Apuntes
Electrónica Digital

Egileak: Iker G., Julen A., Ander P.

Índice:

Introducción:

En este documento vamos a hacer los apuntes necesarios para entender lo que es la electrónica digital, es decir, vamos a crear unos apuntes para poder tener una base sobre la electrónica digital. La finalidad de estos apuntes es poder llevar a cabo el montaje de un coche sigue-líneas con estos apuntes y una guía aparte.

Apuntes:

Lo primero que tenemos que saber es que la electrónica se divide en dos tipos diferentes: la analógica y la digital. Ahora que sabemos que hay dos tipos de electrónica, vamos a ver cual es la diferencia entre ambas.

Diferencia entre analógica y digital:

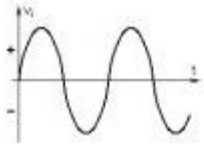
Primero vamos a explicar brevemente en qué consiste cada una de ellas:

- Electrónica analógica: Es la parte de la electrónica que trabaja con señales analógicas, es decir, que trabaja con corrientes y tensiones que varían continuamente de valor en el transcurso del tiempo, o con señales que siempre tienen el mismo valor de tensión y de intensidad.

- Electrónica digital: Es la parte de la electrónica que trabaja con señales digitales, es decir que trabaja con valores de corrientes y tensiones eléctricas que solo pueden poseer dos estados en el transcurso del tiempo.

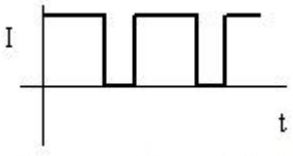
Sabiendo esto podemos diferenciar los dos tipos de electrónica dependiendo de las señales con las que trabajan:

- Señales analógicas:



Corriente alterna (AC) Corriente continua (DC)

- Señales digitales:



Corriente alterna pulsatoria (2 valores)

Ahora que ya sabemos diferenciar ambas electrónicas, podemos empezar a descubrir la electrónica digital.

Electrónica digital:

Como hemos dicho anteriormente, la electrónica digital es la parte de la electrónica que trabaja con las señales digitales, las cuales, solo pueden poseer dos estados, normalmente: 0 o 1. Cuando nos referimos al 0 de una señal digital, nos referimos al valor mínimo de la corriente, y cuando nos referimos al 1 al máximo.

Ahora, vamos a hablar de los dos grupos de sistemas digitales que existen: la combinacional y la secuencial.

- Combinacional: En este tipo de sistemas, las salidas en cualquier instante de tiempo dependen del valor de las entradas en ese mismo instante de tiempo.

- Secuencial: En este tipo de sistemas, la salida del sistema va a depender del valor de las entradas en ese instante de tiempo y del estado del sistema.

Por esto, podemos decir que los sistemas combinacionales son **sistemas sin memoria**, y que los sistemas secuenciales son **sistemas con memoria.**

Ahora ya sabemos lo más básico de la electrónica digital, para empezar a estudiarla, vamos a usar las puertas lógicas y algunas operaciones lógicas. Para ello, primero vamos a explicar lo que es una variable binaria:

- Variable binaria: Es una variable que sólo puede tener dos valores (por ejemplo 1 o 0), y los cuales se refieren a dos estados diferentes (por ejemplo encendido o apagado).

Podemos usar este tipo de variable para saber el estado en el que se encuentra uno de los elementos que tengamos en nuestro sistema, por ejemplo, podemos decir que una lámpara tiene un valor de 1 cuando está encendida y un valor de 0 cuando está apagada.

Como en la mayoría de los sistemas electrónicos, tenemos elementos de salida y de entrada, los cuales, sólo van a tener dos estados y con los que vamos a “jugar” a la hora de hacer nuestros sistemas.

Tenemos que tener en cuenta que cuando hablamos de los estados y de darles un valor, estamos hablando de el cambio de estado del elemento y no del trabajo que esté haciendo. Imaginemos que tenemos un interruptor, si en reposo (valor 0) está abierto, cuando lo accionemos (valor 1) estará cerrado, pero si el interruptor funcionase al revés, es decir, en reposo (valor 0) está cerrado, cuando lo accionemos (valor 1) estará abierto.

Ahora que entendemos cómo usar las variables binarias, vamos a ver los sistemas en los que las usaremos.

Sistema combinacional:

Las salidas de este tipo de sistemas son el valor de sus entradas en un momento dado, sin que entren estados anteriores de las entradas o de las salidas. En electrónica digital la lógica combinacional está formada por ecuaciones simples a partir de las operaciones básicas del álgebra de Boole.

Estos circuitos están compuestos únicamente por puertas lógicas interconectadas entre sí, sin ningún biestable o celda de memoria, lo que los hace sistemas que carecen de memoria alguna.

Sistema secuencial:

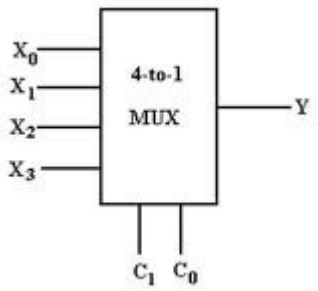
Hay momentos que los valores de las salidas no dependen exclusivamente de los valores de las entradas sino también dependen del valor anterior o el estado interno.

Multiplexores y demultiplexores

En el diseño de circuitos digitales es habitual encontrarse de forma reiterada con las mismas necesidades que requieren las mismas soluciones. Esto da lugar a un conjunto de subcircuitos que aparecen con mucha asiduidad a la hora de proporcionar una solución a un problema. Estos elementos se estudian en profundidad ya que por una parte ahorran tiempo de diseño (no debe "reinventarse la rueda") y por otra parte están disponibles con circuitos integrados comerciales o en forma de bloques en los programas de diseño electrónico.

Multiplexores

Este es el primero de estos elementos o subcircuitos comunes a muchos diseños. Un multiplexor es un circuito combinacional de varias entradas de señal y una única salida. Dispone además de unas entradas de selección que permiten redirigir cualquiera de las entradas elegidas a la salida. En sentido estricto, las entradas de selección se deben considerar exactamente igual que las otras entradas de señal, pero debido a su particular propósito ("selección") suelen dibujarse separadas. En la figura siguiente se muestra un multiplexor de 4 a 1:



Multiplexor 4 a 1

Se suele indicar 4 a 1 para proporcionar el número de entradas de señal que puede manejar (X0, X1, X2 y X3). Como se ha comentado antes, las entradas de selección (C1 y C0) aunque son propiamente entradas del circuito, se dibujan en un lateral (en la parte inferior en este caso) con el objeto de clarificar su función.

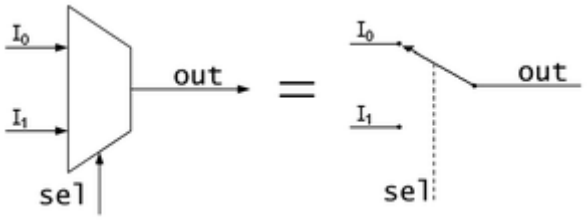
Por otra parte, se puede observar que una vez fijado el número de entradas, queda unívocamente fijado el número de entradas de selección ya que si dispone de N entradas de selección se podrán seleccionar 2^N entradas de señal. La

función lógica de este circuito es muy sencilla de obtener (incluyendo únicamente los términos que dan lugar a 1) y es la siguiente :

- Y = X0·C1'·C0′ + X1·C1'·C0 + X2·C1·C0′ + X3·C1·C0

Por ejemplo, el primer término refleja que la salida será 1 si X0 = 1 y la combinación C1C0 = 00, que corresponde a la selección de la entrada X0. y así para los otros 3 términos de la suma de productos.

