Józef GROMBA[(1),(2)], Grzegorz SIEKLUCKI[(1)], Sylwester SOBIERAJ[(1)]

(1) Akademia Górniczo-Hutnicza w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Katedra Energoelektroniki i Automatyki Systemów Przetwarzania Energii
(2) F.H.U. Auspol, dział Automatyki i Budowy maszyn

doi:10.15199/48.2019.11.57

Pomiar i analiza czasu wykonania algorytmów PID2DOF w sterownikach Siemens S7-1500

Streszczenie. *Celem artykułu jest analiza przeprowadzonych pomiarów czasów wykonania algorytmu PID w sterowniku PLC S7-1500. Wyniki eksperymentu pokazują zmienny czas obliczeń algorytmu regulatora dla różnych jego realizacji. Sterowanie cyfrowe w czasie rzeczywistym musi być obliczone w zdefiniowanym reżimie czasowym, który jest uzależniony od czasu próbkowania regulatora. Pomiary zostały wykonane przy pomocy instrukcji „RUNTIME". Mierzonym algorytmem był zaproponowany przez autorów regulator PID2DOF oraz firmowe rozwiązanie PID Compact V2, który również jest o strukturze 2DOF.*

Abstract*. The aim of this paper is the analysis of performed measurements of PID algorithm execution times in PLC S7-1500. The lack of constant calculation time of the controller algorithm for its various realizations is presented. Real-time digital control must be calculated in a defined time regime which depends on the sampling time of the controller. The measurements were made using the "RUNTIME" instruction. The algorithm was measured by the PID2DOF controller proposed by the authors (which was written in SCL Language) and the original PID Compact V2 (Siemens) solution.* ***Measurement and analysis of the calculation time of pid2dof algorithms in Siemens s7-1500 controllers.***

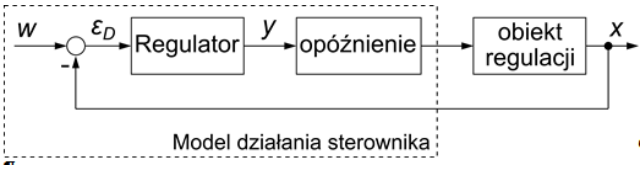
Słowa kluczowe: PLC, Siemens S7, czas obliczeń algorytmu, czasy dostępów pamięci, próbkowanie, regulator PID2DOF
Keywords: PLC, Siemens S7, algorithm calculation time, memory access times, sampling, PID2DOF controller.

Wstęp

W programowaniu sterowników PLC mogą pojawić się problemy z realizacją założonych celów. Przyczyną mogą być zbyt długie czasy obliczeń sterowania (czasy wykonania algorytmu). Z uwagi na fakt szerokiego zastosowania regulatorów o dwóch stopniach swobody (PID2DOF) [1] również w PLC, w artykule przeprowadzono analizę czasu obliczeń algorytmu dla sterownika S7 firmy Siemens. Badania zostały przeprowadzone na sterowniku S7-1500 z jednostką CPU 1515 (6ES7 515-2AM01-0AB0) programowanym przy pomocy TIA Portal V15.

Cyfrowe przetwarzanie danych w czasie rzeczywistym

Sterowanie w czasie rzeczywistym różnymi obiektami automatyki związane jest z bezwzględnym przestrzeganiem twierdzenia o próbkowaniu wraz z towarzyszącymi ograniczeniami [2]. Wówczas, jakość regulacji założona w procesie projektowania systemu jest możliwa do spełnienia (praktyka nie odbiega znacznie od teorii). Dodatkowo, czas wykonania algorytmu regulacji powinien być około 10. krotnie krótszy od przyjętego czasu próbkowania T_p. Przy spełnieniu tego założenia czas od pomiaru sygnału wyjściowego obiektu do wysterowania go może zostać pominięty. W przeciwnym wypadku cyfrowy układ automatycznej regulacji należy analizować zgodnie z rysunkiem 1, gdzie przyjęto oznaczenia zgodnie ze sterownikami S7 i regulatorem PID Compact V2 [3]: w - wartość zadana, y - sterowanie wyznaczone przez regulator, x - sygnał wyjściowy (pomiarowy) z obiektu.



