Manual Técnico Sensor De Color Con Alerta De Estado

Aponte Einer Fabian

Bautista Jonathan Camilo

Quiroga Marlon Daniel

Sistemas Embebidos

Universidad ECCI

2019-1

1. INTRODUCCIÓN

En el presente manual se realizará una especificación a fondo de cada uno de los componentes utilizados a nivel de la programación para un sistema de detección de color, teniendo en cuenta que el programa se desarrollo vía web utilizando una

herramienta de programación compatible para la utilización de la tarjeta núcleo FE446RE, además de una herramienta para enviar y recibir comandos como lo es el programa Coolterm.

Inicialmente se desarrolla el programa en lenguaje c, orientada a objetos.

2. OBJETIVOS

a. OBJETIVO GENERAL

Implementar una especificación completa sobre el programa para que futuros técnicos puedan modificar el software.

b. OBJETIVOS ESPECIFICOS

- Instruir al técnico en el uso adecuado del software, así como comprenderlo en su totalidad.
- Describir los archivos relevantes del sistema los cuales orientan a la correcta configuración de este.
- Implementar el uso adecuado de las herramientas utilizadas para el correcto funcionamiento del software.

3. ALCANCE

El software desarrollado esta destinado a ser implementado en un sistema de detección de colores, con la posibilidad de que a cada color detectado por medio de un buzzer y con la configuración implementada tenga diferentes sonidos. El sistema esta limitado a detectar solo 4 colores, además al detectar un color diferente a los predeterminados debe devolver un mensaje de error o de color no reconocido, estos mensajes serán devueltos en comandos hexadecimales.

4. REQUERIMIENTOS

a. HARDWARE

Fuente de alimentación de 5V DC generalmente es tomada de la tarjeta núcleo FE446RE.

Tarjeta núcleo FE446RE

b. SOFTWARE

Como la programación se realiza mediante una plataforma online llamada MBED, los requerimientos para la aplicación son mínimos, se debe contar con un computador con una velocidad de trasferencia de datos como se ha especificado en el programa, en este caso debe ser de mínimo de 115200, además de contar con acceso a internet.

i. LENGUAJE DE PROGRAMACIÓN

Como se mencionó en la introducción el lenguaje utilizado para trabajar en la plataforma MBED es lenguaje C orientado a objetos.

ii. OTRAS TECNOLOGÍAS

Se necesita un programa secundario para el envió de comandos y recepción de datos provenientes del software implementado, en este caso se utiliza la herramienta Coolterm, esta herramienta nos que permite tener una terminal para nuestros puertos serie ya que tiene un menú de configuración muy completo, permitiendo elegir desde una lista los puertos disponibles y seleccionar su velocidad y demás parámetros; además despliega los datos recibidos tanto en ASCII como en hexadecimal.

c. INSTALACION y CONFIGURACIONES

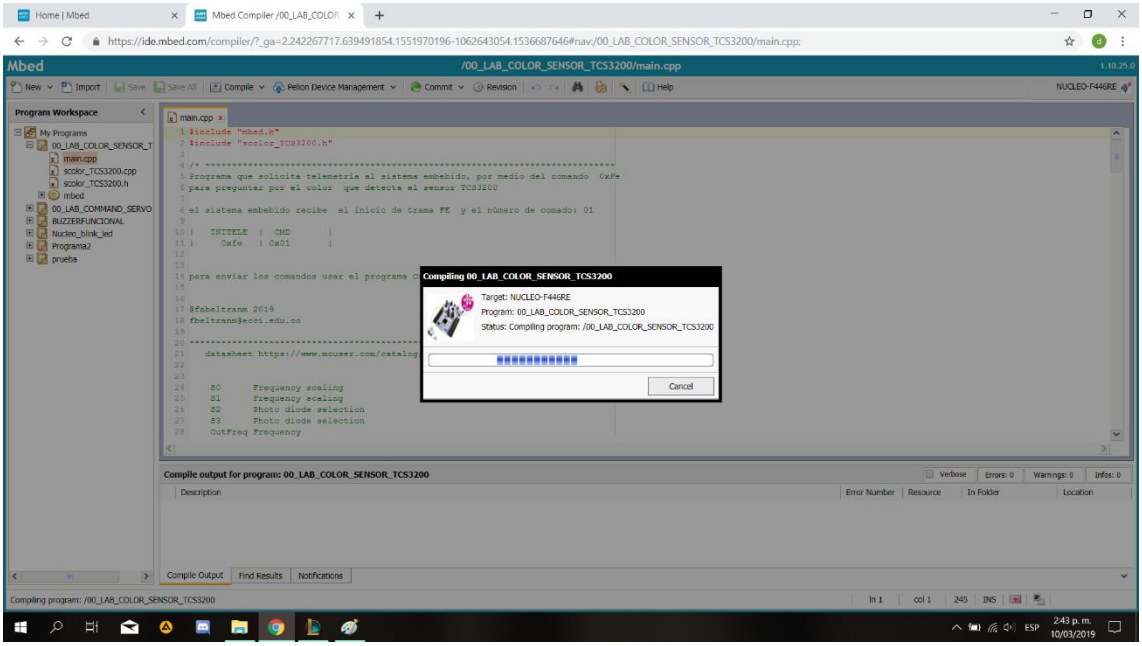
Para obtener una programación correcta de la tarjeta Núcleo FE446RE es importante saber como se realiza la programación mediante la plataforma MBED.

