Józef OKULEWICZ*

MODELOWANIE STEROWANIA POJAZDAMI TRANSPORTU INDUKCYJNEGO

Streszczenie

W przypadku transportu indukcyjnego rozwój techniczny umożliwia coraz łatwiejsze modyfikowanie rozmieszczenia tras wózków w celu dopasowywanie ich do aktualnych potrzeb transportowych. Poważniejsza trudność jest związana z koniecznością dopasowania sterowania do zmienionej trasy i natężenia ruchu.

W pracy przedstawiono możliwość tworzenia języka programowania na potrzeby sterowania konkretnym urządzeniem. W szczególności dla potrzeb sterowania układem transportu indukcyjnego zmodyfikowano język programowania utworzony dla potrzeb sterowania układami transportu przenośnikowego. Język programowania zastosowano w symulatorze układów transportu przenośnikowego. Program sterujący w modelu może być bezpośrednio wykorzystany w realnym układzie transportowym. Takie podejście umożliwia użytkownikowi dopasowanie we własnym zakresie programu sterowania układem transportowym do aktualnych potrzeb.

Słowa kluczowe: transport indukcyjny, system, sterowanie, symulacja

1. WSTĘP

Na obszarze po którym planuje się przemieszczanie ładunków przy użyciu wózków indukcyjnych rozmieszcza się tzw. pętle indukcyjne, wzdłuż których poruszają się wózki [3,5]. Poszczególne trasy różnią się częstotliwością, która jest rozpoznawana przez układ napędowy wózka. Po uruchomieniu wózka na zadaną częstotliwość pracy linii i włączenia zasilania linii, wózek przemieszcza się wzdłuż przewodu tworzącego pętlę indukcyjną. Na trasie są przygotowane miejsca, w jakich wózek może się zatrzymywać. W takich miejscach ładunki są nakładane na wózek lub są z niego zdejmowane. Ruch wózka może być inicjowany przez operatora lub odbywać się w odpowiednim rytmie czasowym umożliwiającym za i rozładunek. Bardzo atrakcyjne wydają się wózki sterowane automatycznie [4].

Projektowanie układu transportowego polega nie tylko na określeniu tras, po których poruszają się wózki. Trzeba także wyznaczyć liczbę wózków umożliwiających zrealizowanie napływających zadań transportowych. Każde zadanie określa miejsce pobrania ładunku i miejsce jego odłożenia. Na tej podstawie można dla danego strumienia ładunków określić pracochłonność zadania transportowego. Polega to na określeniu czasu potrzebnego dla przemieszczenia ładunków. Po uwzględnieniu drogi powrotnej, jaką wózki muszą przebyć aby ponownie pobrać ładunek, wyznacza się zapotrzebowanie na pracę przewozową. Trzeba przy tym uwzględnić to, czy wózek może pobrać jedną lub więcej jednostek ładunkowych oraz to, czy są to jednostki na tej samej trasie czy na różnych trasach. Uwzględnienie tych czynników powoduje, że obliczenia stają się skomplikowane, a sterowanie wymaga rozpoznawania ładunków nakładanych na wózek. Do tego dochodzi jeszcze konieczność uwzględnienia intensywności zgłaszania ładunków w poszczególnych miejscach załadunkowych i struktury rozmieszczenia miejsc docelowych dla ładunków. Powoduje to, że liczba wózków może być wyznaczona z dużym przybliżeniem i musi być zweryfikowana po zrealizowaniu układu transportowego.

* Politechnika Warszawska, Wydział Transportu

Ze względu na złożoność projektowania zarówno układu przestrzennego transportu jak i jego sterowania, tworzenie układu transportu indukcyjnego ma sens dla stabilnych strumieni zgłoszeń ładunków. Przeważnie układ transportu indukcyjnego tworzy się jednorazowo, dla zadanego rozmieszczenia miejsc załadunku i wyładunku. Zazwyczaj też nie ulega on zmianom w okresie swego funkcjonowania. Jakiekolwiek zmiany w położeniu tras i punktów przeładunkowych są bowiem utrudnione. Może to być niedogodnością w przypadku zmian w zakresie strumieni ładunków przetwarzanych przez układ transportowy.

Jednak postępy w rozwoju techniki komputerowej mogą ułatwiać tworzenie elastycznych układów transportu indukcyjnego. Z tego względu może być potrzebna możliwość łatwej zmiany oprogramowania sterującego tym rodzajem transportu. Jednak każdorazowa zmiana wymaga ingerencji firmy specjalistycznej i nie może być zrealizowana przez użytkownika we własnym zakresie. Z jednej strony konieczne może być zmodyfikowanie rozmieszenia tras wózków. Z drugiej – nie mniej ważnej– może być niezbędne dopasowanie sterowania do zmienionej trasy i natężenia ruchu. Mogłoby to być realizowane przez użytkownika pod warunkiem uzyskania dostępu do oprogramowania sterującego. Nie ma jednak takiego zwyczaju przy obecnej małej częstości takich zmian. Poza tym użytkownik nie jest zainteresowany w opanowaniu złożonego specjalizowanego oprogramowania sterującego. Podejmowanie takich działań przez użytkownika miałoby sens gdyby sterowanie układami tego typu zostało uproszczone.

