

Lógica de circuitos. Circuitos combinacionales y secuenciales.

Tema 9

ABACUS NT

Índice

- 1. Introducción**
- 2. Lógica de circuitos**
 - 2.1. Señales.**
 - 2.2. Sistemas**
 - 2.3. Lógica binaria**
- 3. Álgebra de Boole**
 - 3.1. Teoremas del álgebra de Boole**
 - 3.2. Funciones lógicas**
- 4. Circuitos combinaciones**
- 5. Diseño de circuitos combinacionales**
 - 5.1. Simplificación de funciones**
 - 5.1.1. Método algebraico**
 - 5.1.2. Método gráfico de Karnaugh**
- 6. Circuitos secuenciales**
 - 6.1. Biestables**
 - 6.2. Principales módulos estándar**
 - 6.3. Diseño de circuitos secuenciales**
 - 6.3.1. Técnicas de diseño actuales**
- 7. Conclusión.**
 - 7.1. Sistema educativo**
- 8. Bibliografía**

1. Introducción

El estudio de los circuitos electrónicos se puede clasificar en dos amplias categorías: circuitos **analógicos** (utiliza magnitudes con valores continuos) y circuitos **digitales** (utiliza magnitudes con valores discretos).

En este tema nos vamos a centrar en los **sistemas digitales**, los cuales se basan en la electrónica digital que utiliza circuitos en los que sólo existen dos estados posibles (0 y 1).

En la actualidad estamos rodeados de **infinidad de dispositivos** que integran circuitos digitales, como lavadoras, televisores, ordenadores, etc.

Estos circuitos digitales podemos dividirlos en dos grandes grupos: los circuitos **combinacionales** y los circuitos **secuenciales**, los cuales serán estudiados en detalle a lo largo de este tema

2. Lógica de circuitos

2.1. Señales.

Una **señal** es una variación de una magnitud física que se utiliza para transmitir información. Las señales suelen **clasificarse** en dos grupos, las señales analógicas, las cuales toman valores continuos, y las señales digitales, que toman valores discretos.

Las **ventajas** del uso de las señales digitales es que simplifican el diseño de los circuitos electrónicos, facilitan la detección de errores, y son más robustas frente al ruido, es por ello que los ordenadores fundamentalmente utilizan circuitos digitales.

2.2. Sistemas

Un sistema es un **conjunto** ordenado que contribuye a un fin, el cual, ante una entrada, da como respuesta una salida. En función de las dependencias de la salida, los sistemas podemos clasificarlos en:

- **Sistema sin memoria:** La salida depende únicamente de los valores de la entrada $H(x(t)) = y(t)$
- **Sistema con memoria:** La salida también depende del estado en el que se encuentra el sistema, $H(s(t), x(t)) = y(t)$.

Adicionalmente, dependiendo de cuándo puede cambiar sus señales, los sistemas se clasifican en:

- **Asíncronos**
- **Síncrono**

2.3. Lógica binaria

La lógica binaria establece relaciones funcionales entre variables binarias. Cada función puede representarse por una **tabla de verdad**. Las principales son:

Variables booleanas		Funciones combinacionales			
x	y	x AND y	x OR y	x XOR y	NOT x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

3. Álgebra de Boole

Definición

El álgebra de Boole es un conjunto cualquiera A en el que se han definido dos operaciones binarias denominadas suma lógica (+), también conocida como OR, y producto lógico (\bullet), también conocido como AND, y una operación unitaria denominada complemento, también conocida como NOT.

Se dice que A es un álgebra de Boole si cumple las siguientes propiedades:

Propiedad de cierre

El conjunto A es cerrado para las dos operaciones, es decir, $\forall a, b \in A$:

$$a+b \in A$$

$$a \bullet b \in A$$

Ley asociativa

$$a \bullet b \bullet c = a \bullet (b \bullet c) , \forall a, b, c \in A$$

Ley conmutativa

$$a \bullet b = b \bullet a , \forall a, b \in A$$

Ley distributiva

$\forall a, b, c \in A$, se verifica:

$$a \bullet (b+c) = a \bullet b + a \bullet c$$

$$a+(b \bullet c) = (a+b) \bullet (a+c)$$

Complementario

$\forall a \in A, \exists a'$ tal que:

$$a+a' = 1$$

$$a \bullet a' = 0$$

3.1. Teoremas del álgebra de Boole

Idempotencia

$\forall a \in A$, se verifica:

$$a+a = a$$

$$a \bullet a = a$$

Elemento identidad

$$a+0 = a$$

$$a \bullet 1 = a, \forall a \in A$$

Absorción

$$a+a \bullet b = a$$

$$a \bullet (a+b) = a, \forall a, b \in A$$

Teoremas De Morgan

$$\overline{(a \bullet b)} = \overline{a} + \overline{b}$$

$$\overline{(a+b)} = \overline{a} \bullet \overline{b}, \forall a, b \in A$$

3.2. Funciones lógicas

Las funciones lógicas son expresiones que contienen sumas y restas de variables binarias (sólo 2 valores lógicos) que pueden estar complementadas o no. El resultado de una función, que será un valor lógico, depende de sus variables.

Las funciones lógicas pueden representarse de distintas formas:

Forma algebraica

Función de ejemplo: $F(a,b,c) = a+a \bullet c+a \bullet b$

(Utilizaremos esta misma función en las próximas representaciones)

Tabla de Verdad

Se representan todos los valores que puede tomar una función:

a	b	c		F(a,b,c)
0	0	0		0
0	0	1		0
0	1	0		1
0	1	1		0
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

Forma canónica

Se define como **término canónico de una función lógica**, a todo producto o suma (minterm y maxterm respectivamente), en el que aparecen todas las variables de la función en su forma directa o complementada. Notamos como a' al complementario de a .

Minterm:

Se toman las salidas de la tabla de verdad **que son 1** y se expresa como suma de términos producto (para la tabla de verdad anterior tenemos):

$$F(a,b,c) = b + a \cdot c + a \cdot b \cdot c = a' \cdot b \cdot c' + a \cdot b' \cdot c + a \cdot b \cdot c$$

Maxterm:

Se toman las salidas de la tabla de verdad **que son 0** y se expresa como producto de suma de términos (para la tabla de verdad anterior tenemos):

$$F(a,b,c) = 0 \cdot c \cdot (b+c) \cdot a \cdot (a+b) = (a'+b'+c') (a'+b'+c) (a'+b+c) (a+b'+c') (a+b+c')$$

Mapa de Karnaugh

Es una representación gráfica, utilizada en el diseño de circuitos para simplificar funciones booleanas. Se desarrolla en el siguiente punto.

Circuitos lógicos

Es una representación gráfica, donde las operaciones se representan utilizando **puertas lógicas**.

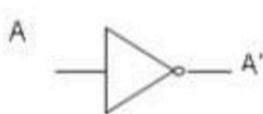
Las puertas lógicas son el bloque fundamental de construcción de todos los circuitos lógicos digitales, de forma que las funciones lógicas se implementan interconectando puertas. Para representar las puertas lógicas se utilizan las siguientes notaciones:

Símbolos distintivos: corresponden a los utilizados comúnmente por la industria digital.

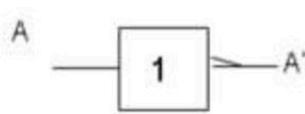
Símbolos rectangulares: en 1984 se desarrolló una nueva norma para la representación de símbolos lógicos llamada IEEE/ANSI 91-1984.

A continuación, se representan las principales puertas lógicas empleando la notación de símbolos distintivos (a) y la de símbolos rectangulares (b). Además, se muestra la tabla de verdad asociada a cada puerta lógica (c):

Puerta NOT: Negación lógica de la entrada ($1 \rightarrow 0 ; 0 \rightarrow 1$)



a)



b)

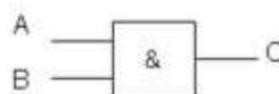
a	s
0	1
1	0

c)

Puerta AND: Salida a 1 si ambas entradas están a nivel 1



a)



b)

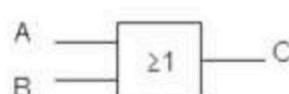
a	b	s
0	0	0
0	1	0
1	0	0
1	1	1

c)

Puerta OR: Salida a 1 si alguna entrada a 1



a)

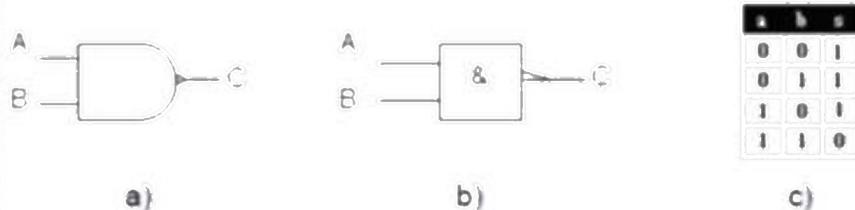


b)

a	b	s
0	0	0
0	1	1
1	0	1
1	1	1

c)

