

# RAPPORT DE PROJET

## RESEAU AVANCEE

MSILNI Yassine

&

UZAN Gabriel

&

BOUND AOUI Sarah

&

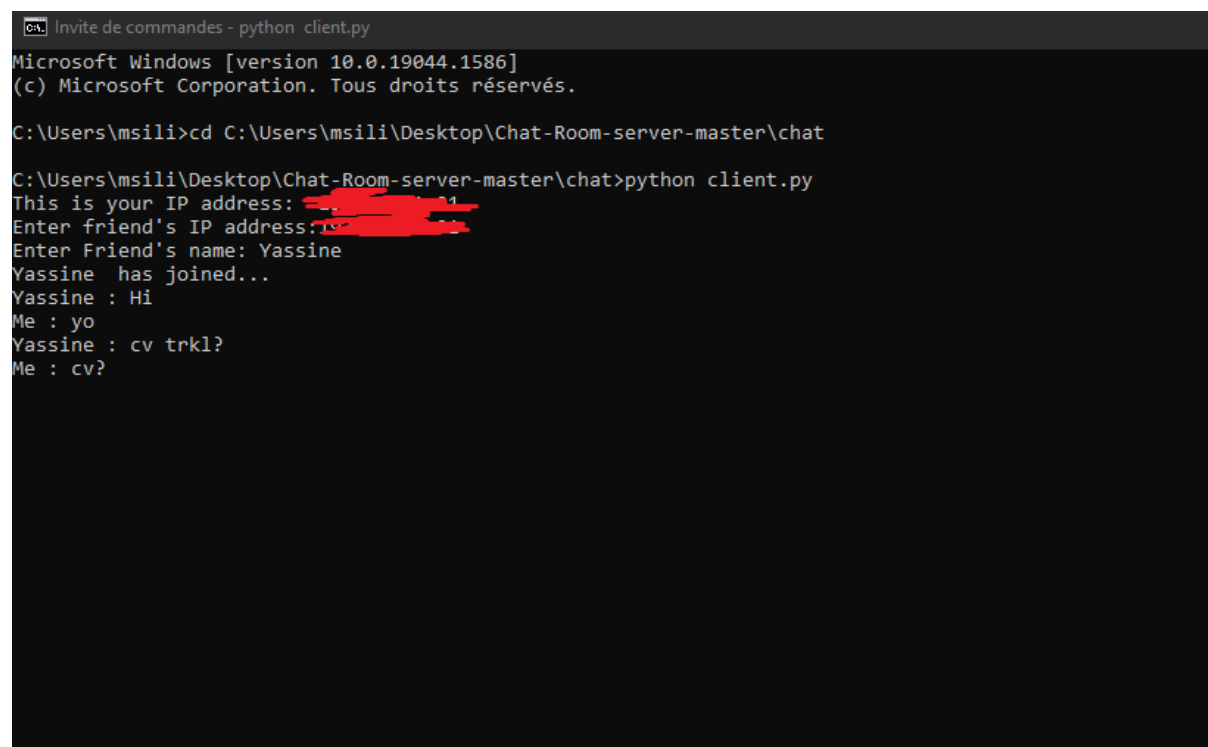
ABASSE Anil

Référent de matière : Monsieur Martin <[steven.martin@lri.fr](mailto:steven.martin@lri.fr)>

Chargée de TD : Madame TOUATI <[h.h.h.touati@gmail.com](mailto:h.h.h.touati@gmail.com)>

## Introduction

Le projet de Réseau a pour but de nous faire utiliser nos connaissances en programmation réseau mais aussi nos connaissances en réseau DHCP/TCP pour réaliser une application sur des nouvelles machines que sont les Raspberry Pi. L'application ainsi que le langage de programmation sont aux choix des groupes, en effet la seule condition est l'utilisation et la mise en place donc d'un réseau ADHOC sur les Raspberry Pi et donc par extension une application qui met en relation les machines. Voici à quoi pourrait ressembler l'interface de cette application :

A screenshot of a Windows command prompt window. The title bar reads 'Invite de commandes - python client.py'. The window shows the following text:

```
Microsoft Windows [version 10.0.19044.1586]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\msili>cd C:\Users\msili\Desktop\Chat-Room-server-master\chat

C:\Users\msili\Desktop\Chat-Room-server-master\chat>python client.py
This is your IP address: 192.168.1.1
Enter friend's IP address: 192.168.1.2
Enter Friend's name: Yassine
Yassine has joined...
Yassine : Hi
Me : yo
Yassine : cv trkl?
Me : cv?
```

