Rapport de Projet

adam.mahraoui
emil.toulouse
samy.achache

16 Mai 2022

Sommaire

1 Introduction

1.1 Présentation du projet

En cette période de pandemie mondiale, une technique plutôt ancienne s'est vu utilisée par une grande partie de la population Mondiale est bien le QR-Code. En effet, il a été très utilisé par la plupart des gouvernements notamment la France pour pouvoir transmettre des informations facilement, partout et sans perte de donnée.

Notre projet de S4, est un logiciel dans lequel l'algorithmique a une part très importante, il s'agira donc d'étudier le QR-code. 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-codes 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.

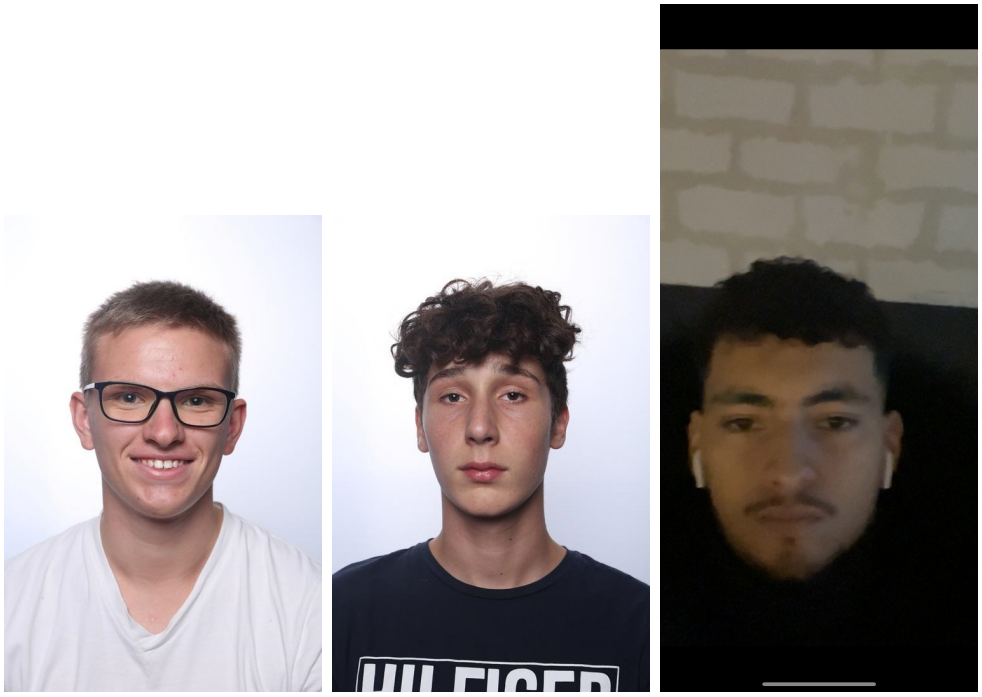
1.2 Qui sont les membres du groupe ?

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 tres 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 répondront 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 convertir un lien en un QR code ;
- Réaliser une interface pour le logiciel ;

3 Etat de l'art

3.1 Encodage d'information

3.1.1 Les premiers encodages

Une des premières manières d'encoder une information a été de décaler l'alphabet un certain nombre de fois d'un message texte pour que celui-ci reste secret. En effet, le seul moyen de lire le message était de savoir le nombre de décaillage de l'alphabet pour pouvoir accéder à la vraie information. Cette méthode fut utilisée par Jules Cesar dans ses correspondances secrètes.

C'est à la fin de la première guerre mondiale qu'est apparue la nécessité de crypter les messages (bien que les techniques de chiffrement existaient déjà depuis fort longtemps).

C'est un Néerlandais résidant en Allemagne, le Dr Arthur Scherbius qui mit au point à des fins commerciales la machine Enigma, servant à encoder des messages.

Le modèle A de la machine fût présentée en 1923 au Congrès Postal International de Berne. L'idée fit son chemin et la marine de guerre allemande reprit le projet en 1925 et en confia son évolution au service du chiffrement du ministère de la guerre allemand. Le modèle Enigma M3 fût finalement adoptée par la Wehrmacht le 12 janvier 1937.

Enigma possédait un fonctionnement particulièrement simple : l'objet était équipé d'un clavier pour la saisie du message, de différentes roues pour le codage, et enfin d'un tableau lumineux pour le résultat.

A chaque pression d'une touche du clavier, une lettre du panneau lumineux s'illuminait. Il y avait ainsi 3 roues de codage, appelées Brouilleur Rotor , qui reliaient le clavier au panneau lumineux. Les machines Enigma peuvent donc chiffrer un texte selon $10^16combinaisonsdifférentes$!

3.1.2 les premiers encodages informatiques

En tant qu'étudiants à L'EPITA nous avons eu besoin d'utiliser le protocole SSH afin de pouvoir envoyer ou recevoir nos TP d'informatique. Ce protocole utilise deux types de clé : une clé publique et une clé privée. La clé publique est distribuée sur les systèmes auxquels on souhaite se connecter quant à la clé privée elle reste uniquement sur le poste à partir duquel on se connecte. La clé privée doit rester secrète et doit être gardée précieusement par l'utilisateur. Un agent ssh permet de stocker le mot de passe de la clé privée pendant que la session de l'utilisateur reste ouverte.

3.1.3 Encodage d'un QR-code

L'encodage se fait en 3 étapes :

- Créer une chaîne de bits de données à partir du message que l'on veut encoder. Cette chaîne inclut les caractères du message d'origine que l'on veut encoder, ainsi que quelques bits d'information qui disent au décodeur QR quel type de code QR c'est.
- Générer le code correcteur d'erreurs (redondance). Il s'agit d'un processus complexe, visant à assurer la validité de la lecture de l'information même si le QR-code subit des dommages
- Choisir la meilleure forme de masque afin d'optimiser la lecture du QR code.

3.2 Decodage d'information

3.2.1 Les premiers decodages

Pour le cassage d'enigma, les Polonais inventèrent la Bombe qui permettait de connaître les réglages Enigma. Seulement, à partir de 1938, c'est l'opérateur lui-même qui établissait le réglage. Pour remédier à ce problème, les polonais trouvèrent la solution: chaque message contenait soit une répétition de mots soit des mots récurrents (appelés femelles).

Ceci était un indice quant au noyau (réglage de base des rotors). Pour découvrir ce réglage, les Polonais utilisaient ensuite la Grille (cartes perforées correspondant à toutes les permutations du noyau). Ces cartes étaient empilées les unes sur les autres par rapport à la position des femelles .

Ensuite, il s'agissait de chercher le point où une série de perforations se recouvrait du haut en bas de la pile.

3.2.2 Decodage d'un QR-code

Avec l'encodage spécifique d'un QR-code on peux retrouver facilement le format verifier celui-ci recouper l'information contenu dedans en desmasquant les données pour en fin les extraires et revenir aux messages initial ou peux importe ce que contenait le code.

3.3 Représentation de l'information

L'information est de nos jours, retransmise de mille et une façons. On peut d'abord penser au traditionnel journal de papier que nos parents et grands-parents transportaient constamment avec eux pour pouvoir y lire les dernières nouvelles. Ensuite venait la révolution des postes télévision et leurs chaines qui pouvaient y transmettre en continu des flots illimités de contenu de tous types. Vint ensuite avec l'avènement d'internet et de l'informatique la dernière

grande révolution dans la circulation de l'information, n'importe qui peut y transporter n'importe quel type d'information via divers moyens tels que les réseaux sociaux, les différents forums, les plateformes de partage de vidéos etc... Et avec la crise sanitaire, on peut voir un nouveau moyen déjà bien établi avant certes mais qui l'est encore plus actuellement : Le QR code, avec un simple smartphone et ces quelques carrés blancs et noirs, on peut accéder à tous types d'informations allant du nom et prénom d'une personne au menu du restaurant dans lequel on se trouve.

