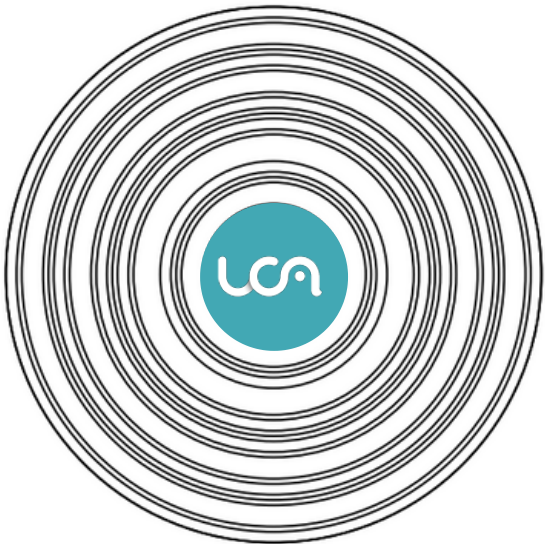
UNIVERSITÉ Clermont Auvergne

TP RÉSEAUX

Création d'un nouveau protocole graphique incluant de l'information - Modèle en Cercle

Par :

- Daoud Mohammed Ryad
- Lahmouri Ikram



2023

TABLE DES MATIÈRES

INTRODUCTION

Le présent rapport a pour objectif de décrire le processus de création d'un protocole graphique pour l'affichage d'informations. Dans le cadre de ce travail pratique, nous avons choisi d'utiliser l'affichage en cercles comme modèle . Cette forme simple mais efficace nous a permis de mettre en place un protocole graphique facilement compréhensible et intuitif pour l'utilisateur.

Nous avons commencé par étudier les différentes méthodes existantes pour l'affichage d'informations, afin de mieux comprendre les enjeux et les défis liés à cette tâche. Nous avons ensuite développé notre propre protocole graphique en nous inspirant de ces méthodes, en intégrant les éléments clés de l'affichage graphique, tels que la taille, la couleur et la position.

Le protocole graphique que nous avons créé a été testé et validé auprès d'un panel d'utilisateurs, afin de nous assurer de son efficacité et de sa compréhensibilité. Les résultats de ces tests ont été analysés et interprétés, et les conclusions de cette étude sont présentées dans ce rapport.

lien github de projet

https://github.com/ryaddaoud21/Protocol_Graphique.git

CHOIX DE MODÈLE

AVANTAGES :

Le choix du modèle en cercle pour l'affichage des informations dans notre protocole graphique peut être justifié par plusieurs raisons. Tout d'abord, les cercles sont des formes géométriques simples, facilement reconnaissables et compréhensibles par un grand nombre de personnes. De plus, les cercles ont des propriétés intéressantes, notamment une symétrie parfaite et une équidistance entre tous les points de leur contour.

Cela permet de représenter de manière équitable des données qui ont une importance similaire dans notre protocole graphique. En outre, l'utilisation de cercles permet une organisation claire et facile à suivre des informations, en plaçant les cercles de manière circulaire et en les reliant par des lignes. Enfin, la nature circulaire du modèle peut représenter de manière intuitive des notions de continuité et de cycle, qui peuvent être pertinentes pour certains types de données à afficher.

INCONVÉNIENTS :

Cependant, il y a également des inconvénients à considérer. Tout d'abord, l'utilisation de cercles peut limiter la capacité à représenter des formes plus complexes ou des informations qui nécessitent des représentations spatiales précises.

En outre, l'utilisation de cercles de différentes tailles peut être trompeuse, car la perception de la taille est relative et peut être affectée par d'autres facteurs, tels que l'espace disponible pour l'affichage ou la résolution de l'écran. Enfin, les cercles ne permettent pas de représenter les relations entre les données, contrairement à d'autres formes graphiques, telles que les diagrammes en arbre ou les graphiques en toile d'araignée.

En résumé, le choix d'un modèle en cercle pour l'affichage dépend du contexte et des objectifs de l'analyse. Si la simplicité et la clarté de la représentation sont privilégiées, et que les relations entre les données ne sont pas un facteur important, alors les cercles peuvent être un choix approprié. Cependant, si la complexité des données nécessite une représentation spatiale précise ou si les relations entre les données sont importantes, il peut être préférable de considérer d'autres formes graphiques.

