Contrôler un objet distant

Dans ce TP, nous allons devoir sauver le monde. Nous sommes le 16 novembre 2022 et la centrale nucléaire de Winden est sur le point d'exploser. Le circuit de refroidissement a été détruit et si le niveau d'eau passe en dessous de 10 %, le cœur nucléaire va créer un trou de ver spatio-temporel. Une première explosion a endommagé le pare-feu qui protégeait le réseau interne de la centrale du reste du monde. Comme hacker, vous pouvez prendre le contrôle du système de refroidissement et sauver le monde.

Nous allons devoir nous familiariser avec ce réseau industriel.

Nous allons étudier un réseau industriel appelé Modbus. Il est apparu en 1979 à une époque où l'internet n'existait pas encore et est toujours très populaire dans l'industrie.

À l'origine, Modbus était construit sur un bus série qui connectait différents équipements appelé **secondaires** ou *slaves* et un **primaire** ou *master* qui gère les communications. Chaque secondaire a un numéro unique ou adresse. Les adresses sont comprises entre 1 et 247. Le primaire n'a pas besoin d'une adresse puisque toutes les communications ont lieu avec lui. Le primaire envoie une requête au secondaire et le secondaire répond au primaire. Des communications directes entre deux secondaires ne sont pas possibles.

Un équipement Modbus gère l'information sous forme des registres :

- Les registres qui peuvent prendre une valeur binaire *on* ou *off*. Si le master peut modifier l'état et, bien sûr, le lire, est appelé un ***coil***. Si la valeur binaire peut être uniquement lue, il s'agit d'un ***discrete input***.
- Les registres sur 16 bits. Ils sont utilisés pour représenter une valeur comme un courant électrique, une température, une vitesse de rotation… De même, si leur valeur ne peut être que lue par le primaire, il s'agit d'un ***input register*** sinon, si elle peut être également modifiée par le primaire, il est appelé ***holding register***.

Un équipement Modbus peut avoir jusqu'à 10000 registres de ces 4 catégories.

Modbus est un protocole requête/réponse. Le primaire envoie une requête à l'adresse d'un équipement pour lire ou écrire un de ses registres.

Une trame Modbus est une séquence de caractères commençant par un octet avec l'adresse du secondaire, suivi d'une commande (ou code de fonction) spécifique à chaque catégorie de registre :

- 1 pour lire un *coil*
- 2 pour lire un *discrete input*
- 3 pour lire un *input register*
- 4 pour lire un *holding register*
- 5 pour écrire un *holding register* et
- 6 pour écrire un *input register*

La suite de la trame contient les données, puis un CRC pour valider qu'il n'y a pas d'erreur de transmission dans la trame.

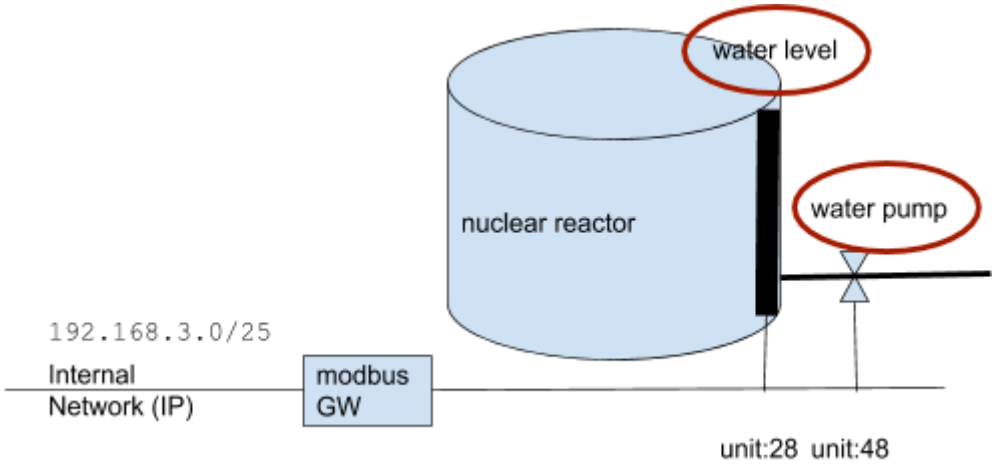
La partie donnée peut être différente dans la requête et la réponse.

