Unidad 1.2 – SISTEMAS DE REPRESENTACIÓN DE LA INFORMACIÓN

CÓDIGOS BINARIOS

Contenido

- Introducción
- Elementos de un código binario
- Propiedades
- Clasificación
- Códigos generales
- Códigos BCD
- Códigos alfanuméricos
- Códigos detectores de errores
  - Métodos de detección
- Códigos correctores de errores
  - Métodos de corrección

Referencias (Aula Virtual)

> Wakerly J. (2001). DISEÑO DIGITAL. Capítulo 2: Sistemas y Códigos Numéricos.

> Velasco et al (2011). Fundamentos de computadores. Cap.1: Los números y los sistemas de representación. Cap. 2: Representación de los números en un computador. Cap. 3: Otros tipos de representaciones.

> Nelson et al (1996). Análisis y Diseño de Circuitos Lógicos Digitales. Cap. 1: Sistemas Numéricos y Códigos.

Introducción

Se puede definir la codificación como la conversión de un conjunto de símbolos en otro.

- **Inicialmente la codificación ha sido utilizada con el propósito de representar hechos o situaciones complejas mediante elementos más simples.**
- **También para esconder en un conjunto de símbolos a los que se le atribuye un significado preestablecido, sucesos o información que solo será accesible para quién conozca su significado (encriptación).**
- **Actualmente la codificación se utiliza básicamente con el propósito de transmitir información.**
- **En el caso de las computadoras, la información se codifica en forma automática a pulsos eléctricos para ser interpretados por los circuitos electrónicos del sistema.**

CÓDIGOS BINARIOS

Se define como código binario al conjunto palabras binarias (cadenas de bits), generalmente de igual longitud, que responden a leyes y propiedades preestablecidas, con las que puede realizarse una representación unívoca de eventos.

- El origen se remonta a principios del **siglo XIX**, en base a una teoría algebraica desarrollada por el matemático inglés **George Boole**.
- Utilizando eventos y proposiciones de **sólo dos estados** -verdadero y falso-, esta teoría constituye el fundamento de las actuales computadoras digitales.
- Codificar en forma binaria, reproduce la información con una **estructura mucha más extensa** (desventaja), pero esta desventaja se ve ampliamente compensada con su **simplicidad** (ventaja).
- A partir del concepto binario, dependerá de la organización, agrupamiento y **leyes de formación**, la generación de los distintos códigos.

Wikipedia: El código binario es el sistema numérico usado para la representación de textos, o procesadores de instrucciones de computadora, utilizando el sistema binario (sistema numérico de dos dígitos, o dos bits: el "0" /cerrado/ y el "1" /abierto/) ← (puede ser invertido)

ELEMENTOS DE UN CÓDIGO

Código binario natural de 3 bits

Longitud (3 bits)

2^2	2^1	2^0	
			← pesos
0	0	0	← primera palabra
0	0	1	
0	1	0	
0	1	1	Palabras consecutivas
1	0	0	
1	0	1	
1	1	0	
1	1	1	← última palabra

Extensión o capacidad de codificación = 2^n (ej. 3 bits con 8 palabras)

“Posición más significativa” (mayor peso)

ponderado

“Posición menos significativa” (menor peso)

Código Johnson de 4 bits, progresión a izquierda, marca 1

Longitud (4 bits)

0	0	0	0	← primera palabra
0	0	0	1	
0	0	1	1	
0	1	1	1	Palabras consecutivas
1	1	1	1	
1	1	1	0	
1	1	0	0	
1	0	0	0	← última palabra

Extensión o capacidad de codificación = 2*n (ej. 4 bits con 8 palabras)

No ponderado

PROPIEDADES DE LOS CÓDIGOS BINARIOS

- **Códigos binarios cíclicos:** Son aquellos donde la primera y última palabra del código son adyacentes (difieren en 1 bit). **Ej. Gray**.
- **Códigos binarios continuos:** La representación de palabras consecutivas son adyacentes. **Ej. Gray**.
- **Capacidad de codificación:** Cantidad de combinaciones diferentes que forman el código, en relación directa con su ley de formación y la cantidad de bits por palabra. La máxima capacidad de codificación para n bits es 2^n. **Ej. Binario natural**.
- **Distancia:** Es la cantidad bits que deben modificarse para pasar de una palabra a otra cualquiera. **Distancia mínima** es la menor distancia entre cualquier par de elementos del código. **Ej. todos los códigos**.
- **Códigos ponderados:** Cada posición de los bits en una palabra representa un valor o peso numérico determinado (similar a los sistemas numéricos posicionales). **Ej. Binario natural, Aiken, 1 de N**.
- **Códigos no ponderados:** Cada bit dentro de la palabra del código se ubica de acuerdo a una determinada ley de formación, sin que le sea asignada una significación numérica particular. **Ej. Gray, Johnson**.

CLASIFICACIÓN DE LOS CÓDIGOS BINARIOS

- **Según su aplicación**
  - **Códigos numéricos**
    - **Estándares**
    - **BCD**
  - **Códigos alfanuméricos**

- **Según la ley de formación**
  - **Ponderados**
  - **No ponderados**

Ley de formación: reglas o condiciones bajo las cuales se forman las palabras

Nivel de seguridad: condiciones bajo las cuales el código puede detectar o corregir error o no

- **Según el nivel de seguridad**
  - **Sin seguridad**
  - **Detectores de errores**
  - **Correctores de errores**

CÓDIGOS GENERALES TÍPICOS

BINARIO NATURAL

n = 3 bits	
DEC	2^2 2^1 2^0
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

- **Tipo**: Binario numérico.
- ***Ciclo***: No cíclico.
- **Continuidad**: No continuo.
- ***Cap. de codificación***: 2^n.
- ***Distancia mínima***: 1.
- ***Ley de formación***: Ponderado.
- ***Aplicación***: Codificación numérica general

GRAY (O REFLEJADO)

n = 3 bits	
DEC	G_2 G_1 G_0
0	0 0 0
1	0 0 1
2	0 1 1
3	0 1 0
4	1 1 0
5	1 1 1
6	1 0 1
7	1 0 0

línea de reflexión

- **Tipo**: Binario numérico.
- ***Ciclo***: Cíclico.
- ***Continuidad***: Continuo
- ***Cap. de codificación***: 2^n.
- ***Distancia mínima***: 1.
- ***Ley de formación***: No ponderado.
- ***Aplicación***: Álgebra binaria y circuitos lógicos.

