Circuitos Digitais I
Eletrônica Digital Combinacional

Fernando Passold, Dr. Eng.
Profesor Titular III
Engenharia Elétrica
E-mail: fpassold@upf.br

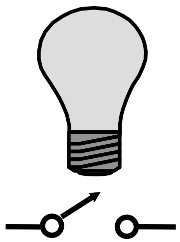
Introdução

- Sistemas Numéricos
  - Códigos Digitais
  - Sistemas numéricos mais utilizados
  - Formação dos sistemas numéricos
  - Conversão entre diferentes bases numéricas
  - Otros códigos binários (Gray, ASCII)
  - Números binários (inteiros) com sinal
    - Notação con sinal, complemento 1 y 2.
    - Soma e subtração em complemento 2

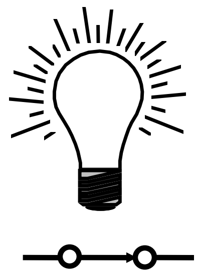
- Álgebra de Boole
  - Portas Lógicas básicas (tabela verdade, símbolo, equações)
  - Análises
  - Propriedades de Álgebra de Boole
  - Minimização de funções algébricas (e de circuitos combinacionais digitais – 1ª parte)

Introdução

Mundo Digital

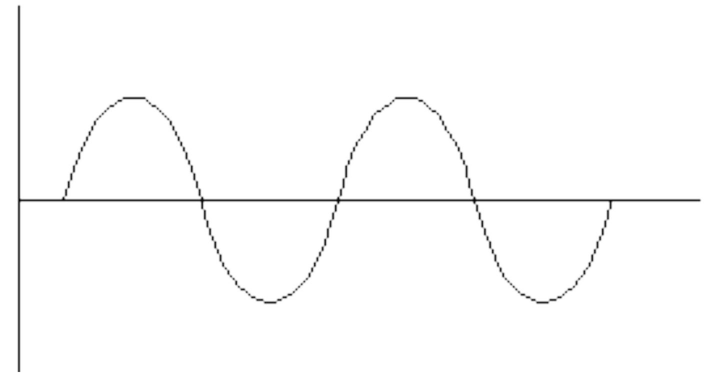


Desligado
Falso
Não existe
"**0**"

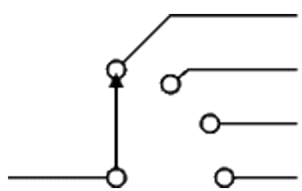


Ligado
Verdade
Existe
"**1**"

Mundo Analógico



Semáforo



valores discretos

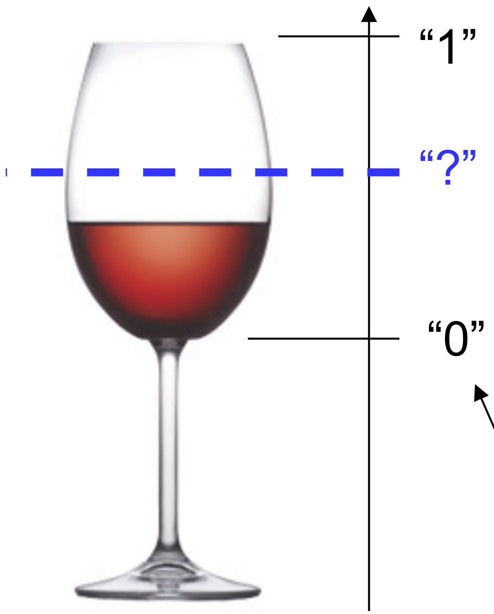
Infinitos estados;

Qualquer valor;

Valor en qualquer instante de tempo.

Introdução

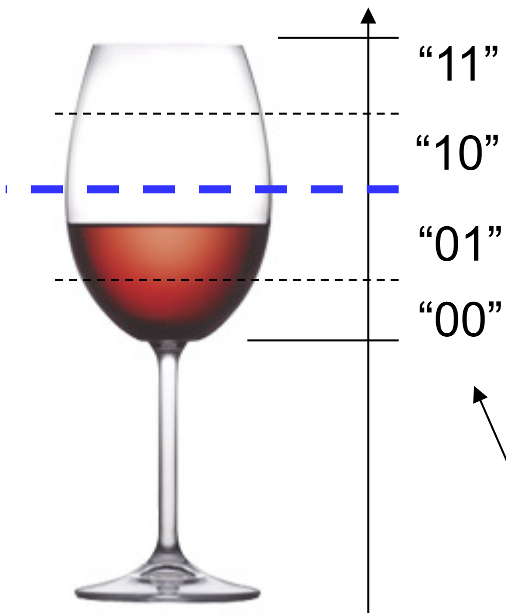
- Estados binários: "0" <u>ou</u> "1":



Note: usando apenas 1 bit!

Introdução

- Estados binários: "0" ou "1":

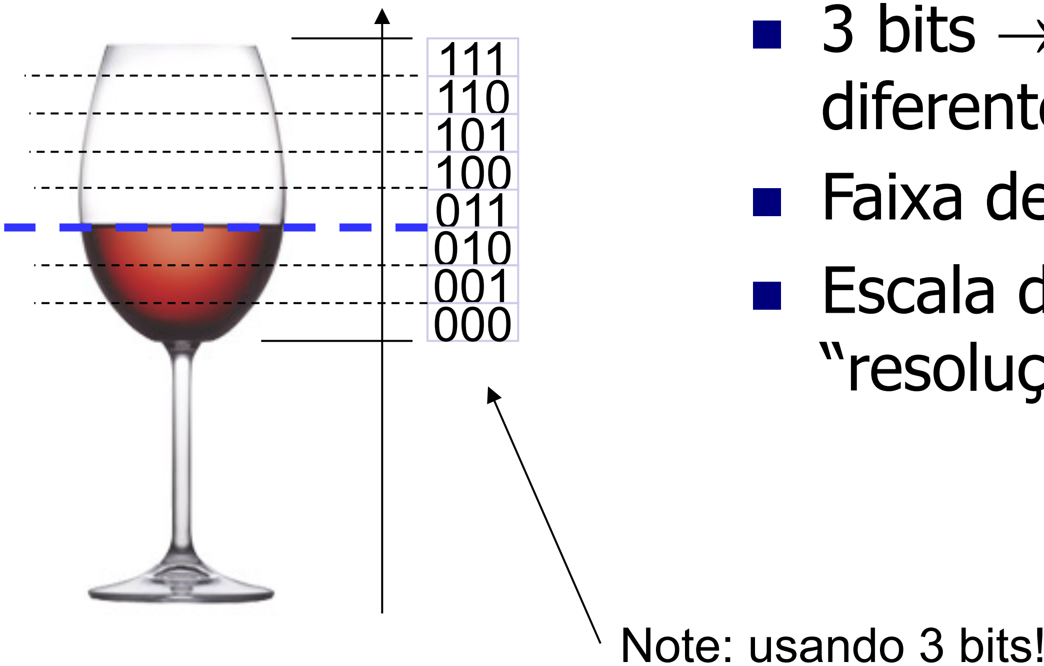


Note: usando 2 bits!

- Proceso de Digitalização de Sinais:
- Problemas de Quantização y Codificação

Introdução

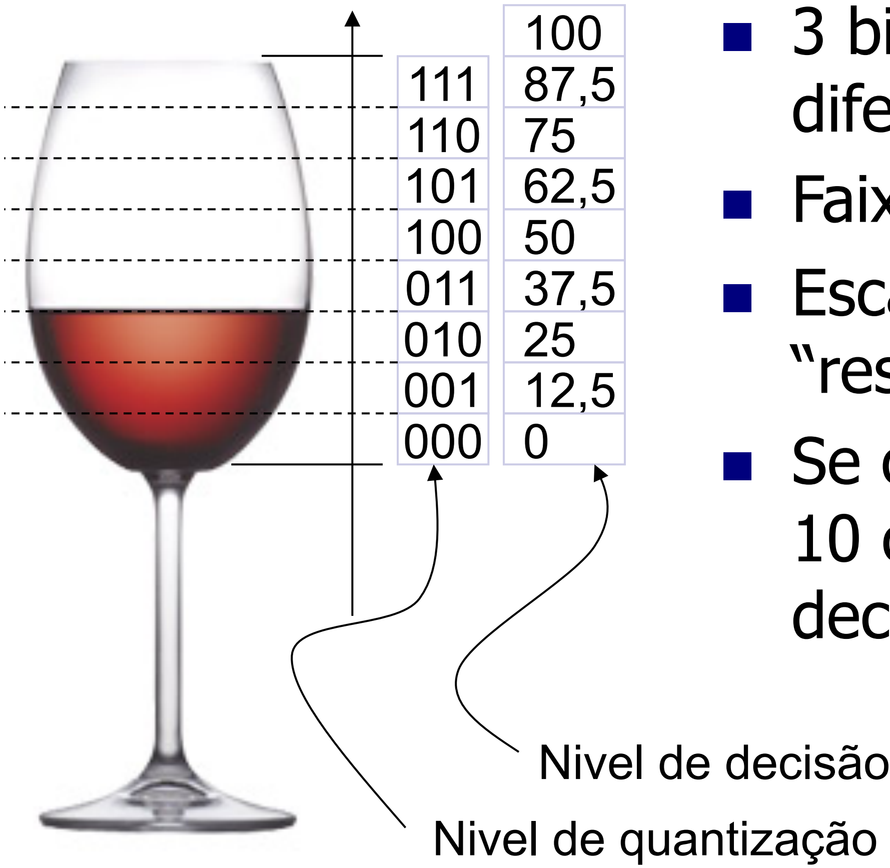
- Estados binários: "0" ou "1":



  - 3 bits → 2^3 = 8 valores diferentes;
  - Faixa de excursão: 0 ~ 100%:
  - Escala de quantização, ou "resolução": 100/8 = 12,5

Introdução

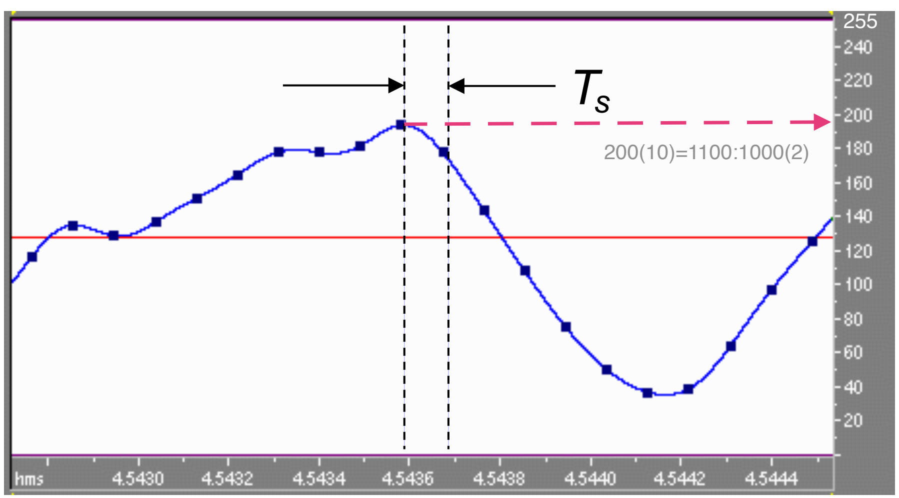
- Estados binários: “0” ou “1”:



- 3 bits → 2^3 = 8 valores diferentes;
- Faixa de excursão: 0 ~ 100%:
- Escala de quantização, ou “resolução”: 100/8 = 12,5
- Se o valor real é 35 → erro de: -10 o +2,5 (depende do nivel de decisão).

