Haute Ecole Arc - Ingénierie

Rapport  
Connected Factory

Rapport Travail d’Automne 2016 – Projet 15DLM-TA655

Superviseur : Huber Droz

Abstract

Ce rapport présente le déroulement de l’exploration du groupement de spécification Opc-Ua.

Les objectifs principaux n’ont pas pu être réalisés pour des raisons techniques qui seront détaillées dans ce rapport.

//TODO traduire en anglais

Table des matières

[Introduction 5](#_Toc473061796)

[Rappel des objectifs du projet 5](#_Toc473061797)

[Automatisation complète à la taille de lot 1 5](#_Toc473061798)

[Contraintes 5](#_Toc473061799)

[Méthodologie 6](#_Toc473061800)

[Analyse 6](#_Toc473061801)

[Spécification OPC UA 6](#_Toc473061802)

[Comparaison des implémentations existantes du protocole Opc-Ua 7](#_Toc473061803)

[digitalpetri/ua-client-sdk et digitalpetri/ua-server-sdk 7](#_Toc473061804)

[eclipse/milo 7](#_Toc473061805)

[ctron/de.dentrassi.camel.milo 7](#_Toc473061806)

[FreeOpcUa/python-opcua 8](#_Toc473061807)

[Interfaçage sur les machines 8](#_Toc473061808)

[Conclusion 8](#_Toc473061809)

[Références 9](#_Toc473061810)

# Introduction

Opc Ua est l’acronyme de « Open Platform Communication Unified Architecture. ». C’est un protocole de communication de machine à machine développé par la fondation OPC. C’est protocole standard d'intégration de systèmes et de communications pour l'automatisation, l’industrie 4.0 et l’internet des objets « Internet of Things ».

L’objectif de ce projet est d’implémenter et tester un prototype de connexion de machines en utilisant la technologie OPC Unified Architecture. Le travail s’inscrit dans une vision stratégique de la Haute Ecole Arc du « Smart and Micro Manufacturing » visant à soutenir les industriels de l’arc jurassien à faire face à la concurrence mondiale dans ce domaine.

# 

# Rappel des objectifs du projet

## Automatisation complète à la taille de lot 1

Moulage par injection, fabrication additive, reprise en usinage et contrôle qualité

On automatise la fabrication additive et réalise une ligne de fabrication interconnectée dans laquelle les informations circulent en continu pour la personnalisation de pièces de série avec des technologies Industrie 4.0. Un robot à sept axes établit l'interconnexion entre la presse à injecter et la machine de fabrication additive, un autre robot (ou le même pour les besoins du prototype) reprend la pièce à la sortie de la machine de fabrication additive pour l’amener à une machine d’usinage 5-axes en vue d’une opération de reprise. Les pièces sont ensuite contrôlées par vision dans un labo de métrologie (manutention par l’homme dans un premier temps), la machine de mesure fournit cependant les résultats sur une page internet spécifique.

Une fois que la presse a injecté une poignée en plastique sur des ciseaux de bureau et qu'un code Datamatrix a été inscrit, le robot à sept axes retire la pièce et son support de la bande transporteuse de la cellule d'injection. La paire de ciseaux est identifiée par scanner grâce à son code et l'étape de production suivante commence. Le robot charge/décharge la chambre de fabrication. La machine de fabrication additive ajoute une forme 3D en plastique personnalisée sur la poignée de ciseaux.

Le robot charge/décharge ensuite la pièce ainsi personnalisée pour l’amener sur une machine d’usinage 5-axes en vue d’une opération de reprise.

Une pièce unique de taille de lot 1 est ainsi produite.

Avant la remise des pièces par le robot, elles sont soumises à un contrôle qualité par vision.

## Contraintes

Les machines concernées par le projet sont les suivantes :

1. Centre d'usinage 5axes Mikron XSM400U
   * Commande Heidenhain HR410M
2. Presse d'injection Arburg Allrounder 170S 180-70
   * Commande Selogica
3. Robot 6 axes Staübli (RX90 ou TX60L)
   * Contrôleur Staübli CS8

## Méthodologie

Dans un premier temps j’ai étudié le fonctionnement du protocole Opc Ua.

Dans un deuxième temps, j’ai cherché une implémentation fonctionnelle et pratique d’utilisation.

Troisièmement, j’ai appris à m’en servir afin de faire fonctionner un système client et serveur capable de communiquer au travers du protocole Opc Ua.

Finalement, j’ai étudié la spécificité des machines à interfacer.

# Analyse

Dans cette partie, je vais détailler la technologie Opc-Ua et les diverses possibilités étudiées sur les implémentations existantes à utiliser pour le projet seront détaillées. Le choix définitif sera justifié.

## Spécification OPC UA

La spécification OPC est définie le protocole de communication entre un client et un server. Le client envoie une requête au serveur et celui-ci lui répond puis se met en attente. Cependant, le client peut demander au server de l’informer lors d’un changement sur une donnée spécifique. Dans ce cas, le serveur enverra des données au client de manière autonome.

La fonctionnalité principale est la possibilité d’accéder et de modifier une donnée. Pour ce faire, chaque donnée possède un nombre d’attribut minium définit. Elle doit posséder un nom et une valeur. En plus de ça certaines informations supplémentaires viennent s’ajouter. L’instant à laquelle la donnée a été lue fait aussi partie des attributs minimum. Cette information peut correspondre au moment où l’information a été envoyée par le nœud où la machine est connectée ou le moment où le serveur l’a reçue. Finalement, le dernier attribut est la qualité de la donnée.

