Bogusław TWARÓG, Zbigniew GOMÓŁKA, Ewa ŻESŁAWSKA, Paweł KRUTYS
Uniwersytet Rzeszowski, Polska

System nadzorujący i sterujący przebieg procesu technologicznego

Wstęp

Współczesne złożone procesy przemysłowe wymagają zaawansowanych narzędzi i technologii wytwarzania sterowanych programowalnymi kontrolerami mikroprocesorowymi. Dzięki rozwojowi technologii sieciowych zaczęto tworzyć oprogramowanie, którego głównym celem jest monitorowanie i kontrolowanie pracy indywidualnych elementów sterujących. Systemy zwane SCADA (Supervisory Control And Data Acquisition) potrafią zbierać aktualne dane pomiarowe z czujników na obiekcie, a następnie po ich przetworzeniu przedstawić odpowiednią wizualizację graficzną i umożliwić zdalne sterowaniem procesem produkcji, a także alarmować w przypadku wystąpienia określonych nieprawidłowości.

Niestety, systemy tego typu nie są tanie i często obsługują tylko wybraną platformę systemową, dlatego też ograniczają możliwość wprowadzenia ich dla szerokiego grona użytkowników. Drogą do rozwiązania tego problemu może być „lekka" aplikacja napisana w języku programowania, który jest niezależny od systemu operacyjnego, a także w środowisku, które jest dostępne dla każdego na zasadzie licencji GNU General Public License. Java jako uniwersalna, wydajna i przenośna platforma dla komputerów stacjonarnych, laptopów, telefonów komórkowych czy tabletów w pełni wpisuje się w przyjęte założenia.

Zrealizowany projekt jest zaawansowanym systemem sterowania nadrzędnego i akwizycji danych (SCADA), umożliwiającym integratorom systemów tworzyć wyrafinowane aplikacje sterująco-monitorujące dla wszystkich gałęzi przemysłu. Wszystkie funkcje sterowania i monitorowania są już wbudowane w system, natomiast projektant musi tylko zaprojektować inteligentną architekturę układu. Wykorzystano wielozadaniowość i wszystkie potężne cechy systemów operacyjnych oraz wbudowany mechanizm zdarzeniowy (event-driven), umożliwiający osiągnięcie wysokiej sprawności pracy i utrzymania integralności danych. Także graficzny interfejs użytkownika systemu zapewnia przejrzystość i wydajność procesu wizualizacji danych technologicznych. Obecnie do nadzoru i sterowania wykorzystuje się grafikę czasu rzeczywistego i kierowane zdarzeniowo aktualizacje informacji, a wszystko w dowolnym systemie operacyjnym.

Aplikacja systemu komunikuje się z urządzeniami obiektowymi, takimi jak sterowniki PLC, instrumenty pomiarowe i inne. Ponieważ wszystkie dane są monitorowane i zapisywane, system szybko reaguje na zaistniałe sytuacje zgodnie z zaprogramowaną procedurą i żądaniami operatora.

Protokoły komunikacyjne w sieciach przemysłowych

Modbus jest uznanym protokołem warstwy aplikacji do komunikacji pomiędzy urządzeniami, głównie w celu wymiany danych typowych dla branży automatyki. Swą popularność zyskał dzięki prostocie zastosowanych w nim rozwiązań, jawności specyfikacji protokołu, a ponadto takim cechom, jak: dostęp do łącza na zasadzie *Master – Slave* (*Query – Response*), zabezpieczenie komunikatów przed przekłamaniami, potwierdzenie wykonania rozkazów i sygnalizacja błędów oraz mechanizmy unikające zawieszania się systemu. Pozwala to na łatwą implementację w dowolnym urządzeniu posiadającym mikrokontroler i w znacznym stopniu wpływa na obniżenie kosztów.

