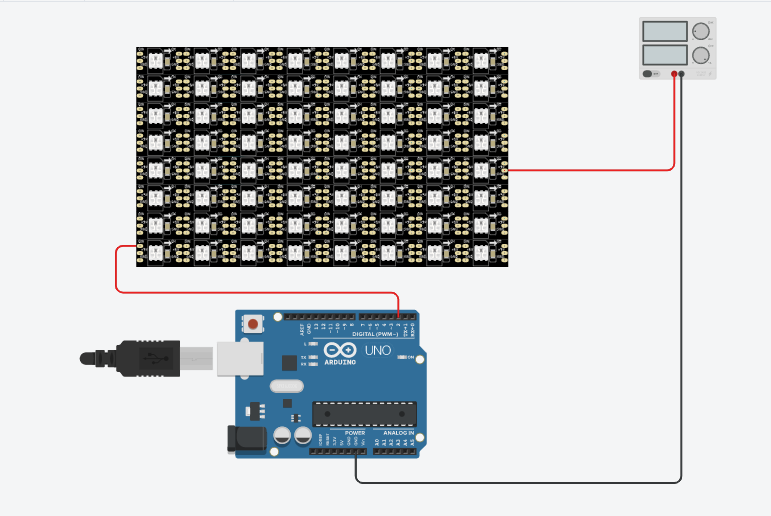
**17/09/2021**

Se hace la lectura del documento y se comienza con el prototipado del circuito

<https://www.tinkercad.com/things/30irhVGsUJx-exquisite-vihelmo/editel?sharecode=QeH9-G-r-W29xbARQEft_GBHeOOqjtv8OLDDFRzYSH8>



**18/09/2021**

Se actualiza un cable que faltaba del circuito

Diagrama

Descripción generada automáticamente

Inicio de la codificación

#include <Adafruit\_NeoPixel.h>

const int pinDatos=2;

const int numPixeles=8\*8;

void setup(){

  //Ya tenemos el serial para recibir la información del imagen de

  //la bandera desde Qt

  Serial.begin(9600);

  pinMode(2,OUTPUT);

}

void loop()

{

  if (Serial.available()>0){

  //Se Establece el menu para el usuario

  Serial.print("BIENVENIDO");

  Serial.print("Ingrese los datos de la bandera");

    //Falta determinar como se orgnizará esta información

  }

**19/09/2021**

Se hace la conexión de la parte trasera de la placa. Se le envía el voltaje y la GND a cada uno de los LEDs. El circuito se ve igual, pero con la segunda foto acercada se ven las conexiones.

Diagrama

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente con confianza media

IDEACIÓN PARA EL INFORME INICIAL Y PRUEBAS DEL CIRCUITO

#include <Adafruit\_NeoPixel.h>

const int pinDatos=2;

const int numPixeles=8\*8;

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

Serial.begin(9600);

pinMode(2,OUTPUT);

}

int main(){

for(int i =0; i<64; i++){

digitalWrite(2,1);

}

}

void loop()

{

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Ingrese los datos de la bandera");

//Falta determinar como se orgnizará esta información

}

}

**20/09/2021**

Actualización de las conexiones del circuito. Se conectaron los voltajes y tierras en serie a los inputs de las tiras de LEDs. Además, se cambió la conexión de los DIN para que queden desde cada output de una tira hacia el input de la siguiente.

Interfaz de usuario gráfica

Descripción generada automáticamente

Actualización del código de prueba. Se intenta probar el encendido de los LEDs pero se tiene que la tira de la parte superior no funciona aún. Se intentó con un color elegido aleatoriamente.

