Sławomir JASZCZAK, Krzysztof MAŁECKI, Rafał SOKOŁOWSKI
WYDZIAŁ INFORMATYKI, ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY,
ul. Żołnierska 49, 71-210 Szczecin

Wykorzystanie OPC i Modbus w zarządzaniu siecią sterowników PLC

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

Dr inż. Krzysztof MAŁECKI

Od 1997 jako asystent w Instytucie Informatyki Politechniki Szczecińskiej prowadził badania dotyczące projektowania układów. Od 2001 zatrudniony na stanowisku adiunkta w ZUT w Szczecinie. Bieżące zainteresowania naukowe związane są z modelowaniem i symulacją ruchu drogowego. W szczególności modelowanie procesów z zastosowaniem automatów komórkowych.

e-mail: kmalecki@wi.zut.edu.pl

Streszczenie

W artykule omówiono metodykę syntezy sprzętowej i programowej sieci sterowników PLC, opierając się na standardach OPC i Modbus. Przedstawiono fizyczną warstwę komunikacji pomiędzy sterownikami z wykorzystaniem protokołu Modbus, a następnie wyjaśniono technologię OPC i jej znaczenie w pracy z urządzeniami w relacji klient-serwer.W części końcowej przedstawiono, w jaki sposób skorzystać z biblioteki OpenOPC, tak aby móc wymieniać dane pomiędzy komputerem klasy PC i sterownikiem przemysłowym.

Słowa kluczowe: Open Process Control, Modbus, sterownik PLC.

Management of the PLC controllers network using OPC and Modbus

Abstract

In this paper a methodology of the hardware and software synthesis of the PLC controllers network, based on OPC and Modbus standards, is described. The main idea is an optimization process of logical algorithms implemented in the direct digital control level using micro PLC controllers. Generally, when micro PLCs are used to control devices, there are weak opportunities to implement difficult optimization algorithms, because of the limited syntactic of the programming language and hardware limits. In such cases some supervisory system at the upper level is needed to optimize low level controllers. The authors propose using OPC technology to realize this purpose i.e. to implement an optimization application (OPC client) using a personal computer connected with low level controllers with a help of a serial RS485 network. At the beginning a logical diagram of the system. (Fig. 1) with a procedure for preparing a physical level of the PLC controllers based on RS485 standard with some Modbus implementation details in micro PLCs are described. Additionally, the physical level of communication between PLCs using Modbus (Fig. 2) protocol and the OPC technology (Fig. 3) and its importance in the interaction between devices based on the client-server relation (Figs. 4, 5) are precisely explained. In the final part of this paper a detailed example of using the developed system i.e. the way of using an OpenOPC library to prepare a communication channel and successfully exchange data between a personal computer and a master PLC controller is given and described.

Keywords: Open Process Control, Modbus, PLC.

Inż. Rafał SOKOŁOWSKI

Magistrant I roku kierunku Informatyka na Wydziale Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie. Reaktywator wydziałowego Koła Naukowego "Grupa .NET", jednocześnie aktywny uczestnik spotkań oraz pasjonat innowacyjnych technologii Microsoft. Pośród swoich zainteresowań dzieli także zamiłowanie do mikrokontrolerów oraz ich zastosowaniach w aplikacjach dedykowanych.

e-mail: rasokolowski@wi.zut.edu.pl

1. Wprowadzenie

Sterowniki PLC klasy micro są zwykle wykorzystywane do realizacji zadań sterowniczych nieskomplikowanych procesów produkcyjnych w warstwie sterowania bezpośredniego. W praktyce możliwe jest zastosowanie większej liczby tego typu sterowników i zabezpieczenie komunikacji między nimi przy wykorzystaniu typowego protokołu komunikacji np. Modbus. Jeden ze sterowników realizuje wówczas funkcje stacji nadrzędnej, pozostałe są stacjami podrzędnymi. Dzięki temu możliwa jest wymiana danych pomiędzy poszczególnymi stacjami i rozpraszanie algorytmu sterowania. Naturalne ograniczenia sterowników PLC typu micro wynikają z ograniczeń języków ich programowania np. LD lub IL i uniemożliwiają implementację skomplikowanych algorytmów optymalizacji np. autotuning algorytmu PID i inne. Rozwiązaniem tego problemu jest rozbudowa systemu sterowania o warstwę sterowania nadrzędnego, z wykorzystaniem komputera PC z serwerem OPC, umożliwiającego swobodne pobieranie i wysyłanie danych w sieci sterowników PLC. Zastosowanie protokołu OPC daje możliwość analizowania i przetwarzania danych maszynowych w celu optymalizacji algorytmów sterowania bezpośredniego.