Puerta NAND: Operación NOT AND



4. Circuitos combinaciones

Especificaciones

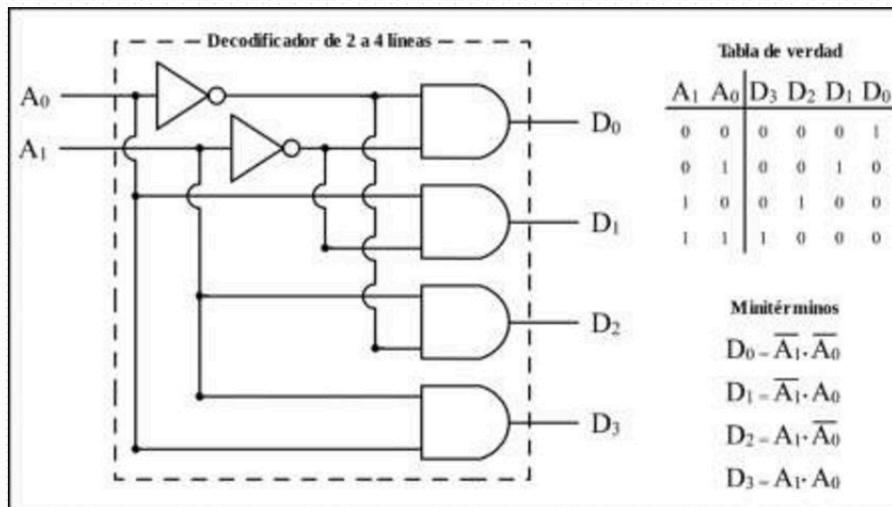
Los circuitos combinacionales dan solución a problemas que pueden expresarse como funciones de conjuntos finitos, las cuales suelen representarse con tablas o expresiones.

Un **ejemplo** podría ser el siguiente, en el que tenemos un sensor, y en función del valor de salida del sensor queremos que la luz de un semáforo se comporte de la siguiente forma:

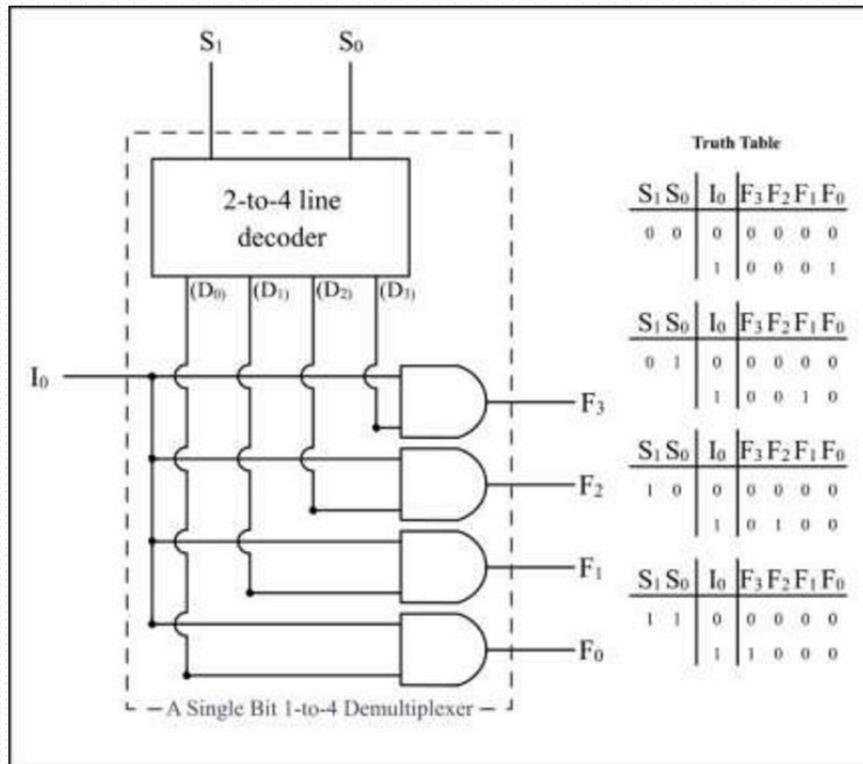
Salidas del Sensor (entrada del circuito)	X_1	Luz del semáforo (Salida del circuito)	y_1
Cerca	1	Rojo	0
Lejos	0	Verde	1

Alguno de los principales módulos estándares de circuitos combinaciones son:

- **Decodificador binario:** Este módulo tiene n entradas, y 2^n salidas. En función de la representación binaria del conjunto de entradas, cuando ésta represente el número i , todas las salidas estarán desactivadas, a excepción de la salida x_i . Su inverso es el codificador.

