OPC UA, un protocole sécurisé pour l'automatisme industriel

Mise en œuvre d'un serveur OPC UA sécurisé et de sa supervision

Culture Sciences de l'Ingénieur

La Revue 3E.I

Édité le 05/06/2024

école normale supérieure paris-saclay

Louis LALAY[1] – Anthony JUTON[2]

[1] *Étudiant agrégé de sciences de l'ingénieur, doctorant à Télécom Paris*

[2] *Professeur agrégé, ENS Paris Saclay, DER Nikola Tesla*

Cette ressource fait partie du N°112 de La Revue 3EI de mai 2024.

L'objectif de cette ressource est de présenter une application pratique de OPC UA, utilisant un nano-ordinateur raspberry Pi 4 bon marché (90 euros), pour simuler un système de château d'eau dialoguant en OPC UA avec le superviseur. Ce dispositif facile à dupliquer peut alors être un support pour des travaux pratiques autour du protocole OPC UA (supervision, étude des mécanismes de sécurisation du protocole). L'objectif étant l'étude d'OPC UA en tant que protocole sécurisé d'automatisme industriel, nous avons limité au maximum le besoin de connaissance de Linux et de python.

En cas de difficulté ou d'éléments peu clairs dans la ressource, il est possible d'échanger sur la liste https://groupes.renater.fr/sympa/info/revue3ei pour résoudre les problèmes et améliorer la ressource.

Pour travailler sur une seule machine (ce qui nous semble moins pédagogique), il est possible de remplacer le nano-ordinateur au choix :

- Par une machine virtuelle (de préférence Linux pour pouvoir utiliser OpenSSL),
- En exécutant les scripts python directement sur le système d'exploitation Windows où fonctionne Panorama

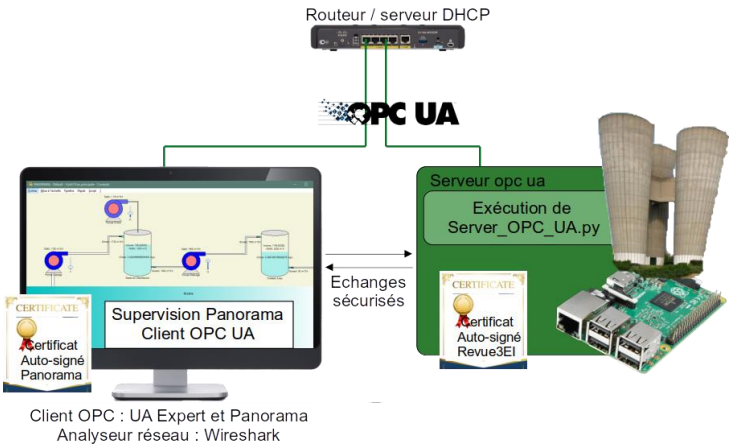

Figure 1 : Schéma synoptique de l'installation simulée présentée dans cette ressource

La ressource commence par une rapide présentation d'OPC UA et de son fonctionnement pour pouvoir aborder ensuite la mise en œuvre dans le cadre de l'activité pratique présentée. Cette présentation peut être avantageusement complétée par les vidéos d'Hervé Discours [6]. Le lecteur pourra ensuite approfondir avec les supports mis à disposition par OPC Uacademic [2].

1 - Présentation de OPC UA

1.1 - Histoire de la fondation [1]

Dans les années 90, Microsoft COM et DCOM[1] dominent la communication industrielle. En 1995, Fisher-Rosemount, Intellution, Opto 22 et Rockwell Software s'associent pour créer un standard de communication basé sur COM et DCOM nommé OPC, raccourcis de OLE[2] for Process Control. La fondation OPC (opcfoundation.org) est officiellement créée en 1996. Fin 1996, OPC-DA[3], une version simplifiée des spécifications OPC voit le jour : c'est la version qui a rendu OPC populaire. Cette version se base sur des protocoles Microsoft, et n'est donc supportée que par Windows.

En 2006, OPC UA[4] est créé afin de devenir indépendant de la plateforme hôte. Cette version est donc compatible avec plus de services. L'ancienne version est alors appelée OPC Classic.

Depuis, la fondation a surtout mis à jour ces deux protocoles et créé des liens avec différents industriels. En effet, un organisme peut intégrer la fondation pour faciliter la mise en œuvre des spécifications OPC à son propre matériel. La fondation OPC compte actuellement 850 membres, dont tous les acteurs importants de l'automatisme industriel (Siemens, Rockwell Automation, Wago, B&R), de la supervision (Arc Informatique, Codra, …), de la robotique industrielle (Fanuc, Staubli, Omron, …), de la production d'électricité (ABB, Alstom, General Electric, Schneider Electric, …), et même des fabricants de composants (Microchip, ST Microelectronics, ...)

Aujourd'hui, OPC signifie Open Plateform Communications. Dans la suite, on s'intéressera uniquement à OPC UA, dans un contexte multiplateforme et sécurisé.

1.2 - Domaines d'utilisation principaux

Le protocole OPC UA est surtout utilisé en automatisme, pour la communication entre un superviseur et des automates, ou pour la communication des automates entre eux. OPC UA permet la communication sécurisée de données, avec une multitude de services utiles à la supervision d'installations industrielles, par exemple :

- Alarmes,
- Horodatage,
- Historisation,
- etc.

Intérêt majeur de OPC UA par rapport aux protocoles de supervision traditionnels (Modbus TCP, BACnet Profinet, …), la communication se fait au travers de sessions ouvertes entre des clients et des serveurs et sécurisées par le populaire protocole SSL. Le client (un logiciel de supervision par exemple) envoie des requêtes aux serveurs (des automates par exemple) pour obtenir des informations du processus en cours et/ou pour donner des consignes afin de modifier ce processus.

---

[1] Distributed Component Object Model
[2] Microsoft Object Linking & Embedding
[3] OPC for Data Acces
[4] OPC Unified Architecture

1.3 - Rôle de la fondation OPC

Figure 2 : Logo de la fondation OPC

La fondation OPC est en charge du développement et du maintien des spécifications du protocole. C'est ensuite aux constructeurs de mettre en œuvre ces spécifications pour créer un serveur OPC UA ou un client OPC UA. Par exemple, Siemens implémente un serveur OPC UA sur ses automates S7-1200. De la même manière, Codra, entreprise développant le logiciel de supervision Panorama, implémente un client OPC UA sur Panorama, ce qui permet ainsi la supervision sécurisée des automates Siemens. Siemens et Codra faisant partie de la fondation OPC, leurs implémentations du protocole est validée par la fondation.

2 - Fonctionnement de OPC UA

Cette partie présente différentes possibilités offertes par OPC UA et leur fonctionnement.

2.1 - Session de communication

On s'intéresse ici à la structure de la communication OPC UA, le support utilisé, les différentes couches et leur rôle.

OPC UA utilise les couches TCP/IP (couches 4 et 3) sur des couches liaison de données / physique Ethernet ou Wifi (couches 2 et 1). Une fois la connexion TCP établie, une session OPC UA est ouverte au niveau de la couche 5 (voir Figure 3). Les interactions entre les clients et les serveurs requièrent un modèle à état. Les informations de l'état sont définies dans la session, qui est une connexion logique entre les clients et les serveurs. Les sessions sont indépendantes des protocoles de communication sous-jacents, et ne sont fermées que sur une requête de fermeture ou l'inactivité d'un client. Dans une session, on retrouve par exemple des détails sur les utilisateurs (nom d'utilisateur, mot de passe, autorisations, etc.), le mode de communication (Publisher/Subscriber ou Request/Response), etc.

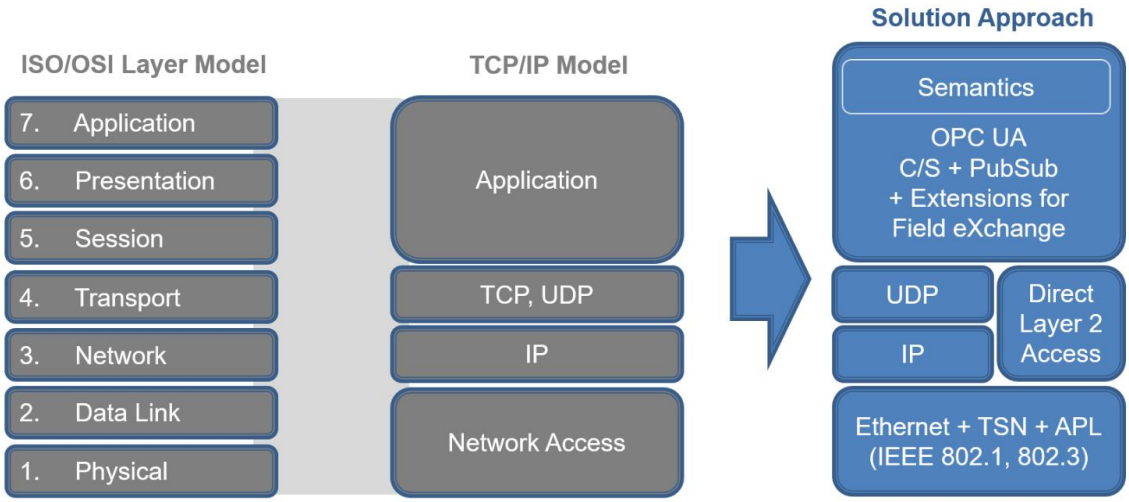

Figure 3 : Couches réseaux OPC UA (source OPC UAcademic)

2.2 - Sécurité

La sécurité est un des points forts de OPC UA, comparé aux anciens protocoles de supervision (Modbus par exemple). La sécurisation des communications en OPC UA s'appuie sur OpenSSL, comme HTTPs (cf ressource [13]).

Le service de sécurisation est indépendant du fonctionnement de l'application OPC UA, ce service de sécurisation reposant sur la *Communication Stack*. La *Communication Stack* communique via un Secure Channel. Un Secure Channel est une connexion logique longue durée entre un client et un serveur. Il est mis en place par un échange de certificats X.509 (client et serveur) et de clés publiques (clé client et clé serveur) aboutissant à un échange de clés de chiffrement symétrique permettant le chiffrement des échanges suivants, jusqu'à la fermeture du Secure Channel. Une application OPC UA ignore tout message qui ne suit pas la politique de sécurité. Une session est associée à un unique *Secure Channel*. Lors d'une connexion, le *Secure Channel* est d'abord ouvert, puis la session est ensuite ouverte en utilisant ce canal sécurisé, comme cela sera présenté dans la partie pratique un peu plus loin dans cette ressource.

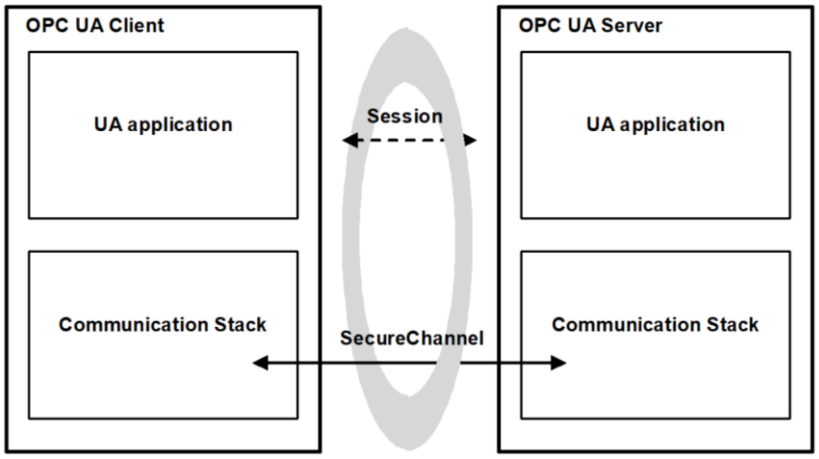

Figure 4 : Sécurité (source opcfoundation.org)

Pour les échanges de données via le *Secure Channel*, il existe 3 niveaux de sécurité :

- Aucun
- Avec chiffrement
- Avec chiffrement et authentification

2.3 - Discovery

Le service *Discovery* est décrit en détail dans la Partie 12.4 de la référence OPC UA. Ce service permet aux applications OPC UA de rechercher d'autres applications. En général, ce sont les serveurs qui proposent ce service afin que des clients s'y connectent, mais certains clients peuvent avoir une connexion inversée.

Avant même de créer une session, le service de *Discovery* permet d'identifier un serveur. Cela permet aussi d'avoir une vue d'ensemble de l'application, avec par exemple les variables qu'elle propose. Les variables étant désignées par un nom explicite, le développeur évite ainsi les erreurs d'adressage ou d'unité, typiques de modbus (on y désigne l'adresse de la variable et on récupère sa valeur brute, sans unité).

