Rapport de soutenance 2

adam.mahraoui
emil.toulouse
samy.achache

1 avril 2022

Sommaire

1 Introduction **4**
1.1 Présentation du projet 4
1.2 Qui sont les membres du groupe ? 4

2 Pourquoi ce projet ? **6**
2.1 Les objectifs du projet 6

3 Encodage de l'information **7**
3.1 Choix du mode et de du niveau de correction 7
3.2 Prémice de l'encodage 8
3.3 codage de correction d'erreurs 9
3.4 Creation de la structure final 10

4 Encodage du QR-Code **10**
4.1 Paternes de détection de position et les séparateurs 11
4.2 Les "timing Patterns" 12
4.3 Paternes d'alignements 12
4.4 Les reserveds areas 14
4.5 Le remplissage du QR-Code 15
4.6 Le Data Masking 16

5 Chargement de l'image **20**

6 Decodage du QR Code **22**
6.1 Lecture du QR Code et ses differentes version 22
6.2 La longueur du message 23
6.3 Le Décodage 23
6.3.1 Le masque 23
6.3.2 Lecture des différentes informations 23
6.3.3 lecture du message 23

7 Interface graphique **25**

8 Site Web **26**

9 Organisation du projet **28**
9.1 Tableau de répartition des tâches 28
9.2 Planning prévisionnel 28
9.3 Ressenti personnel 29
9.3.1 Adam Mahraoui 29
9.3.2 Emil Toulouse 29
9.3.3 Samy Achache 29

10 Conclusion **30**

1 Introduction

1.1 Présentation du projet

Notre projet de S4, est un logiciel dans lequel l'algorithmique a une part très importante, il s'agira d'étudier le QR-code très en vogue en ce moment dû à la crise sanitaire. Ce projet est divisé en plusieurs sous-programmes. Le premier sera l'encodage d'un texte ou lien hypertexte sous forme de QR-code et d'avoir la possibilité de récupérer ce résultat sous forme d'image au format classique.

Par la suite nous avons jugé intéressant que notre logiciel propose également le décodage de ces QR-code et avec la possibilité de voir sous forme textuelle le contenu d'un QR-code. Nous avons donc besoin d'un algorithme de décodage de QR-code ainsi que de regrouper tout cela dans une GUI afin que l'expérience avec notre logiciel n'en soit que meilleur. Avant d'aller plus loin et sans plus attendre voici notre groupe.

1.2 Qui sont les membres du groupe ?

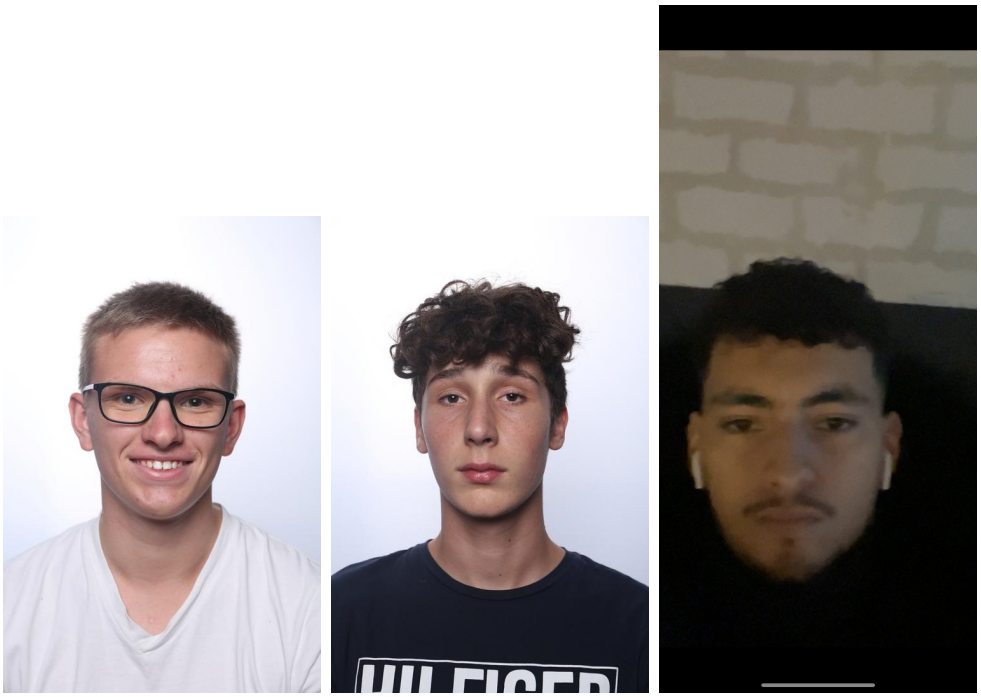
Notre groupe est composé de quatres personnes actuellement, Emil Toulouse, Adam Mahraoui, Icham Benabbas et Samy Achache.

- Emil Toulouse : Le QR-code pour moi est un outil qui a un fort potentiel pour transmettre une information et simplifier des actions de manière rapide. Dans cette optique la proposition de faire notre version du QR-code m'as bien plu. C'est pour cela que j'ai accepté de rejoindre l'équipe. Ayant l'ambition de bien comprendre cette technique qui semble magique, je compte me positionner sur la partie encodage d'un QR-code.

- Adam Mahraoui : La curiosité est un trait de personnalité qui me caractérise bien et c'est pour cela que cette idée projet m'a plu. J'ai toujours utilisé les QR-codes en me demandant comment cela fonctionnait, je n'ai jamais vraiment eu le temps de chercher la réponse à cette question mais ce projet va me permettre d'y répondre. Ce thème me permet de revoir le traitement de l'image et mêler ça avec de l'algorithmie sur un sujet intéressant et concret.

- Samy Achache : Je baigne dans le monde de l'informatique depuis l'âge de quatre ans, à travers les jeux videos, console, Pc. Mon expérience informatique avant Epitas resume plus à l'utilisation a la création. Plus tard je souhaite idealementme spécialiser dans une branche particulière de l'informatique qui est la cybersécurité. L'avantage de ce projet est

le choix de notre logiciel. Effectivement faire un projet autour des QR-Codes sachant le contexte actuelle (pandemie) ou l'utilisation des QR-Codes est devenu repandue connaitre leur fonctionnement me semble trezs important pour moi.