Par exemple, pour lire un *holding register*, la requête contient l'adresse du premier registre à lire, et le nombre de registres à lire. La réponse contient le nombre d'octets transmis suivi de leur valeur.

Pour écrire sur un registre, les données de la trame seront l'adresse du registre et les données à écrire. La réponse est le même message.

QUESTION

En reprenant le schéma du réseau de la centrale nucléaire, donnez l'adresse de l'unité modbus qui contrôle la pompe à eau.
….



Les secondaires manipulent l'information sous forme de registres, soit binaires, soit sur 16 bits.

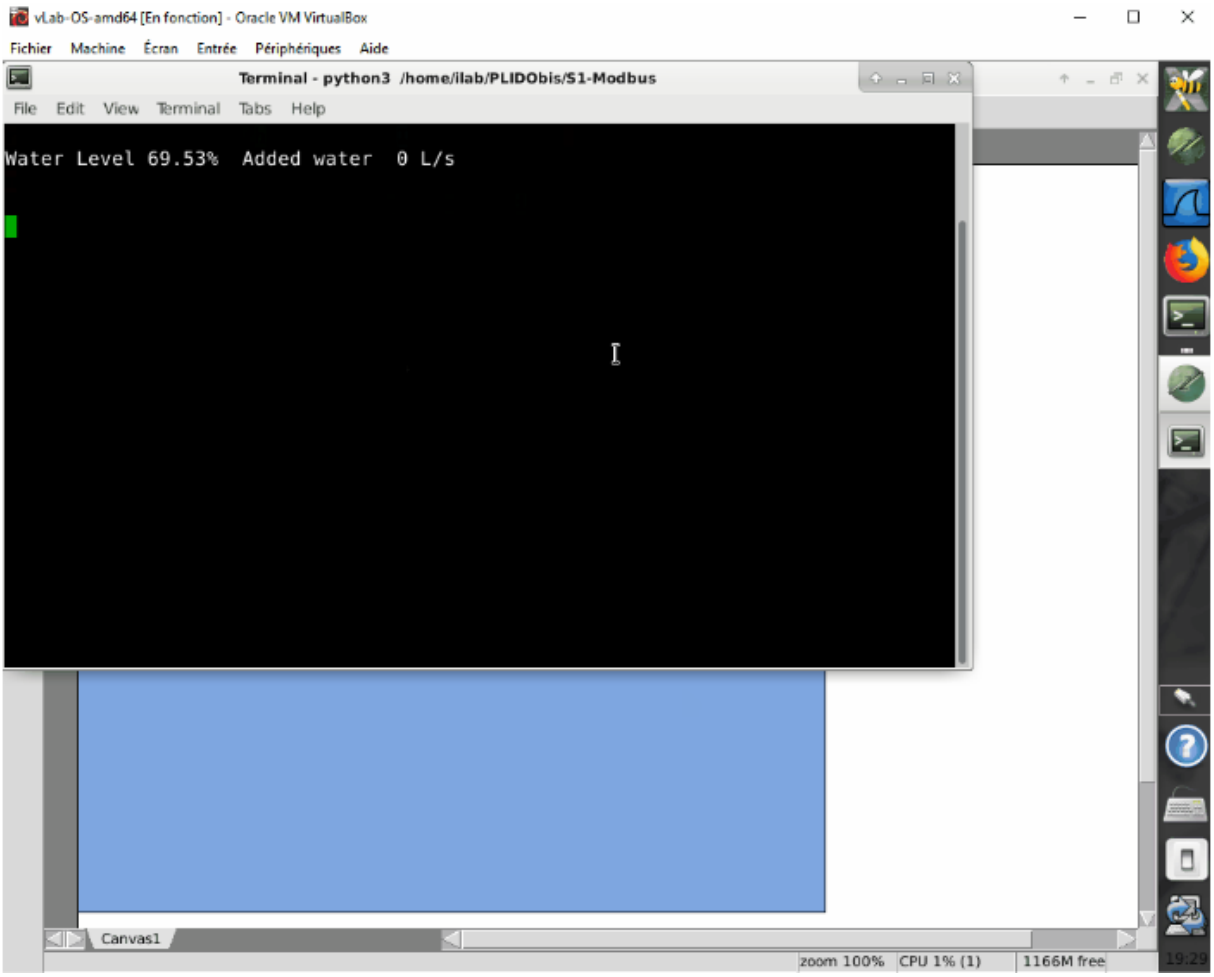
Démarrage du processus de la centrale nucléaire

