STUDIA INFORMATICA 2003
Volume 24 Number 3 (55)

Włodzimierz GAJDA
Katolicki Uniwersytet Lubelski, Wydział Matematyczno-Przyrodniczy

SIECIOWY SYSTEM PROGRAMOWANIA ROBOTÓW

Streszczenie. Artykuł prezentuje *Sieciowy System Programowania Robotów*. Zadaniem systemu jest umożliwienie programowania robotów przez sieć TCP/IP. Każdy robot dostępny w systemie jest podłączony do dowolnego komputera w sieci. Bez względu na fizyczną lokalizację każdy użytkownik może programować dowolne roboty i udostępniać własne podprogramy sterujące robotami innym użytkownikom zarejestrowanym w systemie.

Słowa kluczowe: robotyka, programowanie robotów, sieciowy system sterowania.

NETWORKED SYSTEM FOR ROBOTS PROGRAMMING

Summary. The paper presents networked system for robots programming. The system can be used to control and program robots through the TCP/IP network. Each robot present in the system can be connected to arbitrary chosen host in our network. Every user registered in the system can control the behaviour of any robot regardless of its physical location. Users can also export their subprograms; exported subprograms can be called by other users.

Keywords: robotics, robot programming, networked control system.

1. Specyfikacja warunków pracy systemu

Artykuł prezentuje sieciowy system programowania i kontroli robotów. Zakłada się, że sieć wykorzystuje do komunikacji protokoły rodziny TCP/IP. Termin **robot** oznacza urządzenie, które jest podłączone do komputera poprzez dowolny interfejs (np. port szeregowy, port równoległy, USB, łącze wykorzystujące podczerwień lub dedykowaną kartę IO). Komunikacja komputer-robot może być dwukierunkowa: robot może zarówno odbierać sygnały sterujące od komputera, jak i produkować dane i przekazywać je do komputera. Nie

czyni się żadnych założeń dotyczących języka programowania robotów. Konieczne jest jedynie, by sterowanie robota było możliwe z poziomu dowolnego języka programowania umożliwiającego komunikację protokołem TCP.

Dysponuje się m robotami ($m > 0$), które są podłączone do dowolnych spośród n komputerów ($n > 0$). Powyższe założenie nie ogranicza liczby robotów, jakie można podłączyć do jednego komputera (ograniczenia te mogą wynikać z fizycznych właściwości robotów oraz komputerów). Przyjmuje się oznaczenia:

$R = \{R_i: i=1, ..., m\}$ – zbiór robotów

$U = \{U_i: i=1, ..., k\}$ – zbiór użytkowników.

W zajęciach bierze udział k użytkowników (każdy użytkownik ma do dyspozycji jeden z komputerów w pracowni $0 \le k \le n$).

Użytkownicy mogą zarejestrować podprogramy oraz zdarzenia. Zbiór podprogramów dostępnych w systemie oznacza się jako

$P = \{P_i: i=1, ..., r\}, \; r \ge 0,$

a zbiór zdarzeń

$E = \{E_i: i=1, ..., s\}, \; s \ge 0.$

System przechowuje informacje dotyczące robotów, użytkowników, podprogramów oraz zdarzeń w postaci zbiorów R, U, E, P.

Dla każdego robota, procedury oraz zdarzenia jest określona lista uprawnień *ACL (ang. Access Control List)*. Lista uprawnień jest zbiorem użytkowników uprawnionych do kontroli danego obiektu. Zatem dla każdego elementu zbiorów R, P, E określa się zbiory:

$ACL(R_i) \subset U$, dla $i=1, 2, ..., m$;

$ACL(P_j) \subset U$, dla $j=1, 2, ..., r$;

$ACL(E_k) \subset U$, dla $k=1, 2, ..., s$.

System umożliwia rejestrację i wyrejestrowanie robotów, użytkowników, procedur oraz zdarzeń. Operacje takie sprowadzają się do dodawania oraz usuwania elementów zbiorów R, U, P, E. Zarządzanie uprawnieniami w systemie polega na wykonywaniu operacji dodawania i usuwania elementów zbiorów $ACL(R_i)$, $ACL(P_j)$, oraz $ACL(E_k)$ dla i=1, 2, ..., m; j=1, 2, ..., r; k=1, 2, ..., s.

