**Sławomir JASZCZAK**, Krzysztof BAŁAZY
WYDZIAŁ INFORMATYKI, ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY,
ul. Żołnierska 49, 71-210 Szczecin

# Wieloplatformowy klient OPC do zarządzania sterownikami programowalnymi w warstwie sterowania bezpośredniego

**Dr inż. Sławomir JASZCZAK**

Od 1994 jako asystent na Wydziale Techniki Morskiej Politechniki Szczecińskiej prowadzi badania w zakresie sterowania pojazdami głębinowymi z wykorzystaniem metod sztucznej inteligencji. Od 2002 zatrudniony na stanowisku adiunkta w Zakładzie Sztucznej Inteligencji. Bieżące zainteresowania wiążą się z implementacją złożonych algorytmów sterowania dyskretnego i cyfrowego na platformie wykonawczej PLC.

*e-mail: sjaszczak@wi.zut.edu.pl*

**Krzysztof BAŁAZY**

Student III roku kierunku Informatyka na Wydziale Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie. Aktywny członek koła naukowego "SECred" oraz kontroler CaCert. Pośród swoich zainteresowań dzieli także zamiłowanie do Linuksa oraz Wolnego i Otwartego Oprogramowania.

*e-mail: kbalazy@wi.zut.edu.pl*

**Streszczenie**

W artykule omówiono oprogramowanie typu klient OPC, przeznaczone do współpracy z serwerami OPC, działającymi w sieci Ethernet. Wykorzystano bibliotekę OpenOPC, umożliwiającą implementację funkcji klienckich, niezbędnych do współpracy serwerem OPC. Działanie aplikacji testowano w systemie, składającym się z kilku sterowników PLC, włączonych w sieć Ethernet, sterujących obiektami rzeczywistymi. Zaletą rozwiązania jest możliwość łączenia się z dowolnym urządzeniem z zainstalowanym systemem operacyjnym.

**Słowa kluczowe**: Open Process Control, PLC, OpenOPC, klient OPC.

## Multiplatform OPC client managing PLC controllers in the direct digital level

**Abstract**

In this paper, a multiplatform client software dedicated for the cooperation with OPC servers is described. The main idea is to create a lean OPC client, using an OpenOPC library, which is able to make connection with any OPC server and which is operation system - independent. Generally, the OpenOPC library enables easy implementation of client functions necessary to exchange data with a selected OPC server, like making connection, writing and reading values of OPC tagnames. A ready-made client application was tested in the system, based on several PLC controllers, connected with the Ethernet network and controlled real time plants. A significant advantage of the proposed solution is the opportunity of making connection with any OPC server installed on a device with Windows, or OSX systems. At the beginning, a logical diagram of the system (Fig. 1) with a detailed description is presented. In the next chapter, an example of the use of an implemented lean OPC client is described. There are two possible structures of control systems: a distributed (Fig. 2) and centralized (Fig. 3), which can be operated using an OPC client. Additionally, a selected real time control system i.e. “Modular servo” (Fig. 4) is explained in more detail in relation to the OPC protocol. In the final part of this paper, an example of using the developed OPC client to prepare a communication channel (Fig. 5) and exchange data (Fig. 6) between the application and a selected PLC controller is given and described.

**Keywords**: Open Process Control, PLC, OpenOPC, OPC client.

## 1. Wstęp

OPC to otwarty standard wykorzystywany w automatyce przemysłowej i w systemach biznesowych oraz zarządzania, przedsiębiorstw przemysłowych. Dzięki temu projektowane aplikacje są zdolne do efektywnej współpracy i wymiany danych maszynowych na poziomie systemu operacyjnego. OPC został tak zaprojektowany, aby łączyć aplikacje bazujące na systemach operacyjnych ogólnego stosowania (np. Microsoft Windows) ze sprzętem sterującym i oprogramowaniem aplikacyjnym, stosowanymi do nadzorowania i sterowania procesami technologicznymi.