JOHNSON (O PROGRESIVO)

n = 4 bits	
DEC	J_3 J_2 J_1 J_0
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 1 1 1
4	1 1 1 1
5	1 1 1 0
6	1 1 0 0
7	1 0 0 0

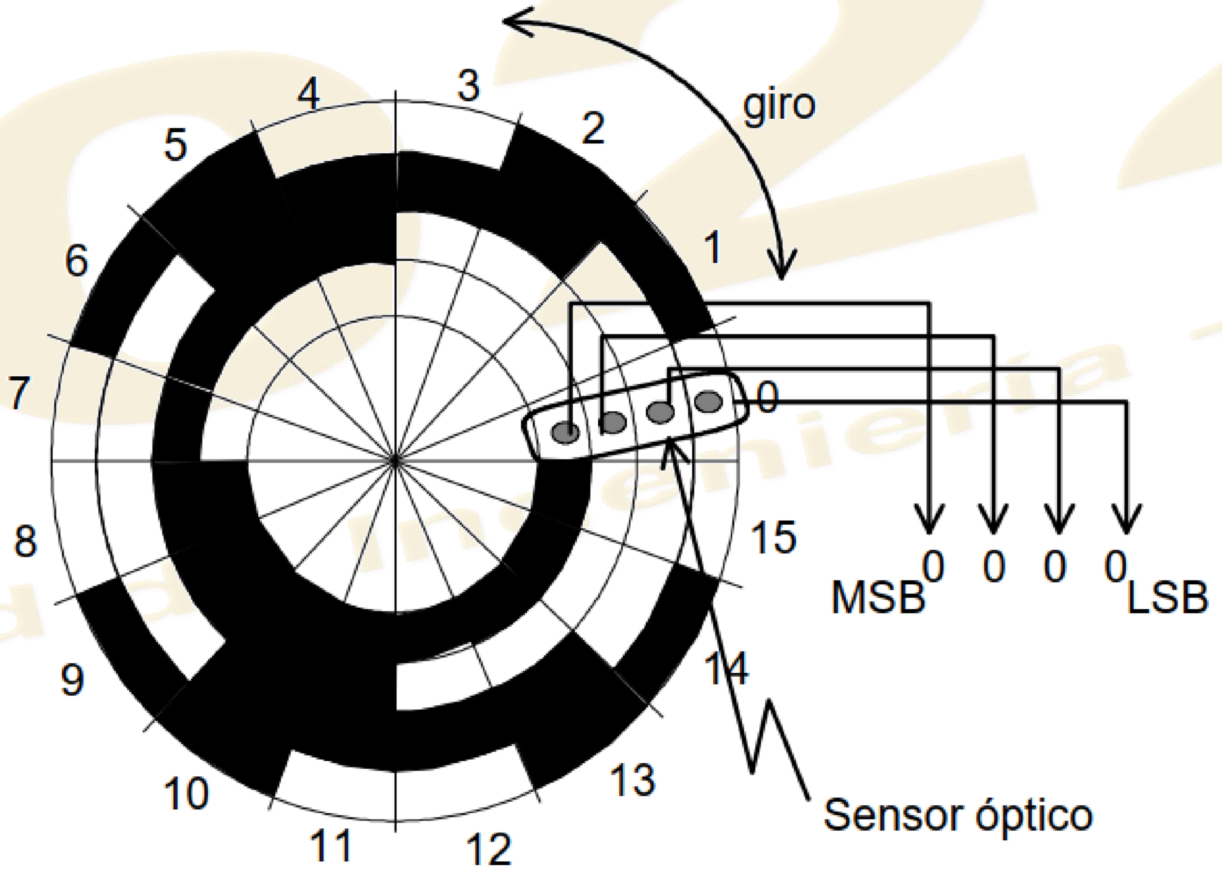
- ***Tipo***: Binario numérico.
- ***Ciclo***: Cíclico.
- ***Continuidad***: Continuo.
- ***Cap. de codificación***: 2*n.
- ***Distancia mínima***: 1.
- ***Ley de formación***: No ponderado, progresión de 1 hacia la izquierda.
- ***Aplicación***: Sistemas de conteo.

CÓDIGOS GENERALES TÍPICOS

CÓDIGO GRAY O REFLEJADO DE 4 BITS

D	G_3	G_2	G_1	G_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

Disco generador de código Gray de cuatro bits. Lectura mediante un conjunto de cuatro sensores ópticos.



Sensor
blanco = 0
negro = 1

CÓDIGOS BCD

(Wakerly, pg. 48; Nelson, pg. 61; Velasco, pg. 64)

- Su denominación proviene de las siglas en inglés ***Binary Coded Decimal*** (Decimal Codificado en Binario).
- Cada dígito decimal se codifica **aisladamente** por un conjunto de bits que respondan a una determinada ley de formación, de modo que se sigue manteniendo la **estructura básica decimal**.
- Sus estructuras resultan **más simples**, ya que solo es necesario considerar cualquier configuración de bits que contengan por lo menos **diez palabras** diferentes para ser asignados a los diez dígitos. La **longitud mínima es de 4 bits**.
- Estos códigos pueden ser ponderados, no ponderados y aún aleatorios, pero siempre es conveniente el empleo de ordenaciones binarias que presenten alguna **ley de formación predecible** o que los relacione con los dígitos que representan.

CÓDIGOS BCD PONDERADOS

BCD NATURAL – Ponderado – Sin seguridad ($d_{mín}$ = 1)										
DEC	0	1	2	3	4	5	6	7	8	9
8421	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

el código más utilizado

genera palabras ambigüas

BCD AIKEN (2421) – Ponderado - Sin seguridad ($d_{mín}$ = 1)										
DEC	0	1	2	3	4	5	6	7	8	9
2421	0000	0001	0010	0011	0100	1011	1100	1101	1110	1111

BCD 5421 - Ponderado - Sin seguridad ($d_{mín}$ = 1)										
DEC	0	1	2	3	4	5	6	7	8	9
5421	0000	0001	0010	0011	0100	1000	1001	1010	1011	1100

genera palabras ambigüas

pesos con valores atípicos

BCD 631-1 - Ponderado - Sin seguridad ($d_{mín}$ = 1)										
DEC	0	1	2	3	4	5	6	7	8	9
631-1	0000	0010	0101	0100	0110	1001	1000	1010	1101	1100

BCD POSICIONAL (Pesos 9876543210)- Ponderado - Con seguridad ($d_{mín}$ = 2)					
DEC	0	1	2	3	4
Peso	0000000001	0000000010	0000000100	0000001000	0000010000
DEC	5	6	7	8	9
Peso	0000100000	0001000000	0010000000	0100000000	1000000000

Codificación más simple, la marca puede ser 1 o 0.