Membres de l'équipe

2 Pourquoi ce projet ?

La technique des QR-codes n'est pas récente, en effet elle existe depuis les années 2000. Elle permet de stocker une information sous forme d'image (du texte, une connexion internet, envoyer un courriel, une vidéo en ligne...).

C'est une technique simple à utiliser et comme son nom l'indique "Quick response Code" qui permet d'obtenir une réponse rapide et simple : il suffit juste de scanner cette image qui ressemble à un damier pour que quelque chose se passe, cela semble presque magique.

Nous voulons comprendre les rouages derrière cette méthode de partage de donné. Comment une simple image peut-elle permettre de faire autant de chose ? Et comment est-il possible de transformer une information en un "damier corrompu" lisible par tous les appareils électroniques.

2.1 Les objectifs du projet

Pour ce projet qui a pour but final de rendre un logiciel capable de lire un QR code ou de transformer un lien en celui-ci, nous nous fixons plusieurs objectifs intermédiaires pour rendre un travail complet, fonctionnel et qui répondra aux exigences demandées. Voici la liste de nos principaux objectifs intermédiaires qui devront être atteints pour la soutenance finale :

- Chargement d'une image contenant un QR Code ;
- Détection du QR code sur l'image ;
- Implémentation de l'algorithme permettant de lire le QR Code;
- Implémentation de l'algorithme permettant de creer son QR Code;
- Réaliser une interface pour le logiciel ;

3 Encodage de l'information

Nous avons decider de nous pencher au plus vite sur la transformation de la donnée utilisateur en une nouvelle chaine binaire correspondant à ce qui va completer le QR-code. Le principe est simple sur le papier et plus compliqué dans sa mise en place. En effet la creation d'un QR-code (les données contenues) doivent respecter un certain pattern et des paramétrages bien précis pour que le résultat final soient utilisable.

3.1 Choix du mode et de du niveau de correction

Les QR-codes peuvent stocker un certain nombre d'informations et de type différents. Le choix d'un mode correspond à la palette de caractère accepter pour l'encryption du message initial.

Numeric mode is for decimal digits 0 through 9.

Alphanumeric mode is for the decimal digits 0 through 9, as well as uppercase letters (not lowercase!), and the symbols $, %, *, +, -, ., /, and : as well as a space. All of the supported characters for alphanumeric mode are listed in the left column of this alphanumeric table.

Byte mode, by default, is for characters from the ISO-8859-1 character set. However, some QR code scanners can automatically detect if UTF-8 is used in byte mode instead.

Tableau des modes QR code

Parmis ces 3 choix nous avons retenu le plus complet (Byte mode) qui prendra en compte tout les caractères du clavier AZERTY classique. Cependant nous avons pris le choix de ne pas prendre en compte un quatrième mode le Kanji Mode puisqu'il inclut des caratères supplémentaires que nous avons juger peux utile pour la charge de travail apporté.

Passons au niveau de correction. Ce niveau de correction est un paramétrage simple ayant pour objectif de spécifier la complexité du QR-code afin que,meme apres degradation partiel, il reste opérationnel.Nous avions le choix entre plusiseurs paramétrage et nous avons retenu le mode M qui une fois mise en place dans sa totalité pourra être utlisable même après 15 pourcent de dégradation.

3.2 Prémice de l'encodage

Nous avons opté pour réaliser une structure contenant tout les possibles paramètres de l'encodage du QR-code avec notament sa taille, sa version, la donnée traité etc... La structure est disponible dans le projet.

```c
struct encdata
{
    // Version of the QR code.
    int version;

    // Mode indicator.
    // 1 -> byte mode
    char mi;

    // Length of the input string.
    size_t size;

    // Error correction level.
    char correction_level;

    // Length of the encoded data.
    size_t len;

    // Length necessary for the encoded data.
    size_t nlen;

    // Encoded data (binary).
    char *data;

    // Error correction Codewords Per Block
    size_t ec;

    // Number of blocks in group 1
    size_t block1;

    // Number of Data Codewords in Each of Group 1's Blocks
    size_t group1;

    // Number of Blocks in Group 2
    size_t block2;

    // Number of Data Codewords in Each of Group 2's Blocks
    size_t group2;
};
```

structure d'encodage QR code

Une fois créer il nous faut donc la remplir.Cela passe par le choix de la version (la plus petite possible). Le mode ainsi que l'erreur de correction étant choisi nous devons passez à la taille/version qu'aura le QR-code.Pour cela rien de plus simple chaque version du QR-code peut contenir un certain nombre d'octect et que donc en connaissant ce tableau des version nous pouvons facielement trouver le version adapté.

Nous allons enfin pouvoir commencer a remplir le champ de la data. Tous d'abord nous ecrivons "0100" qui correspond au choix du mode, ici Byte, puis nous ecrivons la taille de la donnée en binaire par exemple :

"Hello World" contient 11 charactères donc $11(10) = 1011(2)$.

Pour respecter les nomenclatures nous devons impérativement faire en sortes que cette information soient représenté sur 9 bits. Nous ajoutons donc des 0 pour que cela correspondent aux demandes.

Pour le mode alphanumérique, chaque caractère alphanumérique est représen -té par un nombre .Pour chaque paire de caractères, on récupère la représentation numérique du premier caractère et multipliez-la par 45. Ajoutez ensuite ce nombre à la représentation numérique du deuxième caractère.On convertis maintenant ce nombre en une chaîne binaire de 11 bits, en remplissant à gauche avec des 0 si nécessaire.Si la chaine est de longeur impaire le dernier charactère sera représenté sur 6 bits.

Pour avoir la bonne représentation de l'information la correction d'erreur impose une certaine taille pour la data encodée. Ainsi avec le mode choisis nous pouvons determiner cette taille. Un tableau est fourni avec le nombres de bloc erreur nécessaire. On multiplie par 8 ce nombre pour obtenir la longeur nécessaire de la chaine data.Une fois cette longeur de chaine connue nous devons donc remplir les espaces manquants. On commence par ajouter au maximum quatres zéro pour tomber remplir la taille nécessaire. Si ce n'est pas le cas on mets quatre 0.

Maintenant nous devons finir la chaine d'une manière bien spécifique. Il faut tout d'abord ajouter des 0 jusqu'a tomber sur une longeur de chaine divisible par 8 une fois réaliser. Nous terminons le processus par completer la chaine jusque la taille nécessaire avec 11101100 00010001 chacun leur tour.

