

QRGB++: App sustentable, solidaria e innovadora para aumentar la densidad de información acumulada en un Código QR

Ibar Federico Anderson ⁽¹⁾

Resumen: El proyecto QRGB++ consiste en el desarrollo de una aplicación de Códigos QR, donde el script o código fuente está desarrollado en el software Python para utilizar el sistema de colores RGB, con el fin de generar un Código QR con mayor capacidad de almacenamiento de información, eficiencia en el uso de recursos y un impacto positivo en el medio ambiente. Este innovador método permite almacenar hasta tres veces más información en el mismo espacio, al combinar tres códigos QR (rojo, verde y azul) en uno solo, mejorando también la seguridad de los datos mediante la redundancia.

Además de optimizar el uso del papel y la tinta (aunque todavía no se ha desarrollado el sistema de escáner para su lectura impresa en formato papel), a futuro, contribuiría potencialmente a la reducción de residuos, deforestación y huella de carbono, alineándose con el ecodiseño y las tendencias sostenibles. Actualmente, su desarrollo está pensado solo para su uso y distribución en soporte digital.

El código QRGB se genera mediante librerías de código abierto como `qrcode[pil]`, `Pillow` y `opencv-python`, integrando una interfaz gráfica (GUI) con `Kivy` para facilitar la creación y decodificación de los códigos. Esta accesibilidad técnica promueve la inclusión tecnológica y la democratización del uso de herramientas avanzadas, permitiendo que diseñadores con conocimientos básicos de programación puedan utilizarlo gratuitamente. Por ello, es solidario y se ha desarrollado con un ejemplo de aplicación para la Universidad de Palermo y sus diversos enlaces de la Facultad de Diseño y Comunicación.

Desarrollado con el software Python como intérprete del script, este corre en la plataforma de `Visual Studio Code`. Superados los desafíos técnicos con la mezcla de colores, QRGB++ ofrece un aumento significativo en la densidad de datos, con potenciales aplicaciones en seguridad y almacenamiento de alta densidad. Su uso en proyectos sociales y educativos mejora la accesibilidad para personas con discapacidades, fomenta la creatividad y la solidaridad, y refuerza la sostenibilidad en todas las etapas del ciclo de vida del producto, demostrando cómo la tecnología y el diseño pueden converger para generar un impacto positivo en la sociedad.

Palabras clave: App - Códigos QR - Sistema aditivo - colores RGB - Script Python - IDE Visual Studio Code

[Resúmenes en inglés y portugués en las páginas 33-35]

⁽¹⁾ **Ibar Federico Anderson.** Secretariado de Ciencia y Tecnología, Departamento de Diseño Industrial, Universidad Nacional de La Plata, Argentina.

Introducción

El presente artículo describe la aplicación innovadora QRGB++, que utiliza el sistema de colores RGB para generar códigos QR con mayor capacidad de almacenamiento, eficiencia en el uso de recursos y un impacto positivo en el medio ambiente. Esta herramienta tiene múltiples conexiones con la agenda del futuro enfocada en sostenibilidad, innovación y triple impacto. Desde la perspectiva de la sustentabilidad y el reciclaje, QRGB++ reduce la necesidad de imprimir múltiples códigos QR, disminuyendo el consumo de papel y tinta, lo que reduce la generación de residuos y el uso de recursos naturales. Esto se alinea con la prioridad del cuidado del medio ambiente y el reciclaje mencionada en la frase. Además, el enfoque innovador de esta aplicación combina tecnología, sustentabilidad y ecodiseño, lo que demuestra un impacto positivo en la sociedad. Al aplicar el método 3-en-1 mediante el uso de capas de color (rojo, verde y azul), se incrementa la densidad de información acumulada, permitiendo almacenar tres códigos QR en uno solo. Esta optimización no solo contribuye a la sostenibilidad ecológica al reducir el uso de papel, sino que también está en sintonía con el ecodiseño, una corriente avanzada de los países desarrollados, que busca evitar el consumo excesivo de recursos.

Por otro lado, la innovación tecnológica de QRGB++ radica en su implementación mediante Python, utilizando librerías de código abierto como Kivy, Pillow, qrcode[pil] y opencv-python, lo que permite su acceso gratuito a cualquier diseñador con conocimientos de programación. Este aspecto refuerza el impacto positivo en la sociedad, ya que promueve la inclusión tecnológica y la democratización del acceso a herramientas avanzadas de generación y manipulación de códigos QR. El script corre en Visual Studio Code, lo que facilita su uso mediante una interfaz gráfica intuitiva. En resumen, QRGB++ no solo representa una innovación técnica en la densificación de la información, sino que también refuerza la sostenibilidad al reducir el consumo de recursos, fomenta la creatividad y se alinea con las corrientes de diseño más evolucionadas, integrando tecnología, ecología y responsabilidad social.