CÓDIGOS BCD NO PONDERADOS

cuatro variantes para el código Johnson

Johnson / progresivo (4 variantes) - No ponderado - Sin seguridad (d_{min} = 1)

DEC	Johnson M1	Johnson M1 Marca 1, progresión a Izquierda	Johnson M0	Johnson M0 Marca 0, progresión a Izquierda	Johnson M1D	Johnson M1D Marca 1, progresión a Derecha	Johnson M0D	Johnson M0D Marca 0, progresión a Derecha
0	0 0 0 0 0		1 1 1 1 1		0 0 0 0 0		1 1 1 1 1	
1	0 0 0 0 ←1		1 1 1 1 ←0		1 → 0 0 0 0		0 → 1 1 1 1	
2	0 0 0 1 1		1 1 1 0 0		1 1 0 0 0		0 0 1 1 1	
3	0 0 1 1 1		1 1 0 0 0		1 1 1 0 0		0 0 0 1 1	
4	0 1 1 1 1		1 0 0 0 0		1 1 1 1 0		0 0 0 0 1	
5	1 1 1 1 1		0 0 0 0 0		1 1 1 1 1		0 0 0 0 0	
6	1 1 1 1 0		0 0 0 0 1		0 1 1 1 1		1 0 0 0 0	
7	1 1 1 0 0		0 0 0 1 1		0 0 1 1 1		1 1 0 0 0	
8	1 1 0 0 0		0 0 1 1 1		0 0 0 1 1		1 1 1 0 0	
9	1 0 0 0 0		0 1 1 1 1		0 0 0 0 1		1 1 1 1 0	

BCD EXCESO 3 – No ponderado – Sin seguridad (d_{min} = 1)

DEC	0	1	2	3	4	5	6	7	8	9
XS-3	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

Existe toda una familia de códigos de exceso. Este es el más conocido

los códigos aleatorios son útiles en casos especiales

BCD ALEATORIO – No ponderado – Sin seguridad (d_{min} = 1)

DEC	0	1	2	3	4	5	6	7	8	9
Aleat	1101	0010	0110	0011	1111	1000	1010	0001	1001	1110

OTRA CODIFICACIÓN DE DÍGITOS

Bajo los mismos principios que los códigos BCD se pueden codificar los símbolos ("dígitos") octales y hexadecimales.

- **El código BCO (*Binary Coded Octal - Octal Codificado en Binario*) se forma con cualquier arreglo de tres bits como mínimo, que permita una capacidad de codificación máxima de $2^3 = 8$ combinaciones. No hay límite máximo establecido.**

Ejemplo

BINARIO CODIFICADO OCTAL NATURAL - Ponderado								
OCTAL	0	1	2	3	4	5	6	7
$2^2\ 2^1\ 2^0$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1

- **Similarmente, el código BCH (*Binary Coded Hexadecimal - Hexadecimal Codificado en Binario*) se forma con una estructura de cuatro bits como mínimo, para lograr las dieciséis palabras que requieren los símbolos hexadecimales. No hay límite máximo.**

Ejemplo

BINARIO CODIFICADO HEXADECIMAL NATURAL – Ponderado								
HEXA	0	1	2	3	4	5	6	7
$2^3\ 2^2\ 2^1\ 2^0$	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
HEXA	8	9	A	B	C	D	E	F
$2^3\ 2^2\ 2^1\ 2^0$	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1

CÓDIGOS ALFANUMÉRICOS

(Velasco, pg. 53)

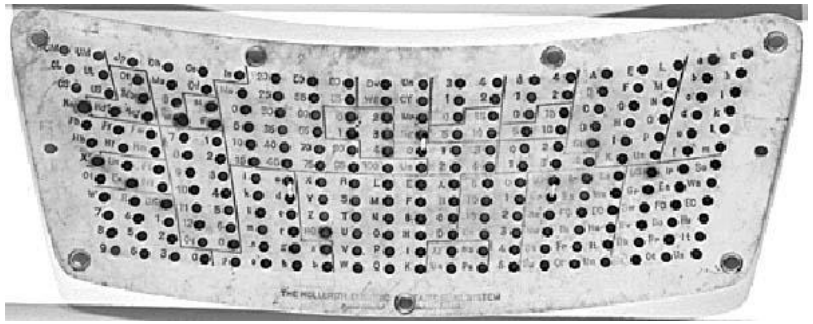
- Al difundirse la actividad informática, se vio la necesidad de extender el campo de la codificación no solo a los números, sino también a los caracteres alfabéticos, signos, símbolos especiales y algunas órdenes simples para los sistemas de computación.
- Debieron ampliarse los códigos binarios para que presenten una mayor cantidad de palabras, con diferentes organizaciones de bits, dando lugar a la generación de los llamados códigos alfanuméricos o códigos de caracteres.
- En un principio, debido a la falta de estandarización, aparecieron una gran cantidad de códigos diversos, que impedía el intercambio de software y hardware entre fabricantes.
- Una natural evolución en este aspecto, ha demostrado que una estandarización en la codificación de la información resulta mucho más beneficiosa tanto para los fabricantes y programadores como para los usuarios.

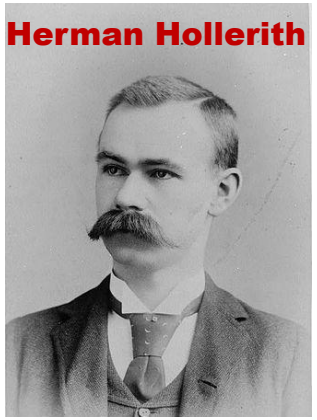
CÓDIGOS ALFANUMÉRICOS TÍPICOS

CÓDIGO HOLLERITH							
CAR	109876543210	CAR	109876543210	CAR	109876543210	CAR	109876543210
esp	000000000000	0	000000000001	C	100000001000	O	010001000000
@	000100010000	1	000000000010	D	100000010000	P	010010000000
#	000100001000	2	000000000100	E	100000100000	Q	010100000000
$	010100001000	3	000000001000	F	100001000000	R	011000000000
&	100000000000	4	000000010000	G	100010000000	S	000000000101
*	010100010000	5	000000100000	H	100100000000	T	000000001001
–	010000000000	6	000001000000	I	101000000000	U	000000010001
.	100100001000	7	000010000000	J	010000000010	V	000000100001
/	000000000011	8	000100000000	K	010000000100	W	000001000001
‘	000100001001	9	001000000000	L	010000001000	X	000010000001
may	001001000000	A	100000000010	M	010000010000	Y	000100000001
min	101001000000	B	100000000100	N	010000100000	Z	001000000001

Código utilizado para representar letras, números o símbolos especiales, sobre tarjetas perforadas estándar de 80 columnas.
Herman Hollerith desarrolló la tecnología de procesamiento de tarjetas perforadas de datos para el censo de los Estados Unidos de América de 1890.
Fundó la compañía Tabulating Machine Company (1895) la cual fue una de las tres compañías que se unieron para formar la Computing Tabulating Recording Corporation (CTR), luego renombrada IBM.