3.4 Présentation de l'information a un utilisateur

Pour decoder un QR-Code, un utilisateur lambda peut utiliser son smartphone, scanner le QR Code depuis sa caméra ou alors en téléchargent une application spécifique. Il peut aussi via des applications spécifiques charger une image depuis son téléphone et la escaner via l'application pour obtenir l'information.

Pour transformer créer lui-même un QR Code, il est possible d'utiliser différent site web qui transformera l'information en une image téléchargeable contenant le QR code

4 Description fonctionnelle

4.1 Nos ambitions

Comme dit précédemment, il s'agit d'un logiciel. Celui-ci aura pour but premier la création et lecture de QR-code. Pour plus de practicité ce logiciel sera accompagné de son interface graphique ayant pour objectif principal de rendre la navigation entre création et lecture simple. De plus nous souhaitons ajouter une possibilité de récupérer des QR-codes anciennement créer pour les réutiliser à d'autres occasions ou si l'ancien a subi trop de dommages. Tout cela sans devoir refaire entièrement le QR-code. Il s'agira d'un dossier spécifique où l'utilisateur du programme pourra retrouver tous les QR-code qu'il aura créé.

4.2 Organisation pratique

Pour ce projet, nous comptons utiliser les ressources que l'on a pu utiliser par le passé, via le language C. Nous comptons utiliser les bibliothèques SDL que l'on utilisera pour le traitement de l'image et GTK pour l'interface car nous avons déjà pu nous faire la main dessus via notre S3. Pour l'architecture du projet nous allons utiliser un repo GIT qui nous permettra d'avoir une bonne structure via la forme arborescente de l'architecture GIT

5 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 compléter le QR-code. Le principe est simple sur le papier et plus compliqué dans sa mise en place. En effet la création d'un QR-code (les données contenues) doivent respecter un certain pattern et des paramétrages bien précis pour que les résultats finaux soient utilisables.

5.1 Choix du mode et de du niveau de correction

Les QR-codes peuvent stocker un certain nombre d'informations et de types différents. Le choix d'un mode correspond à la palette de caractères accepter pour l'encryptage 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

Parmi ces 3 choix nous avons retenu le plus complet (Byte mode) qui prendra en compte tous les caractères de la table ASCII classique. Cependant nous avons pris le choix de ne pas prendre en compte un quatrième mode le Kanji Mode puisqu'il inclut des caractères supplémentaires que nous avons jugés peut 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, même après dégradation partielle, il reste opérationnel. Nous avions le choix entre plusieurs paramétrages et nous avons retenu le mode M qui une fois mise en place dans sa totalité pourra être utilisable même après 15 pourcents de dégradation.

5.2 Prémice de l'encodage

Nous avons opté pour réaliser une structure contenant tous les possibles paramètres de l'encodage du QR-code avec notamment sa taille, sa version, la donnée traitée 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 passer 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'octets et que donc en connaissant ce tableau des versions nous pouvons facilement trouver la version adaptée.

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

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

Pour respecter les nomenclatures nous devons impérativement faire en sortes que cette information soit représentée sur 8 bits. Nous ajoutons donc des 0 pour que cela correspond aux demandes.

Pour le byte mode, chaque caractère alphanumérique est représenté par un nombre .Pour chaque nombre nous allons devoir trouver sa représentation binaire sur 8 bits. Si la représentation est plus courte que prévue nous allons tout simplement ajouter des 0 par la gauche de notre nombre afin de compléter et arriver à 8 bits sans changer la valeur.

Pour avoir la bonne représentation de l'information la correction d'erreur impose une certaine taille pour la data encodée. Ainsi avec le mode choisi nous pouvons déterminer cette taille. Un tableau est fourni avec le nombre de bloc erreur nécessaire. On multiplie par 8 ce nombre pour obtenir la longueur nécessaire de la chaine data. Une fois cette longueur de chaine connue nous devons donc remplir les espaces manquants. On commence par ajouter aux maximums quatre zéros pour tomber remplir la taille nécessaire. Si ce n'est pas le cas on met quatre 0.

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

5.3 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 valeurs numériques qui seront utilisées comme coefficients du polynôme du message.

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

0100 00001011 01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100 01100100 00001110 11000001 00011110 11000100 00001111 11000111 00010100 00011101 01011000 11110010 01101101 01001111 01000011 0011

Le codage de correction d'erreur utilise une division de polynôme. Pour ce faire, deux polynômes sont nécessaires. Le premier polynôme à utiliser est appelé le polynôme de message.

On convertit 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 un polynôme messager qui contient notre information sous une forme bien spécifique. Le message polynôme sera divisé par un polynôme générateur. Le polynôme générateur est un polynôme qui est créé en multipliant (x - 0) ... (x - n-1) où n'est pas le nombre de mots de code de correction d'erreur qui doit être générée. Dans notre cas particulier à=2. Maintenant il est temps de diviser le polynôme messager par le polynôme de générateur. Ceci est fait de la même manière que la division de polynôme. Nous divisons autant de fois que la plus grande puissance du polynôme messager. Afin d'obtenir des valeurs de puissance similaire au polynôme générateur et de reprendre les coefficients comme valeur de correction finale. Avec par exemple pour HELLO WORLD :

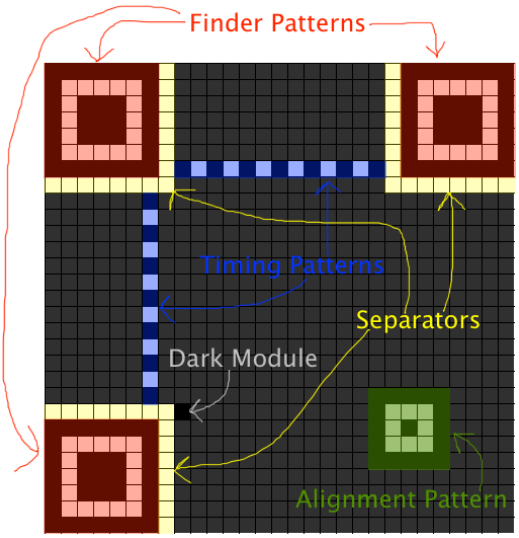
196 35 39 119 235 215 231 226 93 23

Nous reprenons donc ces valeurs et nous les mettons sous leur représentation binaire. Cette suite de caractères va être ajoutée à la longue chaine précédemment utilisée pour encoder la data. Nous avons donc la représentation finale de la donnée assez indigeste en binaire d'Hello World. Maintenant créons une image de QR-code à partir de ses données.

6 Création 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).

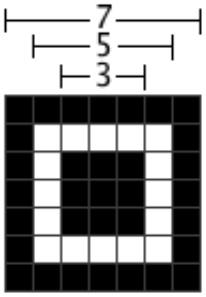
Tous les QR-Codes possèdent les mêmes éléments, seuls leurs emplacements diffèrent en fonction de taille et donc de leur version. On peut retrouver alors 4 paternes : les paternes de détection de position, les séparateurs, les "timings patterns" et les paternes d'alignements.