El presente trabajo, titulado QRGB++, consiste en el desarrollo de una aplicación en Python para la generación de códigos QR mediante la técnica de generación aditiva de colores (RGB). Este método innovador utiliza tres capas de colores –rojo, verde y azul– cada

una representando un conjunto de datos distinto, lo que permite almacenar hasta tres veces más información en el mismo espacio en comparación con los códigos QR tradicionales en blanco y negro. Al combinar tres códigos QR en uno solo mediante colores, se incrementa la capacidad de almacenamiento y se mejora la seguridad de la información, dificultando la falsificación y manipulación, y permitiendo la implementación de redundancia que aumenta la robustez frente a daños o errores de lectura.

El uso de códigos QRGB también tiene un impacto positivo en la sostenibilidad y el ecodiseño, al reducir la necesidad de imprimir múltiples códigos, disminuyendo el uso de papel y tinta. Esto contribuye a la reducción de residuos, el ahorro de recursos naturales, y la disminución de la demanda de papel, lo que a su vez reduce la deforestación y el consumo de agua y energía en la producción de papel, resultando en una menor huella de carbono y un impacto reducido en los ecosistemas. Además, al reducir el volumen de papel transportado, se disminuye la emisión de gases contaminantes durante el transporte. Los códigos QRGB optimizan el uso del espacio en los diseños impresos, permitiendo la creación de materiales más compactos y eficientes, promoviendo un enfoque minimalista y funcional, especialmente cuando se combinan con papeles reciclados o sostenibles. Su capacidad para extender la vida útil de un producto impreso, al no necesitar ser reemplazado con frecuencia, es relevante en campañas informativas que se actualizan constantemente. Integrar QRGB en campañas de sostenibilidad educa sobre la importancia de reducir el consumo de papel y otros recursos, destacando el papel de la tecnología y el diseño en la protección del medio ambiente.

Desde el punto de vista de la programación, el script en Python para la generación de códigos QRGB utiliza varias bibliotecas clave: `qrcode[pil]` para la generación de códigos QR, `Pillow` para la manipulación de imágenes, y `opencv-python` para el procesamiento avanzado de imágenes. La implementación incluye una interfaz gráfica de usuario (GUI) creada con `Tkinter`, que permite al usuario generar y decodificar códigos QR con logotipos superpuestos, así como combinar imágenes QR de diferentes colores. El segundo script está diseñado para ser más modular y extensible, incorporando bibliotecas clave como `Kivy` para la interfaz gráfica, `Pillow` para la manipulación de imágenes, `qrcode[pil]` para la generación de códigos QR, y `opencv-python` para el procesamiento avanzado de imágenes. Para instalar las bibliotecas necesarias, se deben utilizar los siguientes comandos: `pip install kivy`, `pip install pillow`, `pip install qrcode[pil]`, y `pip install opencv-python`. El script facilita la selección de archivos mediante `FileChooserIconView` y la apertura de enlaces en el navegador, lo que aumenta la flexibilidad y la funcionalidad de la aplicación.

Además, el desarrollo y la ejecución del código se realiza utilizando `Visual Studio Code`, un entorno de desarrollo integrado (IDE) que facilita la escritura, depuración y ejecución del código Python de manera eficiente. Este IDE ofrece herramientas integradas y extensiones que permiten una programación más efectiva y un manejo adecuado de las bibliotecas utilizadas en el proyecto. A pesar de enfrentar problemas con la mezcla de colores y la recuperación precisa de la información, el método propuesto demuestra un aumento significativo en la densidad de datos dentro de un solo código QR. El trabajo futuro se centrará en optimizar el algoritmo y explorar aplicaciones potenciales en seguridad de datos y almacenamiento de información de alta densidad.

Los códigos QRGB no solo representan una innovación tecnológica, sino que también tienen un impacto significativo en el diseño ecológico y la accesibilidad. Su implementación en proyectos de diseño social puede mejorar la accesibilidad al proporcionar información de manera visualmente atractiva para personas con discapacidades, facilitar la educación en comunidades vulnerables y fomentar la creatividad y solidaridad en proyectos artísticos o campañas de sostenibilidad. Al relacionar QRGB con creatividad solidaria, se demuestra cómo la tecnología y el diseño convergen para crear un impacto social positivo, alineándose con los principios del ecodiseño y maximizando la sostenibilidad a lo largo del ciclo de vida del producto o servicio.

