Informatique L3 - S6

Réseaux II

Rapport

Mehdi Znata
Diab mohammad ibrahim
Dinedane Abdelkader

No. Groupe 2 et 3

Charges de TP :

Mr. Youssef BOUAZIZ
Mr. Minh Hieu NGUYEN

Lien GitHub Du Projet :

https://github.com/Thorfin2/ReseauxII_L3

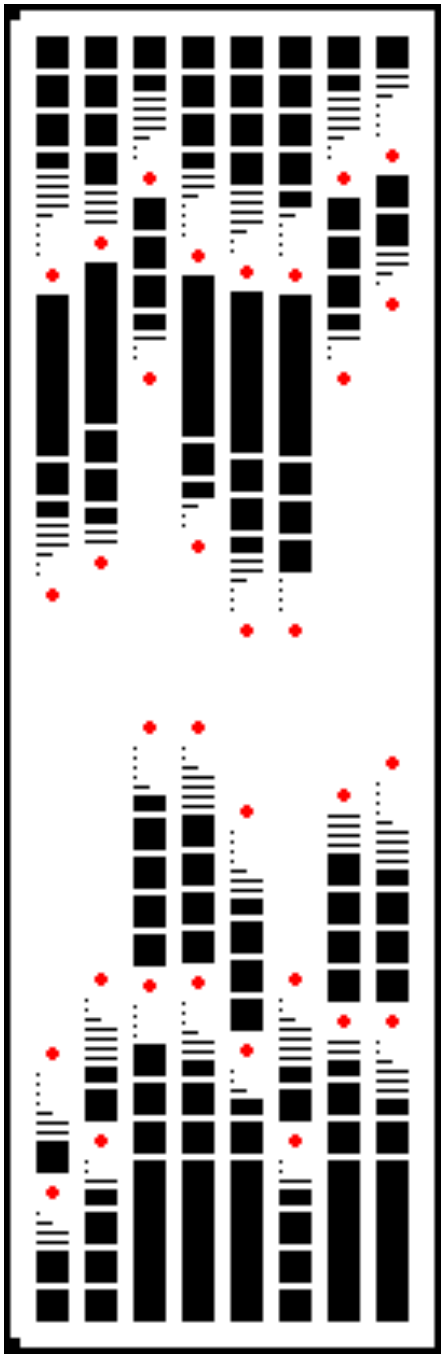
Table des matières

Message : Bonne Lecture !

```
/usr/local/bin/python3.12 /Users/mehdiznata/Desktop/TP_reseauxII/main.py
Please enter a message to encode: Bonne Lecture !
The encoded message is:
 [['%%%$####@....' '^%$###@..']
 ['%%%%####' '^%%$##']
 ['%$####@..' '%%%$#..']
 ['%%%$##@....' '^%$@..']
 ['%%%%####@..' '^%%$##@...']
 ['%%%%$@....' '^%%%....']
 ['%$####@..' '%%%$#..']
 ['%##@....' '%%##@.']]

The decoded message is:
 Bonne Lecture !


Process finished with exit code 0
```



I. Introduction :

La méthode graphique dans ce code implique l'utilisation de symboles et de calculs matriciels pour encoder un message. Chaque symbole a une représentation graphique qui est utilisée pour la visualisation du message.

II. Documentation du code :

Première partie du code :

La classe SymbolEncoder est utilisée pour effectuer le calcul matriciel qui encode le message, tandis que le dictionnaire symbol_values attribue une valeur numérique à chaque symbole.

La méthode encode_num reçoit une valeur numérique en paramètre et renvoie une chaîne de symboles qui représente cette valeur.
La méthode decode_str reçoit une chaîne de symboles en paramètre et renvoie la valeur numérique qu'elle représente.

La méthode encode_with_matrix_A est responsable de l'encodage du message. Elle prend une chaîne à encoder, la décompose en paires de caractères, et calcule la matrice produit matrix_A avec chaque paire de caractères encodée en ASCII en tant que vecteur colonne.