Structure d'un QR code

6.1 Paternes de détection de position et les séparateurs

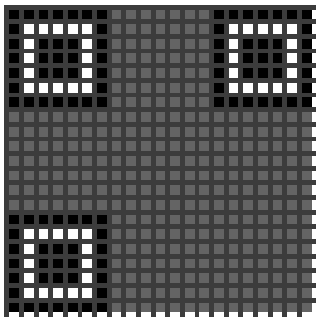
Les paternes de détection de position sont les 3 gros carrés visibles très facilement sur un QR-code et ils sont au nombre de 3. Un paterne se compose d'un carré noir extérieur de 7 pixels sur 7 pixels, d'un carré blanc intérieur de 5 pixels sur 5 pixels et d'un carré noir uni au centre de 3 pixels sur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

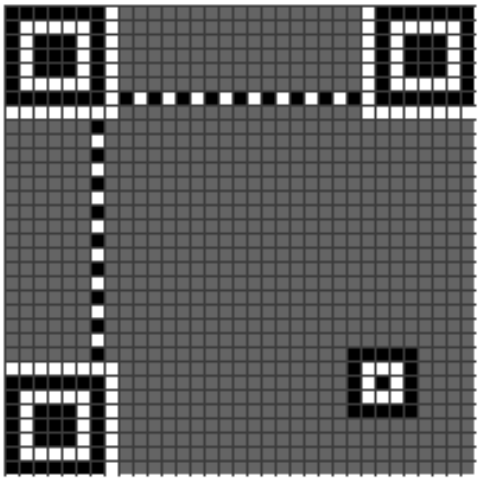


Localisations des paternes de détection de position

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

6.2 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"

6.3 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.

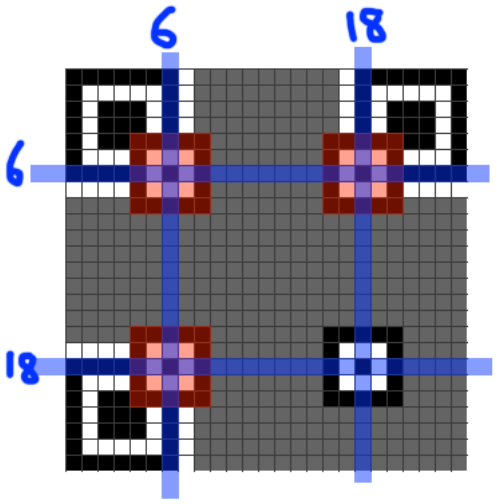


Illustration des coordonées possible des paternes d'alignement

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);
```

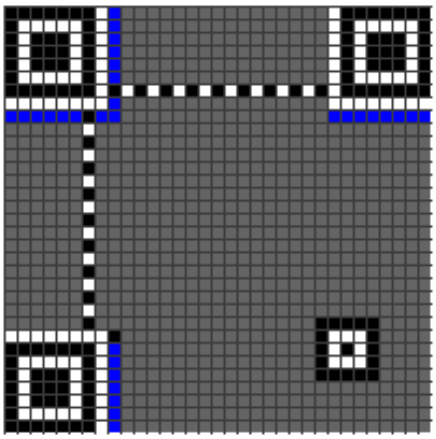
Condition pour placer un paterne avec comme centre (a,b)

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.

6.4 Les reserveds areas

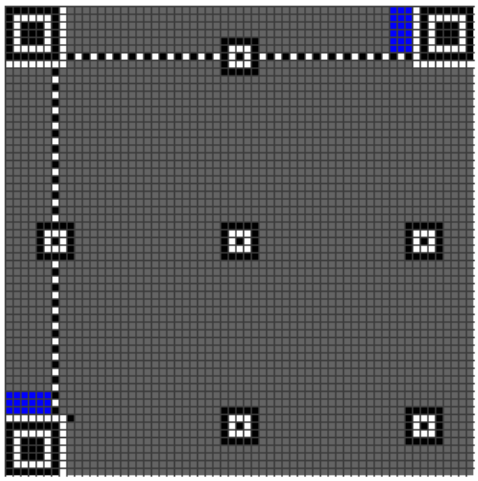
Les “reserved areas” ou nommées en Français “zones de réserve” sont des parties du QR codé qui doivent être marquées pour les différencier des pixels qui sont dit “libres” et donc où la data de ce que l'on doit encoder peut-être placé. Ces zones de réserve seront utilisées pour stocker la version et 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

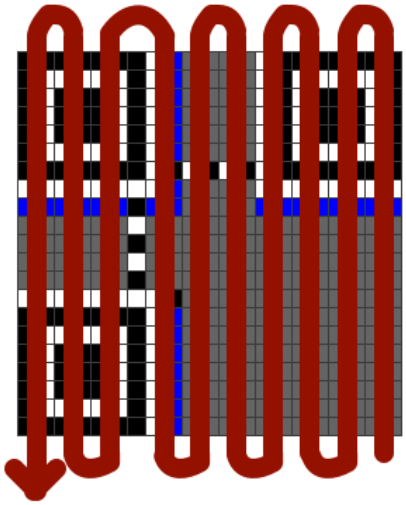
Pour les QR-Codes possédant une version supérieure ou égale à 7, deux autres zones de réserve doivent être marquées pour pouvoir stocker la version. Les zones sont deux rectangles de 6x3 au-dessus du paterne de détection de position inférieure gauche et à droite du paterne de détection de position supérieure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



Localisation des zones de reserve pour le format

6.5 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 traverserait 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. Chaque fois que notre colonne de remplissage a atteint un bord de la matrice, il faut passer 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.

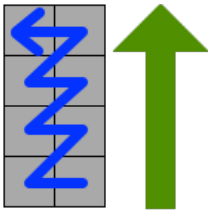


Illustration du remplissage dans la premiere colonne

6.6 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 tous 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.

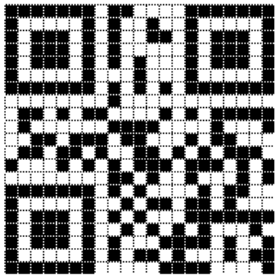
La troisième règle de pénalité recherche des modèles sombre-clair-sombre-sombre-sombre-clair-sombre qui ont quatre modules lumineux de chaque côté.

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 :

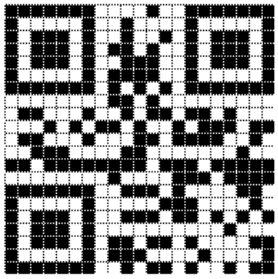
- Comptez le nombre total de modules dans la matrice.;
- Comptez combien de modules sombres il y a dans la matrice.;
- Calculez le pourcentage de modules de la matrice qui sont sombres : (darkmodules / totalmodules) * 100;
- Déterminez le multiple précédent et suivant de cinq de ce pourcentage. Par exemple, 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.;

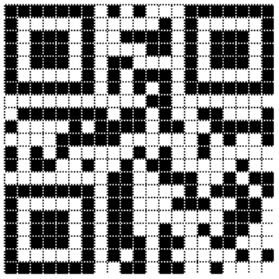
Pour compléter l'évaluation d'un code QR, additionnez les quatre scores de pénalité. Le total est le score de pénalité global du code QR.Les images suivantes montrent huit codes QR, un pour chaque motif de masque. Les huit codes QR de cet exemple encodent les mêmes données. Comme indiqué, le modèle de 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.



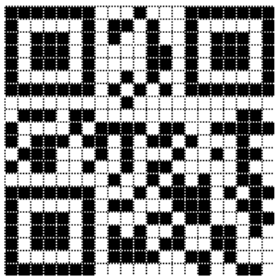
Mask Pattern 0
Penalty 1: 180
Penalty 2: 90
Penalty 3: 80
Penalty 4: 0
Total: 350



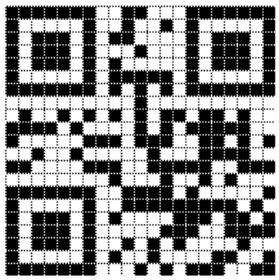
Mask Pattern 1
Penalty 1: 172
Penalty 2: 129
Penalty 3: 120
Penalty 4: 0
Total: 421



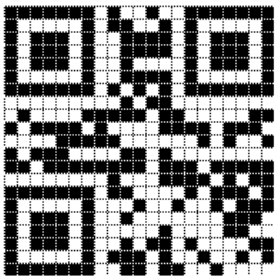
Mask Pattern 2
Penalty 1: 206
Penalty 2: 141
Penalty 3: 160
Penalty 4: 0
Total: 507



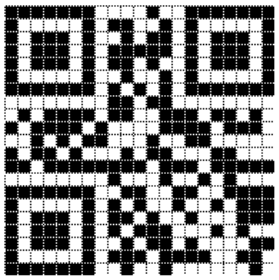
Mask Pattern 3
Penalty 1: 180
Penalty 2: 141
Penalty 3: 120
Penalty 4: 2
Total: 443



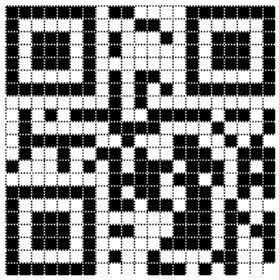
Mask Pattern 4
Penalty 1: 195
Penalty 2: 138
Penalty 3: 200
Penalty 4: 0
Total: 553



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

7 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 processus 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 réalisés avec différents moyens, nous allons utiliser une méthode déjà vue 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 différentes variables pour afficher l'image telle que la texture, le rendu, la surface et la fenêtre dont on précise la dimension sur cette fenêtre 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};
```

