„Prawdziwy” czas w rozproszonych układach sterowania

Wojciech Grega

W artykule omówiono zjawiska, które mogą wpływać na dynamikę systemu sterowania - realizację zadań czasu rzeczywistego w sterowniku oraz dynamikę połączeń telekomunikacyjnych występujących w pętli sprzężenia zwrotnego. Przedstawiono narzędzia, które umożliwiają modelowanie i symulowanie tych zjawisk.

Charakterystyczną cechą współczesnych układów sterowania jest powszechne wykorzystanie komputerowych systemów przetwarzających sygnały, poczynając od pomiarów poprzez układy transmisji sygnałów do regulatora i od regulatora, po sterowniki urządzeń wykonawczych. Idea stabilizującego sprzężenia zwrotnego jest ciągle podstawową zasadą działania takich układów sterowania, ale systemy mikroprocesorowe stały się ich nieodłącznym komponentem przetwarzającym i analizującym w czasie rzeczywistym dane.

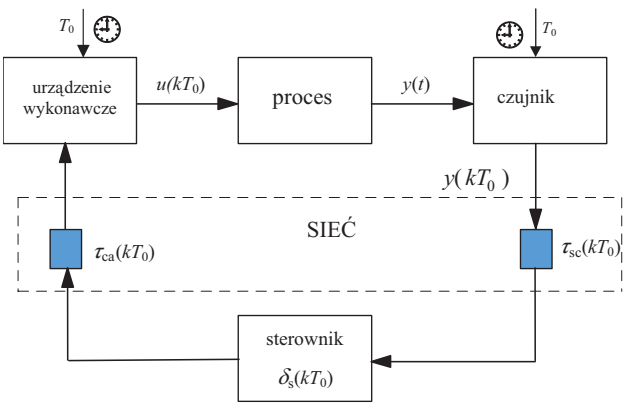
Masowy rozwój technologii systemów wbudowanych (*embedded*) wyposażonych w interfejsy komunikacyjne powoduje, iż współczesne układy sterowania i monitorowania rzadko występują jako samodzielne jednostki sterujące. Zazwyczaj są one połączone z innymi urządzeniami automatyki poprzez kanały teletransmisyjne, wymieniając dane i tworząc systemy sterowania rozproszonego [1].

Doskonałym przykładem jest elektronika pokładowa samochodu osobowego VW Phaeton, którą wspiera 61 systemów mikroprocesorowych (wbudowanych), z czego 35 jest połączonych poprzez trzy sieci CAN wymieniające w czasie rzeczywistym około 2500 sygnałów i komunikatów [2]. Dodatkowo, pokładowa sieć światłowodowa obsługuje systemy multimedialne samochodu.

Takie rozwiązania automatyki mogą wnosić do modelu dynamiki systemu sterowania pewne komplikacje, związane z charakterystycznym sposobem pracy tych systemów, polegającym na próbkowaniu stanu procesu w dyskretnych przedziałach czasu i na oddziaływaniu na proces również w określonych odstępach czasu, a zatem pracy przy ograniczeniach przepływu strumienia danych w pętli sprzężenia zwrotnego.

Obecność kanału transmisji danych w pętli sprzężenia zwrotnego może być nie tylko źródłem ograniczeń ilości przesyłanych danych, ale także może generować opóźnienia lub nawet spowodować utratę danych. Jest to szczególnie istotne w systemach sterowania krytycznych czasowo i krytycznych ze względu na bezpieczeństwo (*safety critical systems*) czyli takich, w których niewielkie naruszenia warunków przepływu danych w czasie rzeczywistym skutkują obniżeniem jakości sterowania, a nawet zagrożeniem bezpieczeństwa całego procesu. Na pewno są nimi elektroniczne układy sterujące w samochodzie. Formułowane są wymagania [2] dotyczące czasów reakcji dla komputerowych systemów sterowania w samochodzie osobowym – systemy awaryjne, stabilizacji toru jazdy i hamowania: 100 μs; napędy: 1ms; systemy poprawiające komfort jazdy: 10–100 ms. I wreszcie sam komputer sterujący, ze swoim charakterystycznym sposobem działania polegającym na realizacji poszczególnych zadań w odpowiedzi na przerwania wewnętrzne lub zewnętrzne, może istotnie wpływać na dynamikę systemu sterowania.

Aby to sobie uświadomić, wystarczy rozważyć następujący przykład (rys. 1). Jest to klasyczna struktura układu ze sprzężeniem zwrotnym, zawierająca model obiektu sterowania reprezentujący proces, dla którego są realizowane określone zadania sterowania. Czujnik jest aktywowany zegarem, który odczytuje wartości zmiennych procesu, przetwarza je na postać cyfrową i przesyła do regulatora, który odbiera aktualną wartość sterowania, wylicza sterowanie i przesyła do urządzenia wykonawczego (aktuatora). Ten przetwarza otrzymane dane cyfrowe na odpowiadające im wartości analogowe (prądy, napięcia itd.), aktualizuje ste-



Rys. 1. Model rozproszonego układu sterowania: τ_{sc} – opóźnienie pomiarów, τ_{ca} – opóźnienie sterowania, $\delta_s(kT_0)$ – opóźnienie przetwarzania danych

