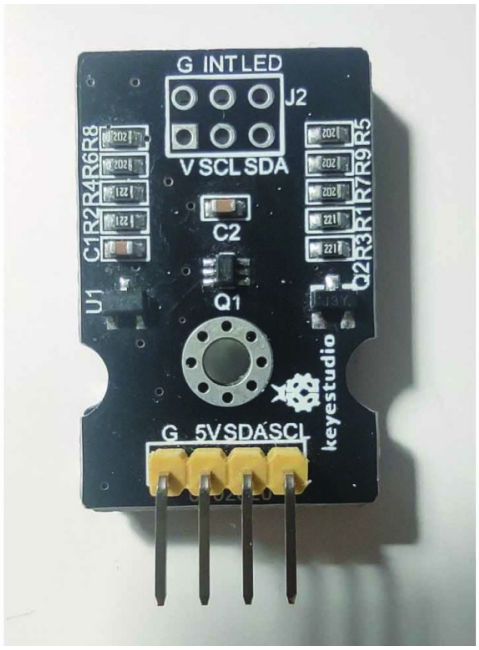
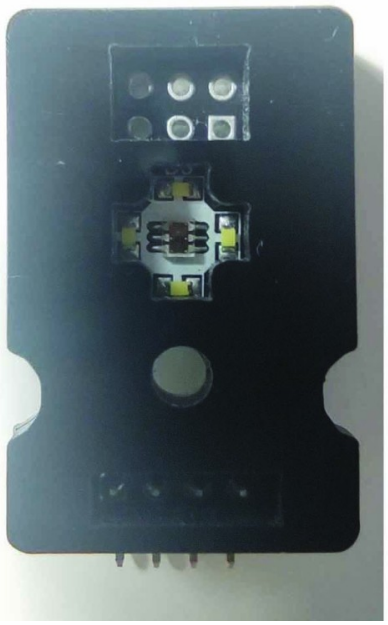
Práctica 13: Cómo percibir el color con Arduino

Para percibir un determinado color, vamos a utilizar uno de los sensores que vienen en nuestro kit, el TCS34725, al que por comodidad denominaremos '**sensor de color**'.

En uno de sus lados, como ves en la primera foto, puedes ver **4 pines**, que son los que conectaremos a nuestra protoboard y de esta a nuestro Arduino.

En el otro lado, como se muestra en la segunda foto, puedes ver el sensor que, rodeado de **cuatro LED blancos** se encarga de determinar la cantidad de **rojo**, **verde** y **azul** que contiene el objeto que le coloquemos delante.

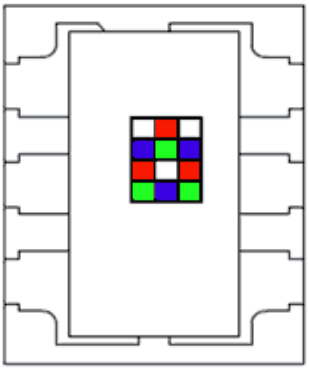


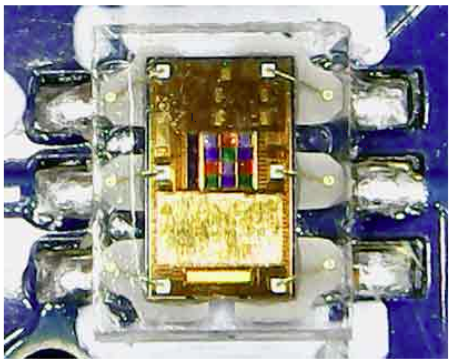


¿Cómo funciona?

