**Glosario de programación Arduino**

En este apartado encontrarás listadas en orden alfabético las instrucciones de programación utilizadas a lo largo del curso junto con una breve reseña de su utilidad.

analogRead (pin); // lee el voltaje del ‘pin’ de tipo analógico. Lo usamos para leer

valores captados por los sensores y mostrarlos a través del

Monitor Serie.

analogWrite (pin, VALOR); // escribe un pseudo valor analógico de 0 a 255 en pines PWM.

Lo usamos para “encender” o “apagar” componentes,

permitiendo “regular la potencia”.

break; // fuerza el quiebre o salida de una función; los usamos utiliza

dentro de ‘switch’.

Byte // introduce variables de tipo ‘byte’, es decir, que trabajan con

números enteros cortos de 8 bits. Su rango es de 0 a 255.

case // sirve para enumerar los casos posibles dentro de ‘switch’.

Permite definir secuencia de instrucciones, dependiendo del

valor que adopte la variable.

constrain (x, a, b); // fuerza a ‘x’ a permanecer entre los límites de ‘a’ y ‘b’.

default : // establece un caso por defecto dentro de ‘switch’ para que la

programación siga este camino en caso de no coincidir con

las condiciones (case) de ninguno de los casos detallados.

delay (ms); // define un tiempo de demora o retardo. 1000 ms = 1 seg.

digitalRead (pin); // lee el estado del ‘pin’. Lo usamos para leer valores booleanos

captados por los sensores y mostrarlos a través del

Monitor Serie (0 o 1).

digitalWrite (pin, LOW ó HIGH); // escribe un estado ‘LOW’ o ‘HIGH’ en ‘pin’. Lo usamos

para “encender” o “apagar” componentes.

FALSE // valor constante, significa ‘falso’.

Float // introduce variables de tipo ‘flotante’, es decir, que trabajan

con números decimales.

for (inicialización; condición; expresión){...} // bucle que se repite determinada cantidad de

veces.

HIGH // valor constante, significa ‘alto’, equivalente a ‘255’ en

analogWrite.

if (CONDICIÓN){...} // ejecuta las siguientes instrucciones si la condición es ‘TRUE’.

if (CONDICIÓN){...} else {...} // si la condición es ‘TRUE’ ejecuta un set de instrucciones. Sino

(’else’) le indica que haga otro.

#include <...> // sirve para incluir librerías dentro de nuestros programas.

INPUT // valor constante, significa ‘entrada’. Nos permite colocar pines

como “Entradas”

Int // introduce variables de tipo ‘entero’, es decir, que trabajan con

números enteros 16 bits. Su rango es de 32767 a -32768.

Long // introduce variables de tipo ‘largo’, es decir, que trabajan con

números enteros de 32 bits. Su rango es de -2147483648

hasta 2147483647.

LOW // valor constante, significa ‘bajo’, equivalente a ‘0’ en

analogWrite.

map (valor, a1, a2, b1, b2); // a un ‘valor’ dentro de los rangos ‘a1’ y ‘a2’ le asigna su

equivalente dentro del rango comprendido entre ‘b1’ y ‘b2’.

max (x,y); // al comparar entre dos valores, el programa se queda con el de

mayor valor.

min (x,y); // al comparar entre dos valores, el programa se queda con el de

menor valor.

OUTPUT // valor constante, significa ‘salida’. Nos permite colocar pines

como “Salidas”.

pinMode (pin, INPUT o OUTPUT); // configura el ‘pin’ como entrada o salida.

pulseIn (pin, VALOR); // lee un pulso (’HIGH o ‘LOW’) que ingresa por un determinado

pin y entrega como resultado el tiempo en microsegundos

que duró dicho pulso.

Serial.begin (9600); // abre el puerto serie y fija la velocidad de transmisión de datos

Serial.print (dato, tipo de dato); // imprime un valor en el puerto serie.

Serial.println (dato, tipo de dato); // imprime un valor en el puerto serie e incluye un salto

de línea para facilitar la lectura.

Serial.Read (dato, tipo de dato); // lee la información que se encuentra en el puerto serie.

switch (variable) { case... // controla el flujo del programa permitiéndonos especificar

diferentes instrucciones (’case’) que serán ejecutadas en

base a una condición.

TRUE // valor constante, significa ‘verdadero’.

void // sirve para definir una función que no devuelve valores.

void loop // sirve para contener el programa de nuestro robot, que se

reproducirá cíclicamente.

void setup // sirve para recoger la configuración de nuestro programa.

while (CONDICIÓN){...} // bucle que se repite mientras dure la ‘cond