Metoda projektowania układów sterowania autonomicznych robotów mobilnych Część 2. Przykład zastosowania

Piotr Trojanek[**], **Cezary Zieliński**[*], **Tomasz Kornuta**[**], **Tomasz Winiarski**[**]

[*]Przemysłowy Instytut Automatyki i Pomiarów PIAP,
[**]Instytut Automatyki i Informatyki Stosowanej Politechniki Warszawskiej

Streszczenie: Ten dwuczęściowy artykuł przedstawia metodę projektową umożliwiającą określenie zarówno struktury, jak i sposobu działania układów sterowania autonomicznych robotów. Część pierwsza koncentruje się na aparacie formalnym i ogólnych przesłankach metody projektowej, natomiast część druga przedstawia przykład obrazujący sposób stosowania metody opisanej w części pierwszej artykułu.

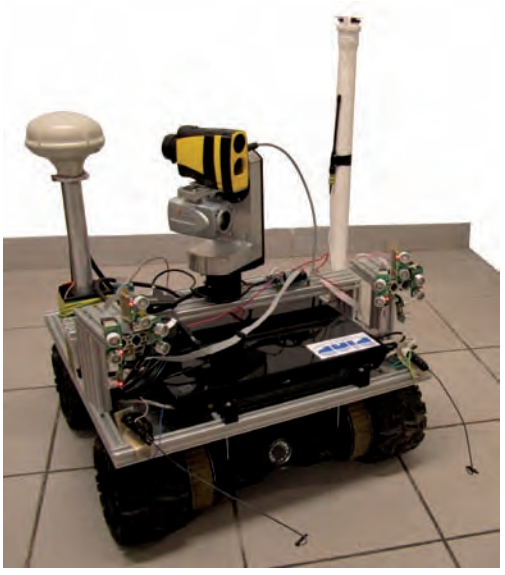
Słowa kluczowe: układ sterowania robota, roboty autonomiczne

W pierwszej części artykułu przedstawiono podstawy teoretyczne metody projektowania układów sterowania robotów. Przykład zastosowania tej metody jest przedmiotem tej części artykułu.

1. Projekt układu sterującego robotem Scout

Podstawowym pytaniem, na które musi odpowiedzieć projektant jest: *z ilu agentów powinien się składać system oraz za jakie zachowania powinny być odpowiedzialne poszczególne agenty?* Odpowiedź na tak postawione pytanie uzależniona jest od zadań, jakie stawiane są systemowi. Należy jednak pamiętać, że zbyt duża liczba agentów prowadzi do rozrostu komunikacji między nimi. Natomiast zbyt mała liczba prowadzi do nadmiernej komplikacji układu sterującego każdego z nich. Zarówno struktura automatu skończonego staje się skomplikowana, jak i funkcje przejścia określane są skomplikowanymi definicjami. Istotne jest też logiczne pogrupowanie zadań, tak aby struktura układu sterującego była zrozumiała. Tak więc potrzebna jest tu rozwaga. Problem dotyczy raczej sztuki projektowania, ale opisana w części pierwszej formalizacja zagadnienia wraz z poniżej opisanymi wskazówkami powinny ułatwić rozwiązanie tak sformułowanego problemu. Wspomniane wskazówki zostaną przedstawione na przykładzie projektu, który został zrealizowany w rzeczywistości.

Wpierw zostanie sformułowany problem. Układ sterowania robotem mobilnym Scout (rys. 1) ma autonomicznie przemieścić tego robota do wcześniej wskazanego celu. Na początku cel jest wskazywany przez operatora, a następnie robot autonomicznie ma dojechać do tego celu, omijając przeszkody oraz unikając przechyłów grożących wywróceniem się pojazdu. Wskazane jest, aby robot posiadał wielostopniowy układ zabezpieczający przed katastrofami, z tym że to operator powinien móc określić, które z zabezpieczeń powinny być aktualnie włączone.



Rys. 1. Robot Scout
Fig. 1. The Scout robot

Dopiero po sformułowaniu problemu należy określić niezbędne zasoby sprzętowe potrzebne do jego rozwiązania. Większość problemów projektowych rozwiązywana jest iteracyjnie. Określenie niezbędnych zasobów sprzętowych przeplata się z formułowaniem zadań cząstkowych stawianych przed projektowanym systemem. Iteracje zazwyczaj wiążą się ze zwiększaniem szczegółowości informacji, ale również mogą być spowodowane pojawieniem się konkurencyjnego rozwiązania. W przedstawianym przykładzie oczywistym jest, iż musi istnieć napędzana platforma jezdna umożliwiająca dotarcie do celu. Cel ruchu może być określany względem aktualnego położenia robota poprzez odpowiednie nakierowanie kamery i mechanicznie sprzężonego z nią dalmierza laserowego. Oba te czujniki mogą być zainstalowane w ruchomej głowicy o dwóch stopniach swobody, umieszczonej na platformie mobilnej. W trakcie jazdy robot powinien omijać przeszkody – do realizacji tego celu robot może być wyposażony w czujniki ultradźwiękowe. Dodatkowo można go zabezpieczyć mechanicznymi zderzakami (wąsami), zatrzymującymi pojazd w momencie, gdy najedzie na przeszkodę. Inklinometry potrzebne są, by unikać nadmiernych przechyłów, natomiast kompas oraz GPS umożliwiają bieżącą lokalizację robota.

