Protocoles de communication graphique 2D

Fatima-Ezzohra AKHTEN et Arthur DELAIN

Abstract

Réseaux, L3 Informatique UCA Clermont-Ferrand,2019

Introduction

Vous avez sûrement déjà remarqué des pictogrammes fait de petits carrés noirs et blancs que ce soit sur des affichages publicitaires, sur un e-billet, sur l'emballage des produits de consommation, ou même dans un magazine, et vous vous êtes demandé de quoi s'agit-il? En effet, c'est un code 2D appelé plus généralement Tags, QR Codes ou Flashcodes; l'héritier du code barre à une dimension mais ayant une technologie plus avancée.

Les codes 2D permettent de passer d'un support physique (papier, écran...) au format électronique en une fraction de seconde, dans le but de traiter ou capturer des données (URL, mail, carte de visite...).

Ils se caractérisent par le fait qu'ils contiennent des informations à la fois verticalement et horizontalement. Par conséquent les codes barres 2D contiennent beaucoup plus d'informations (stockées dans une matrice 2D) qu'un code barre classique à une dimension.

Le format des codes à barres 2D est généralement issu de deux technologies d'encodage : datamatrix et QR Code. Ces technologies d'encodage permettent de stocker une grande quantité d'information sur un espace réduit en comparaison des codes à barres à une seule dimension.

Pour notre projet, on s'est basé sur la deuxième technologie d'encodage qu'est le QR Code, dont le format est direct, qui permet de contenir directement un grand nombre de données dans le code, et qui peut être décodé rapidement.

Sujet

RESEAUX 2
TP PROTOCOLE GRAPHIQUE

L'objectif de ce TP est de concevoir un protocole de communication « graphique ». Un exemple de protocole de communication graphique est le QR-CODE (voir en haut à droite de cette feuille pour un exemple de message graphique).

Vous devrez concevoir un nouveau protocole de A à Z. Vous pouvez vous inspirer du QR-CODE mais devrez proposer vos propres solutions aux différents problèmes posés.

Les informations seront « dessinées » dans une matrice dont vous proposerez vous même les dimensions (plusieurs tailles « fixes » ou formes peuvent être proposées en fonction de la taille du message). Chaque « cellule » de votre matrice constitue l'unité d'information de base (par exemple, pour le QR-CODE, cette unité d'information est le bit).
Votre matrice devra contenir trois types d'informations :

- Le message à transmettre en lui même
- Les informations permettant de détecter et éventuellement corriger des erreurs
- Les informations spécifiques à votre protocole

N'oubliez pas que votre message graphique sera imprimé et que lors de l'impression la taille des cellules (en pixel) peut être modifiée et les couleurs altérées (si vous utilisez des couleurs différentes). De même, l'orientation de votre dessin peut être modifiée.

Exemple d'informations spécifiques à votre protocole (non exhaustif) :

- Information permettant de repérer la matrice au milieu d'une page pouvant contenir d'autres informations
- Information permettant de calculer la taille d'une cellule en pixel
- Information permettant de connaître la position du premier bit du message
- Information permettant de connaître la taille du message
- Information permettant de connaître combien de bits sont codés dans chaque cellule.
- Si vous utilisez des couleurs, informations permettant de retrouver chacune des couleurs utilisées et les bits qu'elles codent
- …

Travail à rendre

1) Rapport détaillé présentant votre protocole graphique et justifiant vos choix. Vous devrez en particulier expliquer le rôle de chacune des cellules de votre matrice ainsi que la taille maximum du message pouvant être transmis à l'aide de votre protocole.
2) Un programme permettant de saisir un message texte quelconque et qui générera la matrice correspondante respectant votre protocole (vous pouvez utiliser le langage de votre choix, en justifiant éventuellement ce choix).
3) Optionnel : proposer une méthode permettant d'ajouter un logo dans votre matrice sans que cela n'altère la fonction de décodage et sans perdre de cellules « codantes ».

Part I
Fonctionnement

• 1 - Tout d'abord, il faut analyser les données à encoder et paramétrer le niveau de code correcteur (expliqué plus loin). Cela permet d'analyser les entrées afin d'obtenir la variété des caractère à encoder.

• 2 - Il faut ensuite convertir les données dans un flux de bytes.

