INFORME 2 PARCIAL 2 INFORMATICA II

ANDRÉS FELIPE RENDÓN VILLADA
DANIEL ANDRÉS AGÜDELO GARCÍA

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

Índice

1. Introducción

La programacion orientada a objetos (POO) representa un cambio importante en el paradigma de la programacion. Pues esta facilita en gran medida la recursividad del codigo y realizar diversos procesos que con funciones serian bastante extensos y engorrosos de realizar utilizando otros recursos de progamacion. Como por ejemplo copiar y pegar n veces un mismo codigo o crear n cantidad de funciones para un determinado proposito. Por este motivo se realiza la siguiente actividad evaluativa en la asignatura INFORMATICA II de la Universidad de Antioquia , la cual consiste en aprender a realizar el muestreo y sobremuestreo de los datos contenidos en una imagen (pixeles) , los cuales hacen posible mediante la combinación de 3 colores (rojo,verde y azul), el visualizar tonalidades de cualquier color presentes en una imagen , el objetivo de esta actividad es la implementacion de algoritmo que permita realizar el cambio en la escala de una imagen para que esta pueda ser montada en una matriz de leds de un tamaño nxn, cabe aclarar que no se permite el uso de librerias que faciliten el procesamiento de imagenes , es decir el programa presentado por los estudiantes debe ser 100 porciento implementado por nosotros los estudiantes de los programas ingenieria en telecomunicaciones e ingenieria electronica.

2. Planteamiento del problema

como se explico anterior mente en el apartado de introduccion del presente informe, el ejercicio propuesto consiste en que se nos brindara una imagen de un determinado tamaño (no necesariamente cuadrada). se debe implementar un codigo en c++ , utilizando el entorno de QTcreator, el programa implementado debe estar en la capacidad de redimensionar la imagen (alargarla o contraerla segun sea el caso), y ademas tambien debe generar un archivo .txt con la posicion de cada pixel y la representacion RGB de los pixeles de la imagen escalada , la informacion contenedia en el txt debe copiarse y luego pegarse en el codigo de la plataforma de tinkercad para simular la imagen original en el montaje de los led desarrollado previamente en dicha plataforma. adicional se debe realizar, un informe , un manual de funcionamiento y un video donde se exponga el montaje diseñado.

2.1. Ideas para la resolucion del problema

Dado que para la solución del problema no podiamos hacer uso de librerias o metodos , que facilitaran el procesamiento de imagenes , tuvimos bastantes ideas para el desarrollo de la actividad , primero pensamos en tratar la imagen como si fuera un archivo .txt para obtener la informacion contenida en esta y luego reescribir esta infomracion en otro txt usando los pixeles necesarios para visualizar la imagen en la matriz de 16x16 leds. al principio esta parecia ser la idea mas optima para la resolucion del problema pero despues de analizar mucho la descartamos. ya que no sabiamos que pixeles estabamos usando para llevarlos a la matriz.

La segunda idea consistio en que debiamos encontrar el pixel de la mitad con sus respectivas coordenadas (x,y) y apartir de este pixel empezariamos a desplazarnos en diferentes direcciones hasta obtner los 256 valores asociados a los pixels que ibamos a mostrar en nuestra matriz esta idea tambien se desecho ya que si la imagen tenia algun dibujo en una esquina o fuera del area recorrida no se mostraria en la matriz de leds.

La tercer idea y la que implementamos finalmente consistio en que dividiriamos el area de la imagen en pequeñas submatrinces de nxn dimensiones, estas submatrices representarian 1 pixel en la matriz de 16x16. cabe mencionar que de cada submatriz obtendriamos la combinacion rgb y luego se promediarian estos valores para obtener la combinacion de cada color en el rango de 0-255. esta fue la opcion que implementamos finalmente , ya que no importaba donde estuviese dibujado un patron en la foto , este se mostraria en la matriz sin importar el

tamaño de la imagen.