Componentes del Código QR

El código QR, abreviatura de “Quick Response”, es un tipo de código de barras bidimensional que fue desarrollado en 1994 por la empresa japonesa Denso Wave. Su propósito inicial era rastrear piezas en la industria automotriz, pero su capacidad para almacenar grandes cantidades de datos y su facilidad de escaneo ha llevado a su adopción en una variedad de sectores, desde la publicidad hasta la gestión de inventarios. Los códigos QR se han convertido en una herramienta poderosa para la codificación y transferencia rápida de información.

Los códigos QR están compuestos por varios elementos clave que permiten su funcionamiento efectivo. Los Finder Patterns son tres grandes cuadrados ubicados en las esquinas del código QR que ayudan a los escáneres a identificar y orientar el código correctamente. Estos patrones proporcionan una referencia para alinear el código durante el escaneo. Los Alignment Patterns son pequeños cuadrados adicionales que se encuentran dentro del código QR y que ayudan a ajustar el escaneo en caso de que el código esté inclinado o distorsionado, asegurando que la información se pueda leer correctamente incluso si el código está en un ángulo. Los Timing Patterns son líneas en zigzag que se extienden entre los Finder Patterns y que determinan la anchura de los módulos, facilitando la interpretación precisa de los datos almacenados.

El Data Área es la sección del código QR donde se almacenan los datos codificados. Esta área puede variar en tamaño dependiendo de la capacidad de almacenamiento requerida. Adicionalmente, el Error Correction es un componente crucial que incluye secciones de información adicional destinadas a recuperar datos si el código está dañado. Este mecanismo permite que el código QR siga siendo legible a pesar de daños parciales, garantizando así una mayor fiabilidad en la transferencia de datos.

En resumen, los códigos QR son una herramienta versátil y eficiente para la transferencia de información, gracias a su capacidad para almacenar una gran cantidad de datos y su facilidad de escaneo. Su diseño estructurado, que incluye patrones de localización, alineación, temporización y corrección de errores, asegura una alta precisión en la lectura y recuperación de datos, lo que los hace útiles en una amplia gama de aplicaciones y sectores.

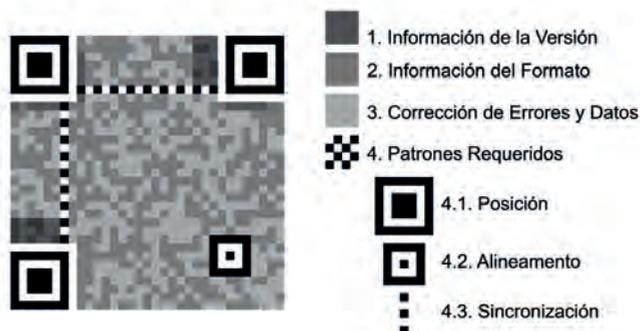


Figura 1. Código QR como convencionalmente se lo conoce. Fuente: https://commons.wikimedia.org/wiki/File:C%C3%B3digo_QR_Ejemplo_de_Estructura.svg

Hipótesis del funcionamiento del código QRGB

El principio de funcionamiento del código QRGB se fundamenta en el modelo aditivo de colores RGB (Red, Green, Blue); ampliamente utilizado en sistemas digitales para la generación de colores. En este modelo, la combinación de luz roja, verde y azul en diferentes intensidades permite representar una amplia gama de colores visibles. Este mismo principio se utiliza en el diseño del QRGB, donde cada componente de color, rojo, verde y azul, representa una capa de datos independiente.

En el QRGB, tres códigos QR separados se generan utilizando los colores primarios del sistema RGB: uno rojo (255, 0, 0), uno verde (0, 255, 0) y uno azul (0, 0, 255). Posteriormente, estas capas de color se combinan en una sola imagen mediante superposición aditiva. Cada capa mantiene su integridad; lo que significa que, al extraer una capa específica del código combinado, se puede recuperar la información original asociada a ese color.

Además de los colores primarios, en el entrecruzado de las capas rojo, verde y azul, se forman los colores secundarios: cian (Cyan), que se genera mediante la combinación de verde y azul, con valores RGB (0, 255, 255); magenta, que surge al combinar rojo y azul, con valores RGB (255, 0, 255); y amarillo (Yellow), que es el resultado de la combinación de rojo y verde, con valores RGB (255, 255, 0).

La superposición de las tres capas, rojo, verde y azul, crea el color blanco (255, 255, 255); mientras que la ausencia de superposición de cualquier capa genera el color negro (0, 0, 0). En el contexto del QRGB, los colores blanco y negro cumplen roles específicos. El blanco (255, 255, 255) representa los cuadrados de datos tradicionales del código QR; en el QRGB, estos cuadrados negros del código QR convencional se transforman en blancos mediante el modelo aditivo RGB cuando las tres capas coinciden en la misma posición. El negro (0, 0, 0) corresponde al fondo tradicionalmente blanco del código QR; en el QRGB,

este fondo se invierte a negro para maximizar el contraste visual y permitir la superposición precisa de las capas de color.