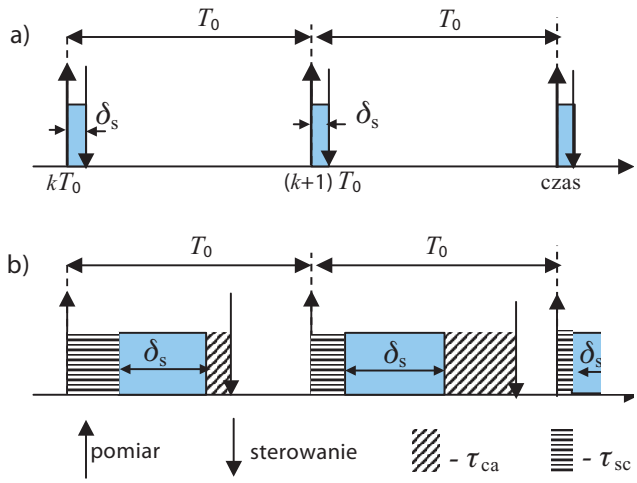
prof. dr hab. inż. Wojciech Grega
– Akademia Górniczo-Hutnicza w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, Katedra Automatyki

rowanie procesu i podtrzymuje je do czasu uzyskania kolejnego sterowania. Dane pomiarowe i sterujące są przesyłane jako oddzielne pakiety, stąd poszczególne komponenty takiego układu sterowania winny być modelowane poprzez równania z czasem ciągłym lub dyskretnym. Odpowiednie sygnały są oznaczone na rys. 1 jako funkcje „t" (czas ciągły) lub „k" (czas dyskretny).

Sieć transmisji danych może wprowadzać zmienne opóźnienia (τ_{ca}, τ_{sc} – rys. 1). Opóźnienia występujące w układzie (spowodowane zarówno transmisją danych jak i czasem obliczeń w sterowniku) mogą mieć charakter zdeterminowany lub przypadkowy. Charakter opóźnień transmisji sygnału zależy od rodzaju wykorzystywanej sieci przemysłowej, a w szczególności od stosowanej w tej sieci metodzie kontroli dostępu do medium komunikacyjnego [3]. W szczególności sieci typu token-bus, token-ring czy inne sieci o cyklicznej wymianie informacji (np. WorldFIP) są klasyfikowane jako sieci deterministyczne.

Zadaniem sterownika w rozproszonym układzie sterowania z rys. 1 jest:

- odczytanie pomiarów dostarczonych poprzez sieć
- wyliczenie sygnału sterującego
- wysłanie sygnału sterującego przez sieć do urządzenia wykonawczego (aktuatora).



Rys. 2. Dwa harmonogramy czasowe dla układu sterowania z rys. 1

Operacje te mogą być aktywowane zegarami, synchronizowanymi zegarami lub zdarzeniami. W przykładzie z rys. 1 działania czujnika i aktuatora są aktywowane zegarami o synchronizowanych częstotliwościach. Jest to równocześnie podstawowy cykl T_0 pętli sprzężenia zwrotnego. Działanie sterownika jest aktywowane zdarzeniem, którym w tym przypadku jest uzyskanie przez sterownik nowej wartości pomiaru $y(kT_0)$.

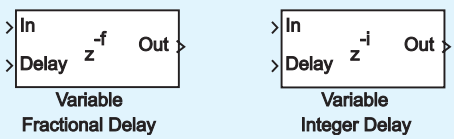
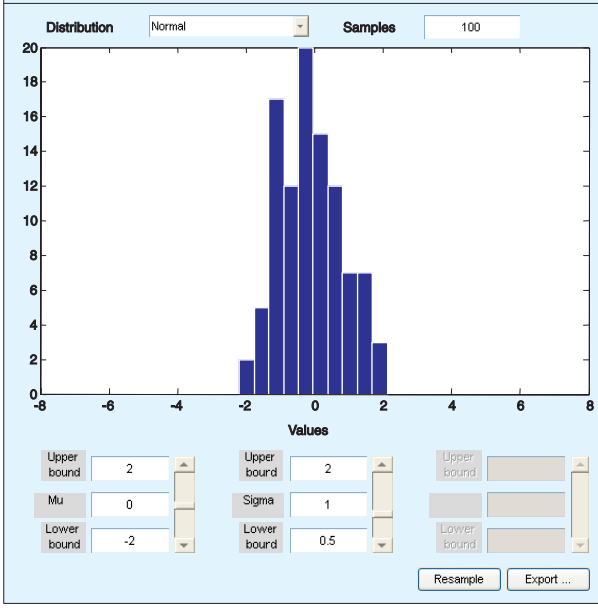
Opóźnienie $\delta_s(kT_0)$, obserwowane w sterowniku w każdym cyklu obliczeń, jest związane z realizacją zadań obliczeniowych. Jego wielkość może się zmieniać w każdym kroku i zależy od stopnia złożoności algorytmu oraz priorytetu przydzielonego zadaniu wyliczenia sterowania w systemie czasu rzeczywistego implementowanym w sterowniku.

Rys. 2 przedstawia przykładowy harmonogram realizacji zadań w sterowniku i jego otoczeniu [3].

