PRÁCTICA 2. Encender un LED de forma intermitente

Materiales. 1 led y 1 resistencia de 220 Ω (rojo-rojo-marrón).

Explicación. Debemos conseguir que un led se encienda de forma intermitente de la siguiente manera: en cada ciclo, el encendido durará medio segundo y el apagado un segundo. Para ello, conectaremos el ánodo del led al pin 13. El cátodo del led irá a una resistencia de 220 Ω para proteger el led. La resistencia irá a tierra. Hecho esto, ya sabemos que cuando el pin 13, que es digital, esté en alto (5 V), el led se encenderá y cuando el pin 13 esté en bajo (0 V), el led se apagará.

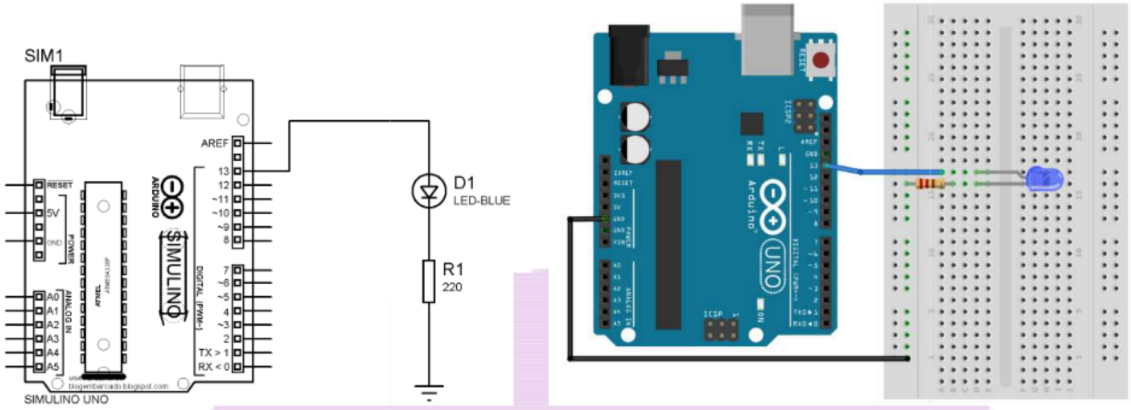
Esquema de un sketch. Dividimos cada sketch en tres partes:

1. Declaración de variables y constantes globales.
   Por ejemplo, si escribimos `const int led=13;` estamos declarando una constante entera llamada `led` de valor 13. Por ser una constante no podremos modificar su valor durante el sketch.
   Por ejemplo, si escribimos `int pepito=12;` estamos declarando una variable entera llamada `pepito` a la que inicialmente le hemos asignado el valor de 12. No es obligatorio asignar un valor inicial a una variable.

2. `void setup() {... }`
   Esta parte del sketch va entre llaves y se ejecuta una única vez. Aquí es donde decidiremos qué pines de Arduino vamos a utilizar y si van a ser de entrada o salida. También podemos inicializar los que creamos conveniente.
   Los pines digitales pueden ser de entrada o de salida y son: 0, 1, 2, …, 12 y 13. Los pines digitales solo pueden en alto o HIGH (5 V) y en bajo o LOW (0 V). Los pines analógicos siempre son de entrada y son: A0, A1, …, A4 y A5. Los pines analógicos pueden tomar cualquier valor entre 0 V y 5 V.
   Si escribimos `pinMode(12,OUTPUT);` estamos declarando el pin digital 12 como salida. Por tanto, Arduino va a poder poner este pin en HIGH o en LOW según se lo indiquemos en el sketch.
   Si escribimos `digitalWrite(12,HIGH);` Arduino pondrá el pin de salida 12 en alto.
   Si escribimos `digitalWrite(12,LOW);` Arduino pondrá el pin de salida 12 en bajo.
   Si escribimos `pinMode(9,INPUT);` estamos declarando el pin digital 9 como entrada. Por tanto, Arduino va a poder leer si este pin está en alto o en bajo cuando se lo indiquemos. De momento, no vamos a necesitar leer pines; ni digitales ni analógicos.

3. `void loop() {... }`
   Esta parte del sketch va entre llaves y se ejecuta constantemente; esto es, después de ejecutarse la última línea se volverá a ejecutar la primera y se repetirá el bucle.
   Si escribimos `delay(500);` la ejecución del programa se detendrá durante 500 ms; esto es, medio segundo. Así, la función delay detiene la ejecución del programa el tiempo que le indiquemos entre paréntesis en milisegundos.
   Si en una línea escribimos `//` , entonces a continuación, en esa línea, podemos poner un comentario.

Esquemas eléctricos



Sketch

```
const int led=13; //conectaré el ánodo del LED al pin 13

void setup() {
pinMode(led,OUTPUT); //declaro el pin al que conectaré el LED como de salida
}

void loop() {
  digitalWrite(led,HIGH); //pongo el pin al que conectaré en LED en alto ...
  delay(500);             //durante medio segundo
  digitalWrite(led,LOW);  //pongo el pin al que conectaré el LED en bajo...
  delay(1000);            //durante un segundo
}
```

Ampliación. ¿Cómo harías para que el encendido fuese de 1 s y el apagado de 0,3 s?