3.3 codage de correction d'erreurs

Il est maintenant enfin temps de commencer à générer des mots de code de correction d'erreurs. Nous créerons des mots avec des valeur numériques qui seront utilisés comme coefficients du polynôme du message.

L'étape de codage des données a abouti aux mots de code de données suivants pour HELLO WORLD sous la forme d'un code 1-M.

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000000 11101100 00010001 11101100 00010001 11101100 00010001

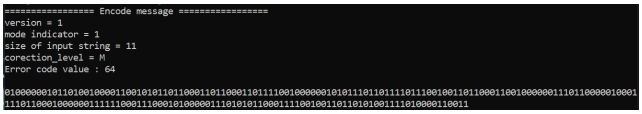
On convertis ces nombres binaires en décimal : 32, 91, 11, 120, 209, 114, 220, 77, 67, 64, 236, 17, 236, 17, 236, 17

Ces nombres sont les coefficients du polynôme du message. En d'autres termes: 32x 15 + 91x 14 + 11x 13 ... et ainsi de suite

Nous avons donc utiliser un polynome messager et un polynome generateur afin de pouvoir avoir une partie du qr code ayant pour role d'avoir une récuperation d'information si le QR-code est endomagé. le but etant de trouver ou l'information manque et de la retrouver parmis ces codes de correction.

3.4 Creation de la structure final

Pour cette dernière partie nous allons mettre bout à bout toutes les manipulations précedentes. Le but etant de créer la chaine de 1 et 0 final que nous allons mettre dans le QR-code. Nous ecrivons d'abord le message initial et nous y ajoutons les codes erreurs. Cela creer donc une chaine de caractère utilisable dans la creation de matrice et de l'image résultat du QR-code.



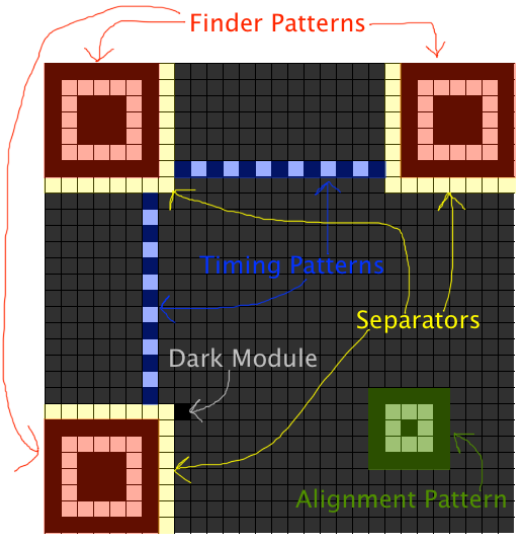
structure d'encodage finale QR code

Nous pouvons voir toute les informations finales qui seront utilisées pour la suite de l'encodage du QR-code.

4 Encodage du QR-Code

Un QR-Code peut être représenté sous la forme d'une matrice carrée contenant des 1 pour les cases noires et des 0 pour les blanches pour représenter la donnée stockée. Pour générer cette matrice nous avons besoin de la version V du QR-Code pour pouvoir calculer sa taille, en effet sa largeur et sa longueur seront égales à (((V-1)*4)+21).

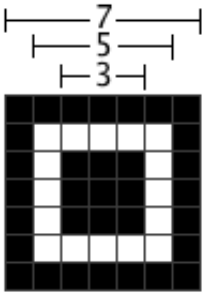
Tous les QR-Codes possèdent les mêmes éléments, seuls leurs emplacements diffèrent en fonction de taille et donc de leur version. On peut retrouver alors 4 paternes : les paternes de détection de position, les séparateurs, les "timings patterns" et les paternes d'alignements.



Structure d'un QR code

4.1 Paternes de détection de position et les séparateurs

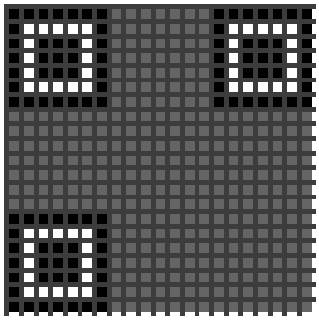
Les paternes de détection de position sont les 3 gros carrés visibles très facilement sur un QR-Code et ils sont au nombre de 3. Un paterne se compose d'un carré noir extérieur de 7 pixels sur 7 pixels, d'un carré blanc intérieur de 5 pixels sur 5 pixels et d'un carré noir uni au centre de 3 pixels sur 3 pixels.



Dimension d'un paterne de détection de position

Leurs positions sont calculées en fonction de la longueur de la matrice nommée "size" : Pour le premier, le coin supérieur gauche se situe en (0,0),

pour le deuxième en (size-7,0) et pour le dernier en (0, size-7). Ils permettent au lecteur de reconnaître le code avec précision et de lire avec rapidité les informations contenues dans le QR-Code. Ils indiquent aussi l'orientation de la structure.

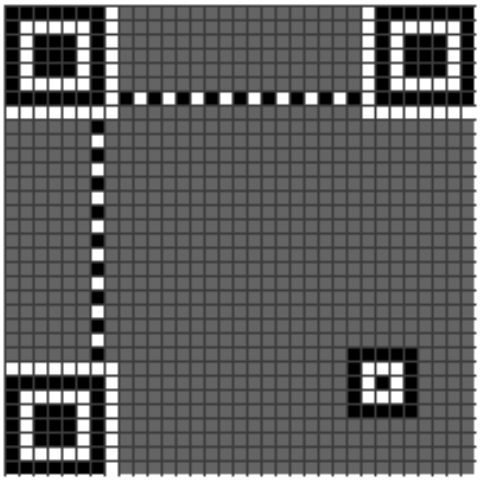


Localisations des paternes de détection de position

Pour ce qui est des séparateurs, se sont des lignes blanches qui se situent sur les bords exterieurs des paternes précedents, ils permettent comme leur nom l'indique de séparer la donnée que contient l'image aux paternes.