System realizuje przekierowywanie żądań pomiędzy robotami a użytkownikami i umożliwia, by użytkownik U_i mógł programować l_i ($0 \le l_i \le m$, $i=1, 2, ... ,k$) dowolnych spośród m robotów.

2. Cechy systemu

Jednym z celów, jakie od początku przyświecały realizacji projektu, było stworzenie otwartej specyfikacji. Zdefiniowanie i udokumentowanie interfejsu systemu umożliwia dołączanie do systemu robotów pochodzących od różnych producentów i posiadających różne właściwości. Dzięki temu jest możliwość stopniowego wzbogacania pracowni o kolejne urządzenia. Równie ważne jest to, że w systemie mogą być wykorzystane urządzenia tak odmienne jak robot przemysłowy, kamera cyfrowa czy cyfrowy miernik napięcia elektrycznego. W ten sposób proponowany system może zostać wykorzystany zarówno do prowadzenia zajęć z robotyki, jak i przeprowadzania doświadczeń chemicznych oraz fizycznych sterowanych programowo.

Ponieważ każdy robot może być kontrolowany przez wielu użytkowników, zatem stwarza to możliwość przesyłania danych przekazywanych przez robota (np. oscyloskop) do wszystkich użytkowników. W ten sposób każdy uczestnik zajęć dysponuje na bieżąco wynikami pomiarów. W podobny sposób możemy zrealizować tworzenie bazy danych przeprowadzonych pomiarów.

Najważniejszą cechą systemu jest umożliwienie wywoływania procedur zdefiniowanych przez innych użytkowników. Stwarza to możliwość stopniowego konstruowania coraz bardziej złożonego systemu oraz ułatwia organizacje pracy grupowej.

3. Architektura systemu

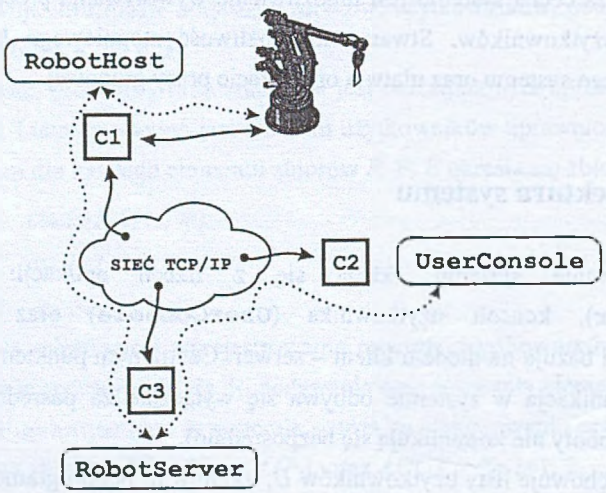
Oprogramowanie systemu składa się z trzech aplikacji: serwera robotów (`RobotServer`), konsoli użytkownika (`UserConsole`) oraz sterownika robota (`RobotHost`) i bazuje na modelu klient – serwer. Centralnym punktem systemu jest serwer robotów. Komunikacja w systemie odbywa się wyłącznie za pośrednictwem serwera (tj. użytkownicy i roboty nie komunikują się bezpośrednio).

Serwer przechowuje listy użytkowników U, robotów R, podprogramów P oraz zdarzeń E. Ponadto serwer przechowuje listy uprawnień. Każdy robot, procedura oraz zdarzenie posiadają własną listę uprawnień zawierającą nazwy uprawnionych użytkowników. Jeśli użytkownik jest obecny na liście ACL danego robota, oznacza to, że może wysyłać do robota instrukcje sterujące.

Uruchomienie systemu należy rozpocząć od uruchomienia programu serwera. Serwer robotów może być uruchomiony na dowolnym komputerze w sieci. Po uruchomieniu serwera listy R, U, P, E są puste; można więc rozpocząć rejestracje robotów i użytkowników w systemie.

Rejestracja robota w systemie polega na uruchomieniu programu **RobotHost** na komputerze, do którego robot został podłączony. Program **RobotHost** steruje pracą robota i jest związany z konkretnym urządzeniem. Każdy robot posiada własny charakterystyczny program sterujący. Cechą wspólną wszystkich programów sterujących pracą robotów jest fakt, że odbierają one komunikaty od serwera i zamieniają je na sygnały wywołujące działanie robota. Dołączenie do systemu nowego typu robota wymaga napisania programu **RobotHost** dedykowanego dla tego robota. Rejestracja robota dodaje nowy element do zbioru *R*, zaś wyrejestrowanie robota z systemu usuwa go ze zbioru *R*.