Interfaz de usuario gráfica

Descripción generada automáticamente

CÓDIGO

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 64

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

//Serial.begin(9600);

//pinMode(2,OUTPUT);

leds.begin();

for( int i = 0; i<LED\_COUNT; i++){

leds.setPixelColor(i, 20, 241, 55);

};

leds.show();

}

void loop()

{

/\*

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Ingrese los datos de la bandera");

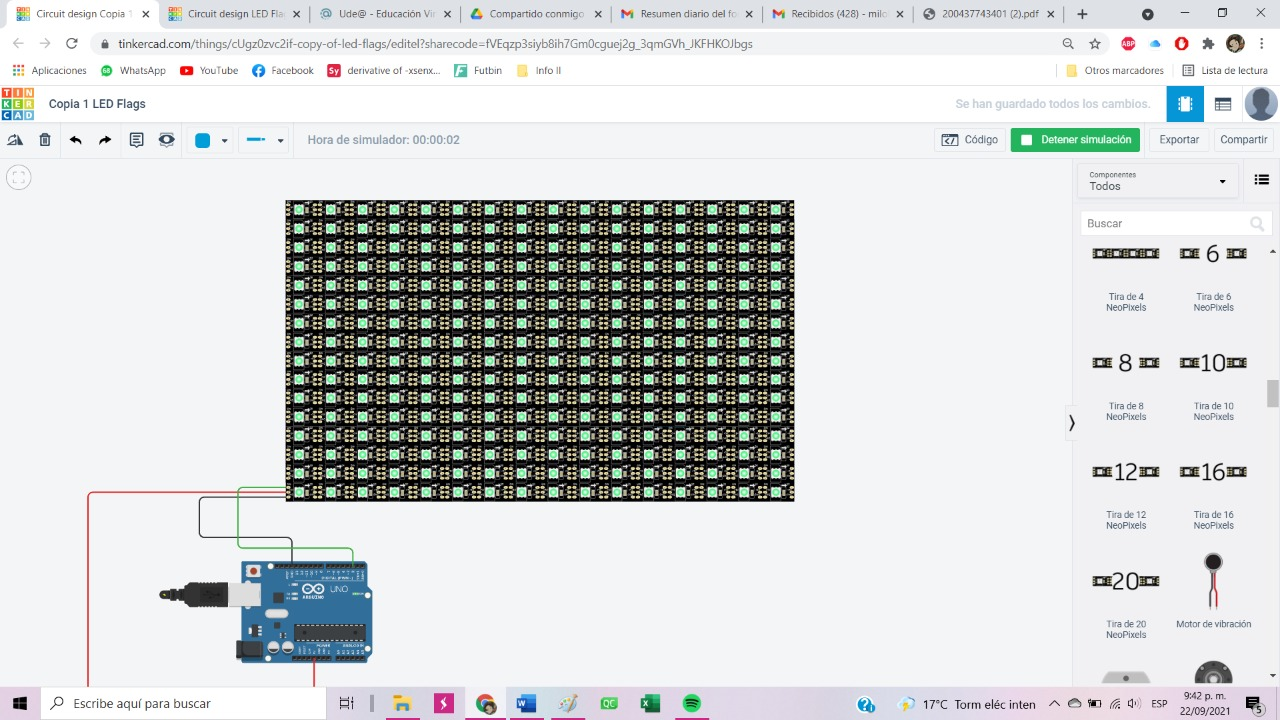
//Falta determinar como se orgnizará esta información

\*/

}

**22/09/2021**

Actualización y ampliación de la matriz



Actualización con la matriz de 16x16

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

//Serial.begin(9600);

//pinMode(2,OUTPUT);

leds.begin();

for( int i = 0; i<LED\_COUNT; i++){

leds.setPixelColor(i, 20, 241, 55);

};

leds.show();

}

void loop()

{

/\*

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Ingrese los datos de la bandera");

//Falta determinar como se organizará esta información

\*/

}

**23/09/2021**

Se comienza con la organización de la información para Arduino.

