MINI PROJET MODBUS TCP

Compte-rendu

Alexis Brunet – Damien Briquet
INSA CVL – 4A MRI-SA

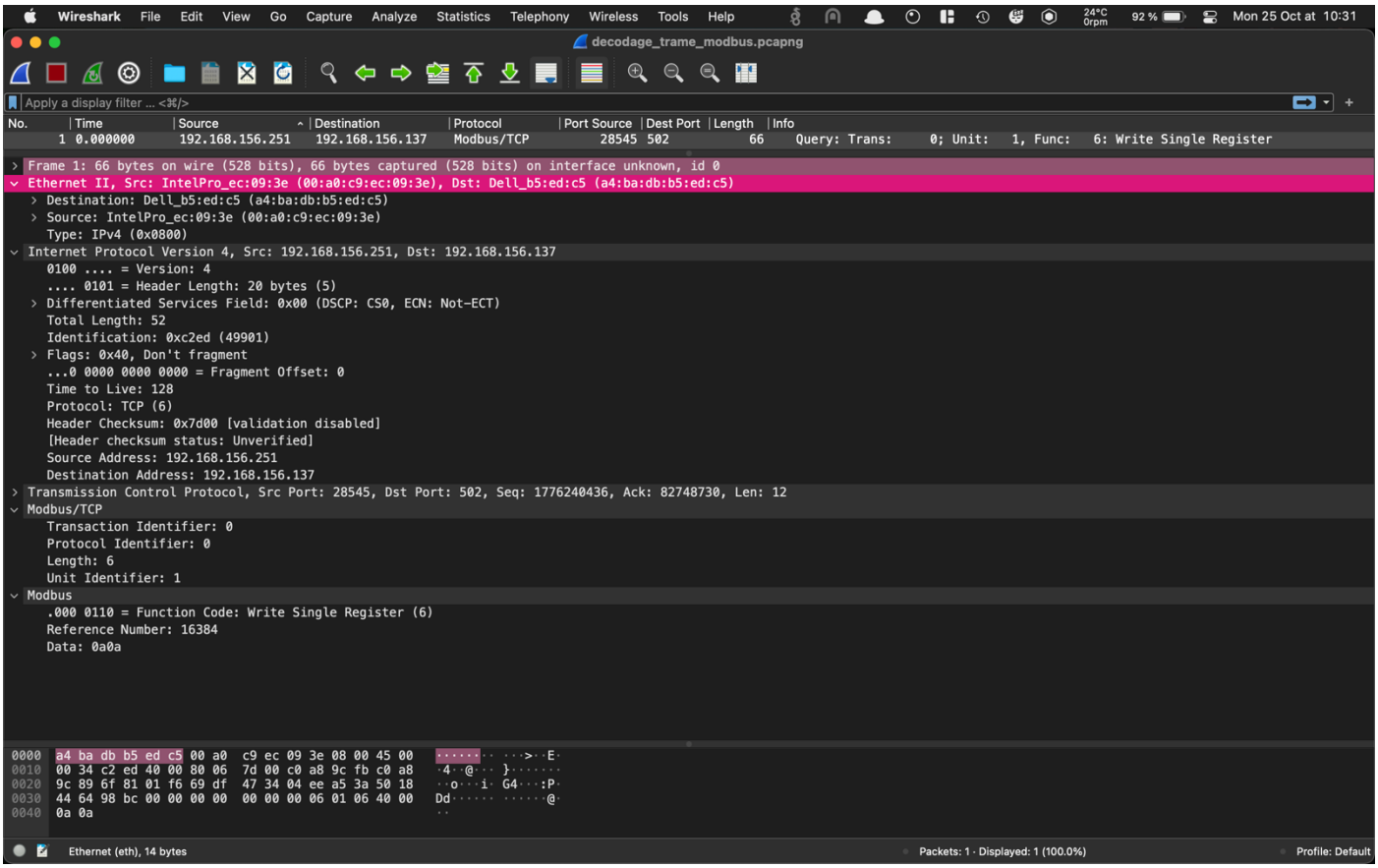
Sommaire

Introduction

Modbus est un protocole de communication de la couche 7 du modèle OSI (Application). Il y a 2 modes de Modbus : le RTU et TCP.
Modbus RTU fonctionne sur le mode maitre-esclave, ayant que le maitre actif, c'est-à-dire que c'est le seul qui peut lire et écrire dans chaque esclave qui sont eux passifs. Une trame ModBus RUT est sur 16 bits, contenant le numéro de l'esclave concerné, la fonction à traiter, la donnée et le code de vérification d'erreur (ou contrôle de redondance cyclique).
Le protocole ModBus TCP est l'un des protocoles Ethernet le plus utilisé. Modbus TCP fonctionne sur le mode client-serveur, ayant que les clients actifs, c'est-à-dire qu'ils peuvent se connecter au serveur ModBus, qui est passif, afin de lire et écrire dessus. La trame est de la même forme que le ModBus RTU. Le client par l'intermédiaire d'une trame requête, va demander des informations au serveur et en retour le serveur va envoyer à son tour une trame réponse pour lui donner les informations demandées. Sur les réseaux, les équipements sont identifiés de manière unique par leur adresse MAC et IP. Chaque carte réseau contient une adresse MAC (Medium Access Control)(de la forme de 6 nombres hexadécimaux séparés par le caractère « : » ou « - »). Cette adresse est utilisée dans les en-têtes de trame pour identifier la source et la destination sur un LAN (Local Area Network – Réseaux Local). Si le serveur ModBus n'est pas sur le même réseau que les clients dans ce cas nous allons utiliser les adresses IP, qui permet à un équipement d'être vue sur le réseau mondial et afin de s'affranchir du matériel (Adresse MAC). Chaque interface réseau se voit attribuer une adresse IP unique. Ces adresses sont des nombres sur 32 bits. Elles sont généralement affichées sous la forme de quatre nombres décimaux séparés par des points : ce format est appelé notation pointée.

Décodage trame ModBus TCP

Afin de comprendre la trame ModBus TCP, nous l'avons importer dans le logiciel Wireshark (fichier decodage_trame_modbus.pcapng dans le dossier).

