Spis treści

1 Wstęp.

1.1 Rys historyczny regulacji automatycznej.

Kontrola nad przebiegiem procesów, automatyczne utrzymywanie poziomu wielkości fizycznych na stałym poziomie, a nawet sprzężenie zwrotne to zagadnienia, które zaprzątały umysły ludzi już od czasów starożytności. Okresem przełomowym okazała się jednak dopiero XVII-wieczna rewolucja przemysłowa.

Kamieniem milowym dla sterowania realizowanego bez czynnego udziału człowieka było wynalezienie przez Jamesa Watta w 1784 roku odśrodkowego regulatora, który stabilizował prędkość obrotową maszyn parowych. Kolejnymi wyznacznikami postępu w tej dziedzinie były: użycie regulatora ciśnienia gazu (1820 r.), bimetalicznego regulatora temperatury (Ure, 1830) i hydraulicznego serwomechanizmu do sterowania maszyną parową przez Sickela (1853 r.). Jeszcze przed pierwszą wojną światową miało miejsce zainstalowanie pneumatycznego, dwustawnego regulatora temperatury, skonstruowanie przez G. Dalena układu automatycznej regulacji wielkości i częstotliwości pulsacji płomienia. W czasach dwudziestolecia międzywojennego dostępne były laboratoryjne regulatory PI, wykorzystujące silniki w celu uzyskania działania całkującego. Pojęcie transmitancji widmowej pojawiło się na początku lat czterdziestych za sprawą Harrisa, a niedługo potem Brown wprowadził pojęcie transmitancji operatorowej. Ziegler i Nichols opracowali zasady doboru nastaw regulatorów pneumatycznych w 1942r. Niedługo potem przekonano się do zalet elektroniki i zaczęto wdrażać regulatory cyfrowe do sterowania procesami przemysłowymi. Następnym punktem przełomowym stało się wynalezienie mikroprocesora przez W. Hoffa. Zapoczątkowało to burzliwy rozwój sterowania cyfrowego (J.R Rogazzini, G. Frankiln, L.A Zadeh, E.J Jury, K. Åström). W 1984 r. opracowano algorytm samo nastrajania regulatorów EXACT. W latach dziewięćdziesiątych rozpowszechniło się zastosowanie technologii informatycznych w automatyce. Głównymi powodami było zastępowanie urządzeń analogowych cyfrowymi, powstawanie języków programowania typu Visual, a co ważniejsze języków programowania ukierunkowanych na regulację i sterowanie (między innymi przeznaczonych do programowania sterowników PLC).[1]

1.2 Aktualny stan wiedzy.

Nieustanny rozwój systemów sterowania, a także szeroko pojętej automatyki, to powody tego, że coraz więcej procesów kontrolowanych jest bez udziału człowieka. Jednym z problemów są złożone obiekty o charakterystycznych cechach, do sterowania których zastosowanie klasycznych regulatorów, w szczególności regulatora PID nie daje zadowalających efektów. Implementacja niekonwencjonalnych regulatorów w sterownikach programowalnych okazuje się polepszać jakość działania układu.[12] Trwają poszukiwania rozwiązań dla problemów sterowania nieliniowymi modelami i adaptacyjnej samoregulacji PID.[13] Doskonalone są metody oparte na logice rozmytej, a także znane od dziesiątek lat systemy przeciwdziałające nasyceniu integratora (anti-windup).[14] Rozległość płaszczyzny zastosowań regulatorów proporcjonalno-różniczkująco-całkujących powoduje pojawianie się coraz to nowych badań i opracowań porównujących wyniki doboru nastaw i konfiguracji w różnych dziedzinach przemysłu i nie tylko.[15] Inżynierowie modyfikując strukturę regulatorów tworzą nowe algorytmy sterowania, często dające znacznie lepsze efekty w rozważanych przypadkach.[16][17]

1.3 Układy regulacji na tle aktualnego stanu wiedzy.

Podstawowa struktura układów regulacji to struktura jednoobwodowa, która może nie zapewnić satysfakcjonującej jakości regulacji w przypadku obiektów o wysokiej inercji, właściwościach nieliniowych lub z dużymi opóźnieniami, albo specyficznych procesów. Dlatego istnieją inne. Między innymi struktury takie jak: kaskadowa, zamknięto otwarta, prosta układu regulacji stosunku, selekcyjna, a także ich hybrydy. Wszystkie one posiadają szereg własności predestynujących je do pełnienia odpowiednich funkcji. Na przykład układ regulacji stosunku wykorzystuje się do utrzymania określonej proporcji dwóch mierzonych wielkości. Każda struktura wymaga odpowiedniego dostrojenia.[2] Powstaje wiele prac badawczych, których celem jest sprawdzenie, czy dana struktura sprawdzi się przy sterowaniu wybranym rodzajem obiektu, lub zaimplementowanie danego układu w mikrokontrolerze.[10][11] Większość z nich wykorzystuje pakiet MATLAB/Simulink. Oryginalnym aspektem niniejszej pracy jest wykorzystanie środowiska CODESYS do implementacji, a następnie symulacji kaskadowej struktury regulacji wraz z przykładowym obiektem

sterowania. Motywacją do wykonania takiego projektu była chęć odkrycia metody do zwiększenia praktycznych możliwości edukacji zdalnej z dziedziny systemów sterowania i programowalnych sterowników logicznych w czasie pandemii, dzięki wykorzystaniu używanego w przemyśle, ale dostępnego i darmowego oprogramowania.

2 Cel, zakres, założenia i efekt końcowy pracy.

2.1 Cel pracy.

Celem pracy było sprawdzenie czy w środowisku programowania CODESYS jest możliwe wykonanie i symulacja poprawnie działającego systemu regulacji kaskadowej z wirtualnym obiektem sterowania. Następnie dobranie właściwych parametrów zaimplementowanych regulatorów w celu uzyskania odpowiednich przebiegów wielkości regulowanej.

2.2 Zakres pracy.

- Zapoznanie się z regulacją kaskadową wybranego procesu,
- zapoznanie się ze środowiskiem programowania CODESYS,
- implementacja algorytmu regulatora PID w programie,
- implementacja modelu obiektu sterowania w programie,
- zbudowanie systemu regulacji kaskadowej,
- przeprowadzenie symulacji,
- dobór nastaw regulatorów,
- określenie jakości regulacji wskaźnikiem ISE,
- stworzenie wizualizacji on-line pozwalającej obserwować sygnały i zmieniać parametry układu podczas jego pracy,
- prezentacja wyników,
- przedstawienie zagadnień związanych z wykonanymi w ramach projektu działaniami i dotyczących teoretycznych aspektów pracy.

