hexadecimal:

Se utiliza la división repetida, en este caso por 16. Ejemplo, el decimal 424_{10} en hexadecimal sería:



El resto de **10** equivale a la letra **A**; recordar que hexadecimal va del 0 al 9 y de la A (equivalente al 10) a la F (equivalente al 15). Por lo tanto, la conversión queda:

$424_{10} = 01A8_{16} = 1A8_{16}$ (notar que, como en cualquier sistema, el 0 de más a la izquierda carece de significado)

3.4.9. Conversión de hexadecimal a binario:

Cada dígito hexadecimal se reemplaza por su equivalente binario de 4 bits, de acuerdo a la siguiente tabla:

Dígito Hexadecimal	Equivalente Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Dígito Hexadecimal	Equivalente Binario
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Por ejemplo, el hexadecimal $4EB3_{16}$ en binario natural sería:

$4EB3_{16} = \underbrace{0100}_{4}\underbrace{1110}_{E}\underbrace{1011}_{B}\underbrace{0011}_{3}{}_{2}$

O lo que es lo mismo 100111010110011_2, ya que el 0 de más a la izquierda es irrelevante.

3.4.10. Conversión de binario a hexadecimal:

Los dígitos binarios se agrupan de a 4 bits, de derecha a izquierda, y cada agrupación se reemplaza por su equivalente hexadecimal. Ejemplo, el binario natural 1001011110110_2 en hexadecimal sería:

completo con 0

$\underbrace{0001}_{1}\underbrace{0010}_{2}\underbrace{1111}_{F}\underbrace{0110}_{6}{}_{2} = 12F6_{16}$

4. CODIGOS BINARIOS

Un *código* es una correspondencia biunívoca entre cantidades y símbolos. Cualquier código se compone de una determinada cantidad **S** de símbolos, los que agrupados en un número **n** de dígitos determinan **C** cantidades o combinaciones distintas según la siguiente relación:

$$C = S^n$$

Por ejemplo, para el caso del código binario (S=2), si utilizo 3 dígitos (n=3), dispongo de $2^3 = 8$ combinaciones o cantidades distintas (000, 001, 010, 011, 100, 101, 110 y 111).
Los códigos numéricos de uso más frecuente en el área de las técnicas digitales son: el decimal, el binario natural, el binario reflejado o Gray, el BCD natural (decimal codificado a binario), el BCD exceso tres, y el hexadecimal.
En las técnicas digitales, en lo que respecta a los códigos binarios, se utilizan dos símbolos: el 0 y el 1. Un criterio similar se adopta para los otros códigos numéricos que requieren menor cantidad de símbolos que el decimal: se usan los símbolos necesarios, del "0" en adelante. En el código hexadecimal, que requiere dieciséis símbolos, se utilizan los diez del decimal con el agregado de las 6 primeras letras del alfabeto, (preferentemente en mayúsculas). Existen también códigos alfabéticos (relación entre letras y símbolos) y alfabético-numéricos o alfanuméricos que reúnen a ambos. Los más utilizados en este campo son el ASCII (American Standard Code for Information Interchange) de uso universal y el EBCDIC (Extended Binary Code Decimal Interchange Code), creado por IBM y empleado fundamentalmente en sus computadoras.
Antes de ver los códigos binarios más utilizados, es conveniente tener presente las siguientes definiciones:

- **Código continuo o adyacente:** son los códigos cuyos números correspondientes a cantidades sucesivas difieren sólo en un bit.
- **Código cíclico:** cuando además de la condición anterior, también se da que la última combinación del código es adyacente a la primera.

4.1. Código Gray

Se denomina también código de cambio mínimo o reflejado. Es un código *adyacente*, ya que al pasar de una posición a otra sólo se cambia un bit, y también es *cíclico*. Se lo utiliza mayormente en transductores digitales de posición o desplazamiento.
Por ejemplo, el código gray de 4 bits sería:

Decimal	Binario Natural	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

Decimal	Binario Natural	Gray
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

4.2. Código de Johnson

Es un código *continuo* y *cíclico*, que con **n** bits es capaz de codificar **2n** números. Un código de Johnson de 5 bits sería:

Decimal	Binario Natural	Johnson
0	0000	00000
1	0001	00001
2	0010	00011
3	0011	00111
4	0100	01111
5	0101	11111
6	0110	11110
7	0111	11100
8	1000	11000
9	1001	10000

