

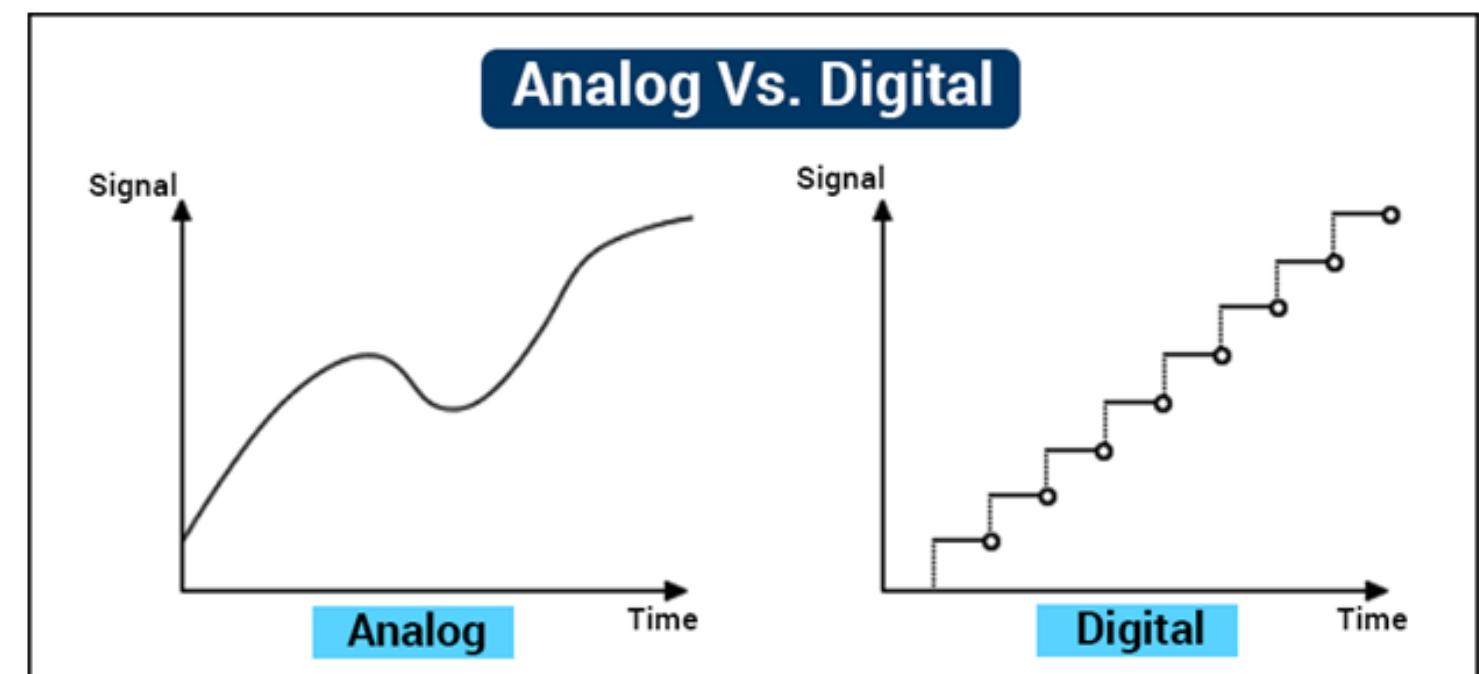
# ¿Digital o analógico?



# ¿Digital o analógico?



Es tema de debate la definición y clasificación de sistemas y señales respecto al área de la electrónica digital y analógica, mientras que entre estas ramas de la electrónica no existe comparación, más bien, son complementarias.

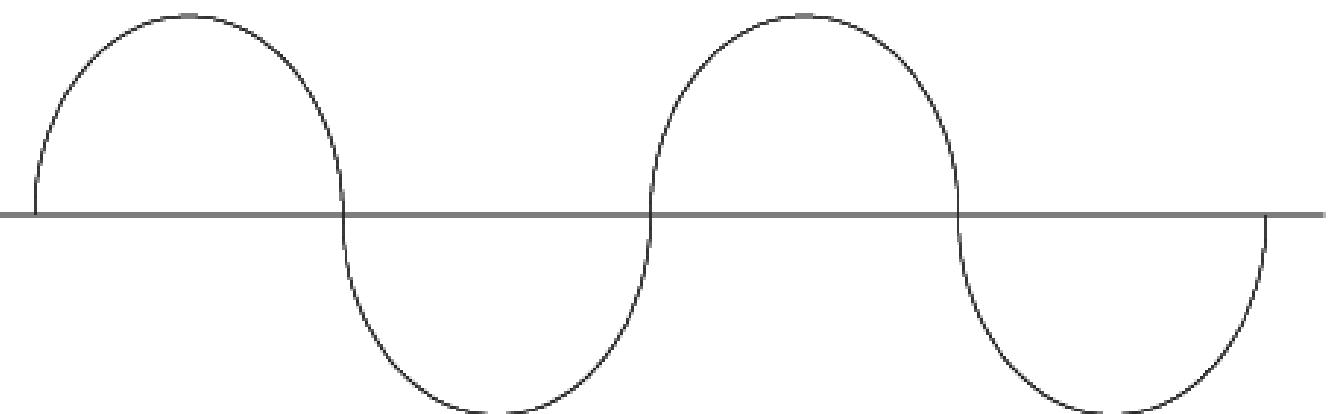


# Sistemas analógicos



La electrónica analógica consiste en el diseño de circuitos y sistemas cuya principal características es que las señales con las que trabaja pueden tomar valores infinitos, es decir, si se trabajan con 5 volts, el sistema puede manejar valores de voltaje de 4.6V, 4.06V, 4.006V, 4.0006V etc.