Les premier bits d'information represente souvent le mode (alpha numeric,...). Il faut donc encoder les données et les concaténer entre elles avec le code correcteur Par exemple le mode QrCode est sur 4bits :

Indicateur binaire	Type charactère
0001	Numérique
0010	Alphanumérique
0100	Binaire
1000	Kanji (caractère Japonais)

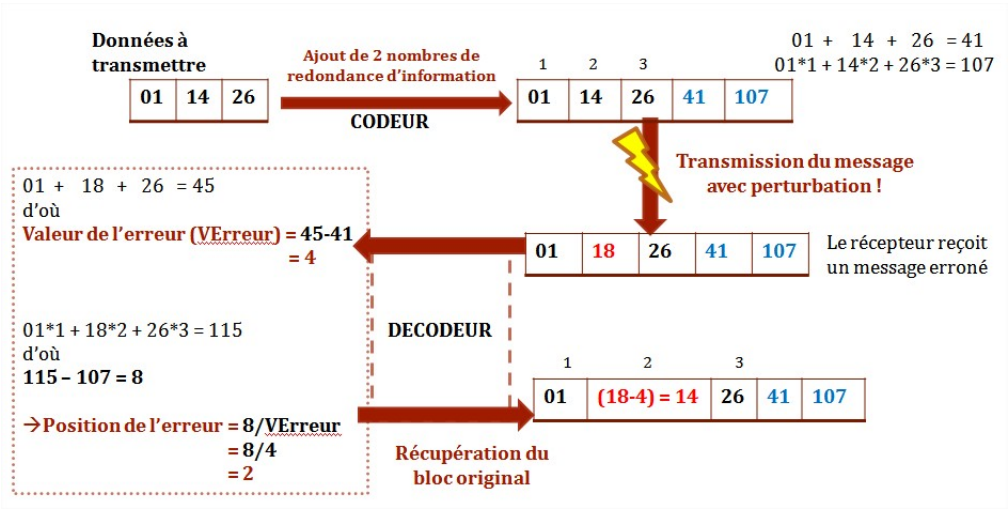
(non exhaustif)
On peut ensuite mettre en place après l'avoir choisi le niveau de correction du code de correction d'erreur afin de créer une structure de messages finale (block de bits)

Et ensuite on peut creer les différents ajout complémentaire:
Il faut indiquer la version (si il y en a plusieurs taille de Code 2D), le format , taille du message , ...

• 3 - On implémente de la verification et correction des erreurs. On sépare le message en bits en blocs de bits de données et on genère le codes correcteurs. Le code correcteur le plus répandu est Redd-Salomon, il peut avoir différents niveau :

Niveau	Redondance	Indicateur binaire
L	environ 7 %	01
M	environ 15 %	00
Q	environ 25 %	11
H	environ 30 %	10

L'illustration la plus courante de ça mise en place sur internet est celle ci :



(Il y a aussi le Code de Hamming et d'autres moins connus) Si il n'y a pas de niveau de code correcteur spécifié, pour le Qr Code par exemple la plus petite version de celui ci est utilisée.
Il y a bien sur différents codes correcteurs et manière de vérifier les informations. Par exemple on peut générer un hashcode du message et l'intégrer dans celui pour que lors du scan du Code 2D l'on puisse savoir directement si le message n'a pas été corrompu.

• 4 - On insère ensuite les données avec le code correcteur une matrice.
On peut ensuite placer les différents éléments:

→ les motifs de positionnement qui sont des séparateurs afin distinguer les differents éléments ainsi que de reperer le Code 2D dans un document.

→ les motifs de synchonisation/Timming patterns permettant de percevoir les contrastes entre les modules (clairs et foncés) et il permettent de determiner la version du Qr code (avec ça taille.)

→ les motifs de d'alignement qui sont pareil mais plus petit que les motifs de positionnement, il servent dans le cas ou il ya des versions très grande du Code 2D, cela facilite la lecture en cas de deformation de la matrice.

→ zone tranquille autour du symbol/matrice. Il s-agit de zone vide afin de bien d'elimiter les differents elements.

→Placement le message en bits (avec ses informations complémentaires).

