CAPÍTULO 5:

# LED RGB

# Led RGB

Ya encendimos nuestras primeras luces, diodos leds simples. Ahora veremos como encender un led que puede combinar 3 diodos de colores diferentes, a diferentes intensidades, generando una infinita gama de tomos bajo el concepto de síntesis aditiva.

Veremos nuevas funciones y trabajaremos con PWM, no dejes de prestar suma atención a ese tema, ya que va a servir para muchos de los proyectos a lo largo de nuestro curso Bots.

## Consejo:

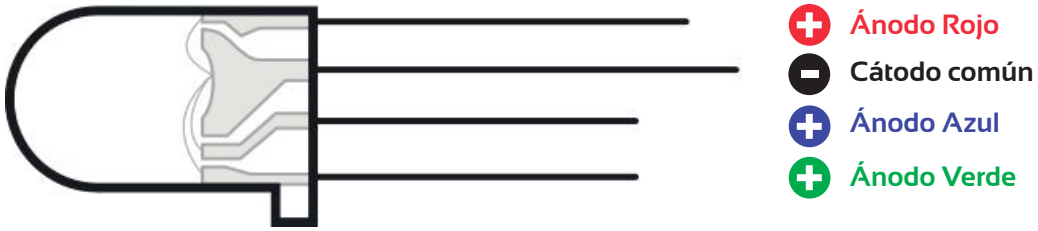
Verificá que tipo de led RGB tenés en la mano: pueden ser de ánodo o cátado común. La pata larga del led se conecta al positivo o al negativo, podés hacer la prueba con la pila del control remoto.

# LEDs RGB

**Es un LED con tres diodos en su interior: uno rojo, uno verde y otro azul.** De ahí su nombre RGB, el cual proviene de las siglas inglesas para Red (rojo), Green (verde) y Blue (azul).

**Estos LEDs poseen cuatro patas y pueden ser de dos tipos**: con un ánodo común o con un cátodo común, lo cual determinará la circulación de la energía en su circuito. **En los proyectos de BOTS trabajamos con LEDs RGB de cátodo común.**

Un LED RGB de cátodo común significa que su terminal más larga es un cátodo (polo negativo) mientras que las otras tres -todas de diferente largo, pero más cortas que el cátodo- son los ánodos que representan cada color, tal y como podemos ver a continuación.




La principal ventaja de los LED RGB es que pueden reproducir casi cualquier color de una manera perfecta, pudiéndose utilizar para reproducir imágenes y videos, o para iluminar una sala con un color determinado. Jugando con la cantidad aditiva de cada unos de los tres colores básicos se pueden obtener infinidad de colores y tonos diferentes.

## Síntesis aditiva del color

**Cuando nos referimos a la síntesis aditiva hablamos de la formación de los colores a través de la suma de diferentes luces en sus distintas longitudes de onda.** Este modelo de colores considera el blanco como la suma de toda luz en máxima proporción del espectro visible, y es el que se usa para la separación del color. Gracias a la síntesis aditiva somos capaces de ver y reproducir los colores de las diferentes pantallas.

Hay cinco premisas fundamentales que debes tener en cuenta:

1. Los colores primarios aditivos son: Rojo, Verde y Azul (RGB).
2. La suma de dos colores primarios a partes iguales origina un color secundario:

   Rojo + Verde = Amarillo
   Rojo + Azul = Violeta (Magenta)
   Azul + Verde = Celeste (Cyan)

3. El blanco teórico se forma por la unión de los tres colores primarios a partes iguales con la máxima saturación posible (255). Esta suma de todas las luces en partes iguales da por resultado la luz blanca (la cual contiene a todos los colores aditivos).

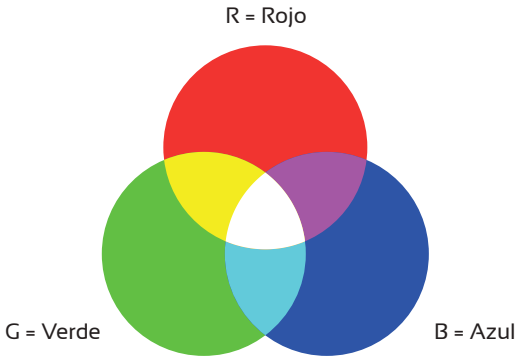
   Azul + Verde + Rojo = Blanco teórico

4. La ausencia de colores primarios de síntesis aditiva origina el negro. El negro es la ausencia de luz, sin luz el ojo no percibe color.