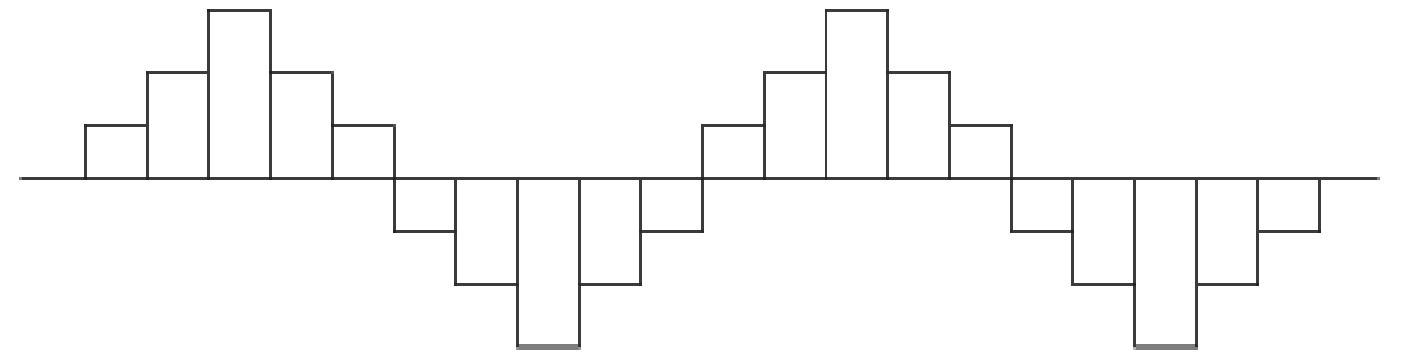
**Analog**



# Sistemas digitales



## Digital



A diferencia de un sistema analógico, un sistema digital únicamente puede trabajar con valores discretos de voltaje, es decir, niveles de voltaje preestablecidos y fijos, generalmente estos voltajes son 0V, 3.3V en CMOS y 5V para TTL.

# Mundo analógico



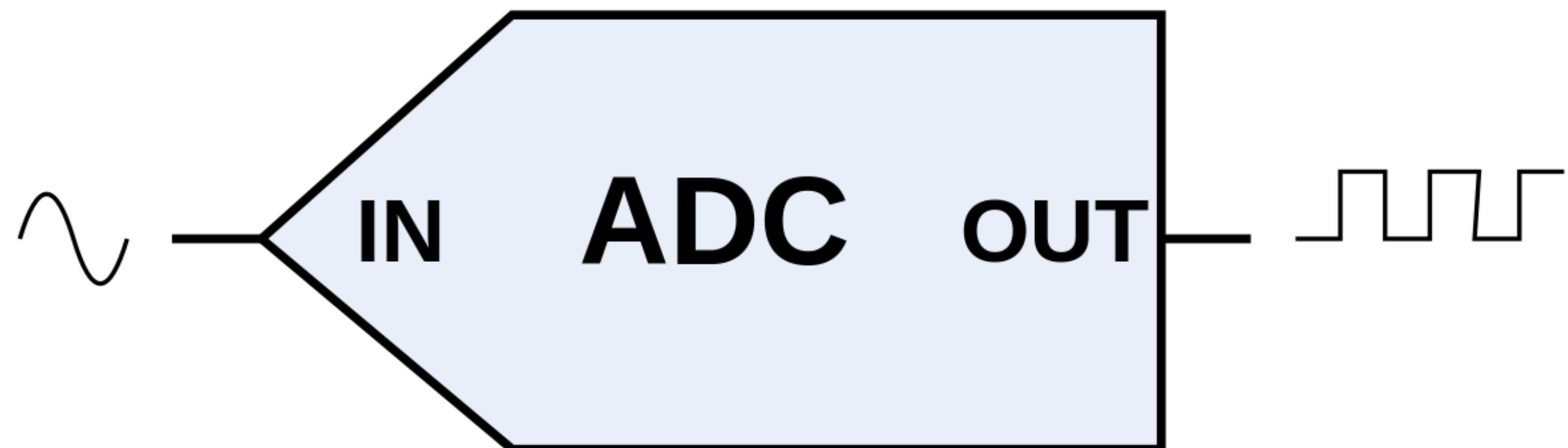
Como humanos, nuestros sentidos nos dan la capacidad de percibir nuestra realidad de manera "analógica", es decir, somos capaces de percibir gamas y matices de luz, sonido y temperatura, entonces, ¿cómo un sistema digital puede interpretar o funcionar en un mundo analógico?



# Convertidor analógico-digital.



El encargado de hacer la "traducción" de señales analógicas a digitales es el llamado ADC (Analog-Digital Converter), este módulo se encarga de tomar los valores analógicos de voltaje de un puerto y transformarlo en una señal digital.



# Resolución del ADC



El microcontrolador ATmega328 cuenta con un ADC de 8 canales, este ADC, según los datos del fabricante tiene una resolución de 10 bits, es decir, puede tomar un rango de valores digitales de 0 a 1,023. El equivalente al valor analógico dependerá de la referencia o del voltaje de alimentación del microcontrolador.

