Informe Parcial 2

Informática II

Abdy Sánchez Salazar
Juan Pablo Vargas Mosquera

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

Índice

1. Análisis del Problema

El análisis del problema represento un verdadero reto ya que tuvimos que hacer varios análisis, tuvimos que mirar las grabaciones varias veces las grabaciones que estaban en el classroom para tener todo claro, lo que podíamos y no podíamos utilizar para este parcial 2. Al principio habíamos planteado hacer un montaje de un circuito de una matriz de 8x8 utilizando tiras de 8 neopixeles pero a medida que fuimos avanzando y lo cambiamos a 16x16 para que se pudiera ver mejor la bandera que fuéramos a presentar, La primer idea que tuvimos fue era utilizar una clase que contuviera el **Qimage** donde se heredarían los metodos para poder leer los pixeles y guardarlos en una lista muy larga, para después pasarlos por nuestras funciones de redimensionamiento, submuestreo y sobremuestreo, hicimos la clase de forma de que al ejecutarlo para que el para que el programa lea la imagen solo sería que ingrese el nombre de la imagen junto a **.jpg**, por ejemplo *Brazil.jpg*. El programa como habíamos planteado antes nos funcionaba para mostrar las banderas, pero solo las de franjas horizontales, al poner banderas de franja verticales el programa entregaba la información, pero al pasarlo al montaje en **TINKERCAD** no lo representaba bien. El análisis que tuvimos para el problema fue que teníamos que hallar una forma de mantener las proporciones o la simetría cuando lo fuéramos a redimensionar a nuestra matriz que es de 16x16 para próximamente entregarlo de forma que las funciones que nos ofrece el **TINKERCAD** lo leyera y mostrara en los leds la información deseada.

2. Esquema de Tareas

Definimos las tareas para el desarrollo del algoritmo de la siguiente manera:

- La primera tarea fue realizar un análisis para saber que nos pedían en este problema llamado parcial 2, todo esto antes de ponernos a tirar código.
- Procedimos a informarnos y leer que podíamos utilizar como el **Qimage**, los leds **RGB** y no dejar vacíos para poder empezar con el código y el montaje.
- Hicimos una clase que heredara el **Qimage** para poder usar sus atributos, siempre y cuando sin violar los requisitos mínimos que nos die9ron en el parcial.
- Luego nos pusimos a hacer el montaje en **TINKERCAD**, donde nos guiamos un poco de la grabación de la clase y lo que habíamos aprendido del parcial anterior.
- Después de que usáramos una función para encender los leds y ver que si estuviera bien montando el circuito y que no hubiera problemas en las conexiones.
- También empezamos a hacer la creación y pruebas de la clase.
- Haremos la migración de código de Qt a **TINKERCAD**.
- Pruebas finales para ver que no haya fallas en el algoritmo y se muestre en los leds lo que se desea.

3. Algorítmo

Para más practicidad utilizamos una matriz de neopixels de 16X16 para que en el programa **TINKERCAD** pueda procesarlo de una manera eficiente, para que a los ojos y experiencia del usuario resulte amigable y pueda distinguir lo ingresado por él. Con ayuda de la librería **Qimage** podremos cargar imágenes a QT, lo cual solo lo utilizaremos para saber la cantidad de pixeles que tiene la imagen cargada y luego pasarlos en nuestros algoritmos de sub y sobremuestreo, según las dimensiones de la imagen ingresada.

4. Consideraciones en la implementación del Algoritmo

Al analizar las alternativas propuestas de la solución y su respectivo algoritmo, el uso excesivo de memoria para el guardado de información de los valores enteros por el color de cada pixel en su componente **RGB** en la imagen. Por lo cual es fundamental buscar los métodos en las clases ya integradas en Qt que nos permitan la modificación de pixeles en la misma imagen.

5. Clases Implementadas

La única clase que utilizamos es la "objeto imagen", donde básicamente va heredar los atributos de **Qimage** para que podamos cargar y obtener la información de los pixeles de la imagen, que luego se convertirá en un objeto el cual empezara a ser la materia prima para nuestro algoritmo.