## Analyse Globale

Nous allons réaliser un projet en python en créant un simple chat avec une interface graphique ou plusieurs personnes pourront rejoindre un salon et discuter entre eux, ce projet devra respecter le modèle ADHOC comme précédemment dit, nous allons donc répartir nos classes de façon que chacune d'entre elles respectent la disposition ADHOC mais nous devons aussi faire en sorte qu'elles soient supportables par nos machine Raspberry Pi. Ce projet devra être réalisé sur une durée de huit semaines. Pour implémenter notre chat, nous allons devoir implémenter de quelques fonctionnalités. Une interface graphique afin d'afficher le salon et la discussion divisé en 2 parties qui nous servira aussi d'interface avec l'utilisateur.

Nous allons aussi configurer les Raspberry Pi en réseau ADHOC permettant une connectivité DHCP de type CLIENT + SERVER intra Raspberry Pi en utilisant le Raspbian.

## Plan de développement

Dans le Plan de Développement nous allons lister toutes les fonctionnalités et sous-fonctionnalités qui vont être développées et implémentées dans le projet.

Nous allons suivre une méthode simple : diviser notre travail sous deux catégories : les fonctionnalités principales et les sous-fonctionnalités secondaire mais tout de même importante pour avoir un projet plus complet et agréable d'utilisation.

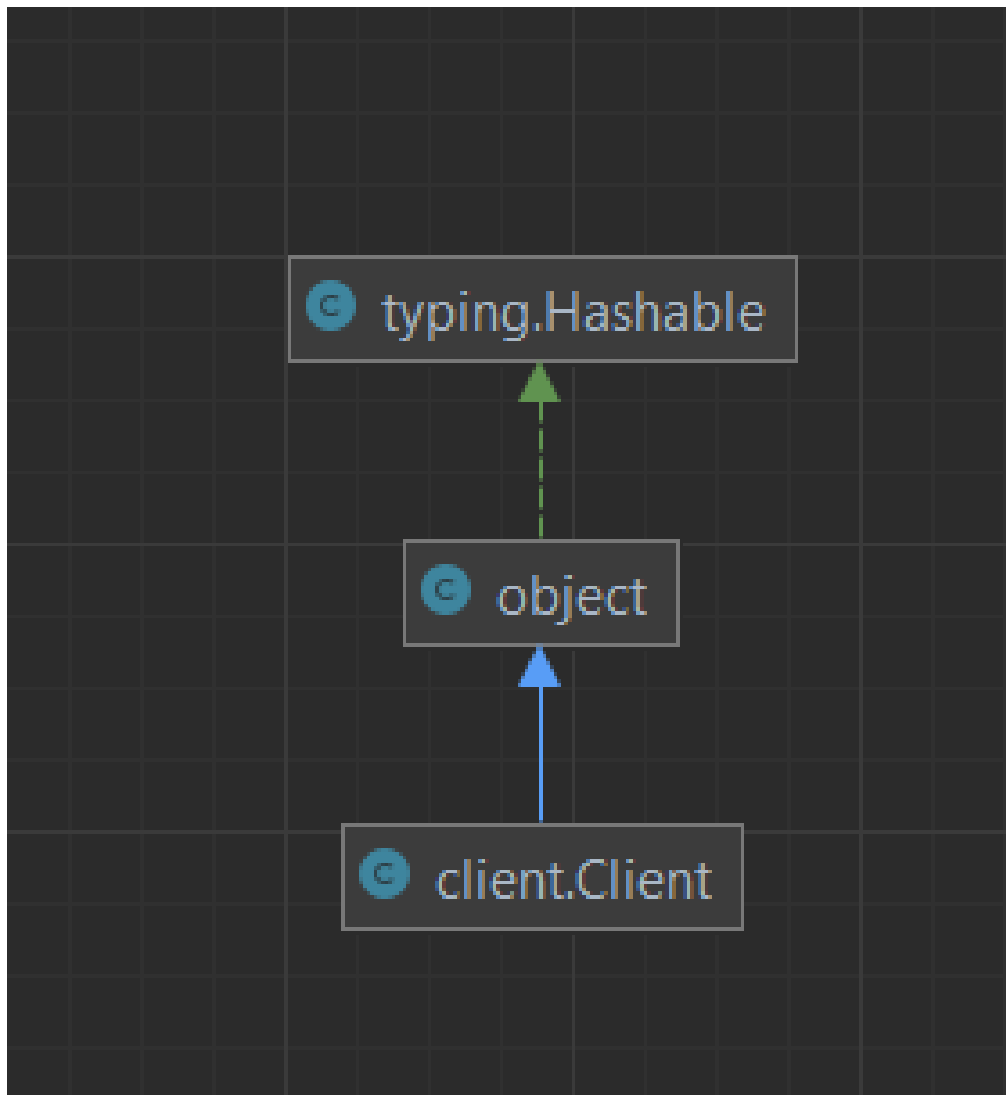
### Fonctionnalités :