Utilizando este método, se puede obtener la función algebraica de cualquier multiplexor. En la figura siguiente se muestra el símbolo que normalmente se utiliza para representar multiplexores, en este caso, particularizado para un multiplexor 2 a 1



Multiplexor 2 a 1 y circuito eléctrico equivalente

En la parte derecha de la figura se observa el equivalente eléctrico del multiplexor. Se puede observar que tiene forma de trapecio girado 90º. El circuito de selección actúa como un conmutador que conecta la entrada requerida a la salida para que pueda fluir la señal.

Demultiplexores

Los demultiplexores efectúan la operación inversa de los multiplexores y se utilizan para distribuir una señal que llega por una única entrada a un conjunto de salidas. De la misma forma, las entradas de selección permite elegir a cual de las salidas se redirige la señal de entrada.

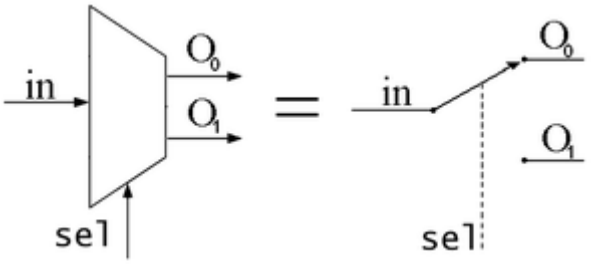
En este caso, se cumple igualmente que N entradas de selección va a permitir distribuir la señal en 2^N salidas. Si consideramos una única entrada X y un conjunto de salidas Y0, Y1, Y2, Y3, ya se puede deducir que tendrá dos entradas de selección,por ejemplo, S0 y S1.

Las funciones de cada una de las salidas se pueden expresar de forma independiente y son las siguientes:

- Y0 = X·S1'·S0′
- Y1 = X·S1'·S0
- Y2 = X·S1·S0′
- Y3 = X·S1·S0

En la que por ejemplo, la salida Y0 sólo tiene un 1 cuando la entrada X=1 y la combinación S1S0 = 00 indica que es la salida seleccionada. Al igual que en el caso anterior, el patrón de cada función algebraica permite desarrollar las funciones de salida para cualquier demultiplexor.

El símbolo genérico de este elemento digital se muestra en la siguiente figura particularizado para un demultiplexor 1 a 2. Se puede observar que tiene también la forma de un trapecio (girado 90º).



Símbolo del demultiplexor y equivalente eléctrico (Extraído de la Wikipedia)

En la parte derecha se muestra el equivalente eléctrico del demultiplexor. Funciona como un conmutador que permite conectar la entrada con la salida deseada para que fluya la señal.

Codificador y Decodificador:

En un sentido general, se puede decir que un codificador es un circuito hecho para pasar información de un sistema a otro con clave diferente, y en tal caso un decodificador sería el circuito o dispositivo que retorne los datos o información al primer sistema. Debido a que el caso que nos ocupa es el de la lógica digital, y en especial la aritmética binaria, hemos de dar sentido más directo a los términos «codificador» y «decodificador».

Un codificador es un bloque combinacional hecho para convertir una entrada no binaria en una salida de estricto orden binario. En otras palabras, es un circuito integrado por un conjunto de componentes electrónicos con la habilidad para mostrar en sus terminales de salida un word binario (01101, 1100, etc.), equivalente al número presente en sus entradas, pero escrito en un código diferente. Por ejemplo, un Octal-to-binary encoder es un circuito codificador con ocho entradas (un terminal para cada dígito Octal, o de base 8) y tres salidas (un terminal para cada bit binario).

Operaciones lógicas:

Las operaciones lógicas, también llamadas álgebra de boole, son las operaciones matemáticas que se usan en el sistema binario, sistema de numeración que solo usa el 0 y el 1. Para poder trabajar con este sistema, nosotros vamos a usar las variables binarias que hemos estado explicando anteriormente.

Para entender las operaciones lógicas, vamos a empezar con las dos operaciones básicas (las cuales son la base del resto de operaciones. En estas operaciones vamos a utilizar dos variables binarias:

Nosotros hemos decidido llamar a las variables: ***a*** y ***b***.

a + b

0+0=0
0+1=1
1+0=1
1+1=1

a x b

0x0=0
0x1=0
1x0=0
1x1=1

Estas dos son las operaciones en las que nos vamos a centrar, la suma de las variables y la multiplicación de las mismas.

- La suma: En este caso podemos observar que con que una sola de las variables sea un 1, el resultado final será un 1.

- La multiplicación: Al contrario que en la suma, en este caso podemos ver que con que una sola de las variables sea un 0, el resultado final será un 0.

Al ser operaciones matemáticas se basan en los principios de estas.

Ahora que conocemos estas dos operaciones, podemos empezar con las puertas lógicas.

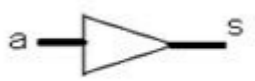
Puertas lógicas:

Las puertas lógicas son componentes electrónicos representados por un símbolo con una o dos entradas y una sola salida. Estos elementos, realizan una función (operación lógica) para dar un valor a la salida, recogiendo los valores que tengan en las entradas. Hay muchos tipos de puertas lógicas, por ahora vamos a explicar las siguientes:

- Puerta lógica “igualdad”. (=)
- Puerta lógica “NO” o “NOT”. (!=)
- Puerta lógica “O” u “OR”. (+)
- Puerta lógica “AND”. (x)
- Puerta lógica “NOR”. (!+)
- Puerta lógica “NAND”. (!x)

Puerta lógica igualdad:

La puerta lógica igualdad, da siempre a la salida, el valor de la entrada, por lo cual:



Donde **S = a**

Entrada a	Salida S
0	0
1	1

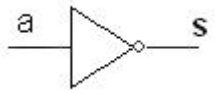
Tabla de la verdad

Algo de lo que no habíamos hablado hasta ahora es la **tabla de la verdad**, esta tabla representa los valores que tienen las entradas y salidas de la puerta teniendo en cuenta la operación lógica que utiliza.

Para entender esta puerta lógica, podemos usar un ejemplo: Nosotros tenemos una lámpara (**S**) la cual es controlada por un interruptor (**a**), nuestro interruptor en estado de reposo, está abierto (valor 0), por lo que nuestra lámpara estará apagada (valor 0) y cuando nuestro interruptor esté accionado (valor 1), la lámpara estará encendida (valor 1).

Puerta lógica NO o NOT: (Negación)

La puerta lógica NOT, es una puerta en la que la salida siempre es contraria al valor de la entrada, por lo que:



Donde **S** = **!a**

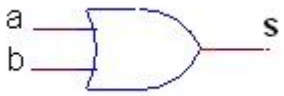
Entrada a	Salida S
0	1
1	0

Tabla de la verdad

Para entender esta puerta lógica vamos a poner un ejemplo. Imaginemos que tenemos un interruptor normalmente cerrado (NA) (Valor 0), y tenemos una lámpara controlada por el interruptor, por lo que, cuando el pulsador esté en reposo (Valor 0), la lámpara estará encendida (Valor 1), y cuando el interruptor esté accionado (Valor 1) la lámpara estará apagada (Valor 0).