Powyżej ogólnie sformułowany problem sterowania trzeba podzielić na proste zadania. Zadanie uważamy za proste,

jeżeli jesteśmy w stanie określić zarówno zasoby sprzętowe niezbędne do jego realizacji, jak i algorytm sterujący, który doprowadzi do jego wykonania. Układ sterujący Scoutem ma do zrealizowania następujące proste zadania:

1) sterowanie silnikami platformy jezdnej (ruch Scouta),
2) wykrywanie przeszkód,
3) określenie położenia platformy jezdnej,
4) określenie orientacji (wraz z pochyleniem) platformy jezdnej,
5) wyznaczenie trajektorii ruchu,
6) nasłuch poleceń operatora,
7) realizacja ruchów ręcznych platformą,
8) realizacja ruchów ręcznych głowicą,
9) przerwanie pracy autonomicznej,
10) informowanie operatora o stanie systemu,
11) konfiguracja systemu (włączanie/wyłączanie zabezpieczeń),
12) określenie kierunku do celu,
13) określenie odległości do celu,
14) ruchy głowicą.

Powyższe zadania muszą być zrealizowane za pomocą efektorów i receptorów stanowiących sprzęt elektroniczny i mechaniczny składający się na ciało (postać) robota. Oczywiście zasoby sprzętowe muszą zostać tak dobrane, aby możliwa była realizacja wymienionych zadań. Scout jest wyposażony w efektory, czyli: silniki platformy jezdnej, silniki głowicy i ekran konsoli operatora, oraz proprioreceptory, czyli: kodery (enkodery) silników platformy jezdnej, kodery (enkodery) silników głowicy, i klawiaturę konsoli (lub mysz albo joystick), a nadto w następujące eksteroreceptory: kamerę, dalmierz laserowy, GPS, kompas, inklinometry, zderzaki (wąsy) i czujniki ultradźwiękowe.

Zadaniem projektanta jest skojarzenie zadań z zasobami sprzętowymi niezbędnymi do ich realizacji. Innymi słowy należy wskazać, które efektory i receptory są niezbędne do realizacji każdego z zadań. W prezentowanym przykładzie poszczególne zadania wymagają następujących zasobów sprzętowych:

1) sterowanie silnikami platformy jezdnej (ruch Scouta): silniki i enkodery platformy jezdnej,
2) wykrywanie przeszkód: czujniki ultradźwiękowe, wąsy,
3) określenie położenia platformy jezdnej: GPS, enkodery platformy jezdnej,
4) określenie orientacji platformy jezdnej (wraz z pochyleniem): kompas, inklinometry, enkodery platformy jezdnej,
5) wyznaczenie trajektorii ruchu: GPS, kompas, inklinometry, czujniki ultradźwiękowe, wąsy (do określenia aktualnej pozycji robota i przeszkód) oraz enkodery silników głowicy i dalmierz (do określenia celu),
6) nasłuch poleceń operatora: klawiatura,
7) realizacja ruchów ręcznych platformą: klawiatura,
8) realizacja ruchów ręcznych głowicą: klawiatura,
9) przerwanie pracy autonomicznej: klawiatura,
10) informowanie operatora o stanie systemu: ekran,
11) konfiguracja systemu (włączanie/wyłączanie zabezpieczeń): klawiatura,
12) określenie kierunku do celu: enkodery silników głowicy, GPS, kompas, inklinometry,
13) określenie odległości do celu: dalmierz,
14) ruchy głowicą: silniki i enkodery głowicy oraz urządzenia do wprowadzania poleceń operatora.

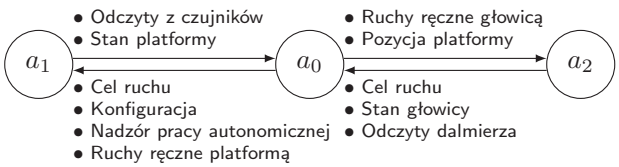
Przy definiowaniu struktury systemu należy dążyć do takiej definicji agentów, aby zadania przydzielone każdemu z nich mogły być wykonane sekwencyjnie. Ponadto każdy z agentów musi dysponować niezbędną informacją do wykonania zadania. Nie oznacza to, że musi ją zdobyć za pomocą własnych czujników – może tę informację uzyskać od innego agenta. Innymi słowy wykorzystuje albo czujniki lokalne, albo zdalne. Należy jednak pamiętać, że przesyłanie informacji między agentami zajmuje czas, a wszelkie opóźnienia w pętlach sprzężenia zwrotnego utrudniają stabilizację układu i pogarszają jakość sterowania. W związku z tym należy przyjąć generalną regułę, która mówi, że jeżeli agent nie jest w stanie samodzielnie uzyskać niezbędnej dla niego informacji, to musi uzyskać ją od agenta bezpośrednio z nim skomunikowanego. Innymi słowy, należy wykluczyć przesyłanie informacji przez wielu pośredników. Odstępstwo od tej generalnej reguły może być tolerowane jedynie wówczas, gdy niezbędna informacja nie musi być uaktualniana często.

