Parcial 2

Informa2 S.A.S.

Holman Londoño Restrepo

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

Índice

1. Clases implementadas. **2**

2. Esquema clases implementadas. **2**

3. Módulos de código. **2**

4. Estructura circuito. **3**

5. Problemas durante la implementacion **4**

6. Manual. **5**

1. Clases implementadas.

Para el desarrollo de este parcial me pareció mucho más óptimo implementar simplemente una clase, ya que así eran menos los punteros que debía declarar en el main que hicieran referencias a estas clases, además, de que desde mi perspectiva, implementar más clases complicaba innecesariamente la codificación; la clase que implemente se llama Métodos la cual es la encargada de realizar todos los procesos de sub y sobre muestreo según sea el caso, el constructor de esta clase recibe tres parámetros los cuales son "DimensionImagen", este parámetro representa la dimensión de la imagen a codificar y es el resultado de multiplicar el Ancho y el alto de esta imagen, el segundo parámetro es .ªnchoImgOriginal", como su nombre lo dice se usa para almacenar el valor del ancho de la imagen a tratar, y el último parámetro es .ªltoImgOriginal.el cual recibe el valor de alto de la imagen que ingresa el usuario, solo quise que recibiera estos parámetros, ya que son valores indispensables para la realización de las funciones de submuestreo y sobremuestreo y son valores que varían en cada imagen. En el interior de esta clase están implementadas 4 funciones, SubMuestreoAncho, SubMuestreoAlto, SobreMuestreoAncho, SobreMuestreoAlto, estas funciones reciben un doble puntero que apunta a la matriz donde se almacenan los datos RGB que están dentro de la imagen ingresada por el usuario y devuelven un doble puntero que hace referencia a la matriz que resulta luego de hacer el proceso que tiene por nombre la función llamada.

2. Esquema clases implementadas.

- **Clase implementada.**
  - 'Metodos'
    - -Parametros:
      - DimensionImagen
      - AnchoImgOriginal
      - AltoImgOriginal
    - - Funciones:
      - - int **SubMuestreoAncho(int **MatrizImagen);
      - - int **SubMuestreoAlto(int **MatrizImagen);
      - - int **SobreMuestreoAncho(int **MatrizImagen);
      - - int **SobreMuestreoAlto(int **MatrizImagen);
    - - Atributos privados:
      - - int **MatrizFilas
      - - int **MatrizImagenCodificada
      - - int AnchoFinal=16
      - - int AltoFinal=9
      - - int AnchoImgO
      - - int AltoImgO
      - - unsigned long Dimension

3. Módulos de código.

- Objeto en main para la clase metodos:

```
QImage Imagen;
fstream Archivo ("ValorRGB.txt", fstream::out);
int **MatrizImg, **MatrizImgC, **MatrizImgF;
int Ancho, Alto, Ancho2=16, Alto2=9;
string FilePath = "../Prueba2_Parcial_2/Imagenes/", FileName;
unsigned long PosPixel = 0, Dimension;

cout << "Ingrese la ruta de la imagen a procesar: ";
getline(cin, FileName); cout << endl;
```

```
FilePath.append(FileName);
if(Imagen.load(FilePath.c_str())){
    cout << "Imagen Cargada con exito. " << endl;
    Ancho = Imagen.width();
    Alto = Imagen.height();
    Dimension = Ancho*Alto;
    Metodos IM(Dimension, Ancho, Alto);//Interaccion con
                                       //Clase "metodos".
```

- Llamado desde main a las funciones dentro de la clase metodos:

```
if(Ancho2 != Ancho){
        if(Ancho > Ancho2){
            MatrizImgC = IM.SubMuestreoAncho(MatrizImg);
        }
        else{
            MatrizImgC = IM.SobreMuestreoAncho(MatrizImg);
        }
    }
    else {
        MatrizImgC = MatrizImg;
    }
    if(Alto2 != Alto){
        if(Alto >= Alto2){
            MatrizImgF = IM.SubMuestreoAlto(MatrizImgC);
        }
        else {
            MatrizImgF = IM.SobreMuestreoAlto(MatrizImgC);
        }
    }
    else {
        MatrizImgF = MatrizImgC;
    }
```

4. Estructura circuito.

El circuito realmente no fue algo que demandara mucho trabajo, ya que las tiras neopixel hacen el proceso de corrido de datos automáticamente y se encargan de procesar los datos ingresados para mostrar el color requerido, así que esto también simplifica mucho el código del Arduino el cual solo necesitó de unas pocas líneas de código para cumplir con lo deseado, este es el código implementado en Arduino fue el siguiente:

```
#include <Adafruit_NeoPixel.h>

#define LED_PIN 2
#define LED_COUNT 144

Adafruit_NeoPixel Leds(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

void setup()
{
  Leds.begin();
  int MatrizImg[LED_COUNT+1][3]= {/*Reemplace este
```