Rys. 2a ilustruje sytuację, kiedy opóźnienia w transmisji danych, a także czas przetwarzania danych przez sterownik są pomijalne w stosunku do wartości cyklu T_0. W każdej chwili kT_0, $k = 1, 2, 3, ...$, są dostępne dane o stanie procesu, a wyliczone sterowanie jest realizowane przez urządzenie wykonawcze.

Inaczej jest w sytuacji przedstawionej na rys. 2b. Czasy transmisji danych

Tab. 1. Przykładowe narzędzia środowiska MATLAB/Simulink wspomagające modelowanie układów rozproszonych

Schemat bloku lub jego interfejs	Opis bloku
	Bloki Variable Fractional Delay oraz Variable Integer Delay umożliwiają wprowadzenie zmiennego opóźnienia sygnału wejściowego zgodnie z wektorem podanym na wejście „Delay"
	Blok generujący losowe wartości o rozkładzie jednostajnym lub normalnym
	RandTool: interaktywny generator i graficzny interfejs umożliwiający prezentację 22 rozkładów losowych z określonymi przez użytkownika parametrami. Dostępny w przyborniku Statistic Toolbox.

i ich przetwarzania nie są już pomijalnie małe. Modelowanie działania np. dyskretnego algorytmu PID:

$$u(kT_0) = u((k-1)T_0) + q_0 y(kT_0) + q_1 y((k-1)T_0) + q_1 y((k-2)T_0)$$

prowadzi do stwierdzenia, że przy relacjach czasowych (przykład z rys. 2b) nie można wyliczyć i zastosować sterowania dla chwili kT_0. Dla tej chwili nie tylko nie jest dostępne wyjście procesu – pomiar $y(kT_0)$ – ale także nie są zrealizowane zadania: obliczeniowe w sterowniku i przesyłu sterowania do aktuatora. Najwcześniej sterowanie może być zastosowane w chwili:

$$t = kT_0 + \Delta(kT_0)$$

gdzie: $\Delta(kT_0) = \tau_{sc}(kT_0) + \tau_{ca}(kT_0) + \delta_s(kT_0)$.

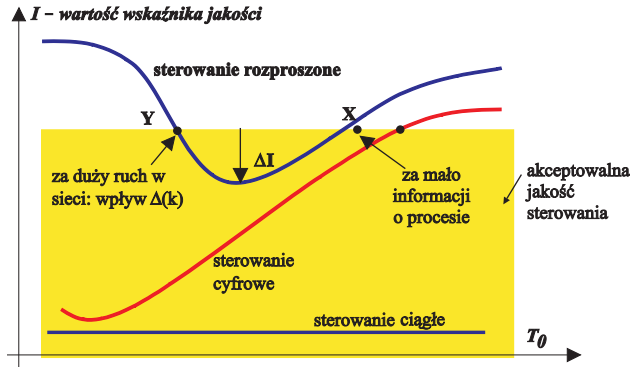
To będzie „prawdziwy" czas, po którym sygnał sterujący zostanie dostarczony do procesu. Może on istotnie różnić się od czasu wynikającego ze stałego cyklu założonego na początku.

Warto też zauważyć, że dla warunków realizacji obliczeń i transmisji danych (rys. 2b), proponowany algorytm sterowania PID nie spełnia zasady przyczynowości, bowiem wymaga danych, które w chwili kT_0 nie są dla sterownika jeszcze dostępne.

Należy zatem stwierdzić, że tradycyjne modele stosowane przy projektowaniu dyskretnych układów sterowania, zakładające periodyczny dopływ danych do sterownika z okresem T_0 i co najwyżej stałe opóźnienie, mogą okazać się zbyt uproszczone dla rozproszonych układów sterowania, a więc nie odzwierciedlać prawdziwych relacji czasowych tam występujących.

Opisane powyżej osobliwości można identyfikować, modelować i symulować, a następnie formułować zadania uodpornienia rozproszonego układu automatyki na tego typu zakłócenia poprzez doskonalenie algorytmów sterowania. To ostatnie podejście wynika z obserwacji, iż pewne niedotrzymanie punktualności w dostarczeniu sygnału sterującego lub przejściowy deficyt informacji o stanie procesu, wywołane np. niedoskonałościami kanału transmisji danych, nie dyskwalifikują rozproszonego systemu sterowania, a tylko obniżają jakość sterowania. A jakość tę można w znacznym zakresie przywrócić poprzez właściwy wybór i strojenie algorytmu sterowania (rys. 3).

Jedną z metod określania zakresu akceptowalnej jakości sterowania (czyli położenia punktów X i Y na



Rys. 3. Zależność wartości wskaźnika jakości sterowania od cyklu próbkowania/sterowania, ΔI – przywracanie jakości sterowania

rys. 3) jest wykonanie badań symulacyjnych. O ile do określenia położenia punktu X na ogół wystarczają standardowe biblioteki modelujące liniowe układy z czasem dyskretnym, o tyle do określenia składników przesunięcia czasowego $\Delta(k)$ zazwyczaj potrzebne są bardziej wyspecjalizowane narzędzia symulacyjne.