Rys.1. Model regulacji cyfrowej realizowanej w oparciu o PLC

W układach automatycznej regulacji obiekt i regulator zwykle zapisuje się w postaci transmitancji i podaje się opóźnienie. Wówczas analiza oraz synteza układu jest standardowa – dynamika układów o parametrach skupionych. W przypadku gdy czas obliczeń sterowania na podstawie sygnałów pomiarowych jest porównywalny z czasem próbkowania występuje konieczność uwzględnienia go w postaci elementu opóźniającego:

$$(1) \qquad G_{OA}(s) = e^{-\tau s} \approx \frac{(s-s_{z1})(s-s_{z2})(s-s_{z3})\cdots}{(s-s_{b1})(s-s_{b2})(s-s_{b3})\cdots}$$

gdzie: τ jest opóźnieniem (czas od wyzwolenia przetwornika AC do uzyskania sygnału sterującego na wyjściu CA), s_{zi} są zerami transmitancji, s_{bj} jej biegunami.

Z wyrażenia (1) widać, że element opóźniający, który przedstawia się w postaci funkcji wymiernej ma nieskończoną liczbę biegunów (pierwiastków mianownika), stąd jest to element nieskończenie wymiarowy. Jeżeli opóźnienie (rysunek 1) będzie znaczne to dokładna analiza układu jest praktycznie niemożliwa. Jeżeli natomiast zostanie określony zakres opóźnienia, to można analizować najgorsze przypadki – $\tau = \tau_{max}$.

Realizacja PID2DOF w sterowniku S7: Formy zapisu danych w PLC

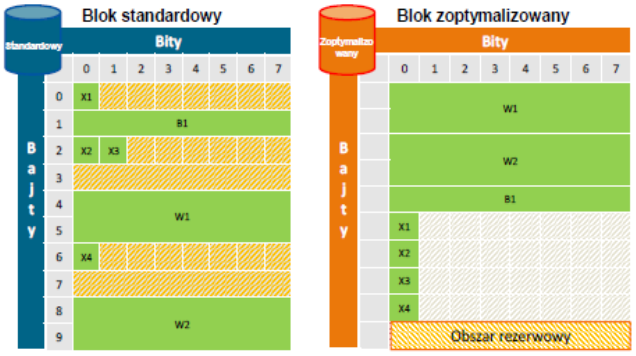
W sterownikach PLC firmy Siemens serii S7 istnieje kilka możliwości zapisu danych w pamięci kontrolera.

Dzięki odpowiedniemu dobraniu rodzaju pamięci, kreator systemu ma możliwość maksymalnego wykorzystania możliwości sterownika i realizację złożonych algorytmów przy założeniu możliwie najkrótszego czasu wykonania obiegu pętli programu.

Oprogramowanie TIA Portal daje dostęp do dwóch głównych obszarów pamięci sterownika PLC – pamięć lokalną oraz pamięć globalną. Do pamięci lokalnej dostęp możliwy jest jedynie w obrębie danego bloku FC (Function), FB (Function Block) lub OB (Organization Block). Istnieje jednak możliwość dostępu do pamięci lokalnej za pośrednictwem interfejsu wejść/wyjść danego bloku lub dostęp do bloku instancji. Pamięć globalna umożliwia dostęp do danych z każdego miejsca programu PLC oraz HMI. Do pamięci globalnej zalicza się pamięć sprzętową (bitową, liczniki, timery) oraz bloki danych DB będące strukturą do przechowywania, sortowania danych.

