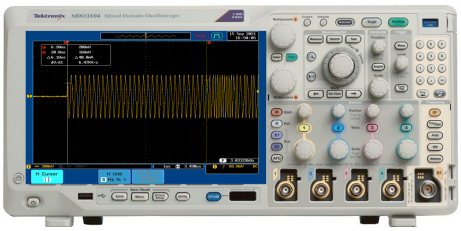
BTS CIEL 2ème année

PHYSIQUE/INFORMATIQUE

Décodeur Ethernet 10BASE-T

Lycée Jean Rostand, Villepinte



But du projet

Ce projet vise à explorer et à comprendre les principes fondamentaux des réseaux Ethernet à travers l'étude de l'encodage et du décodage des signaux de communication. L'objectif principal est de concevoir un système permettant de capturer et d'analyser des trames Ethernet en utilisant des outils physiques (oscilloscope) et des scripts Python pour automatiser et visualiser le processus. Ce projet permet également de se familiariser avec les bases de la programmation orientée objet (POO) et la création d'interfaces graphiques avec PyQT5 pour l'interaction avec les utilisateurs.

Mots-clés

- Ethernet
- NRZ (Non-Return-to-Zero)
- Encodage Manchester
- Décodage de trame
- Signal numérique
- Oscilloscope
- Python
- Programmation Orientée Objet (POO)
- Interface graphique
- PyQT5
- Cybersécurité

Structure du projet

Ce projet est organisé en cinq parties progressives. Chaque partie comporte des objectifs précis, des activités à réaliser, et des consignes spécifiques.

- **Partie 1 :** Étude des signaux NRZ/Manchester
- **Partie 2 :** Visualisation et acquisitions des trame Ethernet 10BASE-T
- **Partie 3 :** Décodage d'une trame Ethernet
- **Partie 4 :** Programmation Orientée Objet (POO)
- **Partie 5 :** Interface graphique (PyQT5)

Partie 1 : Étude des signaux NRZ/Manchester

Objectifs de la partie

- Comprendre les principes des encodages NRZ et Manchester.
- Écrire un script Python pour générer ces signaux.

Travail à faire :

- Activité **Capytale** : https://capytale2.ac-paris.fr/web/c/997a-3712751.
- Avec un **codeSpace** ou environnement cloné **GIT**

1. Créer un module Python contenant deux fonctions d'encodage : **`encode_NRZ`** et **`encode_Manchester`**.
2. Permettre à un programme principal d'utiliser ces fonctions via un **`import`**, le message binaire étant passé en argument à ce programme.
3. Dans le programme principal, les fonctions implémentées devront répondre aux assertions suivantes :
   - Pour NRZ:

     sortie = ([0.0, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0006, 0.0007, 0.0008, 0.0009, 0.001, 0.0011, 0.0012, 0.0013, 0.0014, 0.0015, 0.0016, 0.0017, 0.0018, 0.0019, 0.002, 0.0021, 0.0022, 0.0023, 0.0024, 0.0025, 0.0026, 0.0027, 0.0028, 0.0029, 0.003, 0.0031, 0.0032, 0.0033, 0.0034, 0.0035, 0.0036, 0.0037, 0.0038, 0.0039], [5, 5, 5, 5, 5, 5, 5, 5, 5, 5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5])

     assert encode_NRZ("00001000001100000", 5, -5, 10) == sortie , "Résultat invalide"
   - Pour Manchester:

     sortie = ([0.0, 0.0001, 0.0002, 0.0003, 0.0004, 0.0005, 0.0006, 0.0007, 0.0008, 0.0009, 0.001, 0.0011, 0.0012, 0.0013, 0.0014, 0.0015, 0.0016, 0.0017, 0.0018, 0.0019, 0.002, 0.0021, 0.0022, 0.0023, 0.0024, 0.0025, 0.0026, 0.0027, 0.0028, 0.0029, 0.003, 0.0031, 0.0032, 0.0033, 0.0034, 0.0035, 0.0036, 0.0037, 0.0038, 0.0039], [5, 5, 5, 5, 5, -5, -5, -5, -5, -5, -5, -5, -5, -5, -5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, -5, -5, -5, -5, -5, 5, 5, 5, 5, 5, -5, -5, -5, -5, -5])

     assert encode_manchester("00001000001100000", 5, -5, 10) == sortie , 'Résultat invalide'
4. L'execution du script à la CLI sauvegardera un graphique sous forme d'image png avec `plt.save('fig.png')`.
5. Tester les fonctions sur un ensemble d'exemples.

