OPC UA - Python

Requisitos:

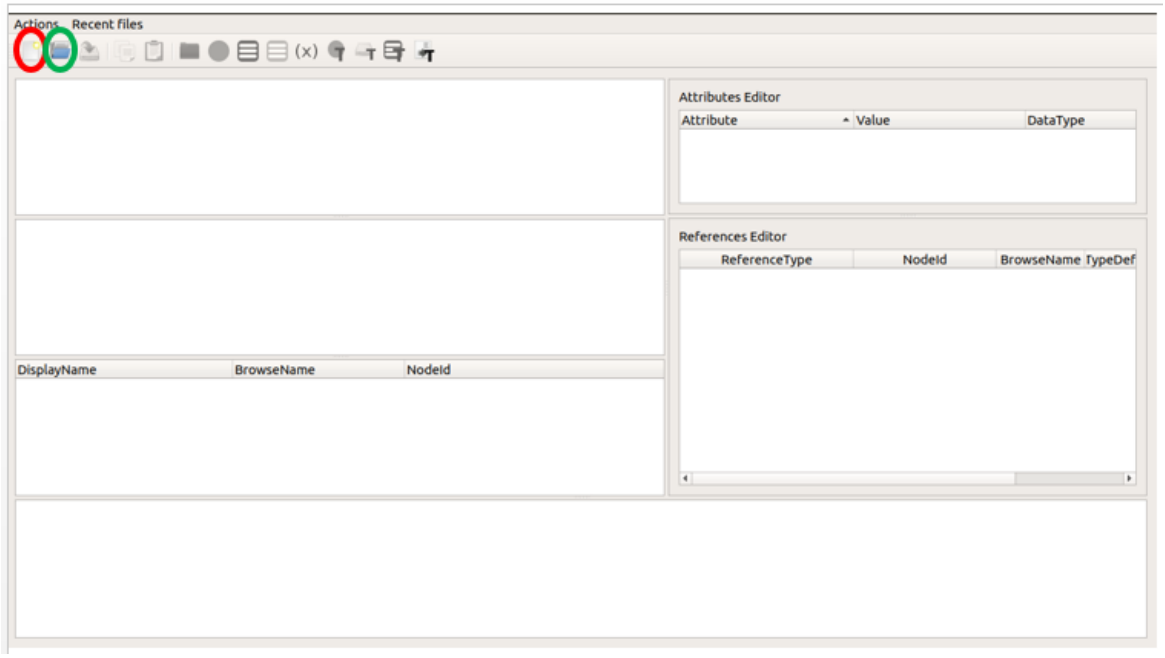
- Instalar la biblioteca “python-opcua” para Python:
  En Windows:
  py –m pip install opcua
  En Linux:
  apt install python-opcua
  apt install python-opcua-tools

  Tomado de: https://github.com/FreeOpcUa/python-opcua

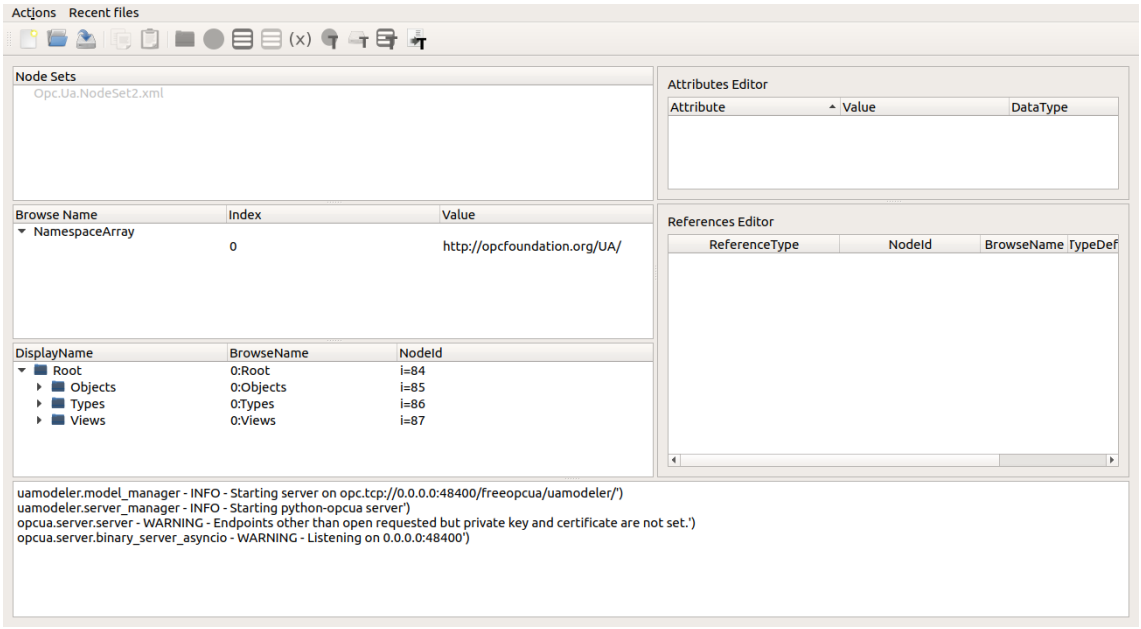
Para crear el modelo de información en formato xml:

- Instalar el programa “Free OPC UA Modeler”
  Instrucciones de instalación en: https://github.com/FreeOpcUa/opcua-modeler

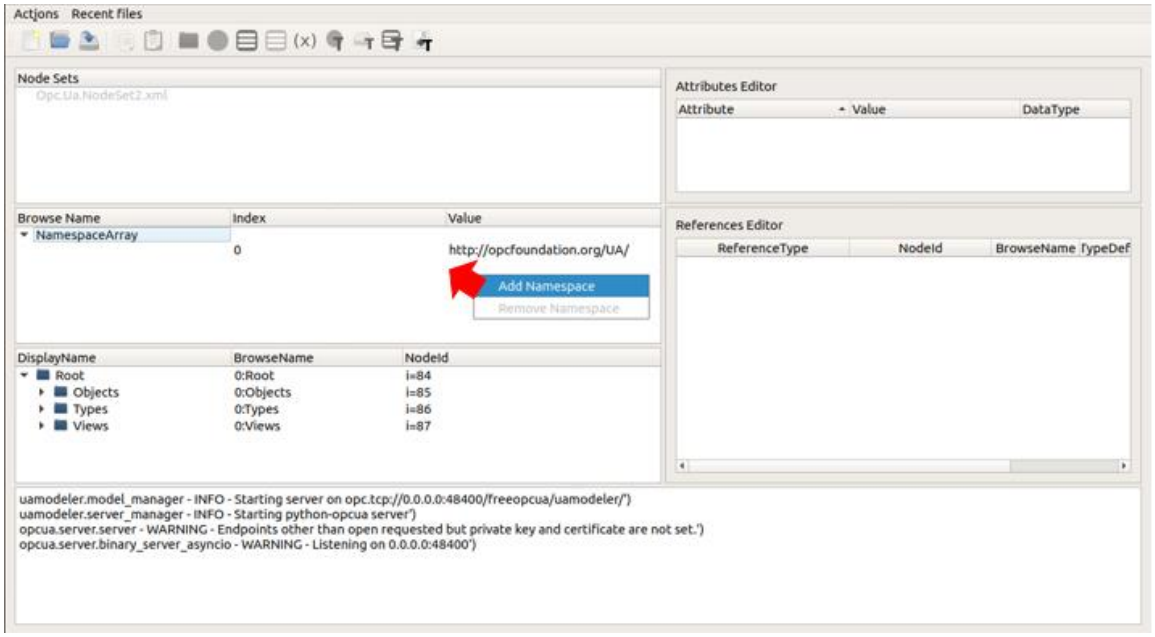
Al ejecutar el programa, se podrá crear un archivo xml con varios “nameSpaces”, “objetos” y “variables”. Al darle clic en la opción marcada con un círculo rojo, se creará un nuevo archivo. Al darle clic a la opción marcada con un círculo verde, se podrá abrir un archivo previamente creado para modificarlo.