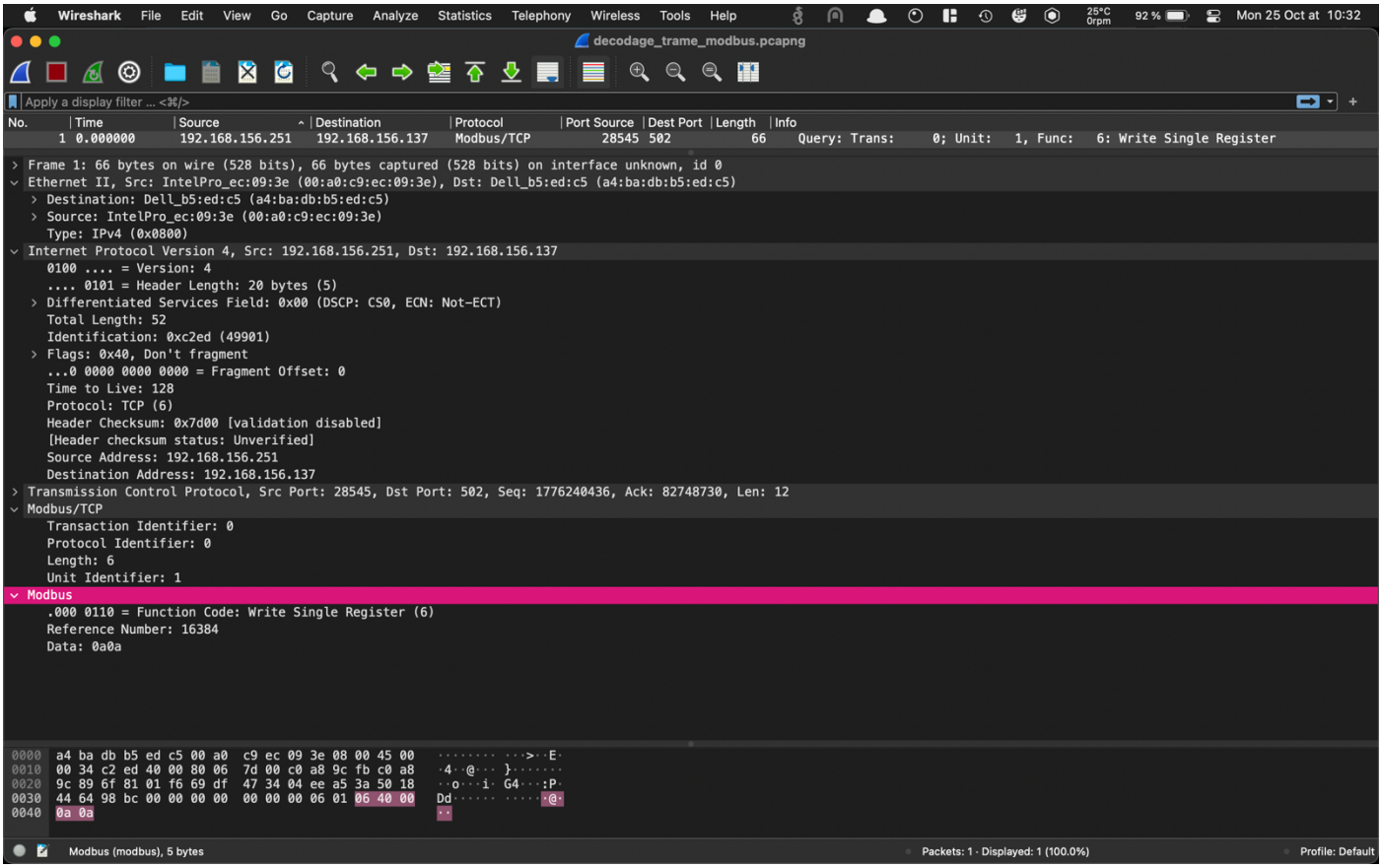


Nous avons pu en déduire les informations suivantes :

- Adresse IP Source : 192.168.156.251
- Adresse IP Destination : 192.168.156.137
- Port Source : 28545
- Port Destination : 502

À partir de ces informations nous pouvons constater que le serveur ModBus TCP correspond à la machine dont l'IP est 192.168.156.137 car le port utilisé sur cette machine est le 502 qui correspond à un serveur ModBus TCP. Donc en conséquent l'autre IP correspond à un client. De plus cette trame nous montre une requête du client vers le serveur car la source est le client.