TrueTime – prawdziwy czas

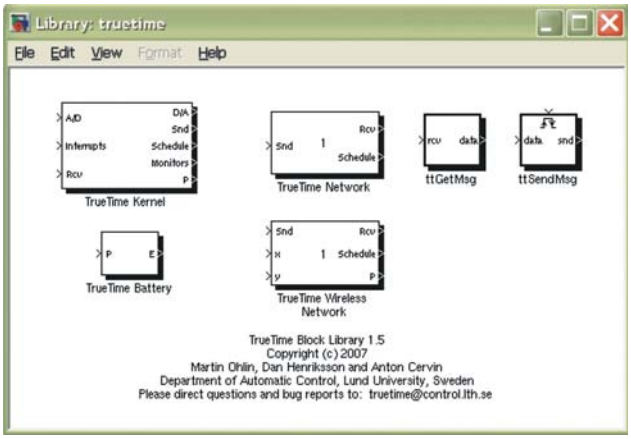
Przystępując do symulacji rozproszonego układu sterowania komputerowego należy zadbać o narzędzia umożliwiające modelowanie jego podstawowych elementów składowych. Podstawowych bloków, umożliwiających modelowanie układu z rys. 1, dostarczają biblioteki środowiska MATLAB/Simulink [4] oraz jego przybory *Signal Processing, Communication Toolbox* i inne. Dynamikę liniową i nieliniową procesu, regulatory ciągłe i dyskretne, bufory i bloki modelujące opóźnienia bez trudu zestawiamy i konfigurujemy ze standardowych bloków dostępnych w bibliotece Simulink. Najbardziej istotnymi dla symulacji zjawisk występujących w układach rozproszonych będą funkcje (bloki) probabilistyki oraz symulacji opóźnień. Przy odpowiednim połączeniu zawartych w przybornikach funkcji można uzyskać różnorodne rozkłady opóźnień (tab. 1).

Standardowe przybory programu MATLAB i środowisko Simulink nie dostarczają narzędzi umożliwiających symulację różnych protokołów transmisji danych ani innych efektów „sieciowych" wpływających na dynamikę układu sterowania z rys. 1, takich jak np. utrata danych w wyniku złych warunków propagacji sygnału w sieciach bezprzewodowych lub też sposób synchronizacji poszczególnych elementów składowych systemu rozproszonego.

Przybornik *TrueTime* [5, 6] jest bezpłatnym narzędziem stworzonym na bazie programu MATLAB/Simulink, służącym do symulacji sieci transmisji danych w sieciach przewodowych i bezprzewodowych

Tab. 2. Algorytmy szeregujące zadania bloku TrueTime Kernel

Nazwa	Skrót	Zasada tworzenia kolejki zadań
Fixed-priority	*prioFP*	Priorytet zadań
Rate-monotonic	*prioRM*	Czas trwania zadań
Deadline-monotonic	*prioDM*	Maksymalny czas wykonania zadania, określony podczas jego definiowania
Earliest-deadline-first	*prioEDF*	Czas pozostały do zakończenia wykonania zadania



Rys. 4. Bloki biblioteki TrueTime, v. 1.5 [5]

jak i modelowania zależności czasowych występujących w sterownikach umieszczonych w takiej sieci.

Z poziomu programu Simulink przybornik ten udostępnia bloki przedstawione na rys. 4, które po zainicjalizowaniu są gotowe do łączenia z dowolnymi elementami wchodzącymi w skład bibliotek programu Simulink.

Blok *TrueTime Kernel* (rys. 4) można traktować jako model komputera znajdującego się w sieci i przetwarzającego dane w czasie rzeczywistym – zgodnie ze zdefiniowanymi przez użytkownika zadaniami. Zadania definiowane są za pomocą funkcji *ttCreateTask* lub *ttCreatePeriodicJob*. Blok *TrueTime Kernel* symuluje węzeł sieci z jądrem czasu rzeczywistego, przetwornikami A/D, D/A, wejściem przerwań zewnętrznych (może to być zdarzenie lub sygnał zegara) oraz wyjściem poboru mocy, używanym w przypadku, gdy chcemy symulować węzeł zasilany baterią.

Konfigurowanie bloku polega na określeniu liczby wejść kanałów A/D lub wyjść D/A oraz algorytmu szeregującego zadania, dostępnego w czterech wariantach opisanych w tab. 2. Dodatkowo użytkownik może ustalić zużycie energii oraz szybkość pracy procesora jednostki (związane z poborem energii) – parametry można zmieniać podczas trwania symulacji.

Blok *Network* symuluje metodę dostępu do sieci transmisji danych (*Media Access Control, MAC*) oraz transmisje pakietów w sieci. Należy podkreślić, że biblioteka *TrueTime* umożliwia symulacje warstwy MAC oraz transmisji danych w sieciach przewodowych i bezprzewodowych na poziomie pakietów – jest to najmniejsza jednostka, o jakiej informacje może uzyskać użytkownik. Wszystkie dane dzielone są na pakiety o założonej wielkości, a następnie wysyłane za pośrednictwem symulatorów sieci.