Puerta lógica O u OR: (Suma)

La puerta lógica OR, es una puerta que necesita (al menos) dos entradas, las cuales simula como si estuvieran conectadas en paralelo, por eso:



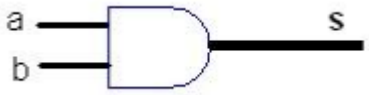
Donde **S** = **a** + **b**

Entrada a	Entrada b	Salida S
0	0	0
0	1	1
1	0	1
1	1	1

Para entender esta puerta lógica vamos a poner un ejemplo. Imaginemos que tenemos dos pulsadores, los cuales controlan una lámpara, nosotros queremos que cuando cualquiera de los dos interruptores esté pulsado la lámpara se encienda, por lo que si el interruptor 1 está sin accionar (Valor1 0) y el interruptor 2 está sin accionar (Valor2 0) la lámpara estará apagada. Pero en cuanto accionemos cualquiera de los interruptores (da igual cual) (Valor1 1 y Valor2 0)la lámpara estará encendida (Valor 1).

Puerta lógica AND: (Multiplicación)

La puerta lógica AND, es una puerta que necesita (al menos) dos entradas, las cuales simula como si estuvieran conectadas en serie, por eso:



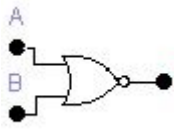
Donde **S** = **a * b**

Entrada a	Entrada b	Salida S
0	0	0
0	1	0
1	0	0
1	1	1

Para entender esta puerta lógica, vamos a utilizar un ejemplo. Imaginemos que tenemos dos interruptores los cuales controlan una puerta de seguridad y queremos que la puerta necesite pulsar ambos interruptores a la vez, entonces si solo pulsamos el interruptor 1 (Valor1 1) y el interruptor 2 no (Valor2 0) la puerta estaría cerrada (Valor 0) pero si pulsamos ambos interruptores a la vez (Valor1 1 y Valor 2 1) la puerta estaría abierta (Valor 1).

Puerta lógica NOR: (Suma invertida)

La puerta lógica NOR, hace lo contrario que la puerta lógica OR, es decir, hace la suma de las entradas y el valor de la salida será el contrario al resultado de la suma, por eso:



Donde **S !=* a + b**

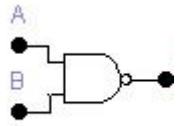
* al poner "!" delante de "=" hacemos que sea lo contrario en vez de el igual.

Entrada a	Entrada b	Salida S
0	0	1
0	1	0
1	0	0
1	1	0

Basándonos en el ejemplo que hemos usado para entender la puerta OR, imaginemos que tenemos dos interruptores, los cuales controlan una lámpara, en este caso la lámpara **solo se encendería (Valor 1)** cuando **ambos interruptores estén en reposo (Valor 0).** En cualquier otro caso la lámpara estaría apagada.

Puerta lógica NAND: (Multiplicación invertida)

La puerta lógica NAND, hace lo contrario a la puerta lógica AND, es decir hace la multiplicación de las entradas y el valor de la salida será el contrario al resultado de la multiplicación, por eso:



Donde **S != a * b**

Entrada a	Entrada b	Salida S
0	0	1
0	1	1
1	0	1
1	1	0

Basándonos en el ejemplo que hemos usado para entender la puerta AND, imaginemos que tenemos dos interruptores los cuales controlan una luz de emergencia, por lo cual queremos que la luz esté encendida sólo cuando no haya ningún interruptor accionado, este es nuestro caso, esta luz **solo se enciende (Valor 1) cuando ambos interruptores están apagados (Valor1 0 y Valor 2 0).**

Ahora que entendemos cómo funcionan las puertas lógicas, vamos a explicar cómo combinarlas para conseguir lo que llamamos un **circuito lógico combinacional**.

Puerta lógica XOR: (OR exclusiva)

La puerta lógica XOR, funciona de la siguiente manera: La salida SOLO será un 1 cuando solamente una de las entradas es un 1, es decir, siempre que las entradas sean diferentes, las salida será un 1, y mientras las entradas sean iguales, la salida será un 0.

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Donde: $A \cdot \overline{B} + \overline{A} \cdot B \equiv (A + B) \cdot (\overline{A} + \overline{B})$

Para entender mejor cómo funciona esta puerta lógica, vamos a poner un ejemplo: Imaginemos que tenemos una lámpara la cual es controlada desde dos puntos diferentes, y en ambos puntos tenemos un interruptor para apagarla, para apagar la lámpara, basta con pulsar uno de los interruptores.

Puerta lógica XNOR: (NOR exclusiva)

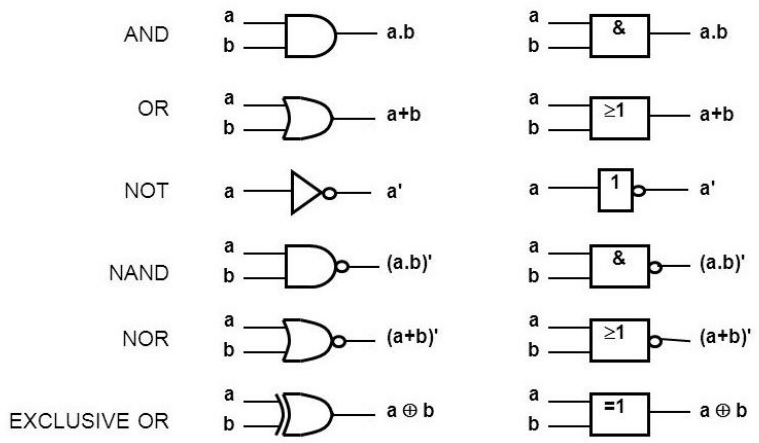
La puerta lógica XNOR, hace la misma labor que la XOR pero dando un resultado negado, es decir, lo contrario a lo que nos da la XOR, por lo que en la salida sólo será un 1 cuando ambas entradas sean iguales, en caso contrario la salida será un 0.

INPUT		OUTPUT
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

Donde: $A \cdot B + A \cdot \overline{B} = A \oplus B$.

Para entender mejor esta puerta lógica vamos a poner un ejemplo: Imaginemos que tenemos una puerta, la cual estará abierta siempre que los guardias que hemos puesto estén de acuerdo, cuando los dos guardias que tiene la puerta estén en desacuerdo para dejar pasar a alguien la puerta se cerrará.

Simbología de las puertas lógicas:



Tradicional ANSI

Circuitos lógicos combinacionales:

Los primero que vamos a explicar es cómo hacer una tabla de la verdad, estas tablas nos van a ayudar a saber cuantas entradas tenemos y cuál es el estado de cada una, y a su vez, el estado que tendrá la salida en cada uno de los casos.

Tablas de la verdad:

Como hemos dicho antes, lo primero que tenemos que hacer es saber el número de entradas que vamos a tener, una vez tengamos las entradas las nombraremos con letras (a, b, c…) y añadiremos la salida a la que llamaremos “S”. Lo siguiente que tenemos que hacer es rellenar la tabla poniendo los diferentes estados de las entradas, y el respectivo estado de la salida.