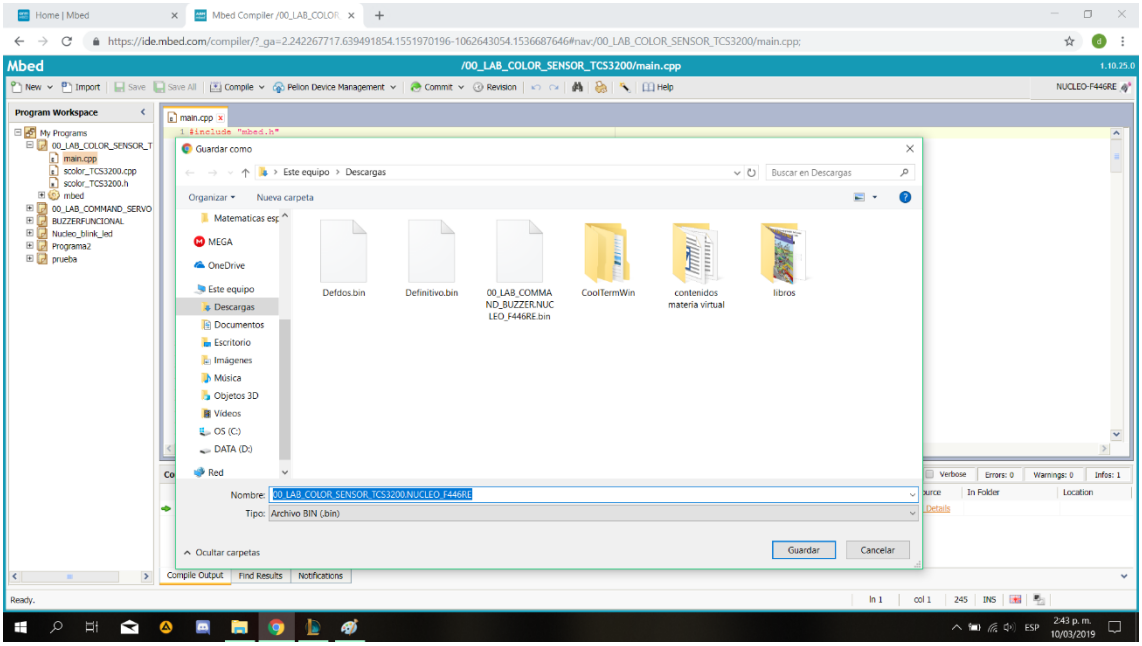
En primer lugar, se debe crear un usuario en la página principal de la plataforma MBED https://www.mbed.com/en/ el uso de la plataforma es muy sencillo y similar a otros programadores.

El código se encuentra disponible en la dirección https://os.mbed.com/users/Darstack/code/Primeraentrega/

Nos encontraremos con la opción de importar el código con todos sus archivos correspondientes, el código será guardado en la cuenta creada con el nombre que se le asigne.

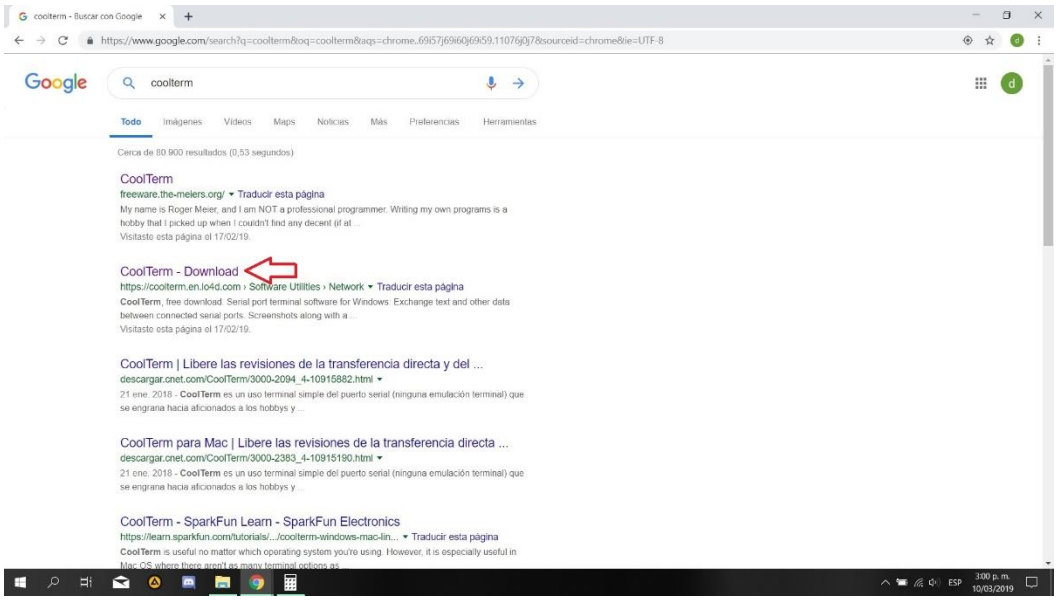
Después de haber importando el programa, se procede a realizar la compilación de este, como es una plataforma de programación online, se compila y luego se descarga un archivo que se guarda en una ruta especifica.