***Tipo*: Alfanumérico para tarjetas perforadas.**
***Ciclo*: No cíclico.**
***Continuidad*: No continuo**
***Capacidad de codificación*: 4096 limitado.**
***Distancia mínima*: 1.**
***Ley de formación*: Ponderado.**
***Aplicación*: Ingreso de datos (obsoleto).**





CÓDIGOS ALFANUMÉRICOS TÍPICOS

CÓDIGO ASCII								
6 5 4	6 5 4	6 5 4	6 5 4	6 5 4	6 5 4	6 5 4	6 5 4	bits
0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	3 2 1 0
NUL	DEL	SP	0	@	P	`	p	0 0 0 0
SOH	DC1	!	1	A	Q	a	q	0 0 0 1
STX	DC2	"	2	B	R	b	r	0 0 1 0
ETX	DC3	#	3	C	S	c	s	0 0 1 1
EOT	DC4	$	4	D	T	d	t	0 1 0 0
ENQ	NAK	%	5	E	U	e	u	0 1 0 1
ACK	SYN	&	6	F	V	f	v	0 1 1 0
BEL	ETB	'	7	G	W	g	w	0 1 1 1
BS	CAN	(	8	H	X	h	x	1 0 0 0
HT	EM	)	9	I	Y	i	y	1 0 0 1
LF	SUB	*	:	J	Z	j	z	1 0 1 0
VT	ESC	+	;	K	[	k	{	1 0 1 1
FF	FS	,	<	L	\	l	|	1 1 0 0
CR	GS	-	=	M	]	m	}	1 1 0 1
SO	RS	.	>	N	^	n	~	1 1 1 0
SI	US	/	?	O	_	o	DEL	1 1 1 1

- **Tipo: Alfanumérico.**
- **Ciclo: No cíclico.**
- **Continuidad: No continuo.**
- **Capacidad de codificación: 128.**
- **Distancia mínima: 1.**
- **Ley de formación: No ponderado.**
- **Aplicación: Entrada-salida de datos en sistemas de computación personales.**

ASCII

Su nombre es acrónimo de American Standard Code for Information Interchange o Código Estándar Estadounidense para el Intercambio de Información.

En 1963 la American Standards Association aprobó el lanzamiento de este código, sentando las bases de lo que sería el futuro de los dispositivos electrónicos a la hora de representar los diversos caracteres en los sistemas de computación.

Es un código originariamente de 7 bits empleado en pequeños sistemas de computación para traducir caracteres del teclado al lenguaje del sistema. Posteriormente ampliado y se utiliza en la mayoría de los sistemas tipo PC.

CÓDIGOS ALFANUMÉRICOS TÍPICOS

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F
0000 0	NUL 0	DLE 16	DS 32	48	SP 64	& 80	- 96	112	128	144	160	176	{ 192	} 208	\ 224	0 240
0001 1	SOH 1	DCI 17	SOS 33	49	65	/ 81	97	113	a 129	j 145	161	177	A 193	J 209	225	1 241
0010 2	STX 2	DC2 18	FS 34	SYN 50	66	82	98	114	b 130	k 146	s 162	178	B 194	K 210	S 226	2 242
0011 3	ETX 3	TM 19	35	51	67	83	99	115	c 131	l 147	t 163	179	C 195	L 211	T 227	3 243
0100 4	PF 4	RES 20	BYP 36	PN 52	68	84	100	116	d 132	m 148	u 164	180	D 196	M 212	U 228	4 244
0101 5	HT 5	NL 21	LF 37	RS 53	69	85	101	117	e 133	n 149	v 165	181	E 197	N 213	V 229	5 245
0110 6	LC 6	BS 22	ETB 38	UC 54	70	86	102	118	f 134	o 150	w 166	182	F 198	O 214	W 230	6 246
0111 7	DEL 7	IL 23	ESC 39	EOT 55	71	87	103	119	g 135	p 151	x 167	183	G 199	P 215	X 231	7 247
1000 8	8	CAN 24	40	56	72	88	104	120	h 136	q 152	y 168	184	H 200	Q 216	Y 232	8 248
1001 9	RLF 9	EM 25	41	57	73	89	105	\ 121	i 137	r 153	z 169	185	I 201	R 217	Z 233	9 249
1010 A	SMM 10	CC 26	SM 42	58	cent 74	! 90	| 106	: 122	138	154	170	186	202	218	234	250
1011 B	VT 11	CU1 27	CU2 43	CU3 59	. 75	$ 91	, 107	# 123	139	155	171	187	203	219	235	251
1100 C	FF 12	IFS 28	44	DC4 60	< 76	* 92	% 108	@ 124	140	156	172	188	204	220	236	252
1101 D	CR 13	IGS 29	ENQ 45	NAK 61	(77	) 93	_ 109	' 125	141	157	173	189	205	221	237	253
1110 E	SO 14	IRS 30	ACK 46	62	+ 78	; 94	> 110	= 126	142	158	174	190	206	222	238	254
1111 F	SI 15	IUS 31	BEL 47	SUB 63	| 79	¬ 95	? 111	" 127	143	159	175	191	207	223	239	255

EBCDIC (Extended Binary Coded Decimal Interchange Code) es un código estándar de 8 bits usado por computadoras mainframe IBM.

IBM adaptó el EBCDIC del código de tarjetas perforadas en los años 1960.

- **Tipo: Alfanumérico.**
- **Ciclo: No cíclico.**
- **Continuidad: No continuo.**
- **Capacidad de codificación: 256.**
- **Distancia mínima: 1.**
- **Ley de formación: No ponderado.**
- **Aplicación: Entrada-salida de datos en sistemas de computación IBM.**

CÓDIGOS ALFANUMÉRICOS TÍPICOS

ISO-8859-15															
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	¡	¢	£	€	¥	Š	§	š	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	Ž	µ	¶	·	ž	¹	º	»	Œ	œ	Ÿ	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

ISO 8859-15 o Latin-9
es la parte 15 de ISO 8859, un estándar de codificación de caracteres definido por la Organización Internacional para la Estandarización (S.I.O.).

Codifica en 8 bits el alfabeto y otros caracteres importantes para almacenar textos en inglés, francés, alemán, español y portugués en ordenadores.