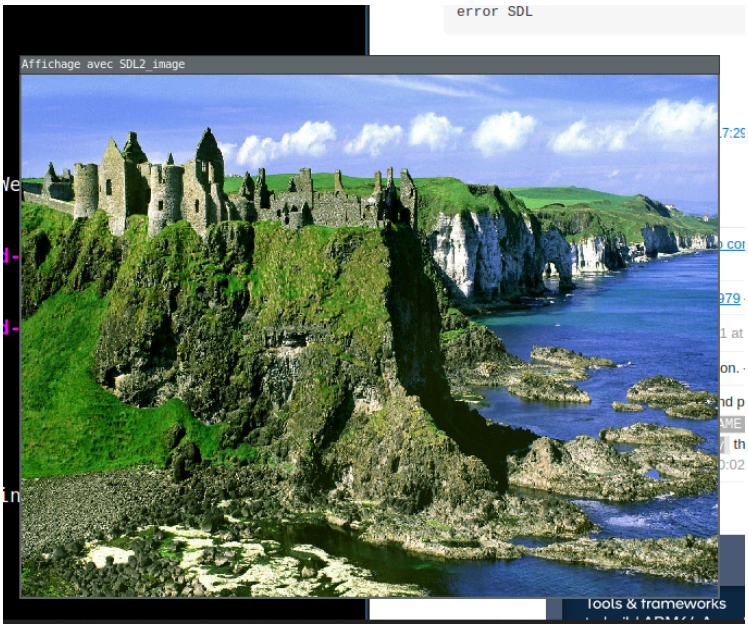
Les différentes Variables

Après cela tout est simple, il suffit de traiter chaque variable et de traiter les possibles cas d'erreur et quand tous ces cas sont traités et donc que le code peut fonctionner, 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

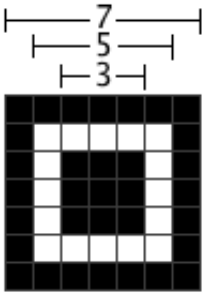
8 Decodage du QR Code

8.1 Convertion de l'image en matrice

Avant toute chose nous devons trouver un moyen de récupérer chaque pixel utile du QR code contenu dans l'image. Nous avons fait le choix de récupérer l'information sous forme de matrice qui sera utile pour le décodage.

Maintenant que nous savons ce que nous cherchons et que nous savons la forme que nous allons donner à nos informations il faut traiter l'image pour en déduire ce que l'on souhaite. On commence donc par parcourir l'image (noir et blanche uniquement) pixel par pixel à la recherche d'un pixel noir. Ce premier pixel noir aura pour intérêt de nous donner la taille de zone morte autour du QR code mais aussi les coordonnées du coin supérieur gauche du QR codent. Grâce à ces coordonnées nous pouvons aussi déterminer la taille du QR code. La zone morte étant de la même taille de part et d'autre du QR code il suffit de prendre la taille de l'image et de retirer la taille deux fois des zones mortes.

Nous poursuivons notre parcours jusqu'à retomber sur un pixel blanc. Le but est donc de déterminer la longueur en pixels du carré noir large de 7 cases selon la documentation. Cette longueur divisée par 7 nous donne la résolution des carrés du QR codé contenu dans l'image. Par exemple si j'ai besoin de 70 px pour parcourir toute la bande noire, j'aurais une résolution de 10. Ainsi les carrées du QR codé font 10x10 px.



Echelle des carrés du QR code

Maintenant que nous avons toutes les informations nous pouvons faire un second parcours mais cette fois plus cibler. Nous allons commencer notre parcours aux coordonnées du coin supérieur gauche que nous avons trouvé auparavant. et nous allons nous déplacer vers la droite de 10 px en 10 px car la résolution du carré dans notre exemple est de 10. On peut donc adapter le parcours à cette variable afin que toutes les représentations de QR codent fonctionné. Nous récupérons les informations de la couleur de chacun des carrés du QR code que nous stockons dans une chaine de caractères.

C'est ainsi que nous pouvons transformer une image de QR code en une matrice qui sera utilisable et utile pour le programme de décodage.

8.2 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 déchiffrer étape par étapes. 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

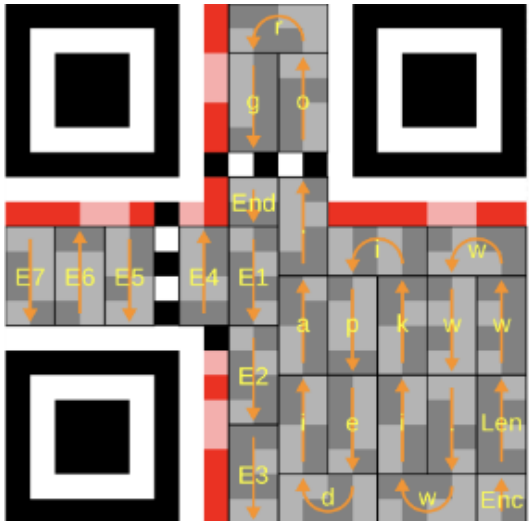


Fig 1 - Structure du QR code, on retrouve la longueur du message ainsi que l'encodage

On peut ainsi décoder le type d'encodage du QR Code, qui sont les suivants: Alphanumérique, Numérique, Bytes mode et Kanji (représentant les caractères spéciaux mandarin par exemple).

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

8.3 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 situé en bas à droite du QR code. (Voir fig 1) Cette longueur de message correspondra au nombre de caractères une fois le message décodé.

