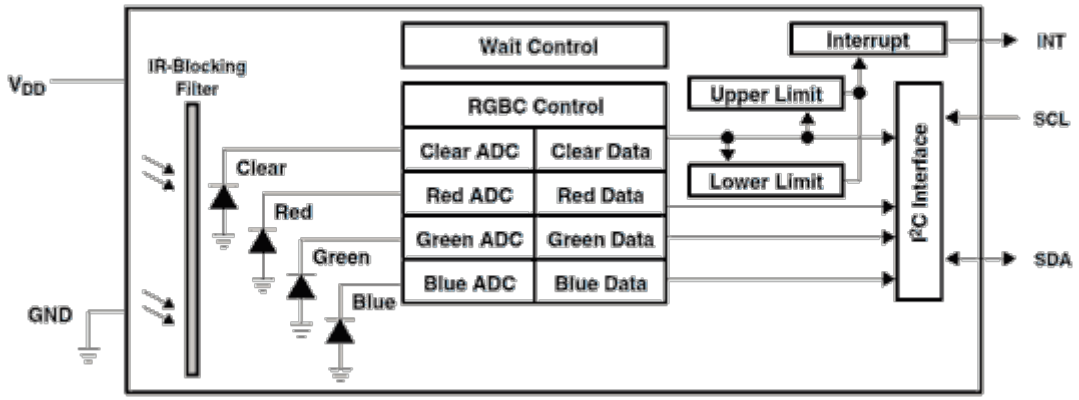
[I2C] Sensor de color TCS34725: Descripción y código para Arduino

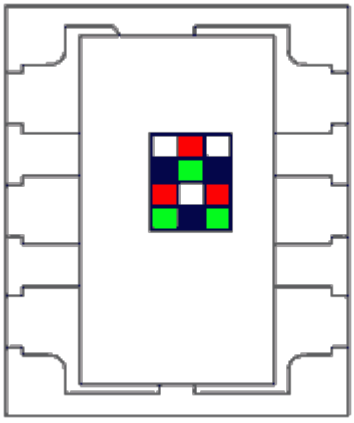
- TCS34725 Color Sensor (AMS)
- Data sheet

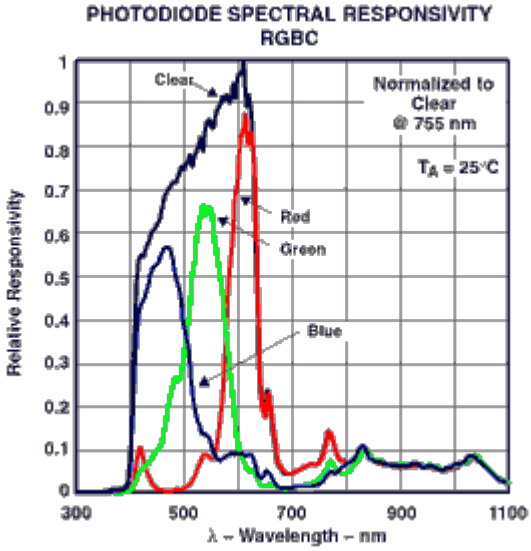
Chip de color TCS34725

- El TCS34725 es un sensor de color digital que podemos emplear con un procesador como Arduino para obtener los medir los valores RGB del color de un objeto o luz.
- El TCS34725 es un integrado completo que realiza un tratamiento digital de la medición de color, proporcionando los valores RGB y Clear (medición total sin filtrar).
- La comunicación con el sensor se realiza por I2C por lo que su lectura desde un procesador como Arduino es muy sencilla.
- Incorpora un filtro de infrarrojos, lo que mejora su precisión ante el ruido del entorno.
- El tiempo de medición y la ganancia es ajustable por software. Dispone de una amplia sensibilidad y un amplio rango dinámico de 3.800.000:1, pudiendo funcionar incluso tras un cristal oscuro.
- Podemos encontrarlo en módulos comerciales que incorporan un LED de luz neutra (4150°K) junto con un MOSFET integrado para su control, por lo que podemos controlar el encendido del LED desde el código.
- Además de la comunicación I2C, el TCS34725 incorpora un pin de interrupción junto con un umbral inferior y superior. Cuando el nivel de luz está fuera del rango de los umbrales, el TCS34725 genera una interrupción que permanece activa hasta que es reseteada por el controlador.
- A diferencia de otros sensores de color como el TCS3200, que únicamente son capaces de detectar colores básicos, el TCS34725 es capaz de proporcionar una medición RGB relativamente precisa del color medido.
- Por supuesto, como cualquier sensor el TCS34725 no es perfecto y tiene desviaciones típicas de cualquier sensor, por lo que no obtendremos una medida totalmente precisa. En general, será necesario calibrar el color.
- El TCS34725 es un sensor óptico que incorpora una matriz de 3×4 fotodiodos, junto con 4 conversores analógico digital de 16bits (ADC) que realizan la medición de los fotodiodos.