Copyright 2026

Este sensor óptico consta de una matriz de **3x4 fotodiodos** filtrados para rojo, verde, azul, y sin filtro (clear). También cuenta con **4 conversores analógico digital** de 16bits (ADC) que realizan la medición de los fotodiodos:





Fuente: https://www.luisllamas.es/arduino-sensor-color-rgb-tcs34725/

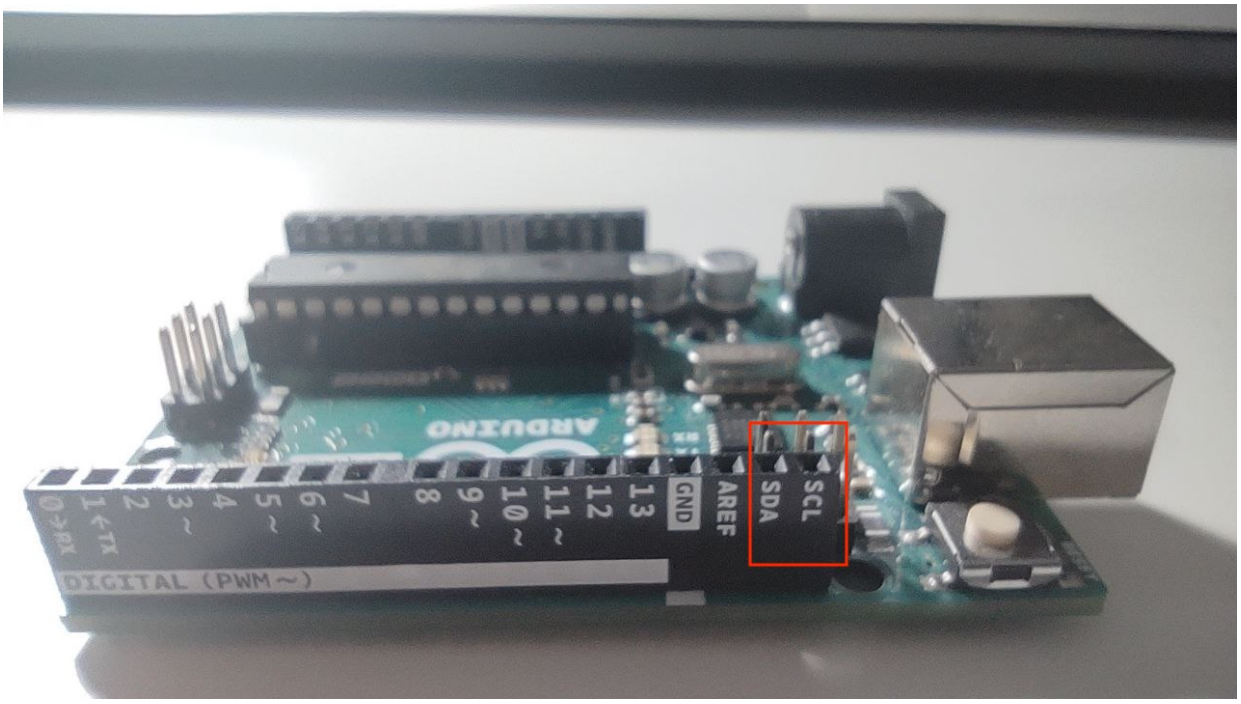
Los conversores ADC integran la medición de los fotodiodos, que es transferida a los registros internos del TCS34725. La conexión de los módulos que integran el TCS34725 es sencilla, ya que la comunicación se realiza a través de I2C.

Comunicación I2C

Aunque no vamos a entrar a explicar cómo funciona la comunicación **I2C** (inter integrated circuits), debemos saber que se trata de un protocolo basado en la existencia de dos líneas de comunicación, una de ellas lleva la **señal de reloj** y es conocida como (**SCL**), y la otra **linea** lleva los **datos** y es conocida como (**SDA**).

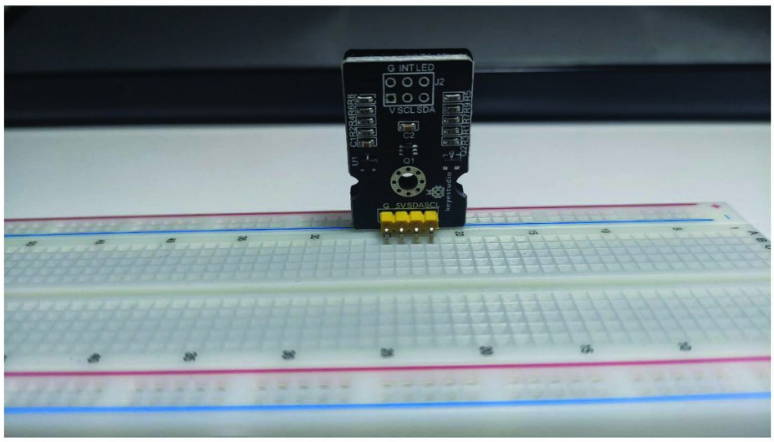
Los Pines **SDA** y **SLC** se encuentran especificados en todos los componentes que usan este tipo de protocolo de comunicación, como podemos ver si volvemos a la primera foto de arriba.