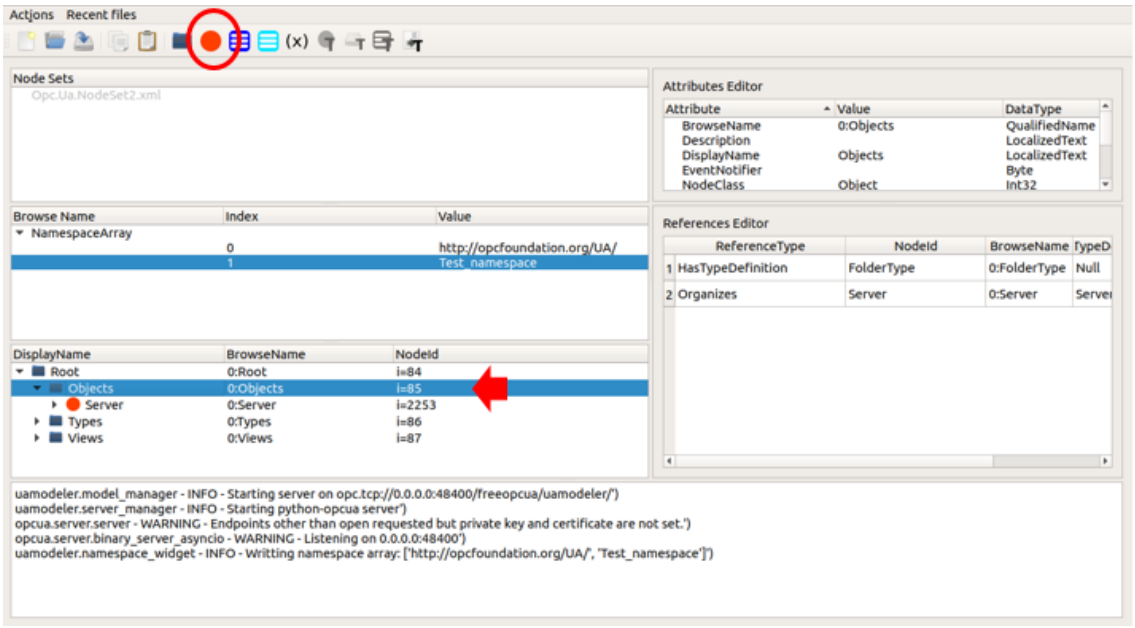
Al crear un archivo nuevo, se observará lo siguiente:



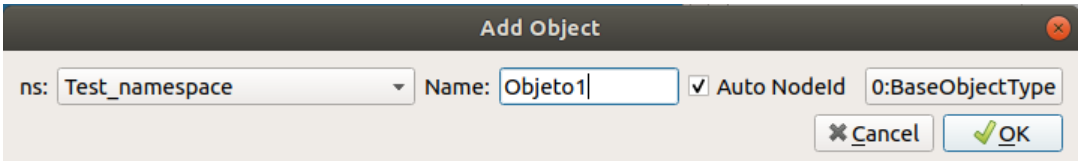
Para agregar un “namespace” dar clic derecho donde se observa la flecha roja, y seleccionar “Add Namespace” En ese instante se habilitará la opción de ingresar el nombre de este espacio.



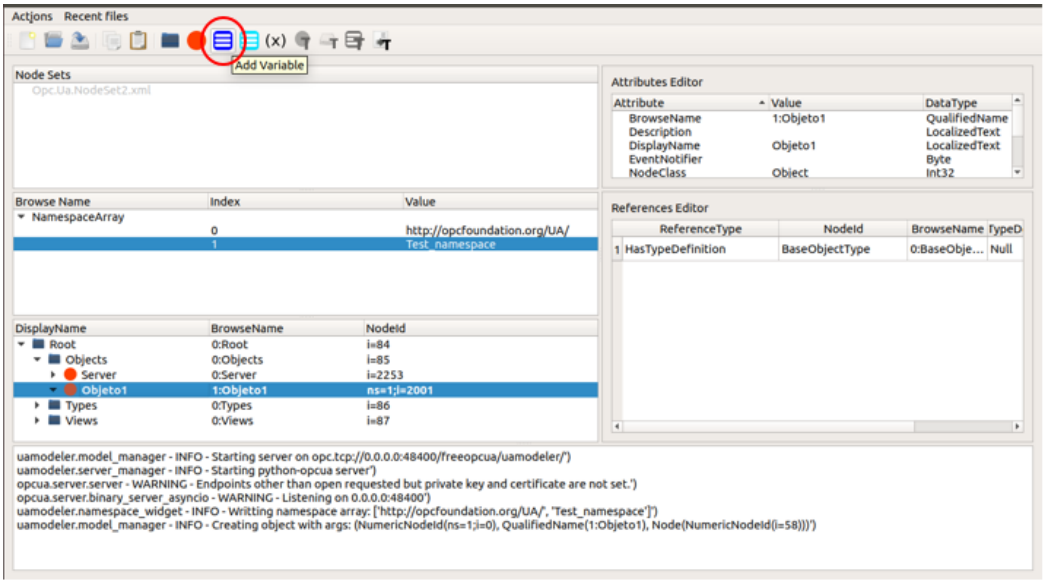
Seguidamente para añadir objetos en este espacio de trabajo, se selecciona el namespace y el directorio objetos. Se debe dar clic al icono circular naranja (mostrado con un círculo rojo)



