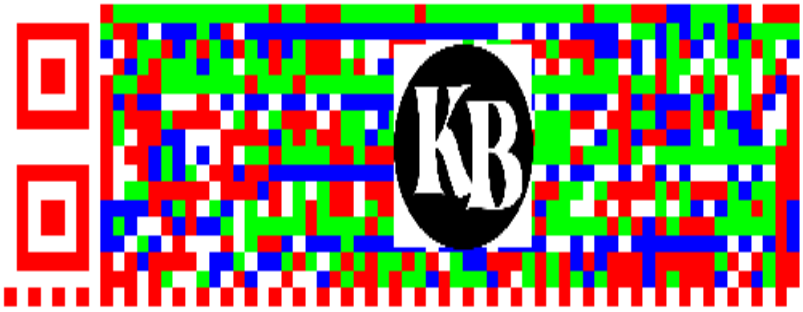
LABONNE Benjamin L3 Informatique

AL_HENDI Khaled UCA

https://github.com/belabon25/KBCode

RESEAU 2

KBCode



12/05/2023

Clermont-Ferrand

Description du code

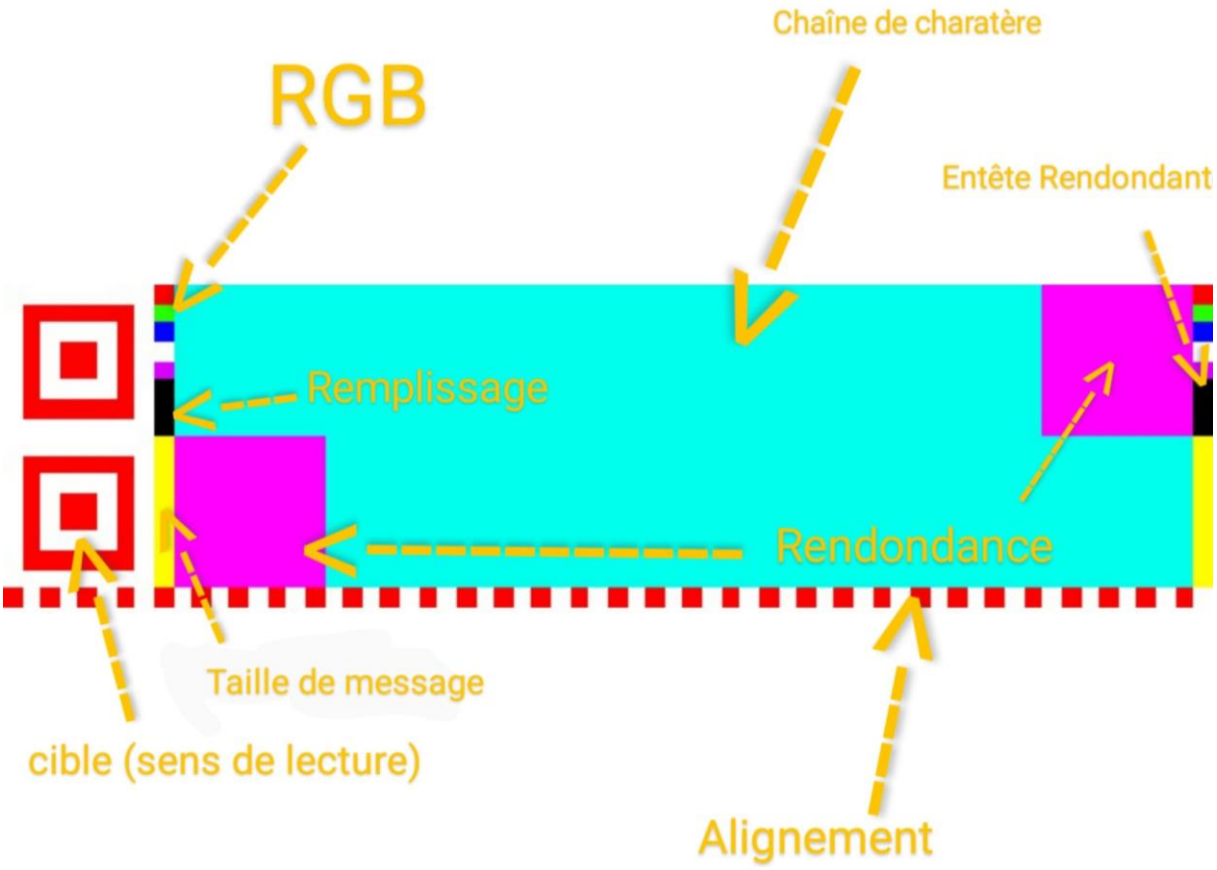
Dans notre projet de Réseau 2, nous avons créé un code graphique pour permettre de communiquer des informations en transformant des chaînes de charactères en matrice binaire qu'on convertit en un code coloré.