Introdução

- Como quantizar valores de tensão negativos ?
  - Também existem várias formas.
- O exemplo seguinte mostra o caso para arquivos digitais de aúdio em formato ***.WAV com 8 bits**:



Note: o sinal é "fatiado" (**amostrado** ou **discretizado**) no tempo.

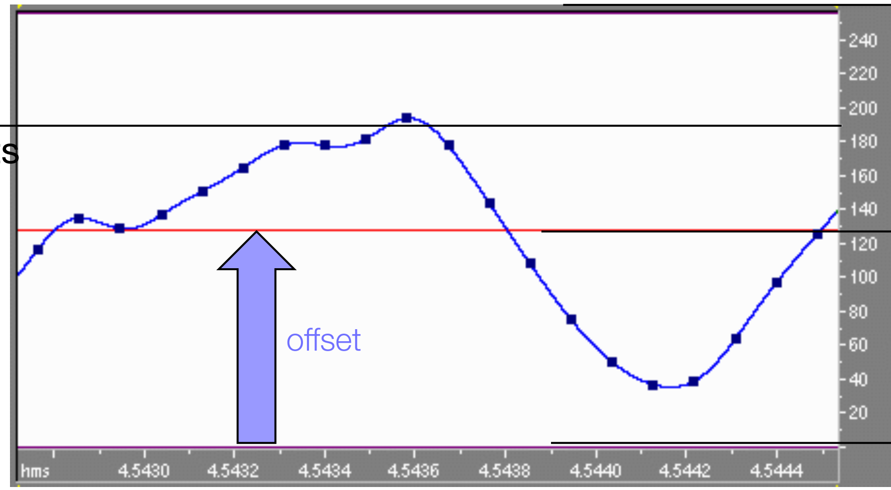
T_S: período de amostragem.

Obs.: O teorema de amostragem de Nyquist ou Shanon determina valores adequados para T_S (não estudado nesta disciplina, faltam fundamentos de Laplace e Diagramas Espectrais).

Arquivo digital de áudio em formato ***.WAV com 8 bits**:

$11111111_{(2)}$ = Vmáx

195 = $10000000_{(2)}$ = 0 Volts



128 = $10000000_{(2)}$ = 0 Volts

$00000000_{(2)}$ = Vmín

- O eixo vertical da figura é graduado no valor das amostras quantizadas com 8 bits : 0 a 255 (2^8).
- Note que o eixo de tensão, 0 Volts, sofre um "**offset**" (deslocamento) para o código (número): 128.
  - Pode-se assim representar valores negativos de tensão sem necesidade de usar código binário com sinal. A forma de onda quantizada acima, no formato decimal é : 118,135,130,138,151,165,179,179,182,195,179,144,109,78,51,37,39,62,97,123. (← unsigned char em "C")
- O que representa os seguintes valores quantizados de tensão (em Volts), supondo que Vmáx=255 Volt e que Vmín= Volts:
  - -10,+7,+2,+10,+23,+37,+51,+51,+54,+67,+51,+16,-19,-50,-77,-91,-89,-66,-31,-5 (Volts)

Isto é:
$255_{(10)}$ = FFh → 255 Volts
Cada "1" → 1 Volt, mas... Não esquecer "offset", então:

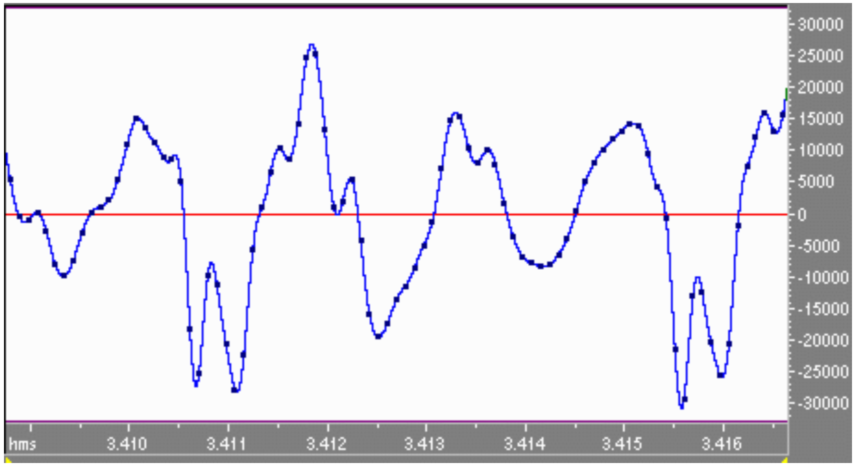
Arquivo de áudio digital PCM em formato ***.WAV de 16 bits**

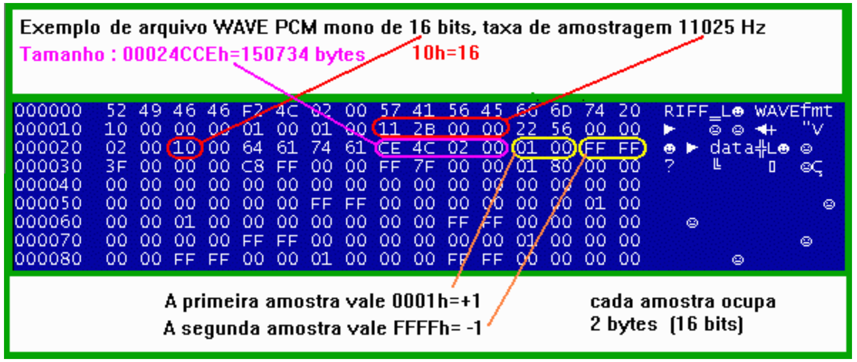
- Um arquivo de áudio digital PCM no formato ***.WAV de 16 bits** usa condificação com **sinal-complemento de 2**.

- Valores positivos são codificados de 0000h=0 até 7FFFh=+32767 e valores negativos são codificados de FFFFh=-1 até 8001h=-32767. O zero é codificado como: 0000H=0.

- A primera figura representa esta codificação (eixo vertical):

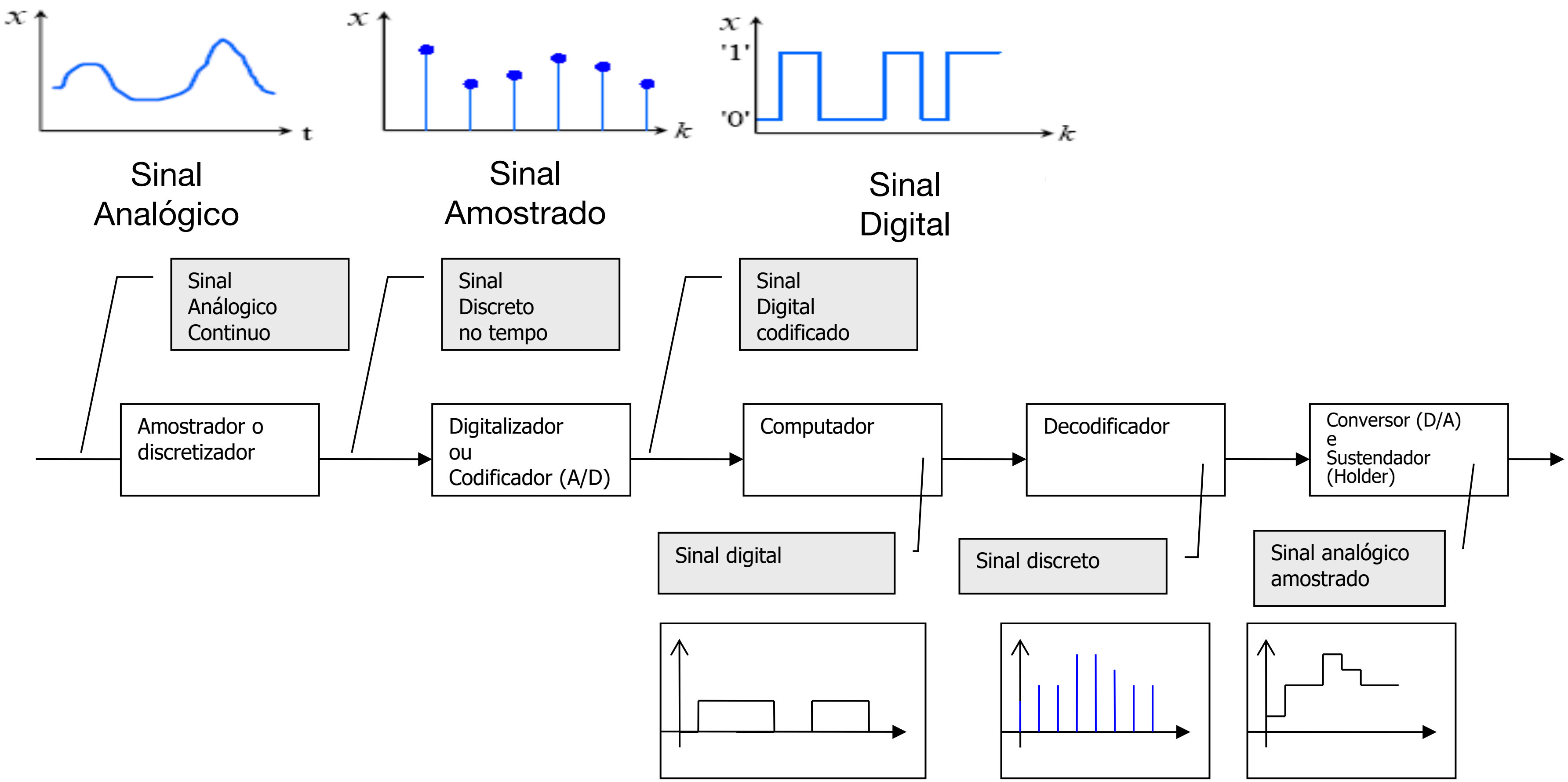
- A segunda figura representa a parte inicial de um arquivo *.WAV de 16 bits.



