



# MINI PROJET MODBUS TCP

Compte-rendu

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

# Sommaire

Introduction.....	2
Décodage trame ModBus TCP .....	3
Étude de trames ModBus TCP avec Wireshark.....	5
Client ModBus TCP .....	7
Liens .....	8

# Introduction

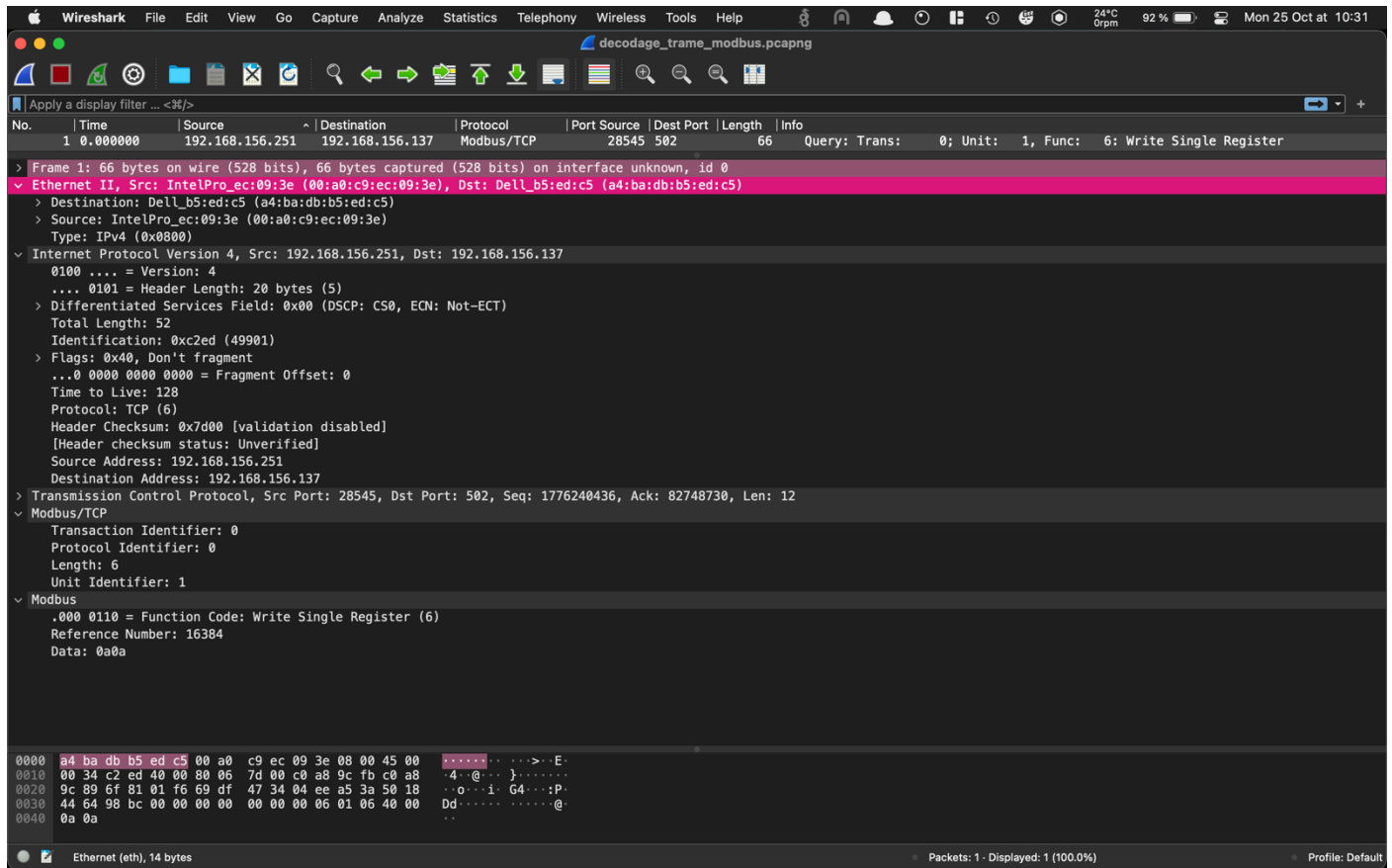
Modbus est un protocole de communication de la couche 7 du modèle OSI (Application). Il y a 2 modes de Modbus : le RTU et TCP.

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 RTU 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).