4.3. Códigos Decimales Codificados en Binario (BCD)

El intercambio de información con los sistemas suele resultar mas cómodo para el operador si las cifras aparecen en decimal. Pero los sistemas operan en binario. Esto origina una serie de codificaciones en las que cada dígito decimal se sustituye por un binario de al menos 4 bits.
Existen códigos BCD **ponderados** y **no ponderados**. En los ponderados, a cada posición binaria se le asigna un peso, y el decimal equivalente se obtiene sumando los pesos cuyo coeficiente sea “1”. De esta manera se tienen los BCD natural (o código 8421), el Aitken (2421), el 5421, etc. El Aitken es además *autocomplementario*, esto es el número decimal **N** y el **9-N** son complementarios (cambian los ceros por los unos y viceversa).
En los códigos no ponderados, las posiciones no tienen asignado peso. De este tipo es el BCD exceso 3, en el que a cada dígito decimal **N** se le asigna el binario natural que correspondería a **N+3**.

Decimal	BCD Natural 8421	Aitken 2421	5421	BCD Exceso 3
0	0000	0000	0000	0011
1	0001	0001	0001	0100
2	0010	0010	0010	0101
3	0011	0011	0011	0110
4	0100	0100	0100	0111
5	0101	1011	1000	1000
6	0110	1100	1001	1001
7	0111	1101	1010	1010
8	1000	1110	1011	1011
9	1001	1111	1100	1100

5. Códigos detectores y correctores de errores

Un sistema de comunicación ideal se puede representar por tres componentes esenciales:

- Transmisor o fuente.
- Canal o medio de almacenamiento.
- Receptor.

El proceso de comunicación involucra el flujo de información a través de un medio, el cual va del transmisor al receptor. En la práctica, el medio o canal está sujeto a una diversidad de perturbaciones que resultan en una distorsión del mensaje que se está transmitiendo. Estas perturbaciones se denominan *ruido*, y la forma en la cual el ruido puede aparecer depende del canal. En cualquier caso, siempre se trata de minimizar las pérdidas debidas al ruido y recuperar el mensaje original.
Una manera de minimizar los efectos del ruido es mediante la utilización de códigos que permitan detectar que ocurrió un error en la transmisión, e incluso corregir el error detectado. Tenemos entonces dos tipos de códigos:

- **Códigos de detección de errores:** envían información adicional junto con los datos, lo que permite deducir si ocurrió un error (pero no cual), y en este caso pedir la retransmisión.
- **Códigos de corrección de errores:** envían información redundante junto con los datos, que permite deducir si ocurrió un error, y llegado el caso permiten corregirlo.

5.1 Códigos de paridad

En estos códigos el transmisor agrega un bit adicional a los bits de datos, de manera tal que la cantidad total de bits sea par o impar. El receptor comprueba la paridad de los datos recibidos, y en función de esta comparación determina si hubo o no error en la transmisión.
Dentro de los códigos de paridad tenemos los de *paridad par* (la cantidad total de bits en 1 es par) y los de *paridad impar* (la cantidad total de bits en 1 es impar). Estos tipos de códigos sólo permiten detectar si ocurrió un error (pero no corregirlo). La desventaja de este tipo de códigos es que si en el mensaje transmitido se produce un error en un número par de bits, la paridad del mensaje no cambia y no se detecta la ocurrencia del error.
Por ejemplo, para un mensaje de 8 bits de datos, el bit de paridad sería:

Bits de Datos	Bit de Paridad Par	Bit de Paridad Impar
11010010	0	1
10010001	1	0

Para el primer caso del ejemplo, el mensaje transmitido sería **11010010**0 si se eligiera paridad par, o **11010010**1 si se eligiera paridad impar.

5.2 Código de Hamming

Este código permite detectar la ocurrencia de error en uno o dos bits, y también permite corregir errores en un solo bit. Es entonces un código que permite la corrección de errores.
En este código el transmisor añade a los bits de datos un determinado número de bits detectores-correctores, y estos bits se intercalan en un orden determinado en el mensaje a transmitir. El receptor, en función del mensaje recibido (datos + bits detectores), puede conocer si ocurrió un error en la transmisión, e incluso corregir el bit erróneo.
En este código se necesitan **m** bits de paridad para una cantidad $\mathbf{2^m-1-m}$ bits de datos, o sea que se transmiten $\mathbf{2^m-1}$ bits en total (datos + paridad). Por ejemplo, digamos que tenemos 4 bits de datos, necesitamos entonces 3 bits de paridad (**m=3**) ya que con esta cantidad se cubren los 4 bits de datos (2^3-1-3 = 4).
Luego el transmisor ordena los bits de datos y de paridad según la siguiente ley de formación: si en total hay 2^m-1 bits, los bits de las posiciones $\mathbf{2^k}$ (con $\mathbf{0 \le k \le m-1}$) serán bits de paridad, y el resto de las posiciones serán bits de datos. Siguiendo en nuestro ejemplo de 4 bits de datos, los bits de paridad estarían en las posiciones 1 (2^0), 2 (2^1), y 4 (2^2). Es decir:

7	6	5	4	3	2	1
D	D	D	P	D	P	P

D = bit de dato
P = bit de paridad

El valor de cada bit de paridad se elige de modo que el número total de unos en un número específico de bits sea par, y estos grupos se eligen de forma tal que ningún bit de datos se cubra con la misma combinación de bits de paridad. Esto es lo que proporciona al código su capacidad de corrección. El algoritmo es el siguiente: el valor del bit de paridad de la posición $\mathbf{2^k}$ estará en función de los bits en las posiciones que tengan en 1 al bit **k** en su representación binaria. Traducido al castellano:

Para el bit de paridad 1, **k** = 0 (ya que $2^0 = 1$) ¿Qué posiciones tienen su bit 0 en 1? Las posiciones:

- $001_2 = 1_{10}$
- $011_2 = 3_{10}$
- $101_2 = 5_{10}$
- $111_2 = 7_{10}$

Entonces el bit de paridad 1 tomará su valor en función de los bits número 1, 3, 5 y 7.

Para el bit de paridad 2, **k** = 1 (ya que $2^1 = 2$) ¿Qué posiciones tienen su bit 1 en 1? Las posiciones:

- $010_2 = 2_{10}$
- $011_2 = 3_{10}$
- $110_2 = 6_{10}$
- $111_2 = 7_{10}$

Entonces el bit de paridad 2 tomará su valor en función de los bits número 2, 3, 6 y 7.

Para el bit de paridad 4, **k** = 2 (ya que $2^2 = 4$) ¿Qué posiciones tienen su bit 2 en 1? Las posiciones:

- $100_2 = 4_{10}$
- $101_2 = 5_{10}$
- $110_2 = 6_{10}$
- $111_2 = 7_{10}$

Entonces el bit de paridad 4 tomará su valor en función de los bits número 4, 5, 6 y 7.

Veamos un ejemplo. Un equipo transmisor necesita enviar paquetes de 4 bits, codificándolos por Hamming. Vimos que para 4 bits se necesitan adicionar 3 bits de paridad, con lo que el total de bits a transmitir serán 7.
Supongamos que el paquete de datos a enviar es **1011**, entonces el código de Hamming sería:

Datos (sin paridad)	**D7**	**D6**	**D5**	**P4**	**D3**	**P2**	**P1**
1011	**1**	**0**	**1**		**1**		
P4	1	0	1	**0**			
P2	1	0			1	**0**	
P1	1		1		1		**1**
Datos (con paridad)	**1**	**0**	**1**	**0**	**1**	**0**	**1**

Entonces el código que enviará el transmisor será **1010101**. Cuando el receptor recibe este código, recalcula los bits de paridad para determinar si ocurrió un error:

Código recibido	**D7**	**D6**	**D5**	**P4**	**D3**	**P2**	**P1**	**Prueba de Paridad**	**Error**
	1	**0**	**1**	**0**	**1**	**0**	**1**		
P4	1	0	1	**0**				**Ok**	0
P2	1	0			1	**0**		**Ok**	0
P1	1		1		1		**1**	**Ok**	0

En este caso, el receptor determina que no hay error de paridad, entonces el código recibido es correcto y no hubo error en la transmisión.

Supongamos ahora que la transmisión se ve afectada por ruido, y en lugar de llegar **1010101** al receptor, llega **1010001**. Cuando el receptor reciba este código y recalcule la paridad:

Código recibido	D7	D6	D5	P4	D3	P2	P1	Prueba de Paridad	Error
	1	**0**	**1**	**0**	**0**	**0**	**1**		
P4	1	0	1	**0**				**Ok**	0
P2	1	0			**0**	**0**		**Error**	**1**
P1	1		1		**0**		**1**	**Error**	**1**

En este caso el receptor detecta que existen errores en la paridad del código recibido. Ahora bien, ¿cómo determina cual es el bit erróneo? Si observamos la columna **Error**, vemos que $\mathbf{P_4P_2P_1}$ forman el binario $\mathbf{011_2}$, que en decimal es **3**. Esto significa que el tercer bit del código recibido es el erróneo, y es el que debe cambiarse. O sea, si en **1010001** se cambia el tercer bit, se obtiene **1010101** que es el código correcto.