Zintegrowane środowisko projektowania i testowania regulatorów do sterowania rozproszonego

Aleksander Simicz

We współczesnych rozwiązaniach przemysłowych istotnym zagadnieniem jest konieczność współpracy urządzeń należących do różnych producentów. Środowisko sterowania rozproszonego składa się m.in. ze sterowników przemysłowych, których zadaniem jest akwizycja danych pomiarowych oraz kontrola położenia urządzenia wykonawczego, komputera nadrzędnego sterującego procesem, przewodowego lub bezprzewodowego medium komunikacyjnego transmitującego dane (można wykorzystać jeden lub więcej protokołów komunikacyjnych) oraz ewentualnie bazy danych archiwizującej proces. W celu ujednolicenia styku pomiędzy urządzeniami powstała fundacja OPC Foundation [1] licząca 300 członków na całym świecie. Członkami OPC są wiodący dystrybutorzy systemów sterowania i monitorowania oraz firma Microsoft. Fundacja ta wprowadziła standard OPC (*OLE for Process Control*), będący wdrożeniem stworzonej przez Microsoft technologii OLE (*Object Linking and Embedding*) do zastosowań w przemyśle. W 1996 roku powstała pierwsza specyfikacja tego standardu.

W artykule pokazano przykład zintegrowanego środowiska sterowania rozproszonego układem zbiorników. Stanowisko to znajduje się w Katedrze Automatyki w Akademii Górniczo–Hutniczej w Krakowie. Sterowanie odbywało się za pośrednictwem sieci telefonii komórkowej GPRS. Wykorzystano standardy: MATLAB, OPC i Modbus. Problem ten został obszerniej przedstawiony w pracy [2].

Koncepcja środowiska

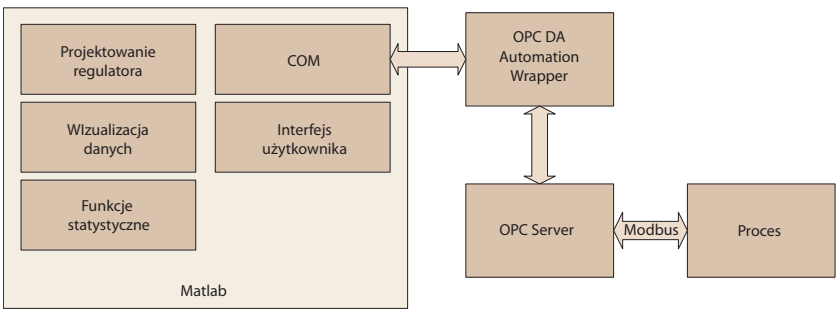
Środowisko uruchomieniowe regulatorów rozproszonych powinno mieć następującą funkcjonalność:

- narzędzia wspomagające projektowanie algorytmów sterowania i optymalizacji – w celu syntezy najlepszego regulatora dla danych zastosowań
- funkcje statystyczne – badanie właściwości medium komunikacyjnego
- obróbka i przetwarzanie danych pomiarowych
- interfejs użytkownika – funkcje pozwalające na uruchomienie aplikacji z wybranymi parametrami oraz wizualizację wyników
- interfejs sprzętowy – pozwalający na komunikację z urządzeniem fizycznym.

Środowisko, które spełnia powyższe wymagania nosi nazwę CACSD (*Computer Aided Control System Design*) – komputerowo wspomagane projektowanie systemów sterowania [3]. Rys. 1 przedstawia zaproponowana architekturę takiego systemu.

MATLAB pełni tu rolę środowiska CACSD. Ma on liczne funkcje pozwalające na syntezę, symulację i uruchomienie regulatorów. Dostęp do danych procesowych umożliwia, zdobywająca coraz większą popularność, technologia OPC. OPC Server udostępnia zasoby sprzętowe urządzeń powiązanych z procesem (sterowników przemysłowych), za pomocą obiektów COM. Popularnym protokołem przemysłowym służącym do komunikacji ze sterownikami jest Modbus. Aby komunikować się z OPC Serverem, należy zaimplementować w środowisku uruchomieniowym OPC Clienta. W MATLAB-ie można wykorzystać do tego celu specjalizowaną bibliotekę OPC Toolbox [4]. Niestety jest to rozwiązanie kosztowne. Indywidualna licencja komercyjna kosztuje 1000 USD (według danych z lutego 2006). Innym rozwiązaniem jest wykorzystanie darmowej biblioteki nakładkowej OPC Automation Wrapper, z którą można się komunikować dzięki standardowym funkcjom COM dostępnym w MATLAB-ie.

Standard OPC jest wykorzystywany również w systemie iFIX. Jest to jednakże system klasy SCADA, przystosowany do nadzorowania procesów przemysłowych, o znacznie mniejszych możliwościach w zakresie syn-



Rys. 1. Architektura środowiska

Aleksander Simicz – Katedra Automatyki, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Akademia Górniczo-Hutnicza, Kraków

tezy regulatora i pracy w trybie *off line* niż MATLAB, który jest typowym narzędziem CACSD. IFIX ponadto nie pozwala na tak dużą rozdzielczość czasową wykresów jak MATLAB.

