Politechnika Wrocławska

MODELOWANIE CYFROWE W PRAKTYCE INŻYNIERSKIEJ

Janusz Staszewski

Oficyna Wydawnicza Politechniki Wrocławskiej
Wrocław 2021

Recenzent
Eugeniusz ROSOŁOWSKI

Korekta
Stanisław GANCARZ

Skład komputerowy
Janusz STASZEWSKI

Projekt okładki
Janusz STASZEWSKI

OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ
Wybrzeże Wyspiańskiego 27, 50-370 Wrocław
http://www.oficyna.pwr.edu.pl;
e-mail: oficwyd@pwr.edu.pl
zamawianie.ksiazek@pwr.edu.pl

ISBN 978-83-7493-166-3
DOI: 10.37190/jasta2021

Motto:

„Czucie i wiara silniej mówią do mnie
Niż mędrca szkiełko i oko"

(Adam Mickiewicz „Romantyczność")

Podręcznik ten dedykuję koleżankom i kolegom z Zespołu Automatyki i Sterowania w Energetyce na Wydziale Elektrycznym Politechniki Wrocławskiej, w podziękowaniu za wspólne lata dotychczasowej pracy.

Spis treści

1. Wstęp

Adam Mickiewicz, pisząc słowa będące mottem do tego podręcznika, nie zdawał sobie sprawy z ogromnego postępu technicznego, jaki czeka ludzkość w najbliższych wiekach. Bezpowrotnie minęły czasy, gdy wystarczyły *„czucie i wiara”*. Dziś wiedza i doświadczenie są niezbędne, aby tworzyć nowe, przynajmniej w dziedzinie szeroko rozumianych nauk przyrodniczych. Jednak nie należy zupełnie dyskredytować *„czucia i wiary”*, bowiem *„czucie”* nieraz pozwala na wybranie odpowiedniej drogi badań, gdy np. jest za mało danych, a *„wiara”* daje nadzieję na ukończenie tychże badań z pozytywnym rezultatem.

Dobra znajomość układów automatyki jest podstawowym warunkiem zapewnienia prawidłowego sterowania nimi. Współczesny inżynier ma kilka możliwości, aby zapoznać się z obiektami. Metoda sprawdzona i zawsze dobra, to badania rzeczywistego obiektu. Jednak ma ona dużo wad. Po pierwsze konieczny jest fizyczny dostęp do obiektu, co nie zawsze jest możliwe. Po drugie przeprowadzenie pełnych, potrzebnych badań jest czasochłonne i niekiedy wręcz niemożliwe, np. w przypadku projektowania sterowania obiektem, który jest w użyciu (ograniczony dostęp). Praktycznie utrudnione i często bardzo kosztowne jest także przetestowanie zachowania obiektu w różnych, często skrajnych warunkach pracy. Niektóre badania obiektu mogą się okazać dla niego szkodliwe, albo wręcz niszczące.

Na przeciwległym końcu badań znajdują się symulacje komputerowe, które są stosunkowo szybkie i mało kosztowne. Jednak wymagają stworzenia matematycznego modelu obiektu, co nie zawsze możliwe jest metodami teoretycznymi i niekiedy wymaga, co prawda bardzo uproszczonych, ale jednak praktycznych badań rzeczywistego obiektu. Nawet najlepszy matematyczny model obiektu daleki jest od rzeczywistości i w teoretycznych symulacjach komputerowych niemożliwe jest przewidzenie większości sytuacji, które mogą wystąpić w pracy rzeczywistej.

Najbardziej wyszukana symulacja komputerowa nie zastąpi bezpośredniego, praktycznego zapoznania się z układem sterowania. Rozwiązaniem pośrednim, które łączy w sobie badania praktyczne i symulacje komputerowe jest badanie modelu fizycznego obiektu wykonanego na podstawie modelu matematycznego. Co prawda dalej jest to model obiektu, ale, ponieważ wykonany jest z rzeczywistych układów elektronicznych, ze wszystkim swoimi wadami (na szczęście), zatem bardziej zbliżony jest do obiektu rzeczywistego. Model fizyczny, podobnie jak obiekt rzeczywisty, podatny jest na zakłócenia oraz może pracować w środowisku rzeczywistym albo zbliżonym do rzeczywistego.

2. Eksperymentalne metody wyznaczania modeli matematycznych liniowych ciągłych rzeczywistych układów regulacji automatycznej

Celem analizy obiektu rzeczywistego jest poznanie jego właściwości statycznych i dynamicznych oraz wyznaczenie modelu matematycznego. Z matematycznego punktu widzenia, każdy obiekt można opisać równaniem różniczkowo-całkowym, które łatwo jest sprowadzić do równania tylko różniczkowego. Jednak dużo wygodniejszy w analizie jest opis obiektu tzw. metodami operatorowymi. W przypadku układów ciągłych, najczęściej wykorzystywanym narzędziem, jest przekształcenie Laplace'a [5], [6], które, bardzo upraszczając, zamienia równanie różniczkowe *n*-tego rzędu w wielomian *n*-tego rzędu. Oprócz większej prostoty rozwiązania np. równania kwadratowego w porównaniu z równaniem różniczkowym 2. stopnia, opis obiektu z wykorzystaniem transformaty Laplace'a jest czytelniejszy dla inżyniera, np. w przypadku obiektu opisanego równaniem różniczkowym 1. stopnia:

$$T\frac{dy(t)}{dt} + y(t) = u(t) \tag{2.1}$$

gdzie:

$u(t), y(t)$ – sygnał, odpowiednio wejściowy i wyjściowy obiektu (w dziedzinie czasu t),

k – wzmocnienie obiektu,

T – stała czasowa obiektu, określająca szybkość zmian odpowiedzi na zadany sygnał wejściowy.

Transmitancja operatorowa (tzw. funkcja przejścia) opisująca ten obiekt jest postaci:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{k}{Ts+1} \tag{2.2}$$

gdzie:

$U(s), Y(s)$ – sygnał, odpowiednio wejściowy i wyjściowy obiektu (w dziedzinie zespolonej s),

W rozdziale tym zostaną przedstawione dwie metody, czasowa i częstotliwościowa, wyznaczania modeli matematycznych najpopularniejszych obiektów regulacji.

2.1. Charakterystyki czasowe obiektów regulacji

Wyznaczenie charakterystyki czasowej obiektu regulacji jest najpopularniejszą i najprostszą metodą jego analizy. Pozwala również na ocenę jego parametrów stycznych, takich jak np. wartość ustalona lub czas ustalenia. W celu zbadania zachowania różnych obiektów w dziedzinie czasu stosuje się standardowe sygnały wejściowe (rys. 2.1), takie jak [5]:

- impuls Diraca, dany wzorem:

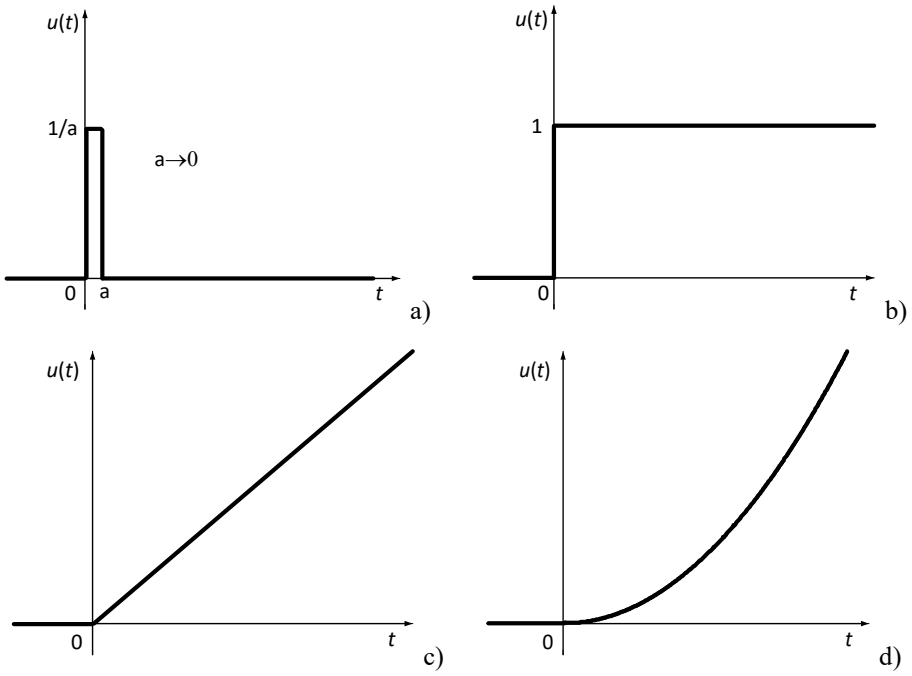
$$u(t) = \delta(t) = \begin{cases} 0 \text{ dla } t \neq 0 \\ +\infty \text{ dla } t = 0 \end{cases} \tag{2.3}$$

- skok jednostkowy (inne nazwy: skok położenia, funkcja Heaviside'a), dany wzorem:

$$u(t) = \mathbf{1}(t) = \begin{cases} 0 \text{ dla } t < 0 \\ 1 \text{ dla } t \geq 0 \end{cases} \tag{2.4}$$

- skok prędkości, dany wzorem:

$$u(t) = t\mathbf{1}(t) \tag{2.5}$$



Rys. 2.1. Standardowe sygnały wymuszające:
a) impuls Diraca, b) skok jednostkowy, c) skok prędkości, d) skok przyspieszenia

- skok przyspieszenia, dany wzorem:

$$u(t) = \frac{t^2}{2}\mathbf{1}(t) \tag{2.6}$$

Impuls Diraca to sygnał „czysto teoretyczny", jednak charakteryzuje się bardzo dużą zaletą praktyczną. W przypadku układów stabilnych odpowiedź każdego układu na impuls Diraca przy czasie $t \to \infty$, zdąża zawsze do wartości 0, niezależnie od parametrów układu. Jest to bardzo wygodne przy ocenie i porównaniu zachowania się różnych obiektów.

Z inżynierskiego punktu widzenia najbardziej użyteczny jest skok jednostkowy. Praktycznie stosowany jest zawsze, gdy np. nastawiana jest wartość temperatury w pomieszczeniu, zadawana prędkość obrotowa silnika, czy też wartość ciśnienia w komorze klimatycznej. To, jaki model obiektu regulacji przyjąć, zależy w znacznym stopniu właśnie od kształtu odpowiedzi układu na zadany skok jednostkowy.

W jaki sposób w praktyce wyznaczyć odpowiedź obiektu na skok jednostkowy? Nie jest to wcale trudne. Poniżej przedstawiono proste przykłady:

- Obiekt regulacji: pomieszczenie, dla którego projektowany jest regulator temperatury. Po załączeniu grzania (co odpowiada podaniu skoku jednostkowego) rejestrujemy temperaturę w funkcji czasu. Niezbędne są dwa przyrządy pomiarowe: zegarek i termometr. Ponieważ temperatura jest wartością fizyczną wolnozmienną, wystarczy zwykły zegarek, nawet bez pomiaru sekund. Klasa termometru zależy od dokładności, z jaką chcemy rejestrować temperaturę.
- Obiekt regulacji: komora klimatyczna, dla której projektowany jest regulator ciśnienia. Po zadaniu wartości ciśnienia (co odpowiada podaniu skoku jednostkowego) rejestrujemy ciśnienie w funkcji czasu. Także i w tym przypadku wystarczą dwa przyrządy pomiarowe: zegarek i ciśnieniomierz. Dobór tych przyrządów, analogicznie jak w przykładzie powyżej.

Analizując odpowiedź obiektu na skok jednostkowy, dobieramy model matematyczny o odpowiedzi skokowej najbardziej zbliżonej. Poniżej opisane zostaną modele najpopularniejszych obiektów regulacji, wraz z ich analizą i metodą wyznaczania podstawowych parametrów.

2.1.1. Obiekt inercyjny I rzędu

Obiekt inercyjny I rzędu [5] to jeden z popularniejszych obiektów regulacji. Jego transmitancja dana jest wzorem (2.2), a odpowiedź na skok jednostkowy (równanie (2.7)) widoczna jest na rys. 2.2.

$$y_1(t) = k\left[1 - e^{\left(-\frac{t}{T}\right)}\right]\mathbf{1}(t) \tag{2.7}$$

Parametr k wyznaczany jest wprost z rys. 2.2. Jeżeli jednak sygnał wejściowy jest postaci $u(t) = u\mathbf{1}(t)$, to oczywiście:

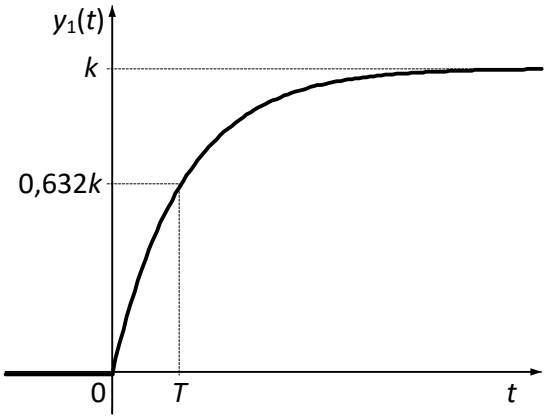
$$k = \frac{y_{ust}}{u} \tag{2.8}$$

gdzie:

u – wartość zadanego sygnału wejściowego,

y_{ust} – wartość ustalona sygnału wyjściowego.

Parametr T wyznaczany jest w sposób podany na rys. 2.2. Dla $t = T$ oraz $u(t) = u\mathbf{1}(t)$, odpowiedź układu, zgodnie ze wzorem (2.7) przyjmuje postać: $y(T) = k\left(1 - e^{-1}\right)u = 0,632u$. Zatem aby uzyskać wartość T, należy odczytać wartość czasu dla $y = 0,632\ y_{\text{ust}}$.



Rys. 2.2. Odpowiedź na skok jednostkowy obiektu inercyjnego I rzędu

2.1.2. Obiekt inercyjny I rzędu z opóźnieniem

Transmitancja obiektu inercyjnego I rzędu z opóźnieniem dana jest wzorem:

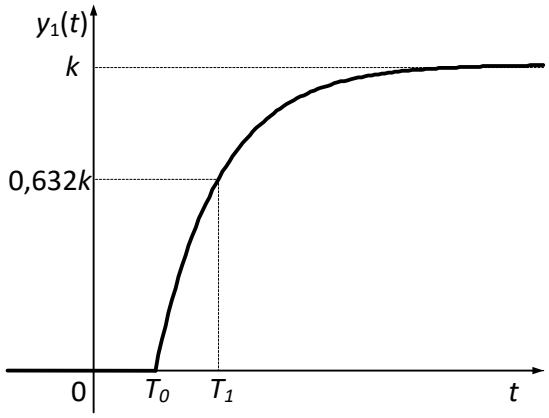
$$G(s) = \frac{k}{Ts + 1}e^{-sT_0} \tag{2.9}$$

gdzie T_0 – wartość opóźnienia.

Odpowiedź na skok jednostkowy takiego obiektu przedstawiona jest na rys. 2.3. Jak widać, praktycznie jest to połączenie obiektu inercyjnego I rzędu, danego wzorem (2.2), z elementem opóźniającym idealnym o wzorze:

$$G_{op}(s) = e^{-sT_0} \tag{2.10}$$

Parametry k oraz T_1 wyznaczane są identycznie jak to opisane zostało w podrozdziale 2.1.1, natomiast wartość opóźnienia T_0 można odczytać wprost z rys. 2.3. Stąd już bezpośrednio: $T = T_1 - T_0$.



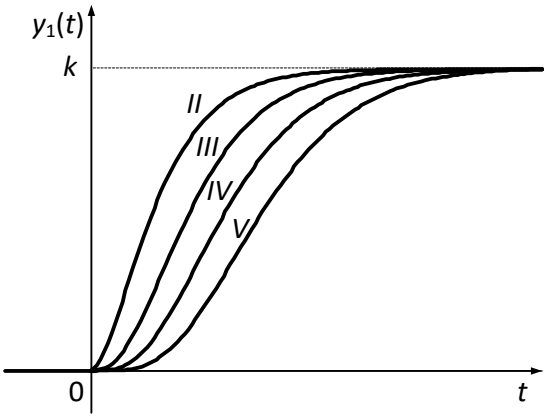
Rys. 2.3. Odpowiedź na skok jednostkowy obiektu inercyjnego I rzędu z opóźnieniem

2.1.3. Obiekt inercyjny II i wyższego rzędu

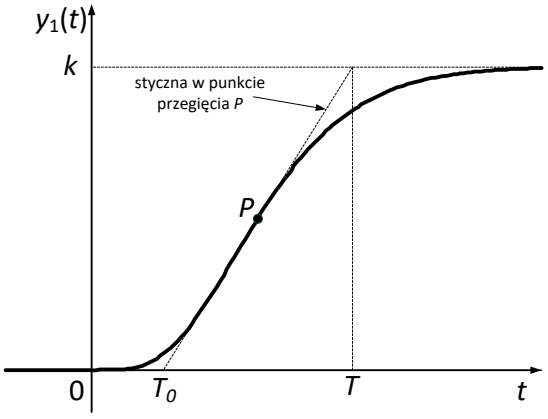
Odpowiedź na skok jednostkowy obiektów inercyjnych II [5] i wyższych rzędów, z pewnym przybliżeniem przypomina odpowiedź na skok jednostkowy obiektu inercyjnego I rzędu z opóźnieniem (rys. 2.4).

W przypadku, gdy nie jest wymagana duża dokładność, można taki obiekt aproksymować modelem obiektu inercyjnego I rzędu z opóźnieniem, czyli zgodnie ze wzorem (2.9) (tzw. model Küpfmüllera) [2]. Obniżenie rzędu inercyjności wymaga zatem wprowadzenia członu opóźniającego.

Parametr k wyznaczany jest identycznie jak to opisano w podrozdziale 2.1.1 (wzór (2.8)), natomiast sposób wyznaczania parametrów T_0 i T widoczny jest na rys. 2.5. Jak widać, wyznaczenie wartość czasów T_0 i T może być obarczone dużym błędem, bowiem nawet niewielkie odchylenie położenia punktu przegięcia P, powoduje zmianę konta nachylenia stycznej w tym punkcie, a co za tym idzie, często znaczną zmianę wartości odczytanych czasów T_0 i T.



Rys. 2.4. Odpowiedź na skok jednostkowy obiektów inercyjnych II i wyższych rzędów



Rys. 2.5. Parametry odpowiedzi na skok jednostkowy obiektów inercyjnych II i wyższych rzędów

Należy pamiętać, że im większy jest stosunek T_0/T, tym model obiektu ma gorsze własności regulacyjne.

Dużo mniejsze błędy można osiągnąć, stosując tzw. model Strejca [2]. Model obiektu inercyjnego II (i wyższego) rzędu dany jest wówczas wzorem:

$$G(s) = \frac{k}{(Ts+1)^n} \tag{2.11}$$

lub, jeżeli nie można znaleźć całkowitej wartości n:

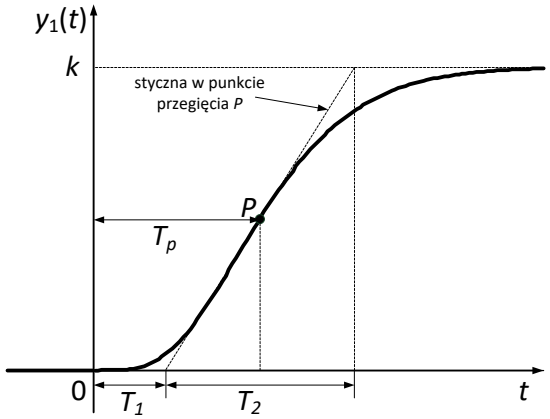
$$G(s) = \frac{k}{(Ts+1)^n} e^{-sT_0} \tag{2.12}$$

Parametr k wyznaczany jest identycznie jak to opisano w podrozdziale 2.1.1 (wzór (2.8)). Sposób wyznaczenia parametrów n i T (oraz T_0 w przypadku wzoru (2.12)) opisany jest poniżej.

W tabeli 2.1 zamieszczone są dane liczbowe klasy obiektów danych wzorem (2.11) poczynając od obiektów stopnia 2., na obiektach stopnia 5. kończąc. Na rys. 2.6 widoczne są parametry odpowiedzi skokowej układu danego wzorem (2.11), występujące w tabeli 2.1. Tabela zawiera dane tylko dla całkowitych wartości n. Oczywiście, nic nie stoi na przeszkodzie, aby dokonać numerycznych obliczeń T_1/T_2 dla n będącym liczbą rzeczywistą, jednak tabela może wtedy osiągnąć bardzo duże rozmiary, tym większe, im mniejszy krok zmian wartości parametru n.

Tabela 2.1. Dane liczbowe klasy obiektów danych wzorem (2.11) [2]

Rząd układu n	$a = T_1/T_2$	$b = Tp/T$
2	0,104	1
3	0,218	2
4	0,319	3
5	0,410	4

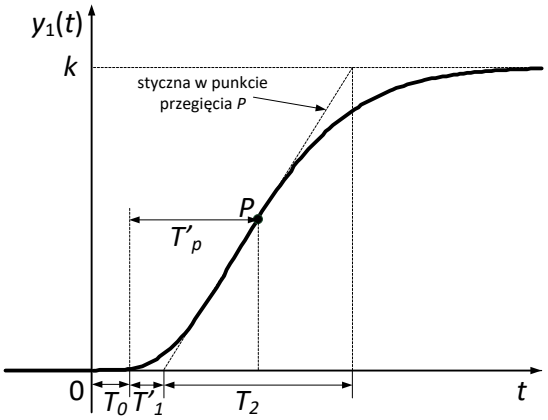


Rys. 2.6. Parametry odpowiedzi skokowej obiektu danego wzorem (2.11)

Sposób postępowania, w celu wyznaczenia parametrów n i T przedstawia się następująco (patrz rys. 2.6):

- po zarejestrowaniu odpowiedzi badanego obiektu na skok jednostkowy, wyznaczamy punkt przegięcia P,
- rysujemy styczną do krzywej odpowiedzi skokowej w punkcie przegięcia,
- odczytujemy wartości czasów T_1 i T_2,

- dalej, w zależności od stosunku $a = T_1/T_2$, postępujemy zgodnie z jednym z poniższych algorytmów:
  1. wartość *a* bliska jest wartości z drugiej kolumny tabeli 2.1:
     - przyjmujemy model obiektu dany wzorem (2.11),
     - dla odpowiedniej wartości *a*, odczytujemy wartość *n* z pierwszej, a wartość *b* z trzeciej kolumny,
     - z wykresu (rys. 2.6) odczytujemy wartość T_p oraz obliczamy wartość *T* zgodnie ze wzorem: $T = T_p/b$,
  2. wartość *a* znacznie odbiega od wartości z drugiej kolumny tabeli 2.1 (znajduje się pomiędzy tymi wartościami):
     - przyjmujemy model obiektu dany wzorem (2.12),
     - ponieważ nie jest możliwe wyznaczenie całkowitej wartości *n*, zatem z tabeli odczytujemy mniejszą wartość *n* (przyjmujemy mniejszy rząd obiektu),
     - dla przyjętej wartości *n* odczytujemy z tabeli nową wartość *a* oraz wartość zmiennej *b*,
     - obliczamy nową wartość T_1' zgodnie ze wzorem: $T_1' = aT_2$,
     - obliczamy wartość opóźnienia T_0 , zgodnie ze wzorem: $T_0 = T_1 - T_1'$ (podobnie jak na rys. 2.5),
     - z wykresu (rys. 2.7) odczytujemy wartość T_p' oraz obliczamy wartość *T* zgodnie ze wzorem: $T = T_p'/b$.



Rys. 2.7. Parametry odpowiedzi skokowej obiektu danego wzorem (2.12)

2.1.4. Obiekt oscylacyjny II i wyższego rzędu

Odpowiedź na skok jednostkowy obiektów oscylacyjnych II [5] i wyższych rzędów wygląda w zasadzie bardzo podobnie (rys. 2.8), zatem, gdy nie jest wymagana duża dokładność modelu, można każdy obiekt oscylacyjny aproksymować modelem obiektu II rzędu.

Transmitancja modelu oscylacyjnego II rzędu dana jest wzorem [8]:

$$G(s) = \frac{k}{T^2 s^2 + 2nTs + 1} \tag{2.13}$$

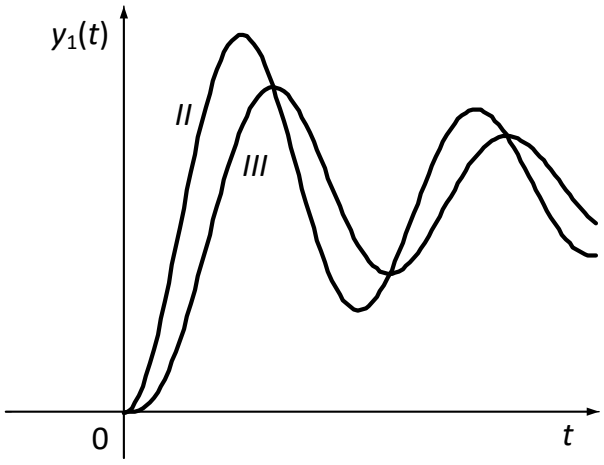
gdzie:

k – wzmocnienie obiektu,

T – stała czasowa obiektu, określająca szybkość zmian odpowiedzi na zadany sygnał wejściowy,

n – współczynnik tłumienia.

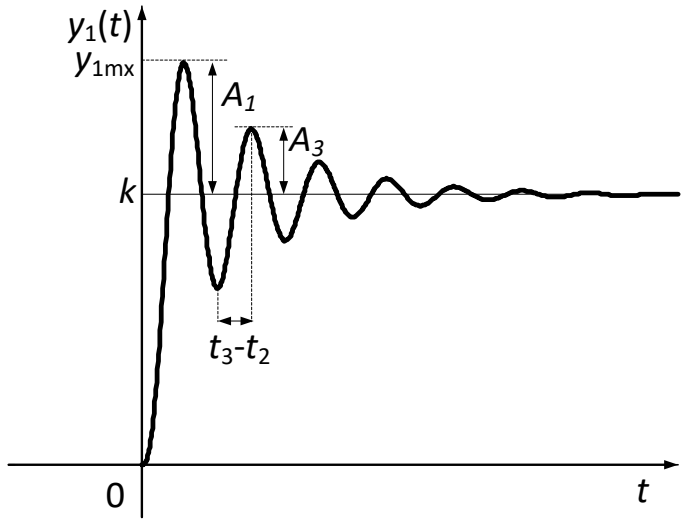
Odpowiedź obiektu na skok jednostkowy ma charakter oscylacyjny, gdy n przyjmuje wartości z przedziału: $0 < n < 1$. W przypadku gdy wartość współczynnika tłumienia przyjmuje wartości $n \geq 1$, odpowiedź układu ma charakter inercyjny (patrz podrozdział 2.1.3).



Rys. 2.8. Odpowiedź na skok jednostkowy obiektów oscylacyjnych II i III rzędu

Odpowiedź takiego układu na skok jednostkowy [8] dana jest wzorem:

$$y_1(t) = k\left[1 - \frac{e^{\left(-\frac{n}{T}t\right)}}{\sqrt{1-n^2}}\sin\left(\frac{\sqrt{1-n^2}}{T}t + \arccos(n)\right)\right]\mathbf{1}(t) \tag{2.14}$$



Rys. 2.9. Parametry odpowiedzi na skok jednostkowy obiektu oscylacyjnego II rzędu [8]

Jak widać na rys. 2.9, parametr k odczytuje się identycznie jak w przypadku uprzednio opisanych obiektów.

Parametry n i T wyznacza się ze wzorów [8]:

$$n = \frac{\ln\left(\frac{A_1}{A_3}\right)}{\sqrt{4\pi^2 + \left(\ln\left(\frac{A_1}{A_3}\right)\right)^2}} \tag{2.15}$$

$$T = \frac{\sqrt{1-n^2}\,(t_3 - t_2)}{\pi} \tag{2.16}$$

gdzie wartości A_1, A_3, t_2, t_3 odczytywane są wprost z charakterystyki na rys. 2.9. Jak można zauważyć, indeksy przy tych parametrach odpowiadają numerom kolejnych wierzchołów odpowiedzi na skok jednostkowy.

2.2. Charakterystyki częstotliwościowe obiektów regulacji

Wyznaczenie charakterystyki częstotliwościowej obiektu regulacji jest kolejną metodą jego analizy. Niestety nie jest to metoda tak prosta, jak w przypadku charakterystyki czasowej i w przypadku zdecydowanej większości obiektów rzeczywistych niemożliwa do praktycznej realizacji. Wymaga bowiem podania na wejście obiektu sygnału sinusoidalnego, obserwacji sygnału na wyjściu

obiektu i badaniu wzmocnienia układu oraz przesunięcia fazowego pomiędzy sygnałem wyjściowym i wejściowym. Zastosowanie tej metody ogranicza się do niewielkiej grupy obiektów, na wejście których można podać sygnał napięciowy sinusoidalny (np. z generatora funkcyjnego) i na wyjściu którego występuje również sygnał napięciowy. Dobrym przykładem jest tu rzeczywisty wzmacniacz napięciowy, który jest obiektem inercyjnym I rzędu.

Poniżej, przykładowo, zostanie zaprezentowane wyznaczanie charakterystyki częstotliwościowej (tzw. charakterystyki Nyquista) tylko układu inercyjnego I rzędu oraz sposób wyznaczania jego parametrów poprzez analizę tej charakterystyki.

2.2.1. Obiekt inercyjny I rzędu

Transmitancja operatorowa obiektu I rzędu w postaci częstotliwościowej (widmowej) powstaje poprzez podstawienie wzoru (2.17) do równania (2.2).

$$s = j\omega \tag{2.17}$$

gdzie:

j – jednostka urojona, $j = \sqrt{-1}$,

ω – pulsacja układu, $\omega = 2\pi f$, gdzie f – częstotliwość.