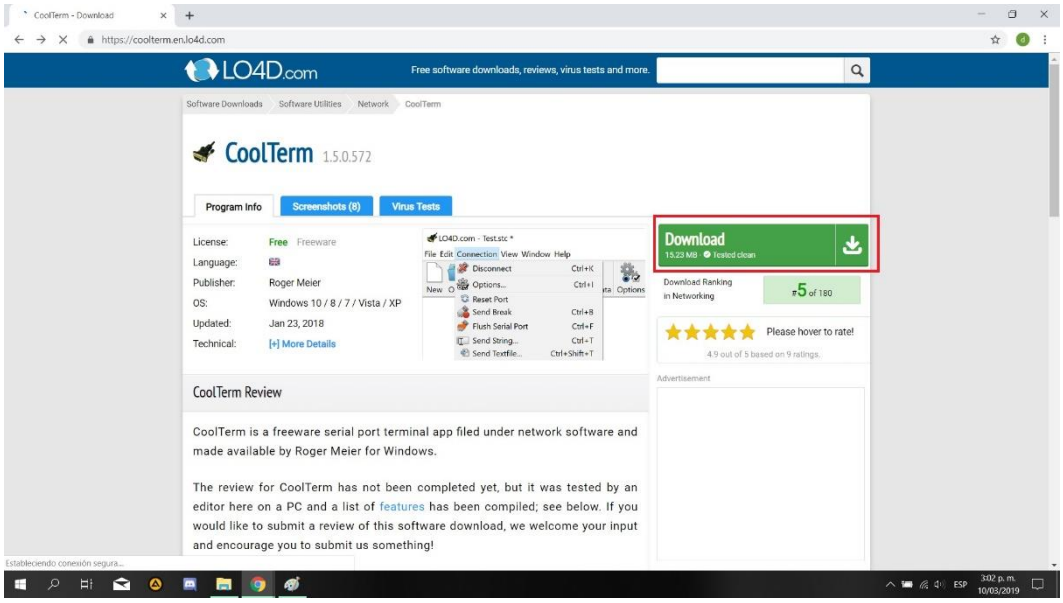


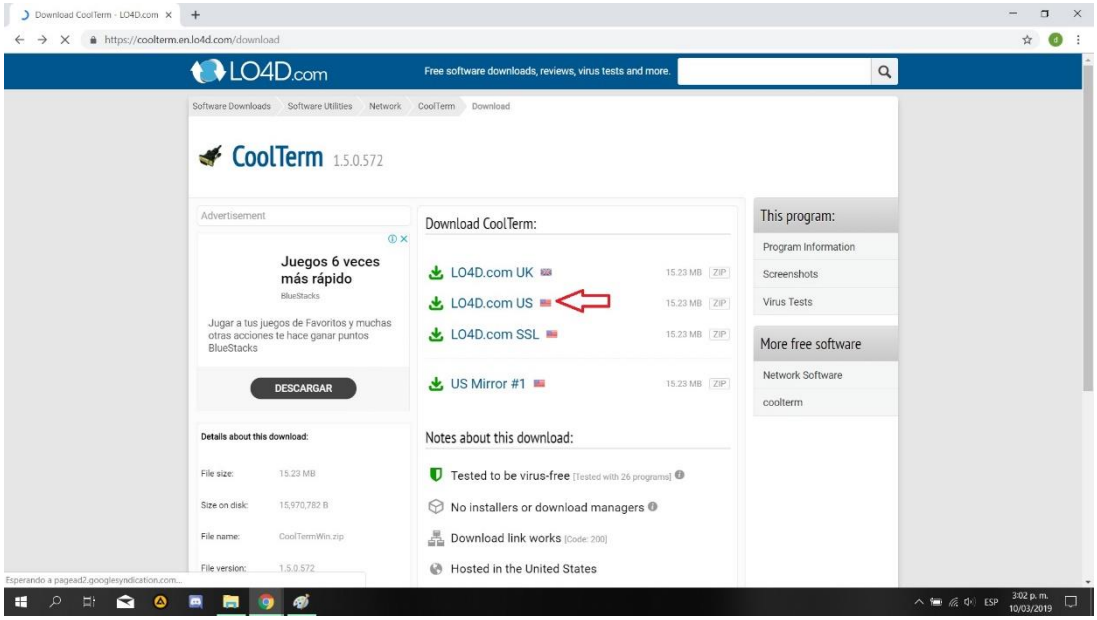


Como ultimo para realizar la programación de la tarjeta núcleo FE446RE lo que se debe hacer es conectar la tarjeta en uno de los puertos USB del computador, luego se procede a copiar el archivo previamente descargado lo copiamos en el almacenamiento de la tarjeta.

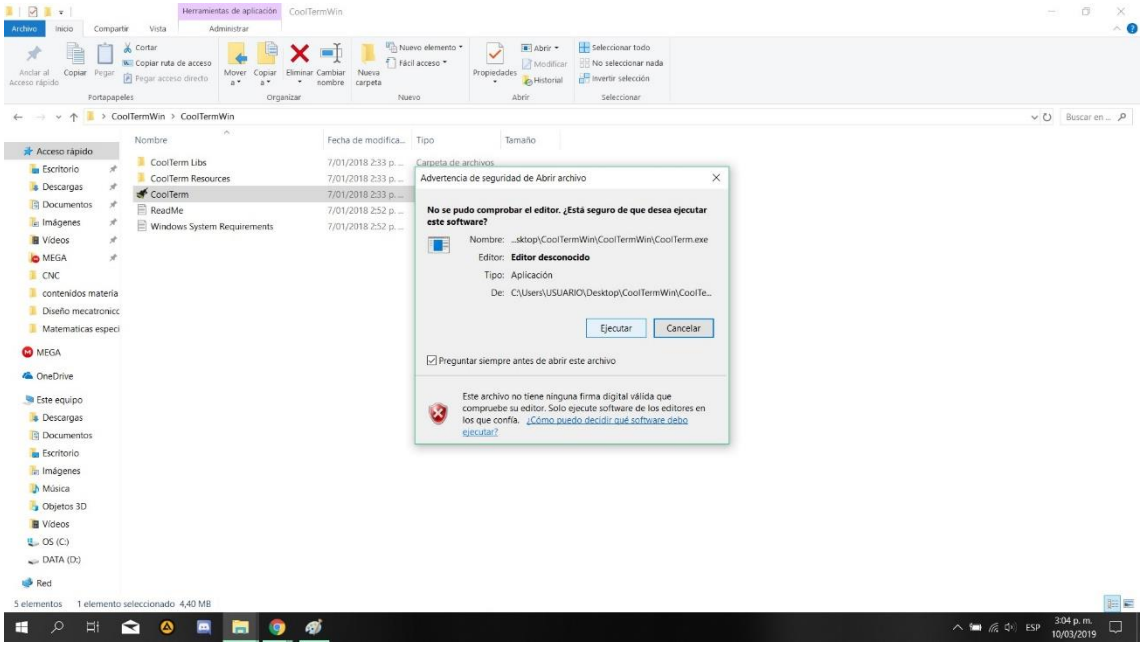
Para lograr una comunicación con el sistema y el programa como tal, se debe descargar la herramienta Coolterm para esto nos vamos a la pagina oficial y se realiza la descarga.


coolterm - Buscar con Google
https://www.google.com/search?q=coolterm&oq=coolterm&aqs=chrome..69i57j69i60j69i59.11076j0j7&sourceid=chrome&ie=UTF-8
Google
coolterm
Todo
Imágenes
Videos
Maps
Noticias
Más
Preferencias
Herramientas
Cerca de 80.900 resultados (0,53 segundos)
CoolTerm
freeware.the-meiers.org/ ▾ Traducir esta página
My name is Roger Meier, and I am NOT a professional programmer. Writing my own programs is a hobby that I picked up when I couldn't find any decent (if at ...
Visitaste esta página el 17/02/19.
CoolTerm - Download
https://coolterm.en.lo4d.com › Software Utilities › Network ▾ Traducir esta página
CoolTerm, free download. Serial port terminal software for Windows. Exchange text and other data between connected serial ports. Screenshots along with a ...
Visitaste esta página el 17/02/19.
CoolTerm | Libere las revisiones de la transferencia directa y del ...
descargar.cnet.com/CoolTerm/3000-2094_4-10915882.html ▾
21 ene. 2018 - CoolTerm es un uso terminal simple del puerto serial (ninguna emulación terminal) que se engrana hacia aficionados a los hobbys y ...
CoolTerm para Mac | Libere las revisiones de la transferencia directa ...
descargar.cnet.com/CoolTerm/3000-2383_4-10915190.html ▾
21 ene. 2018 - CoolTerm es un uso terminal simple del puerto serial (ninguna emulación terminal) que se engrana hacia aficionados a los hobbys y ...
CoolTerm - SparkFun Learn - SparkFun Electronics
https://learn.sparkfun.com/tutorials/.../coolterm-windows-mac-lin... ▾ Traducir esta página
CoolTerm is useful no matter which operating system you're using. However, it is especially useful in Mac OS where there aren't as many terminal options as ...
ESP
3:00 p. m.
10/03/2019