8.4 Le Décodage

8.4.1 Le masque d'inversion

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. Le principal défi que nous avons eu à affronter par rapport au masque est le choix pair ou impair de lignes. Cette notion floue et très peu documentée nous a donné du fil à retordre, et de nombreuses heures de corrections de code et de tests ont testé nos nerfs.

Fig 2 - Masque gris sur le QR Code

8.4.2 Lecture des différentes informations

Dans un QR code d'autres informations sont à extraire. Nous sommes donc à la recherche de la version du QR Code, son type d'encodage et enfin la longueur du message. Ces informations permettant de mettre en place différents traitements lors de l'extraction du message du QR codent et du décodage des informations extraites.

8.5 Parcours du QR Code et extraction du message

8.5.1 Parcours du QR Code : quatre parcours pour des QR Codes simples

Comme vous avez pu le voir avec le premier schéma montrant le sens de la lecture d'un QR Code, il y a trois directions à traiter : — Un premier parcours vertical d'un octet, de bas vers le haut — Un parcours horizontal d'un octet, pour la partie haute du QR Code, de droite à gauche — Un autre parcours horizontal pour la partie basse du QR Code, de droite à gauche — Un second parcours vertical d'un octet, de haut en bas Le parcours permet d'extraire les valeurs contenues dans chaque octet codé. Les valeurs à l'intérieur de chaque octet dépend du parcours que nous souhaitons effectuer, comme il est possible de le remarquer sur la figure suivante (Fig 3)

8.5.2 lecture du message

Après avoir extrait les valeurs de chaque octet, l'objectif est de le stocker dans une liste avant de convertir chaque valeur extraite (suivant l'encodage choisi pour le QR Code), et renvoyer la valeur finale du message alors décodé à l'utilisateur. Les parcours ont été modifiés afin de permettre une implémentation plus pratique et efficace de la fonction de parcours principale. Maintenant tout type de QR Code peut-être défricher via à la réalisation de ses différents parcours en code.

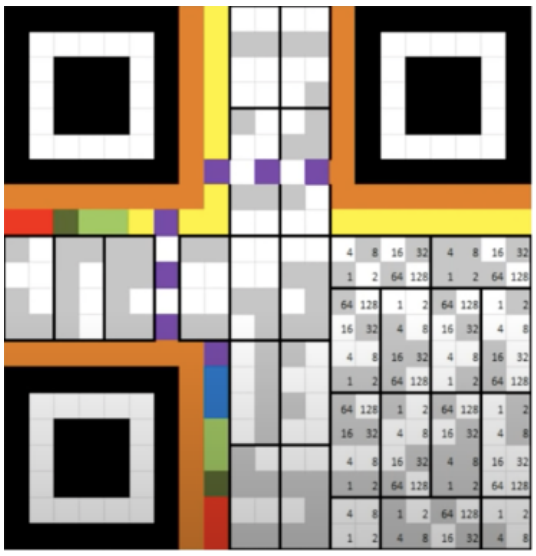


Fig 3 - QR Code et sa partie en octets

L'inversion du QR Code et l'implémentation des différents parcours étant réalisé, il reste à mettre en place la fonction principale, qui va utiliser toutes les informations que nous avons pu recueillir jusqu'à maintenant, afin de faire ressortir le message.

9 Interface graphique

9.1 Premiere conception de l'interface

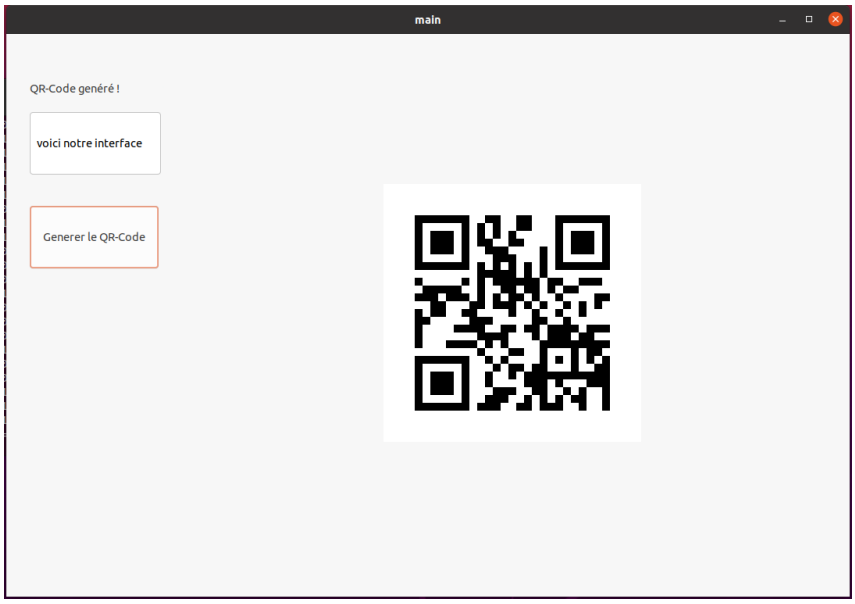
Pour que notre projet soit réellement utilisable et qu'il paraisse moins austère pour les utilisateurs lambda nous avons choisi dans un premier temps de réaliser une interface avec la librairie GTK3 en l'associant à l'outil Glade.

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

9.2 Conception finale de l'interface

Après après avoir réalisé dans un premier temps une seule fenêtre de l'interface, je me suis mis à avancer dessus. En effet, pour la toute première partie sur l'encodage d'un QR-Code j'avais utilisé l'outil Glade en association avec la librairie GTK. Cependant lorsque j'ai voulu faire l'interface finale du projet en créant donc les 2 fenêtres manquantes (l'accueil principal et la partie de décodage d'un QR-Code) je me suis aperçu que c'était plus compliqué que ce que je pensais. J'ai donc décidé de tout recommencer au propre pour la soutenance finale.

Pour ce faire j'ai utilisé uniquement la libraire GTK sans l'aide de glade. Pour ce faire j'ai commencé par créer un la fonction main qui va permettre de lancer notre application et ainsi afficher notre fenêtre principale.

Pour la fenêtre principale j'ai créé une fonction qui se décompose en plusieurs petites parties. J'ai commencé par la déclaration de la fenêtre générale window en tant que fenêtre générale . Ensuite j'ai déclaré tous les widgets qui vont composer mon interface (bouton, image, zone de texte et box. . .)