Otrzymuje się wówczas równanie widmowe postaci:

$$G(j\omega) = \frac{k}{j\omega T + 1} \tag{2.18}$$

Z równania (2.18) można wydzielić część rzeczywistą i urojoną:

$$G(j\omega) = \mathrm{Re}\{G(j\omega)\} + j\,\mathrm{Im}\{G(j\omega)\} \tag{2.19}$$

gdzie część rzeczywista i urojona dane są odpowiednio wzorami:

$$\mathrm{Re}\{G(j\omega)\} = \frac{k}{1 + (\omega T)^2} \tag{2.20}$$

$$\mathrm{Im}\{G(j\omega)\} = \frac{-k\omega T}{1 + (\omega T)^2} \tag{2.21}$$

Równanie (2.18) można przedstawić także w innej postaci:

$$G(j\omega) = \left|G(j\omega)\right| e^{-j\arg\{G(j\omega)\}} \tag{2.22}$$

gdzie:

$\left|G(j\omega)\right|$ – moduł transmitancji (funkcji przejścia) dany wzorem:

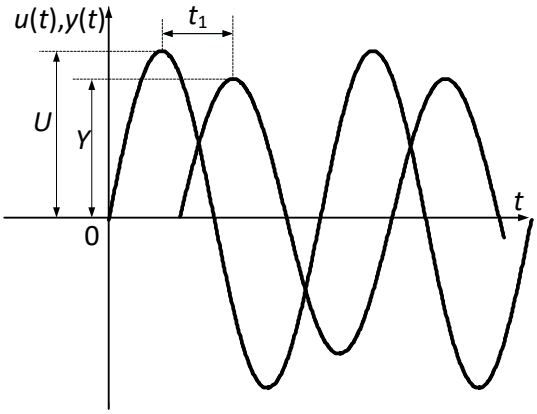
$$G(j\omega)=\sqrt{\mathrm{Re}^2\left\{G(j\omega)\right\}+\mathrm{Im}^2\left\{G(j\omega)\right\}}=\frac{k}{\sqrt{1+(\omega T)^2}} \tag{2.23}$$

$\arg\left\{G(j\omega)\right\}$ – argument transmitancji dany wzorem:

$$\arg\left\{G(j\omega)\right\}=\mathrm{arctg}\frac{\mathrm{Im}\left\{G(j\omega)\right\}}{\mathrm{Re}\left\{G(j\omega)\right\}}=-\mathrm{arctg}(\omega T) \tag{2.24}$$

Algorytm wyznaczania charakterystyki częstotliwościowej obiektu inercyjnego I rzędu jest następujący [8]:

- na wejście obiektu rzeczywistego podajemy (np. z generatora funkcyjnego) sygnał napięciowy sinusoidalny, o stałej amplitudzie i zmiennej częstotliwości,
- dla zadanej częstotliwości, mierzymy amplitudę sygnału wyjściowego oraz przesunięcie fazowe pomiędzy sygnałem wyjściowym i wejściowym; sposób odczytu parametrów widoczny jest na rys. 2.10 [8],



Rys. 2.10. Sposób pomiaru amplitudy sygnałów
oraz wzajemnego przesunięcia fazowego między nimi

- obliczamy moduł transmitancji operatorowej (wzmocnienie układu) zgodnie ze wzorem:

$$\left|G(j\omega)\right|=\frac{Y}{U} \tag{2.25}$$

gdzie:

U – amplituda sygnału wejściowego,
Y – amplituda sygnału wyjściowego.

- obliczamy argument transmitancji operatorowej (fazę układu) zgodnie ze wzorem:

$$\arg\left\{G(j\omega)\right\} = -t_1 f 360° \tag{2.26}$$

gdzie:

t_1 – przesunięcie fazowe pomiędzy sygnałami wyjściowym i wejściowym,

f – częstotliwość sygnału wejściowego.

- powtarzamy wyznaczanie modułu oraz argumentu transmitancji dla różnych częstotliwości tak, aby w miarę równomiernie pokryć zmianę argumentu w przedziale (0…90°); liczba pomiarów zależna jest od dokładności, z jaką chcemy wyznaczyć charakterystykę częstotliwościową; uwaga: oczywiście wyznaczenie $\arg\left\{G(j\omega)\right\} = 0$ oraz $\arg\left\{G(j\omega)\right\} = 90°$ jest z praktycznego punktu widzenia niemożliwe, wymagałoby bowiem podania sygnału wejściowego o częstotliwościach odpowiednio 0 i ∞ Hz.
- wykreślamy charakterystykę Nyquista na płaszczyźnie zespolonej jedną z dwóch metod (rys. 2.11):
  - wprost, zaznaczając na płaszczyźnie zespolonej dla każdej częstotliwości wektor o długości równej $|G(j\omega)|$ przesunięty o kąt równy $\arg\left\{G(j\omega)\right\}$; wierzchołki tychże wektorów utworzą charakterystykę Nyquista,
  - obliczając:

$$\mathrm{Im}\left\{G(j\omega)\right\} = \left|G(j\omega)\right| \sin\left(\arg\left\{G(j\omega)\right\}\right) \tag{2.27}$$

$$\mathrm{Re}\left\{G(j\omega)\right\} = \left|G(j\omega)\right| \cos\left(\arg\left\{G(j\omega)\right\}\right) \tag{2.28}$$

Obliczone dla różnych częstotliwości punkty $\left(\mathrm{Im}\left\{G(j\omega)\right\}, \mathrm{Re}\left\{G(j\omega)\right\}\right)$ tworzą charakterystykę.

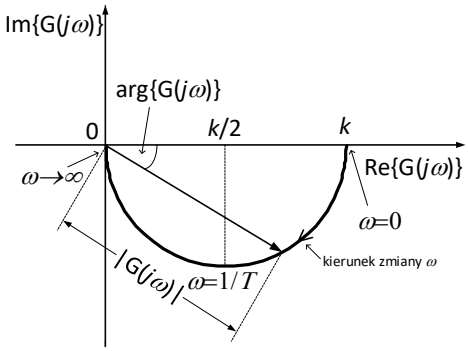
Wzmocnienie k, wyznacza się wprost z charakterystyki, po przedłużeniu jej w kierunku malejących wartości ω do 0 włącznie. Zatem praktycznie jest to wartość $\mathrm{Re}\left\{G(j\omega)\right\}$ dla $\omega = 0$ (rys. 2.11).

Wartość stałej czasowej T liczy się przekształcając wzór (2.24) do wzoru:

$$T = \frac{1}{\omega} \mathrm{tg}\left(\arg\left\{G(j\omega)\right\}\right) \tag{2.29}$$

Teoretycznie, wartość T można wyznaczyć dla dowolnego pomiaru i powinna ona być w każdym przypadku taka sama. Jednak w praktyce jest to mało realne, zatem zaleca się wyznaczenie wartości T dla pomiaru, którego punkt

$\left(\mathrm{Im}\left\{G\left(j\omega\right)\right\},\mathrm{Re}\left\{G\left(j\omega\right)\right\}\right)$ najmniej odbiega od charakterystyki (leży na tej charakterystyce).



Rys. 2.11. Charakterystyka Nyquista obiektu I rzędu

W przypadku, gdy w wyniku pomiaru obliczona wartość $\arg\left\{G\left(j\omega\right)\right\}=45°$ wzór (2.29) upraszcza się do postaci:

$$T=\frac{1}{\omega} \tag{2.30}$$

3. Dyskretne modele rzeczywistych układów regulacji automatycznej

W dobie rozwoju techniki cyfrowej najprostszym sposobem wykonania fizycznego modelu obiektu jest wykorzystanie do tego celu sterownika mikroprocesorowego. Oczywiście taki model będzie miał charakter układu dyskretnego, jednak przy odpowiednio dużych częstotliwościach próbkowania model dyskretny może być zbliżony do modelu ciągłego. Biorąc pod uwagę fakt, że w dzisiejszych czasach sterowanie cyfrowe praktycznie wyparło sterowanie analogowe, dyskretny model fizyczny obiektu staje się bardziej adekwatny, a błędy które wprowadza, w porównaniu z modelem ciągłym, są proporcjonalnie niezbyt duże i stosunkowo łatwe do wyeliminowania poprzez stosowanie algorytmów korekcyjnych [3], [5]–[7].

3.1. Przejście z układu ciągłego do układu dyskretnego

Przejście z transmitancji układu ciągłego na transmitancję układu dyskretnego to problem czysto obliczeniowy. W zależności od tego czy znane są bieguny transmitancji czy też nie, można zastosować metodę residuów albo metodę przekształcenia biliniowego. Poniżej zostaną opisane obydwie te metody.

3.1.1. Metoda residuów

Metoda residuów [5] możliwa jest do zastosowania tylko w przypadku gdy znane są bieguny transmitancji układu ciągłego. Jeżeli bieguny nie są określone w sposób jawny, można ją stosować dla obiektów I i II rzędu, gdyż wyznaczenie biegunów to sprawa banalnie prosta. W przypadku obiektów rzędu III i wyższego wyznaczenie biegunów to problem numeryczny, nierzadko skomplikowany. Znaczącym ułatwieniem może być zastosowanie pakietów matematycznych, takich jak np. Matlab.

Pomijając całą teorie związaną z transformatą Laplace'a i przekształceniem **Z** [5], [7], z matematycznego punktu widzenia wyznaczenie transmitancji dyskretnej na podstawie znajomości transmitancji ciągłej polega na podstawieniu do wzoru:

$$G_{OE}(z) = \frac{z-1}{z}\mathbf{Z}\left\{\frac{G_O(s)}{s}\right\} \tag{3.1}$$

gdzie:

G_{OE} – transmitancja dyskretna obiektu z ekstrapolatorem zerowego rzędu (w praktyce w zasadzie nie stosuje się ekstrapolatorów wyższych rzędów),

$\mathbf{Z}\{\cdot\}$ – przekształcenie **Z** wyrażenia w nawiasie,

G_O – transmitancja ciągła obiektu.

W ogólnej postaci wyznaczanie transmitancji dyskretnej $G(z)$ układu o transmitancji ciągłej $G(s)$ dane jest wzorem:

$$G(z) = \sum_{k=1}^{m} \left[\left(s - s_k\right) G(s) \frac{z}{z - e^{sT_p}} \bigg|_{s=s_k} \right] \tag{3.2}$$

gdzie:

s_k – k-ty biegunem transmitancji ciągłej,

m – liczba biegunów transmitancji ciągłej,

T_p – okres próbkowania.

Wzór (3.2) jest prawdziwy tylko dla biegunów pojedynczych. Jeżeli którykolwiek biegun jest p-krotny ($p > 1$), to we wzorze (3.2), pod sumą, dla takiego bieguna wystąpi składnik:

$$\frac{1}{(p-1)!} \left\{ \frac{d^{p-1}}{ds^{p-1}} \left[\left(s - s_k\right)^p G(s) \frac{z}{z - e^{sT_p}} \right] \right\} \bigg|_{s=s_k} \tag{3.3}$$

gdzie:

p – krotność bieguna,

$\frac{d^{p-1}}{ds^{p-1}}(.)$ – pochodna rzędu $(p-1)$ wyrażenia w nawiasie.

Obiekt regulacji dany wzorem (3.4) jest jednym z częściej występujących w praktyce.

$$G_O(s) = \frac{k}{T^2 s^2 + 2nTs + 1} \tag{3.4}$$

gdzie:

k – wzmocnienie układu,

T – stała czasowa układu,

N – współczynnik tłumienia, dla $0 < n < 1$ odpowiedź układu na skok jednostkowy ma charakter oscylacyjny, a dla $n \geq 1$ – inercyjny.

Poniżej algorytm postępowania prowadzący do przejścia z transmitancji układu ciągłego danego wzorem (3.4) na transmitancję układu dyskretnego, metodą residuów:

- zakładamy czas próbkowania T_p,
- obliczamy bieguny transmitancji (pierwiastki mianownika):

$$s_{1,2} = \frac{-n \pm \sqrt{n^2 - 1}}{T},$$

zatem wzór (3.4) przyjmie postać:

$$G_O(s) = \frac{k}{T^2} \frac{1}{(s - s_1)(s - s_2)}$$

- podstawiamy $G_o(s)$ wprost do wzoru (3.1):

$$G_{OE}(z) = \frac{z-1}{z} \mathbf{Z}\left\{ \frac{G_O(s)}{s} \right\} = \frac{z-1}{z} \frac{k}{T^2} \mathbf{Z}\left\{ \frac{1}{s(s - s_1)(s - s_2)} \right\}$$

- występują 3 pojedyncze bieguny $s = 0$, $s = s_1$ oraz $s = s_2$, zatem w celu obliczenia przekształcenia $\mathbf{Z}\{\cdot\}$, można zastosować wzór (3.2):

$$\mathbf{Z}\left\{ \frac{1}{s(s - s_1)(s - s_2)} \right\} = \sum_{k=1}^{3} \left[(s - s_k) G(s) \frac{z}{z - e^{sT_p}} \bigg|_{s=s_k} \right] =$$

$$= (s-0) \frac{1}{s(s - s_1)(s - s_2)} \frac{z}{z - e^{sT_p}} \bigg|_{s=0} + (s - s_1) \frac{1}{s(s - s_1)(s - s_2)} \frac{z}{z - e^{sT_p}} \bigg|_{s=s_1} +$$

$$+ (s - s_2) \frac{1}{s(s - s_1)(s - s_2)} \frac{z}{z - e^{sT_p}} \bigg|_{s=s_2} =$$

$$= \frac{1}{s_1 s_2} \frac{z}{z-1} +,, \frac{z}{z-}, + \frac{1}{s_2(s_2 - s_1)} \frac{z}{z - e^{s_2 T_p}}$$

lub zapisując w prostszy sposób:

$$\mathbf{Z}\left\{ \frac{1}{s(s - s_1)(s - s_2)} \right\} = z\left(\frac{a}{z-1} + \frac{b}{z-c} + \frac{d}{z-e} \right)$$

gdzie:

$$a = \frac{1}{s_1 s_2}, \quad b = \frac{1}{s_1(s_1 - s_2)}, \quad c = e^{s_1 T_p}, \quad d = \frac{1}{s_2(s_2 - s_1)}, \quad e = e^{s_2 T_p},$$

- podstawiając powyższe równanie do wzoru (3.1), po sprowadzeniu do wspólnego mianownika i uproszczeniu ostatecznie otrzymamy:

$$G_{OE}(z) = \frac{k}{T^2} \frac{Az^2 + Bz + C}{z^2 + Dz + E} \tag{3.5}$$

gdzie:

$A = a + b + d$,

$B = -ae - ac - be - b - dc - d$,
$C = ace + be + dc$,
$D = -e - c$,
$E = ce$.

Wzór (3.5) można wprost wykorzystać do przejścia z transmitancji układu ciągłego danego ogólnym wzorem (3.4) (o biegunach pojedynczych) na transmitancję układu dyskretnego, metodą residuów. Jest to szczególnie wygodne w przypadku modelowania układu o zmiennych parametrach.

3.1.2. Przekształcenie biliniowe

W przypadku obiektów rzędu III i wyższego wyznaczenie biegunów to nierzadko skomplikowany problem numeryczny. W sytuacji, gdy nie ma możliwości skorzystania z pakietu matematycznego, takiego jak np. Matlab oraz, gdy nie jest wymagana duża dokładność konwersji można wykorzystać tzw. przekształcenie biliniowe [5], [7].

Przechodząc od przekształcenia Laplace'a do przekształcenia **Z**, dokonywane jest podstawienie:

$$z = e^{sT_p} \tag{3.6}$$

gdzie T_p – okres próbkowania.

Przekształcając wzór (3.6), otrzymuje się:

$$s = \frac{1}{T_p}\ln(z) \tag{3.7}$$

Biorąc pod uwagę tylko pierwszy wyraz rozwinięcia funkcji ln(z) w szereg Taylora, wzór (3.7) ulegnie znacznemu uproszczeniu:

$$s = \frac{2}{T_p}\frac{z-1}{z+1} \tag{3.8}$$

Powyższe podstawienie nosi nazwę przekształcenia biliniowego.

Poniżej algorytm postępowania prowadzący do przejścia z transmitancji układu ciągłego na transmitancję układu dyskretnego, z wykorzystaniem przekształcenia biliniowego, na przykładzie najczęściej występującego w praktyce układu o transmitancji ciągłej danej wzorem (3.4):

- podstawiamy $G_{OE}(s)$ wprost do wzoru (3.1):

$$G_{OE}(z) = \frac{z-1}{z}\mathbf{Z}\left\{\frac{G_O(s)}{s}\right\} = \frac{z-1}{z}\mathbf{Z}\left\{\frac{k}{s\left(T^2s^2 + 2nTs + 1\right)}\right\}$$

- obliczamy transformatę $\mathbf{Z}\{\cdot\}$ podstawiając w miejsce operatora s przekształcenie biliniowe (3.8)

$$\mathbf{Z}\left\{\frac{k}{s\left(T^2s^2+2nTs+1\right)}\right\}\Bigg|_{s=\frac{2}{T_p}\frac{z-1}{z+1}}=\frac{k}{\frac{2}{T_p}\frac{z-1}{z+1}\left[T^2\left(\frac{2}{T_p}\frac{z-1}{z+1}\right)^2+2nT\frac{2}{T_p}\frac{z-1}{z+1}+1\right]}$$

- wstawiając powyższe równanie do wzoru (3.1), po przekształceniach i uproszczeniu ostatecznie otrzymamy:

$$G_{OE}(z)=\frac{a\left(z+1\right)^3}{bz^3+cz^2+dz} \tag{3.9}$$

gdzie:

$$a=\frac{kT_p^3}{2},$$

$$b=4T^2+4nTT_p+T_p^2,$$

$$c=-8T^2+2T_p^2,$$

$$d=4T^2-4nTT_p+T_p^2.$$

Wzór (3.9) można wprost wykorzystać do przejścia z transmitancji układu ciągłego danego ogólnym wzorem (3.4) na transmitancję układu dyskretnego, wykorzystując przekształcenie biliniowe. Jest to szczególnie wygodne w przypadku modelowania układu o zmiennych parametrach.

3.1.3. Porównanie metody residuów z przekształceniem biliniowym

Porównując dwa sposoby przejścia z układu ciągłego do dyskretnego: z wykorzystaniem metody residuów, wzór (3.5) oraz z wykorzystaniem przekształcenia biliniowego, wzór (3.9), można mylnie stwierdzić, że pod względem obliczeniowym, lepsza jest ta druga, bo daje prostszy wzór końcowy (co prawda licznik i mianownik wyższego rzędu, ale mniej dodatkowych współczynników). Tak przynajmniej wygląda na wzorach ogólnych. Jednak w praktyce tak nie jest. Zastosowanie przekształcenia biliniowego, w przypadku układów wyższego rzędu to dużo żmudnych obliczeń i przekształceń. Dodatkowo jest metodą uproszczoną, bo bierze pod uwagę tylko pierwszy wyraz rozwinięcia funkcji $ln(z)$ w szereg Taylora. Metoda ta jest stosowana, gdy niejawne (trudne, bądź niemożliwe do wyznaczenia) są bieguny transmitancji układu. W metodzie

residuów kolejny biegun to „tylko" kolejny składnik sumy wzoru (3.2) i prostsze przekształcenia.

Poniżej zostanie zaprezentowanie porównane odpowiedzi na skok jednostkowy układu ciągłego oraz wyznaczonych dwoma metodami (residuów i przekształcenie biliniowe) odpowiadających mu układów dyskretnych. W celu porównania zostaną wyznaczone transmitancje układu dyskretnego z wykorzystaniem dwóch wzorów (3.5) oraz (3.9).

Badany przykładowy obiekt opisany jest transmitancją ciągłą daną wzorem:

$$G_O(s) = \frac{1}{s^2 + 3s + 2} \tag{3.10}$$

Zatem parametry obiektu są następujące:

– wzmocnienie $k = 0{,}5$,

– stała czasowa $T = 0{,}7071$,

– współczynnik tłumienia $n = 1{,}0607$.

Transmitancja ma dwa bieguny: $s_1 = -1$ oraz $s_2 = -2$.

W celu przekształcenia na układ dyskretny zakładamy czas próbkowania $T_p = 0{,}1$ s. Przy doborze częstotliwości próbkowania należy wziąć pod uwagę twierdzenie o próbkowaniu (tw. Shannona), które mówi, że częstotliwość próbkowania $f_p = 1/T_p$ musi być większa niż dwukrotność najwyższej składowej częstotliwości w mierzonym sygnale [5]–[7].

W pierwszym kroku obliczona zostanie transmitancja układu dyskretnego metodą residuów zgodnie ze wzorem (3.5). Na początek obliczone zostaną współczynniki a, b, c, d, e oraz A, B, C, D, E występujące w tym wzorze:

$$a = \frac{1}{s_1 s_2} = 0{,}5,$$

$$b = \frac{1}{s_1\left(s_1 - s_2\right)} = -1,$$

$$c = e^{s_1 T_p} = 0{,}9048,$$

$$d = \frac{1}{s_2\left(s_2 - s_1\right)} = 0{,}5,$$

$$e = e^{s_2 T_p} = 0{,}8187,$$

$$A = a + b + d = 0,$$

$$B = -ae - ac - be - b - dc - d = 0{,}0045,$$

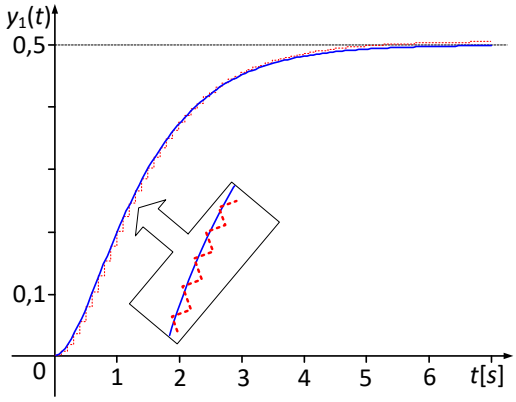
$$C = ace + be + dc = 0{,}0041,$$

$$D = -e - c = -1{,}7236,$$

$$E = ce = 0{,}7408.$$

Równanie (3.5) będzie ostatecznie postaci:

$$G_{OE1}(z) = \frac{0,0045z + 0,0041}{z^2 - 1,724z + 0,741} \tag{3.11}$$



Rys. 3.1. Porównanie odpowiedzi na skok jednostkowy układu ciągłego (linia ciągła) i jego odpowiednika dyskretnego, obliczanego metodą residuów (linia przerywana)

Na rysunku 3.1 widoczne jest porównanie odpowiedzi na skok jednostkowy układu o transmitancji ciągłej danej wzorem (3.10) oraz jego odpowiednika dyskretnego o wzorze (3.11). Jak widać obydwie charakterystyki są zbliżone do siebie kształtem. Charakterystyka układu dyskretnego ma widoczne tzw. schodki, które będą tym mniejsze, im krótszy czas próbkowania.

W drugim kroku obliczona zostanie transmitancja układu dyskretnego z wykorzystaniem przekształcenia biliniowego, zgodnie ze wzorem (3.9).

Na początek obliczone zostaną współczynniki a, b, c, d:

$$a = \frac{kT_p^3}{2} = 0,00025,$$

$$b = 4T^2 + 4nTT_p + T_p^2 = 2,31,$$

$$c = -8T^2 + 2T_p^2 = -3,98,$$

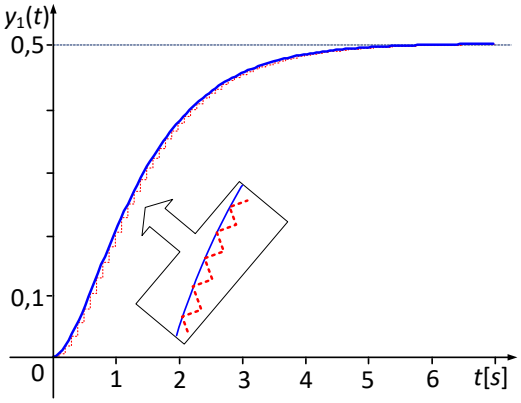
$$d = 4T^2 - 4nTT_p + T_p^2 = 1,71.$$

Równanie (3.9) będzie ostatecznie postaci:

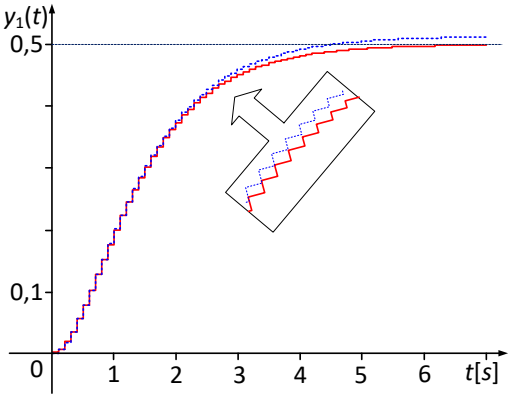
$$G_{OE2}(z) = \frac{0,00025(z+1)^3}{2,31z^3 - 3,98z^2 + 1,71z} \tag{3.12}$$

Na rysunku 3.2 widoczne jest porównanie odpowiedzi na skok jednostkowy układu o transmitancji ciągłej danej wzorem (3.10) oraz jego odpowiednika

dyskretnego o wzorze (3.12). Jak widać obydwie charakterystyki są także zbliżone do siebie kształtem. Charakterystyka układu dyskretnego ma widoczne tzw. schodki które będą tym mniejsze, im krótszy czas próbkowania. Rysunek 3.2 praktycznie niewiele różni się od rys. 3.1, zatem przekształcenie biliniowe wprowadzając pewne ograniczenia (tylko pierwszy wyraz rozwinięcia w szereg Taylora funkcji logarytm naturalny), nie wpływa znacząco na błąd konwersji.



Rys. 3.2. Porównanie odpowiedzi na skok jednostkowy układu ciągłego (linia ciągła) i jego odpowiednika dyskretnego, obliczanego z wykorzystaniem przekształcenia biliniowego (linia przerywana)



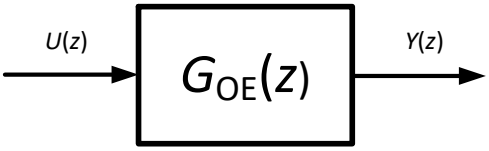
Rys. 3.3. Porównanie odpowiedzi na skok jednostkowy układu dyskretnego obliczanego metodą residuów (linia ciągła) oraz z wykorzystaniem przekształcenia biliniowego (linia przerywana)

Na rysunku 3.3 widoczne jest porównanie odpowiedzi na skok jednostkowy układu dyskretnego obliczonego metodą residuów oraz z wykorzystaniem

przekształcenia biliniowego. Różnice widoczne są dopiero przy powiększeniu charakterystyk. Na rysunkach 3.1, 3.2 i 3.3 zaprezentowano charakterystyki dla konkretnego przykładu. Oczywiście dla innych układów, szczególnie wyższych rzędów, różnice te mogą być większe.

3.2. Przejście z transmitancji układu dyskretnego do równania różnicowego

Numeryczna realizacja modelu obiektu dyskretnego nie jest możliwa wprost na podstawie transmitancji dyskretnej tego obiektu. Konieczne jest jeszcze przejście z układu dyskretnego do równania różnicowego [5]–[7]. Jednak jest to operacja bardzo prosta.



Rys. 3.4. Dyskretny model obiektu z sygnałami wejściowymi i wyjściowymi

Jak widać na rys. 3.4, model obiektu można zapisać inaczej jako iloraz transformaty **Z** sygnałów wyjściowego oraz wejściowego:

$$G_{OE}(z) = \frac{Y(z)}{U(z)} \tag{3.13}$$

Lewą stronę równania (3.13) można zapisać w innej postaci:

$$\frac{b_p z^p + b_{p-1} z^{p-1} + \ldots + b_1 z + b_0}{a_q z^q + a_{q-1} z^{q-1} + \ldots + a_1 z + a_0} = \frac{Y(z)}{U(z)} \tag{3.14}$$

i dalej mnożąc mianownik i licznik przez z^{-q}:

$$\frac{b_p z^{p-q} + b_{p-1} z^{p-q-1} + \ldots + b_1 z^{-q+1} + b_0 z^{-q}}{a_q + a_{q-1} z^{-1} + \ldots + a_1 z^{-q+1} + a_0 z^{-q}} = \frac{Y(z)}{U(z)} \tag{3.15}$$

Po przekształceniu otrzymamy:

$$\begin{aligned} &\left(b_p z^{p-q} + b_{p-1} z^{p-q-1} + \ldots + b_1 z^{-q+1} + b_0 z^{-q}\right) U(z) = \\ &= \left(a_q + a_{q-1} z^{-1} + \ldots + a_1 z^{-q+1} + a_0 z^{-q}\right) Y(z) \end{aligned} \tag{3.16}$$

W celu wyznaczenia równania różnicowego można skorzystać z twierdzenia o opóźnieniu w dziedzinie czasu:

$$\mathbf{Z}\left\{f\left(n-m\right)\right\} = z^{-m}F(z) + z^{-m}\sum_{k=-m}^{k=-1}\left(f(k)z^{-k}\right) \tag{3.17}$$

Zakładając zerowe warunki początkowe, wzór (3.17) przyjmie uproszczoną postać:

$$\mathbf{Z}\left\{f\left(n-m\right)\right\} = z^{-m}F(z) \tag{3.18}$$

Poddając równanie (3.16), działaniu twierdzenia o opóźnieniu w dziedzinie czasu, z założonymi zerowymi warunkami początkowymi (wzór (3.18)), po przekształceniach otrzymamy wprost równanie różnicowe w postaci ogólnej:

$$\begin{aligned} y(n) = \frac{1}{a_q}\Big[& b_p u\left(n+p-q\right) + b_{p-1}u\left(n+p-q-1\right) + \\ & +\ldots + b_1 u\left(n-q+1\right) + b_0 u\left(n-q\right) + \\ & -a_{q-1}y(n-1) - \ldots - a_1 y(n-q+1) - a_0 y(n-q)\Big] \end{aligned} \tag{3.19}$$

W celu lepszego, praktycznego zobrazowania, wykonajmy operacje jak powyżej na przykładzie transmitancji ciągłej II rzędu danej ogólnym wzorem (3.4). W pierwszej kolejności rozpatrzymy transmitancję układu dyskretnego wyznaczoną metodą residuów (wzór (3.5)):

$$\frac{k}{T^2}\,\frac{Az^2+Bz+C}{z^2+Dz+E} = \frac{Y(z)}{U(z)} * \frac{z^{-2}}{z^{-2}}$$

$$k\left(A+Bz^{-1}+Cz^{-2}\right)U(z) = T^2\left(1+Dz^{-1}+Ez^{-2}\right)Y(z)$$

$$y(n) = \frac{k}{T^2}\left[Au(n)+Bu(n-1)+Cu(n-2)-Dy(n-1)-Ey(n-2)\right] \tag{3.20}$$

Dla przykładowego obiektu o transmitancji danej wzorem (3.10) otrzymamy równie różnicowe postaci:

$$y(n) = 0,0045\ u(n-1) + 0,0041\ u(n-2) + 1,724\ y(n-1) - 0,0741\ y(n-2)$$

Następnie rozpatrzymy transmitancję układu dyskretnego wyznaczoną metodą przekształcenia biliniowego (wzór (3.9)):

$$\frac{a\left(z+1\right)^3}{bz^3+cz^2+dz} = \frac{Y(z)}{U(z)}$$

$$\frac{a\left(z^3+3z^2+3z+1\right)}{bz^3+cz^2+dz} = \frac{Y(z)}{U(z)} \quad \Big| * \frac{z^{-3}}{z^{-3}}$$

$$a\left(1+3z^{-1}+3z^{-2}+z^{-3}\right)U(z) = \left(b+cz^{-1}+dz^{-2}\right)Y(z)$$

$$y(n) = \frac{1}{b}\{a[u(n) + 3u(n-1) + 3u(n-2) + u(n-3)] +$$

$$-cy(n-1) - d(y(n-2)\} \tag{3.21}$$

Dla przykładowego obiektu o transmitancji danej wzorem (3.10) otrzymamy równie różnicowe postaci:

$$y(n) = 1,082 * 10^{-4} u(n) + 3,247 * 10^{-4} u(n-1) + 3,247 * 10^{-4} u(n-2) +$$

$$+1,082 * 10^{-4} u(n-3) + 1,723 y(n-1) - 0,740 y(n-2)$$

Wzory (3.20) i (3.21) mogą zostać wprost zaimplementowane w sterowniku mikroprocesorowym.