2.3 Założenia co do sposobu realizacji i efektu końcowego pracy oraz założenia upraszczające.

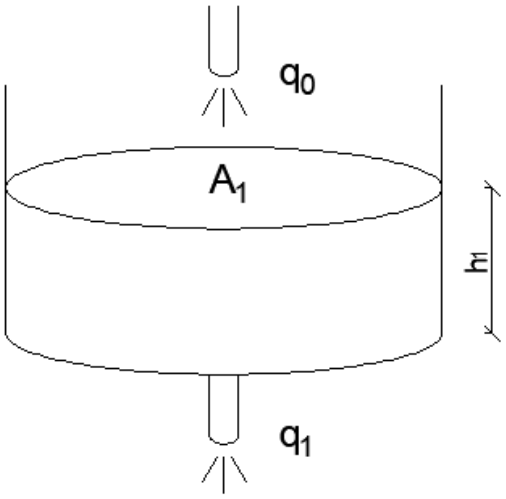
Nie zostało to bezpośrednio wskazane w temacie pracy, ale cały program został zrealizowany w zasymulowanym sterowniku PLC. Użyto w tym celu środowiska programowania CODESYS V3.5 SP16 Patch 3 na bezpłatnej licencji CODESYS Development System. Projekt nie zakładał przetestowania programu na fizycznym sterowniku PLC z powodu sytuacji epidemiologicznej w kraju i z tego właśnie powodu nie zostało to uczynione. Program był testowany na symulowanym, wirtualnym sterowniku CODESYS Control Win V3 x64 v3.5.16.30. Okres pętli programu wynosił 20ms. Główna część została zaprogramowana w języku CFC (Continous Function Chart). Kryterium wyboru tego języka był najbardziej przystępny wygląd ukończonego programu przypominający schemat blokowy realizowanego systemu. Bloki funkcyjne zostały zaprogramowane w języku ST (Structured Text). Model obiektu sterowania jest teoretyczny. Parametry obiektu celowo zostały dobrane tak, aby stosunkowo szybko osiągał on stan ustalony, aby skrócić czas symulacji. Było to ważniejsze niż interpretacja fizyczna obiektu, ponieważ głównym celem pracy jest zbudowanie kaskadowego układu sterowania. Stała czasowa pierwszej jego części jest mniejsza niż drugiej, co jest dobrym powodem do zastosowania takiego właśnie układu regulacji. Dla uproszczenia obiektu przyjęto model liniowy natężenia przepływu.

Efektem końcowym pracy jest działająca symulacja systemu kaskadowej regulacji dwóch połączonych zbiorników. Zaimplementowane od podstaw algorytmy regulatorów PID i model obiektu zbudowany z bloków funkcyjnych zaprogramowanych tak, aby odzwierciedlały dynamikę obiektów inercyjnych pierwszego rzędu działają zgodnie z założeniami. Wizualizacja pozwala obserwować w czasie rzeczywistym zmiany wybranych sygnałów (takich jak: sygnał zadany, uchyby regulacji, sygnały sterujące, wielkości regulowane) na wykresie, a także wartości liczbowe niektórych z nich (takich jak: wartość zadania, wielkości regulowane, wskaźnik jakości ISE), oraz wartość liczbową określającą jakość regulacji według kryterium kwadratowego ISE. Daje ona również możliwość zmiany on-line wartości zadanej jak i parametrów regulatorów PID (takich jak: współczynniki wzmocnienia K, stałe czasowe całkowania Ti, stałe czasowe różniczkowania Td, dolne i górne ograniczenia sygnałów sterujących).

3 Model obiektu sterowania.

3.1 Zbiornik.

Wielkością regulowaną jest poziom wody w zbiorniku *h*, a sygnałem sterującym natężenie przepływu wody wpływającej do zbiornika *q0*.



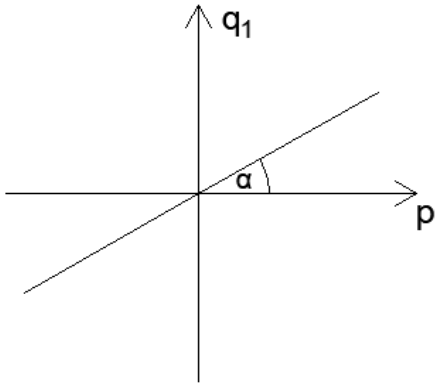
Rys. 3.1 Rysunek poglądowy zbiornika, gdzie q_0, q_1 to odpowiednio natężenia przepływu wody wpływającej i wypływającej, A_1 - pole przekroju zbiornika, h_1 - poziom wody.

Natężenie przepływu wody wypływającej określone jest wzorem:

$$q_1(t) = c \cdot p(t) \tag{3.1}$$

Współczynnik proporcjonalności:

$$c = tg(\alpha) \tag{3.2}$$



Rys. 3.2 Zależność natężenia przepływu q_1 od ciśnienia na dnie zbiornika p.

Ciśnienie na dnie zbiornika:

$$p(t) = \rho \cdot g \cdot h(\mathrm{t}) \tag{3.3}$$

Po podstawieniu (3.3) do (3.1):

$$q_1(t) = k \cdot h(t) \tag{3.4}$$

gdzie:

$$k = \rho \cdot g \cdot c \tag{3.5}$$

Wiedząc, że objętość wody w zbiorniku:

$$V(t_1) - V(t_0) = \int_{t_0}^{t_1} q_o(t) - q_1(t) \tag{3.6}$$

Po przyjęciu zerowych warunków początkowych i obustronnym zróżniczkowaniu względem czasu:

$$\frac{dV(t)}{dt} = q_o(t) - q_1(t) \tag{3.7}$$

Dodatkowo wiedząc, że objętość wody w zbiorniku:

$$V(t) = A \cdot h(t) \tag{3.8}$$

Po obustronnym zróżniczkowaniu względem czasu:

$$\frac{dV(t)}{dt} = A \cdot \frac{dh(t)}{dt} \tag{3.9}$$

Porównując (3.7) i (3.9) oraz po podstawieniu (3.4):

$$A \cdot \frac{dh(t)}{dt} = q_o(t) - k \cdot h(t) \tag{3.10}$$

Po transformacji Laplace'a przy zerowych warunkach początkowych:

$$\begin{gathered} A \cdot s \cdot H(s) = Q_o(s) - k \cdot H(s) \\ H(s) \cdot (A \cdot s + k) = Q_o(s) \\ \frac{H(s)}{Q_o(s)} = \frac{1}{A \cdot s + k} \end{gathered} \tag{3.11}$$

Postać transmitancji operatorowej obiektu inercyjnego I rzędu:

$$G(s) = \frac{K}{s \cdot T + 1} \tag{3.12}$$

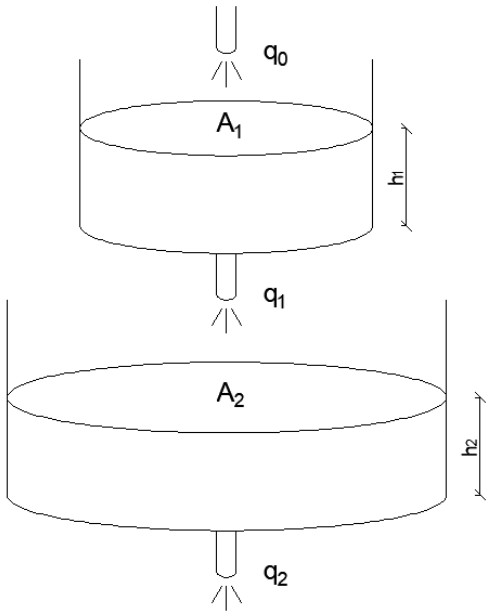
gdzie:

$$\begin{gathered} K = \frac{1}{k} \\ T = \frac{A}{k} \end{gathered} \tag{3.13}$$

Na podstawie [9].