2.4 - Espace d'adresses

Une fois que la connexion est établie et sécurisée, il est possible d'accéder à l'espace d'adresses et de commencer à échanger des messages.

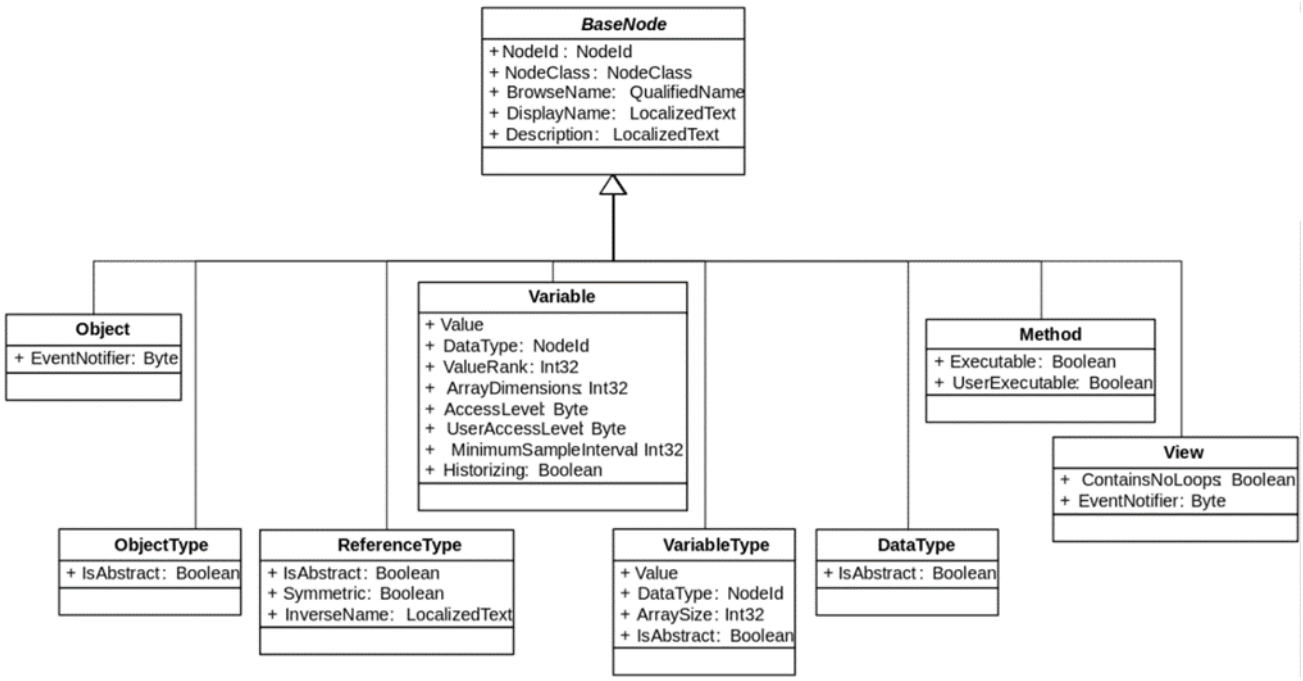

Figure 5 : Address Space (Source OPC UAcademic)

L'espace d'adresses définit comment est organisée une application OPC UA. C'est ici que se trouvent les variables par exemple. Chaque élément de l'espace d'adresse est un nœud, qui peut être une des 8 classes présentées sur la figure 5 ci-dessus et définies comme suit :

- **Object** : Utilisé pour représenter des systèmes, des composants, des objets réels, etc. Les objets sont liés entre eux par des références ;
- **ObjectType** : Définition pour les objets ;
- **ReferenceType** : Définition pour les références ;
- **Variable** : Stocke des données pour un objet ;
- **VariableType** : Définition pour les variables ;
- **DataType** : Type pour les données ;
- **Method** : Fonction ;
- **View** : Définit un sous ensemble de l'espace d'adresses. La vue par défaut est l'espace d'adresses en entier.

Un exemple d'espace d'adresses simple est proposé sur la figure 6. La racine de l'espace d'adresse est l'URL du serveur. On a ensuite (à la fin de la liste ici) un objet présentant les informations relatives au serveur (mode de communication, type de sécurité, etc.) puis les objets liés au process, avec des valeurs associées. L'espace d'adresses se construit à partir de l'installation réelle, en encapsulant les variables dans des objets. On peut ensuite encapsuler les objets dans d'autres objets plus grands. Les *View* permettent ensuite de sélectionner une partie de l'arborescence.

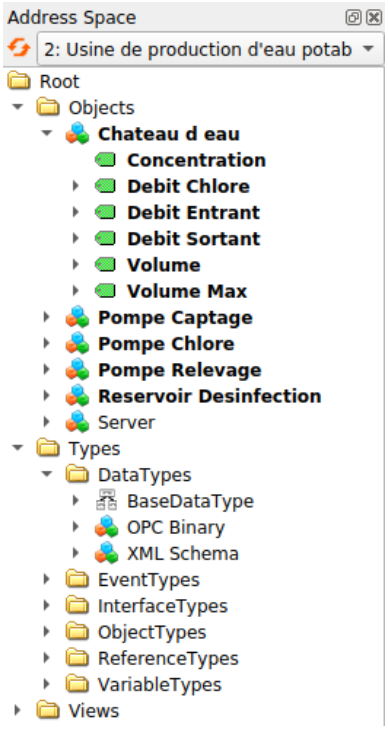

Figure 6 : Espace d'adresse de l'application château d'eau

2.5 - Modes de communication

Il existe deux modes de communication entre les applications. Les deux formes peuvent être adoptées et vont dépendre des cas d'utilisation. Dans les deux types de communication, les variables sont horodatées. La figure 7 présente les deux modes de communication.

Client/Server : Ce mode de communication est établi entre un client et un serveur. Lors de l'ouverture de la session, le serveur met en place un abonnement du client sur les variables qui l'intéressent. La session restant ouverte, le client n'est notifié que lorsque les valeurs ont changé. Le temps de rafraîchissement des valeurs côté serveur est réglable.

Publication/Subscription : Ce mode de communication met en relation un éditeur et des abonnés. Les applications n'échangent pas les messages directement, mais passent par un intermédiaire. Les *Publishers* envoient leurs données à l'intermédiaire sans savoir s'il y a des abonnés. De leur côté, les abonnés signalent à l'intermédiaire qu'ils sont intéressés par une donnée, sans savoir si la donnée est encore mise à jour. Lorsqu'une donnée est modifiée (et uniquement lorsqu'elle est modifiée), les abonnés sont notifiés du changement et peuvent récupérer la donnée à jour. Cela permet à plusieurs clients de s'abonner aux mêmes données sans ouvrir de session supplémentaire.

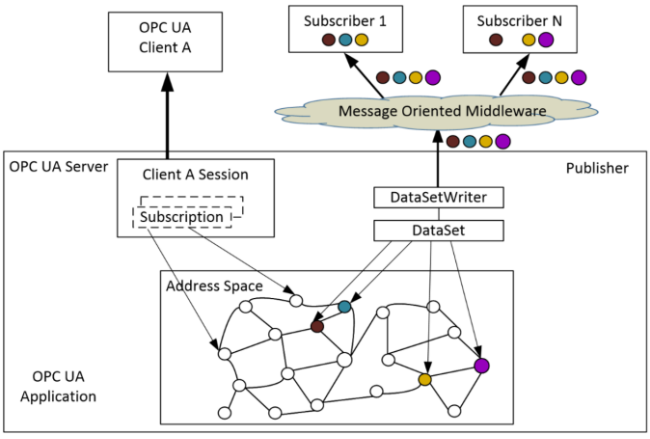

Figure 7 : Modes de communication (source Partie 1 6.6)

2.6 - Alarmes

La Partie 9 de la référence OPC UA explicite le fonctionnement des alarmes dans leur détail.

Les applications OPC UA permettent la création d'alarmes, et donc de simplifier la supervision de processus. Une alarme se déclenche par exemple lorsqu'une mesure sort d'une plage de valeur définie et alerte l'opérateur (via le logiciel de supervision).

L'application présentée ici en exemple gagnerait à utiliser les alarmes pour indiquer par exemple un débordement du château d'eau.

3 - Mise en œuvre du serveur OPC UA / château d'eau

Dans cette partie, on s'intéresse à la mise en œuvre d'un serveur OPC UA simulant un château d'eau, en 3 étapes.

- Démarrage d'un serveur OPC UA FreeOpcUa, qui implémente une grande partie des fonctionnalités OPC UA en langage Python (et en C++, inutilisé ici). Il n'est pas nécessaire de maîtriser python pour mettre en place ce qui suit. Le Client OPC UA est le logiciel gratuit UAExpert.
- Sécurisation de la connexion, toujours avec le serveur OPC UA FreeOpcUa et le client UAExpert
- Supervision avec le logiciel Panorama, client OPC UA, du serveur OPC UA simulant un château d'eau. Ce serveur utilise là encore FreeOpcUa.

3.1 - OPC UA avec Python

FreeOpcUa est un projet open source et ne fournit donc aucune garantie, mais il est très complet pour en faire un projet de démonstration sur un nano-ordinateur comme une carte Raspberry Pi. FreeOpcUa implémente les fonctionnalités essentielles de OPC UA, notamment la sécurisation des échanges.

3.2 - Installation du serveur

On configure un serveur OPC UA en langage Python. L'objectif est d'implémenter les fonctionnalités de base du serveur, afin de pouvoir observer les trames avec des outils tels que Wireshark, pour mettre en évidence les échanges, non sécurisés pour l'instant, via OPC UA.

Installation du système d'exploitation sur le nano-ordinateur

Pour installer le système d'exploitation Raspberry OS sur le nano-ordinateur raspberry Pi 4 et l'accès à distance, on peut suivre le document référencé ici [8]. On obtient ainsi un nano-ordinateur avec un système d'exploitation linux dont le bureau à distance est accessible via VNC Viewer. Il est aussi possible d'utiliser une machine virtuelle ou de faire tourner le serveur directement sur l'OS du client.

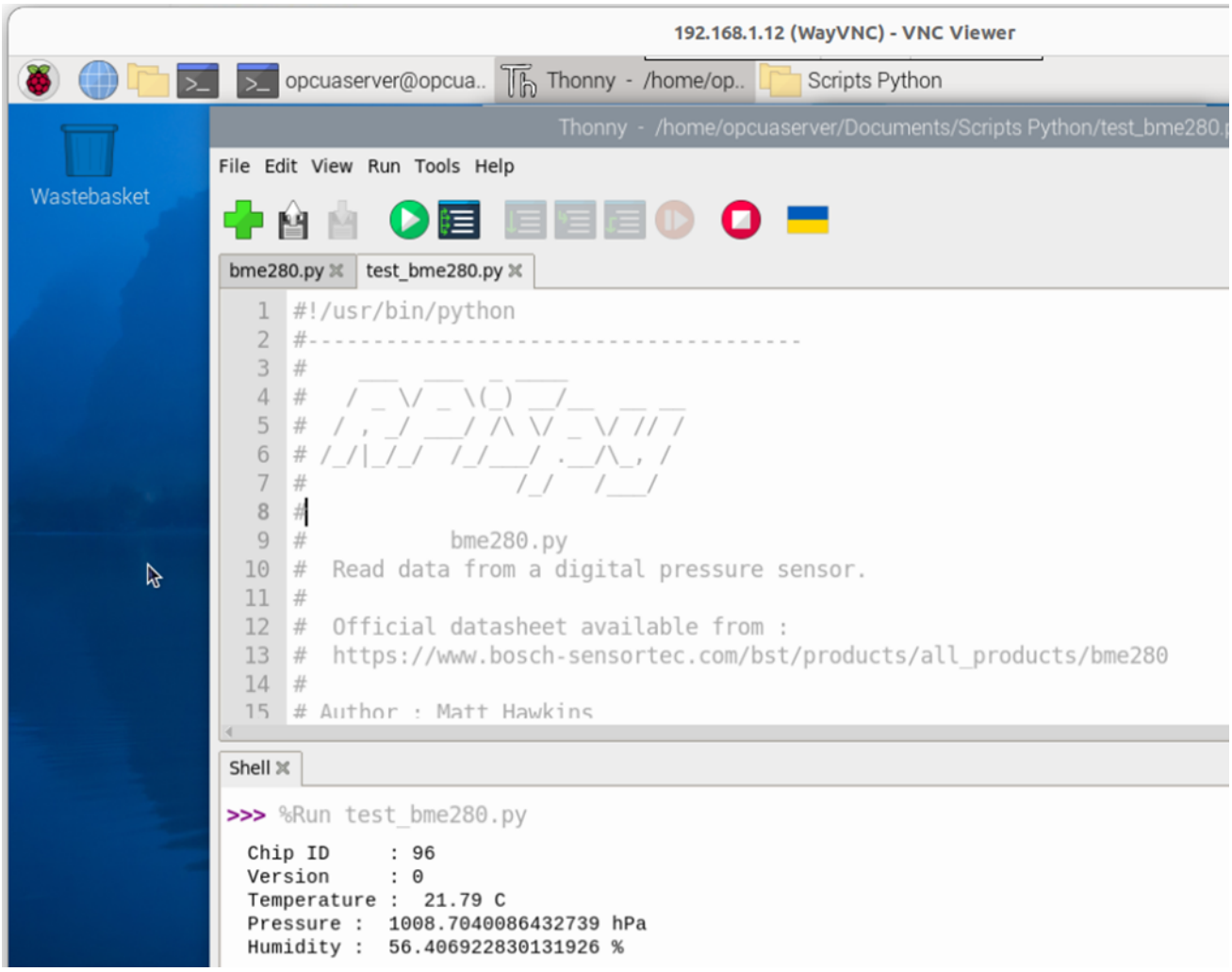

Figure 8 : Exécution d'un programme test_bme280.py via l'accès au bureau à distance du nano-ordinateur Raspberry Pi 4 par VNC Viewer