Stosując powyżej nakreślone zasady, należy rozpatrzyć kilka struktur układu sterowania. Przy projektowaniu układu sterowania Scoutem wzięto pod uwagę struktury: 3-, 4-, 5- i 6-agentowe. Struktura 3-agentowa okazała się najlepsza, a więc ona zostanie tu opisana. Dzieląc układ sterowania na agenty, należy wziąć pod uwagę zarówno podział zasobów sprzętowych, jak i logiczne grupowanie zadań. Dla struktury 3-agentowej należy wskazać trzy grupy zadań. Są to: określenie celu ruchu (agent a_2), nawigacja autonomiczna (agent a_1) oraz komunikacja z operatorem (agent a_0). Tak więc trzem agentom można przypisać następujące zadania. Agent a_1 sterujący platformą jezdną (nawigujący autonomicznie): steruje silnikami platformy jezdnej, lokalizuje platformę jezdną (określa jej położenie i orientację), wykrywa przeszkody, generuje trajektorię ruchu platformy jezdnej. Realizacja tych zadań wymaga, aby agent a_1 był wyposażony w silniki platformy jezdnej oraz GPS, kompas, inklinometry, czujniki ultradźwiękowe i wąsy. Do wyznaczenia trajektorii ruchu niezbędna jest informacja o celu ruchu. Ponieważ agent a_1 nie ma bezpośredniego dostępu do głowicy oraz stowarzyszonych z nią czujników, musi ona być uzyskana od agenta a_2. Sugerowałoby to bezpośrednie skomunikowanie obu agentów. Oczywiście jest to możliwe, ale nie jest konieczne, ponieważ wyznaczenie celu ruchu i autonomiczna nawigacja w jego kierunku są realizowane rozdzielnie w czasie, a co więcej, czas przełączenia między jednym zadaniem a drugim jest znaczny. Tak więc można zrezygnować z bezpośredniego połączenia między agentami a_1 i a_2. Uprości to implementację układu sterowania. Ruchy ręczne platformą mobilną wymagają dostępu do zadajnika położenia, z którego będzie korzystał operator. W tym przypadku zadajnikiem jest klawiatura komputera.

Sugerowałoby to przydzielenie klawiatury agentowi a_1, ale operator musi również sterować ruchami głowicy i najlepiej, jeżeli do tego celu użyje tego samego zadajnika. To natomiast sugeruje przydzielenie klawiatury agentowi a_2. Ponieważ agenty a_1 i a_2 nie zostały skomunikowane bezpośrednio, zdecydowano się przyporządkować klawiaturę koordynatorowi a_0.

Jak wspomniano, agent a_2 steruje ruchem głowicy, aby wyznaczyć cel ruchu, a więc realizuje dwa zadania: steruje silnikami głowicy i wyznacza cel ruchu. To operator powoduje ruch głowicy za pomocą klawiatury, ale to urządzenie zostało przyporządkowane koordynatorowi. Wynika z tego, że potrzebna jest bezpośrednia komunikacja pomiędzy koordynatorem i agentem a_2. Należy zwrócić uwagę, że w tym przypadku nie możemy zrezygnować z wymagania bezpośredniości połączenia między tymi agentami, gdyż reakcja systemu na wciśnięcie klawisza klawiatury musi być szybka i co więcej, jest to operacja powtarzana wielokrotnie w pętli. Do wyznaczenia celu ruchu we współrzędnych globalnych potrzebna jest znajomość aktualnego położenia Scouta. Tę informację można uzyskać od agenta a_1 za pośrednictwem koordynatora. Ta informacja przekazywana jest jednokrotnie, wtedy gdy agent a_2 ma wyznaczyć cel ruchu. Co więcej, czas wyznaczenia tego celu nie jest krytyczny, a więc możemy zadowolić się łącznością pośrednią.

Agent a_0, komunikujący się z operatorem, realizuje interfejs z operatorem oraz koordynuje pracę całości systemu. Dodatkowo agent ten przechowuje informację niezbędną do realizacji zadań przez agenty a_1 i a_2. Zlecając tym agentom realizację któregoś z przypisanych im zadań, przesyła informację niezbędną do realizacji tego zadania. Wynikowa struktura układu sterowania została przedstawiona na rys. 2. Pokazano tam też, jakie informacje przekazywane są między agentami.



Rys. 2. Podział systemu na agenty
Fig. 2. Decomposition of the system into agents

Gdy struktura układu sterowania jest określona, można przystąpić do definicji zachowań poszczególnych agentów. Każde zachowanie określane jest przez parę funkcji: funkcję przejścia oraz warunek końcowy (terminalny). Zachowanie odpowiedzialne jest za realizację pojedynczego zadania. Funkcja przejścia określa, jak będzie ewoluował stan agenta, natomiast warunek końcowy określa, kiedy ma zostać przerwane to zachowanie. Aby zdefiniować te funkcje, trzeba określić ich argumenty oraz wartości, które mają wytworzyć, a więc obrazy: ${}_xc_j$ i ${}_yc_j$ dla $j = 0, 1, 2$. Ścisłe matematyczne zdefiniowanie tych obrazów zajmuje dużo miejsca, więc na potrzeby tego artykułu ograniczymy się jedynie do zaznaczenia ich na rysunkach: 3, 5 i 7. Dla zwięzłości pominięte zostaną również funkcje odpowiedzialne za agregację danych z eksteroreceptorów w odczyty czujników wirtualnych.

Graf stanu agenta a_1 sterującego platformą jezdną (rys. 4) obrazuje, w jaki sposób dochodzi do przełączania między realizowanymi zadaniami. W grafach tego typu węzły odpowiadają kodowi realizującemu zachowanie ((10)cz.1). Parametrem każdego zachowania jest funkcja przejścia ${}^mf'_{c_1}$ definiująca zadanie. Zakończenie realizacji zadania (zachowania) następuje wskutek spełnienia warunku końcowego ${}^mf_{\tau_1}$ sprawdzanego wewnątrz kodu ((10)cz.1). Agent a_1 realizuje trzy zadania (zachowania), a więc $m = 1, \ldots, 3$. Decyzja dotycząca wyboru następnego zadania do realizacji podejmowana jest na podstawie sprawdzenia spełnienia warunków etykietujących łuki wychodzące z węzła reprezentującego właśnie zakończone zachowanie. Na rys. 4 powroty ze stanów ${}^1a_1$, ${}^2a_1$ oraz ${}^3a_1$ do stanu ${}^0a_1$ odbywają się wskutek spełnienia odpowiednich warunków końcowych, stąd umieszczono je jako etykiety przy tych łukach, natomiast przejścia ze stanu ${}^0a_1$ do wymienionych trzech stanów zależą od polecenia wydanego przez operatora. Polecenie to pojawi się w buforze ${}_xc_{T_{10}}$. W stanie ${}^0a_1$ dokonywana jest selekcja zadania do wykonania. Bez wnikania tu w szczegóły matematyczne każdej z funkcji przejścia można określić ich ogólne struktury – podać związki pomiędzy ich argumentami a wytwarzanymi wartościami.