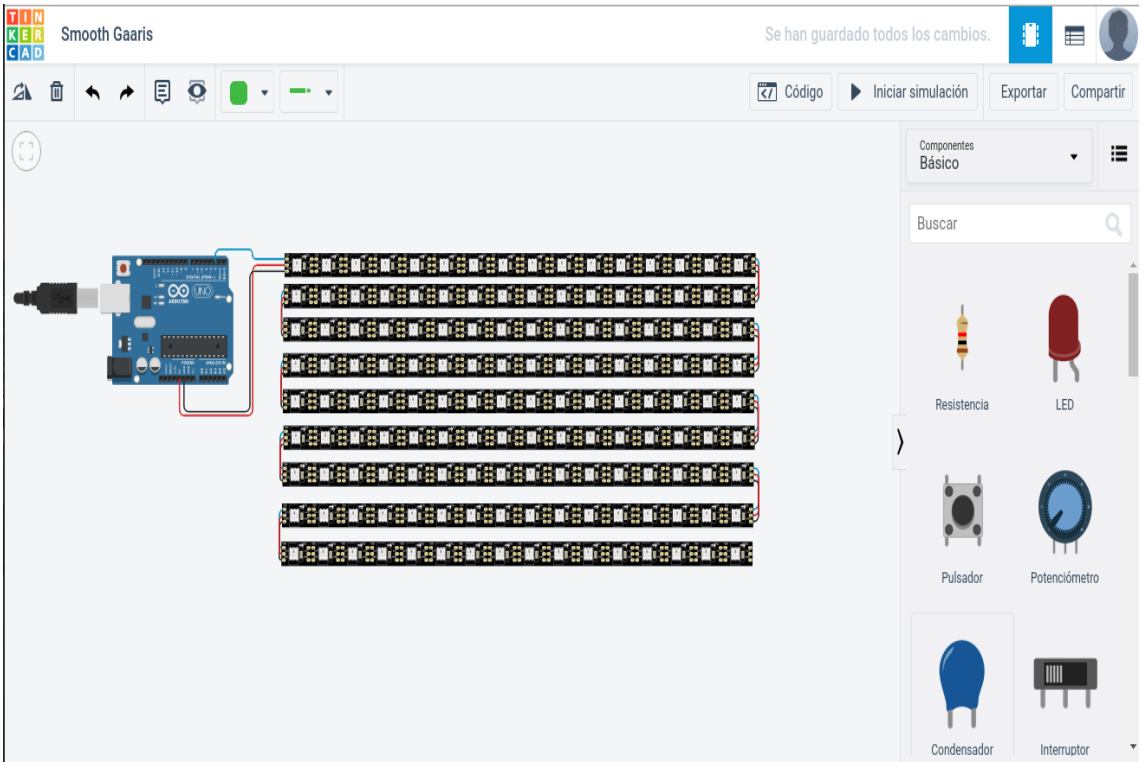
```
  comentario con los datos del txt*/};

  for(int i=0; i<LED_COUNT; i++){
    Leds.setPixelColor(i, MatrizImg[i][0], MatrizImg[i][1], MatrizImg[i][2]);
  }
  Leds.show();
}

void loop()
{

}
```

- Esta es la estructura del circuito:



5. Problemas durante la implementacion

Durante el desarrollo de este parcial tuve varios problemas los cuales detuvieron mucho el avance que tenía en este, el primero fue el tiempo, ya que estamos finalizando semestre luego de un paro de tres meses así que estamos teniendo actividades evaluativas prácticamente cada semana en diferentes materias, además también debemos cumplir con la entrega de las prácticas de laboratorio de esta materia y el desarrollo de estas demanda mucho tiempo, otro problema fue desarrollar la lógica del submuestreo, ya que este el código que demando más tiempo y razonamiento, hasta el punto que considero que tomo más tiempo el desarrollar la lógica que escribir el código.

6. Manual.

1. Debe guardar la imagen que quiere procesar en la carpeta directorio del programa.
2. Al correr el programa este le pedira que ingrese el nombre de la imagen, es importante que solo ingrese el nombre y la extension, sin la ruta, debido a que la ruta de la carpeta ya se encuentra almacenada en una variable dentro del programa.
3. Luego de ingresar el nombre de la imagen junto con su extension, le saldra un mensaje confirmando que la imagen fue tratada con exito.
4. Despues de leer el mensaje de confirmacion se debe dirigir a la carpeta 'build' que crea el programa automaticamente y alli se encontrara un TXT llamado 'ValorRGB.txt', en este se encuentra almacenada la informacion que debe introducir en el Arduino.
5. Por ultimo, copie todo lo que esta dentro del TXT y peguelo en el Arduino, hay un comentario que le indica donde debe pegar este codigo, presione el boton iniciar simulacion y verá la imagen representada en la matriz de leds.