Nous allons utiliser l'émulateur de réseau.

- Cliquez sur la deuxième icône du menu de droite.
- Allez sur le menu File > Open
- Choisissez le fichier /home/ilab/PLIDO/S1-Modbus/modbus.imn
- Cliquez sur la flèche verte dans le menu de gauche pour lancer l'émulation.
- Double-cliquez sur l'icône de la centrale nucléaire pour ouvrir un terminal.

- Accédez au répertoire /home/ilab/PLIDO/S1-Modbus
- et tapez : python3 ./winden.py

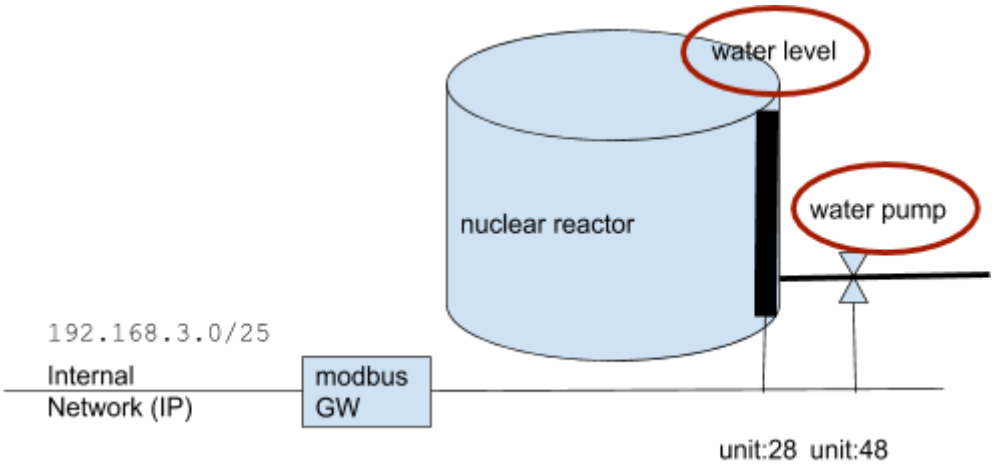
  La fenêtre va afficher le niveau d'eau.



Celui-ci baisse de plus en plus, faites vite ! Prenez le contrôle de la centrale. Cliquez sur le crâne et une autre fenêtre terminal va s'ouvrir : vous pourrez y exécuter votre code.

Vous êtes tombé sur un plan du réseau de la centrale qui donne les adresses des équipements.

On voit qu'il y a deux types de réseaux. Le réseau Internet en bas à gauche de la figure et le réseau Modbus qui contrôle une pompe pour l'eau et un équipement de mesure du niveau d'eau dans la cuve. Une passerelle (GW pour *gateway*) interconnecte les deux réseaux.



Plus précisément, la passerelle Modbus (*Modbus GW*) a l'adresse IPv4: **192.168.3.36/25.**

En cherchant sur le site web des fabricants de contrôleurs Modbus, vous trouvez ces informations sur les équipements de la centrale :

Pour le niveau d'eau :

Register address	description	Register Type	unit	size
00 FF	water level	input register	unsigned integer percentage * 100 ie 7512 correspond to 75.12%	2 bytes

Pour la pompe :

Register address	description	Register Type	unit	size
02 00	rotation speed	holding register	Unsigned integer giving pump rotation speed. 0 : stop pump	2 bytes

Le primaire est connecté au bus RS-485 via un port USB. Il est également possible de déporter le primaire du bus en utilisant une passerelle. Elle possède une adresse IP sur le réseau et si l'on émet des requêtes Modbus sur le bon port, elles seront transmises sur le bus. La réponse sera retournée à l'émetteur toujours encapsulée dans la connexion TCP/IP. Cela permet d'accéder à distance à des bus Modbus et à centraliser le traitement de l'information sur des serveurs.