Este esquema asegura que los datos se distribuyan eficientemente entre las capas de colores primarios y secundarios; con redundancia adicional al usar el entrecruzado de colores. Esta técnica permite incrementar la densidad de información en un código QR, al triplicar su capacidad de almacenamiento sin aumentar el espacio físico requerido. Además, se mejora la seguridad de los datos gracias a la redundancia inherente en las capas. Este enfoque innovador demuestra cómo el modelo RGB puede ser aplicado no solo en visualización de colores, sino también en almacenamiento y transmisión eficiente de información.

Por último, la implementación en Python, mediante librerías de código abierto como Pillow, qrcode[pil] y opencv-python, asegura la flexibilidad, accesibilidad y sostenibilidad del sistema; haciéndolo adecuado para proyectos con impacto social y ambiental positivo.

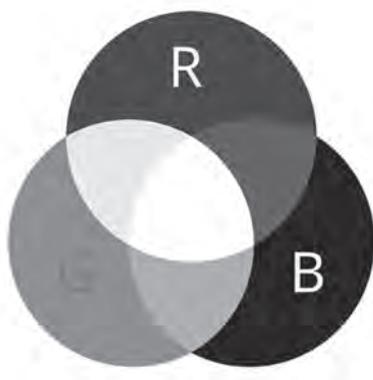


Figura 2. Modelo aditivo de colores rojo (Red), verde (Green) y azul (Blue) conocido por sus siglas en inglés: RGB. Fuente: https://commons.wikimedia.org/wiki/File:Venn_diagram_rgb.svg

Materiales y Metodología

En el desarrollo de este proyecto, se utilizaron diversas bibliotecas de Python para implementar la generación y decodificación de códigos QRGB. A continuación, se enumeran las bibliotecas empleadas y su respectiva utilidad dentro del script:

- **Kivy:** Utilizada para crear la interfaz gráfica de usuario (GUI). Kivy es una biblioteca de código abierto que permite la creación de interfaces interactivas y multiplataforma. En este caso, se utilizó para diseñar y gestionar la ventana principal de la aplicación, permitiendo a los usuarios ingresar los datos que serán codificados en las diferentes capas del código QRGB y para seleccionar los archivos a través de un explorador de archivos visual.
- **Pillow (PIL):** Es una biblioteca de Python para la manipulación de imágenes. En el script, se utiliza para combinar imágenes y superponer logotipos sobre los códigos QR generados. Además, permite el manejo de imágenes durante el proceso de decodificación, extrayendo las capas de color correspondientes (rojo, verde y azul).

- *qrcode[pil]*: Esta biblioteca es clave para la generación de los códigos QR. Permite crear los códigos QR para las capas individuales de color (rojo, verde y azul) que luego son combinadas para formar el código QRGB. Incluye también opciones para ajustar los niveles de corrección de errores y otros parámetros del código QR.
- *opencv-python (cv2)*: Utilizada para el procesamiento avanzado de imágenes y la decodificación de los códigos QR. Esta biblioteca permite la lectura y decodificación de los códigos QR mediante la detección de los patrones de datos en las imágenes combinadas, facilitando la extracción de la información de cada capa de color.
- *os*: Se utiliza para realizar operaciones con el sistema de archivos, como la creación y almacenamiento de los archivos generados, incluyendo los códigos QR y las imágenes decodificadas. Esta biblioteca permite también comprobar la existencia de archivos, como los logotipos a ser superpuestos, y acceder a las rutas de almacenamiento.
- *webbrowser*: Utilizada para abrir enlaces en el navegador web. Esta función es implementada en la GUI de Kivy para permitir que los usuarios abran enlaces directamente desde la aplicación, si se incluye un enlace en las capas de color del código QRGB.

En el proceso de implementación, para la generación del Código QRGB se utilizaron las bibliotecas *qrcode[pil]* y *Pillow* para generar tres códigos QR individuales, cada uno con los datos correspondientes a una de las capas de color (rojo, verde y azul). Estos códigos se combinaron luego en una sola imagen utilizando la técnica de superposición de colores. La interfaz Gráfica (GUI) se llevó adelante con la biblioteca *Kivy* que facilitó la creación de la interfaz gráfica que permite a los usuarios interactuar con el sistema, ingresando los datos y seleccionando imágenes para el logotipo.

En la decodificación manual se emplearon las bibliotecas *Pillow* y *opencv-python* para analizar y decodificar manualmente el código QRGB superpuesto, extrayendo los datos almacenados en cada una de las capas de color.