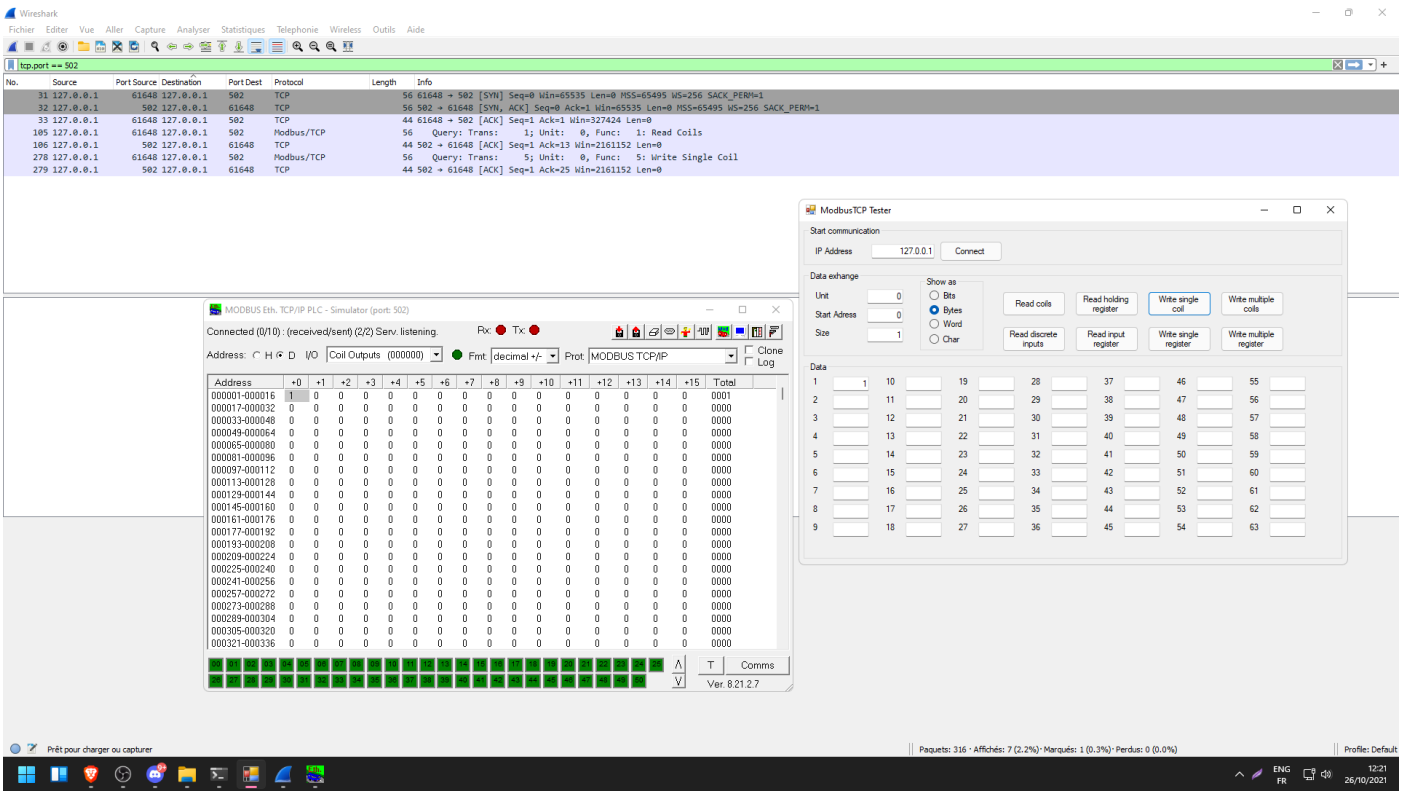
Maintenant nous cherchons à localiser la trame ModBus.



Nous remarquons que la trame ModBus correspond au 5 derniers hexadécimaux de toute la trame. Le 06 correspond à la fonction d'écrire dans le registre unique, avec les données 0a0a envoyé en hexadécimal.