Jak dotąd upraszczanie takie dotyczy tworzenie programu sterującego i polega na umieszczaniu na ekranie komputera odpowiednich symboli reprezentujących określone obiekty i procedury. Faktycznie, uzyskuje się znaczne uproszczenie w tworzeniu programu sterującego, gdyż jest on generowany przez komputer na podstawie wprowadzonych symboli. Dodatkowo na podstawie wprowadzonych danych można utworzyć model układu transportowego wraz z animacją jego funkcjonowania. Model symulacyjny można także wykorzystać do dobierania metodą kolejnych prób potrzebnej liczby wózków tak, by wszystkie ładunki zostały rozwiezione do miejsc przeznaczenia.

Wadą takiego podejścia jest stosunkowo wysoki koszt oprogramowania wspomagającego projektowanie, co czyni go nieopłacalnym dla użytkownika, który rzadko dokonywałby zmian sterowania. Aczkolwiek koszt ten nie musi być znaczący w porównaniu do kosztu realizacji układu transportowego i kosztu jego niewłaściwego funkcjonowania.

Innym sposobem takiego uproszczenia programowania jest proponowana metoda tworzenia języka programowania do sterowania określonym rodzajem obiektów. Zakłada się przy tym, że zakres instrukcji sterujących programu ogranicza się do potrzeb konkretnej klasy zadań. Dzięki temu zarówno sprawdzające modelowanie symulacyjne jak i samo sterowanie działaniem układu transportowego może być realizowane przez użytkownika we własnym zakresie, za pomocą tego samego języka programowania. Do zrealizowania tej idei potrzeba tylko aby język programowania był odpowiednio prosty. Musi on powstać poprzez zasadniczą zmianę sposobu programowania komputerów.

2. BUDOWA INSTRUKCJI PROGRAMU

Dotychczas stosowane sposoby komputerowego wspomagania sterowania urządzeniami polegają albo na wykorzystaniu istniejącego języka programowania wysokiego poziomu i uzupełnienie go o odpowiednie procedury umożliwiające programowanie konkretnego urządzenia, albo na utworzenie specjalizowanego języka programowania dopasowanego do możliwości sterowanego urządzenia. Jednak taki nowy język jest zwykle wzorowany na istniejących językach programowania. Co więcej, w trakcie ponad 60 lat rozwoju programowania większość powstałych języków programowania jest wzorowana na języku FORTRAN [9,10,11]. Można więc przyjąć, że ten rozwój został zdominowany przez jeden

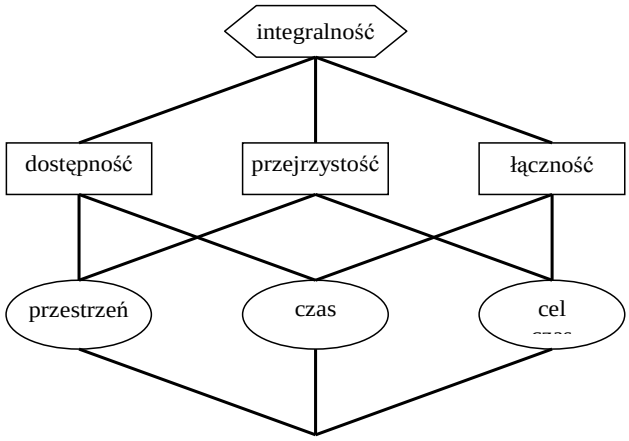
sposób myślenia o posługiwaniu się komputerem i o jego programowaniu [1]. Skutkiem tego nastąpiło swego rodzaju zamrożenie zarówno rozwoju struktury komputerów jak i metod programowania [2].

Na sposób wykonywania programu niekorzystnie wpłynęło traktowanie komputera jako maszyny matematycznej. Od czasu umieszczenia instrukcji programu razem z danymi w tej samej pamięci komputera [8], program rezyduje na stałe w pamięci i każda instrukcja jest przypisana do określonego miejsca w przestrzeni. Jednak wynikające z tego oczywiste ograniczenia komputera są traktowane jako jego ułomności i przesłaniane w językach wysokiego poziomu.