4.2 Les "timing Patterns"

Ils sont au nombre de deux, ils se composent de deux lignes avec une alternance de couleur partant du coin inférieur droit du paterne de détection de position haut gauche, jusqu'aux deux autres paternes. L'une est positionnée sur la 6-ème ligne et l'autre sur la 6-ème colonne de la matrice, de plus elles commencent et finissent toujours par un pixel noir. Ces lignes permettant au lecteur de déterminer facilement la taille du QR-code et donc d'obtenir la version de celui-ci.



Localisation des "timing patterns"

4.3 Paternes d'alignements

Tout comme les paternes de détection de position, ce sont des carrés qui permettent de bien cadrer l'image du QR de faciliter sa lecture même quand l'image est positionnée sur une surface qui n'est pas plate. Un paterne d'alignement se compose d'un carré noir de 5 pixels sur 5 pixels, d'un carré blanc intérieur de 3 pixels sur 3 pixels et d'un seul pixel noir au centre.

Ils sont présents uniquement sur les QR-Codes de versions supérieures à 2 et leurs positions sont déterminées par une table qui pour chaque version attribue une liste de valeur possible pour les coordonnées (x, y) du centre de chaque paterne. Une fois que nous avons la liste de ces valeurs il faut les tester une par une pour voir s'il est possible de placer le carré dans la matrice sans empiéter sur un paterne de détection.

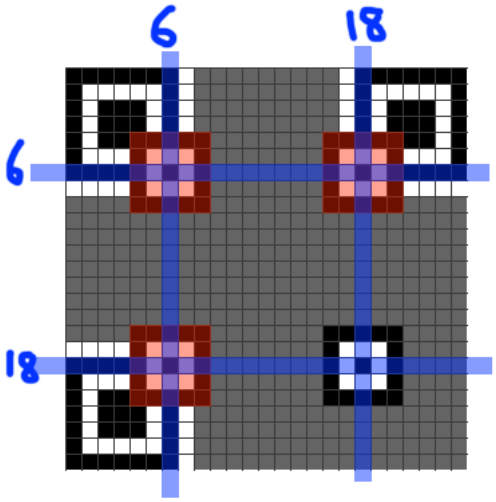


Illustration des coordonées possible des paternes d'alignement

Pour ce faire, j'ai créé un tableau à 2 dimensions qui pour un indexe V donné (correspondant la version du QR-Code) la liste des coordonnées (a,b) possibles. Via une boucle "for" je teste alors si chaque combinaison est possible avec cette condition :

```
if( mat[size_p*(a+2)+(b+2)] =='2' && mat[size_p*(a+2)+(b-2)] == '2' &&

mat[size_p*(a-2)+(b-2)] =='2' && mat[size_p*(a-2)+(b+2)] == '2')
    _align(a, b, mat, size_p);
```

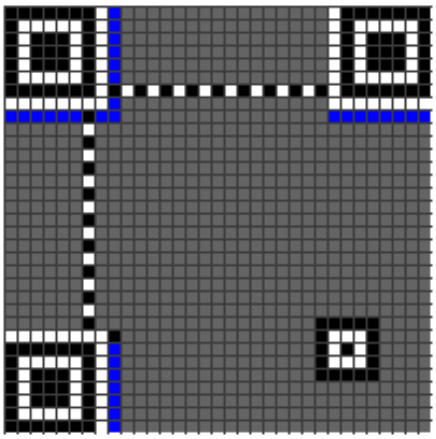
Condition pour placer un paterne avec comme centre (a,b)

Ici, à partir des coordonnés du centre du carré, je vérifie si les 4 coins du carré sont bien disponibles et donc si je peux appeler la fonction "align" pour placer le paterne.

4.4 Les reserveds areas

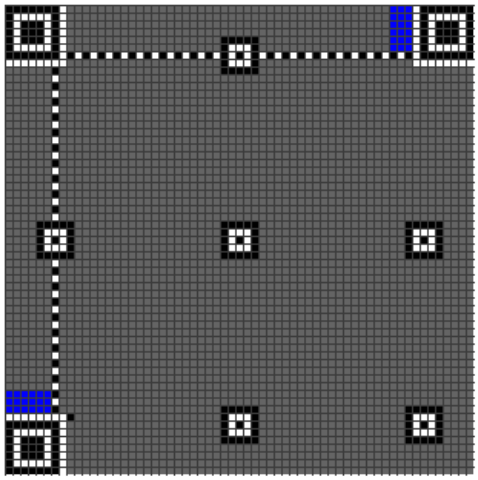
Les “reserved areas” ou nommées en Français “zones de réserve” sont des parties du QR codé qui doivent être marquées pour les différencier des pixels qui sont dit “libres” et donc où la data de ce que l'on doit encoder peut-être placé. Ces zones de réserve seront utilisées pour stocker la version et le format du QR-Code. Étant donné que nous représentons notre QR code sous la forme d'une matrice constituée de 0 ou 1 pour blanc ou noir et de 2 pour les zones vides et libres, nous avons ajouté le 3 pour marquer ces zones et donc avoir la possibilité via des conditions de ne pas utiliser ces zones lors du remplissage de la matrice.

Les premières zones de reserve sont situées le long des séparateurs, celles ci sont présentes pour stocker le format



Localisation des zones de reserve pour le format

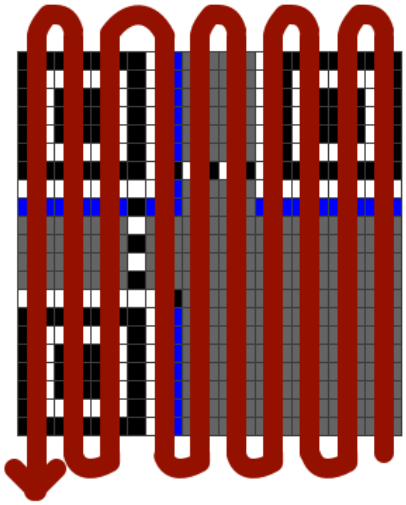
Pour les QR-Codes possédant une version supérieure ou égale à 7, deux autres autres zones de réserve doivent être marquées pour pouvoir stocker la version. Les zones sont deux rectangles de 6x3 au-dessus du paterne de détection de position inferieure gauche et à droite du paterne de détection de position supérieure droit



Localisation des zones de reserve pour le format

4.5 Le remplissage du QR-Code

Une fois que les différents patterns sont placés dans la matrice il faut désormais placer chaque pixel de la chaîne de string généré précédemment dans la matrice. Tout ce processus peut se représenter par un "serpent" qui traverserai la matrice en zigzag.