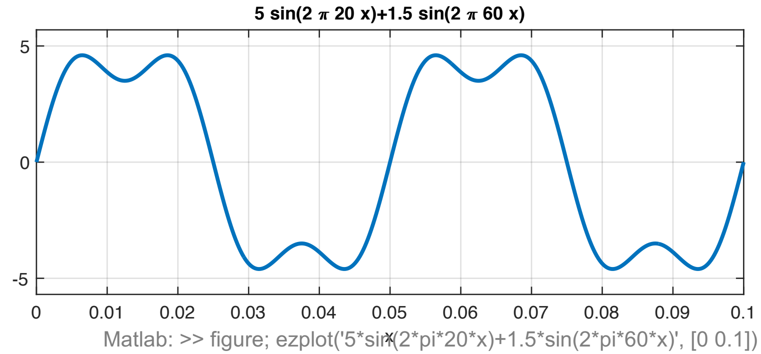


Obs.: Cada Conjunto de 8-bits => 1 byte.

Digitalização de um sinal...

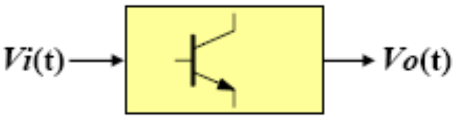


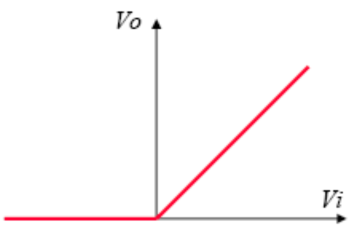
Sinais Contínuos x Discretos

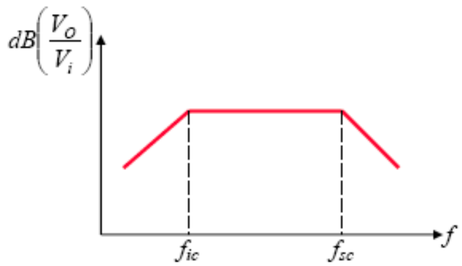


Matlab: >> figure; ezplot('5*sin(2*pi*20*x)+1.5*sin(2*pi*60*x)', [0 0.1])

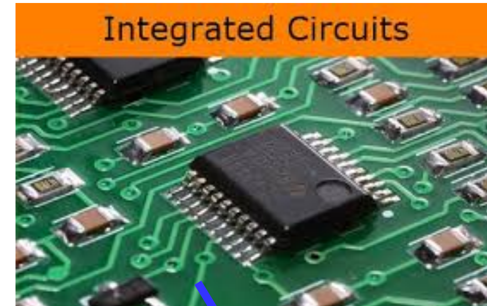
- Sinal Analógico:
  - Ex.: $y(t) = 5\sin(2\pi 20t) + 1,5\sin(2\pi 60t)$
  - Eletrônica analógica: transístores na faixa linear, β.
  - Modelo matemático (forma de caracterizar):
    - Função transferência: $V_o(t) = f(V_i(t))$
    - Equação diferencial (modelo do circuito).
    - Função de transferência no plano-s: $\frac{V_o(s)}{V_i(s)} = H(s)$
    - Resposta em frequência: $|H(jw)|$



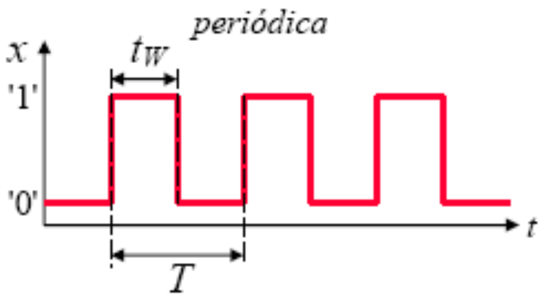


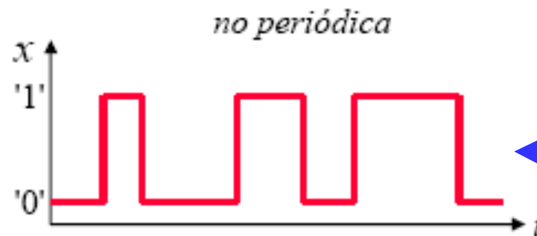


Sinais Contínuos x Discretos



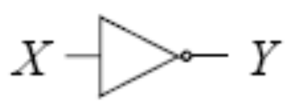
- Sinal digital (binário): trem de pulsos:



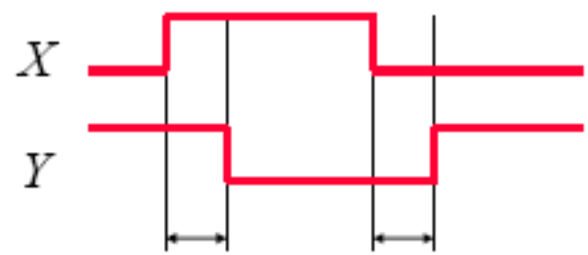


Cada trilha

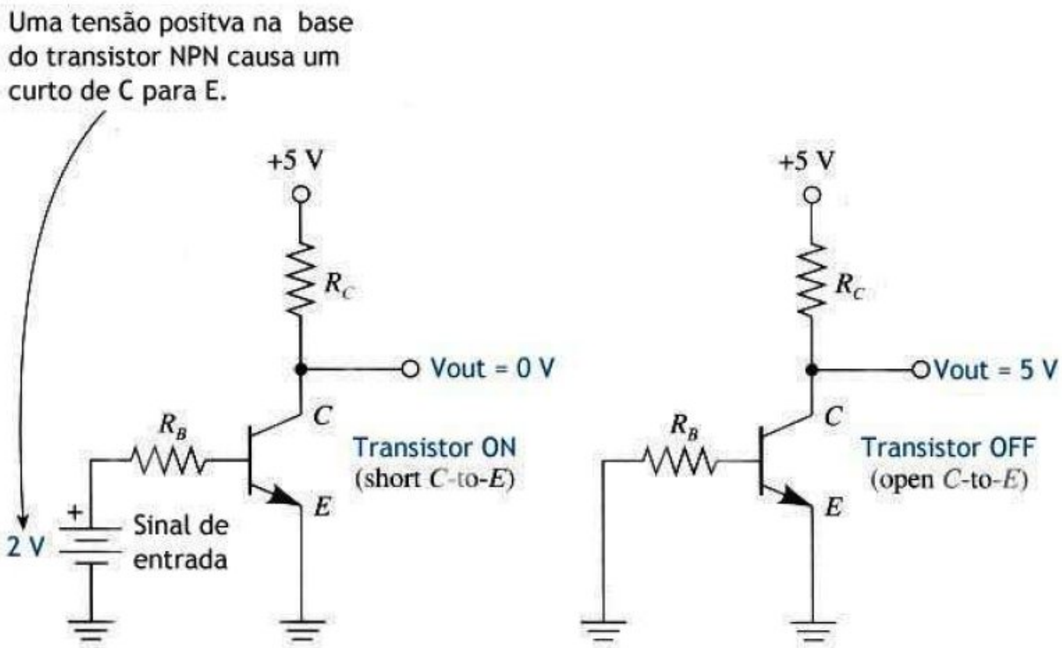
- Transistor ➔ interruptor (corte/saturação)
- Caracterização:
  - Tabelas verdade
  - Diagramas de transição



X	Y
0	1
1	0



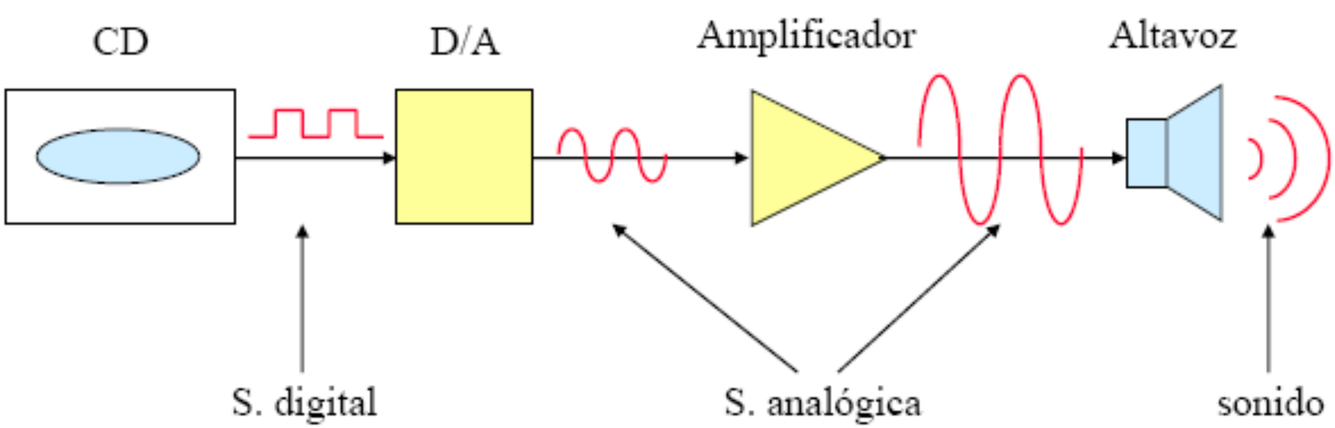
Transistor operando como interruptor



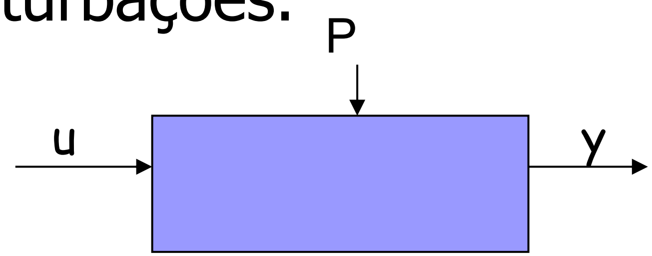
1. Em um transistor PNP, se uma tensão positiva for aplicada entre sua base e o emissor, a junção coletor-emissor torna-se um circuito fechado (pode-se dizer que o "transistor foi ligado" ou que está "**saturado**").
2. Aplicar uma tensão negativa ou nula (0 V) entre a base e o emissor, abre a junção coletor-emissor (pode-se dizer que o "transistor foi desligado" ou que está em estado de "**corte**").