Dodatkowy parametr określający zoptymalizowany dostęp do bloków danych (Optimized block access)

wymusza symboliczny dostęp do zmiennych zdeklarowanych w danym bloku. Wyłączenie tego parametru pozwala na adresowanie absolutne zmiennych. Różnice pomiędzy blokiem standardowym, a blokiem zoptymalizowanym przedstawiono na rysunku 2. Najistotniejszą różnicą pomiędzy blokami danych jest sortowanie zmiennych w zależności od ich typu (blok zoptymalizowany). Dzięki sortowaniu zmiennych w bloku zoptymalizowanym sterownik nie dodaje obszarów rezerwowych pomiędzy zmiennymi różnego typu dzięki czemu blok zajmuje mniejszy obszar pamięci.



Rys.2. Budowa bloku standardowego oraz zoptymalizowanego w S7-1500 [4]

Dla pamięci globalnej i lokalnej istnieje możliwość dodania funkcji podtrzymywania (Retentivity). Pozwala to na przeniesienie danych z pamięci operacyjnej do pamięci stałej w sytuacji awaryjnej np.: w przypadku awarii zasilania. Każdy rodzaj pamięci cechuje się innym czasem dostępów do zmiennych, zostało to przedstawione w dalszej części artykułu..

Realizacja PID2DOF w sterowniku S7:Algorytm regulatora PID2DOF

Teoria regulacji PID2DOF została szczegółowo omówiona w [1], gdzie zaprezentowany został również najpopularniejszy obecnie regulator o dwóch stopniach swobody (2DOF). Jego implementacja dla sterowników S7 została zaprezentowana w [5]. W niniejszym artykule analizowana jest realizacja cyfrowa regulatora, którą przedstawiono na rysunku 3, gdzie: K_P - wzmocnienie regulatora, T_I - czas zdwojenia, T_D - czas wyprzedzenia, T_1 - stała czasowa filtru części różniczkującej, T_t - czas śledzenia dla układu anti-windup, T_s - czas próbkowania.

W sterownikach PLC serii S7 zaimplementowane są również firmowe bloki regulatorów PID2DOF (PID Compact V2) i czas realizacji obliczeń tego bloku został również zamieszczony w tabeli 1 (PID_{10}).

Realizacja PID2DOF w sterowniku S7: Implementacja algorytmu

Doświadczenia mające na celu zbadanie czasów dostępów do różnych rodzajów pamięci sterownika PLC polegały na implementacji zaproponowanego przez autorów algorytmu PID2DOF w języku SCL. Wcześniej omawiany regulator został zrealizowany w bloku FB, aby sprawdzić dostęp do pamięci lokalnej oraz w bloku FC w celu sprawdzenia dostępów do różnych konfiguracji

pamięci globalnej. Algorytm regulatora przyjął następującą postać (FB):

```
#"eD" := (#"c" * #"w-in") - #"x-in";
#"ep" := (#"b" * #"w-in") - #"x-in";
#"e" := #"w-in" - #"x-in";
#"yD" := #"alpha D" * (#"eD" - #"eD-1") + #"beta D" * #"yD-1";
#"yP" := #"Kp" * #"ep";
#"ya_1" := (#"Kp" / #"TI") * #"e" + #"yc";
#"yI" := #"alpha I" * (#"ya_1-1" + (#"beta I" * #"ya_1")) + #"yI-1";
#"yA" := #"yD" + #"yP" + #"yI";
#"y" := #"yA";
IF #"yA" > #"ymax" THEN
   #"y" := #"ymax";
END_IF;
IF #"yA" < #"ymin" THEN
   #"y" := #"ymin";
END_IF;
#"yc" := (#"y" - #"yA") / #"Tt";
#"eD-1" := #"eD";
#"yD-1" := #"yD";
#"y-1" := #"y";
#"ya_1-1" := #"ya_1";
#"yI-1" := #"yI";
```

Pomiar czsu wykonywania algorytmu: Metody pomiarów całkowitego czasu wykonania programów

Oprogramowanie TIA Portal daje trzy możliwości pomiaru czasów wykonywania algorytmów. -

Pierwszy ze sposobów polega na wykorzystaniu zmiennych tymczasowych w bloku głównym (OB1). Metoda ta pozwala na pomiar czasu wykonania pętli głównej programu. Po wyłączeniu w atrybutach bloku „Main" zoptymalizowanego dostępu otrzymujemy możliwość odczytania zmiennych informujących o najdłuższym, najkrótszym oraz poprzednim czasie cyklu.