Podstawową wadą rozwiązania jest silne uzależnienie od systemu operacyjnego. Do poprawnej pracy OPC są potrzebne dwa komponenty COM (ang. Component Object Model) oraz DCOM (ang. Distributed Component Object Model). Mechanizm COM z racji tego, że ma tylko zastosowanie lokalne został rozszerzony o dostęp zdalny, tworząc nowy mechanizm DCOM. Główną koncepcją miało być przezroczystość mechanizmów (właściwość powodująca postrzeganie systemu przez użytkownika jako całości, a nie poszczególnych składowych) i bezproblemowe użytkowanie standardu. Niestety, od kiedy technologia OLE nie jest wspierana przez Microsoft przygotowanie klienta jak i serwera jest trudne i czasochłonne. Następnym problemem w mechanizmie jest to że dane są przekazywane symetrycznie (tworzenie obiektów po obu stronach), co uniemożliwia wykorzystanie zapór sieciowych wbudowanych w system oraz instalowanych na komputerze. Zaletą jest obsługa wielu klientów przez jeden serwer OPC.

OPC jest standardem otwartym, dlatego dostępne są różne biblioteki do implementacji własnych klientów i serwerów z wykorzystaniem dowolnego języka programowania. Są to m.in. Light OPC [9], Utgard [10], czy wymieniony wcześniej OpenOPC. Opisywana wyżej wada w postaci wymaganego systemu operacyjnego może zostać wykluczona z wykorzystaniem biblioteki OpenOPC, która jest napisana w języku Python. Biblioteka ta pozwala na wykorzystanie pośrednika między klientem, a serwerem OPC. Wykorzystanie takiego pośrednika pozwala pozbyć się problemów, związanych z wykorzystaniem systemu operacyjnego jak i problemów z zaporami ogniowymi w komunikacji miedzy obiektami. Pośrednik musi mieć zainstalowany system operacyjny Windows i najlepiej, aby był podłączony do tej samej sieci w której znajduje się serwer. Na pośredniku należy zainstalować odpowiednią usługę, dostępną z biblioteki, która jest przedłużeniem klienta. Takie rozwiązanie umożliwia uniknięcie problemów z zabezpieczeniem interfejsu DCOM, jak i problemów z zaporą ogniową, gdy klient i serwer są na osobnych systemach.

## 2. Opis rozwiązania

Biblioteka do komunikacji między klientem, a serwerem wykorzystuje protokół oparty na bibliotece PyRO (ang. Python Remote Objects). Takie połączenie pozwala na zdalne operacje na obiektach, dzięki czemu klient jest niezależny od systemu. Usługa ta, jak również i sam OPC, pozwala na obsługę wielu klientów przez jednego pośrednika, a ponadto pozwala każdemu klientowi na połączenie się z dowolnym serwerem OPC dostępnym w sieci pośrednika.

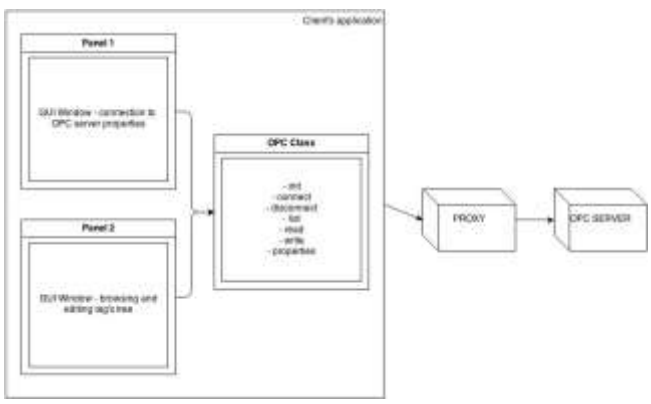
Takie rozwiązanie połączenia między systemami pozwala na zastosowanie różnego rodzaju modułów, umożliwiających łączenie się do zasobów zdalnych, ukrytych za zaporami ogniowymi. Przykładem może być sieć niedostępna dla użytkowników z zewnątrz, lecz użytkownik posiada dostęp do niej przez VPN lub

nawet przez SSH. Ruch między tymi systemami nie jest wtedy w żaden sposób ograniczany.

W oparciu o powyższe informacje został wykonany wieloplatformowy klient OPC. Klient składa się z dwóch części. Pierwszą częścią jest klient właściwy, który komunikuje się ze sterownikiem PLC z wykorzystaniem protokołu OPC. Drugą częścią, która może być uruchomiona na tym samym komputerze, na którym znajduje się klient lub na osobnym systemie. Taki zestaw pozwala na połączenie się dowolnego urządzenia z systemami Windows, Linux czy OSX z serwerem OPC, sprzężonym ze sterownikiem PLC. Dodatkowo w programie został dodany moduł do obsługi serwerów SOCKS tak, aby umożliwić zdalne łączenie do zabezpieczonych sieci.