Installation du module serveur OPC UA

On installe tout d'abord le module python Free OPC-UA nommé **asyncua**. Le nano-ordinateur ne servant que de serveur OPC UA, on ne s'encombre pas du système d'environnements virtuels python, d'où l'option --break-system-packages.

```
pip install asyncua --break-system-packages
```

3.3 - Test simple du serveur OPC UA

Pour commencer, on teste sur une configuration simple :

- Pas de sécurité sur le serveur (prog *test_server_OPC_UA.py*)
- Juste une variable qui s'incrémente sur le serveur,
- Un client tout prêt : UA Expert,
- L'analyseur de réseau Wireshark pour observer les échanges.

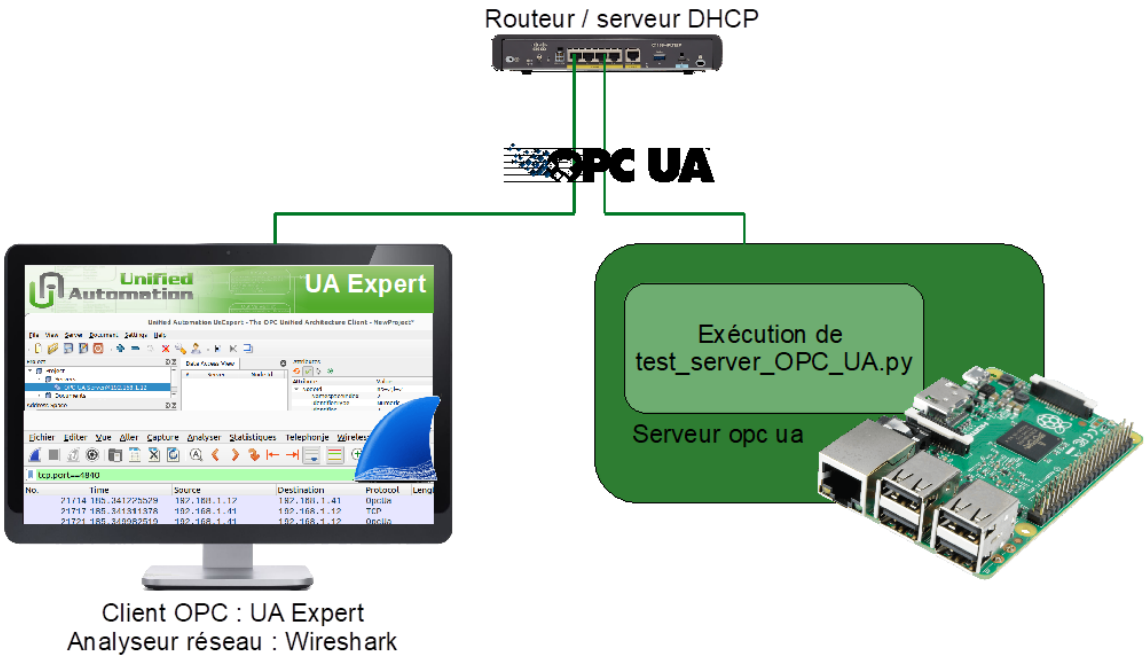

Figure 9 : Première application avec un serveur OPC UA simple, le client UA Expert et Wireshark

Côté serveur (Raspberry Pi 4 avec le module python asyncua)

Le programme du serveur *test_server_OPC_UA.py* est fourni avec cette ressource. Attention, l'adresse IP indiqué ligne 9 doit être l'adresse IP du serveur (donc de la raspberry Pi) :

```
#Programme de test du serveur OPC UA
from opcua import Server, ua
import time

def main():
    # Instanciation du serveur
    server = Server()
    # URL de la Raspberrypi qui hÃ©berge le serveur
    server.set_endpoint("opc.tcp://192.168.1.12:4840/UA/SampleServer")
    # nom du serveur
    server.set_server_name("OPC-UA-Server")
    # Sécurité (rien sur le programme de test)
    server.set_security_policy([ua.SecurityPolicyType.NoSecurity])
    # Nom de l'espace d'adresse
    name = "progDeTest_OPCUA"
    noeud_test = server.register_namespace(name)
    # Création d'un noeud racine
    node = server.get_objects_node()
    # Ajout d'un noeud pour les variables que l'on veut partager
    monObjet    = node.add_object(noeud_test, "monObjet")
    # Création de 2 variables, dont une accessible en écriture
    maVariable1 = monObjet.add_variable(noeud_test,"variableALire",0)
    maVariable2 = monObjet.add_variable(noeud_test,"variableAEcrire",0)
    maVariable2.set_writable()
    ancienneValeur2 = maVariable2.get_value()
    valeur1 = 3.14

    # Demarrage du serveur
    server.start()
    while True:
        # Lecture de la valeur de la variable accessible en écriture
        nouvelleValeur2 = maVariable2.get_value()
        if nouvelleValeur2 != ancienneValeur2 :
            ancienneValeur2 = nouvelleValeur2
            print("valeur recue : ",nouvelleValeur2)
        # Incrémentation de la valeur de maVariable1
        valeur1 = valeur1 + 0.1
        maVariable1.set_value(valeur1)
        time.sleep(1)

if __name__ == "__main__":
    main()
```

On exécute le programme sur la raspberry Pi, avec l'éditeur python Thonny par exemple :

Quelques lignes de commande permettent de vérifier le bon fonctionnement du serveur OPC, directement depuis sur un terminal du nano-ordinateur Raspberry Pi, avec bien sûr l'adresse IP correcte :

- uals --url=opc.tcp://192.168.1.12:4840
- uals --url=opc.tcp://192.168.1.12:4840 --nodeid i=85
- uaread --url=opc.tcp://192.168.1.12:4840 --nodeid "ns=2;i=2"
- uaread --url=opc.tcp://192.168.1.12:4840 --path "0:Objects,2:monObjet,2:variableALire"

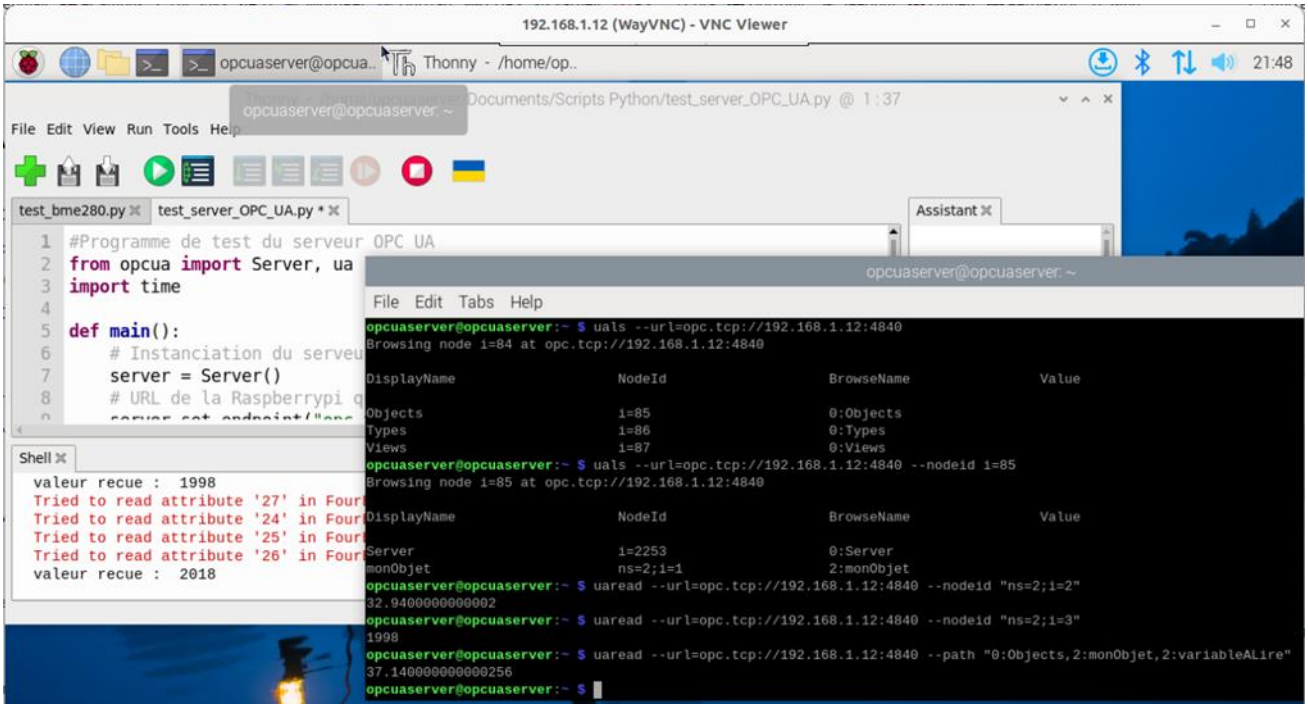

Figure 10 : Bureau du nano-ordinateur Raspberry Pi, serveur OPC UA avec l'IDE Thonny pour l'exécution du programme et la console pour les tests

Côté client (UA Expert sur le PC)

UA Expert [9] est un client OPC UA gratuit, robuste, permettant d'utiliser la plupart des fonctionnalités OPC UA.

Télécharger, installer et lancer UA Expert.

Ajouter le serveur :

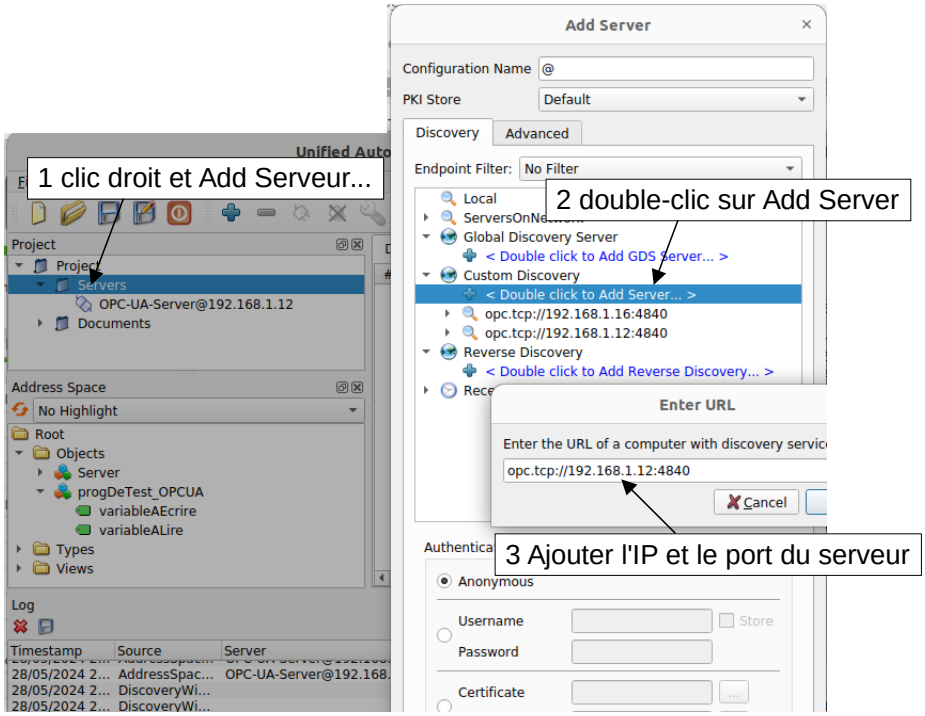

Figure 11 : Ajout d'un serveur sur le client UA Expert

Une fois le serveur dans la liste des Servers de la zone Project, un clic droit dessus permet de s'y connecter.