Funkcja przejścia ${}^1f_{c_1}$ (składająca się z ${}^1f'_{c_1}$ i ${}^1f_{\tau_1}$) agenta a_1 odpowiedzialna jest za ręczne sterowanie platformą mobilną przez operatora.

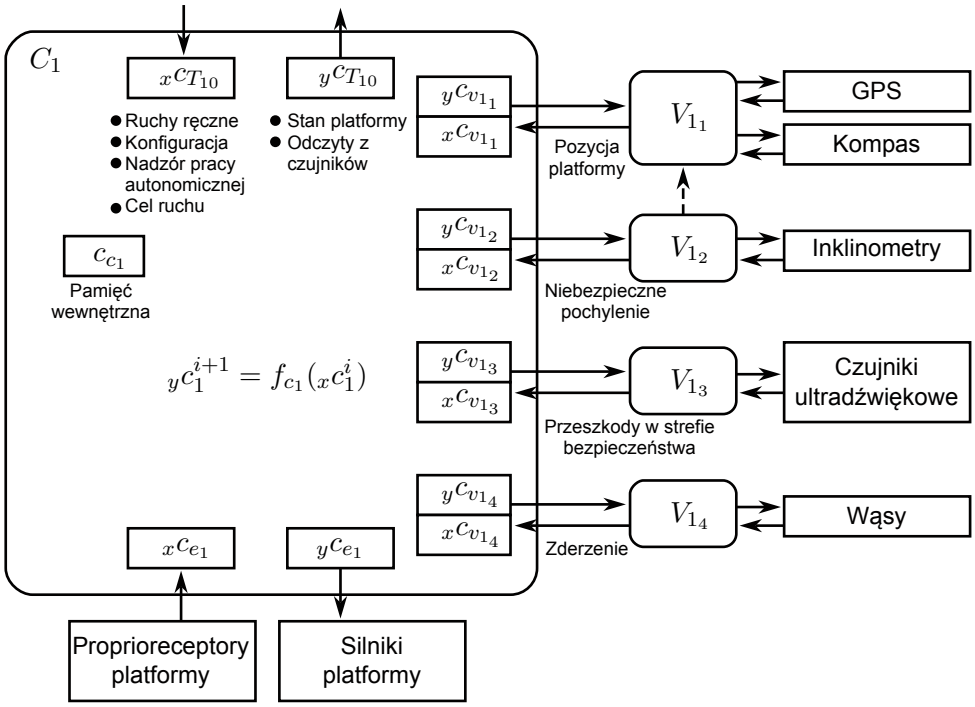
$$ {}^1f_{c_1} \triangleq \left\{ \begin{bmatrix} {}_yc_{e_1} \\ {}_yc_{T_{10}} \\ c_{c_1} \end{bmatrix} = {}^1f'_{c_1}({}_xc_{T_{10}},\ {}_xc_{V_{1_{(2,3,4)}}}, c_{c_1}) \right\} \quad (1) $$

Operator naciskając klawisz przekazuje swoje intencje koordynatorowi a_0, a ten z kolei przekazuje za pośrednictwem bufora transmisyjnego informację o pożądanym makrokroku ruchu platformy jezdnej do wejściowego bufora transmisyjnego ${}_xc_{T_{10}}$ agenta a_1, stąd ta wartość musi stanowić argument funkcji przejścia ${}^1f'_{c_1}$. Jeżeli parametry konfiguracyjne zawarte w pamięci c_{c_1} agenta a_1 wskazują, że należy w trakcie ruchów ręcznych dbać o to, aby platforma nadmiernie się nie przechyliła oraz nie zderzyła się z ewentualną przeszkodą, to przy generacji makrokroku ruchu trzeba wziąć pod uwagę informacje uzyskane z czujników wirtualnych, a więc ${}_xc_{V_{1_2}}$, ${}_xc_{V_{1_3}}$ i ${}_xc_{V_{1_4}}$. Informacje z tych czujników wykorzystywane są do wygenerowania dopuszczalnego makrokroku ruchu, umieszczanego w ${}_yc_{e_1}$, stworzenia odpowiedzi koordynatorowi, przesyłanej poprzez ${}_yc_{T_{10}}$, oraz zapamiętania w pamięci c_{c_1}, informacji dotyczącej wykonywanego zlecenia. To zachowanie powtarzane jest do chwili, gdy poprzez bufor transmisyjny agent uzyska operatorskie polecenie zakończenia ruchów ręcznych.