Représentation simplifier du remplissage de la matrice d'un QR-Code

Pour placer les bits de données nous devons commencer en bas à droite de la matrice et remonter dans une colonne de 2 pixels de large.

Lorsque notre colonne atteint une zone où il ne peut plus écrire, nous utilisons la colonne de 2 pixels suivante qui commence immédiatement à la gauche de la colonne précédente et nous continuons ensuite de remplir cette colonne vers le bas. A chaque fois que notre colonne de remplissage a atteint un bord de la matrice, il faut passez à la colonne suivante et changer de direction.

Si un paterne ou une zone de réserve est rencontrée, le bit de données est placé dans le pixel inutilisé suivant.

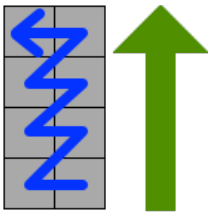


Illustration du remplissage dans la premiere colonne

4.6 Le Data Masking

Maintenant que les modules ont été placés dans la matrice, le meilleur motif de masque doit être déterminé. Un motif de masque modifie les modules qui sont sombres et ceux qui sont clairs selon une règle particulière. Le but de cette étape est de modifier le code QR pour le rendre aussi simple que possible pour un lecteur de code QR.

Si un module dans le code QR est "masqué", cela signifie simplement que s'il s'agit d'un module clair, il doit être changé en module sombre, et s'il s'agit d'un module sombre, il doit être changé en module clair. En d'autres termes, le masquage signifie simplement basculer la couleur du module. La documentation du code QR définit huit modèles de masque qui peuvent être appliqués au code QR. Par exemple, pour le motif de masque 1, chaque ligne paire de la matrice QR est masquée, et pour le motif de masque 2, chaque troisième colonne de la matrice QR est masquée.Les modèles de masque doivent uniquement être appliqués aux modules de données et aux modules de correction d'erreurs. Pour ce faire nous appliquons le mask sur tout les zones de la matrice et nous reconstruisons les modules cassés si nécessaire suite aux masking.

Une fois qu'un modèle de masque a été appliqué à la matrice QR, il reçoit un score de pénalité basé sur quatre conditions d'évaluation qui sont définies dans la spécification du code QR. Un encodeur de code QR doit appliquer les huit modèles de masque et évaluer chacun d'eux. Quel que soit le modèle de masque qui entraîne le score de pénalité le plus bas, c'est le modèle de masque qui doit être utilisé pour la sortie finale.Les quatre règles de pénalité peuvent être résumées comme suit :

- La première règle donne au code QR une pénalité pour chaque groupe de cinq modules ou plus de même couleur dans une rangée (ou une colonne). ;
- La deuxième règle donne au code QR une pénalité pour chaque zone 2x2 de modules de même couleur dans la matrice. ;

- Implémentation de l'algorithme permettant de lire le QR CodeLa troisième règle donne au code QR une pénalité importante s'il existe des modèles qui ressemblent aux modèles de recherche.;
- La quatrième règle pénalise le code QR si plus de la moitié des modules sont sombres ou clairs, avec une pénalité plus importante pour une différence plus importante.;

Pour la première condition d'évaluation, vérifiez chaque ligne une par une. S'il y a cinq modules consécutifs de la même couleur, ajoutez 3 à la pénalité. S'il y a plus de modules de la même couleur après les cinq premiers, ajoutez 1 pour chaque module supplémentaire de la même couleur. Ensuite, vérifiez chaque colonne une par une, en vérifiant la même condition. Additionnez le total horizontal et vertical pour obtenir le score de pénalité 1.

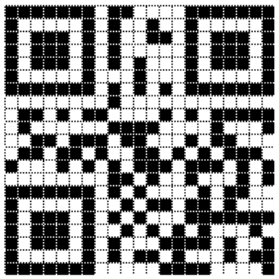
Pour la deuxième condition d'évaluation, recherchez les zones de la même couleur qui sont au moins 2x2 modules ou plus. La spécification du code QR indique que pour un bloc de couleur unie de taille m × n, le score de pénalité est de 3 × (m - 1) × (n - 1). Cependant, la spécification du code QR ne précise pas comment calculer la pénalité lorsqu'il existe plusieurs façons de diviser les blocs de couleur unie.

La troisième règle de pénalité recherche des modèles sombre-clair-sombre-sombre-sombre-clair-sombre qui ont quatre modules lumineux de chaque côté.

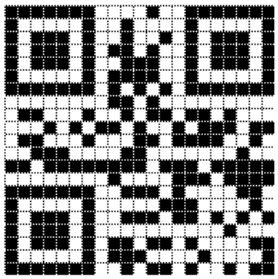
La condition d'évaluation finale est basée sur le rapport des modules clairs aux modules sombres. Pour calculer cette règle de pénalité, procédez comme suit :

- Comptez le nombre total de modules dans la matrice.;
- Comptez combien de modules sombres il y a dans la matrice.;
- Calculez le pourcentage de modules de la matrice qui sont sombres : (darkmodules / totalmodules) * 100;
- Déterminez le multiple précédent et suivant de cinq de ce pourcentage. Par exemple, pour 43
- Soustrayez 50 de chacun de ces multiples de cinq et prenez la valeur absolue du résultat. Par exemple, —40 - 50— = —-10— = 10 et —45 - 50— = —-5— = 5.;
- Divisez chacun d'eux par cinq. Par exemple, 10/5 = 2 et 5/5 = 1.;
- Enfin, prenez le plus petit des deux nombres et multipliez-le par 10. Dans cet exemple, le nombre inférieur est 1, donc le résultat est 10. C'est le score de pénalité 4.;

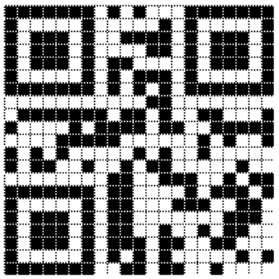
Pour compléter l'évaluation d'un code QR, additionnez les quatre scores de pénalité. Le total est le score de pénalité global du code QR.Les images suivantes montrent huit codes QR, un pour chaque motif de masque. Les huit codes QR de cet exemple encodent les mêmes données. Comme indiqué, le modèle de masque avec le score de pénalité le plus bas est le modèle de masque 0. Par conséquent, dans cet exemple, l'encodeur QR doit utiliser le modèle de masque 0 lors de la sortie du code QR final.