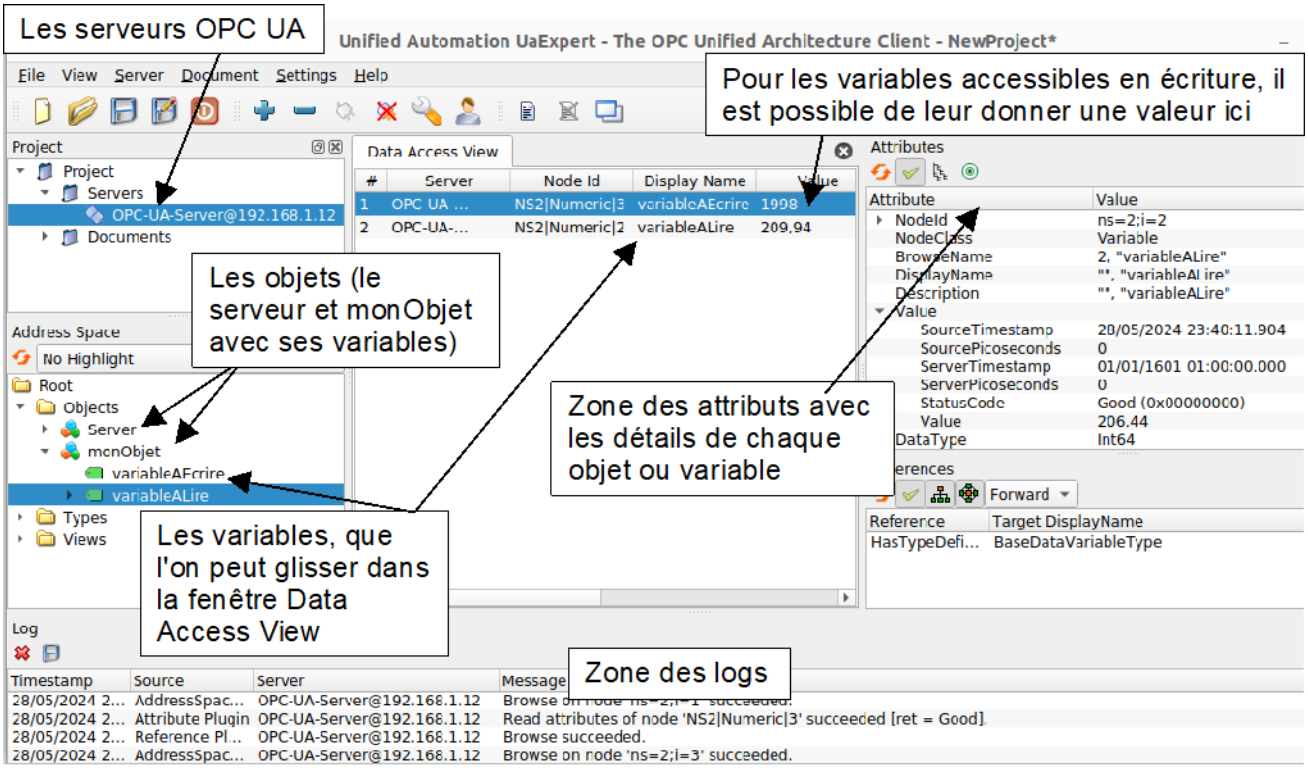

Figure 12 : Utilisation de UA Expert pour afficher et modifier les variables supervisées

Espionnage des échanges (Wireshark sur le PC)

Le logiciel analyseur de réseau Wireshark [10] permet d'observer les échanges entre le client et le serveur. Il est possible de l'installer sur le PC et/ou sur le nano-ordinateur raspberry Pi. L'espionnage de la mise en place de la connexion est intéressant :

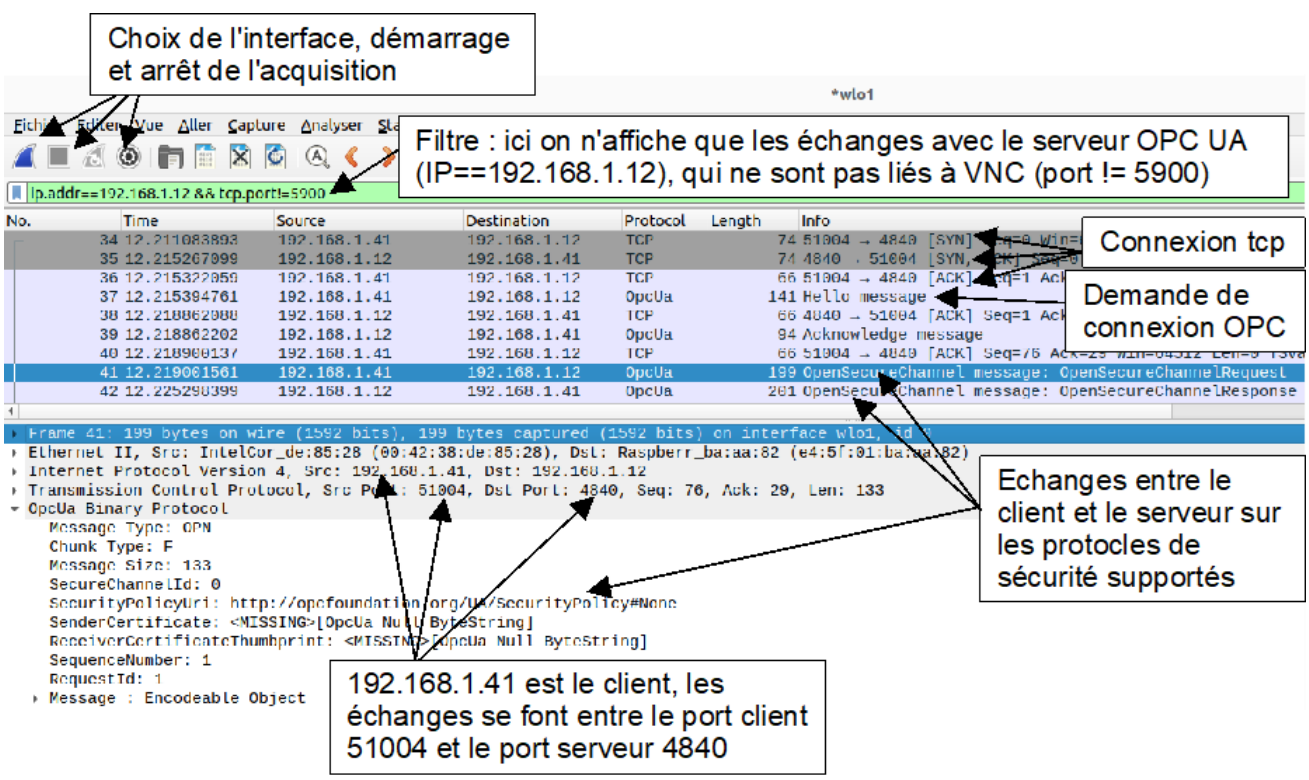

Figure 13 : Espionnage des échanges entre le client et le serveur OPC UA avec Wireshark, lors de la connexion

L'espionnage des échanges une fois la connexion établie permet de mettre en évidence le mode publisher/subscriber.

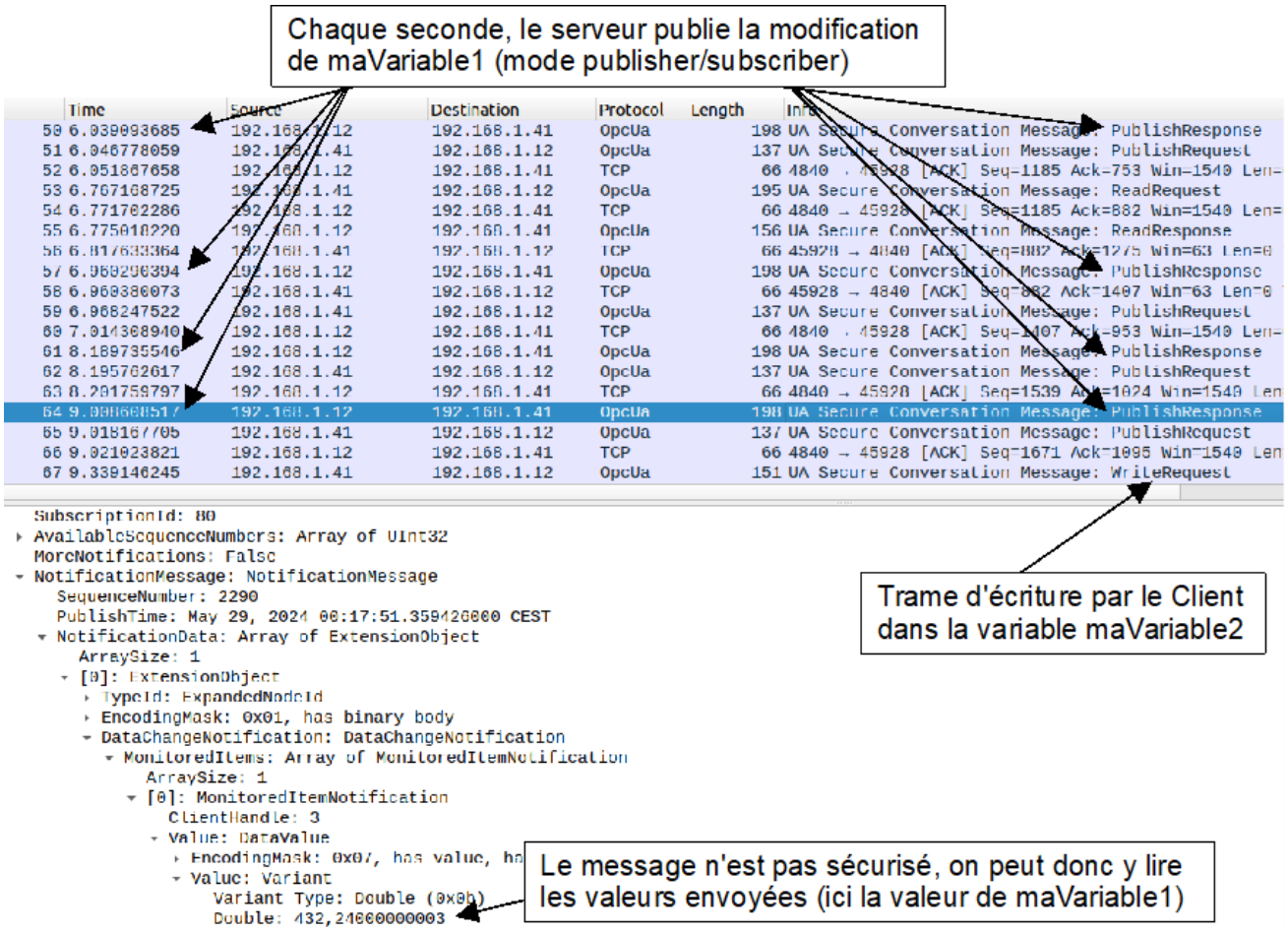

Figure 14 : Espionnage des échanges des valeurs maVariable1 (Serveur -> Client) et maVariable2 (Client->Serveur)

3.4 - Connexion sécurisée OPC UA

Dans ce 2ème exemple, on reprend l'exemple simple, en ajoutant la sécurisation de la connexion. On choisit le chiffrement des échanges et l'authentification (SignAndEncrypt).