Wówczas warunek końcowy:

$$ {}^1f_{\tau_1} \triangleq \{ {}_xc_{T_{10}} = \text{stop} \} \quad (2) $$

zostanie spełniony. Powyższy skrótowy zapis dotyczy oczywiście pewnej składowej bufora transmisyjnego, co nie zostało wyraźnie zaznaczone. W dalszej części tego artykułu konsekwentnie nie będą wyróżniane składowe poszczególnych struktur danych (pamięci wewnętrznej oraz



Rys. 3. Struktura agenta a_1
Fig. 3. Structure of the agent a_1

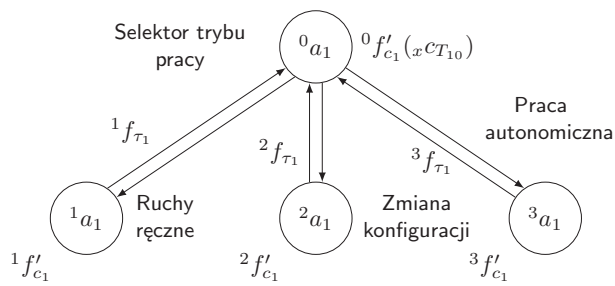
buforów/obrazów), aby uniknąć wprowadzania zbyt wielu oznaczeń, które nie miałyby większego znaczenia dla tego poglądowego wykładu.

Funkcja przejścia ${}^2f_{c_1}$ odpowiedzialna jest za zmianę konfiguracji:

$$ {}^2f_{c_1} \triangleq \left\{ \begin{bmatrix} {}_yc_{T_{10}} \\ c_{c_1} \end{bmatrix} = {}^2f'_{c_1}({}_xc_{T_{10}}, c_{c_1}) \right\} \tag{3} $$

czyli odebranie celu ruchu od koordynatora bądź włączanie/wyłączanie czujników bezpieczeństwa (aktywacja/dezaktywacja ich wpływu). Ponieważ zmiana konfiguracji odbywa się jednokrokowo, więc funkcja przejścia (3) zmienia marker zawarty w c_{c_1} z *FALSE* na *TRUE*, a więc w skrótowej postaci można zapisać warunek końcowy jako

$$ {}^2f_{\tau_1} \triangleq \{c_{c_1} = TRUE\} \tag{4} $$

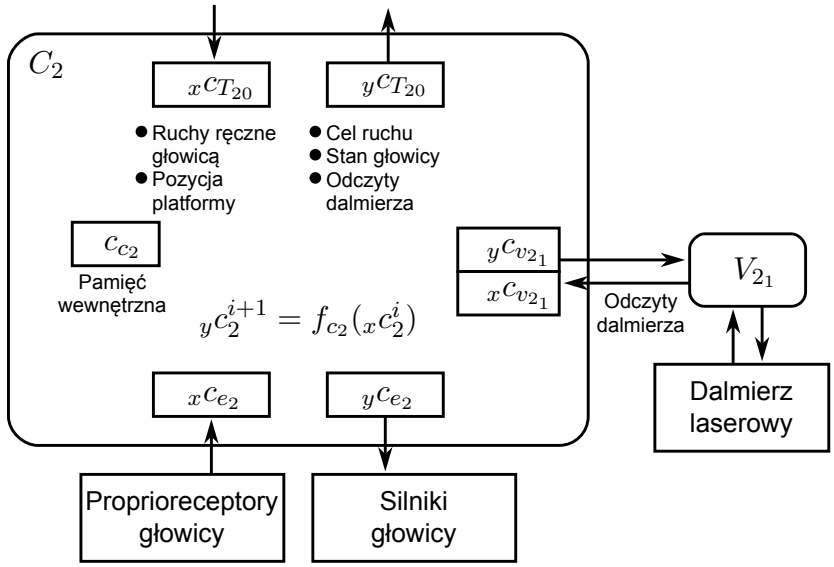


Rys. 4. Graf stanu agenta a_1
Fig. 4. State graph of the agent a_1

Funkcja przejścia – ${}^3f_{c_1}$ odpowiedzialna jest za ruch autonomiczny:

$$ {}^3f_{c_1} \triangleq \left\{ \begin{bmatrix} {}_yc_{T_{10}} \\ {}_yc_{V_1} \\ {}_yc_{e_1} \\ c_{c_1} \end{bmatrix} = {}^3f'_{c_1}({}_xc_{V_{1_{(1,3)}}}, {}_xc_{e_1}, c_{c_1}) \right\} \tag{5} $$

Do wygenerowania kolejnego makrokroku ruchu funkcja ${}^3f'_{c_1}$ wykorzystuje algorytm *Vector Field Histogram (VFH)* [1, 2]. Do tego celu przede wszystkim używa informacji uzyskanej z czujnika wirtualnego V_{1_3}, a przekazywanej układowi sterowania agenta poprzez bufor ${}_xc_{V_{1_3}}$. Czujnik ten przekształca odczyty uzyskane z czujników ultradźwiękowych na postać akceptowaną przez algorytm *VFH*. Algorytm w swej pierwotnej postaci zakłada, że otrzyma odczyty ze skanera laserowego, a więc będzie dysponował informacją wysokiej rozdzielczości o położeniu przeszkód. Czujnik wirtualny zajmuje się przekształceniem (powieleniem) informacji o małej rozdzielczości, tak aby dostosować się do wymaganego przez algorytm *VFH* formatu danych. Algorytm *VFH* wyznacza nową pożądaną prędkość liniową i obrotową robota, umieszczając ją w ${}_yc_{e_1}$. Do tego celu korzysta z wiedzy o rozkładzie przeszkód w otoczeniu robota, aktualnym stanie robota (jego prędkości liniowej i obrotowej oraz położeniu i orientacji, uzyskanych z enkoderów za pośrednictwem ${}_xc_{e_1}$), zagregowanych odczytach GPS i kompasu (otrzymanych za pośrednictwem ${}_xc_{V_{1_1}}$) oraz informacji o celu ruchu (przechowywanej w c_{c_1}). Algorytm *VFH* w każdym kroku określa rozkład wiarygodności istnienia przeszkód w sektorach kątowych okalających robota. Jako kierunek ruchu wybierany jest sektor, którego kierunek jest zgodny w najwyższym stopniu



Rys. 5. Struktura agenta a_2
Fig. 5. Structure of the agent a_2

z kierunkiem ruchu do celu, a jednocześnie wiarygodność braku przeszkód w tym sektorze jest wysoka.