CoolTerm - Download
https://coolterm.en.lo4d.com
LO4D.com
Free software downloads, reviews, virus tests and more.
Software Downloads
Software Utilities
Network
CoolTerm
CoolTerm 1.5.0.572
Program Info
Screenshots (8)
Virus Tests
License:
Free
Freeware
Language:
Publisher:
Roger Meier
OS:
Windows 10 / 8 / 7 / Vista / XP
Updated:
Jan 23, 2018
Technical:
[+] More Details
Download
15.23 MB
Tested clean
Download Ranking in Networking
#5 of 180
Please hover to rate!
4.9 out of 5 based on 9 ratings.
Advertisement
CoolTerm Review
CoolTerm is a freeware serial port terminal app filed under network software and made available by Roger Meier for Windows.
The review for CoolTerm has not been completed yet, but it was tested by an editor here on a PC and a list of features has been compiled; see below. If you would like to submit a review of this software download, we welcome your input and encourage you to submit us something!
Estableciendo conexión segura...
ESP
3:02 p. m.
10/03/2019

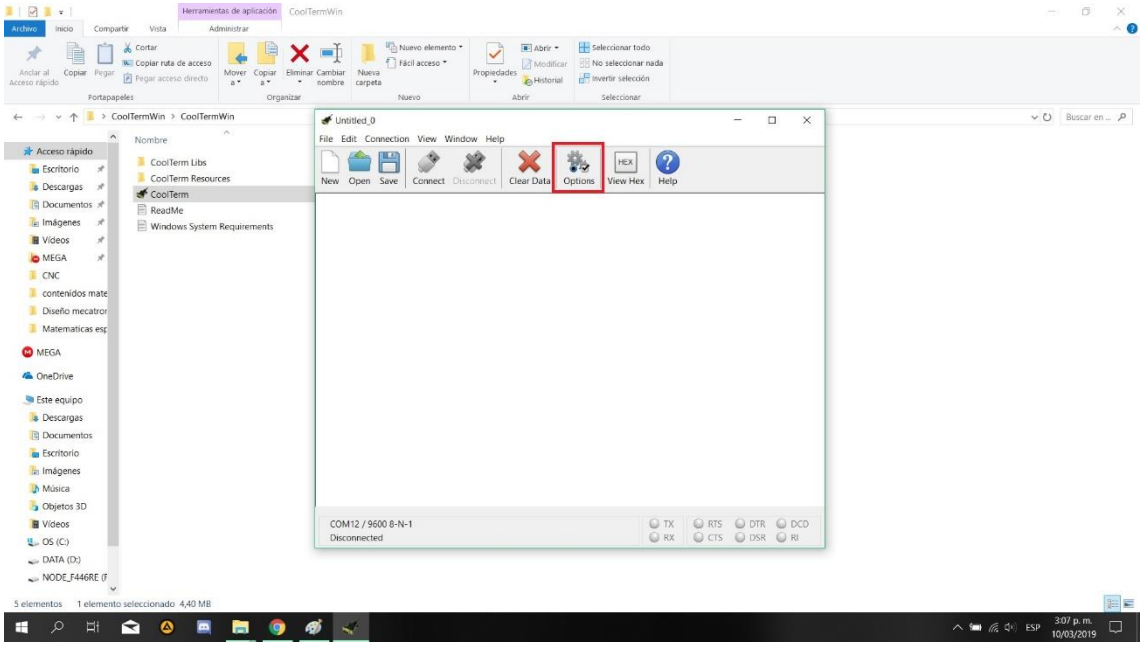

Download CoolTerm - LO4D.com
https://coolterm.en.lo4d.com/download
LO4D.com
Free software downloads, reviews, virus tests and more.
Software Downloads
Software Utilities
Network
CoolTerm
Download
CoolTerm 1.5.0.572
Advertisement
Juegos 6 veces más rápido
BlueStacks
Jugar a tus juegos de Favoritos y muchas otras acciones te hace ganar puntos BlueStacks
DESCARGAR
Details about this download:
File size: 15.23 MB
Size on disk: 15,970,782 B
File name: CoolTermWin.zip
File version: 1.5.0.572
Download CoolTerm:
LO4D.com UK
LO4D.com US
LO4D.com SSL
US Mirror #1
15.23 MB
ZIP
Notes about this download:
Tested to be virus-free [Tested with 26 programs]
No installers or download managers
Download link works [Code: 200]
Hosted in the United States
This program:
Program Information
Screenshots
Virus Tests
More free software
Network Software
coolterm
Esperando a pagead2.googlesyndication.com...
ESP
3:02 p. m.
10/03/2019