6. Esquema de la Clase

- Primero se incluye la clase objeto imagen.
- Luego definimos la ruta donde se extraerá la imagen.
- Añadimos el nombre de la imagen a la ruta.
- Cargamos la Imagen deseada.
- Después se procede a hacer una comparación de pixeles para saber que método tendrá que aplicársele a la imagen cargada.
- Se le aplica el método de sub o sobre muestreo, si son iguales las dimensiones de la imagen a la de los leds sigue normal el programa.
- En los distintos métodos de redimensionamiento entrara a sus respectivos *For* con los cuales se irán guardando en un apuntador llamado colores.
- Después de que ejecutemos el programa e ingresemos el nombre de la imagen, el programa hará todo como y simplemente nos imprimirá en la pantalla la información que vamos a copiar y pegar sin la última coma en el **TINKERCAD** para mostraren los leds la imagen ingresada anteriormente.

7. Módulos del código implementado

Los módulos se encuentran en el GitHub que serían estos 3 ya que nosotros solo manejamos una clase, los módulos son:

- Main.cpp
- Objeto imagen.cpp
- Objeto imagen.h

8. Estructura del circuito montado

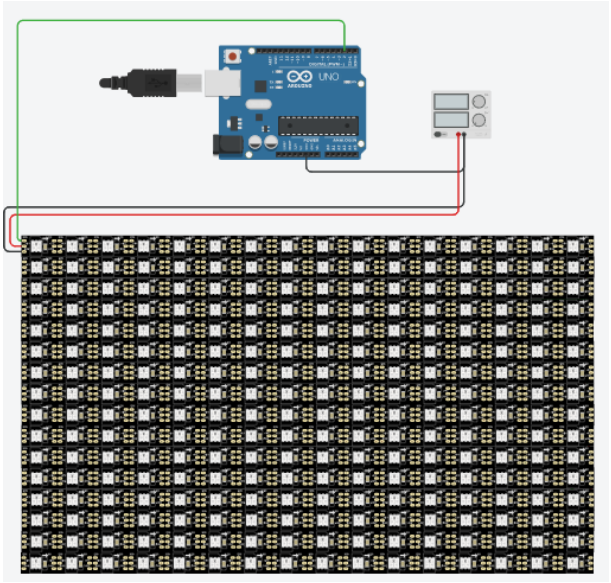

Figura 1: Montaje TINKERCAD

```
#include <Adafruit_NeoPixel.h>
#define LED_COUNT 256
#define LED_PIN 2

Adafruit_NeoPixel leds(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
// C++ code
//
byte Arr[16][16][3]=

{
//Aqui va copiado lo que nos imprime el QT sin la ultima coma.
};



void setup()
{
  leds.begin();
  int cont = 0;
```

```
  for(int fil = 0; fil < 16;fil++){
    for(int col = 0;col <16;col++){

leds.setPixelColor(cont,Arr[fil][col][0],Arr[fil][col][1],Arr[fil][col][2]);

      cont++;
    }
  }

  leds.show();
}
void loop()
{
}
```

9. Problemas Presentados durante la implementación

El mayor problema que se nos presento fue que con el método que ya teníamos planeados al principio nos imprimía bien las banderas horizontales, pero al ser banderas verticales o al tener diagonales u otro tipo de figura los hacia duplicado o solo imprimía un color, entonces nos tocó cambiar la clase desde el principio porque teníamos una lista muy grande donde estaba guarda la información que nos daba el algoritmo, entonces nos tocó hacer un vector que contiene otro vector y este vector a su vez contiene tres datos que son unos unsigned int que serían los **RGB**, con la cual si logramos mostrar los que deseábamos en los led y todas las banderas. Ya en el arduino no tuvimos no tuvimos problemas.

10. Manual

1. Lo primero es guardar la imagen que el usuario escoja en la carpeta imagen.
2. Guardar la imagen en .jpg.
3. Ejecutar el programa.
4. Escribir el nombre de la imagen anexando el .jpg.
5. Copiar el resultado desde la consola, omitiendo la ultima coma.
6. Pegue el resultado en el **TINKERCAD** y Corra la simulacion.