Modbus

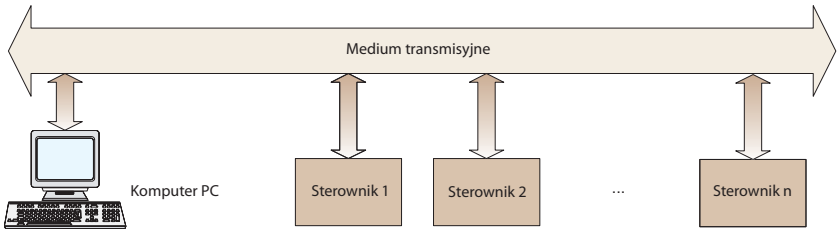
Modbus [5] jest protokołem typu *master-slave*, stosowanym w urządzeniach przemysłowych, pierwotnie przeznaczonym dla sterowników logicznych firmy Modicon. Komputer (najczęściej PC) jest węzłem nadrzędnym (*master*), sterującym pracą sieci o topologii magistrali, natomiast pozostałe węzły są węzłami podrzędnymi (slaves). Na rys. 2 jest przedstawiona architektura sieci, w której zastosowano protokół Modbus.

Wprawdzie standard ten został opracowany w latach 70. dla sieci sterowników firmy Modicon, ale jest on nadal popularny w systemach SCADA. Polecenia użyte w specyfikacji służą do wykonania operacji na zasobach tych sterowników – wejściach i wyjściach: cyfrowych, analogowych, licznikowych oraz rejestrach ogólnego przeznaczenia.

Węzły podrzędne w sieci Modbus są jednoznacznie identyfikowane przez adres. Węzeł nadrzędny może wysłać komunikat do pojedynczego węzła, lub do wszystkich węzłów (*broadcast*). Po otrzymaniu indywidualnego komunikatu sterownik wysyła odpowiedź. Taki cykl jest nazywany transakcją i jest przedstawiony na rys. 3. [6].

Komunikaty zawierające polecenia i odpowiedzi mają identyczną strukturę. Część informacyjna ramki składa się z trzech pól:

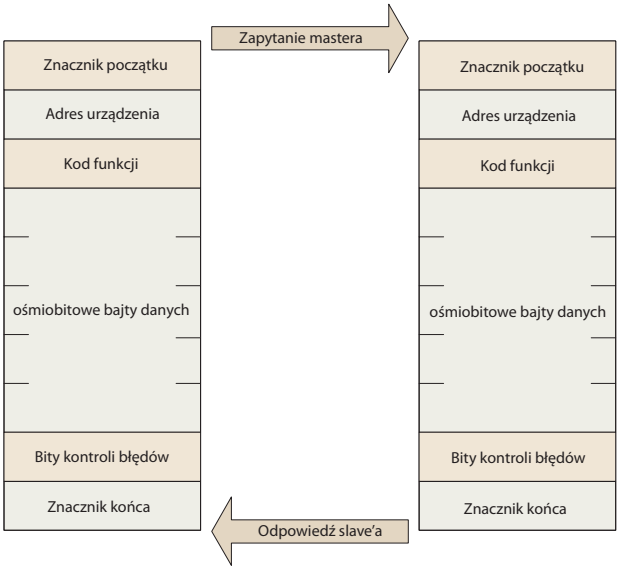
1) adres węzła podrzędnego(długość 1 bajt) – liczba z zakresu 1...247, adres 0 jest adresem *broadcastowym*. Ramka będąca odpowiedzią z węzła podrzędnego zawiera adres tego węzła
2) kod polecenia (długość 1 bajt) – funkcja (liczba z zakresu 1...127) jaką ma wykonać węzeł podrzędny, na przykład, może to być: odczyt lub zapis rejestru lub grupy rejestrów, wejść lub wyjść, odczyt zawartości rejestrów statusowych. W odpowiedzi pole to jest wykorzystywane do pozytywnego lub negatywnego potwierdzenia wykonania polecenia. W wypadku pozytywnego wykonania operacji, węzeł podrzędny umieszcza w tym polu kod polecenia. W wypadku błędów, dodatkowo najstarszy bit tego pola jest ustawiony na wartość 1. Jeżeli węzeł podrzędny nie może wykonać polecenia (*exception response*), to ponadto w polu danych zostaje umieszczony kod błędu, co umożliwia węzłowi nadrzędnemu określenie przyczyny jego wystąpienia
3) pole danych (długość zmienna, może być zerowa) – argumenty poleceń przesyłane przez węzeł nadrzędny. W odpowiedzi pole to może zawierać żądane przez węzeł nadrzędny dane lub kod błędu, jeżeli jest to odpowiedź szczególna (*exception response*).