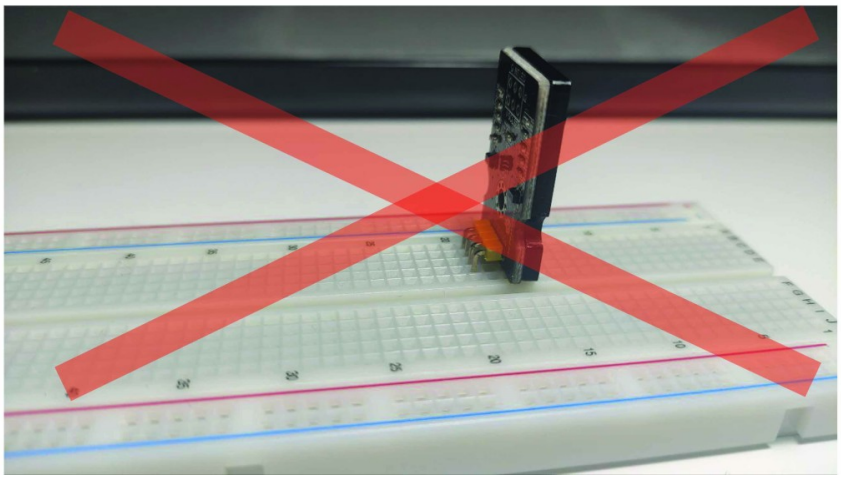
Por tanto, para hacer funcionar nuestro sensor, alimentamos nuestro sensor de color desde Arduino mediante los pines G y 5V y conectamos el pin SDA y SCL del sensor a los 2 pines de nuestro UNO SCL y SDA:



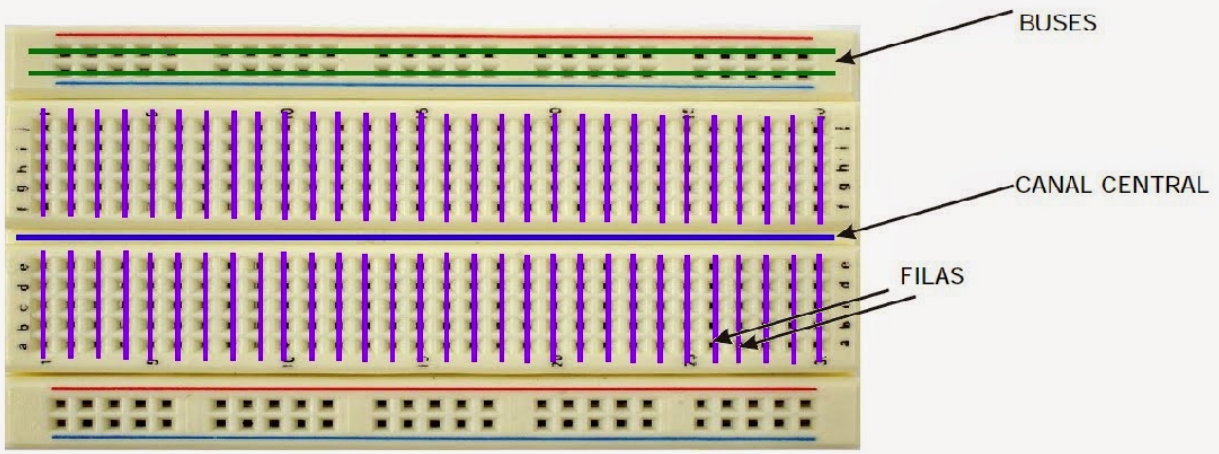
¿Cómo conectamos el sensor a nuestro Arduino?