Alternatywnym rozwiązaniem problemu mogłoby być tworzenie języka programowania dostosowanego do potrzeb sterowanego urządzenia, bez balastu innych możliwych jego zastosowań. Zmiana polega na tym, że komputer nie powinien być traktowany jako maszyna matematyczna lecz po prostu, jako urządzenie sterowane za pomocą odpowiednich komend. Przy takiej zmianie podejścia język programowania ogranicza się do obiektów odpowiadających obiektom systemu technicznego, a sfera operacji matematycznych jest ukryta przed użytkownikiem.

Celem programowania jest spowodowanie działania urządzenia w pożądanych sekwencjach. Za pomocą instrukcji odpowiedniego języka programowania można sterować funkcjonowaniem takiego urządzenia. Może to być nie tylko komputer, ale także inne urządzenia do niego podłączone.

Ponieważ komendy języka programowania powinny umożliwić realizowanie celu działania systemu, dlatego przy formułowaniu innego podejścia do programowania można skorzystać z ogólnych własności systemów.



Rys.1. Struktura pojęć systemowych
Źródło: opracowanie własne

W stosunku do systemu jakim jest sterowane programowo urządzenie, odnoszą się trzy niezależne logiczne kategorie przestrzeni, czasu i celu działania. Dotyczą one zarówno obiektów, na jakich są wykonywane operacje obliczeniowe jak i poleceń języka programowania, gdyż to z jego pomocą realizowane są cele użytkowania tego urządzenia. Na ich bazie można utworzyć kratę logiczną wiążącą pojęcia dotyczące systemu (rys.1). Stanowi ona źródło postulatów, jakie system powinien spełniać w celu zachowania swej integralności [6]. W proponowanej metodzie tworzenia języka programowania przyjmuje się, że instrukcje języka programowania są traktowane jako oddzielne systemy. Zatem każda z nich musi zapewnić „*dostępność*”, „*przejrzystość*” i „*łączność*” względem elementów, na które oddziałuje. Odpowiednio do tego instrukcja sterująca oprócz komendy zawiera trzy sekcje,

które nazwano odpowiednio: WOBEC, JAKO i OTO. Dla zrealizowania postulatu dostępności w sekcji WOBEC podaje się zakres obiektów, wobec których ma być podjęte działanie, określając w ten sposób „*gdzie*" i „*kiedy*" działanie ma być wykonane. Kolejna część instrukcji tworzy sekcję o nazwie JAKO i określa „*gdzie*" i „*co*" ma być wykonane. Ponadto określa się „*kiedy*" i „*co*" ma być wykonane, za co odpowiada trzecia część instrukcji tworząca sekcję o nazwie OTO. Postulaty te względem obiektów matematycznych są spełnione z definicji. W maszynie matematycznej są one realizowane poprzez umieszenie wszystkich obiektów w pamięci i wykonywanie operacji w jednym module przetwarzającym. Dzięki temu jest ułatwiona realizacji postulatów integralności w języku programowania. Znajdują się w nich odpowiedniki wymienionych trzech grup instrukcji. Do pierwszej grupy należą instrukcje typu „*for*", do drugiej – instrukcje typu „*if*", „*case*", „*while*", a do trzeciej grupy – instrukcje typu „*go*" oraz wywołania funkcji i procedur.

3. PROGRAMOWANIE UKŁADU TRANSPORTU INDUKCYJNEGO

Rozpatruje się sterowanie w układzie transportu indukcyjnego, w którym jest kilka tras poruszania się wózków. Każdy wózek porusza się po przydzielonej mu trasie. Trasy ułożone są wokół wielu punktów nadania i odbioru ładunków. W danym źródle ładunków pojawiają się ładunki do różnych miejsc docelowych. Przy tym miejsca docelowe ładunków z danego źródła mogą należeć do różnych tras. Wózek może pobierać jeden ładunek. Ładunek jest pobierany w danej stacji jeśli jego miejsce docelowe należy do trasy, po jakiej porusza się dany wózek.

Do obsługi sterowania wózkami zaproponowano następujące komendy: *start*, *stop*, *pobierz*, *zostaw*. Dla tych komend można opracować specjalizowany język programowani wg przedstawionych zasad. Jednak można też wykorzystać podobieństwa pomiędzy transportem indukcyjnym a przenośnikowym. Np. obiekty poruszają się po wyznaczonych trasach, nie mogą się wymijać, stojący obiekt zatrzymuje obiekty poruszające się za nim, itp. Dlatego do sterowania wykorzystano język utworzony do sterowania układami transportu przenośnikowego [7], uzupełniony o dwie komendy *pobierz (p)*, *zostaw (z)* (rys.2).

Ze względu na prosty zestaw komend sterujących, poszczególne sekcje instrukcji są rozpoznawane na podstawie początkowego znaku. Nie wykorzystano też wszystkich możliwości znaczeniowych zaproponowanej składni języka.