Aby połączyć się z serwerem należy znać jego adres sieciowy oraz jego nazwę sieciową, dodatkowo należy podać adres i port sieciowy, na którym „nasłuchuje" pośrednik. Połączenie z serwerem następuje dwuetapowo, najpierw jest tworzone połączenie do bramki, a następnie po ustanowieniu połączenia, rozpoczyna się proces łączenia z serwerem. Gdy wszystkie zasoby są dostępne i nie występują problemy na połączeniach, klient uzyskuje połączenie z serwerem otrzymując możliwość edycji zmiennych.

Pośrednik w połączeniu jest przeźroczysty i w żaden sposób nie wpływa na połączenie. Takie rozwiązanie umożliwia wykorzystanie DCOM bez potrzeby używania systemu Windows na komputerze, gdzie został uruchomiony klient.



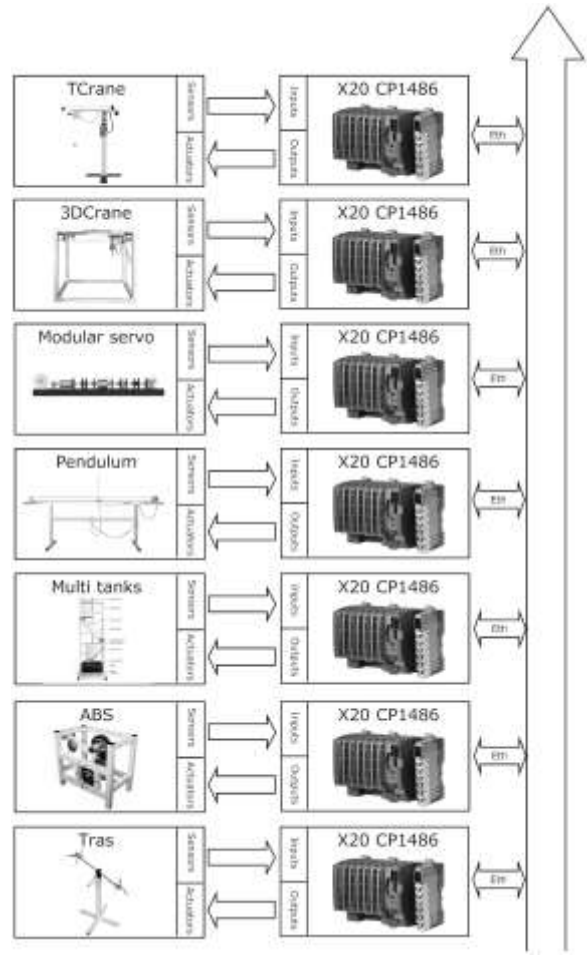

Rys. 1. Schemat logiczny systemu (Źródło: opracowanie własne)
Fig. 1. Logical diagram of the system

## 3. Przykład zastosowania

Zaprojektowaną aplikację typu klient OPC zastosowano w systemie zarządzania siecią sterowników PLC klasy X20 CP 1486 marki Bernecker&Reiner, włączonych w infrastrukturę sieciową Wydziału Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie. Zgodnie ze schematem rys. 2, sterowniki są wykorzystywane w następujących układach sterowania bezpośredniego marki Inteco: TCrane (model dźwigu towarowego), 3DCrane (model przemysłowej suwnicy transportowej), Modular servo (serwomechanizm z elementami pomiarowymi), Pendulum (model wahadła odwróconego), Multi tanks (model obiektu wielozbiornikowego), ABS (model antypoślizgowego systemu samochodowego), Tras (model dwuśmigłowego systemu aerodynamicznego).

Wymienione obiekty mechatroniczne są połączone z modułami wejście/wyjście wielu sterowników programowalnych X20CP1486, poprzez urządzenia pomiarowe i wykonawczo-nastawcze. Sterowniki programowalne pełnią funkcję indywidualnych platform wykonawczych algorytmów sterowania cyfrowego np. PID, fuzzy PID, dead beat i innych dla konkretnego obiektu.

Innym rozwiązaniem, przedstawionym na rys. 3 jest zastosowanie jednego sterownika z rozproszonymi układami wejść/wyjść. Jest to przykład scentralizowanego układu sterowania, gdzie na jednej platformie wykonawczej, realizowanych jest jednocześnie wiele algorytmów sterowania.