3.2 Kaskada dwóch zbiorników.

Wielkością regulowaną jest poziom wody w drugim zbiorniku *h2*, a sygnałem sterującym natężenie przepływu wody wpływającej do pierwszego zbiornika *q0*.



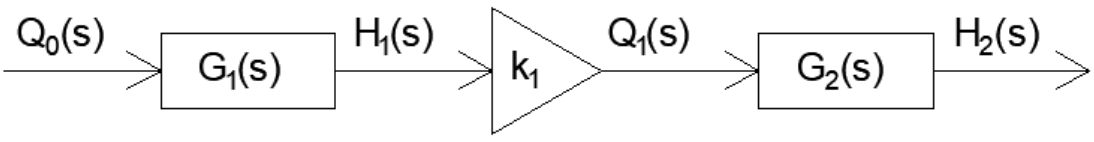
Rys. 3.3 Rysunek poglądowy kaskady dwóch zbiorników, gdzie q_0, q_1, q_2 to odpowiednio natężenia przepływu wody: wpływającej do pierwszego zbiornika, wypływającej z pierwszego i jednocześnie wpływającej do drugiego, wypływającej z drugiego , *A1, A2* - pola przekroju zbiorników pierwszego i drugiego, *h1, h2* - poziom wody w zbiorniku: pierwszym, drugim.

Zgodnie ze wzorem (3.12) transmitancje pojedynczych zbiorników prezentują się następująco:

$$G_1(s) = \frac{K_1}{s \cdot T_1 + 1} \tag{3.14}$$

$$G_2(s) = \frac{K_2}{s \cdot T_2 + 1} \tag{3.15}$$

Transmitancja (G_z) takiego układu to iloczyn transmitancji dwóch pojedynczych zbiorników i wzmocnienia o wartości k_1.



Rys. 3.4 Schemat blokowy przedstawiający transmitancję zastępczą modelu G_Z.

Jest tak, ponieważ sygnał wejściowy przemnożony przez transmitancję $G_1(s)$ daje na wyjściu sygnał $H_1(s)$, a więc poziom wody w pierwszym zbiorniku. Wejściem do drugiego musi być natomiast natężenie wody wypływającej z pierwszego

reprezentowane przez sygnał $Q_1(s)$. Aby go uzyskać należy pomnożyć poziom wody w pierwszym zbiorniku przez stałą k_1 zgodnie ze wzorem (3.4). Jest to jednoznaczne z podzieleniem transmitancji G_1 przez K_1, co może wydawać się bezsensowne mając na uwadze to jak wygląda wzór (3.14). Celem takiego działania jest dostęp do sygnału reprezentującego poziom wody w zbiorniku pierwszym i było ono zamierzone. Transmitancję zastępczą całego obiektu przedstawia zatem następujący wzór:

$$G_z = G_1(s) \cdot k_1 \cdot G_2(s) \tag{3.16}$$

Na podstawie [9].

3.3 Parametryzacja.

Założono następujące przepływy q przy następujących poziomach wody h w zbiornikach:

$$\begin{aligned} h_1 = 2\ [m] \ &\rightarrow q_1 = 1 \left[\frac{m^3}{s}\right] \\ h_2 = 4{,}5\ [m] \ &\rightarrow q_2 = 1{,}5 \left[\frac{m^3}{s}\right] \end{aligned} \tag{3.17}$$

oraz następujące pola przekrojów zbiorników:

$$A_1 = 0{,}5\ [m^2], A_2 = 0{,}833\ [m^2] \tag{3.18}$$

Stałe:

$$g = 9{,}81\ \left[\frac{m}{s^2}\right], \rho_w = 997\ \left[\frac{kg}{m^3}\right] \tag{3.19}$$

Ze wzorów (3.1) i (3.3):

$$\begin{aligned} c_1 &= \frac{q_1}{\rho \cdot g \cdot h_1} = 5{,}11 \cdot 10^{-5}\ [\frac{m^4 \cdot s}{kg}] \\ c_2 &= \frac{q_2}{\rho \cdot g \cdot h_2} = 3{,}41 \cdot 10^{-5}\ [\frac{m^4 \cdot s}{kg}] \end{aligned} \tag{3.20}$$

Ze wzoru (3.5):

$$\begin{aligned} k_1 &= \frac{1}{2}\left[\frac{m^2}{s}\right] \\ k_2 &= \frac{1}{3}\left[\frac{m^2}{s}\right] \end{aligned} \tag{3.21}$$

Po podstawieniu wyników (3.21) i (3.18) do wzorów (3.13):

$$\begin{aligned} K_1 = 2\ \left[\frac{s}{m^2}\right], K_2 = 3\ \left[\frac{s}{m^2}\right] \\ T_1 = 1\ [s], T_2 = 2{,}5\ [s] \end{aligned} \tag{3.22}$$

Po podstawieniu wyników (3.22) do wzorów (3.14) i (3.15) otrzymano ostateczne postaci transmitancji:

$$G_1(s) = \frac{2}{s+1} \tag{3.23}$$

$$G_2(s) = \frac{3}{2{,}5 \cdot s + 1} \tag{3.24}$$

Na podstawie [9].

3.4 Implementacja.

W celu implementacji algorytmu spełniającego funkcję członu inercyjnego pierwszego rzędu zamodelowano cyfrowo system ciągły. Określono przybliżoną transmitancję Z układu stosując zasadę ekwiwalentowania całki według niejawnej metody prostokątów(ang. backward Euler)[3]:

$$s = \frac{z-1}{T_p \cdot z} \tag{3.25}$$

gdzie T_p - okres próbkowania.