4. Praktyczna implementacja modelu obiektów liniowych w sterowniku mikroprocesorowym

Praktyczne zaimplementowanie równania różnicowego w sterowniku mikroprocesorowym to sprawa bardzo prosta nawet dla początkującego programisty. Należy jednak pamiętać o kilku ważnych sprawach, które niekiedy są warunkiem koniecznym dla prawidłowego działania modelu:

- zalecane jest, aby wybrany sterownik zawierał mikroprocesor 32-bitowy, zmiennoprzecinkowy, RISC-owy;
- sterownik mikroprocesorowy powinien być wyposażony zarówno w przetwornik A/C jak i C/A (mogą być wbudowane, mogą być układami zewnętrznymi) o odpowiednich parametrach:
  - czas konwersji krótszy niż zakładany minimalny czas próbkowania,
  - rozdzielczość przetworników powinna zapewniać wymaganą maksymalną dokładność przetwarzania np. przetwornik A/C 10-bitowy zapewni przykładowo pomiar temperatury w zakresie od 32 do 42°C z rozdzielczością 0,1°C,
  - przetworniki powinny być bipolarne. W przypadku stosowania przetworników unipolarnych, należy pamiętać o dodaniu składowej stałej do mierzonego napięcia (niwelującej wartości ujemne), a w przypadku obliczeń dających w wyniku wartości ujemne, o dodaniu wartości stałej przed wysłaniem na przetwornik C/A;
- wybór minimalnego czasu próbkowania jest zależny od typu stosowanego mikroprocesora: im „szybszy" i bardziej wydajny obliczeniowo mikroprocesor, tym możliwy krótszy czas próbkowania;
- kolejne próbki równania różnicowego należy obliczać w jednakowych odstępach czasu równych czasowi próbkowania. Możliwe jest to jedynie w procedurze obsługi przerwania od timera mikroprocesorowego, gdyż tylko wtedy gwarantowana jest powtarzalność czasu. Częstym błędem jest odczyt timera w pętli, z pominięciem procedury przerwań. Niestety nie gwarantuje to powtarzalności czasu;
- czas trwania procedury obsługi przerwania (sumaryczny czas wykonywania wszelkich instrukcji), musi być mniejszy od czasu próbkowania (czyli czasu wywołania kolejnych przerwań);
- gdy czas trwania procedury obsługi przerwania jest zbyt długi, a niezalecane jest wydłużenie czasu próbkowania, możliwe jest skrócenie wykonywania niektórych instrukcji stosując przykładowo poniższe operacje zastępcze (dotyczy języka C), których czas działania jest zdecydowanie krótszy:
  - operację potęgowania pow(a, b), w przypadku gdy b jest liczbą całkowitą, można zastąpić operacją mnożenia np. zamiast:

```
y=pow(x,3);
```
użyć:
```
y=x*x*x;
```

- złożone funkcje matematyczne można zastąpić rozwinięciem danej funkcji w szereg potęgowy (liczba wyrazów rozwinięcia zależy od żądanej dokładności obliczeń), np. zamiast: $y = \sin(x)$ użyć pierwszych trzech wyrazów rozwinięcia funkcji w szereg:

$$y = x - \frac{x*x*x}{6} + \frac{x*x*x*x*x}{120}$$

- funkcję *for*, w przypadku niewielkiej liczby powtórzeń, można zastąpić ciągiem instrukcji, np. zamiast:

```
for(i=0;i=3;i++){
  y[i]=2*x+0.5*i;}
```
użyć:
```
y[0]=2*x;
y[1]=2*x+0.5;
y[2]=2*x+1;
y[3]=2*x+1.5;
```

- w przypadku skoku jednostkowego możemy założyć, że wartości kolejnych próbek sygnału wejściowego są jednakowe, czyli:
  $u(n) = u(n-1) = u(n-2) = \ldots$ itd.,
  co znacznie upraszcza postać równania różnicowego,
- w przypadku sygnału wyjściowego, próbki $y(n-1)$, $y(n-2)$, $y(n-3)$, … itd., to kolejne wartości obliczeń numerycznych w poprzednich krokach próbkowania (tzw. historia obliczeń), zatem należy je zapamiętać. Liczba zapamiętanych wartości wstecz, to w postaci ogólnej stopień mianownika transmitancji dyskretnej obiektu;
- z uwagi na nie zawsze prawidłowe wartości liczbowe obliczeń w stanach przejściowych, należy ograniczyć wielkości wysyłane do przetwornika C/A do jego zakresu pracy, np. w przypadku bipolarnych przetworników 12-bitowych, do zakresu od –4096 do +4095;
- wszelkie obliczenia najlepiej wykonywać na liczbach znormalizowanych, zatem wynik pomiaru sygnału sinusoidalnego dowolnym przetwornikiem unipolarnym lub bipolarnym, o dowolnej rozdzielczości, należy znormalizować do przedziału <–1; +1>;
- z uwagi na zaokrąglenia numeryczne wszelkie obliczenia warto wykonywać na liczbach zmiennoprzecinkowych o standardzie podwójnej precyzji;
- zakładając niezmienność obiektu w czasie oraz stałość czasu próbkowania, wartości współczynników stałych w równaniach różnicowych (a, b, c, d,

e, *A*, *B*, *C*, *D*, *E* we wzorze (3.5) oraz *a*, *b*, *c*, *d*, *e* we wzorze (3.9)) należy policzyć *off-line*. Skróci to czas obliczeń w procedurze obsługi przerwania, a co za tym idzie, może pozwolić na wybór krótszego czasu próbkowania;

- każda zmiana czasu próbkowania, a także zmiana choć jednego parametru obiektu wymaga powtórnego przeliczenia współczynników równania różnicowego. Zmiana czasu T_p, jest dodatkowo tożsama ze zmianą czasu wystąpienia przerwania. Niestety bardzo często poczatkujący programiści o tym zapominają.

W rozdziale tym zostanie zaprezentowana przykładowa realizacja modelu obiektu opisanego transmitancją operatorową daną ogólnym wzorem (3.4), przy zastosowaniu przekształcenia biliniowego. Na wejście obiektu zostanie podany skok jednostkowy o wartości *u*, zatem można założyć, że:

$$u(n) = u(n-1) = u(n-2) = u(n-3)$$

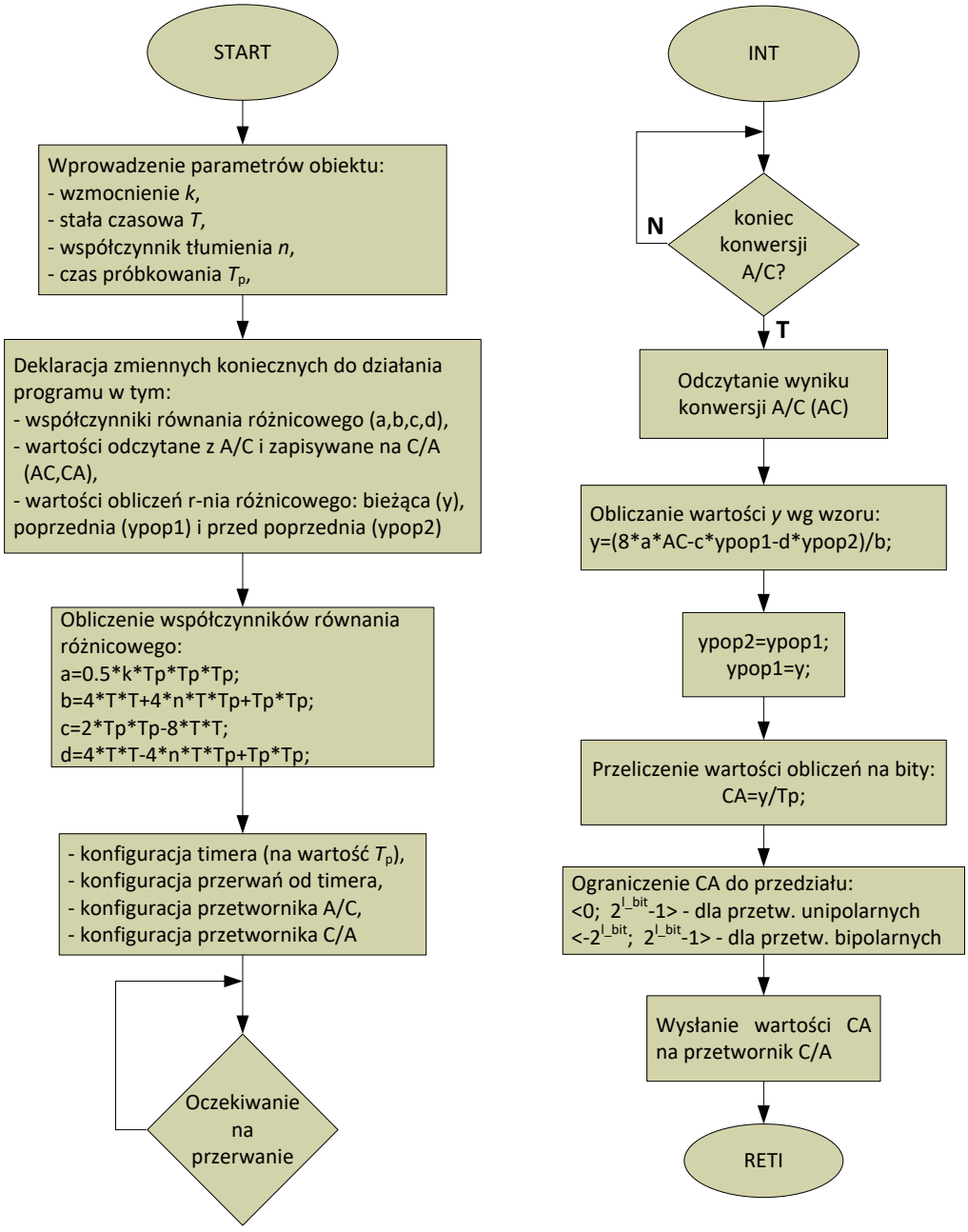
co uprości równanie (3.21) do postaci:

$$y(n) = \frac{1}{b}\left[8a\,u(n) - c\,y(n-1) - d\,y(n-2)\right] \tag{4.1}$$

Na rysunku 4.1 widoczny jest algorytm realizacji modelu obiektu (3.4) zgodnie z powyższym równaniem różnicowym. Algorytm po lewej stronie dotyczy programu głównego, a po prawej podprogramu obsługi przerwania od timera. Program główny zawiera tylko deklaracje zmiennych oraz obliczenia *off-line* współczynników *a*, *b*, *c*, *d*, równania (4.1). Operacje te wykonywane są jednorazowo po uruchomieniu programu. Przerwanie od timera wywoływane jest co czas próbkowania T_p. Obsługa przerwania składa się z trzech zasadniczych części:

- odczyt przetwornika A/C,
- obliczenie wartości *y* (dla kolejnego kroku próbkowania) zgodnie ze wzorem (4.1) wraz z odpowiednim dopasowaniem wyniku obliczeń oraz zapamiętanie niezbędnej „historii” obliczeń,
- zapis przetwornika C/A.

Algorytm zakłada użycie przetworników A/C i C/A o rozdzielczości bitów zapisanej w zmiennej *l_bit*. W przypadku przetworników unipolarnych odczyt (w przypadku przetwornika A/C) i zapis (w przypadku przetwornika C/A) będą w przedziale $<0; 2^{l\_bit}-1>$, natomiast w przypadku przetworników bipolarnych w przedziale $<-2^{l\_bit}; 2^{l\_bit}-1>$.



Rys. 4.1. Algorytm realizacji modelu obiektu (3.4) zgodnie z równaniem różnicowym (4.1)

4.1. Przykład praktycznej implementacji modelu obiektu z wykorzystaniem języka C

Każda rodzina sterowników mikroprocesorowych posługuje się własnym językiem programowania zwanym asemblerem. Programowanie w asemblerze choć jest najefektywniejszą formą programowania, jest bardzo czasochłonne i wymaga dużego doświadczenia. Producenci zdecydowanej większości sterowników mikroprocesorowych oferują dziś darmowe kompilatory języka C, który jest zdecydowanie prostszy od asemblera i bardziej uniwersalny, gdyż, niezależnie od typu mikroprocesora, programy tworzone są w podobny sposób.

Dla modelu obiektu opisanego transmitancją operatorową daną ogólnym wzorem (3.4), zakładamy parametry jak we wzorze (3.10), czyli:

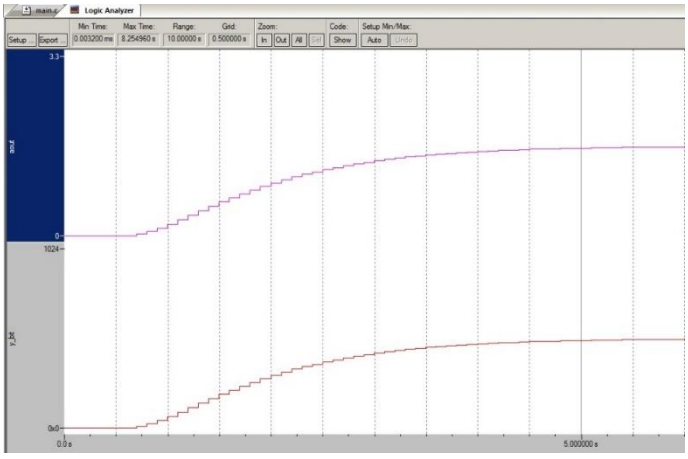
– wzmocnienie $k = 0{,}5$,
– stała czasowa $T = 0{,}7071$,
– współczynnik tłumienia $n = 1{,}0607$,
– czas próbkowania $T_p = 0{,}1$ s.

Przykładowy program w języku C realizujący model obiektu w sterowniku mikroprocesorowym, zostanie przedstawiony w postaci bardzo ogólnej, tzn. z pominięciem kwestii programowania konfiguracji mikroprocesora (ta jest specyficzna dla każdego typu). Szczegółowo zostanie przestawiony tylko sam algorytm realizacji równania. Zakładamy, że przetworniki A/C i C/A są unipolarne, o liczbie bitów $l\_bit = 10$ i czasie konwersji krótszym niż czas próbkowania T_p.

```
// Modelowanie obiektu o transmitancji G(s)=k/(T^2*s^2+2*n*T+1)
// z wykorzystaniem przekształcenia biliniowego s=(2/Tp)*(z-1)/(z+1)
void tc0 (void) __irq;   // deklaracja podprogramu obsługi przerwania od timera
// deklaracja parametrów transmitancji:
double k=0.5;          // wzmocnienie obiektu
double T=0.7071;       // stała czasowa obiektu
double n=1.0607;       // współczynnik tłumienia obiektu
double Tp=0.1;         // czas próbkowania w [s]
// deklaracja pozostałych zmiennych:
unsigned int l_bit=10;         // liczba bitów przetworników A/C i C/A
unsigned int AC=512;     // wartość napięcia odczytana z przetwornika A/C w bitach
  // odpowiada podaniu napięcia skoku jednostkowego równego połowie maksymalnego
  // zakresu przetwornika A/C w przypadku przetwornika 10-bitowego
double a, b, c, d;       // współczynniki równania różnicowego
double y;                // obliczona wartość wyjścia
double ypop1=0;          // wartość wyjścia w poprzednim kroku
double ypop2=0;          // wartość wyjścia 2 kroki wstecz
unsigned int CA;         // wartość wyjścia w bitach, podawane na przetwornika C/A
// Program główny
int main (void){
  // obliczenie współczynników równania różnicowego:
```

```
 a=0.5*k*Tp*Tp*Tp;
 b=4*T*T+4*n*T*Tp+Tp*Tp;
 c=2*Tp*Tp-8*T*T;
 d=4*T*T-4*n*T*Tp+Tp*Tp;
 // konfiguracja timera (odliczanie czasu Tp)
 // konfiguracja przerwania od timera
 // konfiguracja przetwornika A/C
 // konfiguracja przetwornika C/A
 // start timera
 while (1);                    // niekończąca się pętla - oczekiwanie na przerwanie
}
// Procedura obsługi przerwania od timera
void tc0 (void) __irq  {
 // oczekiwanie na koniec konwersji przetwornika A/C
 // odczytanie przetwornika A/C i zapisanie wyniku konwersji do zmiennej AC
 // realizacja równania różnicowego:
 y=(8*a*AC-c*ypop1-d*ypop2)/b;
 ypop2=ypop1;   // zapisanie poprzedniej wartości obliczeń jako wartość przed poprzednią
 ypop1=y;       // zapisanie bieżącej wartości obliczeń jako wartość poprzednią
 CA=y/Tp;       // przeliczenie wartości obliczeń na bity
// ograniczenie wartości bitowej do zakresu od 0 do 2^l_bit-1, (dla przetwornika 10-bit. do 1023)
 if (CA <0)
   CA=0;
 if (CA >1023)
   CA=1023;
 // wysłanie wartości CA na przetwornik C/A
}
```

Na rysunku 4.2 widoczna jest odpowiedź modelu obiektu danego wzorem (3.10), na skok jednostkowy o wartości 3,3 V.



Rys. 4.2. Odpowiedź modelu obiektu danego wzorem (3.10) na skok jednostkowy

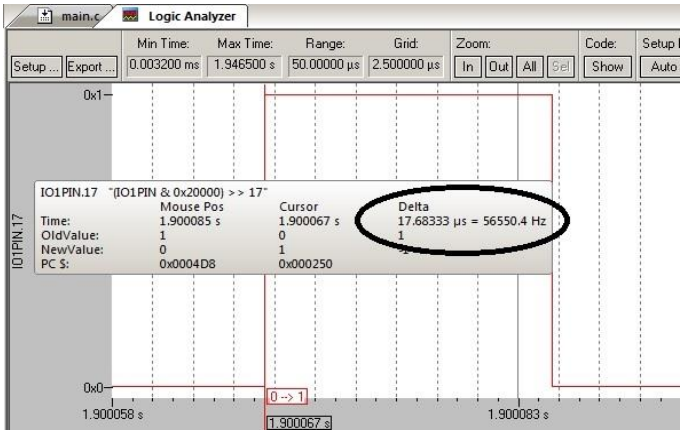
Całość została zrealizowana na sterowniku mikroprocesorowym firmy Phillips LPC2138 rodziny ARM z rdzeniem ARM7TDMI-S z wykorzystaniem środowiska Keil µVision v4.03a. Sterownik ma wbudowane przetworniki A/C i C/A unipolarne, 10-bitowe, o zakresie pomiarowym od 0 do 3,3 V. Wykres górny to sygnał wprost na wyjściu przetwornika C/A. Wykres dolny przedstawia wartość, w bitach, podawaną na ten przetwornik. Jak widać na rysunku, model obiektu zachowuje się zgodnie z oczekiwaniami. Wartość ustalona to 50% wartości zadanej (wzmocnienie $k = 0{,}5$), czyli napięciowo 1,65 V, a bitowo 512.

Warto wyznaczyć czas trwania procedury obsługi przerwania (czas wykonywania wszystkich instrukcji). Pozwoli to na określenie maksymalnej dopuszczalnej częstotliwości próbkowania.

Możliwe są 2 sposoby wyznaczenia tego czasu:

- teoretyczny, polegający na zsumowaniu czasu wykonania wszystkich operacji w obsłudze przerwania,
- praktyczny: najprościej wstawić pierwszą instrukcję procedury przerwania zmieniającą stan dowolnego wyjścia cyfrowego np. na „1” logiczne oraz ostatnią – zmieniającą stan wyjścia cyfrowego na stan przeciwny, czyli logiczne „0”; wystarczy wówczas zmierzyć czas trwania „jedynki” logicznej na tym wyjściu cyfrowym, wykorzystując oscyloskop lub licznik czasu.

Praktyczny sposób wyznaczenia tego czasu jest widoczny na rys. 4.3. Jak widać w badanym przypadku wynosi on ok. 17,68 µs, co pozwala na maksymalną częstotliwość próbkowania ok. 55 kHz.



Rys. 4.3. Praktyczny pomiar czasu trwania procedury obsługi przerwania

4.2. Przykład praktycznej implementacji modelu obiektu z wykorzystaniem programowalnego sterownika logicznego PLC

Realizacja modelu obiektu w sposób opisany w podrozdziale 4.1, wymaga dość dobrej znajomości budowy i działania sterownika mikroprocesorowego oraz umiejętności programowania w języku C lub asemblerze.

Zadanie będzie zdecydowanie prostsze z wykorzystaniem specyficznego sterownika mikroprocesorowego jakim jest programowalny sterownik logiczny PLC. Jego budowa i sposób działania powodują, że użytkownik powinien mieć tylko bardzo podstawową wiedzę w dziedzinie sterowników mikroprocesorowych, a języki programowania (tekstowe lub graficzne) są o wiele łatwiejsze do opanowania niż język C, o asemblerze nawet nie wspominając. W sumie jedyną wadą sterownika PLC są zdecydowanie gorsze parametry czasowe. O ile klasyczny sterownik mikroprocesorowy ma timery pozwalające na odliczanie bardzo krótkich czasów, nawet na poziomie nanosekund, to w przypadku sterownika PLC najkrótszy czas to zazwyczaj 1 ms, zatem maksymalna częstotliwość próbkowania nie może przekraczać wartości 1 kHz. Zdecydowanie dłuższy jest również czas wykonywania wszelkich instrukcji, co ogranicza liczbę operacji możliwych do implementacji w procedurze obsługi przerwania. Nie stanowi to jednak dużej wady, ponieważ zdecydowana większość obiektów automatyki przemysłowej, to obiekty o parametrach wolnozmiennych w czasie.

W podrozdziale tym zostaną zaprezentowane 2 przykładowe programy realizujące model obiektu. Jeden napisany w najczęściej używanym języku graficznym Ladder, drugi w coraz bardziej popularnym języku tekstowym SCL. Podobnie jak to było poprzednio, zostaną pominięte kwestie konfiguracji sterownika PLC, choć z praktycznego punktu widzenia jest ona nieporównywalnie prostsza niż w klasycznych sterownikach mikroprocesorowych. Nie konfiguruje się bowiem przetworników A/C i C/A oraz timera, a tylko tzw. przerwanie czasowe (ang.: *Cyclic Interrupt*), którego konfiguracja ogranicza się do podania czasu występowania przerwania.

Wykorzystany został sterownik firmy Siemens S7 1214C DC/DC/DC oraz środowisko programowe TiA Portal V15. Sterownik ma wbudowany 10-bitowy, unipolarny przetwornik A/C, o zakresie pomiaru napięcia od 0 do +10 V. Dodatkowo został wykorzystany zewnętrzny moduł *Signal Board* zawierający 12-bitowy, bipolarny przetwornik, o zakresie pracy od –10 do +10 V. Zakresowi napięcia +10 V odpowiada wartość 27 648 w tzw. standardzie *S7 analog format*. Dla –10 V jest to odpowiednio –27 648. W realizacji modelu obiektu wykorzystane zostaną tylko dodatnie wartości napięć. Należy dodać, że standard *S7 analog format* jest niezależny od liczby bitów przetwornika, czyli taki sam dla przetwornika np. unipolarnego 10-bitowego (mającego „klasyczny” zakres bitowy

w przedziale <0; 1023>) i 12-bitowego (mającego „klasyczny” zakres bitowy w przedziale <0; 4095>).

Dla modelu obiektu opisanego transmitancją operatorową daną ogólnym wzorem (3.4) zakładamy następujące parametry:

– wzmocnienie $k = 1$,
– stała czasowa $T = 3$,
– współczynnik tłumienia $n = 0{,}3$,
– czas próbkowania $T_p = 0{,}01$ s,

czyli transmitancja obiektu dana jest wzorem:

$$G_O(s) = \frac{1}{9s^2 + 1{,}8s + 1} \tag{4.2}$$

Ponieważ współczynnik tłumienia $n < 1$ zatem odpowiedź modelu obiektu na skok jednostkowy będzie miała charakter oscylacyjny.

Zadanie realizowane jest wg algorytmu blokowego na rys. 4.1. Wszystkie używane zmienne (tzw. Tagi) są deklarowane w oknie *Tag Table* (tabela 4.1).

Tabela 4.1. Wykaz zmiennych (Tagów) stosowanych w programie realizującym model obiektu z wykorzystaniem sterownika PLC

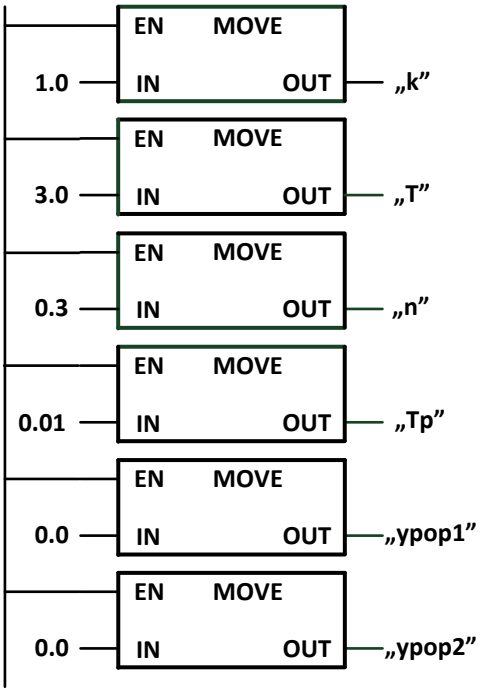
Nazwa Tagu	Typ zmiennej	Opis
k	Real	wzmocnienie obiektu
T	Real	stała czasowa obiektu
Tp	Real	okres próbkowania
n	Real	współczynnik tłumienia obiektu
a	Real	współczynnik równania różnicowego
b	Real	współczynnik równania różnicowego
c	Real	współczynnik równania różnicowego
d	Real	współczynnik równania różnicowego
ypop1	Real	wartość poprzednia wyjścia modelu obiektu
ypop2	Real	wartość przed poprzednia wyjścia modelu obiektu
AC	Int	wejście modelu obiektu w standardzie *S7 analog format* (odczyt przetwornika A/C)
AC_real	Real	wejście modelu obiektu w standardzie *S7 analog format* (odczyt przetwornika A/C) zamienione na *Real*
y	Real	wyjście modelu obiektu
CA_real	Real	wyjście modelu obiektu w standardzie *S7 analog format* – zmienna typu *Real*
CA	Int	wyjście modelu obiektu w standardzie *S7 analog format* – zmienna typu Int (na przetwornik C/A)

W przypadku sterownika PLC warunki początkowe zamieszcza się w bloku *Startup*, który wykonywany jest jednorazowo po uruchomieniu programu. Z uwagi na fakt, że nie używamy klasycznego timera, a przetworniki A/C i C/A nie wymagają konfiguracji, program główny *Main* nie zawiera żadnej instrukcji. Konfiguracja przerwania czasowego (*Cyclic Interrupt*), odbywa się poza programem, w warstwie hardware'owej, na etapie konfiguracji sprzętu i ogranicza się do podania czasu występowania przerwania.

Oczywiście zadawanie parametrów transmitancji w bloku *Startup* nie wyklucza ich zmiany w trakcie działania programu. Jeżeli zakładamy zmianę parametrów obiektu (czyli transmitancji) w trakcie działania programu, to obliczanie współczynników równania różnicowego *a*, *b*, *c*, *d* należy przenieść z bloku *Startup* do bloku programu głównego *Main*.

4.2.1. Realizacja z wykorzystaniem języka Ladder

W podrozdziale tym zostanie szczegółowo omówiona realizacja programu w języku Ladder. Na rysunkach 4.4 i 4.5 przedstawiono zawartość bloku *Startup*.

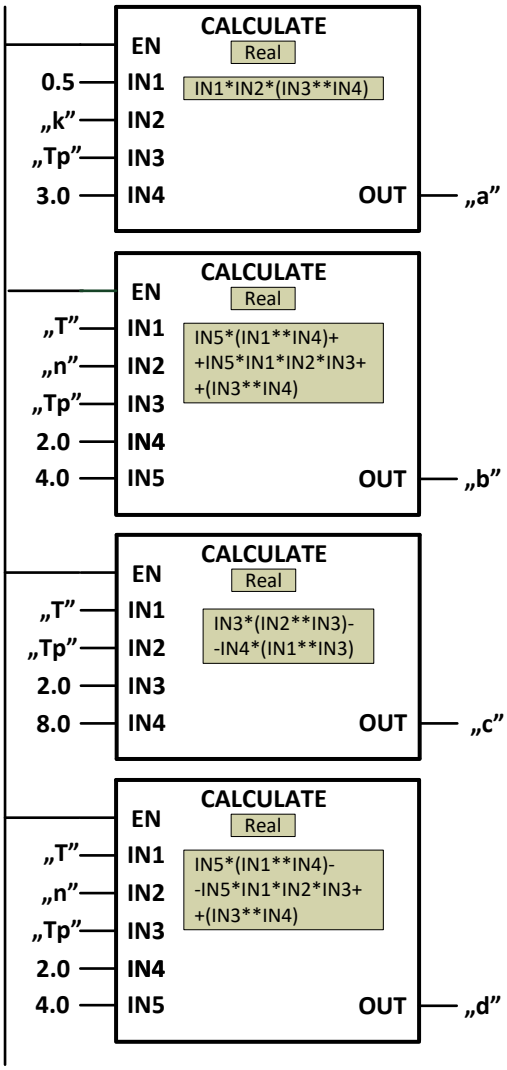


Rys. 4.4. Zadawanie parametrów początkowych w bloku *Startup*

Na rysunku 4.4 widoczne jest zadawanie parametrów początkowych:

- zadawanie parametrów obiektu, czyli transmitancji go opisującej: k, T, n,
- zadawanie czasu próbkowania T_p,
- zerowanie początkowych wartości obliczeń równania różnicowego z dwóch poprzednich kroków *ypop1* i *ypop2*.