Rys. 2. Schemat rozproszonego układu sterowania bezpośredniego (Źródło: opracowanie własne)
Fig. 2. Diagram of the distributed direct control system

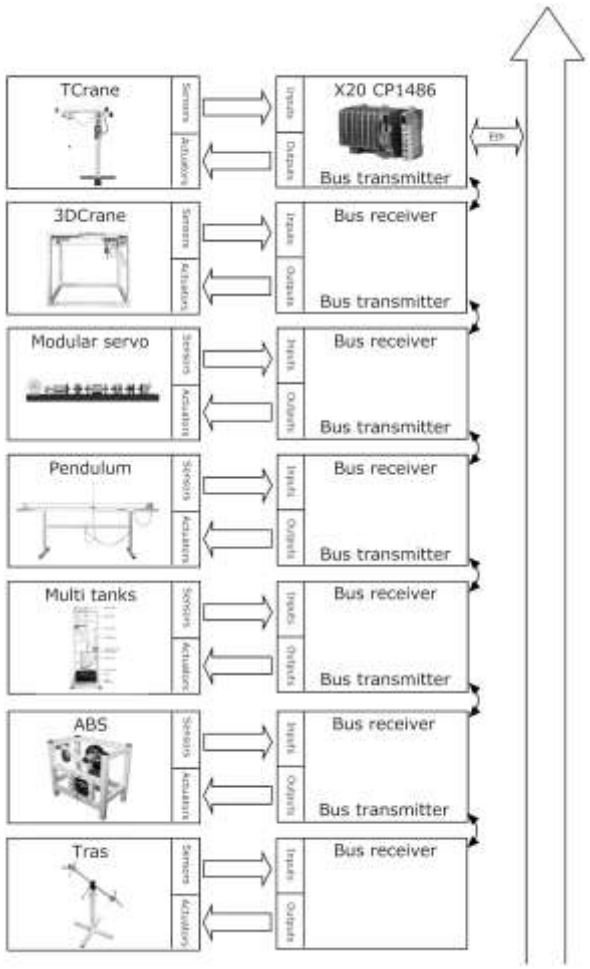
Poza realizacją algorytmów sterujących, w przedstawionym rozwiązaniu, sterowniki programowalne są również serwerami OPC, umożliwiając zdalne zarządzanie oprogramowaniem sterującym warstwy bezpośredniej. W przypadku sterowników klasy X20CP1486 można skonfigurować autonomiczny serwer OPC, dzięki czemu nie ma potrzeby uruchamiania programu komunikacyjnego na poziomie systemu operacyjnego komputera, który będzie wykorzystywany do zarządzania PLC. Na etapie konfiguracji serwera OPC istotne jest określenie listy zmiennych OPC tzw. tagów wraz z ich konfiguracją tj. typ, atrybuty write/read, zabezpieczenie hasłem itp. W rozważanym systemie przyjęto następujące grupy zmiennych OPC :

1. Zmienne reprezentujące typ algorytmu sterowania,
2. Zmienne reprezentujące obiekt (w przypadku rozwiązania scentralizowanego – rys. 3),
3. Zmienne reprezentujące parametry nastawne algorytmów cyfrowych.

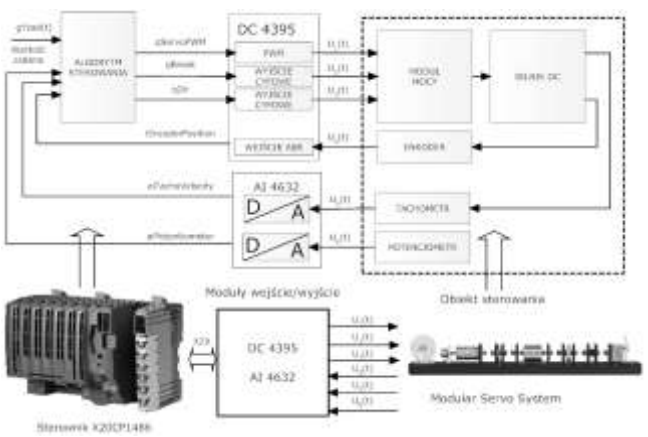
Biorąc pod uwagę możliwe struktury systemu sterującego tj. z rozproszoną siecią sterowników rys. 2 i rozproszonymi modułami wejście/wyjście rys. 3 można przedstawić dwa scenariusze kooperacji klienta OPC z systemem sterowania bezpośredniego :

1. Klient OPC łączy się indywidualnie z każdym ze sterowników i poprzez listę zmiennych umożliwia zmianę algorytmu sterowania i jego parametrów nastawnych.
2. Klient OPC łączy się z jednym sterownikiem i poprzez listę zmiennych umożliwia zmianę typu algorytmu i jego parametrów nastawnych dla wszystkich obiektów.