Rys. 2. Architektura sieci pracującej zgodnie z protokołem Modbus

W protokole Modbus stosowane są dwa tryby transmisji ramek:

1) ASCII – tryb znakowy
2) RTU – tryb binarny.



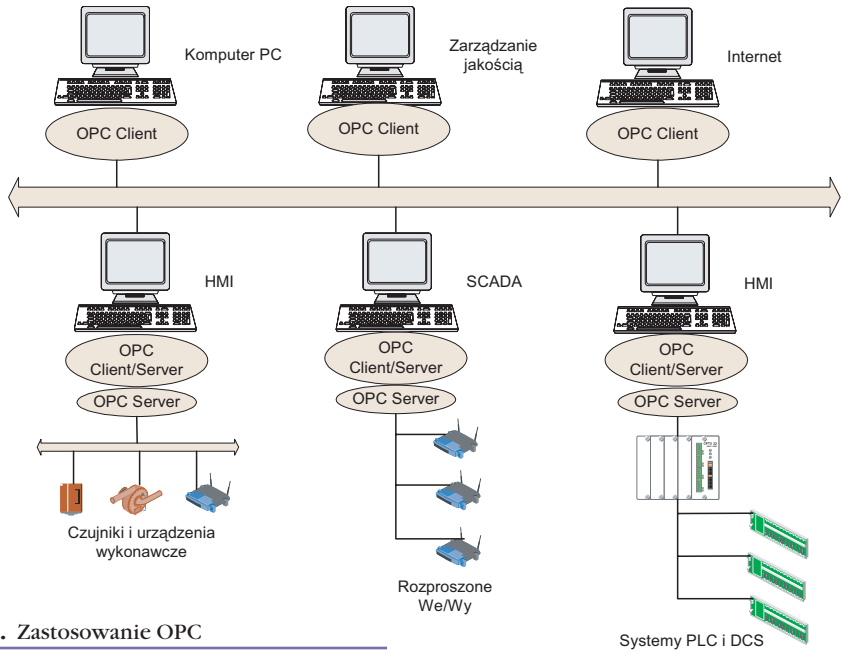
Rys. 3. Transakcja w protokole Modbus

OPC

OPC (*OLE for Process Control*) [7] jest standardem przemysłowym wymiany danych. OPC bazuje na architekturze klient–serwer, którzy wymieniają dane ze sobą za pomocą technologii COM/DCOM. Klient i serwer mogą pracować na tej samej maszynie lub też na różnych maszynach. Przykład systemu wykorzystującego OPC jest przedstawiony na rys. 4.

1. **OPC Server** – moduł, służący do wymiany danych pomiędzy aplikacjami a procesem. OPC Server jest dostosowany do konkretnego sterownika przemysłowego, pozwala na dostęp do jego zasobów za pomocą interfejsu OPC.
2. **OPC Client** – aplikacja pozwalająca na dostęp do zasobów dowolnego serwera OPC z poziomu aplikacji klienta napisanej w języku programowania ogólnego przeznaczenia (C, C++, Visual Basic). OPC Client może być modułem pozwalającym aplikacjom MS Office (np. Excel) korzystać z danych OPC.

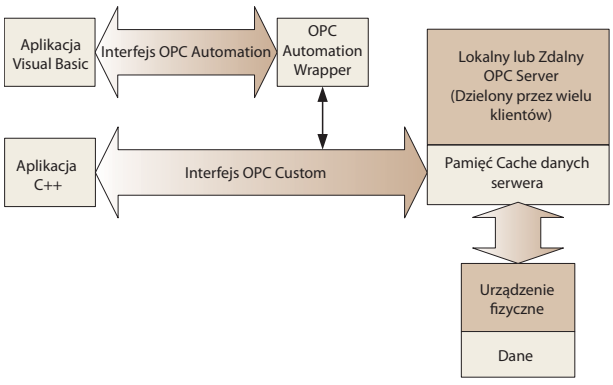
Aplikacja klienta komunikuje się z serwerem OPC poprzez interfejsy typu *custom* i *automation* [8]. Interfejs custom musi być zawsze zaimplementowany przez serwer OPC, natomiast interfejs automation jest opcjonalny. Zastosowanie standardowej biblioteki nakładkowej wrapper pozwala na utworzenie interfejsu automation. Interfejs *custom* [9] jest stosowany dla języków mających wskaźniki do funkcji (C, C++). Interfejs *automation* jest stosowany dla innych języków (Visual Basic, MATLAB). Użycie tych interfejsów ilustruje rys. 5 [8].



