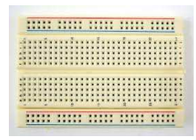
LOS DIODOS LED RGB

OBJETIVOS

- Ampliar la experiencia con los pines PWM.
- Conocer los LED RGB.
- Presentar la función random().

MATERIAL REQUERIDO.

	Arduino Uno o similar. *Esta sesión acepta cualquier otro modelo de Arduino*
	Una Protoboard.
	Un diodo **LED RGB**, independiente, o bien, con montura keyes.
	Una **resistencia** de 330 Ohmios.
	Algunos **cables** de Protoboard..

LOS DIODOS RGB

Hasta ahora hemos usado varias combinaciones de LEDS, pero siempre de un color definido. Habitualmente los rojos y amarillos son los más fáciles de conseguir, pero se pueden comprar también en tonos azules, verdes y hasta blancos. No suele haber grandes diferencias entre ellos excepto en el color.

Pero a veces es interesante disponer de una luz piloto que cambie de color según las condiciones. Por ejemplo, todos identificamos el verde como una señal de OK, mientras que el rojo indica problemas y el amarillo… bueno pues algo intermedio.

Poner varios diodos para hacer esto es engorroso y complica el diseño, así que estaría bien disponer de un diodo al que podamos indicar que color queremos que muestre. Esto es un **LED RGB**.

Para quien este acostumbrado al diseño por ordenador ya está familiarizado con la idea de que podemos generar cualquier color en la pantalla con la mezcla, en diferentes grados de tres colores básicos:

Red : Rojo

Green: Verde

Blue: Azul

Es decir RGB, uno de esos acrónimos que surgen continuamente en imagen, TV, etc.

Un **LED RGB** es en realidad la unión de tres LEDs de los colores básicos, en un encapsulado común, compartiendo el Ground (cátodo es otro nombre más para el negativo).

En función de la tensión que pongamos en cada pin podemos conseguir la mezcla de color que deseemos con relativa sencillez

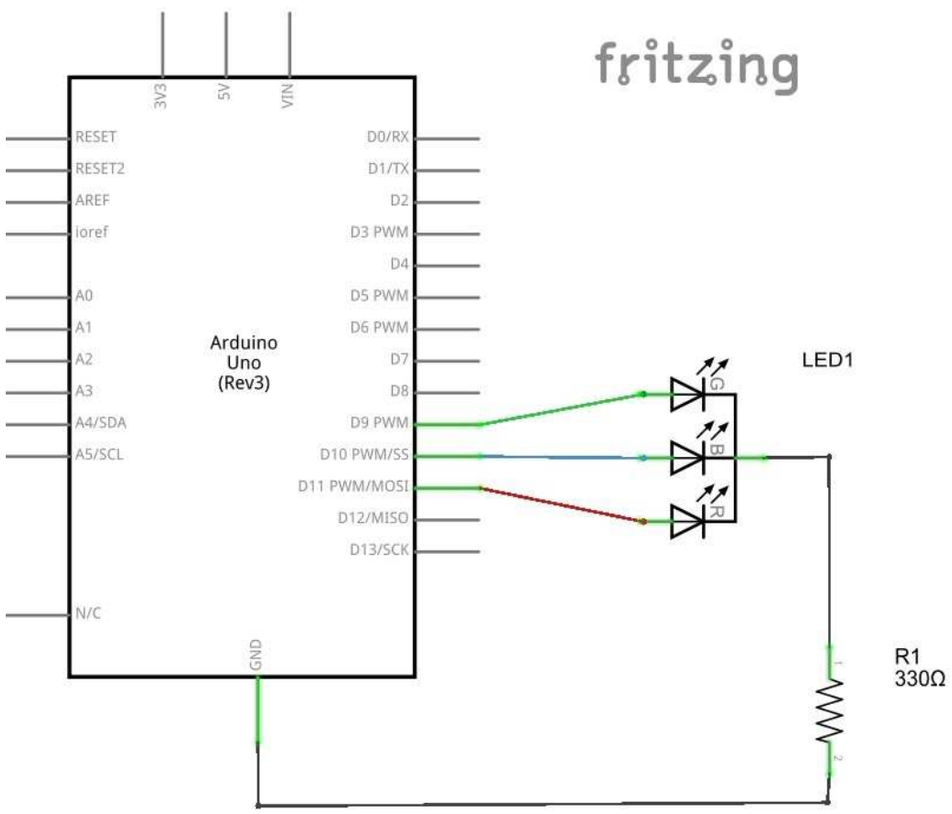
Para quien haya dibujado con lápices de colores o acuarelas, las mezclas de colores de arriba les resultará extraña. Esto es porque cuando pintamos en un papel blanco, la mezcla de colores es substractiva: Si mezclamos los tres colores obtenemos negro, o por lo menos algo oscuro

En cambio cuando pintamos con luz directamente, la mezcla es aditiva y obtenemos blanco al mezclar los tres colores básicos. Las reglas de mezcla de color en ambos casos son opuestas.