Metodología

La metodología para la generación y decodificación de códigos QRGB se basa en el uso de capas de color RGB para aumentar la densidad de información en los códigos QR. Primero, se generan tres códigos QR independientes, uno para cada capa de color: rojo, verde y azul, donde cada código almacena un conjunto distinto de datos, lo que permite triplicar la cantidad de información almacenada. Luego, estas capas se superponen en una sola imagen utilizando la biblioteca *Pillow*, que permite combinar las tres imágenes manteniendo la integridad de la información. Después, se puede superponer un logotipo en el centro del código QRGB utilizando las funciones de *Pillow*, sin afectar la legibilidad gracias a la corrección de errores que implementa el código QR.

La interfaz gráfica para la generación y decodificación de los códigos QRGB se crea con la biblioteca *Kivy*, lo que facilita la interacción del usuario mediante una GUI intuitiva que permite ingresar los datos de cada capa de color, seleccionar archivos de imágenes y ejecutar la generación o decodificación del código. Para la decodificación, se utilizan las

bibliotecas Pillow y opencv-python, que permiten extraer las capas de color del código QRGB y leer los datos correspondientes en cada una. Finalmente, las imágenes generadas, así como las capas decodificadas, se guardan en directorios específicos del sistema utilizando la biblioteca, lo que facilita el acceso a los archivos para futuros usos o consultas. Este enfoque permite una mayor capacidad de almacenamiento de datos en los códigos QR, mejora la seguridad al utilizar capas de color, y su implementación modular con Python y bibliotecas de código abierto asegura la flexibilidad del sistema.

Resultados y discusión

Se ilustra el proceso de funcionamiento paso a paso, corriendo el software.



```

1  import Kivy
2  from kivy.app import App
3  from kivy.uix.boxlayout import BoxLayout
4  from kivy.uix.label import Label
5  from kivy.uix.button import Button
6  from kivy.uix.textinput import TextInput
7  from kivy.uix.image import Image as KivyImage
8  from kivy.core.window import Window
9  from kivy.uix.popup import Popup
10 from PIL import Image
11 import qrcode
12 import os
13 import cv2
14
15 # Rutas para guardar y cargar archivos
16 FOLDER_PATH = r"C:\Users\Usuario\Documents\Archivos\Fede\Tutoria\1"
17
18 # Función para crear un código QR con un logo superpuesto
19 def create_qr_with_logo(data, color, logo_path, qr_version=10, box_size=10):
20     qr = qrcode.QRCode(
21         version=qr_version,
22         error_correction=qrcode.constants.ERROR_CORRECT_H,
23         box_size=box_size,
24         border=4
25     )
26     qr.add_data(data)
27     qr.make(fit=True)
28
29     img = qr.make_image(fill_color=color, back_color="white").convert("RGBA")
30
31     if not os.path.exists(logo_path):
32         raise FileNotFoundError(f"Logo file not found: {logo_path}")
33
34     logo = Image.open(logo_path).convert("RGBA")
35     basewidth = img.size[0] // 4
36     wpercent = (basewidth / float(logo.size[0]))
37     hsize = int(float(logo.size[1]) * float(wpercent))
38     logo = logo.resize((basewidth, hsize), Image.LANCZOS)
39
40     pos = ((img.size[0] - logo.size[0]) // 2, (img.size[1] - logo.size[1]) // 2)
41     img.paste(logo, pos, logo)
42
43     return img
44
45 # Combina las imágenes QR asegurándose de que todas tengan el mismo tamaño
46 def combine_qr_images(img1, img2, img3, logo_path):
47     size = img1.size
48     if img2.size != size or img3.size != size:
49         raise ValueError("All QR images must be the same size")

```

Figura 3. Script Python en Visual Studio Code. Fuente: Elaboración propia.



Figura 6. Se podrá ingresar a la carpeta de archivos para seleccionar la imagen Logo.png (que debe estar previamente configurada) para cargar desde una carpeta denominada y configurada como Tutorial. Es importante que la ruta de acceso a la imagen Logo.png esté bien preconfigurada con el script de Python que es la línea (que en este caso es el Logo de la Universidad de Palermo, pero bien podría ser otra imagen Logo, cualquiera). La ruta de acceso especificada en el script para buscar la imagen logo.png en una carpeta llamada Tutorial es: `FOLDER_PATH=r'C:\Users\Usuario\Documents\Archivos_Fede\Tutorial'`, se espera que el archivo logo.png esté ubicado dentro de esta carpeta. Si necesitas asegurarte de que el script busque correctamente logo.png, la ruta completa sería: `logo_path=os.path.join(FOLDER_PATH, "logo.png")`. Luego se debe seleccionar la opción “Generar QRGB” para que se genere el código QRGB. Fuente: Elaboración propia.