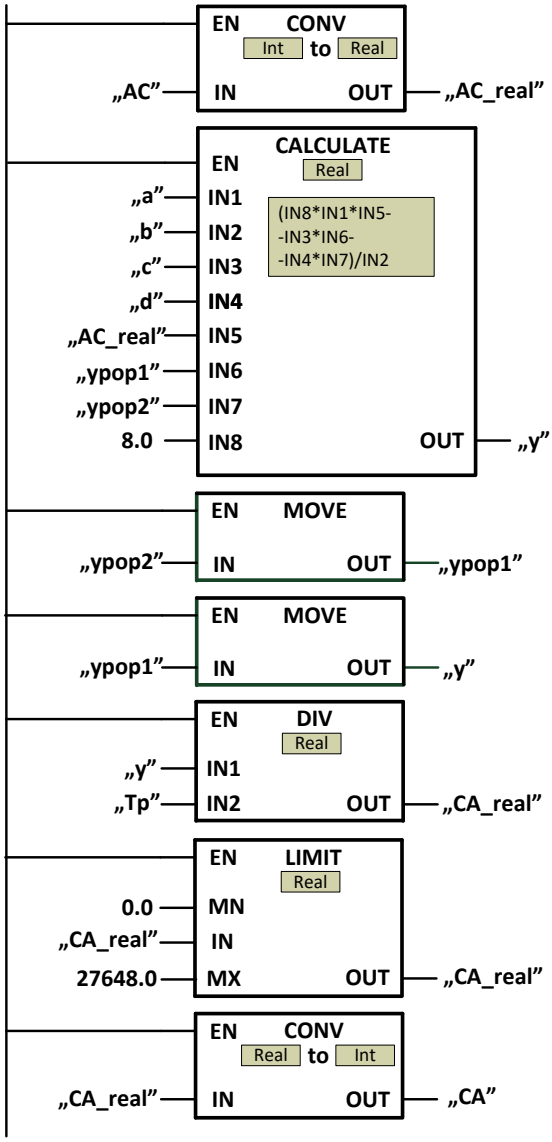
Natomiast na rys. 4.5 przedstawiono obliczenie współczynników a, b, c, d równania różnicowego danego ogólnym wzorem (3.9) dla transmitancji o parametrach zadeklarowanych jak na rys. 4.4.



Rys. 4.5. Obliczanie współczynników równania różnicowego w bloku *Startup*

Wykorzystana została do tego celu funkcja *Calculate*. W przypadku starszych sterowników, w których nie ma takiej funkcji, należy użyć klasycznych operacji algebraicznych. I tak przykładowo, przy obliczaniu współczynnika a danego wzorem: $a = 0,5kT_p^3$, wystarczy ten wzór zapisać w innej, prostszej postaci $a = 0,5kT_pT_pT_p$ i czterokrotnie użyć operacji mnożenia *Mul*.

Obliczanie kolejnych próbek równania różnicowego realizowane jest w bloku *Cyclic Interrupt* (rys. 4.6).



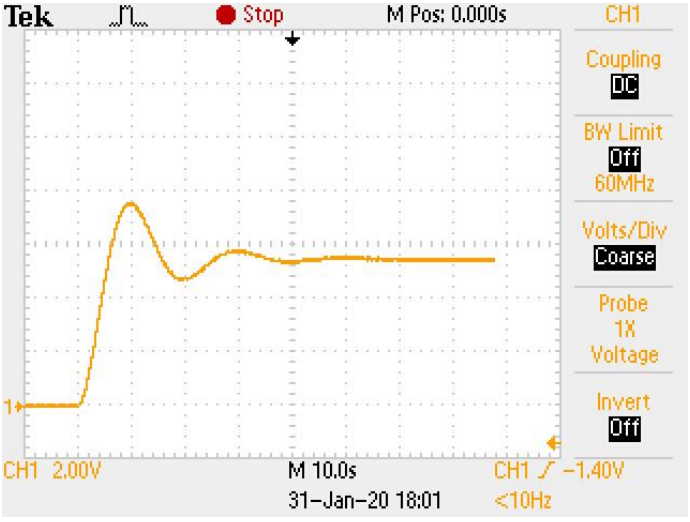
Rys. 4.6. Realizacja równania różnicowego w bloku *Cyclic Interrupt*

Wartość odczytana z przetwornika A/C jest oczywiście typu *Integer* i występuje w standardzie *S7 analog format*, czyli w przedziale od 0 do 27 648. Jest to format uniwersalny niezależny od rozdzielczości przetwornika. Ponieważ obliczenia kolejnych próbek y równania różnicowego dokonywane są na liczbach typu *Real*, zatem uprzednio następuje konwersja wartości odczytanej z przetwornika A/C z typu *Int* na *Real*. Operacja ta nie jest konieczna w nowszych wersjach oprogramowania. Można bowiem użyć liczby np. *Int* w bloku algebraicznym działającym na liczbach typu *Real*. Konwersja nastąpi wówczas automatycznie.

Po wykonaniu obliczenia bieżącej wartości y, konieczne jest zapamiętanie wartości poprzedniej i przed poprzedniej. Będą one niezbędne w następnym kroku (następnym przerwaniu). Operacja dzielenia $CA\_real = y/T_p$, „przeskalowuje" obliczoną wartość do standardu *S7 analog format*.

Z uwagi na nie zawsze prawidłowe wartości liczbowe obliczeń w stanach przejściowych, należy ograniczyć wielkości wysyłane do przetwornika C/A do jego zakresu pracy, czyli do przedziału <0; 27 648>. W tym celu użyta została funkcja *Limit*. Ostatnia operacja to zamiana liczby typu *Real* na *Int* przed wysłaniem do przetwornika C/A.

W przypadku sterownika PLC nie ma konieczności kontrolowania czasu trwania procedury obsługi przerwania (czas wykonywania wszystkich instrukcji). W sytuacji, gdy czas ten przekracza czas próbkowania T_p, sterownik nie przejdzie w tryb pracy *Run*. Należy wówczas albo zwiększyć wartość T_p, albo ograniczyć czas wykonywania instrukcji, np. poprzez zastąpienie instrukcjami o krótszym czasie wykonania.



Rys. 4.7. Odpowiedź na skok jednostkowy modelu obiektu II rzędu o transmitancji danej wzorem (4.2) i $T_p = 0{,}01$ s

Na rysunku 4.7 widoczna jest odpowiedź modelu obiektu na skok jednostkowy o wartości 15 000 w standardzie *S7 analog format*, co opowiada napięciu ok. 5,43 V. Odpowiedź została zarejestrowana na oscyloskopie i jest zgodna z oczekiwaniami.

4.2.2. Realizacja z wykorzystaniem języka SCL

Język tekstowy SCL staje się coraz bardziej popularny ze względu na duże podobieństwo z językami typu C czy Pascal. Co prawda wymaga od użytkownika większej wiedzy i umiejętności programowania niż w przypadku języka graficznego Ladder, ale za to pisanie programów jest sprawniejsze i sam program jest czytelniejszy niż w przypadku języka graficznego, którego instrukcje zajmują stosunkowo dużo miejsca. Poniżej fragment programu bloku *Startup*, zawierający zadawanie parametrów początkowych oraz obliczanie współczynników równania różnicowego.

```
// Realizacja modelu obiektu II-rzędu o transmitancji G(s)=k/(T^2*s^2+2*n*T*s+1)
//STARTUP  - zadawanie wartości początkowych obiektu oraz obliczanie współczynników
//   a,b,c,d równania różnicowego
"k" := 1; // wzmocnienie obiektu
"T" := 3; // stała czasowa obiektu [s]
"n" := 0.3; // współczynnik tłumienia obiektu (n<1 - układ oscylacyjny, n>1 - układ inercyjny)
"Tp":=0.01; // okres próbkowania dla modelu obiektu [s]
                // (musi być zgodny z czasem przerwania cyklicznego)
"ypop1" := 0; // wartość poprzedniej próbki wyjścia z modelu obiekt
"ypop2" := 0; // wartość przed poprzedniej próbki wyjścia z modelu obiektu
// obliczanie współczynników a,b,c,d równania różnicowego
"a" := 0.5 * "k" * ("Tp" ** 3);
"b" := 4 * ("T" ** 2) + 4 * "n" * "T" * "Tp" + ("Tp" ** 2);
 "c" := 2 * ("Tp" ** 2) - 8 * ("T" ** 2);
 "d" := 4 * ("T" ** 2) - 4 * "n" * "T" * "Tp"+ ("Tp" ** 2);
```

Program główny *Main* nie zawiera żadnej instrukcji. Obsługa przerwania cyklicznego wywoływanego z czasem T_p, widoczna jest poniżej.

```
// CYCLIC INTERRUPT Przerwanie cykliczne co 10 ms (musi być zgodne z wartością Tp)
// Realizacja modelu obiektu II-rzędu o transmitancji G(s)=k/(T^2*s^2+2*n*T*s+1)
// Realizacja równania różnicowego:
//       y(n)=(a(u(n)+3*u(n-1)+3*u(n-2)+u(n-3))-c*y(n-1)-d*y(n-2))/b
//       zakładamy, że u(n)=u(n-1)=u(n-2)=u(n-3)
// Parametry k, n, T, Tp zadane wstępnie w STARTUP
// współczynniki a, b, c, d obliczone w STARTUP
```

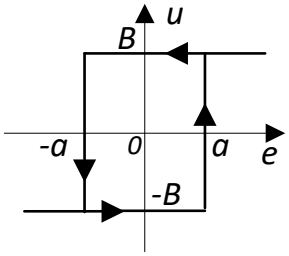
```
"AC_real":= INT _TO_ REAL ("AC"); // konwersja wartości odczytanej z przetwornika A/C
"y" := (8 * "a" * "AC_real" - "c" * "ypop1" - "d" * "ypop2") / "b";   // obliczenie kolejnej
                // próbki równania różnicowego
"ypop2" := "ypop1";       // wartość poprzednia y staje się przed poprzednią
"ypop1" := "y";               // wartość bieżąca y staje się wartością poprzednią
"CA_real" := "y" / "Tp";  // przeskalowanie wyjścia do standardu S7 analog format
"CA_real":= LIMIT(MN := 0, IN := "CA_real", MX := 27648.0);    // ograniczenie wartości
                 // do przedziału  <0; 27648.0>
"CA":= REAL_TO_INT("CA_real");    // konwersja na Int i wysłanie na przetwornik C/A
```

5. Praktyczna implementacja modelu elementów nieliniowych w sterowniku mikroprocesorowym

Nieliniowość to cecha układu polegająca na tym, że wartość wyjściowa nie jest wprost proporcjonalna do wartości wejściowej. Z matematycznego punktu widzenia w przypadku modelu nieliniowego nie obowiązuje zasada superpozycji, co oznacza, że odpowiedź układu na sumę kilku wymuszeń nie jest równa sumie odpowiedzi na każde z tych wymuszeń. W przyrodzie większość zjawisk ma cechę nieliniowości i opisywana jest funkcjami nieliniowymi. Często jednak nieliniowość jest na tyle niewielka, że obiekt zastępuje się modelem liniowym w celu uproszczenia jego opisu. Zazwyczaj obiekt zastępowany jest kilkoma modelami liniowymi dla różnych zakresów zmian wartości wejściowych.

W przypadku wystąpienia wyraźnej nieliniowości obiektu przyjęcie liniowego opisu prowadzi do zbyt dużych błędów, co uniemożliwia realizację sterowania. Niekiedy celowo wprowadza się nieliniowość do układu regulacji, aby uzyskać pożądane cechy układu [5].

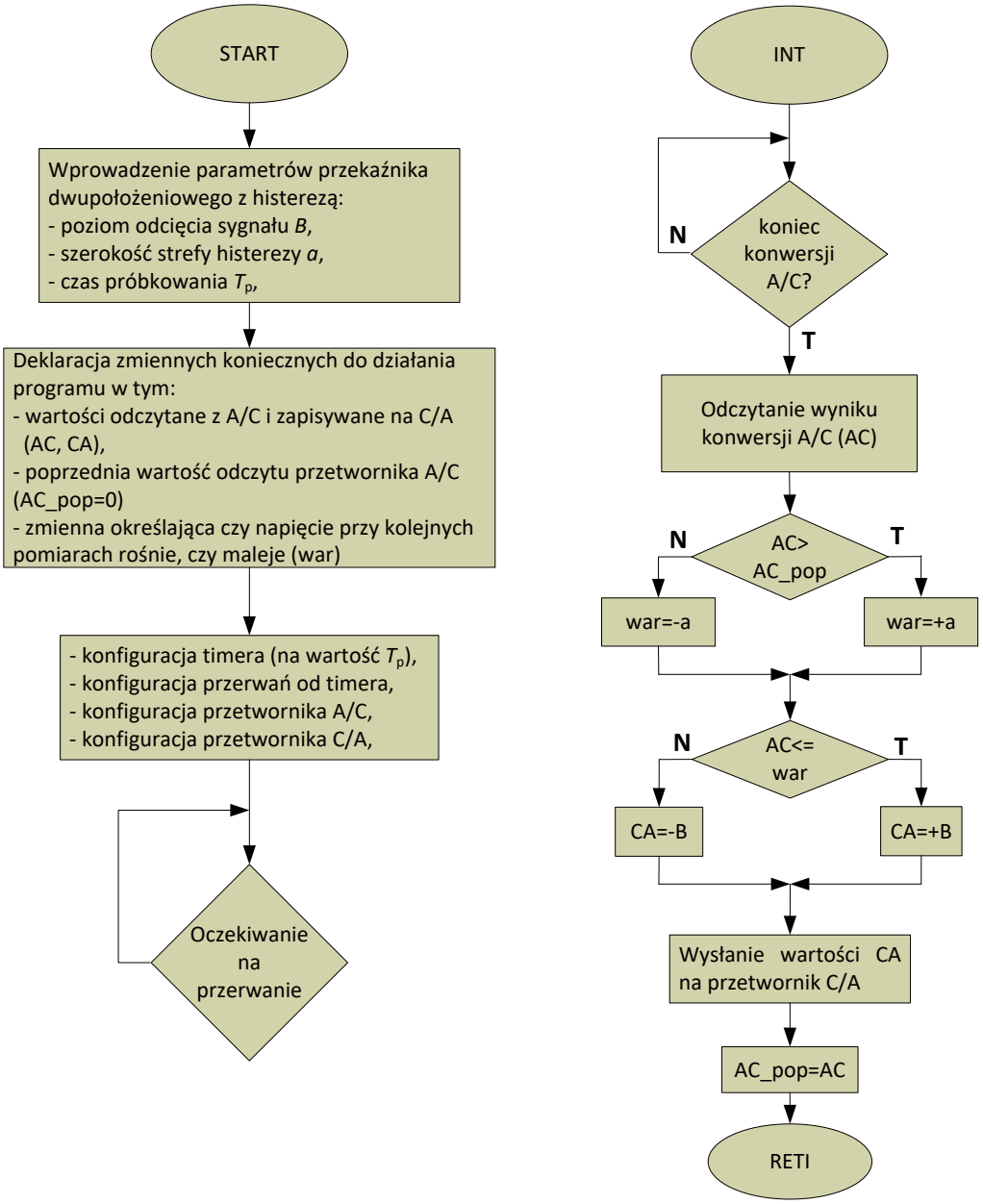
Modelowanie elementów nieliniowych jest o wiele prostsze niż to wynika z teorii układów nieliniowych. Nie jest konieczna bowiem znajomość funkcji opisujących elementy nieliniowe lub innych metod je definiujących [5], [8]. W zasadzie cały algorytm opiera się na realizacji kilku prostych funkcji typu warunkowego. W rozdziale zostanie zaprezentowana realizacja przykładowego elementu nieliniowego jakim jest przekaźnik dwupołożeniowy z histerezą o charakterystyce jak na rys. 5.1. Jest to element dość często występujący w praktyce i tak przykładowo, gdy zmienna B przyjmuje tylko wartości dodatnie, to taką charakterystykę ma np. klasyczny bimetal stosowany w żelazkach.



Rys. 5.1. Przekaźnik dwupołożeniowy z histerezą

Model elementu nieliniowego zostanie zaimplementowany w trzech językach programowania: język C (klasyczny sterownik mikroprocesorowy) oraz języki Ladder i SCL (programowalny sterownik logiczny PLC).

Na rysunku 5.2 widoczny jest algorytm realizacji modelu przekaźnika dwupołożeniowego z histerezą.



Rys. 5.2. Algorytm realizacji modelu przekaźnika dwupołożeniowego z histerezą

5.1. Przykład praktycznej implementacji modelu elementu nieliniowego z wykorzystaniem języka C

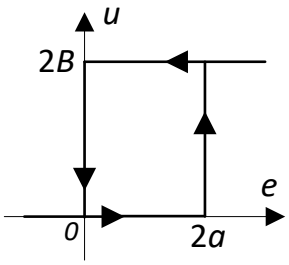
Przykładowy program w języku C realizujący model elementu nieliniowego w sterowniku mikroprocesorowym, zostanie przedstawiony w postaci bardzo ogólnej, tzn. z pominięciem kwestii programowania konfiguracji mikroprocesora (ta jest specyficzna dla każdego typu). Szczegółowo zostanie przestawiony tylko sam algorytm realizacji przekaźnika.

Całość została zrealizowana na sterowniku mikroprocesorowym firmy Phillips LPC2138 rodziny ARM z rdzeniem ARM7TDMI-S z wykorzystaniem środowiska Keil µVision v4.03a. Sterownik ma wbudowane przetworniki A/C i C/A unipolarne, 10-bitowe, o zakresie pomiarowym od 0 do 3,3 V i czasie konwersji krótszym niż zakładany czas próbkowania T_p.

W celu praktycznej realizacji modelu przekaźnika dwupołożeniowego z histerezą zakładamy jego parametry:

– poziom odcięcia sygnału $B = 256$ (wartość w bitach), co odpowiada napięciu ok. 0,83 V,
– szerokość strefy histerezy $a = 128$ (wartość w bitach), co odpowiada napięciu ok. 0,41 V,
– czas próbkowania $T_p = 0{,}001$ s.

Przetworniki A/C i C/A są unipolarne, czyli nie ma fizycznej możliwości realizacji przekaźnika dwupołożeniowego z histerezą zgodnie z rys. 5.1. Zostanie zatem zrealizowany model przekaźnika zgodnie z rys. 5.3.



Rys. 5.3. Przekaźnik dwupołożeniowy z histerezą
(w wersji dla unipolarnych przetworników A/C i C/A)

Poniżej przykładowy program w języku C realizujący model przekaźnika dwupołożeniowego z histerezą.

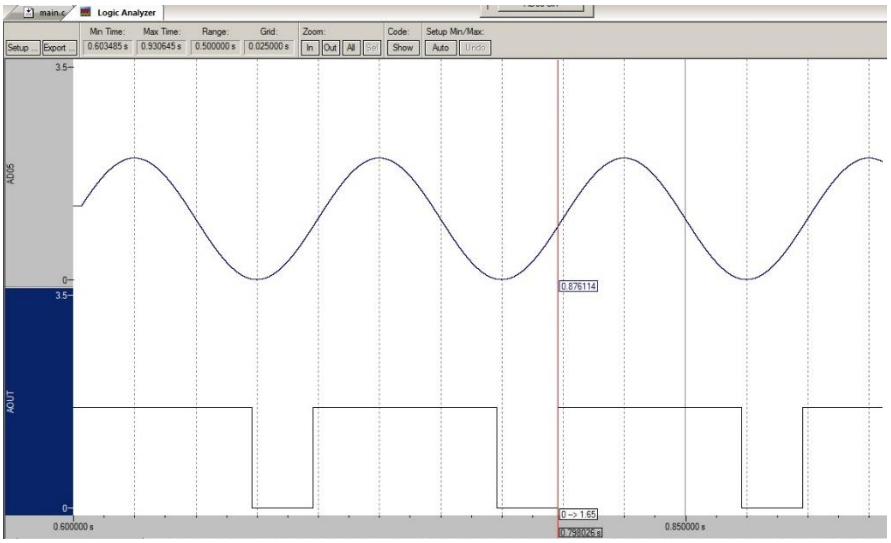
```
// Modelowanie elementu nieliniowego: przekaźnik dwupołożeniowy z histerezą
void tc0 (void) __irq;      // deklaracja podprogramu tc0 obsługi przerwania od timera
// Deklaracja parametrów elementu nieliniowego (podawane w bitach);
```

```
//   dla przetworników 10-bitowych mogą przyjmować wartości z przedziału <0; 1023>:
int B=256;      // wartość wyjściowa przekaźnika (w bitach), czyli poziom odcięcia sygnału;
                //  odpowiada napięciu (256/1024)*3,3V=0,83V
int a=128;        // szerokość strefy histerezy (w bitach);
                  // odpowiada napięciu (128/1024)*3,3V=0,41V
int Tp=0.001;    // czas próbkowania (w sekundach)
// deklaracja pozostałych zmiennych:
unsigned int AC;       // wartość napięcia odczytana z przetwornika A/C w bitach
unsigned int AC_pop=0; // wartość napięcia w bitach - pomiar poprzedni
int war;  // wartość porównywana z sygnałem wejściowym
          // jeżeli sygnał wejściowy <= war, to wyjście przyjmuje wartość 0
          // jeżeli sygnał wejściowy > war, to wyjście przyjmuje wartość  2B.
          // Zmienna war przyjmuje 2 wartości:
          // war=2a - gdy sygnał wejściowy rośnie
          // war=0 - gdy sygnał wejściowy maleje. W praktyce jednak przyjęto war=2 (bitowo),
          //   ponieważ przetwornik nie mierzy napięć ujemnych,
          //   a rzeczywisty sygnał wejściowy rzadko przyjmuje wartość 0 (szumy i zakłócenia)
unsigned int CA;       // wartość wysyłana na przetwornik C/A

int main (void) {
 // konfiguracja timera (odliczanie czasu Tp)
 // konfiguracja przerwania od timera
 // konfiguracja przetwornika A/C
 // konfiguracja przetwornika C/A
 // start timera
 while (1);                    // niekończąca się pętla - oczekiwanie na przerwanie
}
// Procedura obsługi przerwania od timera
void tc0 (void) __irq {
 // oczekiwanie na koniec konwersji przetwornika A/C
 // odczytanie przetwornika A/C i zapisanie wyniku konwersji do zmiennej AC
 if((AC)>AC_pop)      // sprawdzenie kierunku zmian napięcia
   war=2*a;             // napięcie rośnie
 else
   war =2;              // napięcie maleje
 if (AC<=war)
   CA=0;               // przy malejącym napięciu wyjście przyjmuje wartość równą 0
 else
   CA=2*B;             // przy rosnącym napięciu wyjście przyjmuje wartość równą +2B
 // wysłanie wartości CA na przetwornik C/A
 AC_pop=AC;      // zapamiętanie poprzedniego wyniku pomiaru
}
```

Powyższy program, to oczywiście wersja najprostsza, niedoskonała. W praktyce, aby sprawdzić, czy zmiany napięcia na wejściu przekaźnika mają tendencję rosnącą czy malejąca, należy dokonać sprawdzenia kilkukrotnego, aby wykluczyć przypadkowe zakłócenia.

Działanie przekaźnika dwupołożeniowego z histerezą widoczne jest na rys. 5.4. Wykres górny przedstawia zewnętrzny sygnał podawany wprost na wejście elementu nieliniowego. Wykres dolny, to wyjście przekaźnika. Całość wyskalowana w woltach.



Rys. 5.4. Odpowiedź przekaźnika dwupołożeniowego z histerezą (wykres dolny) na sygnał wejściowy typu sinus (wykres górny)

5.2. Przykład praktycznej implementacji modelu elementu nieliniowego z wykorzystaniem programowalnego sterownika logicznego PLC

W podrozdziale tym zostaną zaprezentowane dwa przykładowe programy realizujące model elementu nieliniowego. Jeden w języku graficznym Ladder, drugi w języku tekstowym SCL. Podobnie jak to opisano poprzednio, nie ma potrzeby konfiguracji sterownika PLC, za wyjątkiem przerwania czasowego (*Cyclic Interrupt*), którego konfiguracja ogranicza się do podania czasu występowania przerwania.

Wykorzystany został sterownik firmy Siemens S7 1214C DC/DC/DC oraz środowisko programowe TiA Portal V15. Tym razem nie zostały wykorzystane: wbudowany 10-bitowy, unipolarny przetwornik A/C, o zakresie pomiaru napięcia od 0 do +10 V oraz moduł *Signal Board* zawierający 12-bitowy, bipolarny przetwornik, o zakresie pracy od –10 do +10 V. Został za to zastosowany dodatkowy moduł zewnętrzny wejść i wyjść analogowych SM 1234, zawierający

cztery przetworniki A/C i dwa przetworniki C/A, 14-bitowe, bipolarne, o zakresie pracy od –10 V do +10 V.

Zakresowi napięcia +10 V odpowiada wartość 27 648 w tzw. standardzie *S7 analog format*. Dla –10 V jest to odpowiednio –27 648.

Zadanie realizowane jest wg algorytmu blokowego na rys. 5.2. Wszystkie używane zmienne są deklarowane w oknie *Tag Table* (tabela 5.1).

Tabela 5.1. Wykaz zmiennych (Tagów) stosowanych w programie realizującym model elementu nieliniowego z wykorzystaniem sterownika PLC

Nazwa Tagu	Typ zmiennej	Opis
B	Int	wartość wyjściowa przekaźnika (czyli poziom odcięcia sygnału) w standardzie *S7 analog format* <–27 648; +27 648>
a	Int	szerokość strefy histerezy w standardzie *S7 analog format* <–27 648; +27 648>
AC	Int	wejście modelu elementu nieliniowego w standardzie *S7 analog format* (odczyt przetwornika A/C)
AC_pop	Int	wejście modelu elementu nieliniowego w standardzie *S7 analog format* (odczyt przetwornika A/C) – wartość poprzednia
war	Int	wartość porównywana z sygnałem wejściowym: • jeżeli sygnał wejściowy <= war, to wyjście przyjmuje wartość –B, • jeżeli sygnał wejściowy > war, to wyjście przyjmuje wartość +B, • zmienna war przyjmuje 2 wartości: o war = +a – gdy sygnał wejściowy rośnie, o war = –a – gdy sygnał wejściowy maleje.
CA	Int	wyjście modelu przekaźnika w standardzie *S7 analog format* (wartość wpisywana na przetwornik C/A)

Program główny *Main* nie zawiera żadnej instrukcji. Konfiguracja przerwania czasowego (*Cyclic Interrupt*), odbywa się poza programem, w warstwie sprzętowej, na etapie konfiguracji i ogranicza się do podania czasu występowania przerwania.

5.2.1. Realizacja z wykorzystaniem języka Ladder

W podrozdziale tym zostanie szczegółowo omówiona realizacja programu w języku Ladder. Na rysunku 5.5 przedstawiono zawartość bloku *Startup*, gdzie

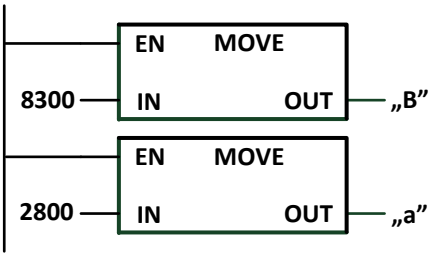
określone są parametry początkowe przekaźnika w standardzie *S7 analog format*:

B = 8300, co odpowiada napięciu ok. 3 V,

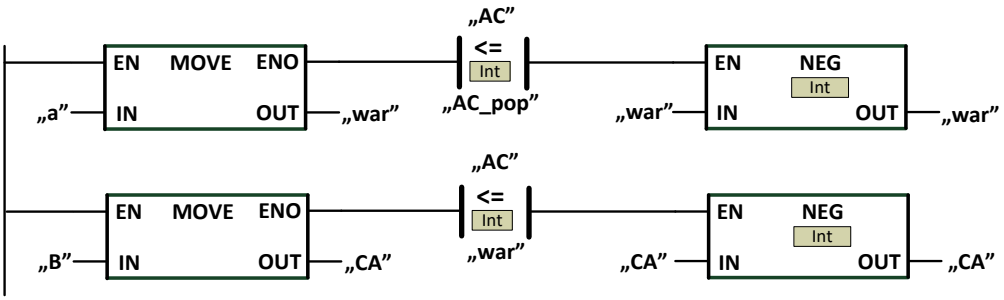
a = 2800, co odpowiada napięciu ok. 1 V.

Oczywiście zadawanie parametrów w bloku *Startup* nie wyklucza ich zmiany w trakcie działania programu.

W bloku *Cyclic Interrupt* (rys. 5.6). wywoływanym co 100 ms, zamieszczona została programowa realizacja przekaźnika dwupołożeniowego z histerezą. Napięcie wejściowe podawane na przekaźnik (zmienna *AC* odczytywana z przetwornika A/C) porównywane jest z wartością poprzednią (*AC_pop*). Jeżeli wartość napięcia na wejściu przekaźnika rośnie, to zmienna *war* = *a*, jeżeli maleje to *war* = –*a*. W kolejnym kroku zmienna *AC* porównywana jest ze zmienną *war*. Jeżeli napięcie przekracza wartość zmiennej *war*, to na wyjście elementu nieliniowego (przetwornik C/A) podawana jest wartość +*B*, jeżeli nie, to CA = –*B*.



