Progetto Industrial Informatics

Cliccando il titolo si verrà reindirizzati al progetto di GitHub

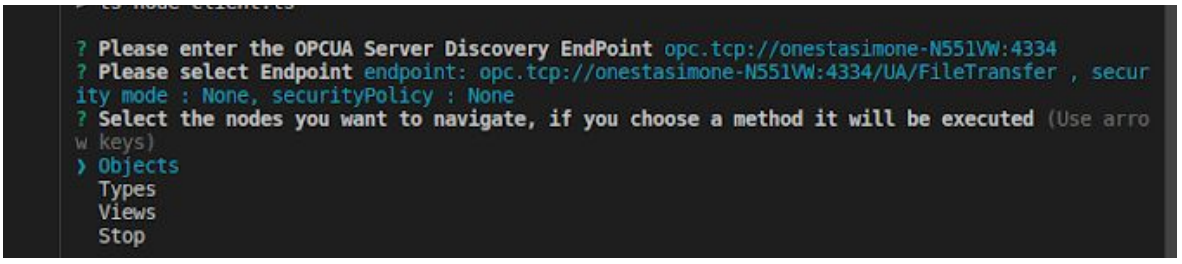
L'obiettivo del progetto è l'elaborazione di un client e di un server OPCUA in nodejs che:

1) Sfruttino il meccanismo di file transfer per l'invio di script
2) Definiscano e richiamino metodi per l'esecuzione degli scripts all'interno del Server.

Il Server sarà inserito all'interno di un dispositivo Raspberry.
Per quanto concerne il meccanismo OPC UA file transfer è stato utilizzato un ObjectType FileType, il quale offre dei metodi standard per la gestione del file come: apertura, lettura, scrittura ecc… ai quali è stato aggiunto un metodo custom per l'esecuzione. Gli script sono organizzati all'interno di una cartella "script" presente all'interno del server, la quale viene mappata all'interno di un Object OPCUA insieme ai metodi per la gestione dei file(addFileObject e RemoveFile).

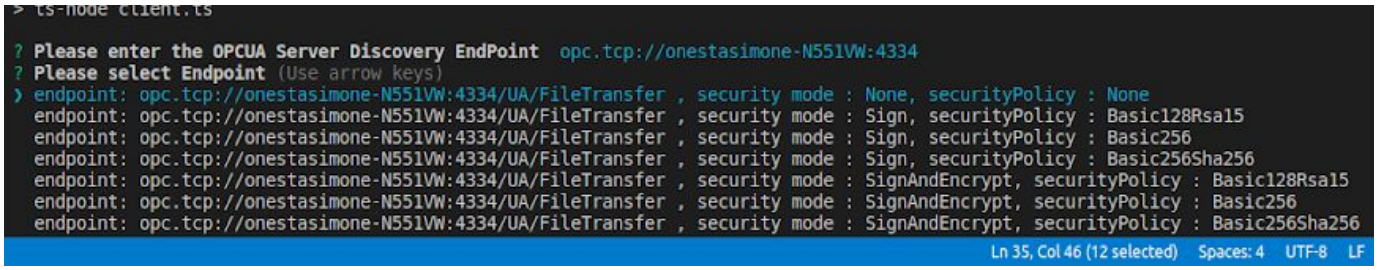
Client

Il client è stato implementato per riuscire a navigare all'interno dell'Address space del Server ed eseguire i metodi esposti da quest'ultimo.
Come interfaccia grafica è stata utilizzato il modulo di nodejs Inquirer e di seguito viene proposta un'anteprima:



Come prima azione il Client richiede l'immissione dell'indirizzo del Discovery Endpoint del Server dal quale otterrà la lista di Session Endpoint disponibili, attraverso il servizio di **GetEndpoints,** che verranno visualizzati a video indicando:

- Endpoint Url
- Security Mode
- Security Policy



L'utente potrà selezionare il Session Endpoint desiderato e verrà creata ed attivata una sessione.
Una volta stabilita la connessione e creata la sessione l'utente sarà capace di navigare l'address space a partire dalla RootFolder.
Selezionando un nodo il Client verificherà se quest'ultimo risulta essere un nodo Object o un nodo Method e :

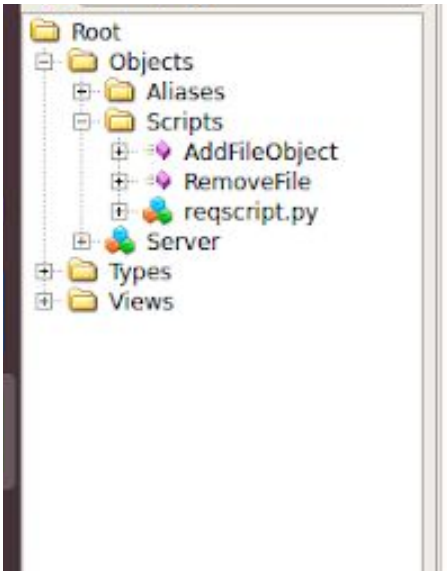
- Se il nodo è di tipo Object effettuerà il browse del nodo
- Se il nodo è di tipo Method eseguirà il metodo. Per l'esecuzione sarà necessario inserire i parametri di ingresso che verranno richiesti al nodo in modo dinamico e successivamente verrà stampato a video il risultato della chiamata con il relativo stato.