Para conectarlo, necesitaremos la protoboard, como ya hemos dicho al principio. Es muy importante realizar las conexiones como te indico en la imagen de abajo:





Nuestro sensor ha de estar paralelo a las líneas azul y rojo, no perpendicular. ¿Por qué? Porque la protoboard internamente está conectada de la siguiente forma:



Copyright 2026

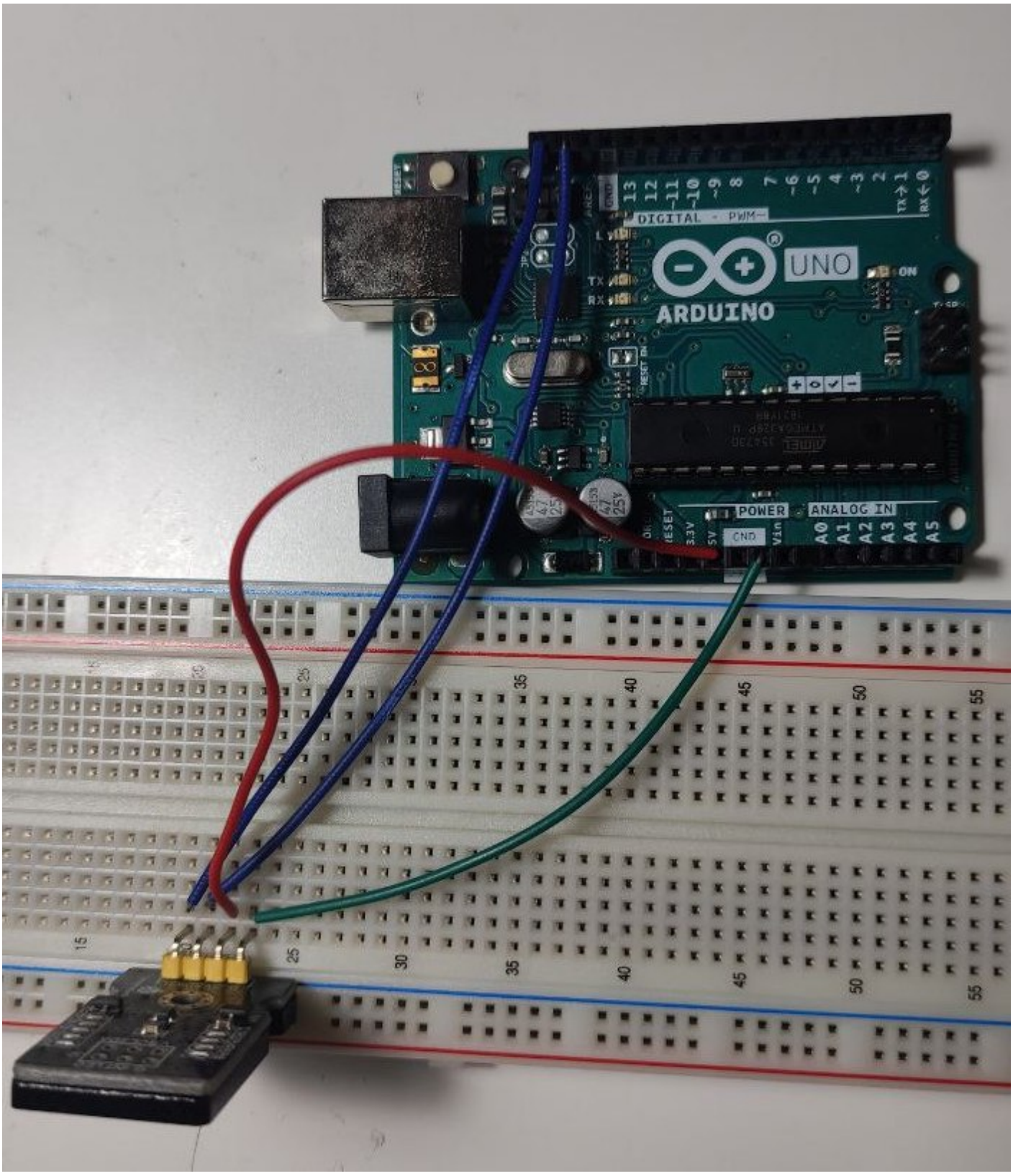
Si conectamos nuestro sensor con sus pines perpendiculare a las líneas azuly rojo, lo que estamos haciendo es cortociruitar el sensor y conectar todos los pines entre sí, lo que provocará que lo rompamos y probablemente salga humo... cosa que, obviamente, queremos evitar.