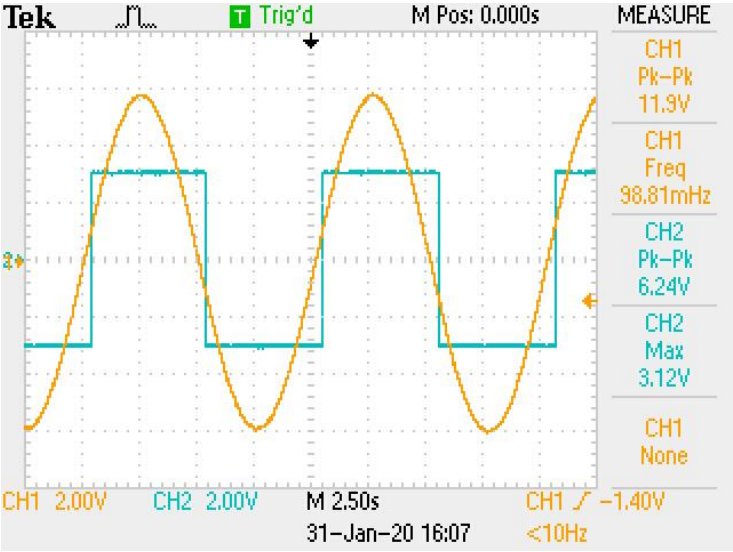
Rys. 5.5. Zadawanie parametrów początkowych w bloku *Startup*



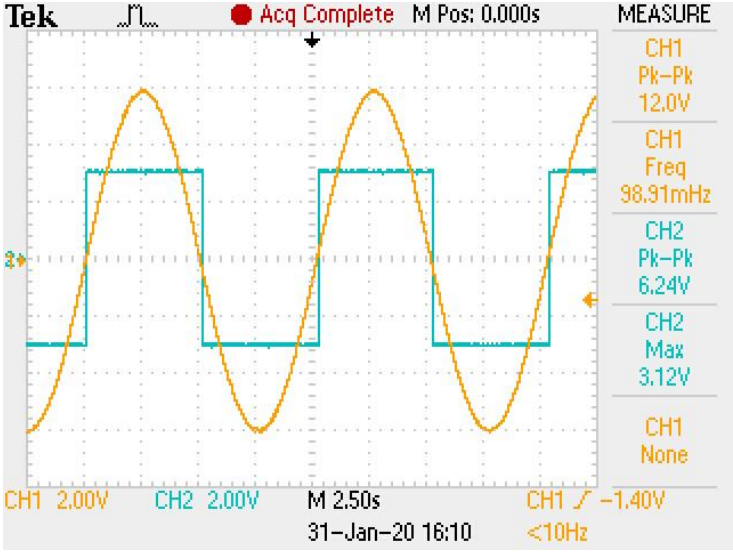
Rys. 5.6. Numeryczna realizacja przekaźnika dwupołożeniowego z histerezą w bloku *Cyclic Interrupt*

Na rysunku 5.7 widoczna jest odpowiedź modelu przekaźnika dwupołożeniowego z histerezą na sygnał wejściowy typu sinus, zarejestrowana na oscyloskopie. Warto zwrócić uwagę na histerezę. Objawia się ona tym, że charakterystyki przecinają się w punkcie ok. +1 V (czyli wartość *a*), w przypadku gdy napięcie rośnie oraz –1 V, w przypadku gdy napięcie maleje.

Dla porównania na rys. 5.8 zamieszczona została odpowiedź modelu przekaźnika dwupołożeniowego bez histerezy ($a = 0$), na sygnał wejściowy typu sinus. Obydwie charakterystyki przecinają się dokładnie w punkcie 0 V.



Rys. 5.7. Odpowiedź modelu przekaźnika dwupołożeniowego z histerezą na sygnał wejściowy typu sinus



Rys. 5.8. Odpowiedź modelu przekaźnika dwupołożeniowego bez histerezy na sygnał wejściowy typu sinus

5.2.2. Realizacja z wykorzystaniem języka SCL

Realizacja modelu przekaźnika dwupołożeniowego z histerezą w języku tekstowym SCL jest równie prosta (jak nie prostsza) jak w języku Ladder. Poniżej fragment programu bloku *Startup*, zawierający zadawanie parametrów początkowych elementu nieliniowego.

```
// Realizacja modelu przekaźnika dwupołożeniowego z histerezą
//STARTUP  - zadawanie wartości początkowych elementu nieliniowego
// początkowe parametry przekaźnika dwupołożeniowego z histerezą
"B" := 8300;    // wartość wyjściowa przekaźnika (czyli poziom odcięcia sygnału)
                // (w standardzie S7 analog format - odpowiada napięciu ok. 3 V)
"a" := 2800;    // szerokość strefy histerezy
                // (w standardzie S7 analog format - odpowiada napięciu ok. 1 V)
```

Program główny *Main* nie zawiera żadnej instrukcji. Obsługa przerwania cyklicznego wywoływanego z czasem T_p, widoczna jest poniżej.

```
// Realizacja przekaźnika dwupołożeniowego z histerezą.
// CYCLIC INTERRUPT. Przerwanie cykliczne co 100 ms
// Parametry przekaźnika:
// B - wartość wyjściowa przekaźnika w standardzie S7 analog format <-27648; +27648>
// a - szerokość strefy histerezy w standardzie S7 analog format <-27648; +27648>
// AC - odczyt przetwornika A/C (wejście modelu elementu nieliniowego w standardzie
//              S7 analog format)
// AC_pop - odczyt przetwornika A/C - wartość poprzednia
// war - wartość porównywana z sygnałem wejściowym:
// - jeżeli sygnał wejściowy <= war, to wyjście przyjmuje wartość –B,
// - jeżeli sygnał wejściowy > war, to wyjście przyjmuje wartość +B,
// - zmienna "war" przyjmuje 2 wartości:
//     - war=+a - gdy sygnał wejściowy rośnie,
//     - war=-a - gdy sygnał wejściowy maleje.
// CA - wyjście modelu przekaźnika w standardzie S7 analog format (podawane
//           na przetwornik C/A)
IF ("AC" > "AC_pop") THEN // sprawdzenie kierunku zmian napięcia
  "war" := "a";       // napięcie rośnie
ELSE
  "war":=-"a";      // napięcie maleje
END_IF;

IF ("AC"<="war") THEN
  "CA":=-"B";          // przy malejącym napięciu wyjście (przetwornik C/A) przyjmuje wartość -B
ELSE
  "CA" := "B";         // przy rosnącym napięciu wyjście (przetwornik C/A) przyjmuje wartość +B
END_IF;
"AC_pop" := "AC";  // zapamiętanie bieżącego wyniku pomiaru jako poprzedni
```

6. Filtracja cyfrowa

Zadaniem filtracji [5]–[7] jest przepuszczenie, najlepiej bezstratne, składowych sygnału w żądanym, określonym paśmie częstotliwości i eliminacja, najlepiej całkowita, składowych o innych częstotliwościach. Z uwagi na pasmo przenoszonej częstotliwości filtry dzielimy na:

- dolnoprzepustowe,
- górnoprzepustowe,
- pasmowoprzepustowe

lub odwrotnie, ze względu na pasmo blokowanej częstotliwości:

- dolnozaporowe,
- górnozaporowe,
- pasmowozaporowe.

Filtry cyfrowe można przedstawić za pomocą ogólnej transmitancji w dziedzinie **Z**:

$$F(z) = \frac{\sum_{k=0}^{N-1} a(k) z^{-k}}{1 + \sum_{k=0}^{M-1} b(k) z^{-k}} \tag{6.1}$$

gdzie:

$a(k)$ i $b(k)$ to k-te współczynniki filtru,

N – liczba współczynników a,

M – liczba współczynników b,

natomiast czynnik z^{-k} reprezentuje opóźnienie o k próbek (twierdzenie o próbkowaniu przy zerowych warunkach początkowych) [5]-[7].

W przeciwieństwie do filtrów analogowych, które mają zawsze nieskończenie długą odpowiedź impulsową, filtry cyfrowe dzielimy na dwa rodzaje, ze względu na czas odpowiedzi impulsowej:

- filtry rekursywne (NOI) charakteryzujące się nieskończoną odpowiedzią impulsową,
- filtry nierekursywne (SOI) charakteryzujące się skończoną odpowiedzią impulsową.

Filtry rekursywne wytwarzają kolejne próbki sygnału wyjściowego jako sumę ważoną próbek sygnału wejściowego i wyjściowego. Ich algorytm dany jest wzorem:

$$y(n) = \sum_{k=0}^{N-1} \big(a(k)x(n-k)\big) - \sum_{k=0}^{M-1} \big(b(k)y(n-k)\big) \tag{6.2}$$

gdzie:

$x(n)$ – n-ta próbka sygnału wejściowego,

$y(n)$ – n-ta próbka sygnału wyjściowego,

$a(k)$, $b(k)$ – k-te współczynniki filtru.

Jak widać, wzór (6.2) jest niepraktyczny do numerycznej realizacji. Wymagana jest bowiem znajomość N próbek sygnału wejściowego i M próbek sygnału wyjściowego, czyli „historii" obliczeń.

Znacznie prostszy w realizacji jest filtr nierekursywny, którego algorytm dany jest wzorem:

$$y(n) = \sum_{k=0}^{N-1} \left(a(k)x(n-k) \right) \tag{6.3}$$

Jest to szczególny przypadek filtru rekursywnego o zerowych współczynnikach b, którego odpowiedź jest skończona i równa długości okna filtru:

$$T_f = NT_p \tag{6.4}$$

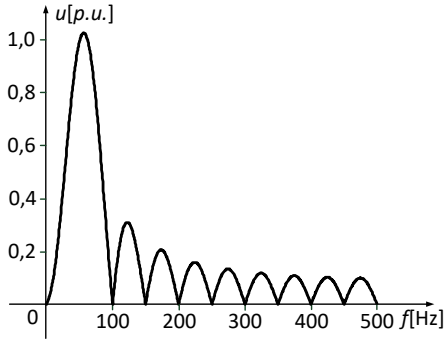
Wzór (6.1) przyjmuje wówczas postać:

$$F(z) = \sum_{k=0}^{N-1} a(k) z^{-k} \tag{6.5}$$

Istnieje bardzo dużo rozmaitych standardowych filtrów cyfrowych. Nie jest rolą tego podręcznika ich opis. Szczegóły czytelnik może znaleźć w literaturze [5]–[7]. W celu praktycznej realizacji filtru cyfrowego zostanie zaprezentowany przykładowy filtr, o oknie kosinusoidalnym danym wzorem:

$$a_c(n) = \frac{2}{N} \cos\left(\frac{2\pi n}{N} \right) \tag{6.6}$$

Charakterystyka widmowa takiego filtru zamieszczona jest na rys. 6.1. Jak widać, jest to typowy filtr dolnoprzepustowy.



Rys. 6.1. Charakterystyka częstotliwościowa filtru o oknie kosinusoidalnym

7. Praktyczna implementacja filtracji cyfrowej w sterowniku mikroprocesorowym

Algorytm filtru nierekursywnego jest stosunkowo prosty do numerycznej realizacji. Z algebraicznego punktu widzenia to suma iloczynów próbek sygnału ze współczynnikami filtru. Współczynniki filtru mogą być policzone odrębnym narzędziem np. w pakiecie Matlab i wprowadzone do programu w postaci tablicy. Mogą być również policzone *off-line* w samym programie, co zostanie zrobione w tym rozdziale. Oczywiście przy zmianie długości okna filtracyjnego konieczne jest powtórne przeliczenie współczynników.

Początkującym programistom problem może sprawić dostęp do „historii" próbek. Jeżeli realizujemy algorytm filtracji w pakietach matematycznych typu np. Matlab, to deklarujemy cały sygnał wejściowy, wobec tego w każdej chwili mamy dostęp do dowolnej próbki sygnału wstecz. W przypadku pomiarów *on-line*, a z takimi mamy do czynienia w układach rzeczywistych, nie jesteśmy w stanie, z oczywistych względów, zapamiętać wszystkich próbek mierzonego sygnału wejściowego. Konieczne jest zapamiętanie minimalnej liczby próbek równej długości okna filtracyjnego. Realizowane jest to w ten sposób, że tworzona jest tablica o rozmiarze długości okna filtracyjnego N, do której zapisywane są na bieżąco próbki mierzonego sygnału wejściowego. Po wypełnieniu tablicy jest ona nadpisywana kolejnymi, nowymi próbkami. Algorytm filtracyjny musi być realizowany w każdym kroku. Mamy dostęp do N próbek wstecz, jednak nie zawsze jest to dokładnie N elementów tablicy wstecz. Przykładowo niech $N = 10$, zatem tablica próbek jest postaci $\{x_0, x_1, x_2, ..., x_8, x_9\}$. Jeżeli aktualnie mierzona próbka jest zapisywana jako ostatni, dziesiąty element tablicy (x_9), to oczywiście nie ma problemu: poprzednia próbka jest zapisana w komórce x_8, a najstarsza w komórce x_0. Jeżeli natomiast bieżąca próbka zapisywana jest do komórki np. x_5, to poprzednie próbki znajdują się w komórkach tablicy o numerach:

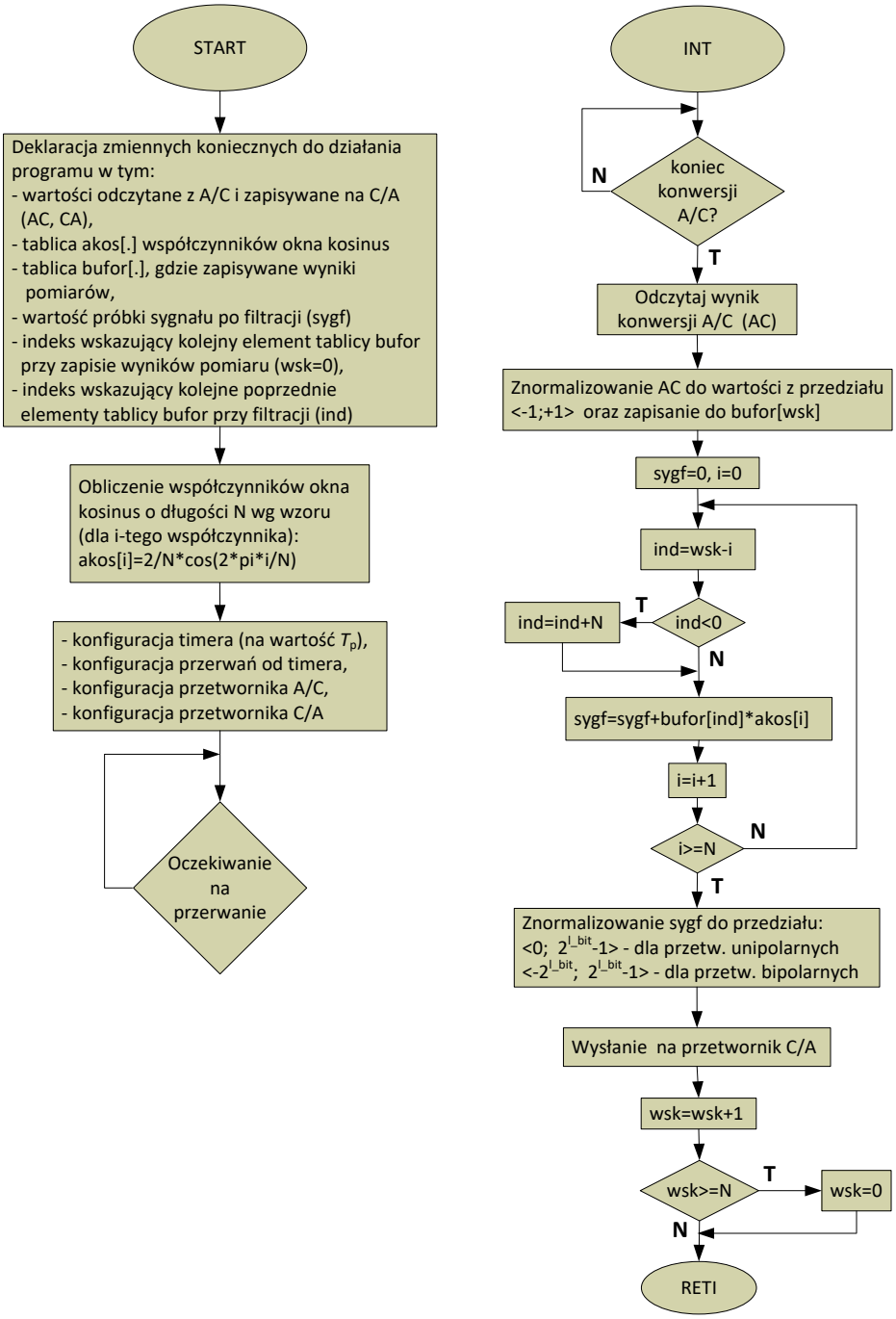
$$x_4, x_3, x_2, x_1, x_0, x_9, x_8, x_7, x_6 .$$

Sprawa niby oczywista, ale często bywa przyczyną błędów.

Na rysunku 7.1 widoczny jest algorytm realizacji filtracji cyfrowej. Wartości współczynników okna kosinusoidalnego liczone są *off-line*. Cały algorytm filtracji wykonywany jest w przerwaniu od timera.

Kilka słów wyjaśnienia wymaga operacja normalizacji zmiennej AC do wartości z przedziału <–1; +1> przed zapisaniem do pamięci. Do zmiennej AC zapisywane są wartości wprost z przetwornika A/C, czyli w postaci bitowej, zależnej oczywiście od rozdzielczości przetwornika oraz od typu, czy jest unipolarny czy bipolarny. Konieczne jest, aby przy jakichkolwiek operacjach na sygnale

(w tym wypadku algorytm filtracji) znormalizować jego wartości do przedziału <–1; +1>, co uniezależni działanie każdego algorytmu od stosowania różnego typu przetworników.



Rys. 7.1. Algorytm realizacji filtracji cyfrowej

Przykładowo, w przypadku 10-bitowego, unipolarnego przetwornika A/C wartości odczytane będą w przedziale <0; 1023>. Normalizacja do przedziału <–1; +1>, wymaga wykonania operacji:

$$AC/512 - 1.$$

Podobnie w przypadku np. 12-bitowego, bipolarnego przetwornika A/C wartości odczytane będą w przedziale <–4096; 4095>. Normalizacja do przedziału <–1; +1>, wymaga wykonania operacji:

$$AC/4096.$$

Algorytm zakłada użycie przetworników A/C i C/A o rozdzielczości bitów zapisanej w zmiennej *l_bit*. W przypadku przetworników unipolarnych odczyt (w przypadku przetwornika A/C) i zapis (w przypadku przetwornika A/C) będą w przedziale <0; $2^{l\_bit}-1$>, natomiast w przypadku przetworników bipolarnych w przedziale <-2^{l_bit}; $2^{l\_bit}-1$>.

7.1. Przykład praktycznej implementacji filtracji cyfrowej z wykorzystaniem języka C

Przykładowy program w języku C, realizujący algorytm filtracji cyfrowej w sterowniku mikroprocesorowym, zostanie przedstawiony w postaci bardzo ogólnej, tzn. z pominięciem kwestii programowania konfiguracji mikroprocesora (ta jest specyficzna dla każdego typu). Szczegółowo zostanie przestawiony tylko sam algorytm realizacji filtracji.

Całość została zrealizowana na sterowniku mikroprocesorowym firmy Phillips LPC2138 rodziny ARM z rdzeniem ARM7TDMI-S z wykorzystaniem środowiska Keil µVision v4.03a. Sterownik ma wbudowane przetworniki A/C i C/A, unipolarne, 10-bitowe, o zakresie pomiarowym od 0 do 3,3 V i czasie konwersji krótszym niż zakładany czas próbkowania T_p.

Procedurze filtracji został poddany sygnał wejściowy o częstotliwości 50 Hz, amplitudzie ok. 1,65 V oraz składowej stałej dokładnie 1,65 V (przetwornik A/C unipolarny!). Parametry napięciowe zostały dobrane tak, aby maksymalnie wykorzystać zakres pomiarowy unipolarnego przetwornika A/C. Sygnał składa się z dwóch harmonicznych: podstawowej oraz trzeciej o udziale 50% w stosunku do podstawowej. Nie występują przesunięcia fazowe w harmonicznych.

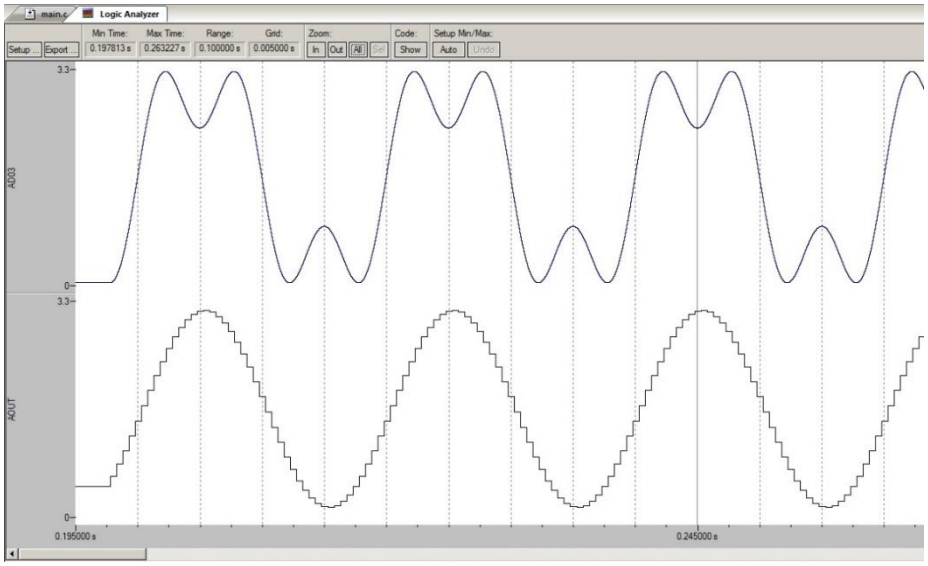
```
// Algorytm filtracji cyfrowej sygnału
#include <math.h>    // konieczne, bo używana funkcja kosinus do liczenia współczynników
                     // okna filtracyjnego
void tc0 (void) __irq; // deklaracja podprogramu tc0 obsługi przerwania od timera0
// deklaracja zmiennych:
```

```
unsigned int AC;       // wartość napięcia odczytana z przetwornika A/C w bitach

double Tp=0.0005;   // czas próbkowania w [s] fp=2 kHz, wówczas cały okres przebiegu (50Hz)
    // mieści się w 40 próbkach (2000 Hz/50 Hz), zatem w celu zapewnienia filtracji
    // pełnookresowej należy przyjąć N=40.
unsigned int N=40;    // długość okna filtracyjnego
double akos[40];      // tablica współczynników okna kosinus
double bufor[40];     // tablica, gdzie zapisywane dane pomiarowe (znormalizowane do
                      //  wartości z przedziału <-1;+1>)
double sygf;          // próbka sygnału po filtracji
unsigned int CA;      // próbka sygnału  po filtracji znormalizowana do przedziału
                      // <0;  1023> - podawana bezpośrednio na C/A
unsigned int i;       // indeks używany w pętli for
unsigned int ind;     // indeks wskazujący kolejny nr próbki przy algorytmie filtracji
unsigned int wsk=0;  // indeks wskazujący bieżące miejsce zapisu wyniku pomiaru do tablicy
                      // bufor (musi być zawsze w przedziale  <0; N-1>)
// Program główny
int main (void){
 // obliczenie off-line współczynników okna kosinus o długości N
 double pi=3.1415926;          // wartość  liczby pi
 for(i=0;i<N;i++)
  akos[i]=2./N*cos(2*pi*i/N);  // obliczanie kolejnego współczynnika okna kosinus
 // konfiguracja timera (odliczanie czasu Tp)
 // konfiguracja przerwania od timera
 // konfiguracja przetwornika A/C
 // konfiguracja przetwornika C/A
 // start timera
 while (1);     // niekończąca się pętla - oczekiwanie na przerwanie
}
// Procedura obsługi przerwania od timera
void tc0 (void) __irq  {
 // oczekiwanie na koniec konwersji przetwornika A/C
 // odczytanie przetwornika A/C i zapisanie wyniku konwersji do zmiennej AC
 bufor[wsk]=AC/512.0-1;   // zapisywanie tablicy kolejnymi danymi pomiarowymi
                          // (znormalizowane wartości z przedziału <-1;+1>)
 // filtracja cyfrowa oknem kosinus o długości N
 sygf=0;                  // wstępne zerowanie
 for(i=0;i<N;i++){
  ind=((wsk-i)+N)%N;      // zapewnia zawsze nr elementu tablicy bufor odnoszący się do
                          // poprzedniej wartości pomiaru
  sygf+=bufor[ind]*akos[i];
  }
 CA = (sygf+1)*512;       // próbka sygnału po filtracji znormalizowana do przedziału
                          //  <0;  1023>
// ograniczenie wartości bitowej do zakresu od 0 do 2^l_bit-1, (dla przetwornika 10-bit. do 1023)
 if (CA <0)
```

```
  CA=0;
 if (CA >1023)
  CA=1023;
 // wysłanie wartości CA na przetwornik C/A
 wsk++;                  // kolejny element tablicy bufor
 wsk=wsk%N;              //  zapewnienie zakresu  zmian wsk w przedziale <0; N-1>
    // odpowiada sprawdzeniu czy wsk nie przekracza N, jeżeli tak, to wsk zostaje wyzerowane
}
```

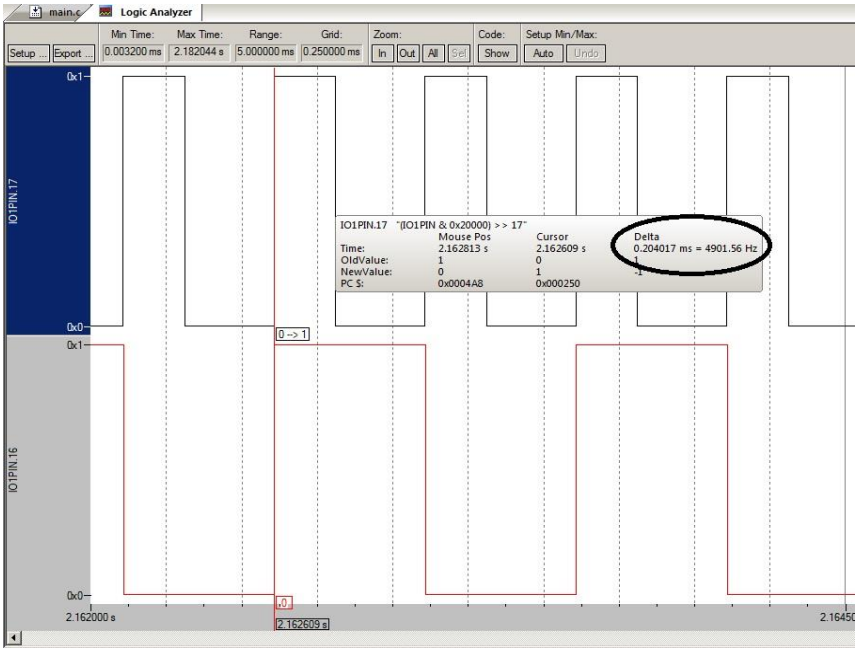
Na rysunku 7.2 widoczny jest praktyczny efekt działania filtracji cyfrowej wg powyższego algorytmu. Wykres górny, to sygnał wejściowy, obserwowany bezpośrednio na przetworniku A/C, wykres dolny to sygnał po filtracji cyfrowej obserwowany wprost na przetworniku C/A.



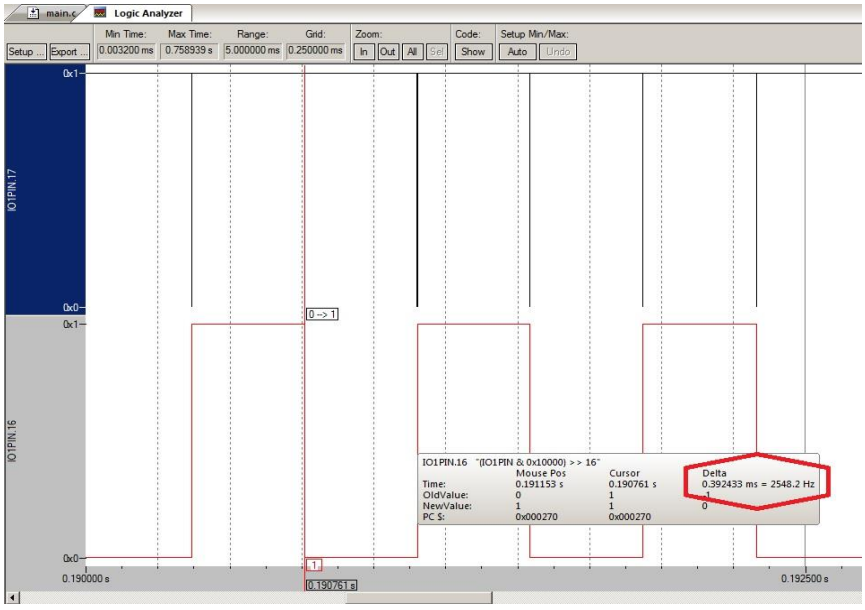
Rys. 7.2. Praktyczny efekt działania filtracji cyfrowej

Wybrano częstotliwość próbkowania f_p = 2 kHz. Ponieważ filtrowany sygnał ma częstotliwość 50 Hz, zatem aby zapewnić filtrację pełnookresową, należy wybrać okno filtru o długości N = 40 (2000 Hz/50 Hz). Warto zwrócić uwagę, że np. podwojenie częstotliwości próbkowania wymusza podwojenie wartości N, zatem także podwojenie czasu obliczeń filtracji. Należy więc przy każdej zmianie częstotliwości próbkowania kontrolować czas trwania procedury obsługi przerwania (sumaryczny czas wykonywania wszelkich instrukcji). Na rysunku 7.3, na wykresie górnym, widoczny jest praktyczny pomiar czasu trwania procedury obsługi przerwania wynoszący ok. 0,2 ms. Każde zbocze wykresu dolnego to rozpoczęcie procedury przerwania, która jest wywoływana co 0,5 ms. Na podstawie analizy tych czasów można podjąć błędną decyzję o możliwości

podwojenia częstotliwości próbkowania. Jednak wtedy, aby filtracja była pełnookresowa, należy podwoić także wartość *N*.



Rys. 7.3. Praktyczny pomiar czasu trwania procedury obsługi przerwania



Rys. 7.4. Praktyczny pomiar czasu trwania procedury obsługi przerwania przy zwiększonej dwukrotnie częstotliwości próbkowania

Na rysunku 7.4 na wykresie górnym, widoczny jest praktyczny pomiar czasu trwania procedury obsługi przerwania przy zwiększonej dwukrotnie (do wartości $f_p = 4$ kHz) częstotliwości próbkowania. Zatem wartość N musi wynosić teraz 80 (4000 Hz/50 Hz). Czas wykonywania wszystkich instrukcji (w tym procedury filtracji) wynosi ok. 0,39 ms, zatem znacznie przekracza okres próbkowania (T_p = 0,25 ms). Ponieważ kolejne przerwanie nie może wystąpić zanim zakończy się obsługa poprzedniego, zatem następuje ze znacznym opóźnieniem, co widoczne jest na dolnym wykresie rys. 7.4.

7.2. Przykład praktycznej implementacji filtracji cyfrowej z wykorzystaniem programowalnego sterownika logicznego PLC

W podrozdziale tym zostaną zaprezentowane dwa przykładowe programy realizujące filtrację cyfrową. Jeden w języku graficznym Ladder, drugi w języku tekstowym SCL. Podobnie jak to było poprzednio, nie ma potrzeby konfiguracji sterownika PLC, za wyjątkiem przerwania czasowego (*Cyclic Interrupt*), którego konfiguracja ogranicza się do podania czasu występowania przerwania.

Wykorzystany został sterownik firmy Siemens S7 1214C DC/DC/DC oraz środowisko programowe TiA Portal V15. Zastosowano dodatkowy moduł zewnętrzny wejść i wyjść analogowych SM 1234, zawierający cztery przetworniki A/C i dwa przetworniki C/A, 14-bitowe, bipolarne, o zakresie pracy od –10 V do +10 V. Zakresowi napięcia +10 V odpowiada wartość 27 648 w tzw. standardzie *S7 analog format*. Dla –10 V jest to odpowiednio –27 648.