Sistema misto (analógico-digital)

- Circuitos de conversão:
  - Conversores A/D e D/A.
- Exemplo:
  - Reprodutor de CD
  - Placa de áudio de PC



Sistemas Digitais.

- A teoria de Sistemas permite descrever um sistema atravéz de diagramas, o que permite ver suas diferentes partes, para formular um modelo matemático que relaciona 3 tipos de variables: entradas, saídas y perturbações.
- 

$y = f(u,P)$
- No caso de Sistemas Digitais, sistemas criados pelo homem, os modelos se aproximam bastante mais que em outros sistemas físicos reais.

Vantagens dos sistemas Digitais

- 1. Sistemas digitais são generalmente mais fáceis de projectar.
  - Porque os circuitos utilizados são circuitos de comutação ($0 \rightleftarrows 1$), onde o importante não é tratar de forma exata os valores de tensão e corrente (como em eletrônica analógica), mas apenas a faixa que alcançan (nível lógico "Alto" ou "Baixo": "HIGH" or "LOW").
- 2. Guardar (estocar) informação é (bem) mas fácil.
  - Isso é possível graças a circuitos de comutação especiais (biestáveis) que podem guardar informação e mantê-las por tanto tempo quanto seja necessário (sistemas digitais sequenciais: digitiais II). Em alguns casos, necessitando energia (memórias voláteis), em outros nem isso.
- 3. Exatidão e precisão melhores.
  - Sistemas digitaiss podem manipular tantos digitos de precisão cuanto se necessite simplemente se adicionando mais circuitos de comutação (aumentando os bits). Em sistemas analógicos, a precisão está geralmente limitada a 3 ou 4 dígitos porque os valores de tensão e corrente dependen diretamente (da qualidade e) dos componentes do circuito e estos ainda são afetados por perturbações aleatórias (ruído, temperatura → "drift").

Sistemas numéricos binarios mais usados:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binário

octal

decimal

hexadecimal

- Base binária: 2 símbolos: "0" e "1"
- Base decimal: 10 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9.
- Base octal: 8 símbolos: 0, 1, 2, 3, 4, 5, 6, 7.
- Base hexadecimal: 16 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Formação de um sistema numérico

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binaria

octal

decimal

hexadecimal

$$N_b = a_{m-1} \times b^{m-1} + \ldots + a_0 \times b^0 + a_{-1} \times b^{-1} + \ldots + a_{-n} \times b^{-n}$$

$$N_b = \sum_{i=-n}^{m-1} a_i \times b^i$$

Onde:

N_b = número na base deseada;

a = símbolo que compõe o número na base desejada;

m e *n* = exponentes, índices, ou "posição" do símbolo dentro do número que está sendo formando.

Formação de um sistema numérico

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binaria
octal
decimal
hexadecimal

$$N_b = a_{m-1} \times b^{m-1} + \ldots + a_0 \times b^0 + a_{-1} \times b^{-1} + \ldots + a_{-n} \times b^{-n}$$

- Ex$_1$) $1042_{(10)} = 1 \times 10^3 + 4 \times 10^1 + 2 \times 10^0$
- Ex$_2$) $176_{(8)} = 1 \times 8^2 + 7 \times 8^1 + 6 \times 8^0$

$$= 64 + 56 + 6$$

$$= 126$$

Formação de um sistema numérico

- Ex_3: $100110_{2} \rightarrow ?_{10}$
  Utilizando a equação geral: $$N_b = \sum_{i=-n}^{m-1} a_i \times b^i$$

5	4	3	2	1	0	← "posição" ("índice")
1	0	0	1	1	0	(2)

$= 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 32 + 0 + 0 + 4 + 2 + 0$

$= 38_{(10)}$

Formação de um sistema numérico

$$N_b = \sum_{i=-n}^{m-1} a_i \times b^i$$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binaria

octal

decimal

hexadecimal

- Ex$_4$:

$$\begin{array}{l} 3\ \ 2\ \ 1\ \ 0,-1\,-2\,-3 \\ 1\,0\,1\,1\,,\,0\,1\,1_{\,(2)} \end{array} = 1\times 2^3 + 1\times 2^1 + 1\times 2^0 + 1\times 2^{-2} + 1\times 2^{-3}$$

$$= 8 + 2 + 1 + 0{,}25 + 0{,}125$$

$$11{,}375$$

Formação de um sistema numérico

$$N_b = \sum_{i=-n}^{m-1} a_i \times b^i$$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binaria
octal
decimal
hexadecimal

- $Ex_{5:}$ $453_{(5)}$=?
- $Ex_{6:}$ 3456H = $3456_{(16)}$ = $?_{(10)}$ ← unsigned char a=0x3456;

Formação de um sistema numérico

$$N_b = \sum_{i=-n}^{m-1} a_i \times b^i$$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binaria

octal

decimal

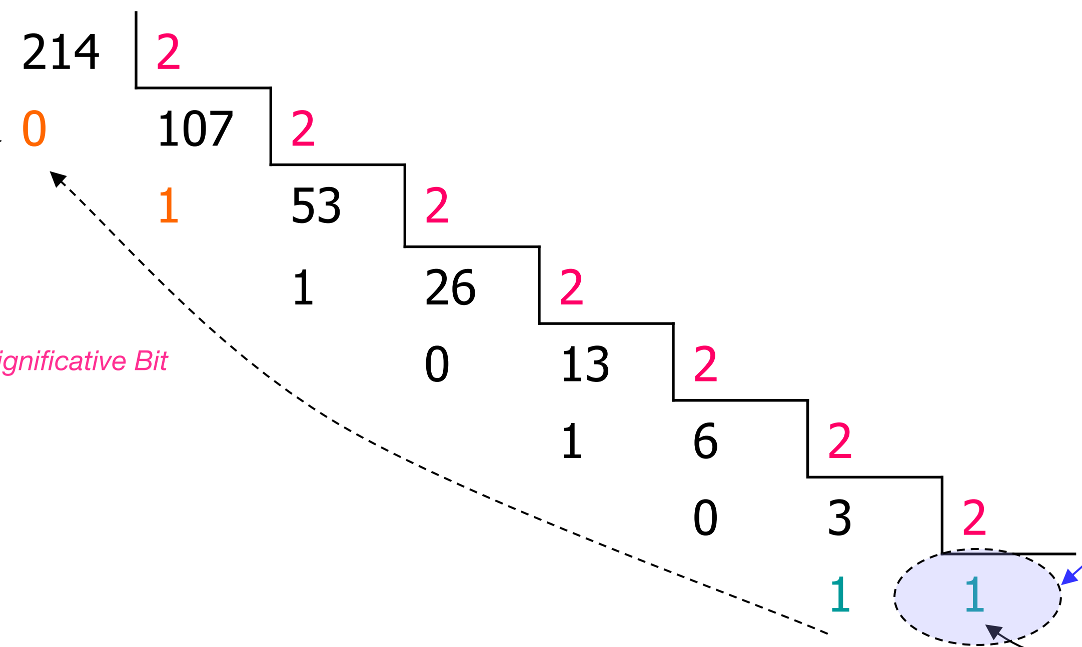
hexadecimal

- $Ex_{5:}$ $453_{(5)}$=?
- $Ex_{6:}$ 3456H = $3456_{(16)}$ = $?_{(10)}$ ← unsigned char a=0x3456;

Resposta: $13398_{(10)}$

Conversão de Códigos

- Método da divisão:
  - Ex$_7$: $214_{(10)} = ?_{(2)} \quad = \quad 11010110_{(2)}$



$$N_b = \sum_{i=-n}^{m-1} a_i \times b^i$$

LSB

=Least Significative Bit

Obs.: A divisão continua até o resultado atingir um número dentro dos símbolos das base de destino

MSB *=Most Significative Bit*

Conversão de Códigos

- **Método de la división:**
  - Ex_8: $214_{(10)} = ?_{(16)} = D6H$

$$
\begin{array}{r|l}
214 & 16 \\ \hline
-208 & 13 \\ \hline
6 &
\end{array}
$$

LSB

MSB

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

binaria

octal

decimal

hexadecimal

Conversão de Códigos

- Ex_9: $117_{(10)} = ?_{(5)} \quad = \quad 432_{(5)}$

Conversão de Códigos

- Outro método: por aproximações sucessivas...
- Exemplo: $113_{(10)} = ?_{(2)}$

Note:

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$\Rightarrow$

$113 - 2^6 = 49$

$49 - 2^5 = 17$

$17 - 2^4 = 1$

1×2^0

$\Rightarrow$

6	5	4	3	2	1	0
1	1	1	0	0	0	1

Conversão de Códigos

=1x2^2+1x2^0
= 4 + 1
= 5

- De binário à octal

$$\underline{10}\ \underline{110}\ \underline{010}\ \underline{110}\ \underline{101}_{(2)}$$

$$= 2 \quad 6 \quad 2 \quad 6 \quad 5_{(8)}$$

- De octal à binário

$$\underline{5}\ \underline{2}\ \underline{7}\ \underline{3}_{(8)}$$

$$= 101\ 010\ 111\ 011_{(2)}$$

- De binario à hexadecimal

$$\underline{1101}\ \underline{0111}\ \underline{1110}\ \underline{0111}\ \underline{0101}_{(2)}$$

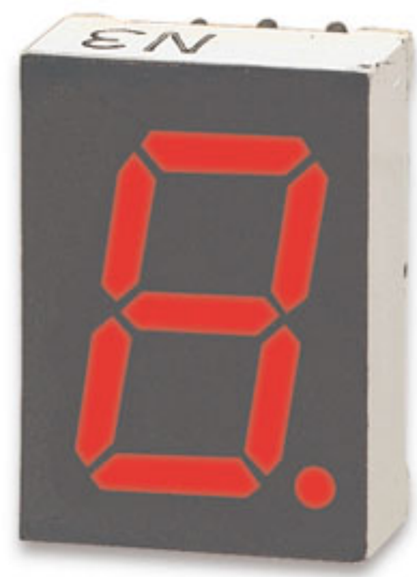
$$= D \quad 7 \quad E \quad 7 \quad 5_{(16)}$$

Note: cada conjunto de 3 bits → 1 símbolo octal;
cada conjunto de 4 bits → 1 símbolo hexadecimal.