Po podstawieniu (3.25) do (3.12) otrzymano transmitancję układu dyskretnego:

$$G(z) = \frac{K \cdot T_p \cdot z}{(T_p + T) \cdot z - T} \tag{3.26}$$

W celu określenia równania różnicowego podzielono licznik i mianownik przez najwyższą potęgę zmiennej *z:*

$$\frac{Y(z)}{X(z)} = \frac{K \cdot T_p}{T_p + T - T \cdot z^{-1}} \tag{3.27}$$

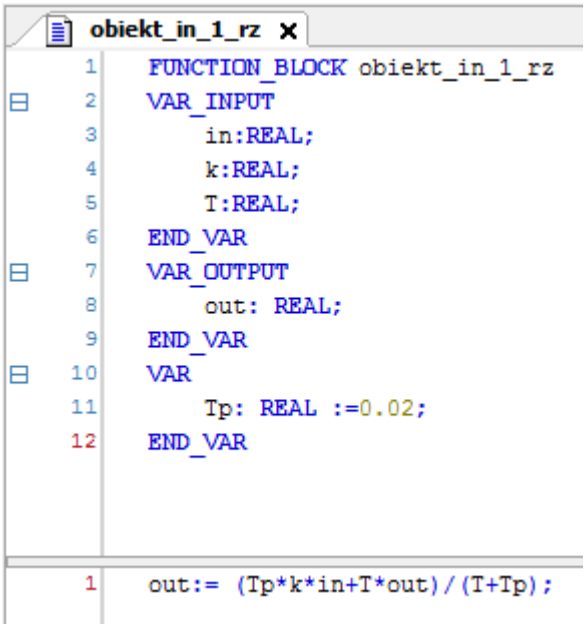
Stąd:

$$(T_p + T) \cdot Y(z) - T \cdot Y(z) \cdot z^{-1} = K \cdot T_p \cdot X(z) \tag{3.28}$$

Postać czasowa:

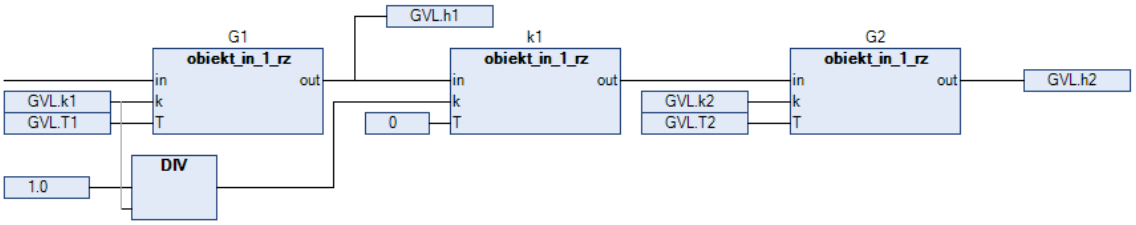
$$Y(z) = \frac{K \cdot T_p \cdot X(k) + T \cdot Y(k-1)}{T_p + T} \tag{3.29}$$

W takiej wersji zostało ono zaimplementowane w programie w języku ST jako Function Block o nazwie *obiekt_in_1_rz*. Czas próbkowania T_p wynosi tyle samo co czas wywołania programu sterownika (20ms).[8]



Rys. 3.5 Implementacja obiektu inercyjnego I rzędu w CODESYS.

Obiekt sterowania został zamodelowany zgodnie z rys. 3.4 i wzorem (3.16). Function Block *obiekt_in_1_rz* pełni również funkcję wzmocnienia k1 dzięki ustaleniu w nim stałej czasowej T równej 0 i wzmocnienia k równego 1/K1. Zmienne globalne GVL pozwolą na wykorzystanie pożądanych wielkości w czasie wizualizacji. Bloczek *DIV* odpowiada za operację dzielenia.



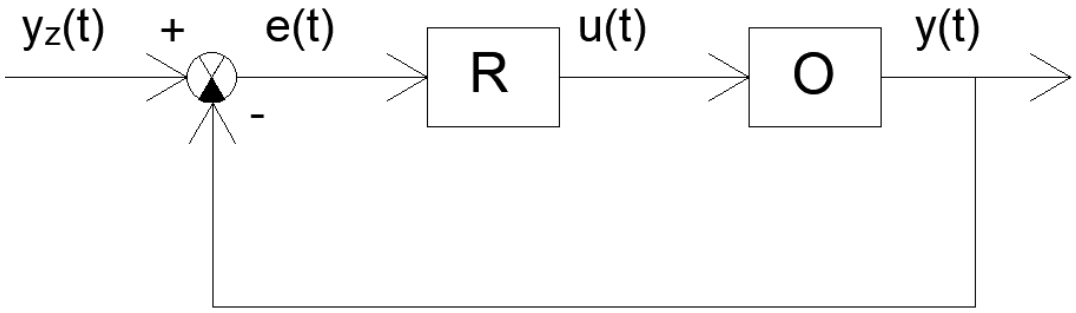
Rys. 3.6 Realizacja modelu obiektu sterowania w CODESYS.

4 Regulator PID.

4.1 Wprowadzenie teoretyczne

Układ regulacji automatycznej to taki układ, który zapewnia wymaganą zmienność wielkości regulowanej bez ingerencji człowieka. Istotnym elementem jest sprzężenie zwrotne, które pozwala na kontrolowanie tej wielkości w celu wyznaczenia uchybu, aby następnie móc go minimalizować. Uchybem regulacji nazywa się różnice wartości zadanej i regulowanej. Podstawowym zadaniem jakiegokolwiek regulatora jest

zapewnienie pożądanego zachowania obiektu poprzez odpowiednią zmianę wielkości sygnału sterującego.[4]

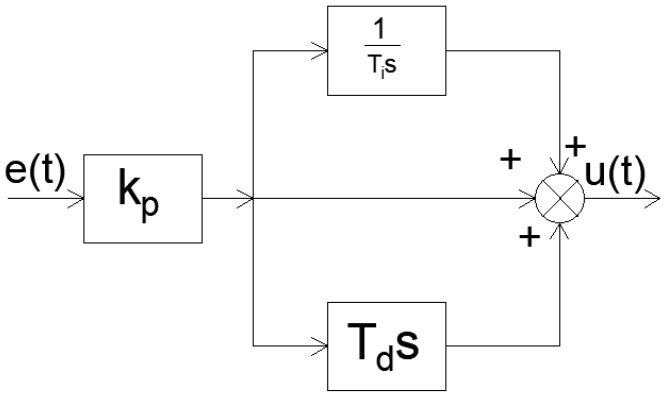


Rys. 4.1 Schemat blokowy regulacji automatycznej. R – regulator, O – obiekt sterowania, $y_z(t)$ – wartość zadana wielkości regulowanej, e(t) – uchyb regulacji, u(t) – wielkość sterująca, y(t) – wielkość regulowana.

Regulatory można podzielić ze względu na ich właściwości dynamiczne na następujące rodzaje:

- P – regulatory proporcjonalne,
- I – regulatory całkujące,
- PI – regulatory proporcjonalno-całkujące,
- PD – regulatory proporcjonalno-różniczkujące,
- PID – regulatory proporcjonalno-całkująco-różniczkujące.

Włączenie do układu regulatora ze sprzężeniem zwrotnym powinno spowodować optymalną pracę tego układu. Połączenie sterowania proporcjonalnego, całkowego i różniczkowego to regulator PID. Zalety takiego połączenia w stosunku do konfiguracji wymienionych na liście przed nim to zmniejszenie błędów stanów ustalonych i przejściowych, jednoczesne zachowanie stabilności i tłumienia na dostatecznym poziomie. [5] Ogólną strukturę PID można sprowadzić do równoległej kombinacji trzech podstawowych członów, czyli P, I, oraz D.