Leyenda	
NUL	Código de control C0
PAD	Código de control C1
!	Carácter idéntico en ISO 8859-1 y en ISO 8859-15
€	Carácter en ISO 8859-15 diferente del que se encuentra en la misma posición en ISO-8859-1

CÓDIGOS DETECTORES DE ERRORES

(Wakerly, 58; Nelson, pg. 65)

Para que un código binario pueda detectar y/o corregir errores de transmisión, debe tener características especiales, generalmente basadas en la distancia mínima

Teorema: **En un código binario pueden detectarse errores en *n-1* bits por palabra, si y solo si su distancia mínima es por lo menos *n*.**

- **En el manejo y transmisión de información codificada, es posible que se produzcan errores debido a la presencia de ruido.**
- **Estos errores se traducen en el cambio de estado de uno o más bits de la información, lo que produce una recepción de datos distintos a los que fueron emitidos.**
- **Cuando un código binario utiliza todas las combinaciones posibles de su estructura, no es factible la detección de errores pues una alteración en una palabra produce otra palabra que también pertenece al código.**
- **Cuando no se emplean todas las combinaciones posibles, es más factible detectar un error, siempre y cuando el error produzca un elemento no perteneciente al código.**

CÓDIGOS DETECTORES DE ERRORES

- **El criterio general para la detección de errores consiste en incorporar algún tipo de redundancia en la información que lleva cada palabra del código, de este modo, si una parte del elemento se ve afectada durante la transmisión, utilizando la información redundante sería posible detectar la presencia del error.**

$$\text{Redundancia} = 1 - \frac{\text{Bits de información}}{\text{Total de bits}}$$

$$\text{Eficiencia} = \frac{\text{Bits de información}}{\text{Total de bits}}$$

Redundancia + Eficiencia = 1 (ó 100%)

Ejemplos

Código de 7 bits + bit de paridad
- **Redundancia = 12,5%**
- **Eficiencia = 87,5%**

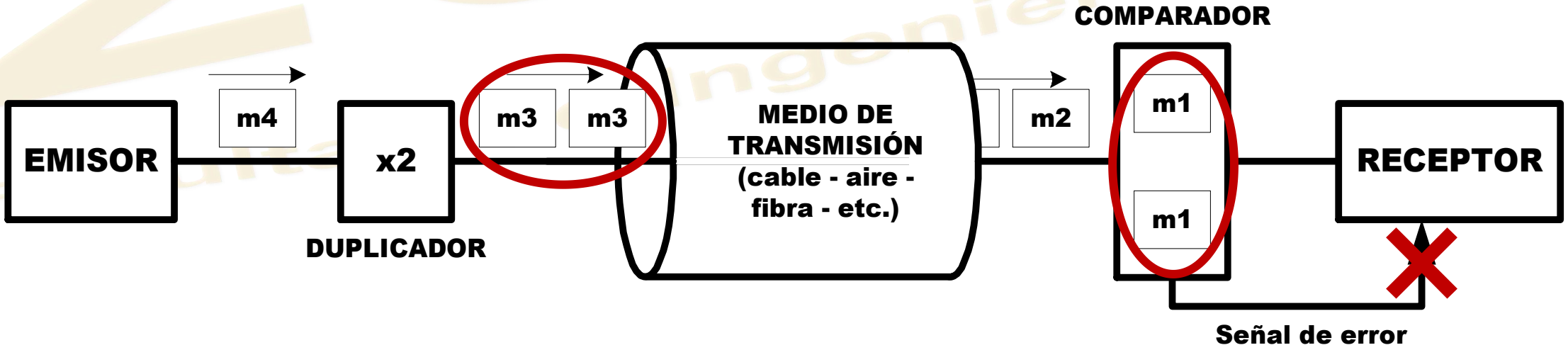
Código de 4 bits + 3 bits de paridad
- **Redundancia = 42,9%**
- **Eficiencia = 57,1%**

MÉTODOS DE DETECCIÓN DE ERRORES

RETRANSMISIÓN

Consiste en transmitir dos veces cada palabra del código o todo el mensaje. Si en ambos conjuntos hay diferencias se ha detectado el error. Sería prácticamente imposible que se produjese el mismo error en el mismo lugar para ambas transmisiones.

La ventaja es la simplicidad del proceso para detectar errores. Las desventajas son evidentes. Este método presenta una redundancia del 100% y una eficiencia nula.



MÉTODOS DE DETECCIÓN DE ERRORES

USO DE CÓDIGOS M DE N

- Son códigos de N bits de longitud con M (< N) bits fijos en un estado.
- Su distancia mínima es **mayor que la unidad**, pero no representan estructuras muy eficientes.
- Son capaces de detectar una **cantidad impar** de errores; cuando el número de errores en un elemento sea par, pasarán inadvertidos.
- Presentan una **baja capacidad de codificación**, a menos que se utilicen muchos bits y longitudes grandes.
- La cantidad de palabras se calculan como ----> N! / (M! x (N-M)!)

Ejemplo

BCD 2 DE 5 – Ponderado - d_{min} = 2 – Cap. de codificación = 10 (limitado)										
DEC	0	1	2	3	4	5	6	7	8	9
01236	01100	11000	10100	10010	01010	00110	10001	01001	00101	00011

$$\underbrace{N = 5}_{\text{longitud código}} \quad \underbrace{M = 2}_{\text{marcas}} \longrightarrow \frac{5!}{2! \times (5-2)!} = \underbrace{10}_{\text{cantidad palabras}}$$

MÉTODOS DE DETECCIÓN DE ERRORES

BITS DE PARIDAD

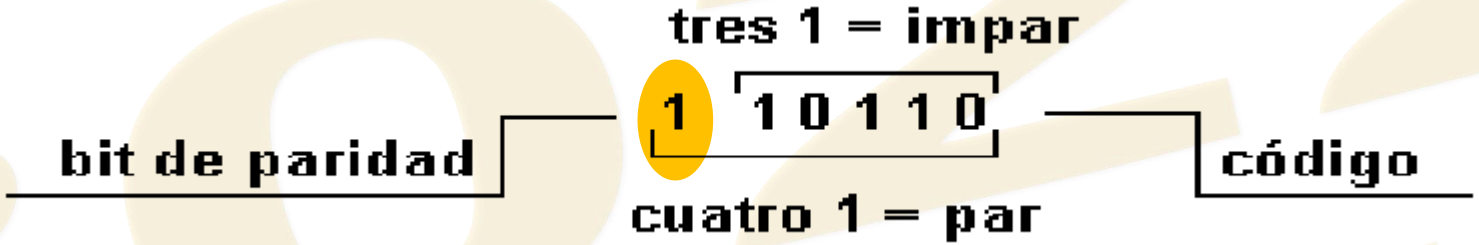
(Nelson, pg. 66)

- Es una técnica muy eficiente empleada para aumentar la distancia en un código.
- Consiste en agregar a cada palabra del código, en posiciones determinadas, uno o más bits adicionales que incorporan información redundante.
- Se puede aplicar en dos variantes: paridad par o paridad impar.
- Cada bit de paridad se encarga de mantener constante la paridad de 1´s en la palabra del código. Si se produce un error de transmisión y cambia cualquier bit de la palabra, la paridad no coincide y el error se detecta.
- Una misma palabra puede tener más de un bit de paridad que controle la totalidad de los bits o una parte de ellos.