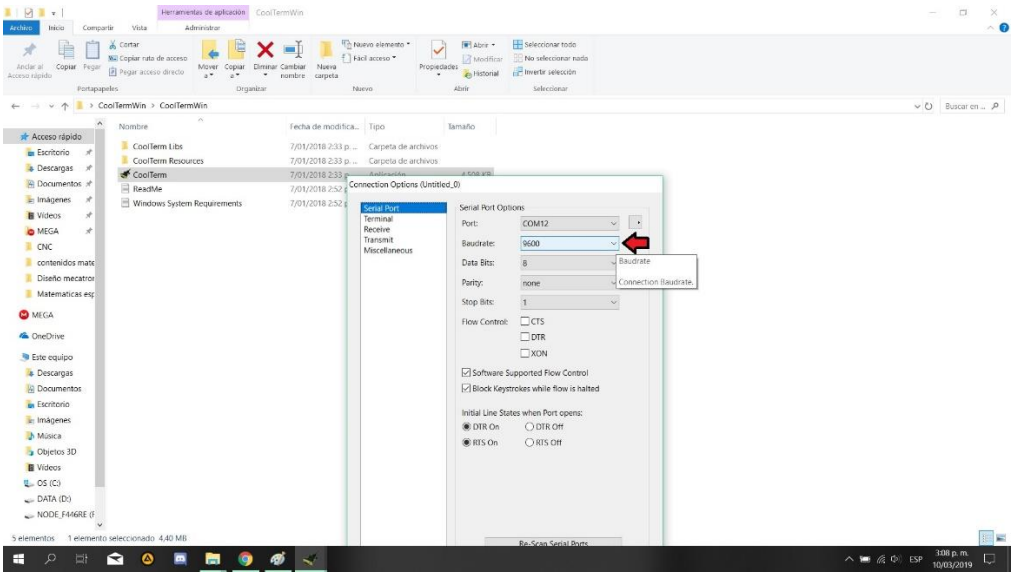
Al crear la carpeta y ubicarla en la ruta escogida a preferencia procedemos a ejecutar la herramienta.

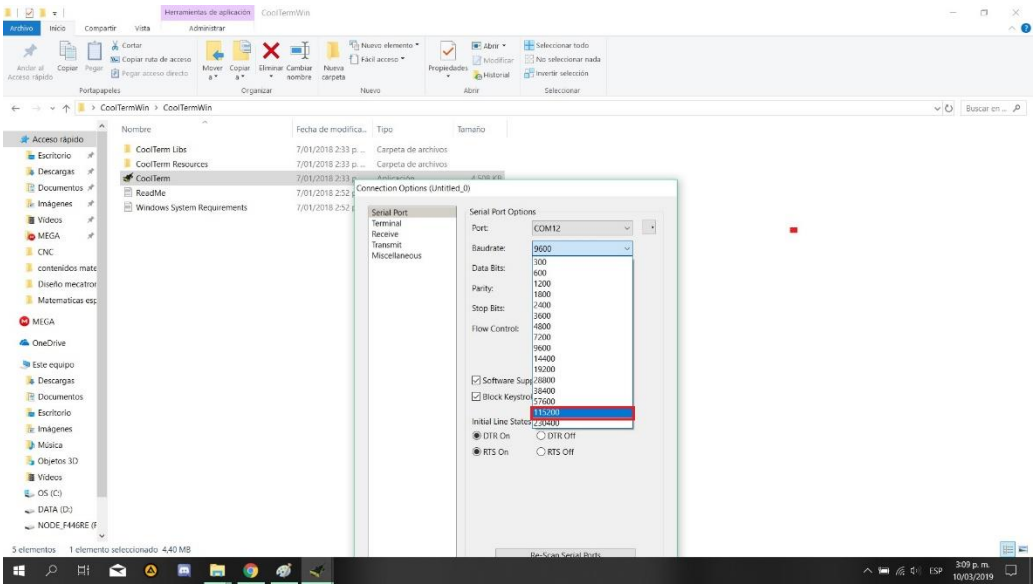


Ahora se procede a realizar la configuración del Coolterm para poder utilizarlo acorde a las especificaciones de nuestro programa. Es necesario realizar esta configuración cada vez que se vaya a probar el programa ya que las configuraciones no quedan guardadas, también es importante realizar la configuración y la comunicación con la tarjeta con esta conectada al computador.

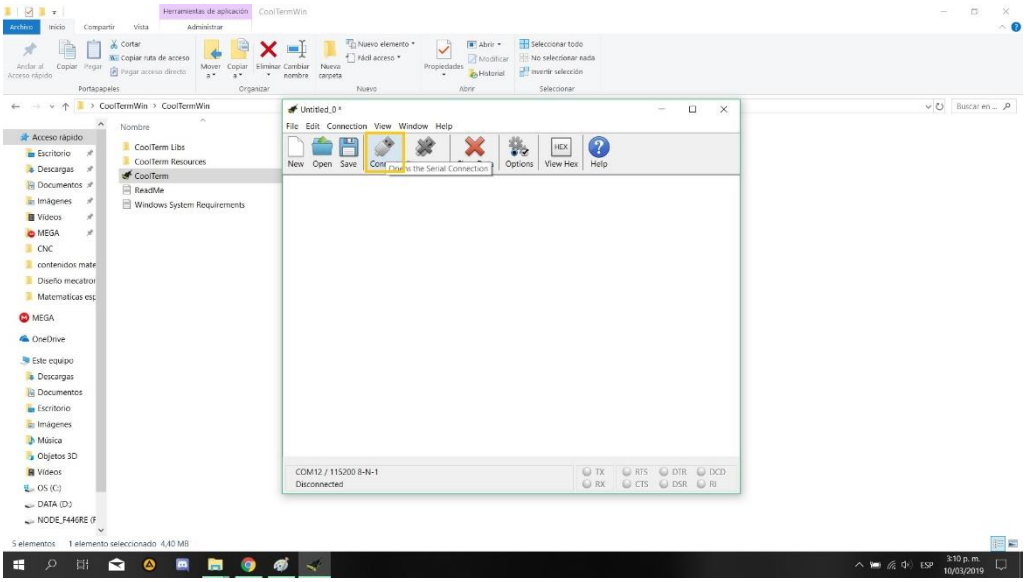
En primer lugar, configurar la velocidad de trasferencia de datos, se utiliza la velocidad especificada en el programa que en este caso son 115200, para esto vamos a option y luego damos click en baudrate.