Rejestracja użytkownika polega na uruchomieniu programu **UserConsole** oraz zalogowaniu do systemu. Program **UserConsole** umożliwia jedynie wysyłanie prostych komunikatów do robotów (program nie zawiera środowiska programowania). Programowanie robotów w językach wysokiego poziomu (C, C++, Pascal, Logo) realizuje się wykorzystując bibliotekę **UserConsoleLib**. Biblioteka ta (dostępna w formacie DLL, LIB oraz w postaci kodu źródłowego) zawiera zestaw funkcji realizujących proces rejestracji w systemie oraz komunikację z robotami. Każdy zarejestrowany użytkownik traktowany jest jako jeden z elementów zbioru *U*.



Rys. 1. Proces komunikacji w systemie. C1, C2 oraz C3 symbolizują komputery, linie ciągłe oznaczają łącza fizyczne, zaś linie przerywane – otwarte połączenia komunikacyjne

Fig. 1. Communication process. C1, C2 and C3 symbolise computers, solid lines represent physical connections and dotted lines stand for opened communication channels

Użytkownicy mogą rejestrować w systemie podprogramy i zdarzenia wykorzystując bibliotekę `UserConsoleLib`. Zarejestrowane procedury są elementami zbioru *P*, zaś zarejestrowane zdarzenia – elementami zbioru *E*.

Podczas rejestracji robota, zdarzenia lub procedury serwer tworzy – początkowo pustą – listę uprawnień *ACL* dla rejestrowanego obiektu. Wyrejestrowanie obiektu powoduje usunięcie odpowiedniej listy *ACL*.

Wykorzystując panel administracyjny serwera można wykonywać operacje na listach ACL obiektów zarejestrowanych w systemie. Administrator może przyznawać uprawnienia dowolnym użytkownikom do kontroli dowolnych spośród obiektów obecnych w systemie.

Programowanie robotów można realizować na dwa sposoby. Pierwszy sposób polega na przygotowywaniu programu `RobotHost`. Każdy program `RobotHost` jest wykonywany na komputerze, do którego fizycznie podłączono robota. Program udostępnia pewien zestaw komend sterujących pracą robota. Użytkownik wysyła komunikaty, które sprowadzają się do wywoływania funkcji oferowanych przez program `RobotHost`.

Drugi sposób polega na pisaniu programów wykorzystujących bibliotekę `UserConsoleLib`. Programy takie możemy pisać np. w środowisku Borland Builder lub gcc. Wówczas program sterujący pracą robota jest wykonywany na komputerze użytkownika, zaś sterowanie odbywa się poprzez wywołanie funkcji wysyłających komunikaty do robota. Jeśli użytkownik pracuje na komputerze A, robot jest podłączony do komputera B, zaś serwer jest uruchomiony na komputerze C, wówczas:

- program napisany przez użytkownika jest wykonywany na komputerze A;
- program w trakcie wykonania wywołuje funkcje sterujące pracą robota;
- wywołanie funkcji powoduje wysłanie odpowiednich komunikatów z komputera A do serwera pracującego na komputerze C;
- serwer (po sprawdzeniu uprawnień) wysyła komunikaty do komputera B, do którego robot jest podłączony;
- oprogramowanie `RobotHost` uruchomione na komputerze B, po odebraniu komunikatu od serwera, generuje odpowiednie sygnały sterujące robotem.

Pierwszy proponowany sposób służy do programowania konkretnych urządzeń fizycznych. Program `RobotHost` udostępnia pewien zbiór podprogramów dotyczących konkretnego urządzenia fizycznego (implementacja podprogramów nie odwołuje się poprzez sieć do innych urządzeń fizycznych).

Drugi proponowany sposób programowania służy do tworzenia robotów składających się z wielu urządzeń fizycznych, podłączonych do różnych komputerów. Program wykonywany przez użytkownika komunikuje się z wieloma robotami (dokładniej z wieloma programami `RobotHost`).