(voir schema Qr Code pour exemple)

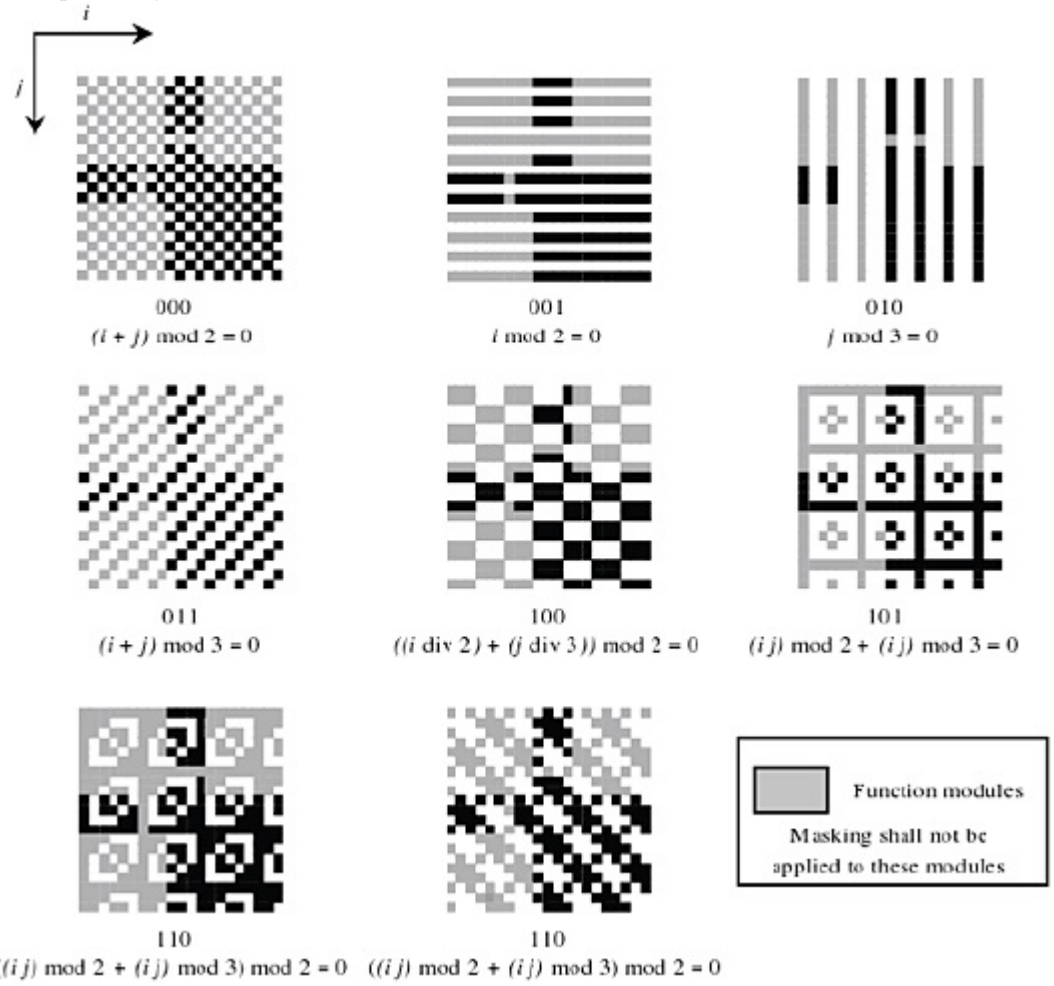
• 5 - On génère la matrice et l'évaluation du résultat retourné.
Afin d'avoir une bonne repartition des élements dans le code 2d on utilise des masques de patterns. Cela permet d'optimiser la balance entre les modules noirs et les modules blancs (cela améliore donc la répartition) et aussi de minimiser