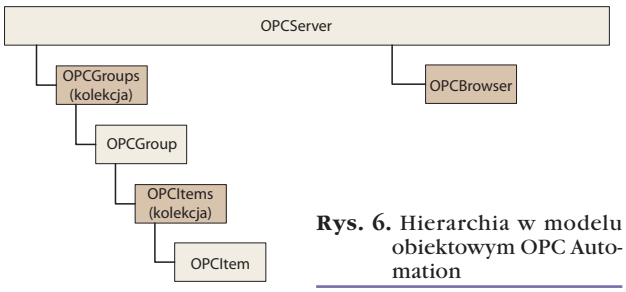
Rys. 4. Zastosowanie OPC

Zasoby serwera OPC mają hierarchię zilustrowaną na rys. 6 [10]. Aplikacja zewnętrzna napisana w języku skryptowym np. w MATLAB-ie lub Visual Basicu może dokonywać na nich operacji zapisu/odczytu (read/write). Są to następujące obiekty:

1) **OPCServer** – należy stworzyć instancję tego obiektu, aby odnieść się do innych obiektów. Zawiera kolekcję OPCGroups Collection oraz obiekt OPCBrowser. Aplikacja tworzy ten obiekt za pomocą metody Connect (w Visual Basicu jest to metoda OPCServer.Connect)



Rys. 5. Typowa architektura OPC



Rys. 6. Hierarchia w modelu obiektowym OPC Automation

2) **OPCGroups** – kolekcja zawierająca wszystkie obiekty typu OPCGroup, które klient stworzył, będąc w obrębie obiektu OPCServer

3) **OPCGroup** – instancja obiektu tego typu jest tworzona w celu otrzymywania informacji o stanie oraz do obsługi procesu akwizycji danych dla kolekcji OPCItem, na którą wskazuje

4) **OPCItems** – kolekcja zawierająca wszystkie obiekty OPCItem stworzone przez klienta w ramach obiektu OPCServer i odpowiadające obiektowi OPCGroup, dla którego została stworzona aplikacja

5) **OPCItem** – obiekt przechowujący dane odnośnie do definicji obiektu terminalnego (np. rejestru, zmiennej, wejścia/wyjścia analogowego/cyfrowego), aktualnej wartości, statusu, czasu ostatniej aktualizacji

6) **OPCBrowser** – przeglądarka nazw obiektów typu OPCItem występujących w serwerze OPC. Istnieje tylko jedna instancja obiektu OPCBrowser dla instancji obiektu OPC Server.

MATLAB

MATLAB jest popularnym oprogramowaniem typu CACSD. Umożliwia dokonywanie obliczeń, wizualizacji, oraz programowanie w łatwym w użyciu środowisku, z wykorzystaniem intuicyjnej notacji matematycznej [4], [11]. Typowymi zastosowaniami są:

- obliczenia matematyczne
- symulacja i prototypowanie
- analiza, eksploracja i wizualizacja danych
- tworzenie aplikacji z możliwością budowy graficznego interfejsu użytkownika
- akwizycja danych pomiarowych
- implementowanie algorytmów w czasie rzeczywistym.

MATLAB jest interaktywnym systemem, którego podstawowym elementem jest bezwymiarowa tablica danych. Pozwala to na rozwiązywanie problemów,

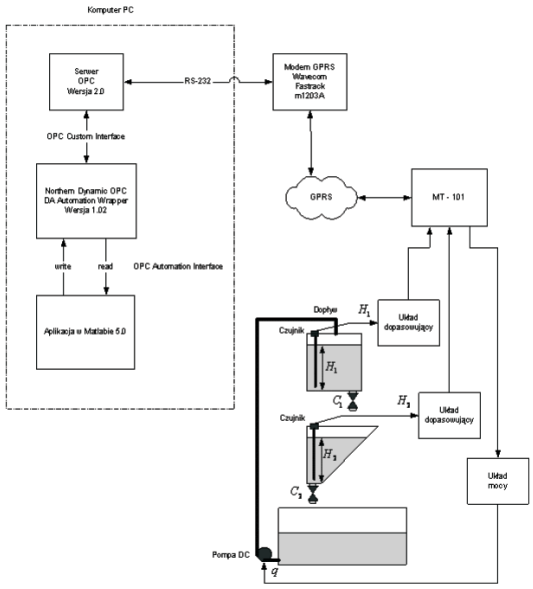
w których występują wektory i macierze. W ciągu wielu lat MATLAB ewoluował, dostosowując się do swoich użytkowników. W ośrodkach akademickich jest wykorzystywany do nauki podstawowych i zaawansowanych zagadnień naukowo-inżynieryjnych. W przemyśle stał się narzędziem umożliwiającym zarówno teoretyczne, jak i praktyczne badania nad różnymi algorytmami sterowania, co pozwala poprawić wydajność produkcji. Ważną cechą MATLAB-a jest modularność. System składa się z kolekcji kompletnych zestawów funkcji (m-plików), pozwalających na rozszerzenie środowiska MATLAB-a w celu rozwiązywania określonej klasy problemów. Kolekcje te noszą nazwę *toolboxów*. Na przykład, istnieją odrębne przyborniki (*toolboxy*) dla takich dziedzin, jak: przetwarzanie sygnałów, sterowanie, sieci neuronowe, logika rozmyta, falki, oraz wiele innych.