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

//Serial.begin(9600);

//pinMode(2,OUTPUT);

leds.begin();

for( int i = 0; i<LED\_COUNT; i++){

leds.setPixelColor(i, 20, 241, 55);

};

leds.show();

}

void loop()

{

/\*

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Ingrese los datos de la bandera");

//Falta determinar como se organizará esta información

\*/

}

void salida(\*\*\*int arreglo){

int NumLed = 255;

int red =0;

int green =0;

int blue =0;

for(int a=0;a<16;a++){

for(int b=0;b<16;b++){

for(int c=0;c<3;c++){

red = \*(\*(\*(arreglo+a)+b)+0);

green = \*(\*(\*(arreglo+a)+b)+1);

blue = \*(\*(\*(arreglo+a)+b)+2);

leds.setPixelColor(NumLed, red, green, blue);

}

NumLed--;

}

}

}

COMIENZO DE PRUEBAS CON ARREGLOS ALEATORIOS

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

//Definimos el arreglo que se pega de Qt

int arr[16][16][3]=

{

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

};

int\*\*\* arreglo = arr;

void salida(\*\*\*int arreglo){

int NumLed = 0;

int red =0;

int green =0;

int blue =0;

for(int a=15;a=>0;a--){

for(int b=0;b<16;b++){

for(int c=0;c<3;c++){

red = \*(\*(\*(arreglo+a)+b)+0);

green = \*(\*(\*(arreglo+a)+b)+1);

blue = \*(\*(\*(arreglo+a)+b)+2);

leds.setPixelColor(NumLed, red, green, blue);

}

NumLed++;

}

}

}

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

//Serial.begin(9600);

//pinMode(2,OUTPUT);

leds.begin();

salida(\*\*\*arreglo);

leds.show();

}

void loop()

{

/\*

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Ingrese los datos de la bandera");

//Falta determinar como se organizará esta información

\*/

}

**24/09/2021**

AVANCE CÓDIGO DE LA MATRIZ EN QT

int prueba[12][12]; //tamanio matriz que simula la imagen

int p=1;

for(int i=0 ; i<4;i++){ //este for es para rellenar la matriz, simulando una imagen

for(int j=0; j<4; j++){

prueba[i][j]=p++; //Rellene la matriz que esta simulando la imagen

}

}

int matriz[4][4]; //Matriz de LEDs

int altura=12;

int ancho=12;

AVANCE CÓDIGO ARDUINO TINKERCAD

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

//Definimos el arreglo que se pega de Qt

int arr[16][16][3]=

{

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

};

int \*\*\*arreglo=arr;

void salida(\*\*\*int arreglo, Adafruit\_NeoPixel leds){

int NumLed = 0;

int red =0;

int green =0;

int blue =0;

for(int a=15;a=>0;a--){

for(int b=0;b<16;b++){

for(int c=0;c<3;c++){

red = \*(\*(\*(arreglo+a)+b)+0);

green = \*(\*(\*(arreglo+a)+b)+1);

blue = \*(\*(\*(arreglo+a)+b)+2);

leds.setPixelColor(NumLed, red, green, blue);

}

NumLed++;

}

}

}

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

Serial.begin(9600);

pinMode(2,OUTPUT);

}

void loop()

{

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Presione uno para imprimir la información ingresada");

Serial.flush();

int caso = Serial.parseInt();

if(caso==1){

leds.begin();

salida(\*\*\*arreglo);

leds.show();

}

}

**25/09/2021**

Nueva versión Arduino

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

//Definimos el arreglo que se pega de Qt

int arr[16][16][3]=

{

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

};

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

Serial.begin(9600);

pinMode(2,OUTPUT);

}

void loop()

{

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Presione uno para imprimir la información ingresada");

Serial.flush();

int caso = Serial.parseInt();

if(caso==1){

leds.begin();

for(int a=15;a=>0;a--){

for(int b=0;b<16;b++){

for(int c=0;c<3;c++){

red = arr[a][b][c];

green = arr[a][b][c+1];

blue = arr[a][b][c+2];

leds.setPixelColor(NumLed, red, green, blue);

}

NumLed++;

}

}

leds.show();

}

}

**CODIGO SIN ERRORES (SOLUCION ERROR CONDICIONALES)**

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

//Definimos el arreglo que se pega de Qt

int arr[16][16][3]=

{

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

};

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

Serial.begin(9600);

pinMode(2,OUTPUT);

}

void loop()

{

if (Serial.available()>0){

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Presione uno para imprimir la información ingresada");