les occurrences de patterns indésirables

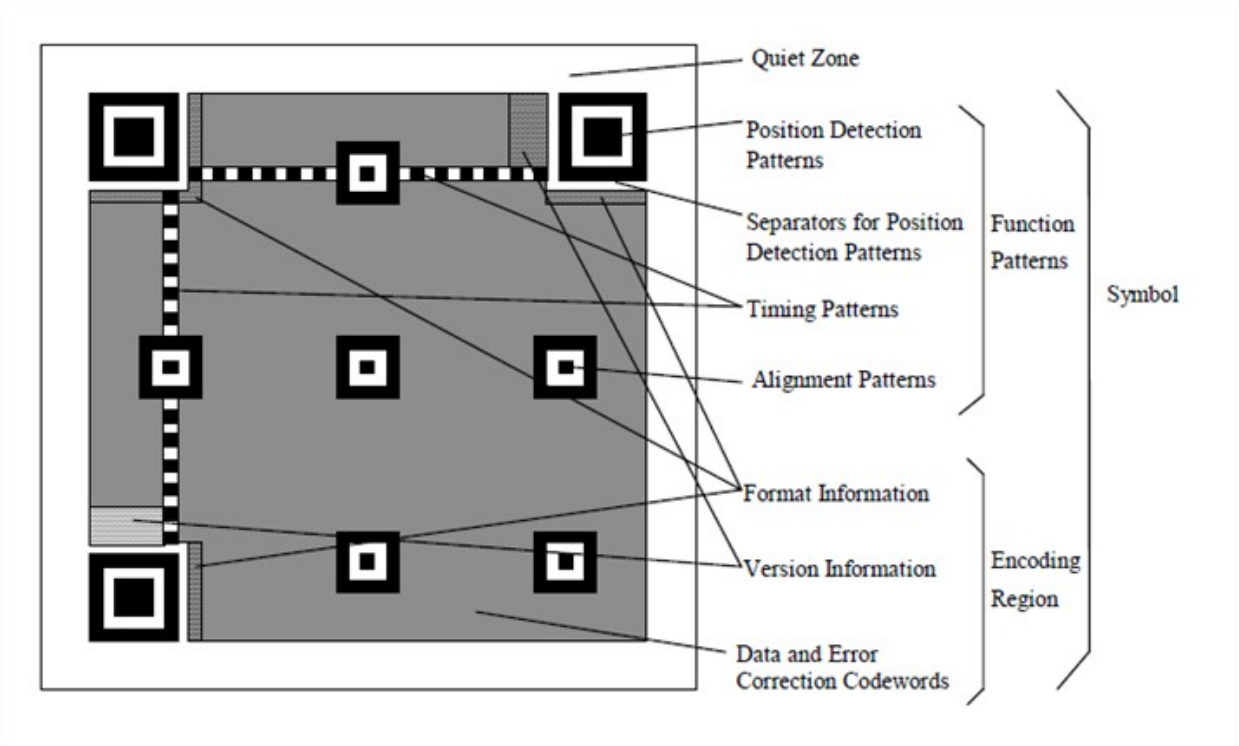
On creer donc différents masques à appliquer et on applique celui/ceux juger le meilleurs (note donné à chaque masque)

Application du masque

Exemple du QR code :



On peut ensuite intégrer les differentes information de format et de version à des endroits spécifique. Exemple du QR code:



• 6 - Génération du Code 2D au format image.

(lecture
1 - Reconnaître les bits 1 ou 0.
2 - Identifier le taux de code correcteur (avec masque -¿ format).
3 - Identifier la version/format du Code.
4 - Découvrir région à décoder.
5 - Lire les données et le code correcteur.
6 - Détecter/Corriger les erreurs.
7 - Décoder données.
8 - résultat.
)

On peut prendre comme exemple un logiciel open source de générateur de Qr Code ZXing ,ecrit en java.

Part II
Mise en place

Le langage uilisé dans notre projet est le java, avec swing pour générer une interface graphique. Le logiciel est développé sous l'IDE Intel ij.

Le nom de notre Code 2D est Code Fort par son allure en carré "concentrique" rappelant une muraille avec un donjon comme zone d'inclusion

0.1 Encodage et versionnage:

Versionnage : Apres avoir définit un versionnnage se voulant proportionnel à la taille du message à encoder nous avons décider pour des raisons de simplicité de le laisser fixe à 32x32 cases pour le message en bits en lui même. (Il n'est donc pas necessaire de l'indiquer, mais celui ci pourra etre ajouter au information de format (8 bit pr le format et 8 bit pour la version et 8 bits pour la taille))

Ecodage: Nous avons choisi comme encodage le bit avec un pixel blanc pour le 0 et un noir pour le 1 (traduit en tableau de Boolean dans le code). Les élements sont défini selon leur Code ASCII. Il seront dont encodé sur 8 bits (lettres, nombres et charactères speciaux) Notre Version seul peut donc encoder un maximum de 128 caractères. (Cela sans indication d'encodage pouvant par la suite être integré)