System MATLAB składa się z pięciu części:

- środowisko projektowe – zestaw narzędzi i pomocy ułatwiający korzystanie z plików i funkcji MATLAB-a, zawiera okno linii poleceń, historię komend, edytor z *debuggerem* oraz przeglądarki do wyświetlania pomocy, przestrzeni roboczej, plików oraz ścieżki przeszukiwania
- biblioteki matematyczne MATLAB-a – rozległa kolekcja algorytmów obliczeniowych mająca podstawowe funkcje jak: suma, sinus, kosinus, arytmetyka liczb zespolonych, oraz bardziej zaawansowane: odwracanie macierzy, obliczanie wartości własnych, funkcje Bessela i szybką transformatę Fouriera
- język MATLAB – jest to wysokopoziomowy tablicowo-macierzowy język programowania; ma funkcje, struktury danych, obsługę wejścia/wyjścia oraz cechy obiektowe; pozwala na pisanie zarówno krótkich skryptów, jak i bardziej złożonych aplikacji
- grafika – MATLAB pozwala prezentować w dogodny sposób wektory i macierze jako postaci wykresy, które można swobodnie opisywać, drukować i kopiować do innych aplikacji (np. Paint albo Word). Są dostępne funkcje wysokiego poziomu do wizualizacji danych dwu- i trójwymiarowych, przetwarzania obrazów, animacji oraz niskiego poziomu do ustalenia parametrów wykresu oraz tworzenia graficznego interfejsu użytkownika
- interfejs programistyczny MATLAB-a (API) – biblioteka pozwalająca na pisanie programów w języku C i Fortranie. Umożliwia wywołanie procedur z poziomu MATLAB-a (dynamic linking), wywoływanie MATLAB-a jako środowiska uruchomieniowego oraz zapis i odczyt MAT-plików.

Przykład zintegrowanego systemu sterowania

Na rys. 7 jest przedstawiony układ laboratoryjny służący do sterowania systemem zbiorników z wodą, wykorzystujący technologię GPRS. Moduł telemetryczny MT–101 firmy AB-MiCRO jest sterownikiem przemysłowym, mającym kartę SIM. Moduł ten jest używany w celu odczytania informacji o poziomach wody w dwóch



Rys. 7. Budowa stanowiska laboratoryjnego sterowania rozproszonego

górnych zbiornikach oraz do sterowania pompą. Układ MT–101 komunikuje się z komputerem nadrzędnym za pomocą ramek protokołu Modbus przesyłanych przez sieć telefonii komórkowej. Dokładniejsze omówienie tego zagadnienia, sposób uruchomienia stanowiska, znajduje się w [2]. Informacje dotyczące układu telemetrycznego i jego zasobów są zawarte w [12 i 13].

Przykładowa konfiguracja serwera OPC

Konfiguracja serwera OPC integrującego komponenty systemu z rys. 7 wymaga napisania pliku XML. Wszystkie informacje przeznaczone dla serwera OPC muszą się znajdować w obrębie znacznika <opc>. W obrębie tego znacznika występują dwa znaczniki [14]:

(1) <configure> – globalne ustawienia serwera. Parametry znacznika są następujące:
   (a) *internet_mode* – określający tryb pracy drajwera (zwykła praca przez Internet lub praca przez Internet z wykorzystaniem programu łączącego Internet z APN-em)
   (b) *csv_path* – określający ścieżkę gdzie składowane będą pliki CSV
   (c) *udp_port* – *port*, na którym pracuje serwer
   (d) *timestamp* – informuje jakim stemplem czasowym mają być opatrzone dane
   (e) *debug* – opcja zapisu do pliku całej komunikacji serwera

(2) <network> – definiuje pojedyncze urządzenie MT (moduł telemetryczny) w APN-ie. Parametry znacznika są następujące:
   (a) *name* – dowolna unikalna nazwa urządzenia (domyślnie jest to numer IP)
   (b) *ip_receiver* – adres IP, pod który będą wysyłane ramki (jest to adres IP modułu MT - 101)
   (c) *udp_port* – port udp, pod który będą wysyłane ramki

