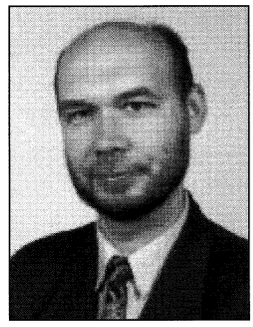
Cezary Zieliński

POLITECHNIKA WARSZAWSKA
INSTYTUT AUTOMATYKI I INFORMATYKI STOSOWANEJ

Narzędzie do szybkiej weryfikacji koncepcji układu sterowania urządzenia mechatronicznego

Prof. dr hab. inż. Cezary ZIELIŃSKI



Jest profesorem Politechniki Warszawskiej w Instytucie Automatyki i Informatyki Stosowanej. Od dwudziestu lat naukowo zajmuje się robotyką. Jego prace koncentrowały się na zagadnieniach związanych z: programowaniem robotów, architekturą sterowników systemów robotycznych, wykorzystaniem informacji z różnorodnych czujników do sterowania robotami. W wyniku tych prac powstało kilka języków-bibliotek służących do konstruowania i programowania systemów jednorobotowych, a następnie wielorobotowych. Najnowszy system (MRROC++) wykorzystywany jest do prowadzenia badań w kilku laboratoriach.

Streszczenie:

Konstruując złożone urządzenie mechatroniczne bardzo często musimy zweryfikować swoją koncepcję układu sterowania. Sam układ sterujący zazwyczaj składa się z jednego bądź więcej mikrokomputerów wykonujących programy realizujące we współpracy ze specjalizowanym sprzętem poszczególne zadania systemu. Przed zbudowaniem takiego dedykowanego systemu sterowania dobrze jest sprawdzić, czy wszystkie jego części są w stanie wykonywać właściwie zaplanowane zadania. Większość producentów złożonych elementów, z których składamy system, dostarcza narzędzia do konstruowania oprogramowania zdolnego realizować przypisane mu funkcje. Niestety niejednokrotnie narzędzia programistyczne pochodzące od różnych producentów wymagają zarówno innych platform sprzętowych jak i systemów operacyjnych. Niemniej jednak bardzo często zawierają oprogramowanie zdolne do porozumiewania się poprzez sieć z wykorzystaniem protokółów TCP/IP lub UDP/IP. Niniejszy artykuł pokazuje jak można wykorzystać to do szybkiej weryfikacji koncepcji układu sterowania. Gdy koncepcja zostanie zweryfikowana pozytywnie, można wykonać układ i oprogramowanie dedykowane, które lepiej i taniej będzie realizowało te same funkcje.

Abstract

In the development of complex electromechanical systems, e. g. robot systems, it is often necessary to produce prototype systems to demonstrate the viability of the concept. This, typically, requires that different computer hardware operating with different software to be interconnected. Towards this aim, UDP/IP or TCP/IP networking interfaces provide a useful medium. The paper presents a universal switch that can handle the communication between diverse software modules of any large robot system. The only assumption is made that each component has the capability of using UDP/IP and resides on a node of a real computer network supporting that protocol. The switch has internal data buffering capability, so the speed of data production and consumption may be radically different. It can handle a large number of data producers and consumers. The effectiveness of the solution has been tested on a pilot training and control system for an Underwater Robotic Vehicle (URV).

