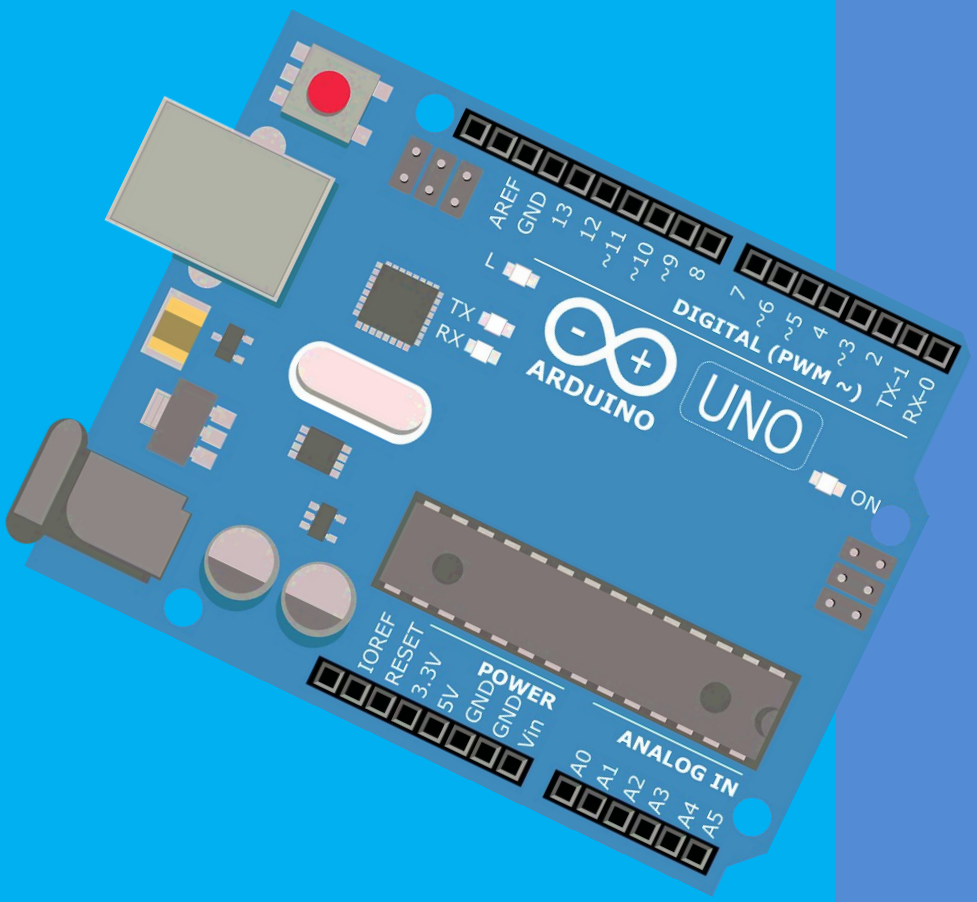
ARDUINO NIVEL 1

Primeras líneas de código a aprender

Instructor: Konrad Peschka



Primeras líneas de código a aprender

Un programa es básicamente el equivalente a una receta de cocina… pero destinado a un público distinto.

Mientras que las personas somos razonablemente buenas interpretando las instrucciones, generalmente vagas, de una receta de cocina, cuando programamos quien debe entendernos es un ordenador que espera instrucciones precisas respecto a lo que debe hacer y que además carece por completo de la imaginación o capacidad de improvisación humana.

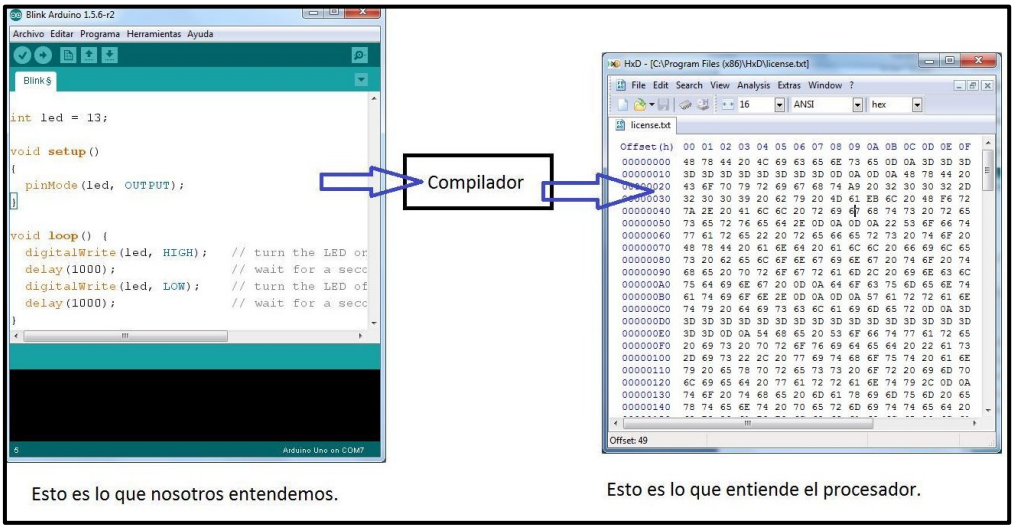
Por ello se desarrollan los lenguajes de ordenador, para dar instrucciones a una máquina de forma:

- Precisa: Sin ambigüedades inherentes a la comunicación humana.
- Unívoca: Solo se puede interpretar de una manera.
- Concisa: Preferiblemente órdenes cortas.

Arduino se programa en una variante de C++ , que es un lenguaje muy extendido por sus características. Un programa es una serie de instrucciones que se ejecutan en secuencia (salvo que indiquemos expresamente condiciones precisas en las que esta secuencia se altera).

Un programa interno comprueba que la sintaxis de nuestro programa es acorde a la norma, y si hay cualquier cosa que no le convence dará un error y finalizará la comprobación obligándonos a revisar lo que hemos escrito.

Cuando el comprobador acepta nuestro programa, invoca otro programa que traduce lo que hemos escrito a instrucciones comprensibles para el procesador de nuestro Arduino. A este nuevo programa se le llama compilador.



El compilador convierte nuestras instrucciones (código fuente) en instrucciones del procesador (código ejecutable).

Estructura de un programa Arduino

Un programa de Arduino consiste en dos secciones o funciones básicas:

- **Setup**: Sus instrucciones se ejecutan solo una vez, cuando se arranca el programa al encender Arduino o cuando pulsamos el botón de reset. Generalmente incluye definiciones e inicializaciones de ahí su nombre.
- **Loop:** Sus instrucciones se van ejecutando en secuencia hasta el final…. Y cuando acaba, vuelve a empezar desde el principio haciendo un ciclo sin fin.

Cuando abrimos la interfaz de programación de Arduino, él nos escribe ya estas dos funciones

Nótese que el principio de cada función es indicado por la apertura de llave **“ { “** y el fin de la misma corresponde al símbolo de cerrar llaves **“ } “.**

De hecho, el conjunto de instrucciones contenidas entre una apertura y cierre de llaves se llama bloque y es de capital importancia a la hora de que nuestro Arduino interprete de una u otra manera las instrucciones que le damos. Es imperativo que a cada apertura de una llave corresponda un cierre de llave.

Cualquier cosa que escribamos precedido por **“// “** son comentarios, y serán ignorados. Es decir, podemos dejarnos mensajes dentro del código, (que de otro modo darían errores). El compilador ignorará cualquier cosa entre // y el fin de línea

// A partir de aquí todo será ignorado porque esto es un comentario.

Nuestro primer programa en Arduino

Basándonos en las estructuras del programa vistas, y en instrucciones básicas, haremos un programa que le indique a Arduino que active su pin 10 como de salida digital y después encenderemos y apagaremos esta señal lo que hará que el LED que tiene conectado en el puerto 10 se encienda o apague al ritmo que marquemos.

Para indicar al sistema que deseamos usar el pin 10 como salida digital utilizamos la instrucción:

- **pinMode (10, OUTPUT) ;**

La función de Arduino pinMode permite configurar a cada pin, de forma individual, como entrada o como salida

- 10. Es el número del pin que se quiere configurar, este número está escrito en la tarjeta Arduino que se usará.
- INPUT. Configura el pin como entrada.
- INPUT_PULLUP. Se le agrega al pin una resistencia de pull-up interna y se configura como entrada.
- OUTPUT. El pin se configura como salida digital.

Estas definiciones se harán solo una vez al principio, en la función setup(). La nuestra quedará, con una única instrucción que declara que vamos a usar el pin 10 como salida digital:

```
void setup()
      {
         // Inicializa el pin 10 como salida
         pinMode( 10, OUTPUT) ;
      }
```

- Es importante fijarse en que, a pesar de ser una única instrucción, hemos delimitado el bloque de esta función mediante abrir y cerrar llaves.
- Obsérvese que la instrucción finaliza en **";"** . la sintax obliga a acabar las instrucciones con un punto y coma que delimite la orden. Si se omite generará un error.