Kolejną możliwością pomiarową jest wykorzystanie instancji „RT-INFO" dzięki, której otrzymujemy możliwość odczytu czasów wykonania wskazanego bloku organizacyjnego (OB). Podobnie jak w przypadku pierwszego ze sposobów istnieje możliwość odczytania najdłuższego, najkrótszego, poprzedniego oraz dodatkowo obecnego czasu cyklu.

Ostatnią metodą pomiaru wykorzystaną w przedstawionym artykule jest wywołanie instrukcji „RUNTIME". W przypadku tej metody istnieją dwie możliwości jej wykorzystania. Pierwszą z nich jest pomiar całego programu. W tym celu należy wywołać instrukcję "RUNTIME" w OB1. Pomiar rozpoczyna się od pierwszego wywołania, a wynik zwracany jest po drugim wywołaniu. Instrukcja pomiaru obejmuje wszystkie działania CPU, które mogą wystąpić podczas wykonywania programu, np. przerwania spowodowane zdarzeniami wyższego poziomu lub komunikacją. Drugą możliwością jest odczyt poszczególnych bloków FC, FB lub poszczególnych sekwencji poleceń. Punkt początkowy pomiaru ustawiany jest za pomocą pierwszego wywołania instrukcji. Następnie wywoływany jest mierzony blok lub sekwencja poleceń.

W ostatnim kroku ponowne wywołanie instrukcji pomiaru odczytuje wewnętrzny licznik procesora i oblicza aktualny czas pracy bloku programu lub sekwencji poleceń.

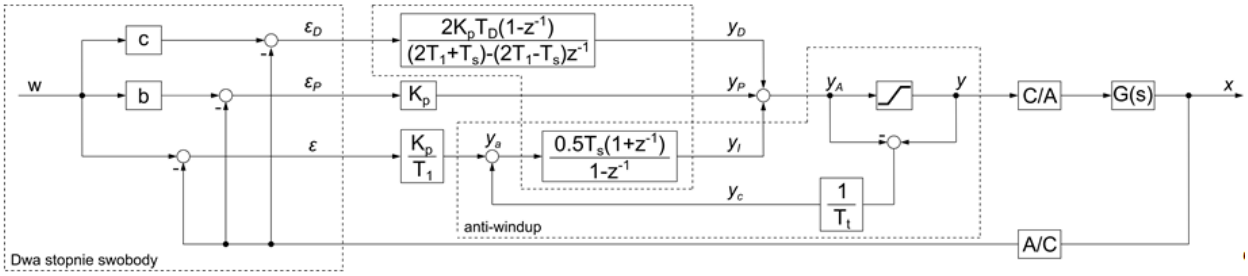
Pomiar czasu wykonywania algorytmu: Wyniki pomiarów

Przeprowadzone badania obejmowały dziesięć różnych sposobów przechowywania danych przez sterownik PLC.

Pamięć globalna:

PID_1 – Blok danych z wyłączonym zoptymalizowanym dostępem oraz włączoną funkcją podtrzymania wartości zmiennych.

PID_2 – Blok danych z włączonym optymalnym dostępem oraz włączoną funkcją potrzymania wartości zmiennych.



Rys.3. Regulator PID2DOF z techniką anti-windup jako system przetwarzania danych (filtr cyfrowy)

PID_3 – Blok danych z wyłączonym optymalnym dostępem oraz wyłączoną funkcją potrzymania wartości zmiennych.

PID_4 – Blok danych z włączonym optymalnym dostępem oraz wyłączoną funkcją potrzymania wartości zmiennych.