5. El color complementario o inverso de cada primario aditivo se puede definir como el color que le falta a ese primario para ser blanco:

   Rojo Complementario: Cian
   Verde Complementario: Magenta
   Azul Complementario: Amarillo

**Nota:** para que puedas crear tus propios colores puede ser útil utilizar un *mezclador digital*. Estos abundan en la web y te ayudarán a saber qué luces debes usar y en qué proporción para obtener el color que deseas de manera rápida. Para que pruebes, te recomendamos usar esta ***Rueda de Colores RGB***.




## El uso de las resistencias con LEDs

Para limitar la corriente que fluye a un LED, se debe usar una resistencia en una conexión en serie. Esto significa que debes conectar el ánodo del LED a uno de los terminales de la resistencia y debes conectar el otro terminal de la resistencia al positivo de la fuente de alimentación. Luego debes conectar el terminal del cátodo del LED al negativo de la fuente de alimentación. Ya que los dos terminales del LED no están directamente conectados a la alimentación, la electricidad debe fluir a través de la resistencia, lo que limita la corriente.

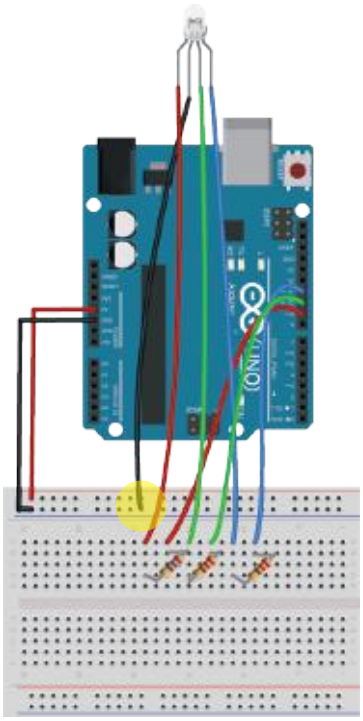
En el caso del LED RGB es necesario utilizar tres 3 resistencias, una por cada ánodo de color. Para saber qué resistencia corresponde en cada proyecto te recomendamos que leas los artículos sobre "Ley de Ohm" y "Cálculo de resistencias".

## Diferencias entre LED RGB de cátodo común y ánodo común

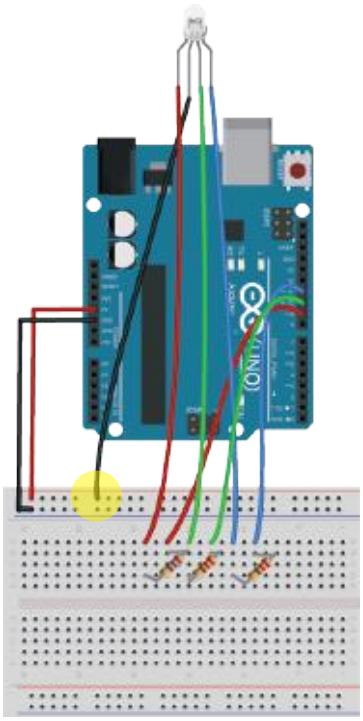
Hasta aquí hemos visto que la principal diferencia entre estos LEDs es la disposición de sus terminales. El LED de cátodo común tiene una terminar negativa (el cátodo) y tres positivas (un ánodo para cada color); mientras que los LEDs de ánodo común poseen una terminal positiva (el ánodo) y tres terminales negativas para cada color.

A continuación vamos a desarrollar el esquema del circuito electrónico y el código de programación de cada uno de ellos.