Serial.flush();

int caso = Serial.parseInt();

if(caso==1){

int NumLed = 0;

int red =0;

int green =0;

int blue =0;

leds.begin();

for(int a=15;a>0 || a==0;a--){

for(int b=0;b<16;b++){

for(int c=0;c<3;c++){

red = arr[a][b][c];

green = arr[a][b][c+1];

blue = arr[a][b][c+2];

leds.setPixelColor(NumLed, red, green, blue);

}

NumLed++;

}

}

leds.show();

}

}

}

CAMBIO EN EL SERIAL DE ARDUINO

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

//Definimos el arreglo que se pega de Qt

int arr[16][16][3]=

{

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

};

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

Serial.begin(9600);

pinMode(2,OUTPUT);

}

void loop()

{

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("Presione uno para imprimir la información ingresada");

if (Serial.available()>0){

Serial.flush();

int caso = Serial.parseInt();

if(caso==1){

int NumLed = 0;

int red =0;

int green =0;

int blue =0;

leds.begin();

for(int a=15;a>0 || a==0;a--){

for(int b=0;b<16;b++){

for(int c=0;c<3;c++){

red = arr[a][b][c];

green = arr[a][b][c+1];

blue = arr[a][b][c+2];

leds.setPixelColor(NumLed, red, green, blue);

}

NumLed++;

}

}

leds.show();

}

}

}

**26/09/2021**

AVANCE SOBREMUESTREO EN QT

#include <iostream>

#include <QImage>

using namespace std;

void **Azul**();

void **Verde**();

int **main**()

{

string filename="../Parcial2/imagen.jpg";

QImage im( filename.c\_str());

int anchoim=im.width();

int altoim=im.height();

//Sobremuestreo

int recorrerx=0,recorrery=0,ledsmatrix;

float promedio=0,redimx,redimy;

int matrixleds[16][16]; //Se simula la matriz de LEDs de Tinkercad

ledsmatrix=16; //Ya que se trabaja con una matriz cuadrada se puede coger el alto o el ancho, en este caso se coge el ancho de la matriz de leds

//Se divide la imagen entre el tamaño de la matriz de leds para saber cuantos elementos se necesitan para obtener un dato de la matriz ya redimensionada

redimx=anchoim/ledsmatrix; //Cantidad de filas a redimensionar

redimy=altoim/ledsmatrix; //Cantidad de columnas a redimensionar

for(int filas=0;filas<ledsmatrix;filas++){ //Recorre el ancho de la matriz de leds

for(int columnas=0;columnas<ledsmatrix;columnas++){ //Recorre el alto de la matriz de leds

for(int itery=0, redimfila=recorrery ;itery<redimy;itery++,redimfila++){ //Recorre las filas a redimensionar

for(int iterx=0, redimcol=recorrerx;iterx<redimx;iterx++,redimcol++){ //Recorre las columnas a redimensionar

int rojo= im.pixelColor(redimfila,redimcol).red();

promedio = ( promedio ) + ( ( rojo) / (redimx\*redimy) ) ; //Se saca el promedio de la fila y las columnas a redimensionar

//Con lo que se obtiene un dato para la matriz ya redimensionada, estos datos se obtienen uno a uno

//este promedio debe hacerlo con los 3 colores, Rojo, azul, verde

}

}

recorrerx=recorrerx+redimx;

matrixleds[filas][columnas]=promedio; //Se guarda el promedio obtenido en la debida posicion de la matriz de leds

promedio=0;

//este arreglo debe hacerlo con los 3 colores, rojo, azul, verde. Guardando estos colores en matrices, int rojo[tamanioLeds][tamaniosLeds], asi seria el tamanio del arreglo rojo

cout<< "posicion filas: "<<filas<<" posicion columnas: "<<columnas<<" : "<<matrixleds[filas][columnas]<<endl;

}

recorrery=recorrery+redimy;

recorrerx=0;

}

Verde();

Azul();

return 0;

}

/\*

string filename="../Parcial2/imagen2.jpg";

