AUTOMATYKA • 2009 • Tom 13 • Zeszyt 3

Konrad Kułakowski*

Robust – model komunikacji Mindstorms NXT-PC**

1. Wprowadzenie

W ostatnim czasie można zaobserwować pojawienie się na rynku kilku prostych i tanich platform robotowych, przeznaczonych dla osób zaczynających swoją przygodę z robotami mobilnymi. Dzięki temu studenci kierunków technicznych takich jak automatyka i robotyka, informatyka, czy informatyka stosowana zyskują świetne narzędzie do wprawiania się w konstrukcji systemów sterowania takimi robotami, a prowadzący zajęcia otrzymują wspaniałe narzędzie pracy dydaktycznej. Do takich konstrukcji należy choćby platforma Hexor II [18], czy Mindstorm NXT [3]. Przedmiotem niniejszej pracy będzie właśnie owa druga konstrukcja, a ściślej, model komunikacyjny platformy Robust pozwalający na efektywne i niezawodne zdalne sterowania prostym robotem mobilnym wykonanym z wykorzystaniem zestawu Lego Mindstorms NXT.

We wprowadzeniu do prezentowanej pracy została przedstawiona potrzeba tworzenia takiej platformy oraz zostały omówione: Lego Mindstorms NXT i Lejos – wbudowana maszyna wirtualna Javy na platformie sprzętowej Mindstorms. Pozostała część pracy jest poświęcona omówieniu modelu komunikacyjnego platformy Robust oraz przedstawieniu aktualnego stanu jej implementacji.

1.1. Motywacja

Sercem zestawu *Lego Mindstorms NXT* sterującym całym urządzeniem jest *Intelligent Brick* – inteligentna kostka zawierająca procesor serii ARM oraz wyposażona 256 kilobajtów pamięci ROM oraz 64 kilobajty pamięci RAM. Całość uzupełnia kontroler protokołu *Bluetooth* (BT) oraz port USB 2.0. Te nader skromne, w porównaniu z obecnie dostępnymi na rynku komputerami klasy PC, zasoby sprzętowe nie pozwalają na uruchamianie za ich pomocą bardziej skomplikowanych aplikacji. Głównym ograniczeniem jest niewielka ilość pamięci dostępna w urządzeniu. Także sam procesor ARM7TDMI nie zawsze oferuje wystarczającą wydajność. W takiej sytuacji jedynym rozwiązaniem dla osób chcących

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie
** Praca finansowana przez MNiSW w ramach grantu: N N516 228735

stworzyć nieco bardziej zaawansowaną aplikację sterującą z wykorzystaniem *Mindstorms NXT* jest przeniesienie całości sterowania na komputer PC i zarządzanie inteligentną kostką zdalnie z wykorzystaniem protokołu *Bluetooth*. Eksperymenty przeprowadzone przez autora dowiodły jednak, że w praktyce takie rozwiązanie również nie działa w sposób należyty. Częste odczyty danych sensorycznych za pośrednictwem protokołu BT, niezbędne do nawigacji w czasie rzeczywistym poruszającym się robotem powodują przepełnienie łącza komunikacyjnego pomiędzy komputerem PC i NXT. W rezultacie algorytm sterujący otrzymuje dane przeterminowane i niewiarygodne, a część komunikatów zostaje bezpowrotnie zgubiona.

Rozwiązaniem tej sytuacji zaproponowanym przez autora jest hybrydowa platforma sterująca urządzeniami klasy *Mindstorms NXT* umożliwiająca z jednej strony przeniesienie zaawansowanej logiki sterującej urządzeniem na komputer klasy PC, a z drugiej zapewniająca właściwą, zbliżoną do warunków oferowanych przez systemy czasu rzeczywistego, obsługę elementów sensorycznych.

1.2. Lego Mindstorms NXT

Lego Mindstorms NXT to programowalny zestaw *Lego* przeznaczony do tworzenia prostych robotów mobilnych [9, 3]. Został wprowadzony na rynek w czerwcu 2006 roku, zastępując starszą konstrukcję *Lego Mindstorms RCX*. W skład sprzedawanego zestawu wchodzą standardowo:

- cztery czujniki:
  - odległości (*ultrasonic sensor*),
  - dźwięku (*sound sensor*),
  - dotyku (*touch sensor*),
  - światła (*light sensor*),
- cztery serwomechanizmy wyposażone w tachometry,
- programowalny sterownik – „inteligentna kostka" (*intelligent brick*), do którego podłączone są wszystkie czujniki oraz silniczki.

