PARCIAL 2

Informe del desarrollo del proyecto de Informa2
S.A.S.
Análisis y diseño.

Juan Fernando Muñoz López.
Carlos Daniel Lora Larios.

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

Índice

1. Análisis del problema y consideraciones para la alternativa de solución propuesta.

1.1. Primeras impresiones sobre el problema:

Para este nuevo reto, partimos con una muy significativa experiencia del último proyecto desarrollado para Informa2 S.A.S, en el que tres desarrolladores se juntaron para que cada uno fuera un pilar fundamental de ese producto final del que nos sentimos muy orgullosos, y por cierto, del cual obtuvimos muy buenas apreciaciones por parte de los encargados de Informa2 S.A.S, donde efectivamente cumplimos el reto en todos los aspectos con creces; de esa experiencia de trabajo nos quedaron muchos aprendizajes y aspectos en los cuales mejorar, con esa misma motivación del primer proyecto, partimos para este en donde queremos seguir evolucionando como equipo, desarrolladores en la praxis del conocimiento adquirido y por mejorar.

Nuevamente, Informa2 S.A.S, nos pone en la mesa un reto en el cual debemos integrar tanto el desarrollo de Software y Hardware, aspecto del cual hemos procurado seguir progresando y potencializar para la vida laboral. En este caso el reto es el de procesamiento de imágenes, redimensión de las mismas para su posterior visualización a través de entorno de desarrollo de Hardware Arduino.

A priori, parece sencillo, pero una vez hemos analizado los diferentes factores que rodean la posible solución de este reto, hemos concluido que es un reto que nos va demandar una gran astucia e imaginación para dar una solución viable y asequible para los usuarios de este sistema de muestreo de imágenes digitales que se quieren para la visualización de banderas para los juegos olímpicos de París 2024. Una vez analizado esto es como dábamos inicio a nuestra propuesta de solución para este reto, en el que a manera de bitácora a través de un repositorio evidenciaremos la evolución de nuestras apreciaciones para ello.

1.2. Análisis del problema.

Posterior a una breve interacción con la librería QImage, análisis del manejo de imágenes y su información a través de la misma, además de una breve discusión y pertinente retroalimentación de ideas. Consideramos que ya tenemos ciertas ideas base las cuales serán las directrices en la consecución del desarrollo.

Para el cual queremos mantener ese algoritmo sencillo de trabajo que se enunciaba a groso modo en los inicios de este informe “procesamiento de imágenes, redimensión de las mismas para su posterior visualización a través de entorno de desarrollo de Hardware Arduino”, pero obviamente intentando desarrollar cada uno de estos puntos de manera clara y metodológica para que los objetivos sean alcanzados.

Nuestra consideración general es la brindar al usuario una experiencia cómoda, sencilla y eficiente. Por lo que hemos pensado que desde el primer momento el trabajo de usuario sea mínimo bajo un gran trabajo de desarrollo de nuestra parte.

Por ello, es que hemos analizado que como se nos propone en el reto para dar inicio desde el entorno de Qt se le pida al usuario por consola, a través de un pseudo menú, la ruta en la que se encuentra la imagen que se desea procesar, una vez pasado esto creemos que es pertinente que la imagen sea tratada como una clase u objeto, en el que además de tener su procesamiento a través de la librería QImage, con la cual se importará la imagen para acceder a la información base de la misma, para que de esta manera tenga ciertos métodos y atributos cuenta de la información de la misma, para posteriormente deba ser llevada a la implementación de la plataforma Tinkercad. Nuestra apreciación es que dentro de sus atributos debe conocer cuáles son sus dimensiones en la unidad de pixeles (alto, base), además hemos considerado que es necesario generar una clase que se derive de esta, como si tratase de un pixel o un led de la matriz de la implementación de Tinkercad, puesto que hemos generado un esquema sobre el tratamiento de la información que esperamos desarrollar en el proyecto, en donde generar esta clase "hijo" es importante para la ejecución de las otras partes.