QImage im( filename.c\_str());

int ancho=im.width();

int alto=im.height();

unsigned int pixelY=227;

unsigned int pixelX=227;

cout<<"Ancho: "<<ancho<<endl;

cout<<"Alto: "<<alto<<endl;

cout<<"intensidad del rojo: "<<im.pixelColor(pixelX,pixelY).red()<<endl;

cout<<"intensidad del verde: "<<im.pixelColor(pixelX,pixelY).green()<<endl;

cout<<"intensidad del azul: "<<im.pixelColor(pixelX,pixelY).blue()<<endl;

return 0;

}

}

\*/

void **Azul**(){

cout<<endl<<endl<<endl<<"Representacion del azul: "<<endl;

string filename="../Parcial2/imagen.jpg";

QImage im( filename.c\_str());

int anchoim=im.width();

int altoim=im.height();

int prueba[anchoim][altoim]; //Matriz que simula la imagen

int datos=1;

for(int pixelX=0 ; pixelX<anchoim;pixelX++){ //este for es para rellenar la matriz

for(int pixelY=0; pixelY<altoim; pixelY++){

prueba[pixelX][pixelY]=datos++; //Lo que se hace es que se rellena la fila y columna correspondiente empezando con un 1 y terminando con un 144

}

}

int recorrerx=0,recorrery=0,ledsmatrix;

float promedio=0,redimx,redimy;

int matrixleds[16][16]; //Se simula la matriz de LEDs de Tinkercad

ledsmatrix=16; //Ya que se trabaja con una matriz cuadrada se puede coger el alto o el ancho, en este caso se coge el ancho de la matriz de leds

//Se divide la imagen entre el tamaño de la matriz de leds para saber cuantos elementos se necesitan para obtener un dato de la matriz ya redimensionada

redimx=anchoim/ledsmatrix; //Cantidad de filas a redimensionar

redimy=altoim/ledsmatrix; //Cantidad de columnas a redimensionar

for(int filas=0;filas<ledsmatrix;filas++){ //Recorre el ancho de la matriz de leds

for(int columnas=0;columnas<ledsmatrix;columnas++){ //Recorre el alto de la matriz de leds

for(int itery=0, redimfila=recorrery ;itery<redimy;itery++,redimfila++){ //Recorre las filas a redimensionar

for(int iterx=0, redimcol=recorrerx;iterx<redimx;iterx++,redimcol++){ //Recorre las columnas a redimensionar

int azul= im.pixelColor(redimfila,redimcol).blue();

promedio = ( promedio ) + ( ( azul) / (redimx\*redimy) ) ; //Se saca el promedio de la fila y las columnas a redimensionar

//Con lo que se obtiene un dato para la matriz ya redimensionada, estos datos se obtienen uno a uno

//este promedio debe hacerlo con los 3 colores, Rojo, azul, verde

}

}

recorrerx=recorrerx+redimx;

matrixleds[filas][columnas]=promedio; //Se guarda el promedio obtenido en la debida posicion de la matriz de leds

promedio=0;

//este arreglo debe hacerlo con los 3 colores, rojo, azul, verde. Guardando estos colores en matrices, int rojo[tamanioLeds][tamaniosLeds], asi seria el tamanio del arreglo rojo

cout<< "posicion filas: "<<filas<<" posicion columnas: "<<columnas<<" : "<<matrixleds[filas][columnas]<<endl;

}

recorrery=recorrery+redimy;

recorrerx=0;

}

}

void **Verde**(){

cout<<endl<<endl<<endl<<"Representacion del verde: "<<endl;

string filename="../Parcial2/imagen.jpg";

QImage im( filename.c\_str());

int anchoim=im.width();

int altoim=im.height();

int recorrerx=0,recorrery=0,ledsmatrix;

float promedio=0,redimx,redimy;

int matrixleds[16][16]; //Se simula la matriz de LEDs de Tinkercad

ledsmatrix=16; //Ya que se trabaja con una matriz cuadrada se puede coger el alto o el ancho, en este caso se coge el ancho de la matriz de leds