Para entenderlo mejor vamos a usar un ejemplo:
Tenemos dos interruptores los cuales controlan una caja fuerte, por lo cual, tenemos dos entradas (a y b) y una salida (S). Queremos que la caja se abra sólo cuando pulsamos ambos interruptores a la vez, por lo que sabemos que para que la salida (S) sea un 1, ambas entradas (a y b) tienen que estar accionadas, es decir el valor de ambas debe ser un 1. Sabiendo esto, podemos hacer la tabla:

Primero haremos la tabla en sí, por lo que haremos una tabla con 2 entradas y 1 salida (como cada entrada tiene 2 posibles estados, la tabla tendrá 4 posiciones diferentes):

Entrada “a”	Entrada “b”	Salida “S”

Ahora tenemos que rellenar la tabla con los posibles estados de las entradas:

Entrada “a”	Entrada “b”	Salida “S”
0	0	
1	0	
0	1	
1	1	

Para terminar, tenemos que poner el estado que tendrá la salida dependiendo del valor que tengan las entradas:

Entrada “a”	Entrada “b”	Salida “S”
0	0	0
1	0	0
0	1	0
1	1	1

Una vez hayamos hecho la tabla de la verdad, vamos a sacar la **función lógica** de nuestro problema.

Sacar la función lógica:

El siguiente paso es sacar la función lógica, para ello usaremos la tabla de la verdad que hemos hecho en el anterior paso.

Vamos a coger todas las opciones en las que la salida sea un 1, en nuestro caso solo hay una, la última. Ahora vamos a multiplicar las entradas, poniendo invertidas las entradas con el valor 0 y normal las que tengan el valor 1 (en nuestro caso solo tenemos dos 1). Por lo que nuestra función quedaría así:

S = a * b

Hay casos en los que nos encontraremos con más de una opción con un 1 en la salida, en estos casos podemos usar dos métodos diferentes:

- Suma de productos (minterms): En este método, vamos a hacer una suma de las funciones de cada línea con un 1 como salida, es decir haremos el mismo procedimiento de antes pero al final juntaremos todas las funciones que tengamos en una suma.

- Productos de sumas (maxterms): En este método, haremos funciones de las líneas en las que tengamos un 0 como salida, y sumaremos el valor de las entradas invirtiendo las que tengan como valor un 1, para terminar multiplicaremos estas sumas.

Para entender mejor estos métodos vamos a poner un ejemplo de una tabla de la verdad con varias opciones:

A	B	C	S
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

Esta es la tabla de la verdad en la que vamos a basarnos para usar los métodos explicados antes.

Método 1: “minterms”

Con este método tenemos que tener en cuenta que a la hora de hacer las funciones tenemos que poner a la inversa las entradas que tengan como valor un 0.

En este caso tenemos que hacer las funciones de las líneas con un 1 en la salida, en este caso tendríamos 5 funciones:

S = !a * !b * !c
S = !a * !b * c
S = !a * b * !c
S = a * !b * c
S = a * b * c

Ahora vamos a juntar todas las funciones en una sola, por lo que:

S = !a * !b * !c + !a * !b * c + !a * b * !c + a * !b * c + a * b * c

Esta sería la función de esta tabla de la verdad usando el método “minterms”.

Método 2: “maxterms”

Con este método tenemos que tener en cuenta que a la hora de hacer las funciones, tenemos que poner a la inversa las entradas que tengan como valor un 1.

En este caso tenemos que hacer las funciones de las líneas con un 0 en la salida, en este caso tendríamos 3 funciones:

S = a + !b + !c
S = !a + b + c
S = !a + !b + c

Ahora vamos a juntar las funciones en una sola, por lo que:

S = (a + !b + !c) * (!a + b + c) * (!a + !b + c)

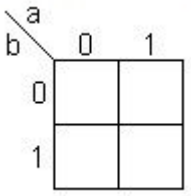
A estas funciones tan largas las llamamos **Expresión Canónica de la Función Lógica.** Una vez tenemos estas expresiones las podemos simplificar usando el **método de karnaugh**.

Simplificación de las funciones con el método de Karnaugh:

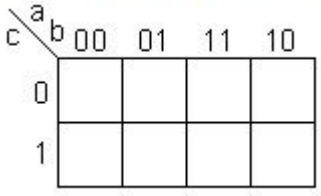
Para poder usar el método de Karnaugh, lo primero que tenemos que saber es cómo hacer una tabla de Karnaugh. Lo primero que tenemos que saber es que la cantidad de casillas de la tabla la define la cantidad de variables que tengamos, para eso podemos usar la siguiente fórmula: 2^n. Donde "n" es el número de variable que tengamos, por lo que si tenemos, por ejemplo, 3 variables, tendremos una tabla con 8 casillas.

Ahora que sabemos cómo conseguir el número de casillas que vamos a necesitar vamos a explicar cómo situar estas casillas. A la hora de hacer la tabla tenemos tanto columnas como filas, para saber cómo van situadas las casillas tenemos que ver algún ejemplo:

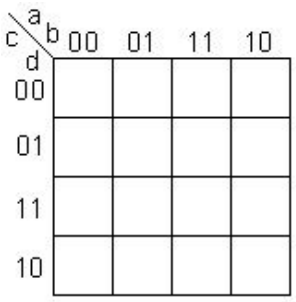
En el caso de tener 2 variables, la tabla quedaría así:



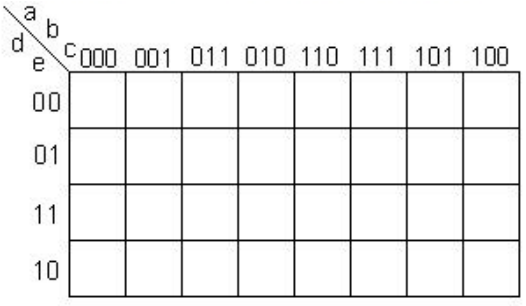
En el caso de tener 3 variables, la tabla quedaría así:



En el caso de tener 4 variables, la tabla quedaría así:



En el caso de tener 5 variables, la tabla quedaría así:



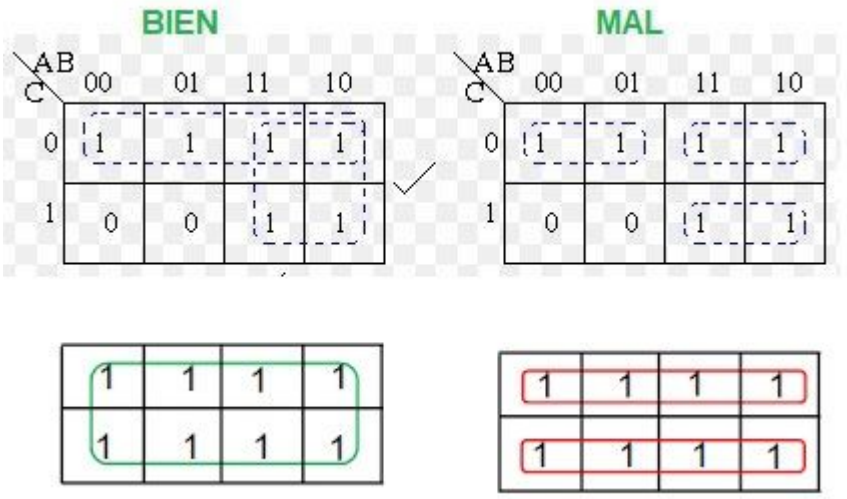
Tras ver estos ejemplos, nos damos cuenta de que situamos las variables primero en las columnas y luego en las filas y que, cuando tenemos más de dos variables, empezamos a tener más de una variable en las columnas/filas.