PID_5 – Pamięć sprzętowa „PLC Tags" z wyłączoną funkcją potrzymania wartości zmiennych.

PID_6 – Pamięć sprzętowa „PLC Tags" z włączoną funkcją potrzymania wartości zmiennych.

Pamięć lokalna:

PID_7 – Blok danych instancji z wyłączoną funkcją podtrzymania wartości zmiennych.

PID_9 – Blok danych instancji z włączoną funkcją podtrzymania wartości zmiennych.

PID_{10} – Regulator PID Compact V2 z zapisem parametrów wejściowych oraz wyjściowych w pamięci sprzętowej „PLC Tags"

Pamięć lokalna wraz z globalną:

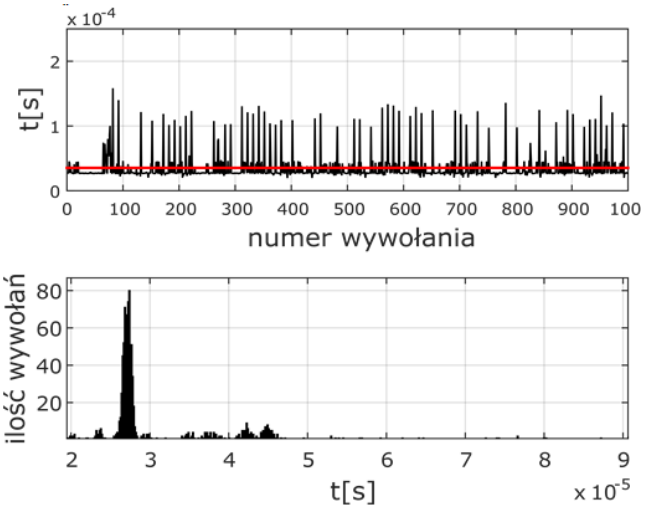
PID_8 – Zmienne wejściowe oraz wyjściowe przetrzymywane w pamięci sprzętowej „PLC Tags" z wyłączoną funkcją potrzymania wartości zmiennych natomiast zmienne wewnętrzne zapisywane w instancji z wyłączoną funkcją podtrzymania.

Tabela 1. Zestawienie czasów wykonania wybranych procedur PID2DOF (gdzie PID_{10} jest implementacją regulatora PID producenta) z techniką anti-windup dla sterownika S7-1500 z CPU 1515.

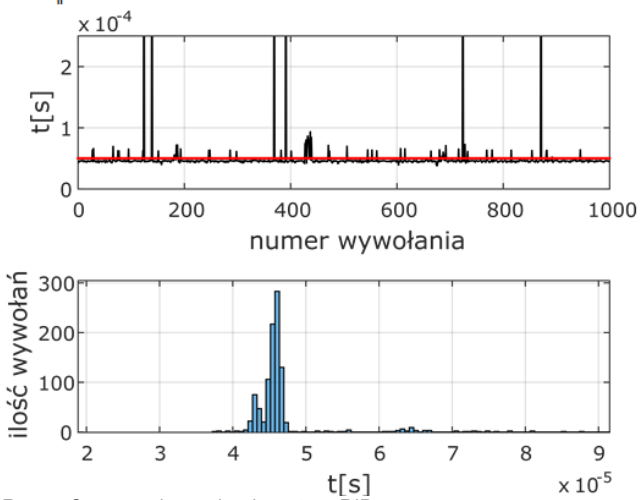
	Wartość min	**Wartość max**	**Wartość średnia**
PID_1	19.45 µs	158.40 µs	35.32 µs
PID_2	14.10 µs	579.03 µs	36.20 µs
PID_3	19.19 µs	597.28 µs	37.01 µs
PID_4	19.59 µs	645.63 µs	37.02 µs
PID_5	19.27 µs	629.15 µs	38.77 µs
PID_6	14.83 µs	1200 µs	40.42 µs
PID_7	30.35 µs	1100 µs	47.02 µs
PID_8	22.52 µs	633.92 µs	47.66 µs
PID_9	37.72 µs	649.18 µs	50.08 µs
PID_{10}	76.02 µs	1200 µs	90.08 µs