Vamos a montar un pequeño circuito que nos permita gobernar el color que emite uno de éstos LEDs de RGB.

ESQUEMA DEL CIRCUITO

.



El montaje supone sencillamente conectar el negativo (el pin más largo) a Ground mediante una resistencia que limite la intensidad, y luego identificar los pines de colores:

El pin más largo en estos LED es el GND.

Al lado de GND hay dos pines a un lado y uno solitario al otro. Por lo normal el solitario es el rojo R.

Así pues el pin out (patillaje) de un RGB LED suele ser R, GND, G, B.

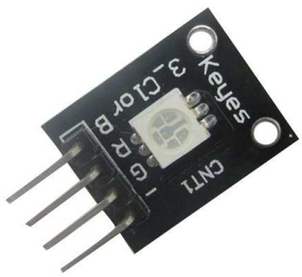
De todos modos conviene asegurarse leyendo las especificaciones del fabricante, o bien identificando cada PIN. *Para identificarlos basta conectar el GND a nuestro Arduino e ir probando cada una de las patas independientemente para ver qué color producen.*

Si tu RGB tiene una montura Keyes, no tendrás que hacer esto, porque los pines vienen marcados y GND viene rotulado como -.

Atención, en contra de la norma habitual, en este caso el cable rojo no indica la tensionVcc, sino el pin de gobierno del LED rojo.

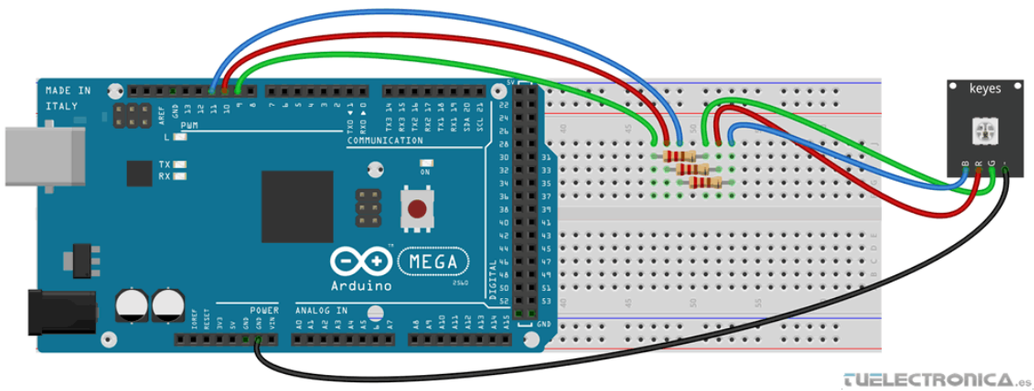
Módulos Keyes en el mercado:

No necesita resistencias, estan incluidas. Va directo.