Código BCD (Binary Code Decimal)

Usa sempre 4 bits para representar os números: 0,1,2, ... , 8,9

BCD	decimal	BCD	decimal
0000	0	0101	5
0001	1	0110	6
0010	2	0111	7
0011	3	1000	8
0100	4	1001	9

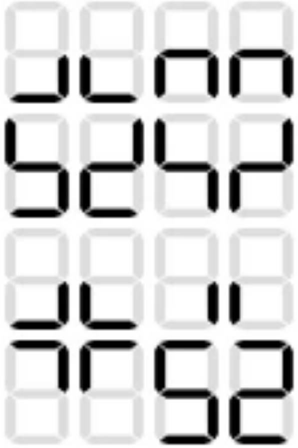


7 Segments Display

Código BCD (Binary code decimal)

- Ex_1: Número $137_{(10)} = ?_{(2,BCD)}$

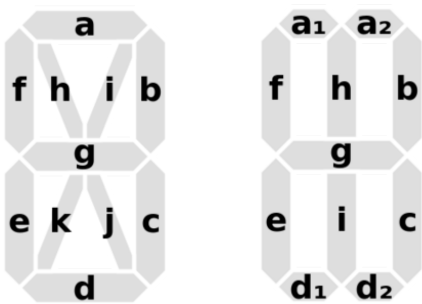
$137_{(10)}$ =		1000	1001	(binário)
$137_{(10)}$ =	0001	0011	0111	(BCD) *
	↓	↓	↓	
	1	3	7	

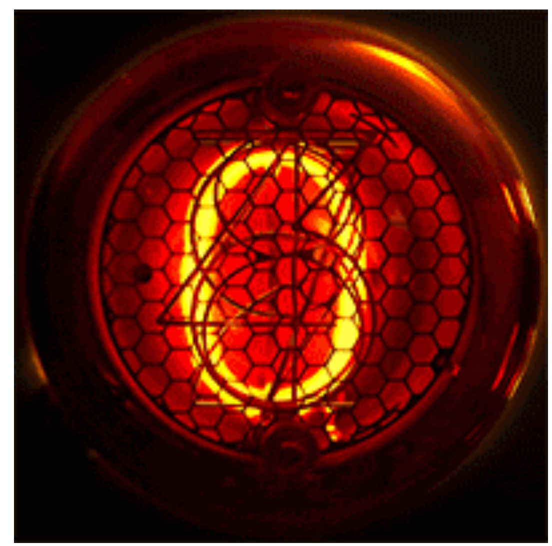


* Note que o código BCD exigiu 12 bits (dígitos),
Enquanto que em binário puro teriam sido "gastos" soment 8 bits
para representar este número.

Outros Displays

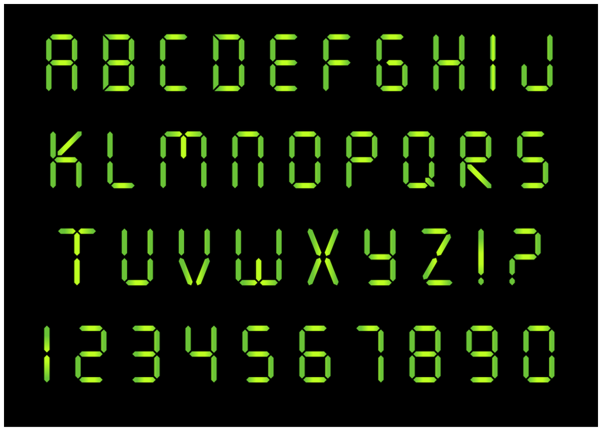
- 11 segmentos:

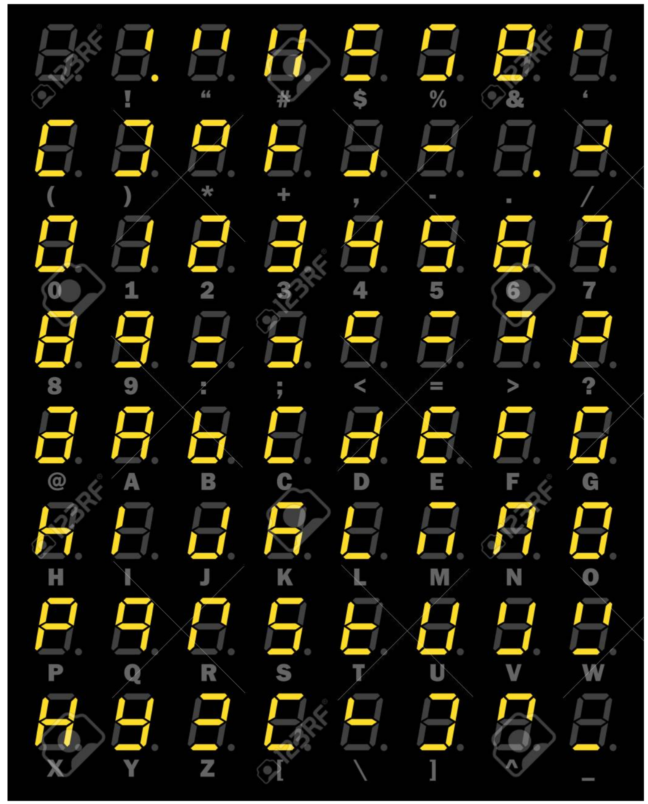


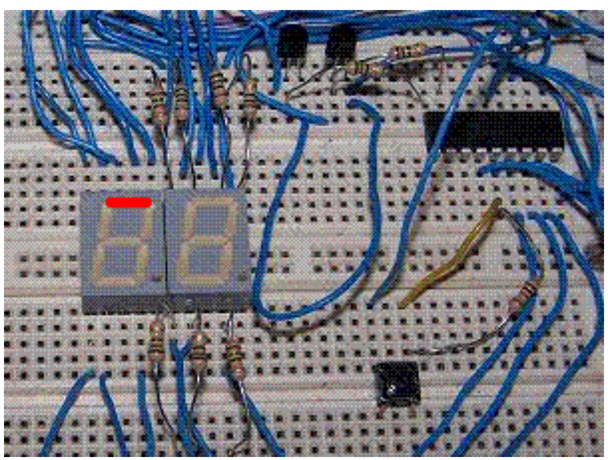


- 14 segmentos:

Outros usos para Displays:







Códigos não ponderados.

- Código excesso-3: valor + 3

  Propiedade : seu complemento 9 é o mesmo como seu complemento lógico.

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

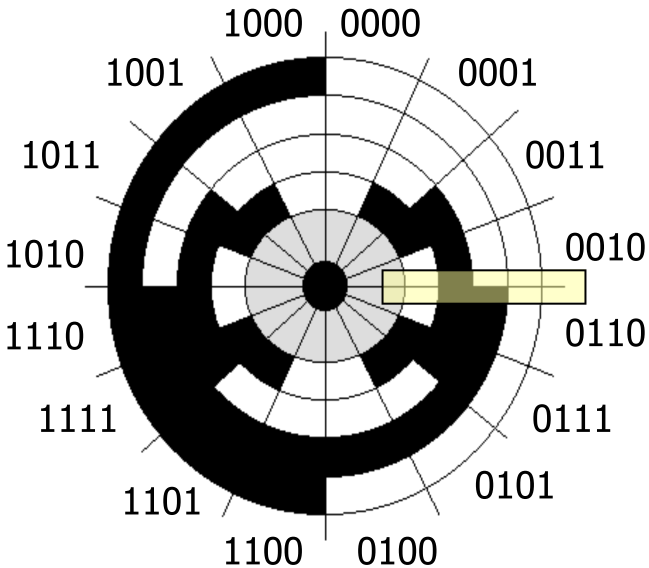
Código Gray

- Código Gray: clase especial de códigos de "distância um".

"Distância um": diferença de um único bit com a palabra código próximo ou adjacente independiente da direção.

Código gray	decimal
000	0
001	1
011	2
010	3
110	4
111	5
101	6
100	7

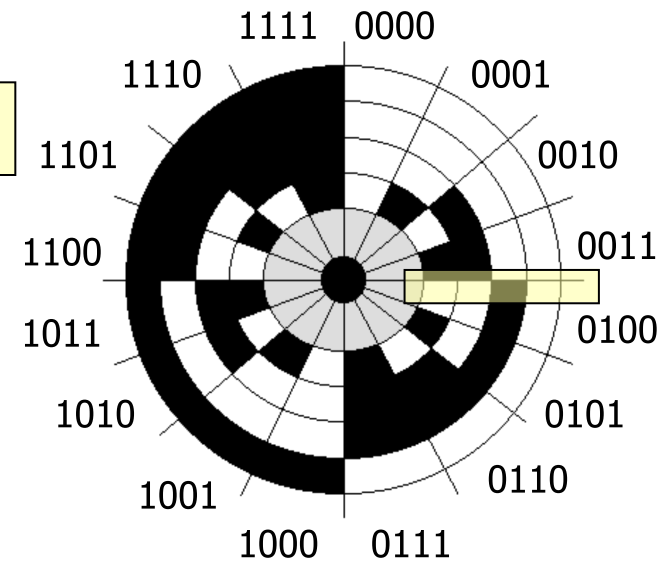
Disco Gray:

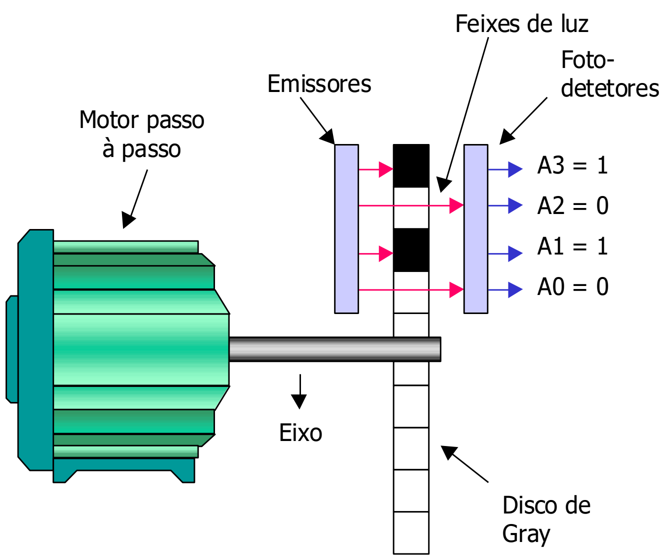


Repare a seqüência:

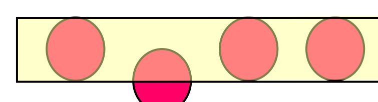
Ref	Gray	Dec	Bin
0	0000	0	0000
1	0001	1	0001
2	0011	3	0010
3	0010	2	0011
4	0110	6	0100
5	0111	7	0101
6	0101	5	0110
7	0100	6	0111
8	1100	12	1000
9	1101	13	1001
10	1111	15	1010
11	1110	14	1011
12	1010	10	1100
13	1011	11	1101
14	1001	9	1110
15	1000	8	1111

Disco Binário:



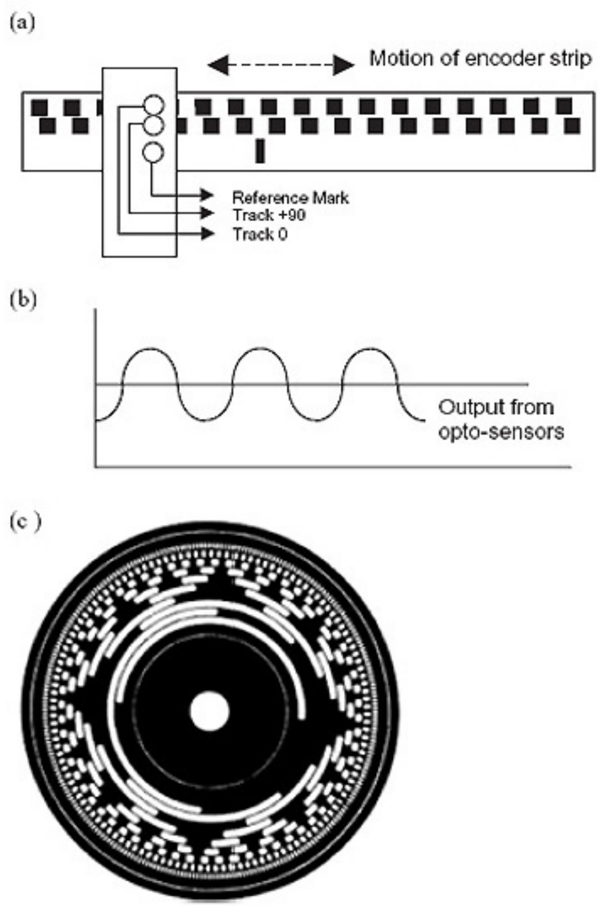


Note o **código Gray**: entre uma variação e outra do código, **somente um bit se altera**. Evita erros de leitura se alguns dos foto-detectores se encontra desalinhado frente a seus “companheiros”:



Exemplo de Uso de Código Gray

- Encoder absoluto: sensor de posição angular!



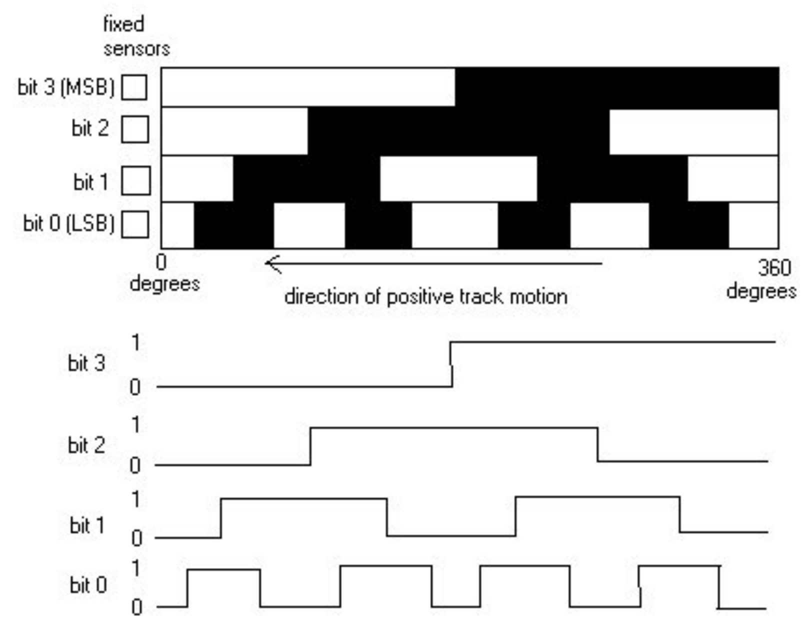
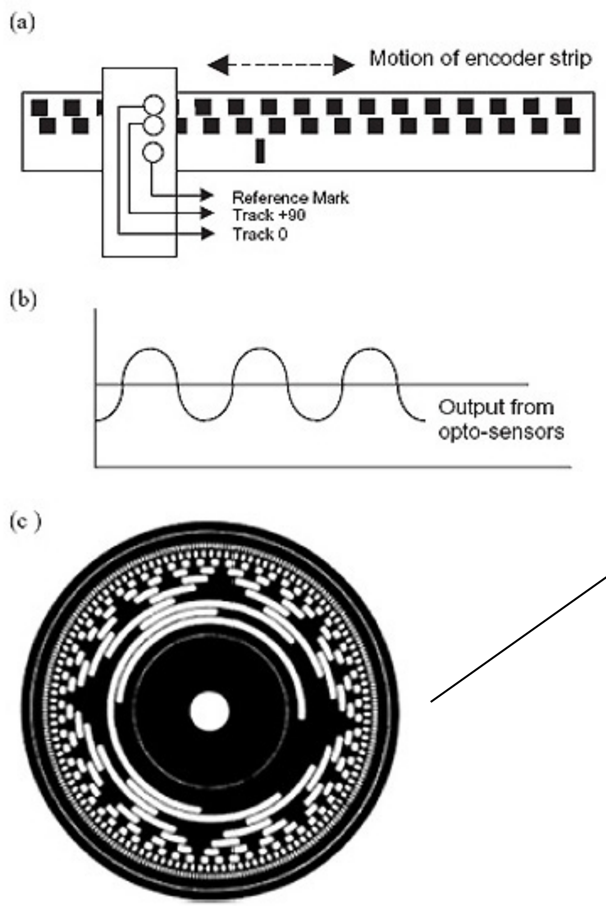
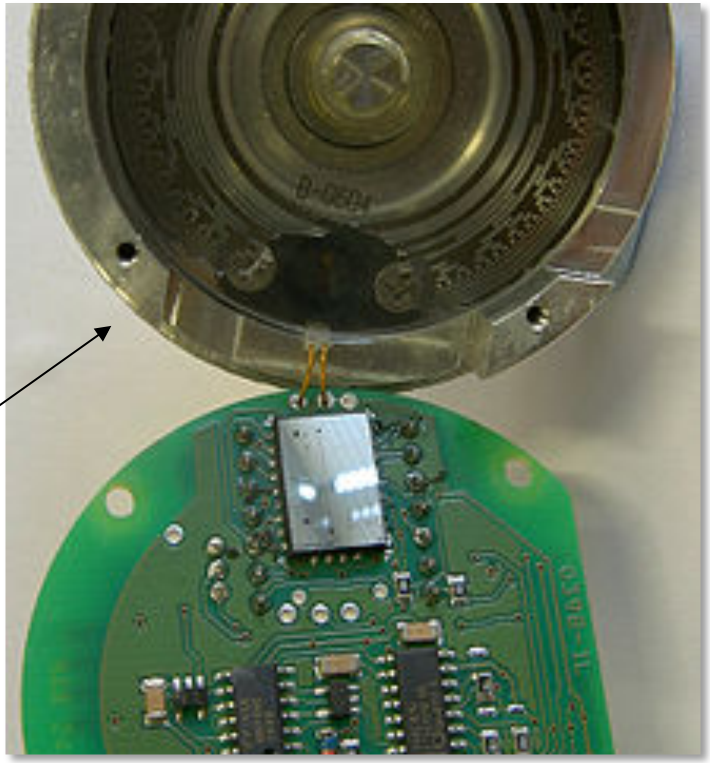


Fig 2. 4-Bit gray code absolute encoder disk track patterns

Exemplo de Uso de Código Gray

- Encoder absoluto: sensor de posição angular!





Um codificador rotativo absoluto de código Gray com 13 trilhas. No topo podem ser vistos o invólucro, o disco óptico e a fonte de luz; na parte inferior pode ser visto o elemento sensor e os componentes de suporte.

Código Binário:: Não é Encoder Absoluto!

