

# **Informe Escrito**

Parcial 2 - Segunda parte

**Julian Taborda Ramirez**

**Samuel Ruiz Vargas**

Informatica II  
Universidad de Antioquia  
Medellín  
Septiembre de 2021

# Índice

<b>1. Clases Implementadas</b>	<b>2</b>
1.1. QT . . . . .	2
1.1.1. imagen . . . . .	2
1.1.2. QImage . . . . .	2
1.2. Tinkercad . . . . .	2
1.2.1. Adafruit NeoPixel . . . . .	2
<b>2. Esquema de las Clases</b>	<b>2</b>
2.1. imagen . . . . .	2
2.1.1. Metodo Sobremuestreo . . . . .	2
2.1.2. Metodo Submuestreo . . . . .	3
2.1.3. Metodo Guardar . . . . .	3
<b>3. Interacción de las Clases</b>	<b>3</b>
<b>4. Estructura del Circuito</b>	<b>4</b>
4.1. 22/09 . . . . .	4
4.2. 23/09 . . . . .	4
<b>5. Problemas Presentados</b>	<b>4</b>
5.1. 23/09 . . . . .	4
5.1.1. Muestreos . . . . .	4
5.1.2. Retorno de informacion . . . . .	5
5.2. 24/09 . . . . .	5
5.3. 25/09 . . . . .	5
5.4. 26/09 . . . . .	5
5.5. 26/09 . . . . .	5
<b>6. Manual de Uso Rápido</b>	<b>6</b>
6.1. QT . . . . .	6
6.1.1. Consideraciones previas . . . . .	6
6.1.2. Uso de la interfaz . . . . .	6
6.1.3. Retorno de la información . . . . .	6
6.2. Tinkercad . . . . .	6
6.2.1. Ingreso de la información . . . . .	6
6.2.2. Visualizacion . . . . .	6

## **1. Clases Implementadas**

### **1.1. QT**

Clase(s) Implementadas en QT.

#### **1.1.1. imagen**

Clase creada por nosotros. En esta clase tratamos toda la información relacionada con la imagen, además de esto realizamos las técnicas de muestreo y guardado de archivo haciendo uso de algunos métodos.

#### **1.1.2. QImage**

Clase propia de QT. Esta clase es la encargada de cargar la imagen seleccionada y además brindarnos los métodos para abstraer la información de cada pixel de la imagen.

### **1.2. Tinkercad**

Clase(s) Implementadas en Tinkercad.

#### **1.2.1. Adafruit NeoPixel**

Clase implementada por Tinkercad. Esta clase es la encargada de hacer la gestión de los Neopixel usados en el montaje, además nos brinda los métodos para configurarlos.

## **2. Esquema de las Clases**

### **2.1. imagen**

Esta clase cuenta con 2 atributos que son el ancho y alto de la imagen, los cuales son usados en el constructor, en el main y en algunos métodos. Los métodos principales de esta clase son los de muestreos, puesto que son la carta fuerte del programa al ser los encargados de hacer el redimensionamiento y entregar la representación.

#### **2.1.1. Metodo Sobremuestreo**

Comienza por sustraer toda la información de la imagen, posteriormente usamos un algoritmo basado en la multiplicación de la información. Para este proceso tomamos la información de un pixel y la repartimos entre varias entradas de la matriz 16x16 y repetimos este proceso con cada pixel de la imagen original, finalmente, usamos esta información para hacer el guardado invocando el método guardar desde el main.

### **2.1.2. Metodo Submuestreo**

Empieza por sustraer toda la información de la imagen, posteriormente usamos un algoritmo basado en el promedio para hacer el submuestreo. Para ello sacamos toda la información de un trozo de la imagen original y hacemos un promedio de todos los datos, para seguidamente entregarle la información a la matriz 16x16 y que se haga el guardado invocando el método guardar desde el main.

### **2.1.3. Metodo Guardar**

Este método recibe la información a guardar en forma de 3 arreglos bidimensionales que representan el red, green, blue de la imagen (redimensionada si es el caso) y reorganiza esta información de manera que pueda ser directamente copiada y pegada en el código de Tinkercad, finalmente guarda esta información en un .txt.

## **3. Interacción de las Clases**

La interacción entre las clases es muy sencilla. En el caso de QT la clase imagen solo tiene interacciones con la clase QImage, esto debido a que dentro de la misma se requiere poder cargar una imagen y guardar sus representaciones RGB por pixel así como saber su anchura y altura, una vez tenemos esta información no se producen más interacciones entre clases (exceptuando los tipos de datos como string, fstream, etc.).