EXÉCUTION

OUTILES :

Dans le cadre de notre travail, nous avons utilisé Python version 3.9 comme langage de programmation pour la réalisation de notre protocole graphique. Nous avons également utilisé Pycharm, un environnement de développement intégré (IDE) très pratique pour la création de notre code.

En ce qui concerne l'affichage de notre modèle graphique basé sur des cercles, nous avons opté pour la bibliothèque Matplotlib. Cette bibliothèque est très utile pour la création de graphiques en Python, elle offre une grande variété de styles et de formats pour l'affichage graphique. Nous avons particulièrement apprécié la simplicité d'utilisation de cette bibliothèque et son rendu graphique très clair et esthétique.

Pour exécuter le fichier Python sous Windows :

- Ouvrez l'invite de commande de Windows
- Accédez au répertoire où se trouve le fichier Python à exécuter .
- Une fois que vous êtes dans le répertoire où se trouve le fichier Python, vous pouvez l'exécuter en saisissant la commande "python Protocol.py" dans l'invite de commande.
- Appuyez sur Entrée pour exécuter le fichier Python. Si tout se passe bien, vous devriez voir les résultats de l'exécution du programme s'afficher dans l'invite de commande.

EXÉCUTION

TRAITEMENT DE DONNÉES :

la première étape du traitement de données consiste à saisir une chaîne de caractères via le code python. Cette chaîne de caractères est ensuite modifiée en utilisant la logique affine pour obtenir chaque bit et le mettre sous sa forme entière dans un tableau. Ce tableau est alors composé d'entiers représentant les bits de l'information.

Le deuxième traitement consiste à ajouter un protocole de contrôle et de correction d'erreur. Nous avons choisi le code de Hamming pour cette opération pour deux raisons. Tout d'abord, nous ne transmettons pas beaucoup d'informations pour le moment, et le code de Hamming semble être plus efficace car notre protocole est plus facile à lire lorsque peu d'informations sont transmises. Deuxièmement, le code de Hamming offre un bon compromis entre la détection d'erreur et la correction d'erreur.

INTERFACE :

Les deux premiers bits sont réservés au protocole et non à l'information. Ils sont utilisés pour indiquer la version et le début de lecture du protocole.

Pour chaque bit d'information et de correction, le protocole trace un cercle si la valeur du bit est égale à 1, en utilisant un rayon fixé à une valeur donnée appelée "z", qui représente l'écart entre chaque bit. Si la valeur du bit est égale à 0, le protocole n'affiche rien mais augmente le rayon de la valeur de "z". Cette opération est répétée pour chaque bit jusqu'à la fin de l'information.

Lecture d'information :

La lecture commence en repérant le cercle le plus au centre et en lisant les deux premiers bits qui donnent la version et l'écart de lecture entre les cercles (représenté par la valeur z). Ensuite, à chaque étape, on saute de z et si un cercle est détecté, on ajoute 1 à notre tableau d'informations, sinon on ajoute 0. Les positions de puissances de 2 contiennent des informations de correction d'erreur, tandis que les autres bits contiennent les données de l'information. Si trop d'erreurs sont détectées, le protocole peut être relu.

Information Supplémentaire :

Le protocole inclut des informations supplémentaires telles que les bits de parité du code correcteur, ainsi que deux bits utilisés pour indiquer la version et la taille de l'écart à lire.

Robustesse :

D'une certain manière ce code est assez robuste car même si on perd une très grande partie de l'image générée par le protocole, avec du café par exemple où elle se déchire nous pourrons quand même lire le protocole même si il ne reste qu'un dixième de l'image.

Contrôle d'erreur :

La vérification d'erreur se fait après avoir récupéré les bits de parité positionnés aux puissances de deux. Ces bits de parité permettent de vérifier si les bits d'information sont corrects ou s'ils ont été altérés lors de la transmission. Si des erreurs sont détectées, le protocole de Hamming permet de les corriger automatiquement en utilisant les bits de parité. Cela permet d'assurer l'intégrité de l'information transmise et d'éviter les erreurs de lecture.