2.2. Para el Montaje

Para el montaje usamos la plataforma de autodesk TINKERCAD la cual nos permitio realizar la simulacion de las imagenes en la matriz realizada con tiras led de neopixel, las cuales recibian la señal de voltaje y datos por medio de un arduino para su funcionamiento. sin embargo el codigo que usariamos para realizar la simulacion se realizo en el IDE QTcreator ya que en esta plataforma no es posible realizar el manejo de archivos para obtener los datos del txt donde guardariamos las combinaciones de los valores RGB asociados a cada pixel, lo cual presenta una gran desventaja a la hora de realizar la simulacion pues constantemente toca estar abriendo el archivo txt copiando la informacion contenida en este y pegandolo en la plataforma de tinkercad para poder realizar la simulacion.

2.3. Para el codigo de QT

Como trabajar directamente con una imagen de tamaño 400x400 o de 120x90 es muy dificil tanto para saber si estan sacando bien el promedio de los colores o si esta recorriendo ordenadamente, y un punto de apoyo muy importante a la hora de programar es la depuracion, todos los codigos implementados tuvieron, entonces se hicieron prototipos de funciones, estos prototipos se trabajaban con matrices mas pequeñas por ejemplo, se creaba una matriz de 12x12 donde se suponia que era la imagen, haciendo un doble for que todas las entradas las convirtiera en numeros consecutivos, al igual que se suponia que la matriz de leds era 4x4, esto con el fin de trabajar y depurar con numeros mas pequeños.

Al trabajar con estas dimensiones mas pequeñas era mas facil rayar y sacar los promedios, o volver las imagenes mas grandes, esto nos ayudaba con la depuracion y saber en que precisa posicion esta generando problemas, y ya al tener el prototipo de la funcion listo, solo era implementarlo a un tamaño mas grande con las imagenes, y deberia funcionar perfectamente.

3. Clases implementadas

El codigo funciona con una sola clase, la cual tiene 4 funciones dentro, de las 4 opciones que podrian tener la imagen. Como nuestra matriz de leds es de 16x16 puede suceder que la imagen exija solo sobremuestreo, que la imagen exija solo submuestreo, que la imagen exija submuestreo en las filas y sobremuestreo en las columnas o que la imagen exija sobremuestreo en las filas y submuestreo en las columnas.

La funcion de submuestreo en la funcion mas importante del codigo, ya que las demas funciones se basan en este. Esta funcion se trabaja la imagen como si fuera una matriz, entonces se trate de convertir una imagen con alto y ancho igual o mayor a 16 (ya que la matriz de leds es de 16x16) a una matriz de 16x16.

La funcion de sobremuestreo, esta funcion es para imagenes donde su alto y ancho son menores a 16. Tomamos una imagen pequeña y la agrandamos 16x16 veces, por ejemplo si el tamaño de la imagen es 10x10 entonces la agrandamos hasta 160x160, y luego de tener la imagen ampliada, la vamos a convertir en 16x16 con la funcion de submuestreo.

La funcion de Submuestreo y Sobremuestreo en una misma imagen, por ejemplo con imagenes tamaño 300x10. Es parecido a las dos funciones anteriores, se dejan las filas con la misma dimension y las columnas se amplian por 16 veces usando una parte del sobremuestreo, y ya despues de tener esta imagen redimensionada se aplica la funcion de submuestreo.

La funcion Sobremuestreo y Submuestreo en una misma imagen, por ejemplo con una imagen tamaño 10x300. Es parecido a las dos primeras funciones, las filas se amplian por 16 veces, o sea se hace un sobremuestreo y las columnas se mantienen con su dimension original, y ya despues se aplica a esta imagen ya redimensionada la funcion de submuestreo.

En cada una de las funciones anteriores, la imagen ya redimensionada a matrices de 16x16 que representaban el promedio de la intensidad de los colores rojo, verde y azul, lo unico que se hacia era un doble for que recorriera estas matrices, se creara/guardara en un .txt en el formato requerido para tinkercad.