$$1111111111_2 = 1,023_{10}$$

# ADC en Arduino UNO

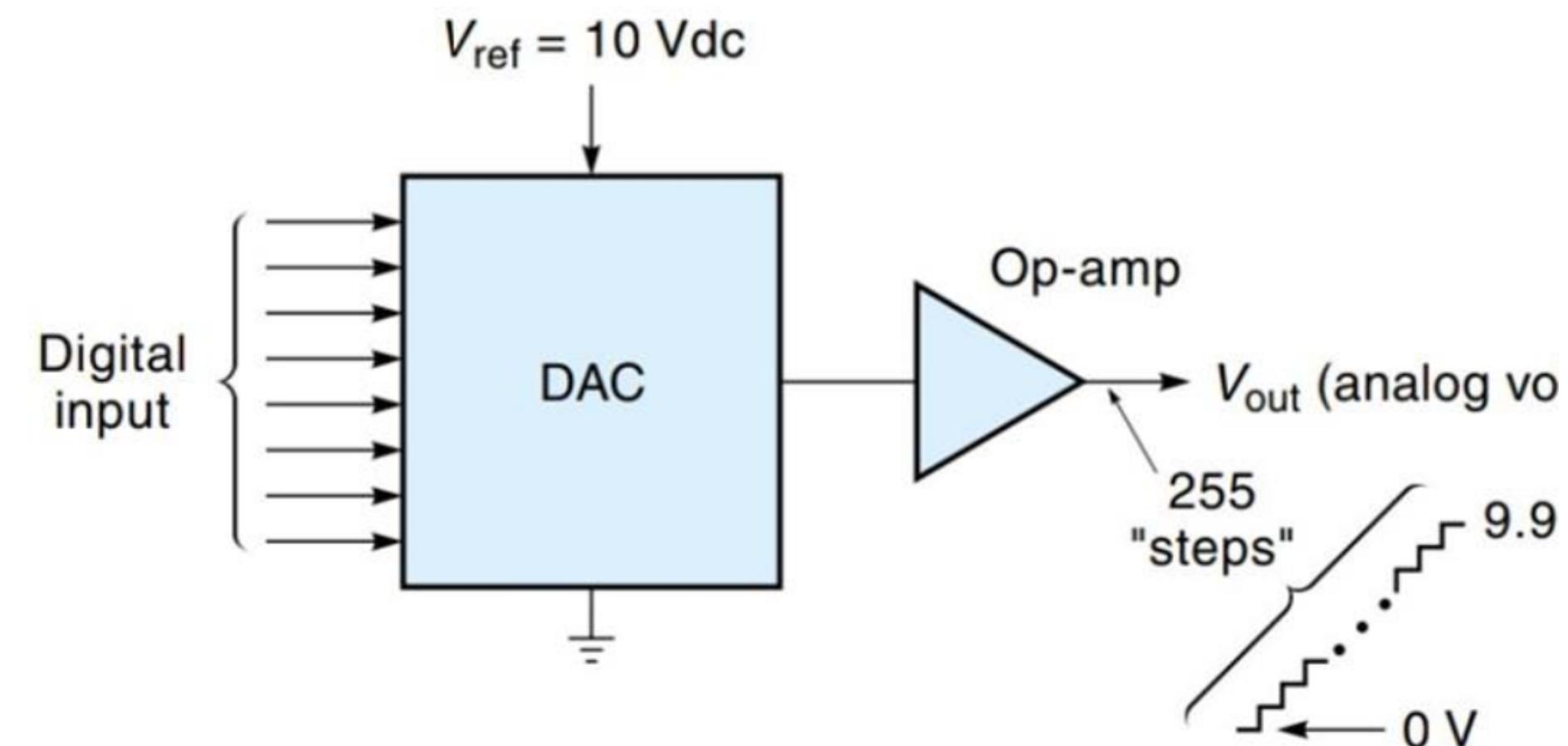
La tarjeta tipo UNO se alimenta con un voltaje de 5V, tomando en cuenta que el ADC tiene una resolución de 10 bits, para poder saber la resolución analógica, simplemente dividimos el voltaje de alimentación entre la resolución en bits.

$$\frac{5V}{1,023} = 0.004887V$$

Esta sería la "sensibilidad" del microcontrolador ATmega328.

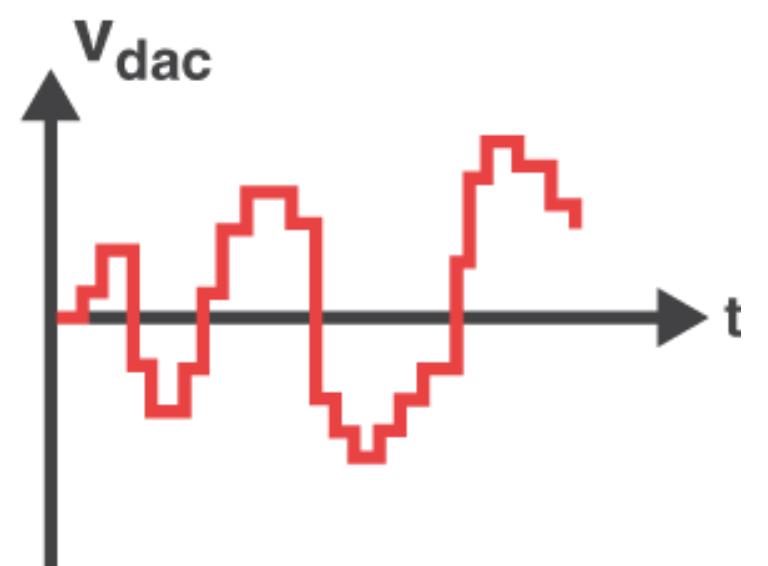
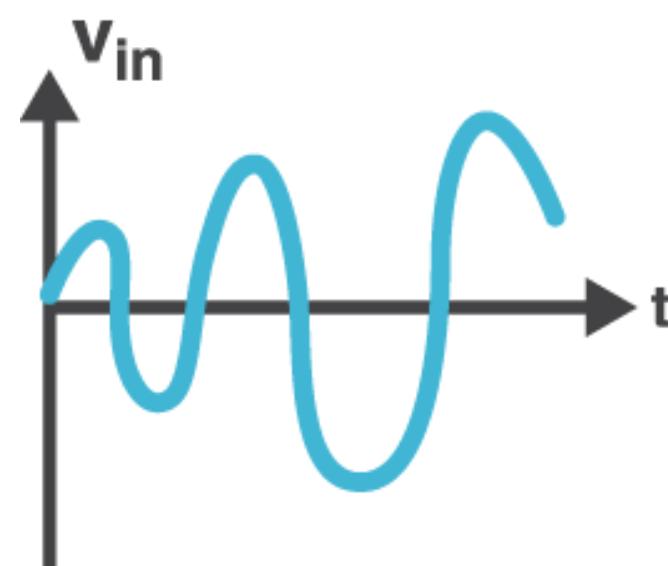
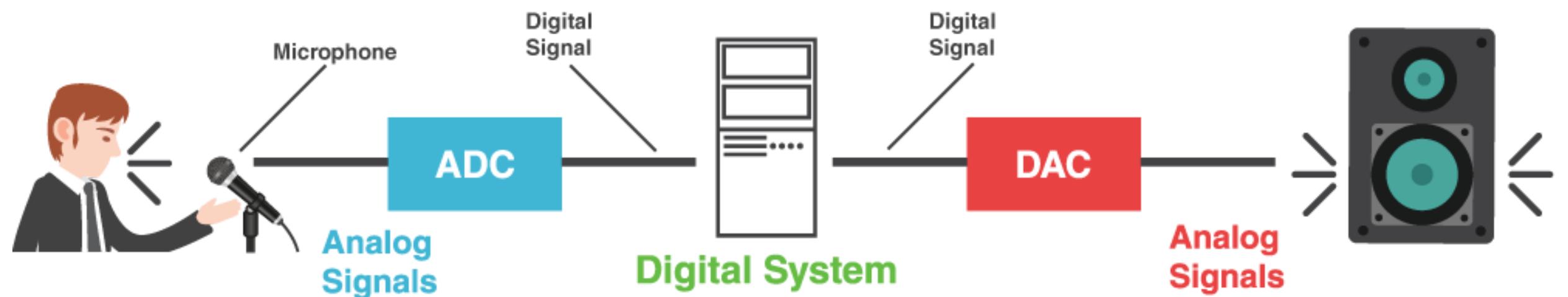