Para entender las tablas tenemos que entender los números que situamos por encima y a la izquierda de la tabla, estos números representan en valor de las variables, en caso de ser 0 representaría la variable invertida. Los valores de las variables están representadas en el orden en el que vemos situadas las variables.

Podemos ver que los números que representan los valores no siguen un orden "lógico" pero en realidad sí que lo hace. En realidad tienen ese orden por el hecho de que en cada paso **sólo puede cambiar de valor una de las variables**.

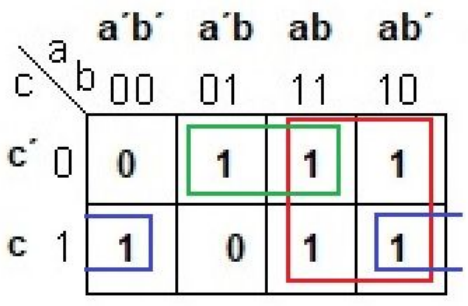
Ahora que sabemos cómo se hacen las tablas de Karnaugh, tenemos que saber cómo rellenarlas. Este paso es sencillo, ya que lo único que tenemos que hacer es **rellenar la tabla con el valor que tenga la salida** respecto al valor de las entradas. Para hacer esto, podemos usar tanto la tabla de la verdad como la la función de la expresión canónica.

El siguiente paso se llama agrupación de unos. En este paso lo que hacemos es juntar en cuadrados o rectángulos **únicamente los unos**, y siempre en grupos de 2^n (es decir, 2, 4, 8…)



También tenemos que tener en cuenta que podemos usar el extremo de la tabla para empezar por el otro extremos, y un par de casos especiales más.

Para sacar la función de las agrupaciones de unos tenemos que ir fijándonos en las agrupaciones que hemos hecho, y cada agrupación nos dará una función en cuestión. En cada agrupación nos tenemos que fijar si las variables cambian de valor, si este es el caso, la variable será eliminada de la función:



3 grupos de UNOS, Verde y Azul de dos UNOS y Rojo de 4.

Verde: a cambia y b y c quedan como están = término para la función = b x c´

Azul: a cambia y b y c no cambian = término para la ecuación = b´ x c

Rojo: c y b cambian y solo a no cambia = término para la ecuación = a

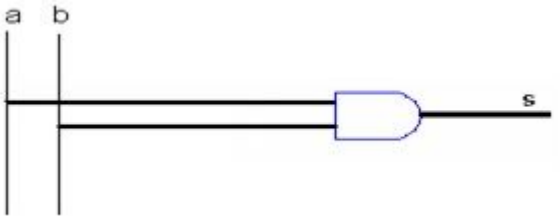
La Ecuación Simplificada Será:

S = b x c´ + b´x c + a

Con esto habríamos simplificado nuestra función con el método de minterms, para hacerlo en maxterms tendríamos que agrupar los 0 en vez de los 1 y tendríamos que usar la lógica negativa y el producto de sumas. Ahora vamos a explicar los circuitos lógicos con puertas lógicas.

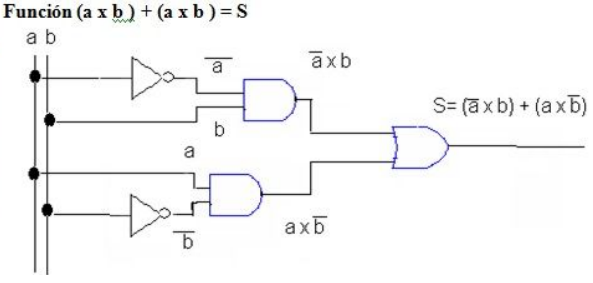
Circuitos lógicos con puertas lógicas:

Una vez tenemos nuestra tabla de la verdad y nuestra función lógica, lo siguiente es hacer el circuito lógico usando las puertas lógicas explicadas anteriormente. Para eso, lo primero que vamos a hacer es poner tantas líneas verticales como variables de entrada tengamos (en el caso del ejemplo vamos a usar dos variables) y sacamos líneas horizontales para cada variable del producto de la función hasta la puerta que tengamos que usar.



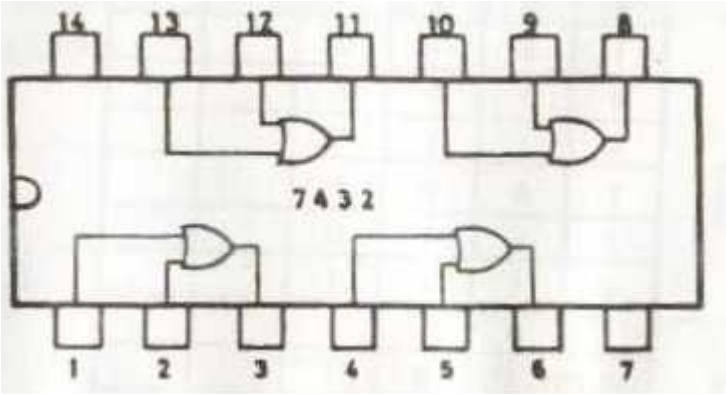
De esta forma tendríamos nuestro primer circuito lógico terminado. Obviamente, este circuito en muy sencillo, por ello vamos a poner otro ejemplo un poco más complicado:

Entrada a	Entrada b	Salida S
0	0	0
0	1	1
1	0	1
1	1	0



En este caso hemos añadido la tabla de la verdad del circuito junto a este.

Hay que tener en cuenta que las puertas lógicas no suelen venderse por separado, sino que se venden incorporadas en los llamados circuitos integrados.



Este es por ejemplo el circuito integrado 7432 el cual tiene 4 puertas lógicas OR y las patillas para la alimentación.

Para ver la información completa podéis visitar las páginas web añadidas en la bibliografía.

Características de las puertas lógicas:

Antes de hablar de las características de las puertas lógicas tenemos que diferenciarlas en varios grupos:

- RTL
- DTL
- ECL
- TTL
- IIL
- CMOS
- NMOS
- BiCMOS

Estos grupos son lo que llamamos familias lógicas, cada una representa uno de los posibles diseños para el montaje de los circuitos integrados.

Las características más importantes de las puertas lógicas son las siguientes:

- Características de transferencia: Cuando sepamos el número de puertas lógicas que vamos a conectar a la salida, la tensión de alimentación y la temperatura, podremos conseguir una curva que nos muestra la relación entre la tensión de salida y la de entrada. De esta curva destacan una serie de valores que debemos que tener en cuenta:
    - **VIL**: Es la tensión de entrada requerida para un nivel lógico bajo en la entrada de la puerta. Es decir, será el valor máximo de tensión permisible para el 0.
    - **VIH**: Es la tensión de entrada necesaria para obtener un nivel alto a la entrada de la puerta. Al contrario que el valor anterior, éste será la tensión mínima permisible para tener un 1.
    - **VOL**: Es la tensión de salida de la puerta en nivel bajo.
    - **VOH**: Es la tensión de salida en nivel alto.