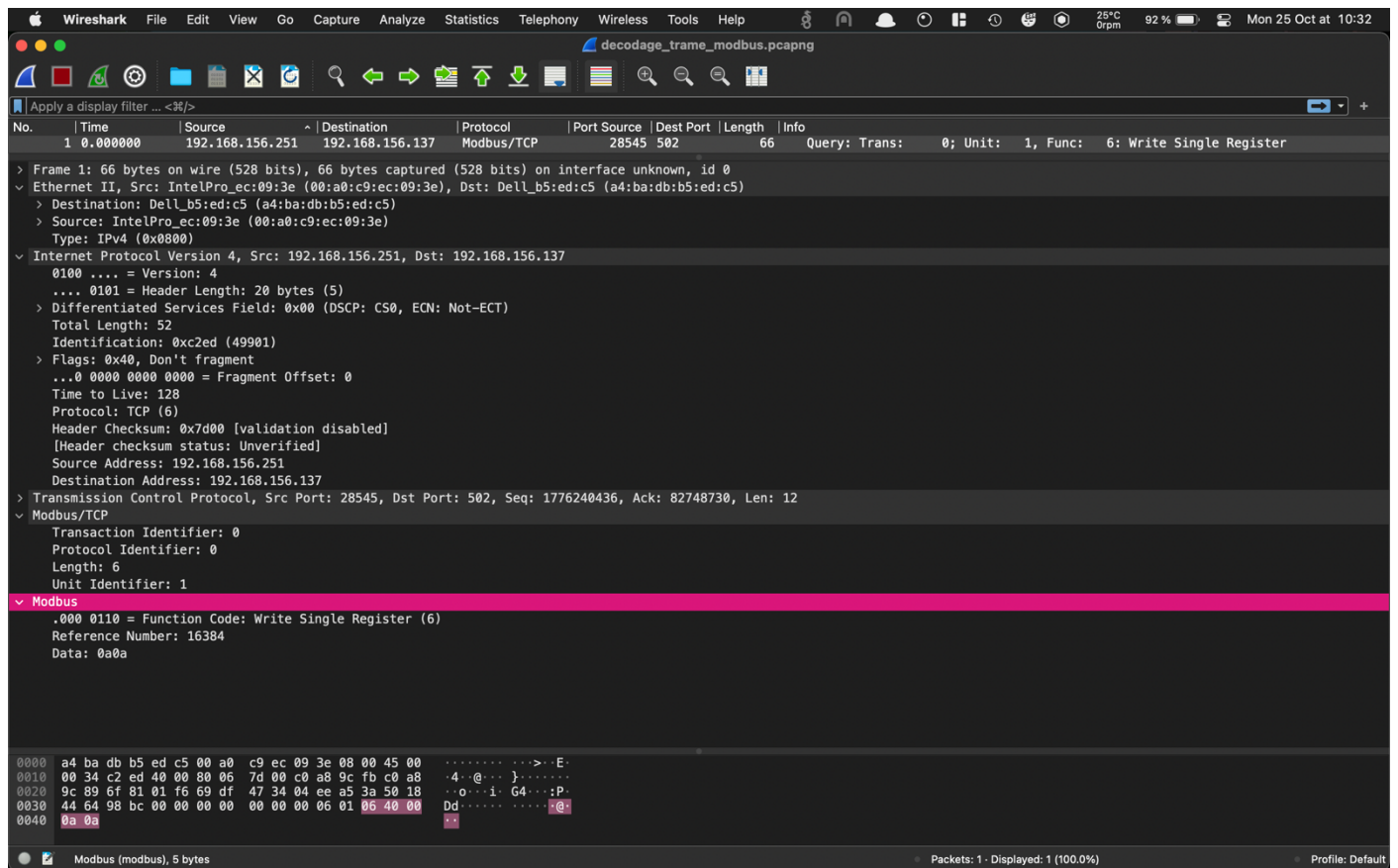


Nous avons pu en déduire les informations suivantes :

- 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)

The screenshot displays the Wireshark network protocol analyzer interface. The packet list on the left shows several TCP and Modbus/TCP packets. The main pane shows the details of a selected Modbus/TCP packet, including the query and response data. Overlaid on the bottom right is the 'ModbusTCP Tester' application window, which includes a 'Start communication' section with an IP address field set to 127.0.0.1 and a 'Connect' button. Below this is a 'Data exchange' section with various read and write operation buttons. The bottom part of the window shows a 'Data' table with 16 rows and 16 columns, representing a 16x16 matrix of coil outputs. The table is currently empty, with all cells showing '0'. The status bar at the bottom of the application indicates 'Paquets: 316 - Affichés: 7 (2.2%) - Marqués: 1 (0.3%) - Perdus: 0 (0.0%)' and 'Profil: Default'.