# Convertidor digital-analógico.

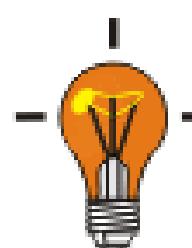
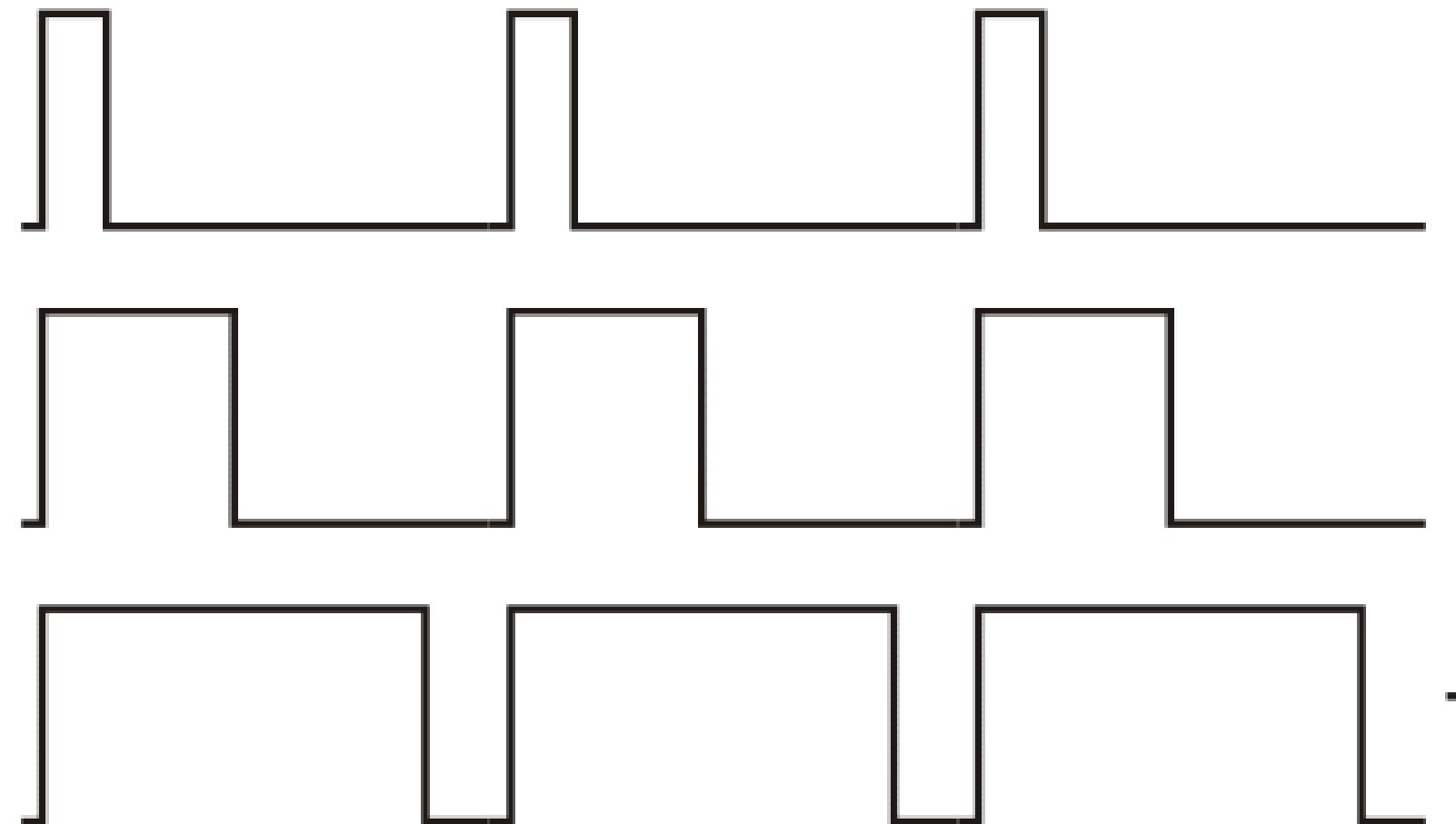


Por otro lado, la versión "inversa" del ADC es el DAC (Digital-analog converter), este dispositivo funciona de manera inversa, convierte la señal digital a una señal "analógica". El ATmega328 NO cuenta con DAC, pero existe una manera de "engañar" a nuestros sentidos para generar señales "analógicas".