W artykule omówiono metodykę tworzenia warstwy fizycznej sieci sterowników PLC typu micro w oparciu o standard RS485 wraz ze szczegółami implementacji protokołu Modbus/RTU w sterownikach VersaMax Micro GE..

Omówiono ponadto sposób wykorzystania technologii OPC na przykładzie komercyjnego serwera OPC KEPServerEX wraz ze sposobem wykorzystania biblioteki OpenOPC w implementacji algorytmów optymalizacji w warstwie sterowania bezpośredniego.

W rozdziale końcowym przedstawiono praktyczną aplikację z wykorzystaniem zaprojektowanej sieci sterowników.

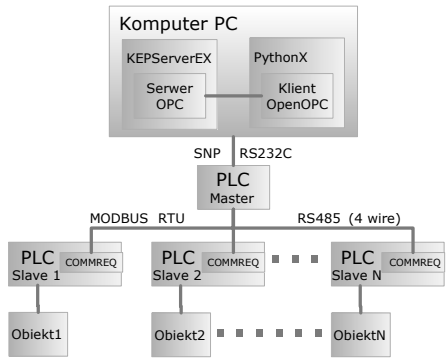
2. Synteza sprzętowo-programowa rozwiązania

Ogólny schemat fizyczny systemu zarządzania siecią sterowników PLC przedstawiono na rysunku 1.

W warstwie sterowania bezpośredniego DDC (ang Direct Digital Control) działają sterowniki klasy micro – w tym przypadku sterowniki GE VersaMax Micro, obsługujące rzeczywiste urządzenia, wykorzystując sygnały logiczne i analogowe.

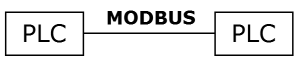
Każdy z tych sterowników, komunikując sie w protokole Modbus RTU, łączy sie ze sterownikiem nadrzędnym – VersaMax, który pełni funkcję sterownika nadrzędnego. Komunikacja w sieci sterowników wymaga użycia standardów RS-422 lub RS-485 (lub RS-232 z konwerterem na jeden RS-422 lub RS-485). Sterowniki są podłączane ze sobą szeregowo. Do wykorzystania rozwiązania w polaczeniem 2-przewodowym [4], przewód w postaci skrętki ekranowanej sprawdza sie na magistralach w odległości do 15 metrów pomiędzy odbiorcami [7]. Sterownik nadrzędny (Master)

musi sie znajdować na jednym z końców magistrali. W całej sieci może znajdować sie do 247 urządzeń [3].



Rys. 1. Schemat logiczny systemu
Fig. 1. Logical diagram of the system

Protokół połączenia szeregowego Modbus, jest otwartym standardem komunikacji służącym do przesyłu danych pomiędzy sterownikami PLC i urządzeniami pokrewnymi. Szeregowy standard Modbus zezwala na komunikacje używając wyłącznie znaków alfanumerycznych (Modbus ASCII) lub wyłącznie danych binarnych (Modbus RTU) [4]. Schemat połączenia pomiędzy urządzeniami został zaprezentowany na rysunku 2.



Rys. 2. Schemat połączenia w protokole Modbus
Fig. 2. Connection diagram in the Modbus protocol

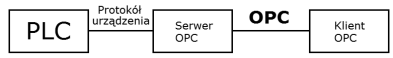
Protokół Modbus, powszechnie używany w praktyce przemysłowej, działa na zasadzie Master- Slave. Sterownik nadrzędny (Master) kieruje zapytania do jednego lub więcej sterowników podrzędnych (Slave) [4]. Takim zapytaniem może być także polecenie do zapisu danych lub wymuszenie określonej akcji w sterowniku podrzędnym. Każde zapytanie może być kierowane do konkretnego odbiorcy lub wszystkich jako transmisja rozgłoszeniowa (ang. broadcast) [4]. Każdy odbiorca, który otrzyma zapytanie musi na nie odpowiedzieć, dopiero po pełnej wymianie komunikatów transmisję uznaje sie za zakończoną pomyślnie.