- La matriz de 3×4 está formada por fotodiodos filtrados para rojo, verde, azul, y sin filtro (clear). Todos los sensores están filtrados para infrarrojos.

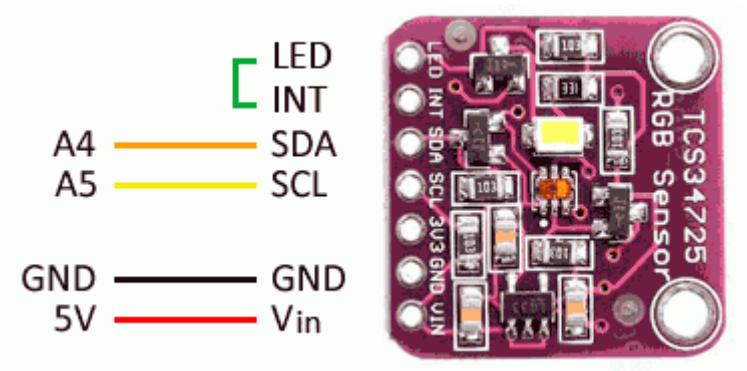




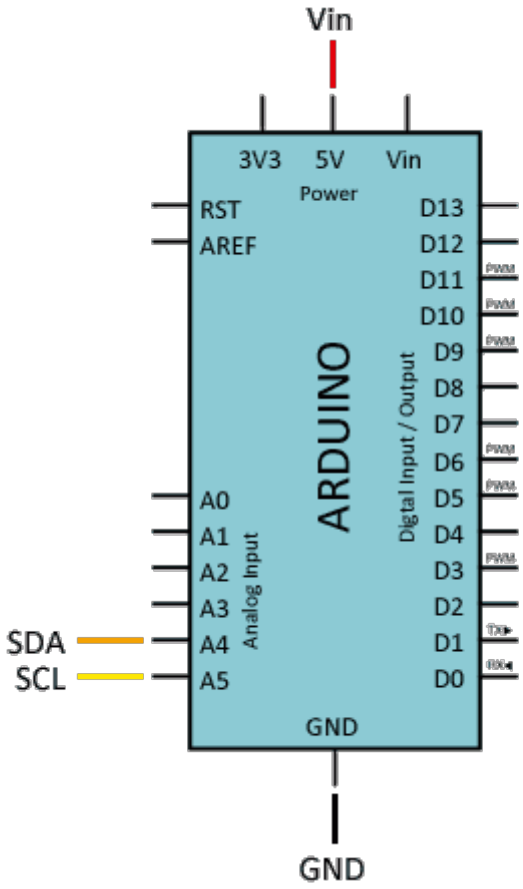
- Los conversores ADC integran la medición de los fotodiodos, que es transferida a los registros internos del TCS34725, que incorporan doble buffer para asegurar la integridad de os datos.
- El estado del sensor y el estado de energía del sensor está controlado por una máquina de estados interna, que controla todas las funciones del TCS34725.

Montaje y código para Arduino

- La conexión de los módulos que integran el TCS34725 es sencilla, ya que la comunicación se realiza a través de I2C.
- La tensión de alimentación del TCS34725 es de 3,3 V, pero normalmente los módulos comerciales integran una salida Vin que permite alimentar a 5 V. Por tanto, simplemente alimentamos el módulo desde Arduino mediante GND y Vin y conectamos el pin SDA y SCL de Arduino con los pines correspondientes del sensor.



- El pin LED controla el encendido del LED de luz neutra integrado en el módulo. La conexión de este pin puede ser la siguiente:
  - Dejar sin conectar para mantenerlo encendido continuamente.
  - Conectar a GND para apagar continuamente.
  - Conectar a un pin digital, para controlar su encendido con `digitalWrite()`.
  - Conectarlo al pin INT y controlar el encendido con `setInterrupt()` de la librería.
- Vista desde Arduino, la conexión sería la siguiente,



- **Código**
  - Para realizar la lectura del sensor TCS34725 usaremos la librería desarrollada por Adafruit, disponible en este enlace. La librería incorpora varios ejemplos de uso que conviene revisar.
  - A continuación vamos a ver algunos ejemplos de uso del sensor TCS34725.