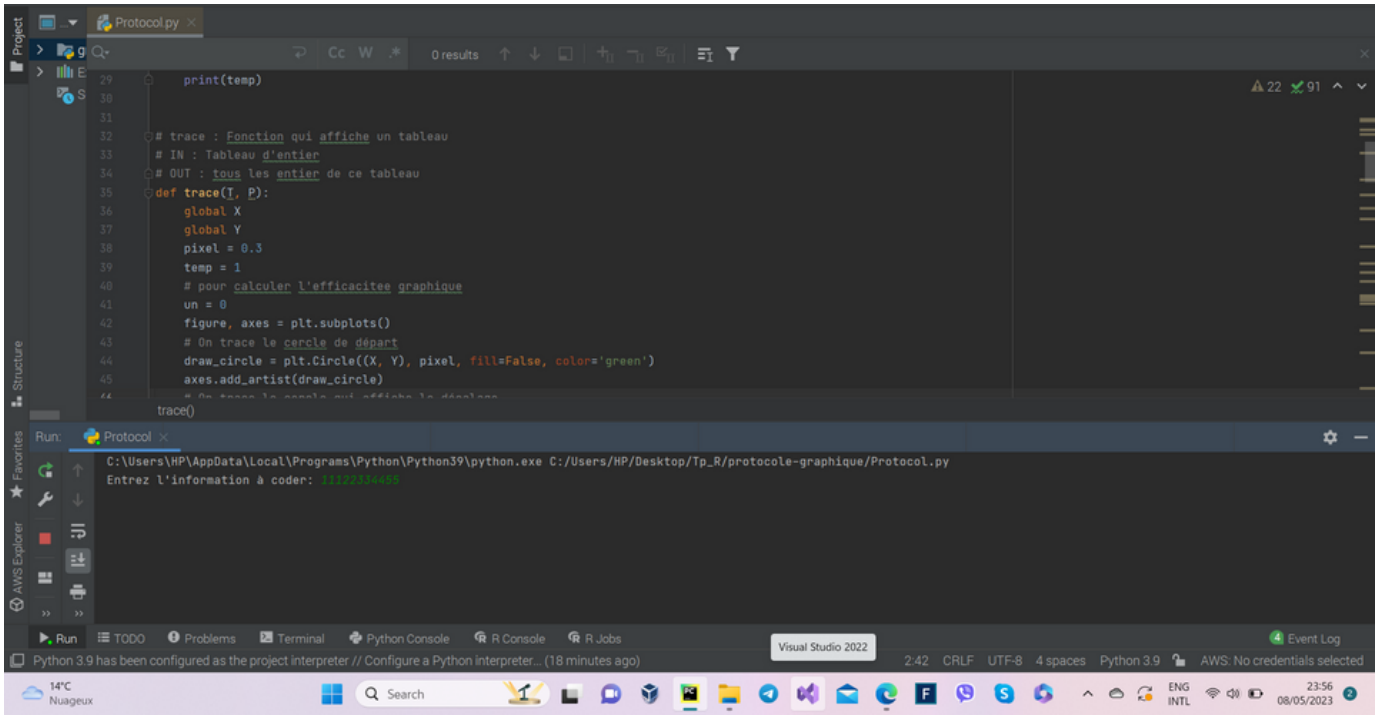
Esthétique

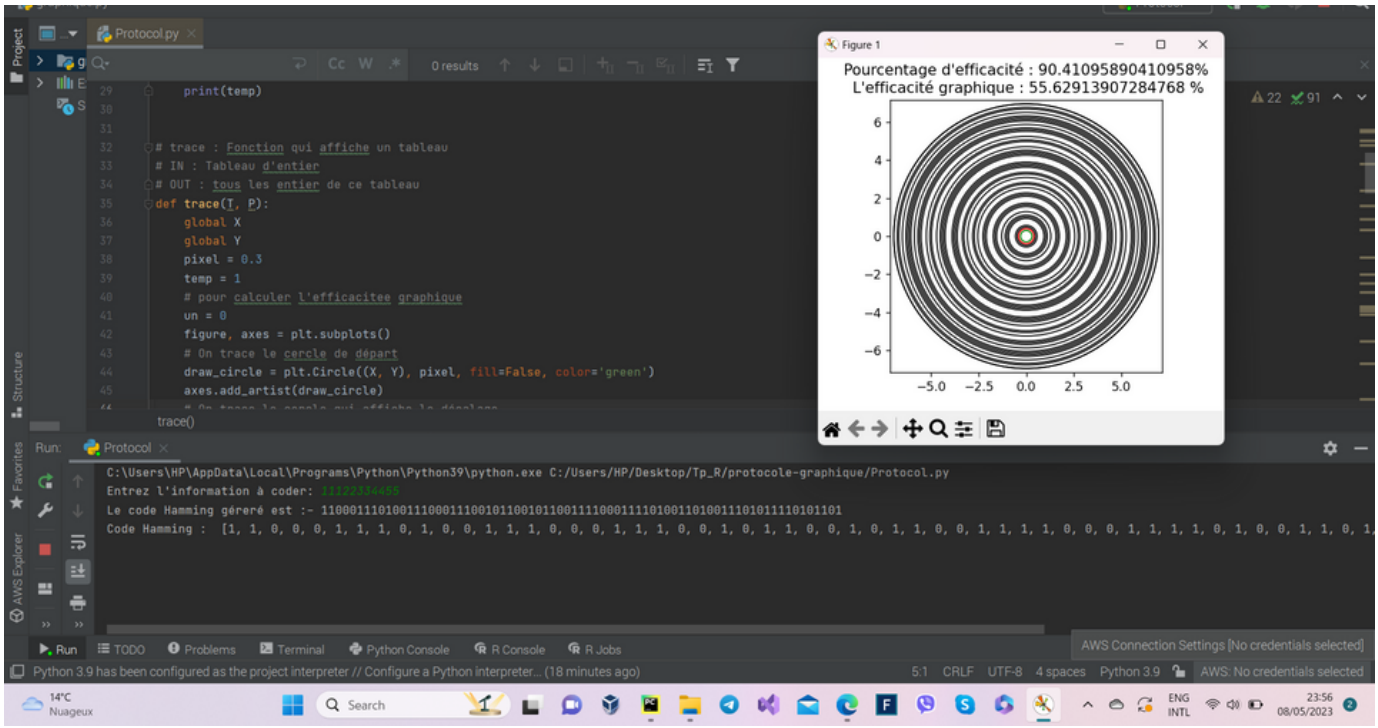
Lorsque le nombre de bits à traiter est faible, le protocole présente une certaine esthétique. Cependant, lorsqu'il y a un grand nombre