Python GUI Widget OPC UA Client

**Intro**

Il client è stato realizzato sfruttando le seguenti API OPC UA in Python:

<https://github.com/FreeOpcUa/python-opcua>.

Per lo sviluppo dell’interfaccia grafica si è fatto uso del framework PyQt5:

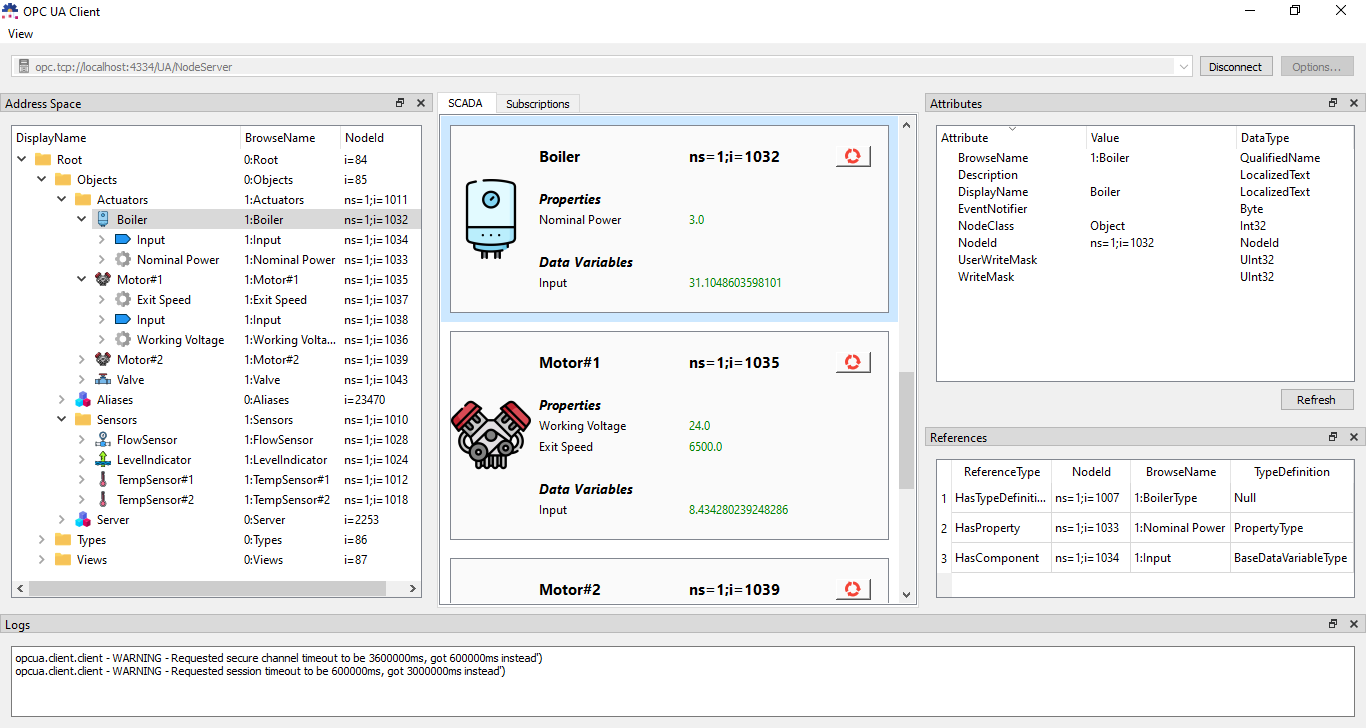
<https://www.qt.io/qt-for-python>.

Il Client offre la possibilità di visualizzare tutti gli endpoint offerti da un server OPC UA e connettersi ad esso in maniera sicura, tramite l’utilizzo di certificati che possono essere importati o generabiti attraverso l’applicazione stessa.

Dal Client è possibile navigare l’Address Space del server a cui ci si è connessi, visualizzando per ogni nodo in esso contenuto attributi e reference.

È inoltre possibile sottoscriversi per riceve aggiornamenti periodici di variabili di interesse.

Infine, il client è stato progettato per permettere di visualizzare degli appositi widget per monitorare e controllare le informazioni relative a oggetti custom, definibili dal server.

