Las señales electricas se transforman en audio o en video.En el caso del formato analógico , la traducción de los datos esta en forma de impulsos eléctricos, en los que la variación reside en la amplitud , pero e el caso de las señales digitales la traducción se hace al sistema binario de ceros y unos donde las difeentes amplitudes representan los 0 y los 1.

La tegnologia digital cambia las señales anallogicas en digitales, permitiendo la transmisión del código binario que luego será reconstruida mediante el software oportuno.

La diferencia entre ambos es la calidad .La tegnologia digital deja mucho que desear

**Tipos de datos**

Arduino permite manejar los siguientes tipos de datos:

* Byte. Almacena un valor numérico de 8 bits. Tienen un rango de 0-255.
* Int. Almacena un valor entero de 16 bits con un rango de 32,767 a -32,768.
* Long. Valor entero almacenado en 32 bits con un rango de 2,147,483,647 a -2,147,483,648.
* Float. Tipo coma flotante almacenado en 32 bits con un rango de 3.4028235E+38 a -3.4028235E+38.
* Arrays Se trata de una colección de valores que pueden ser accedidos con un número de índice (el primer valor del índice es 0). Ejemplos de utilización:
  + Definición y asignación. int myArray[] = {value0, value1, value2...}
  + Definición. int myArray[5]; // declara un array de 6 enteros
  + Asignación del cuarto componente. myArray[3] = 10;
  + Recuperar el cuarto componente y asignarlo a x. x = myArray[3];

**Operadores aritméticos**

* Empleando variables, valores constantes o componentes de un array pueden realizarse operaciones aritméticas y se puede utilizar el operador cast para conversión de tipos. Ej. int a = (int)3.5; Además pueden hacerse las siguientes asignaciones:
* x ++. Lo mismo que x = x + 1.
* x --. Lo mismo que x = x - 1, or decrements x by -1.
* x += y. Lo mismo que x = x + y, or increments x by +y.
* x -= y. Lo mismo que x = x - y .
* x \*= y. Lo mismo que x = x \* y.
* x /= y. Lo mismo que x = x / y.
* Para su utilización en sentencias condicionales u otras funciones Arduino permite utilizar los siguientes operadores de comparación:
* x == y. x es igual a y.
* x != y. x no es igual a y.
* x < y, x > y, x <= y, x >= y.
* Y los siguientes operadores lógicos:
* Y lógico: if (x > 0 &#38;&#38; x < 5). Cierto si las dos expresiones lo son.
* O lógico: if (x > 0 || y > 0). Cierto si alguna expresión lo es.
* NO lógico: if (!x > 0). Cierto si la expresión es falsa.
* El lenguaje de Arduino presenta las siguientes constantes predefinidas:
* TRUE / FALSE.
* HIGH/LOW. Estas constantes definen los niveles de los pines como HIGH o LOW y son empleados cuando se leen o escriben en las entradas o salidas digitales. HIGH se define como el nivel lógico 1 (ON) o 5 V. LOW es el nivel lógico 0, OFF, o 0 V.
* INPUT/OUTPUT. Constantes empleadas con la función pinMode() para definir el tipo de un pin digital usado como entrada INPUT o salida OUTPUT. Ej. pinMode(13, OUTPUT);

## Sentencias condicionales

El lenguaje de arduino permite realizar sentencias condicionales if, if... else, for, while, do... while. Su utilización es similar a las funciones correspondientes en C.

## Entradas y salidas digitales y analógicas

**9.1. Función pinMode(pin, mode)**

Función usada en la function setup() para configurar un pin dado para comportarse como INPUT o OUTPUT. Ej. pinMode(pin, OUTPUT); configura el pin número 'pin' como de salida. Los pines de Arduino funcionan por defecto como entradas, de forma que no necesitan declararse explícitamente como entradas empleando pinMode().

**9.2. Función digitalRead(pin)**

Lee el valor desde un pin digital específico. Devuelve un valor HIGH o LOW. El pin puede ser especificado con una variable o una constante (0-13). Ej. v = digitalRead(Pin);

**9.3. Funcion digitalWrite(pin, value)**

Introduce un nivel alto (HIGH) o bajo (LOW) en el pin digital especificado. De nuevo, el pin puede ser especificado con una variable o una constante 0-13. Ej. digitalWrite(pin, HIGH);

**9.4. Función analogRead(pin)**

Lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023. Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT o OUTPUT.

**9.5. Función analogWrite(pin, value)**

Escribe un valor pseudo-analógico usando modulación por ancho de pulso (PWM) en un pin de salida marcado como PWM. Esta función está activa para los pines 3, 5, 6, 9, 10, 11. Ej analogWrite(pin, v); // escribe 'v' en el 'pin' analógico. Puede especificarse un valor de 0 - 255. Un valor 0 genera 0 V en el pin especificado y 255 genera 5 V. Para valores de 0 a 255, el pin alterna rápidamente entre 0 V y 5 V, cuanto mayor sea el valor, más a menudo el pin se encuentra en HIGH (5 V). Por ejemplo, un valor de 64 será 0 V tres cuartas partes del tiempo y 5 V una cuarta parte. Un valor de 128 será 0 V la mitad del tiempo y 5 V la otra mitad. Un valor de 192 será 0 V una cuarta parte del tiempo y 5 V tres cuartas partes.

Componentes

* Motor paso a paso: convierte los pulsos digitales en pequeños pasos de movimiento en una dirección hacia adelante o hacia atrás.Se mueve solo un paso a la vez,pero puede girar indefinidamente repitiendo la secuencia de pasos.