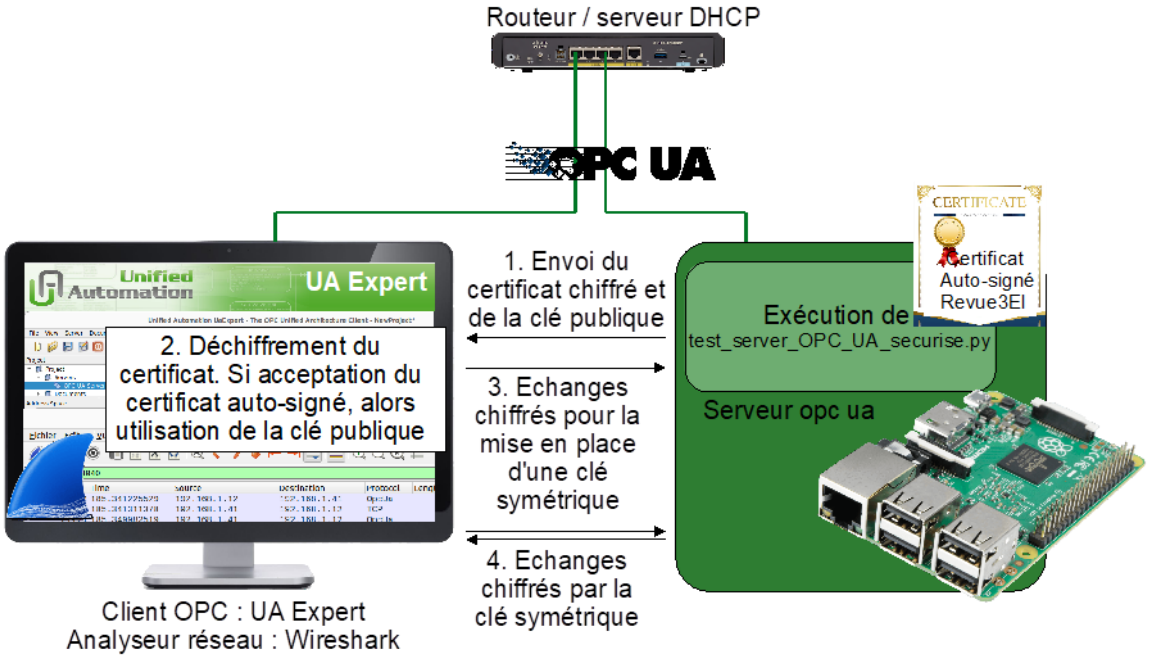

Figure 15 : Mise en place de la connexion OPC-UA sécurisée

La meilleure pratique est de privilégier l'utilisation d'une autorité de certification au lieu d'un certificat autosigné. Cependant, dans un environnement pédagogique, cela est peu accessible. L'authentification est donc basée ici sur des certificats X.509 auto-signés, associés à des clés asymétriques : le serveur envoie sa clé publique et un certificat chiffré par sa clé privée. Le client vérifie que la clé publique déchiffre bien le certificat. Il reste à approuver le certificat (une autre machine pourrait se faire passer pour le serveur), ce qui pour des certificats auto-signés se fait manuellement.

Génération des certificats

La génération de certificats auto-signés est gratuite, mais il faut approuver le certificat serveur manuellement, soit à la demande du client lors de la connexion, soit en ajoutant le certificat dans le dossier des certificats approuvés. Le serveur approuve le certificat du client (il répond à tous les clients).

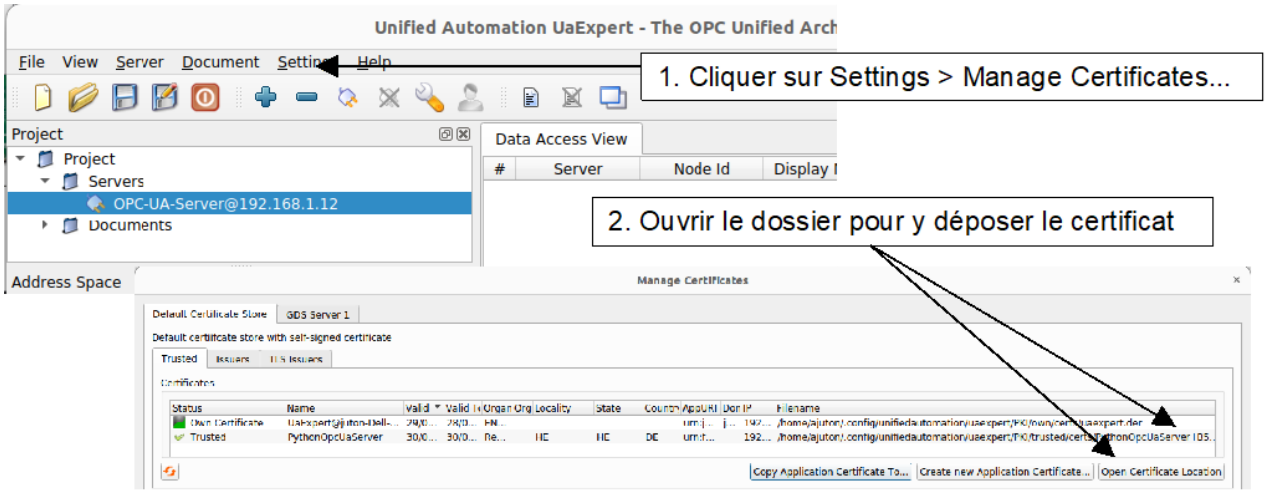

Figure 16 : Management des certificats par UAExpert, avec le dossier de dépôt des certificats

UA Expert crée lui-même le certificat client (*Own Certificate* sur la figure sous-dessous).

Pour le serveur, FreeOPCUA propose un générateur de certificats X.509 auto-signés, avec la configuration pré-remplie, à adapter quelque peu pour que le certificat soit conforme et accepté aussi bien par UAExpert que Panorama (pour la suite) :

On crée le fichier ssl.conf (fourni en pièce jointe à cette ressource). OpenSSL est installé de base sur linux.

```
[ req ]
default_bits = 2048
default_md = sha256
distinguished_name = subject
req_extensions = req_ext
x509_extensions = req_ext
string_mask = utf8only
prompt = no

[ req_ext ]
basicConstraints = CA:FALSE
nsCertType = client, server
keyUsage = nonRepudiation, digitalSignature, keyEncipherment, dataEncipherment, keyCertSign
extendedKeyUsage= serverAuth, clientAuth
nsComment = "OpenSSL Generated Certificat"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
subjectAltName = URI:urn:freeopcua:python:server, IP: 192.168.1.12

[ subject ]
countryName = FR
stateOrProvinceName = IDF
localityName = Saclay
organizationName = Revue3EI
commonName = PythonOpcUaServer
```


Figure 17 : Fichier ssl.conf pour la génération du certificat

On génère alors la clé avec OpenSSL :

```
openssl genrsa -out key2.pem 2048
```

On génère ensuite le certificat X.509, aux formats pem (ascii) et der (binaire) :

```
openssl req -x509 -days 365 -new -out certificate2.pem -key key2.pem -config ssl.conf
openssl x509 -outform der -in certificate2.pem -out certificate2.der
```

Linux permet d'afficher le certificat X.509 généré :

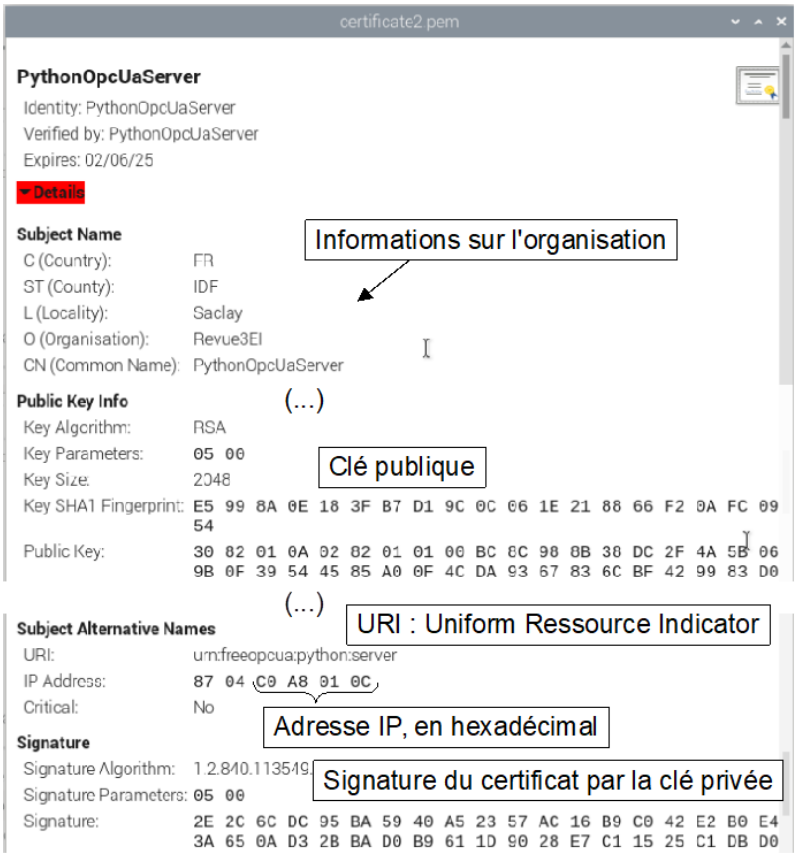

Figure 18 : Affichage du certificat généré

Démarrage du serveur

Une fois certificate2 et key2 copiés dans le dossier du fichier *test_server_OPC_UA_securise.py*, on lance le serveur, avec la sécurité activée :

test_server_OPC_UA_securise.py

```
4
5  def main():
6      # Instanciation du serveur
7      server = Server()
8      # URL de la Raspberrypi qui héberge le serveur
9      server.set_endpoint("opc.tcp://192.168.1.12:4840/UA/SampleServer")
10     # nom du serveur
11     server.set_server_name("OPC-UA-Server")
12     # Sécurité
13     server.set_security_policy([ua.SecurityPolicyType.Basic256Sha256_SignAndEncrypt])
14     server.load_certificate("certificate2.der")
15     server.load_private_key("key2.pem")
16     # Nom de l'espace d'adresse pour éviter les ambiguïtés de noms de noeuds
17     name = "progDeTest_OPCUA"
```

Figure 19 : Début du fichier test_server_OPC_UA_securise.py avec l'intégration du certificat et de la clé

Démarrage du client

Il est possible de se connecter au serveur. Celui-ci n'acceptant que les connexions sécurisées désormais, il propose son certificat au client, que celui-ci doit accepter. 2 possibilités avec UAExpert :

Soit on accepte manuellement le certificat au moment de la première connexion :

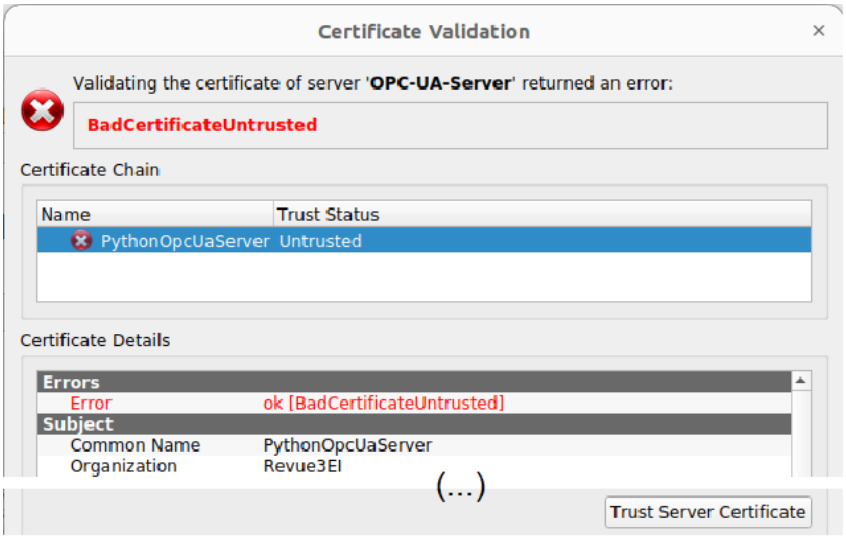

Figure 20 : Fenêtre UAExpert de demande l'acception du certificat

Il est aussi possible de copier à l'avance le certificat dans le dossier des certificats de confiance (voir **Erreur ! Source du renvoi introuvable.**).

