Análisis e Informe

Parcial 2

Danier Santiago Ortega Restrepo
c.c.1036681098
danier.ortega@udea.edu.co

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

Índice

1. RECURSOS

Despues de leer las indicaciones del parcial 2, primero se tendran en cuenta los recursos disponibles para la realizacion de este, se estudiaran conceptos de manejo de leds RGB(red green and blue), uso de la plataforma Tinkercad, arduino, lenguaje C++(Qt creator), manejo y libreria para el uso de tiras led neopixel, informacion multimedia relacionada con el proyecto y enfasis en el tiempo de realizacion.

Algunos recursos probables pero de analizar, circuito integrado, entorno de desarrollo arduino, ya sea online o descargar la app.

2. PROPUESTA

Una propuesta para empezar el parcial seria empezar por practicar con algunos leds RGB y practicar su funcionamiento y asi practicar su funcionamiendo y maniobrabilidad de colores a conveniencia del usuario.

3. IDEA

En la imagen siguiente vemos una idea de partida para el parcial en donde se trabaja desde la parte en que se ingresan los datos a la placa de arduino en el entorno tikercad para su posterior funcionamiento.

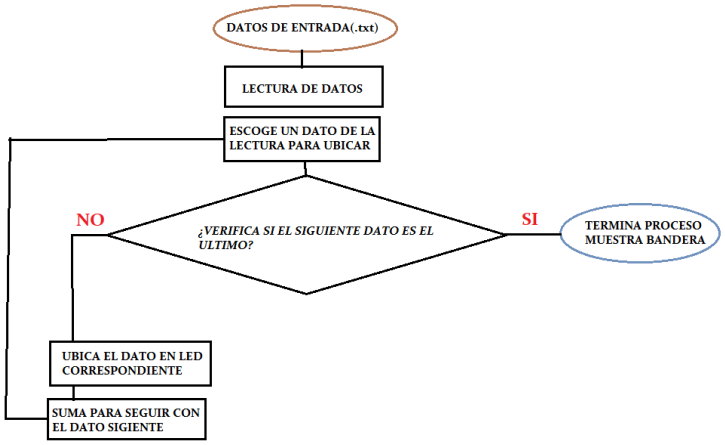

Figura 1: Diagrama de idea general

Practicar el procesamiento de imagenes dictado por el profesor y buscar ayudas para la implementacion de esta parte fundamental, ya que a partir de el manejo adecuado de la imagen dependera la adaptacion a el arreglo de leds.

Crear funciones para lectura de imagen, adaptacion de imagen para el tamaño 8x8 sin importar la dimension de la imagen agregada, implementar una clase que trabaje con los pixeles que contega la imagen y retorne bits para implementarlos en la placa arduino y trasmitirlos a los leds 8x8.

Adaptar el codigo c++ trabajado en Qt al arduino y sus condiciones, ademas de tener en cuenta el tamaño de las variables para evitar lentitud en el procesamiento de los leds en tinkercad.

Leer archivos PNG en Qt, para esto se debe investigar como se trabaja la lectura y organizar estos datos de salida de tal forma que se puedan ingresar a la placa de arduino por las entradas seriales.

Crear la matriz de leds e implementar un metodo de organizacion de tal forma que se pueda trabajar con cada uno de los leds de manera acertada, verificar metodos de submuestreo y muestreo para las imagenes, estudiar algoritmos de estos metodos ateriormente mencionados.

4. IDEA DE TRABAJO DE IMAGENES

Segun la guia explicada por el porfesor en la clase, primero hay que leer la imagen deseada, analizar los datos de dicha imagen como sus dimensiones, verificar los colores de cada uno de sus pixeles determinados como red,green y blue (rgb). Luego de esto investigar o diseñar un algoritmo de submuestreo para adaptar la imagen a nuestro arreglo de tiras neopixel, en caso de que la imagen sea menor al arreglo realizar un muestreo para adaptar la imagen al arreglo para esto se deberan tener en cuenta algunos metodos de interpolacion de imagenes.