4. Esquema

La estructura de las funciones de la clase se basan en dos codigos, uno el sobremuestreo funcional con imagenes y otro volver las imagenes que se necesiten mas grandes, con esas dos maneras se pueden trabajar cualquier imagen, pero dependiendo de las dimensiones de la imagen se deben tirar por un lado o por el otro.

Se hizo 4 condiciones, submuestreo, sobremuestreo, submuestreo y sobremuestreo a la vez y sobremuestreo y sub muestreo a la vez, y dependiendo de las dimensiones de la imagen entra en una de esas 4 condiciones que sera donde la redimensionara a una matriz de 16x16

El sobremuestreo lo que hace es tomar una imagen, y trabajarla como si fuera una matriz, con sus columnas o filas mayor o igual a 16, y las divide en varias submatrices, y el tamanio de las submatrices es la division del altoXancho de la imagen con 16, y a esas submatrices se le saca el promedio de la intencidad del rojo, verde y azul, y al tener este promedio se guarda en las posiciones de otras matrices creadas de dimension 16x16, y ya al final al tener el promedio de los colores rojo, verde y azul, y guardarlo en matrices, ya para que imprimiera y generara el .txt solo es con un doble for que recorra toda la matriz de rojo, verde y azul e imprimera en el formato correspondiente.

El submuestreo lo que hace es tomar la imagen pequeña y redimensionarla en submatrices de 16x16 leds, cambiando su dimension a alto*16Xancho*16, convirtiendo la imagen mas grande, entonces un solo pixel de esta imagen se multiplicaria 16*16 veces, haciendo la imagen grande, y ya al tenerla de tamaños mas considerables se hacia todo lo mismo en la funcion de sobremuestreo.

El submuestreo y sobremuestreo en una misma imagen, es la implementacion de las dos funciones anteriores pero con la condicion de que filas sea mayor o igual a 16 y que las columnas sea menor a 16, en este caso se hace los for, donde las filas se mantienen con sus dimensiones pero las columnas las multiplico por 16, anchando la imagen, ejemplo una imagen 100x10, quedaria redimensionada a 100x160, y esta imagen ya redimensionada se aplica la funcion de sobremuestreo.

El sobremuestreo y submuestreo en una misma imagen, es lo mismo que la funcion descrita anteriormente pero cambiando filas y columnas, en este caso serian imagenes como por ejemplo de tamaño 10x100, entonces las filas se

ampliarian, multiplicandolas por 16 y las columnas mantienen su dimension, redimensionando la imagen a 160x100, e igualmente que se ha estado mencionando, al tener esta imagen de tamaño mayor o igual a 16x16 se aplica la funcion de submuestreo

5. Dificultades para la solución del problema

Algunos de las dificultades que presentamos al momento de realizar la solución de la actividad , fue la poca información que encontrabamos acerca del tema , en plataformas como youtube el material que encontrabamos era bastante escaso y en su mayoria estaba en ingles por lo que muchos conceptos no quedaban bastante claros.

Por otra parte al momento de realizar el montaje debimos buscar la documentación de la libreria que permitia encender los leds de las tiras de neopixel y tambien para entender como se debian conectar las tiras entre si para conseguir formar la matriz.

Tambien debimos estudiar los metodos asociados a la clase QImage para obtener la idea de como podriamos realizar el escalamiento de la imagen al tamaño que necesitabamos para la matriz sin utilizar estos metodos.

La manera como imprimia la redimension de 16x16 en tinkercad en un pincipio nos genero espejo, por lo que fue necesario revertir la direccion de los neopixeles al igual que cambiar un orientacion.

En la forma de implementar el sobremuestreo, se estaba haciendo un codigo nuevo que trabajara en la ampliacion de imagenes hasta llegar a 16x16, pero con imagenes como 7x9 es dificil convertirlas a 16x16 ya que al dividirlos daria un numero con decimales, y al ser una imagen tan pequeña cada pixel era importante y contada, generando muchos errores y mucho tiempo perdido en la solucion de esta forma.