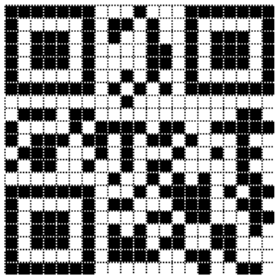
Mask Pattern 0
Penalty 1: 180
Penalty 2: 90
Penalty 3: 80
Penalty 4: 0
Total: 350



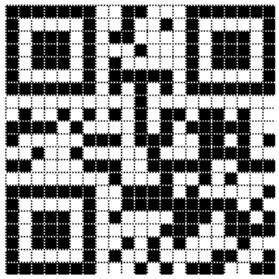
Mask Pattern 1
Penalty 1: 172
Penalty 2: 129
Penalty 3: 120
Penalty 4: 0
Total: 421



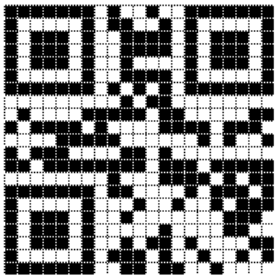
Mask Pattern 2
Penalty 1: 206
Penalty 2: 141
Penalty 3: 160
Penalty 4: 0
Total: 507



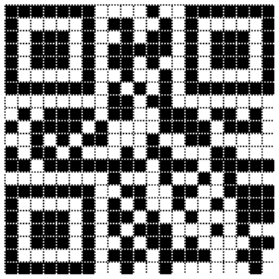
Mask Pattern 3
Penalty 1: 180
Penalty 2: 141
Penalty 3: 120
Penalty 4: 2
Total: 443



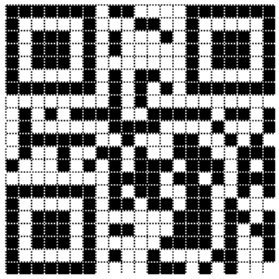
Mask Pattern 4
Penalty 1: 195
Penalty 2: 138
Penalty 3: 200
Penalty 4: 0
Total: 553



Mask Pattern 5
Penalty 1: 189
Penalty 2: 156
Penalty 3: 200
Penalty 4: 2
Total: 547



Mask Pattern 6
Penalty 1: 171
Penalty 2: 102
Penalty 3: 80
Penalty 4: 4
Total: 357



Mask Pattern 7
Penalty 1: 197
Penalty 2: 123
Penalty 3: 200
Penalty 4: 0
Total: 520

Masking sur une même données

5 Chargement de l'image

Pour réaliser la lecture de QR code, il faut d'abord charger cette image que l'algorithme puisse charger l'image du QR code , et donc pouvoir après tout un process de traitement de l'image de pouvoir lire ce dernier. En effet, afin de charger une image plusieurs moyens sont possibles différent programme peuvent être réalise avec différent moyen, nous allons utiliser une méthode déjà vu à EPITA en utilisant la Bibliothèque SDL. Premièrement, nous avons une image dans le format que l'on souhaite nous n'avons plus qu'à appliquer le programme sur cette image. Voici l'image de base que nous souhaitons afficher via le programme que nous créons.

Photo Initial

Pour ce faire, nous initialisations les differentes variables pour afficher l'image tel que la texture, le rendu, la surface et la fenetre dont on precise la dimension sur cette fenetre lors de l'affichage de l'image.

```
SDL_Window *window = NULL;
SDL_Renderer *renderer = NULL;
SDL_Surface *picture = NULL;
SDL_Texture *texture = NULL;
SDL_Rect dest_rect = {0, 0, 640, 480};
```

Les différentes Variables

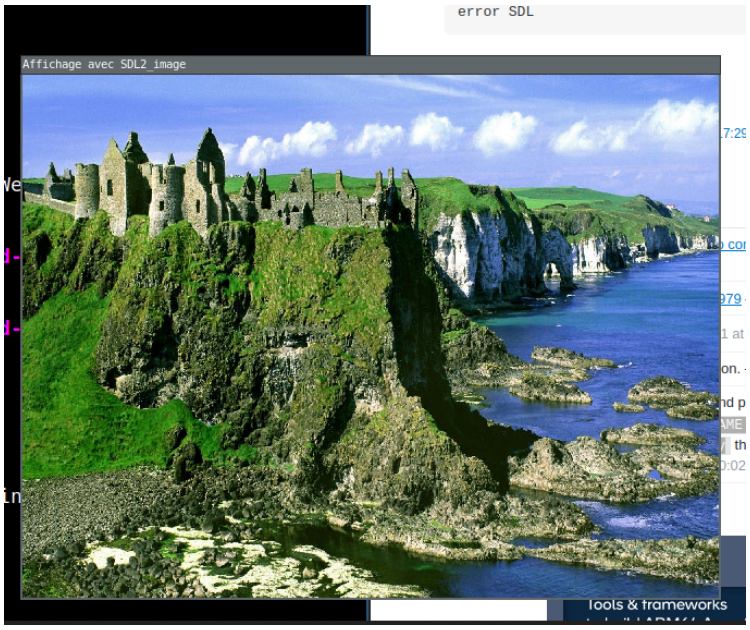
Après cela tout est simple, il suffit de traiter chaques variables et de traiter les possibles cas d'erreur et quand tous ces cas sont traités et donc que le code peut fonctionne, il suffit de copier le rendu puis de l'afficher. Pour rendre cela plus intéressant, on met un délai de 5 secondes lors de l'affichage de l'image.

```
SDL_RenderPresent(renderer);
SDL_Delay(5000);

clean_resources(window, renderer, texture);
return EXIT_SUCCESS;
```

La derniere étape

Et voici ce que donne le résultat final, après exécution du code de l'image initial.