- Características de entrada: Estas características son necesarias para sacar el mayor rendimiento posible de las puertas lógicas. Como en el caso anterior, tenemos dos valores fundamentales, que se utilizarán para los distintos diseños:
    - **IIL**: Es la corriente que sale por la entrada de una puerta cuando se encuentra a nivel bajo.
    - **IIH**: Es la corriente que entra por la entrada de una puerta cuando se halla a nivel alto.

- Características de salida: Normalmente, la tensión de salida de una puerta se establece como resultado de combinar las características de salida de esa puerta con las de entrada de las siguientes puertas conectadas a ella. Caben destacar dos valores:
    - **IOL**: Es la corriente de salida que entra cuando ésta se halla a nivel bajo. (**Isink**)
    - **IOH**: Es la corriente que fluye cuando la salida se encuentra a nivel alto. (**Isource**)

- Características en régimen transitorio: Esta característica es la que llamamos velocidad de conmutación de las puertas y es la que nos dice si nuestro sistema reacciona con mayor o menor rapidez. Aquí encontramos cuatro valores a destacar:
    - **tpHL**: Es el tiempo de retraso en una transición a la salida desde un nivel alto a un nivel bajo.
    - **tpLH**: Es idéntico al anterior pero cuando hay una transición de nivel bajo a alto.
    - **tr**: También llamado tiempo de subida, nos mide el momento en que la señal pasa desde un 10% del valor final hasta el instante que alcanza el 90%, en una transición de nivel bajo a alto.
    - **tf**: También llamado tiempo de bajada, que es igual al anterior pero en un cambio de nivel alto a bajo.

- Capacidad de carga: Es la característica que nos indica el número de puertas lógicas adicionales que puedes conectar a la salida de una, también tenemos que tener en cuenta la tensión de funcionamiento de cada una de las puertas a la hora de conectarlas.

Las familias más usadas son la TTL y la CMOS, las cuales vamos a explicar a continuación.

Familia TTL:

Esta familia usa la lógica del **"transistor - transistor"** (en inglés *"transistor -transistor logic"* de ahí TTL), es una tecnología para construir circuitos digitales, la cual, usa transistores bipolares como entrada y salida del circuito.

Las características principales de la familia TTL son las siguientes:

- Tensión: La tensión nominal de funcionamiento de los circuitos construidos con la lógica TTL tiene como rango entre 4,75V y 5,25V, normalmente se usan 5V para su funcionamiento.

- Niveles lógicos: Los circuitos fabricados con esta tecnología tienen los niveles lógicos definidos por el rango de tensión comprendida, con un rango entre 0V y 0,8V para el estado bajo y de 2,2V y Vcc para el estado en alto.

- Velocidad de transmisión: Esta característica es la parte fuerte y la débil de esta familia, ya que, tiene una velocidad de transmisión muy elevada la cual puede llegar hasta los 400 MHz, pero esto hace que el consumo de esta familia sea de los más elevados, por ello hay más de una versión de los circuitos.

- Calidad de la señal: Las señales de salida TTL se degradan rápidamente si no se transmiten a través de circuitos adicionales de transmisión, es decir, no pueden viajar más de 2 m por cable sin graves pérdidas.

Familia CMOS:

Esta familia usa el funcionamiento de los transistores de efecto de campo, o MOSFET, para conseguir las funciones lógicas

Las características principales de esta familia son las siguientes:

- Consumo: Gracias a la alta impedancia de entrada de los transistores que utiliza (MOSFET) le permite tener un bajo consumo de potencia estática.

- Calidad de la señal: Los circuitos CMOS son robustos frente a ruido o degradación de señal debido a la impedancia del metal de interconexión.

- Simplicidad: Los circuitos fabricados con este método son sencillos de diseñar.

- Coste: Dado que la tecnología de fabricación está muy desarrollada, podemos conseguir los materiales a un precio más asequible que los demás materiales.

- Velocidad de cálculo: Dado que este sistema utiliza transistores MOSFET, la velocidad de comunicación entre ellos es más lenta que con otras tecnologías, esto hace que la velocidad de cálculo de este sistema sea algo más lenta.

- Latch-up: Al usar transistores MOSFET, estos tienen la posibilidad de encontrarse con el fallo que llamamos *"Latch-up"* el cual puede estropear el circuito completamente, para evitarlo tenemos que utilizar los métodos de diseño adecuados.

- Tensión: La tensión de alimentación de los dispositivos fabricados con este método tienen un gran rango, el cual es desde los 3V hasta los 18V.

Biestables:

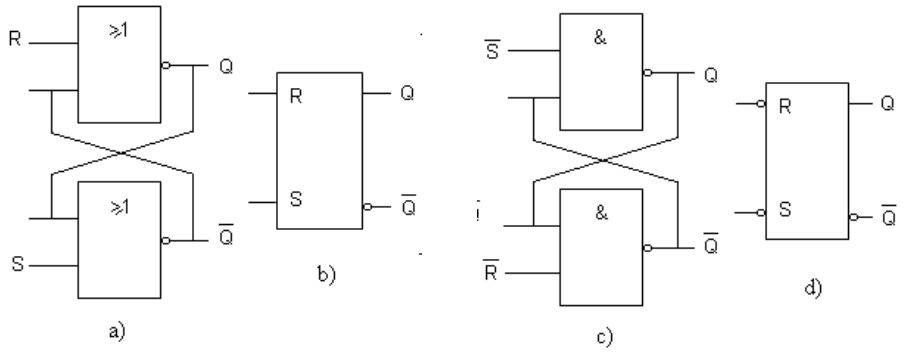
En electrónica, un biestable es un circuito que tiene dos estados estables y puede almacenar información. Se puede hacer que cambie de estado aplicando distintas señales a las diferentes entradas, a las cuales llamamos entradas de control, además estos circuitos suelen tener una o dos salidas. Es el elemento de almacenamiento básico en lógica secuencial. Los circuitos biestables tienen la capacidad de permanecer en uno de dos estados posibles durante un tiempo indefinido en caso de no recibir ningún cambio en las entradas.
Ahora vamos a explicar algunos de los biestables más utilizados:

- Biestable R-S.
- Biestable D.
- Biestable T.
- Biestable JK.

Biestable R-S:

Podemos encontrar dos tipos de biestables R-S: los **biestables R-S asíncronos** y los **biestables R-S síncronos**.

Biestable R-S asíncrono:
Este biestable tiene solamente dos entradas de control: Reset (R) y Set (S). Está compuesto por dos puertas lógicas, estas puertas pueden ser NAND o NOR.



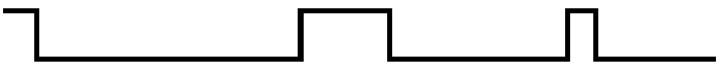
Aquí podemos ver un biestable R-S compuesto por dos puertas NOR (**a** y **b**) y otro compuesto por dos puertas NAND (**c** y **d**). [**a** y **c** representan **b** y **c** internamente]

La tabla de la verdad de un biestable R-S sería la siguiente:

R	S	Q (NOR)	Q (NAND)
0	0	q	N. D.
0	1	1	0
1	0	0	1
1	1	N. D.	q
N. D.= Estado no deseado q= Estado de memoria			

Para entender mejor el funcionamiento de este biestable, vamos a representar los pulsos:

Esta señal representa la entrada S (Set)

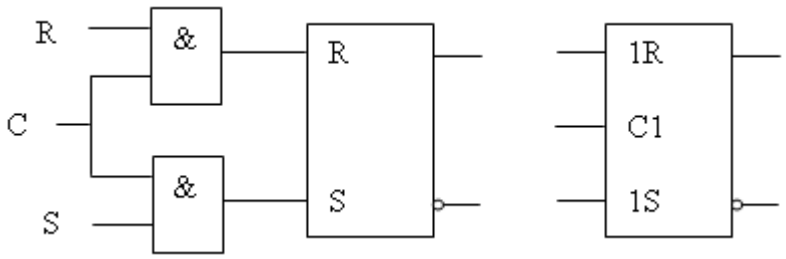


Esta señal representa la entrada R (Reset)

Esta señal representa la salida Q

No hemos representado la salida **Q negada** dado que esta sería la contraria a la señal de la salida **Q**.