Rys. 4.2 Schemat blokowy przedstawiający równoległą strukturę idealnego regulatora PID, odpowiadający transmitancji tego regulatora.

Transmitancja takiego układu wyraża się wzorem:

$$G_r(s) = k_p\left(1 + \frac{1}{T_i \cdot s} + s \cdot T_d\right) \tag{4.1}$$

Transmitancja operatorowa *G(s)* to jedno z podstawowych pojęć w teorii układów regulacji automatycznej. Nazywa się tak stosunek transformaty odpowiedzi *Y(s)* do transformaty wymuszenia *X(s)* przy zerowych warunkach początkowych. Biorąc pod uwagę układ liniowy o stałych parametrach skupionych o jednym wejściu *x(t)* i jednym wyjściu *y(t)*.[4]

Struktura z rys. 4.2 umożliwia nastawianie poszczególnych parametrów regulacji bez wpływu na pozostałe nastawy. Nastawy regulatora to: wzmocnienie proporcjonalne k_p, stała czasową całkowania (czas zdwojenia) T_i i stała czasowa akcji różniczkowania (czas wyprzedzenia) T_d. Idealny algorytm PID to algorytm astatyczny, co oznacza, że jego odpowiedź na skokową funkcję Heaviside'a nie stabilizuje się na stałym poziomie. Nie ma on realizacji technicznej.[2] Dla liniowych regulatorów o działaniu ciągłym z idealnym elementem różniczkującym postać czasowa przedstawia się następująco:

$$u(t) = k_p \cdot \left(e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau + T_d \cdot \frac{de(t)}{dt}\right) \tag{4.2}$$

Charakterystykę czasową układu otrzymuje się wprowadzając na wejście układu sygnał w postaci skoku jednostkowego o wartości e_{st}.

$$e(t) = 1(t) \cdot e_{st} \tag{4.3}$$

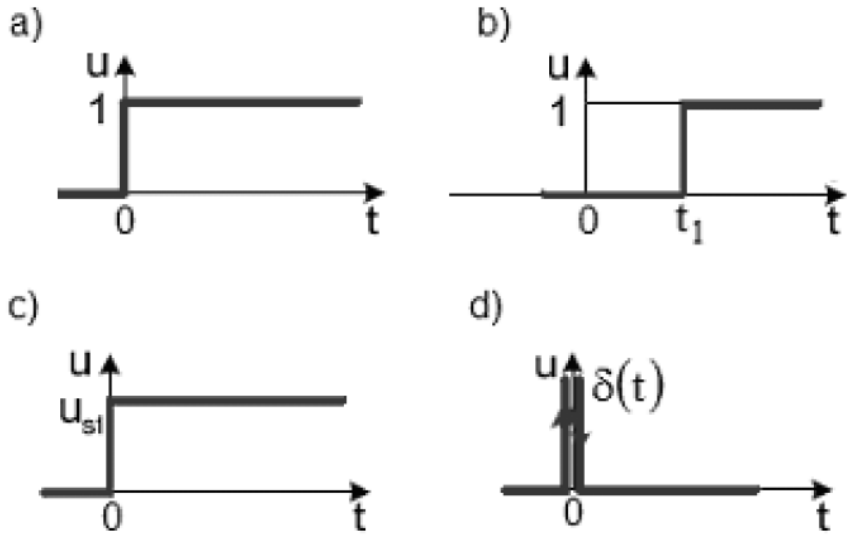
$$u(t) = k_p \cdot e_{st} \cdot \left(1 + \frac{1}{T_i} \cdot t + T_d \cdot \delta(t)\right) \tag{4.4}$$

Wymuszenie jednostkowe (skok jednostkowy, funkcja Heaviside'a):

$$1(t) = \begin{cases} 0 & dla\ t < 0, \\ 1 & dla\ t \geq 0 \end{cases} \tag{4.5}$$

Impuls Diraca (impuls jednostkowy, delta Diraca):

$$\delta(t) = \frac{d1(t)}{dt} \tag{4.6}$$



Rys. 4.3 Sygnały wymuszeń: a) skok jednostkowy, b) skok jednostkowy z przesunięciem, c) skok o dowolnej wartości, d) delta Diraca.[5]

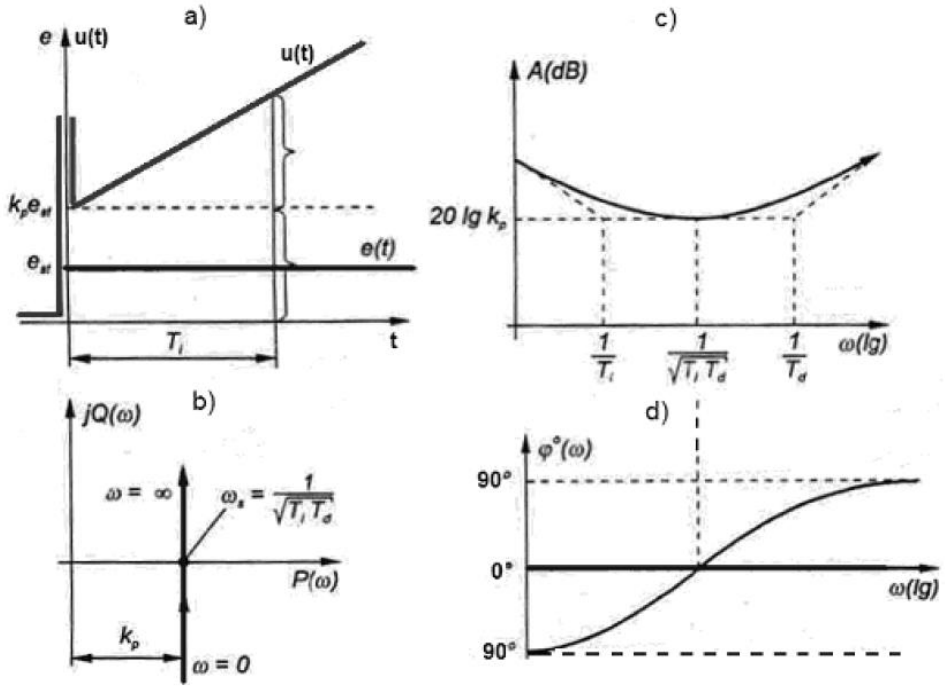
Transmitancja widmowa z (4.1) ma postać:

$$G_r(s) = \frac{u(j\omega)}{e(j\omega)} = k_p\left(1 + \frac{1}{T_i \cdot j\omega} + j\omega \cdot T_d\right) \tag{4.7}$$

Stąd część rzeczywista i urojona wyraża się jako:

$$P(\omega) = k_p, \qquad Q(\omega) = k_p \cdot \left(T_d \cdot \omega - \frac{1}{T_i \cdot \omega}\right) \tag{4.8}$$

Na podstawie powyższych równań otrzymuje się charakterystykę czasową (skokową) i charakterystyki częstotliwościowe.