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01		SOH
^B	2	02		STX
^C	3	03		ETX
^D	4	04		EOT
^E	5	05		ENQ
^F	6	06		ACK
^G	7	07		BEL
^H	8	08		BS
^I	9	09		HT
^J	10	0A		LF
^K	11	0B		VT
^L	12	0C		FF
^M	13	0D		CR
^N	14	0E		SO
^O	15	0F		SI
^P	16	10		DLE
^Q	17	11		DC1
^R	18	12		DC2
^S	19	13		DC3
^T	20	14		DC4
^U	21	15		NAK
^V	22	16		SYN
^W	23	17		ETB
^X	24	18		CAN
^Y	25	19		EM
^Z	26	1A		SUB
^[	27	1B		ESC
^\	28	1C		FS
^]	29	1D		GS
^^	30	1E	▲	RS
^-	31	1F	▼	US

Dec	Hex	Char
32	20	
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

0.2 Code correcteur:

Apres avoir commencer de créer un systeme de correction d'erreur avec Redd Salomon, la tache fut simplifié en intégrant seulement un hashcode des donnéees générées cela afin lors de la lecture de verifier l'intégrité du message. (le hashcode nouvelement créé avec le message scanné est comparé à l'ancien hashcode indiqué en début de message et dont la taille est pour notre projet de 32bits (norme java))

0.3 Insertion (les differents motifs):

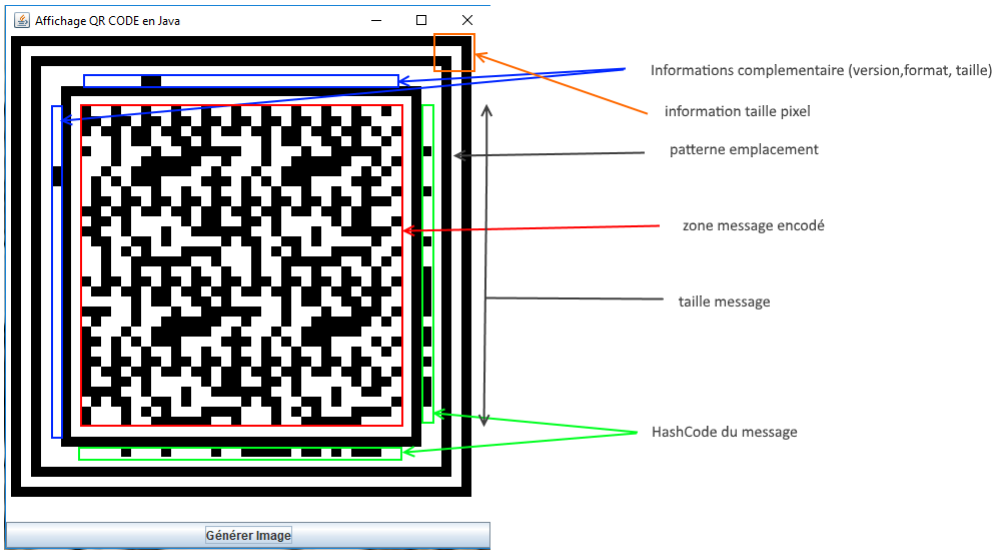
Un pemier contour au message encodé est créé afin de faire une zone tranquille/tampon autour de celui-ci. Et afin de séparer le message des autres informations indiqué (version dans le cas future ou l'on en rajouterai, format,...). Ce dernier sera lui même entouré d'un contour (double pixel : blanc puis noir) servant de pattern de positionnement.
C'est l'alterance entre ces différents contour permet de calculer la taille d'un pixel

0.4 generation matrice (masque,format,version) :

On peut ensuite rajouter les différentes informations complémentaires dans la zone entre le contour exterieur et le contour du message.
On applique ensuite un masque (choisi par default ici, il aurait fallu caluler celui ayant la meilleur répartion vis à vis des autres).

0.5 Generation Code 2D:

Après avoir optenu une matrix de 0 et de 1 (utilisation du Boolean[][]) on peut directement générer le Code 2D Graphique correspondant en plaçant des pixels noirs aux emplacements des 1 (comme dit precedemment).



Part III
Utilisation

Lancer le .jar (avec java) placé dans la racine du projet:
Etapes:

1) rentrer le texte dans la fentre pop-up
2) generer l'image correspondante (bouton correspondant)