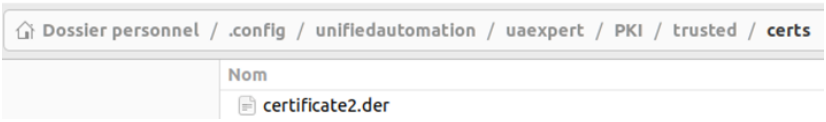

Figure 21 : Fichier certificat copié dans le dossier des certificats de confiance de UAExpert

La communication sécurisée est alors opérationnelle :

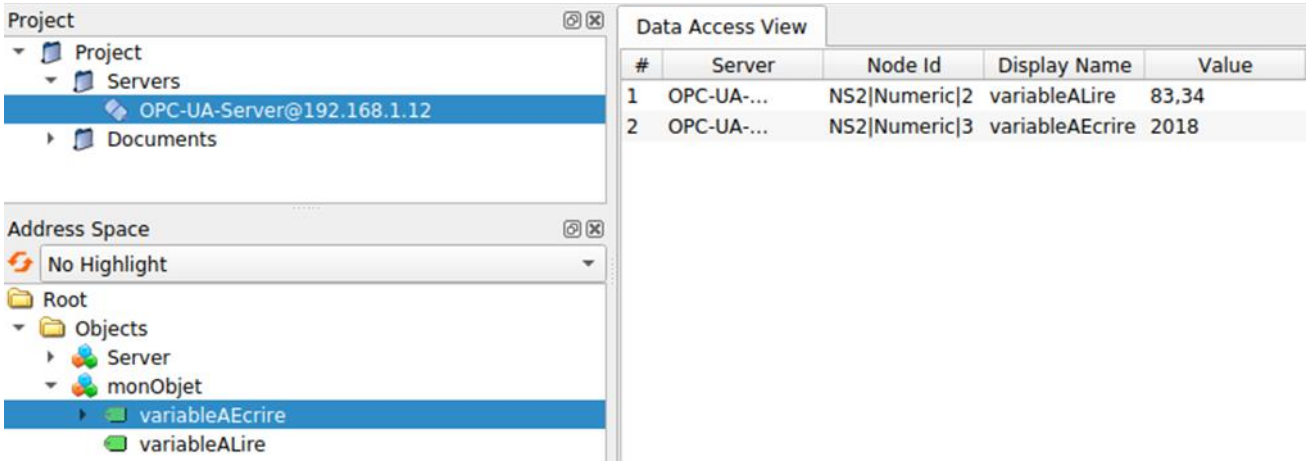

Figure 22 : Communication du client UAExpert avec serveur OPC UA

Observation des échanges OPC UA sécurisés par wireshark

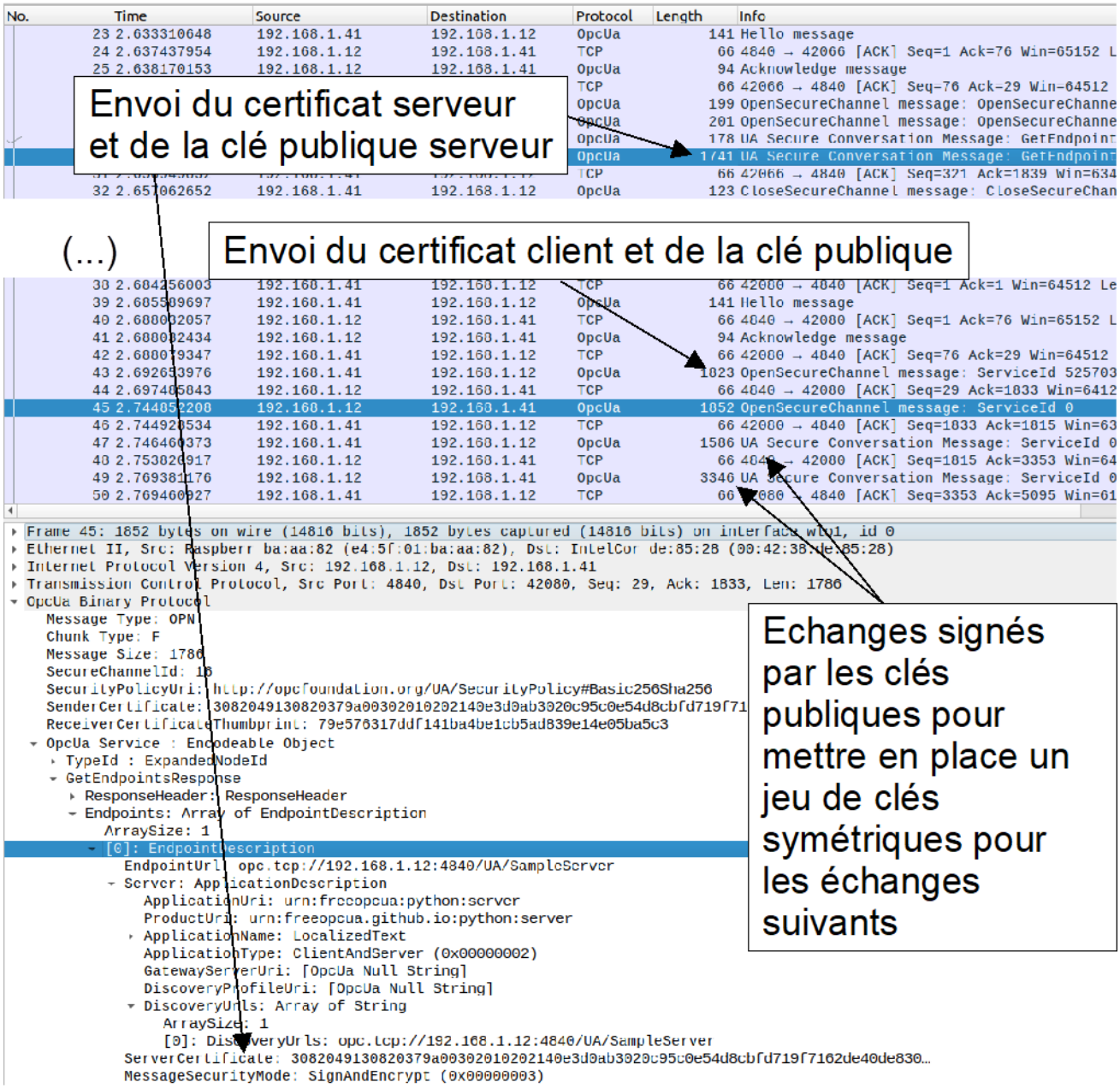

Figure 23 : Mise en place des échanges sécurisés lors de la connexion du client UA Expert au serveur OPC UA

3.5 - Supervision d'un château d'eau simulé via OPC UA

Les échanges supervisés étant en place entre UAExpert et le serveur OPC UA, il est possible de lancer le serveur OPC UA du château d'eau et le client OPC UA sur le superviseur Panorama.

Démarrage du serveur

Sur le serveur, l'ensemble des fichiers nécessaires est fourni en pièce jointe à cette ressource.

Figure 24 : Fichiers du dossier ScriptsPython, pour l'exécution du simulateur de château d'eau

Il faut lancer d'abord le fichier *simulation_usine_eau.py* qui gère le fonctionnement simulé du château d'eau. Ensuite, on exécute *server_OPC_UA.py* qui, comme son nom l'indique gère la mise à disposition des variables par OPC. Les échanges entre les 2 process (simulation_usine_eau et server_OPC_UA) se font par l'intermédiaire de 3 fichiers texte : *fichier_commande_OPC.txt*, *fichier_retour_pompes.txt* et *fichier_retour_reservoirs.txt*.

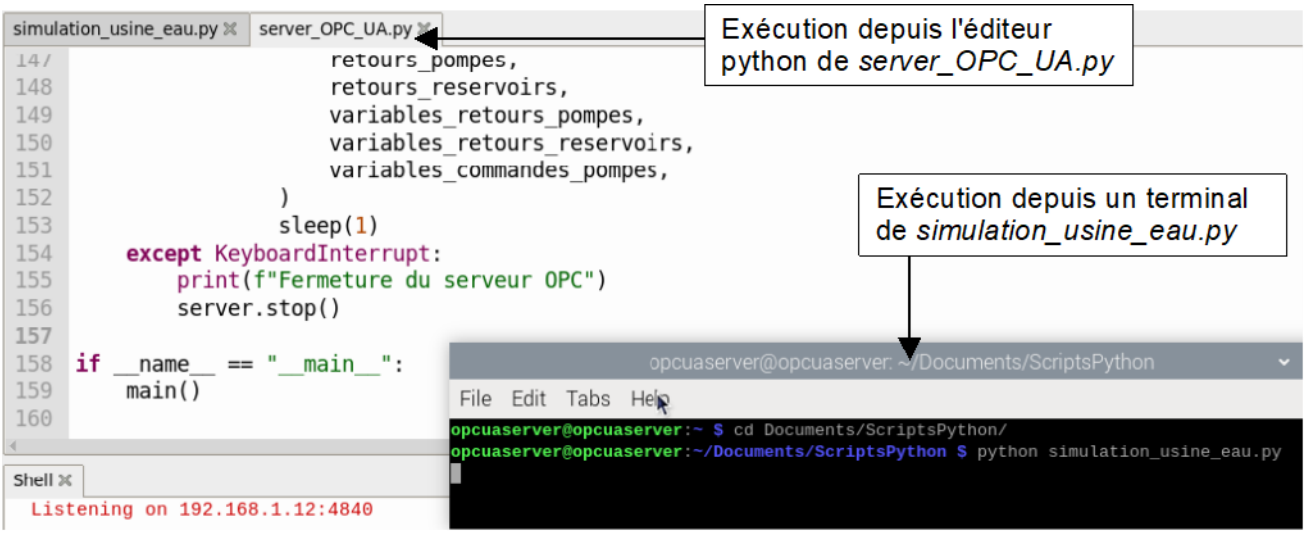

Figure 25 : Copie d'écran du serveur Raspberry Pi lors de la simulation du château d'eau

Vérification du bon fonctionnement du serveur depuis un client UAExpert

Depuis UAExpert, il est possible de se connecter au simulateur de château d'eau pour vérifier son bon fonctionnement.

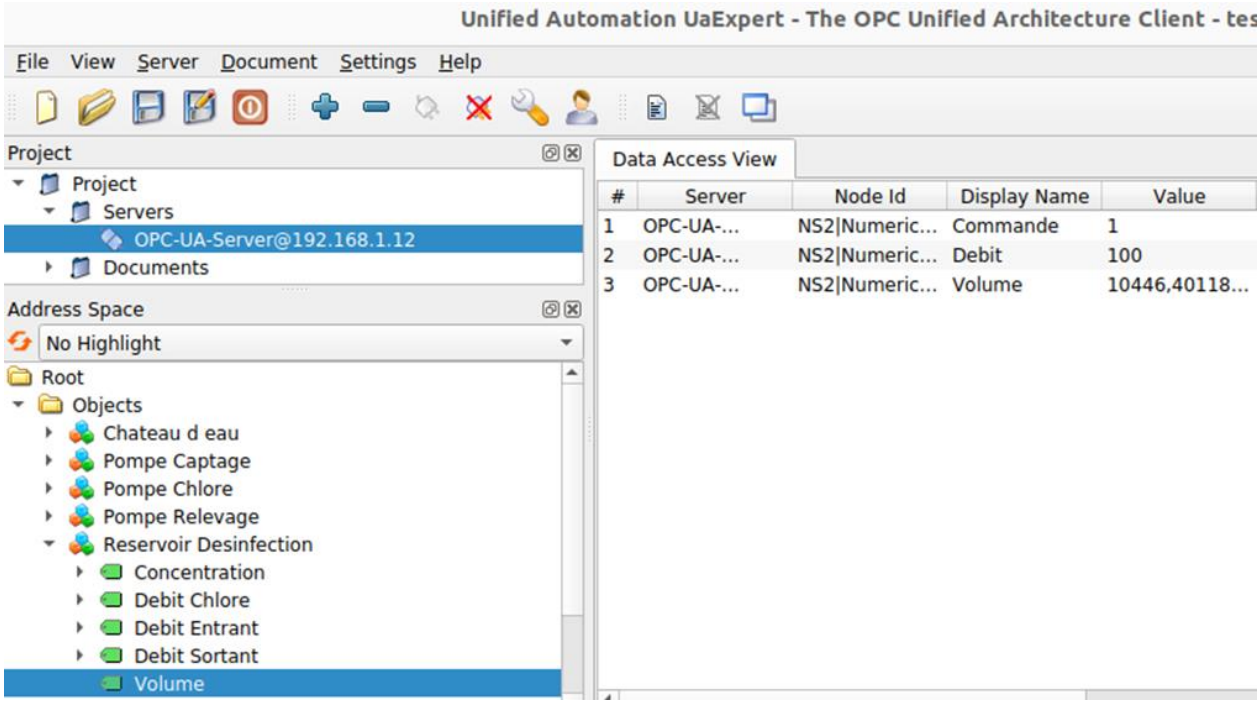

Figure 26 : Supervision des variables du simulateur de château d'eau depuis UAExpert

Cela permet de vérifier le bon fonctionnement mais aussi de mettre en évidence l'intérêt du service Discovery de OPC-UA, fournissant l'ensemble des variables accessibles.