# Ejemplo de uso



# Pulse Width Modulation



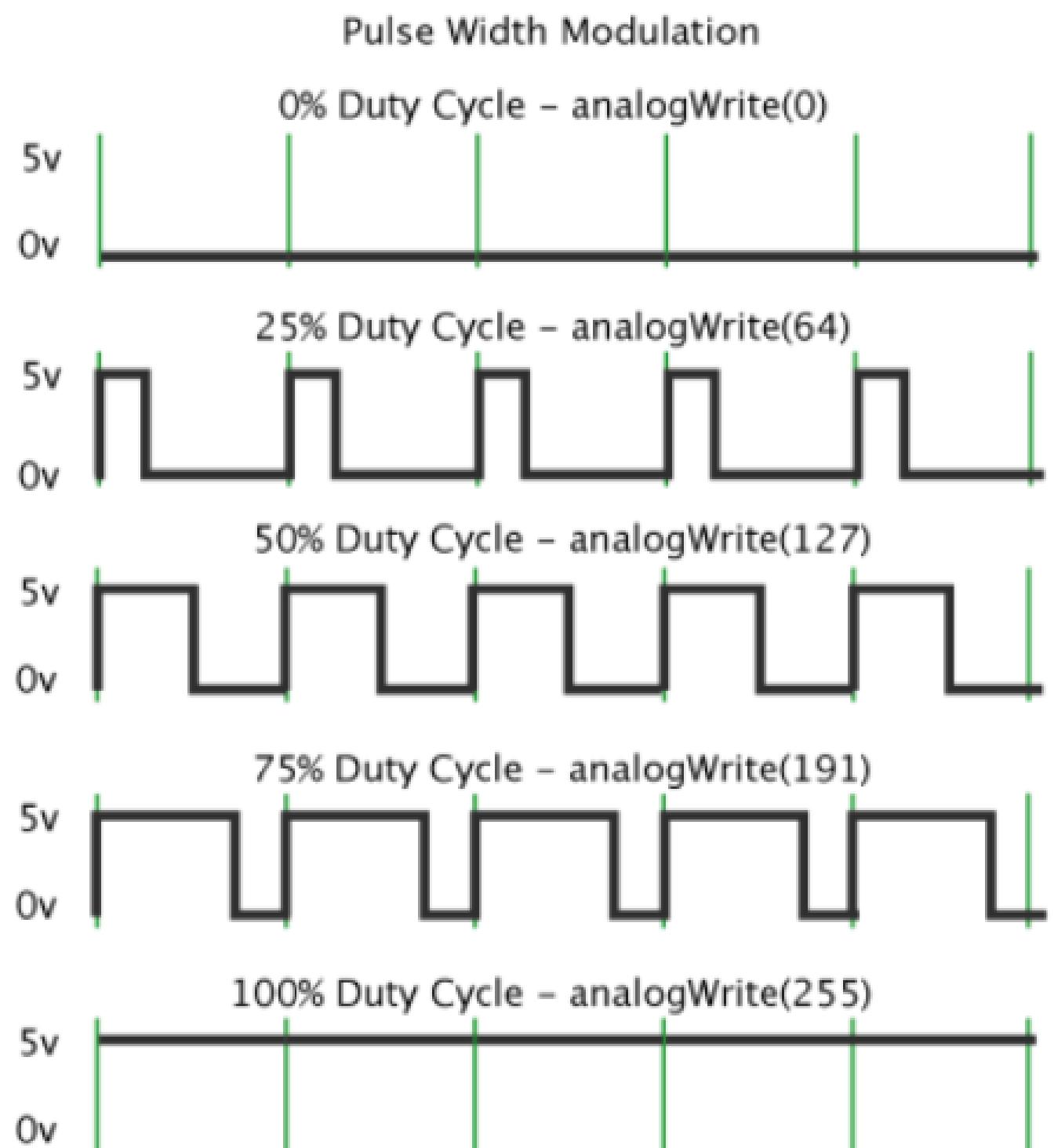
La manera de "engañar" a nuestros sentidos es a través del PWM, función que se encuentra en el ATmega328, este método funciona para controlar el tiempo de encendido y apagado de un puerto dentro de un intervalo de tiempo definido llamado "periodo".

# Pulse Width Modulation

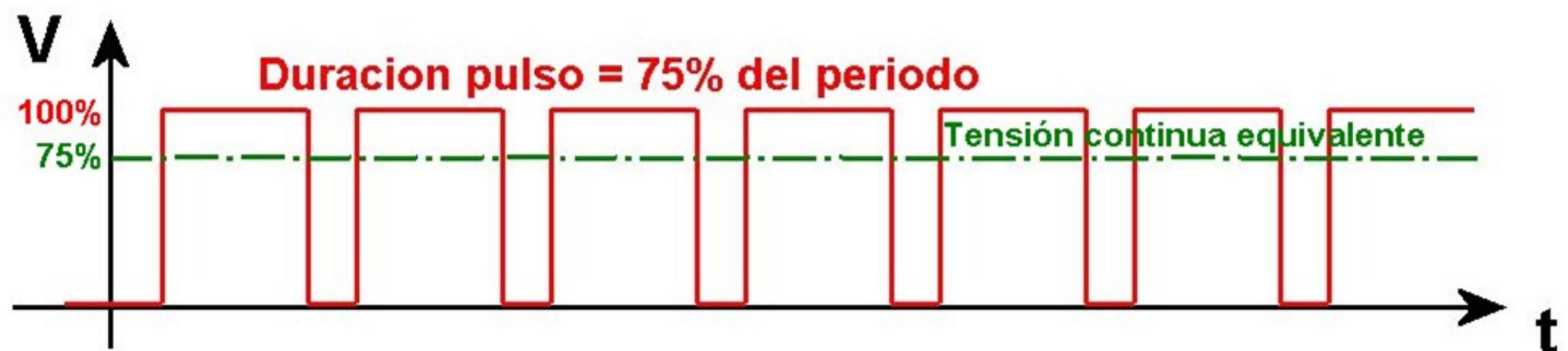
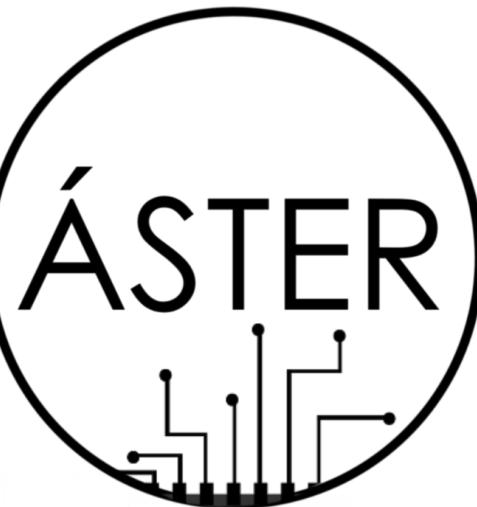
La tarjeta Arduino UNO cuenta con 6 salidas PWM de 8 bits, es decir, sólo pueden tomar valores de 0 a 255, el voltaje promedio a la salida será directamente proporcional al ciclo de trabajo multiplicado por el voltaje de referencia, es decir:

$$\% \text{DutyCycle} \times V_{\text{ref}} = V_{\text{out}} \text{ (promedio)}$$

$$\% 50 \times 5V = 2.5V$$



# Pulse Width Modulation

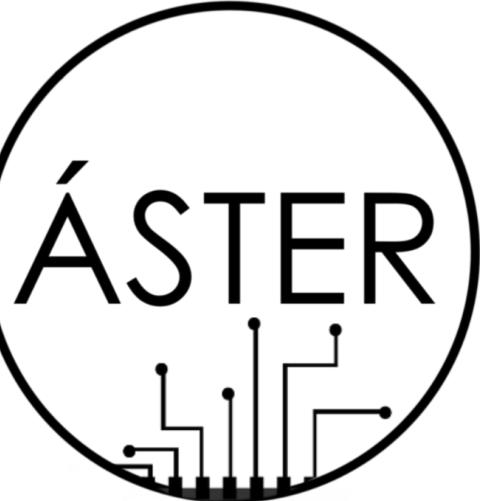




Tiempo de descanso y de restablecer humanidad.(:

# ¿Cómo usar el ADC y el PWM en Arduino?

Por suerte, el Arduino IDE cuenta por defecto con funciones dedicadas al uso y al manejo de estas herramientas de una manera fácil y rápida.



# pinMode



Palabra reservada

```
pinMode(3,OUTPUT);
```

Pin impreso en  
la tarjeta

OUTPUT o INPUT  
dependiendo si es  
salida o entrada

Esta función nos permite configurar como entrada o salida (INPUT, OUTPUT) los pines de nuestra tarjeta, no importa si es analógico o digital, tenemos que pensar si "entran o salen" datos o señales.(1)

# digitalRead



Palabra reservada

digitalRead(3);

Pin impreso en  
la tarjeta

Esta función nos permite leer una señal digital en el pin que nosotros le coloquemos dentro del parentesis, como resultado nos retornará un HIGH si el valor es alto o 5V o un LOW si el valor es bajo o 0V, funciona tanto en pines digitales como análogicos.(2)

# digitalWrite



Palabra reservada

digitalWrite(3,HIGH);

Pin impreso en  
la tarjeta

Argumento de  
estado del pin

Esta función nos permite colocar o modificar el estado lógico o digital de un pin, esta función funciona tanto en pines digitales como analógicos, acepta sólo dos parámetros HIGH y LOW.(3)

# analogRead

Palabra reservada

analogRead(A0);

Pin impreso en  
la tarjeta



Esta función nos permite leer el valor analógico en un pin dentro de un rango entre 0V y el valor de referencia o alimentación, esta función únicamente funciona en pines analógicos y deberán estar previamente configurados como entradas(INPUT).(4)

# analogWrite



Palabra reservada

Pin impreso en  
la tarjeta

Argumento del  
%duty cycle

analogWrite(6,255);

Esta función nos permite desplegar un señal PWM a través del pin seleccionado previamente como salida, como segundo parámetro nos pide un número de 8bits, es decir, sólo trabaja con valores de 0 a 255 y representa el ciclo de trabajo del PWM, siendo 0 el 0% y 255 el 100%. Esta función únicamente trabaja sobre los pines denotados con el símbolo ~ indicado en la placa.

(5)

# Problema al usarlo



`analogRead` nos entrega valores de 10 bits, es decir, puede adquirir un valor de hasta 1,023 mientras que `analogWrite` únicamente recibe valores de 0 a 255. ¿Cómo podemos solucionar esta pequeña problemática?

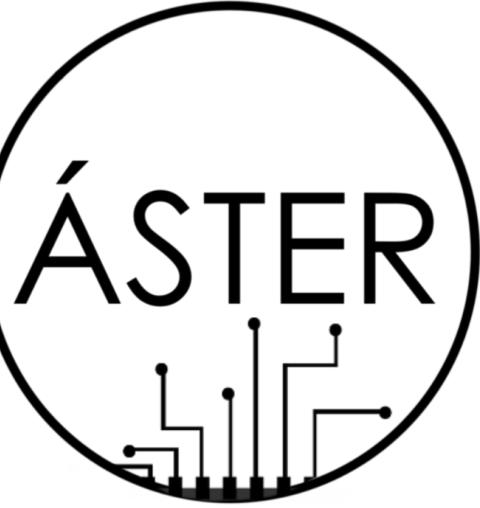
```
analogRead(A0);  
analogWrite(6,255);
```

# Primera solución

La documentación oficial de Arduino (5) nos dice que una de las manera más fáciles es dividir el valor del analogRead entre 4, esto para "ajustar " la proporción entre una variable y otra.

$1023/4 = 255.75$  ; si usamos una variable del tipo `int`, el valor se truncará en 255.

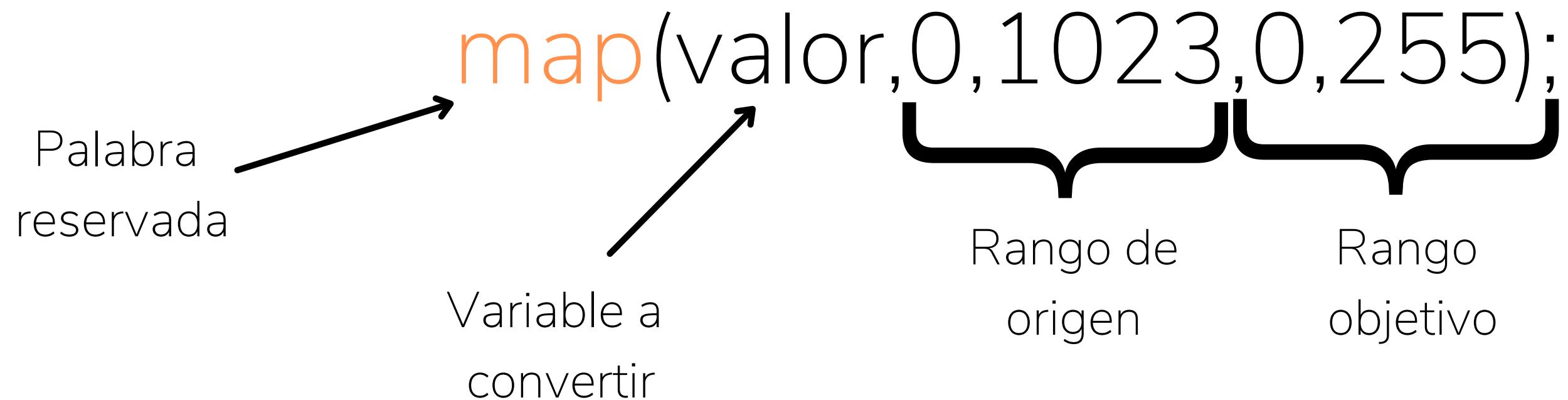
```
analogRead(A0);  
valor=analogRead(A0)/4;  
analogWrite(6,valor);
```