Biestable R-S síncrono:
Este biestable tiene tres entradas de control: Reset (R), Set (S) y Clock (C). En este caso, además de tener el Set y el Reset, nos encontramos una entrada para el "Clock", esta entrada tendrá un pulso con una frecuencia constante, y es el pulso complementario para las otras dos entradas, es decir, para que las entradas de Set y Reset puedan actuar sobre la salida tendrán que estar en uno de los flancos (altos o bajos, dependiendo de cómo conectemos el Clock) del Clock.



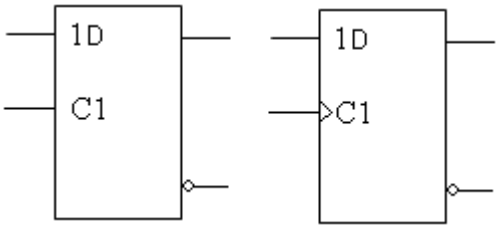
Este sería el esquema de un biestable R-S síncrono compuesto por dos puertas NAND.

La tabla de la verdad de este biestable sería el siguiente:

C	R	S	Q (NOR)
0	X	X	q
1	0	0	q
1	0	1	1
1	1	0	0
1	1	1	N. D.
X=no importa			

Biestable D:

Este biestable tiene una entrada de control (D) y otra para el Clock (C), además de dos salidas (Q y Q negada).



Aquí podemos ver el esquema de dos biestables D, la diferencia entre ellos es que el Clock se dispara en los flancos o en los niveles, esto los diferencia en dos tipos: Biestable D por nivel (Alto o Bajo) y Biestable D por flanco (Bajada o Subida).

La tabla de la verdad de ambos es la siguiente:

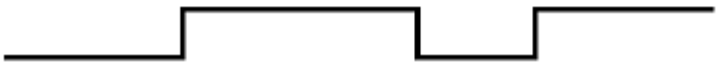
D	Q	$Q_{siguiente}$
0	X	**0**
1	X	**1**
X=no importa		

Los pulsos de este biestable serían los siguientes:

Esta señal representa la entrada de control C (Clock)

Esta señal representa la entrada de control D (Set)

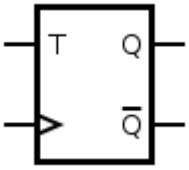
Esta señal representa la salida Q (en caso de ser un biestable D de nivel)



Esta señal representa la salida Q (en caso de ser un biestable D de flanco)

Biestable T:

El biestable T cambia de estado cada vez que la entrada de sincronismo o reloj se dispara mientras la entrada T está a nivel alto. Si la entrada T está a nivel bajo, el biestable retiene el nivel previo.



La tabla de la verdad de este biestable es la siguiente:

T	Q	$Q_{siguiente}$
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Para entenderlo mejor vamos a poner unos pulsos como ejemplo:

Esta señal representa la entrada de control T.

Esta señal representa la entrada del Reloj.

Esta señal representa la salida Q del biestable.

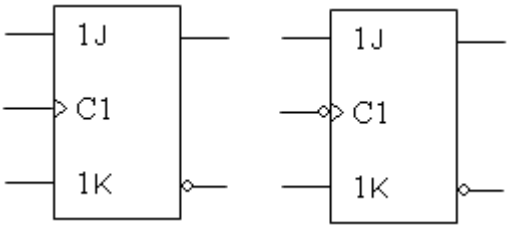
Biestable JK:

Es uno de los tipos de biestable más usados. Su funcionamiento es idéntico al del biestable S-R en las condiciones SET, RESET y de permanencia de estado. La diferencia está en que el biestable J-K no tiene condiciones no válidas como ocurre en el S-R.

La tabla de la verdad de este biestable es el siguiente:

J	K	Q	$Q_{siguiente}$
0	0	0	**0**
0	0	1	**1**
0	1	X	**0**
1	0	X	**1**
1	1	0	**1**
1	1	1	**0**
X=no importa			

Existen dos tipos de biestable JK: por flanco y Maestro-Esclavo. Dado que el segundo no se usa mucho, vamos a explicar únicamente el de flanco.



Estos son los dos tipos de biestable JK por flanco que podemos encontrar, la única diferencia entre ambos es que en uno de ellos la entrada C (clock) está negada y en el otro no. Cuando la entrada de control del Clock no está negada, llamamos a este tipo de biestable JK de flanco de subida; y en el caso contrario, les llamamos biestable JK de flanco de bajada.

Para entender el funcionamiento de este biestable vamos a usar un ejemplo com pulsos:

Biestable JK de flanco de bajada:

Este sería la señal de la entrada de control C (Clock)

Esta señal representa la entrada de control J (Set)

Esta señal representa la entrada de control K (Reset)

Esta señal representa la salida Q

En el caso del biestable JK de flanco de subida:

Esta señal representa la entrada de control C (Reloj)

Esta señal representa la entrada de control J (Set)

Esta señal representa la entrada de control K (Reset)

Esta señal representa la salida Q

Sistema decimal, binario, hexadecimal y código BCD:

Ahora vamos a explicar alguno de los sistemas de numeración que podemos llegar a utilizar o ver a la hora de programar este tipo de circuitos. Los sistemas que vamos a explicar son los siguientes:

- Decimal: Este es el sistema de numeración que conocemos todos, el cual tiene una base de 10 números.
- Binario: Este sistema de numeración es el que utilizan , por ejemplo, los ordenadores y muchos otros aparatos electrónicos. Este sistema tiene una base de 2 números, siendo siempre 1 o 0.
- Hexadecimal: Este sistema de numeración puede ser algo más complicado que los dos anteriores, dado que este sistema tiene una base de 16, en la cual contiene números del 0 al 9 y letras desde la A hasta la F.
- BCD: El código BCD sirve para hacer la conversión entre los sistemas de numeración anteriores, BCD o *"Binary-Coded Decimal"* nos ayudará a pasar un número que tengamos en el sistema decimal al sistema binario.

Sistema decimal:

Este sistema es el que usamos las personas normalmente, es un sistema de numeración con una base de 10, empezando en el 0 hasta llegar al 9. Una vez lleguemos al 9 sumamos 1 al número de la izquierda, cuando el número de la izquierda es 0 no se suele poner.