4. Komunikacja w systemie

Do komunikacji pomiędzy serwerem a użytkownikami oraz pomiędzy serwerem a robotami system wykorzystuje protokół RCP/1.0 opisany w pracy [4]. Ponieważ zasadniczym celem jest stworzenie systemu możliwie uniwersalnego, stąd stosowany protokół jest możliwie ogólny. Protokół RCP/1.0 jest protokołem warstwy aplikacji bazującym na protokole TCP.

Serwer po uruchomieniu czeka pasywnie na nadchodzące połączenia na porcie 12000. Program `RobotHost` otwiera połączenie TCP z serwerem na zadanym porcie. Połączenie to jest otwarte aż do momentu wyrejestrowania robota z systemu. Użytkownicy otwierają połączenie z serwerem programem `UserConsole` lub wywołując procedury inicjalizacyjne biblioteki `UserConsoleLib`. Połączenie serwer-użytkownik jest otwarte aż do chwili wylogowania użytkownika z systemu.

Jedynymi kanałami komunikacyjnymi w systemie są otwarte połączenia serwer-robot oraz serwer-użytkownik. Użytkownicy i roboty nie komunikują się bezpośrednio.

Pierwszym żądaniem wysyłanym zarówno przez robota jak i użytkownika po otwarciu połączenia z serwerem jest żądanie rejestracji. Jeśli rejestracja robota powiodła się (informuje o tym odpowiedź serwera), wówczas serwer może przekierowywać komunikaty otrzymywane od użytkowników do danego robota. Prawidłowa rejestracja użytkownika w systemie umożliwia danemu użytkownikowi wysyłanie sygnałów sterujących przez serwer do robotów. Komunikaty pochodzące od użytkownika i adresowane do konkretnego robota są przekazywane robotowi wówczas, gdy dany użytkownik jest obecny na liście uprawnień ACL robota.

Druga klasa zapytań i odpowiedzi zdefiniowanych przez protokół RCP/1.0 dotyczy transferu danych. Zdefiniowane zapytania umożliwiają transfer zapytań od użytkowników do robotów oraz przekazywanie odpowiedzi przez roboty do użytkowników. Nie nałożono żadnych ograniczeń dotyczących przekazywanych ciągów bajtów. Rola protokołu ogranicza się do zapewnienia przekazywania komunikatów pomiędzy użytkownikami i robotami. Określenie składni i semantyki zapytań wiąże się z konkretnym robotem i jego programem `RobotHost`.

Po zakończeniu pracy roboty oraz użytkownicy przesyłają do serwera żądanie wyrejestrowania z systemu.

5. Przykłady robotów

W obecnym stanie rozwoju system jest stosowany wyłącznie do prowadzenia zajęć z robotyki w szkole średniej. Elementami napędowymi robotów są silniki prądu stałego, silniki krokowe oraz serwomechanizmy. Kontrolery elektroniczne wykorzystywane w systemie są podłączane do portu równoległego komputera. Szczegóły dotyczące sterowania silnikami krokowymi opisano w pracy [2].

Mechanika robotów jest wykonywana z klocków Lego Technic. Wykonane roboty to chwytak, frezarka oraz model wiertarki.

6. Zastosowania

Wykorzystując omawiany system można programować roboty w dowolnym języku, dzięki czemu zajęcia z robotyki mogą stanowić kontynuację czy pewne urozmaicenie ogólnych przedmiotów dotyczących programowania w językach np. C++ lub Pascal. Jest to szczególnie istotne w odniesieniu do szkół średnich. W ciągu ostatnich kilku lat coraz częściej mówi się o korzyściach wynikających z nauczania programowania robotów na poziomie szkoły średniej [3, 6, 7]. Wykorzystanie omawianego systemu umożliwia realizację nauczania robotyki podczas lekcji z przedmiotu „elementy informatyki".

Drugi ważny aspekt dotyczy otwartości systemu. System umożliwia stopniowe dołączanie różnorodnych robotów. Ponadto kolejno dodawane urządzenia mogą współpracować pomimo różnic w budowie, czy metodach sterowania. Robot dołączany do systemu jest widziany przez interfejs udostępniany programem `RobotHost`. Pozwala to na programowanie robota bez konieczności znajomości szczegółów jego budowy oraz metod sterowania. Oczywiście zanim będziemy mogli używać konkretnego urządzenia w roli robota w systemie musimy dla niego przygotować oprogramowanie `RobotHost`.