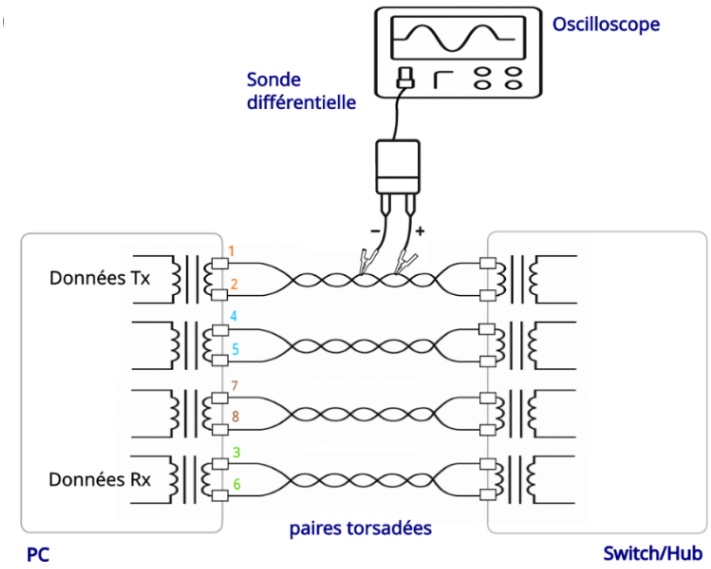
Partie 2 : Visualisation et acquisitions des trame Ethernet 10BASE-T

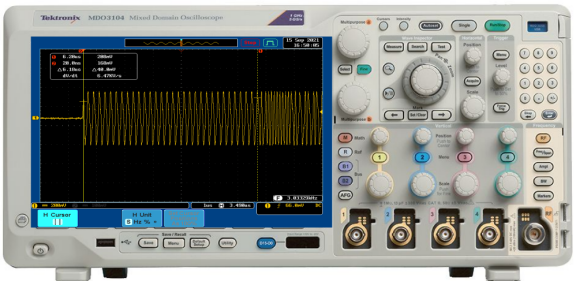
Objectifs de la partie

- Visualiser les trames Ethernet 10BASE-T
- Analyser et valider les caractéristiques de la transmission.
- Décoder manuellement le début de quelques trames.
- Enregistrer les trames dans des fichiers CSV.

Activité :

TP PHYSIQUE: TP1b: Codage Manchester et trame Ethernet





Enregistrer plusieurs trames complètes dans des fichiers CSV en indiquant le modèle de l'oscillo

Partie 3 : Décodage d'une trame Ethernet

Objectifs de la partie

- Analyser des trames Ethernet capturées avec un oscilloscope.
- Décoder manuellement et par script un signal Manchester réel.

Activité :

- Activité Capytale : https://capytale2.ac-paris.fr/web/c/4ff2-3798512.
- À l'aide de l'activité 2 précédente, élaborer un programme python, exécutable depuis la console permettant d'extraire les informations décodées en hexadécimal depuis un fichier CSV passé en paramètre (à télécharger depuis capytale).

  Exemple:

  Exécution du Script

  Commande :

  ```
  python mon_script.py /chemin/fichier.CSV
  ```

  Retour console :

  ```
  Destination :   ff:ff:ff:ff:ff:ff
  Source :        dc:4a:3e:41:e4:7c
  Ethernet Type : 86 dd   (Cette trame est de type ARP)
  Données :       40 23 45 ...etc.
  ```

- Coder le script dans un repo Git et pousser celui-ci vers le repo assignement fourni en début d'année (créer un dossier Partie 2).

Remarques :

- Exclure toute intégration de graphiques avec Matplotlib dans le script.
- Les variables (*ex. période d'échantillonnage, débit*) doivent être déclarées au début du script.
- Aucune valeur numérique ne doit être insérée directement dans les fonctions.

Partie 4 : Programmation Orientée Objet (POO)

Objectifs de la partie

- Structurer le script en utilisant des classes et des méthodes.
- Calculer dynamiquement le nombre d'échantillons par bit à partir du fichier CSV.

- Modifier la fonction permettant de charger les données du fichier CSV afin de récupérer également l'intervalle entre deux échantillons ("Sample Interval").
- Développer le module (fichier) Python correspondant au diagramme de classes ci-dessous.
- Implémenter chaque classe dans ce module.
- Établir le diagramme de séquence de votre programme principal, un lien vers ce document devra être établi dans un fichier README.md.
- Créer un programme principal qui lance l'application en important les classes du module.