Seq. **Gray** (4-bits) X Seq. **Binária** (4-bits)

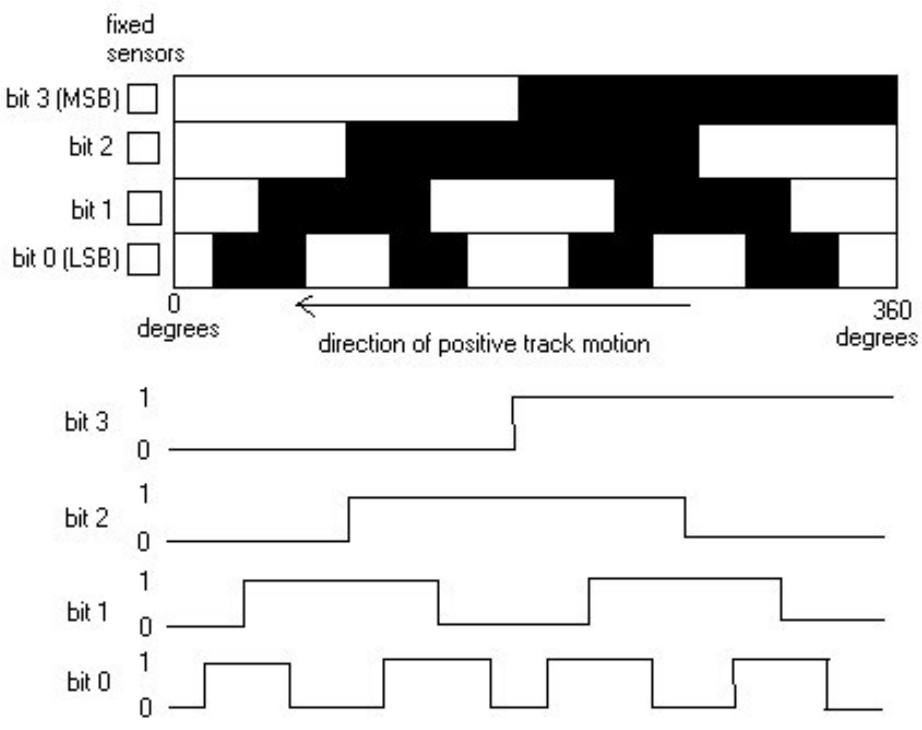


Fig 2. 4-Bit gray code absolute encoder disk track patterns

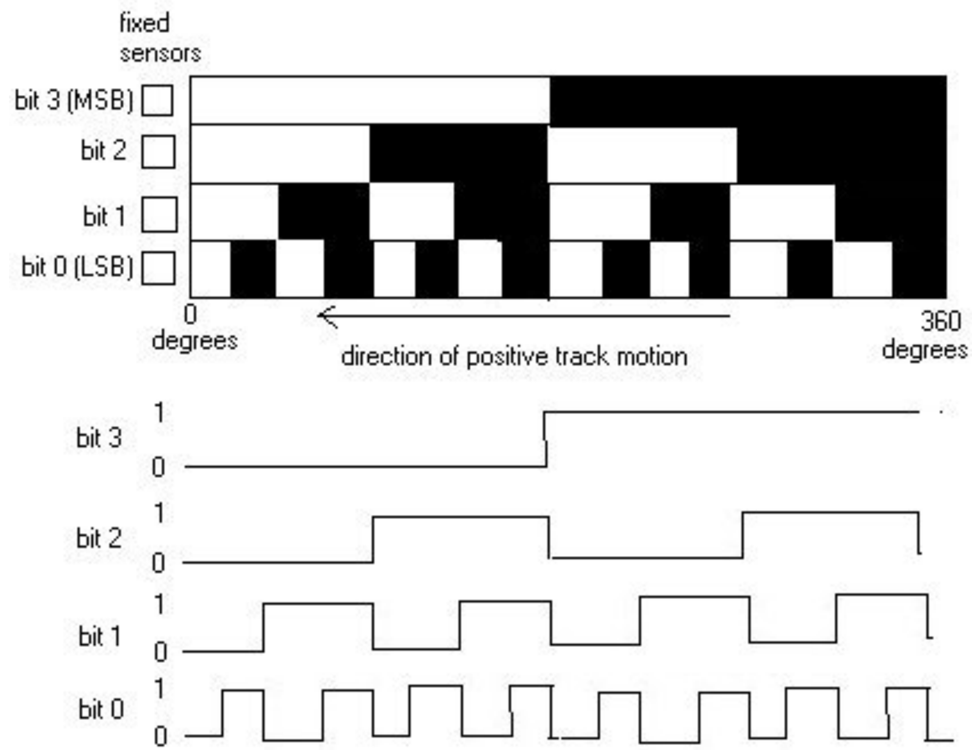
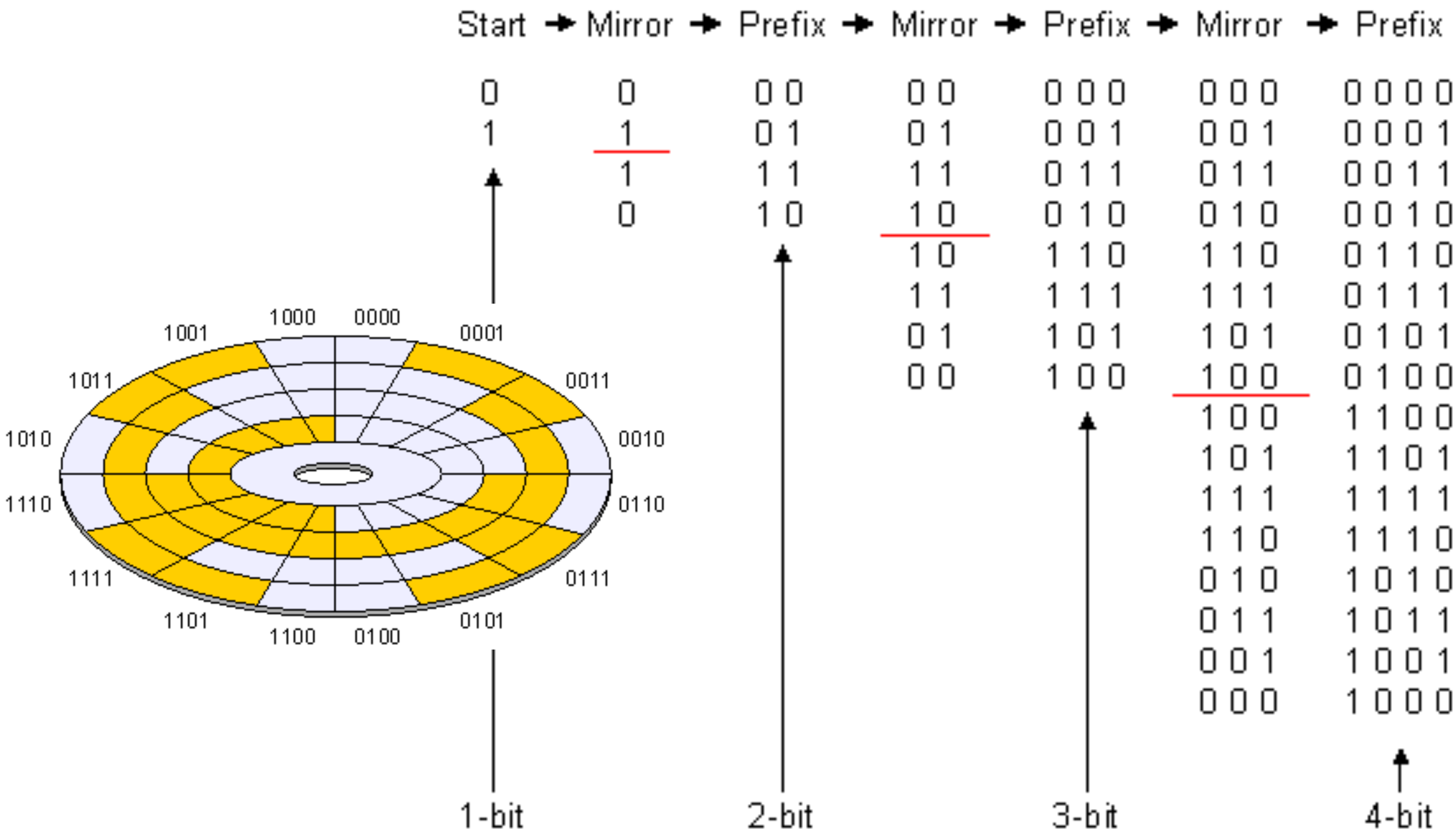


Fig 3 4-Bit binary code absolute encoder disk track patterns

Menos erros
A cada passo: 1 bit apenas varia de estado!

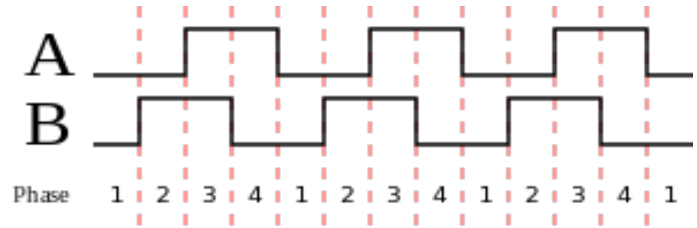
"Saltos grandes" => + erros

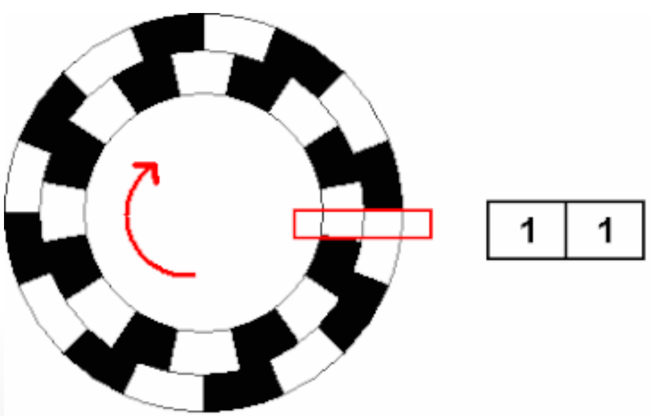
Sequência do Código Gray:



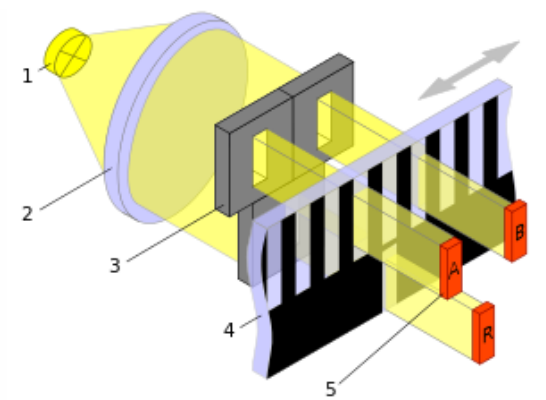
Encoder Absoluto x Encoder Incremental ou Relativo

Duas ondas quadradas em quadratura. A direção do movimento é indicada pelo sinal da diferença de fase A-B que, neste caso, é negativa porque A segue B.