Tab. 3. Właściwości bloku TrueTime Network [5, 9]

Nazwa pola	Użycie
Bandwidth allocations	Pole aktywne dla protokołu *FDMA* – wektor określający udział pasma dla węzłów nadających. Poszczególne wartości odnoszą się do kolejnych węzłów w sieci, suma elementów wektora nie może przekroczyć 1.
Slot size	Pole aktywne dla protokołu *TDMA* – na podstawie tej wartości wyliczona zostaje wielkość okna czasowego na przeprowadzenie transmisji: t = slotsize/datarate.
Cyclic schedule	Aktywne dla protokołu *TDMA* – wektor identyfikatorów (ID) gałęzi sieci określający cykliczną kolejność wysyłania wiadomości. Wartość zero oznacza, że żadna z gałęzi nie może przesyłać informacji w tym czasie.
Total switch memory	Pole aktywne dla modelu i protokołu *Switched Ethernet* – maksymalna wielkość wewnętrznej pamięci, dostępnej do przechowywania wiadomości.
Switch buffer type	Dla modelu *Switched Ethernet* – określa sposób alokowania pamięci:· ■ *Common buffer* – wszystkie wiadomości przechowywane są w kolejce FIFO (*First In First Out*) i współdzielą jeden obszar pamięci· ■ *Symmetric output buffers* – pamięć podzielona jest na równe części, po jednej dla każdego wyjścia przyłączonego do przełącznika. Kiedy kolejka danego wyjścia wyczerpie dostępny dla niego rozmiar pamięci, nie zostaną przyjęte kolejne wiadomości.
Switch overflow behavior	Dla protokołu *Switched Ethernet* pole to określa akcje podejmowane przez urządzenie w przypadku otrzymania nowej wiadomości, gdy wyczerpany został limit pamięci:· ■ *Retransmit* – otrzymana wiadomość zostaje zignorowana, zostaje wysłany komunikat do węzła nadawcy z żądaniem powtórzenia transmisji wiadomości· ■ *Drop* – otrzymana wiadomość zostaje tylko zignorowana.

Inicjalizacja bloku *Network* odbywa się jeden raz dla każdego węzła, który z niego korzysta. Określa się podczas niej identyfikator ID sieci, adres rozpatrywanego węzła i nazwę obsługi przerwania, które powinno być wywołane po nadejściu wiadomości.

Obecnie dostępna wersja 1.5 biblioteki *TrueTime* udostępnia 6 modeli sieci przewodowych (protokoły: CSMA/CD, CSMA/AMP, RoundRobin (Token Bus), FDMA, TDMA, Switched Ethernet) oraz dwa modele sieci bezprzewodowych: 802.11b (WLAN) i 802.15.4 (ZigBee). Od maja 2009 r. jest dostępna wersja „beta" biblioteki TrueTime 2.0, zawierająca blok *TrueTime Ultrasound Network*, który można wykorzystać do symulacji lokalizacji obiektów mobilnych.

Przy konfiguracji modelu sieci istotny jest parametr *Loss probability*, który określa prawdopo-

Rys. 5. Przykładowe okno konfiguracyjne bloku *TrueTime Network* [5]

dobieństwo utraty wiadomości podczas transmisji. Wiadomości utracone generują ruch w sieci, ale nie docierają do odbiorcy.

Parametr *Minimum frame size* określa minimalną wielkość wiadomości w bitach – wiadomości krótsze są uzupełniane bitami. Pola niewypełnione (nieaktywne) na rys. 5 są używane tylko do określonych protokołów komunikacyjnych (tab. 3).

Działanie bloku *TrueTime Wireless Network* jest analogiczne do przedstawionego powyżej bloku sieci przewodowych. Zasadniczą różnicą jest wprowadzenie funkcji charakteryzującej możliwość straty sygnału oraz wejść określających współrzędne geograficzne dołączonych węzłów. Te ostatnie umożliwiają symulację wpływu warunków propagacji.

Węzły sieci modelowane za pomocą bloku *Kernel* mogą korzystać z zewnętrznego źródła zasilania modelowanego blokiem *TrueTime Battery*. Blok ten ma jeden parametr – jest to energia początkowa wyrażona w watosekundach. Poziom aktualnego naładowania baterii może być modyfikowany podczas trwania symulacji. Ustawienie „on-line" tego parametru jest pomocne przy symulacji programowego przełączania węzłów w stan uśpienia.

