Tema 1

Introducción a los Sistemas Digitales

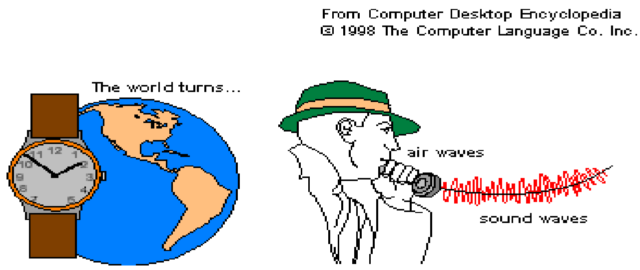
Contenidos

- Sistemas Analógicos y Digitales
- Sistema de Numeración Binario
- Representación de Números Enteros
- Conclusiones

Sistemas Analógicos y Digitales

Magnitud Analógica

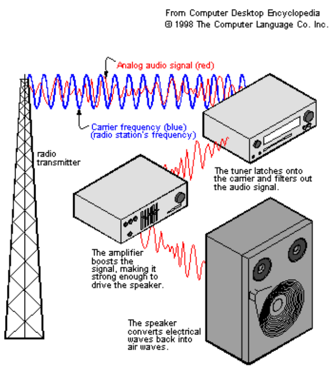
- **Magnitud Analógica**: la que toma cualquier valor contínuo dentro de un rango.
- Todas las magnitudes físicas son analógicas. *El mundo es analógico*
- Ejemplos:
  - Temperatura, velocidad, voz, hora, ...

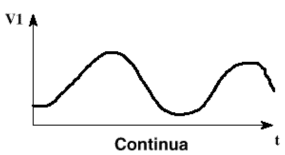


Magnitud Digital

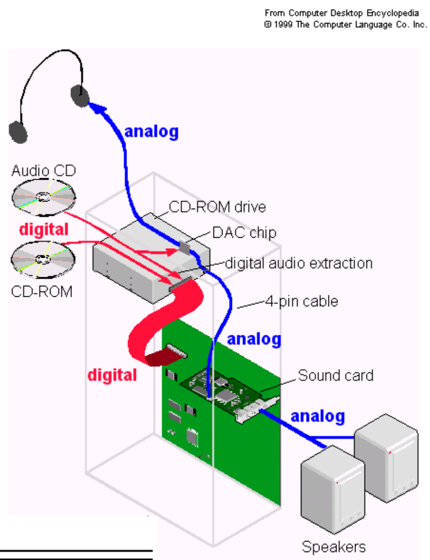
- **Magnitud Digital**: la que toma un valor discreto dentro de un rango finito.
- *En la vida real se utilizan valores discretos.*
- Ejemplos:
  - Panel de temperatura en la calle: 21ºC ó 22ºC. No 21.5ºC
  - Relojes digitales

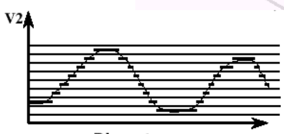
Sistemas Electrónicos Analógicos vs. Digitales





Continua

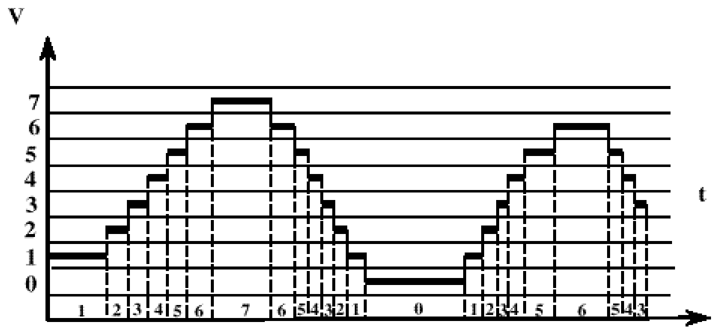




Discreta

Codificación Digital

- A cada nivel se le asigna un código
- Más niveles -> Aumento de la resolución
  -> Aumento de la Complejidad