Całość uzupełniona jest zestawem klocków umożliwiających zbudowanie różnego rodzaju robotów mobilnych, zarówno kołowych jak i konstrukcji kroczących.

Początkowo zestaw *Lego Mindstorms* był pomyślany jako zaawansowana technicznie zabawka dla dzieci. Z czasem odkryto jego przydatność w nauczaniu robotyki [10] i sztucznej inteligencji [16], na poziomie szkoły średniej oraz studiów licencjackich i magisterskich [1, 16]. Dzięki swojej prostocie oraz niskiej, w porównaniu z profesjonalnymi zestawami jak np. dostarczane przez *Mobile Robots Research* [14], cenie, stał się świetną platformą dydaktyczną i badawczą [8, 5, 7, 19]. O jego wielkiej popularności może świadczyć szereg prac poświęconych tworzeniu inteligentnych modeli z wykorzystaniem *Lego Mindstorms* [12, 3]. Oprócz standardowego wbudowanego systemu operacyjnego (*firmware*) pozwalającego tworzyć proste programy sterujące z wykorzystaniem środowiska *LabView*, dostępne są inne systemy udostępniające możliwość programowania w językach, takich jak np.: Java, Ada, C, C++, Objective C, Python, Perl, C#, Ruby, Lisp, Prolog i inne.

1.3. Lejos

Jednym z niestandardowych systemów operacyjnych działających pod kontrolą jednostki operacyjnej ARM7TDMI na platformie *Lego Mindstorms NXT* jest *Lejos* – wbudowana maszyna wirtualna *Javy* [4, 15, 2]. Pomimo tego, że wiele konstrukcji dostępnych w obecnie obowiązującym standardzie *Javy* nie jest dostępnych w *Lejos*, to przyzwyczaiwszy się do ograniczeń składniowych akceptowanego w tym środowisku dialektu języka, z powodzeniem można pisać całkiem skomplikowane programy sterujących robotem pod tę platformę. Specyficzny interfejs umożliwiający obsługę urządzeń takich jak czujniki czy serwomechanizmy znajduję się w pakiecie *lejos.**.

Programy pod *Lejos* są kompilowane skrośnie (*cross compiled*) na komputerze PC a potem instalowane na kostce NXT (*Intelligent Brick*).

2. Platforma ROBUST

Platforma *Robust* składa się z dwóch głównych części: programu sterującego kostką NXT, oraz biblioteki rezydującej na komputerze klasy PC i zapewniającej komunikację ze sprzętem (rys. 1). Obie części są napisane w *Javie*, przy czym część rezydująca na *Mindstorms NXT (RobustNXT)* działa w środowisku *Lejos*, natomiast do skorzystania z biblioteki *(RobustPC)* wystarcza posiadanie dowolnej maszyny wirtualnej *Java* w wersji 6 kompatybilnej z aktualnie używanym systemem operacyjnym.

Rys. 1. Platforma Robust

Biblioteka *RobustPC* udostępnia użytkownikowi możliwość synchronicznej i asynchronicznej komunikacji z robotem. Komunikacja asynchroniczna jest zrealizowana za pomocą mechanizmu pozwalającego na subskrypcję po stronie bibliotecznej danego zdarzenie sensorycznego. Komunikacja synchroniczna jest zaimplementowana jako zdalne wywołanie metody z wykorzystaniem własnego, wewnętrznego protokołu komunikacji z wykorzystaniem protokołu BT (*Bluetooth*).