Supervision par Panorama

Le populaire logiciel de supervision Panorama de Codra [3] inclut la possibilité de créer des clients et des serveurs OPC UA. Comme Codra fait partie de la fondation OPC, leur produit est régulièrement validé et mis aux normes. Le logiciel est disponible en version d'essai pour faire des projets de taille raisonnable pour l'enseignement (moins de 50 variables / 4h de connexion) [3]. Panorama ne fonctionne que sous Windows.

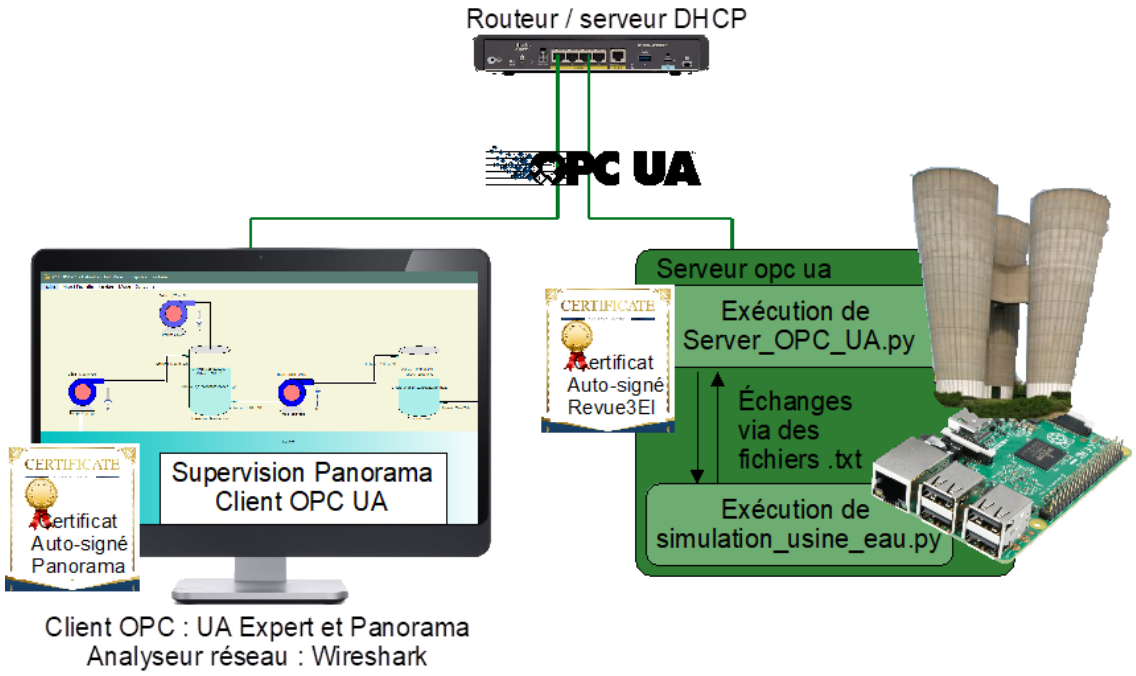

Figure 27 : Schéma de la configuration réseau pour la supervision du château d'eau simulé

Depuis Panorama Studio, ouvrir le fichier Panorama.ini du dossier SimuUsineMars2024 fourni avec cette ressource. SimuUsineMars2024 a été créé avec Panorama version 2023, il pourra donc être

ouvert avec une version plus récente. Aucune garantie par contre pour son ouverture avec une version plus ancienne.

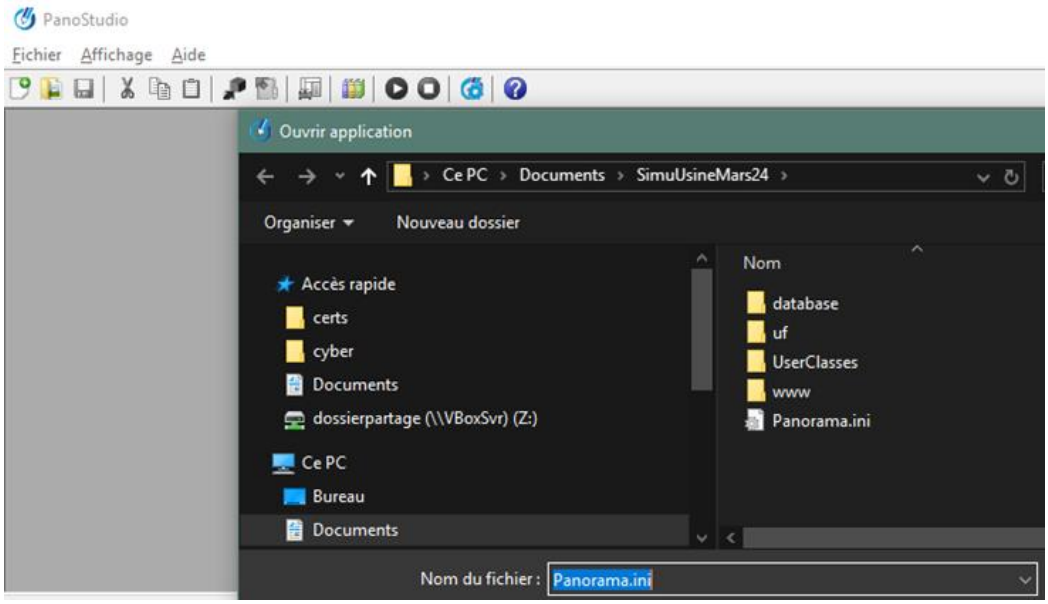

Figure 28 : Ouverture du projet

Ajuster la configuration OPC UA du projet.

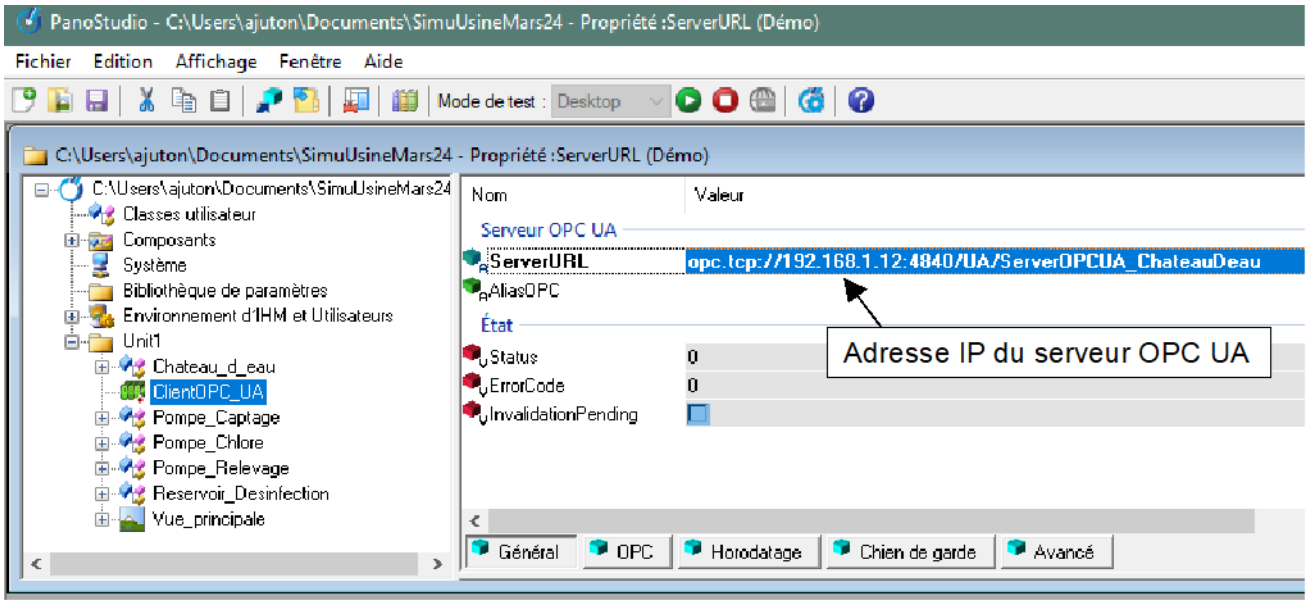

Figure 29 : Configuration de l'adresse IP du serveur OPC-UA / simulateur de château d'eau

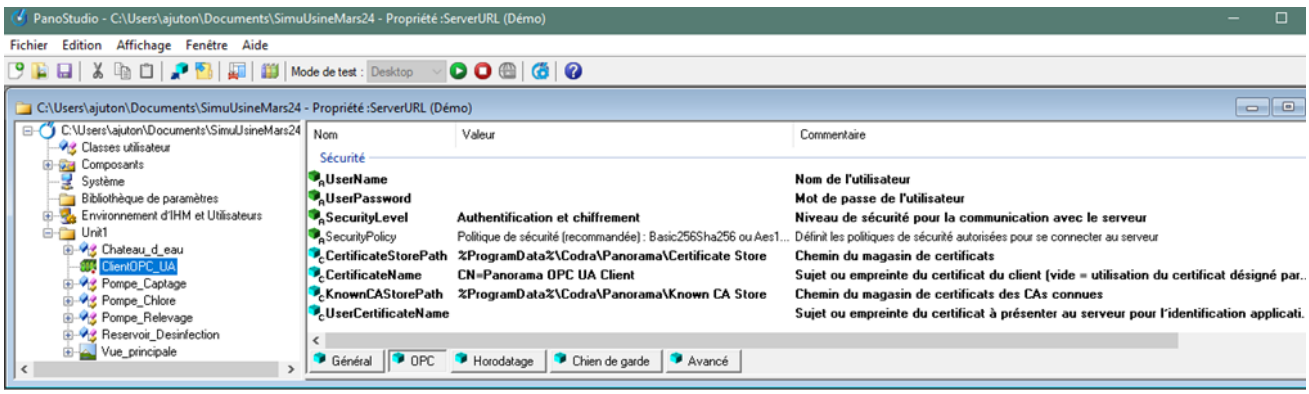

Figure 30 : Configuration des dépôts de certificat du client OPC UA

Pour se connecter de manière sécurisée, il ne manque plus à OPC-UA qu'un certificat SSL (X.509). Pour le créer depuis Windows (OpenSSL ne fonctionne que sous linux), Codra propose une solution (aide de Panorama rubriques *Création et installation du certificat Client OPC UA* et *Configuration de la sécurité*) ainsi qu'une fiche technique *FAQ080-V2.1 Création de certificats pour les fonctions Panorama* [12].

1. Installer [11] et démarrer PowerShell en administrateur

2. Exécuter les commandes suivantes (un peu différente de celles proposées par Panorama pour gérer l'utilisation de certificat auto-signé), en adaptant le chemin de destination du couple certificat/clé. Attention, la clé et le certificat doivent avoir le même nom, seule l'extension les différenciant.

```
PS C:\Users\ajuton\Downloads> $Cert = New-SelfSignedCertificate `
>> -Type Custom `
>> -Subject "CN=Panorama OPC UA Client" `
>> -HashAlgorithm sha256 `
>> -KeyAlgorithm RSA `
>> -KeyLength 4096 `
>> -KeyExportPolicy Exportable `
>> -Provider "Microsoft Enhanced RSA and AES Cryptographic Provider" `
>> -KeySpec KeyExchange `
>> -NotAfter (Get-Date).AddYears(5) `
>> -CertStoreLocation "Cert:\LocalMachine\My" `
>> -KeyUsage DigitalSignature,NonRepudiation,KeyEncipherment,DataEncipherment `
>> -TextExtension("2.5.29.37={text}1.3.6.1.5.5.7.3.2",
"2.5.29.17={text}URL=urn:localhost:CODRA:Panorama OPC UA Client")

PS C:\Users\ajuton\Downloads> Export-Certificate -Cert $Cert -FilePath
C:\Users\ajuton\Downloads\MyOpcUaClientCert5.der

PS C:\Users\ajuton\Downloads> $thumb = $Cert.Thumbprint

PS C:\Users\ajuton\Downloads> $CertPasswordEmpty = new-object System.Security.SecureString

PS C:\Users\ajuton\Downloads> Export-PfxCertificate -Cert "Cert:\LocalMachine\My\$thumb" -FilePath
C:\Users\ajuton\Downloads\MyOpcUaClientCert5.pfx -Password $CertPasswordEmpty
```