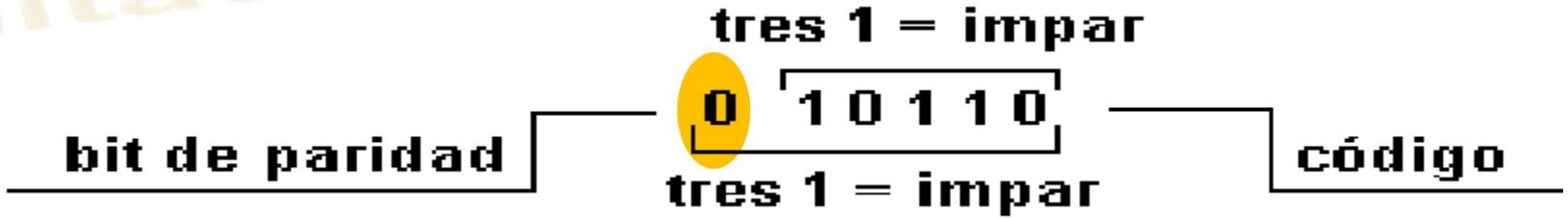
MÉTODOS DE DETECCIÓN DE ERRORES

BITS DE PARIDAD (BdeP)

- **Paridad par (PP):** El bit toma valor 0, si la cantidad de 1´s de la correspondiente palabra del código es par –toma valor 1 en caso contrario–, de modo que siempre se transmitan una cantidad par de 1´s.



- **Paridad impar (PI):** El bit toma valor 0, si la cantidad de 1´s de la correspondiente palabra del código es impar –toma valor 1 en caso contrario–, de modo que siempre se transmitan una cantidad impar de 1´s.

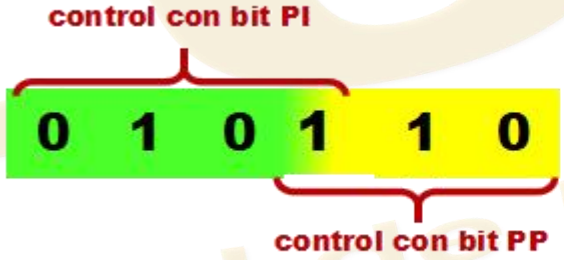


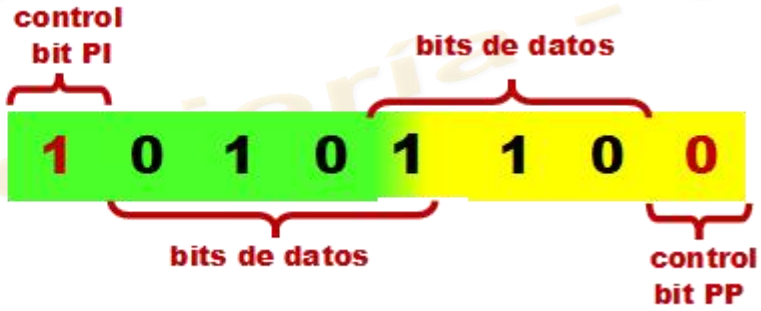
MÉTODOS DE DETECCIÓN DE ERRORES

BITS DE PARIDAD (BdeP)

- **Cantidad:** En una misma palabra de código, se pueden incorporar tantos BdeP como sea conveniente. Se pueden combinar los BdeP par con BdeP impar

- **Control:** Cada BdeP puede controlar un grupo de bits de datos, según criterio del usuario. Inclusive el control puede solaparse entre los grupos.





- **Posición:** Cada BdeP se puede puede ubicar en cualquier lugar de la palabra binaria.

MÉTODOS DE DETECCIÓN DE ERRORES

Ejemplo:

Código BCD Exceso 3 con bit de paridad par ubicado como prefijo.

BCD XS-3 CON BIT DE PARIDAD PAR					
DEC	BPP	$X_3X_2X_1X_0$	DEC	BPP	$X_3X_2X_1X_0$
0	0	0 0 1 1	5	1	1 0 0 0
1	1	0 1 0 0	6	0	1 0 0 1
2	0	0 1 0 1	7	0	1 0 1 0
3	0	0 1 1 0	8	1	1 0 1 1
4	1	0 1 1 1	9	0	1 1 0 0

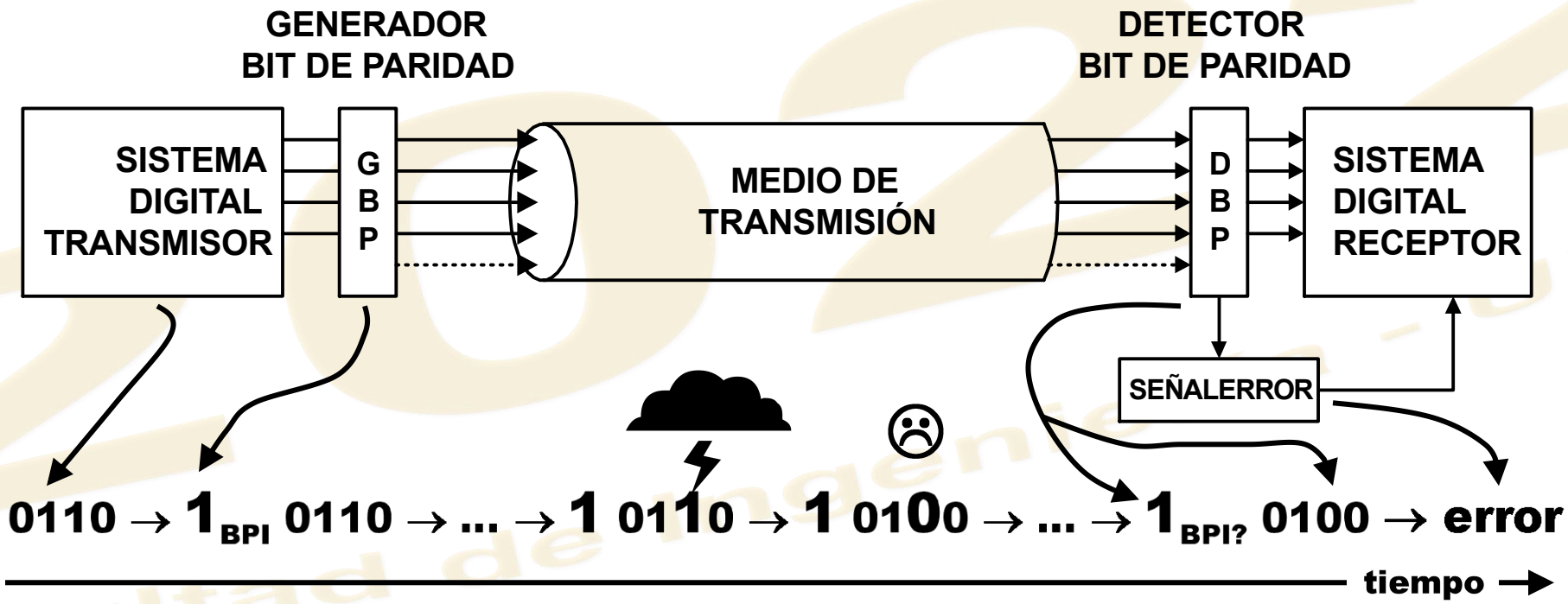
Ejemplo:

Código BCD Johnson con bit de paridad impar ubicado como sufijo.