2.1. Model komunikacji

Jednym z głównych zadań platformy Robust jest zapewnienie właściwej jakości komunikacji z wykorzystaniem protokołu BT pomiędzy modułami *RobustNXT* i *RobustPC*. Aby to osiągnąć, w przyjętym rozwiązaniu aktywne sprawdzanie stanu czujników platformy NXT ograniczono do modułu *RobustNXT*, wprowadzając jednocześnie po stronie

RobustPC mechanizm subskrypcji na zdarzenia (*listeners*). Użytkownik chcący monitorować stan wybranego czujnika musi zatem stworzyć odpowiednią klasę implementującą akcję, która ma zostać wywołana w momencie wystąpienia interesującego go zdarzenia, a następnie zarejestrować ją w odpowiednim obiekcie udostępniającym metody dostępowe do wybranego czujnika. Przykładem może być interfejs komunikacji asynchronicznej *RobustAPITouchAsync* (rys. 2) dla standardowego czujnika dotyku znajdującego się w każdym zestawie z Mindstorms NXT. Dostarcza on metodę *registerTouchListener* oraz interfejs *TouchLister*.

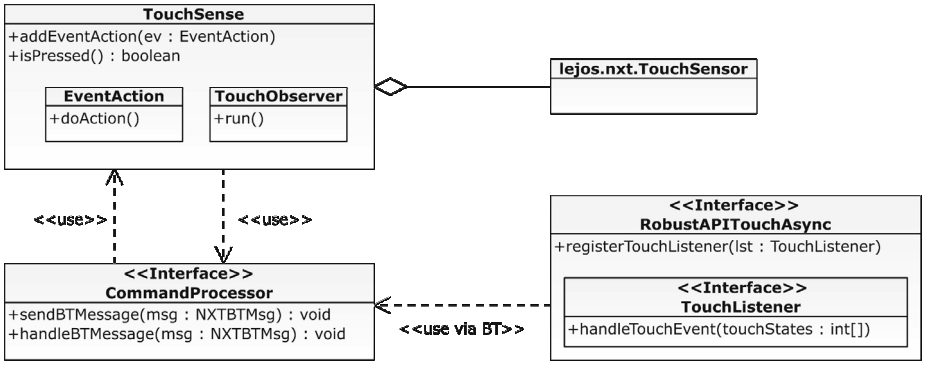
Programista, aby zostać powiadomiony w sposób asynchroniczny o zmianie stanu czujnika dotyku, musi stworzyć własną implementację interfejsu *TouchListener* i zarejestrować ją za pomocą metody *registerTouchListener()* w obiekcie implementującym interfejs *RobustAPITouchAsync*. Po rejestracji, w momencie zmiany stanu czujnika dotyku, zostanie na podanym obiekcie wywołana metoda *handleTouchEvent*, gdzie wartość tablicy *touchStates* będzie oznaczała stan czujnika, a numer komórki tablicy będzie wskazywał, którego (może być ich kilka) czujnika przekazana zmiana dotyczy.

```
public interface RobustAPITouchAsync extends RobustAPI {
        public interface TouchListener {
                void handleTouchEvent(int[] touchStates);
        }
        public void registerTouchListener(TouchListener
touchListener);
}
}
```

Rys. 2. Interfejs komunikacji asynchronicznej dla standardowego czujnika dotyku

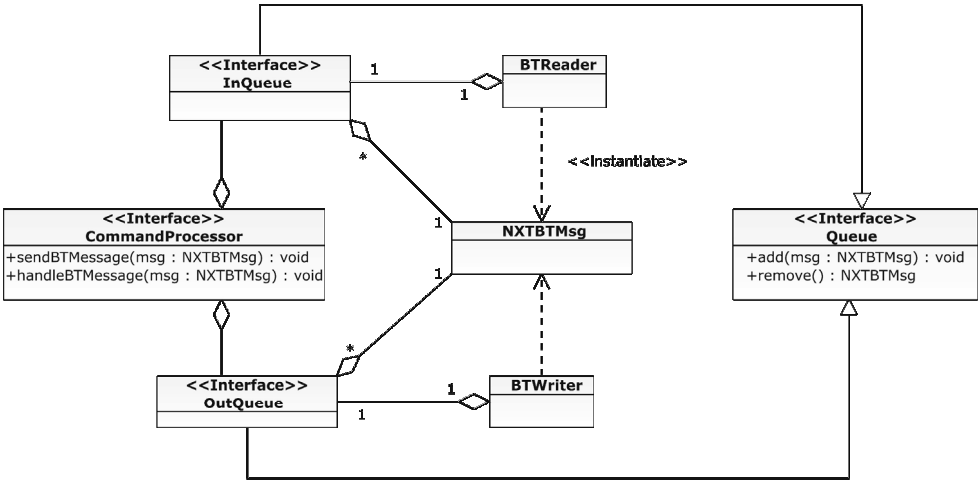
W praktyce, w momencie rejestracji obiektu klasy *TouchListener*, następuje wysłanie poprzez łącze komunikacyjne prośby o subskrypcję na zdarzenie zmiany stanu czujnika dotyku do klasy implementującej funkcjonalność interfejsu *CommandProcessor*, co skutkuje właściwą rejestracją w obiekcie klasy *TouchSense* zapewniającej obsługę czujnika po stronie *RobustNXT*. W tym rozwiązaniu, aktywne próbkowanie stanu czujnika jest ograniczone do klasy *TouchObserver*, która, będąc jedną ze składowych obiektu *TouchSense*, działa w osobnym wątku wbudowanej maszyny wirtualnej *Lejos* (rys. 3). Komunikat niosący informację o stanie czujnika zostaje wysłany do modułu *RobustPC* tylko wtedy, gdy stan się rzeczywiście zmieni. Tym samym komunikacja za pośrednictwem łącza BT zostaje zredukowana do niezbędnego minimum.

