

PRÁCTICAS DE ARDUINO

Práctica nº 1: Encendido y apagado LED

En esta práctica veremos **como encender un LED mediante las salidas de Arduino**. Para ello, veremos el principio de funcionamiento y el esquema eléctrico necesario.

En algunos libros o páginas web veréis que, a veces, conectan directamente el LED a una salida digital o analógica de Arduino. Aunque esto funciona (luego veremos porque) es una mala práctica. En general, **los LED deben conectarse siempre a través de una resistencia**.

Para entender la importancia y el papel de esta resistencia, y poder calcular su valor, es necesario entender **cómo funciona un LED**.

¿QUÉ ES UN LED?

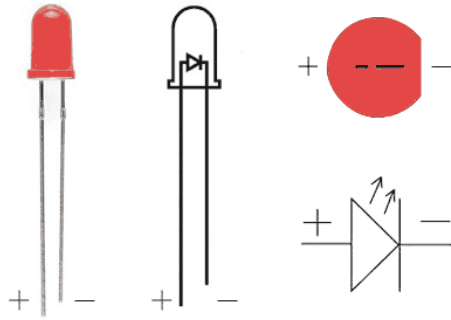
Un LED es un diodo emisor de luz, es decir, **un tipo particular de diodo que emite luz** al ser atravesado por una corriente eléctrica. Los diodos (emisor de luz, o no) son unos de los dispositivos electrónicos fundamentales.

Recordemos que **diferenciamos entre dispositivos eléctricos y electrónicos**. Los dispositivos eléctricos engloban resistencias, condensadores y bobinas, e integran el campo de electricidad. Los dispositivos electrónicos, surgen del uso de materiales semiconductores, y dan lugar al campo de la electrónica.

Un diodo es una **unión de dos materiales semiconductores con dopados distintos**. Sin entrar en detalles, esta diferencia de dopado hace que genere una barrera de potencial, que como primera consecuencia hace que el paso de corriente en uno de los sentidos no sea posible.

Aquí tenemos la primera característica de los diodos, tienen polaridad, es decir, **solo dejan pasar la corriente en un sentido**. Por tanto, tenemos que conectar correctamente la tensión al dispositivo.

La patilla larga debe ser conectada al voltaje positivo (ánodo), y la corta al voltaje negativo (cátodo).



Regla mnemotécnica:

- La patilla “más” larga es la positiva.
- La patilla “menos” larga es la negativa.

La otra consecuencia de la barrera de potencial es que, incluso conectando el dispositivo con la polaridad correcta, a baja tensión los electrones siguen sin poder atravesar el dispositivo. Esto ocurre hasta alcanzar **un cierto valor de tensión que llamamos tensión de polarización directa (V_d)**, que depende del tipo de diodo.

A partir de esta tensión decimos que **el diodo está polarizado y la corriente puede atravesarlo libremente** con una resistencia casi nula. La tensión que realmente está alimentando al diodo es la diferencia entre la tensión aplicada, y la tensión de polarización directa del diodo.

Como vemos, en el momento que superamos la tensión de polarización, y dado que la resistencia del diodo es muy pequeña, se genera una gran corriente que destruirá el diodo. Por ese motivo, **necesitamos una resistencia que limite la cantidad de corriente que circula por el diodo.**

En resumen, si no ponemos una resistencia, el sistema solo tiene dos estados.

- Si alimentamos a una tensión inferior a V_d , el LED no luce.
- Si alimentamos a una tensión superior a V_d , el LED se rompe.

En cualquier caso, **no conseguiremos hacer lucir el LED sin utilizar una resistencia** de valor adecuado.

¿POR QUÉ FUNCIONA CONECTANDO A UNA SALIDA DE ARDUINO DIRECTAMENTE?

Como hemos adelantado esto, a veces veréis en tutoriales en Internet que algunos conectan un LED directamente a una salida de Arduino, sin usar una resistencia. Efectivamente, esto funciona y el LED luce sin romperse. ¿Cómo puede ser posible esto?

Esto funciona porque **Arduino tiene una limitación de 20mA** en sus salidas. Esta limitación hace que el LED no se funda, aunque realmente se está comportando como si fuera un cortocircuito. Simplemente Arduino no puede dar más corriente.