CÁTODO COMÚN



ÁNODO COMÚN

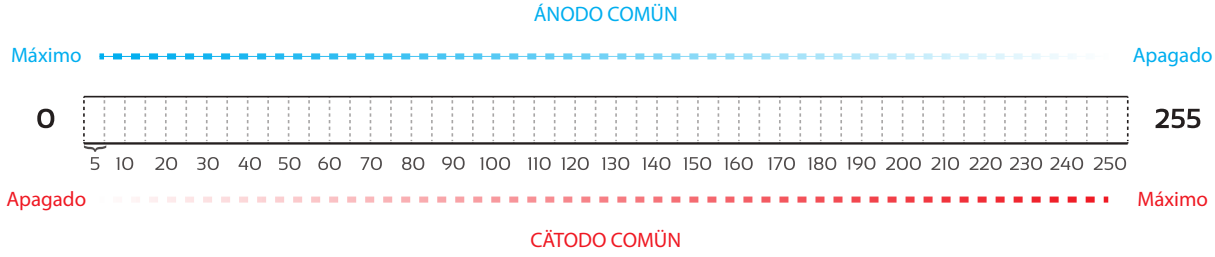


A simple vista, la conformación del circuito eléctrico es la misma, sin embargo, si prestamos atención, la terminal más larga del LED RGB de cátodo común se encuentra conectada al GND (-), mientras que en el otro lo conectamos a 5v (+).

```
void loop() {

 analogWrite(LED_ROJO, 255); // Color rojo ON
 delay(1000);
 analogWrite(LED_ROJO, 0); // Color rojo OFF
 analogWrite(LED_VERDE, 255); // Color verde ON
 delay(1000);
 analogWrite(LED_VERDE, 0); //Color verde OFF
 analogWrite(LED_AZUL, 255); // Color azul ON
 delay(1000);
 analogWrite(LED_AZUL, 0); // Color azul OFF

}
```

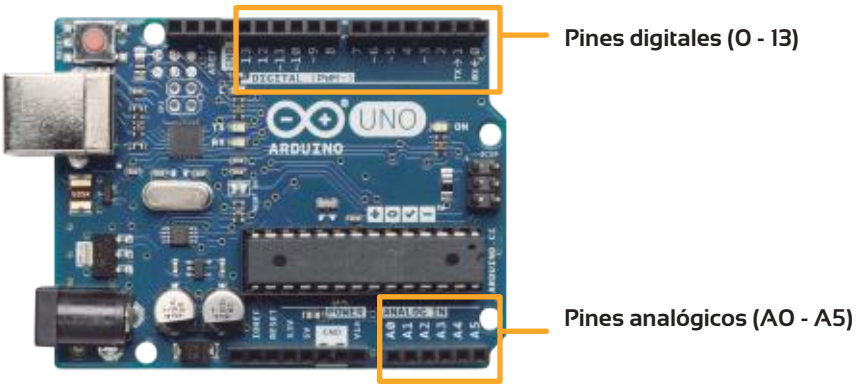
```
void loop() {

 analogWrite(LED_ROJO, 0); // Color rojo ON
 delay(1000);
 analogWrite(LED_ROJO, 255); // Color rojo OFF
 analogWrite(LED_VERDE, 0); // Color verde ON
 delay(1000);
 analogWrite(LED_VERDE, 255); //Color verde OFF
 analogWrite(LED_AZUL, 0); // Color azul ON
 delay(1000);
 analogWrite(LED_AZUL, 255); // Color azul OFF

}
```

Los códigos de programación parecen iguales, sin embargo observamos que para el encendido y apagado de cada uno de los diodos de color se invierten los valores del parámetro en la función analogWrite. En la programación del LED RGB de ánodo común, al conectar los diodos a los pines PWM, y darles el valor 0 (cero), estos se comportan como GND (-) y generan la diferencia de voltaje necesarios para completar el circuito.



# Otras funciones asociadas a los pin

Los pin, además de poder configurarse como entradas o salidas, pueden tener un funcionamiento analógico o digital según dónde los conectemos en la placa de Arduino.




# Funciones analógicas

A diferencia de las funciones digitales, donde ambas trabajan con los pins del 0 al 13, aquí existen dos alternativas.

• **analogRead** (significa "lectura analógica"): esta función sirve para realizar lecturas en los **pines analógicos**, cuyos valores sólo van del **A0 al A5** (ubicados a la izquierda de la placa arduino en la imagen de la página anterior). Los pines analógicos, a diferencia de los digitales, **por defecto son** INPUT por lo que no es necesario usar el pinMode si vamos a usarlos como entrada.