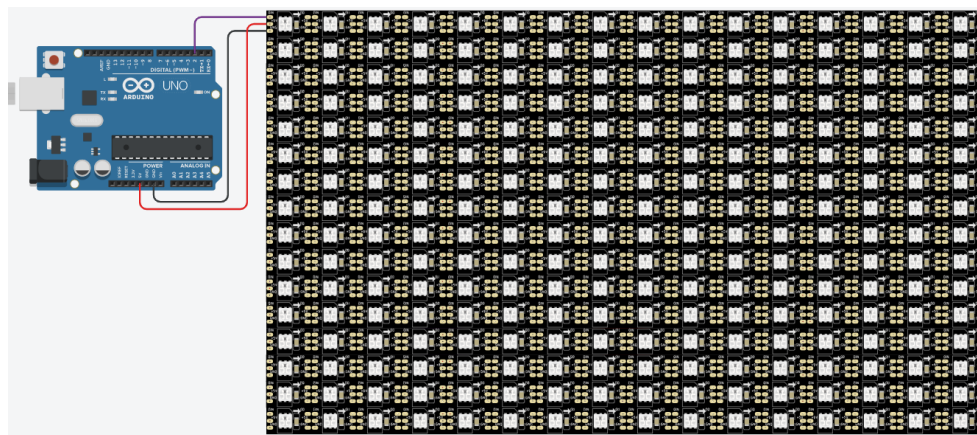
## 4. Estructura del Circuito

### 4.1. 22/09

Iniciamos con una estructura simple que consta de un Arduino uno R3 conectado a una matriz de leds formada por 11 tiras de 16 Neopixeles; Para ello conectamos el puerto digital 2, 5V y GND a la primera tira de Neopixel, posteriormente conectamos las salidas de dicha tira con la siguiente tira y repetimos este proceso con todas las tiras de Neopixel.

### 4.2. 23/09

Actualizamos la matriz de leds a una 16x16, esto debido a que queremos que la matriz sea lo más fácil de tratar posible.



## 5. Problemas Presentados

### 5.1. 23/09

En este día, luego de plantear el código en QT, nos encontramos con la siguientes problemáticas:

#### 5.1.1. Muestreos

Hasta el momento no hemos evaluado la técnica necesaria para realizar un submuestreo o sobremuestro según la imagen elegida, como su vez no hemos implementado la carga respectiva de la imagen.

### **5.1.2. Retorno de informacion**

Para esta parte del programa, aún no tenemos muy claro como vamos a entregar la información requerida por Tinkercad de tal manera que se adapte a la representación de la matriz.

### **5.2. 24/09**

En este día realizamos la parte más sencilla del código que es sacar los datos de la imagen original, nos topamos con el problema de que no podíamos introducir todos los datos en una sola matriz, puesto que en algunos casos hay demasiada información, por lo que decidimos separar los datos en 3 arreglos, uno para los rojos, otro para los verdes y el último para los azules.

### **5.3. 25/09**

Para este día se hizo la mayor parte del programa de QT, debido a esto se nos impusieron varios predicamentos, entre ellos el tratamiento en matrices, esto debido a que en algunos casos una matriz tridimensional contenía demasiada información, por lo tanto nos vimos en la necesidad de repartir la información en distintas matrices bidimensionales; Otro problema que detectamos fue la fusión de colores, a pesar de no haber hecho aun los algoritmos de submuestreo y sobremuestreo, haciendo las pruebas de representar una imagen que ya contaba con las dimensiones de nuestra matriz de leds nos dimos cuenta de que en algunos casos los colores podían llegar a verse distorsionados, sin embargo debido a la información brindada por el profesor determinamos que era algo común y correcto que ocurra en algunos procesos de redimensionamiento de imágenes.

### **5.4. 26/09**

En este día terminamos los métodos correspondientes para realizar el submuestreo y el sobremuestro partiendo de una imagen que cumpla con las condiciones iniciales, en este proceso tuvimos varios inconvenientes tales como pensar que manera podíamos hacer dichas representaciones de la forma más fiel que nos fuera posible. Además tuvimos problemas con la forma en la que obteníamos cierta información desde el constructor, el main y otros métodos, por lo tanto se reestructuraron partes del código para tratar la información de manera más eficiente.

### **5.5. 26/09**

En este último día de trabajo nos encontramos con el problema de que nuestro programa no era capaz de recibir imágenes con mucha información, por lo tanto cambiamos la estructura de los datos de arreglos bidimensionales comunes a vectores anidados, puesto que pueden contener más información sin romper el programa y con esto el problema quedo solucionado.

## **6. Manual de Uso Rápido**

### **6.1. QT**

#### **6.1.1. Consideraciones previas**

1. El usuario debe tener en cuenta que la imagen debe estar en la ubicación correcta.
2. Se le recomienda al usuario que la imagen tenga un nombre sencillo de recordar y/o escribir, puesto que se requerirá más adelante.

#### **6.1.2. Uso de la interfaz**

En la interfaz, debe ingresar el nombre de la imagen correspondiente y su respectiva extensión; Para evitar errores, en caso de que la imagen no pueda ser cargada se le indicará mediante un mensaje.

#### **6.1.3. Retorno de la información**

Una vez finalizado el programa de manera exitosa, automáticamente se creará un archivo ".txt", en la carpeta saves del proyecto, dicho txt está completamente listo para ser copiado y pega en el código .ino del Tinkercad.

### **6.2. Tinkercad**

#### **6.2.1. Ingreso de la información**

Teniendo ya copiada la información que nos fue retornada en el txt brindado por QT, procedemos a abrir el código del proyecto de Tinkercad e inmediatamente buscamos la zona designada para dicha información la cual está marcada entre dos comentarios y pegamos ahí la información, con cuidado de no dañar el resto del código.

#### **6.2.2. Visualizacion**

Habiendo seguido los pasos anteriores y suponiendo que todo se hizo de forma correcta, solo nos queda iniciar la simulación y disfrutar los colores.