//Se divide la imagen entre el tamaño de la matriz de leds para saber cuantos elementos se necesitan para obtener un dato de la matriz ya redimensionada

redimx=anchoim/ledsmatrix; //Cantidad de filas a redimensionar

redimy=altoim/ledsmatrix; //Cantidad de columnas a redimensionar

for(int filas=0;filas<ledsmatrix;filas++){ //Recorre el ancho de la matriz de leds

for(int columnas=0;columnas<ledsmatrix;columnas++){ //Recorre el alto de la matriz de leds

for(int itery=0, redimfila=recorrery ;itery<redimy;itery++,redimfila++){ //Recorre las filas a redimensionar

for(int iterx=0, redimcol=recorrerx;iterx<redimx;iterx++,redimcol++){ //Recorre las columnas a redimensionar

int verde= im.pixelColor(redimfila,redimcol).green();

promedio = ( promedio ) + ( ( verde) / (redimx\*redimy) ) ; //Se saca el promedio de la fila y las columnas a redimensionar

//Con lo que se obtiene un dato para la matriz ya redimensionada, estos datos se obtienen uno a uno

//este promedio debe hacerlo con los 3 colores, Rojo, azul, verde

}

}

recorrerx=recorrerx+redimx;

matrixleds[filas][columnas]=promedio; //Se guarda el promedio obtenido en la debida posicion de la matriz de leds

promedio=0;

//este arreglo debe hacerlo con los 3 colores, rojo, azul, verde. Guardando estos colores en matrices, int rojo[tamanioLeds][tamaniosLeds], asi seria el tamanio del arreglo rojo

cout<< "posicion filas: "<<filas<<" posicion columnas: "<<columnas<<" : "<<matrixleds[filas][columnas]<<endl;

}

recorrery=recorrery+redimy;

recorrerx=0;

}

}

ARDUINO

#include <Adafruit\_NeoPixel.h>

//Icluimos el pin de control de la salida

#define LED\_PIN 2

//Incluimos el numero de pixeles a controlar

#define LED\_COUNT 256

//Definimos el LED NeoPixel

Adafruit\_NeoPixel leds(LED\_COUNT, LED\_PIN, NEO\_GRB + NEO\_KHZ800);

//Definimos el arreglo que se pega de Qt

int arr[16][16][3]=

//--------- AQUI SE PEGA LA SALIDA DEL PROGRAMA DE QT---------------

{

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0},{255,233,0}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

{{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200},{20,47,200}},

};

//---------------------HASTA AQUI---------------------------

void setup(){

//Ya tenemos el serial para recibir la información del imagen de

//la bandera desde Qt

Serial.begin(9600);

pinMode(2,OUTPUT);

}

void loop()

{

//Se Establece el menu para el usuario

Serial.print("BIENVENIDO");

Serial.print("\n");

Serial.print("Presione uno para imprimir la información ingresada");

Serial.print("\n");

Serial.flush();

int caso = Serial.parseInt();

if(caso==1){

int NumLed = 0;

int red =0;

int green =0;

int blue =0;

leds.begin();

for(int a=0;a<15;a++){

for(int b=0;b<16;b++){

red = arr[a][b][0];

green = arr[a][b][1];

blue = arr[a][b][2];

/\*Serial.print("numled ");

Serial.print(NumLed);

Serial.print("\n");

Serial.print("red ");

Serial.print(red);

Serial.print("\n");

Serial.print("green ");

Serial.print(green);

Serial.print("\n");

Serial.print("blue ");

Serial.print(blue);

Serial.print("\n");\*/

leds.setPixelColor(NumLed, red, green, blue);

NumLed++;

}

}

leds.show();

Serial.print("llego");

}

}

**27/09/2021**

ACTUALIZACIÓN SUMUESTREO

#include <iostream>

#include <QImage>

using namespace std;

void **Azul**();

void **Verde**();

int **main**()

{

string filename="../Parcial2/brasil.jpg";

QImage im( filename.c\_str());

int anchoim=im.width();

int altoim=im.height();