Figura 7. Se generará un código QRGB como el que muestra la imagen. Fuente: Elaboración propia.



Figura 8. Luego de haberse generado el código QRGB y almacenado como imagen superposed_qr en la carpeta Tutorial, aparece un cartel en una ventana avisándonos que la operación se ha realizado exitosamente. Fuente: Elaboración propia.

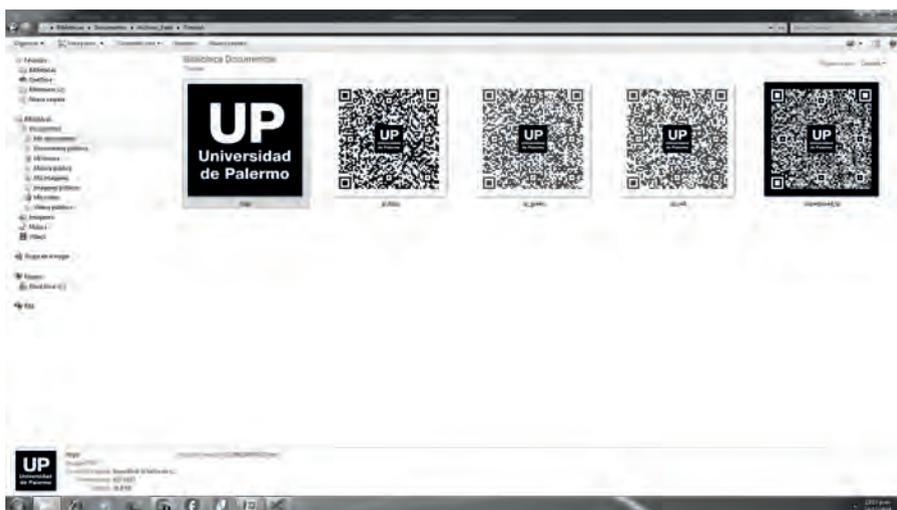


Figura 9. Al ingresar a la carpeta Tutorial nos encontraremos con cinco (5) archivos: Logo.png y los tres (3) archivos qr rojo verde y azul correspondientes a los pasos intermedios de la codificación RGB (llamados respectivamente qr_blue, qr_green y qr_red) y junto a ellos el archivo superposed_qr generado. Todos archivos de imágenes de formato .png. Fuente: Elaboración propia.

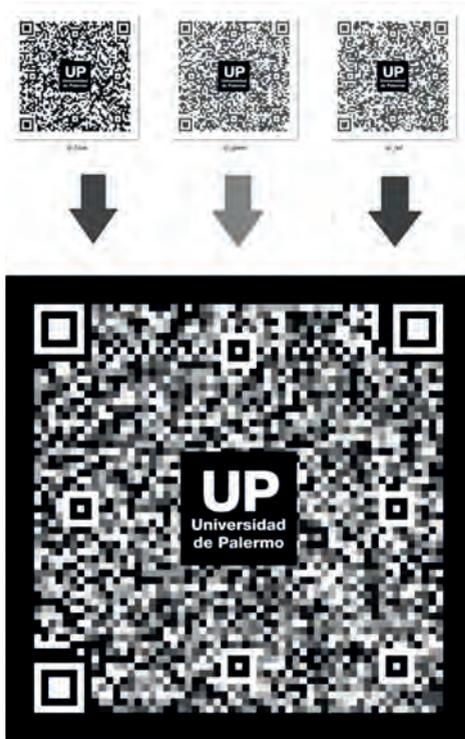


Figura 10. Los tres (3) archivos qr rojo verde y azul correspondientes a los pasos intermedios de la codificación RGB (llamados respectivamente qr_blue, qr_green y qr_red) son las etapas previas a generar la superposición del pixelado en el archivo superposed_qr. Todos archivos de imágenes de formato .png. Fuente: Elaboración propia.