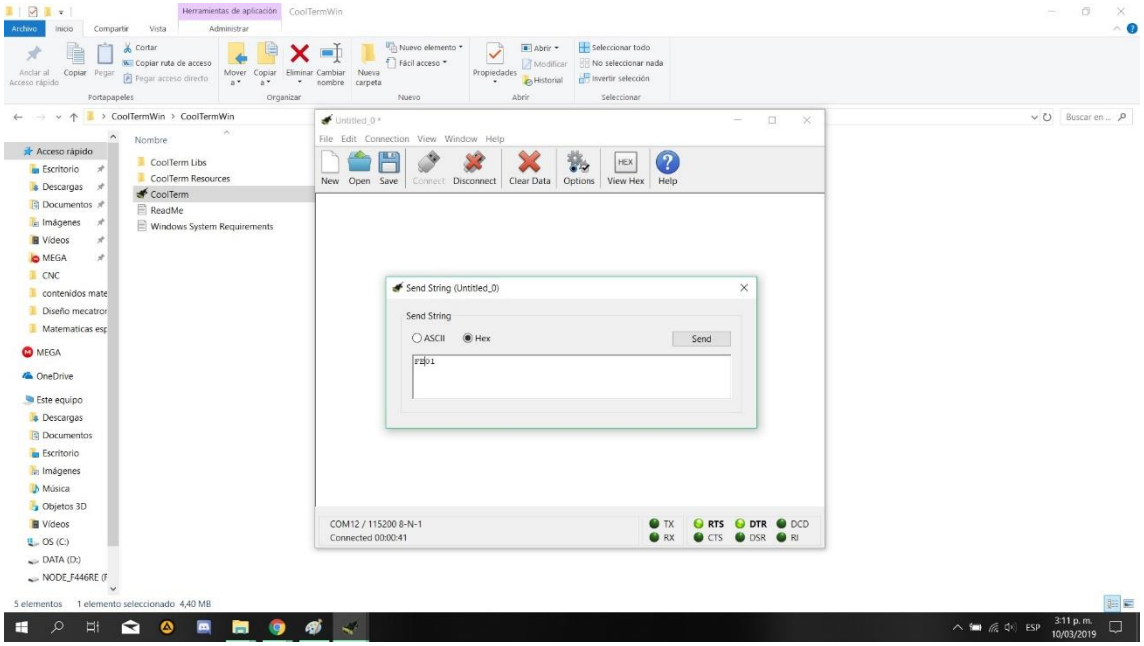




Luego de realizar los pasos anteriores se hace la conexión entre la interfaz y la tarjeta para esto solo damos click en la opción connect.



Para enviar los comandos para realizar la lectura de colores y verificar el funcionamiento correcto del programa se procede a oprimir las teclas ctrl + t, luego damos click en hex (hexadecimal), escribimos FE01 y le damos enviar.



5. PROGRAMA

Ahora se encuentra la descripción del programa como tal, como se menciono en las secciones anteriores el lenguaje de programación esta orientado a objetos y se utilizó C.

Al momento de realizar la importación del programa se generan tres archivos con código, el primero es el main.cpp que es el código principal, luego encontramos el archivo scolor_TCS3200.cpp y el archivo scolor_TCS3200.h, estos dos últimos son códigos para la configuración del sensor TCS3200.

- **scolor_TCS3200.h**

```
- #define SCALE_100   1
- #define SCALE_20    2
- #define SCALE_2     3
- #define POWER_DOWN  4
-
- class scolor_TCS3200 {
-   public:
-     scolor_TCS3200(PinName s0, PinName s1, PinName s2, PinName s3, Pin
    Name s_in);
-     long ReadRed();      // retorno el tiempo en alto de OutFreq para
    Rojo en ns
-     long ReadGreen();    // retorno el tiempo en alto de OutFreq para
    verde en ns
-     long ReadBlue();     // retorno el tiempo en alto de OutFreq color
     azul en ns
-     long ReadClear();    // retorno el tiempo en alto de OutFreq sin f
    iltro en ns
-     void SetMode(uint8_t mode);
-   private:
-     DigitalOut _s0;
-     DigitalOut _s1;
```

```
-     DigitalOut _s2;
-     DigitalOut _s3;
-     DigitalIn _s_in;
-     Timer timer;
-     long pulsewidth();
-
- };
```

En este primer archivo se define lo que es la clase que se utilizara para definir el sensor, se definen los atributos públicos como la lectura del censado que realiza el sensor para la detección de color, estos atributos entregan el periodo de la señal, de tal manera están definidos como variables long, también se les da nombre a lo que son los pines de entrada y salida del sensor, en la parte superior se definen las escalas de lectura, según el datasheet se configura la escala de lectura dependiendo de la configuración de los pines como se ve a continuación:

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

Como atributos privados se definen las entradas y salidas de nuestro sensor, se define el timer para la lectura del tiempo de la señal y se define la variable long pulsewidth que es la que almacenara el ancho de pulso.

- **scolor_TCS3200.cpp**

```
- #include "mbed.h"
- #include "scolor_TCS3200.h"
-
-
- scolor_TCS3200::scolor_TCS3200(PinName s0, PinName s1, PinName s2, Pin
    Name s3, PinName s_in) :
- _s0(s0), _s1(s1), _s2(s2), _s3(s3), _s_in(s_in)
- {
-     SetMode(SCALE_100);
-
- };
-
-
- long scolor_TCS3200::ReadRed()   { _s2=0;  _s3=0; return pulsewidth();
    }
- long scolor_TCS3200::ReadBlue()  { _s2=0;  _s3=1; return pulsewidth();
    }
- long scolor_TCS3200::ReadClear() { _s2=1;  _s3=0; return pulsewidth();
    }
- long scolor_TCS3200::ReadGreen() { _s2=1;  _s3=1; return pulsewidth();
    }
-
- void scolor_TCS3200::SetMode(uint8_t mode) {
-     switch (mode){
-         case SCALE_100:  _s0= 1; _s1=1; break;
-         case SCALE_20:   _s0=1 ; _s1=0; break;
-         case SCALE_2:    _s0=0 ; _s1=1; break;
-         case POWER_DOWN: _s0=0 ; _s1=0; break;
```

```
    }
};

long  scolor_TCS3200::pulsewidth() {
    while(!_s_in);
    timer.start();
    while(_s_in);
    timer.stop();
    float pulsewidth_v = timer.read_us();
    timer.reset();
    return pulsewidth_v;
};
```

En el código presente se toman trabajan con los atributos definidos en la clase anterior, como se puede observar se toman los pines nombrados en la seccion anterior, así como tambien se define la escala con la que trabajaremos el sensor.

Para obtener unos intervalos para la configuración de los colores a detectar, se debe configurar los pines S2 y S3 según como lo muestra el datasheet y como se ve a continuación:

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Esta configuración se aplica como se ve en el código y nos retornara el ancho de pulso de la señal, seguidamente se realiza el switch para la selección de la escala de medición, como ya se ha mencionado se esta trabajando al 100%, y se finaliza con la función para obtener el periodo de la señal, mientras en S_in se encuentre negado se activara el timer, cuando esta condición cambien el timer se detiene, se almacena la medición en micro segundos y se resetea el timer, retornando la medida almacenada.