Ejemplo 1: Leer valores RGB Ejemplo 2: Clasificar colores Ejemplo 3: Clonar color en tira WS2812B

Ejemplo 1

- En este primer ejemplo, muy sencillo, leemos los valores del sensor y los mostramos por puerto serie para su visualización.

tcs34725_rgb.ino

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

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

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS,
TCS34725_GAIN_1X);

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

  if (!tcs.begin())
  {
    Serial.println("Error al iniciar TCS34725");
    while (1) delay(1000);
  }
```

```
}

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);
}
```

Ejemplo 2

- En el segundo ejemplo, vamos a leer los valores RGB e intentar determinar el color que estamos leyendo. Para que sea más sencillo convertimos los valores RGB a HSV con la librería ColorConverter.
- Para determinar el color empleamos el Hue del color medido. Para un ajuste fino será necesario calibrar el sensor ensayando colores y ajustando los valores de los condicionales.

tcs34725_clasificar.ino

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

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_1X);

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


  if (!tcs.begin())
  {
    Serial.println("Error al iniciar TCS34725");
    while (1) delay(1000);
  }
}

void loop()
{
  uint16_t clear, red, green, blue;

  tcs.setInterrupt(false);
  delay(60); // Cuesta 50ms capturar el color
  tcs.getRawData(&red, &green, &blue, &clear);
  tcs.setInterrupt(true);
```

```
  // Hacer rgb medición relativa
  uint32_t sum = clear;
  float r, g, b;
  r = red; r /= sum;
  g = green; g /= sum;
  b = blue; b /= sum;

  // Escalar rgb a bytes
  r *= 256; g *= 256; b *= 256;

  // Convertir a hue, saturation, value
  double hue, saturation, value;
  RGBConverterLib::RgbToHsv(static_cast<uint8_t>(r), static_cast<uint8_t>(g),
static_cast<uint8_t>(b), hue, saturation, value);

  // Mostrar nombre de color
  printColorName(hue * 360);

  delay(1000);
}

void printColorName(double hue)
{
  if (hue < 15)
  {
    Serial.println("Rojo");
  }
  else if (hue < 45)
  {
    Serial.println("Naranja");
  }
  else if (hue < 90)
  {
    Serial.println("Amarillo");
  }
  else if (hue < 150)
  {
    Serial.println("Verde");
  }
  else if (hue < 210)
  {
    Serial.println("Cyan");
  }
  else if (hue < 270)
  {
    Serial.println("Azul");
  }
  else if (hue < 330)
  {
    Serial.println("Magenta");
  }
  else
  {
    Serial.println("Rojo");
  }
}
```

Ejemplo 3

- En el último ejemplo, vamos a usar una tira Led RGB WS2812b para “clonar” el color que leemos en el sensor de color.

- El ejemplo es muy simple, por un lado leemos el color con el TCS34725. Por otro, usamos la librería Fastled para hacer un efecto estela en la tira WS2812b. La tabla de conversión gamma adapta los colores leídos a la percepción de color que tiene el ojo humano.

tcs34725_clonar.ino

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

#define DATA_PIN    6
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB
#define BRIGHTNESS    250
#define FRAMES_PER_SECOND 250

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_1X);

const unsigned long INTERVAL = 2000;
const int NUM_LEDS = 16;
CRGB leds[NUM_LEDS];

byte gammatable[256];

float r, g, b;

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

  if (!tcs.begin())
  {
    Serial.println("Error al iniciar TCS34725");
    while (1) delay(1000);
  }

  delay(1000);
  FastLED.addLeds<LED_TYPE, DATA_PIN, COLOR_ORDER>(leds, NUM_LEDS);
  for (int i=0; i<256; i++)
  {
      float x = i;
      x /= 255;
      x = pow(x, 2.5);
      x *= 255;
      gammatable[i] = x;
  }
}


void loop()
{
  readColor();
  calculateLeds();
  FastLED.show();
  FastLED.delay(1000 / FRAMES_PER_SECOND);
```

```
}

void readColor()
{
  uint16_t clear, red, green, blue;

  tcs.setInterrupt(false);
  delay(60);
  tcs.getRawData(&red, &green, &blue, &clear);
  tcs.setInterrupt(true);

  uint32_t sum = clear;
  r = red; r /= sum;
  g = green; g /= sum;
  b = blue; b /= sum;

  r *= 256;
  g *= 256;
  b *= 256;
}

void calculateLeds()
{
  fadeToBlackBy(leds, NUM_LEDS, 20);

  uint8_t pixel;
  unsigned long tCurrent = millis();
  pixel = (tCurrent % INTERVAL) * NUM_LEDS / INTERVAL;
  leds[pixel] = CRGB(gammatable[(int)r], gammatable[(int)g], gammatable[(int)b]);
}
```

- Este es el resultado: https://youtu.be/RUASzylRWtk