Przy tworzeniu języka programowania przyjęto, że instrukcje sterujące są przypisane do odcinków tras wózków. Są one aktywowane przez poruszające się po tych trasach obiekty i wykonywane w kolejności zapisania. Określają one, na jakim odcinku trasy instrukcja jest wykonywana, jaki powinien być spełniony warunek oraz chwilę wykonania instrukcji.

Trasy poruszania się wózków są tworzone z modułów w postaci odcinków prostych. Są one podzielone na moduły, które umożliwiają poruszanie się obiektu pełniącego funkcję wózka. Każdy wózek ma cechę określającą trasę, do jakiej jest przydzielony. Kierunek poruszania się wózka po danym module ustala się na podstawie tej cechy obiektu. Dzięki niej wózek może także zmieniać kierunek działania modułu, po jakim się porusza. Ponadto wybrane moduły służą do zatrzymywania obiektów. W takich momentach ładunki są pobierane lub zdejmowane z obiektu.

Ze względu na pierwotne przeznaczenie języka do sterowania układami transportu przenośnikowego, za jego pomocą trzeba oddziaływać na moduły aby zmieniać kierunek ich działania, a tym samym kierunek ruchu obiektów. W rzeczywistym układzie transportowym wózek porusza się wzdłuż przewodu pętli indukcyjnej do jakiej został przydzielony i dlatego instrukcje zmiany kierunku działania modułu nie byłyby wykonywane.

Domyślnie instrukcje są przydzielane do modułów wg zawartości sekcji WOBEC. Jednak można przypisać instrukcję do dowolnego modułu za pomocą poprzedzającego tę instrukcję adresu w postaci nazwy modułu poprzedzonej znakiem „:”.

Instrukcje z pustymi sekcjami JAKO i OTO są wykonywane jednorazowo w chwili początkowej po uruchomieniu programu. Instrukcje obiektowe, których sekcja WOBEC rozpoczyna się od „#” są aktywowane przy każdym uruchomieniu modułu, do jakiego są przypisane. Instrukcje modułowe, których sekcja WOBEC rozpoczyna się od „@” o niepustych sekcjach JAKO i OTO są aktywowane przez obiekty docierające do modułów. Obiekty ruchome są identyfikowane wg cechy przypisywanej im w chwili ich powstawania.

```
<program> ::= <linia>{<linia>}
<linia> ::= <instrukcja> | * text
<instrukcja> ::= <WOBEC> <komenda>{/<nazwa>} <JAKO> <OTO>
<WOBEC> ::= @<nazwa>{<relacja><nazwa>} | #<nazwa>
<JAKO> ::= <puste> | #<nazwa> | #<rozrzut>
<OTO> ::= <puste> | <znak><wartość>
<nazwa> ::= * | tekst
<komenda> ::= St | Sp | we | le | pr | go | do | v* | q* | p* | z* |
<relacja> ::= : | < | >
<znak> ::= < puste > | + | @
<wartość> ::= liczba | <rozkład> | <rozrzut>
<rozkład> ::= <typ>(liczba){, liczba })
<typ> ::= W | N | R | E
<rozrzut> ::= liczba /<część>{; liczba /<część>}{; liczba }
<część> ::= prawdopodobieństwo
```

Rys.2. Zapis syntaktyki proponowanego języka programowania.
Źródło: opracowanie własne

W modelu obiekty pełniące funkcję ładunków są przemieszczane po modułach za pomocą obiektów pełniących rolę wózków. Dlatego też nie mają one przydzielonej trasy i po pojawieniu się na danym module „spadają” z niego i pozostają na stosie obok niego. Z tego miejsca mogą być pobierane na wózki.

Ładunki pojawiają się w źródłach w ustalonych odstępach czasu. Dla uproszczenia przyjęto, że odcinki czasu są stałe. Jednak w ogólności mogą to być losowe odcinki czasu, lub zadane dowolnym harmonogramem.

Jeśli w danym źródle może pojawić się kilka rodzajów ładunków, to jest określone prawdopodobieństwo pojawienia się ładunku danego rodzaju.

Wózek z określonej trasy może pobrać ładunek na stacji postojowej jeśli cecha ładunku określa miejsce docelowe należące do trasy wózka. Jeśli jest ładunek do pobrania i przewiezienia na trasie wózka, to wózek się zatrzymuje. Jeśli nie ma ładunku, to wózek mija stację.

Wózek zatrzymuje się jeśli ma ładunek do danego punku. Następnie startuje po zadanej chwili. Jeśli nie ma ładunku do tego punktu to wózek nie zatrzymuje się. Bezwarunkowa komenda startu spowoduje rozpoczęcie ruchu, tylko wtedy gdy było zatrzymanie wózka. Jeśli wózek nie zatrzymał się, to ta komenda niczego nie spowoduje.