Warunek końcowy, który sprawdzany jest po każdej iteracji algorytmu *VFH*, zdefiniowany jest jako:

$$
\begin{aligned}
{}^3f_{\tau_1} \triangleq \quad & \big\{(\text{uchyb} \approx 0) \vee ({}_x c_{T_{10}} = \text{npo}) \vee \\
& c_{c_1}[2]({}_x c_{V_{1_2}} > \text{np}) \vee c_{c_1}[3]({}_x c_{V_{1_3}} < \text{dk}) \vee \\
& c_{c_1}[4]({}_x c_{V_{1_4}} = \text{wak})\big\}
\end{aligned} \tag{6}
$$

gdzie npo – nowe polecenie operatora, np – nadmierny przechył, dk – dystans krytyczny do przeszkody, wak – wąs aktywny (natrafił na przeszkodę). Poprzez ${}_x c_{T_{10}}$ operator realizuje nadzór nad działaniem autonomicznym, natomiast uchyb określany jest jako różnica pomiędzy celem ruchu zapamiętanym w c_{c_1} a aktualnym położeniem robota uzyskiwanym z GPS, a więc otrzymanym z czujnika wirtualnego V_{1_1} poprzez ${}_x c_{V_{1_1}}$. Wartość zmiennych boole'owskich $c_{c_1}[k]$, $k = 2, 3, 4$ (gdzie k odpowiada numerowi czujnika wirtualnego) określana jest przez polecenie konfiguracyjne operatora, który decyduje o aktywacji lub nie poszczególnych zabezpieczeń – robi to funkcja przejścia (3).

Agent a_2 odpowiedzialny jest za sterowanie ruchami głowicy i wyznaczenie celu ruchu. Struktura wewnętrzna agenta a_2 przedstawiona jest na rys. 5, natomiast graf jego stanów pokazuje rys. 6.

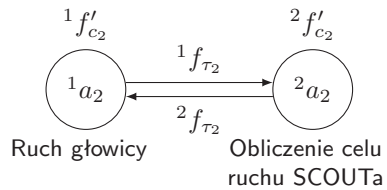
Funkcja przejścia ${}^1f'_2$ odpowiedzialna jest za przemieszczanie głowicy:

$$
{}^1f_{c_2} \triangleq \left\{ \begin{bmatrix} {}_y c_{e_2} \\ {}_y c_{T_{20}} \end{bmatrix} = {}^1f'_{c_2}({}_x c_{T_{20}}, {}_x c_{e_2}) \right\} \tag{7}
$$

Wpływ na silniki poruszające głowicą wywierany jest poprzez ${}_y c_{e_2}$. Rozkazy determinujące ruchy głowicy otrzymywane są poprzez ${}_x c_{T_{20}}$, natomiast stan głowicy przekazywany jest koordynatorowi poprzez ${}_y c_{T_{20}}$. Stan głowicy określany jest na podstawie odczytów z enkoderów uzyskiwanych poprzez ${}_x c_{e_2}$.

Warunek końcowy, który sprawdzany jest po realizacji każdego makrokroku ruchu, zdefiniowany jest jako:

$$
{}^1f_{\tau_2} \triangleq \quad \big\{ {}_x c_{T_{20}} = \text{stop} \big\} \tag{8}
$$



Rys. 6. Graf stanu agenta a_2
Fig. 6. State graph of the agent a_2

Gdy operator wyda polecenie stop, ruch głowicy jest przerywany, natomiast agent a_2 przechodzi do wyznaczenia parametrów celu.

Funkcja przejścia ${}^2f'_2$ zajmuje się określeniem celu ruchu robota.

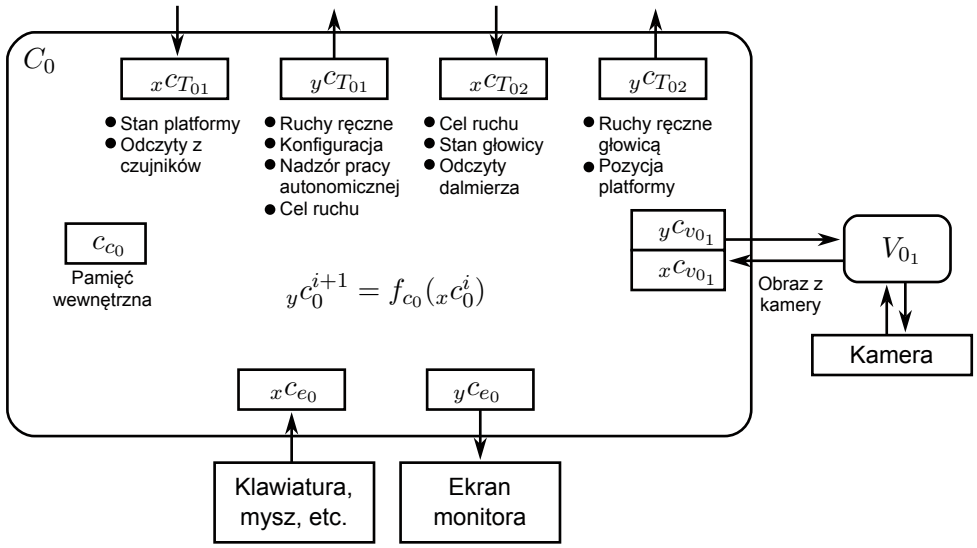
$$
{}^2f_{c_2} \triangleq \left\{ \begin{bmatrix} {}_y c_{T_{20}} \\ c_{c_2} \end{bmatrix} = {}^2f'_{c_2}({}_x c_{T_{20}}, {}_x c_{e_2}, {}_x c_{V_{2_1}}) \right\} \tag{9}
$$