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15	Total
000001-000016	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0001
000017-000032	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000033-000048	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000049-000064	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000065-000080	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000081-000096	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000097-000112	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000113-000128	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000129-000144	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000145-000160	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000161-000176	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000177-000192	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000193-000208	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000209-000224	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000225-000240	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000241-000256	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000257-000272	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000273-000288	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000289-000304	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000305-000320	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
000321-000336	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000

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.

The screenshot shows a Wireshark capture of a network packet. The packet list at the top shows 7 packets. Packet 4 is selected, showing a Modbus/TCP query. The details pane shows the Modbus/TCP header with Transaction Identifier 1, and the Modbus data section showing a Read Coils (1) function code. The raw data pane shows the hexadecimal and ASCII representation of the packet data.

No.	Time	Source	Destination	Protocol	Port Source	Dest Port	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	61648	502	56	61648 → 502 [SYN] Seq=4003807997 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
2	0.000036	127.0.0.1	127.0.0.1	TCP	502	61648	56	502 → 61648 [SYN, ACK] Seq=905127930 Ack=4003807998 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
3	0.000049	127.0.0.1	127.0.0.1	TCP	61648	502	44	61648 → 502 [ACK] Seq=4003807998 Ack=905127931 Win=327424 Len=0
4	7.521120	127.0.0.1	127.0.0.1	Modbus/TCP	61648	502	56	Query: Trans: 1; Unit: 0, Func: 1: Read Coils
5	7.521131	127.0.0.1	127.0.0.1	TCP	502	61648	44	502 → 61648 [ACK] Seq=905127931 Ack=4003808010 Win=2161152 Len=0
6	26.312115	127.0.0.1	127.0.0.1	Modbus/TCP	61648	502	56	Query: Trans: 5; Unit: 0, Func: 5: Write Single Coil
7	26.312129	127.0.0.1	127.0.0.1	TCP	502	61648	44	502 → 61648 [ACK] Seq=905127931 Ack=4003808022 Win=2161152 Len=0

Frame 4: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF\_{...}\_Loopback, id 0

Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 61648, Dst Port: 502, Seq: 4003807998, Ack: 905127931, Len: 12

Modbus/TCP

Transaction Identifier: 1

Protocol Identifier: 0

Length: 6

Unit Identifier: 0

Modbus

.000 0001 = Function Code: Read Coils (1)

Reference Number: 0

Bit Count: 32

0000 02 00 00 00 45 00 00 34 90 1f 40 00 80 06 00 00 .....E..4..@.....

0010 7f 00 00 01 7f 00 00 01 f0 d0 01 f6 ee a5 42 fe .....B.....

0020 35 f3 27 fb 50 18 04 ff 2a 3e 00 00 00 01 00 00 5..P...>.....

0030 00 06 00 01 00 00 00 20 ..... ..

Modbus (modbus), 5 bytes

Packets: 7 · Displayed: 7 (100.0%)

Profile: Default

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

The screenshot shows a Wireshark capture of a network packet. The packet list at the top shows 7 packets. Packet 6 is selected, showing a Modbus/TCP query. The details pane shows the Modbus/TCP header with Transaction Identifier 5, and the Modbus data section showing a Write Single Coil (5) function code. The raw data pane shows the hexadecimal and ASCII representation of the packet data.

No.	Time	Source	Destination	Protocol	Port Source	Dest Port	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	61648	502	56	61648 → 502 [SYN] Seq=4003807997 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
2	0.000036	127.0.0.1	127.0.0.1	TCP	502	61648	56	502 → 61648 [SYN, ACK] Seq=905127930 Ack=4003807998 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
3	0.000049	127.0.0.1	127.0.0.1	TCP	61648	502	44	61648 → 502 [ACK] Seq=4003807998 Ack=905127931 Win=327424 Len=0
4	7.521120	127.0.0.1	127.0.0.1	Modbus/TCP	61648	502	56	Query: Trans: 1; Unit: 0, Func: 1: Read Coils
5	7.521131	127.0.0.1	127.0.0.1	TCP	502	61648	44	502 → 61648 [ACK] Seq=905127931 Ack=4003808010 Win=2161152 Len=0
6	26.312115	127.0.0.1	127.0.0.1	Modbus/TCP	61648	502	56	Query: Trans: 5; Unit: 0, Func: 5: Write Single Coil
7	26.312129	127.0.0.1	127.0.0.1	TCP	502	61648	44	502 → 61648 [ACK] Seq=905127931 Ack=4003808022 Win=2161152 Len=0