- **Main.cpp**

```
#include "mbed.h"
#include "scolor_TCS3200.h"
#define INITELE 0xFE
#define RESPUESTA1 0xFF01
#define RESPUESTA2 0xFF02
#define RESPUESTA3 0xFF03
#define RESPUESTA4 0xFF04
#define RESPUESTA5 0xFF05
#define CMD  0x01
#define DO  1300
#define RE  1500
#define MI  1600
#define FA  1800
#define SO  2000


Serial command(USBTX, USBRX);
//                        S0,    S1,    S2,    S3,    OUT
scolor_TCS3200 scolor(PA_8, PB_10, PB_4, PB_5, PB_3);

// definición de las funciones
void setup_uart();
void leer_datos();
```

```
- void leer_color();
- uint8_t cmd;
- PwmOut mybuzzer(PA_5);
-
- int main() {
-
-     setup_uart();
-     while(1){
-         leer_datos();
-         if (cmd==CMD){}
-             leer_color();
-     }
- }
-
-
-
- void setup_uart(){
-     command.baud(115200);
- }
-
-
- void leer_datos(){
-     while(command.getc()!= INITELE);
-     cmd=command.getc();
- }
-
-
- void leer_color(){
-
-     mybuzzer.write(0);
-     long     red = scolor.ReadRed();
-     long     green = scolor.ReadGreen();
-     long     blue = scolor.ReadBlue();
-     long     clear = scolor.ReadClear();
-
-     long frqred;
-     long frqgreen;
-     long frqblue;
-     long frqclear;
-     int8_t sel_color;
-     printf("RED: %5d     GREEN: %5d     BLUE: %5d     CLEAR: %5d    \n
    ", red, green, blue, clear);
-
-     frqred = ( ( 1.0/red )* 1000.0 );
-     frqgreen = ( ( 1.0/green ) * 1000.0);
-     frqblue = ( (1.0/blue) *1000.0 );
-     frqclear = ( (1.0/clear) *1000.0 );
-     printf("RED: %5d     GREEN: %5d     BLUE: %5d     CLEAR: %5d    \n
    ", frqred, frqgreen, frqblue,frqclear);
-  ////////////////////////////////////////////////////////////////
    /////////////////
-  /*|||||||||||||||||Seleccionando los diferentes colores.|||||||
    |||||||||||||||*/
-  /*||||||||||||||||||||||||||||||Color rojo||||||||||||||||||||
    |||||||||||||||*/
-  ////////////////////////////////////////////////////////////////
    /////////////////
-     if ( frqred >= 30.0 and frqred <= 500.0) {
-
-         if( frqgreen >= 0.0 and frqgreen <= 20.0 ) {
-
-             if ( frqblue >= 5.0 and frqblue <= 29.0 ) {
```

```
-
-                 //printf ( "tiende a rojo \n" );
-                 mybuzzer.period_us(DO);
-                 mybuzzer.write(0.5);
-                 wait(5);
-                 mybuzzer.write(0);
-                 sel_color=1;
-
-             }
-         }
-     }
- ////////////////////////////////////////////////////////////////////
- /*°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°Color verde°°°°°°°°°°°°°°°°°°°°
- °°°°°°°°°°°°°°°°*/
- ////////////////////////////////////////////////////////////////////
-     if ( frqgreen >= 11.0 and frqgreen <= 21.0) {
-
-         if( frqred >= 8.0 and frqred <= 20.0 ) {
-
-             if ( frqblue >= 10.0 and frqblue <= 22.0 ) {
-
-             printf ( "tiende a verde  \n" );
-             mybuzzer.period_us(RE);
-             mybuzzer.write(0.5);
-             wait(5);
-             mybuzzer.write(0);
-             sel_color=2;
-
-             }
-         }
-     }
- ////////////////////////////////////////////////////////////////////
- /*°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°Color azul°°°°°°°°°°°°°°°°°°°°
- °°°°°°°°°°°°°°°°*/
-  ////////////////////////////////////////////////////////////////////
-     if ( frqblue >= 10.0 and frqblue <= 70.0) {
-
-         if( frqgreen >= 5.0 and frqgreen <= 26.0 ) {
-
-             if ( frqred >= 5.0 and frqred <= 25.0 ) {
-
-             //printf ( "tiende a azul \n" );
-             mybuzzer.period_us(MI);
-             mybuzzer.write(0.5);
-             wait(5);
-             mybuzzer.write(0);
-             sel_color=3;
-
-             }
-         }
-     }
- ////////////////////////////////////////////////////////////////////
- /*°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°°Color amarillo°°°°°°°°°°°°°°°°°°
- °°°°°°°°°°°°°°°°*/
-  ////////////////////////////////////////////////////////////////////
```

```
-     if ( frqblue >= 0.0 and frqblue <= 25.0 ) {
-
-         if( frqgreen >= 20.0 and frqgreen <= 40.0) {
-
-             if ( 20.0 >= frqred <= 46.0 ) {
-
-             printf ( "tiende a amarillo \n" );
-             mybuzzer.period_us(FA);
-             mybuzzer.write(0.5);
-             wait(5);
-             mybuzzer.write(0);
-             sel_color=4;
-
-             }
-         }
-     }
- ////////////////////////////////////////////////////////////////////
   ///////////////////
- /*°°°°°°°°°°°°°°°°°°°°°°°°°°°°°Color no found°°°°°°°°°°°°°°°°°°
   °°°°°°°°°°°°°°°*/
-  ////////////////////////////////////////////////////////////////////
   ///////////////////
-
-  switch ( sel_color ) {
-
-     case 1:
-     int32_t enviar = RESPUESTA1 ;
-     char txt [6] ;
-     printf ( txt, "%04X" , RESPUESTA1 );
-     break;
-     case 2:
-
-     int32_t enviar = RESPUESTA2 ;
-     char txt [6] ;
-     printf ( txt, "%04X" , RESPUESTA2 ) ;
-
-     break;
-
-     case 3:
-
-     int32_t enviar = RESPUESTA3;
-     char txt [6] ;
-     printf ( txt, "%04X" , RESPUESTA3 );
-
-     break;
-
-     case 4:
-
-     int32_t enviar = RESPUESTA4;
-     char txt [6] ;
-     printf ( txt, "%04X" , RESPUESTA4 );
-     break;
-
-     default:{
-
-     int32_t enviar = RESPUESTA4;
-     char txt [6] ;
-     printf ( txt, "%04X" , RESPUESTA4 );
-
-     }
- }
- }
```