Opisany model obejmuje m.in. sytuację, w której dysponujemy jednym robotem, zaś w zajęciach bierze udział wielu uczestników. Każdy uczestnik może napisać własny kod sterujący pracą robota, a następnie go przetestować. Administrator przyznaje kolejno uczestnikom uprawnienia umożliwiające sterowanie robotem. Sytuacja taka występuje np. wówczas, gdy mamy do czynienia z drogim sprzętem. Wykorzystanie systemu sieciowego z jednej strony eliminuje konieczność przełączania robota, z drugiej strony umożliwia każdemu uczestnikowi rzeczywistą pracę z robotem.

Drugi przykład dotyczy sytuacji, w której pracownia jest wyposażona w kilka robotów, zaś liczba uczestników zajęć jest większa od tej liczby robotów. Realizacja zajęć wymaga

wtedy, aby każdy z uczestników zajęć napisał kilka programów sterujących pracą każdego z robotów. Wykorzystując listy uprawnień do sterowania robotami można łatwo rekonfigurować system pozwalając kolejnym uczestnikom programować kolejne roboty.

LITERATURA

1. Gajda W.: The System of Networked, Mechanical LOGO Turtles, Proceedings of EUROLOGO 2001, Linz 2001.
2. Gajda W.: Mechaniczne żółwie, V Lubelskie Akademickie Forum Informatyczne, Kazimierz Dolny 2001.
3. Gajda, W.: „Komputery, roboty i dzieci". XVIII Konferencja "Informatyka w Szkole", Mielec 2001.
4. Gajda W.: Robots Communication Protocol RCP/1.0, Materiały konferencyjne „Informatyka – badania i zastosowania", Kazimierz Dolny 2003.
5. Martin, F. G.: Circuits to control: Learning engineering by designing Lego Robots, Phd., MIT 1994.
6. Piasecki M., Rogaliński P.: Nauczanie podstaw programowania za pomocą zestawów MinStorms. XVII Konferencja „Informatyka w Szkole", Mielec 2001.
7. Piasecki, M., Rogaliński P.: Dydaktyka eksperymentalna z wykorzystaniem programowalnych klocków Lego Mindstorms. Proceed. of XVIII Krajowa Konferencja Polioptymalizacja i Komputerowe Wspomaganie Projektowania, Mielno 2000.

Recenzent: Prof. dr hab. inż. Bolesław Pochopień

Wpłynęło do Redakcji 26 marca 2003 r.

Abstract

The paper presents networked system for robots programming. The main aim of this work is to create the system that enhance the courses of robotics. We propose an open system that uses network to facilitate remote control of robots. The system is open in the sense that we can use arbitrary device that uses the protocol described in the paper.

We consider the laboratory containing $n > 0$ computers. We assume that computers are connected and use TCP/IP protocols for communication. There are $m > 0$ robots present in

our laboratory. Each robot can be connected to arbitrary computer in the network. There are $0 \le k \le n$ users that can control robots. Users can register events and procedures in the system. Thus the system consists of four sets: the set of robots, the set of users, the set of events and the set of procedures. We denote these sets with

$U = \{U_i: i=1, ..., k\}$,

$R = \{R_i: i=1, ..., m\}$,

$P = \{P_i: i=1, ..., r\}$,

$E = \{E_i: i=1, ..., s\}$.

We define *Access Control List* for every element of the sets R, P and E. The ACL is a set of users that can control the given object:

$ACL(R_i) \subset U, \quad i=1, 2, ..., m;$

$ACL(P_j) \subset U, \quad j=1, 2, ..., r;$

$ACL(E_k) \subset U, \quad k=1, 2, ..., s.$

The system acts as redirector of requests. It redirects the requests sent by users to appropriate robots. The user U_i can control l_i ($0 \le l_i \le m$, $i=1, 2, ... ,k$) arbitrary robots in the system.

Adres

Włodzimierz GAJDA: Katolicki Uniwersytet Lubelski, Wydział Matematyczno-Przyrodniczy, Al. Racławickie 14, 20-950 Lublin, Polska, gajdaw@kul.lublin.pl