Chaque vecteur résultant est converti en une liste de nombres encodés en utilisant la fonction encode_num. Les listes de nombres encodés sont ensuite concaténées dans une matrice encodée qui commence et se termine par des nombres qui représentent la longueur de la chaîne originale et l'entier rep_matrix_A qui sera utilisé pour déduire les valeurs de la matrice matrix_A afin que la partie réceptrice puisse décoder le message extrait.

La méthode decode_with_matrix_A fait l'inverse de la méthode encode_with_matrix_A : elle prend une matrice encodée, la décode en utilisant la méthode decode_str pour chaque élément de la matrice, puis multiplie chaque vecteur décodé par l'inverse de la matrice matrix_A pour obtenir la liste originale de paires de caractères. La chaîne originale est ensuite reconstituée en utilisant les codes ASCII correspondant à chaque paire de caractères.

2ème partie du code :

Dans cette partie, deux classes sont définies : Shape et MessageDrawer. La classe Shape définit une forme rectangulaire avec les attributs x et y représentant les coordonnées de son coin supérieur gauche, la largeur et la hauteur du rectangle, et la couleur du rectangle.

La classe MessageDrawer, dont le constructeur reçoit la matrice encodée encoded_matrix, et initialise les valeurs de x, y, mH (hauteur maximale), shapes_top et shapes_bottom. Cette classe contient également les méthodes utilisées pour donner la représentation graphique du message représenté par la matrice encodée.

La méthode draw_encoded_matrix remplit la liste des symboles de la partie supérieure du "dessin". Elle remplit également la liste des symboles de la partie inférieure du "dessin".
Les dimensions des symboles (formes) sont calculées comme suit : La valeur numérique sym_value du symbole est apportée du dictionnaire symbol_values. Si cette valeur est <= 10 ==> largeur du rectangle = sym_value, hauteur du rectangle = 1. Si cette valeur est > 10 ==> largeur du rectangle = 10, hauteur du rectangle = sym_value/10.

Ces méthodes sont appelées par la méthode draw_encoded_matrix qui dessine les rectangles représentant le message sur une image en utilisant la bibliothèque PIL (Python Imaging Library) qui est enregistrée sous le nom encoded_message.png.

Python a été utilisé pour ce projet pour plusieurs raisons. Tout d'abord, Python est un langage de programmation très populaire et facile à apprendre, ce qui m'a permis de commencer rapidement à travailler sur mon projet. De plus, Python dispose d'une grande communauté de développeurs et de nombreuses bibliothèques open source, telles que NumPy ou Pillow, mettre en œuvre certaines fonctionnalités de mon projet plus facilement, comme la création des graphiques et la génération des matrices

III. Documentation du protocole :

Chaque message est encadré par une bordure de 2 pixels, qui est le double de la plus petite valeur qu'un symbole peut avoir (qui est 1 pixel).

Un message se compose de 2 parties : une partie supérieure, qui contient le message dans son "sens correct" (sens 1), et une partie inférieure qui contient le "message inversé" (sens 2).
Les petits carrés (de 3 pixels) aux 2 coins gauche de l'image aident à trouver son orientation correcte et indiquent le début du message.
-Dans le "sens 1", le message doit être lu de gauche à droite. Chaque colonne doit être lue de haut en bas.
-Dans le "sens 2", le message doit être lu de droite à gauche. Chaque colonne doit être lue de haut en bas.

La première et la dernière colonne de chaque partie (inférieure/supérieure) contiennent les informations suivantes :

Valeurs qui composent la matrice utilisée pour encoder le message
Taille du message

Une colonne est composée de deux lettres séparées par 4 pixels blancs, un "cercle" rouge de rayon 2 pixels (==> diamètre de 4 pixels), et 4 pixels blancs ==> 12 pixels au total.