Étude de trames ModBus TCP avec Wireshark

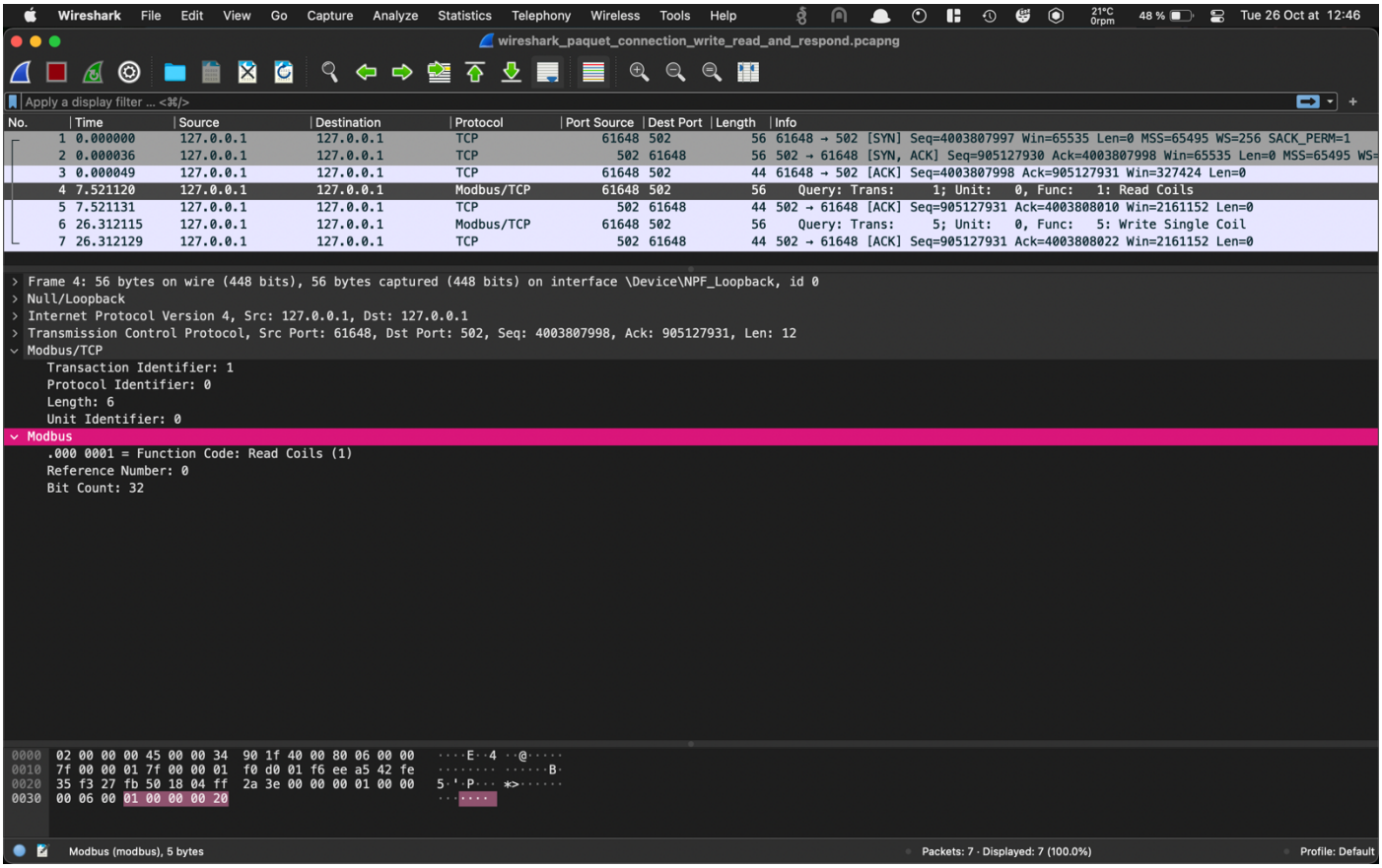
Après avoir trouvé et compris une trame ModBus, nous allons maintenant intercepter les trames, à l'aide de Wireshark, qui transitent entre un serveur et un client ModBus TCP fournis par notre enseignant. (Le fichier de capture est wireshark_paquet_connection_write_read_and_respond.pcapng)