No obstante es esto es una práctica totalmente desaconsejada por varios motivos. En primer lugar porque supone forzar de forma innecesaria la salida de Arduino, lo que puede acortar su vida a largo plazo. Por otro porque 20mA es, en general, una corriente demasiado elevada para un Led.

¿QUÉ TIPOS DE LED USAREMOS EN ELECTRÓNICA?

Existe una gran gama de LED disponibles, desde los LED habituales de pequeña potencia, a los LED de gran potencia empleados en iluminación. Estos últimos requieren etapas adicionales de potencia (drivers) para poderlos encender desde un autómata.

Dentro de los LED de pequeña potencia, que son los que vamos a emplear con más frecuencia, los más habituales **son los encapsulados tradicionales de LED de 3mm o 5mm.**



También podemos encontrar **LED opacos (diffused) o LED transparentes (clear).** Los LED opacos están pensados para “encenderse” ellos mismos (por ejemplo, para hacer un panel de mando). Por el contrario, los LED transparentes están pensados para iluminar un área, pero no al propio LED.



Adicionalmente, **encontraremos LED con diferentes ángulos.** Los LED con un ángulo de iluminación más pequeño tienen un haz más cerrado, por lo que concentran la luz en un área estrecha. Por el contrario, los LED con ángulos más amplios concentran una cantidad de luz inferior hacia delante, y a cambio iluminan un área mayor.

Por último, veréis que **algunos LED tienen el encapsulado de un color**. Este color es simplemente para identificar el color de la luz emitida por el LED sin tener que encenderlo, pero **no tiene ninguna influencia en el color de la luz emitida**, que sólo depende de la construcción interna del LED (personalmente, yo los prefiero con el encapsulado sin colorear).



CALCULAR EL VALOR DE LA RESISTENCIA

Hemos dicho que lo principal para hacer funcionar un LED **es calcular el valor de la resistencia necesaria**. Para calcular el valor de tensión necesaria para alimentar un LED necesitamos conectar 3 parámetros:

- La tensión de alimentación (V_{cc})
- La tensión de polarización directa del LED (V_d)
- La corriente nominal del LED (I_n)

Calcular el valor de la resistencia es sencillo. Como hemos dicho, la tensión que soporta el LED es la diferencia entre la tensión aplicada y la tensión de polarización directa del LED.

Aplicando la ley de Ohm, con el valor de la intensidad nominal del LED

$$V = V_{cc} - V_d = I_{nominal} * R$$

Por lo que el valor de la resistencia resulta

$$R = \frac{V_{cc} - V_d}{I_{nominal}}$$

Dado que las resistencias comerciales tienen valores normalizados, no encontraréis una resistencia con el valor exacto que hayáis calculado. En este caso, **elegiremos la resistencia normalizada inmediatamente superior al valor calculado**, para garantizar que la corriente es inferior a la nominal.

La tensión de alimentación V_{cc} es conocida para nosotros. En caso de aplicar una fuente de alimentación o una batería, V_{cc} es la tensión nominal de la misma. En el caso de una salida digital o analógica de Arduino, V_{cc} dependerá del modelo que estemos usando (5V o 3.3V) pero también es conocido.

Recordar que **aunque uséis una salida analógica PWM la tensión entregada a la carga es siempre Vcc.**

Respecto a la tensión de polarización y la corriente nominal **dependen de los materiales y constitución interna del diodo.** En el caso de diodos LED convencionales de 3mm y 5mm, **dependen principalmente del color y luminosidad.**

No obstante, en la mayoría de las ocasiones **el propio vendedor facilita estos valores** en el anuncio. En caso de duda deberemos acudir al Datasheet del LED para consultar los valores nominales.

En la siguiente tabla os adjuntamos unos valores generales de la tensión de polarización V_d típica para cada color. También os figura el valor de la resistencia necesaria, en Ohmios, para distintos valores de tensión de alimentación V_{cc} .