La función analogRead sirve para leer el voltaje presente en un determinado pin analógico con una resolución de 10 bits, lo que significa que el valor de lectura final podrá oscilar entre 0 y 1023.

Su sintaxis correcta es: **analogRead(pin);**

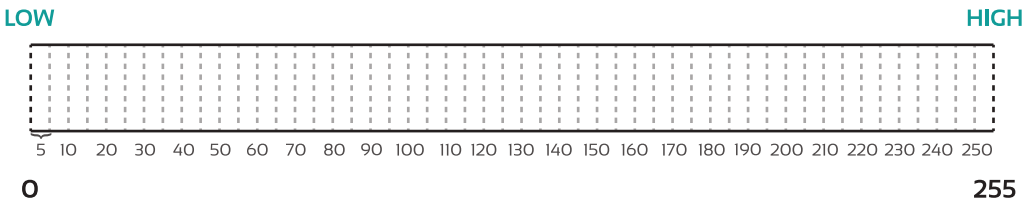
Ejemplo:
valor = analogRead(2); // hace que 'valor' sea igual al estado leído en el pin A2

• **analogWrite** (significa "escritura analógica"): esta instrucción sirve para **grabar un pseudo-valor analógico** a través de un proceso de modulación por ancho de pulso (PWM) **a alguno de los pin de Arduino marcado como PWM**; es decir que la función analogWrite sólo estará disponible para los pines **3, 5, 6, 9, 10 y 11 (señalados con un "~" por delante)**, que son los capaces de entregar una salida analógica.

Su sintaxis correcta es: **analogWrite(pin, valor);**

El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre con un margen entre 0 y 255. Esto es útil , por ejemplo, cuando queremos trabajar con valores intermedios para manejar las luces diodo LED y no sólo usando HIGH (equivalente a 255 en esta escala) o LOW (0), los únicos dos valores posibles con digitalWrite.

En este gráfico vemos la relación entre HIGH/LOW y la cantidad de valores intermedios que ofrece el analogWrite a través de los pines PWM.




Ejemplo:
```
analogWrite(11, 182); //escribe el valor "182" en el pin 11
```

Para que te quede más claro el uso de las funciones analógicas con pines PWM, veamos un ejemplo donde se trabaja con luces RGB. Este programa lo que hace es definir los pines 9, 10 y 11 (capaces de recibir señales PWM) como leds de diferentes colores los cuales, a través de la función analogWrite se prenden con diferentes intensidades antes de apagarse:

```
int ledRojo = 9;
int ledVerde = 10;
int ledAzul = 11;

void setup() { // no es necesario configurar entradas y salidas
}

void loop() {
        analogWrite(ledRojo, 255);   // enciende ledRojo al 100% (255 = 5V)
        analogWrite(ledVerde, 127);  // enciende ledVerde al 50% (valor 127)
        analogWrite(ledAzul, 0);     // mantiene ledAzul apagado
        delay(1000);                 // espera de 1 segundo

        analogWrite(ledRojo, 50);    // enciende ledRojo con valor intermedio 50
        analogWrite(ledVerde, 213);  // enciende ledVerde con valor intermedio 213
        analogWrite(ledAzul, 189);   // enciende ledAzul con valor intermedio 189
        delay(1000);                 // espera de 1 segundo

        analogWrite(ledRojo, 0);     // apaga ledRojo
        analogWrite(ledVerde, 0);    // apaga ledVerde
        analogWrite(ledAzul, 0);     // apaga ledAzul
        delay(1000);                 // espera de 1 segundo
}
```