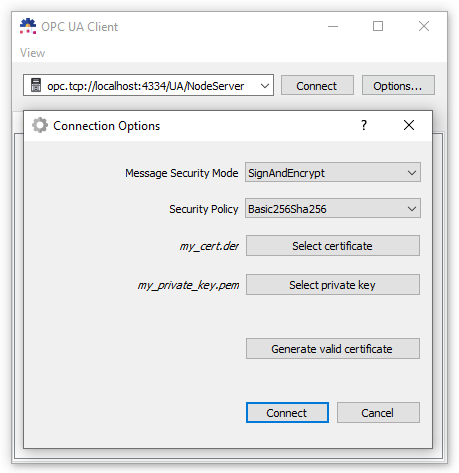
****

**Connessione**

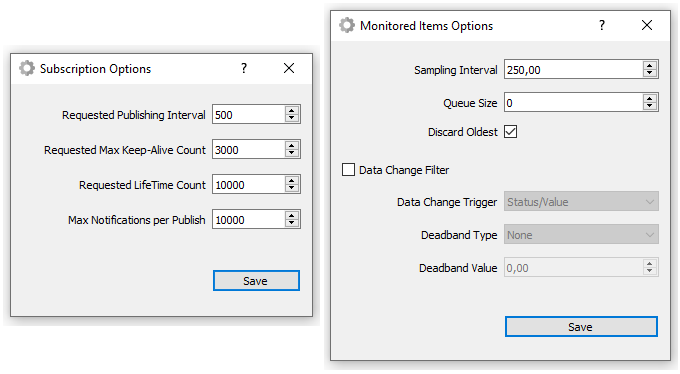
**Nella barra in alto è possibile inserire l’URI del server a cui ci si vuole collegare.**

**Cliccando sul bottone Connect ci si collega utilizzando le ultime impostazioni memorizzate associate all’URI in questione mentre, cliccando su Options, si apre una finestra che mostra tutti i possibili Session Endpoint offerti dal server e che permette di selezionare un certificato per realizzare il canale sicuro.**

**Se necessario, è possibile generare un certificato valido per l’applicazione, cliccando sull’apposito bottone.**

****

**Prima che avvenga effettivamente la connessione, l’utente ha la possibilità di inserire i parametri che caratterizzeranno la subscription e la creazione dei monitored items, attraverso le seguenti interfacce.**

****

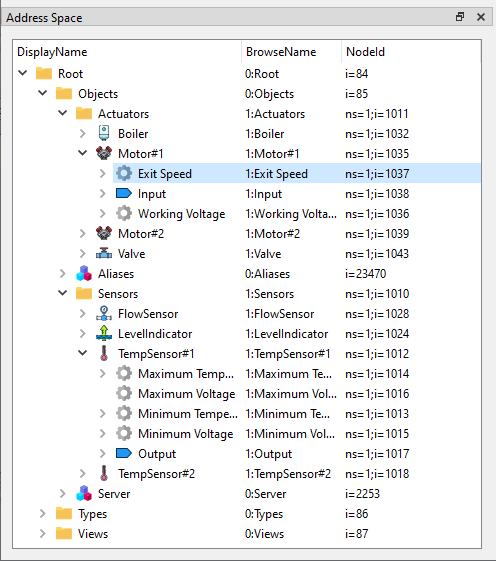
**In qualsiasi momento è possibile disconnettersi dal server.**

**Address Space**

**Una volta connessi è possibile esplorare l’Address Space del server tramite una visualizzazione ad albero.**

**Per ogni nodo è possibile visualizzare display name, browse name, namespace index e node id.**

**Attraverso questa vista è possibile effettuare la browse sui diversi nodi.**

****

**Attributi e Reference**

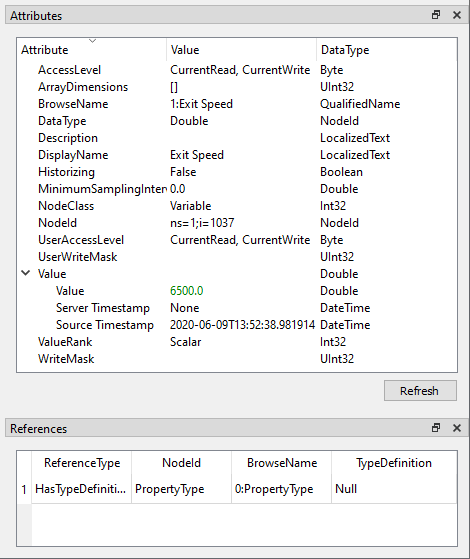
**Ogni volta che si seleziona un nodo nell’address space vengono visualizzate tutte le sue informazioni: attributi e reference.**