Frame 6: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF\_{...}\_Loopback, id 0

Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 61648, Dst Port: 502, Seq: 4003808010, Ack: 905127931, Len: 12

Modbus/TCP

Transaction Identifier: 5

Protocol Identifier: 0

Length: 6

Unit Identifier: 0

Modbus

.000 0101 = Function Code: Write Single Coil (5)

Reference Number: 0

Data: ff00

Padding: 0x00

0000 02 00 00 00 45 00 00 34 90 6d 40 00 80 06 00 00 .....E..4..m@.....

0010 7f 00 00 01 7f 00 00 01 f0 d0 01 f6 ee a5 43 0a .....C.....

0020 35 f3 27 fb 50 18 04 ff 2b 49 00 00 00 05 00 00 5..P...+I.....

0030 00 06 00 05 00 00 ff 00 ..... ..

Data (modbus.data), 2 bytes

Packets: 7 · Displayed: 7 (100.0%)

Profile: Default

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.

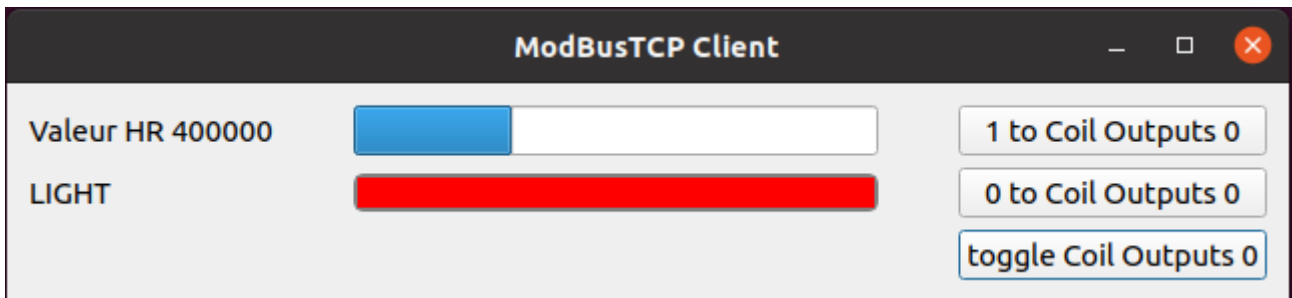


## Client ModBus TCP

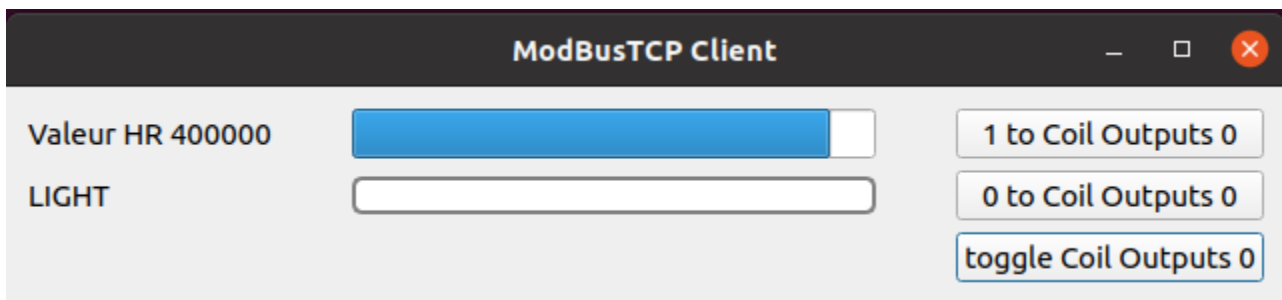
Après avoir compris les trames ModBus, nous pouvons maintenant développer un client ModBus afin d'interagir avec un serveur ModBus. Tout d'abord, le serveur ModBus que nous allons utiliser est le suivant : mod\_RSsim.exe (<https://sourceforge.net/projects/modrssi2/>). Il nous est demandé de réaliser 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'écriture 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 apparaît 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 Output 0 : affecte le bit 0 à la case 0 du Coil Output
- toggle Coil Output 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](https://github.com/AlexTheGeek/ModBusTCP_Client_Python).

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