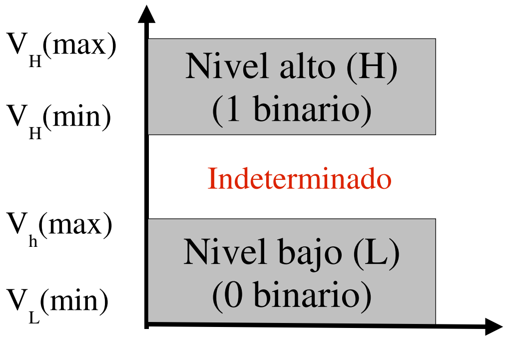


Dígitos Binarios

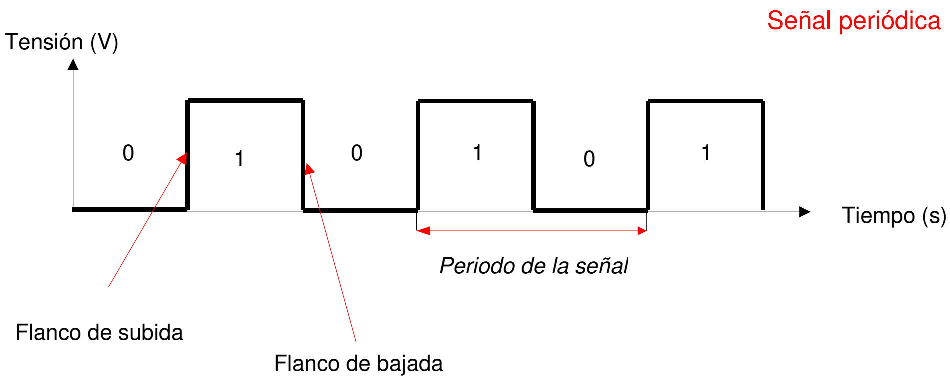
- Dentro de las magnitudes digitales la más utilizada es la binaria
- **Magnitud Binaria**: la que toma 1 de 2 valores posibles
  - Todas son asimilables a pares de valores (sí/no), (verdadero/falso), (0/1).
  - La informática se basa en las magnitudes binarias
  - Ej: Tener gafas (sí/no)
- Los dos dígitos binarios (0 y 1) se denominan **bits**.Representan niveles de tensión
  - **tensión alta -> 1** **tensión baja -> 0**

Niveles Lógicos

- 2 niveles porque es muy fácil distinguirlos y los dispositivos son muy fáciles (equivalente a baratos) de fabricar.
- Los niveles lógicos equivalen a niveles de voltaje, que varían según la tecnología empleada



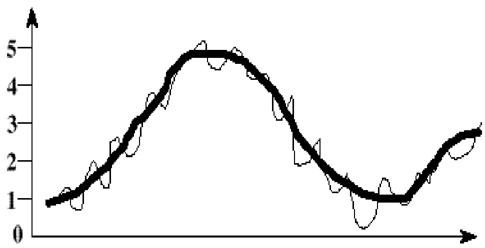
Formas de Onda Digitales



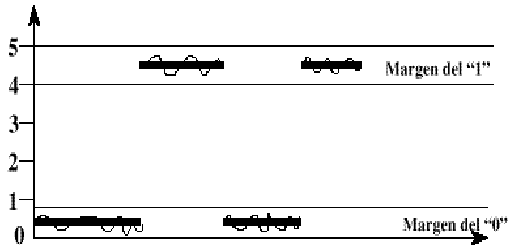
$$frecuencia = \frac{1}{T}$$

Ruido

- Señales analógicas: las perturbaciones modifican el valor de la señal
- Señales digitales: la señal sólo se ve afectada si la perturbación es superior al margen de tensión



Señal Analógica



Señal Digital

Ventajas de los Sistemas Digitales

- Mayor facilidad de diseño (sólo hay 2 niveles)
- Fácil almacenamiento
- Mayor rapidez en el procesamiento de la información
- Los circuitos pueden ser totalmente integrados
- Mayor inmunidad al ruido