Relacja w sekcji WOBEC umożliwia rozdzielenie zakresu działania instrukcji jako całości i samej komendy. Instrukcja jest wykonywana gdy obiekt dociera do modułu. Są przy tym dwie fazy. Najpierw działa instrukcja, a dopiero jak zostanie uznana za instrukcję aktywną, następuje wykonanie komendy.

Działanie instrukcji polega na wykonaniu sekcji JAKO, której istotną funkcją jest sprawdzenie warunków wykonania komendy. Jeśli warunki są spełnione, to na podstawie relacji w sekcji WOBEC określa się jakiego modułu dotyczy komenda.

4. PRZYKŁAD MODELOWANIA STEROWANIA

Zadane jest rozmieszczenie stacji nadania i odbioru ładunków. W modelu umieszczono układ modułów tworzących trasy, z zachowaniem proporcji do układu rzeczywistego. Przy wykorzystaniu stosunku odległości rzeczywistej do długości trasy w modelu można określić prędkość poruszania się wózków w [cm/s]. Wynosi ona w tym przypadku v = 78 [cm/s] ≈ 46,7 [m/min].

Średnie zapotrzebowanie na przewozy ładunków pomiędzy poszczególnymi stacjami wynika z harmonogramu produkcji i jest zadane w tab. 1.

Tabela 1. Zapotrzebowanie na przewozy ładunków pomiędzy stacjami

	0	1	2	3	4	5	6	7	suma
0		212		296		184	134		826
1			212						212
2				15	210				225
3					189				189
4	488								488
5	308				124				432
6	53							64	117
7	38				34				72
suma	887	212	212	311	557	184	134	64	2561

Źródło: opracowanie własne

Przyjęto założenie, że zapotrzebowanie na przewozy w każdej ze stacji powstaje w stałych odstępach i jest rozłożone równomiernie w czasie. Na tej podstawie dla każdego punktu nadania wyznaczono odstęp czasu pomiędzy kolejnymi ładunkami:

$$dt_i = \frac{T_D}{L_i}, \tag{1}$$

gdzie:
dt_i – odstęp pomiędzy ładunkami wysyłanych z i-tej stacji,
T_D – dysponowany czas pracy (16 godz.)
L_i – liczba ładunków wysyłanych z i-tej stacji.

Następnie na podstawie tab.1 dla każdej stacji nadawczej wyznaczono prawdopodobieństwo wystąpienia ładunku skierowanego do odpowiednich stacji odbiorczych (2):

$$p_{ij} = \frac{L_{ij}}{L_i}, \tag{2}$$

gdzie:
p_{ij} – prawdopodobieństwo wystąpienia ładunku ze stacji i-tej do j-tej,
L_{ij} – liczba ładunków wysyłanych ze stacji i-tej do j-tej,

Na podstawie rozmieszczenia stacji nadawczo odbiorczych wyznaczono odległości pomiędzy poszczególnymi stacjami nadania i odbioru (tab.2).

Przy realizacji przewozów należałoby przyjąć, że wózki poruszają się w jednym kierunku i po wyładowaniu ładunku mogą pobrać ładunek ze stacji znajdującej się na jego trasie. Takie założenie znacząco skomplikowałoby obliczenia. Dlatego przyjęto czas powrotu

równy czasowi dowiezienia. Zapotrzebowanie na pracę przewozową oszacowano wg przybliżonego wzoru (3) i zestawiono w tab. 3.

Tabela 2. Odległości pomiędzy poszczególnymi stacjami [cm]

	0	1	2	3	4	5	6	7
0	0	1586	3509	4504	6791	3273	4657	5965
1	7344	0	1923	2917	5205	1687	3071	4379
2	5422	7008	0	995	3283	0	0	0
3	4427	6013	7935	0	2288	7700	9084	10391
4	2139	3725	5648	6642	0	5412	6796	8104
5	8541	10127	0	4114	6402	0	1384	2692
6	7157	8743	0	2730	5018	10430	0	1307
7	5849	7436	0	1423	3711	9123	10507	0

Źródło: opracowanie własne

$$P = L_{ij} \cdot (2 \cdot T_{ij} + 2 \cdot T_0) \tag{3}$$

gdzie:
P – zapotrzebowanie na pracę przewozową,
L_{ij} – liczba ładunków wysyłanych ze stacji i-tej do j-tej,
T_{ij} – czas przejazdu trasy pomiędzy stacją i-tą a j-tą,
T_0 – czas postoju w stacji w celu załadunku lub wyładunku.