## delay (retardo)

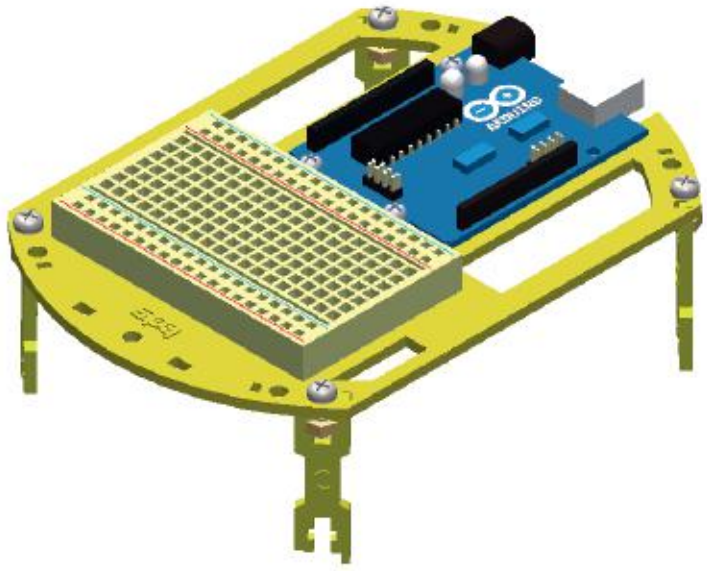
Lo que hace esta instrucción es detener la ejecución del programa durante determinada cantidad de tiempo, la cual debe expresarse en milisegundos. Recuerda que 1000 ms = 1 seg

Su sintaxis correcta es: **delay(ms);**

Ejemplo:
```
delay(1000); // introduce una espera de 1 seg antes de seguir ejecutando el programa.
```

# Ensamble

A continuación veremos la estructura y componentes necesarios encender los leds RGB.

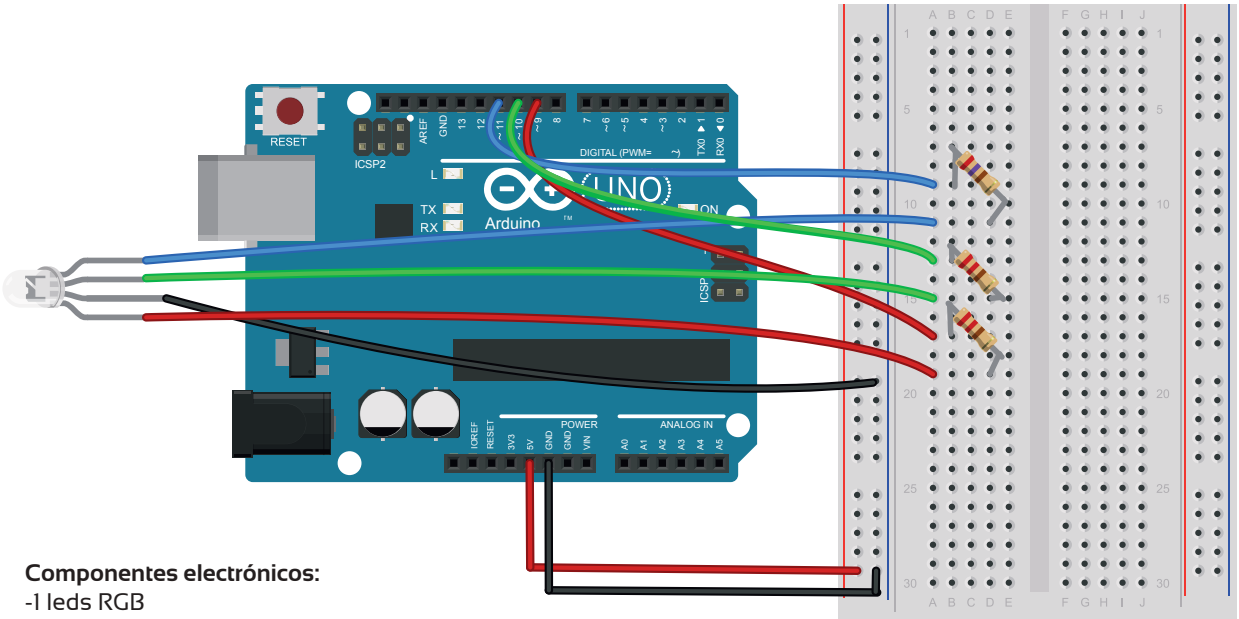


**Materiales:**
-Estructura de Hola Mundo

NOTA: Podemos usar la estructura de Luces para mi robot, no hay falta que desarmes el robot. Si cambiaremos la electrónica.