Modelowanie rozproszonego układu sterowania

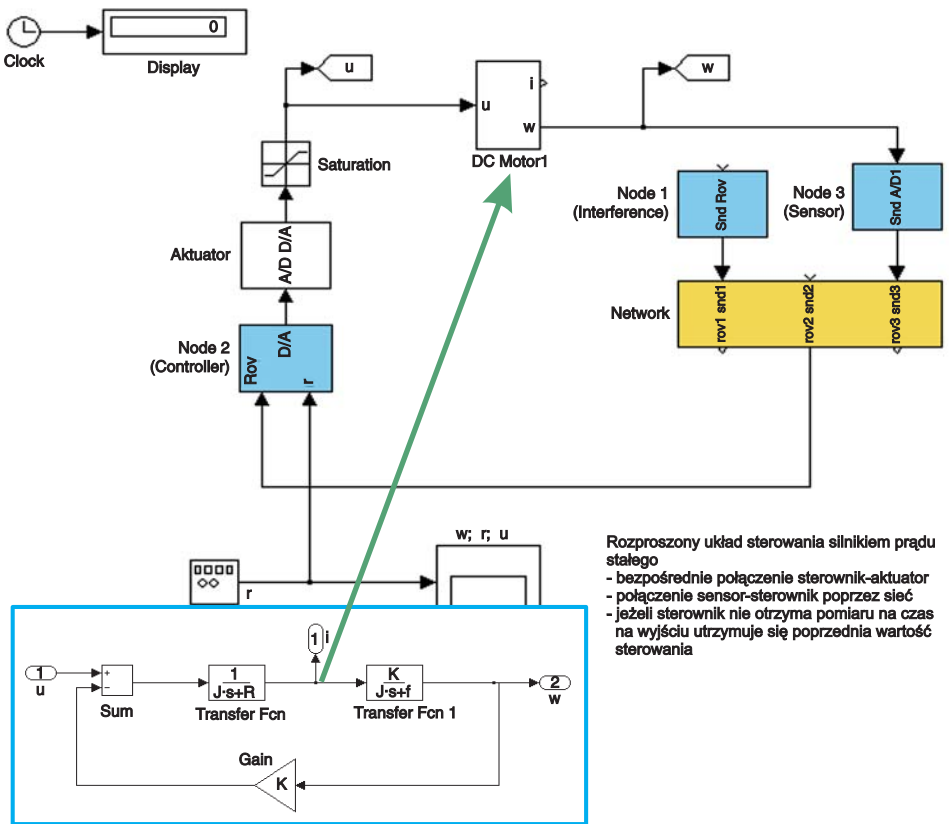
Rys. 6 przedstawia przykład modelu symulacyjnego rozproszonego układu sterowania cyfrowego skonfigurowanego w środowisku *Simulink/TrueTime*. Jak wynika z rysunku, połączenie sieciowe występuje tylko między czujnikiem a regulatorem. *Plant,* to w tym przypadku liniowy model silnika prądu stałego [7]:

$$H(s)=\frac{\omega(s)}{u(s)}=\frac{A}{(1+\tau_1 s)(1+\tau_2 s)}$$

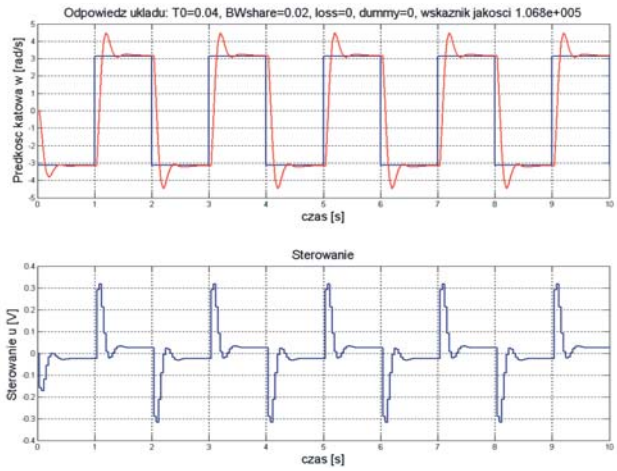
gdzie: ω – prędkość kątowa wału, u – napięcie sterujące. Stałe czasowe modelu wynoszą: τ_1 = 503 ms, τ_2 = 1,4 ms, a wzmocnienie A = 10^7 rad/Vs. W układzie zastosowano próbkowanie z okresem T_0 = 40 ms, co zgadza się z okresem próbkowania zalecanym dla tego modelu przez odpowiednie kryteria (18 – 50 ms). Zastosowano dyskretny regulator PI (*Node* 2 na rys. 6) o transmitancji

$$C(z)=k_p\left(1+k_i\frac{z}{z-1}\right)$$

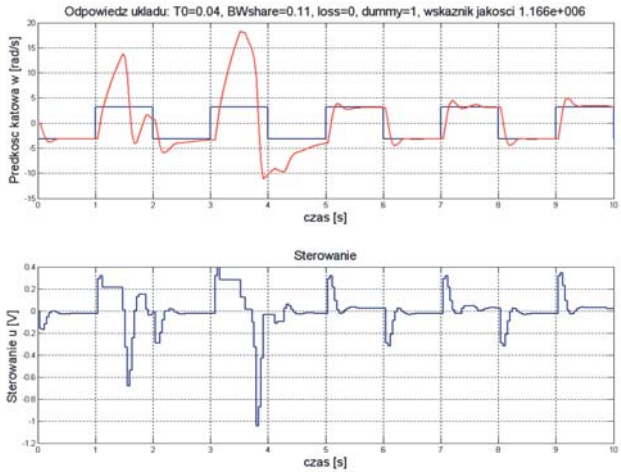
i nastawach: k_p=46*10^{-3}, k_i=93*10^{-3}.



Rys. 6. Schemat symulacyjny rozproszonego układu sterowania silnikiem prądu stałego [9]



Rys. 7. Odpowiedź układu na falę prostokątną, niskie obciążenie sieci [9]



Rys. 8. Odpowiedź układu na falę prostokątną, sieć obciążona [9]

Podstawowe założenia modelu z rys. 6 to:

- bezpośrednie połączenie między regulatorem a urządzeniem wykonawczym (aktuatorem)
- synchronizacja zegarowa – zarówno w przypadku czujnika jak regulatora
- zastosowanie bufora umieszczonego w regulatorze i podtrzymującego poprzednią wartość otrzymaną z czujnika aż do momentu nadejścia nowej próbki
- blok sieci *TrueTime Network*, w którym zdefiniowano sieć Ethernet (protokół CSMA/CD) o następujących parametrach: *date rate* = 80 000 (bit/s), *minimum frame size* = 400 (bitów).