Każdy z przedstawionych przypadków był wywoływany oraz mierzony indywidualnie w przerwaniu cyklicznym co 0.1s. Przerwanie to wyznacza czas próbkowania T_s dla algorytmu regulatora. Czasy wykonania instrukcji zapisywano w bloku danych po wyjściu procesora z przerwania cyklicznego. Wyniki pomiarów zestawiono w tabeli 1 w kolejności od najkrótszego do najdłuższego czasu średniego wykonania algorytmu. Na przedstawionych wykresach linią poziomą przedstawiono wartość średnią z 1000 wywołań algorytmu, oraz histogramy przedstawiające najbardziej znaczące (najczęściej występujące) czasy wywołania algorytmu regulatora.

Na rysunku 4 przedstawiono wyniki pomiarów czasów realizacji algorytmu PID2DOF zrealizowanego na zmiennych globalnych w standardowym bloku danych z podtrzymaniem. Przedstawiony przebieg cechuje się najmniejszym czasem średnim oraz najmniejszym czasem maksymalnym wykonania algorytmu.



Rys.4. Czasy wykonania algorytmu PID_1

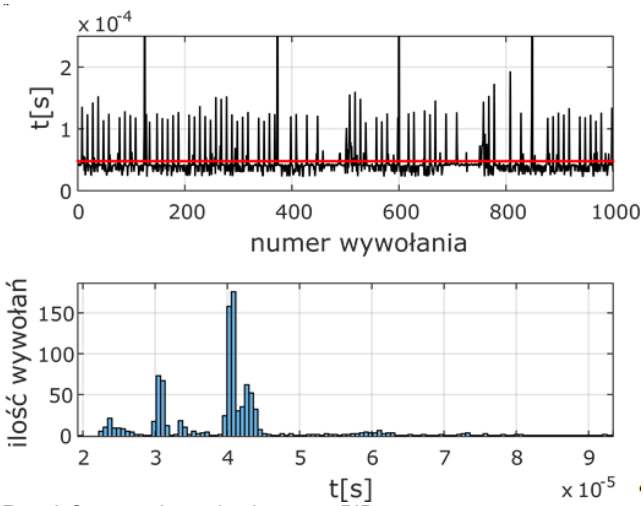


Rys.5. Czasy wykonania algorytmu PID_9

Największy czas średni jak i największy czas minimalny zaobserwowano dla regulatora PID_9 (rysunek 5). Natomiast PID_9 jest najbardziej stabilnym rozwiązaniem pod kątem czasu przetwarzania danych w porównaniu do pozostałych propozycji realizacji zapisu danych. W przedstawionym przypadku powodem tak długiego czasu wykonania algorytmu jest wykorzystanie pamięci lokalnej z funkcją podtrzymania. Podobną sytuację można zaobserwować dla przypadku PID_6 gdzie zarejestrowano najdłuższy czas średni dla zmiennych globalnych – bitowych z włączoną funkcją „Retain". W obu przypadkach zdeklarowane zmienne przechowywane są w tym samym obszarze pamięci sterownika [6].

W celu zbadania większości przypadków mogących wystąpić w procesie programowania systemu

automatycznej regulacji przeprowadzono doświadczenie polegające na wykorzystaniu dwóch rodzajów zmiennych – globalnej oraz lokalnej. Został zdefiniowany blok FB, którego parametry wejściowe oraz wyjściowe były przechowywane w „PLC Tagach". Natomiast pozostałe zmienne zostały zapisane w pamięci Static oraz Temp w zależności od potrzeby podtrzymania wartości w kolejnym wywołaniu bloku. Wyniki doświadczenia przedstawiono na rysunku 6. Zauważyć można niewielką różnicę czasów średnich pomiędzy rozwiązaniem PID_7 i PID_8. Powodem takiego zachowania jest w pierwszym przypadku wykorzystanie zmiennych lokalnych do przechowania danych, natomiast w drugim koniecznością przepisywania wartości zmiennych przy każdorazowym wywołaniu bloku FB.