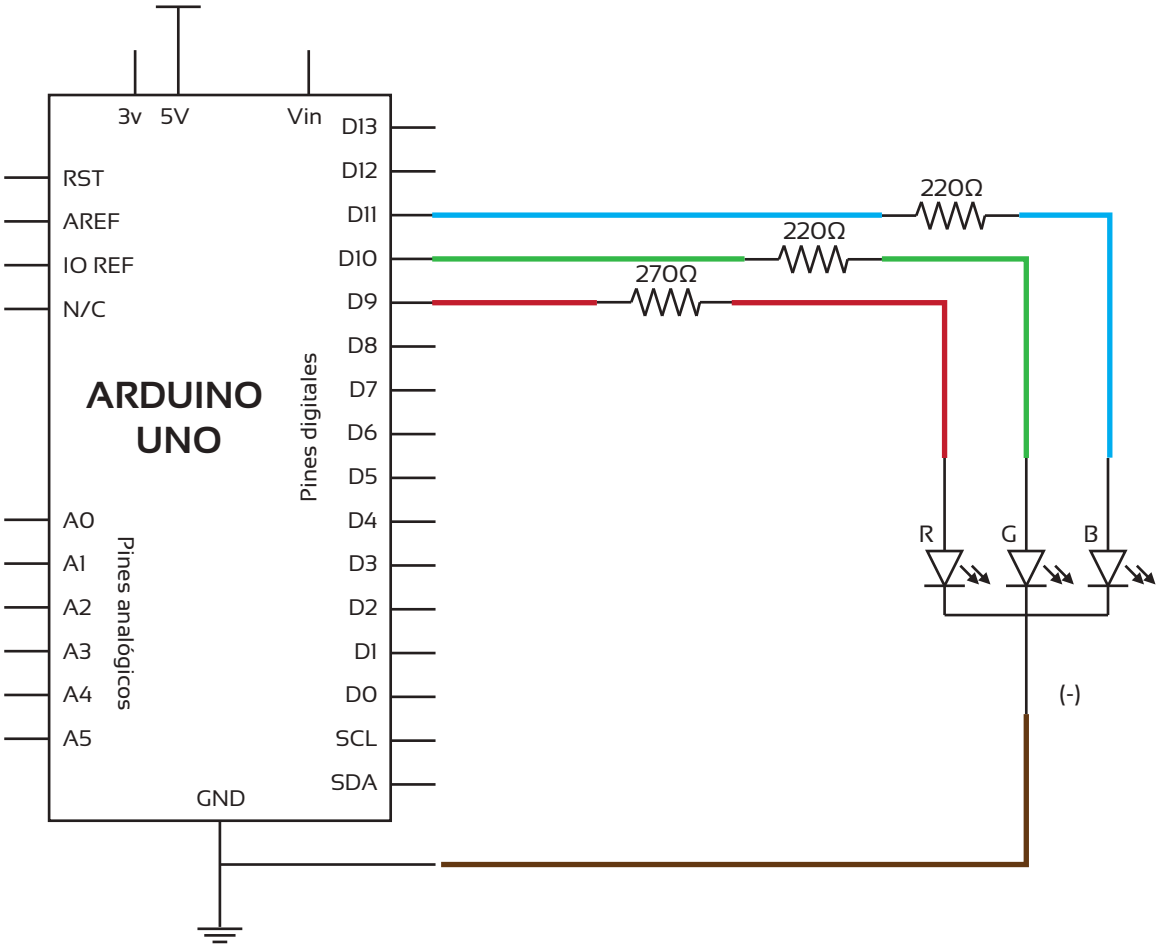
# Circuito electrónico

Finalmente observamos las conexiones de nuestro circuito electrónico.