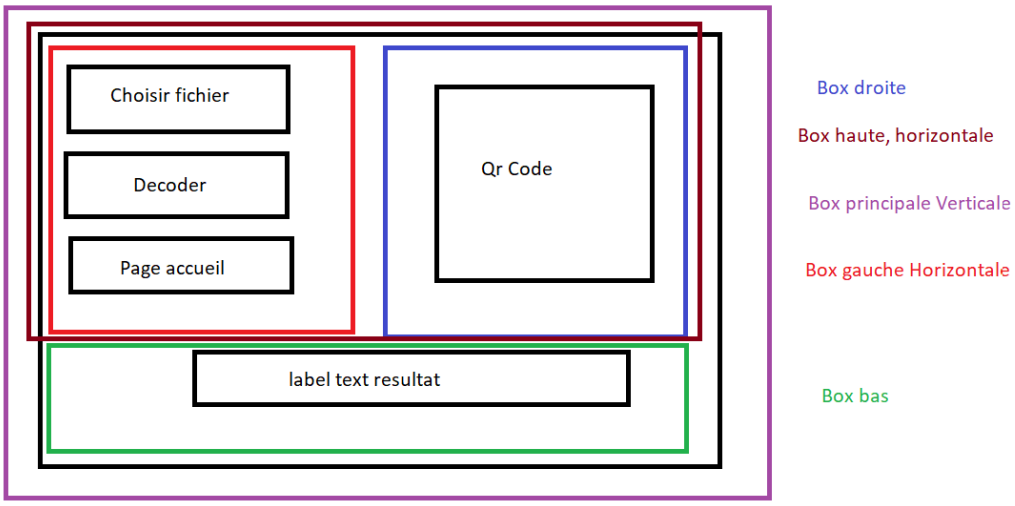
```
GtkWidget * generale_box_generale = gtk_box_new(GTK_ORIENTATION_VERTICAL, 50);

GtkWidget * generale_box_haut = gtk_box_new(GTK_ORIENTATION_VERTICAL, 50);

GtkWidget * generale_box_bas = gtk_box_new(GTK_ORIENTATION_HORIZONTAL, 50);

GtkWidget * generale_bouton_decode = gtk_button_new_with_label("Decoder un QR-Code");

GtkWidget * generale_bouton_encode = gtk_button_new_with_label("Generer un QR-Code");

GtkWidget * generale_logo  = gtk_image_new_from_file("logo.jpg");
```

Detection des signaux de la fenetre principale

Par la suite j'ai découpé la fenêtre en plusieurs gtk box qui compose notre page d'accueil et qui permet de stocker les différents widgets. Les box peuvent être de deux types : horizontales ou verticales, cela indique comment sont affichés widgets s'il y en a plusieurs. C'est avec ces paramètres et le nombre de box que nous pouvons jouer pour constituer notre interface. Cette partie se fait d'abord au préalable sur papier pour avoir une idée bien précise de la découpe pour ensuite la retranscrire à l'écrit.

```
gtk_container_add(GTK_CONTAINER (generale_window), generale_box_generale);

gtk_container_add( GTK_CONTAINER (generale_box_generale), generale_box_haut);

gtk_container_add( GTK_CONTAINER (generale_box_generale), generale_box_bas);

gtk_container_add( GTK_CONTAINER (generale_box_bas), generale_bouton_decode);

gtk_container_add( GTK_CONTAINER (generale_box_bas), generale_bouton_encode);

gtk_container_add( GTK_CONTAINER (generale_box_haut), generale_logo);
```

Creation des box pour la page d'accueil



Schema de decoupage de la fenêtre de décodage

Une fois les box établis et ajoutés correctement il ne reste plus qu'on y mis les widgets un à un en faisant attention à l'ordre d'ajout pour qu'il soit cohérent avec le résultat voulu sur l'interface. Il faut aussi dans certains cas ajouter des marges pour aligner et déclarer les boutons au sein des box pour rendre le tout plus esthétique et ergonomique.

Une fois que le visuel nous convient il ne reste plus qu'à donner une utilité aux différents boutons. Pour cela il faut alors détecter les signaux émis par l'utilisateur pour ensuite les lier à des actions comme par exemple ouvrir une nouvelle fenêtre ou alors afficher une image.

```
//detection signaux

g_signal_connect(generale_bouton_encode,"clicked", G_CALLBACK(create_encode),NULL);

g_signal_connect(generale_bouton_decode,"clicked", G_CALLBACK(create_decode),NULL);
```

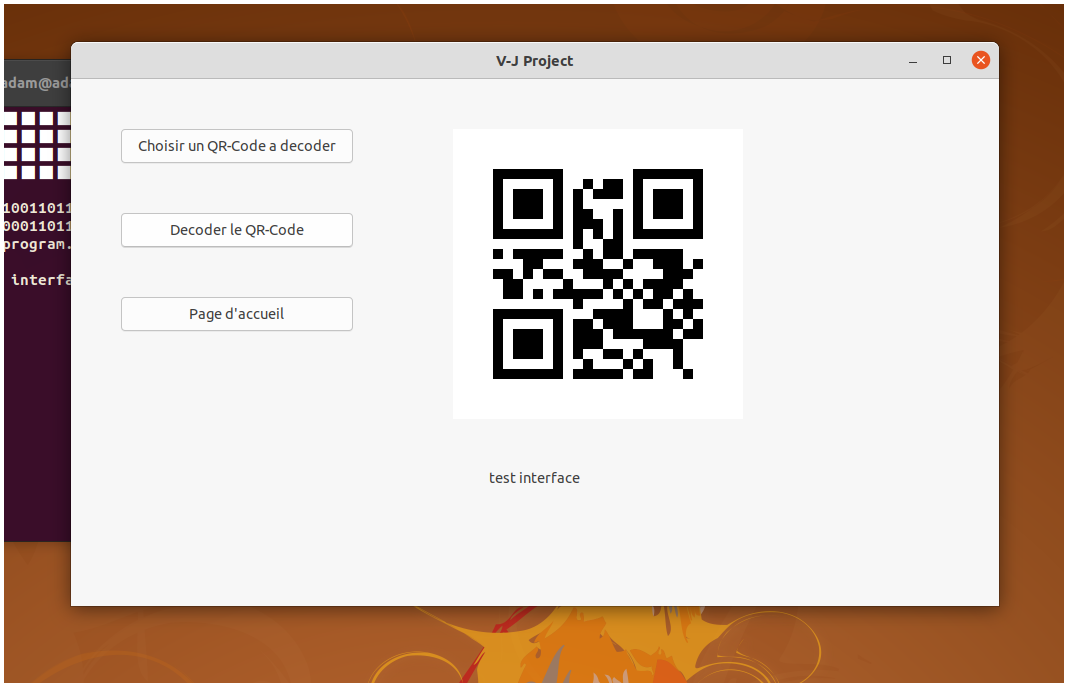
Detection des signaux de la fenetre principale

Toute cette fonction ainsi que ces étapes est a reproduire pour chaque fenêtre. Pour ce qui est de l'ajout de la fonction de d'encodage : Lorsque l'utilisateur écrit du texte dans le widget entry et qu'il click ensuite sur le bouton encoder nous détectons alors le signal et nous alors par la suite appeler notre fonction d'encodage avec comme paramètre le texte écrit. Par la suite nous affichons alors l'image résultat.