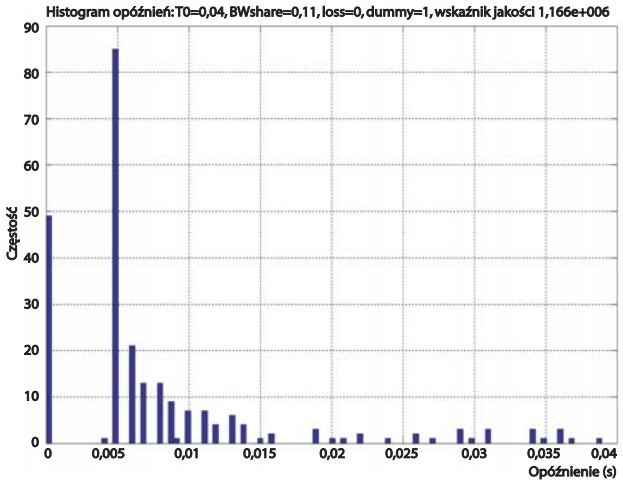
Udział traconych pakietów określany jest jako liczba z przedziału 0–1 w odpowiednim polu właściwości bloku T*rueTime Network*, a ustaloną wartość można odczytać jako parametr *loss*.

Za opóźnienia przetwarzania w regulatorze jest odpowiedzialne dodatkowe zadanie regulatora o wyższym priorytecie (zadanie *dummy*), którego czas trwania ustawiony został na 40 µs. O tym, czy zadanie zostało uwzględnione w symulacji informuje wartość parametru *dummy* (wartość 0 – wyłączone, 1 – włączone).

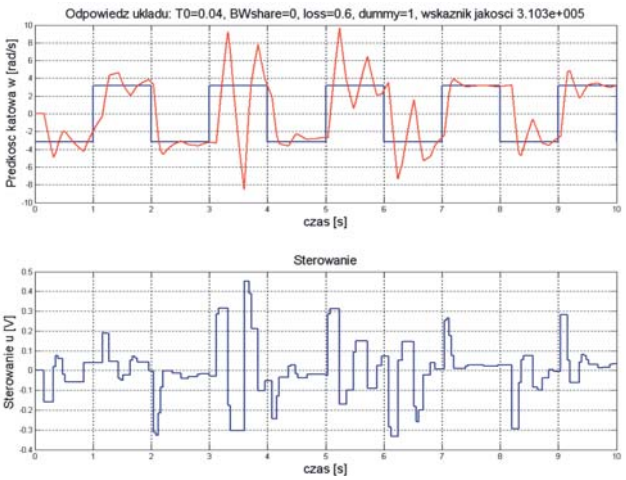
Modelowanie opóźnień przesyłu jest zrealizowane za pomocą dodatkowego węzła sieci *(Interference*). Zadanie generujące ruch w sieci jest wyzwalane w tym węźle co 1 ms, z wykorzystaniem generatora liczb losowych o rozkładzie jednostajnym. Parametr *Bwshare* określa średnią zajętość medium komunikacyjnego.

Blok *Kernel* regulatora (*Node* 2, rys. 6) wywołuje model dyskretnego regulatora PI, obliczającego odpowiednią wartość sterowania. Jakość sterowania (nadążania za sygnałem prostokątnym) oceniano za pomocą kwadratowego wskaźnika jakości.

Rys. 7 przedstawia odpowiedź modelu silnika – nadążanie za zadaną prędkością kątową wału silnika – przy niskim obciążeniu sieci i bez symulacji zadań wewnętrznych sterownika. Rys. 8 przedstawia realizację tego samego zadania sterowania przy dodatkowym, 11 % obciążeniu sieci i symulacji zadania wewnętrznego sterownika. Na rys. 9 przedstawiono histogram występujących w tym modelu opóźnień. Rys. 10 prezentuje wyniki symulacji dla założonej utraty danych na średnim poziomie 60 %. Na rys. 8 i rys. 10 wyraźnie obserwujemy pogorszenie jakości sterowania (należy zwrócić uwagę, że na rys. 8 i rys. 10 jest inna skala osi pionowej).



Rys. 9. Histogram symulowanych opóźnień w sieci Ethernet, *BWshare* = 0,11, słupek zlokalizowany w „0" obrazuje liczbę pakietów, których opóźnienie jest większe niż 40 ms [9]



Rys. 10. Odpowiedź układu na falę prostokątną, sieć nieobciążona, losowa utrata danych (*loss* = 0,6) [9]

Pogorszenie jakości sterowania potwierdza wzrost wartości wskaźnika jakości liczonego za czas 100 cykli wartości zadanej. Dla niskiego obciążenia sieci, opóźnienia były dobrze kompensowane (wyrównywane do wartości stałej) przez zastosowany w sterowniku bufor.

Należy podkreślić, że modelowanie i symulacja metodami „klasycznymi" – z wykorzystaniem standardowych narzędzi środowiska Simulink – nie ujawnią efektów wnoszonych przez rozproszony charakter aplikacji.