Para la implementación de Tinkercad hemos en principio que la matriz tenga un tamaño de 16*16, pues consideramos a través de un breve análisis de eficiencia vs estética que a partir del redimensionamiento de las imágenes ya sea a través del sub muestreo y el sobre muestreo; con estas dimensiones se representará de una manera más que aceptable la información contenida en la imagen base.

La clase "hijo" de la que hemos definido para la imagen consideramos tendrá una gran importancia, pues la imagen tendrá un arreglo con 16*16 elementos de esta clase derivada, cada una de estas representará 1/16 del área total de la imagen, donde ésta deberá saber que valor de color tienen los pixeles de esta área, compararlos, y definir cuál es color más predominante para que la clase "padre" tenga otro vector o arreglo dinámicos de un tamaño de 16*16 donde cada elemento corresponderá en paralelo a pixel o led RGB de la matriz de 16*16 de la implementación de Tinkercad.

Para que esto suceda, una vez se conozcan las dimensiones de la imagen con ayuda de la librería QImage, se determinará a través de un método de la clase que tipo de ejecución se debe de hacer si un sobre muestreo o un sub muestreo de la misma.

-Submuestreo: Para este proceso, el cual se debe ejecutar una vez se analice que la imagen supere las dimensiones de nuestra matriz de implementación de Tinkercad de 16*16, para ello hemos pensado en como en el siguiente ejemplo en cual se muestra una imagen que es la bandera de Colombia con un tamaño de 200*133 pixeles, se hace una división de la imagen de 16*16, donde a cada región le pertenece un espacio de 1/16 del ancho y un 1/16 del alto de la imagen total, precisamente este espacio de área es el que le corresponderá a la clase derivada de la imagen a la que habíamos referenciado como pixel o led RGB. Posteriormente esta analizará los colores de los pixeles, los comparará y los contará para determinar cuál es color en términos de sus componentes RGB más predominante en el área. Esta información será contenida por el arreglo o vector dinámico de la clase imagen que será la información que posteriormente sea llevada a la implementación de Tinkercad.

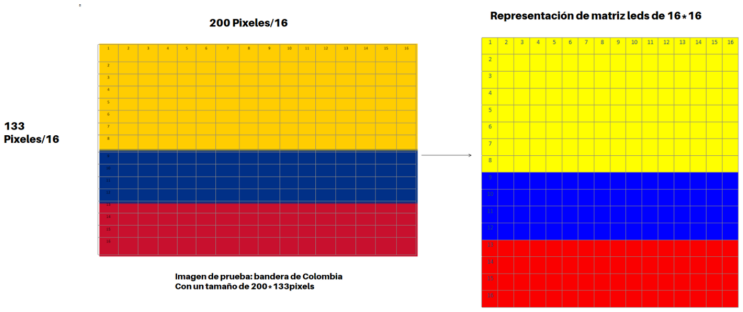

Figura 1: Ejemplo de submuestreo.

-Sobre muestreo:

Para este caso hemos analizado dos posibles soluciones, que nos ayuden a realizar este proceso. Una más eficiente que otra, la primera de ellas consiste en hacer un proceso similar submuestreo, en donde hacemos el mismo proceso de división de la imagen de 16*16. Para hacer el análisis de que, si por ejemplo hay una imagen de n*m pixeles, con n y m ¡=16, lo que se hará es que si por ejemplo la clase de las áreas quedan de 0.n*0.m, entonces cada una de estas áreas de un mismo pixel, serán de ese color para su representación en la matriz.

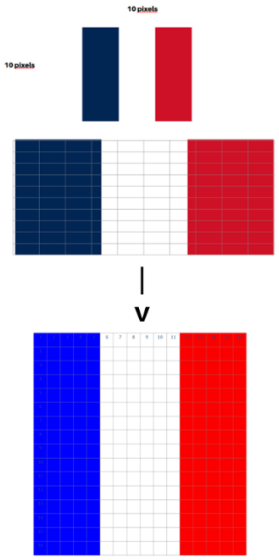

Figura 2: Ejemplo de sobremuestreo.