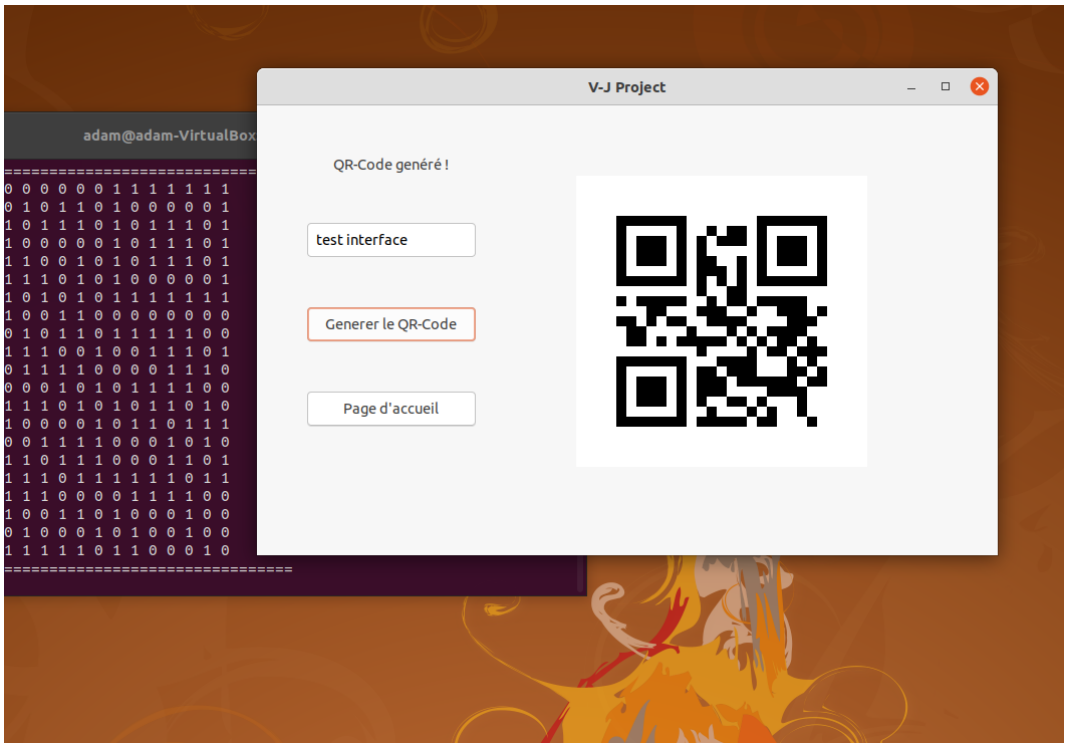
Pour ce qui est du décodage, une fois que l'utilisateur choisit une image et clique sur le bouton décodage nous appelons alors une fonction qui va permettre de transformer notre grille en matrice de 1 et 0 pour ensuite le décoder et afficher le résultat obtenu.

L'avantage à ne pas utiliser Glade c'est que le code est plus clair et plus lisible cependant le travail est plus long et assez redondant sans retour visuel instantané.

Page d'accueil



Page de décodage



Page d'encodage

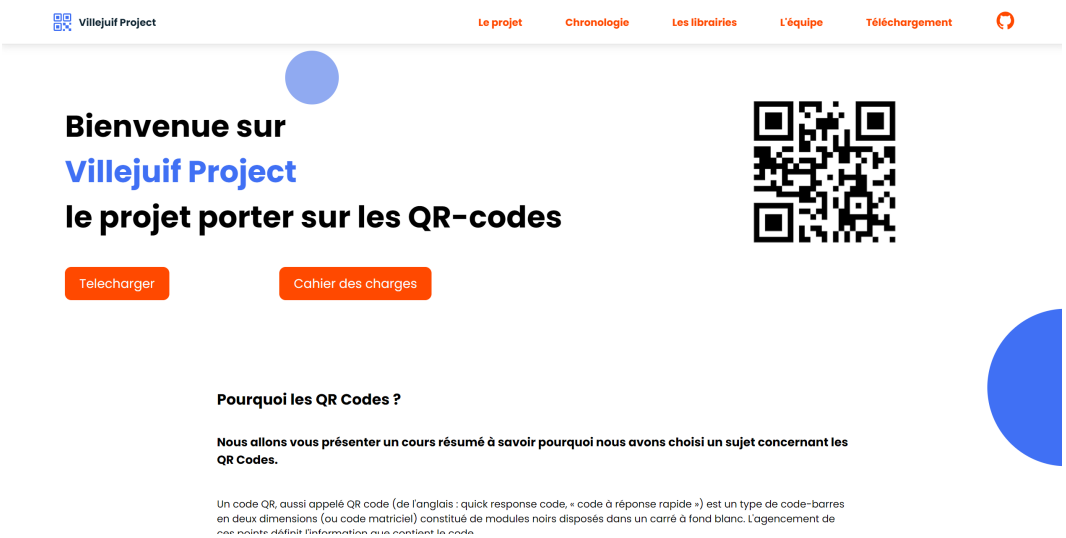
10 Site Web

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

Le HTML forme la structure de la page. Nous avons ajouté du CSS ainsi que JS pour rendre la page jolie et dynamique. Le site actuellement est sur le internet et est accessible depuis emiltoulouse.com et cela pour 2 mois. Il y a des informations sur les groupes, les motivations du projet ainsi que de multiples liens pour le téléchargement documentation et support potentiel pour l'installation ou encore des rapports d'avancement.

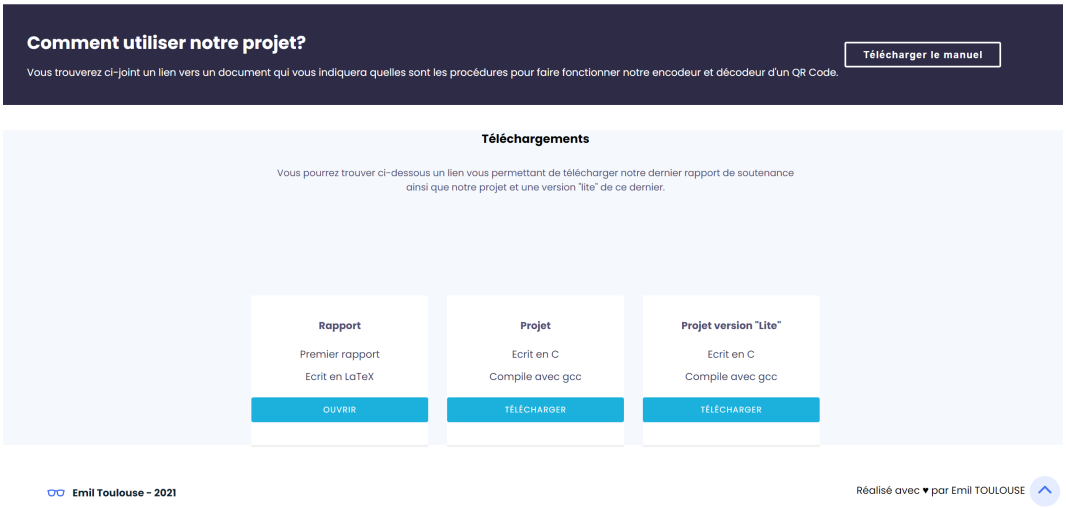
Le but est de faire en sorte que des gens puissent utiliser notre code. Et installer notre projet sur leur machine Linux chez eux pour essayer ou utiliser notre projet.

Voici des aperçus du site actuellement.

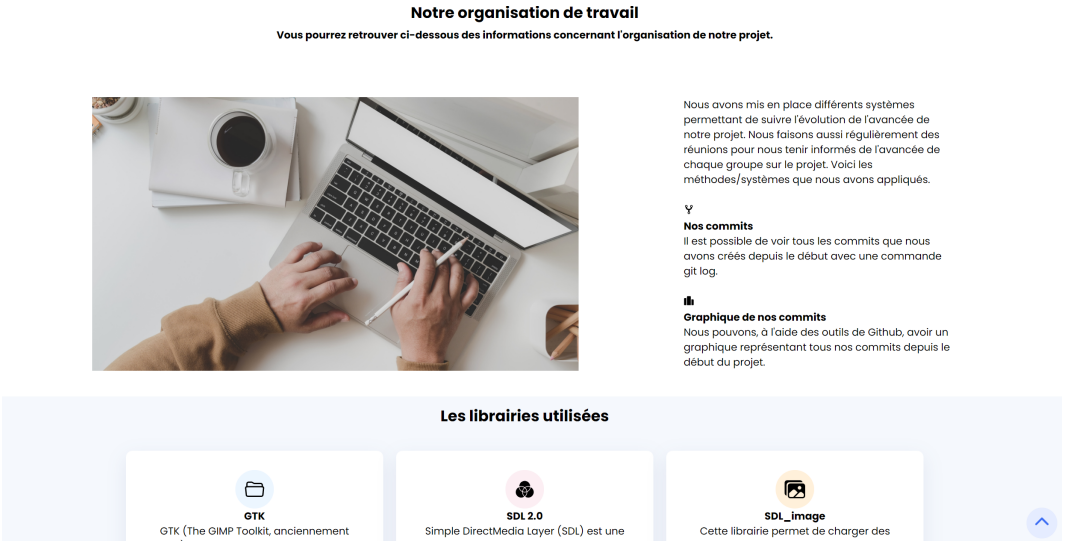


Page d'accueil

La chronologie du projet



Possibilité de télécharger



Nos façons de travailler et librairies

Le site est disponible sur ce lien : https://emiltoulouse.com/QR-code/index.html

Le but de notre site a également été de faire une navigation simple et rapide pour que l'utilisateur puisse aller directement aux informations qui l'intéresse et ainsi utiliser le projet selon ses envies.