Tambien hubo complicaciones en el submuestreo y sobremuestreo en una misma imagen y viseversa, ya que el formato con las que trabajan las imagenes es por ejemplo 390x450, siendo 390 el ancho de la imagen y 450 el alto de la

imagem, o sea, conceptos intercalados a como es una matriz, que asi fue como se trabajaron las imagenes.

Otro problema fue cuando los leds deberian encender totalmente blancos, en este caso algunas promedios eran 256 pero si el neopixel registraba 256,256,256 habia que restarle uno a cualquier color, pero al implementarlo esto en la funcion y siempre ir restando de a uno, y ya un pixel es 0 entonces quedaria en -1, y el -1 como forma de entrada en el neopixel hace que el led se ponga a cambiar de colores.

6. Algoritmo

En la Figura (1), se añade el diagrama de flujo del algoritmo empleado para la solucion del problema.

7. Citación

El video de **Ferrabacus** [**?**]. fue de gran ayuda para entender como podriamos realizar el modelamiento de la imagen en un tamaño mas reducido.

Tambien planeamos usar interfaz grafica usando el video de **Cunrakes** [**?**]. pero obtamos por hacerlo simplemente usando lineas de codigo en una aplicacion de c++

Para la parte del control de los leds usamos la información de la libreria adafruit neopixel de **ARDUINO** [**?**]

8. Inclusión de imágenes

En la Figura (2), representación grafica de los pixels formando la bandera de Colombia.

En la Figura (3), representación del esquema del montaje implementado en tinkercad.

En la Figura (4), representación del codigo con las clases implementado en qtcreator.