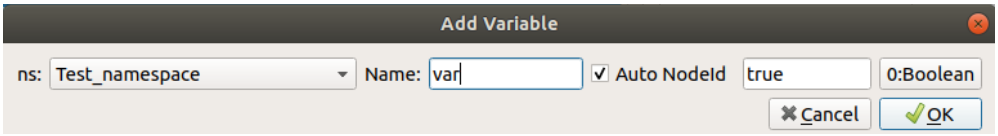
Se desplegará un cuadro donde se podrá indicar el nombre del objeto. Se puede indicar en cual namespace será creado el objeto.



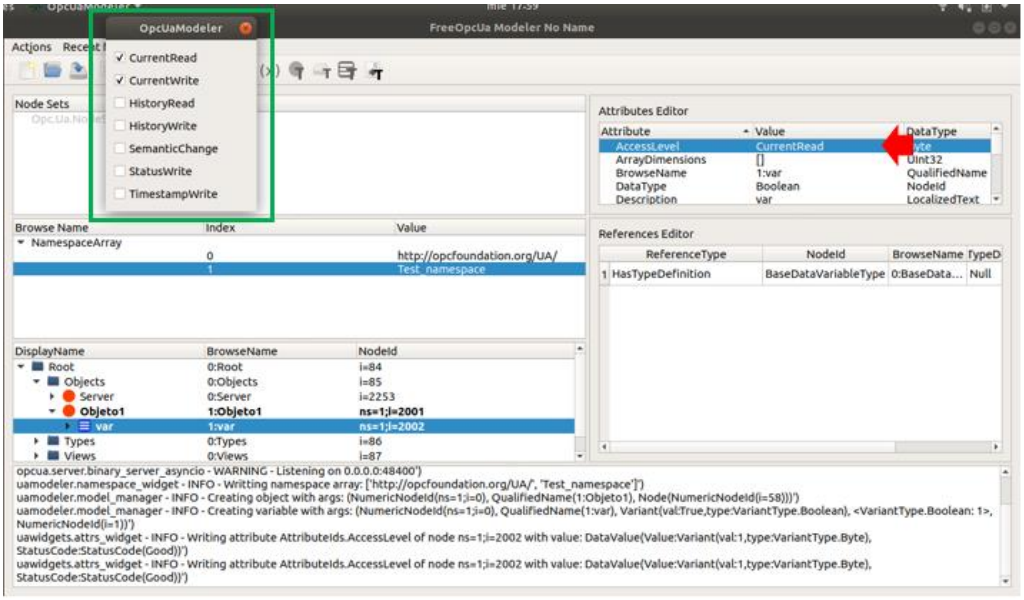
Igualmente para añadir variables los objetos, se selecciona el namespace y el objeto que será el padre. Se debe dar clic al icono cuadrado azul (mostrado con un círculo rojo)