11 Organisation du projet

11.1 Tableau de répartition des tâches

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

11.2 Planning prévisionnel

	Soutenance 1	Soutenance 2	Soutenance 3
Chargement d'image	100%	100%	100%
Prétraitement	50%	100%	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%

11.3 Ressenti personnel

11.3.1 Adam Mahraoui

Pour ma part ce projet a été vraiment intéressent à réaliser. Ma première partie sur l'encodage de l'image résultat avec la transformation de la création de la matrice et le remplissage de celle-ci a vraiment été agréable à manipuler. C'était plutôt plaisant d'avoir un retour visuel sur son travail lorsque je devais remplir la matrice avec les différents éléments qui la composent, essayer de trouver les coordonner et à chaque fois inverser les abscisses et les ordonnées... Une fois mon travail rassemblé avec Émil il ne nous restait plus qu'à régler les petit bugs présent et de comparer notre matrice en plissant les yeux à un QR-Code généré sur le internet et trouver les petites erreurs. Un des moments les plus marquants de ce projet fu lorsque nous avons scanner notre premier QR-Code et que ça a fonctionné, c'était un des évènement le plus marquant pour moi durant ce projet.

En ce qui concerne la partie sur l'interface, même si au départ c'était plutôt compliqué de comprendre comment lier mon interface faite avec Glade à des fonctionnalités de l'encodage j'ai finalement réussi à rendre pour la deuxième soutenance une interface fonctionnelle. Par la suite comme dit précédemment dans le rapport, j'ai préféré faire tout a la main pour mieux comprendre ce que je faisais et avoir un code lisible.

Un autre évènement marquant fut lorsque nous avons réussi à décoder le premier QR code avec l'interface. En effet, lorsque nous avons rassemblé la partie encodage et décodage plusieurs problèmes sont apparus et ce fut un parti plutôt frustrant car nous savions que les parties séparées fonctionnaient mais lorsque nous les regroupions ce n'était pas le cas... Alors lorsque après des multiples essayés, à modifier des petites lignes, à faire et refaire make puis ./main nous avons finalement réussi à décoder le QR avec comme donné fini c'était un vrai soulagement (qui a d'ailleurs été pris en capture d'écran.

Je suis plutôt satisfait de mon travail rendu, l'interface est ergonomique et pourrait tout à fait avec quelques petites améliorations graphiques servir dans un vrai logiciel.

Ce travail de groupe m'a permi de renforcer mon niveau en C et aussi apprendre toujours plus a travailler en équipe. Même si avec quelques problèmes d'organisation et communication interne au groupe je reste quand même satisfait de notre travail rendu.

11.3.2 Emil Toulouse

Ce projet a été vraiment intéressant mettre en place et à gérer. De l'idéation du projet à la création du repo en passant par les heures de correction de bug et de compréhension des polynômes, tout à pour été très intense et enrichissant Ma première partie sur l'encodage de la donne a été assez dure. Plusieurs étapes au début étaient simples et donnait l'impression d'avancer mais arriver à la partie polynomiale pour la correction d'erreur d'un QR code le sujet est devenu plus théorique et mathématique, et donc m'as rendu le code plus compliquer. Une fois mon travail rassemblé avec Adam il ne nous restait plus qu'à régler les petits bugs présents et de comparer notre matrice à un QR-Code généré sur le internet et trouver les petites erreurs. Il nous fallait vraiment trouver un moyen de comparaison plus simple. Et c'est la que mon travail avec SDL sur la création d'une image de la matrice du QR code nous à beaucoup aider et nous donner un second souffle au projet qui était plus ou moins à l'arrêt a cause de cette histoire de polynôme.

Un moment super important preuve tout le travail que nous avons fait à été lorsque nous avons put enfin scanner nos premiers QR code avec nos téléphones.

J'ai par la suite perfectionné l'encodage et je me suis concentré sur le site internet et sur l'aide que je pouvais apporter à droite a gauche notamment sur l'interface graphique faite par Adam. J'ai endossé le rôle de code reviewer pendant la période qui suivait la première présentation. C'est un rôle que je n'avais jamais fait mais qui est intéressant. Arriver dans le code de qqn comprendre ses raisonnements et aider dans un sujet particulier c'est assez inédit pour moi.

Pour la dernière partie je me suis concentré sur la partie du prétraitement. J'ai eu la tâche de transformer une image en une matrice représentative du QR code dans l'image afin que le code de Samy puisse l'utiliser pour sa partie décodage.

Je suis plutôt satisfait du travail rendu, l'interface est simple et pourrait tout à fait avec quelques petites améliorations graphiques servir dans un vrai logiciel. Les créations de QR code marche plutôt bien et le décodage mériterait d'être améliorer. Nous avons un rendu simple mais fonctionnel à petite échelle.

Ce travail de groupe m'a permis de solidifier mon niveau en C. Il m'a également permis de travailler et gérer une équipe. Même si quelques problèmes d'implication et de communication au sein du groupe étaient présent, je reste satisfait de notre travail rendu.

11.3.3 Samy Achache

Ce projet pour ma part est très intéressant, en effet l'avantage de ce dernier et 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 dernière soutenance, nous avons donc terminé tout le projet, mais cela a été très complique, en effet beaucoup de contrainte sont apparu lors de cette dernière soutenance, l'un des moments les plus compliqués a été lors de la liaison de toutes nos parties en rendant le projet utilisable depuis une seule et unique interface.

Cependant, malgré cela, nous avons réussi à surmonter cela grâce à notre communication et notre aide mutuelle, qui ont été déterminantes pour la finalisation de ce projet.

12 Conclusion

Ce projet nous a fait rencontre différents problèmes sur plusieurs points, et dans quasi toutes les sous-parties de la réalisation de ce projet. Mais nous avons réussi à le finalisé et à le rendre totalement fonctionnel. L'expérience a été très enrichissante, sur différents aspects, il nous apporte énormément de savoir sur la matière de la programmation puisque chacun a nettement vu son niveau et sa compréhension du code augmente mais pas seulement puisque en effet, nous avons eu un groupe soudé, qui a su s'aide dans la moindre difficulté rencontres et c'est l'une des qualités essentielles qui nous ont permis de réussir ce projet de QR Code.

Enfin nous avons certes la impression que ce projet nous a apportée beaucoup dans notre développement personnel et en groupe, puisque maintenant nous avons eu une nouvelle expérience dans le travail en groupe. Nous sommes très fiers d'avoir pu achever le projet dans son intégralité.