Tabela 3. Zapotrzebowanie na pracę przewozową pomiędzy poszczególnymi stacjami

	0	1	2	3	4	5	6	7	
0	0	27727	0	60919	0	32047	28108	0	
1	0	0	29561	0	0	0	0	0	
2	0	0	0	1734	36627	0	0	0	
3	0	0	0	0	28129	0	0	0	
4	70760	0	0	0	0	0	0	0	
5	95365	0	0	0	31574	0	0	0	
6	14524	0	0	0	0	0	0	7912	
7	9136	0	0	0	6304	0	0	0	SUMA
	189783	27727	29561	62653	102634	32047	28108	7912	**480423**

Źródło: opracowanie własne

Liczbę wózków niezbędną do zrealizowania przewozów ładunków oszacowano na podstawie sumarycznego zapotrzebowania na pracę przewozową i dysponowanego czasu pracy (4). Wynosi ona w tym przykładzie w = 8,34.

$$w = \frac{P}{T_D} = 8{,}34\,, \tag{4}$$

gdzie:
w – liczba wózków niezbędna do zrealizowania przewozów ładunków.

Generowanie ładunku do przewiezienia jest realizowane za pomocą następującej instrukcji: `#p0 we #1/0.257;3/0.358;5/0.223;6/0.162 +69.73.`

Powoduje ona w module „p0” – odpowiadającym stacji „0” – powstawanie ładunku w stałych odstępach czasu dt_0 = 69,73. Wartość ta wynika ze średniej liczby ładunków, jaka ma się pojawić w tym miejscu. Dla ładunku powstałego w sekcji „p0” określa się losowo jeden z czterech adresów („1”, „3”, „5”, „6”), wg prawdopodobieństwa przypisanego do każdego z nich. Wartości prawdopodobieństwa określono na podstawie struktury ładunków

przemieszczanych ze stacji o adresie „0” (tab.1). Podobnie tworzy się instrukcje generowania ładunków dla pozostałych stacji („1”, „2”, „3”, „5”, „6”, „7”). Jeśli miejsce docelowe ładunku znajduje się przy wspólnym odcinku obu tras, to ładunki mogą być pobierane przez wózki należące do obu tras.

Pobieranie ładunków jest realizowane za pomocą sekwencji instrukcji przypisanych do miejsca załadunku. Przykładowo oprogramowanie stacji o numerze „0” przedstawiono w tab. 5. W poniższym przykładzie są to instrukcje przypisane do modułu „z0”, odpowiadającego stacji o adresie „0”. Ładunki przypisane do tego modułu są gromadzone przy module „p0”.

Gdy obiekt reprezentujący wózek wjeżdża na moduł „z0”, to aktywuje przypisane do niego instrukcje. Pięć początkowych instrukcji określa warunki wykonania komendy „stop” („Sp”). Pojedyncza instrukcja wymaga sprawdzenia, czy na stosie w miejscu „p0” jest ładunek do stacji znajdującej się na trasie tego wózka (np. „@1”), czy wózek ma cechę wynikająca z przypisania go do trasy, na której znajduje się stacja docelowa (np. „#t1”) oraz czy na wózku są wolne miejsca („?0”). Jeśli wszystkie te warunki są spełnione, to wózek jest zatrzymywany. Liczba warunków zatrzymania wózka odpowiada liczbie instrukcji.

Uruchomienie wózka nastąpi po 45 sek. na skutek wykonania instrukcji „@z0 St +45”. Ładunek spełniający warunki zatrzymania wózka zostanie pobrany za pomocą którejś z końcowych pięciu instrukcji. Jeśli wózek nie zatrzyma się to znaczy, że nie jest spełniony któryś z warunków pobrania ładunku i żadna z tych instrukcji nie spowoduje pobrania ładunku, mimo ich aktywowania przez wózek.

Tabela 4. Instrukcje sterujące przypisane do stacji o adresie „0”