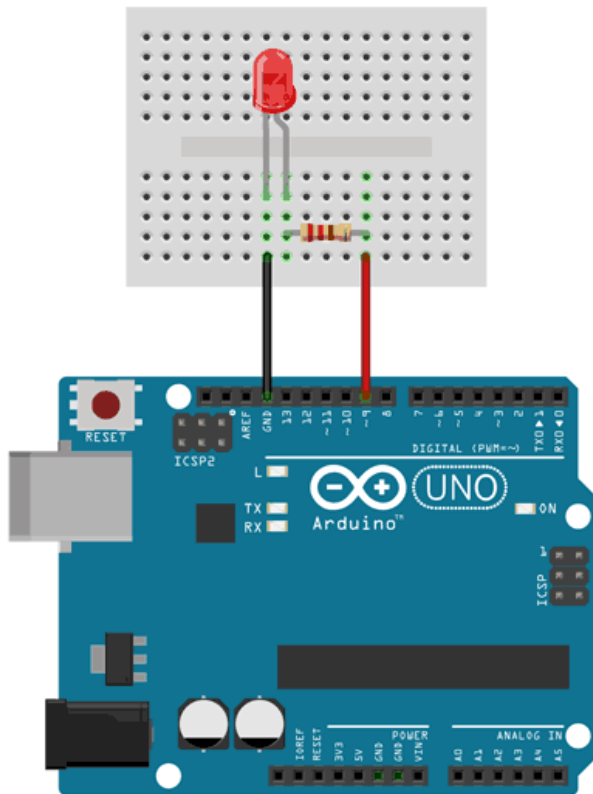
Color	Vdd	Resistencia (Ohmios)			
		3.3V	5V	9V	12V
Infrarrojo	1.4V	150	270	510	680
Rojo	1.8V	100	220	470	680
Naranja	2.1V	100	200	470	680
Amarillo	2.2V	100	200	470	680
Verde	3.2V	10	150	330	560
Azul	3.5V	-	100	330	560
Violeta	3.6V	-	100	330	560
Blanco	3.8V	-	100	330	560

MATERIALES

- Arduino UNO.
- Cable USB tipo A-B.
- 1 LED
- 3mm
- 1 Resistencia de 220Ω.
- Protoboard.
- Cables de conexión.

MONTAJE

La conexión eléctrica es realmente sencilla. Simplemente ponemos la resistencia en serie con el LED. El montaje en una protoboard quedaría de la siguiente forma.



PROGRAMACIÓN

Podemos indicar a Arduino que en un pin determinado coloque un “0” o un “1” lógico (que serán 0 V o 5 V) mediante los siguientes comandos, respectivamente:

digitalWrite(pin,LOW);

digitalWrite(pin,HIGH);