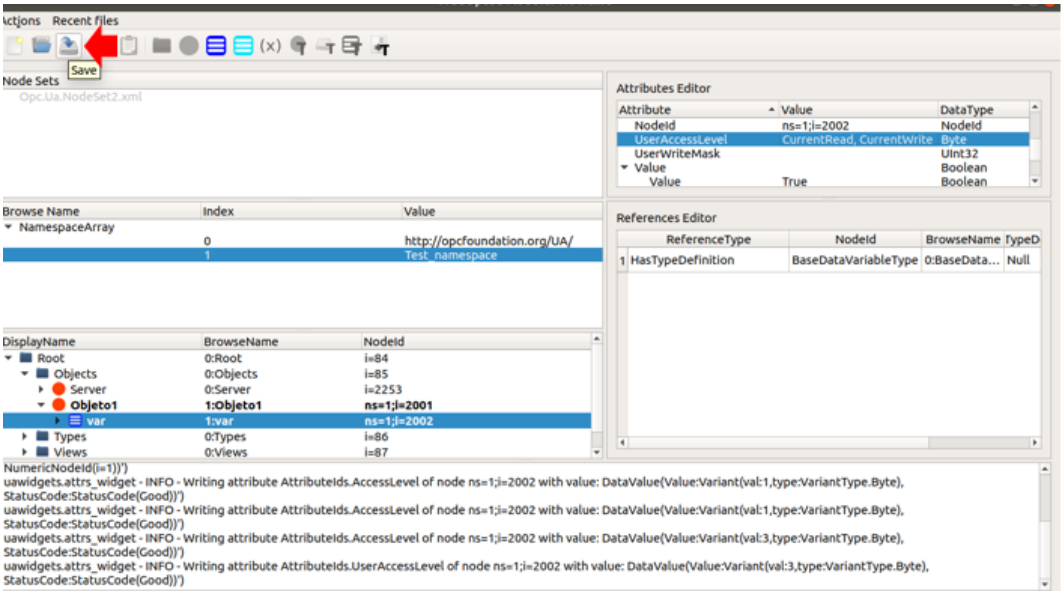
Se desplegará un cuadro donde se podrá indicar el nombre de la variable, el tipo de esta y su valor inicial. Se puede indicar en cual namespace será creado el objeto.



Una vez creada la variable, es importante cambiar su nivel de acceso, por defecto están configuradas como “Solo lectura”, por lo que seleccionando la variable, se dará clic derecho en la opción denominada “Access level” (Indicado con una flecha roja) para que se despliegue el menú (indicado con un cuadro verde). En este se debe seleccionar también la opción de escritura. Este paso debe realizarse para la opción “Access level” y “User Access level”



Finalmente, una vez que se hayan agregado los namespace objetos y variables deseados, se procederá a guardar el archivo, dando clic en el icono señalado.



Seguridad

Si se desea utilizar seguridad en la conexión servidor-cliente, en este caso seguridad tipo "Basic256Sha256_SignAndEncrypt", es necesario crear un certificado y una clave que deberán ser colocados en el mismo directorio donde se encuentre el servidor y crear otro certificado y clave que deberán estar en el mismo directorio donde se encuentre el cliente.

- Para crear este certificado y la clave, descargar el archivo ssl.config que se encuentre en el directorio 4-example en el repositorio https://github.com/lfvm0001/OPCUA_Test
- Modificar el archivo "ssl.config" las opciones de:
  - subjectAltName = URI:urn:opcua:python:server,IP: <COLOCAR IP DEL SERVIDOR>
  - countryName =
  - stateOrProvinceName =
  - localityName =
  - organizationName =
  - commonName =

- En Linux, ejecutar los siguientes comandos en la terminal:

openssl genrsa -out key.pem 2048
openssl req -x509 -days 3650 -new -out certificate.pem -key key.pem -config ssl.conf
openssl x509 -outform der -in certificate.pem -out certificate.der

*con la opción "-days" se puede definer el tiempo de validez del certificado, una vez este expiré será necesario sustituirlo por uno nuevo tanto en el servidor como en el cliente. En este caso el certificado tiene una validez de 10 años.
El certificado es de tipo .der y la llave de tipo .pem