**Componentes electrónicos:**
-1 leds RGB
-2 resistencias de 220Ω
-1 resistencias de 270Ω
-4 cables M-H
-5 cables M-M

# Circuito electrónico en símbolos



# Código de programación

Desarrollaremos el código de programación para el encendido del led RGB. Vamos a configurar las diferentes salidas de nuestro arduino lograr diferentes colores utilizando la tecnica de adición.

```
1  int ledRojo = 9;
2  int ledVerde = 10;
3  int ledAzul = 11;
4
5  int ledTest = 13;
6
7  int rojo = 0;
8  int verde = 0;
9  int azul = 0;
10 int todos = 0;
11
12
13
14
15
16 void setup() {
17
18   pinMode(ledRojo, OUTPUT);
19   pinMode(ledVerde, OUTPUT);
20   pinMode(ledAzul, OUTPUT);
21   pinMode(ledTest, OUTPUT);
22
23   analogWrite(ledRojo, 0);
24   analogWrite(ledVerde, 0);
25   analogWrite(ledAzul, 0);
26
27   digitalWrite(ledTest, LOW);
28
29   for(int i=0;i<3;i++){
30   digitalWrite(ledTest, HIGH);
31   delay(500);
32   digitalWrite(ledTest, LOW);
33   delay(500);
34   }
35
36 }
37
38 void loop() {
39
40   for (rojo = 0; rojo <= 255; rojo++) {
41     analogWrite(ledRojo, rojo);
42     delay(0);
43   }
44   for (rojo = 255; rojo >= 0; rojo--) {
45     analogWrite(ledRojo, rojo);
46     delay(10);
47   }
48   for (verde = 0; verde <= 255; verde++) {
49     analogWrite(ledVerde, verde);
50     delay(10);
51   }
52
```

//INICIO VARIABLES: Definimos las variables que van a vincular salidas del arduino con los leds de colores R G y B. Para las salidas del led RGB utilizaremos las salidas PWM que se controlan con la función analogWrite. El Led de la plaqueta (pin 13) se utilizará como salida digital

Led Rojo pin de salida 9
Led Verde pin de salida 10
Led Azul pin de salida 11

Declaramos tambien las variables rojo, verde, azul y todos, que las utilizaremos para controlar la potencia de salida.

//INICIO SETUP SALIDAS: Configuramos los pines que necesitamos como salidas utilizando las variables declaradas anteriormente. Una vez realizado esto, nos aseguramos que están apagadas las salidas

//ISaludo inicial con función for

//Cerramos for de línea 29

//Cerramos setup de línea 16

//INICIO ENCENDER COLORES: Vamos a comenzar prendiendo y apagando (gradualmente) los leds independientemente, de esta manera nos aseguraremos que todos están bien conectados. El valor máximo que le podemos pasar a la funcion analogWrite es 255. El intervalo entre incremento e incremento será de 40 milisegundos. Luego de esto vamos a comenzar a mezclar los colores:

Rojo + Verde (incremental)
Rojo + Verde + Azul (incremental)
Rojo + Verde + Azul (decremento Verde)
Rojo + Verde + Azul (decremento Rojo)
Todos incremento luego decremento

```
53   for (verde = 255; verde >= 0; verde--) {
54     analogWrite(ledVerde, verde);
55     delay(10);
56   }
57   for (azul = 0; azul <= 255; azul++) {
58     analogWrite(ledAzul, azul);
59     delay(10);
60   }
61   for (azul = 255; azul >= 0; azul--) {
62     analogWrite(ledAzul, azul);
63     delay(10);
64   }
65
66   analogWrite(ledRojo, 255); //
67   analogWrite(ledVerde, 0);
68   analogWrite(ledAzul, 0);
69   for (verde = 0; verde <= 255; verde++) {
70     analogWrite(ledVerde, verde);
71     delay(10);
72   }
73   for (azul = 0; azul <= 255; azul++) {
74     analogWrite(ledAzul, azul);
75     delay(10);
76   }
77   for (verde = 255; verde >= 0; verde--) {
78     analogWrite(ledVerde, verde);
79     delay(10);
80   }
81    for (rojo = 255; rojo >= 0; rojo--) {
82     analogWrite(ledRojo, rojo);
83     delay(10);
84   }
85   for (azul = 255; azul >= 0; azul--) {
86     analogWrite(ledAzul, azul);
87     delay(10);
88   }
89
90   delay(50);
91
92   for (todos = 0; todos <= 255; todos++) {
93     analogWrite(ledRojo, todos);
94     analogWrite(ledVerde, todos);
95     analogWrite(ledAzul, todos);
96     delay(10);
97   }
98   for (todos = 255; todos >= 0; todos--) {
99     analogWrite(ledRojo, todos);
100    analogWrite(ledVerde, todos);
101    analogWrite(ledAzul, todos);
102    delay(10);
103  }
104 }
```

//Escribe un valor entre 0 y 255 en un pin analógico (~). Se puede usar para encender un LED con luminosidad variable o accionar un motor a diferentes velocidades.