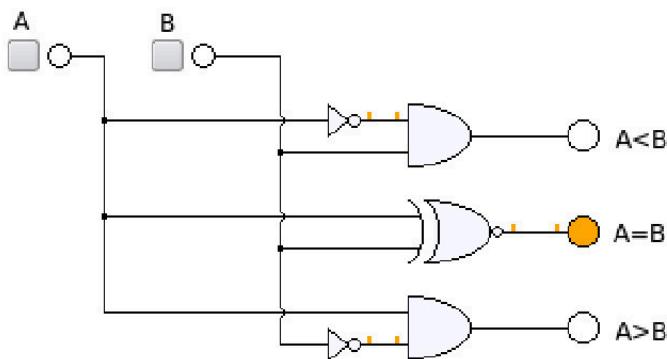


- **Multiplexor:** Se trata de un módulo con n señales de selección, 2^n entradas, y una salida. La salida será igual a la entrada seleccionada por la representación en binario de las señales de selección. Su inverso es el demultiplexor.



- **Comparador:** Tiene 2 entradas de n bits, y 3 salidas cuyas funciones de salida son:
 - Mayor Si $x > y$
 - Igual Si $x = y$

- Menor Si $x < y$



5. Diseño de circuitos combinacionales

Los pasos para diseñar un circuito combinacional, obteniendo su función canónica son:

1. Codificar las especificaciones de alto nivel en especificaciones binarias.
2. Representar las especificaciones en mapas de Karnaugh (matriz donde se representa en función de los valores de las entradas, en código gray, el valor de la salida).
3. Obtener los minterms, formando para ello rectángulos de tamaño 2^n de celdas, del mayor tamaño posible, que agrupen a todos los 1 en el mapa de Karnaugh. También se podría hacer agrupando los maxterms, en ese caso sería agrupando ceros.
4. La función lógica de salida es la suma de los productos conformado por los minterms (valores de las variables que no variaban en los rectángulos conformados en el paso anterior), obteniéndose la representación canónica
5. Finalmente diseñamos el circuito en función de la función lógica obtenida, que será una red de tres niveles: NOT, AND y OR. Para maxterms sería: NOT, OR, AND.

5.1. Simplificación de funciones

En electrónica digital, resulta muy interesante optimizar un circuito, es decir, simplificarlo. Se trata de obtener la representación de la función con el menor número de elementos, lo que nos llevará a la hora de la implementación física de dicha función, a obtener un circuito más sencillo y barato.

Existen varios métodos para simplificar funciones lógicas.

5.1.1. Método algebraico

Utiliza las propiedades y teoremas del álgebra de Boole. Por ejemplo:

$$F = (a + b) \bullet (a + c) = a + (b \bullet c)$$

lo que se consigue de manera directa a partir de la propiedad distributiva.

En este método, el grado de optimización de la expresión final depende de la habilidad del diseñador para aplicar la propiedad más adecuada en cada paso de la simplificación. Al crecer la complejidad de la expresión para simplificar esta tarea se hace cada vez más difícil. Por ello, se utilizan otros métodos que facilitan y automatizan el proceso de simplificación de las funciones lógicas. - los Mapas de Karnaugh, y el método tabular de Quine-McCluskey.

5.1.2. Método gráfico de Karnaugh

Se basa en la representación gráfica de la función en un cuadro denominado mapa de Karnaugh todo válido para simplificar de forma sencilla funciones de hasta 5 ó 6 variables.

Una tabla de verdad tiene tantas filas como términos, sean estos minterms o maxterms, mientras un mapa que el mapa K correspondiente tiene una celda por cada término.

Tabla para 2 Variables de Entrada

a b	0	1
0		
1		

Tabla para 3 Variables de Entrada

c a b	00	01	11	10
0				
1				

Tabla para 4 Variables de Entrada

c d a b	00	01	11	10
00				
01				
11				
10				

Tabla para 5 Variables de Entrada

d e c a b	000	001	011	010	110	111	101	100
00								
01								
11								
10								

Los encabezamientos de las filas y las columnas representan para las variables todas las combinaciones de estados de las mismas que se pueden dar.