//Submuestreo

int recorrerx=0,recorrery=0,ledsmatrix;

float promedio=0,redimx,redimy;

int matrixleds[16][16]; //Se simula la matriz de LEDs de Tinkercad

ledsmatrix=16; //Ya que se trabaja con una matriz cuadrada se puede coger el alto o el ancho, en este caso se coge el ancho de la matriz de leds

//Se divide la imagen entre el tamaño de la matriz de leds para saber cuantos elementos se necesitan para obtener un dato de la matriz ya redimensionada

redimy=anchoim/ledsmatrix; //Cantidad de filas a redimensionar

redimx=altoim/ledsmatrix; //Cantidad de columnas a redimensionar

for(int filas=0;filas<ledsmatrix;filas++){ //Recorre el ancho de la matriz de leds

for(int columnas=0;columnas<ledsmatrix;columnas++){ //Recorre el alto de la matriz de leds

for(int itery=0, redimfila=recorrery ;itery<redimy;itery++,redimfila++){ //Recorre las filas a redimensionar

for(int iterx=0, redimcol=recorrerx;iterx<redimx;iterx++,redimcol++){ //Recorre las columnas a redimensionar

int rojo= im.pixelColor(redimfila,redimcol).red();

promedio = ( promedio ) + ( ( rojo) / (redimx\*redimy) ) ; //Se saca el promedio de la fila y las columnas a redimensionar

//Con lo que se obtiene un dato para la matriz ya redimensionada, estos datos se obtienen uno a uno

//este promedio debe hacerlo con los 3 colores, Rojo, azul, verde

}

}

recorrerx=recorrerx+redimx;

matrixleds[filas][columnas]=promedio; //Se guarda el promedio obtenido en la debida posicion de la matriz de leds

promedio=0;

//este arreglo debe hacerlo con los 3 colores, rojo, azul, verde. Guardando estos colores en matrices, int rojo[tamanioLeds][tamaniosLeds], asi seria el tamanio del arreglo rojo

cout<< "posicion filas: "<<filas<<" posicion columnas: "<<columnas<<" : "<<matrixleds[filas][columnas]<<endl;

}

recorrery=recorrery+redimy;

recorrerx=0;

}

Verde();

Azul();

return 0;

}

/\*

string filename="../Parcial2/imagen2.jpg";

QImage im( filename.c\_str());

int ancho=im.width();

int alto=im.height();

unsigned int pixelY=227;

unsigned int pixelX=227;

cout<<"Ancho: "<<ancho<<endl;

cout<<"Alto: "<<alto<<endl;

cout<<"intensidad del rojo: "<<im.pixelColor(pixelX,pixelY).red()<<endl;

cout<<"intensidad del verde: "<<im.pixelColor(pixelX,pixelY).green()<<endl;

cout<<"intensidad del azul: "<<im.pixelColor(pixelX,pixelY).blue()<<endl;

return 0;

}

}

\*/

void **Azul**(){

cout<<endl<<endl<<endl<<"Representacion del azul: "<<endl;

string filename="../Parcial2/imagen.jpg";

QImage im( filename.c\_str());

int anchoim=im.width();

int altoim=im.height();

int prueba[anchoim][altoim]; //Matriz que simula la imagen

int datos=1;

for(int pixelX=0 ; pixelX<anchoim;pixelX++){ //este for es para rellenar la matriz

for(int pixelY=0; pixelY<altoim; pixelY++){

prueba[pixelX][pixelY]=datos++; //Lo que se hace es que se rellena la fila y columna correspondiente empezando con un 1 y terminando con un 144

}

}

int recorrerx=0,recorrery=0,ledsmatrix;

float promedio=0,redimx,redimy;

int matrixleds[16][16]; //Se simula la matriz de LEDs de Tinkercad

ledsmatrix=16; //Ya que se trabaja con una matriz cuadrada se puede coger el alto o el ancho, en este caso se coge el ancho de la matriz de leds