// en este caso, usando "rojo--" en vez de incrementar, decrementa el valor de la variable de control "rojo" en 1.

//Cerramos loop de línea 88

# ACTIVIDADES Y EJERCITACIÓN

# Cap. 5: Actividades complementarias

En base a los contenidos vistos en el **"Capítulo 5: Luces RGB"** realiza las siguientes actividades complementarias.

## Actividad 1

Simplifica la programación utilizando dentro de una función sólo dos bucles FOR: uno para incrementar y el otro para reducir la intensidad de la luz. Pasa por parámetro qué color quieres generar.

```
función (parámetro) {
        bucle FOR 1 {
                //programación para incrementar la intensidad de la luz
        }
        bucle FOR 2 {
                //programación para reducir la intensidad de la luz
        }
}
```

## Actividad 2

Crea una función donde se pueda pasar por parámetro la intensidad de cada color RGB (rojo, verde y azul) de manera individual para conseguir que el LED se ilumine con el color definido.

Una vez creada la función anterior, ponla a prueba y responde las siguientes preguntas:
**a.** ¿De qué color se ilumina el LED cuando le indicas un valor de R = 56, G = 29, B = 98?

**b.** ¿Y si en nuestra función especificamos los parámetros R = 224, G = 5, B = 227?

# Cap. 5: Respuestas

## Actividad 1

Simplifica la programación utilizando dentro de una función sólo dos bucles FOR: uno para incrementar y el otro para reducir la intensidad de la luz. Pasa por parámetro qué color quieres generar.

```
int ledRojo = 11;
int ledVerde = 10;
int ledAzul = 9;

void setup() {
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAzul, OUTPUT);

  analogWrite(ledRojo, 255);
  analogWrite(ledVerde, 255);
  analogWrite(ledAzul, 255);
}

void loop() {
  prendeyApagaColor(ledRojo);  //recordemos que ledRojo, ledVerde y ledAzul son variables int
  prendeyApagaColor(ledAzul);  // que guardan el número de pin conectado a un color
  prendeyApagaColor(ledVerde);
}

void prendeyApagaColor(int pin){
  for (int i = 255; i >= 0; i--) {
    analogWrite(pin, i);        //relacionamos la intensidad del led con la variable i que aumenta o
    delay(10);                  // o decrece
  }
  for (int i = 0; i <= 255; i++) {
    analogWrite(pin, i);
    delay(10);
  }
}
```

## Actividad 2

Crea una función donde se pueda pasar por parámetro la intensidad de cada color RGB (rojo, verde y azul) de manera individual para conseguir que el LED se ilumine con el color definido.

```
int ledRojo = 11;
int ledVerde = 10;
int ledAzul = 9;

void setup() {
  pinMode(ledRojo, OUTPUT);
  pinMode(ledVerde, OUTPUT);
  pinMode(ledAzul, OUTPUT);
```

```
  analogWrite(ledRojo, 255);
  analogWrite(ledVerde, 255);
  analogWrite(ledAzul, 255);
}

void loop() {
  generarColor(56, 29, 98);
  delay(500);
  generarColor(224, 5, 227);
  delay(500);
  }
```

La siguiente parte varía según tengas un led de cátodo común o un led de ánodo común.

Led de cátodo común:

```
void generarColor(int R, int V, int A) {
  analogWrite(ledRojo, R);
  analogWrite(ledVerde, V);
  analogWrite(ledAzul, A);
  }
```

Led de ánodo común:

```
void generarColor(int R, int V, int A) {
  analogWrite(ledRojo, 255 - R);
  analogWrite(ledVerde, 255 - V);
  analogWrite(ledAzul, 255 - A);
  }
```

Esto es así porque un led de ánodo común funciona al revés (apagado en 255, prendido en 0), por eso en lugar de poner directamente el valor, restamos de 255 el valor deseado.

**a. ¿De qué color se ilumina el LED cuando le indicas un valor de R = 56, G = 29, B = 98?**
El color que se forma es violeta.

**b. ¿Y si en nuestra función especificamos los parámetros R = 224, G = 5, B = 227?**
El resultado de esta combinación es el color fucsia.