Pozycja celu obliczana jest na podstawie aktualnej pozycji platformy jezdnej otrzymanej poprzez ${}_x c_{T_{20}}$ od koordynatora a_0, pozycji głowicy uzyskanej z enkoderów poprzez ${}_x c_{e_2}$ oraz odczytu dalmierza uzyskanego poprzez ${}_x c_{V_{2_1}}$. Pozycja platformy jezdnej ustalana jest przez koordynatora na podstawie odczytów z GPS, kompasu i inklinometrów, i przekazywana agentowi a_2 równocześnie z przesłaniem komendy stop kończącej ruchy ręczne głowicą.

Pamięć wewnętrzna agenta a_2, czyli c_{c_2}, zawiera marker sygnalizujący zakończenie obliczeń, działający identycznie jak ten wykorzystany w (4). Ponieważ obliczenie celu ruchu odbywa się jednokrokowo, warunek końcowy przyjmuje postać:

$$
{}^2f_{\tau_2} \triangleq \{c_{c_2} = TRUE\} \tag{10}
$$

Agent a_0 stanowi interfejs z operatorem oraz koordynuje pracę całego systemu. Jego struktura wewnętrzna przedstawiona jest na rys. 7. Ponieważ agent a_0 zleca wykonanie zadań albo agentowi a_1 albo a_2 w sposób rozłączny, zadania własne wykonywane są równolegle tylko z jednym



Rys. 7. Struktura agenta a_0
Fig. 7. Structure of the agent a_0

z zadań zlecanych tym agentom. Wynika z tego, że można skonstruować jedną funkcję f_{c_0} albo tyle funkcji ${}^m f_{c_0}$, ile łącznie jest zadań zlecanych agentom a_1 i a_2. Ze względu na dominację zadań własnych, wybieramy pojedynczą f_{c_0}.

$$f_{c_0} \triangleq \left\{ \begin{bmatrix} {}_yc_{e_0} \\ {}_yc_{V_{0_1}} \\ {}_yc_{T_{01}} \\ {}_yc_{T_{02}} \\ c_{c_0} \end{bmatrix} = f'_{c_0}({}_xc_{T_{01}},\ {}_xc_{T_{02}},\ {}_xc_{V_{0_1}},\ {}_xc_{e_0},\ c_{c_0}) \right\} \tag{11}$$

Funkcja (11), na podstawie informacji uzyskanej od agenta a_1 poprzez ${}_xc_{T_{01}}$ oraz agenta a_2 poprzez ${}_xc_{T_{02}}$, obrazu z kamery otrzymanego poprzez ${}_xc_{V_{0_1}}$ oraz polecenia od operatora otrzymanego poprzez ${}_xc_{e_0}$, wykorzystując ${}_yc_{e_0}$, wyświetla na ekranie informację o stanie systemu oraz obraz z kamery wraz z krzyżem celowniczym. W c_{c_0} zapamiętywane są niektóre informacje otrzymane od pozostałych agentów, natomiast polecenia dla tych agentów zlecane są poprzez ${}_yc_{T_{01}}$, ${}_yc_{T_{02}}$. Zachowano również możliwość wpływania na kamerę poprzez ${}_yc_{V_{0_1}}$. Może to być wykorzystane do rekonfiguracji kamery, np. zoom.

Cały system powoływany jest do życia w następującej kolejności: wpierw agent a_0, który z kolei powołuje do życia agenty a_1 i a_2. Każdy z agentów powołuje do życia swoje czujniki wirtualne oraz efektory i je inicjuje.

2. Podsumowanie

Dobrze dobrana struktura systemu oraz jednolitość specyfikacji różnych jego funkcji w istotny sposób redukują powstawanie błędów w trakcie implementacji. Jeżeli nawet takie błędy powstaną, łatwo jest je wykryć i naprawić. Zaproponowana metoda projektowania powstała na bazie zarówno prac teoretycznych [3, 4, 6], jak i bogatych doświadczeń w implementacji układów sterowania wieloma robotami [5, 6]. Zaprezentowany tu przykład dotyczący wprowadzenia elementów autonomicznych zachowań do sterowania robotem Scout z natury rzeczy mógł być przedstawiony jedynie szkicowo. Pełna definicja funkcji przejścia i warunków końcowych wymaga szczegółowego określenia struktury buforów, obrazów i pamięci wewnętrznej. To oczywiście wymaga miejsca przekraczającego ramy artykułów publikowanych w PAR.

Należy jeszcze dodać, że zarówno agenty a_0, a_1 i a_2, jak i czujniki wirtualne, z których korzystają, muszą wykrywać sytuacje awaryjne, w szczególności nieprawidłowości w działaniu sprzętu. Minimalnym wymaganiem jest, aby informacja o awarii została przekazana operatorowi, a więc dotarła do agenta a_0. W szczególnych przypadkach można się pokusić o zaprojektowanie alternatywnych zachowań, które nie będą wymagały wykorzystania uszkodzonego sprzętu. Oczywiście konieczne jest wtedy wbudowanie pewnej redundancji w projektowany układ. Na bazie przytoczonego tu formalizmu możliwe jest prowadzenie rozważań i na ten temat.