//Se divide la imagen entre el tamaño de la matriz de leds para saber cuantos elementos se necesitan para obtener un dato de la matriz ya redimensionada

redimx=anchoim/ledsmatrix; //Cantidad de filas a redimensionar

redimy=altoim/ledsmatrix; //Cantidad de columnas a redimensionar

for(int filas=0;filas<ledsmatrix;filas++){ //Recorre el ancho de la matriz de leds

for(int columnas=0;columnas<ledsmatrix;columnas++){ //Recorre el alto de la matriz de leds

for(int itery=0, redimfila=recorrery ;itery<redimy;itery++,redimfila++){ //Recorre las filas a redimensionar

for(int iterx=0, redimcol=recorrerx;iterx<redimx;iterx++,redimcol++){ //Recorre las columnas a redimensionar

int azul= im.pixelColor(redimfila,redimcol).blue();

promedio = ( promedio ) + ( ( azul) / (redimx\*redimy) ) ; //Se saca el promedio de la fila y las columnas a redimensionar

//Con lo que se obtiene un dato para la matriz ya redimensionada, estos datos se obtienen uno a uno

//este promedio debe hacerlo con los 3 colores, Rojo, azul, verde

}

}

recorrerx=recorrerx+redimx;

matrixleds[filas][columnas]=promedio; //Se guarda el promedio obtenido en la debida posicion de la matriz de leds

promedio=0;

//este arreglo debe hacerlo con los 3 colores, rojo, azul, verde. Guardando estos colores en matrices, int rojo[tamanioLeds][tamaniosLeds], asi seria el tamanio del arreglo rojo

cout<< "posicion filas: "<<filas<<" posicion columnas: "<<columnas<<" : "<<matrixleds[filas][columnas]<<endl;

}

recorrery=recorrery+redimy;

recorrerx=0;

}

}

void **Verde**(){

cout<<endl<<endl<<endl<<"Representacion del verde: "<<endl;

string filename="../Parcial2/imagen.jpg";

QImage im( filename.c\_str());

int anchoim=im.width();

int altoim=im.height();

int recorrerx=0,recorrery=0,ledsmatrix;

float promedio=0,redimx,redimy;

int matrixleds[16][16]; //Se simula la matriz de LEDs de Tinkercad

ledsmatrix=16; //Ya que se trabaja con una matriz cuadrada se puede coger el alto o el ancho, en este caso se coge el ancho de la matriz de leds

//Se divide la imagen entre el tamaño de la matriz de leds para saber cuantos elementos se necesitan para obtener un dato de la matriz ya redimensionada

redimx=anchoim/ledsmatrix; //Cantidad de filas a redimensionar

redimy=altoim/ledsmatrix; //Cantidad de columnas a redimensionar

for(int filas=0;filas<ledsmatrix;filas++){ //Recorre el ancho de la matriz de leds

for(int columnas=0;columnas<ledsmatrix;columnas++){ //Recorre el alto de la matriz de leds

for(int itery=0, redimfila=recorrery ;itery<redimy;itery++,redimfila++){ //Recorre las filas a redimensionar

for(int iterx=0, redimcol=recorrerx;iterx<redimx;iterx++,redimcol++){ //Recorre las columnas a redimensionar

int verde= im.pixelColor(redimfila,redimcol).green();

promedio = ( promedio ) + ( ( verde) / (redimx\*redimy) ) ; //Se saca el promedio de la fila y las columnas a redimensionar

//Con lo que se obtiene un dato para la matriz ya redimensionada, estos datos se obtienen uno a uno

//este promedio debe hacerlo con los 3 colores, Rojo, azul, verde

}

}

recorrerx=recorrerx+redimx;

matrixleds[filas][columnas]=promedio; //Se guarda el promedio obtenido en la debida posicion de la matriz de leds

promedio=0;

//este arreglo debe hacerlo con los 3 colores, rojo, azul, verde. Guardando estos colores en matrices, int rojo[tamanioLeds][tamaniosLeds], asi seria el tamanio del arreglo rojo

cout<< "posicion filas: "<<filas<<" posicion columnas: "<<columnas<<" : "<<matrixleds[filas][columnas]<<endl;

}

recorrery=recorrery+redimy;

recorrerx=0;

}

}