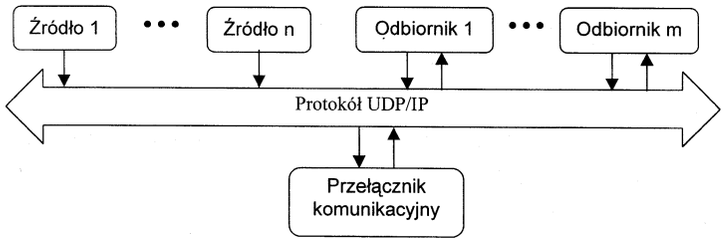
1. Wstęp

Systemy mechatroniczne zazwyczaj mają strukturę warstwową. Każda z warstw zawiera wiele elementów. Warstwa najniższa składa się z urządzeń elektromechanicznych. Kolejna warstwa tworzy rodzaj elektronicznego interfejsu z poprzednią. Jest ona odpowiedzialna za formowanie sygnałów sterujących napędami oraz zbieraniem sygnałów z czujników. Warstwa ta połączona jest z komputerami realizującymi złożony algorytm sterowania. Ów algorytm, w postaci programu sterującego, konstytuuje ostatnią warstwę tej struktury. Oczywiście każda z wyżej wymienionych warstw też może być podzielona nad podwarstwy [4]. Przed zrealizowaniem takiej złożonej struktury dobrze jest wpierw sprawdzić, czy wszystkie jej części składowe złożone razem spełniają swe funkcje należycie. W tej fazie projektowania można zamiast z dedykowanych komputerów wbudowanych skorzystać z uniwersalnych komputerów połączonych siecią. W ten sposób każdy z elementów składowych będzie miał do dyspozycji własny komputer i będzie w związku z tym mógł korzystać z oprogramowania wykorzystującego dowolną platformę programową (np. system operacyjny, język programowania). Producenci takich elementów dostarczają swe produkty z oprogramowaniem (bibliotekami) rzadko współpracującym bezpośrednio z oprogramowaniem innych firm. W takiej sytuacji każdy z elementów systemu będzie realizował swe funkcje dokładnie tak, jak obmyślił to jego producent, a więc zapewne w sposób najbardziej efektywny, ale połączenie tych elementów może stanowić problem. W artykule tym proponowane jest rozwiązanie tego problemu przy założeniu, że dostarczone przez producentów poszczególnych elementów oprogramowanie jest zdolne do komunikacji sieciowej z wykorzystaniem protokółu UDP/IP (User Datagram Protocol/Internet Protocol) [2, 6, 7]. Pomysł został zastosowany do wstępnej weryfikacji działania systemu sterującego robotycznym pojazdem podwodnym.