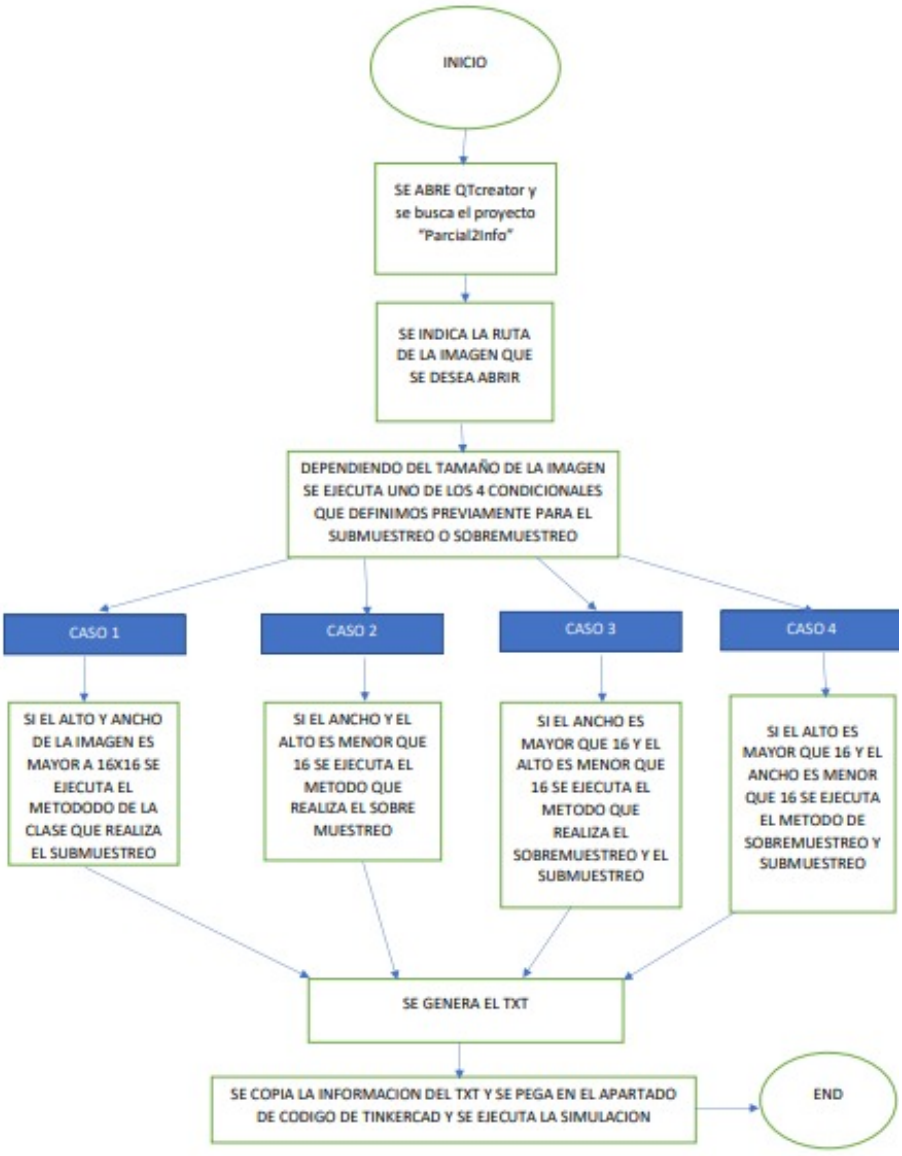

Figura 1: Algoritmo

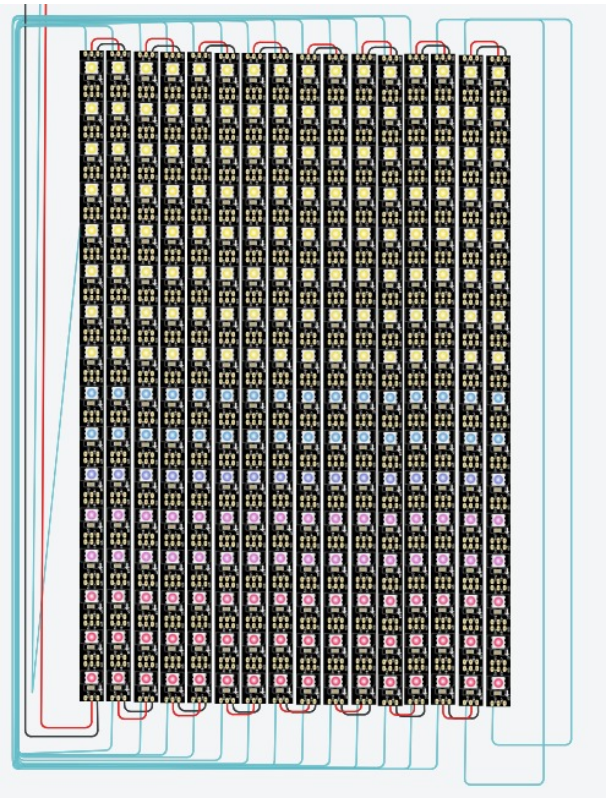

Figura 2: prueba de la matriz

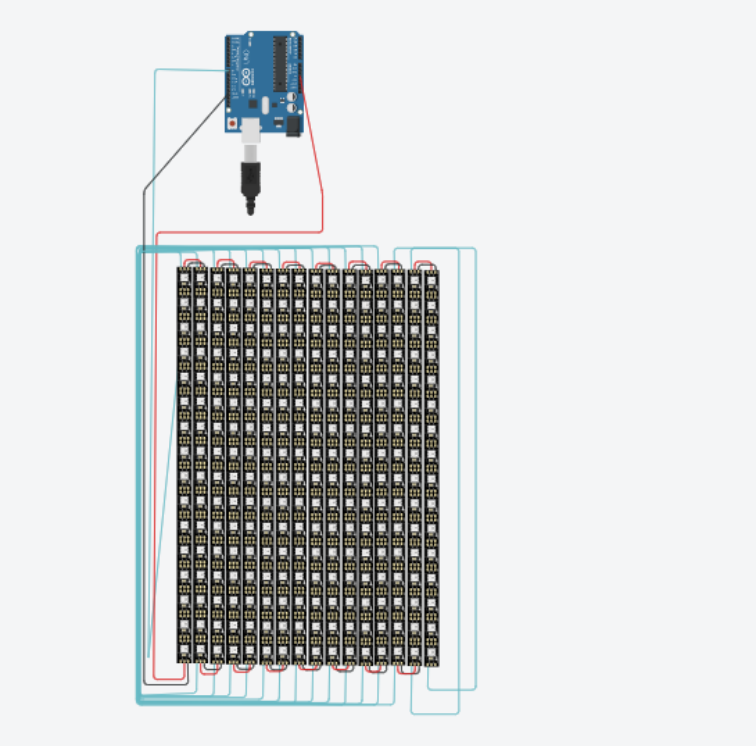

Figura 3: Montaje

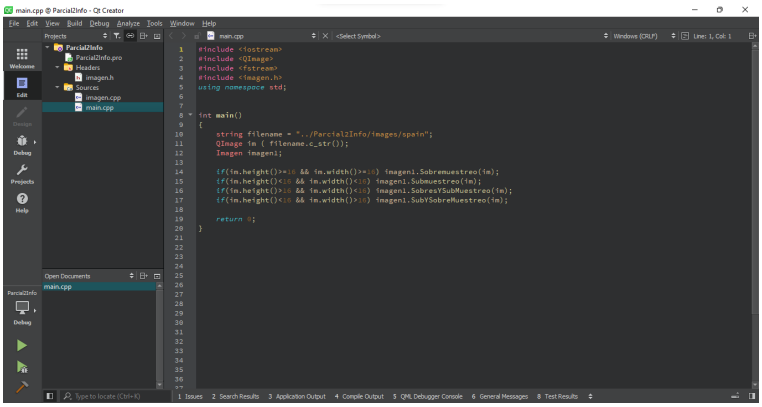

Figura 4: implmentacion qt

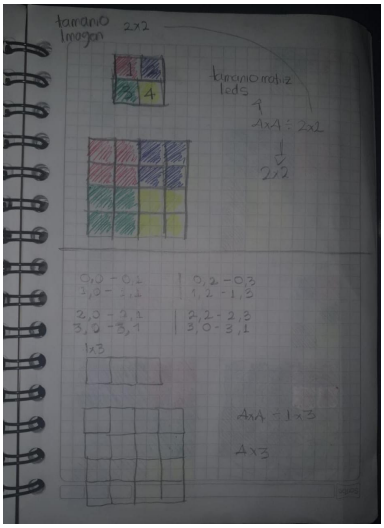

Figura 5: analizis del problema, Sobremuestreo prototipo

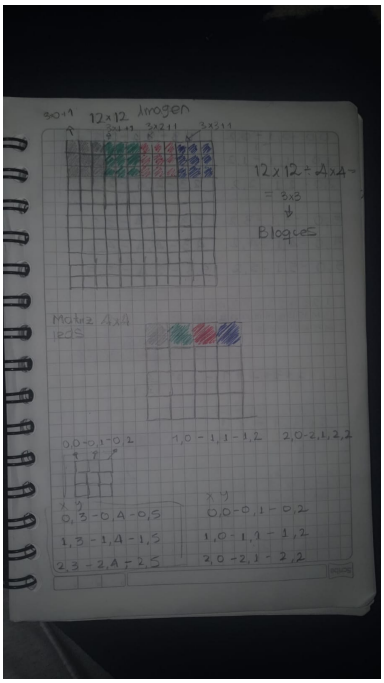

Figura 6: analizis del problema, Submuestreo prototipo

Figura 7: analizis del problema, Sub y sobremuestreo prototipo

Figura 8: analizis del problema, Sobre y submuestreo prototipo