Colocaremos nuestro sensor de la manera correcta y posteriormente usaremos 4 cables para conectar cada uno de los pines de la protoboard con nuestro Arduino UNO:

El pin SCL con SCL
El pin SDA con SDA
El pin G con el GND
El pin 5V con el 5V

Quedando de la siguiente manera:

Copyright 2026

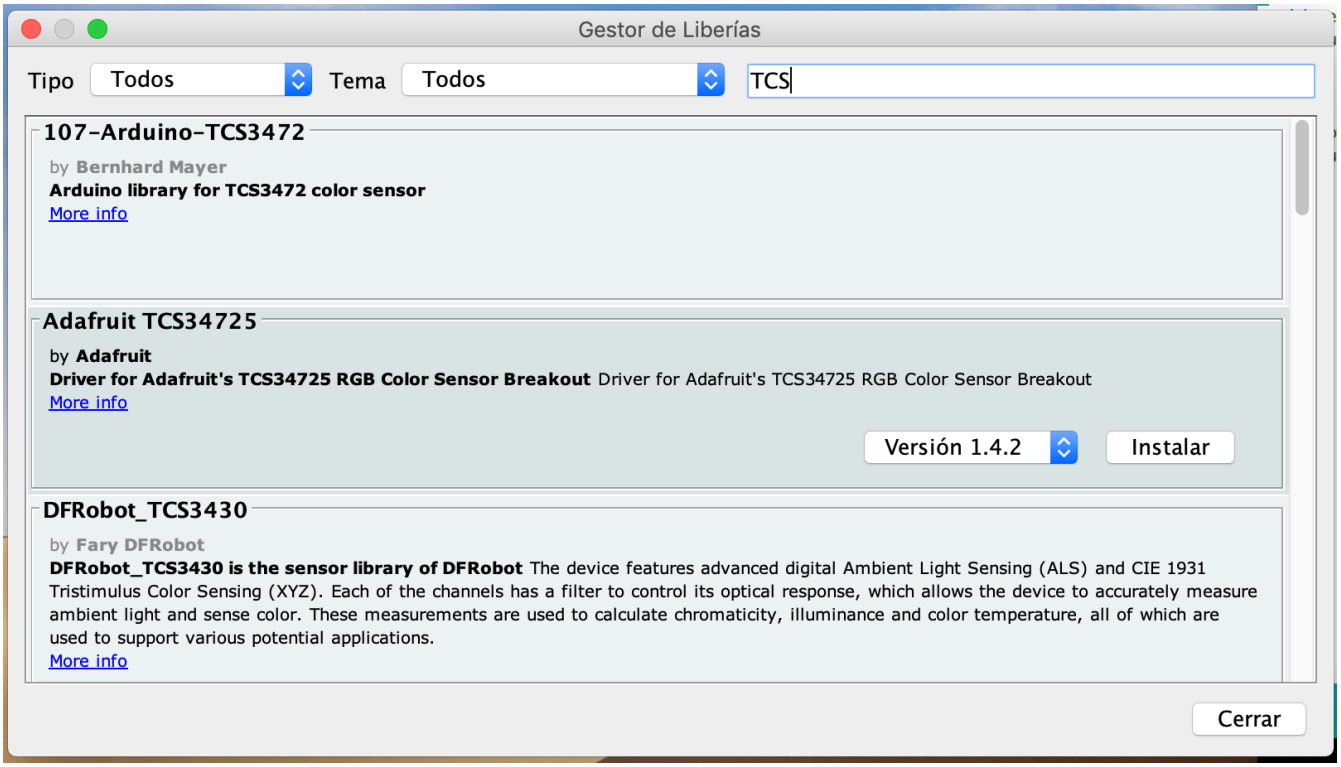


Ahora, pasaremos a ver el código necesario para identificar colores dependiendo de la cantidad de rojo, verde y azul que detecta el sensor.