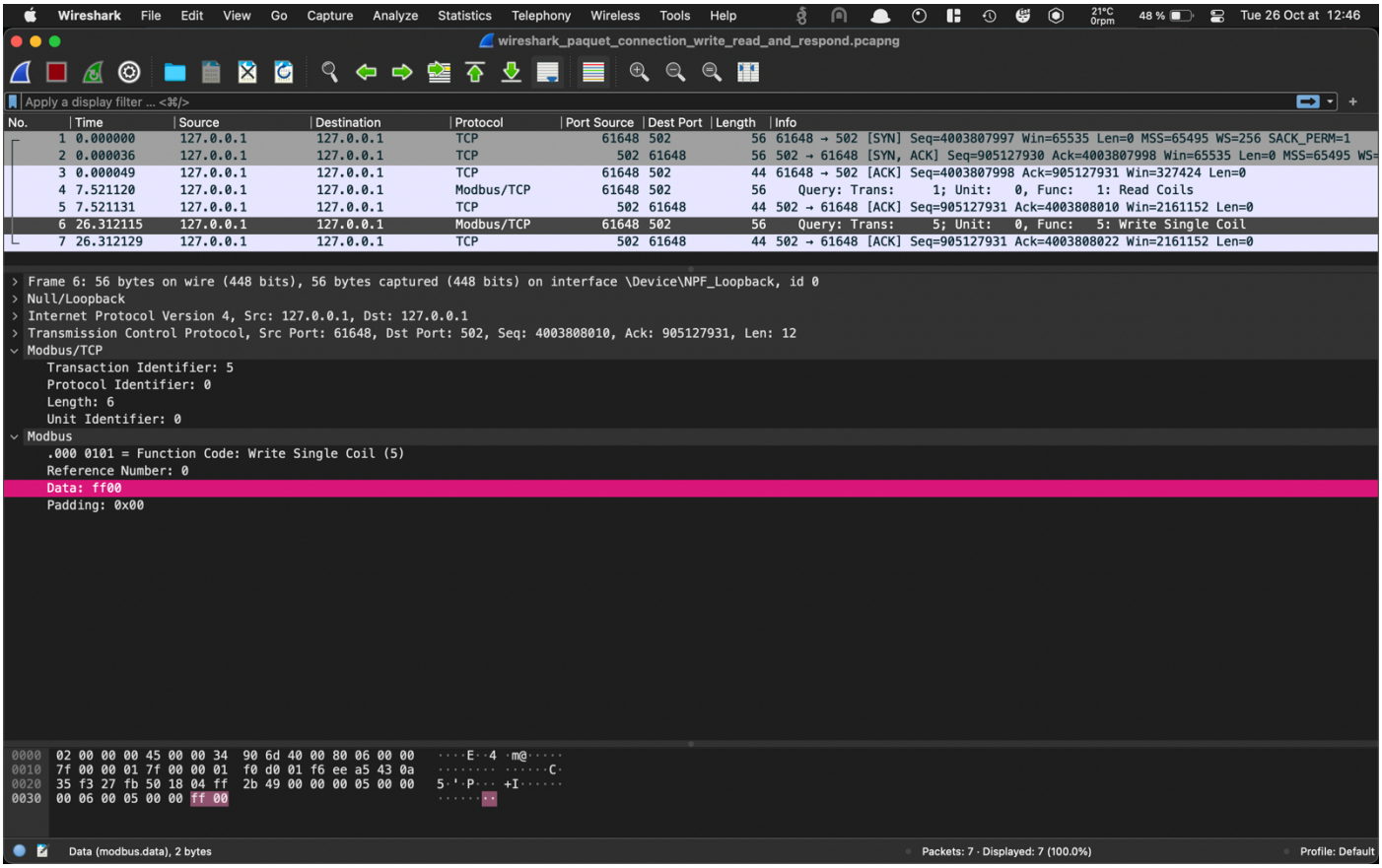
Les trois premières trames correspondent à la connexion du client au serveur avec le protocole TCP qui réalise sa connexion avec les trois poignées de main. Dans la première trame, le client envoie le flag de SYNC pour se synchroniser avec le serveur, dans la trame suivante il répond avec le flag SYNC souhaitant lui aussi se synchroniser avec le client et le flag ACK correspond à la confirmation de réception de son souhait de connexion avec lui (le serveur). Puis la dernière poignée de main correspondant à la troisième trame qui possède le flag ACK qui est l'accusé de réception de la synchronisation du serveur vers le client. Ainsi le client et le serveur peuvent communiquer tous les deux.

Les trames 4 et 6 correspondent aux requêtes faites par le client.

La trame 4 ne lit simplement que les données du Coil.



La trame 6 écrit sur le serveur ModBus, le bit 1 dans la case 0.