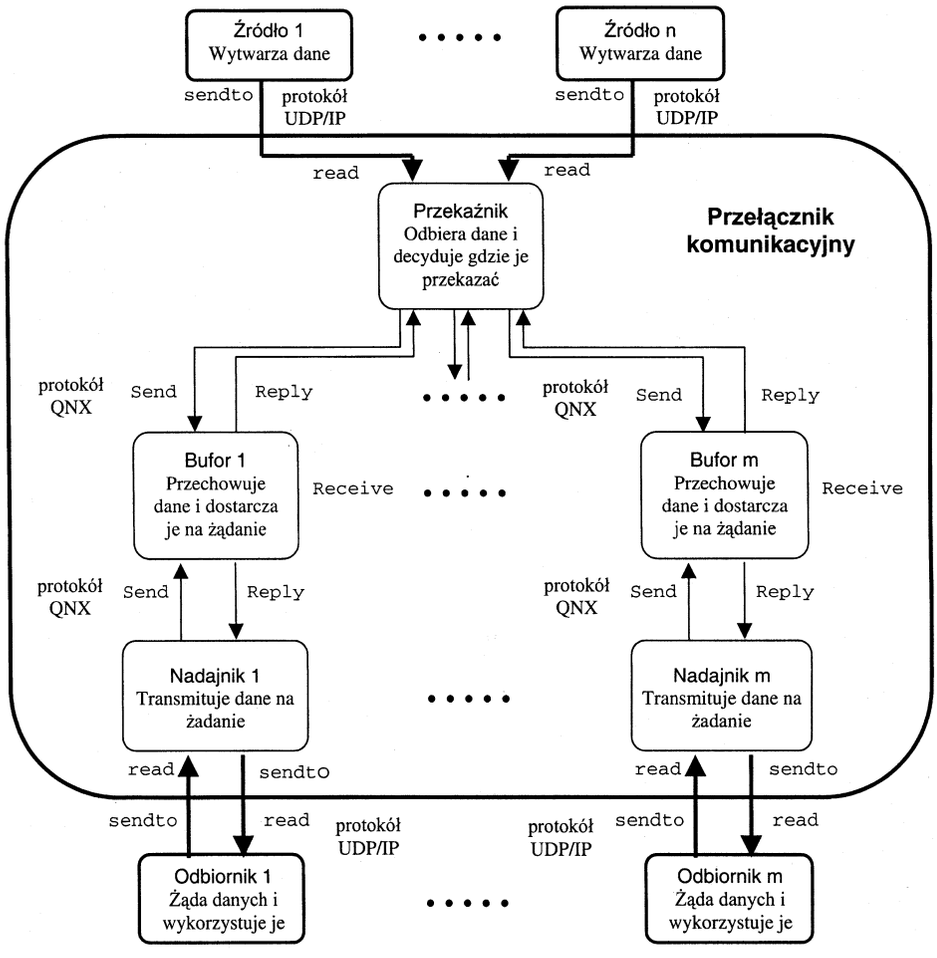
2. Przełącznik komunikacyjny

Często w systemach mechatronicznych warstwa programowa układu sterowania podzielona jest na elementy odpowiedzialne za zbieranie danych z czujników oraz moduły przetwarzające otrzymane dane (np. w celu wytworzenia sygnałów sterujących napędami). Wyłania się tu struktura, w skład której wchodzą producenci danych (źródła) oraz ich konsumenci (odbiorniki). Niestety zazwyczaj źródła i odbiorniki pracują z różnymi częstotliwościami powtarzania swych akcji. Ponadto zazwyczaj dość trudna jest bezpośrednia synchronizacja pracy wielu źródeł oraz wielu odbiorników. Zaproponowany przełącznik komunikacyjny rozwiązuje oba powyższe problemy.

Przełącznik komunikacyjny wykorzystuje oddzielny komputer pracujący pod nadzorem systemu operacyjnego czasu rzeczywistego QNX [8]. Program przełącznika stale bada gniazdko (socket [2]) o ustalonym adresie, by stwierdzić, czy pojawił się jakiś pakiet UDP/IP (datagram) wysłany przez źródło. Nagłówki przysłanych pakietów są analizowane, by ustalić adresata (odbiornik) przesyłki. Na tej podstawie przesłane dane wstawiane są do bufora przypisanego temu odbiornikowi. Gdy odbiornik zwróci się o dane, poprzez inne, ale znane jemu gniazdko, przełącznik prześle je. Gniazdko to nadzorowane jest przez nadajnik. W ten sposób źró-



Rys. 1. Sposób połączenia źródeł danych z ich odbiornikami za pośrednictwem przełącznika komunikacyjnego.



Rys. 2. Wewnętrzna struktura przełącznika komunikacyjnego

dło może wytwarzać dane z dowolną prędkością, a odbiornik może z nich korzystać we właściwym mu tempie. Oczywiście odbiornik otrzymuje zawsze najświeższe dane dostępne w buforze.

Sposób połączenia źródeł, odbiorników oraz przełącznika komunikacyjnego prezentuje rys. 1. Przełącznik może zostać tak skonfigurowany, że będzie obsługiwał dowolną liczbę źródeł i wskazaną liczbę odbiorników. Nie ma ograniczeń co do rozmieszczenia poszczególnych źródeł i odbiorników – poszczególne elementy mogą dzielić wspólny komputer bądź być ulokowane w oddzielnych węzłach sieci. Rozmieszczenie tych elementów podyktowane jest przyjętą architekturą systemu sterującego oraz obliczeniochłonnością zadań wykonywanych przez dany element. Nie ma ograniczeń co do typu i formatu danych przekazywanych między źródłami a odbiornikami. Jest to związane z tym, że przełącznik ich nie analizuje, więc format danych jest mu obojętny. Jedynie format nagłówka jest istotny.

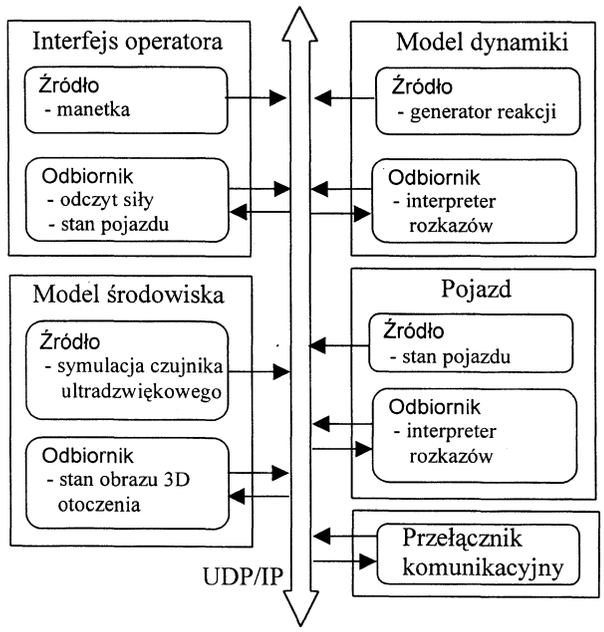
Wewnętrzną strukturę przełącznika komunikacyjnego uwidacznia rys. 2. W jego skład wchodzą trzy rodzaje procesów: jeden przekaźnik oraz tyle buforów i nadajników ile jest odbiorników. Zadaniem przekaźnika jest nasłuch swego gniazdka. Gdy tylko pojawi się pakiet, jego nagłówek jest dekodowany i na tej podstawie określany jest bufor, w którym należy umieścić dane. Dane te będą przechowywane przez bufor właściwy dla odbiornika tych danych aż do momentu pojawienia się nowego pakietu przeznaczonego dla tego samego odbiornika. Odbiornik zwraca się o dane do nadajnika. By zrealizować to żądanie nadajnik komunikuje się z odpowiadającym mu buforem. Wtedy dane przekazywane są do nadajnika, a ten transmituje je do odbiornika poprzez gniazdko znane jedynie temu nadajnikowi i odbiornikowi. Przełącznik komunikacyjny zewnętrznie wykorzystuje do przesyłania pakietów protokół UDP/IP, natomiast wewnętrznie mechanizmy komunikacji międzyprocesowej udostępniane przez sytem operacyjny QNX. W tym przypadku są to spotkania, czyli triada: *Send, Receive, Reply*.