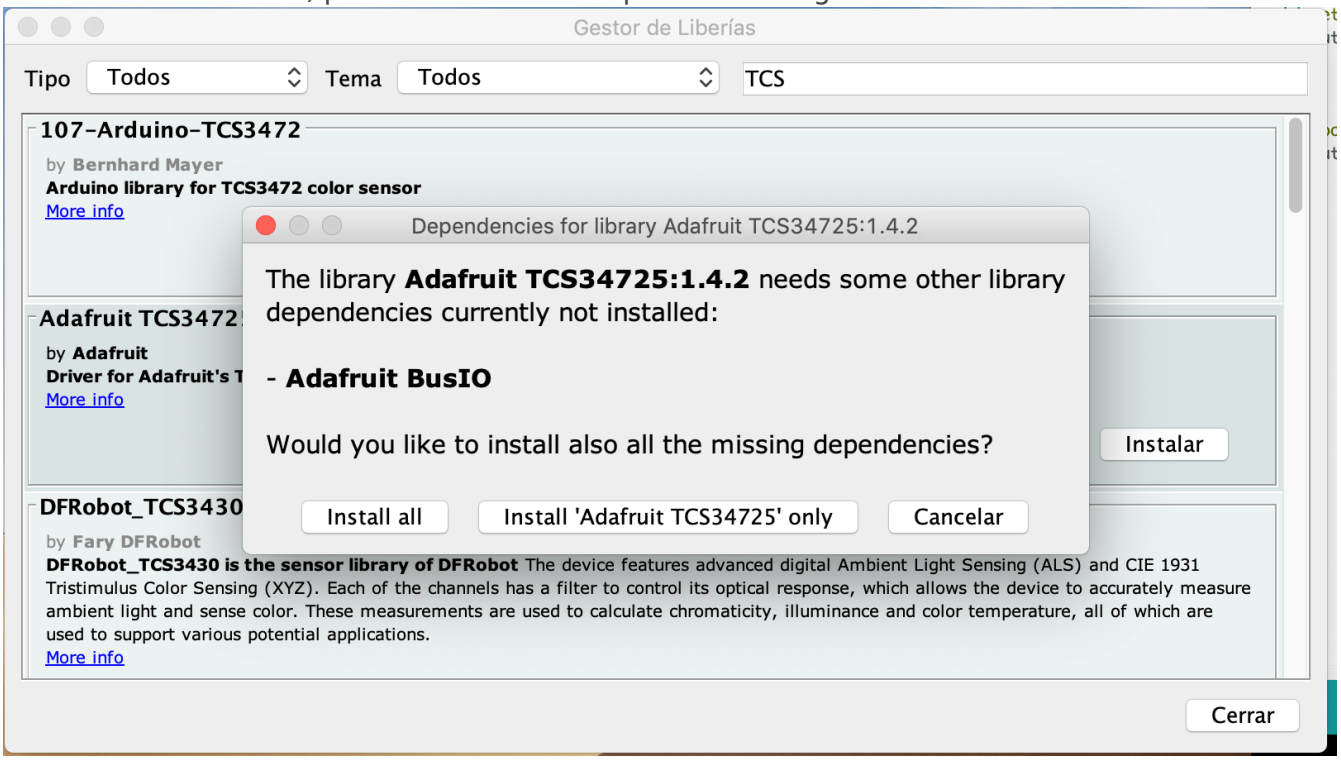
Copyright 2026

Identificamos colores

Lo primero que haremos será descargar una librería, siguiendo el procedimiento que ya realizamos cuando instalamos la librería PulseSensor Playground. En este caso, buscaremos en el gestor de librerías de nuestro Arduino una llamada **Adafruit TCS34725**:



Cuando la instalemos, probablemente nos aparecerá el siguiente aviso:



Haremos clic sobre el botón **Install All** y así ya tendremos instaladas todas las librerías necesarias para que funcione el código siguiente:

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"

/* Initialise with default values (int time = 2.4ms, gain = 1x) */
 Adafruit_TCS34725 tcs = Adafruit_TCS34725();



void setup(void) {
  Serial.begin(9600);

  if (!tcs.begin())
  {
    Serial.println("Error al iniciar TCS34725");
    while (1);
  }