Z uwagi na fakt, że sterownik PLC ma zdecydowanie gorsze parametry czasowe niż standardowy sterownik mikroprocesorowy, przyjęto częstotliwość próbkowania $f_p = 20$ Hz i częstotliwość sygnału poddawanego filtracji 1 Hz. Zatem, żeby filtracja była pełnookresowa należy przyjąć $N = 20$ (20 Hz/1 Hz).

Program główny *Main* nie zawiera żadnej instrukcji. Konfiguracja przerwania czasowego (*Cyclic Interrupt*), odbywa się poza programem, w warstwie sprzętowej, na etapie konfiguracji i ogranicza się do podania czasu występowania przerwania. W bloku *Startup* zamieszczone są obliczenia współczynników okna kosinusoidalnego, wartość liczby π = 3,1415926, początkowe wartości zmiennej *wsk* = 0 oraz zmiennej $N = 20$ (filtracja pełnookresowa).

Dodatkowo w bloku *Data Block* o nazwie *Dane* zadeklarowane zostały dwie tablice, dwudziestoelementowe, typu *Real*:

– *akos* – tablica zawierająca współczynniki filtru kosinus,

– *bufor* – tablica, gdzie zapisywane będą kolejne wyniki pomiarów.

Zadanie realizowane jest wg algorytmu blokowego na rys. 7.1. Wszystkie używane zmienne są deklarowane w oknie *Tag Table* (tabela 7.1).

Tabela 7.1. Wykaz zmiennych (Tagów) stosowanych w programie realizującym filtrację cyfrową z wykorzystaniem sterownika PLC

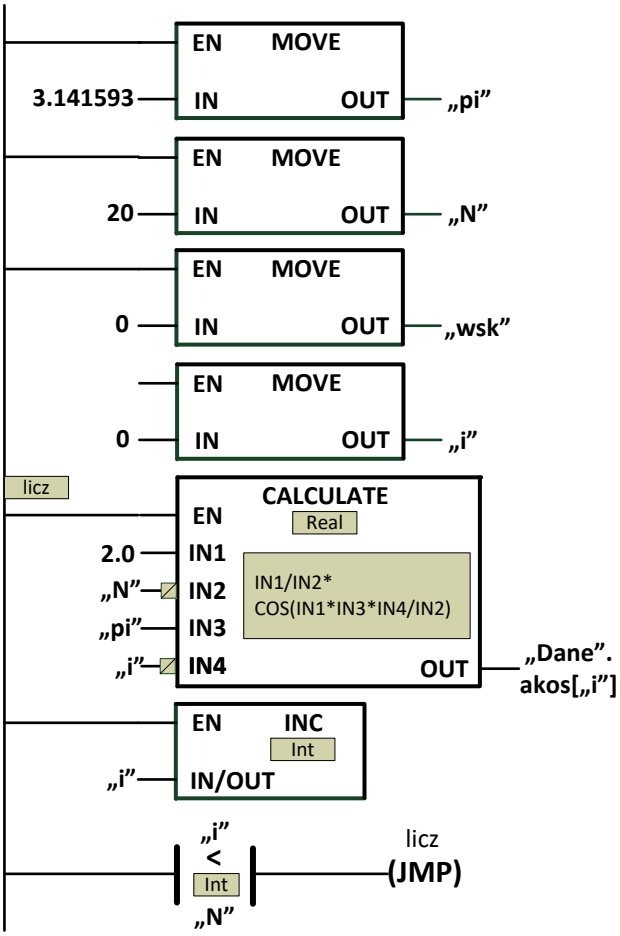
Nazwa Tagu	Typ zmiennej	Opis
N	Int	długość okna filtracyjnego
AC	Int	wartość próbki napięcia sygnału wejściowego w standardzie *S7 analog format* (odczyt przetwornika A/C)
sygf	Real	wartość próbki sygnału po filtracji
wsk	Int	indeks wskazujący kolejny element tablicy *bufor* przy zapisie wyników pomiaru
ind	Int	indeks wskazujący kolejne poprzednie wyniki pomiarów zapisane uprzednio w tablicy *bufor*
CA_real		wartość próbki sygnału po filtracji w standardzie *S7 analog format* – zmienna typu *Real*
CA	Int	wartość próbki sygnału po filtracji w standardzie *S7 analog format* – zmienna typu *Int (*wysyłana na przetwornik C/A)
pi	Real	wartość liczby π
i	Int	indeks do realizacji pętli *for*

7.2.1. Realizacja z wykorzystaniem języka Ladder

W podrozdziale tym zostanie szczegółowo omówiona realizacja programu w języku Ladder.

Na rysunku 7.5 przedstawiono zawartość bloku *Startup*, gdzie m.in. zamieszczone są obliczenia współczynników okna kosinusoidalnego. Do tego celu najwygodniejsza jest funkcja *for*, która niestety nie występuje języku Ladder. Została zastąpiona parą instrukcji *Label* (etykieta, w tym przypadku „*licz*") i *Jmp* (skocz). Instrukcja *Calculate* tym razem została użyta w wersji działającej na liczbach typu *Real*, ale z opcją używania na wejściu liczb typu *Integer* (dostępne w nowszych wersjach oprogramowania). Przed wykonaniem operacji algebraicznych następuje wówczas automatyczna konwersja liczby *Integer* na typ *Real*. Na wejściu bloku *Calculate* jest to zaznaczone w odpowiedni sposób, widoczny na rys. 7.5 przy wejściach *IN2* i *IN4*.

W przypadku filtrów o niezbyt dużej wartości długości okna filtracyjnego *N* możliwe jest, w miejsce pętli ograniczonej instrukcjami *Label* i *Jmp*, *N*-krotne powtórzenie instrukcji *Calculate*. Oczywiście wtedy wejście *IN4* każdej instrukcji *Calculate* przyjmuje kolejne wartości z zakresu < 0; N – 1 >. Zmienna *i* może wobec tego zostać usunięta.



Rys. 7.5. Zadawanie parametrów początkowych i obliczanie współczynników filtru kosinusoidalnego w bloku *Startup*

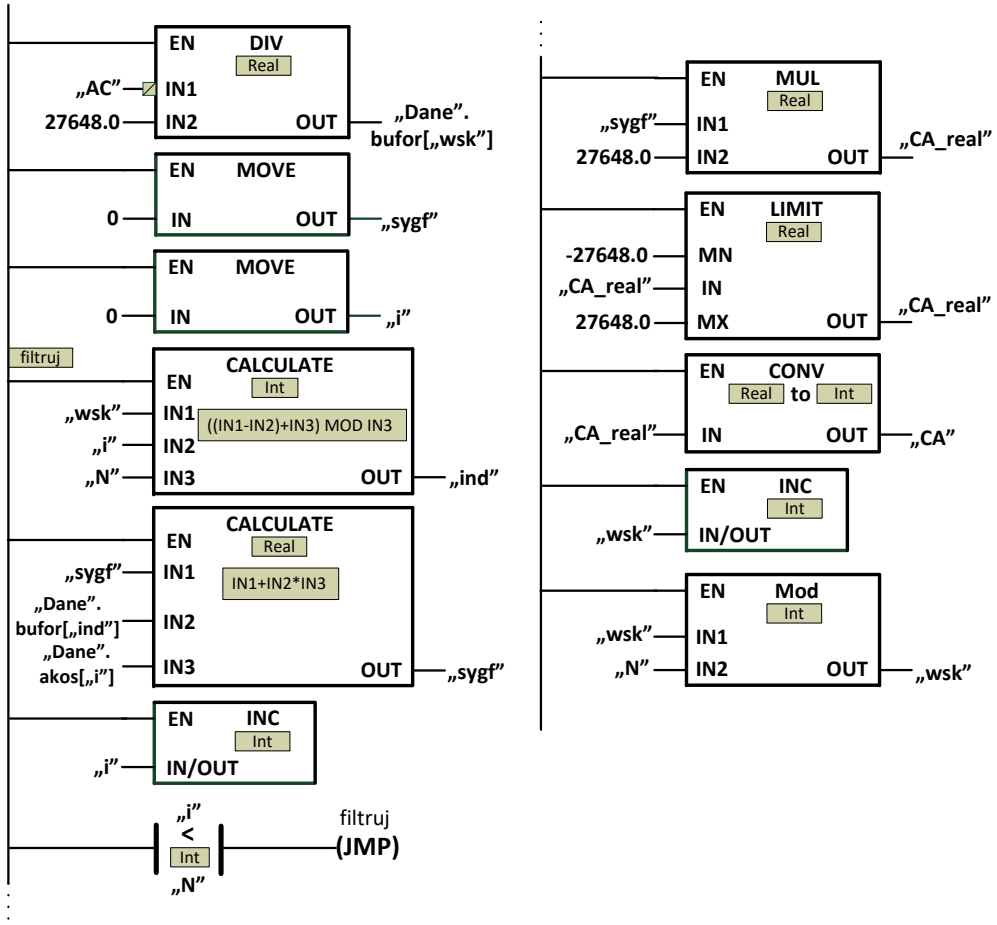
W bloku *Cyclic Interrupt* (rys. 7.6), wywoływanym co 50 ms, zamieszczona została programowa realizacja filtracji cyfrowej.

Wartość odczytana z przetwornika A/C (typu *Integer*), po zamianie na typ *Real* (automatycznie w bloku *DIV*), zostaje podzielona przez liczbę 27 648,0 w celu normalizacji do przedziału <–1; +1> oraz zapisana do tablicy *bufor*.

Realizacja pętli filtracyjnej odbywa się między etykietą „*filtruj*” a instrukcją *JMP* „*filtruj*”. Po wykonaniu w pętli *N* sumowań iloczynów próbek sygnału ze współczynnikami filtru, następuje normalizacja wyniku do standardu *S7 analog format* (z ograniczeniem do przedziału <–27 648; +27 648>, zamiana z typu *Real* na *Integer* i wysłanie do przetwornika C/A.

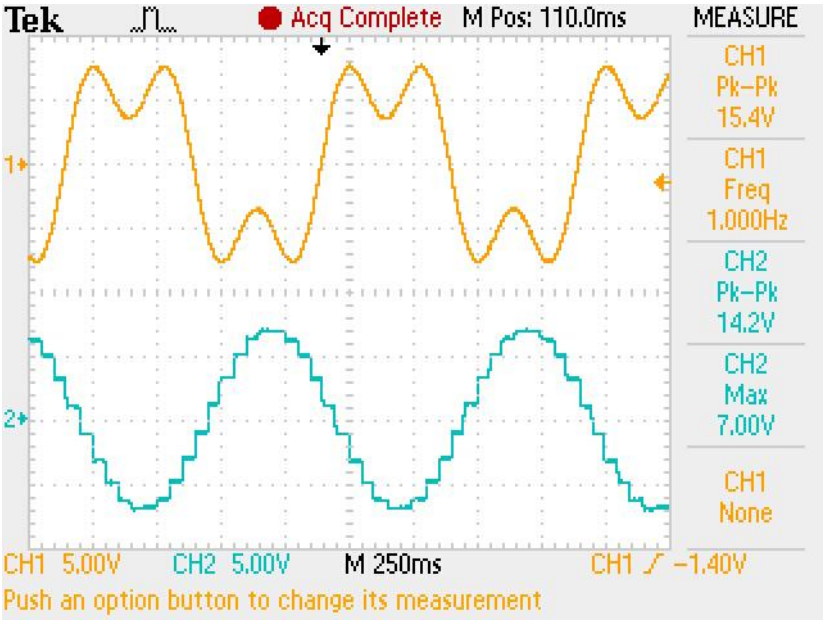
Dwie ostatnie instrukcje realizują przejście do kolejnego elementu tablicy *bufor*, wraz z zapewnieniem zakresu zmian *wsk* w przedziale <0; *N*–1>. Działanie

instrukcji *MOD* (reszta z dzielenia) odpowiada sprawdzeniu czy zmienna *wsk* nie przekracza *N*, jeżeli tak, to zostaje wyzerowana.



Rys. 7.6. Realizacja algorytmu filtracji cyfrowej w bloku *Cyclic Interrupt*

Na rysunku 7.7 widoczne jest zarejestrowane na oscyloskopie działanie filtracji cyfrowej. Jako sygnał wejściowy wykorzystano sygnał uzyskany z karty przetworników C/A firmy National Instruments znajdującej się w komputerze PC. Sygnał o częstotliwości 1 Hz składa się z dwóch harmonicznych: podstawowej o amplitudzie 7 V oraz trzeciej o udziale 50% w stosunku do podstawowej. Górny wykres to sygnał wejściowy. Wykres dolny przedstawia sygnał „odfiltrowany”, czyli efekt działania filtracji.



Rys. 7.7. Efekt działania filtracji cyfrowej

7.2.2. Realizacja z wykorzystaniem języka SCL

Realizacja algorytmu filtracji cyfrowej sygnału w języku tekstowym SCL, w którym występuje pętla *for*, jest znacznie prostsza niż w języku Ladder. Poniżej fragment programu bloku *Startup*, zawierający zadawanie parametrów początkowych elementu nieliniowego.

```
// Realizacja algorytmu filtracji cyfrowej
//STARTUP  - zadawanie wartości początkowych
"pi" := 3.1415926; // wartość liczby pi
"N" := 20;     // długość okna filtracyjnego (przy założonej częstotliwości próbkowania fp=20Hz
               //  i częstotliwości sygnału poddawanego filtracji równej 1Hz).
"wsk" := 0;    // początkowa wartość indeksu wskazującego kolejny element tablicy bufor
               //  przy zapisie wyników pomiaru
// obliczenie off-line współczynników okna kosinus o długości N
FOR "i":= 0 TO ("N"-1) DO
  "Dane".akos["i"] := 2.0 / "N" * COS(2.0 * "pi" * "i" / "N");
END_FOR;
```

Program główny *Main* nie zawiera żadnej instrukcji. Obsługa przerwania cyklicznego wywoływanego z czasem T_p = 50 ms, widoczna jest poniżej.

```
// Realizacja algorytmu filtracji cyfrowej
// filtracja cyfrowa oknem kosinus o długości N
// CYCLIC INTERRUPT - przerwanie cykliczne co 50 ms
// współczynniki okna kosinusoidalnego obliczone w bloku STARTUP i zapisane do tablicy akos
"Dane".bufor["wsk"] := INT_TO_REAL("AC")/27648.0;     // odczyt z A/C, konwersja z Int na
            // Real, normalizacja do przedziału <-1; +1> i zapis do tablicy bufor pod adres wsk
"sygf" := 0;     // wstępne zerowanie wartość próbki sygnału po filtracji
FOR "i" := 0 TO ("N" - 1) DO
  "ind":=(("wsk"-"i")+"N") MOD "N";   // zapewnia zawsze nr elementu tablicy bufor odnoszący
                                      // się do poprzedniej wartości pomiaru
"sygf":="sygf"+"Dane".bufor["ind"]*"Dane".akos["i"];          // kolejny krok filtracji
END_FOR;
"sygf":="sygf"*27648.0;              // normalizacja do standardu S7 analog format
"sygf":= LIMIT(MN := -27648.0, IN := "CA_real", MX := 27648.0);     // ograniczenie wartości
                                                    //  do przedziału  <-27648.0; +27648.0>
"CA":= REAL_TO_INT("syg_f");  // konwersja z Real na Int i wpis na przetwornik C/A
"wsk" := "wsk" + 1;                  // kolejny element tablicy bufor
"wsk" := "wsk" MOD "N";          //zapewnienie zakresu zmian wsk w przedziale <0; N-1>
         // odpowiada sprawdzeniu czy wsk nie przekracza N, jeżeli tak, to zostaje wyzerowane
```

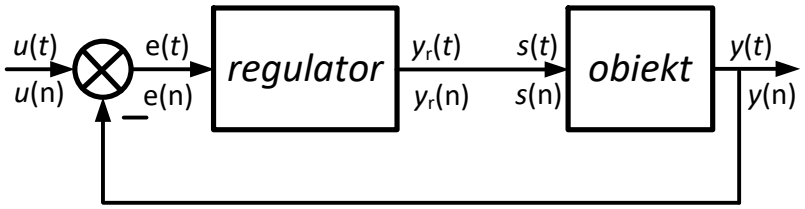
8. Regulator PID w układzie sterowania

W układach regulacji automatycznej oprócz samego obiektu (czy też jego modelu) bardzo ważną rolę pełni korektor (regulator) [5], [6], [8]. Z praktycznego punktu widzenia nie ma bowiem obiektów, które spełniałyby samodzielnie założenia regulacji, czyli uzyskanie żądanych właściwości statycznych i dynamicznych projektowanego układu. Rolą regulatora jest takie wytworzenie sygnału sterującego obiektem, aby te właściwości uzyskać. Na rysunku 8.1 znajduje się schemat blokowy typowego układu regulacji automatycznej, gdzie (w dziedzinie czasu):

$u(t)$ – sygnał wejściowy (zadany),
$e(t)$ – sygnał błędu,
$s(t)$ – sygnał sterujący, równy sygnałowi wyjściowemu $y_r(t)$ z regulatora PID,
$y(t)$ – sygnał wyjściowy (otrzymany jako efekt regulacji),

oraz odpowiednio w dziedzinie dyskretnej n:

$u(n)$, $e(n)$, $s(n)$, $y_r(n)$, $y(n)$.



Rys. 8.1. Typowy układ regulacji automatycznej

Najpopularniejszym typem regulatora stosowanym w praktyce w układach regulacji automatycznej jest regulator PID, czyli proporcjonalno (P)-całkująco (C)-różniczkujący (D). Jego działanie w dziedzinie czasu określone jest wzorem ogólnym:

$$y_{PID}(t) = y_P(t) + y_I(t) + y_D(t) \tag{8.1}$$

gdzie:

$y_P(t) = k_p e(t)$ – człon proporcjonalny,

$y_I(t) = k_p \left[\frac{1}{T_I} \int_0^t e(\tau) d\tau + y_I(0) \right]$ – człon całkujący,

$y_D(t) = k_p T_D \frac{de(t)}{dt}$ – człon różniczkujący,

gdzie:

k_p – współczynnik wzmocnienia, $k_p > 0$,
$e(t)$ – wartość błędu w chwili t,

T_I – czas całkowania, zwany inaczej czasem zdwojenia,

T_D – czas różniczkowania, zwany inaczej czasem wyprzedzenia,

$y_I(0)$ – wartość początkowa członu całkującego.

Zatem ostatecznie:

$$y_{PID}(t) = k_p \left[e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau + y_I(0) + T_D \frac{de(t)}{dt} \right] \tag{8.2}$$

Wzór (8.2) to wzór na regulator idealny, niemożliwy do praktycznej realizacji w tej postaci. W celu praktycznego zastosowania cyfrowego regulatora PID, można, w chwilach próbkowania, zastąpić całkę sumowaniem (wzór Eulera), a różniczkowanie różnicą pierwszego rzędu. Wzór (8.2) przyjmie wówczas postać:

$$y_{PID}(n) = k_p \left[e(n) + \frac{T_p}{T_I} \sum_{i=1}^{n} e(i) + y_I(0) + \frac{T_D}{T_p} \big(e(n) - e(n-1) \big) \right] \tag{8.3}$$

gdzie:

n – numer próbki,

T_p – okres próbkowania.

Wzór (8.3) jest dalej niewygodny do praktycznej realizacji, wymaga bowiem znajomości wartości początkowej $y_I(0)$ członu całkującego oraz całej „historii" sumowania (całkowania). Można jednak zauważyć, że:

$$y_I(1) = \frac{T_p}{T_I} e(1) + y_I(0)$$

$$y_I(2) = \frac{T_p}{T_I} \big[e(1) + e(2) \big] + y_I(0) = \frac{T_p}{T_I} e(2) + y_I(1)$$

$$y_I(3) = \frac{T_p}{T_I} \big[e(1) + e(2) + e(3) \big] + y_I(0) = \frac{T_p}{T_I} e(3) + y_I(2)$$

$$\vdots$$

$$y_I(n) = \frac{T_p}{T_I} e(n) + y_I(n-1) \tag{8.4}$$

Po uwzględnieniu powyższych zależności wzór (8.3) przyjmie postać:

$$y_{PID}(n) = k_p \left[e(n) + \frac{T_p}{T_I} e(n) + y_I(n-1) + \frac{T_D}{T_p} \big(e(n) - e(n-1) \big) \right] \tag{8.5}$$

Dodatkowo, jak widać na rys. 8.1:

$$E(n) = u(n) - y(n) \tag{8.6}$$

Zatem przy założeniu, że $u(n) = u(n-1)$, ostateczna postać wzoru na praktyczną realizację cyfrowego regulatora PID przyjmie postać:

$$y_{PID}(n) = \left\{ k_p \left[u(n) - y(n)\right] + \frac{T_p}{T_I}\left[u(n) - y(n)\right] + y_I(n-1) + \right.$$
$$\left. + \frac{T_D}{T_p}\left[y(n-1) - y(n)\right] \right\} \tag{8.7}$$

Niekiedy, przy założeniu, że:

$$K_P = k_p \qquad K_I = \frac{k_p T_p}{T_I} \qquad K_D = \frac{k_p T_D}{T_p},$$

gdzie:

K_P – wzmocnienie członu proporcjonalnego,

K_I – wzmocnienie członu całkującego,

K_D – wzmocnienie członu różniczkującego,

wzór (8.7) podawany jest w innej postaci:

$$y_{PID}(n) = K_P\left[u(n) - y(n)\right] + K_I\left[u(n) - y(n)\right] + y_I(n-1) +$$
$$+ K_D\left[y(n-1) - y(n)\right] \tag{8.8}$$

Powyższa postać wzoru bardzo dobrze nadaje się do numerycznej realizacji, co zostanie opisane w następnym rozdziale.

9. Praktyczna implementacja modelu regulatora PID w sterowniku mikroprocesorowym

W rozdziale tym zostanie zaprezentowana przykładowa realizacja modelu regulatora PID opisanego równaniem (8.8). W układach automatyki regulator nigdy nie występuje samodzielnie, tylko zawsze w układzie sterowania, zgodnie ze schematem na rys. 9.1. Bardzo ważne jest miejsce wstawienia regulatora na tym schemacie. Musi on zawsze występować przed obiektem, bowiem jego zadaniem jest, na podstawie sygnału błędu uzyskanego po sumatorze, wytworzenie takiego sygnału sterującego obiektem, aby uzyskać wymagane parametry działania układu regulacji.

Regulator PID oraz sumator zostały zaimplementowane na pierwszym sterowniku mikroprocesorowym. Obiekt został zaimplementowany na drugim sterowniku mikroprocesorowym. Sterowniki zostały połączone ze sobą tylko sygnałami analogowymi, zgodnie ze schematem na rys. 9.1. Taki bowiem sposób połączenia występuje w rzeczywistych układach sterowania.

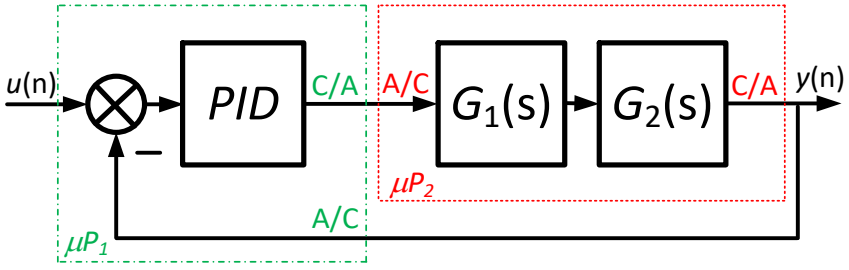
Obiektem regulacji jest szeregowe połączenie dwóch obiektów: pierwszego i drugiego rzędu o transmitancjach danych wzorami:

$$G_1(s) = \frac{k_1}{T_1 s + 1} \tag{9.1}$$

gdzie: $k_1 = 1$, $T_1 = 3$.

$$G_2(s) = \frac{k_2}{T_2^2 s^2 + 2n_2 T_2 s + 1} \tag{9.2}$$

gdzie: $k_2 = 1$, $T_2 = 3$, $n_2 = 0{,}3$.



Rys. 9.1. Analizowany układ regulacji automatycznej

Obiekt dany równaniami (9.1) i (9.2) został zaimplementowany na oddzielnym sterowniku mikroprocesorowym, zgodnie z opisem w rozdziale 3. W przypadku transmitancji $G_1(s)$, przejście z układu ciągłego do układu dyskretnego

zostało dokonane metodą residuów, a w przypadku transmitancji $G_2(s)$ – z zastosowaniem przekształcenia biliniowego.

Równania różnicowe dane są odpowiednio wzorami:

$$y_1(n) = k(1-A)u_1(n-1) + Ay_1(n-1) \tag{9.4}$$

gdzie $A = e^{-\frac{T_p}{T_1}}$.

$$y_2(n) = \left\{ \frac{1}{b} a\left[u_2(n) + 3u_2(n-1) + 3u_2(n-2) + u_2(n-3)\right] + \right.$$
$$\left. - cy_1(n-1) - dy_1(n-2) \right. \tag{9.5}$$

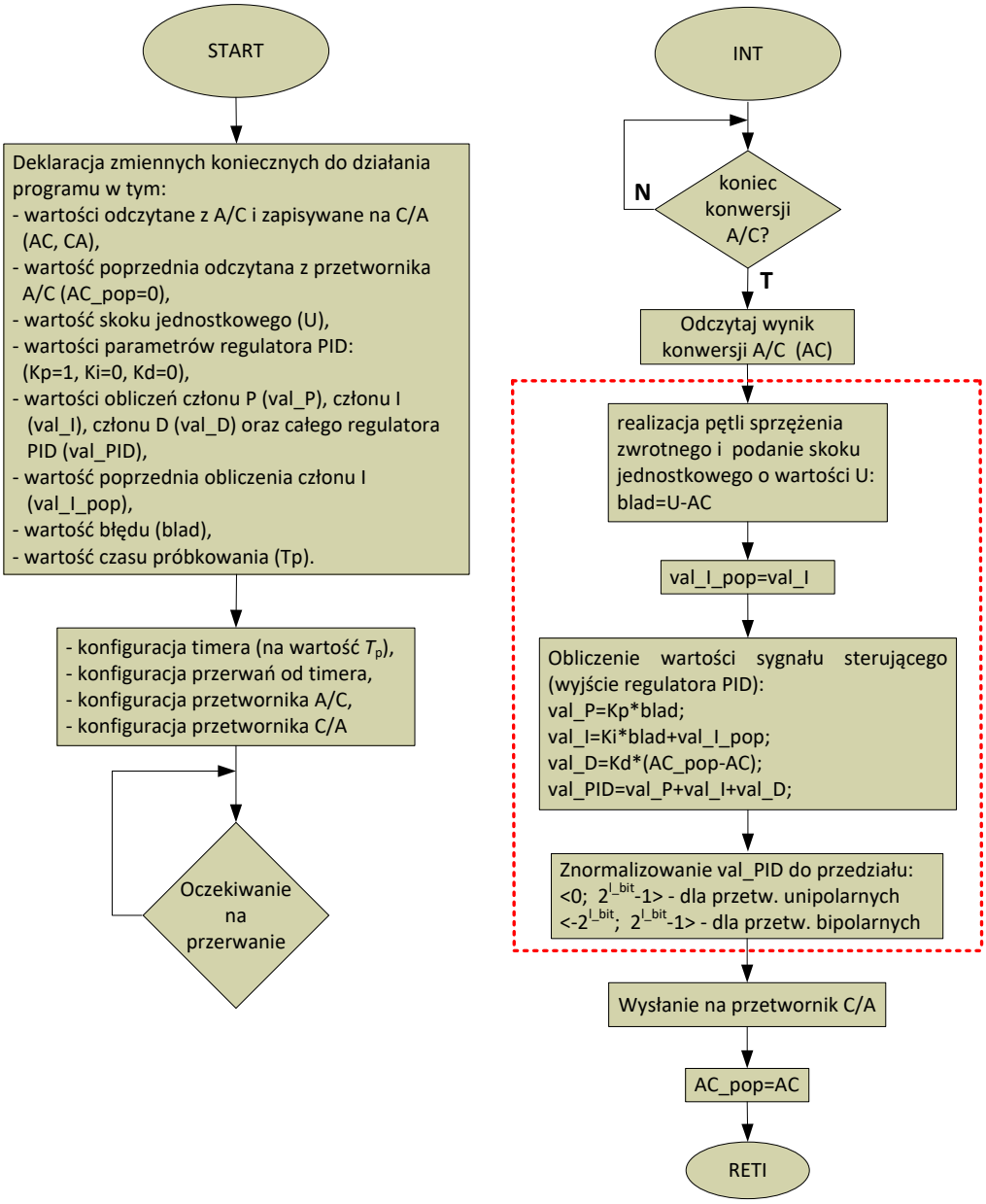
gdzie:

$$a = \frac{kT_p^3}{2},$$
$$b = 4T^2 + 4nTT_p + T_p^2,$$
$$c = -8T^2 + 2T_p^2,$$
$$d = 4T^2 - 4nTT_p + T_p^2.$$

Na rysunku 9.2 widoczny jest algorytm realizacji modelu regulatora PID w układzie sterowania. Po lewej stronie algorytmu znajdują się bloki deklaracji zmiennych oraz konfiguracji układów wejścia–wyjścia. Niektórym zmiennym przyporządkowane są wartości początkowe istotne dla obliczeń numerycznych. Ta część algorytmu wykonywana jest tylko jednorazowo po uruchomieniu programu.

Sam algorytm realizacji modelu regulatora PID w układzie sterowania wykonywany jest w procedurze obsługi przerwania od timera. Główna jego część zaznaczona jest na schemacie blokowym czerwoną ramką. Na wejście regulatora PID podawany jest sygnał *blad*, czyli różnica między wartością zadaną *U*, a wyjściem z obiektu. W ten sposób realizowany jest sumator i pętla sprzężenia zwrotnego oraz podanie skoku jednostkowego o wartości *U*, zgodnie z rys. 9.1. Każdy z członów regulatora obliczany jest oddzielne, na końcu całość jest sumowana. Przed czerwoną ramką wykonywana jest odczyt przetwornika A/C, a za nią zapis przetwornika C/A. Algorytm zakłada użycie przetworników A/C i C/A o rozdzielczości bitów zapisanej w zmiennej *l_bit*. W przypadku przetworników unipolarnych odczyt (w przypadku przetwornika A/C) i zapis (w przypadku przetwornika A/C) będą w przedziale <0; 2^{l_bit}–1>, natomiast w przypadku przetworników bipolarnych w przedziale <–2^{l_bit}; 2^{l_bit}–1>.

Przyjęto czas próbkowania T_p = 0,1 s, bowiem w rzeczywistych układach regulacji rzadko występują czasy krótsze.