La otra versión demanda mucho más trabajo computacional, que creemos es innecesario, si esta versión tras ciertas modificaciones es la que nos llevará a dar por efectivo el proceso. Una vez realizado el proceso de submuestreo o sobre muestreo, como habíamos comentado, la clase de la imagen, tendrá una vector o arreglo dinámico el cual contendrá la información que será necesaria para encender los leds de la matriz RGB, con los colores correspondientes a la imagen. La información de este arreglo o vector dinámico, será organizada a través de una función para posteriormente ser escrita en un archivo .txt, para de esta manera abrirlo copiar esta información en el código de la implementación de Tinkercad, para de esta manera mostrar la representación de la imagen en nuestra matriz, y así culminar el proceso de ejecución.

2. Esquema donde describa las tareas que usted definió en el desarrollo del algoritmo.

2.1. Interacción con Tinkercad en función del manejo de leds RGB.

Empezamos intentando entender cómo funcionaba el Neopixel junto sus comandos en el código, decidimos entonces usar una multiplexacion de Neopixeles para realizar esta tarea, al principio decidimos probar con una matriz de Neopixeles 8x8 pero queríamos que nuestro código fuese algo más accesible y sencillo de entender entonces optamos por una matriz de 16x16(valor que aun podríamos cambiar dependiendo de la complejidad de la implementación),hicimos las respectivas conexiones y le pusimos un suministro de energía conectado a la protoboard para que no hubieran problemas de falta de corriente y que pudieran prender todos los leds, verificamos que todo funcionara colocando algunos valores para el comando de setPixelColor() para ver de qué manera y en qué orden nos prendían los neopixeles y así tener ideas sobre cómo controlarlos.

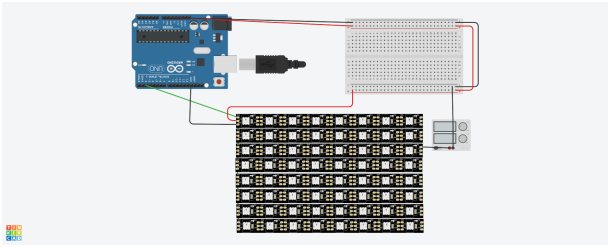

Figura 3: Primera matriz de 8*8.

2.2. Interacción en el entorno de desarrollo QT con imágenes importadas a través de la librería Qimage

Estuvimos experimentando un poco con lo que es la librería de Qimage con la que importamos imágenes de prueba que fueron banderas de países como Colombia, con el fin de hacer pruebas y encaminar el algoritmo hacia lo que nosotros queríamos. Creamos 3 archivos txt, en el que cada uno de ellos representaba los valores RGB de la bandera por pixeles, también intentamos crear dos objetos: uno para la imagen y el otro que representa el área para un pixel. Todo esto con el objetivo de llevarlo todo a términos de una matriz de 16x16, con lo cual consideramos que sería más fácil para llevarlo a los neopixeles para que muestren la imagen proporcionada.

Encontramos que también el hecho de trabajar con la librería de Qimage era algo más intuitivo y manejable.

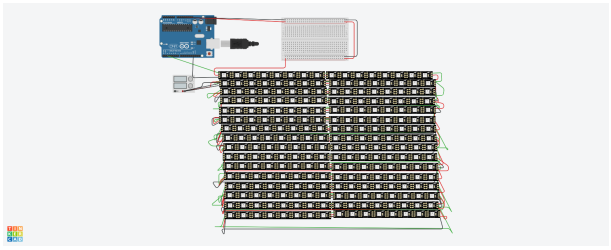

Figura 4: Primera matriz de 16*16.