El orden de las combinaciones de valores debe ser tal que dos adyacentes sólo se pueden diferenciar en un valor.

Con estas configuraciones existe una adyacencia gráfica igual a la adyacencia algebraica. Definimos los términos adyacentes desde el punto de vista lógico como dos términos del mismo tipo mini o maxi) que difieren sólo en una variable.

Por ejemplo, $a \bullet b \bullet c' \bullet d'$ y $a \bullet b \bullet c' \bullet d$ son minterms de cuatro variables adyacentes lógicamente ya que sólo difieren en la variable d. Ambos términos pueden combinarse eliminando una variable ($a \bullet b \bullet c' \bullet d' + a \bullet b \bullet c' \bullet d = a \bullet b \bullet c' \bullet (d + d')$).

En el mapa de Karnaugh indicamos los términos que se pueden combinar rodeándolos con un trazo. Estos términos al combinarse nos darán una expresión con menos variables.

Además, se interpretará que la fila adyacente a la primera por arriba es ¡la última por abajo y la adyacente a la última columna por la derecha es la primera por la izquierda.

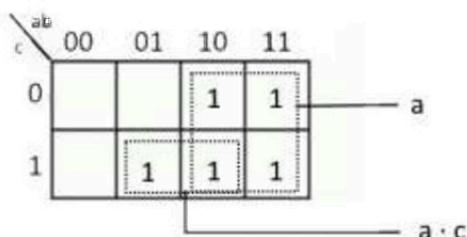
Veamos un ejemplo utilizando minterms:

$$F = (a + b) \bullet (a + c)$$

en primer lugar, obtenemos la notación especial abreviada correspondiente a la primera forma canónica

$$F(a, b, c) = 011_{(2)} + 100_{(2)} + 101_{(2)} + 110_{(2)} + 111_{(2)} = m3 + m4 + m5 + m6 + m7 = \Sigma m (3, 4, 5, 6, 7)$$

a continuación, colocamos un 1 en cada una de las celdas del mapa de Karnaugh correspondientes a las combinaciones obtenidas. Seguidamente, se agrupan unos adyacentes en conjuntos de potencias de 2 (a partir de 2¹), es decir, 2, 4, 8, etc., pudiendo haber intersecciones entre los grupos. Cuanto mayor sea el agrupamiento, mayor será la simplificación. A cada grupo de unos le corresponde un mintérmino, donde se eliminan aquellas variables que aparezcan con valor 0 y 1 dentro del propio grupo y se mantienen, efectuándose entre ellas el producto lógico, aquellas que sólo tengan un valor.



Finalmente, se suman los términos obtenidos lográndose función simplificada.

Como se ha indicado, a esta técnica de resolución se la conoce como trabajo en minterms (simplificación utilizando los unos). Existe otra técnica paralela que trabaja con maxitérminos (simplificación utilizando los ceros). Seguidamente, recogemos con más detalle las reglas a aplicar para la simplificación por minterms.

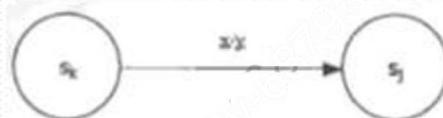
- Cada celda (mintérmino) sobre un mapa de Karnaugh de dos variables tiene dos celdas (minterms) adyacentes lógicamente sobre un mapa de Karnaugh de tres variables, cada mintérmino tendrá tres adyacentes, etc. En general, cada celda en un mapa de Karnaugh de n variables tiene n celdas adyacentes de modo que cada par de celdas adyacentes difieren precisamente en una variable.
- Al combinar las celdas en un mapa de Karnaugh agruparemos un número de minterms que sea potencia de 2. Al agrupar 2 celdas eliminamos 1 variable, al agrupar 4 celdas eliminamos 2 variables, etc. En general al agrupar 2ⁿ celdas eliminamos n variables. Debemos, por tanto, agrupar tantas celdas como sea posible. Cuántas más celdas tenga cada grupo, mayor será la simplificación.

- Debemos cubrir todas las celdas minterms) de la función con el menor número posible de grupos. Un mintérmino está cubierto si está incluido al menos en un grupo. A menor número de grupos, menos términos tendrá la función minimizada. Debemos utilizar cada celda al menos una vez, pero podemos usarlas en más de un grupo. Tan pronto hayamos utilizado todos los minterms al menos una vez nos detenemos.
- La combinación de celdas en el mapa, se hará siempre empezando por los minterms que tienen menor número de celdas adyacentes (los más solitarios en el mapa). Los minterms con varios términos adyacentes ofrecen más combinaciones posibles y, por tanto, deben combinarse más adelante en el proceso de minimización.