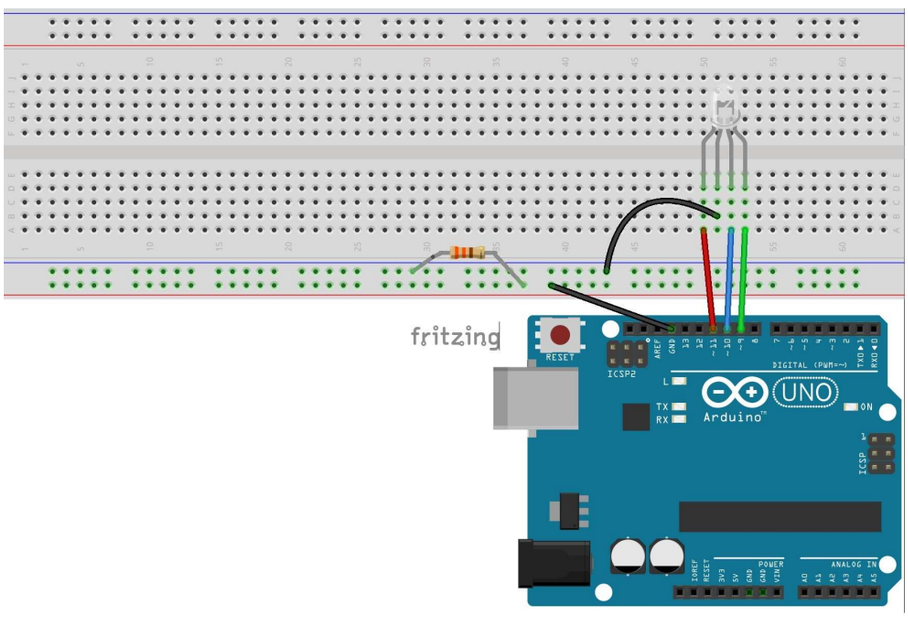


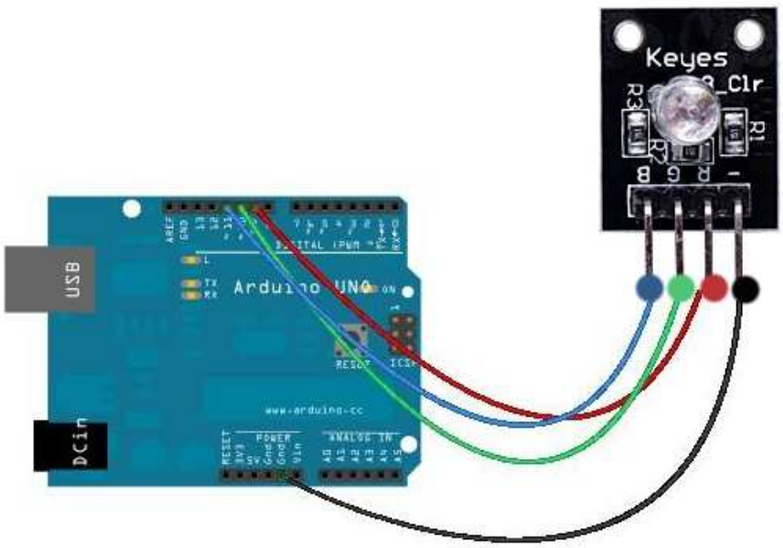
Necesita agregar 3 resistencias, ya que no están incluidas.

Opcion1 Modulo Keyes sin resistencias incluidas.



Opcion 2 Led RGB independiente





En este esquema hemos utilizado los pines 9, 10 y 11. Podemos usar otros pero aseguraros de que puedan hacer PWM(los que tienen ~) para poder poner distintas intensidades.

PROGRAMA DE CONTROL RGB

Dado que nuestra idea es poder mezclar las tonalidades de los componentes RGB para generar diferentes matices de colores, parece buena idea escribir una función que haga esta mezcla de colores y a la que podamos recurrir de forma abstracta y práctica (además de para encapsular una utilidad curiosa, a la que podremos recurrir en futuros ejemplos y de paso insistir en el concepto de función).

Lo primero sería definir en el setup() los pines a usar:

```
void setup()
   {
       for (int i =9 ; i<12 ; i++)
            pinMode(i, OUTPUT);
   }
```

Y después podríamos escribir una función como esta

```
void Color(int R, int G, int B)
    {
        analogWrite(9 , R) ;   // Red    - Rojo
        analogWrite(10, G) ;   // Green - Verde
        analogWrite(11, B) ;   // Blue - Azul
    }
```

De este modo tendríamos fácil llamar a Color (0, 255, 0) para el verde. De hecho vamos a empezar asegurándonos de que tenemos identificados correctamente los pines, escribiendo un sketch como este:

```
void loop()
   {    Color(255 ,0 ,0) ;
        delay(500);
        Color(0,255 ,0) ;
        delay(500);
        Color(0 ,0 ,255) ;
        delay(500);
        Color(0,0,0);
        delay(1000);
   }
```

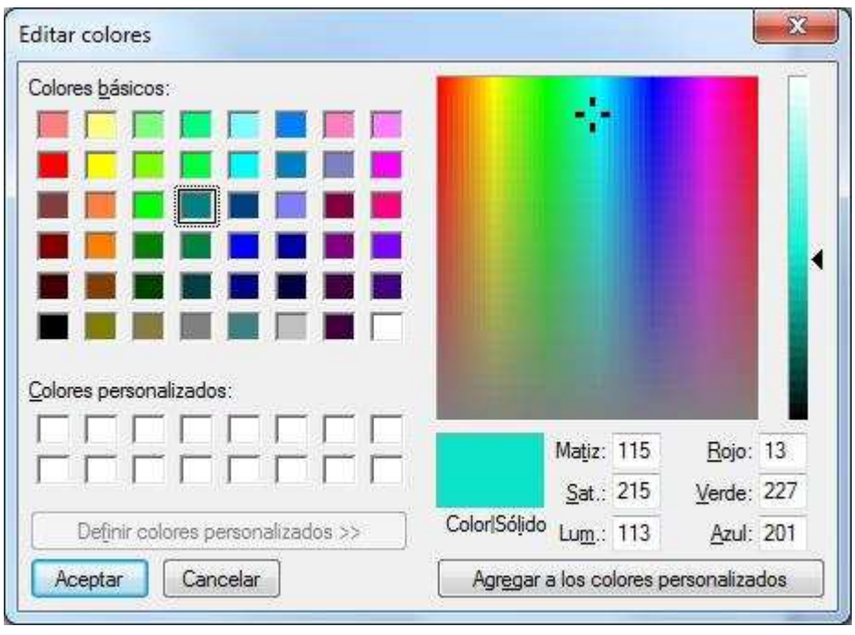
Este programa debería producir una secuencia de rojo, verde, azul, apagado y vuelta a empezar. Es el del Ejemplo 1 de Proteus

Conviene asegurarse de que hemos identificado correctamente los pines del RGB, porque de lo contrario, las mezclas posteriores de colores no serán lo que esperamos.