Chaque lettre est composée de plusieurs rectangles noirs de différentes dimensions (en pixels), séparés par 2 pixels.

Les cercles servent à délimiter les lettres, pour savoir qu'on a atteint la fin d'une lettre. Cela aideraà compter le nombre total de lettres décodées, afin que ce nombre puisse être comparé au nombre total des lettres* du message (*qui est mentionné dans la première et la dernière colonne de chaque "sens"). Et cette comparaison déterminera s'il y avait des lettres manquantesdans le message décodé.

Si certaines lettres manquent, elles peuvent être décodées à partir de la partie inférieure dumessage.
De cette manière, la redondance du message est utilisée comme approche de correction en casd'erreur.
Cela permet également d'insérer des logos dans la partie médiane de l'image encodée. Pour cetteraison, la partie supérieure et la partie inférieure sont inversées.
Ceci sera expliqué dans l'exemple à la fin de cette documentation.

IV. Avantages et inconvénients :

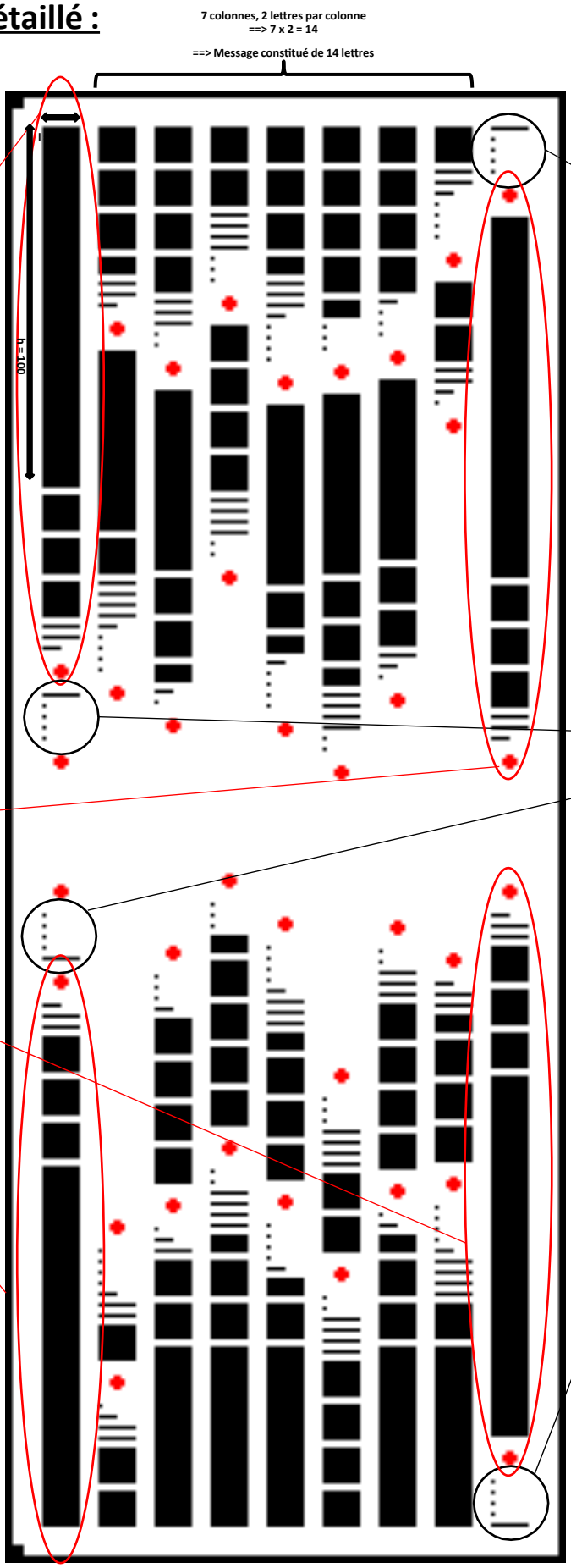
Avantages :