6. Circuitos secuenciales

Especificaciones

El comportamiento de los circuitos secuenciales se representa mediante diagramas de estados, en donde los estados son círculos, y las transiciones flechas, éstas se activan en función de las variables de entrada, por ejemplo:



Las salidas, se representarán en los estados en la máquina de Moore, y en las transiciones en la máquina de Mealy (como en la figura anterior).

6.1.Biestables

Los circuitos secuenciales se van a diseñar a partir de los biestables, y se pueden clasificar según la lógica que utilizan en:

Biestable JK			
Entrada		Q _{antes}	Q _{después}
J	K		
0	0	Q	Q
0	1	Q	0
1	0	Q	1
1	1	Q	Q̄

Biestable D		
Entrada	Q _{antes}	Q _{después}
D		

0	Q	0
1	Q	1

Biestable T		
Entrada	Q _{antes}	Q _{después}
T		
0	Q	<u>Q</u>
1	Q	Q

6.2. Principales módulos estándar

Alguno de los principales módulos estándares de circuitos secuenciales son:

- **Registro:** Almacena un vector de bits mediante la conexión de biestables entre sí.
- **Contador:** Su salida es un número binario codificado según un sistema de representación específico (binario puro, en exceso M, etc.), el cual se irá incrementando de uno en uno hasta llegar a P-1, a partir del cual, reiniciará la cuenta.

6.3. Diseño de circuitos secuenciales

Las fases para diseñar de forma sistemática un circuito secuencial son:

1. Codificar las especificaciones de alto nivel en especificaciones binarias
2. Modelar una máquina de estados (de Mealy o de Moore) que describa el problema.
3. Representar la tabla de transición de estados, y de funciones de salida.
4. Simplificar las funciones de los estados y de las salidas mediante mapas de Karnaugh.

6.3.1. Técnicas de diseño actuales

Actualmente, no se puede abordar el diseño de un sistema complejo sin el uso de un HDL (lenguaje de descripción de hardware), para automatizar las tareas de diseño indicando especificaciones a alto nivel. Los HDL más utilizados son **VHDL**, **Verilog** y **ABEL** (SystemVerilog probablemente sea el próximo estándar de la IEEE).

7. Conclusión.

A modo de síntesis, se podría indicar que, una vez comprendida la técnica de los mapas de Karnaugh, la base para diseñar circuitos combinacionales y secuenciales es sencilla. Sin embargo, los circuitos de las máquinas comerciales actuales son de una enorme complejidad, es por ello que se hace necesario el uso de técnicas avanzadas de diseño asistidas por ordenador, tales como Verilog.

Actualmente con la aparición en el mercado tecnológico de dispositivos reprogramables como son los **FPGA** y **CPLD** y el de los lenguajes de descripción de hardware **VHDL** y **Verilog**, también se podría decantar el estudio de este tema desde el punto de vista del diseño software de los sistemas digitales tanto secuenciales como e incluso para el montaje de pequeños circuitos utilizando dispositivos microcontroladores programables (**PIC**) como **Arduino** o **Raspberry Pi Zero**.

7.1. Sistema educativo

Este tema no tiene una aplicación directa en ningún módulo profesional, aunque podría estudiarse de forma complementaria en:

Grado Medio

- Montaje y Mantenimiento de Equipos (SMR) (PES/SAI)

Grado Superior

- Sistemas informáticos (DAM / DAW) (PES/SAI)
- Fundamentos de hardware (ASIR) (PES/SAI)

8. Bibliografía

- De Anasagasti, Miguel. "Fundamentos de la Computadora" 9^aed 2004 Edt. Paraninfo
- Patterson D.A. y Hennessy JL. "Estructura y diseño de computadoras: la interfaz hardware/software" 4^a Ed. (2005) Edt McGraw-Hill
- Prieto A, Lloris A, Torres JC. "Introducción a la Informática" 4^aed. (2006) Edt. McGraw-Hill
- Jiménez Cembreras, Isabel M^a "Sistemas Informáticos" 2^aEd (2018) Edt. Garceta
- Morris Mano, Diseño Digital, (2017) Edt Pearson