Para encender el LED usaremos la instrucción:

- **digitalWrite(10 , HIGH) ;**

La función Arduino digitalWrite permite escribir valores lógicos digitales en un pin de Salida de una tarjeta Arduino. Está función requiere que el pin haya sido declarado como salida previamente. Requiere de dos parámetros de entrada. El primero es el número de pin (0-13) y el segundo es la condición lógica (HIGH ó LOW).

Para apagar el LED usaremos la instrucción:

- **digitalWrite(10 , LOW) ;**

El 10 indica el pin a utilizar y HIGH, LOW indican el valor que deseamos poner en esa salida, que en Arduino corresponden a 5V para HIGH y 0V para LOW.

- Si en la función loop() escribiéramos estas dos instrucciones seguidas, Arduino cambiaría estos valores tan deprisa que no percibiríamos cambios, así que necesitamos frenarle un poco para que podamos percibir el cambio.

Para hacer este retraso de, digamos, un segundo, utilizaremos:

- **delay(1000) ;** // delay “congela” Arduino n milisegundos

En Arduino “delay” es una función que hace que el procesador espere. Por ejemplo, esta espera permite no hacer nada y esperar hasta la ejecución de la siguiente instrucción durante un retardo de tiempo definido. Entonces esta función tiene un parámetro de entrada del tipo entero, que es la espera en milisegundos.

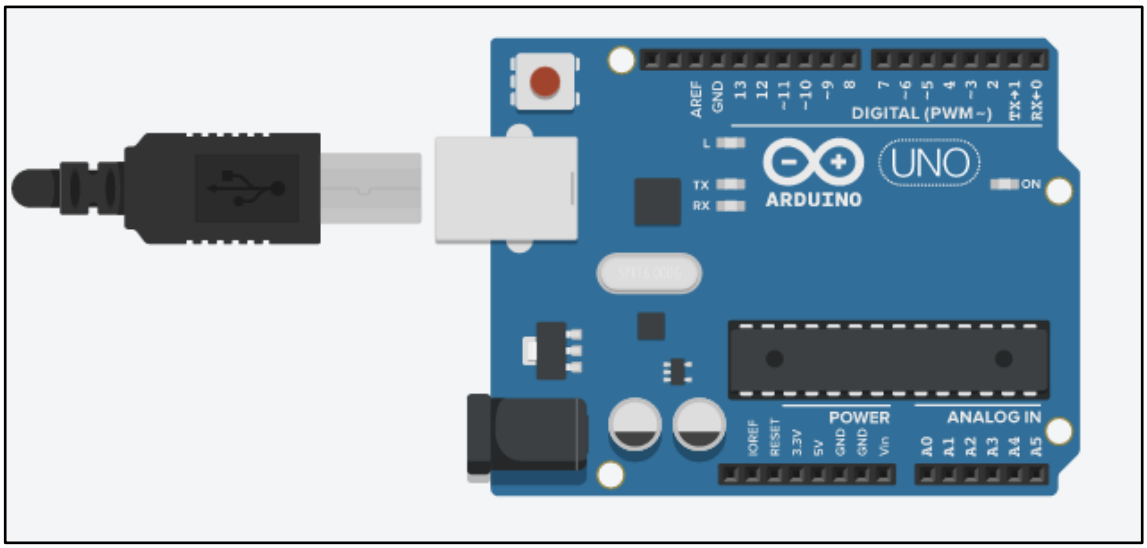
Por tanto, para programar una luz que se enciende y se apaga, tendríamos que generar una secuencia de órdenes (Como en una receta de cocina) que hicieran:

1. Informar a Arduino de que vamos a utilizar el pin10 para escribir valores (en el Setup)
2. Encender el LED (pin10): Poner valor alto (5V) en dicho pin.
3. Esperar un segundo.
4. Apagar el LED (pin10): Poner valor bajo (0V) en dicho pin.
5. Volver a esperar un segundo.

Circuito Propuesto

El circuito propuesto consta de un Arduino que hará titilar el led interno de la placa dispuesto en el pin13, para ello usaremos:

- 1 Arduino uno



Circuito: Titilar led

Solución de la Programación

Programa: Titilar led	

void setup()
{
  pinMode(13, OUTPUT);
}
	// Función Setup
	// llave de apertura del bloque Setup
	//Asignación de puertos como salida
	//llave de cierre del bloque Setup

void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
	//Función Loop
	// llave de apertura del bloque Loop
	//Puerto 13 en estado alto
	// retardo de 1000 millisecond(s)
	//Puerto 13 en estado bajo
	// retardo1000 millisecond(s)