**Se il nodo selezionato appartiene alla Node Class Variable, ovvero è una Data Variable o una Property, il suo valore viene colorato sulla base del suo status code:**

* **Verde → Good / Success**
* **Giallo → Uncertain / Warning**
* **Rosso → Bad / Failure**

**L’utente ha anche la possibilità di modificare il valore di ogni singolo attributo.**

**Le reference visualizzate sono quelle uscenti dal nodo.**

****

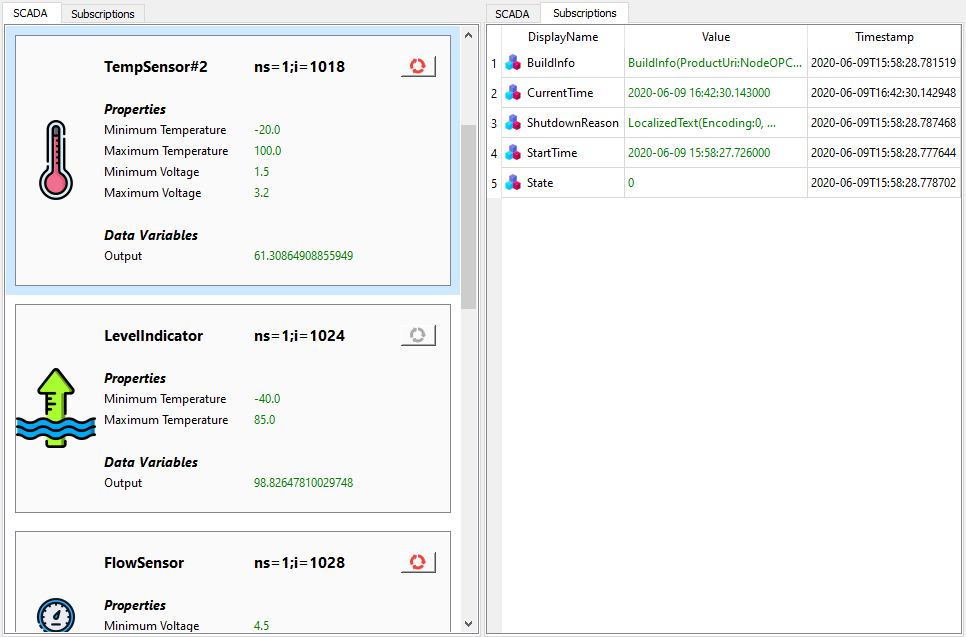
**Interfaccia SCADA e Subscriptions**

**Il Client presenta un’interfaccia SCADA che permette una visualizzazione dettagliata degli oggetti custom, sensori e attuatori, definiti nel server.**

**Una volta effettuata la connessione col server, viene in automatico creata una sottoscrizione a cui vengono associate, in qualità di monitored items, tutte le variabili di tutti gli oggetti custom definiti dal server.**

**All’utente viene offerta la possibilità di bloccare l’aggiornamento automatico dei dati relativi ad un singolo oggetto nello SCADA, rimuovendo tutti i monitored item ad esso associati dalla sottoscrizione.**

**Viene anche data la possibilità di monitorare le variabili di oggetti non custom, mediante la tab Subscriptions.**

****

**Nota:**

**Per avviare il server lanciare dalla directory server il comando *npm install* (se non lo si è fatto prima), e successivamente *npm start*.**

**Per avviare il client bisogna prima installare i requirements attraverso il comando *pip install -r requiremements.txt*, e poi lanciare dalla directory client il comando *python mainwindow.py* .**