Podziękowania

Praca była finansowana z funduszy statutowych PIAP. Pragniemy podziękować następującym studentom Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej, członkom Koła Naukowego *Bionik*, którzy wzięli udział w implementacji przykładowego układu sterowania: Konradowi Banachowiczowi, Piotrowi Miedzikowi, Maciejowi Stafańczykowi oraz Kacprowi Szkudlarkowi.

Bibliografia

1. Borenstein, J., Koren, Y. (1991): *The vector field histogram-fast obstacle avoidance for mobilerobots*. IEEE Transactions on Robotics and Automation 7(3), 278–288.
2. Ulrich, I., Borenstein, J. (1998): *VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots*. In: IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 16–21, pp. 1572–1577.
3. Zieliński, C. (2006): *Transition-Function Based Approach to Structuring Robot Control Software*. In: Kozłowski, K. (Ed.), Robot Motion and Control: Recent Developments, Lecture Notes in Control and Information Sciences, Vol.335, pp. 265–286, Springer Verlag.

4. Zieliński, C. (2010): *Formalne podejście do programowania robotów – struktura układu sterującego*. In: Ambroszkiewicz, S., Borkowski, A., Centarowicz, K., Zieliński, C. (Eds.), Inteligencja wokół nas. Współdziałanie agentów softwareowych, robotów, inteligentnych urządzeń, pp. 267–300, EXIT.
5. Zieliński, C., Szynkiewicz, W., Winiarski, T. (2005): *Applications of MRROC++ Robot Programming Framework*. In: Kozłowski, K. (Ed.), Proceedings of the 5th International Workshop on Robot Motion and Control, RoMoCo'05, Dymaczewo, Poland, pp. 251–257.
6. Zieliński, C., Winiarski, T. (2010): *Motion Generation in the MRROC++ Robot Programming Framework*. International Journal of Robotics Research 29(4), 386–413.

Method of Designing Autonomous Mobile Robot Control Systems. Part 2: An Example

Abstract: This two-part paper presents a general method of designing both the structure as well as the method of operation of autonomous robot control systems. The first part of the paper describes the notation and the general aspects of the design method. The second part presents an example of the utilization of the proposed method of designing robot controllers.

Keywords: robot control systems, autonomous robots

prof. nzw. dr hab. inż. Cezary Zieliński

Od 2008 roku pracuje w Przemysłowym Instytucie Automatyki i Pomiarów. Ponadto jest profesorem nadzwyczajnym Politechniki Warszawskiej na Wydziale Elektroniki i Technik Informacyjnych. W latach 2002-2005 sprawował na tym wydziale funkcję prodziekana ds. nauki i współpracy międzynarodowej, 2005-2008 zastępcy dyrektora Instytutu Automatyki i Informatyki Stosowanej (IAiIS) ds. naukowych, a od 2008 pełni funkcję dyrektora tego instytutu. Od uzyskania habilitacji w roku 1996 pełni rolę kierownika Zespołu Robotyki w IAiIS. Od 2007 roku jest członkiem i sekretarzem Komitetu Automatyki i Robotyki Polskiej Akademii Nauk. Jego zainteresowania badawcze koncentrują się na zagadnieniach związanych z programowaniem i sterowaniem robotów.

e-mail: c.zielinski@ia.pw.edu.pl

mgr inż. Tomasz Kornuta

Absolwent Wydziału Elektroniki i Technik Informacyjnych Politechniki Warszawskiej. W 2003 roku uzyskał tytuł inżyniera, w 2005 tytuł magistra inżyniera. Od 2008 roku pracuje na etacie asystenta w Instytucie Automatyki i Informatyki Stosowanej (IAiIS), w ramach którego prowadzi zajęcia dydaktyczne, a od 2009 roku pełni funkcję Kierownika Laboratorium Podstaw Robotyki. Dodatkowo od 2009 roku jest zatrudniony na etacie konstruktora, w ramach którego realizuje prace związane z grantem europejskim. Od 2005 roku w ramach doktoratu prowadzi badania związane z wykorzystaniem przez roboty paradygmatu aktywnego czucia do analizy otoczenia. Jego główne zainteresowania naukowe obejmują wykorzystanie informacji wizyjnej w robotyce.

e-mail: tkornuta@ia.pw.edu.pl

mgr inż. Piotr Trojanek

Jest doktorantem oraz pracuje jako konstruktor w Instytucie Automatyki i Informatyki Stosowanej Politechniki Warszawskiej. Swoje doświadczenie zdobywał pracując nad układami sterowania robotów mobilnych. Obecnie prowadzi badania dotyczące systemów wieloagentowych oraz zastosowań metod inżynierii oprogramowania w robotyce. Od lat związany ze studenckim kołem naukowym Bionik.

e-mail: piotr.trojanek@gmail.com

dr inż. Tomasz Winiarski

Jest adiunktem w Instytucie Automatyki i Informatyki Stosowanej Politechniki Warszawskiej. Sprawuje funkcje kierownika laboratorium robotyki w macierzystym instytucie, a także opiekuna studenckiego koła naukowego Bionik, które współtworzył. W 2010 roku otrzymał za osiągnięcia naukowe nagrodę indywidualną drugiego stopnia rektora PW. Jego zainteresowania badawcze dotyczą z jednej strony konstrukcji i nawigacji robotów mobilnych z drugiej strony specyfikacji zadań manipulatorów i chwytaków ze szczególnym uwzględnieniem sterowania pozycyjno-siłowego.

e-mail: tmwiniarski@gmail.com