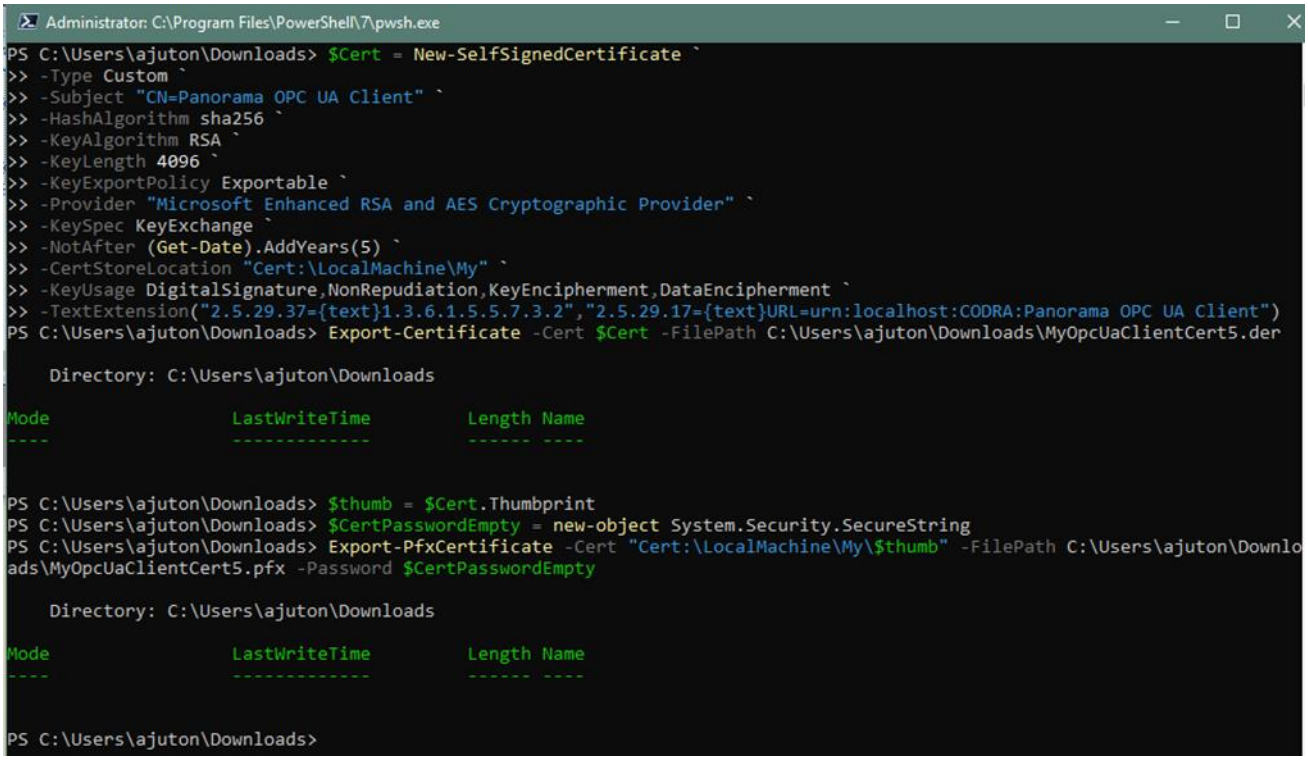

Figure 31 : Copie d'écran des commandes PowerShell de création du certificat X.509 et d'exportation de la clé privée associée

Copier le certificat généré, ainsi que le certificat du serveur OPC UA dans le dossier des certificats de Panorama :

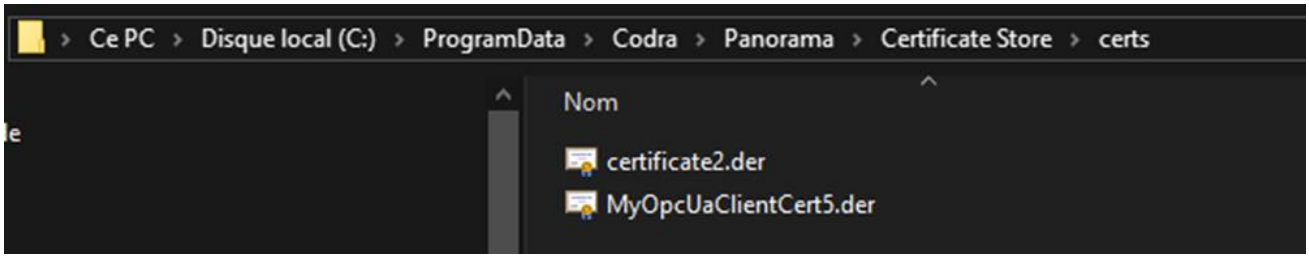

Figure 32 : Dossier certificats de Panorama

Faire de même avec la clé privée associée au certificat Client.

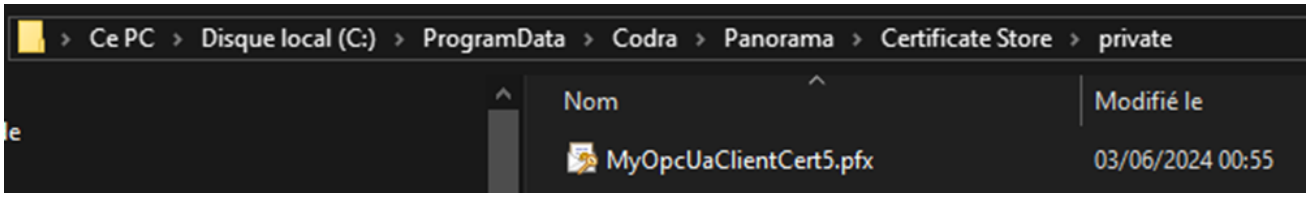

Figure 33 : Dossier clés privées de Panorama

Lancer le logiciel Codra Traceur (installé en même temps que Panorama) et éxécuter la supervision, en choisissant la vue principale. Le château d'eau est alors visualisé et contrôlable (on peut contrôler le débit de chaque pompe).

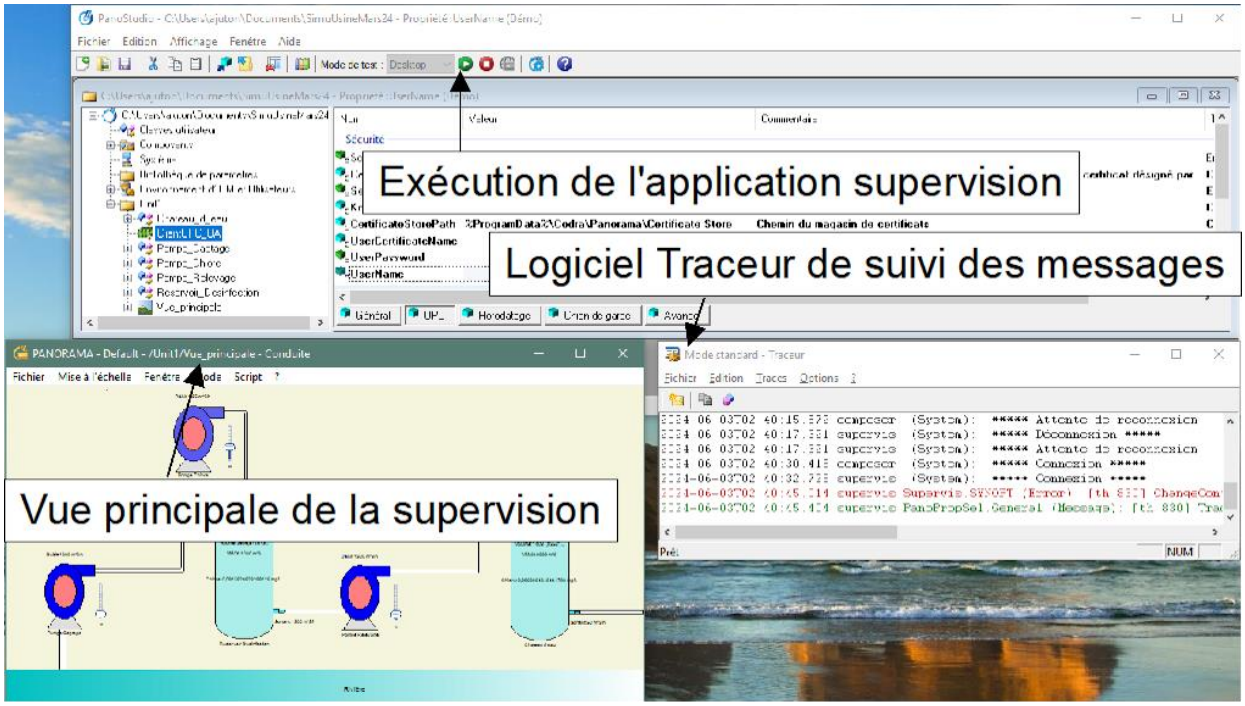

Figure 34 : Vue de la fenêtre de supervision du client OPC UA Panorama

Il est possible de connecter les clients OPC UA UAExpert et Panorama en même temps, montrant par là-même que le serveur accepte plusieurs clients simultanés. Wireshark peut être utilisé pour mettre en évidence la mise en place de la connexion sécurisée et les échanges (peu parlants lorsque la communication est chiffrée)

4 - Conclusion

Cette ressource s'est intéressée essentiellement à l'aspect sécurité d'OPC UA. Une des raisons du succès d'OPC UA est lié à l'utilisation et à la réputation des mécanismes SSL. Le monde de

l'automatisme industriel (OT – Operationnal Technology) tire parti des technologies développées et fiabilisées par l'informatique (IT – Information Technology). Le mode publisher/subscriber et les possibilités temps réels (basées sur la couche réseau TSN Time Sensitive Network) sont d'autres atouts d'OPC UA qui mériteraient également d'être présentés.

Les constructeurs intègrent de plus en plus souvent des serveurs OPC UA à leurs automates programmables. Par exemple, le S7-1200 de Siemens embarque un serveur OPC UA permettant d'accéder aux variables internes de l'automate. Hervé Discours, professeur à l'IUT de Cachan, a fait quelques vidéos très intéressantes sur le sujet [6].

Références :

[1] https://opcfoundation.org

[2] OPCUAcademics propose des ressources pédagogiques sur OPC UA. L'accès gratuit à ces ressources se demande sur la page https://opcfoundation.org/resources/opcuacademic/

[3] Codra, page de présentation de Panorama : https://codra.net/fr/offre-logiciel/plateforme-supervision/logiciel-panorama-suite/ et serveur web de téléchargement de Panorama - https://my.codra.net/

[4] Installation de Raspberry OS sur raspberry Pi4 et mise en place du bureau à distance. https://github.com/ajuton-ens/CourseVoituresAutonomesSaclay/blob/main/Bibliotheques_logicielles/Installation_RaspberryOS_CoVAPSy_v1re2.pdf

[5] Documentation de FreeOPCUA
https://github.com/FreeOpcUa/opcua-asyncio
https://opcua-asyncio.readthedocs.io/en/latest

[6] Chaîne Youtube de Hervé Discours :
OPC UA - Initiation : https://www.youtube.com/watch?v=iN4qKm5W35g
OPC UA – Cybersécurité : https://www.youtube.com/watch?v=58FUQzWxs3Y

[7] Using the BME280 I2C Temperature and Pressure Sensor in Python, Matt Hawkins
https://www.raspberrypi-spy.co.uk/2016/07/using-bme280-i2c-temperature-pressure-sensor-in-python/

[8] Guide Installation de Raspberry OS sur raspberry Pi 4 :
https://github.com/ajuton-ens/CourseVoituresAutonomesSaclay/blob/main/Bibliotheques_logicielles/Installation_RaspberryOS_CoVAPSy_v1re3.pdf

[9] Site officiel de Unified Automation pour le téléchargement de UA Expert
https://www.unified-automation.com/products/development-tools/uaexpert.html

[10] Site officiel de Wireshark : https://www.wireshark.org/

[11] Installation de PowerShell : https://learn.microsoft.com/fr-fr/powershell/scripting/install/installing-powershell

[12] Fiche technique FAQ080-V2.1 Création de certificats pour les fonctions Panorama :
https://my.codra.net/fr/productreleases?productrelease=PS-2023&selection=technicalfiles

[13] Fondamentaux de la sécurité réseau, M. Sechehaye, A. Juton, M. Sauvergeat, février 2024, https://eduscol.education.fr/sti/si-ens-paris-saclay/ressources_pedagogiques/fondamentaux-dela-securite-reseau

Ressource publiée sur Culture Sciences de l'Ingénieur : https://eduscol.education.fr/sti/si-ens-paris-saclay