Cela va nous permettre également d'accéder au système de contrôle de la centrale nucléaire. Donc, dans un premier temps, nous allons devoir localiser la passerelle sur le réseau IP de la centrale.

Pour communiquer avec la centrale, nous allons utiliser un module Python appelé tout simplement pymodbus. Voici un exemple de code pour lire la valeur d'un *coil* qui se trouve dans le programme save_the_world.py (dans le répertoire /home/ilab/PLIDO/S1-Modbus).

```
1   from pymodbus.client.sync import ModbusTcpClient # import modbus for Primary
2
3
4   # open connection with gateway at the specified address
5   client = ModbusTcpClient('127.0.0.1', port=5020)
6   client.connect()
7
8
9   # send a request to write a coil at register address 1 on unit 12
10  client.write_coil(address=1, value=True, unit=12)
11
12  # send a request to read 1 coil at register address 1 on unit 12
13  result = client.read_coils(address=1,count=1, unit=12)
14
15  # display the result
16  print(result.bits[0])
17
18  # finished, close the TCP connection with the gateway.
19  client.close()
```

Si vous lancez ce programme, vous allez avoir une erreur, car le programme ne peut pas ouvrir une connexion TCP avec la passerelle Modbus.

La première étape consiste à modifier l'adresse IP dans le programme précédent pour mettre celle de la passerelle Modbus (192.168.3.36). En relançant le programme, on a cette fois une erreur car le registre n'existe pas (à moins que la centrale ait déjà explosé et donc vous ne pourrez pas vous y connecter ; dans ce cas, on vous donne une seconde chance en remontant dans le temps et en relançant le programme winden.py).

Commençons notre session de *hacking* pour sauver le monde, et rappelez-vous : le temps est limité car le niveau d'eau baisse dangereusement.

La documentation du module pymodbus donne différentes méthodes pour manipuler les registres :

- read_coils,
- read_holding_registers,
- write_register,
- read_discrete_inputs,

- read_input_registers.

Ces méthodes prennent une partie des arguments suivants selon le type de méthode :

* unit: adresse du secondaire sur le bus
* address : l'adresse du registre dans le secondaire désigné
* value : valeur à écrire dans le registre désigné
* count : combien de registres doivent être lus à partir du registre désigné.