Ramka protokołu Modbus (rys. 1) określa format przesyłanych wiadomości i zawiera: adres odbiorcy, kod funkcji reprezentujący żądane polecenie, dane dotyczące funkcji oraz słowo kontrolne zabezpieczające przesyłaną wiadomość. Postać ramki zapytania wysyłanego przez jednostkę *Master* i ramki odpowiedzi jednostki *Slave* jest podobna. Różnica polega na tym, że w polu danych ramki odpowiedzi występują dane, których dostarczenia żądała stacja *Master*. Możliwe są dwa typy transakcji: „rozgłoszenie" (Broadcast), w którym jednostka *Master* wysyła jedynie ramkę rozgłoszenia o adresie 0, który nie jest przypisany do konkretnego urządzenia *Slave*, ale wszystkie urządzenia ją odbierają i wykonują zawartą w niej funkcję, nie odsyłając odpowiedzi oraz „zapytanie – odpowiedź" (*Query – Response*), gdzie jednostka *Master* wysyła zapytanie i oczekuje na odpowiedź od wybranego urządzenia *Slave*.

Znacznik początku	Adres	Kod funkcji	**Dane…**	Suma kontrolna	Znacznik końca
	część informacyjna ramki				

Rys. 1. Ramka protokołu Modbus

W trybie RTU (Real-Time Unit) komunikacja odbywa się z kontrolą czasu rzeczywistego, a bajty są wysyłane binarnie jako znaki ośmiobitowe. Ramka musi być przesłana w całości, tj. dopuszczalna przerwa między znakami nie może przekraczać 1,5 znaku. Dłuższa może być potraktowana jako przerwa kończąca ramkę. Część informacyjna ramki zaczyna się od adresu urządzenia, do którego kierowane jest polecenie. Osiem bitów pozwala na adresowanie 255

urządzeń *Slave*, przy czym 0 jest wspomnianym adresowaniem rozsiewczym (brodcast) do wszystkich [Kasprzyk 2007].

Protokół Modbus TCP/IP, to protokół RTU z dodanym interfejsem TCP, dzięki któremu może działać w sieci Ethernet. Komunikacja w takiej konfiguracji odbywa się na zarezerwowanym dla Modbus porcie numer 502.

Tabela 1

Zestawienie podstawowych funkcji protokołu Modbus

Opis funkcji	Rodzaj zmiennej	Kod funkcji
Read Coils Odczyt stanów wyjść binarnych, np. wyjść PLC	**Single bit** Jednobitowy	0x 01
Read Discrete Inputs Odczyt stanów wejść binarnych, np. wejść PLC	**Single bit** Jednobitowy	0x 02
Read Holding Register Odczyt rejestrów pamiętających	**16-bit Word** Słowo 16-bitowe	0x 03
Read Input Register Odczyt rejestrów wejściowych, np. wartości z wejść analogowych PLC	**16-bit Word** Słowo 16-bitowe	0x 04
Write SingleCoils Zapis jednego wyjścia binarnego, ustawianie przekaźnika	**Single bit** Jednobitowy	0x 05
Write single Register Zapis do jednego rejestru pamiętającego	**16-bit Word** Słowo 16-bitowe	0x 06
Write Multiple Coils Zapis wielu wyjść binarnych, ustawianie przekaźników	**Single bit** Jednobitowy	0x 0F
Write Multiple Registers Zapis do wielu rejestrów	**16-bit Word** Słowo 16-bitowe	0x 10
Diagnostyka, zgłaszanie błędów		0x 08

Programistyczna biblioteka funkcji protokołu Modbus

Biblioteka Jamod jest obiektową implementacją protokołu Modbus, zrealizowana w całości na bazie technologii Java. Dzięki temu osiągnięto intuicyjny i obiektowo zaimplementowany protokół Modbus z naturalnym odwzorowaniem protokołu do abstrakcyjnego modelu klas. Może być ona wykorzystywana na różnych platformach systemowych i urządzeniach. Pozwala na szybką realizację aplikacji typu *Master* lub *Slave* w różnych wariantach transportowych (IP, szeregowy). Pamięć przechowująca zestaw pomiarów procesowych lub stanów wejścia/wyjścia nazywana „obrazem procesu", odzwierciedla wszystko to, co przedstawia stan procesu w określonej chwili czasowej. Poniższa lista prezentuje abstrakcyjne modele dla różnych typów danych:

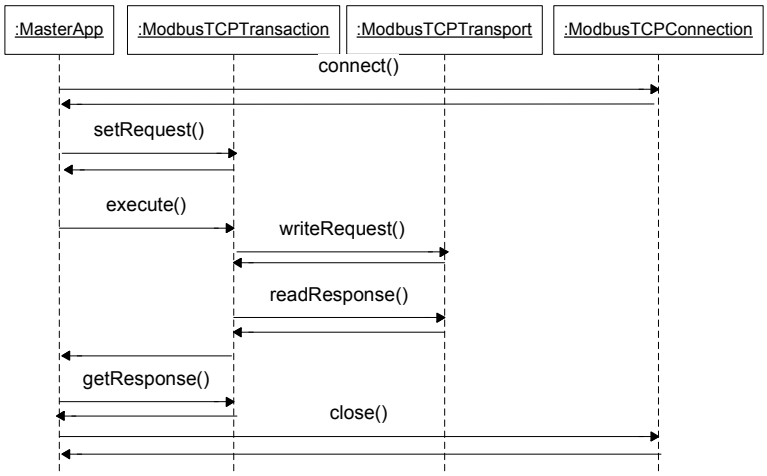
– wejście cyfrowe – DigitalIn (dla wejść dyskretnych);

– wyjście cyfrowe – DigitalOut (dla cewki);
– rejestry wejściowe – InputRegister (dla rejestrów wejściowych);
– rejestry – Register (dla rejestrów wewnętrznych).

W konfiguracji Modbus TCP, typu *Master – Slave*, *Master* nawiązuje połączenie z urządzeniem *Slave* i korzysta z tego połączenia do wysyłania żądań. Każdy cykl żądania i odpowiedzi jest nazywany transakcją. *Master* może sprawdzać i odpytywać (wielokrotnie) dane ze źródła, jak również kontrolować samo urządzenie. Do implementacji w konfiguracji *Master* potrzebne są następujące elementy i ich klasy:

– Połączenie: TCPMasterConnection;
– Transakcja: ModbusTCPTransaction;
– Zapytanie: ModbusRequest (podklasa ReadInputDiscretesRequest);
– Odpowiedź: ModbusResponse (podklasa ReadInputDiscretesResponse).

Rys. 2. przedstawia diagram sekwencji dla przebiegu komunikacji zbudowany dla stacji *Master*.



Rys. 2. Diagram sekwencji dla stacji *Master*

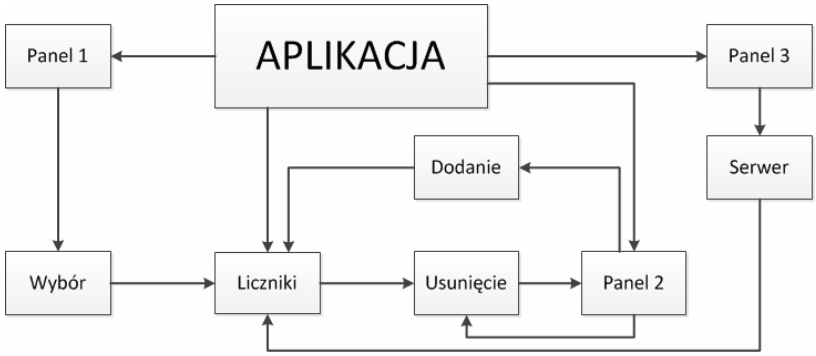
Na początku należy dodać instancje i zmienne aplikacji, które będą potrzebne. Następnie rozpoczyna się otwieranie połączenia, a zaraz po tym przygotowanie żądania i transakcji. Ostatnią częścią jest określenie ilości powtórzeń transakcji i zamknięcie połączenia.

Implementacja systemu wizualizacji i sterowania

Aplikacja służąca do modelowania i symulowania systemów nadzorujących procesy przemysłowe posiada interfejs składający się z okna głównego, służącego

do obserwacji graficznych wizualizacji przetwarzanych danych, a także panelu który może służyć do symulowania różnych sygnałów sterujących poprzez zadawanie konkretnych wartości [Twaróg 2011].