instrukcja	objaśnienie instrukcji
`@z0<p0 Sp @1#t1?0 +0`	na sekcji z0 zatrzymanie obiektu ruchomego jeśli ma on cechę „t1” (jest wózkiem z trasy t1) i przy sekcji p0 znajdują się obiekty z cechą „1” (ładunki do stacji 1) i wiezie on 0 obiektów (ma wolne miejsce na ładunek)
`@z0<p0 Sp @1#t2?0 +0`	podobnie dla obiektu ruchomego z cechą „t2”
`@z0<p0 Sp @3#t2?0 +0`	podobnie dla obiektu ruchomego z cechą „t2” i obiektów na stosie z cechą „3”
`@z0<p0 Sp @5#t1?0 +0`	podobnie dla obiektu ruchomego z cechą „t1” i obiektów na stosie z cechą „5”
`@z0<p0 Sp @6#t1?0 +0`	podobnie dla obiektu ruchomego z cechą „t1” i obiektów na stosie z cechą „6”
`@z0 St +45`	uruchomienie sekcji z0 po czasie 45 sek.
`@z0:p0 p1/1 #t1 +0`	na sekcji z0 pobranie ze stosu przy sekcji p0 ładunku z cechą „1” jeśli ruchomy obiekt ma cechę „t1”
`@z0:p0 p1/1 #t2 +0`	podobnie dla ładunku z cechą „1” jeśli ruchomy obiekt ma cechę „t2”
`@z0:p0 p1/3 #t2 +0`	podobnie dla ładunku z cechą „3” jeśli ruchomy obiekt ma cechę „t2”
`@z0:p0 p1/5 #t1 +0`	podobnie dla ładunku z cechą „5” jeśli ruchomy obiekt ma cechę „t1”
`@z0:p0 p1/6 #t1 +0`	podobnie dla ładunku z cechą „6” jeśli ruchomy obiekt ma cechę „t2”

Źródło: opracowanie własne

5. WYNIKI MODELOWANIA

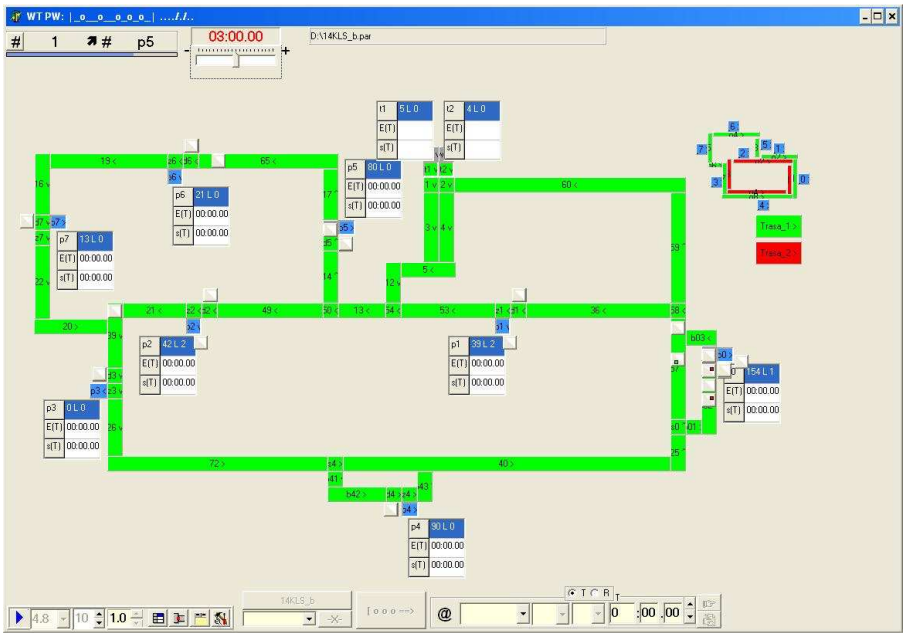
Dla określenia liczby potrzebnych wózków wykorzystano modelowanie symulacyjne układu transportowego dla konkretnego strumienia napływających ładunków. W modelu pominięto zagadnienie rozdzielenia zbioru wózków na poszczególne trasy.

Początkowo w modelu założono brak bocznic przy stacjach postojowych. Jednak w wyniku modelowania okazało się, że powstają kolejki wózków przy stacjach, przy których obie trasy wózków się pokrywają. Ponieważ spowalniało to proces rozwożenia ładunków, w kolejnej wersji tras zaproponowano w dwóch miejscach bocznice, aby zatrzymujące się wózki nie tamowały ruchu tych wózków, które mijają te stacje.

W wyniku modelowania układu transportu indukcyjnego stwierdzono, że przy zastosowaniu 8 wózków ładunki są sukcesywnie pobierane ze stacji początkowych i rozwożone do stacji docelowych. Jedynie w stacji „5” pozostawało znacząco więcej

ładunków niż w innych stacjach. Dlatego niezbędne było dodanie jednego wózka poruszającego się na trasie 1 (rys.3). W wyniku tego opóźnienie z pobieraniem ładunków ze stacji „5” zostało zmniejszone. Oznacza to, że powiększanie liczby wózków powoduje zmniejszanie opóźnienia pobierania ładunków z poszczególnych stacji. Liczba ładunków oczekujących na przewiezienie może być kryterium wyboru liczby wózków (tab.5).