Ventajas de los Sistemas Analógicos

- El mundo real es analógico, por lo que la interación con los sistemas digitales requiere un proceso de conversión A/D - D/A.
- A veces el procesamiento analógico es más sencillo (ej: transmisión de voz por teléfono)
- La información digital consume más recursos

Sistema de Numeración Binario

Sistemas de Numeración

- Sistemas posicionales: el valor de los dígitos depende de la posición dentro del número

3 3

3x10 + 3x1

33

- Sistemas de numeración más utilizados:

Sistema	Base	Dígitos
Decimal	10	[0-9]
Binario	2	[0,1]
Octal	8	[0-7]
Hexadecimal	16	[0-9] U {A, B, .. F}

Sistemas de Numeración

Decimal	Binario	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Sistemas de Numeración

- En general: $N_b = a_{p-1}b^{p-1} + a_{p-2}b^{p-2} + ... + a_0b^0 + a_{-1}b^{-1} + ...$
  - b es la base y los a_i son los coeficientes.
  - Representación $(a_{p-1}a_{p-2}a_{p-3}...a_0a_{-1}...)$
- Ejemplos:
  - $564{,}25_{10} = 5 \times 10^2 + 6 \times 10^1 + 4 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$

    $= 500 + 60 + 4 + 0{,}2 + 0{,}05$
  - $564{,}25_7 = 5 \times 7^2 + 6 \times 7^1 + 4 \times 7^0 + 2 \times 7^{-1} + 5 \times 7^{-2}$

    $= 245 + 42 + 4 + 0{,}286 + 0{,}102$

Cambios de Base

- De cualquier base a base 10 (enteros):
  - expresar en forma polinomial y operar en base 10
- Ejemplo:
  - $BC92_{16} = 11x16^3 + 12x16^2 + 9x16^1 + 2x16^0 = 48274$
  - $101100.11_2 = 1x2^5 + 1x2^3 + 1x2^2 + 1x2^{-1} + 1x2^{-2}$

    $= 32 + 8 + 4 + 0{,}5 + 0{,}25 = 44{,}75$

Cambios de Base

- De base 10 a cualquier base:
  - Se divide por la base sucesivamente, tomando los restos en order inverso
- Ejemplo:
  - 48274_{10} -> base 16

48274	16		
2	3017	16	
	9	188	16
		12	11

  - Resultado: 11 - 12 - 9 - 2 (BC92)

Cambios de Base

- De base 10 a cualquier base (fracionales):
  - Parte entera y fraccional por separado.
  - Parte fraccional: multiplicar el número por la base y tomar la parte entera. Con el resto se repite el proceso hasta obtener el número de decimales deseados.
- Ejemplo:
  - 0.6875_{10} -> base 2 -> 0.1011_2

0.6875	x 2	1
0.375	x 2	0
0.75	x 2	1
0.5	x 2	1
0		

Cambios de Base

- De la base 2 a la base 2^n:
  - Agrupando los bits de n en n, de derecha a izquierda
  - Ejemplo: 179_2 -> base 8

$$179_2 = 10110011_2 = (010)\ (110)\ (011) = 263_8$$

- De la base 2^n a la base n:
  - Expandiendo cada dígito por los n bits correspondientes
  - Ejemplo: $B3_{16}$ -> base 2

$$B3_{16} = (1011)(0011) = 10110011_2$$

Sistema de Numeración Binario

- Formado por sólo 2 dígitos
- Es el más adecuado para sistemas electrónicos digitales
- Requiere muchas más cifras que el decimal para contar una misma cantidad

Decimal	Binario			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
...				