Oprócz asynchronicznego odczytu danych sensorycznych platforma *Robust* umożliwia synchroniczne pobranie wskazań poszczególnych czujników. Wywołanie metody synchronicznej zarezerwowane jest jednak dla sytuacji, w której zachodzi potrzeba jednorazowego odczytu takich danych i nie powinno być używane zbyt często.



Rys. 3. Schemat klas biorących udział w asynchronicznym odczycie danych sensorycznych

Zarówno synchroniczny, jak i asynchroniczny odczyt danych sensorycznych na platformie *Robust* jest zaimplementowany na podstawie specjalnie w tym celu zaprojektowanym mechanizmie transportowym, wykorzystującym dostarczane przez *Lejos* API do komunikacji z użyciem BT (rys. 4).



Rys. 4. Ogólny schemat warstwy komunikacyjne platformy Robust

Sercem tego mechanizmu jest pięć klas: *NXTBTMsg*, *BTReader*, *BTWriter*, *InQueue*, *OutQueue* odpowiedzialnych za odbieranie i wysyłanie wiadomości (rys. 3). W momencie wysyłania komunikatu (reprezentowanego każdorazowo przez instancję klasy *NXTBTMsg*) przez jeden z modułów, jest on wkładany do kolejki komunikatów wychodzących *OutQueue*, z niej jest pobierany przez obiekt klasy *BTWriter*, serializowany do strumienia bajtów, a następnie wysyłany do drugiego z modułów. Proces odbioru komunikatu przebiega analo-

gicznie. Wpierw jest on deserializowany przez obiekt klasy *BTReader*, wkładany do kolejki *InQueue*, po to by w końcu trafić do właściwej instancji klasy *CommandProcessor*, która zajmuje się jego dalszym przetwarzaniem.

2.2. Aktualny stan rozwoju platformy

W chwili obecnej platforma *Robust* posiada wsparcie dla większości standardowych czujników dostarczanych wraz z zestawem *Mindstorms* NXT. Są to: czujnik dotyku, czujnik natężenia światła oraz czujnik odległości. Ponad to zostało dodane wsparcie dla dodatkowych czujników, takich jak: podsystemu rozpoznawania i przetwarzania obrazów (NXTCam-v2) oraz kompasu magnetycznego (CMPS-Nx v. 2.0). Obydwa urządzenia pochodzą od firmy *Mindsensors* [13]. W planach jest implementacja obsługi kolejnych urządzeń pochodzących od niezależnych dostawców. W szczególności jednym z urządzeń, których obsługa zostanie dodana w pierwszej kolejności, będzie żyroskop NGY1044 będący produktem firmy *HiTechnic* [6].

Platforma *Robust* umożliwia również ograniczoną kontrolę dostępnych w zestawie *Mindstorms* NXT serwomechanizmów. Oferowane wsparcie umożliwia swobodną nawigację pojazdem trójkołowym po płaskiej nawierzchni, w którym dwa koła są napędzane, dwoma niezależnymi serwomechanizmami.

Moduł *RobustPC* jest multi-platformowy. Platforma *Robust* została z powodzeniem przetestowana w środowisku *Microsoft Windows (MS Windows XP SP 2)*, *Linux (Mandriva 2008 Professional)* oraz *Mac OS X (Leopard*, *Mac OS X 10.5.6)*.