Kooperacja między klientem OPC, opisanym w rozdziale 2, a jednym z serwerów OPC, zostanie omówiona na przykładzie obsługi systemu sterowania serwonapędu „Modular servo", przedstawionego na rys. 4.




Rys. 3. Schemat scentralizowanego układu sterowania bezpośredniego (Źródło: opracowanie własne)
Fig. 3. Diagram of the centralized direct control system




Rys. 4. Schemat blokowy system sterowania modular servo (Źródło: opracowanie własne)
Fig. 4. Block diagram of the modular servo system

Układ „Modular servo" umożliwia badanie algorytmów sterowania cyfrowego pozycji i prędkości kątowej napędu elektrycznego prądu stałego. Cały układ składa się z kilku modułów, m.in. silnika prądu stałego, modułu tłumiącego oraz inercyjnego. Do obsługi sygnałów sterujących wykorzystano moduł DC4395 – dwa wyjścia cyfrowe do sterowania stanem hamulca (*gBreak ->OPC.Break*) oraz kierunku obrotów (*gDir –> OPC.Dir*) oraz jedno wyjście PWM (*gServoPWM->OPC.PWM*), służące do sterowania mocą silnika. Sygnał wyjściowy z enkodera (pozycja) (*iEncoderPosition->Offset.OPC*) jest również obsługiwany przez moduł licznikowy DC4395.

Aby rozpocząć zdalną obsługę obiektu rzeczywistego należy wywołać odpowiedni serwer OPC poprzez odwołanie do unikalnego adresu IP sterownika PLC (OPC Host) oraz adresu IP pośrednika (Gateway host), co pokazano na rys. 5.


Rys. 5. Połączenie z serwerem OPC poprzez unikalne IP (Źródło: opracowanie własne)
Fig. 5. Connection with an OPC server through unique IP

Po uzyskaniu listy zmiennych, zadeklarowanych wcześniej jako zmienne OPC, można dokonywać zmian ich wartości i dzięki temu wpływać na sposób wykonania programu sterującego w sterowniku PLC, co przedstawiono na rys. 6.


Rys. 6. Modyfikacja wartości wybranej zmiennej OPC (Źródło: opracowanie własne)
Fig. 6. Value Modification of the selected OPC variable

Na takiej samej zasadzie opiera się obsługa pozostałych urządzeń mechatronicznych.

## 4. Wnioski

Przedstawione rozwiązanie w postaci aplikacji typu klient OPC umożliwia swobodne zarządzanie oprogramowaniem sterującym, zaimplementowanym w sieci sterowników PLC, włączonych w sieć Ethernet. W porównaniu do typowego oprogramowania typu SCADA, możliwe jest łączenie omawianego klienta OPC z serwerami OPC, z urządzeń, wykorzystujących systemy MS Windows, Linux czy OSX. Podstawą do uzyskania powyższej niezależności systemowej było wykorzystanie otwartej biblioteki OpenOPC dla języka programowania Python. Biblioteka OpenOPC umożliwia wykorzystanie pośrednika między klientem, a serwerem OPC. Pośrednik pozwala na eliminację problemów, związanych z wykorzystaniem systemu operacyjnego, jak i problemów z zaporami ogniowymi w komunikacji między obiektami.

## 5. Literatura

[1] API OpenOPC for Python: http://openopc.sourceforge.net/api.html
[2] http://python.org/ - oficjalna strona na temat technologii Python
[3] OpenOPC for Python http://openopc.sourceforge.net/
[4] Podrecznik OPC: http://www.commsvr.com/
[5] The Free OPC Server Toolkit: http://www.ipi.ac.ru/lab43/lopc-en.html
[6] Utgard: http://openscada.org/projects/utgard/

*otrzymano / received: 22.04.2014*
*przyjęto do druku / accepted: 02.06.2014* *artykuł recenzowany / revised paper*