(d) *ip_header_receiver* – IP modułu telemetrycznego MT (przy internet_modec = x"0" taki jak <ip_receiver>)
(e) *ip_header_sender* – IP nadawcy (jest to adres IP modemu GPRS)
(f) *timeout* – timeout transmisji w sekundach
(g) *retries* – liczba powtórzeń (przy wystąpieniu timeoutu)
(h) *add_crc* – parametr przyjmuje następujące wartości w zależności od wersji firmware
(i) *unsolicited_status* – wybór mapy pamięci w zależności od typu modułu
(i) *debug* – opcja zapisu komunikacji do pliku
(j) *enable* – określa czy znacznik jest aktywny
(k) w obrębie znaczników <network> występują znaczniki <modbus> definiujące konkretne pojedyncze zapytanie o zasoby za pomocą protokołu Modbus. Jako argumenty przyjmują następujące parametry:
  (i) *name* – nazwa zasobu lub urządzenia typu slave podłączonego do MT
  (ii) *id - adres* Modbus urządzenia MT lub urządzenia typu slave podłączonego do MT
  (iii) *type* – typ zasobu Modbus:
  1. "binary_inputs" – wejścia binarne
  2. "binary_outputs" – wyjścia binarne
  3. "analog_inputs" – wejścia analogowe
  4. "registers" – rejestry
  (iv) *address* – adres zasobu w protokole Modbus. Jest to adres zgodny z mapą pamięci modułu MT
  (v) *size* – rozmiar przestrzeni do odczytania
  (vi) *interval* – co jaki okres (w sekundach) zasoby będą odczytywane samoistnie przez serwer OPC (jeżeli 0 – będą odczytywane wyłącznie z poziomu MATLAB-a)
  (vii) *debug* – opcja zapisu do pliku komunikacji dotyczącej konkretnego zadania
  (viii) *enable* – określa czy znacznik jest aktywny.

Uruchamianie serwera OPC

Program serwera OPC może być uruchomiony z poziomu linii komend systemu DOS/Windows przez podanie, jako parametr, nazwy pliku xml z konfiguracją:

mt_opc.exe <nazwa_pliku>

gdzie <nazwa_pliku> jest nazwą pliku konfiguracyjnego xml. Uruchomienie serwera z konfiguracją zawartą w pliku przyklad.xml będzie miało postać:

mt_opc.exe przyklad.xml

Jeżeli nie podamy nazwy pliku xml z konfiguracją jako parametru, serwer będzie tę nazwę próbował odczytać z pliku *startfile*. Jeżeli natomiast nie znajdzie pliku *startfile*, przyjmie domyślną nazwę *sample.xml*. Domyślnie program uruchamia się w trybie ukrytym i pracuje w tle.

W celu wizualizacji okienka programu należy podać parametr/*show*:

mt_opc.exe /show przyklad.xml

Korzystanie z zasobów serwera OPC z poziomu MATLAB-a

Z poziomu MATLAB-a zostały wykorzystane następujące funkcje pozwalające na komunikację według standardu COM:

(1) <obiekt> = actxserver(<progid>)
tworzy serwer COM, zwraca obiekt COM (oznaczony jako <obiekt>), reprezentujący interfejs serwera (w naszym wypadku serwera OPC). Argument pobierany przez tę funkcję jest łańcuchem znaków będącym identyfikatorem ustalonym przez producenta (w naszym przypadku producenta biblioteki nakładkowej Northern Dynamic OPC DA Automation Wrapper – firmę Northern Dynamic). Obiekt COM wykorzystany w pracy (biblioteka nakładkowa DLL) ma parametr <progid> równy 'NDI.OPCAutomation'

(2) <wartosc> = invoke(<obiekt>, <metoda> [,<arg1>,<arg2>, ... ,<argn>])
pozwala uruchomić funkcję składową (metodę) przekazaną jako zmienna łańcuchowa <metoda> obiektu <obiekt>. Liczba argumentów (<arg1>,<arg2>, ... ,<argn>) zależy od konkretnej metody i może być zerowa. <wartosc> jest wartością zwracaną przez wywoływaną metodę

(3) <wartosc> = get(<obiekt>,<parametr>)
umieszcza w zmiennej <wartosc> wartość parametru <parametr> należącego do obiektu <obiekt>.

Poniżej zostanie podany przykładowy cykl komunikacji (zapis/odczyt) parametrów modułu telemetrycznego MT–101 za pomocą komend MATLAB-a (wydanych z linii poleceń lub z napisanego m-pliku). Komendy MATLAB-a pozwalające na korzystanie z zasobów MT–101 są następujące:

(1) h=actxserver('NDI.OPCAutomation');
utworzenie nowego obiektu serwera ActiveX (technologia COM) dla nakładkowej biblioteki DLL udostępniającej zasoby serwera OPC.

(2) invoke(h, 'Connect', 'OPC.MT_Srv.1');
utworzenie nowego obiektu typu OPCServer dla serwera o nazwie 'OPC.MT_Srv.1'. Nazwę tę otrzymujemy, stosując polecenie:
get(h)
wyświetlony parametr ServerName zawiera szukaną nazwę serwera

(3) czas_serwera = get(h,'CurrentTime');
wyświetlenie czasu aktualnego serwera (w sekundach).

