MECHENG 709/710 (2024) Assignment

OPC UA for Shop-floor Communications (10%)

Introduction:

In this assignment, you will learn the basic concepts and applications of OPC UA in a modern factory, a.k.a. Smart Factory. The machines you will deal with are a CNC lathe, KUKA KR10 and conveyor belt; all are located in the Laboratory for Industry 4.0 Smart Manufacturing Systems (B405.870).

You will be working in **groups of 2.** You should sign into a group on Canvas before 5.00pm on Friday May 10. The assignment requires you to:

- *Inspect* and *understand* the provided OPC UA server and client Python codes
- *Answer* the questions related to the provided source codes
- *Complete* the client codes
- *Run* both server code and client codes to acquire the logged data

The OPC UA standard defines the technical aspects of communication and modelling, i.e. network protocols, definitions of nodes and how they reference each other. It also defines the best practices about how a technical system (e.g. a machine tool) can be adequately represented and modelled. If you are familiar with object-oriented programming (OOP) and unified modelling language (UML), you will recognize many similar traits, although the terminology used may differ.

Background:

The whole shop-floor communication is based on OPC UA, and the shop-floor is represented by an OPC UA aggregation server. Assume that there are two companies in Auckland now that are using the services provided by this shop floor to fabricate their products. Company one has two types of products to make, part A and part B. Both of them require the use of the CNC lathe on the shop-floor. Company two only has one product, part C, not requiring the CNC lathe, instead a transferring operation to an inspection station. Each company has its own OPC UA client to communicate with the shop-floor OPC UA (server).

The clients have three main responsibilities:

1) send operation requests to the shop-floor;
2) monitor the progress of the operations;
3) monitor the status of each machine.

Figure 1 explains the communication scenario.

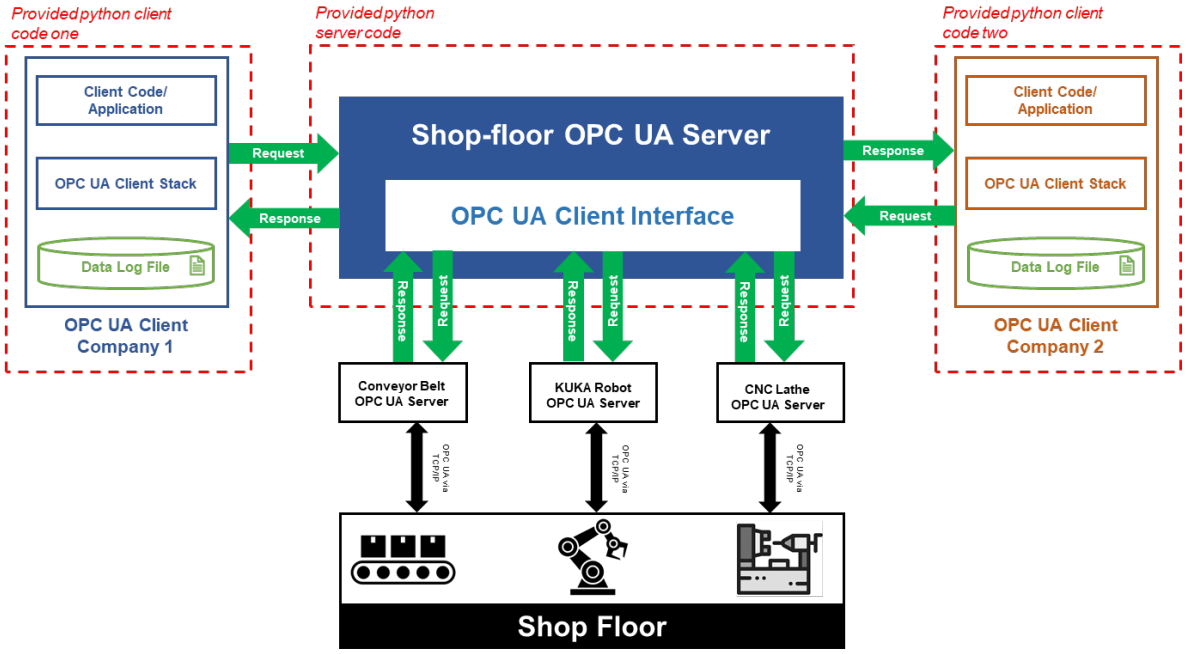

Figure 1: Server and clients communication based on OPC UA

These three pieces of machinery are working together to handle the abovementioned parts (Figure 2). Three different operations have been pre-defined for each part:

- Operation A: Conveyor Belt (transfer) → KUKA Robot (pick up & drop off) → Lathe (1. Turning; 2. Drilling)
- Operation B: Conveyor Belt (transfer) → KUKA Robot (pick up & drop off) → Lathe (1. Turning; 2. Threading)
- Operation C: Conveyor Belt (transfer) → KUKA Robot (pick up & drop off) → Inspection station