BCD JOHNSON CON BIT DE PARIDAD IMPAR					
DEC	$J_4J_3J_2J_1J_0$	BPI	DEC	$J_4J_3J_2J_1J_0$	BPI
0	0 0 0 0 0	1	5	1 1 1 1 1	0
1	0 0 0 0 1	0	6	1 1 1 1 0	1
2	0 0 0 1 1	1	7	1 1 1 0 0	0
3	0 0 1 1 1	0	8	1 1 0 0 0	1
4	0 1 1 1 1	1	9	1 0 0 0 0	0

MÉTODOS DE DETECCIÓN DE ERRORES

Esquema de un sistema físico de transmisión de datos protegido con bit de paridad.



Esta disposición permitiría únicamente la detección de las palabras de código con un error, aunque no puede precisar exactamente cuál es el bit erróneo, por lo tanto no se tiene la posibilidad de corregirlos. Para ello se emplean otros procedimientos más sofisticados.

MÉTODOS DE CORRECCIÓN DE ERRORES

(Wakerly, pg. 61; Nelson, pg. 65)

Para que un código binario pueda detectar y/o corregir errores de transmisión, debe tener características especiales, generalmente basadas en la distancia mínima

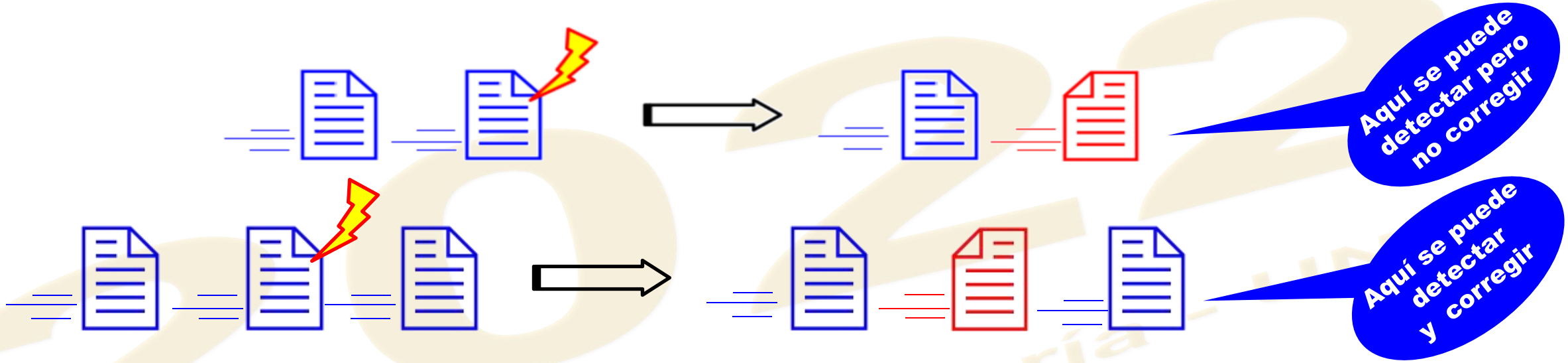
Teorema: **En un código binario pueden detectarse y corregirse errores en *n* bits por palabra, si y solo si su distancia mínima es por lo menos *2*n+1*.**

- **Los códigos correctores se emplean en la transmisión de datos donde no es posible retransmitir cuando se detecta un error, como por ejemplo en los sistemas que trabajan en tiempo real o sistemas de lectura o medición de variables en procesos industriales.**
- **Estos códigos no sólo detectan una condición de error. sino que además proporcionan la suficiente información para determinar cuál o cuáles son los bits erróneos y por consiguiente pueden efectuar la corrección, simplemente invirtiéndolos (por ser binarios).**
- **Actúan con un criterio similar a los códigos detectores de errores, pero requieren mayor redundancia.**

MÉTODOS DE CORRECCIÓN DE ERRORES

RETRANSMISIÓN

Consiste en transmitir tres veces cada palabra del mensaje o todo el mensaje.



- **Si en los tres mensajes (o en cada trio de datos) no hay diferencias, significa que no hay error.**
- **Si hay un mensaje (o una palabra del trío) diferente, hay error y el mensaje o palabra correcta es alguna del duplo que no haya cambiado.**
- **Con este método se puede detectar y corregir error en un solo bit por palabra.**

La ventaja es la simplicidad del proceso para detectar y corregir errores. Las desventajas son evidentes. Este método presenta una redundancia del 100% y una eficiencia nula.

MÉTODOS DE CORRECCIÓN DE ERRORES

Corrección de 1 error con código de $d_{min} = 3$

transmitido	*recibido*	*corrección*	*valor final*
000 111 000 pertenece al código	⇨ error en 1 bit 000 111 010 no pertenece al código	⇨ X_7=000 111 000 pertenece al código	→ elemento más cercano 000 111 000 pertenece al código
		⇨ pertenece al código X_3=000 111 111	

Imposibilidad de corrección de 2 errores con código de $d_{min} = 3$

transmitido	*recibido*	*corrección*	*valor final*
000 111 000 pertenece al código	⇨ error en 2 bit 000 111 101 no pertenece al código	⇨ X_7=000 111 000 pertenece al código	el elemento más cercano es
		→ ?	*indeterminado*
		⇨ pertenece al código X_3=000 111 111	no se sabe que se han producido dos errores

MÉTODOS DE CORRECCIÓN DE ERRORES

Códigos Hamming

(Nelson, pg. 68; Wakerly, pg. 61)