(4) Groups= get(h,'opcgroups');
utworzenie uchwytu do kolekcji OPCGroups

(5) gg=invoke(Groups,'Add');
utworzenie instancji obiektu typu OPCGroup i dodanie go do kolekcji Groups

(6) <obiekt> = invoke(gg.OPCItems, 'AddItem', <ItemID>, <ClientHandle>);
utworzenie obiektu <obiekt> typu OPCItem i dodanie go do kolekcji gg.OPCItems. Jako uchwyt klienta <ClientHandle> była przyjmowana dowolna niepowtarzalna liczba. Parametr <ItemID> identyfikuje zasób serwera OPC. Jego nazwa i zasoby, których może dotyczyć, zależą od pliku konfiguracyjnego z jakim został uruchomiony serwer OPC. Parametr ten ma następującą składnię:
<NazwaSieci>--<NazwaModbus>.<TypZasobu><AdresZasobu>'
Parametr <NazwaSieci> ma taką samą wartość jak parametr name znacznika <network> pliku konfiguracyjnego (w naszym wypadku jest to MT_Tanks)
Parametr <NazwaModbus> ma taką samą wartość jak parametr name znacznika <modbus> pliku konfiguracyjnego (w naszym wypadku jest to slave).
Parametr <TypZasobu> zależy od parametru type znacznika <modbus> pliku konfiguracyjnego, co jest przedstawione w tabeli.
Parametr <AdresZasobu> jest liczbą, która może przyjmować wartości w zależności od parametrów *address* i size znacznika <modbus> pliku konfiguracyjnego. Przykładowo dla parametrów *address* = "64" oraz *size* = "10", <AdresZasobu> będzie mógł przybierać wartości z zakresu od 64 do 73.
Parametr <ItemID> może być uzyskany za pomocą przeglądarki (obiekt OPCBrowser omówiony dalej).
Przykładowy zapis:
zasob=invoke(gg.OPCItems, 'AddItem', 'MT_Tanks--slave.REG64', 1235);
oznacza przypisanie zmiennej zasob obiektu ActiveX przechowującego w sobie parametry związane z rejestrem modułu MT–101. Rejestr ten ma adres 64 (w zapisie szesnastkowym 0x40). W mapie pamięci odczytujemy, że rejestr ten nosi nazwę REG1 (pod taką nazwą jest widoczny w programie MTProg)

(7) invoke(zasob,'Read', <opcja>);
Jeżeli parametr <opcja> ma wartość 1, to nastąpi wysłanie do serwera OPC polecenia odczytu danych dotyczących zasobu zasob z wewnętrznej pamięci serwera (cache). Gdy parametr ten ma wartość 2, wówczas serwer OPC wyśle zapytanie do modułu MT–101 nakazujące odczyt zasobu (device). Jeżeli moduł MT–101 ma zdefiniowane reguły wysyłania danych (opisane w [12]), to serwer OPC nie wyśle modułowi polecenia odczytu zasobu. Zajście zdarzenia w module, określonego odpowiednią regułą w programie MTConfig spowoduje wysłanie określonego obszaru pamięci MT–101. Jeżeli w tym obszarze znajdzie się szukany zasób, to związany z nim obiekt OPCItem zasob zostanie uaktualniony

(8) invoke(zasob,'Write', <wartosc>);
wpisanie wartości <wartosc> do obiektu OPCItem zasob

(9) wartosc=get(zasob,'Value');
odczytanie wartości wartosc z obiektu OPCItem zasob

(10) timestamp=get(zasob,'TimeStamp');
odczytanie czasu aktualizacji obiektu OPCItem zasob

(11) jakosc=get(zasob,'Quality');
odczytanie jakości transmisji dla obiektu OPCItem zasob

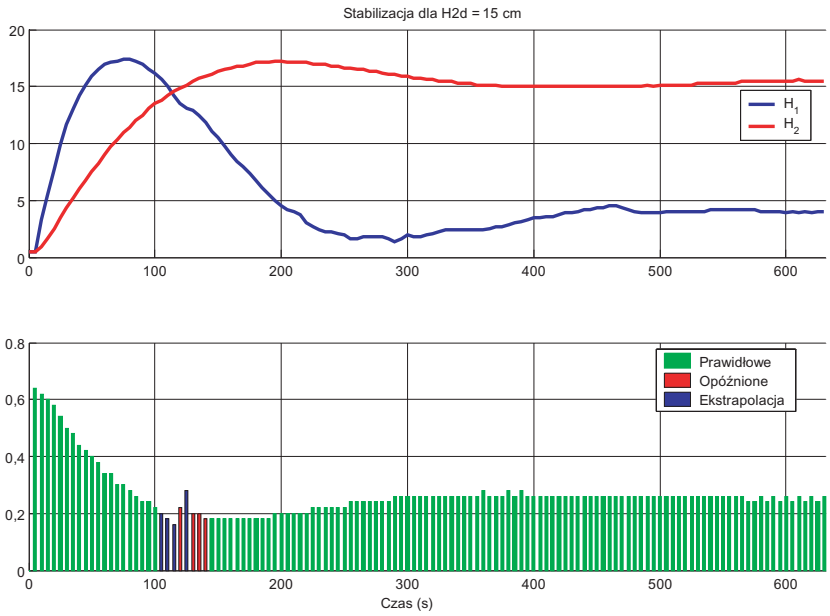
(12) invoke(h, 'Disconnect');
rozłączenie się z serwerem OPC,

W punkcie (9) cyklu komunikacyjnego pokazano odczyt zasobu modułu telemetrycznego. Graficzne funkcje MATLAB-a umożliwiają gromadzenie tych zasobów w postaci tablicy danych, a następnie ich wizualizację. Na rys. 8 pokazano przykład wizualizacji w czasie zmiennego poziomu wody w zbiornikach z rys. 7 oraz sterowania (modulacja szerokości impulsów – PWM).