- **Ce protocole n'impose pas de longueur maximale pour un message, mais bien sûr, si le message est trop grand, l'image résultante sera également très grande, ce qui la rendra difficile à voir et à interpréter (ce qui peut être considérer comme un inconvénient).**
- **L'encodage du message garantit la sécurité de la transmission du message.**
- **La redondance permet à ce protocole graphique de tolérer un grand taux d'erreur, surtoutgrâce à la représentation "inverse" du message.**
- **La possibilité d'ajouter des logos au message encodé.**

Inconvénients :

- **L'approche de correction d'erreur, même si elle est efficace, augmente la taille de la représentation de l'image. On peut se contenter de la partie supérieure du code, mais au prix de ne tolérer aucune erreur. J'ai utilisé la redondance car je n'ai pas trouvé d'autre solution car mon encodage n'est pas basé sur des bits, ce qui a rendu la tâche un peu difficile.**
- **On pourrait dire que l'utilisation de pixels comme unités peut causer des problèmes car ils sont petits. Mais l'utilisation d'espaces suffisants entre chaque symbole rend clair et facile pour le destinataire (ordinateur par exemple) de pouvoir détecter même de petits pixels.**

V. Exemple détaillé :



Dimensions sur l'image originale :

- Bordure : 2px
- Carrés (coins) : 3px

- l = 10, h = 100
  ==> h > 1
  donc la valeur est > 10
  ==> h = valeur/10
  ==> 100 = valeur/10
  ==> valeur = 1000
  ==> &
- l = 10, h = 10
  ==> h > 1
  donc la valeur est > 10
  ==> h = valeur/10
  ==> 10 = valeur/10
  ==> valeur = 100
  ==> %
  - (De même pour les 2 rectangles qui suivent)
  - l = 10, h = 1
    ==> valeur = 10
    ==> #
  - l = 10, h = 1
    ==> valeur = 10
    ==> #
  - l = 5, h = 1
    ==> valeur = 5
    ==> @

==> Symboles : **&%%%##@**

==> Valeur totale = 1000 + 100 + 100 + 100 + 10 + 10 + 5
==> = **1325**

==> Matrice utilisé pour encoder :
A = [[1 3]

Dimensions sur l'image originale :

- l = 10, h = 1
  ==> valeur = 10
  ==> #
- l = 1, h = 1
  ==> valeur = 1
  ==> .
- l = 1, h = 1
  ==> valeur = 1
  ==> .
- l = 1, h = 1
  ==> valeur = 1
  ==> .
- l = 1, h = 1
  ==> valeur = 1
  ==> .