Dalsze badanie tego układu transportowego może mieć na celu sprawdzenie efektu wprowadzenia wózków jednego rodzaju, które ustalałyby swoją trasę na podstawie zapotrzebowania i wiezionego ładunku. Wydaje się bowiem, że po pobraniu ładunku ze stacji „5” do stacji „0” lub „4”, wózek przypisany do trasy 1 niepotrzebnie jedzie obok stacji „6” i „7”. Podobnie wózek przypisany do trasy 1 niepotrzebnie jedzie obok stacji „6” i „7” wioząc ładunek ze stacji „0” do stacji „3”. Odpowiednia modyfikacja sterowania jest możliwa przy użyciu proponowanego języka sterowania.



Rys.3. Przykład ekranu programu symulującego działania 9 wózków
Źródło: opracowanie własne

Tabela 5. Liczba ładunków nie przewiezionych po 3 godz.

stacja	4 + 4 wózki brak bocznic	4 + 4 wózki 2 bocznice	5 + 4 wózki 2 bocznice	5 + 5 wózków 2 bocznice
0	4	5	3	1
1	10	2	3	0
2	13	2	4	0
3	1	1	0	1
4	1	0	0	0
5	18	5	1	1
6	2	3	0	0
7	4	5	0	0
suma	**53**	**22**	**11**	**3**

Źródło: opracowanie własne

6. WNIOSKI

Język programowania utworzony na podstawie proponowanej metody umożliwia sterowanie działaniem przykładowego układu transportu indukcyjnego. Na podstawie wyników modelowania symulacyjnego można wyznaczyć niezbędną liczbę wózków dla zrealizowania zadanego strumienia ładunków.

Współcześnie istnieją już środki techniczne dla zrealizowania przedstawionej koncepcji sterowania. Jednak istotnym utrudnieniem może być konieczność przekonania użytkownika, że powinien mieć możliwość oddziaływania na układ transportowy we własnym zakresie. Dotychczas jest to domeną specjalizowanych firm, które nie stwierdzają takiego zapotrzebowania ze strony potencjalnych użytkowników.

Zanim to nastąpi przedstawiony symulator może mieć zastosowanie w dydaktyce. Umożliwia bowiem obserwowanie zjawisk występujących przy posługiwaniu się transportem indukcyjnym, a przede wszystkim umożliwia aktywne kształtowanie tego transportu za pomocą prostego języka programowania.

LITERATURA

[1] Backus J.: The history of FORTRAN I, II, and III, ACM SIGPLAN Notices, VoI. 13, No. 8, August 1978, s. 165-180. "History of Programming Languages", Wexelblat, 1981, s. 25-45.
[2] Gabriel R. P., Steele G. L. Jr. : *What Computers Can't Do (And Why),* Lisp and Symbolic Computation (LASC), vol. 1, n. 3-4. 1986.
[3] Fijałkowski J.: Transport wewnętrzny w systemach logistycznych, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2003.
[4] Karkula M.: Wybrane aspekty modelowania i symulacji zautomatyzowanych systemów transportu wewnętrznego, XIII Konferencja Logistyki Stosowanej, Zakopane, 2009.
[5] Korzeń Z.: Logistyczne systemy transportu bliskiego i magazynowania. Tom 1.Infrastruktura, Technika, Informacja, Instytut Logistyki i Magazynowania, Poznań 1998.
[6] Okulewicz J.: *Kryteria analizy systemów transportowych*. Międzynarodowa Konferencja Naukowa "Transport XXI w." Warszawa 2004, s. 479-488.
[7] Okulewicz J.: Programming language creation for controlling internal transport devices, in "Advances in transport systems telematics", Silesian University of Technology, Katowice 2007, s. 97 – 105.
[8] Targowski A.: Informatyka – modele systemów i rozwoju. PWE Warszawa 1980.
[9] URL: http://hopl.murdoch.edu.au/, The History of Programming Languages by Diarmuid Pigott.
[10] URL: http://www.levenez.com/lang/, Jun 2007.
[11] URL: http://www.princeton.edu/~ferguson/adw/programming_languages.shtml, The History of Computer Programming Languages by Andrew Ferguson, (*Last Modified: 05-Nov-2004).*

THE MODELING OF AN INDUCTIVE TRANSPORT CONTROL

Abstract

As usual a particular inductive transport system doesn't change during its functional life because of stabile layout of its inputs and outputs. This is a reason of difficulties in preparing its abilities to actual streams of goods, by changing roads for moving cars, and modifying a controlling program to adapt cars movement to modified routs and goods sources.

A possibility of creating a computer programming language for controlling a particular transport system is presented in the paper. The modified language for programming a conveyer transport system was used. The language was used in a simulator of any conveyor transport systems. The advantage of such a solution is that the program can be directly used to control a real transport system. The user is able to adjust a controlling program to actual needs by his own.

Keywords: inductive transport, system, control, simulation