W celu zorientowania się w budowie parametru <ItemID> należy skorzystać z obiektu OPCBrowser. Parametr <ItemID> ma strukturę drzewiastą. Gałęzie i liście

Tabela. Zależność parametru <TypZasobu> od parametru type znacznika <modbus> w pliku konfiguracyjnym serwera OPC

type	<TypZasobu>
binary_inputs	BI
binary_outputs	BO
analog_inputs	AI
registers	REG



Rys. 8. Stabilizacja wody w środkowym zbiorniku na poziomie 15 cm

drzewa są rozdzielone symbolem kropki. Komendy MATLAB-a pozwalające na korzystanie z obiektu OPCBrowser w celu odtworzenia struktury parametru <ItemID> są następujące:

(1) hBrowser=invoke(h, 'CreateBrowser');
utworzenie obiektu OPCBrowser dla serwera ActiveX, dla którego wywołana była wcześniej metoda 'Connect'
(2) invoke(hBrowser, 'MoveToRoot');
przesunięcie obiektu OPCBrowser do korzenia drzewa
(3) Branches=invoke(hBrowser, 'ShowBranches');
przejście do gałęzi drzewa
(4) BranchesCount=get(hBrowser, 'Count');
liczba gałęzi w drzewie
(5) NazwaGalezi=invoke(hBrowser, 'Item', num2str(1));
ustalenie nazwy pierwszej gałęzi (w naszym wypadku będzie to 'MT_Tanks--slave')
(6) invoke(hBrowser, 'MoveDown', NazwaGalezi);
przesunięcie przeglądarki w dół (do gałęzi NazwaGalezi)
(7) invoke(hBrowser,'ShowLeafs');
przejście do liści drzewa
(8) LeafsCount=get(hBrowser, 'Count');
liczba gałęzi w drzewie
(9) invoke(hBrowser, 'Item', num2str(1));
ustalenie nazwy pierwszego liścia. W naszym wypadku będzie to 'BI8' (ósme wejście binarne), tak więc parametr <ItemID> będzie miał w tym wypadku postać 'MT_Tanks--slave.BI8'

Podsumowanie

Opierając się na technologię OPC protokole Modbus oraz środowisku MATLAB-a, zbudowano rozproszony system sterowania. Dzięki temu udało się znacznie uprościć korzystanie z zasobów sprzętowych modułu telemetrycznego MT–101. Moduł ten wykorzystuje technologię GSM/GPRS, co oznacza, że system monitorowania lub sterowania może obejmować obszar geograficzny o zasięgu takim, jaki ma sieć telefonii komórkowej. Dzięki użytym technologiom jest możliwa współpraca urządzeń odległych od siebie o setki kilometrów. Serwer OPC pozwala na dostęp do zasobów wewnętrznych modułu telemetrycznego. MATLAB umożliwia obróbkę tych danych za pomocą narzędzi CACSD. Pozwala to na syntezę układu sterowania oraz monitorowanie zasobów modułu telemetrycznego.

Bibliografia

1. http://www.opcfoundation.org/
2. A. Simicz, *Technologia GPRS w monitorowaniu procesów*, Praca Magisterska (opiekun – prof. dr hab. inż. Wojciech Grega), Katedra Automatyki, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki AGH, Kraków, 2005.
3. W. Grega, K. Kołek, *Simulation and Real-time Control: from Simulink to Industrial Applications*, Proceedings of 11th IEEE International Conference on Computer Aided Control System Design, Glasgow, 2002, pp. 104-109.
4. http://www.mathworks.com/
5. http://rose.iinf.polsl.gliwice.pl/iinf/zui/zima/SK1_SW7_USM3.htm
6. Modicon Modbus Protocol Reference Guide, MODICON Inc., Industrial Automation Systems, 1996.
7. http://www.pep.com.pl/pep/opc_serwery.htm
8. OPC Overview Version 1.0, OPC Task Force, 1998.
9. http://aq.ia.agh.edu.pl/Aquarium/index.html
10. *OPC Data Access Automation Specification Version 2.02*, OPC Foundation,1999.
11. B. Mrozek, Z. Mrozek, MATLAB 6, PLJ, Warszawa, 2001.
12. *Moduł telemetryczny MT-101 – instrukcja obsługi, wersja 1.31*, AB-MICRO, Warszawa, 2004.
13. http://telemetria.pl
14. http://www.abmicro.pl/