# Segunda solución



Usando la función `map()`; que ejecuta una proporción o una "regla de tres" a una variable, de esta manera nosotros podemos ingresar el primer rango a convertir y también ingresar a qué rango queremos ajustar esa variable(6).



# Segunda solución



```
valor=analogRead(A0);  
valor=map(valor,0,1023,0,255);  
analogWrite(6,valor);
```

# Segunda solución

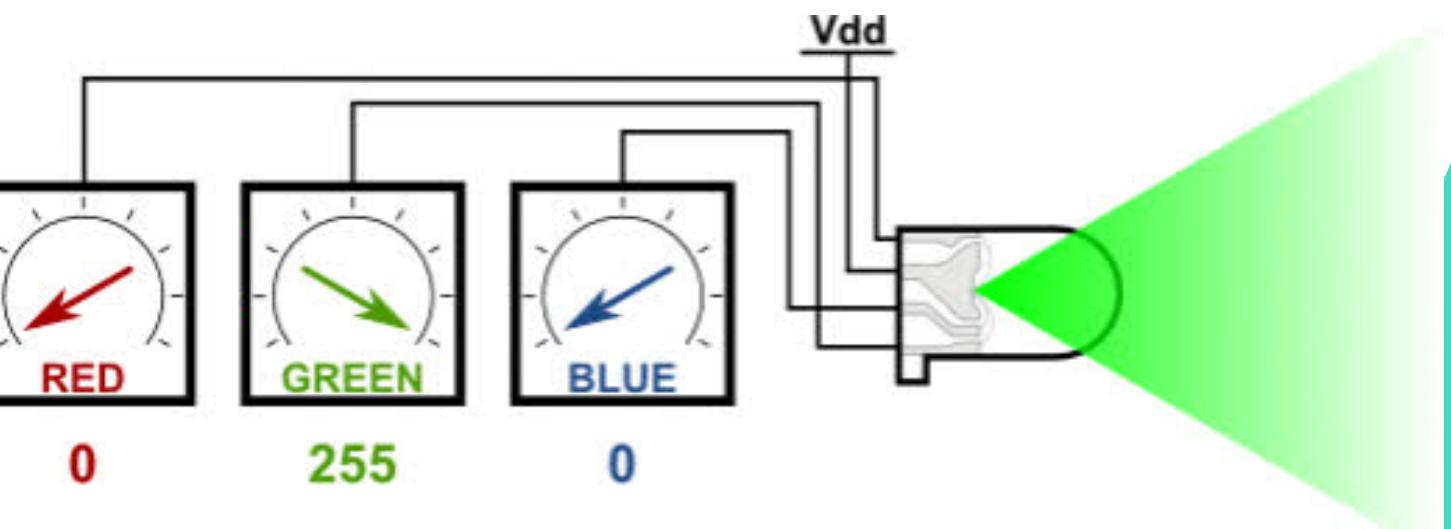


```
analogWrite(6,map(analogRead(A0),0,1023,0,255));
```