E' possibile chiudere il Client selezionando la scelta Stop, che chiuderà la sessione, scollegherà il client e terminerà l'applicazione client

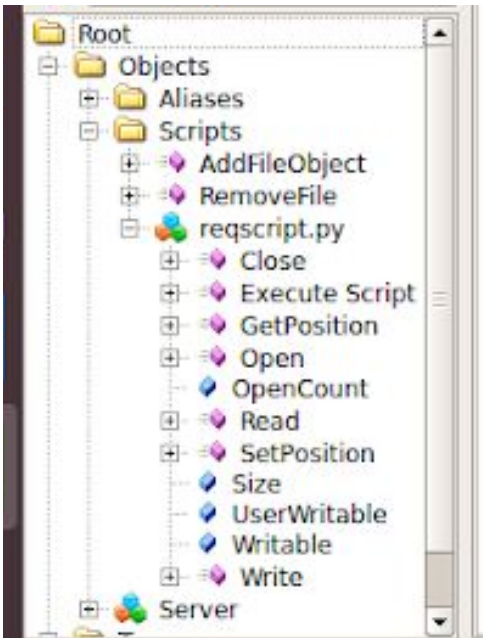
Server

Il server consiste in una serie di moduli scritti in NodeJS, che verrà eseguito su un dispositivo Raspberry Pi 4.
Esso ha l'obiettivo di definire un Address Space personalizzato, che presenta una directory "Scripts" che ospiterà dei file di script che potranno essere eseguiti all'interno del dispositivo, tipicamente scritti in Python o in Typescript. La directory presente sull'Address Space presenta una directory corrispondente nel file system locale, anch'essa denominata "scripts".
All'avvio del server, questo analizza il contenuto della cartella "scripts" in locale e, se vengono riscontrati dei file, questi vengono caricati sull'Address Space, in modo che in caso di caduta inaspettata del server, i file caricati non vengano persi, o comunque vengano esclusi dall'Address Space.
L'oggetto cartella sull'Address Space contiene due metodi:

- *AddFileObject*: si occupa della creazione del file accettandone il nome come parametro e impostando questo come NodeID, in modo che nello stesso namespace se ne possa presentare soltanto una occorrenza. Tale file verrà creato in locale solo nel momento in cui viene eseguita un'operazione di scrittura;
- *RemoveFile*: provvede alla rimozione di un file sia dalla cartella locale, sia dall'Address Space, lanciando una eccezione nel caso in cui questo non venga trovato.



All’aggiunta di un nuovo file, questo verrà aggiunto nell’address space, sotto la cartella in cui il metodo è stato chiamato, mediante l’oggetto messo a disposizione dallo SDK nell’operazione di bind del metodo all’oggetto: la struttura dati context.object, che contiene, appunto, le informazioni sull’oggetto su cui il metodo creato viene richiamato.
Per il trasferimento dei file da client a server è stato utilizzato il modulo `node-opcua-file-transfer`, tramite il quale è possibile istanziare oggetti di tipo file e collegarli a istanze di file sul File System locale
Un file conterrà, oltre ai metodi che consentono di effettuare le operazioni di base sui file (apertura, lettura, scrittura), un metodo personalizzato, “Execute Script”, che si occupa di lanciare un processo figlio che eseguirà il comando associato all’esecuzione del file selezionato, a seconda della sua estensione (ad esempio se il file ha estensione .py verrà lanciato il comando “*python3 nomefile*” mentre per file con estensione .ts sarà invocato il comando “*ts-node nomefile*”).



Il metodo Execute Script, tramite lo stesso oggetto context.object visto in precedenza, rileva in automatico il nome del file e lo utilizza per l'invocazione del comando. Tale metodo accetta come parametro di ingresso un intero, che determina la modalità di esecuzione:

- 0: Esecuzione asincrona, il client non riceve output, applicabile in casi in cui lo script sia un firmware da lasciare in esecuzione sul dispositivo, ad esempio un'accensione intermittente di un LED. In questo caso, viene previsto anche un meccanismo di arresto di un eventuale processo asincrono già in esecuzione, prima del lancio del nuovo comando;
- 1: Esecuzione sincrona, con output di ritorno al client, utile nel caso in cui lo script esegua un'azione sincrona, come una richiesta HTTP.

Nonostante all'interno della documentazione OPC UA il metodo risulta essere sincrono è stato implementata comunque la parte asincrona anche se non compliant.

Renato Sortino O55000405

Simone Onesta O55000431