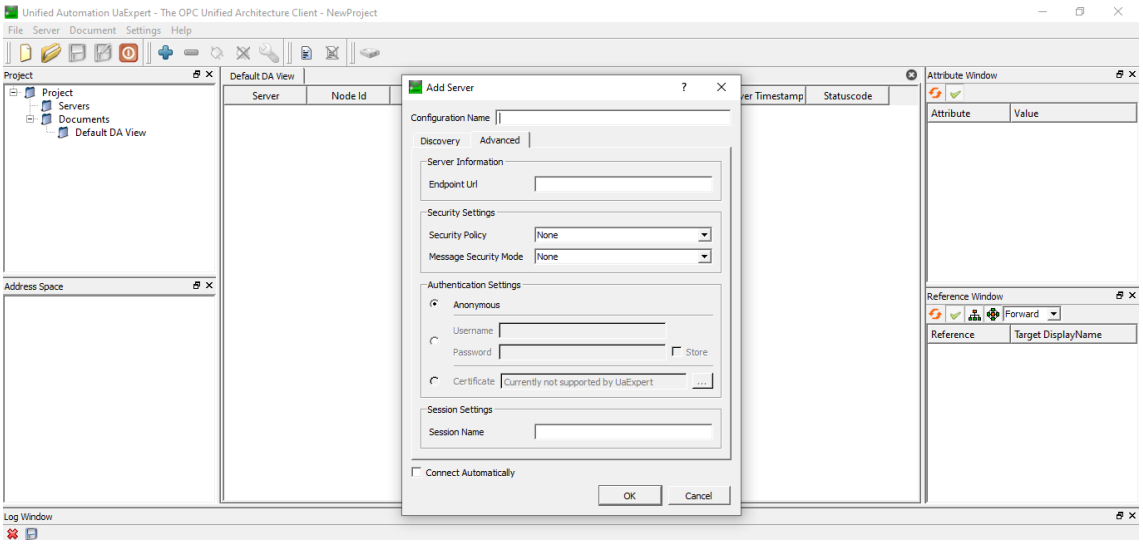
Información tomada de: https://github.com/AndreasHeine/SecurePythonOpcUaServer

GUI de Cliente para realizar pruebas

Para realizar pruebas de conexión y observar los datos, se puede utilizar el GUI UA Expert Client.

- El programa se puede descargar en el siguiente enlace: https://www.unified-automation.com/downloads/opc-ua-clients.html

En este programa, se da clic en el icono "+" para agregar un servidor. Se desplegará un menú donde en la opción "Advance" se debe colocar el Endpoint URL del servidor. Una vez agregado, se podrá conectar a este servidor cuando se encuentre disponible.



En caso de utilizar seguridad, en el programa es necesario definir el tipo de seguridad en "Security Settings" y cargar el certificado y clave del cliente en "Authentification Settings"

Servidor-Cliente en Python

Ambos archivos (server.py y client.py) pueden ser descargados del repositorio: https://github.com/lfvm0001/OPCUA_Test bajo el directorio 4-example

Es necesario descargar también el archivo readXML.py ya que contiene una función utilizada por el servidor y cliente (debe ser colocada en el mismo directorio).
Para ejecutar el servidor o el cliente, ejecutar en consola:

- En Windows:

  py server.py model.xml

  py client.py model.xml

- En Linux:

  python3 server.py model.xml

  python3 client.py model.xml

Si se desea utilizar un modelo de información diferente cambiar el último argumento, por ejemplo: python3 server.py ejemplo.xml.

Tanto en el servidor como en el cliente es necesario agregar la ip y puerto del servidor. Por lo tanto se modificara la línea:

```
url = "opc.tcp://172.16.102.43:4840"
```

La URL debe ser: `“opc.tcp://ip:puerto”`

En este código ejemplo tanto el servidor como cliente, leerán todas los namespace, objetos y variables cargados en el modelo de información e imprimirán su valor en pantalla (utilizando el comando “get_value()”)

Para cambiar el valor de una variable, se utiliza el comando set.value(VALUE), y es necesario conocer el nombre de la variable, su objeto padre y su namespace. Por ejemplo en el cliente, para modificar la variable “timeStamp” del objeto “Other” del namespace “OPCUA_TEST” se tiene en el código:

```
#Escribir el valor de una de las variables
if dicVar["name"] == "timeStamp" and dicVar["parentName"] == "Other" and dicVar["nsName"]== "OPCUA_TEST":
    node = client.get_node("ns=" + str(dicVar["ns"]) + "; i=" + str(dicVar["id"]))
    TIME = datetime.datetime.now()
    node.set_value(TIME)
```

Si se desea cambiar otra variable, sustituir “timeStamp” por la variable deseada, “Other” por el nombre de su objeto padre, y “OPCUA _TEST” por el namespace respectivo. Y finalmente colocar el valor deseado (valor proveniente de un sensor por ejemplo) en la instrucción: node.set_value(<valor>)