Wyjątkiem jest transmisja rozgłoszeniowa, gdzie żaden ze podrzędnych nie odpowiada na zapytanie a sterownik nadrzędny musi chwile odczekać zanim następne zapytanie zostanie sformułowane [4].

Program działający na sterowniku PLC kontroluje zapytania poprzez wysyłanie komunikatów poleceniem COMMREQ, którego działanie objaśniono w [6]. Polecenia COMMREQ musza być adresowane do odpowiednio skonfigurowanego portu szeregowego, który jest podłączony do sieci szeregowej Modbus RTU [4]. Zarówno port nadawcy jak i odbiorcy musza być odpowiednio skonfigurowane.

OPC (ang. OLE for Process Control) zgodnie z [7], jest otwartym standardem komunikacyjnym stosowanym w automatyce przemysłowej i informatycznych systemach wyższych warstw. OPC powstał i został tak zaprojektowany, aby łączyć aplikacje bazujące na systemach operacyjnych ogólnego stosowania (np. Windows) ze sprzętem i oprogramowaniem aplikacyjnym automatyki przemysłowej (urządzenia procesowe), nadzorującym i sterującym procesem technologicznym. Jest to otwarty standard komunikacji, który pozwala używać jednolitych metod dostępu i opisu danych (interfejsu) dla procesu technologicznego. Metody te są niezależne od typu oraz źródła danych.

Jak widać na rysunku 3, serwer OPC umożliwia utworzenie kanału komunikacji sterownik PLC - komputer PC, w tym wypadku klient OPC, reprezentujący stronę komputera PC. W tym wypadku klient OPC reprezentuje stronę komputera PC.



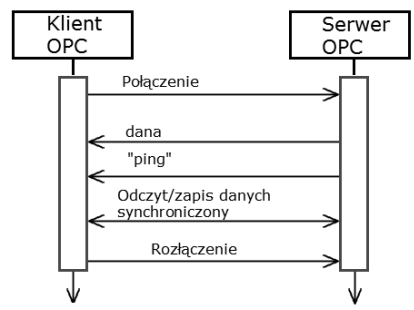
Rys. 3. Sposób wykorzystania protokołu OPC
Fig. 3. The way of using the OPC protocol

Standard ten wymaga platformy DCOM zarówno po stronie klienta, jak i serwera [7]. W przypadku systemów operacyjnych Microsoft Windows powyższy warunek nie stanowi przeszkody, ponieważ ten komponent jest domyślnym składnikiem systemu. Informacja o używanym protokole do komunikacji pomiędzy serwerem OPC a sterownikiem przemysłowym, jest nieistotna z punktu widzenia użytkownika. Co więcej, klient OPC może działać na innym komputerze niż serwer. Dzięki temu rozwiązania wykorzystujące standard OPC mogą być rozproszone w sieci zarówno lokalnej jak i Internecie.

Jak wspomniano w [7], OPC implementuje typowa architekturę klient-serwer, w której klient jest odpowiedzialny za zarządzanie relacjami z serwerem.

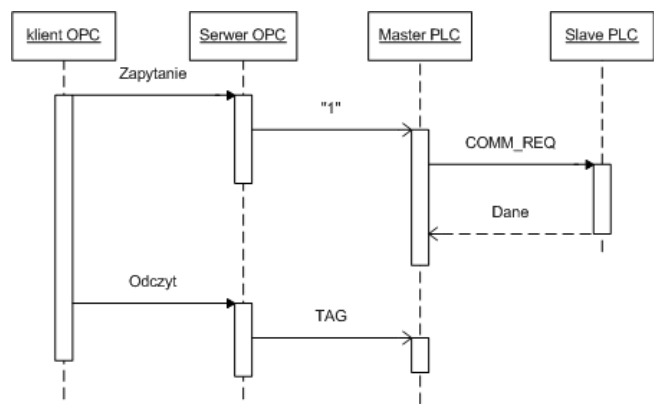
Na rysunku 4 zaprezentowano uproszczony schemat w postaci diagramu sekwencji, który objaśnia komunikację klienta OPC z serwerem OPC. Przykładem takiego serwera może być aplikacja KEPServerEX.4, który gwarantuje dwugodzinną, pełną funkcjonalność w trybie demonstracyjnym.