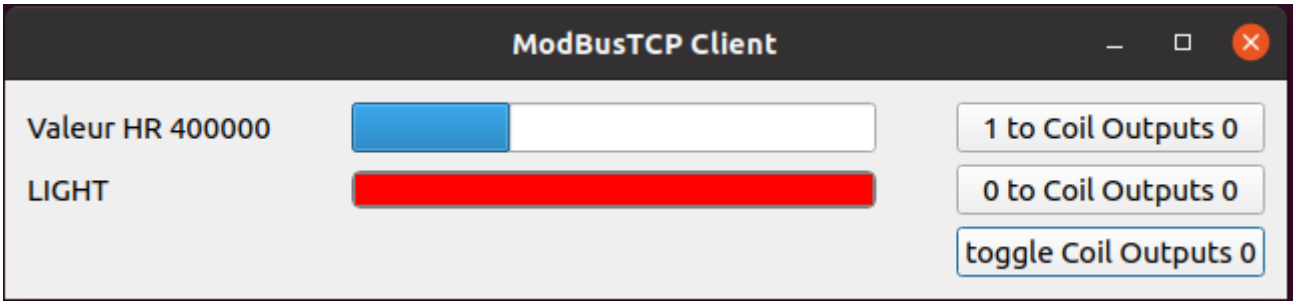


Et les trames 5 et 7 sont des accusés de réception des requêtes du client, comme on peut le voir grâce au flag ACK contenu dans la trame.

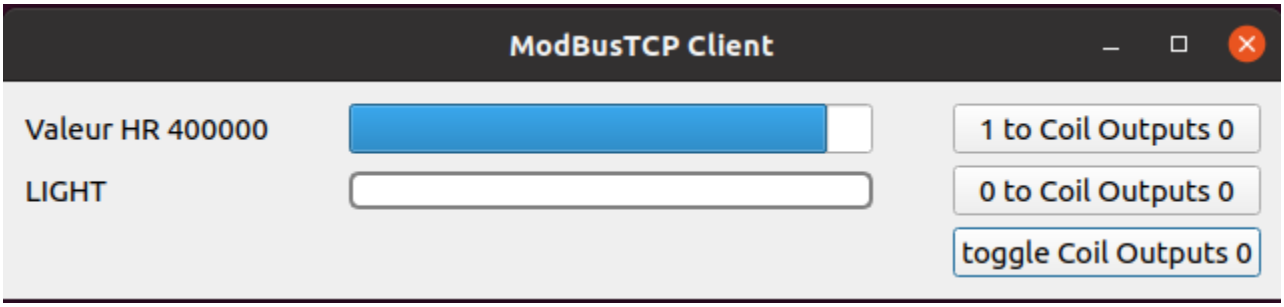
Client ModBus TCP

Après avoir compris les trames ModBus, nous pouvons maintenant développer un client ModBus afin d'interagir avec un serveur ModBus. Tout d'abord, le serveur ModBus que nous allons utiliser est le suivant : mod_RSsim.exe (https://sourceforge.net/projects/modrssim2/). Il nous est demandé de réaliser un client qui permet d'écrire sur une entrée, et de lire sur une sortie. Notre client est développé en Python car nous avons plus de compétence en Python qu'en VB.
Donc nous utilisons les bibliothèques pymodbus, pour la lecture et l'écrire sur le serveur Modbus, et PyQt5 pour l'interface graphique du client. (Le code de notre client se trouve dans le fichier projet.py et sur Github (voir les liens page 9))

Voici 2 captures d'écran de notre client dans 2 situations différentes :



Client avec lumière rouge active



Client avec lumière rouge désactivée

Sur notre interface, nous avons 1 jauge bleu, 1 voyant rouge et 3 boutons.
La jauge bleue correspond à la valeur rentrée dans la case HR 40001 sur 32767. La lumière rouge apparait dès lors que la valeur dans la case HR 40001 est exactement égale à 10000 (voir première capture écran). Ensuite 3 boutons sont sur le côté droit de l'interface :

- 1 to Coil Output 0 : affecte le bit 1 à la case 0 du Coil Output
- 0 to Coil Ouput 0 : affecte le bit 0 à la case 0 du Coil Output
- toglgle Coil Ouput 0 : passe le bit 1 en 0 et inversement automatiquement.

Liens

Vous pouvez aussi retrouver tout le projet sur Github avec des vidéos explicatives : https://github.com/AlexTheGeek/ModBusTCP_Client_Python.

Vidéo de présentation de notre client (installation et test) : https://video.lapinfo.fr/vs/sharing/J8BBYken#!aG9tZV92aWRlby00MjM=