de bits, le protocole devient difficilement lisible, prenant la forme d'un gros cercle noir. Dans ce cas, il est nécessaire de zoomer pour en faciliter la lecture.

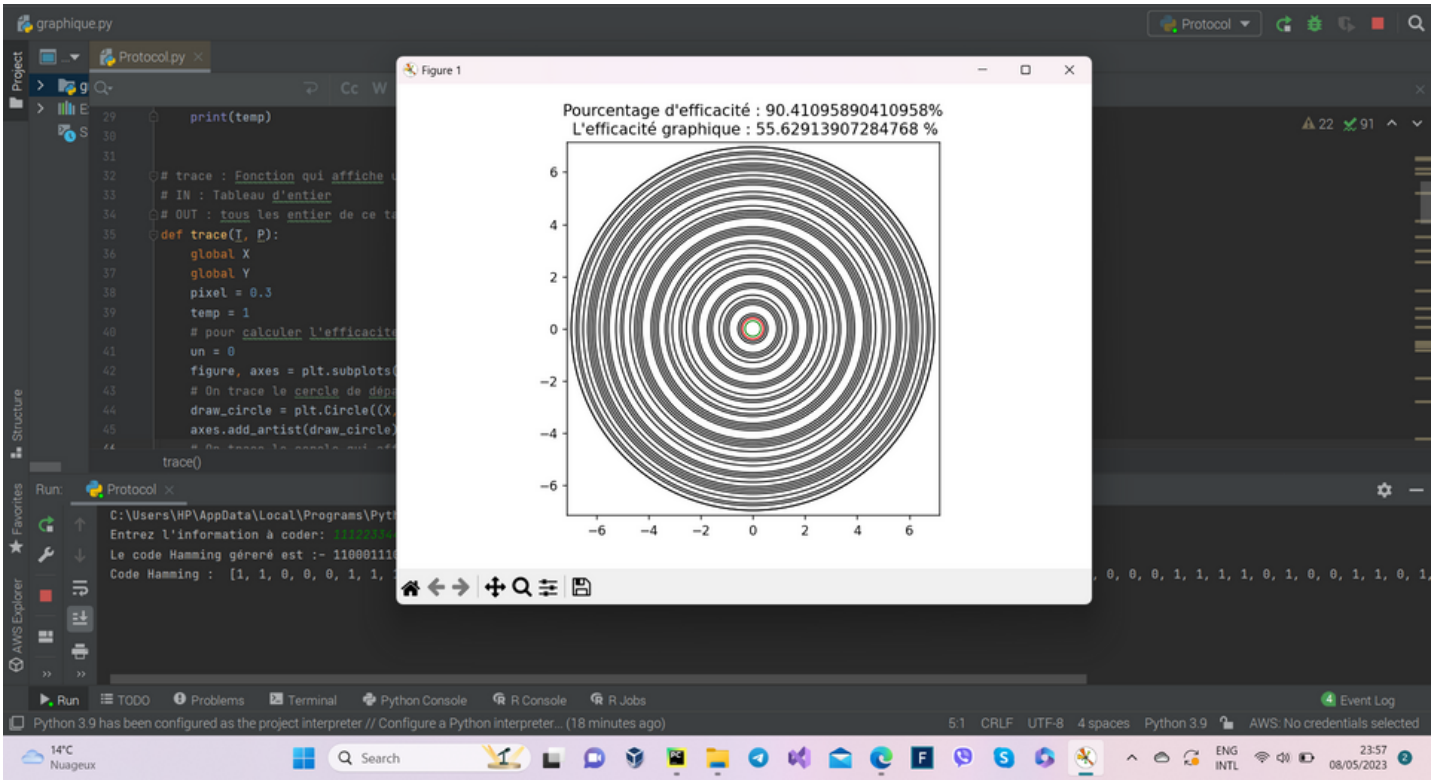
Fonctionnalitées:

- getTab(n): génère une liste de n nombres aléatoires 0s et 1s.
- printTab(tab): affiche une grille 5x5 des nombres dans tab.
- trace(tab, P): génère un graphique qui représente le code de Hamming. La fonction prend une liste de 0s et de 1s en entrée, tab, et un entier P qui représente le pourcentage de perte de données. Le graphique se compose d'une série de cercles concentriques, chaque cercle représentant un bit dans le code de Hamming.
- charToAscii(char): convertit un caractère char en son code ASCII correspondant.
- intToBit(n, b, verbose): convertit un entier n en sa représentation binaire. L'argument optionnel b spécifie la base à laquelle convertir (la valeur par défaut est 2). L'argument optionnel verbose spécifie s'il faut afficher les étapes de la conversion.
- strToBit(string): convertit une chaîne de caractères en sa représentation binaire, en appelant charToAscii et intToBit sur chaque caractère de la chaîne.
- hamming(tab): génère le code de Hamming pour la liste d'entrée de bits tab.
- listToString(s): convertit une liste de caractères en une chaîne de caractères.

Captures et Exemples :






graphique.py
Protocol.py
Figure 1
Pourcentage d'efficacité : 90.41095890410958%
L'efficacité graphique : 55.62913907284768 %
print(temp)
trace()
Run: Protocol
Entrez l'information à coder: 1112233
Le code Hamming géreré est :- 11000111
Code Hamming : [1, 1, 0, 0, 0, 1, 1,
Run TODO Problems Terminal Python Console R Console R Jobs
Event Log
Python 3.9 has been configured as the project interpreter // Configure a Python interpreter... (18 minutes ago)
5:1 CRLF UTF-8 4 spaces Python 3.9 AWS: No credentials selected


Run: Protocol ProGraph
C:\Users\HP\AppData\Local\Programs\Python\Python39\python.exe C:/Users/HP/Desktop/Tp_R/protocole-graphique/Protocol.py
Entrez l'information à coder: uca.fr
Le code Hamming géreré est :- 1110101110001110100001101110110001101100011010
Code Hamming : [1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0]
Run TODO Problems Terminal Python Console R Console R Jobs