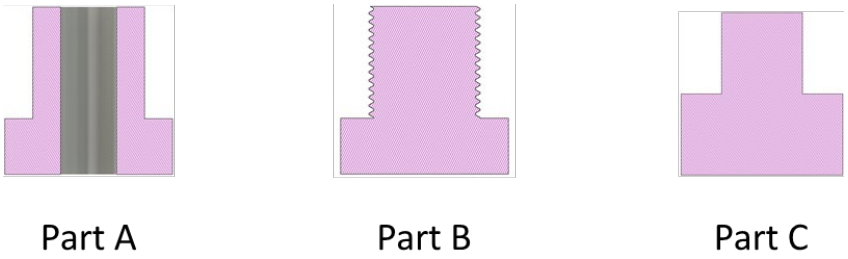

Figure 2: Parts to be produced

The OPC UA server and client codes:

You will download all the provided OPC UA python codes and a question sheet from Canvas, one for the shop-floor server, and two others for the clients. The server code is complete, but both client codes are not.

To do:

Prerequisite: Follow the instructions to set up your Python coding environment.

T1: Download your individual archived files from Canvas, put them into the same folder and open this folder in your Python IDE. (see the document "*Python_and_IDE.docx*")

T2: Answer the questions listed in the "*OPC UA assignment_question sheet.docx*". You can work on this file directly.

T3: Study and understand the codes, then complete the client codes (*Company1_Client.py* and *Company2_Client.py*). All the to-do codes have been marked with *# TODO*; insert your codes by replacing the asterisk (*).

T4: Run all the codes on your PC. You need to run the server code (*OPC_UA_Server.py*) first, then "*Company_1_Client.py*". Wait for all the operations to be DONE from Company 1 before starting "*Company_2_Client.py*".

T5: Generate a pdf version of your completed answer sheet, save your completed client code in *.py* format. You should re-name them as:
Group_{group#}_Answers.pdf
Group_{group#}_ Company1_Client.py
Group_{group#}_ Company2_Client.py

T6: Logged files.

When you successfully run the code, and all the operations from both companies have finished, you should have FOUR *.txt* files automatically generated by the program, these are the logged data from the shop-floor OPC UA server. Save all these *.txt* files into the same folder (don't make a sub-folder for them), and re-name them as:

Group_{group#}_Progress_Client_1.txt
Group_{group#}_Progress_Client_2.txt
Group_{group#}_Machine_Status_Client_1.txt
Group_{group#}_Machine_Status_Client_2.txt

Submissions:

You should put all the submission files into one folder; please do NOT make a sub-folder in your submission folder. Then, zip them into one file, i.e.

Group_{group#}_Answers.pdf *Group_{group#}_Company1_Client.py* *Group_{group#}_Company2_Client.py*	6 Marks
Group_{group#}_Progress_Client_1.txt *Group_{group#}_Progress_Client_2.txt* *Group_{group#}_Machine_Status_Client_1.txt* *Group_{group#}_Machine_Status_Client_2.txt*	4 Marks

Peer assessment:

An individual peer assessment must be submitted at the same time. It needs to have two parts,

Part 1:

Group #	Contributions to the assignment (%)
Your name	
Your group partner's name	

Part 2 (if not 50:50):

Explanations and evidence about the uneven contributions.

On Python:

There are several OPC UA libraries for different programming languages available (https://github.com/open62541/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations). Some are commercial and others are open source.

The OPC-Foundation provides a (powerful) reference implementation for C#. Unfortunately, it is very complex and has limited documentation so we will not use it. The Python-based library FreeOpcUa/python-opcua (https://github.com/FreeOpcUa/python-opcua) is much easier to use. The example/assignment code can be simple but do familiarise yourself with it (and Python language) if you have not used it before.

The assignment is designed to be completed on your own computer. For that purpose, instructions and scripts are provided. These will help you to set up your coding environment and (automatically) download all necessary software

libraries. It is recommended that a more advanced IDE be used than the one provided with the Python distribution, e.g., Visual Studio Code.

Much more information can be found on the Internet, e.g. https://www.stavros.io/tutorials/python/. Please note that you do NOT need to be a (Python-) programming expert to do this assignment.

References

1. All lecturing notes on Canvas
2. "OPC Unified Architecture" by Wolfgang Mahnke, available via the University of Auckland Library. The e-version can be downloaded, from our Libraries and Learning Services, at https://link-springer-com.ezproxy.auckland.ac.nz/book/10.1007/978-3-540-68899-0
3. "OPC Unified Architecture - Interoperability for Industrie 4.0 and the Internet of Things". OPC Foundation. Downloaded at https://opcfoundation.org/wp-content/uploads/2016/05/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN-v5.pdf
4. "OPC Unified Architecture for CNC Systems". OPC Foundation, July 2017. Can be downloaded after registration for free at, ttps://opcfoundation.org/developer-tools/specifications-unified-architecture/opc-unified-architecture-for-cnc-systems/