Rys. 4.4 Charakterystyki: a) czasowa, b) amplitudowo-fazowa, c) logarytmiczna amplitudowa, d) logarytmiczna fazowa.[5]

4.2 Problem nasycenia regulatora.

Do tej chwili zjawiska zachodzące w układzie sterowania były analizowane przy założeniu, że jego poszczególne elementy mają charakter liniowy. Nie ulega jednak wątpliwości, że sygnał sterujący *u(t)*, który jest podawany z regulatora na obiekt regulacji, ma pewne ograniczenia jeśli chodzi o jego wartości. Są to ograniczenia techniczne, na przykład dopuszczalna wartość napięcia, prądu, ciśnienia, prędkości, temperatury i innych wielkości fizycznych reprezentujących wielkość sterującą w układzie sterowania. Aby uwzględnić ten fakt, w różnych miejscach układu regulacji stosuje się nieliniowe bloki: dyskryminatory (ograniczniki).

Rys. 4.5 Dyskryminator (ogranicznik).

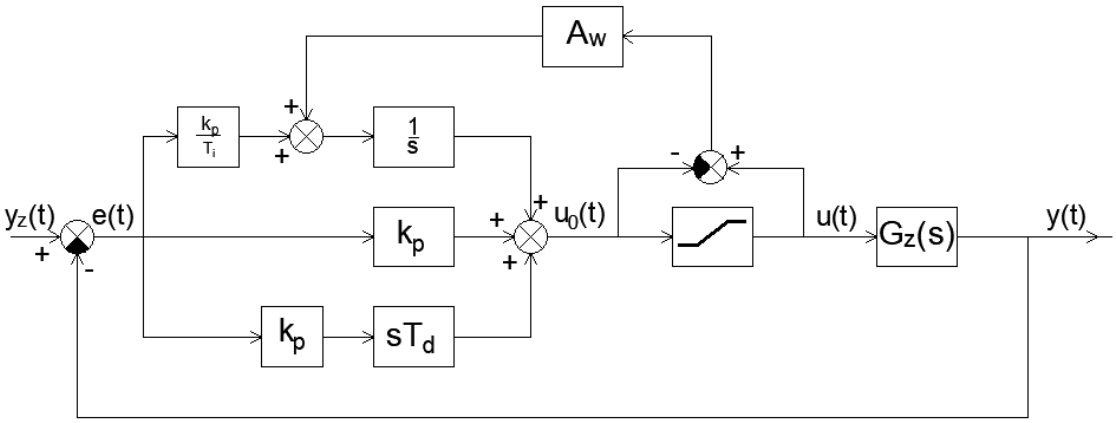
Określenie dolnego i górnego ograniczenia sygnału sterującego powoduje jego „nasycenie". Jeśli sygnał na wyjściu regulatora $u_0(t)$ osiągnie wartość większą od wartości górnego ograniczenia, lub mniejszą od wartości dolnego ograniczenia, to nie zostanie on podany na obiekt sterowania. W takim przypadku sygnał sterujący *u(t)* przyjmie jedną z wartości skrajnych ogranicznika. Uchyb regulacji nie będzie zmniejszał się przez to tak szybko jak wynikałoby to z algorytmu sterowania. Człon całkujący, który sumuje kolejne wartości uchybu przyczynia się do intensyfikacji tego efektu. Dyskryminatory mogą znajdować się także wewnątrz regulatora chroniąc jego elementy przed uszkodzeniem i nie muszą ograniczać wartości sygnału, ale na przykład szybkość jego narastania.

„Nasycenie integratora (ang. integrator windup) objawia się w postaci dużych przeregulowań i długich czasów ustalania, widocznych w odpowiedziach czasowych układu regulacji. Zjawisko to obserwuje się zazwyczaj podczas startu układu lub dużej zmiany wielkości zadanej, gdy występuje znaczna zmiana uchybu regulacji. Przytoczony termin angielski (windup - nakręcić) nawiązuje do całkowicie ściśniętej sprężyny, która w tym stanie traci swoje właściwości dynamiczne. Powrót do normalnego stanu wymaga redukcji ściskającej siły. Zjawisko to występuje w członie całkującym, gdy duży sygnał wymuszający (co objawia się jako składowa stała lub o długim okresie oscylacji) powoduje wystąpienie także dużego wyjściowego sygnału regulatora (nasycenie). Odwrócenie tej tendencji jest niemożliwe w krótkim odcinku czasu, co prowadzi do zaniku sterowania.

Profesjonalne regulatory PID są wyposażone w układy przeciwdziałające nasyceniu integratora. Ogólnie, ich działanie polega na detekcji wystąpienia nasycenia (lub zbliżania się do poziomu nasycenia) i blokowania na ten czas integratora.” [5]

Rozwiązanie problemu nasycenia integratora jest przedstawione na rys. 4.6. Polega ono na pomnożeniu różnicy sygnałów przed ogranicznikiem $u_0(t)$ i za ogranicznikiem $u(t)$, przez współczynnik A_w, a następnie dodaniu powstałego w ten sposób sygnału do sygnału w gałęzi członu I regulatora przed całkowaniem. Jeśli sygnał wychodzący z regulatora mieści się w ramach zdefiniowanych przez wartości skrajne ogranicznika, to system anti-windup nie ma wpływu na pracę układu. Natomiast, jeśli sygnał wychodzący z regulatora przekroczy któreś z ograniczeń, to różnica sygnałów $u_0(t)$ i $u(t)$ wzmocniona przez współczynnik A_w osiągnie wartość o znaku przeciwnym do uchybu $e(t)$ i zmniejszy efekt całkowania, eliminując efekt nasycenia. Wartość tego współczynnika można obliczyć ze wzoru [7] :

$$A_w = \frac{1}{\sqrt{T_i \cdot T_d}} \tag{4.9}$$



Rys. 4.6 Schemat blokowy układu sterowania z systemem anti-windup.