- Affichage de la fenêtre et des différents panels
- Historique déroulant du chat
- Notifications d'arrivées d'une personne et de départ
- Affichage du numéro de client selon sa position de d'arrivé sur le serveur
- Horloge
- Box permettant l'écriture d'un message
- Bouton d'envoi d'un message
- Notifications dans le terminal server
- Retranscription des messages dans le terminal server
- Configuration des Raspberry en ADHOC
- Gestion d'adressage ADHOC

## Conception Générale

Ce projet a été réalisé en utilisant python avec un modèle ADHOC. Ce modèle consiste à prendre en compte l'adressage HOST ainsi que le PORT utilisé pour demander une autorisation de connexion. Nous avons des Raspberry qui servent de clients ainsi que de server car nous sommes en intra connexion nous. Ce modèle permet d'avoir une meilleure lisibilité du projet et de mieux comprendre l'impact d'une connectivité et le changement drastique d'efficacité et de coût par rapport à un autre.

Notre projet contient deux classes principales ayant tout le contenu du projet :



La classe server qui contient tout le code qui gère toute la connectivité serveur client :

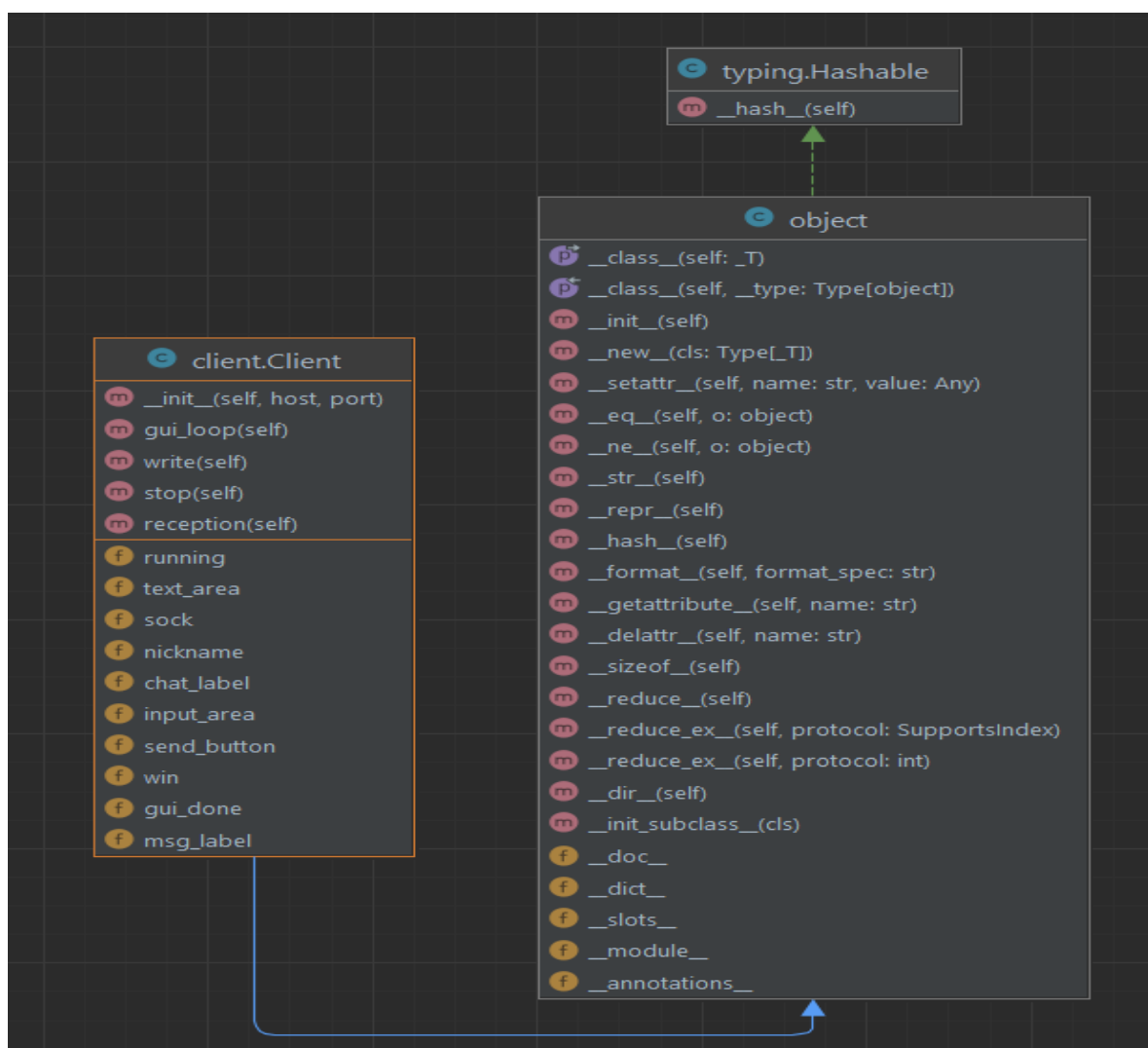
Elle contient 3 méthodes dont nous parlerons plus tard dans la conception détaillée :