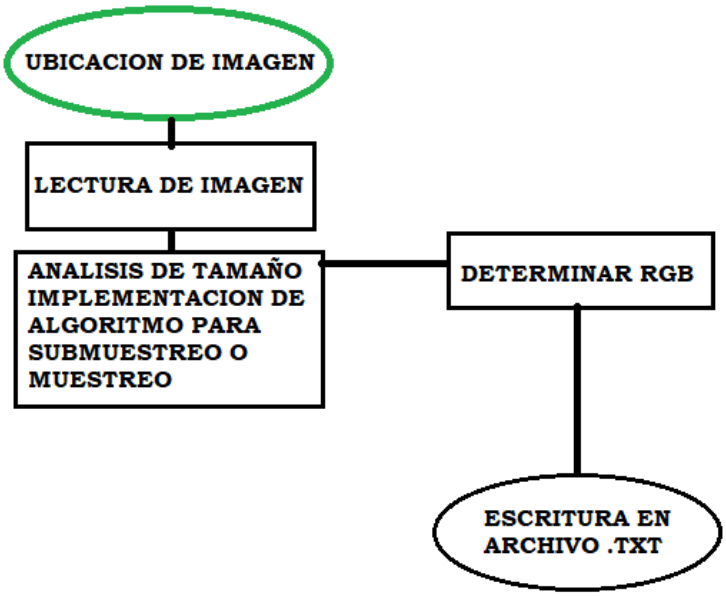

Figura 2: Idea general(Qt c++)

5. PROCESO TINKERCAD

Se creo un arreglo de 8 tiras de neopixel, cada una de las tiras contiene 8 leds para un total de 64 leds, unidas por sus respectivas entradas y salidas, ademas se conectaron a una de las entradas seriales de la placa de arduino en el pin numero 2 por recomendacion de los profesores se agrego un suministro de energia con un paso de corriente de 20A.

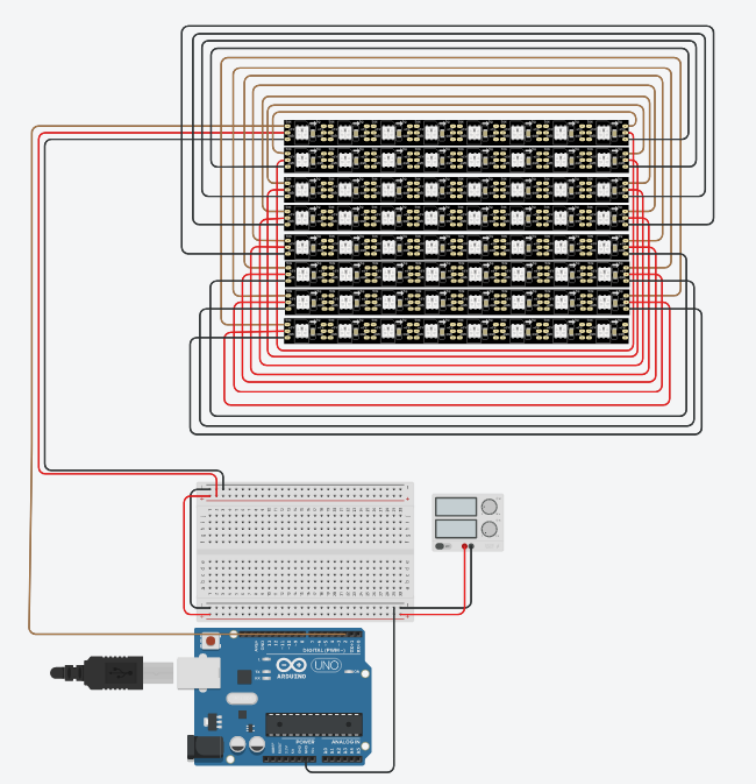

Figura 3: Orden de leds

6. CODIGO ARDUINO

Incialmente se escribio codigo en tinkercad de tal forma que al ingresar datos separados por coma se leyera en el codigo de arduino, se creo un arreglo de tamaño 192 porque cada uno de los datos ingresados llegarian en un orden de tres cada uno por los colores RGB, entonces ya que la cantidad de leds son 64 para cada led corresponde de a tres datos del arreglo, 3x64=192(tamaño del arreglo). Ademas se condiciono para que en caso de que todos los valores de RGB fueran su maximo contenido de memoria o sea 255,255,255 al menos uno se le reducia el valor para que quedara en 254.

```
int fill[192]={
//incio

  //{R,G,B}


//fin
};

#include <Adafruit_NeoPixel.h>

#define  LED_PIN 2

#define LED_COUNT 64

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

void setup(){
  leds.begin();
  for(int x=0, i=0; x < 192; i++){
    int r=fill[x];
    int g=fill[x+1];
    int b=fill[x+2];
    if(r==255 && g==255 && b==255){b--;}
    leds.setPixelColor(i,r,g,b);
    x=x+3;
  }

  leds.show();
}

void loop(){

}
```