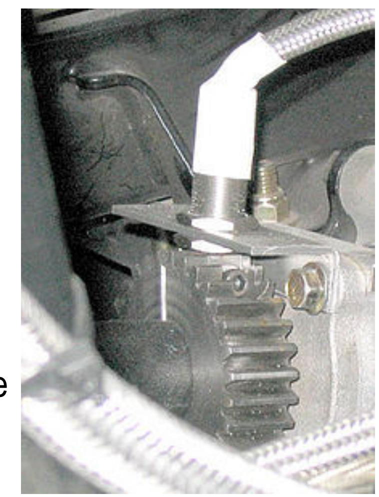




Codificador rotativo, com estados dos sinais A/B correspondentes mostrados à direita



Codificador linear; o sinal R (index) indica que o codificador está localizado em sua posição de referência.



Codificador de quadratura de efeito Hall, detecção de dentes de engrenagem no eixo de transmissão de um veículo.

Código ASCII

32:	33: !	34: "	35: #	36: $	37: %	38: &	39: '	40: (	41:)
42: *	43: +	44: ,	45: -	46: .	47: /	48: 0	49: 1	50: 2	51: 3
52: 4	53: 5	54: 6	55: 7	56: 8	57: 9	58: :	59: ;	60: <	61: =
62: >	63: ?	64: @	65: A	66: B	67: C	68: D	69: E	70: F	71: G
72: H	73: I	74: J	75: K	76: L	77: M	78: N	79: O	80: P	81: Q
82: R	83: S	84: T	85: U	86: V	87: W	88: X	89: Y	90: Z	91: [
92: \	93:]	94: ^	95: _	96: `	97: a	98: b	99: c	100: d	101: e
102: f	103: g	104: h	105: i	106: j	107: k	108: l	109: m	110: n	111: o
112: p	113: q	114: r	115: s	116: t	117: u	118: v	119: w	120: x	121: y
122: z	123: {	124: |	125: }	126: ~	127: ⌂	128: Ç	129: ü	130: é	131: â
132: ä	133: à	134: å	135: ç	136: ê	137: ë	138: è	139: ï	140: î	141: ì
142: Ä	143: Å	144: É	145: æ	146: Æ	147: ô	148: ö	149: ò	150: û	151: ù
152: ÿ	153: Ö	154: Ü	155: ø	156: £	157: Ø	158: ×	159: ƒ	160: á	161: í
162: ó	163: ú	164: ñ	165: Ñ	166: ª	167: º	168: ¿	169: ®	170: ¬	171: ½
172: ¼	173: ¡	174: «	175: »	176: ░	177: ▒	178: ▓	179: │	180: ┤	181: Á
182: Â	183: À	184: ©	185: ╣	186: ║	187: ╗	188: ╝	189: ¢	190: ¥	191: ┐
192: └	193: ┴	194: ┬	195: ├	196: ─	197: ┼	198: ã	199: Ã	200: ╚	201: ╔
202: ╩	203: ╦	204: ╠	205: ═	206: ╬	207: ¤	208: ð	209: Ð	210: Ê	211: Ë
212: È	213:	214: Í	215: Î	216: Ï	217: ┘	218: ┌	219: █	220: ▄	221: ¦
222: Ì	223: ▀	224: Ó	225: ß	226: Ô	227: Ò	228: õ	229: Õ	230: µ	231: þ
232: Þ	233: Ú	234: Û	235: Ù	236: ý	237: Ý	238: ¯	239: ´	240: -	241: ±
242: ‗	243: ¾	244: ¶	245: §	246: ÷	247: ¸	248: °	249: ¨	250: ·	251: ¹
252: ³	253: ²	254: ■	255:						

ascii.cpp (gera tabela ASCII):

```
// Tabela ASCII
// Fernando Passsold, 05/09/2001

#include <stdio.h>
// #include <stdlib.h>
void main() {
   int codigo = 32, coluna=1, linha=1, key, aux;

   for (codigo=32; codigo<256; codigo++) {
      printf("%3d: ",codigo);
      fputchar(codigo);
      coluna++;
      if (coluna<11){
            printf("  ");
      }
      else {
        printf("\n");
            coluna=1;
            linha++;
            if (linha>24){
              // espera uma tecla ser apertada
              while ( (key = getchar()) != '\n' )
                 printf("%c",key);
            linha=1;
        }
      }
   }
}
```

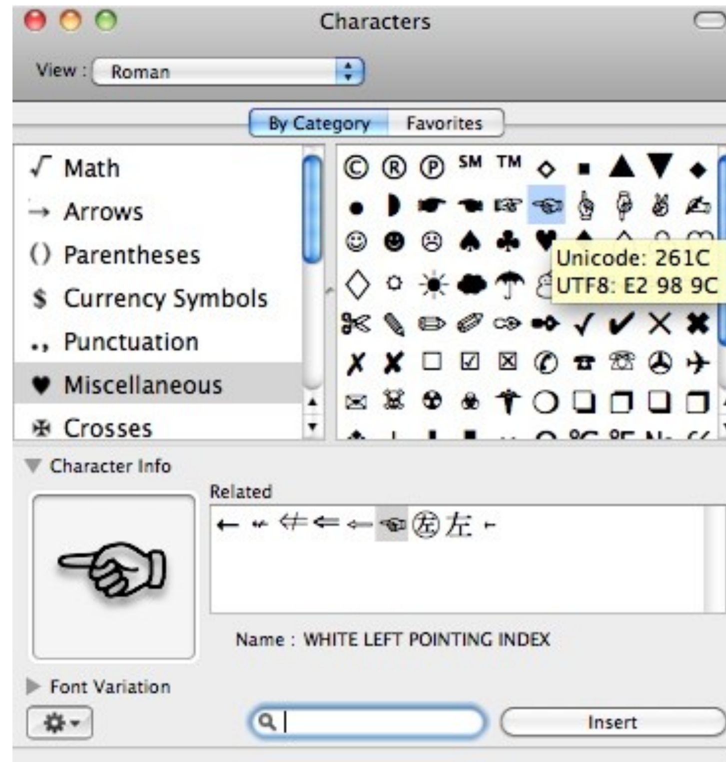
Código UTF-8:

- UTF-8 é uma codificação de caracteres de largura variável usada para comunicação eletrônica. Definido pelo padrão Unicode, o nome é derivado do formato de transformação Unicode (ou conjunto de caracteres codificados universal) - 8 bits.
- O UTF-8 é capaz de codificar todos os 1.112.064 códigos de caracteres válidos em Unicode usando um a quatro bytes (8 bits). Os pontos de código com valores numéricos mais baixos, que tendem a ocorrer com mais frequência, são codificados usando menos bytes. Ele foi projetado para compatibilidade com versões anteriores de ASCII: os primeiros 128 (=2^7) caracteres de Unicode, correspondem com o código ASCII, são codificados usando um único byte com o mesmo valor binário de ASCII, de modo que um texto ASCII válido é também um código UTF-8 válido. Os bytes ASCII não conseguem codificar caracteres não ASCII do UTF-8, mas UTF-8 é seguro para uso na maioria das linguagens de programação e de documento que interpretam caracteres ASCII de uma maneira especial, como "/" (barra) em nomes de arquivos, "\" (barra invertida) em sequências de escape e "%" em comandos “printf()”.
- Os primeiros 128 caracteres (US-ASCII) precisam de um byte. Os próximos 1.920 caracteres precisam de dois bytes para codificar, o que abrange o restante de quase todos os alfabetos de escrita latina e também os alfabetos grego, cirílico, copta, armênio, hebraico, árabe, siríaco, Thaana e N'Ko, bem como os alfabetos diacrítico combinado Marcas. Três bytes são necessários para caracteres no resto do Plano Multilíngue Básico, que contém virtualmente todos os caracteres de uso comum, incluindo a maioria dos caracteres chineses, japoneses e coreanos. Quatro bytes são necessários para caracteres em outros planos de Unicode, que incluem caracteres CJK menos comuns, vários scripts históricos, símbolos matemáticos e emoji (símbolos pictográficos).

Código UTF-8:

- Exemplo: o código Unicode para "€" é U+20AC.

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0xxxxxxx			
U+0080	U+07FF	110xxxxx	10xxxxxx		
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxx x



Notações binárias: números inteiros com sinal.

- Notação Sinal-magnitude
- Complemento a 1
- Complemento a 2
- Sigal-magnitude

Ej. 0 0010110 +22

1 0010110 -22

0 0000000 +0

1 0000000 -0

Números inteiros binários (Notações)

Binário sem sinal	Decimal	Binário com sinal	Decimal ?	Binário C1	Binário C2
0000	0	0000	+ 0 !?	0000	0000
0001	1	0001	1	0001	0001
0010	2	0010	2	0010	0010
0011	3	0011	3	0011	0011
0100	4	0100	4	0100	0100
0101	5	0101	5	0101	0101
0110	6	0110	6	0110	0110
0111	7	0111	7	0111	0111
1000	8	1000	- 0 !?		
1001	9	1001	- 1	1110	1111
1010	10	1010	- 2	1101	1110
1011	11	1011	- 3	1100	1101
1100	12	1100	- 4	1011	1100
1101	13	1101	- 5	1010	1011
1110	14	1110	- 6	1001	1010
1111	15	1111	- 7	1000	1001
	Problemas:	1000			

Códigos de detecção e de correção de erros.

(transmissão digital de dados)

- A transmissão de dados por um canal, pode conter erros causados por interferências, ruído, etc.
- Os dados se corrompem.
- Existem alguns métodos para detectar e corrigir erros.
- O mais comum é acrescentar um bit extra na transmissão.

Detecção de erro com bit de paridade.

Obs: do inglês:
Par = even
Ímpar = odd

- Paridade par e paridade ímpar referem-se a modos de verificação de paridade de comunicação assíncrona.
- É o método mais simples para a detecção de erro.
- Acrescenta um bit adicional de paridade na transmissão.
- A paridade par acrescenta um bit extra em "1" se o conjunto de dados já tiver um número ímpar de bits "1" ou "0" se o número de bits "1" for par.
- A paridade ímpar faz o inverso.
- Exemplo:

7 bits de dados	Contagem de bits "1":	8 bits incluindo paridade	
		Par (even)	Impar (odd)
0000000	0	00000000	00000001
1010001	3	10100011	10100010
1101001	4	11010010	11010011
1111111	7	11111111	11111110