Jak widać każde zadanie wynika z inicjatywy klienta, natomiast zadaniem serwera jest tylko na nie odpowiadać.



Rys. 4. Współpraca pomiędzy klientem OPC a serwerem OPC
Fig. 4. Interaction between OPC client and OPC Server

Na komputerze, na którym uruchomiono serwer OPC jest jednocześnie uruchomiona instancja klienta OPC która odpytuje serwer o wartości zapisane w fizycznej pamięci sterowników PLC. Program napisany w języku Python z instancją klienta OPC realizuje algorytm optymalizacji na podstawie danych z urządzeń rzeczywistych. Na rysunku 5 przedstawiono Uproszczony diagram interakcji pomiędzy poszczególnymi elementami systemu zarządzania.



Rys. 5. Uproszczony diagram interakcji pomiędzy poszczególnymi elementami systemu zarządzania.
Fig. 5. Simplified diagram of the interaction between elements of the management system

3. Przykład aplikacji

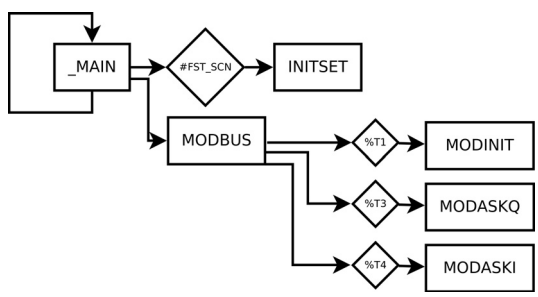
Przykładowe zastosowanie projektowanego systemu może stanowić realizacja algorytmów nadzorujących (ang. supervisory control) algorytmy sterowania cyfrowego, działających w warstwie bezpośredniej. Implementacja tego typu algorytmów w sterownikach klasy micro jest praktycznie niemożliwa ze względu na ograniczenia języka programowania (głównie LD i IL) oraz sprzętowe (ograniczona pamięć programu i szybkość jednostki centralnej).

Innym równie ważnym obszarem zastosowań przedstawionego systemu jest optymalizacja algorytmów sterowania sekwencyjnego, w których wykorzystywane są funkcje zegarowe. Realizacja poszczególnych etapów sekwencji może być uwarunkowana zdarzeniowo i/lub czasowo. We wstępnej fazie implementacji algorytmów sekwencyjnych zazwyczaj wykorzystywane są funkcje zegarowe stałoczasowe, co w praktyce może oznaczać nieoptymalną w sensie wydajności, pracę określonego urządzenia. Z tego powodu w trakcie normalnej pracy urządzenia dokonywane są korekty stałych czasowych tak, aby optymalizować czas wykonania danego ciągu czynności.

W przypadku zastosowania komputera nadrzędnego z uruchomionym serwerem i klientem OPC, możliwe jest zautomatyzowanie procesu modyfikacji wartości stałych czasowych.

Poniżej omówiono przykład realizacji tak postawionego zadania z wykorzystaniem sieci szeregowej sterowników PLC w oparciu o protokół ModbusRTU oraz komputer PC z uruchomionym serwerem OPC, zapewniający wymianę danych pomiędzy sterownikiem nadrzędnym Modbus a aplikacją klient OPC (rys. 1), odpowiedzialną za wyznaczanie optymalnych nastaw funkcji zegarowych.

Komunikacja sterownika nadrzędnego ze sterownikami podrzędnymi w programie polega na odpytywaniu wejść i wyjść sterowników o kolejnych wartościach adresu (”COMMREQ slave address” %R28 o wartości od 2 do ”COMMREQ slave max address” %R29). Schemat działania programu sterownika nadrzędnego przedstawia rys. 6.



Rys. 6. Schemat działania programu sterownika nadrzędnego
Fig. 6. Operating flowchart of the controller's type master programme

Podprogram INITSET wywoływany jest w pierwszej iteracji po uruchomieniu sterownika (#FST SCN). Podprogram Modbus zapewnia wywoływanie podprogramów MODINIT, MODASKI i MODASKQ w odpowiedniej kolejności i z odpowiednimi wartościami adresu sterownika podrzędnego oraz ustawia adresy pamięci %M sterownika nadrzędnego, w którym zostaną zapisane stany wejść/wyjść.