Fonctionnement du KBCode

Le Langage de programmation que nous avons utilisé est Python, nous avons utilisé les bibliothèque numpy pour travailler avec des matrices, reedsolo pour la redondance des données et matplotlib pour générer l'image de notre code.

Chaque lettre est présentée par 8 bits. On les découpe en 4 morceaux tel que chaque morceau présente 2 bits. Nous pouvons ensuite convertir ces morceaux en pixels de couleur distinctes pour les afficher.

Schéma théorique



En-tête

L'en-tête correspond à la première et la dernière colonne de la partie données de notre code.

La dernière colonne est un duplicata de la première dans un soucis de redondance des données.

Les 4 premiers pixels représente la valeur donnée à une couleur (dans le schéma 00 = rouge, 01 = vert, 10 = bleu, 11 = blanc).

L'ordre RGBW peut être librement modifié, le code source prend en charge cela.

Le 5e pixel représente l'encodage. Actuellement, l'UTF-8 et l'ASCII non étendu est disponible

La bande noire est une bande de remplissage (vide) mais pourra servir dans le futur (voir Pistes d'améliorations)

La bande jaune est réservée à la taille de message et mesure 8 pixels. Il est donc théoriquement possible de générer des messages de 2^16 bits !

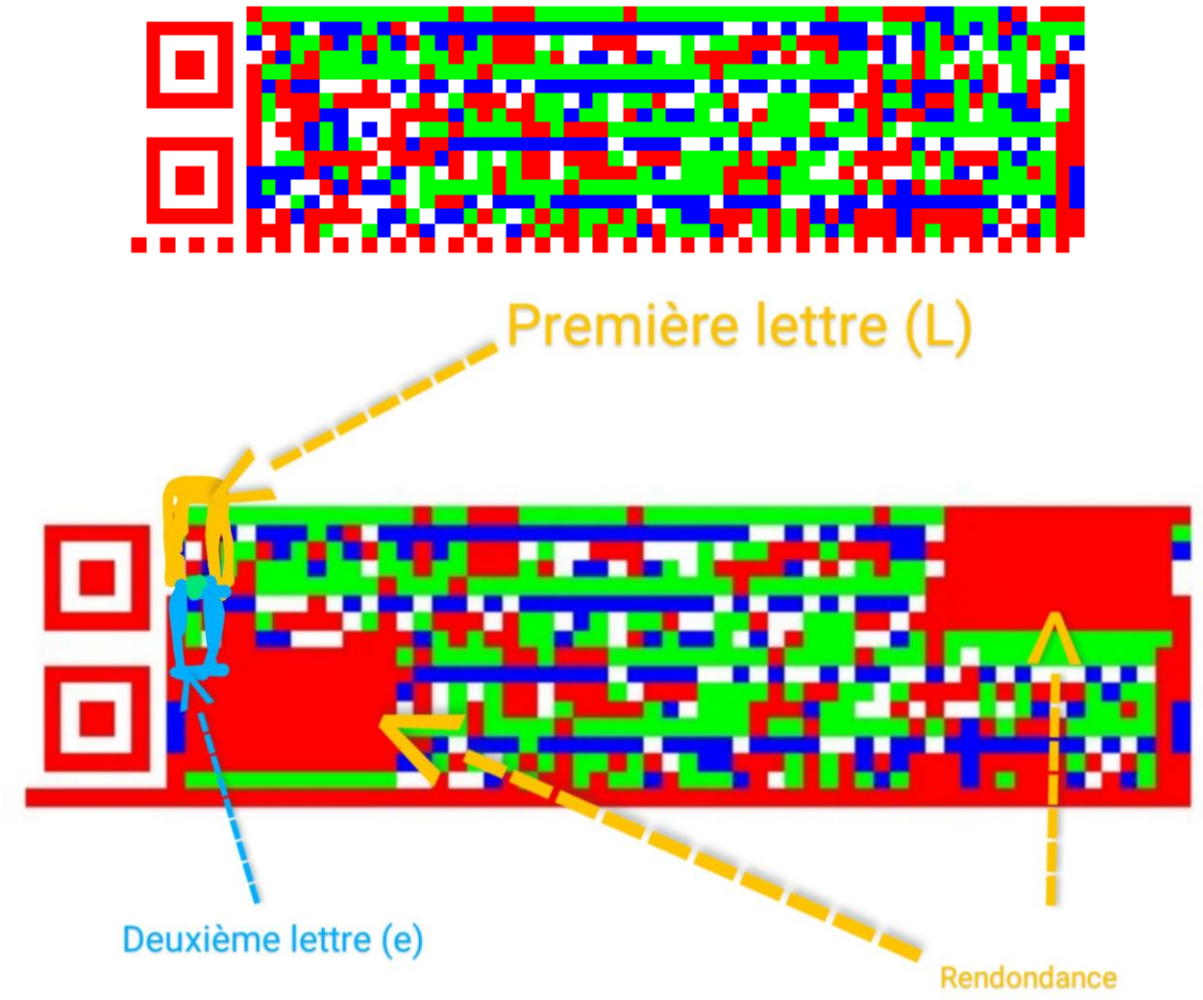
Données et redondance

Le message stocké est découpé en deux parties : le texte et la redondance. Le KBCode assure une redondance de 30%.

Dans le schéma ci-dessus, on peut observer que la partie redondance et données est décalée entre le 8e et le 9e pixel. Ce découpage permet d'assurer que le code reste lisible malgré une déchirure du code qui corromprait toute une partie du message. En contrepartie, la matrice doit être remise dans le bon ordre pour permettre une lecture des données colonnes par colonnes.

Exemple d'encodage et de lecture

```