Vamos a ver como averiguar qué mezcla de RGB necesitamos para conseguir un color determinado. Para quienes uséis Windows disponéis del programa Paint incluido (en el menú de accesorios) y para quienes uséis Mac o Linux tenéis programas similares.

Si arrancáis el Paint(o equivalente) suele tener un selector de colores:

Pulsándolo aparecerá algo parecido a esto:



Si vais pinchando en la zona de colores de la derecha, en la barra vertical aparecen los matices

próximos al que habéis pinchado y podéis elegir el que más os guste. Debajo podéis ver la separación en RGB precisa para conseguir un tono determinado.

Así pues para conseguir ese tono de azulito de la imagen basta con que llaméis a

```
Color(13, 227, 201) ;
```

Dado que Arduino nos permite escribir valores de 0 a 255 en los pines digitales, cuando utilizamos analogWrite(), en la práctica tendremos 255 x 255 x 255 colores diferentes o lo que es igual: 16.581.375 colores posibles.

La función Color() que hemos creado en esta sesión es muy sencilla pero se va añadiendo a otras que hemos ido creando en sesiones anteriores con lo que vamos haciendo una pequeña colección de ellas.

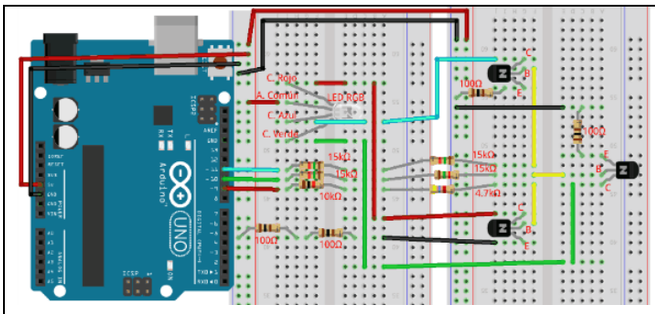
El grupo de desarrollo de Arduino ha ido creando también muchas funciones que están disponibles para incorporar en nuestros programas y que por razones de espacio resultan imposibles de ver más que muy por encima.

Solo como ejemplo introduciremos una de ellas. La función La función random(N) devuelve un valor al azar, comprendido entre 0 y N y en este caso, se presta especialmente bien para generar colores aleatorios en nuestro LED RGB. Probad esto:

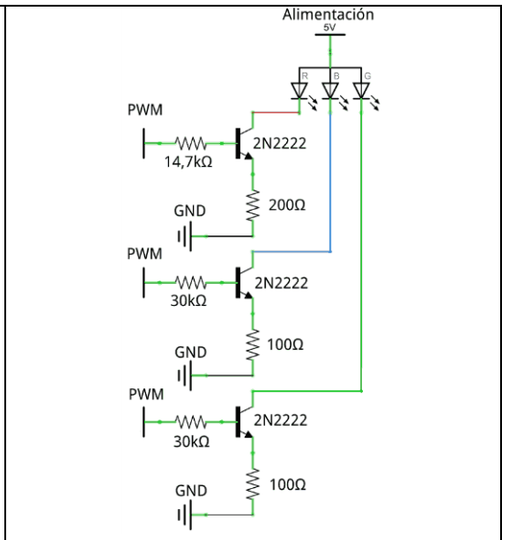
```
  void setup()           //Prog_11_3
      {
         for (int i =9 ; i<12 ; i++)
              pinMode(i, OUTPUT);
      }

   void loop()
  {
        Color(random(255), random(255), random(255)) ;
        delay(500);
  }

void Color(int R, int G, int B)
  {
        analogWrite(9 , R) ;   // Rojo
        analogWrite(10, G) ;   // Green - Verde
        analogWrite(11, B) ;   // Blue - Azul
  }
```

Se generará un ciclo de colores aleatorios bastante psicodélico. Ejemplo 2 de Proteus.

NOTA:



Siempre es bueno no conectar directamente los elementos a controlar, en este caso LEDs. Seria prudente una interfase con transistores.



RESUMEN DE LA SESIÓN

- Ya conocemos la utilidad y manejo de un LED RGB.
- Hemos insistido con la programación de los pines de Arduino como salidas analogías (PWM).
- Hemos introducido la función random() que suele ser sorprendentemente útil en un montón de situaciones diversas.