}
```

```
void loop(void) {
  uint16_t r, g, b, c, colorTemp, lux;

  tcs.getRawData(&r, &g, &b, &c);
  colorTemp = tcs.calculateColorTemperature(r, g, b);
  lux = tcs.calculateLux(r, g, b);

  Serial.print("Temperatura color: "); Serial.print(colorTemp, DEC); Serial.println(" K");
  Serial.print("Lux : "); Serial.println(lux, DEC);
  Serial.print("Rojo: "); Serial.println(r, DEC);
  Serial.print("Verde: "); Serial.println(g, DEC);
  Serial.print("Azul: "); Serial.println(b, DEC);
  Serial.print("Clear: "); Serial.println(c, DEC);
  Serial.println(" ");
  delay(1000);
}
```

Este código está basado en el ejemplo TCS34725, con algunas modificaciones. El original puedes encontrarlo en la IDE de Arduino en **Archivo > Ejemplos > Adafruit TCS34725 > tcs34725**.

Desmenuzando el código

Lo primero que nos encontramos es dos sentencias que comienzan con **#** seguido de la palabra **include.** Como ya hemos visto en prácticas anteriores, esto se utiliza para importar librerías. En este caso importaremos las dos librerías que acabamos de instalar.

```
#include <Wire.h>
#include "Adafruit_TCS34725.h"
```

A continuación, lo que hacemos es crear un objeto sensor de color, de la misma manera que hicimos cuando creamos un objeto para nuestro pulsómetro.

```
 /* Initialise with default values (int time = 2.4ms, gain = 1x) */
 Adafruit_TCS34725 tcs = Adafruit_TCS34725();
```

Copyright 2026

A continuación, encontramos la función **void setup**:

```
void setup() {
  Serial.begin(9600);

  if (!tcs.begin())
  {
    Serial.println("Error al iniciar TCS34725");
    while (1);
  }

}
```

En ella comenzamos la comunicación con el puerto serial. Lo siguiente que haremos es comprobar si nuestro Arduino reconoce al sensor. Si no lo reconoce, entraremos en un bucle con la línea **while (1) delay (1000);** y pasado 1 segundo, volveremos a comprobar si nuestro Arduino ha reconocido al sensor. Si hemos realizado las conexiones correctamente esta línea no llegará a ejecutarse.

Lo próximo que nos encontramos es la función **void loop**, que también la hemos visto en las anteriores prácticas.
En ella, vamos a crear las variables que almacenarán la cantidad de rojo (**r**), verde (**g**), azul (**b**), el valor que reciban los fotodiodos sin filtro (**c**), la temperatura de la luz (**colorTemp**) y la cantidad de luz (**lux**). Serán variables cuyo tamaño sea 16 bits y eso lo sabemos porque se utiliza la palabra **uint16_t**.

Seguidamente, encontramos tres funciones que se encargarán de recibir y procesar la información sobre el color que coloquemos delante del sensor. Con la primera **tcs.getRawData(&r, &g, &b, &c);**, recibimos los datos en bruto y con las dos sentencias siguientes, calculamos la temperatura del color **colorTemp = tcs.calculateColorTemperature(r, g, b)** haciendo uso de los valores RGB; y también la luz **lux = tcs.calculateLux(r, g, b);**.

Los siguientes 7 **Serial.print** imprimen por el puerto serial los valores obtenidos. Al final encontramos un delay de 1 segundo (1000 ms) que nos facilitará la lectura del sensor al darle tiempo para procesar la información.