Aktualnie trwają pracę nad zwiększeniem czytelności i stabilności kodu. Rozważane jest także wprowadzenie częściowej kontroli transmisji komunikatów, co może mieć znaczenie w przypadku słabej jakości połączenia BT.

Docelowo cała platforma zostanie udostępniona za darmo w internecie.

3. Podsumowanie

Powstanie platformy *Robust*, jest wynikiem ponad półrocznej pracy autora nad założeniami konstrukcyjnymi oraz implementacją rozwiązania. Bezpośrednią inspiracją do podjęcia tego tematu był brak analogicznego rozwiązania w oprogramowaniu dostarczanym wraz z robotem. Co prawda standardowy model komunikacji bezprzewodowej pomiędzy *Mindstorms NXT* a komputerem PC umożliwia odczyt danych sensorycznych, ale nie wspiera komunikacji asynchronicznej. Stąd proponowane przez producenta rozwiązanie, wobec ograniczonej przepustowości łącza BT, praktycznie uniemożliwia śledzenie zdalne w czasie rzeczywistym większej liczby czujników.

W prezentowanym artykule został pokazany model komunikacyjny zaimplementowany w platformie *Robust*. Dzięki wprowadzeniu asynchronicznego zdalnego odczytu czujników, znacząco poprawiła się charakterystyka czasowa aplikacji sterujących robotem z wykorzystaniem tej platformy. W przeprowadzonych testach nie wystąpiły problemy

wcześniejsze związane ze zbyt małą wydajnością łącza komunikacyjnego. Pozwala to mieć nadzieję, że *Robust* stanie się ciekawą biblioteką uzupełniającą standardowe rozwiązanie i zyska szerszą popularność wśród tych, którzy systemy sterowania dla *Mindstorms NXT* piszą w *Javie*.

Literatura

[1] Atmatzidou S. *et al*., *The use of LEGO Mindstorms in elementary and secondary education: game as a way of triggering learning*. [w:] Workshop proceedings of International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2008.

[2] Bagnall B., *Building robots with Java Brains*. Variant Press, 2007.

[3] Baum D., *Extreme Mindstorms: an Advanced Guide to Lego Mindstorms*. Appress, ed. 1, 2000.

[4] Ferrari G. *et al*., *Programming LEGO Mindstorms with Java*. Syngress Publishing, Inc., Rockland, MA, USA, 2002.

[5] Heck B.S.N., Scott C., Ferri A.A., *A LEGO Expermment for Embedded Control System Design*. IEEE Control Systems Magazine, 2004.

[6] HiTechnic: http://www.hitechnic.com/.

[7] Iversen T.K. *et al*., *Model-checking real-time control programs: verifying LEGO(R)MINDSTORMSTM systems using UPPAAL*. IEEE Computer Society, 2000.

[8] Klassner F., Anderson S.D., *LEGO MindStorms: not just for K-12 anymore*. Robotics & Automation Magazine, IEEE, 2003.

[9] Lego, *Lego Mindstorms*. http://mindstorms.lego.com/, 2009.

[10] Leva A., *Turning a toy into a didactic industrial regulator*. Proceedings of the 44th IEEE Conference on Decision and Control, Seville 2005.

[12] Miguel A. *et al*., *LEGO Mindstorms Masterpieces – Building and Programming Advanced Robots*. Syngress Publishing, Inc., Rockland, MA, USA, 2003.

[13] Mindsensors: http://www.mindsensors.com.

[14] Mobile Robots Inc.: http://www.activrobots.com/.

[15] Moral J.A.B., *Develop Lejos programs step by step*. on-line manual: http://www.juanantonio.info, 2008.

[16] Parson S., Sklar E., *Teaching AI using LEGO Mindstorms*. American Association for Artifcial Intelligence, 2004.

[17] Rieber J.M., Wehlan H., Allgoewer F., *The Roborace contest*. IEEE Control Systems Magazine, 2004.

[18] Stenzel. *Projekt Hexor*. http://www.stenzel.com.pl, 2009.

[19] Woo E., MacDonald B.A., Trepanier F., *Distributed Mobile Robot Application Infrastructure*. Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2003.