Przywrócenie jakości sterowania w rozważanym modelu może nastąpić przez [1]:

- zwiększanie przepustowości kanałów transmisji danych lub/i ograniczenie ilości przesyłanych danych
- poprawę determinizmu protokołu transmisyjnego
- właściwy dobór okresu próbkowania i sterowania
- algorytmiczną kompensację efektów wnoszonych przez sieć.

Współczesne sterowniki i stacje procesowe DCS mają możliwość odczytywania wejść analogowych wielokrotnie szybciej niż samo wykonanie algorytmu. Poprzez grupowanie takich próbek (co ogranicza ruch w sieci!) i odtwarzanie stanu na podstawie modelu sterowanego procesu, można w pewnym zakresie zmniejszać opóźnienia komunikacyjne [8].

Należy także zwrócić uwagę, że blok modelujący sieć (węzeł *Network* na rys. 6) umożliwia symulację kilku pętli sprzężenia zwrotnego, z algorytmami sterowania realizowanymi przez pojedynczy sterownik. W tym celu należy w pliku inicjalizującym działanie węzła modelującego sterownik (blok typu *TrueTime Kernel*) określić odpowiednią liczbę kanałów wejściowych i wyjściowych, strategie harmonogramowania zadań i parametry algorytmu szeregującego zadania (tab. 2).

Podsumowanie

Rozproszone rozwiązania, zwłaszcza w układach automatyki, stawiają nowe zadania projektantom sprzętu, oprogramowania i algorytmów sterowania. W szczególności obecność kanału transmisji danych w pętli sprzężenia zwrotnego może wnosić do modelu dynamiki systemu sterowania pewne komplikacje związane z opóźnieniami transmisji danych lub nawet z możliwością ich utraty. Jest to szczególnie istotne w systemach sterowania krytycznych ze względu na bezpieczeństwo procesu.W systemach takich istotne jest symulacyjne testowanie wpływu efektów wnoszonych przez sieć na jakość sterowania, a w szczególności ograniczeń czasowych wnoszonych przez procesory i kanały transmisji danych umieszczone w pętli sprzężenia zwrotnego.

Przedstawiona w artykule biblioteka *TrueTime* jest narzędziem umożliwiającym takie badania. Autorzy oprogramowania przygotowali szereg metod – mających dokumentację dostępną z programu MATLAB – bardzo przydatnych do symulacji rozproszonych układów sterowania. Twórcy pakietu postarali się również o wprowadzenie użytkownika w środowisko za pomocą licznych przykładów.

Biblioteka ma też pewne niedostatki. Brakuje interfejsu do modelowania dokładności przetworników A/D i D/A, ale można go zaimplementować za pomocą dostępnych bloków i metod pakietu MATLAB/Simulink. Twórcy biblioteki nie przewidzieli także możliwości symulacji utraty części pakietu danych, składającego się na pełną wiadomość wysyłaną za pośrednictwem sieci.

Uzupełnieniem biblioteki *TrueTime* może być pakiet *Jitterbug* [10], umożliwiający symulację w środowisku MATLAB/Simulink z uwzględnieniem dynamiki przetwarzania zadań przez komputery realizujące zadania czasu rzeczywistego. Można symulować losowe opóźnienia poszczególnych zadań realizowanych przez system operacyjny sterownika i badać wpływ tych zjawisk na jakość sterowania. Jest to kolejny krok w kierunku dokładniejszego, modelowego odwzorowania zależności czasowych, które występują we współczesnych układach sterowania.

Bibliografia

1. Grega W.: *Sterowanie cyfrowe w systemach skupionych i rozproszonych*. Monografie Komitetu Automatyki i Robotyki PAN, 7, Kraków 2004.
2. Leohold J.: *Communication Requirements for Automotive Systems*. Keynote, WFCS 2004, 5th IEEE Workshop on Factory Communication Systems, Wien, http://www.ict.tuwien.ac.at/wfcs2004/
3. Grega W.: *The design of distributed control systems: from theoretical to applied timing*. [w:] Recent Advances in Control and Automation, Academic Publishing House EXIT, The Committee on Automatic Control and Robotics of the Polish Academy of Sciences, Polish Neural Network Society, s. 343–352, 2008.
4. Mrozek B., Mrozek Z.: *MATLAB i Simulink*. Poradnik użytkownika, Wyd. Helion, 2004.
5. [http://www.control.lth.se/truetime/]
6. Ohlin M., Henrikson D., Cervin A.: *TrueTime 1.5-Reference Manual*, Report , Department of Automatic Control, Lund University, 2007.
7. Khan Z. H., Genon-Catalot D., Thiriet J. M.: *Co-design in Heterogeneous Wireless Networked Control Systems*. Proceedings of International Multiconference on Computer Science and Information Technology, October 12–14, Mrągowo 2009.
8. Tutaj A.: *Odporne algorytmy sterowania rozproszonego*, Rozprawa Doktorska, AGH, Kraków, 2008.
9. Hopek B., Krosta B.: *Metody symulacji rozproszonych układów sterowania*. Praca Dyplomowa Akademia Górniczo-Hutnicza, Wydział Elektrotechniki Automatyki Informatyki i Elektroniki (promotor: W. Grega), Kraków 2008.
10. Lincoln B., Cervin A.: Jitterbug: *A tool for analysis of real-time control performance*. 41st IEEE Conference on Decision and Control, Las Vegas, NV, 2002.