Por ultimo se hace el desarrollo del código principal, agregando las librerias de MBED y llamando e incluyendo el código explicado anteriormente.

```
#define INITELE 0xFE
#define RESPUESTA1 0xFF01
#define RESPUESTA2 0xFF02
#define RESPUESTA3 0xFF03
#define RESPUESTA4 0xFF04
#define RESPUESTA5 0xFF05
#define CMD  0x01
#define DO  1300
#define RE  1500
#define MI  1600
#define FA  1800
#define SO  2000
```

Definimos luego los tipos de respuesta que tendrá el sistema, así como el comando que se debe enviar para ejecutar la lectura del color, luego definimos el periodo de diferentes sonidos que tendrá el sistema al detectar los cuatro colores seleccionados y el sonido que realizará al no reconocer un quinto color.

```
Serial command(USBTX, USBRX);
//                      S0,    S1,    S2,    S3,    OUT
scolor_TCS3200 scolor(PA_8, PB_10, PB_4, PB_5, PB_3);

// definición de las funciones
void setup_uart();
void leer_datos();
void leer_color();
uint8_t cmd;
PwmOut mybuzzer(PA_5);
```

Seguimos con el llamado al comando para la comunicación serial, recordemos que la comunicación será por medio de la UART, definimos los pines a los que iran conectados las entradas y salidas del sensor de color, así como la definición de las funciones, la definición del entero de 8 bits cmd y la asignación del pin del PWM para el buzzer.

```
int main() {

    setup_uart();
    while(1){
        leer_datos();
        if (cmd==CMD){}
            leer_color();
    }
}
```

Nuestro main esta conformado por el llamado a la función de la UART, la función leer_datos, se pregunta si la variable cmd que es la que se envia por medio de Coolterm, es la misma que definimos por defecto en la parte superior del programa, de ser esto correcto se procede a llamar la función leer_color, de lo contrario no se realizara ninguna lectura.

```
void setup_uart(){
    command.baud(115200);
}


void leer_datos(){
    while(command.getc()!= INITELE);
    cmd=command.getc();
}


void leer_color(){

    mybuzzer.write(0);
    long     red = scolor.ReadRed();
    long     green = scolor.ReadGreen();
    long     blue = scolor.ReadBlue();
    long     clear = scolor.ReadClear();

    long frqred;
    long frqgreen;
    long frqblue;
    long frqclear;
    int8_t sel_color;
    printf("RED: %5d     GREEN: %5d     BLUE: %5d     CLEAR: %5d    \n ", red, green, blue, clear);

    frqred = ( ( 1.0/red )* 1000.0 );
    frqgreen = ( ( 1.0/green ) * 1000.0);
    frqblue = ( (1.0/blue) *1000.0 );
    frqclear = ( (1.0/clear) *1000.0 );
    printf("RED: %5d     GREEN: %5d     BLUE: %5d     CLEAR: %5d    \n ", frqred, frqgreen, frqblue,frqclear);
```

La función setup_uart nos define la velocidad de trasnferencia de datos, como ya habíamos mencionado trabajamos con 115200.

La función leer_datos nos dice que si se hay algún dato listo para su lectura, de ser así ese dato es almacenado en la variable cmd.

Leer_color nos indica la puesta del pwm en cero, así como la asignación a las variables long de cada color, la lectura proveniente del código anterior, también definimos variables para almacenar la frecuencia de cada señal, que es calculada con la formula mostrada en el código. Imprimimos los valores de periodo y frecuencia almacenados.

```
////////////////////////////////////////////////////////////////////////////////
/*||||||||||||||||Seleccionando los diferentes colores.|||||||||||||||||||||||*/
/*||||||||||||||||||||||||||||||Color rojo||||||||||||||||||||||||||||||||||||*/
////////////////////////////////////////////////////////////////////////////////
    if ( frqred >= 30.0 and frqred <= 500.0) {

        if( frqgreen >= 0.0 and frqgreen <= 20.0 ) {

            if ( frqblue >= 5.0 and frqblue <= 29.0 ) {

                //printf ( "tiende a rojo \n" );
                mybuzzer.period_us(DO);
                mybuzzer.write(0.5);
                wait(5);
                mybuzzer.write(0);
                sel_color=1;

            }
        }
    }
```

Para la detección del color especificado se deben hacer varias lecturas para poder obtener un intervalo de frecuencia para cada lectura de color, el sensor lee azul, verde y rojo de un color en específico, así se pueden con intervalos y varias lecturas definir variables para un color en especial, en este caso los intervalos para los colores de lectura son para la detección del color rojo, mientras los valores de frecuencia se

mantengan en esos intervalos el color tiene a ser rojo, si es correcto el buzzer sonara con el periodo que definimos DO a un ciclo útil del 50% y definimos una variable llamada sel_color=1, se realiza el mismo procedimiento con los colores definidos cambiando el peridodo y la variable sel_color.

```
switch ( sel_color ) {

    case 1:

    int32_t enviar = RESPUESTA1 ;
    char txt [6] ;
    printf ( txt, "%04X" , RESPUESTA1 );

    break;

    case 2:

    int32_t enviar = RESPUESTA2 ;
    char txt [6] ;
    printf ( txt, "%04X" , RESPUESTA2 ) ;

    break;

    case 3:

    int32_t enviar = RESPUESTA3;
    char txt [6] ;
    printf ( txt, "%04X" , RESPUESTA3 );

    break;

    case 4:

    int32_t enviar = RESPUESTA4;
    char txt [6] ;
    printf ( txt, "%04X" , RESPUESTA4 );
    break;

    default:{

    int32_t enviar = RESPUESTA4;
    char txt [6] ;
    printf ( txt, "%04X" , RESPUESTA4 );

    }
```

Finalizamos con un switch que funcionara acorde a la variable Selec_color que definimos para cada color detectado, todo esto para poder enviar al Coolterm respuestas Hexadecimales y no las respuestas comunes como el color leído.

FEx01 Rojo

FEx02 Verde

FEx03 Azul

FEx04 Amarillo

FEX05 Color no found