Affichage de L'image

6 Decodage du QR Code

6.1 Lecture du QR Code et ses differentes version

Afin de lire un Qr code, on doit savoir comment ce dernier fonctionne et donc de connaitre parfaitement sa structure, et donc de pouvoir le dechiffrer etape par etape. Pour cela, nous avons analysé la structure globale du QR Code. Le format d'encodage se situe dans le coin inférieur droit de l'image, et est représenté par 4 bits

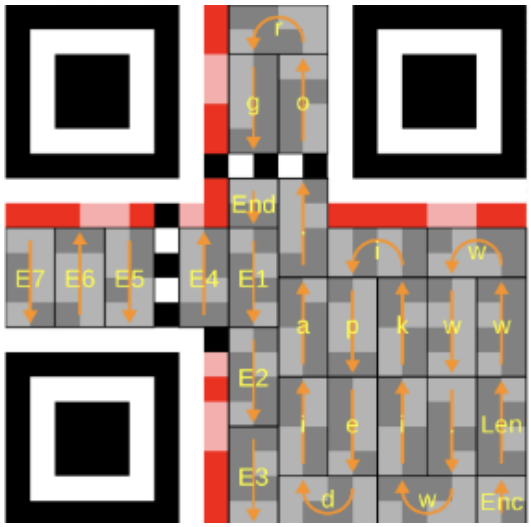


Fig 1 - Structure du QR code, on retrouve la longueur du message ainsi que l'encodage

On peut ainsi décoder le type d'encodage du QR Code, qui sont les suivants: Alphanumérique et Numérique

Quant à la version, elle dépend de la taille du QR Code, c'est à dire du nombre de pixels qui le composent. Un simple parcours nous permet de déterminer la version du QR Code.

6.2 La longueur du message

Dans la même dynamique, nous devons déterminer la longueur du message encodé. Les informations se situent sur les bits au-dessus des quatre pixels de l'encodage, toujours situés en bas à droite du QR Code. (Voir fig 1) Cette longueur de message correspondra aux caractères une fois décodés.

6.3 Le Décodage

6.3.1 Le masque

La première étape de décodage d'un QR Code est son inversion. La lo- gique est d'appliquer un masque gris sur une ligne de pixels sur deux, afin de transformer les pixels noirs en pixels blancs.

Fig 2 - Masque gris sur le QR Code

6.3.2 Lecture des différentes informations

Dans un qr Code d'autres informations pour le decoder sont possible à extraire tel que la version du QR Code, son type d'encodage et enfin la longueur du message. Ces informations permettent de mettre en place différents traitements lors de l'extraction du message du QR code et du décodage des informations extraites.

Parcours du QR Code et extraction du message

6.3.3 lecture du message

Cette partie n'est toujours pas operationnelle etant donnée que nous pouvons lire que la premiere lettre d'un message, le parcours du QR Code etant assez complexe donc l'extraction de ces informations l'est, aussi il faut prendre en compte plusieurs facteurs. Comme l'indique la figure ci contre (fig 3), La partie

regrouppant le message est en bas droite les pixels sont regroupés en blocs de 8 (des octets), et la valeur de chaque bloc permet d'avoir un resultat qui sera interprétée différement selon le type d'encodage du present.

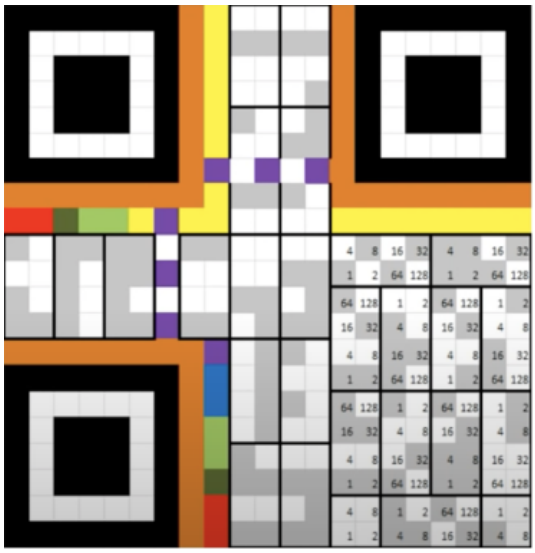


Fig 3 - QR Code et sa partie en octets

7 Interface graphique

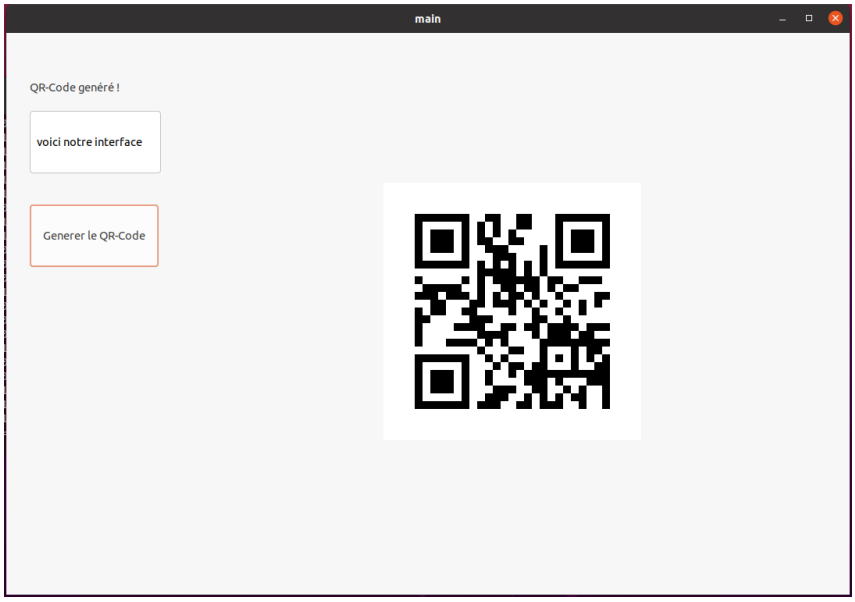
Pour que notre projet soit réellement utilisable et qu'il paraisse moins austère pour les utilisateurs lambda nous avons choisi de réaliser une interface avec la librairie GTK3 en l'associant à l'outil Glade.

La librairie GTK est composée d'un grand nombre de widget qui sont utilisables pour pouvoir modéliser notre interface. Pour pouvoir rendre l'utilisation de GTK plus facile et surtout beaucoup plus visuel le logiciel Glade nous permet de désigner en temps réel le visuel de notre interface.

En effet, Glade permet de générer des fichiers d'interface en .glade qui sont utilisables par GTK. Dans notre code principal il faut penser à bien définir chaque widget avec un nom, qui devra alors être indiqué sur GTK pour ensuite pouvoir les utiliser pour lancer des fonctions par exemple.