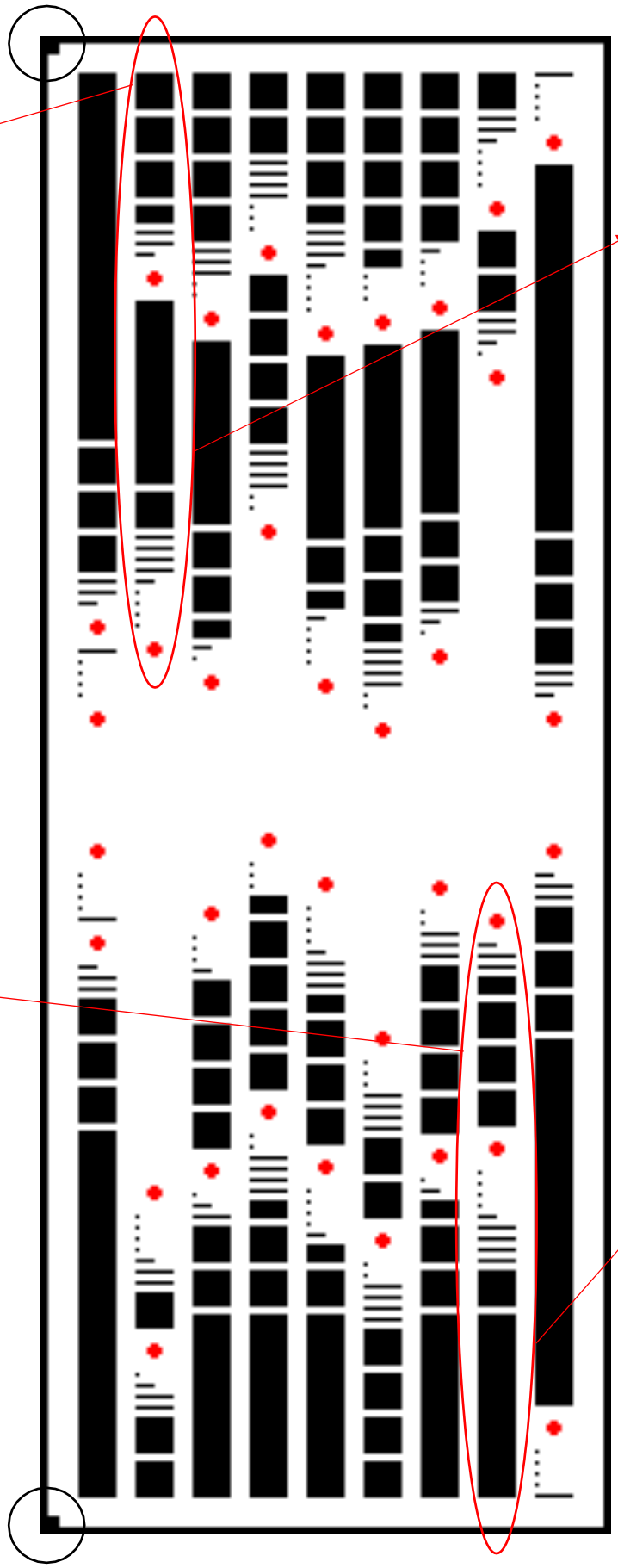
==> Symboles : **#....**

==> Valeur totale = 10 + 1 + 1 + 1 + 1 = **14**

==> Taille du message = 14 :

Hello : 5
(virgule) : 1
(espace) : 1
world! : 6
==> Total = 13
==> Impair
==> Un (espace) sera ajouté à la fin du message
==> Total = 14

Début du message



Dimensions sur l'image originale : 1ère lettre :

- l = 10, h = 10
  ==> h > 1
  donc la valeur est > 10
  ==> h = valeur/10
  ==> 10 = valeur/10
  ==> valeur = 100
  ==> %
  - (De même pour les 2 rectangles qui suivent)
  - l = 10, h = 5
    ==> h > 1
    donc la valeur est > 10
    ==> h = valeur/10
    ==> 5 = valeur/10
    ==> valeur = 50
    ==> $
  - l = 10, h = 1
    ==> valeur = 10
    ==> #
  - l = 10, h = 1
    ==> valeur = 10
    ==> #
  - l = 5, h = 1
    ==> valeur = 5
    ==> @

==> Symboles qui constituent la 1ère lettre : **%%%$##@**

Dimensions sur l'image originale : 2ème lettre :

- l = 10, h = 50
  ==> h > 1
  donc la valeur est > 10
  ==> h = valeur/10
  ==> 50 = valeur/10
  ==> valeur = 500
  ==> ^
- l = 10, h = 10
  ==> h > 1
  donc la valeur est > 10
  ==> h = valeur/10
  ==> 10 = valeur/10
  ==> valeur = 100
  ==> %
  - l = 10, h = 1
    ==> valeur = 10
    ==> #
  - (De même pour les 3 rectangles qui suivent)
  - l = 5, h = 1
    ==> valeur = 5
    ==> @
  - l = 1, h = 1
    ==> valeur = 1
    ==> .
  - (De même pour les 3 rectangles qui suivent)

==> Symboles qui constituent la 2ème lettre : **^%####@**