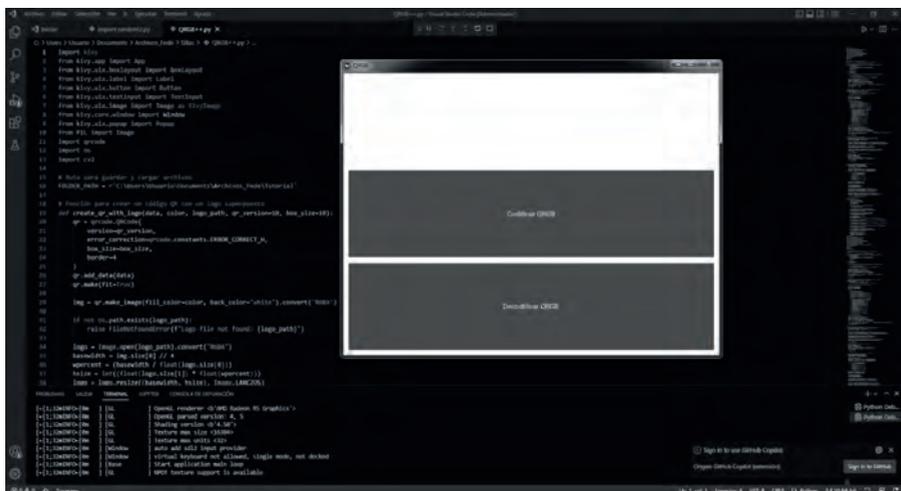


Figura 11. Si ahora cerramos todo y reiniciamos la App (reiniciamos el software o script) deberemos seleccionar la opción “Decodificar QRGB” para leer el código. Fuente: Elaboración propia.



Figura 12. En la carpeta Tutorial deberemos seleccionar el archivo superposed_qr para que se produzca el proceso de decodificación. Fuente: Elaboración propia.



Figura 13. Por lo que el sistema generará tres (3) archivos intermedios (blancos y negros correspondientes al cada uno de los tres colores RGB: rojo, verde y azul) extraídos del archivo superposed_qr para proceder a su decodificación. Fuente: Elaboración propia.



Figura 14. Así es como finalmente se produce la lectura o decodificado de los tres (3) canales rojo, verde y azul con la información respectiva: <https://www.palermo.edu/dyc/congreso-latino/>, <https://dspace.palermo.edu/ojs/index.php/cdc/issue/view/594>, https://www.palermo.edu/dyc/instituto_investigacion/ Fuente: Elaboración propia.

Conclusiones

El concepto de códigos QR en color (QRGB), que combina tres códigos QR en rojo, verde y azul para incrementar la densidad de información, representa una innovación significativa en el ámbito de la codificación y transmisión de datos. A diferencia de los códigos QR tradicionales, los QRGB utilizan la superposición de colores para almacenar hasta tres veces más información en el mismo espacio físico, mejorando así tanto la capacidad de almacenamiento como la seguridad de los datos.

El sistema High Capacity Colored Two-Dimensional Codes (HCC2D) y el Microsoft Tag son desarrollos similares que también emplean colores para aumentar la capacidad de los códigos QR. Sin embargo, estos sistemas no abarcan la superposición de múltiples códigos QR como en el enfoque QRGB. Mientras que el HCC2D se centra en la codificación de información en cada punto de un solo código, el QRGB se destaca por su técnica única de combinar códigos en capas de color.

A pesar de su novedad, el concepto de QRGB no ha sido ampliamente adoptado ni comercializado. La implementación de algoritmos avanzados para la codificación y decodificación de estos códigos es crucial, abordando desafíos como la mezcla de colores y la precisión en la recuperación de datos. La compatibilidad con dispositivos de escaneo actuales y la adaptación de los escáneres para leer códigos QRGB son áreas que requieren atención adicional.

El uso de herramientas de Python, como `qrcode[pil]`, `Pillow`, y `opencv-python`, para la creación de QRGB demuestra una adecuada integración de tecnologías de código abierto. No obstante, la creación de una interfaz de usuario intuitiva y la evaluación de compatibilidad con escáneres en diversas condiciones son aspectos a desarrollar para facilitar su adopción.

Los QRGB tienen aplicaciones potenciales en sectores como la seguridad de documentos, tarjetas de visita digitales, y publicidad interactiva. Su capacidad para almacenar información adicional y su mejora en la seguridad pueden revolucionar el uso de códigos QR, brindando soluciones innovadoras para el almacenamiento y la transmisión de datos en la era digital.

En futuros estudios, será esencial profundizar en la precisión de la decodificación, la compatibilidad con dispositivos existentes, y la creación de una interfaz de usuario amigable. El potencial para aplicar QRGB en criptografía visual, marketing, y educación también merece exploración adicional. Este enfoque innovador, aunque aún en desarrollo, ofrece una base sólida para el avance en la tecnología de códigos QR y su implementación práctica.

