Principales comandos de Arduino en C++

Los siguientes comandos te ayudarán a realizar determinadas funciones en tus programas arduino.

IF

Comprueba si cierta condición se cumple y puede ser usado en conjunto con uno o más operadores de comparación (**==**igual, **!=** distinto, **<** menor, **>** mayor):

```
if (a == b) {
        código
}
```

IF y ELSE

Permite agrupar múltiples comprobaciones.

```
if (a < b) {
        código 1
}
else {
        código 2
}
```

WHILE

Se ejecuta un bloque hasta que la condición deje de cumplirse.

```
while(a > b){
        código
}
```

DO y WHILE

Trabaja de la misma manera que el bucle *while*, con la excepcion de que la condición se comprueba al final del bucle, por lo que este bucle se ejecuta "siempre" al menos una vez.

```
do{
        código
} while (a > b)
```

BREAK

Es usado para salir de los bucles *do*, *for*, o *while*, pasando por alto la condición normal del bucle. Es usado también para salir de una estructura de control *switch*.

```
while (a > b) {
        código
        if(a == 5) {
                break
        }
}
```

FOR

Repite un bloque de código hasta que se cumpla una condición. Se inicializa una variable, se comprueba una condición y ejecuta un bloque, luego se comprueba nuevamente la condición y asi sucesivamente hasta que la condición ya no sea válida.

```
for (int a = 0; a < 10; a++) {
        código
}
```

Este código es equivalente al siguiente

```
int a = 0;
while (a < 10) {
        código
        a++
}
```

SWITCH

Compara el valor de una variable con el valor especificado en las sentencias "case". Cuando se encuentra una sentencia cuyo valor coincide con dicha variable, el código de esa sentencia se ejecuta.

```
switch (variable) {
        case 1:
        código cuando "variable" es igual a 1
        break;
        case 2:
        código cuando "variable" es igual a 2
        break;
        default:
        //código ejecutado cuando ninguna de las sentencias se cumple (es opcional)
}
```

pinMode:

Configura el pin especificado para comportarse como una entrada o una salida.

pinMode(pin, OUTPUT/INPUT)

digitalWrite:

Pone en 1 o 0 un pin de salida (output)

digitalWrite(pin, HIGH/LOW)

digitalRead:

Lee el valor de un pin configurado como entrada (input).

digitalRead(pin)

analogRead:

Lee el valor de tensión en el pin analógico especificado. Se representa con un numero entero entre 0 y 1023.

analogRead(pin)

analogReference:
Configura el voltaje de referencia usado por la entrada analógica. La función *analogRead()* devolverá un valor de 1023 para aquella tensión de entrada que sea igual a la tensión de referencia. Las opciones son:

DEFAULT: Es el valor de referencia analógico que viene por defecto, generalmente 5v o 3,3v.
INTERNAL: Es una referencia de tensión interna que puede ser de 1.1v o 2,56v, dependiendo las versiones.
EXTERNAL: Se usará una tensión de referencia externa que tendrá que ser conectada al pin *AREF*.

analogReference(DEFAULT/INTERNAL/EXTERNAL)