Procesy przekaźnika oraz nadajników przez większość czasu są zablokowane w oczekiwaniu na przesłanie pakietu danych bądź żądania odczytu danych z bufora. Natomiast procesy buforów zablokowane są w oczekiwaniu na przesłanie nowych danych przez przekaźnik bądź na żądanie przekazania danych nadajnikowi. W ten sposób moc obliczeniowa komputera wykonującego program przełącznika komunikacyjnego wykorzystywana jest bardzo oszczędnie. Przełącznik komunikacyjny jest czystym dostarczycielem usług, czyli serwerem, natomiast źródła i odbiorniki są jego klientami. Przy rozsądnych rozmiarach przesyłanych pakietów, a tak jest w dobrze zaprojektowanych systemach mechatronicznych, czynnikiem limitującym przepustowość przełącznika jest fizyczna warstwa połączeń międzykomputerowych. W naszym przypadku zastosowano sieć Ethernet.

3. Wykorzystanie

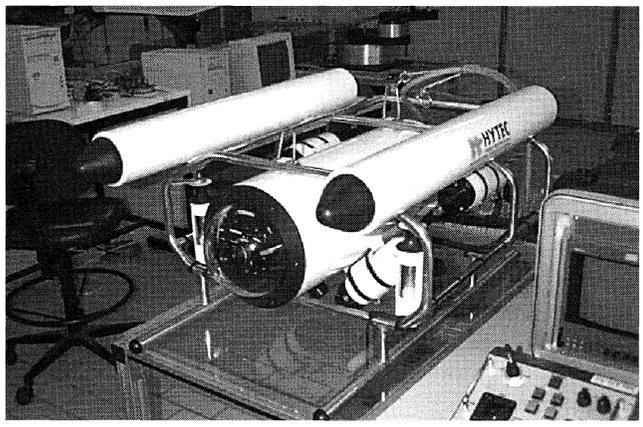
Przełącznik komunikacyjny został wykorzystany do weryfikacji prawidłowości działania zintegrowanego systemu sterowania oraz symulacji działania robotycznego pojazdu podwodnego. Taki system powstał w Robotics Research Centre (RRC) na Nanyang Technological University (NTU) w Singapurze [3,5]. Ten sam interfejs z użytkownikiem jest stosowany zarówno przy uczeniu operatora na symulatorze posługiwania się pojazdem jak i przy późniejszym sterowaniu rzeczywistym robotem. Symulowana jest zarówno dynamika pojazdu jak i podwodne otoczenie robota [1]. Rys. 3 poka-

zuje poszczególne komputery oraz realizowane przez nie zadania, jednocześnie wskazując źródła i odbiorniki danych.



Rys. 3. Struktura systemu sterowania i symulacji robotycznego pojazdu podwodnego

Rzeczywistym pojazdem podwodnym jest Super Safir (rys. 4) wytworzony przez firmę HYTEC Hydro-Technology, a wyposażony w elektronikę przez RRC. Pojazd posiada wiele czujników, między innymi kamerę umożliwiającą obserwację otoczenia. Energia oraz sygnały sterujące pracą czterech silników elektrycznych, a nadto dane z czujników przekazywane są do stacji operatorskiej za pośrednictwem kabla o długości 250 m. W trakcie szkolenia pojazd zastępowany jest symulatorem samego robota jak i jego otoczenia. Symulator pojazdu, z punktu widzenia operatora, zachowuje się w przybliżeniu jak rzeczywisty robot – w ten sam sposób reaguje na rozkazy użytkownika. Efekty wywoływane przez wydawane rozkazy można obserwować na ekranie specjalnego monitora wytwarzającego złudzenie obrazu trójwymiarowego obserwowanego poprzez okulary CrystalEyesVR. Ponieważ pojazd przeznaczony jest do prac przy podwodnych rurociągach i platformach wiertniczych, to co widać na ekranie stanowi konstrukcje rurowe oraz dno morza.