Podprogram LD MODASKI odpytuje sterowniki podrzędne o wartości ich wejść w obszarze adresowym %I.. Pomiędzy zapytaniami wysyłanymi do sieci Modbus RTU musi występować 2-sekundowa przerwa, dlatego przed wywołaniem bloku COMMREQ, licznik czasu odmierza założony czas.

Aby móc swobodnie wymieniać dane między oprogramowaniem optymalizacyjnym a sterownikiem nadrzędnym konieczne jest odpowiednie przygotowanie środowiska. W tym celu należy przeprowadzić instalacje, niżej wymienionych elementów, na komputerze klasy PC z zainstalowanym systemem Microsoft Windows : Python, OpenOPC oraz dowolny serwer OPC, w tym przypadku : ’Kepware.KEPServerEX.V4.

Przykładową procedurę podłączenia do serwera OPC ’Kepware. KEPServerEX.V4’ pokazano w listingu 1. W zaprezentowanym fragmencie kodu zaimportowano bibliotekę OpenOPC. Przy pomocy utworzonej instancji klasy klienta, nawiązano połączenie i odczytano daną maszynową, pochodzącą ze sterownika nadrzędnego. Po udanej wymianie informacji połączenie zostało zamknięte, natomiast instancja wciąż pozostaje. Taka sytuacja pozwala na nawiązanie polaczenia z innym serwerem, bez konieczności alokowania dodatkowej pamięci programu. Polecenia z listingu 1 zostały poniżej dokładniej omówione:

```
>>> import OpenOPC
>>> opc = OpenOPC.client()
>>> opc.connect('Kepware.KEPServerEX.V4')
>>> print opc['SlaveARed.Int16']
(19169, 'Good', '06/24/07 15:56:11')
>>> opc.close()
```

Listing.1. Procedura odpytania serwera OPC o zmienną całkowitą w języku Python z wykorzystaniem OpenOPC
Listing. 1. A querying procedure of the OPC server about an integer variable in the Python language using an OpenOPC
Źródło: opracowanie własne

Pierwsze polecenie listingu jest standardową składnią języka Python, służąca do importowania bibliotek zewnętrznych do programu, w tym wypadku biblioteki OpenOPC.

Pierwszym krokiem do nawiązania połączenia jest stworzenie instancji klienckiej. Można tego dokonać za pomocą powyższego polecenia. Jak widać instancja ta zostaje przypisana do zmiennej opc.

Następnie, odwołując się do metody instancji klasy, nawiązuje się połączenie z serwerem, którego nazwę należy podać jako argument w postaci łańcucha znaków. W tym wypadku nazwa serwera to ’Kepware.KEPServerEX.V4’. Po pomyślnym nawiązaniu połączenia zostanie zwrócona wartość ‘True’.

Oznaczoną wartość na serwerze OPC można odczytać na kilka sposobów. Jeden z nich został zaprezentowany powyżej. Mianowicie odwołując się do instancji klasy poprzez nawiasy kwadratowe tak, jakby to była tablica znaków.

Aby zakończyć połączenie z serwerem należy wywołać metodę ‘close’. Takie działanie pozwala na nawiązanie kolejnego połączenia za pomocą tej samej instancji, nie alokując przy tym dodatkowej pamięci.

4. Wnioski

Wykorzystanie standardu OPC wraz z aplikację typu klient OPC, stworzonej przy użyciu otwartej biblioteki OpenOPC dla języka programowania Python, umożliwia swobodne zarządzanie siecią sterowników przemysłowych PLC, połączonych w standardzie szeregowym i wymieniających dane zgodnie z protokołem Modbus RTU.

5. Literatura

[1] API OpenOPC for Python: http://openopc.sourceforge.net/api.html
[2] http://python.org/ - oficjalna strona na temat technologii Python
[3] KEPServerEX Kepware R technologies: http://www.kepware.com/Products/kepserverex features.asp
[4] Modbus RTU Communications, GFK-2220C ,Suplement GFK-0582, GE Fanuc Automotion, 2003.
[5] OpenOPC for Python http://openopc.sourceforge.net/
[6] Podręcznik użytkownika, Sterowniki programowalne VersaMax Micro i Nano, GFK-1645C - PL, GE Fanuc Automotion, 2002.
[7] Podręcznik OPC: http://www.commsvr.com/

otrzymano / received: 26.04.2012
przyjęto do druku / accepted: 01.06.2012 *artykuł recenzowany / revised paper*