Pour pouvoir lancer des fonctions, nous utilisons certains signaux (cliques simples par exemple ou alors clique maintenue) qui peuvent être détectés et utiliser pour des actions précises.

Ces deux outils utilisés ensemble permettant de réaliser des interfaces épurées et simples idéales pour ce genre de projet ou le côté pratique est plus important que le visuel.



Fenetre pour le décodage

8 Site Web

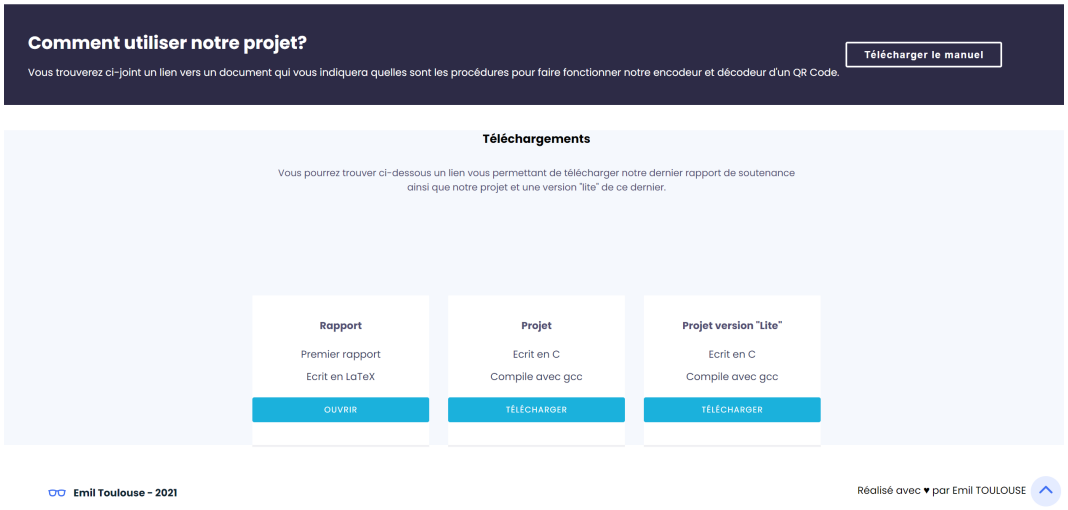
Pour le site web nous avons créer un site avec HTML CSS et JS.

Le HTML forment la structure de la page, et CSS ainsi que JS on pour objectif de rendre la page jolie et dynamique. Le site actuellement n'est pas en ligne mais dans une version locale. Il aura pour but a terme d'être en ligne avec tout les lien pour le téléchargment documentation et support potentiel.

Le but est de faire en sorte que des gens puissent utiliser notre code pour de vrai. Voici des apercu du site acutellement.

Page d'accueil

La chronologie du projet



Possibilité de télécharger

9 Organisation du projet

9.1 Tableau de répartition des tâches

	Emil	Adam	Samy
Chargement d'image			⊕
Pretraitement			⊕
Decodage du QR Code			⊕
Encodage du QR Code	⊕	⊕	
Création d'une image résultat	⊕		
Sauvegarde et chargement d'image		⊕	
Interface graphique		⊕	
Site Web	⊕		

9.2 Planning prévisionnel

	Soutenance 1	Soutenance 2	Soutenance 3
Chargement d'image	100%	100%	100%
Prétraitement	0%	25%	100%
Décodage du QR Code	0%	50%	100%
Encodage du QR Code	75%	100%	100%
Création de l'image résultat	25%	100%	100%
Sauvegarde et chargement d'une image	0%	50%	100%
Interface graphique	0%	25%	100%
Site Web	50%	75%	100%

9.3 Ressenti personnel

9.3.1 Adam Mahraoui

Cette deuxième partie du projet a été assez intéressante pour moi-même si se fut quand même assez complexe sur certains points. La création de l'interface (même si elle n'est pas encore terminée) a vraiment été une partie plutôt ludique pour moi, c'était vraiment plaisant d'avoir un retour visuel de son travail et pas uniquement des résultats sous la forme de 1 ou 0. La partie du remplissage de la matrice fut un peu plus compliquée et casse-tête mais tout aussi intéressante. Ce projet devient de plus en plus concret et utilisable ce qui me motive toujours un peu plus à continuer. L'objectif pour la dernière soutenance est bien évidemment de rendre un projet fini et fonctionnel avec une meilleure interface graphique.

9.3.2 Emil Toulouse

Cette seconde partie de projet à été une réelle opportunité de rattraper nos erreurs et retard pris lors de la premiere écheance.J'ai beaucoup aimé me pencher sur ce qu'etait un QR-code. Mon objectif pour la prochaine soutenance est de réussir à avoir un programme finis avec un rendu satisfaisant pour moi . C'est l'occasion pour moi de faire un projet qui à un réel interet et qui montre ma capacité de créer avec du code.

9.3.3 Samy Achache

Ce projet pour ma part est très intéressant, en effet l'avantage de ce dernier est de choisir son sujet on s'est mis d'accord sur un sujet qui nous intéressait beaucoup la lecture de QR code qui depuis peu nous entoure quotidiennement dans notre vie. Le fait qu'un membre du groupe parte vu qu'il effectue son S4 à l'étranger, a été certes un inconvénient et une difficulté en plus mais cela nous a permis de mieux nous souder et de nous répartir les nouvelles taches. Pour cette deuxième soutenance, malgré les différents problèmes rencontres nous avons vite trouve leurs solutions et nous somme complètements opérationnels et j'espère que nous allons réussir à remplir tous nos différents objectif fixe pour la dernière soutenance.

10 Conclusion

Le temps écoulé entre la première soutenance et la deuxième soutenance, nous a réellement permis de progresser, nous avons pu réaliser entièrement la partie encodage et eu une bonne partie de décode. De plus, à la suite des différents problèmes rencontre et malgré l'absence d'un de nos membres, nous avons pu trouver des solutions, cela ne nous a été que bénéfique, nous avons gagné une meilleure connaissance du langage c et l'entente de notre groupe réduit s'est encore amélioré. Les délais sont respectés et nos objectifs le sont aussi une interface graphique bonne mais un encodage et décodage fonctionnels. Nous sommes optimistes sur la faite d'avoir tout notre projet complètement fonctionnel pour la dernière soutenance.