Rys. 9.2. Algorytm realizacji modelu regulatora PID w układzie sterowania

9.1. Przykład praktycznej implementacji modelu regulatora PID z wykorzystaniem języka C

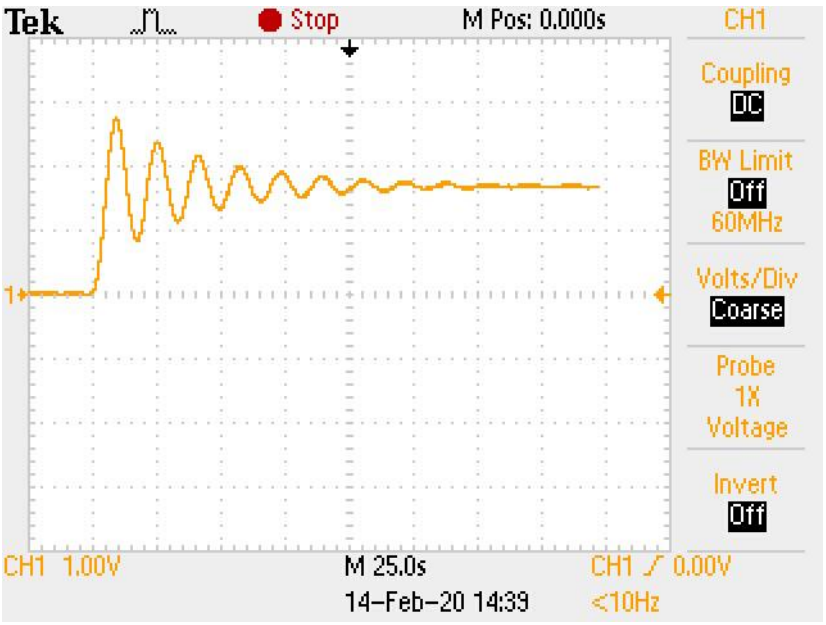
Przykładowy program w języku C realizujący działanie układu sterowania z wykorzystaniem modelu regulatora PID w sterowniku mikroprocesorowym, zostanie przedstawiony w postaci bardzo ogólnej, tzn. z pominięciem kwestii programowania konfiguracji mikroprocesora (ta jest specyficzna dla każdego typu). Szczegółowo zostanie przestawiony tylko sam algorytm realizacji regulatora. Zakładamy, że przetworniki A/C i C/A są unipolarne, o liczbie bitów $l\_bit = 10$ i czasie konwersji krótszym niż czas próbkowania T_p.

```
// Modelowanie układu regulacji z wykorzystaniem regulatora PID
void tc0 (void) __irq;        // deklaracja podprogramu tc0 obsługi przerwania od timera
unsigned int U=1023;          // napięcie skoku jednostkowego w bitach (odpowiada 3,3V)
unsigned int AC;              // wartość odczytana z przetwornika A/C w bitach
unsigned int AC_pop=0;        // wartość poprzednia odczytana z przetwornika A/C w bitach
float Tp=0.1;                 // okres próbkowania [s]
float Kp=1.0, Ki=0.0, Kd=0.0;     // początkowe współczynniki regulatora PID
                                  // co dla takich wartości praktycznie oznacza brak regulatora
unsigned int CA;   // wartość wyjścia układu regulacji w bitach, podawana na przetwornik C/A
float blad;           // wartość błędu (po sumatorze)
float val_P;          // wartość obliczenia członu P
float val_I=0;        // wartość obliczenia członu I
float val_I_pop;      // poprzednia wartość obliczenia członu I
float val_D;          // wartość obliczenia członu D
float val_PID;        // wartość obliczenia PID

// Program główny
int main (void){
  // konfiguracja timera (odliczanie czasu Tp)
  // konfiguracja przerwania od timera
  // konfiguracja przetwornika A/C
  // konfiguracja przetwornika C/A
  // start timera
  while (1);                  // niekończąca się pętla - oczekiwanie na przerwanie
}
// Procedura obsługi przerwania od timera
void tc0 (void) __irq {
  // oczekiwanie na koniec konwersji przetwornika A/C
  // odczytanie przetwornika A/C i zapisanie wyniku konwersji do zmiennej AC
  blad=U-AC;        // wartość błędu; wartość podawana na regulator PID jako różnica wartości
       // zadanej i wyjścia z obiektu; w ten sposób realizowana jest pętla sprzężenia zwrotnego
      // i podanie skoku jednostkowego o wartości U
  val_P=Kp*blad;                  // wartość obliczenia członu P
  val_I_pop=val_I;                // poprzednia wartość obliczenia członu I
```

```
 val_I=Ki*blad+val_I_pop;        // wartość obliczenia członu I
 val_D=Kd*(AC_pop-AC);           // wartość obliczenia członu D
 val_PID=val_P+val_I+val_D;      // wartość na wyjściu regulatora PID
  if(val_PID>1023)
   CA=1023;          // ograniczenie do wartości maksymalnej
 else
   CA=val_PID;
 // wysłanie wartości CA na przetwornik C/A
 AC_pop=AC;       // zapamiętanie bieżącej wartości pomiaru napięcia
                  //   jako wartość poprzednią
}
```

Na rysunku 9.3 widoczna jest zarejestrowana na oscyloskopie odpowiedź na skok jednostkowy (3,3 V) w układzie jak na rys. 9.1, przy parametrach regulatora: $K_P = 1$, $K_I = 0$, $K_D = 0$, co praktycznie oznacza brak regulatora PID. Całość została zrealizowana na sterowniku mikroprocesorowym firmy Phillips LPC2138 rodziny ARM z rdzeniem ARM7TDMI-S z wykorzystaniem środowiska Keil µVision v4.03a. Sterownik ma wbudowane przetworniki A/C i C/A unipolarne, 10-bitowe, o zakresie pomiarowym od 0 do 3,3 V. Jak widać na rysunku układ zachowuje się zgodnie z oczekiwaniami. Wartość ustalona to 50% wartości zadanej, czyli napięciowo 1,65 V, a bitowo 512.



Rys. 9.3. Odpowiedź na skok jednostkowy (3,3V) w układzie jak na rys. 8.2, przy braku regulatora PID

9.2. Przykład praktycznej implementacji modelu regulatora PID z wykorzystaniem programowalnego sterownika logicznego PLC

W podrozdziale tym zostaną zaprezentowane dwa przykładowe programy realizujące model regulatora PID. Jeden w języku graficznym Ladder, drugi w języku tekstowym SCL. Konfiguracja sterownika PLC ogranicza się do podania czasu występowania przerwania.

Zadanie realizowane jest wg algorytmu blokowego na rys. 9.2. Wszystkie używane zmienne są deklarowane w oknie *Tag Table* (tabela 9.1).

Tabela 9.1. Wykaz zmiennych (Tagów) stosowanych w programie realizującym model regulatora PID w układzie sterowania z wykorzystaniem sterownika PLC

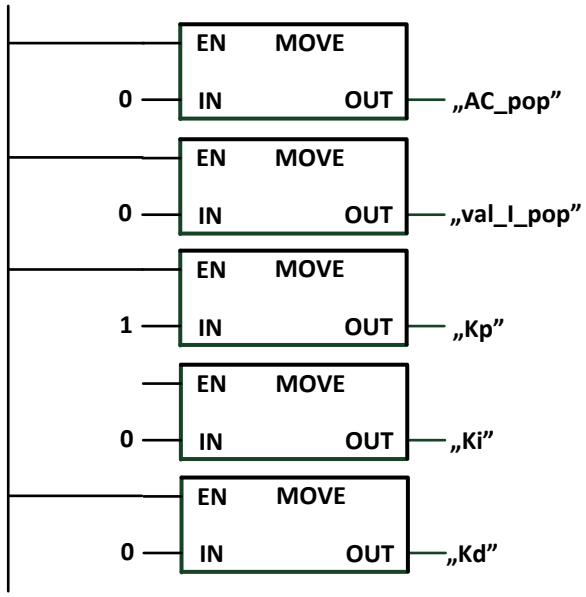
Nazwa Tagu	Typ zmiennej	Opis
U	Int	wartość skoku jednostkowego w standardzie *S7 analog format*
AC	Int	wyjście układu sterowania w standardzie *S7 analog format* (odczyt przetwornika A/C)
AC_pop	Int	wyjście układu sterowania w standardzie *S7 analog format* (odczyt przetwornika A/C) – wartość poprzednia
CA	Int	wyjście regulatora PID w standardzie *S7 analog format* (wysyłane na przetwornik C/A)
blad	Real	wartość błędu; wartość podawana na regulator PID jako różnica wartości zadanej i wyjścia z obiektu; w ten sposób realizowana jest pętla sprzężenia zwrotnego i podanie skoku jednostkowego o wartości *U*
val_P	Real	wartość obliczeń członu P
val_I	Real	wartość obliczeń członu I
val_I_pop	Real	wartość poprzednia obliczeń członu I
val_D	Real	wartość obliczeń członu D
val_PID	Real	wartość na wyjściu regulatora PID
Kp	Real	wartość wzmocnienia członu P
Ki	Real	wartość wzmocnienia członu I
Kd	Real	wartość wzmocnienia członu D

W celu realizacji modelu regulatora PID został wykorzystany sterownik firmy Siemens S7 1214C DC/DC/DC oraz środowisko programowe TiA Portal V15. Do przetwarzania sygnałów analogowych zastosowano dodatkowy moduł zewnętrzny wejść i wyjść analogowych SM 1234, zawierający cztery niezależne

przetworniki A/C i dwa niezależne przetworniki C/A, 14-bitowe, bipolarne, o zakresie pracy od –10 V do +10 V. Zakresowi napięcia +10 V odpowiada wartość 27 648 w tzw. standardzie *S7 analog format*. Dla –10 V jest to odpowiednio –27 648. Zastosowanie zewnętrznego modułu analogowego było konieczne, ponieważ sterownik ma wbudowany tylko przetwornik A/C unipolarny, wykorzystanie którego mogłoby utrudnić lub wręcz uniemożliwić proces regulacji.

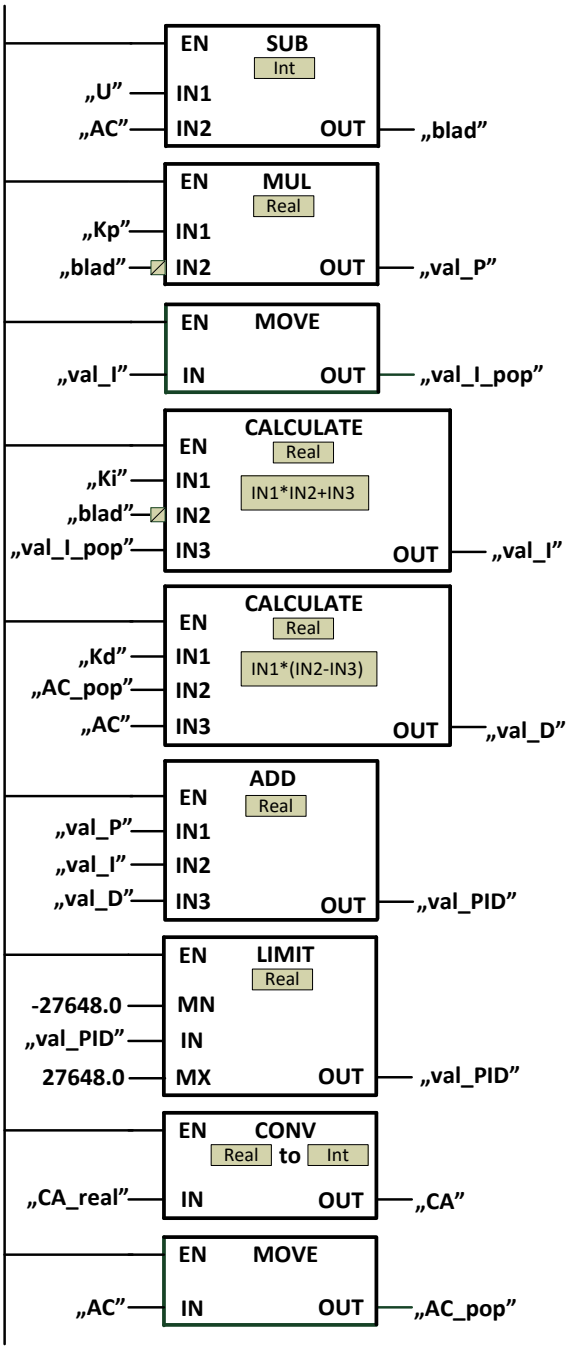
9.2.1. Realizacja z wykorzystaniem języka Ladder

W podrozdziale tym zostanie szczegółowo omówiona realizacja programu w języku Ladder. Na rysunku 9.4 przedstawiono zawartość bloku *Startup*, gdzie zamieszczone są zerowe wartości zmiennych *AC_pop* i *val_I_pop* oraz początkowe wartości wzmocnień $K_P = 1$, $K_I = 0$, $K_D = 0$, co, z praktycznego punktu widzenia, oznacza brak regulatora PID. Program główny *Main* nie zawiera żadnej instrukcji.



Rys. 9.4. Zadawanie parametrów początkowych w bloku *Startup*

W bloku *Cyclic Interrupt* (rys. 9.5), wywoływanym co 100 ms, zamieszczona została programowa realizacja modelu regulatora PID w układzie sterowania. Bardzo ważną rolę pełni tu pierwsza instrukcja obliczająca wartość błędu, czyli wartość podawaną na regulator PID jako różnica wartości zadanej i wyjścia z obiektu. W ten sposób realizowana jest pętla sprzężenia zwrotnego i podanie skoku jednostkowego o wartości *U*.

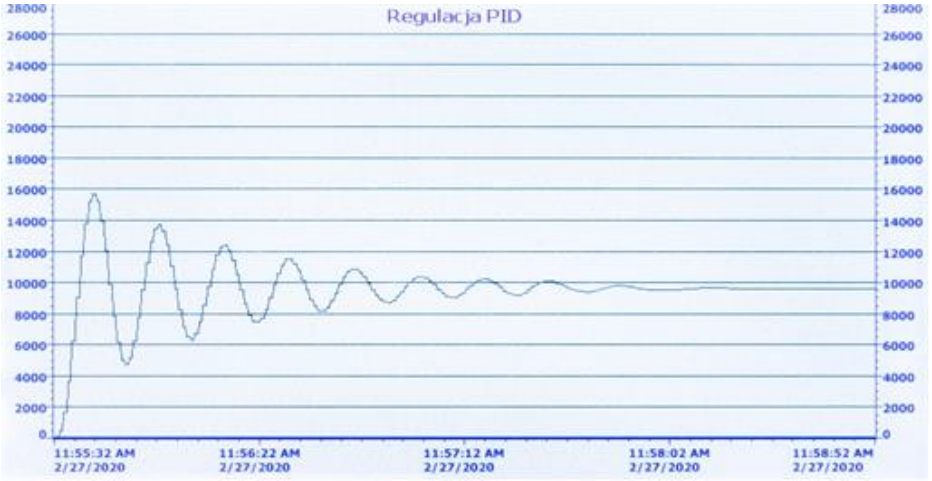


Rys. 9.5. Numeryczna realizacja modelu regulatora PID w układzie sterowania w bloku *Cyclic Interrupt*

Wartość wyjściowa regulatora PID obliczana jest zgodnie z równaniem (8.8). Poszczególne wartości członów regulatora liczone są w oddzielnych instrukcjach, na koniec całość jest sumowana i po ograniczeniu do przedziału <–27 648.0; +27 648> oraz zamianie na typ *Int*, wysyłana na przetwornik C/A. Ostatnią instrukcją jest zapamiętanie wartości pomiaru napięcia, która w następnym cyklu pomiarowym będzie traktowana jako wartość poprzednia.

Na rysunku 9.6 widoczna jest (zarejestrowana na panelu operatorskim HMI) odpowiedź układu sterowania, jak na rys. 9.1, na skok jednostkowy o wartości 20 000 w standardzie *S7 analog format*, co odpowiada napięciu ok. 7,2 V. Wprowadzone parametry regulatora PID to $K_P = 1$, $K_I = 0$, $K_D = 0$, co praktycznie oznacza brak regulatora PID.

Jak widać układ zachowuje się zgodnie z oczekiwaniami. Wartość ustalona to 50% wartości zadanej, czyli napięciowo ok. 3,6 V (ok. 10 000 w standardzie *S7 analog format*).



Rys. 9.6. Odpowiedź na skok jednostkowy w układzie jak na rys. 9.1, przy braku regulatora PID

9.2.2. Realizacja z wykorzystaniem języka SCL

Realizacja modelu regulatora PID w układzie sterowania, zgodnie z rys. 9.1, w języku tekstowym SCL jest prostsza niż w języku Ladder. Poniżej fragment programu bloku *Startup*, zawierający zadawanie parametrów początkowych elementu nieliniowego.

```
// Realizacja modelu regulatora PID w układzie sterowania
//STARTUP  - zadawanie wartości początkowych
```

```
"AC_pop" := 0;      // wartość poprzednia odczytana z przetwornika A/C w standardzie
                    //  S7 analog format
"val_I_pop" := 0;   // poprzednia wartość obliczenia członu I
// wartości początkowe regulatora PID (co oznacza praktycznie: brak regulatora)
"Kp" := 1;
"Ki" := 0;
"Kd" := 0;
```

Obsługa przerwania cyklicznego wywoływanego z czasem T_p, widoczna jest poniżej.

```
// Realizacja modelu regulatora PID w układzie sterowania
// przerwanie cykliczne co 100 ms
"blad" := "U" - "AC";   //wartość błędu; wartość podawana na regulator PID jako różnica
        // wartości zadanej i wyjścia z obiektu; w ten sposób realizowana jest pętla sprzężenia
        // zwrotnego i podanie skoku jednostkowego o wartości U
"val_P" := "Kp" * "blad";                   // wartość obliczenia członu P
"val_I_pop" := "val_I";                     // poprzednia wartość obliczenia członu I
"val_I" := "Ki" * "blad" + "val_I_pop";     // wartość obliczenia członu I
"val_D" := "Kd" * ("AC_pop" - "AC");        // wartość obliczenia członu D
"val_PID" := "val_P" + "val_I" + "val_D";   // wartość obliczenia PID
"CA":=LIMIT(MN:=-27648, IN:=REAL_TO_INT("val_PID"), MX:=27648);      // zamiana na Int,
    //  ograniczenie  do przedziału <-27648; 27648> i wysłanie ma przetwornik C/A
 "AC_pop" := "AC";  // zapamiętanie bieżącej wartości pomiaru napięcia
                    //  jako wartość poprzednią
```

10. Dobór parametrów regulatora PID

Sam fakt zastosowania regulatora PID nie jest absolutnie gwarancją prawidłowego działanie układu regulacji. Mało tego, źle dobrane parametry regulatora mogą zdecydowanie pogorszyć sterowanie. Dlatego tak ważny jest problem doboru nastaw regulatora PID. Nie jest to jednak proste zadanie, nawet dla doświadczonych automatyków. Istnieje wiele algorytmów i metod wspomagających dobór parametrów regulatora. Jednak ich zastosowanie nie daje gwarancji uzyskania wyników optymalnych, a tym bardziej idealnych. Pozwalają one jednak na zgrubny dobór nastaw tak, aby układ sterowania działał prawidłowo, zapewniając w znacznym stopniu osiągnięcie zadanych parametrów odpowiedzi układu. Dokładne „dostrojenie" regulatora zależy od konkretnych potrzeb (np. warunek minimalnego czasu odpowiedzi na skok jednostkowy, warunek braku przeregulowania itp.) i wymaga sporego doświadczenia, gdyż przeprowadza się w sposób „ręczny".

Poniżej zostaną zaprezentowane dwie metody doboru nastaw regulatora PID. Jedna wymaga większego udziału użytkownika, druga przeprowadza dobór nastaw w sposób półautomatyczny.

10.1. Dobór parametrów regulatora PID metodą Zieglera–Nicholsa

Jedną z popularniejszych metod doboru nastaw regulatora PID jest reguła Zieglera–Nicholsa [6], [8]. Nazwa pochodzi od nazwisk jej autorów. Sam algorytm postępowania jest praktycznie identyczny zarówno dla układów ciągłych, jak i dyskretnych. Inny jest jedynie sposób obliczeń nastaw regulatora. W układach dyskretnych istotną rolę pełni czas próbkowania T_p. Niestety osoby niedoświadczone bardzo często zapominają o tym i przy zmianie czasu próbkowania (spowodowanym np. zastosowaniem szybszego mikroprocesora, innego przetwornika A/C czy C/A itp.) nie przeliczają powtórnie nastaw regulatora PID, mylnie rozumując, że skoro obiekt regulacji się nie zmienił, to nie ma takiej potrzeby. Nic bardziej błędnego, może to bowiem spowodować niewłaściwe działanie układu sterowania.

W tym podrozdziale zostanie zaprezentowana metoda dla układów cyfrowych. Klasyczny algorytm postępowania Zieglera–Nicholsa został zmodyfikowany, na potrzeby układów cyfrowych, przez Y. Takahashi'ego [6], [8]. Zgodnie z tą regułą możliwe są dwa sposoby doboru nastaw regulatora. Pierwszy wymaga analizy skoku jednostkowego układu. Drugi doprowadzenia układu do granicy stabilności.

10.1.1. Analiza odpowiedzi układu sterowania na skok jednostkowy

Metoda stosowana jest w przypadku obiektów, których odpowiedź na skok jednostkowy w układzie zamkniętym ma charakter aperiodyczny, zatem funkcja przejścia obiektu może być aproksymowana jednym z poniższych równań:

– w przypadku obiektu statycznego danego wzorem:

$$G_O(s) = \frac{ke^{-sT_0}}{Ts+1} \tag{10.1}$$

– w przypadku obiektu astatycznego danego wzorem:

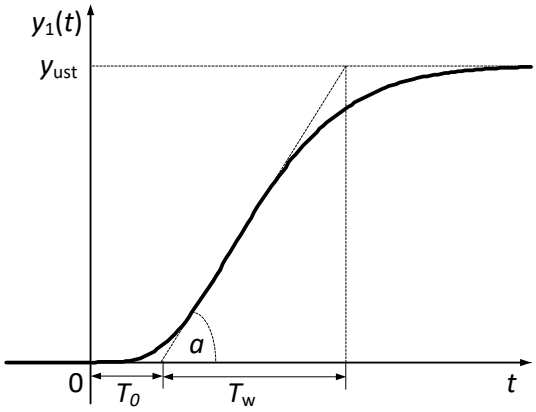
$$G_O(s) = \frac{ke^{-sT_0}}{s} \tag{10.2}$$

Algorytm postępowania jest następujący:

1. zarejestrować odpowiedź na skok jednostkowy układu bez regulatora PID, (w układzie jak na rys. 9.1, z regulatorem, ustawić $K_P = 1$, $K_I = 0$, $K_D = 0$,
2. wyznaczyć parametry T_0, T_w oraz a zgodnie z rys. 10.1, gdzie a to wzmocnienie wyliczane na podstawie zmiany sygnału wyjściowego Δy, czasu w którym doszło do tej zmiany T_w oraz zmiany sygnału wejściowego Δu (czyli skoku jednostkowego), dane wzorem:

$$a = \frac{\Delta y}{T_w \Delta u} \tag{10.3}$$

3. policzyć wartości nastaw regulatora według wzorów danych w tabeli 10.1 w zależności od typu regulatora (T_p – czas próbkowania).



Rys. 10.1. Odpowiedź układu sterowania (bez regulatora PID) na skok jednostkowy

Niekiedy wyznaczenie parametrów T_0, T_w jest mało precyzyjne, szczególnie w przypadku, gdy utrudnione jest wykreślenie stycznej do odpowiedzi na skok jednostkowy. Jest to niewątpliwie wadą tej metody. Jej zaletą jest bezinwazyjność, gdyż zadawanie skoku jednostkowego to naturalny proces w układach sterowania.

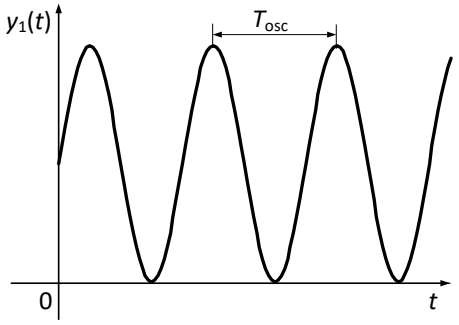
Tabela 10.1. Nastawy regulatora cyfrowego PID wg Takahashi'ego. Metoda skoku jednostkowego [6], [8]

Regulator	K_P	K_I	K_D
P	$\frac{1}{a\left(T_0+T_p\right)}$	0	0
PI	$\frac{0,9}{a\left(T_0+0,5T_p\right)}-0,5K_I$	$\frac{0,27T_p}{a\left(T_0+0,5T_p\right)^2}$	0
PID	$\frac{1,2}{a\left(T_0+T_p\right)}-0,5K_I$	$\frac{0,6T_p}{a\left(T_0+0,5T_p\right)^2}$	$\frac{0,5}{aT_p}$

10.1.2. Doprowadzenie układu do granicy stabilności

Metoda ta nazywana jest inaczej badaniem cyklu granicznego. Stosowana jest w przypadku obiektów, których odpowiedź na skok jednostkowy w układzie zamkniętym ma charakter oscylacyjny. Algorytm postępowania jest następujący:

1. w układzie jak na rys. 9.1, ustawić $K_P = 1$, $K_I = 0$, $K_D = 0$,
2. zwiększając wzmocnienie K_P, doprowadzić układ do granicy stabilności, otrzymując odpowiedź jak na rys. 10.2,



Rys. 10.2. Wyznaczenie T_{osc} układu na granicy stabilności

3. spisać wartość $K_{Pgr} = K_P$, przy której to nastąpiło oraz wyznaczyć okres oscylacji drgań nietłumionych T_{osc}, zgodnie z rys. 10.2,
4. policzyć wartości nastaw regulatora według wzorów danych w tabeli 10.2 w zależności od typu regulatora (T_p – czas próbkowania).

Tabela 10.2. Nastawy regulatora cyfrowego PID wg Takahashi'ego. Metoda granicy stabilności [6], [8]

Regulator	$K_{\mathbf{P}}$	K_I	K_D
P	$0,5K_{Pgr}$	0	0
PI	$0,45K_{Pgr} - 0,5K_I$	$0,54\frac{K_{Pgr}T_p}{T_{osc}}$	0
PID	$0,6K_{Pgr} - 0,5K_I$	$1,2\frac{K_{Pgr}T_p}{T_{osc}}$	$0,075\frac{K_{Pgr}T_{osc}}{T_p}$

Niestety jest to metoda inwazyjna, bowiem doprowadzenie układu do granicy stabilności może doprowadzić do uszkodzenia lub nawet zniszczenia obiektu regulacji. To niewątpliwie wada tej metody. Zaletą jest zdecydowanie prostsze i jednoznaczne wyznaczanie parametrów K_{Pgr} i T_{osc}.

10.1.3. Praktyczna implementacja doboru nastaw regulatora PID

Praktyczna implementacja doboru nastaw regulatora PID została przeprowadzona w układzie jak na rys. 9.1, zgodnie z algorytmem jak na rys. 9.2. Implementacja modelu regulatora PID zgodna z opisem w podrozdziale 9.1 (klasyczny sterownik mikroprocesorowy, język C) oraz podrozdziale 9.2.2 (sterownik PLC, język SCL).

Dobór nastaw regulatora PID został przeprowadzony tylko metodą doprowadzenia układu do granicy stabilności. Poniżej, dla sterownika mikroprocesorowego ARM i sterownika PLC, zostanie zaprezentowane:

– odczyt okresu oscylacji,
– dobór nastaw regulatora PID zgodnie z tabelą 10.2,
– odpowiedź układu sterowania jak na rys. 9.1 z wprowadzonymi nastawami regulatora.

Sterownik mikroprocesorowy ARM

Na rysunku 10.3 widoczne jest wyznaczenie okresu oscylacji T_{osc} = 15,2 s, zarejestrowane na oscyloskopie. Nastąpiło to przy $K_{Pgr} = K_P$ = 1,95. Obliczone, zgodnie z tabelą 10.2, parametry regulatora PID wynoszą:

K_P = 1,16,

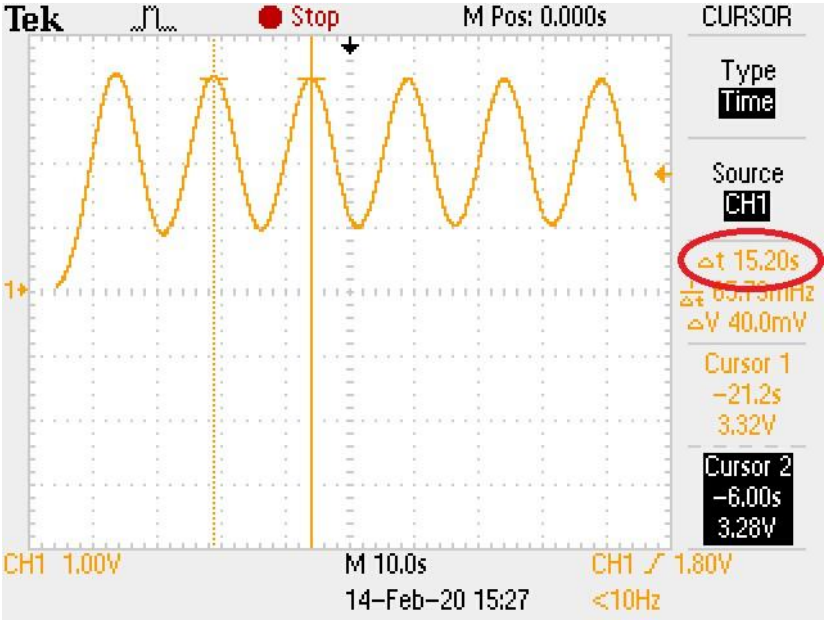
K_I = 0,0154,

K_D = 22,23.

