Parcial 2 - Informa2

Santiago Vélez Arboleda.

Mariana Noreña Vásquez.

Despartamento de Ingeniería Electrónica y
Telecomunicaciones
Universidad de Antioquia
Medellín
Septiembre de 2021

Índice

1. Clases implementadas y estructura

Se implementará una sola clase que tendrá los siguientes métodos:

1. Resize: Este método será el encargado de redimensionar la imagen ingresada al tamaño de la matriz.
2. Promcolor: Obtiene el promedio de los colores consecutivos. Esta es empleada en el método Resize.
3. Colors: Esta función extrae los colores que componen a la imagen redimensionada en sus tres componentes: rojo, verde y azul.
4. Escribir: Toma los datos obtenidos en la función Colors y los introduce en un archivo .txt.

De igual forma, tendrá lo siguientes atributos:

1. Dos instancias de la clase QImage.
2. Dos enteros para el alto y ancho.
3. Una matriz tridimensional
4. Factor de escala para el alto y ancho de la imagen.
5. Numero de Leds.

2. Estructura del circuito

Para el montaje del circuito, se utilizaron los siguientes componentes:

1. x16 Tiras led.
2. 1 Placa de pruebas.
3. 1 Arduino.

Seguidamente, para comunicar las 16 tiras led, se conectó la entrada 2 del arduino con la entrada de la tira led. Seguidamente, la salida se conecta a la entrada de la siguiente para conectarlas en casdada. De una forma similar, se entrega una potencia de 5V proveniente del mismo y por último se conecta a tierra.

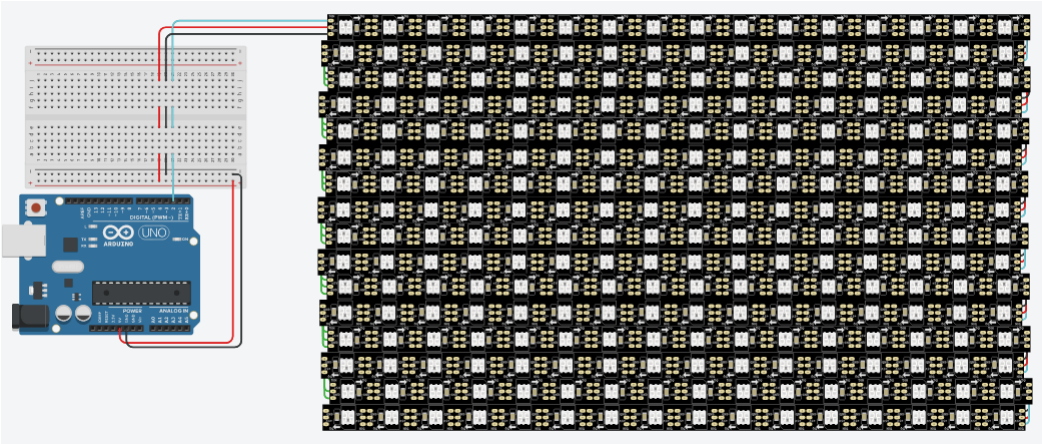

Figura 1: Circuito

3. Manual

El presente circuito es la implementación de una matriz 16x16 compuesta de tiras de NeoPixels, con la finalidad de mostrar en ella una imagen, la cual, posteriormente usted suministrará.
Tenga en cuenta la carpeta informa2, esta contiene los siguientes archivos: pimage.h, pimage.cpp y main.cpp. Estos le permitirán manejar adecuadamente la imagen ingresada. Cuando ejecute la anterior lista de archivos, se le pedirá que proporcione la ruta donde se encuentra la imagen que desea mostrar en la matriz, un ejemplo de la manera adecuada de ingresarla es: /Users/disco/...
Para mayor facilidad, la ubicación del archivo la puede ver accediendo a las propiedades de la imagen cambiando los símbolos por /
Es de vital importancia que siga las instrucciones dadas a continuación, puesto que, de ello dependerá el correcto funcionamiento del programa.
Con respecto a la ejecución del circuito planteado en Tinkercad, es esencial que en primera instancia ejecute la lista de archivos mencionados con anterioridad puesto que al finalizar su ejecución se le brindará un archivo de texto y dentro de este, habrá un código, el cual deberá copiar todo lo que haya dentro de dicho archivo), a continuación, proceda a pegar dichos datos después del signo = de la variable cuyo nombre es matriz, esto es:

byte matriz[filas][leds][leds]=aquí debe de pegar la información

4. Módulos de código

4.1. pimage.h

A continuación, se presenta el código de pimage.h, en el cual se muestra que contiene la clase. Tales como atributos y métodos.

```
#include <iostream>
#include <QImage>
#include "string"
#include <math.h>
#include <fstream>

using namespace std;
constexpr int filas = 3, pixeles = 16;

class pimage
{

private:

    QImage image, *imageE;
    int w = 0, h = 0, x = 1, y = 1;
    int red = 0, green = 0, blue = 0;
    int matriz[filas][pixeles][pixeles];
    float leds = 16, fw = 1.0,fh = 1.0, X = 1.0, Y = 1.0;

public:
    pimage(string img);
    QRgb promcolor(QRgb color1, QRgb color2);
    void resize();
    void colors();
    void escribir();
    ~pimage();
};
```

4.2. main.cpp

A continuación, se presenta el código main.cpp

```
#include "pimage.h"

int main()
{
    string ruta;

    cout << "Ingrese la ruta de la imagen: ";
```

```
    getline(cin, ruta);
    fflush(stdin);

    pimage imagen(ruta);

    imagen.resize();
    imagen.colors();
    imagen.escribir();

    return 0;
}
```

4.2.1. pimage.cpp

Se presenta el código pimage.cpp:

```
#include "pimage.h"

pimage::pimage(string img)
{
    if(image.load(img.c_str()) == false){
        cout << "No se pudo cargar la imagen. Verifique la ruta ingresada" << en
    }
    else{

        image = image.convertToFormat(QImage::Format_RGB888);

    }
}

void pimage::resize()
{

    fw = image.width()/leds;
    fh = image.height()/leds;
    w = round(image.width()/fw);
    h = image.height()/fh;

    imageE = new QImage(w,h,QImage::Format_RGB888);

    for(int m = 0; m < h; m++){
        for(int f = 0; f < w; f++){
            X = f*fw;
            Y = m*fh;
            x = X;
            y = Y;
```

```
            if((X-x > 0.5) && (x+1 < w)){
                x += 1;
                if((Y-y > 0.5) && (y+1 < h)) y += 1;
            }
            imageE->setPixel(f,m,image.pixel(x,y));
        }
    }
    imageE->save("imagenescalada.jpg");
}



void pimage::colors()

{
    for(int f = 0; f < imageE->height(); f++){
        for(int c = 0; c < imageE->width(); c++){
            red = imageE->pixelColor(c,f).red();
            green = imageE->pixelColor(c,f).green();
            blue = imageE->pixelColor(c,f).blue();
            if((red == 255) && (green == 255) && (blue == 255)){
                red = 254;
                green = 254;
                blue = 254;
            }
            matriz[0][f][c] = red;
            matriz[1][f][c] = green;
            matriz[2][f][c] = blue;
        }
    }
}

void pimage::escribir()
{
    ofstream archivo;
    archivo.open("Datos.txt", ios::out);

    archivo << "{" << endl;
    for(int f = 0; f < 3; f++){
        archivo << "{" << endl;
        for(int c = 0; c < pixeles; c++){
            archivo << "{";
            for(int d = 0; d < pixeles; d++){
                archivo << matriz[f][c][d];
                if(d < pixeles-1) archivo << ",";
            }
```

```
            if(c < pixeles-1) archivo << "},";
            else archivo << "}";
        }
        archivo << endl;
        if(f < 2) archivo << "}," << endl;
        else archivo << "}" << endl;
    }

    archivo << "};";
    archivo.close();
}

pimage::~pimage()
{
    delete imageE;
}
```

5. Problemas presentados

1. Reescalado de la imagen: Cuando se estaba desarrollando el código de escalado de la imagen, se tuvieron varios errores. Uno de ellos, fueron los colores de la imagen reescalada ya que en base de los colores y forma de la imagen original, no eran del todo fieles. También, después de realizar el proceso, la imagen resultante era de un color solido diferente a la imagen base. Además, cuando se intentaba hacer el proceso con una imagen base de pequeña resolución, se aplicaba el sobremuestreo, la resolución de la resultante no daba lo esperado (En ocasiones agregaba o quitaba un pixel a la resolución de reescalado).