Rys. 4. Pojazd podwodny Super Safir

Każdy z elementów wyróżnionych na rys. 3 stanowi program (często wielowątkowy) pracujący na oddzielnym komputerze. Ponadto każdy z tych programów korzysta z innej platformy programistycznej. Interfejs operatora został stworzony za pomocą LabVIEW. Jest to środowisko programowe oparte na graficznym języku programowania G [9]. Natomiast model środowiska został stworzony za pomocą WorldToolKit dostarczanego przez Sense8 Corporation. Dla zwiększenia realizmu symulator wytwarza efekty dźwiękowe oraz wytwarza sygnały sprzężenia zwrotnego od siły dla manetki sterującej używanej przez operatora [10]. W ten sposób manetka jest źródłem rozkazu ruchu oraz stanowi swoistego rodzaju czujnik dotykowy – wibruje przy zderzeniu ze środowiskiem oraz zwiększa swe opory ruchu przy wzroście prędkości i przyspieszeniu pojazdu. WorldToolKit stanowi bibliotekę funkcji języka C. Za ich pomocą programista tworzy model środowiska.

Rzeczywisty pojazd dołączony jest do systemu za pomocą sprzęgu RS-232. Sterownik wytwarza sygnały: wskazujące kierunek i prędkość ruchu poszczególnych silników, intensywność oświetlenia wytwarzanego przez dwie lampy, oraz sygnały sterujące ruchem kamery (dwa obroty oraz ogniskowanie i teleobiektyw). W drugą stronę przesyłane są sygnały o aktualnym stanie pojazdu: odczyt kompasu pokładowego, zanurzenie, stan czujników awaryjnych oraz informacja o aktualnym położeniu kamery.

4. Wnioski

Zastosowane rozwiązanie wykorzystujące przełącznik komunikacyjny umożliwia szybkie wykonanie prototypu układu sterującego składającego się z różnorodnego oprogramowania porozumiewającego się za pomocą protokółu UDP/IP. Za dostosowanie różnej prędkości pracy poszczególnych elementów odpowiedzialny jest przełącznik. Nie istnieją istotne ograniczenia co do liczby źródeł danych oraz ich odbiorników. Program przełącznika został napisany w języku C w sposób parametryczny. Parametrem jest tu liczba odbiorników. W zależności od tej liczby różna ilość procesów buforów oraz nadajników zostanie wygenerowana przy tworzeniu przełącznika. Przełącznik został wykorzystany do weryfikacji sprawności działania systemu sterownik-symulator pojazdu podwodnego, potwierdzając słuszność przyjętej koncepcji.

5. Literatura

[1] Barfield, W., Furness III, T. A.: *Virtual Environments and Advanced Interface Design*. Oxford University Press, New York, 1995.

[2] Hunt C.: TCP/IP *Network Administration*. O'Reilly, Cambridge, 1998.

[3] Seet G., Tan K. C., Lau W. S., Low E.: *An Advanced Pilot Training and Control System for Underwater Robotic Vehicles*. Journal of Robotics and Mechatronics, Vol. 12, No. 3, 2000, str. 275-280.

[4] Zielinski C.: *Distributed Software for Robots Systems*. International Journal of Intelligent Robot Design and Production. Vol. 1, No. 1, 1994. str. 11-24.

[5] Zielinski C., Seet G.: *A Prototyping Tool for Validating Complex Robot Systems*, 6th Int. Conf. on Control, Automation, Robotics and Vision, ICARCV'2000, 5-8 December 2000, Singapore. (na CD-ROM).

[6] ---: *TCP/IP Programmers Guide*. QNX Software Systems Ltd. Canada, 1998.

[7] ---: *TCP/IP User's Guide*. QNX Software Systems Ltd. Canada, 1998.

[8] ---: *QNX Operating System – System Architecture*. QNX Software Systems Ltd. Canada, 1997.

[9] ---: *G Programming Reference Manual*. National Instruments, January 1998 Edition

[10] ---: *SideWinder Force feedback SDK – Programmer's Reference*. Microsoft Corporation. USA, 1997.