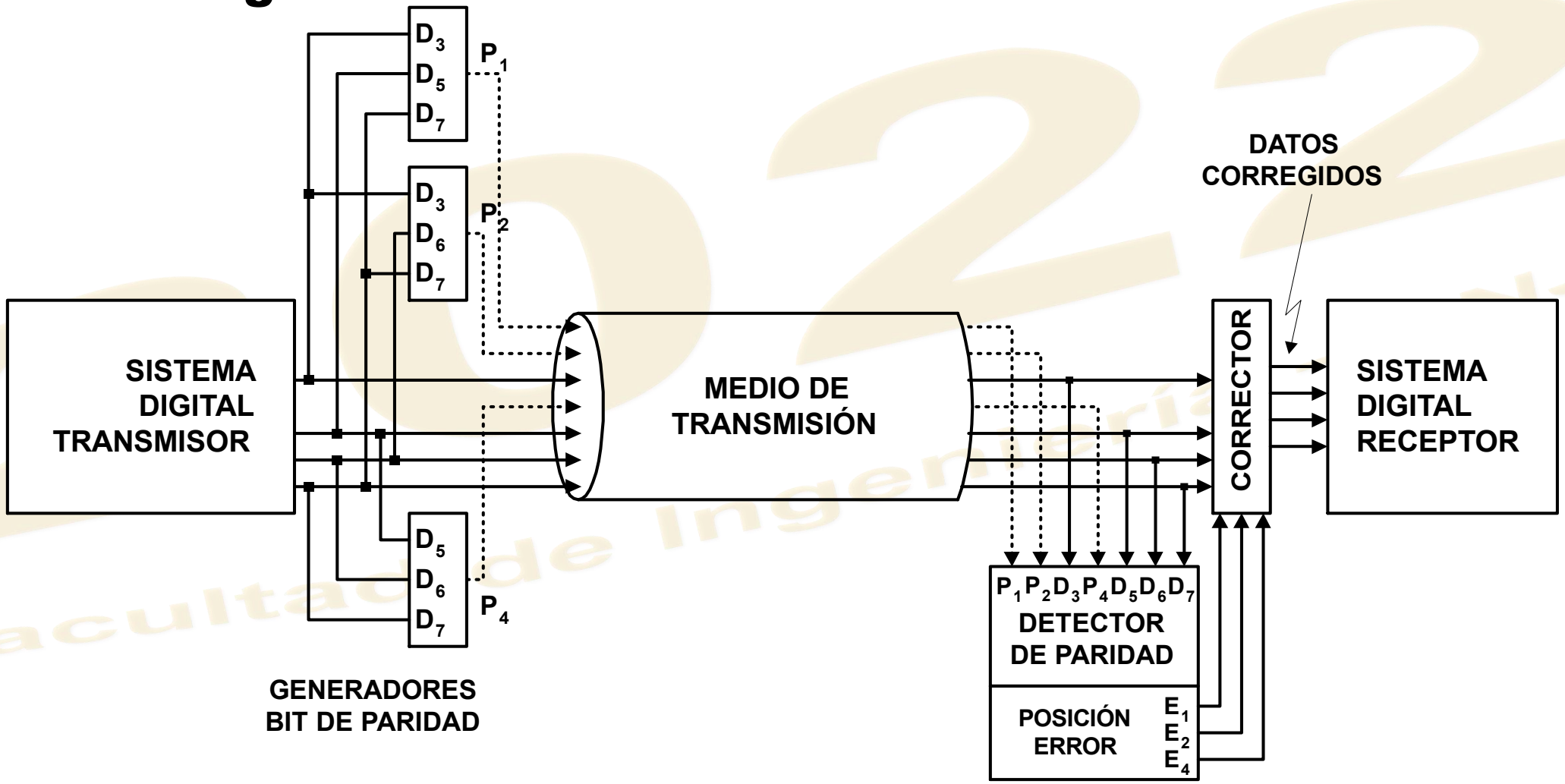
- Según la teoría de Hamming, *para la detección y corrección de errores en un bit en una palabra de un código de n bits, de distancia unitaria, es necesario adicionar p bits, de tal forma que se cumpla la relación*

$$2^p \geq n + p + 1$$

- Con los p bits adicionales se obtiene un nuevo código de n+p bits. Por ej., con notación genérica H(n+p,n), se pueden obtener los códigos Hamming H(7,4), H(15,11), H(31,26).
- En este nuevo código se realizan p detecciones de paridad en los bits del código original. Luego, en el extremo receptor, la comprobación de los códigos de paridad con sus respectivas líneas, generan un número binario cuyo equivalente decimal indica la posición del bit erróneo.
- La corrección se realiza invirtiendo el bit con error identificado por el sistema.

MÉTODOS DE CORRECCIÓN DE ERRORES

Esquema de transmisión–recepción de información protegida por Hamming



MÉTODOS DE CORRECCIÓN DE ERRORES

Generación de código Hamming H(7,4) para código base de 4 bits

Palabra del código base (n = 4)

Genérica	Ejemplo
$(D_7\ D_6\ D_5\ D_3)_2$ →	$(1\ 1\ 0\ 1)_2$

Requiere 3 bits de protección (p = 3)

$P_4\ P_2\ P_1$ →	$P_4\ P_2\ P_1$

los bits P_i serán producidos por los circuitos generadores de paridad del extremo emisor.

Cumple con la relación de Hamming

$2^p \geq n + p + 1$ →	$2^3 = 4 + 3 + 1$

Organización de la protección

P_1 paridad de → $D_7\ D_5\ D_3$	→ $P_1 = 1$ p. impar de → 1 0 1
P_2 paridad de → $D_7\ D_6\ D_3$	→ $P_2 = 0$ p. impar de → 1 1 1
P_4 paridad de → $D_7\ D_6\ D_5$	→ $P_4 = 1$ p. impar de → 1 1 0

el tipo de paridad puede ser cualquiera (par o impar), pero debe ser conocida por emisor y receptor.

Palabra del nuevo código (n + p = 7)

$(D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1)_2$ →	$(1\ 1\ 0\ 1\ 1\ 0\ 1)_2$

la ubicación de los bits de paridad en la palabra es una cuestión formal, pueden estar en cualquier posición, pero debe ser conocida por emisor y receptor.

MÉTODOS DE CORRECCIÓN DE ERRORES

Control de recepción de de código Hamming H(7,4)

Palabra enviada

Genérica	Ejemplo
$(D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1)_2$ →	→ $(1\ 1\ 0\ 1\ 1\ 0\ 1)_2$

Palabra recibida sin error

$(D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1)_2$ →	→ $(1\ 1\ 0\ 1\ 1\ 0\ 1)_2$

Control de paridad en la recepción

P_1 para → $D_7\ D_5\ D_3$ → E_1 →	→ $P_1 = 1$ para → 101 → $E_1 = 0$
P_2 para → $D_7\ D_6\ D_3$ → E_2 →	→ $P_2 = 0$ para → 111 → $E_2 = 0$
P_4 para → $D_7\ D_6\ D_5$ → E_4 →	→ $P_4 = 1$ para → 110 → $E_4 = 0$

los bits E_i generados por los detectores de paridad del receptor, indicarían error ($E_i = 1$) o sin error ($E_i = 0$) en cada grupo parcial de bits.

Palabra recibida con error (por ejemplo en bit D_3)

$(D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1)_2$ →	→ $(1\ 1\ 0\ 1\ 0\ 0\ 1)_2$

Control de paridad en la recepción (paridad impar)

P_1 para → $D_7\ D_5\ D_3$ → E_1 →	→ $P_1 = 1$ para → 100 → $E_1 = 1$
P_2 para → $D_7\ D_6\ D_3$ → E_2 →	→ $P_2 = 0$ para → 110 → $E_2 = 1$
P_4 para → $D_7\ D_6\ D_5$ → E_4 →	→ $P_4 = 1$ para → 110 → $E_4 = 0$

la equivalencia decimal de los bits de error ($E_4\ E_2\ E_1$) indican la posición del bit recibido con error. En este caso ($E_4\ E_2\ E_1$) = $011_2 = 3_{10}$ ⇒ error en bit 3 de la palabra.