Odpowiedź na skok jednostkowy, z tak dobranymi parametrami regulatora PID dana jest na rys. 10.4. Porównując ten wykres z wykresem na rys. 9.3, widoczny jest pozytywny efekt działania regulatora:

– odpowiedź ustala się na wartości zadanej, zatem błąd położenia wynosi 0% (bez regulatora PID, błąd był na poziomie 50%),

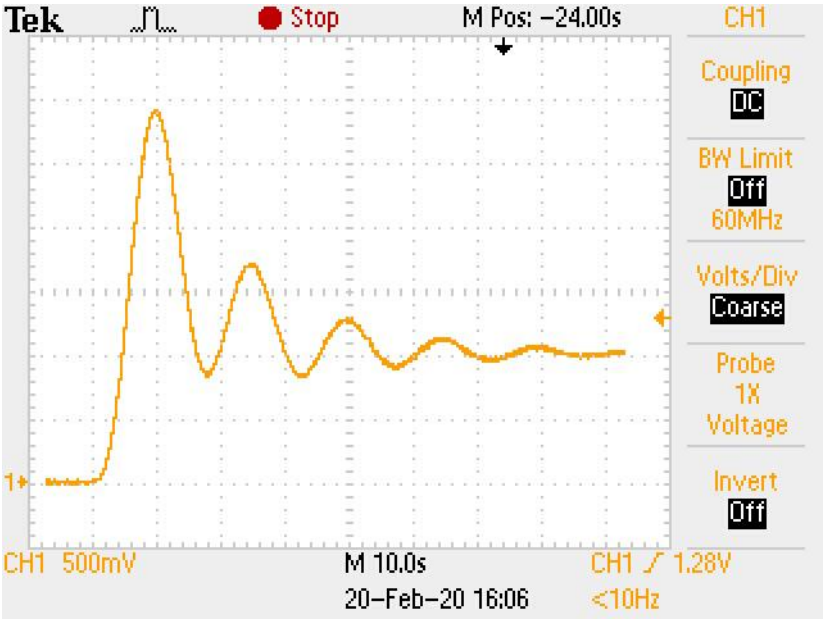
– znacznemu skróceniu uległ czas ustalenia. Teraz wynosi ok. 90 s, a bez regulatora PID było to ok. 175 s.



Rys. 10.3. Wyznaczanie okresu oscylacji w układzie jak na rys. 9.1 z wykorzystaniem tylko regulatora P, K_{gr} = 1,95

Jedyną wadą zastosowania regulatora PID, w tym konkretnym przypadku, jest wzrost przeregulowania do ok. 190%, podczas, gdy bez regulatora wynosiło ono ok. 63%. Oczywiście możliwe jest takie skorygowanie nastaw regulatora PID, aby i ten parametr poprawić. Niestety wiąże się to zazwyczaj z nieznacznym pogorszeniem innych parametrów i wymaga bardzo dobrej znajomości

obiektu oraz praktyki w doborze parametrów regulatora. Trzeba mieć świadomość, że zmiana parametrów jednego członu (P lub I lub D) zazwyczaj nie wystarcza, a brak doświadczenia może pociągnąć za sobą pogorszenie parametrów układu sterowania, a nawet uniemożliwić jego działanie (przejście w stan niestabilności). O ile przy analizie modeli (obiektu i regulatora) nie ma to większego znaczenia (zawsze możliwy jest powrót do parametrów regulatora PID uprzednio obliczonych), o tyle w przypadku układów rzeczywistych, może przynieść katastrofalne, nieraz nieodwracalne skutki.



Rys. 10.4. Odpowiedź na skok jednostkowy (2,0 V) w układzie jak na rys. 9.1, przy parametrach regulatora PID dobranych zgodnie z regułą Zieglera–Nicholsa

Sterownik PLC

Na rysunku 10.5 widoczne jest wyznaczenie okresu oscylacji T_{osc} = 14,8 s, zarejestrowane na panelu operatorskim HMI. Nastąpiło to przy $K_{Pgr} = K_P = 1{,}53$. Obliczone, zgodnie z tabelą 10.2, parametry regulatora PID wynoszą:

$K_P = 0{,}9118$,

$K_I = 0{,}0124$,

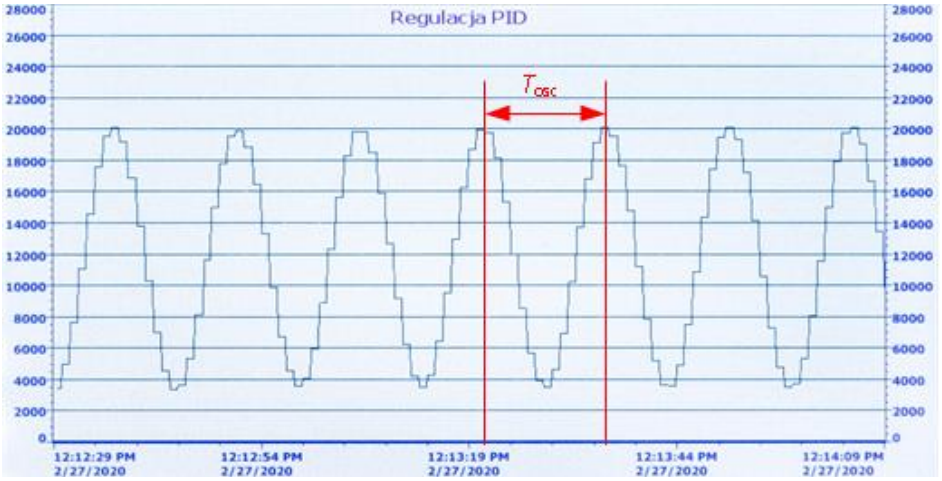
$K_D = 16{,}9830$.

Odpowiedź na skok jednostkowy, z tak dobranymi parametrami regulatora PID znajduje się jest na rys. 10.6. Porównując ten wykres z wykresem na rys. 9.6 widoczny jest pozytywny efekt działania regulatora:

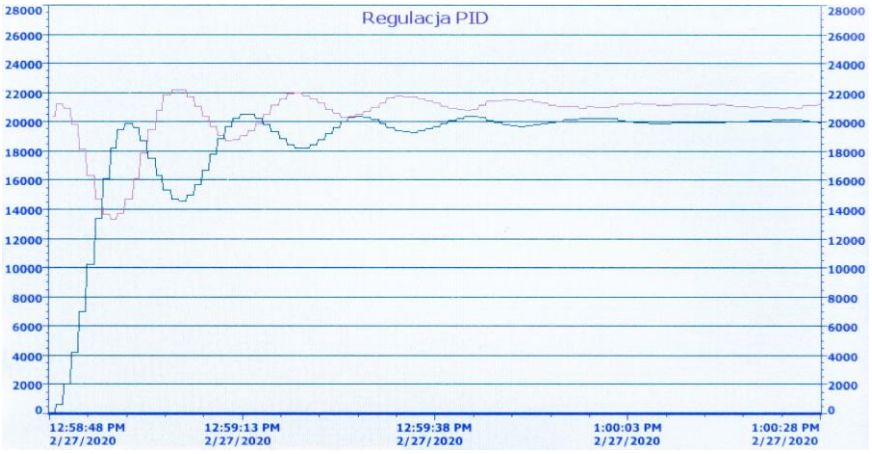
– odpowiedź ustala się na wartości zadanej, zatem błąd położenia wynosi 0% (bez regulatora PID błąd był na poziomie ok. 50%),

– znacznemu skróceniu uległ czas ustalenia. Teraz wynosi ok. 75 s, a bez regulatora PID było to ok. 160s,
– ograniczone, praktycznie do wartości kilku procent zostało przeregulowanie, podczas gdy bez regulatora wynosiło ok. 60%.

Dodatkowo na wykresie widoczny jest sygnał sterowania, czyli sygnał wyjściowy z regulatora PID (linia czerwona).



Rys. 10.5. Wyznaczanie okresu oscylacji w układzie jak na rys. 9.1 z wykorzystaniem tylko regulatora P, K_{gr} = 1,55



Rys. 10.6. Odpowiedź na skok jednostkowy (20 000 w standardzie S7 analog format, co odpowiada ok. 7,2 V) w układzie jak na rys. 9.1, przy parametrach regulatora PID dobranych zgodnie z regułą Zieglera–Nicholsa (linia czerwona – sygnał sterowania)

W tym miejscu nasuwa się naturalne pytanie, dlaczego działanie regulatora PID jest inne w przypadku zaimplementowania na sterowniku mikroprocesorowym ARM, niż w przypadku zaimplementowania na sterowniku PLC, dla teoretycznie takiego samego obiektu i identycznego algorytmu regulatora. W praktyce jednak modele obiektów zostały zaimplementowane na różnych sterownikach, o różnych parametrach (także o różnych parametrach przetworników A/C i C/A), zatem jednak minimalnie różnią się od siebie. Wiarygodniejsze byłoby porównanie działania dwóch identycznych modeli regulatorów PID zrealizowanych na różnych sterownikach, na identycznym obiekcie rzeczywistym, a nie na modelach.

10.2. Dobór parametrów regulatora PID metodą przekaźnikową

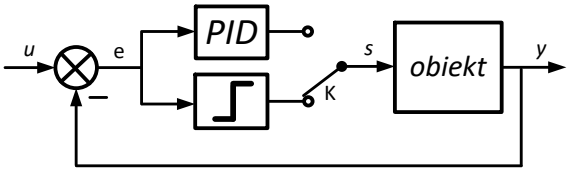
Metoda Zieglera–Nicholsa jako jedna z najpopularniejszych metod doboru nastawa regulatora PID ma też swoje wady, które były sygnalizowane w poprzednich podrozdziałach. Dla przypomnienia:

– w przypadku analizy skoku jednostkowego, często bardzo utrudniony lub niejednoznaczny jest odczyt wymaganych parametrów,
– w przypadku doprowadzenia układu do granicy stabilności, w układach rzeczywistych istnieje duże prawdopodobieństwo uszkodzenia lub nawet zniszczenia obiektu.

Do tego trzeba jeszcze dodać kolejną wadę: obydwie metody są stosunkowo problematyczne przy numerycznej realizacji: w przypadku skoku jednostkowego – odczyt parametrów, a w przypadku granicy stabilności – czasochłonny sposób dochodzenia do tej granicy (ryzyko utraty stabilności przy zbyt dużych zmianach wartości K_P).

Tych wad jest pozbawiona tzw. metoda przekaźnikowa opracowana przez dwóch Szwedów Åströma i Hägglunda [1]. W zasadzie bazuje ona na badaniu cyklu granicznego opracowanym przez Zieglera–Nicholsa. Podstawową różnicą jest doprowadzenie układu do wzbudzenia drgań harmonicznych o niewielkiej, i co najważniejsze, ograniczonej amplitudzie. Uzyskiwane jest to poprzez zastosowanie przekaźnika dwupołożeniowego w miejsce regulatora PID, tak jak to jest widoczne na rys. 10.7. Należy zwrócić uwagę na fakt, że w przypadku podłączenia przekaźnika w miejsce regulatora, wymuszanie na wejściu układu sterowania u musi być równe 0, co jest warunkiem koniecznym uzyskania drgań własnych w układzie. Niewielką wadą tej metody jest brak możliwości jej stosowania w przypadku modeli obiektów I rzędu (drgania własne nigdy nie wystąpią) oraz ograniczone stosowanie w przypadku modeli obiektów II rzędu (utrudnione pojawienie się drgań własnych) [8]. Jednak w przypadku obiektów

rzeczywistych, zawsze jest możliwość ich zamodelowania układem III rzędu, co też najczęściej jest czynione.



Rys. 10.7. Układ do praktycznej realizacji metody przekaźnikowej

Sama procedura doboru nastaw regulatora PID wymaga tylko krótkiego eksperymentu identyfikacyjnego, zatem możliwa jest korekta doboru nastaw, w każdej chwili, co jest szczególnie istotne w przypadku, gdy parametry obiektu ulegają zmianie w czasie. Z tego też względu procedura ta należy do grupy metod zapewniających tzw. autostrojenie regulatora PID [4].

W przypadku pojawienia się drgań własnych, konieczne jest wyznaczenie parametrów cyklu granicznego, jak to zostało pokazane na rys. 10.8. Następnie na podstawie parametrów A i B obliczane jest wzmocnienie graniczne zgodnie ze wzorem (10.4):

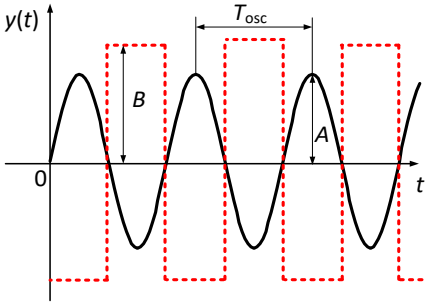
$$K_{Pgr} \approx \frac{4A}{\pi B} \tag{10.4}$$

gdzie:

A – amplituda sygnału wejściowego przekaźnika, czyli sygnału błędu e (sygnał oznaczony linią ciągłą na rys. 10.8),

B – amplituda sygnału wyjściowego przekaźnika, czyli sygnału sterującego s (sygnał oznaczony linią przerywaną na rys. 10.8).

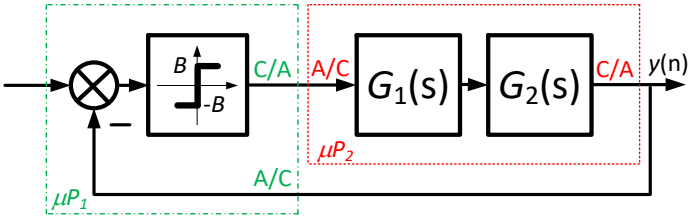
Ostatni krok to obliczenie parametrów regulatora zgodnie z tabelą 10.2, wprowadzenie ich do regulatora i przełączenie klucza „K” na rys. 10.7, aby sprawdzić działanie układu z tak dobranymi parametrami regulatora.



Rys. 10.8. Wyznaczenie parametrów cyklu granicznego

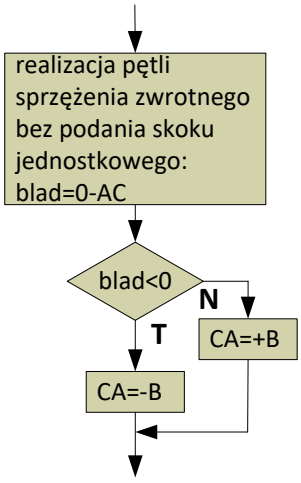
10.2.1. Praktyczna implementacja doboru nastaw regulatora PID

Praktyczna implementacja metody przekaźnikowej doboru nastaw regulatora PID została przeprowadzona w układzie jak na rys. 10.9, zgodnie z algorytmem jak na rys. 9.2, z tym że w miejsce fragmentu algorytmu oznaczanego ramką (czerwona linia przerywana) wystąpi fragment algorytmu z rys. 10.10. Obiekt regulacji, realizowany na innym sterowniku, pozostał bez zmian.



Rys. 10.9. Układ do praktycznej realizacji metody przekaźnikowej

Programy realizujące fragment algorytmu z rys. 10.10, napisane zarówno w języku C, w przypadku sterownika mikroprocesorowego ARM, jak i w językach Ladder i SCL, w przypadku sterownika PLC, opisane zostaną poniżej. Powinny one zastąpić odpowiedni fragment programu realizującego regulator PID opisanego w rozdziale 9 (9.1 – w języku C, 9.2.1 – w języku Ladder i 9.2.2 – w języku SCL).



Rys. 10.10. Fragment algorytmu z rys. 9.2 realizujący działanie przekaźnika dwupołożeniowego w układzie sterowania zgodnie z rys. 10.9

Sterownik mikroprocesorowy ARM

Poniżej znajduje się fragment programu w języku C realizujący model przekaźnika dwupołożeniowego w układzie sterowania jak na rys. 10.9.

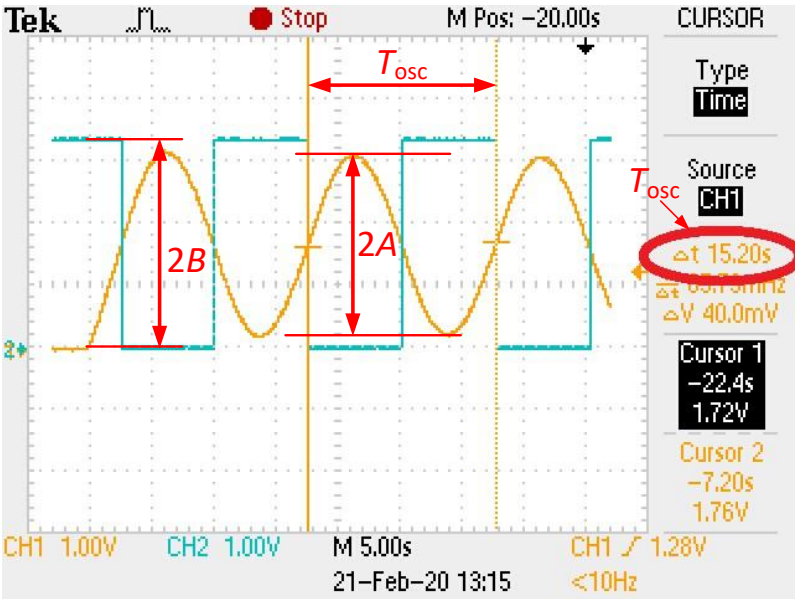
```
blad=0-AC+512.0;   // wartość błędu; wartość podawana na przekaźnik jako różnica
   // wartości zadanej (w tym wypadku równej 0) i wyjścia z obiekt.
   // W ten sposób realizowana jest pętla sprzężenia zwrotnego
   // Dodatkowo dodawana jest wartość 512 // (połowa z wartość B),
   // ponieważ przetworniki A/C i C/A w ARM są unipolarne 10-bitowe
if(blad<0)
  nap_CA=0;           // bo przetwornik C/A unipolarny
else
  nap_CA=1023;      // ograniczenie do wartości maksymalnej
// wysłanie wartości CA na przetwornik C/A
```

Na rysunku 10.11 widoczne są drgania własne zarejestrowane na oscyloskopie, w układzie jak na rys. 10.9. Zaznaczone zostały parametry cyklu granicznego. Wynoszą one odpowiednio:

$2A = 2{,}92$ V,

$2B = 3{,}32$ V,

$T_{osc} = 15{,}2$ s.



Rys. 10.11. Drgania własne w układzie jak na rys. 10.9

Odczytano podwójną amplitudę (wartość międzyszczytową) obydwu sygnałów, gdyż było to prostsze i dokładniejsze. Następnie na podstawie wzoru (10.4) obliczone zostało wzmocnienie krytyczne:

$K_{Pgr} = 1{,}1198$.

Parametry regulatora PID zostały obliczone na podstawie wzorów z tabeli 10.2:

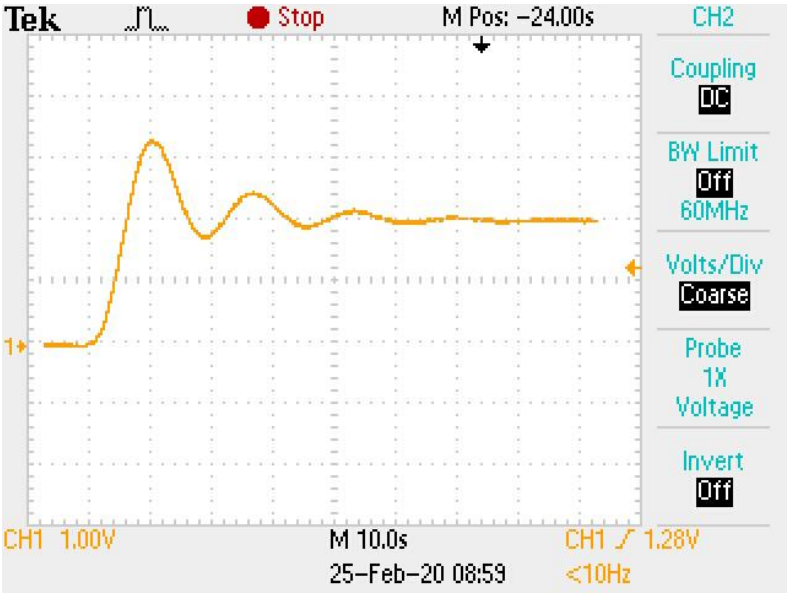
$K_P = 0{,}6675$,

$K_I = 0{,}0088$,

$K_D = 12{,}7661$.

Tak obliczone parametry zostały wprowadzone do regulatora PID. Efekt działania regulatora w układzie sterowania jak na rys. 9.1 widoczny jest na rys. 10.12.

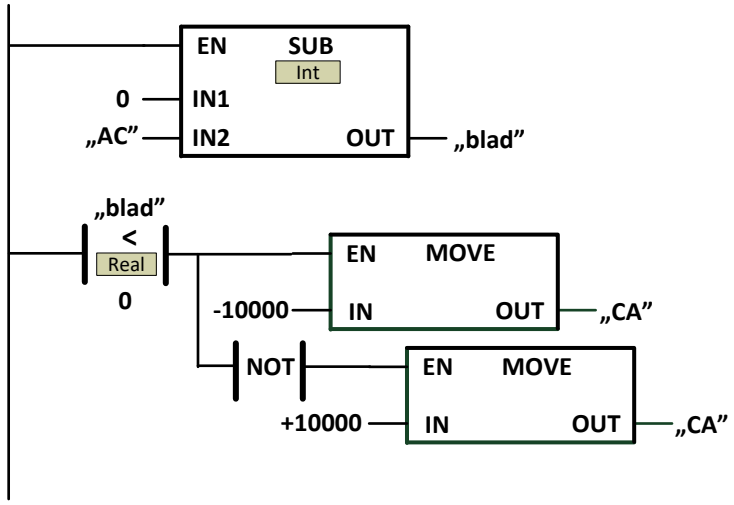
Jak widać układ ustala się na wartości zadanej 2 V po czasie ok. 70 s, a przeregulowanie wynosi ok. 65%, zatem uzyskano parametry lepsze niż w przypadku parametrów regulatora dobranych zgodnie z regułą Zieglera–Nicholsa.



Rys. 10.12. Odpowiedź na skok jednostkowy (2,0 V) w układzie jak na rys. 9.1, przy parametrach regulatora PID dobranych zgodnie z metodą przekaźnikową

Sterownik PLC

Na rysunku 10.13 znajduje się fragment programu realizujący model przekaźnika dwupołożeniowego w układzie sterowania jak na rys. 10.9, zrealizowany w języku Ladder.



Rys. 10.13. Fragment programu w języku Ladder realizujący model przekaźnika dwupołożeniowego w układzie sterowania jak na rys. 10.9

Poniżej identyczny fragment programu zrealizowany w języku SCL:

```
"blad" := - "AC";   //wartość błędu; wartość podawana na przekaźnik jako różnica
    // wartości zadanej (w tym wypadku równej 0) i wyjścia z obiektu;
    // w ten sposób realizowana jest pętla sprzężenia zwrotnego;
IF "blad" < 0 THEN    // jeżeli blad ujemny….
  "CA" := -10000;     // … to na C/A podaj -10000 (odpowiada napięciu ok. -3,62V)
ELSE                  // jeżeli blad >=0 …..
  "CA" := 10000;      // to na C/A podaj +10000 (odpowiada napięciu ok. +3,62V)
END_IF;
```

Na rysunku 10.14 widoczne są drgania własne zarejestrowane na panelu operatorskim HMI, w układzie jak na rys. 10.9. Zaznaczone zostały odczytane parametry cyklu granicznego. Wynoszą one odpowiednio:

$A = 8405$,

$B = 10\,000$,

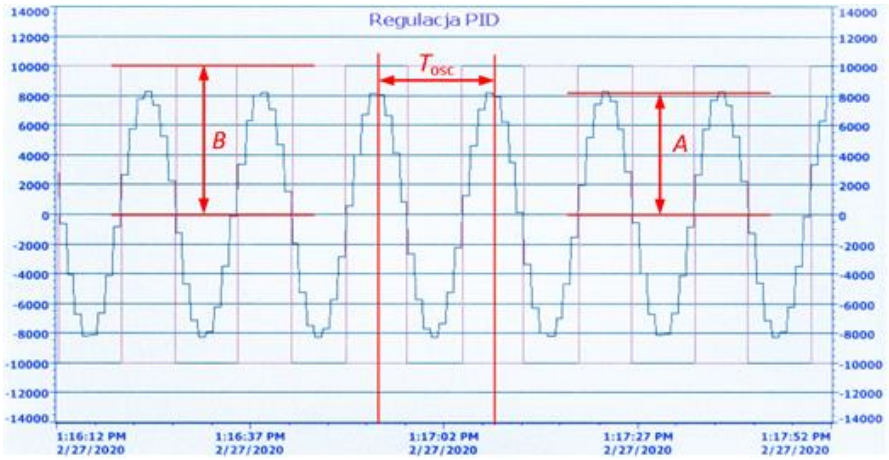
$T_{osc} = 14{,}8$ s.

Wartości A i B są podane w standardzie *S7 analog format*. Następnie na podstawie wzoru (10.4) obliczone zostało wzmocnienie krytyczne $K_{Pgr} = 1{,}0695$, a na podstawie wzorów z tabeli 10.2, parametry regulatora PID, które odpowiednio wynoszą:

$K_P = 0{,}6374$,

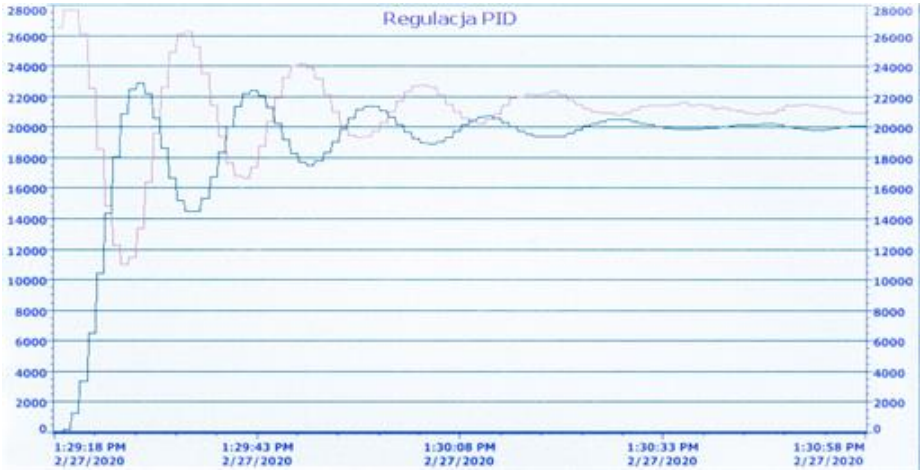
$K_I = 0{,}0087$,

$K_D = 11{,}8717$.



Rys. 10.14. Drgania własne w układzie jak na rys. 10.9

Tak obliczone parametry zostały wprowadzone do regulatora PID. Efekt działania regulatora w układzie sterowania jak na rys. 9.1 widoczny jest na rys. 10.15. Jak widać, układ ustala się na wartości zadanej 20 000 (*S7 analog format*) po czasie ok. 90 s, a przeregulowanie na poziomie 14%, zatem uzyskano parametry nieznacznie gorsze niż w przypadku parametrów regulatora dobranych zgodnie z regułą Zieglera–Nicholsa.



Rys. 10.15. Odpowiedź na skok jednostkowy (20 000 w standardzie S7 analog format, co odpowiada ok. 7,2 V) w układzie jak na rys. 9.1, przy parametrach regulatora PID dobranych zgodnie z metodą przekaźnikową (linia czerwona – sygnał sterowania)

Trudno porównywać ze sobą dwie realizacje autostrojenia i działania układu regulacji, jedną z wykorzystaniem klasycznego sterownika mikroprocesorowego, drugą z wykorzystaniem sterownika PLC. Wszak to różne sterowniki

stworzone do różnych zadań inżynierskich, także różne języki programowania. Niezależnie od różniących się od siebie wyników odpowiedzi modelu układu sterowania z regulatorem PID w tych realizacjach, jedno jest pewne: cel został osiągnięty, układ polepszył swoje działanie. Zostało to zrobione w sposób prostszy i mniej inwazyjny, niż z wykorzystaniem klasycznej metody Zieglera–Nicholsa.

W podrozdziale tym została zaprezentowana tylko najprostsza, zgrubna metoda autostrojenia regulatorów PID. Istnieje wiele algorytmów pozwalających na precyzyjny dobór paramentów (tzw. dostrojenie precyzyjne), na podstawie wyników tejże metody zgrubnej [1], [4].

11. Podsumowanie

Podręcznik ten prezentuje tylko wybrane przykłady realizacji modeli cyfrowych oraz niektóre problemy modelowania cyfrowego. Można powiedzieć, że to wierzchołek góry lodowej. Jednak doświadczony badacz gór lodowych wie, że na podstawie obserwacji i pomiarów wierzchołka góry lodowej można o niej dowiedzieć się bardzo dużo. Celem tego podręcznika było także pokazanie, że modelowanie cyfrowe nie jest trudne i leży w zasięgu możliwości każdego inżyniera-automatyka. Książka zawiera wiele praktycznych porad będących sumą doświadczeń autora w dziedzinie szeroko rozumianych sterowników mikroprocesorowych oraz w dziedzinie automatyki przemysłowej. Doświadczeń zdobytych podczas pracy w przemyśle tuż po zakończeniu studiów oraz wieloletniej pracy w charakterze nauczyciela akademickiego i opiekuna Studenckiego Koła Naukowego.

Autor ma nadzieję, że podręcznik zachęci do modelowania cyfrowego wiele osób, które do tej pory nie miały z tym do czynienia. A osoby mające już jakieś doświadczenie w tej dziedzinie być może podejmą próby modeli bardziej zawansowanych, innych niż te prezentowane.

Wracając do Mickiewiczowskiego cytatu będącego mottem tej książki, dobry inżynier powinien posługiwać się w swojej pracy zawodowej zawsze „*szkiełkiem i okiem*”, lecz nie powinno mu zabraknąć także „*czucia i wiary*”.

Literatura

[1] ÅSTRÖM K.J., HÄGGLUND T., *Automatic tuning of simple regulators with specifications on phase and amplitude margins*, Automatica, Vol. 20, No. 5, 1984, pp. 645–651.

[2] FINDEISEN W., *Technika regulacji automatycznej*, PWN, Warszawa 1965.

[3] HORLA D., KRÓLIKOWSKI A., *Identyfikacja obiektów sterowania. Metody dyskretne* (wydanie II poprawione i rozszerzone), Wydawnictwo Politechniki Poznańskiej, Poznań 2010.

[4] KULA K., *Automatyczne strojenie regulatora PID w układzie on-line na podstawie identyfikacji metodą przekaźnikową*, Zeszyty Naukowe Akademii Morskiej w Gdyni, nr 62, Gdynia 2009.

[5] ROSOŁOWSKI E., *Automatyczne sterowanie i regulacja. Procesy ciągłe i dyskretne*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2020.

[6] ROSOŁOWSKI E., *Cyfrowe przetwarzanie sygnałów w automatyce elektroenergetycznej*, Akademicka Oficyna Wydawnicza EXIT, Warszawa 2002.

[7] SZAFRAN J., WISZNIEWSKI A., *Algorytmy pomiarowe i decyzyjne cyfrowej automatyki elektroenergetycznej*, WNT, Warszawa, 2001.

[8] *Podstawy Automatyki. Ćwiczenia laboratoryjne,* Praca zbiorowa pod red. A. Wiszniewskiego, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2000.