Diagramme de classes UML :

https://drive.google.com/file/d/1-00eV7hbzm5ALyce$_i o_j hkg3CJDgp49/view?usp = sharing$

Rappel : Création de classes en Python

Pour créer une classe en Python, on utilise le mot-clé `class`. Voici les étapes principales de création d'une classe :

1. **Définition de la classe** : Utilisez `class NomDeClasse:` pour définir une nouvelle classe.
2. **Constructeur `__init__`** : Définissez un constructeur pour initialiser les attributs de la classe. Le mot-clé `self` fait référence à l'instance actuelle.
3. **Méthodes** : Ajoutez des fonctions à l'intérieur de la classe pour définir le comportement des objets créés.

Exemple

Voici un exemple de classe simple pour modéliser un étudiant :

```
class Etudiant:
    def __init__(self, nom, age):
        self.nom = nom
        self.age = age

    def se_presenter(self):
        return f"Bonjour, je m'appelle {self.nom} et j'ai
           {self.age} ans."
```

```python
# Exemple d'utilisation
etudiant1 = Etudiant("Alice", 20)
print(etudiant1.se_presenter())
```

Listing 1: Classe `Etudiant`

Création d'un module Python avec importation dans le programme principal

Pour organiser un projet Python, il est souvent utile de créer des modules. Cela consiste à séparer le code en plusieurs fichiers pour une meilleure lisibilité et réutilisabilité.

Étapes principales

1. **Créer un fichier pour la classe** : Placez la définition de votre classe dans un fichier indépendant (par exemple, `personne.py`).
2. **Importer la classe** : Importez la classe dans votre programme principal avec l'instruction `from nom_du_fichier import NomDeClasse`.
3. **Utiliser la classe** : Créez des instances de la classe et utilisez ses méthodes dans le programme principal.

Exemple : Classe `Personne`

Fichier `Personne.py` : Voici la définition de la classe `Personne`.

```python
class Personne:
    def __init__(self, nom, age):
        self.__nom = nom
        self.__age = age

    def repeter(self, argument):
        """Retourne l'argument fourni."""
        return argument
```

Listing 2: Fichier `Personne.py`

Fichier principal `main.py` : Voici comment utiliser la classe `Personne` dans un autre fichier.

```python
from personne import Personne

# Creation d'une instance de la classe Personne
p1 = Personne("Alice", 30)

# Appel d'une methode de la classe
resultat = p1.repeter("Bonjour, je suis une personne!")
print(resultat)
```

Listing 3: Fichier `main.py`

Points importants

- **Nom des fichiers** : Assurez-vous que le fichier contenant la classe porte un nom descriptif et respecte les conventions (par exemple, tout en minuscules, avec des underscores si nécessaire).
- **Organisation des imports** : Placez les importations en haut de votre fichier principal.
- **Réutilisation** : Une classe définie dans un fichier peut être utilisée dans plusieurs programmes, ce qui facilite la maintenance et la modularité.

Partie 5 : Interface graphique (PyQT5)

Objectifs de la partie

- Créer une interface graphique pour exploiter les données des oscilloscopes.
- Ajouter des fonctionnalités interactives pour analyser les trames Ethernet.

- Suivre le tutoriel à l'adresse Tutoriels PyQt5 sur YouTube
- Pour chaque vidéo, réaliser un commit dans le dossier pyQT_tutoriel préalablement créé.

Spécifications :

Interface graphique :

- Menu déroulant pour sélectionner l'oscilloscope (*Tektronix MSO, TDS2012, Rigol IDS*):
  $\longrightarrow$ Créer une méthode distincte pour chaque type d'oscilloscope après analyse des CSV.
- Bouton pour charger un fichier CSV.
- Graphique affichant la trame complète.
- Boutons interactifs pour analyser différentes parties :
  - → Adresse Destination
  - → Adresse Source
  - → Type
  - → Données

Fonctionnalités avancées :

- Zoom sur une partie sélectionnée (graphique dédié).
- Encadré coloré pour localiser la section analysée.
- Décodage hexadécimal affiché.
- Décodage binaire sur le signal zoomé (sauf pour la section Données).

Vidéo illustrative

Vous pouvez vous inspirer de la vidéo suivante:

https://urlz.fr/tn7R