"Le KBCode est l'avenir des QRCodes! Il est de taille variable ce qui
lui permet de contenir beaucoup d'informations en plus d'être posable
sur des surfaces cylindriques"
```



Le sens de lecture est donné par les 2 carrés et par la ligne d'alignement.

Le schéma précédent se base sur une ancienne version du KBCode qui présentait une erreur sur la partie correction d'erreur. Cela nous arrange car le fait que ces parties soit visibles rend la compréhension plus simple.

Le code se lit colonne par colonne et 4 pixels par 4 pixels. Pour lire le code en informatique, nous pensons qu'il est plus simple de reconvertir la partie données de la matrice en deux blocs distincts (texte puis redondance) avant de récupérer les informations (voir fonction incomplète KBDecode)

Pistes d'améliorations

L'amélioration majeure qui peut être ajouté à notre code est l'implémentation d'une redondance de taille variable. En effet, le code source est capable de supporter cela, mais il faut aussi l'ajouter dans l'en-tête du code pour pouvoir le décoder. La présence d'une partie de remplissage est une opportunité de faire cela.

Comme nous possédons de la redondance de données, il est théoriquement possible d'ajouter un logo à notre code. Cependant, l'utilisation de matplotlib pour le construire ne nous donne pas suffisamment de ressources pour en ajouter un dans le code.

Conclusion

Ce projet de réseau 2 nous a permis de mettre en application ce que nous avons appris sur les différents moyens de communiquer.

A première vue, il semble incohérent de travailler sur sa propre version du QRCode dans ce cours, cependant un code graphique est un protocole de communication comme un autre. La présence de technologie de récupération de données perdues est aussi une part importante du cours, il était donc important pour nous d'ajouter à notre code un algorithme viable.