Rys. 6 Czasy wykonania algorytmu PID_8

Podsumowanie

Na przedstawionych wykresach czasów wykonania algorytmu zaobserwować można bardzo dużą ich nierównomierność oraz często pojawiające się pojedyncze skoki (impulsy). Takie zmiany spowodowane są faktem, iż sterownik podczas badań połączony był z oprogramowaniem TIA Portal w trybie Online, co wymuszało komunikację w przerwaniu pomiędzy urządzeniami. Połączenie to zostało wykonane celowo, aby sprawdzić najbardziej niekorzystne przypadki. Podobna sytuacja ma miejsce w rozwiązaniach zdecentralizowanych układów sterowania np.: komunikacja pomiędzy sterownikami PLC lub z grupą rozproszonych peryferii.

Wartość maksymalną czasu obliczeń sterowania można utożsamiać z normą $\| \ \|_\infty$, czyli określeniem najbardziej niekorzystnej sytuacji. Na taj podstawie można przyjąć (zgodnie z opisem sterowania w czasie rzeczywistym we Wprowadzeniu) zalecane minimalne czasy próbkowania, które przedstawiono w tabeli 2.

Tabela 2. Zestawienie minimalnych proponowanych czasów próbkowania dla poszczególnych rozwiązań (gdzie PID_{10} jest implementacją regulatora PID producenta).

	Tp
PID_1	1.58 ms
PID_2	5.97 ms
PID_3	5.97 ms
PID_4	6.45 ms
PID_5	6.29 ms
PID_6	12 ms
PID_7	11 ms
PID_8	6.34 ms
PID_9	6.5 ms
PID_{10}	12 ms

Podsumowując, gdy wymagane czasy próbkowania są mniejsze od zaprezentowanych, to należy stosować szybsze sterowniki przemysłowe lub specjalizowane układy regulacji. Dodatkowo należy zwracać szczególną uwagę na sposób, miejsce deklaracji zmiennych oraz budować optymalny program przy założeniu jak najkrótszego czasu wykonania kodu.

***Podziękowania**: Składamy serdeczne podziękowania F.H.U. Auspol za udzielone wsparcie techniczne w postaci dostarczenia programowalnych sterowników PLC oraz za wypożyczenie programatora wraz z pakietem oprogramowania TIA Portal v15. Adres sieciowy firmy: http://www.auspol.com.pl*

***Autorzy**: mgr inż. Józef Gromba, e-mail: jozefgromba@gmail.com, dr inż. Grzegorz Sieklucki, e-mail: sieklo@agh.edu.pl; mgr inż. Sylwester Sobieraj, e-mail: sobieraj2@gmail.com; AGH w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej, Katedra Energoelektroniki i Automatyki Systemów Przetwarzania Energii, al. Adama Mickiewicza 30, 30-059 Kraków.*

LITERATURA

[1] Åström K.J., Hägglund T., *"PID Controllers*: *Theory, Design, and Tuning"*, International Society for Measurement and Control, 1995.

[2] Tietze U., Schenk Ch.: *„Układy półprzewodnikowe",* WNT Warszawa 1996.

[3] SIEMENS, *SIMATIC S7-1200, S7-1500 PID control*, Function Manual, 10/2018, 74-97, 252-287.

[4] SIEMENS, "*Przewodnik programowania dla S7-1500/S7-1200",* wydanie 3/2019.

[5] Gromba J., Sieklucki G., Sobieraj S., *„Czasy realizacji algorytmów PID2DOF w sterownikach S7"*, Przegląd Elektrotechniczny (w trakcie druku).

[6] SIEMENS, "*S7-1500, ET 200SP, ET 200pro Structure and Use of the CPU Memory"*, Function Manual, 10/2018, A5E03461664-AC.