En résumé: Data = Name, Value, Timestamp and Quality.

Le deuxième protocole permet de monitorer une ressource sur le serveur. Le client peut s’inscrire pour recevoir la nouvelle valeur d’une donnée spécifique si celle-ci venait à être modifiée. Le serveur ne conserve aucune information de l’échange et envoie juste la nouvelle valeur ainsi que le moment où elle a été modifiée mais il n’envoie pas les attributs du nom ou de la qualité.

Le troisième protocole ajoute la fonctionnalité d’historique d’un élément. Le client peut faire une requête sur plusieurs valeurs antérieures d’une donnée.

Opc Ua rend ces protocoles agnostiques aux systèmes d’exploitation et les encapsules dans une nouvelle structure de donnée. Il est possible de faire une requête concernant plusieurs données et toutes vont être lu simultanément et envoyé au client sous la forme d’un seul paquet.

La communication entre le client et le serveur est indépendante des protocoles de communication par réseau bas niveau. Le transport d’information se fait au travers du protocole TCP/IP puis une surcouche en SSL, http ou https est ajoutée. La couche de communication ne sécurise pas seulement les données, il sécurise aussi l’authentification ce qui protège le serveur de l’infiltration ou de la modification par un tiers non autorisé. Cette partie est basée sur la norme de certificat X.509[[1]](#endnote-1) qui comprend trois parties dont la première doit être réalisée manuellement en peer-to-peer puis le reste est automatique.

## Comparaison des implémentations existantes du protocole Opc-Ua

Il existe déjà un bon nombre d’implémentation du protocole Opc-Ua[[2]](#endnote-2) Dans un souci de cohérence et de simplicité, je me suis uniquement penché sur les projets qui implémentent le stack, le client et le serveur. Je les ai testés dans un ordre de préférence de langage de programmation. Tous les projets sont capables de faire de la lecture et de l’écriture de donnée au travers du protocole Opc-Ua.

### digitalpetri/ua-client-sdk et digitalpetri/ua-server-sdk[[3]](#endnote-3)

Ce projet est écrit en Java. La création du client ou du server a échouée lors de l’utilisation de l’exemple fourni.

Aucune classe n’était exécutable et mes efforts pour créer une classe permettant l’utilisation de classe provenant de ce projet n’ont pas été fructueux.

### eclipse/milo[[4]](#endnote-4)

Ce projet est intègre le projet précédant en utilisant Maven.

La classe ClientServerExemple.java lance une exception appelée « UaException : status=Bad\_CertificateTimeInvalid, message=security checks failed ». Cette exception est lancée parce que le fichier contenant le certificat de sécurité est placé dans le dossier des certificats non-reconnu par l’application. La résolution de ce problème proposé par les développeurs eux-mêmes consiste à déplacer ce fichier à la main dans le dossier des certificats reconnus. Cette solution n’a donné aucun résultat.

Il semblerait qu’il ne soit pas possible d’instancier un protocole sans certificat de sécurité.

### ctron/de.dentrassi.camel.milo[[5]](#endnote-5)

Ce projet intègre le projet précédant en utilisant Camel. Apache Camel est un framework Java libre dont le but est de rendre l’intégration d’applications plus facile et plus accessible pour les développeurs.

Je n’ai pas lancé d’exemple avec ce projet c’est un projet qui demande beaucoup de dépendance et qui est tente de rester très haut niveau. Dans un souci d’avoir un contrôle fin sur l’utilisation de la technologie Opc-Ua, ce projet occulte trop d’informations pour adapter le code à des cas spécifiques. L’utilisation de beaucoup de dépendances est aussi un problème car l’erreur est plus difficile à déterminer. Pour ces raisons je me suis tourné vers un projet plus bas niveau.

### FreeOpcUa/python-opcua[[6]](#endnote-6)

FreeOpcUa est un projet open-source développé en python 3.

Les exemples de base sont fonctionnels tant au niveau du client qu’au niveau du serveur. L’importation de XML coté serveur et coté Client ne semble pas fonctionnelle.

Le client graphique est fonctionnel et permet le test de toutes les fonctions de base de la spécification Opc Ua.

## Interfaçage sur les machines

Le but est d’accéder aux variables et fonctions d’une des machines listées au travers du protocole Opc Ua. Ce protocole doit être implémenté par le fabriquant de la commande de la machine.

Après des recherches approfondies, j’ai conclu qu’aucune des commandes spécifiées dans ce projet n’implémentaient ce protocole. Pire, aucune commande actuellement en possession par la He-Arc l’implémente. Un employé de la marque Stäubli a même affirmé que ce protocole n’allait pas être intégré à sa prochaine génération de commande telle que le contrôleur robot CS9[[7]](#endnote-7).

Nous avons donc rencontré un problème insoluble à ce niveau.

# Conclusion

# Références

1. https://fr.wikipedia.org/wiki/X.509 [↑](#endnote-ref-1)
2. <https://github.com/digitalpetri/opc-ua-stack> [↑](#endnote-ref-2)
3. https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations [↑](#endnote-ref-3)
4. https://github.com/digitalpetri/ua-client-sdk [↑](#endnote-ref-4)
5. <https://dentrassi.de/2016/07/18/bringing-opc-ua-to-apache-camel/> [↑](#endnote-ref-5)
6. https://github.com/FreeOpcUa/python-opcua [↑](#endnote-ref-6)
7. http://www.staubli.com/fr/robotique/controleurs-robots/robot-controller-cs9/controleur-robot-cs9/ [↑](#endnote-ref-7)