Figura 4: Codigo de arduino

7. CODIGO QT

En el incio se trabajo con las ayudas suministradas por los profesores para acceder al tamaño de las imagenes, luego se hizo pruebas con un cout para verificar el tamaño de algunas imagenes, despues se hizo pruebas con las ayudas brindadas para obtener la imformacion RGB que contenian los pixeles de la imagen leida. A partir de lo anterior se empezo con la busqueda de algoritmo para la reduccion de una imagen a un conjunto de datos deseado, en este caso necesitabamos reducir el tamaño de una imagen (bandera de algun pais) a el tamaño de una matriz 8x8, se leyo codigo con la implementacion de interpolacion bilineal[1] en foro virtual.

```
7  int main()
8  {
9      //LECTURA DE IMAGEN
10     string filename="../P2/images/mexico.png";
11     QImage im(filename.c_str());
12
13     //IMPRIME TAMAÑO DE IMAGEN
14     cout<<im.width()<<"x"<<im.height()<<" pixels"<<endl;
15
```

Figura 5: Lectura de imagen

7.1. ARREGLOS

Despues de analizar articulos, foros y archivos multimedia[2], se intento imprimir en la terminal desde Qt, una matriz del tamaño de cada imagen lo cual se volvio complicado de trabajar, despues se intento imprimir matrices con imagenes de tamaño pequeño o cercano a la matriz solicitada 8x8, para esto empezamos con con la impresion de uno de sus datos, por ejemplo de una matriz 15x15 se imprimio los datos de cada unos de sus pixeles en color Red, despues el color Green y respectivamente su color Blue.
Ademas al ser imagenes pequeñas se pudo ingresar estos datos RGB a un solo arreglo[n][m] tipo string, gracias a esto se pudo probar algunos arreglos separados por coma en el codigo ubicado en tinkercad.

7.2. CICLOS

Despues de analizar el comportamiento extrayendo imformacion de algunas imagenes de tamaño pequeño a traves de ciclos y condicionales guiados por su ancho y altura, se decidio cortar u omitir algunas filas y columnas con la intencion de reducir su tamaño, se obtuvieron resultaron de reduccion pero ordenes incorrectos.

7.3. RECORRIDO POR PASOS

Se implemento un metodo en el cual se divide el ancho y la altura por el tamaño de la matriz o arreglo deseado en nuestro caso 8x8, al dividir el ancho entre 8 obtendremos un valor con el cual daremos pasos sobre el tamaño ariginal de la imagen sacando valores en cada paso sobre el eje X, de igual forma se divide la altura sobre 8, para despues dar pasos sobre el eje Y. Se crean variables tipo long double para evitar perder distancias al dividir los tamaños por numeros enteros y asi tener pasos mas acertados sobre las imagenes de gran tamaño.

```
//CICLO PARA REDUCIR
string matrizLed[8][8];
long double longiX=im.width();
long double longiY=im.height();

long double pasoX=0.0;
long double pasoY=0.0;

    for(unsigned int y=0; y < 8; y++){
        //cout<<endl;
            for (unsigned int x=0; x < 8; x++){
                matrizLed[x][y]=to_string(im.pixelColor(pasoX,pasoY).red())+","+
                        to_string(im.pixelColor(pasoX,pasoY).green())+","+
                        to_string(im.pixelColor(pasoX,pasoY).blue())+",";
                //cout<<matrizLed[x][y]<<",";
                pasoX=pasoX+(longiX/8);

            }
            pasoY=pasoY+(longiY/8);
            pasoX=0;
    }
```

Figura 6: Reduccion de imagen

8. ESCRITURA EN ARCHIVO DE TEXTO

La parte de esritura sobre un archivo.txt se recopilo los conocimientos de practicas anteriores donde se debe tener en cuenta la verificacion de un archivo abierto y tener en cuenta que despues de modificarlo se debe cerrar. Se creo un ciclo el cual escribia uno por uno los datos con sus respectivos RGB separados por una coma en un archivo llamado datos.txt.

```
    //ESCRITURA EN TXT
    ofstream archivo;
    archivo.open("datos.txt", ios::out);
    if(archivo.fail()){
        cout<<"No se pudo abrir archivo";
        system("pause");
        system("reset");
    }


  //ESCRIBIR EN datos.txt
    for(unsigned int y=0; y < 8; y++){
        cout<<endl;
            for (unsigned int x=0; x < 8; x++){
                cout<<matrizLed[x][y];//muestra datos en terminal
                archivo<<(matrizLed[x][y]);
            }
            archivo<<"\n";
    }

    archivo.close();
```

Figura 7: Escritura en archivo.txt

8.1. datos.txt

En la siguiente imagen se ve como quedan los datos en un archivo de texto listos para copiar y pegar en tinkercad.