Sistema de Numeración Binario

- Rango de representación
  - con n bits pueden representarse **2^n** números diferentes
  - con n bits el rango será: **$[2^{n-1} .. 0]$**
  - Ejemplo: 4 bits -> 2^4 (16_{10}) números diferentes,

    en el rango [15 .. 0]
- ¿Bits necesarios para representar un rango de N números decimales?
  - Respuesta: **$n = \log_2 N$**
  - Ejemplo: 16_{10} -> $n = \log_2 16 = 4$ bits

Arimética Binaria: Suma

- 4 reglas básicas
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 1 0

Bits de acarreo

- Ejemplo:

```
      1   1
      1   1   1    <-  3 bits
+     0   0   1    <-  3 bits
-------------------
  1   0   0   0    <-  4 bits
```

Aritmética Binaria: Resta

- 4 reglas básicas
  - 0 - 0 = 0
  - 1 - 1 = 0
  - 1 - 0 = 1
  - 1 0 - 1 = 1
- Ejemplo:

Bits de acarreo

```
            1
      0 ~1~  0   1    <-  3 bits
 -       0   1   1    <-  3 bits
 -------------------
         0   1   0    <-  3 bits
```


Aritmética Binaria: Multiplicación

- 4 reglas básicas
  - 0 x 0 = 0   0 x 1 = 0
  - 1 x 0 = 0   1 x 1 = 1
- Ejemplo:

```
            1   1   1    <-  3 bits
      x     1   0   1    <-  3 bits
      --------------------
            1   1   1
        0   0   0
    1   1   1
 ---------------------------
 1  0   0   0   1   1    <-  6 bits
```

Representación de Números enteros

Representación de Enteros

- Hasta ahora sólo hemos utilizado binarios enteros positivos
  - con n bits -> $0 \leq N \leq 2^{n-1}$
- ¿Cómo representar números negativos?
  - 2 alternativas
    - Signo-Magnitud
    - Complemento

Representación Signo-Magnitud

- El primer bit representa el signo
  - 0 -> signo positivo, 1 -> signo negativo
- El resto de la cifra representa la magnitud (valor absoluto)

Signo Magnitud

S | MMM ... MM

n-1 n-2 ... 0

- Ejemplos:
  - $7_{10} \rightarrow 111_2$ $\quad n^o \text{ bits} = \lceil \log_2 7 \rceil = 3$
  - $+7_{10} \rightarrow 0111_2$ $\quad n^o \text{ bits} = \lceil \log_2 7 \rceil + \text{signo} = 4$
  - $-7_{10} \rightarrow 1111_2$ $\quad n^o \text{ bits} = \lceil \log_2 7 \rceil + \text{signo} = 4$

Representación Signo-Magnitud

- Rango representable con n bits
  - sin signo : $[2^n\text{-}1 \,..\, 0] \rightarrow$ total $= 2^n$
  - signo-magnitud : $[-(2^{n-1}\text{-}1) \,..\, (2^{n-1}\text{-}1)] \rightarrow$ total $= 2^n$
- Ejemplo: 3 bits

Representación	Signo-Magnitud	Sin signo
000	0	0
001	1	1
010	2	2
011	3	3
100	-0	4
101	-1	5
110	-2	6
111	-3	7

Doble representación para el 0

Representación Signo-Magnitud

- Suma y resta:
  - Caso 1: los números son de signo contrario.
    - Se resta el mayor del menor. El signo resultante será el del mayor.
    - 4 - 3 $\rightarrow 0100 - 1011 \rightarrow +(100 - 011) = +1 \rightarrow 0001$
    - -4 + 3 $\rightarrow 1100 - 0011 \rightarrow -(100 - 011) = -1 \rightarrow 1001$
  - Caso 2: los números son del mismo signo
    - Se suman ambos. El signo resultante será el de los operandos.
    - Puede producirse desbordamiento
    - 4 +5 $\rightarrow 0100 + 0101 \rightarrow +(100 + 101) = +(1001)$ => overflow
    - -4 -5 $\rightarrow 1100 + 1101 \rightarrow -(100 + 101) = -(1001)$ => overflow

Representación Signo-Magnitud

- Producto:
  - Se multiplican ambos operandos. Si son del mismo signo el resultado es positivo. Sinó es negativo.
  - El tamaño del resultado es la suma de los tamaños de los operandos
    - n bits * m bits = n+m bits
  - No hay overflow.