- `diffusion(message)`
- `tri(client)`
- `reception()`

Nous avons ensuite la classe client qui contient le code de l'interface du salon de chat ainsi que le tri des demandes de connexion et de l'envoi de signaux encoded :

- `__init__(self, host, port)`
- `gui_loop(self)`
- `write(self)`
- `stop(self)`
- `reception(self)`

Voici plus en détail notre diagramme de classe avec toutes les méthodes et tous les attributs de nos deux classes :



## Conception Détaillée

### Affichage de la fenêtre et des différents panels

Le JFrame ou TKinter représente notre fenêtre de salon qui contient les différents panels :

- `chat_label`

S'occupant de l'affichage de la box de chat ou une personne peut écrire son message et pour avoir un aperçu de l'envoi.

- `text_area`

S'occupant de l'affichage du chat de la conversation et de l'historique du chat.

- `msg_label`

S'occupant de l'affichage du message au-dessus de la box de chat en rouge.

- `input_area`

S'occupant de l'input du texte dans le chat\_label

- `send_button`

S'occupant de l'affichage du bouton d'envoi du message.

Voici un aperçu :



Notre Tkinter est créé dans la classe client qui utilise aussi plusieurs attributs cités précédemment :

```
self.chat_label = tkinter.Label(self.win, text="Chat:", bg="lightgray")
self.chat_label.config(font=("Arial", 12))
self.chat_label.pack(padx=20, pady=5)

self.text_area = tkinter.scrolledtext.ScrolledText(self.win)
self.text_area.pack(padx=20, pady=5)
self.text_area.config(state='disabled')

self.msg_label = tkinter.Label(self.win, text="Message:", bg="red")
self.msg_label.config(font=("Arial", 12))
self.msg_label.pack(padx=20, pady=5)

self.input_area = tkinter.Text(self.win, height=3)
self.input_area.pack(padx=20, pady=5)

self.send_button = tkinter.Button(self.win, text="Send", command=self.write)
self.send_button.config(font=("Arial", 12))
self.send_button.pack(padx=20, pady=5)
```

### La partie réseau et interactions

Nous avons ensuite les méthodes `__init__(self, host, port)`, `write(self)`, `stop(self)` et `reception(self)`

Les méthodes `write` et `stop` sont utilisées par `gui_loop` et permettent l'envoi du message de manière codée via le `.encode('utf-8')` comme ceci :

```
def write(self):
    message = f"{self.nickname}: {self.input_area.get('1.0', 'end')}"
    self.sock.send(message.encode('utf-8'))
    self.input_area.delete('1.0', 'end')
```

Ainsi que le stoppage de la fenêtre du salon en cas d'erreur ou de départ d'un participant via le `destroy` de la `Windows` ainsi que le `socket.close()` :

```
def stop(self):  
    self.running = False  
    self.win.destroy()  
    self.sock.close()  
    exit(0)
```

Puis pour finir, nous avons les méthodes de réception et `__init__` permettant de gérer la partie réseau qui permet l'envoi du message au server pour l'afficher dans le chat puis de le décoder.