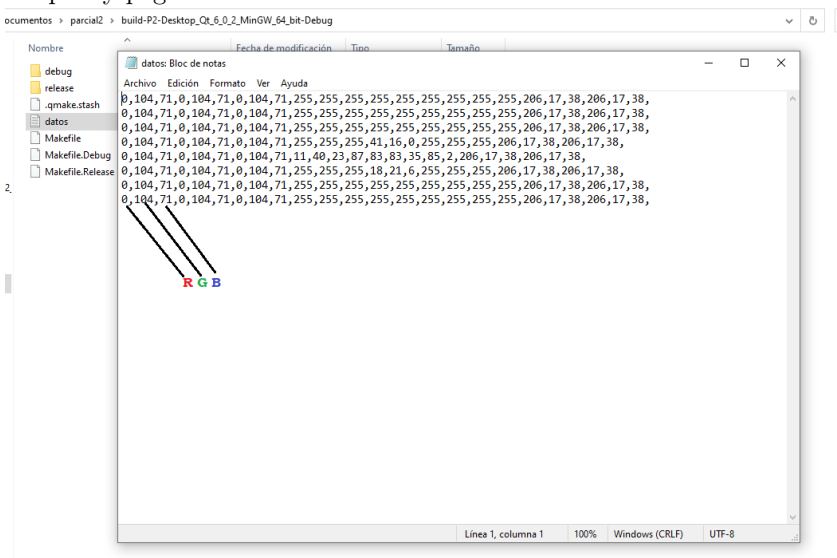

Figura 8: datos.txt

9. EJEMPLO

Aca tenemos un ejemplo de como funciona los datos en el entorno de tinkercad con la bandera de Mexico.

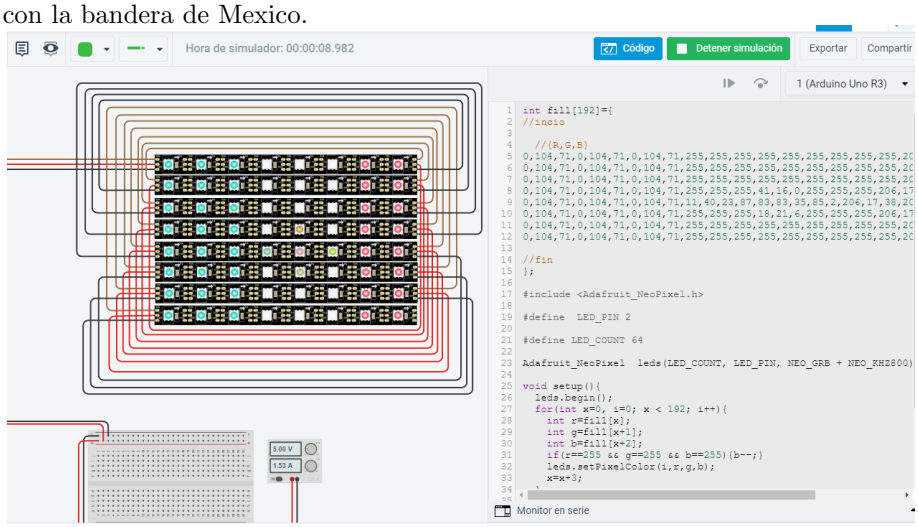

Figura 9: Ejemplo de bandera

10. DIFICULTADES

Los principales problemas que se tuvieron fueron el tiempo ya que por cuestiones laborales y academicas el tiempo se ve reducido, aunque los profes dieron buen plazo de entrega la presion por tiempo de las demas asignaturas generan un trabajo menos detallado y mas rapido. Otro problema fue analizar que metodo se usaria para reducir la imagen, en el metodo que use, al trabajar con variables long double y no tomar valores enteros, hace que el programa responda a las imagenes pequeñas tambien, aunque no de manera muy acertada.

11. CONCLUSION

Este trabajo me trajo conocimientos muy importantes sobre como funcionan las imagenes digitales, la relacion entre los colores RGB en cada pixel son de valor absoluto, ademas me genero un mapa el cual no conocia sobre los diferentes metodos que se usan en muchas plataformas para procesar y manejar las imagenes.

Referencias

[1] "https://stackoverflow.com/questions/21572460/resizing-a-picture-vc/21578107," *stackoverflow*, 2021.

[2] C. Manning, "https://www.youtube.com/watch?v=i15r3di-eyst=1sab$_channel = colinmanning.''$