Representación en Complemento

- No es necesario distiguir entre la suma y la resta
- Dos tipos
  - Complemento a la base - 1 (complemento a 1)
  - Complemento a la base (complemento a 2)

Representación en Complemento a 1

- Los números positivos se representan igual que en signo-magnitud. Los negativos se representan complementando todos los bits
- Rango : $[-(2^{n-1}-1) .. (2^{n-1}-1)] \rightarrow$ total = 2^n
- Ejemplos: n = 4 bits
  - +7 $\rightarrow$ 0111
  - -7 $\rightarrow (0111)' = 1000_{C1}$

Representación	C1
000	0
001	1
010	2
011	3
100	-3
101	-2
110	-1
111	-0

Doble representación para el 0

Representación en Complemento a 1

- Suma y resta:
  - Si el signo es opuesto debe complementarse el negativo
  - Siempre se suma
  - Signo resultante:
    - el de los operandos sin son del mismo signo. Sinó **overflow.**
    - El del mayor de los dos.
  - Ejemplo:
    - $4 - 3 = 4 + (-3) \rightarrow 0100 - 0011 = 0100 + C1(0011) \rightarrow$
      $\rightarrow 0100 + 1100 = 0001$
    - $-4 -5 = (-4) + (-5) \rightarrow 0100 - 0101 = C1(0100) + C1(0101) \rightarrow$
      $\rightarrow 1011 + 1010 = 0101$ => overflow

Representación en Complemento a 1

- Producto:
  - Como en signo-magnitud
- Extensión del signo
  - positivos: $3 \rightarrow 011_2 = 0...011_2$
  - negativos: $-3 \rightarrow 100_2 = 1...100_2$

Representación en Complemento a 2

- Los números positivos se representan igual que en signo-magnitud. Los negativos como 2^n - el número.
- Rango : $[-(2^{n-1}) .. (2^{n-1}-1)] \rightarrow$ total = 2^n
- Ejemplos: n = 4 bits
  - $+7 \rightarrow 0111$
  - $-7 \rightarrow (2^4 - 7)_{10} = (16 - 7)_{10} = 9_{10}$

    $9_{10} = 1001_{C2}$
- Operaciones aritméticas:
  - como en complemento a 1

Representación	C2
000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

Representación en Complemento a 2

- Otras formas de calcular el complemento a 2:
  - C2(número) = C1(número) + 1
    - C2(1011010) = C1(1011010) + 1 = 0100101 + 1 = 0100110
  - De derecha a izquierda copiar todos los bits hasta encontrar el primer '1'. A partir de él (sin incluirlo) complementar el resto de bits.
    - 1 0 1 1 0 | 1 | 0
      ↓ ↓ ↓ ↓ ↓ ↓ ↓
      0 1 0 0 1 | 1 | 0

Hasta el 1er. '1' todos los bits son iguales

El 1er. '1' se mantiene

El resto de valores se complementa

Tabla Comparativa

Bits	Números	Rango SM	Rango C1	Rango C2	Sin Signo
4	$2^4 = 16$	[-7.. +7]	[-8 .. +7]	[0 .. 15]	[0 .. 15]
8	$2^8 = 256$	[-127.. +127]	[-127 .. +127]	[-128 .. +127]	[0 .. 255]
16	$2^{16} = 65536$	[-32767.. +32767]	[-32767.. +32767]	[-32768.. +32767]	[0 .. 65535]
N	2^n	$[-2^{n-1}-1 .. +2^{n-1}-1]$	$[-2^{n-1}-1 .. +2^{n-1}-1]$	$[-2^{n-1} .. +2^{n-1}-1]$	$[0 .. +2^n-1]$

Conclusiones

- Dos tipos de sistemas electrónicos: analógicos y digitales
- Los dígitos binarios equivalen a niveles de tensión
- El sistema de numeración utilizado por los computadores es el binario
- Cambios de base
  - 10 ↔ cualquier base
  - $2 \leftrightarrow 2^n$
- 3 posibles representaciones para los números binarios
  - signo-magnitud
  - complemento a 1
  - complemento a 2