### La partie réseau du server

Nous créons le server via des sockets puis nous utilisons une simple méthode `diffusion(message)` qui envoie tout simplement le signal :

```
client.send(message)  #envoi du signal qui est le msg
```

Nous avons ensuite la méthode la plus difficile et importante du projet, le `handler` `tri(client)` qui envoie le message via le `diffusion(message)` si tout est correcte ou sinon qui fermera l'instance s'il y a une erreur.

Pour finir la partie codage python, nous avons la méthode `reception()` qui gère la création des usernames ainsi que l'acceptation des connexions rentrante.

Le dernier point de cette conception et la mise en place du réseau ADHOC :

Nous sommes passés par plusieurs méthodes tous résultant d'un échec comme la modification du fichier `network` dans le `/opt` ou encore l'installation de packages via le terminal mais seulement a marcher pour nous et elle résulte de plusieurs étapes qui sont :

### La mise à jour système :

```
sudo apt-get update
```

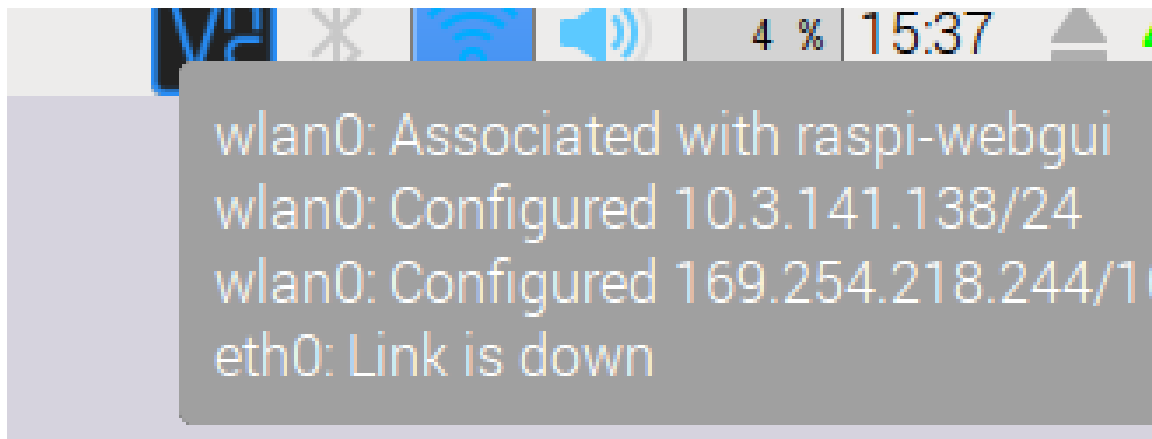
```
sudo apt-get upgrade
```



## L'installation de la bibliothèque Rasp :

```
wget -q https://git.io/voEUQ -O /tmp/raspap && bash /tmp/raspap
```

## La connexion aux hotspot ADHOC raspi webgui :



## La page du server :

The screenshot shows the RaspAP Wifi Portal v1.3.1 web interface. The interface is in French and displays the following information:

- Tableau de bord** (Dashboard)
- Informations d'interface** (Interface Information):
  - Nom de l'interface: wlan0
  - Adresse IP: 10.3.141.1
  - Masque de sous-réseau: 255.255.255.0
  - Adresse Mac: b8:27:eb:7f:28:69
- Statistiques d'interface** (Interface Statistics):
  - Paquets reçus: No Data
  - Octets reçus: No Data
  - Paquets transférés: No Data
  - Octets transférés: No Data
- Informations sans fil** (Wireless Information):
  - Connecté à: Not connected
  - AP Mac Adresse:
  - Bitrate:
  - Niveau du signal: 31 dBm
  - Puissance de transmission: La fréquence
  - Qualité de lien:
- Buttons:** Stop wlan0, Recharger
- Footer:** Informations fournies par ifconfig et iwconfig