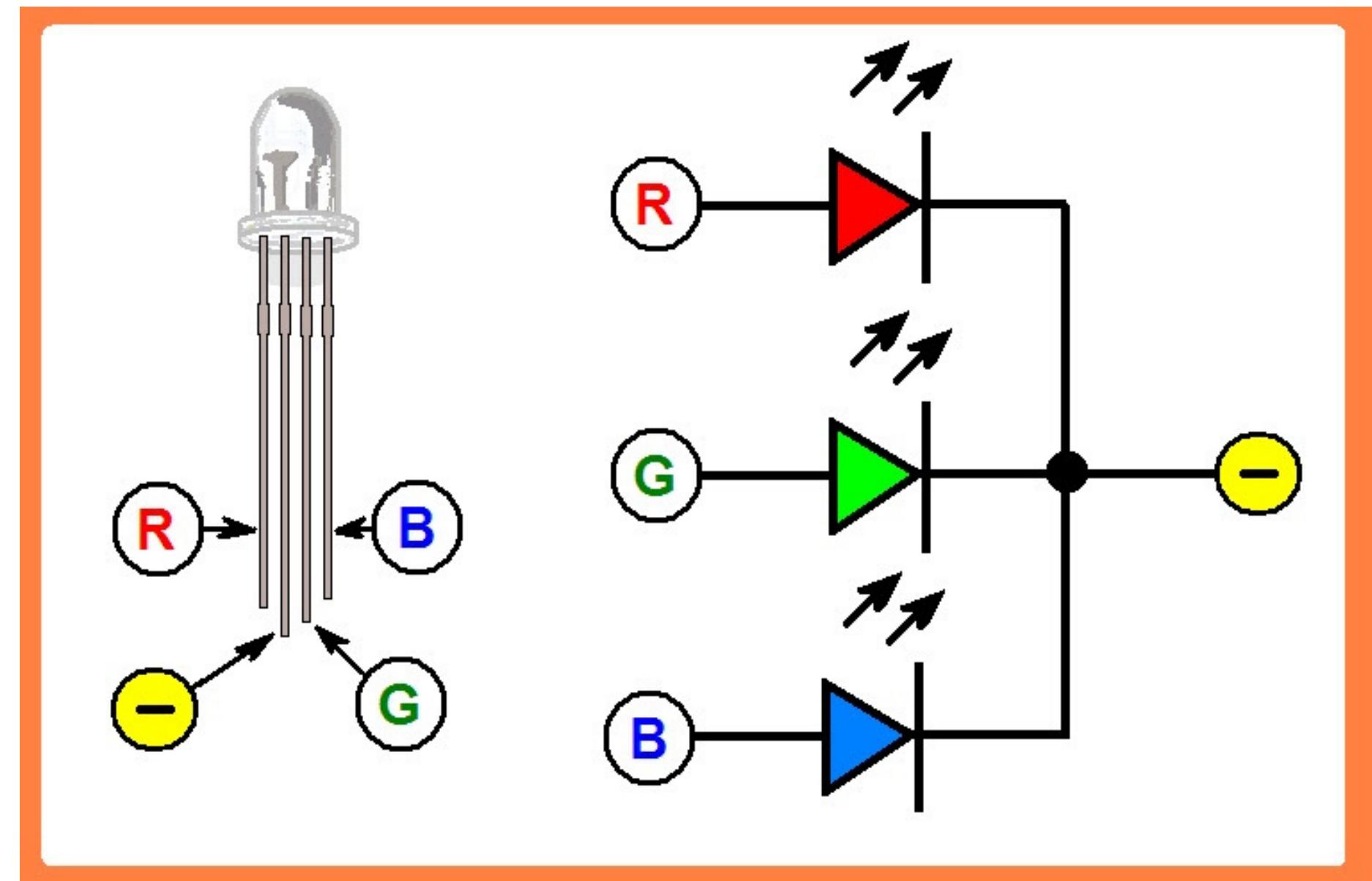
# Control de LED RGB



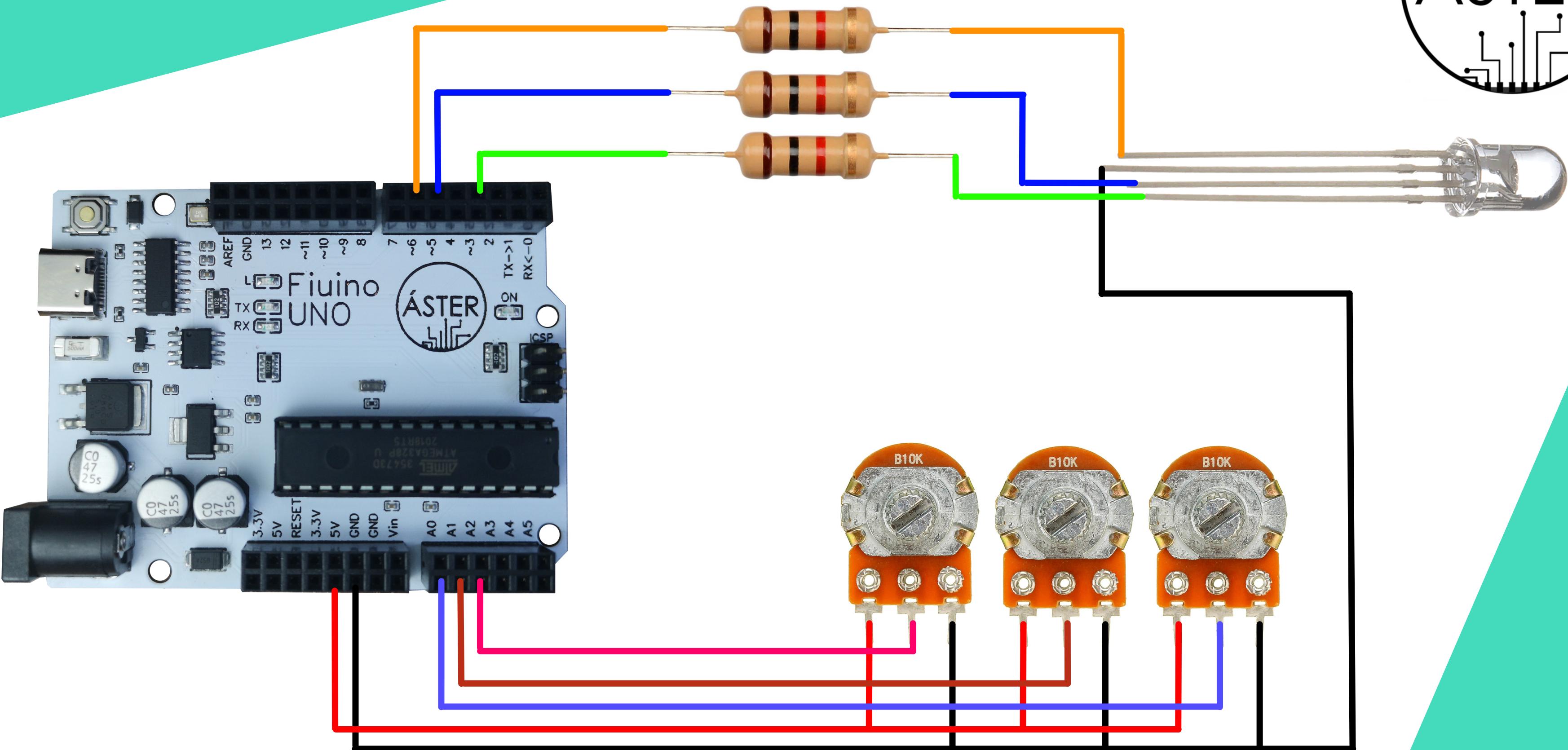
En esta práctica controlaremos el color y el nivel de brillo de un LED RGB mediante 3 potenciómetros o bien, de manera completamente autónoma y aleatoria.



# Control de LED RGB



# Diagrama completo



# Fuentes



1. <https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/>
2. <https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/>
3. <https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/>
4. <https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/>
5. <https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/>
6. <https://www.arduino.cc/reference/en/language/functions/math/map/>
7. <https://www.arduino.cc/reference/en/language/functions/random-numbers/random/>