Bibliografía

- Anderson, I. F. (2024). "QRGB: App for QR Code Generation (3-in-1 Method), Additive Color Generation Method (RGB), Using Python Programming Code, to Increase Accumulated Information Density". *Preprints.org*, pp. 1-47. <https://doi.org/10.20944/preprints202407.1384.v2>
- Anderson, I. F. (2024). "QRGB: App para la generación de códigos QR (método: 3 en 1), o método de generación aditiva de colores (RGB), aplicando librerías de Python de código abierto, para aumentar la densidad de información acumulada". *EdArXiv Preprints*, pp. 1-29. <https://doi.org/10.35542/osf.io/hy2em>
- Anderson, I. F. (2024). "QRGB+: Advanced QR Code Generator with RGB Color Method in Python to Expand Data Capacity". *Journal of Sensor Networks and Data Communications*, Vol. 4, N°. 2, pp. 1-20. Handle: <http://sedici.unlp.edu.ar/handle/10915/169498>
- Anderson, I. F. (2024). "QRGB++ in Python running in Visual Studio Code with a graphical in-terface (pip install kivy) + pip install pillow + pip install qrcode[pil] + pip install opencv-python". *OSF Preprints*, pp. 1-17. <https://doi.org/10.31219/osf.io/g3ame>

Abstract: The QRGB++ project involves the development of a QR Code application, where the script or source code is written in Python software to utilize the RGB color system, generating a QR Code with increased information storage capacity, improved resource efficiency, and a positive environmental impact. This innovative method allows up to three times more information to be stored in the same space by combining three QR codes (red, green, and blue) into one, also improving data security through redundancy.

In addition to optimizing the use of paper and ink (although the scanning system for reading printed paper has not yet been developed), this approach would potentially contribute to reducing waste, deforestation, and the carbon footprint in the future, aligning with eco-design and sustainable trends. Currently, its development is only intended for use and distribution in digital format.

The QRGB code is generated using open-source libraries like `qrcode[pil]`, `Pillow`, and `opencv-python`, integrating a graphical user interface (GUI) with Kivy to facilitate the creation and decoding of the codes. This technical accessibility promotes technological inclusion and the democratization of the use of advanced tools, allowing designers with basic programming knowledge to use them for free. Therefore, it is solidary and has been developed with an application example for the University of Palermo and its various links from the Faculty of Design and Communication.

Developed with Python software as the script interpreter, it runs on the Visual Studio Code platform. After overcoming technical challenges with color blending, QRGB++ offers a significant increase in data density, with potential applications in security and high-density storage. Its use in social and educational projects improves accessibility for people with disabilities, fosters creativity and solidarity, and reinforces sustainability throughout the product's lifecycle, demonstrating how technology and design can converge to create a positive impact on society.

Keywords: App - QR Codes - RGB Additive Color System - Python Script - Visual Studio Code IDE

Resumo: O projeto QRGB++ envolve o desenvolvimento de uma aplicação de Códigos QR, onde o script ou código-fonte é desenvolvido no software Python para usar o sistema de cores RGB, com o objetivo de gerar um Código QR com maior capacidade de armazenamento de informações, eficiência no uso de recursos e um impacto positivo no meio ambiente. Este método inovador permite armazenar até três vezes mais informações no mesmo espaço ao combinar três códigos QR (vermelho, verde e azul) em um só, melhorando também a segurança dos dados por meio da redundância.

Além de otimizar o uso de papel e tinta (embora o sistema de scanner para leitura impressa em formato papel ainda não tenha sido desenvolvido), no futuro, contribuiria potencialmente para a redução de resíduos, desmatamento e pegada de carbono, alinhando-se com o ecodesign e as tendências sustentáveis. Atualmente, seu desenvolvimento está pensado apenas para uso e distribuição em formato digital.

O código QRGB é gerado usando bibliotecas de código aberto como `qrcode[pil]`, `Pillow` e `opencv-python`, integrando uma interface gráfica (GUI) com Kivy para facilitar a criação e decodificação dos códigos. Essa acessibilidade técnica promove a inclusão tecnológica e a democratização do uso de ferramentas avançadas, permitindo que designers com conhecimentos básicos de programação possam utilizá-lo gratuitamente. Por isso, é solidário e foi desenvolvido com um exemplo de aplicação para a Universidade de Palermo e seus diversos links da Faculdade de Design e Comunicação.

Desenvolvido com o software Python como o intérprete do script, ele é executado na plataforma Visual Studio Code. Superados os desafios técnicos com a mistura de cores, o

QRGB++ oferece um aumento significativo na densidade de dados, com potenciais aplicações em segurança e armazenamento de alta densidade. Seu uso em projetos sociais e educacionais melhora a acessibilidade para pessoas com deficiências, fomenta a criatividade e a solidariedade, e reforça a sustentabilidade em todas as etapas do ciclo de vida do produto, demonstrando como a tecnologia e o design podem convergir para gerar um impacto positivo na sociedade.

Palavras-chave: App - Códigos QR - Sistema aditivo de cores RGB - Script Python - IDE Visual Studio Code

[Las traducciones de los abstracts fueron supervisadas por el autor de cada artículo.]