```
void loop(void) {
  uint16_t r, g, b, c, colorTemp, lux;

  tcs.getRawData(&r, &g, &b, &c);
```

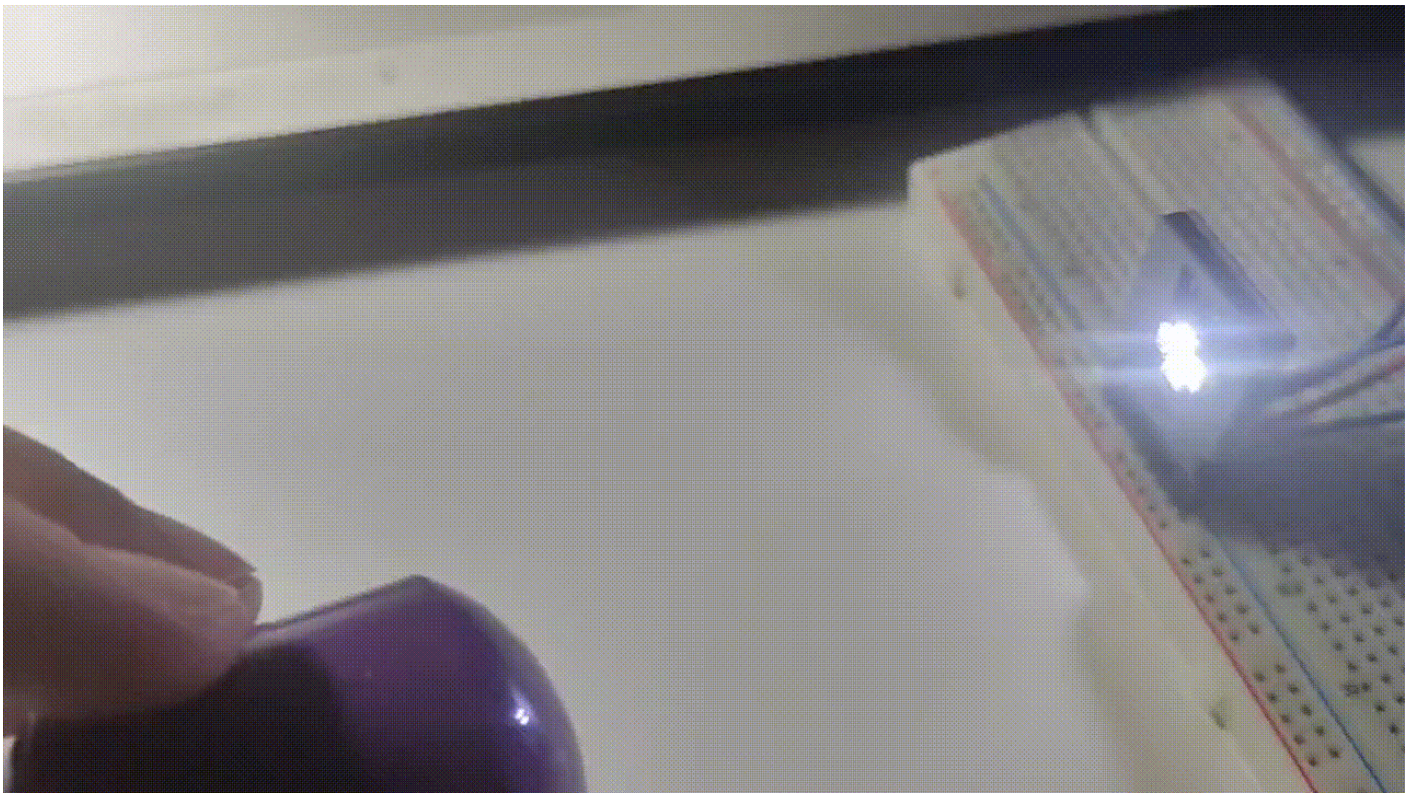
```
    colorTemp = tcs.calculateColorTemperature(r, g, b);
    lux = tcs.calculateLux(r, g, b);

    Serial.print("Temperatura color: "); Serial.print(colorTemp, DEC); Serial.println(" K");
    Serial.print("Lux : "); Serial.println(lux, DEC);
    Serial.print("Rojo: "); Serial.println(r, DEC);
    Serial.print("Verde: "); Serial.println(g, DEC);
    Serial.print("Azul: "); Serial.println(b, DEC);
    Serial.print("Clear: "); Serial.println(c, DEC);
    Serial.println(" ");
    delay(1000);
  }
```

Si ahora copiamos y pegamos este código en la IDE de nuestro Arduino y abrimos el puerto serial:



Cuando acerquemos un objeto de color y abramos el puerto serial nos encontraremos con algo parecido a esto:



Copyright 2026