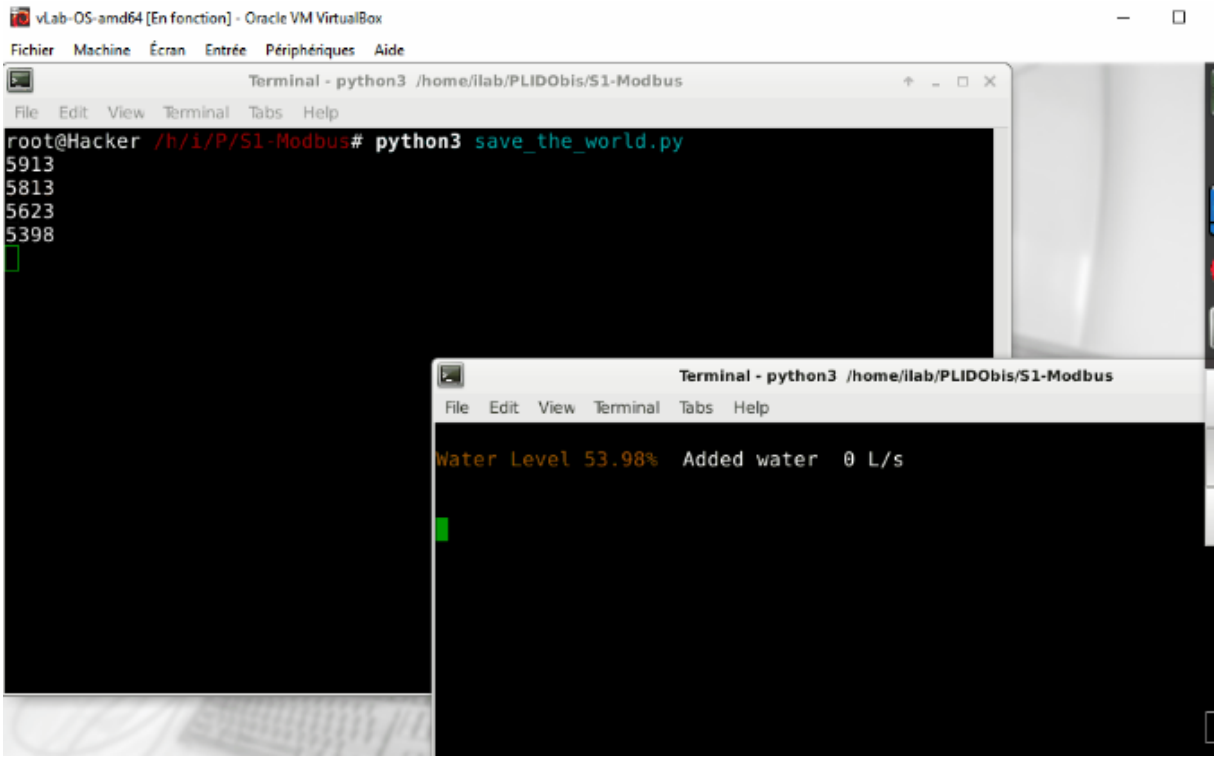
Obtenez le niveau d'eau

Vous avez pu vous connecter à la passerelle Modbus. Le monde est presque sauvé. Il faut maintenant lire le niveau d'eau dans la cuve. Pour cela vous devez encore modifier le programme :

- Pour lire le type de registre approprié en utilisant la bonne méthode,
- Modifier l'unit et l'adresse de registre pour interroger le niveau d'eau,
- Afficher le résultat en utilisant le tableau .registers[] à la place de .bits[].

Vous pouvez également inclure cette interrogation dans une boucle pour l'afficher périodiquement. En python, la fonction sleep() du module time permet d'attendre pendant un nombre de secondes spécifié en paramètre.

Résultat souhaité :



Question :

-Quelle est l'adresse (unit) de l'équipement mesurant le niveau d'eau ?

-Quel registre doit-être lu ?

-De quel type est ce registre ?

Coil Discrete input Input register Holding register

-Quelle méthode doit-être utilisée pour lire le niveau d'eau ?

read_coils, read_holding_registers, write_register, read_discrete_inputs, read_input_registers

-Quels arguments doivent-être fournis ?

unit address value count

Après avoir écrit le programme de surveillance du niveau d'eau, il reste à contrôler la vanne pour ajouter de l'eau, mais sans faire déborder la cuve.

Contrôlez la pompe

Dernière étape, mais le temps presse. On doit maintenant contrôler le niveau d'eau en activant la pompe. Attention à ne pas faire déborder la cuve car ça serait également une catastrophe. Le but est d'écrire une boucle de contrôle où l'on lit le niveau d'eau. S'il est inférieur à un certain niveau, on démarre la pompe, et on l'arrête s'il dépasse un seuil.

-De quel type est-ce registre ?

○ Coil ○ Discrete input ○ Input register ○ Holding register

-Quelle méthode doit-être utilisée pour activer la pompe ?

○ read_coils, ○ read_holding_registers, ○ write_register, ○ read_discrete_inputs, ○ read_input_registers

-Quels arguments doivent-être fournis ?

☐ unit ☐ address ☐ value ☐ count

Conclusion :

On espère que le monde est sauvé. De cet exercice, on peut tirer des leçons sur l'interopérabilité. Sans la documentation, vous n'auriez pas pu résoudre le problème. Bien sûr, il fallait connaitre l'adresse IP de la passerelle Modbus et les numéros des équipements sur le bus et leurs registres associés, mais la documentation doit également nous donner le format de stockage des données dans les registres et les unités utilisés.