## Résultat

```
Invité de commandes - python serveur.py
Microsoft Windows [version 10.0.19044.1586]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\msili>cd C:\Users\msili\PycharmProjects\Projet-Reseau

C:\Users\msili\PycharmProjects\Projet-Reseau>python serveur.py
Server running...
Connexion depuis ('...', 55424)
```

Please choose a nickname

OK Cancel

```
Invité de commandes - python serveur.py
Microsoft Windows [version 10.0.19044.1586]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\msili>cd C:\Users\msili\PycharmProjects\Projet-Reseau

C:\Users\msili\PycharmProjects\Projet-Reseau>python serveur.py
Server running...
Connexion depuis ('...', 55438)
Votre pseudonyme est b'Sarah: .\n'
b'Sarah: .\n' says b'Sarah: Hello world !\n'
b'Sarah: .\n' says b'Sarah: Test 2 !\nHi !\n'
b'Sarah: .\n' says b'Sarah: LOL!\n'
Connexion depuis ('...', 55448)
Votre pseudonyme est b'Yass: .\n'
b'Yass: .\n' says b'Yass: Hello!\n'
b'Yass: .\n' says b'Yass: Hi my friend !\n'
b'Yass: .\n' says b'Yass: \n'
b'Sarah: .\n' says b'Sarah: Hi !\n'
```

Chat

b'Sarah: .\n' s'est connecté au salon  
Vous vous êtes connecté au salon !Sarah: Hello world !  
Sarah: Test 2 !  
Hi !  
Sarah: LOL  
b'Yass: .\n' s'est connecté au salon  
Yass: Hello  
Yass: Hi my friend !  
Yass:  
Sarah: Hi !

Message

Send

Chat

b'Yass: .\n' s'est connecté au salon  
Vous vous êtes connecté au salon !Yass: Hello  
Yass: Hi my friend !  
Yass:  
Sarah: Hi !

Message

Send

## Documentation utilisateur

Pour que l'utilisateur puissent utiliser notre salon de chat plusieurs prérequis sont nécessaire :

En premier lieu un IDE comme PyCharm est nécessaire de préférence mais il est tout de même possible de lancer le jeu uniquement avec un terminal ayant python (sur linux) ou un CMD ayant un pathing python 3 sur Windows. Une fois dans l'IDE, l'utilisateur doit importer le fichier, aller dans la classe client puis exécuter le fichier **APRES** avoir lancer le server avec un python serveur.py dans le terminal/cmd et ainsi le chat se lancera, il faudra juste rentrer un pseudonyme et cliquer sur enter. Dans le cas où l'utilisateur n'utilise pas d'IDE, il aura juste à lancer deux terminaux l'un pour le server et l'autre pour le client (en local bien sûr mais si le server est déjà lancer sur une autre machine seulement le client.py suffit quel que soit la méthode choisies citées au-dessus).

Et pour finir, la connectivité ADHOC entre toutes les machines SAUF si la personne l'utilise en local sur une seule machine.

## Conclusion et perspectives

Nous voilà au terme de ce projet de Réseau dans laquelle nous avons fait l'implémentation de nombreuses fonctionnalités mais malgré ces efforts il nous reste encore beaucoup de chose à faire que ça soit niveau code pour l'optimiser ou l'ajout de nouvelles fonctionnalités pour une meilleur expérience utilisateur qui nous permettra de rendre ce chat plus complet comme l'horloge, les pop-d'alerte ou encore la personnalisation de l'interface mais aussi des commandes comme /kick ou /ban.

Les difficultés rencontrées nous ont permis de comprendre nos mauvaises habitudes sur le manque de commentaires sur le code car nous sommes plusieurs à lire le même code ce qui freine beaucoup la compréhension du code des autres sans commentaires mais aussi l'utilisation de python pour le codage réseau ainsi que la bibliothèque d'interface TKinter.

Le tri des signaux et la méthode très compliquée qu'est le `__init__` en programmation nous ont permis de comprendre l'importance d'une optimisation du code pour une utilisation réseau dépendant de l'utilisation voulu (TCP/DHCP etc.).

Pour finir ce projet nous a permis d'apprendre à programmer à plusieurs tout en respectant une méthode de programmation qui est commune à tous qui dans notre cas est le modèle ADHOC et nous en ressortons une expérience non négligeable qui nous aidera sans l'ombre d'un doute dans notre avenir.