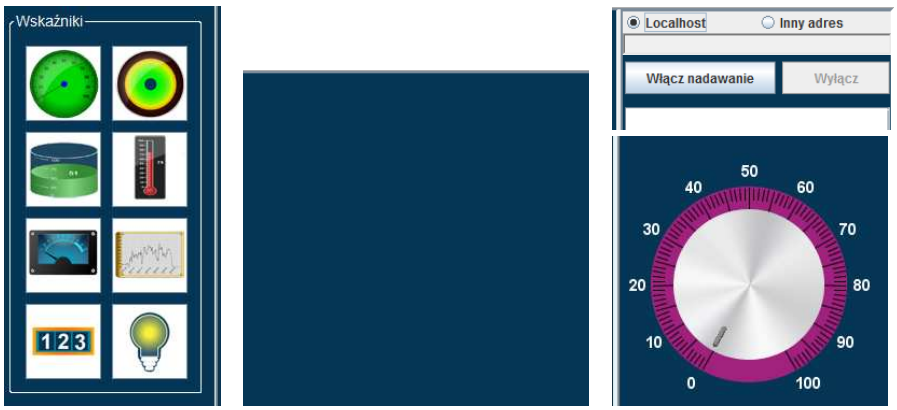
Na rys. 3 został przedstawiony ogólny zarys struktury tworzonego systemu. Obiekt o nazwie Aplikacja reprezentuje interfejs użytkownika.



Rys. 3. Blokowy zarys struktury zaimplementowanego systemu

Z jego poziomu udostępniono kolejne elementy, które mają za zadanie realizować przedstawione wcześniej założenia:

– Panel 1 – wyświetla zminiaturyzowane obiekty liczników, które są dostępne do wyboru. Stosuje również funkcje wyboru liczników;
– Panel 2 – realizuje funkcje głównego panelu aplikacji, udostępnia pole do wyświetlania dla wszystkich dodanych wskaźników;
– Panel 3 – jest odpowiedzialny za symulowanie określonych wartości narzędzia nadawczego, jak również za sterowanie serwerem;



Rys. 4. Główny panel zrealizowanego systemu kontrolno-monitorującego

– Liczniki – jest to zbiór wszystkich obiektów, jakimi są wskaźniki graficzne. Poprzez swoje dynamiczne funkcje potrafią przekazać w sposób graficzny i czytelny wszystkie przetwarzane przez nie wartości.

W tworzonym systemie została zastosowana grupa ośmiu liczników, które stanowią główną funkcję całej aplikacji. Każdy z nich może posłużyć do wizualizacji różnego typu wartości, zaczynając od pomiaru ciśnienia, czy poziomu głośności, a kończąc na obsłudze pomiarów dyskretnych [Binko 2013].

Podsumowanie

Korzyści wynikające ze zrealizowanej aplikacji typu SCADA przedstawiają się w następujących wnioskach: czytelne informacje dotyczące pomiarów, podejmowanie szybkich i bardziej precyzyjnych decyzji związanych z zaistniałymi zdarzeniami, łatwe i szybkie uruchomienie nawet bardzo zaawansowanych systemów automatyki, zwiększenie wydajności procesu produkcji, zmniejszenie czasu wyrobu produktu, zmniejszenie kosztów działalności inżynierów.

Literatura

Binko Ł. (2013), *Graficzna wizualizacja procesów produkcyjnych w środowisku Java*, praca inżynierska, promotor – B. Twaróg.

Kasprzyk J. (2007), *Programowanie sterowników przemysłowych*, WNT.

Twaróg B. (2011), *Rozpoznawanie obrazów wizyjnych w systemach nawigacji robotów mobilnych*, Z. Gomółka, B. Kwiatkowski, „Technical News", 1(33).

Streszczenie

W pracy zaprezentowano zaawansowany system sterowania i akwizycji danych (SCADA), umożliwiający projektantom tworzyć specjalistyczne aplikacje sterująco-monitorujące dla różnych gałęzi przemysłu. Zrealizowana aplikacja komunikuje się z urządzeniami obiektowymi, takimi jak sterowniki PLC, instrumenty pomiarowe w czasie rzeczywistym. Ponieważ wszystkie dane są monitorowane i zapisywane, system szybko reaguje na zaistniałe sytuacje zgodnie z zaprogramowaną procedurą i żądaniami operatora.

Słowa kluczowe: proces technologiczny, sterowanie, protokół Modbus, biblioteka Jamod, wizualizacja, sterowniki przemysłowe, programowanie, Java, SCADA.

System of the supervising and controlling the technological process

Abstract

This paper presents an advanced control and data acquisition (SCADA) system that allows designers to create specialized applications to control and monitoring for a different of industries. Completed application communicates with field devices such as PLCs, measuring instruments in real time. Since all data is monitored and recorded, the system responds quickly to opportunities in accordance with the programmed procedure and requests the operator.

Key words: technological process control, Modbus protocol, Jamod library, visualization, industrial controllers, programming, Java, SCADA.