De forma matemática, el sistema funcionaría de la siguiente manera:

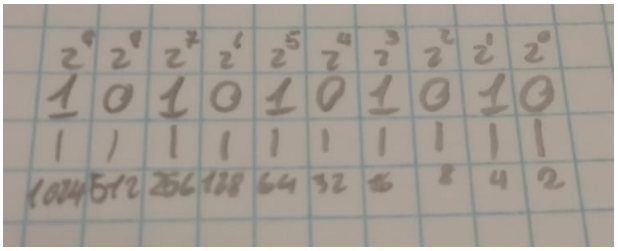
$$
\begin{array}{rcll}
1 & = & 10^0 & \longmapsto \; uno \\
10 & = & 10^1 & \longmapsto \; diez \\
100 & = & 10^2 & \longmapsto \; cien \\
1.000 & = & 10^3 & \longmapsto \; mil \\
10.000 & = & 10^4 & \longmapsto \; diez\ mil \\
100.000 & = & 10^5 & \longmapsto \; cien\ mil \\
1_{,}000.000 & = & 10^6 & \longmapsto \; un\ millón
\end{array}
$$

Multiplicando el número que tengamos por 10 elevado a la cantidad de números que tenga a la derecha.

Sistema binario:

Este sistema es el más usado dentro de los elementos electrónicos, el cual tiene una base de 2, siendo siempre 0 o 1. De esta forma, cada uno de los números tiene un peso de 2 elevado a la cantidad de números que tenga a la derecha, o lo que es igual, cada número tendrá un peso equivalente al doble del número de su derecha.

Para entender mejor este sistema vamos a usar una imagen:



Así podemos ver, que por ejemplo con 10 números en sistema binario podemos llegar al número 1024 en caso de tener solo un 1. Por ejemplo, el número de la imagen será el 1354. A la hora de contar solo tenemos en cuenta el valor de los números que tengan un 1.

Sistema hexadecimal:

Como hemos dicho antes, este código puede ser algo más lioso que los dos anteriores, ya que este sistema contiene número y letras. Este sistema tiene como base el 16, teniendo números desde el 0 al 9 y letras para los restantes, desde la A hasta la F.

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$$

Como podemos apreciar el valor de cada letra sería el siguiente: **A=10, B=11, C=12, D=13, E=14** y **F=15.** Con esto conseguimos que la base sea sobre 16 en vez de sobre 10. El peso de cada número será de 16 elevado a la cantidad de números y letras que tenga a la derecha. En el caso de que el número sea sólo números, se suele poner un “0x” delante.

Código BCD:

Este código es una forma para pasar los números decimales a binarios de una forma sencilla. Este método consiste en separar los números binarios en grupos de 4, con esto conseguimos hacer grupos lo suficientemente grandes como para darle un valor de 9. Cada grupo de 4 dígitos representa uno de los números del número decimal

Ejemplo:

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

Lista de elementos:

Elementos:

- 2 motores de corriente continua (motor CC) acoplados a ruedas neumáticas se encargan de dar tracción.
- La batería nos permitirá alimentar nuestro dispositivo.
- 4 Optoacopladores CNY70.
- Fuente de energía: Batería
- Una Alhambra II
- Tarjeta para el control de los motores: TB6612FNG
- DC/DC: LM2596

FPGA:

La FPGA es un dispositivo programable con bloques de lógica que se programa con un lenguaje especial. Una de las ventajas de la fpga es que superan la potencia de cómputo gracias al paralelismo del hardware, y al controlar las entradas y las salidas ofrece respuestas mucho más rápidas. Con las FPGA puedes controlar robots y printbots.

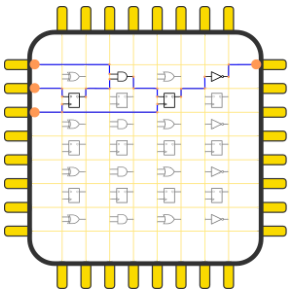
Alhambra II: Es una PCB libre que puedes diseñar circuitos con la herramienta de código abierto que tiene. Tiene los mismos pines que un arduino uno y los shields que tiene son reutilizables la mayoría.

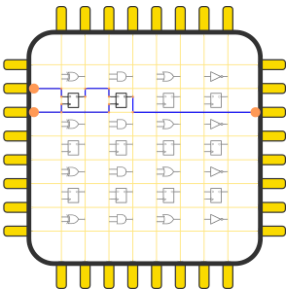
¿Cómo se configura la FPGA? Se descargan bits de configuración contenidos para modificar la funcionalidad de los bloques lógicos y las interconexiones entre ellos. Generalmente los bits se almacenan en una memoria SRAM incluida en cada bloque lógico, que es quien controla la interconexión y funcionalidad de los elementos lógicos de cada bloque lógico. El FPGA no usa un lenguaje de programación sino un lenguaje descriptivo

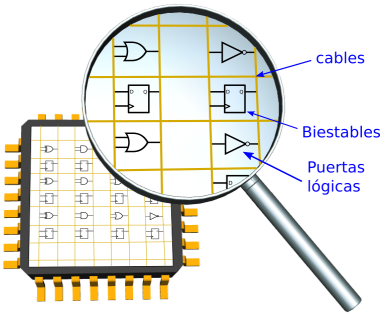
En esta foto se ve cuales son las cuatro formas de programar FPGA y sus características:

Configuration Mode	Preconfiguration Pull-ups	M0	M1	M2	CCLK Direction	Data Width	Serial D_{OUT}
Master Serial mode	No	0	0	0	Out	1	Yes
	Yes	0	0	1			
Slave Parallel mode	Yes	0	1	0	In	8	No
	No	0	1	1			
Boundary-Scan mode	Yes	1	0	0	N/A	1	No
	No	1	0	1			
Slave Serial mode	Yes	1	1	0	In	1	Yes
	No	1	1	1			

Dentro de los fpga están los NO-NOT, O-OR, AND, NOR, NAND y con ellos según los pines y la programación que le pongas al FPGA hará entre ellos un circuito u otro, según el resultado que quieras para tu programa.







Batería:

https://es.aliexpress.com/item/1005001967210715.html

Sensor:

https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=81-CNY70&cPath=1343

Motor:

https://es.aliexpress.com/item/1005001271326638.html?spm=a2g0o.detail.100009.2.44c51911ZaKhn7&gps-id=pcDetailLeftTopSell&scm=1007.13482.95643.0&scm_id=1007.13482.95643.0&scm-url=1007.13482.95643.0&pvid=e37ec38d-7a81-4134-be13-eb6c6927b195&_t=gps-id:pcDetailLeftTopSell,scm-url:1007.13482.95643.0,pvid:e37ec38d-7a81-4134-be13-eb6c6927b195,tpp_buckets:668%230%23131923%230_668%230%23131923%230_668%23888%233325%2314_668%23888%233325%2314_668%232846%238113%231998_668%232717%237563%23539_668%231000022185%231000006605 9%230_668%233480%2315683%23513_668%232846%238113%231998_668%232717%237563%23539_668%233164%239976%23536_668%233480%2315683%23513

Servo:

https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/producto/DetalleProducto.jsp?idIdioma=&idTienda=93&codProducto=999334074&cPath=1343

Alhambra II:

https://www.amazon.es/Alhambra-V-1-0A-Open-FPGA-Board/dp/B07J6TR4WS

Bibliografía:

https://www.areatecnologia.com/electronica/electronica-digital.html
https://www.areatecnologia.com/electronica/karnaugh.html
https://www.neoteo.com/robot-siguelineas-teoria-y-practica/
https://es.linkfang.org/wiki/Sistema_secuencial
https://es.linkfang.org/wiki/Sistema_combinacional