4.3 Implementacja

Algorytm numerycznej reprezentacji regulatora uzyskano przez bezpośrednie zastosowanie numerycznych metod całkowania i różniczkowania do ciągłej postaci regulatora (4.2). Formę zmodyfikowano nieznacznie wymnażając składniki sumy w nawiasie przez współczynnik wzmocnienia k_p, co okaże się zasadne podczas dodawania części algorytmu zapobiegającej nasyceniu integratora:

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i} \cdot \int_0^t e(\tau)d\tau + k_p \cdot T_d \cdot \frac{de(t)}{dt} \tag{4.10}$$

W skrócie:

$$u(t) = u_P(t) + u_I(t) + u_D(t) \tag{4.11}$$

Część proporcjonalna po transformacji Laplace'a:

$$U_P(s) = k_p \cdot E(s) \tag{4.12}$$

Po zamianie na postać dyskretną:

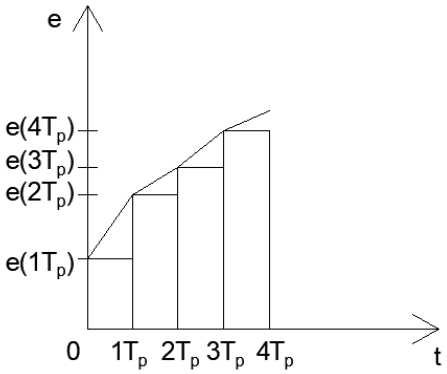
$$U_P(z) = k_p \cdot E(z) \tag{4.13}$$

Po zamianie na postać czasową:

$$u_P(k) = k_p \cdot e(k) \tag{4.14}$$

Przybliżenie całki sumą zgodnie z metodą prostokątów:

$$\int_0^t e(\tau)d\tau \approx T_p \cdot \sum_{m=0}^{k} e(m) \tag{4.15}$$

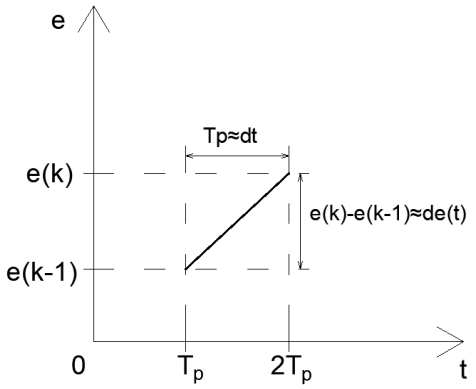


Rys. 4.7 Rysunek poglądowy przedstawiający ideę całkowania numerycznego metodą prostokątów:

$$\int_0^{4 \cdot T_p} e(\tau)d\tau \approx T_p \cdot \sum_{m=0}^{4 \cdot T_p} e(m)$$

Przybliżenie różniczki:

$$\frac{de(t)}{dt} \approx \frac{1}{T_p} \cdot (\, e(k) - e(k-1)) \tag{4.16}$$



Rys. 4.8 Rysunek poglądowy przedstawiający ideę różniczkowania numerycznego.

Po podstawieniu (4.14), (4.15) i (4.16) do (4.10) :

$$u(k) = k_p \cdot e(k) + \frac{k_p \cdot T_p}{T_i} \cdot \sum_{m=0}^{k} e(m) + \frac{k_p \cdot T_d}{T_p} \cdot (\, e(k) - e(k-1)) \qquad (4.17)$$

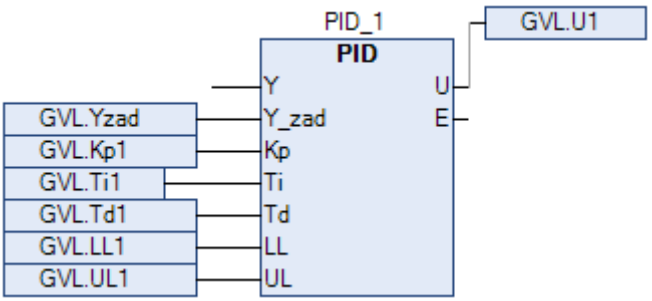
W takiej wersji równanie zostało poszerzone o algorytm anti-windup i zaimplementowane w programie w języku ST jako Function Block o nazwie *PID*. Czas próbkowania T_p wynosi tyle samo co czas wywołania programu sterownika(20ms).

PID

```
FUNCTION_BLOCK PID
VAR_INPUT
    Y: REAL;
    Y_zad: REAL;
    Kp: REAL;
    Ti: REAL;
    Td: REAL;
    LL: REAL;
    UL: REAL;
END_VAR
VAR_OUTPUT
    U: REAL;
    E: REAL;
END_VAR
VAR
    e_k: REAL :=0;
    u_P: REAL :=0;
    u_I: REAL :=0;
    Tp: REAL :=0.02; //czas probkowania (realizacji petli programowej) =20ms=0,02s
    U_diff: REAL :=0;
    u_I_sum: REAL :=0;
    u_D: REAL :=0;
    e_k_1: REAL :=0; //uchyb w chwili poprzedniej
    U_0 :REAL :=0;
END_VAR
```

```
e_k:=Y_zad-Y; //uchyb w danej chwili
E:=e_k; //uchyb w danej chwili wyprowadzony do obl bledu
u_P:=Kp*e_k; // algorytm proporcjonalny
u_I:=Kp*Tp/Ti*e_k+1/SQRT(Ti*Td)*Tp*U_diff; //algorytm calkujacy z anti-windup Aw=1/SQRT(Ti*Td)
u_I_sum:=u_I+u_I_sum; //czesc algorytmu sumujaca wartosc z chwili poprzedniej z aktualna
u_D:=(Kp*Td/Tp)*(e_k-e_k_1); //algorytm rozniczkujacy
U:= u_P+u_I_sum+u_D; //sygnal sterujacy
//anti-windup:
U_0:=U;
//dyskryminator:
IF U >= UL THEN
    U:=UL;
END_IF
IF U <= LL THEN
    U:=LL;
END_IF
//koniec dyskryminatora
U_diff:=U-U_0;
//koniec anti-windup
e_k_1:=e_k; // zapisanie uchybu z danej chwili jako uchybu z chwili poprzedniej
```

Rys. 4.9 Implementacja regulatora PID w CODESYS.



Rys. 4.10 Blok regulatora PID w CODESYS.

5 Regulacja kaskadowa.

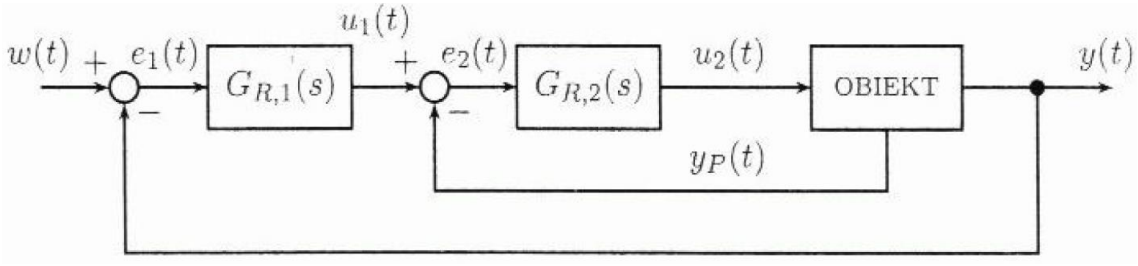
5.1 Struktura regulacji.