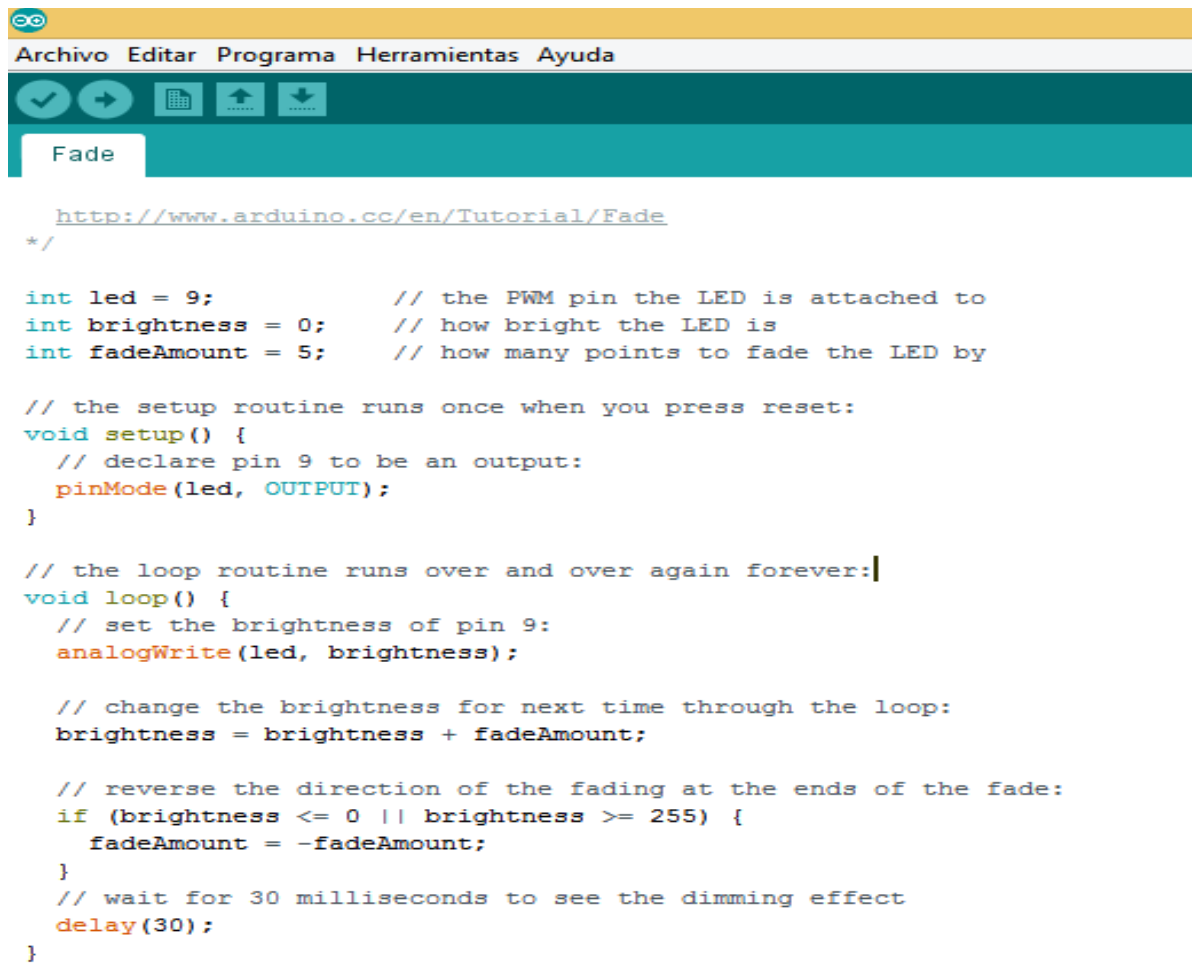
La función delay que hace que el procesador espere. Por ejemplo, esta espera permite no hacer nada y esperar hasta la ejecución de la siguiente instrucción durante un retardo de tiempo definido. Entonces esta función tiene un parámetro de entrada del tipo entero, que es la espera en milisegundos.

delay (retardoMilisegundos);

EJEMPLOS DE CÓDIGO

A continuación se presentan alguno de los códigos para probar a encender LED con nuestro Arduino y el montaje indicado.

Una vez realizado el montaje para comprobar su correcto funcionamiento, cargamos el siguiente código de ejemplo: **Archivo >Ejemplos >Basics > Blink.**



```
http://www.arduino.cc/en/Tutorial/Fade
*/

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Este código es similar a los que hemos visto previamente en ejemplo Blink, pero empleando un LED externo en lugar del LED integrado en la placa. Para ello, solo tenemos que sustituir el número de PIN 13, correspondiente con el LED integrado, por el PIN de la salida a la que vamos a emplear en nuestro caso el 9.

Una vez comprobado el programa de ejemplo, vamos a comenzar a crear nuestro primer código que sirve para encender y apagar un LED:



```
const int ledPIN = 9;

void setup() {
  Serial.begin(9600);    //Iniciar puerto serie
  pinMode(ledPIN , OUTPUT); //definir pin como salida
}

void loop(){
  digitalWrite(ledPIN , HIGH);    // Poner el Pin en HIGH
  delay(1000);                    // Esperar un segundo
  digitalWrite(ledPIN , LOW);     // Poner el Pin en LOW
  delay(1000);                    // Esperar un segundo
}
```

Ejercicios propuestos

1. Comprueba que el programa “Fade” funciona correctamente en tu placa Arduino.
2. Cambia los valores de la función delay en el para que aumente o disminuya el tiempo para ver el efecto de atenuación del led.
3. Comprueba que el programa “Encender y apagar” funciona correctamente en tu placa Arduino.
4. Cambia los valores de la función delay para que pasen 5 segundos (5000 ms) entre el encendido y apagado del led.
5. Cambia los valores de la función delay para que espere 1 minuto entre el encendido y apagado del led.