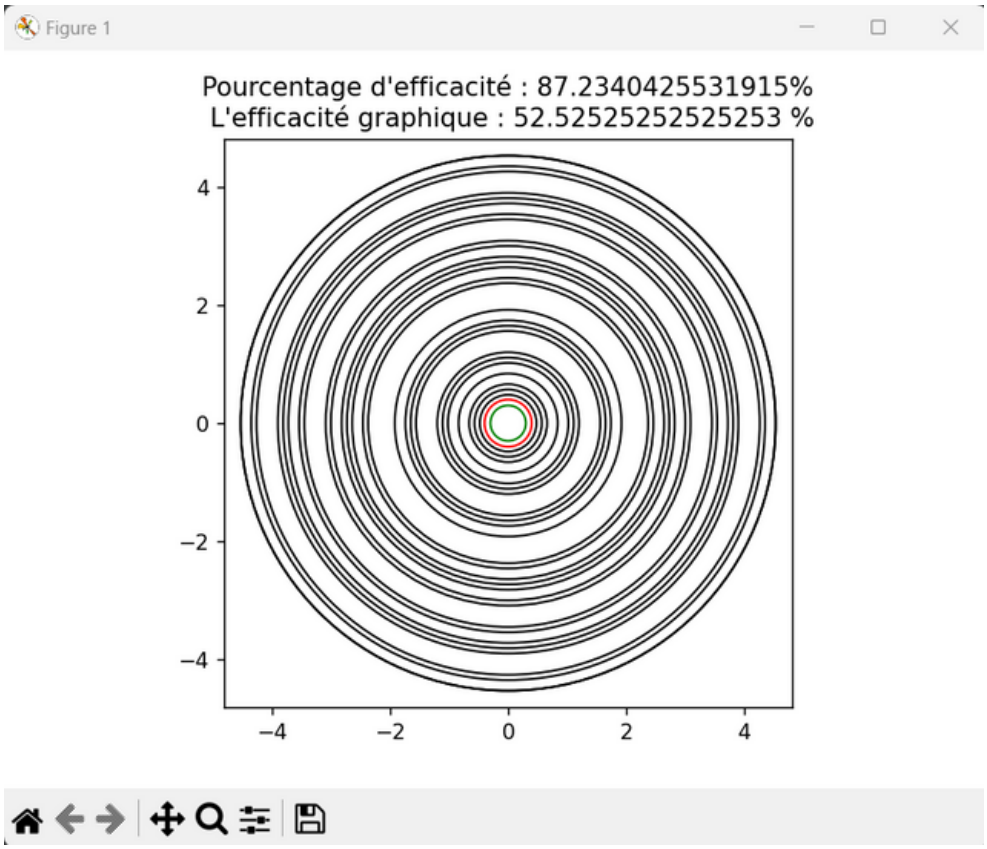


Figure 1
Pourcentage d'efficacité : 87.2340425531915%
L'efficacité graphique : 52.52525252525253 %

Orientation

Notre protocole offre la possibilité de lire l'information sous tous les angles, grâce à sa forme circulaire. Peu importe l'angle de vue, il est possible de récupérer les données en observant simplement les cercles. Il suffit d'une seule vue où les cercles sont visibles pour obtenir l'ensemble des informations.

Améliorations Possible :

- Ajouter une interface utilisateur : pas d'une interface utilisateur, On envisage à ajouter une interface pour faciliter l'utilisation et rendre l'application plus conviviale.

- Amélioration la précision et la robustesse : nous pouvons essayer d'améliorer la précision et la robustesse .

- Implemention de LOGO : Nous pouvons mettre facilement un logo sur notre protocole car celui-ci a besoin d'un seul angle pour lire l'image .

- Optimisation des performances : nous pouvons travailler à optimiser les performances de notre code pour améliorer la réactivité de l'application et pour la gestion des codes plus complexes.

- Intégration des tests unitaires : Pour améliorer la qualité de notre code et éviter les erreurs, vous pouvez intégrer des tests unitaires qui permettent de vérifier que chaque partie de votre code fonctionne comme prévu.

- Optimisation la sécurité : vous pouvons travailler à optimiser la sécurité de votre code pour garantir la confidentialité et l'intégrité des données.

CONCLUSION

En conclusion, la réalisation de ce TP sur le protocole graphique a été très enrichissante pour nous. on a pu découvrir et comprendre le fonctionnement des codes correcteurs d'erreur, ainsi que leur utilisation dans les communications numériques. on a également appris à manipuler des images en Python et à utiliser différentes bibliothèques telles que Pillow et Matplotlib.

L'un des aspects les plus intéressants de ce TP a été la mise en pratique des concepts théoriques à travers la création d'un protocole graphique. on a pu expérimenter avec différentes tailles de messages et différentes taux d'erreurs pour voir comment le protocole se comportait dans des conditions variées.

De plus, on a également pu constater les limites du protocole graphique, notamment en termes d'esthétique lorsque le nombre de bits devient trop élevé. Cela m'a conduit à réfléchir aux améliorations possibles, telles que l'utilisation de couleurs pour faciliter la lecture.

Merci