2.3. Retroalimentación del conocimiento y pruebas.

Nos hemos estado reuniendo para debatir las diferentes soluciones planteadas para la problemática y así dividirnos el trabajo partiendo de varias ideas y análisis que hemos hecho mirando las imágenes, una vez que nos dividimos el trabajo y lo desarrollamos intentamos hacer una retroalimentación para estar al tanto de lo que hizo cada uno y compartir más ideas que fueron surgiendo en el transcurso de este proyecto, cuando nos reunimos nos apoyamos de herramientas para hacer diagramas, dibujos o cosas que nos ayuden a asimilar más fácilmente algunos conceptos que en un principio nos parecen abstractos o confusos. También realizamos pruebas a medida que vamos avanzando tanto en el código como en las interacciones con Tinkercad con el fin de que todo vaya encaminándose al objetivo que queremos alcanzar.

3. Algoritmo diseñado.

En base a lo ya explicado en el análisis nuestra propuesta de algoritmo para la solución se denota a continuación.

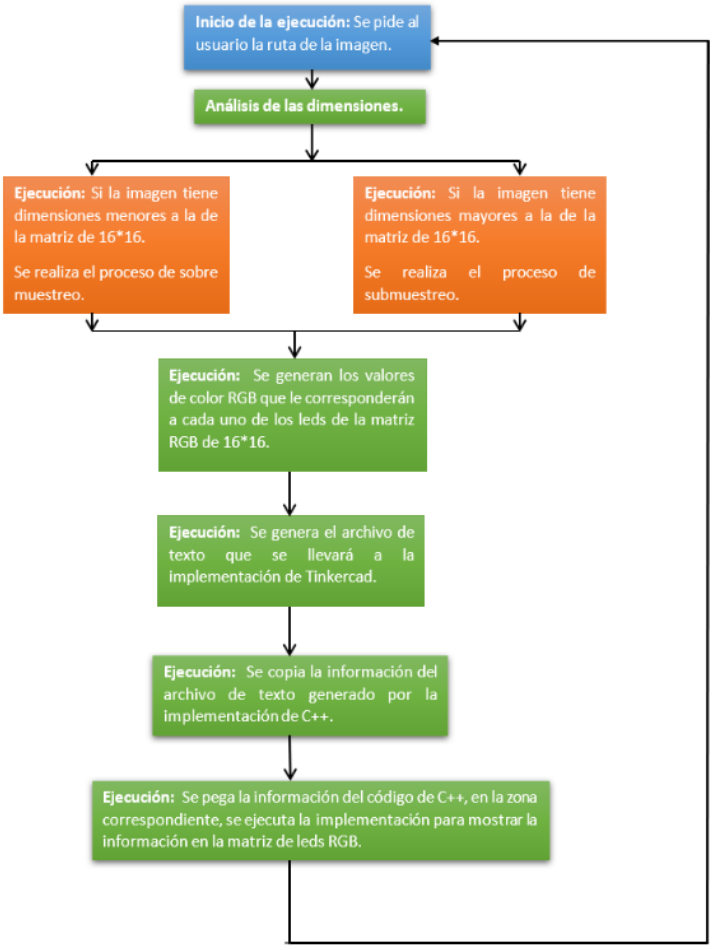

Figura 5: Algoritmo implementado.

4. Consideraciones a tener en cuenta en la implementación.

Es importante notar, que a medida de que se va desarrollando la implementación con sus respectivas pruebas, analizando los pros y contras de cada resultado, se harán las respectivas modificaciones de los elementos de nuestra estructura de trabajo que veamos pertinentes, como el tamaño de nuestra matriz de leds de 16*16 de Neopixels, clases implementadas, el concepto del submuestreo y sobre muestreo porque son ámbitos de los que apenas estamos aprendiendo es por eso que este informe de nuestro análisis del problema, hará las veces de mapa y guía en esta aventura en fuertes aguas en alta mar que apenas comienza.