„Podstawową strukturą jednowymiarowych układów regulacji jest struktura jednoobwodowa o schemacie blokowym jak na rys. 4.1. Skuteczność tej struktury zależy od właściwości obiektu, działających na obiekt zakłóceń, algorytmu sterowania zastosowanego regulatora i od wymaganej jakości regulacji. Dużą rolę odgrywa także dobór wielkości, za pośrednictwem której regulator oddziałuje na proces. Jeżeli układ o strukturze jednoobwodowej z regulatorem konwencjonalnym o algorytmie P, PI, PD, lub PID spełnia pożądane wymagania jakościowe, to nie należy go komplikować.”[2]

„W przypadku obiektów regulacji z dużymi opóźnieniami, o wysokiej inercji lub o właściwościach nieliniowych, jednoobwodowe układy regulacji z regulatorami PID mogą nie zapewnić uzyskania pożądanej jakości regulacji. Możliwości uzyskania pożądanej jakości regulacji poszukuje się przez dobór innej struktury układu regulacji lub wykorzystanie nietypowych algorytmów sterowania. Jednoobwodowa struktura układu regulacji jest także nieprzydatna w przypadku szeregu specyficznych procesów podlegających regulacji. Typowymi nieelementarnymi strukturami układów regulacji są:

- struktura kaskadowa,
- struktura zamknięto-otwarta,
- struktura prosta układu regulacji stosunku,
- struktura kaskadowa układu regulacji stosunku,
- struktura z pomocniczą wielkością wyjściową obiektu,
- struktura selekcyjna (ang. override control)”.[2]

Idea regulacji kaskadowej polega między innymi na dekompozycji układu sterowania na pętle regulacyjne i znalezienie wartości reagującej na zmiany wymuszenia szybciej niż wielkość sterowana w celu osiągnięcia lepszej jakości regulacji. Kiedy prosty układ sterowania z jednym regulatorem nie spełnia wymogów jakościowych, albo prowadzi do skomplikowania regulatora można zaprojektować stosowną strukturę układu regulacji. Kaskada regulatorów powstaje na skutek doprowadzenia dodatkowej wielkości pomocniczej do regulatora.



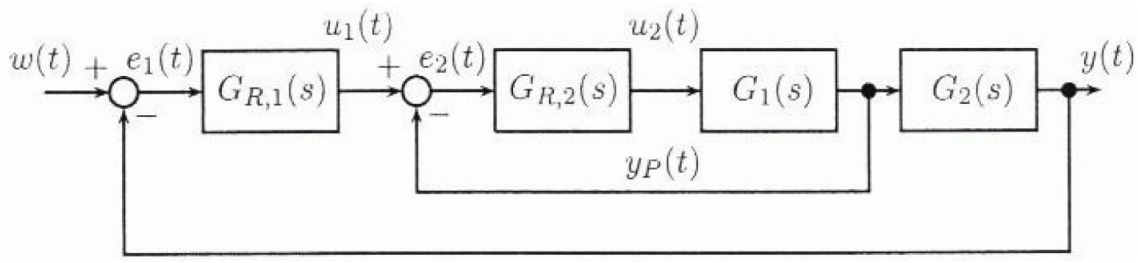
Rys. 5.1 Schemat blokowy układu regulacji kaskadowej, gdzie $G_{R,1}(s)$ – to regulator główny (nadrzędny), a $G_{R,2}(s)$ – regulator pomocniczy (podrzędny).[6]

Cele wykorzystania struktury regulacji kaskadowej:

- Linearyzacja nieliniowości statycznej – kiedy w obiekcie regulacji pojawia się człon nieliniowy. Wpływ nieliniowości powinien zostać zmniejszony poprzez objęcie jej pomocniczym sprzężeniem zwrotnym i doprowadzenie odpowiedniego sygnału do dodanego regulatora.

- Kompensację dynamiki obiektu regulacji – kiedy obiekt regulacji zawiera opóźnienie transportowe lub dużą inercję, pomocnicza wielkość sterująca musi w stanach dynamicznych reagować na zmianę wartości wymuszenia szybciej niż wielkość wyjściowa.

- Poprawa eliminacji zakłóceń – kiedy na obiekt sterowania działają zakłócenia, które muszą być eliminowane przez regulator. W takim przypadku doprowadza się do niego dodatkową informację o ich wartości, co może się wiązać z koniecznością znalezienia wielkości reagującej na zakłócenia szybciej niż wielkość regulowana.[6]



Rys. 5.2 Schemat układu po rozdzielenie obiektu regulacji na dwie części $G_1(s)$ i $G_2(s)$, przy założeniu, że $G_{R,2}(s)$*jest regulatorem liniowym.*[6]

Przy założeniu, że G_1 jest częścią liniową $G_1(s)$ obiektu, regulator główny $G_{R,1}$ steruje obiektem o transmitancji wyrażonej wzorem [6]:

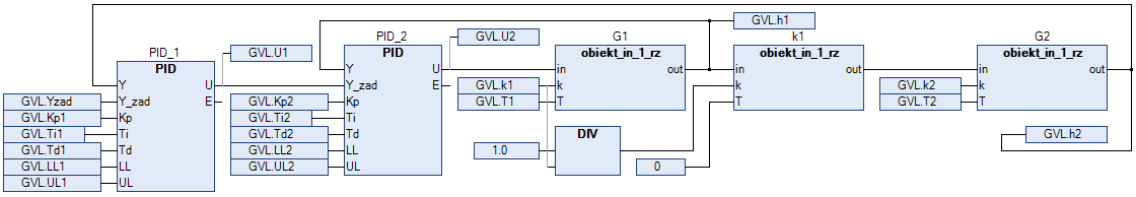
$$G(s) = \frac{Y(s)}{U_1(s)} = G_2(s) \cdot \frac{G_{R,2}(s) \cdot G_1(s)}{1 + G_{R,2}(s) \cdot G_1(s)} \tag{5.1}$$

Transmitancja widmowa obiektu zastępczego dla częstotliwości spełniających warunek $\left|G_{R,2}(j\omega) \cdot G_1(j\omega)\right| \gg 1$ jest określana wzorem [6]:

$$G(j\omega) = \frac{Y(j\omega)}{U_1(j\omega)} = G_2(j\omega) \cdot \frac{G_{R,2}(j\omega) \cdot G_1(j\omega)}{1 + G_{R,2}(j\omega) \cdot G_1(j\omega)} \approx G_2(j\omega) \tag{5.2}$$

Odpowiada to kompensacji dynamiki obiektu $G_1(s)$. [6]

Na potrzeby tej pracy kaskadowy układ regulacji został zbudowany z wcześniej stworzonych bloków funkcyjnych z rys. 3.6 i rys. 4.10:



Rys. 5.3 Realizacja układu regulacji kaskadowej w CODESYS.

5.2 Dobór nastaw regulatorów.

Jednym ze sposobów na dobieranie wartości parametrów k_p, T_i, T_d jest metoda prób i błędów, która w przypadku rzeczywistego procesu technologicznego może okazać się kosztownym rozwiązaniem. Dlatego istnieją ustalone zasady takie jak metoda Zieglera-Nicholsa oparta na odpowiedzi skokowej.

Polega ona na tym, że obiekt regulacji pobudza się skokiem jednostkowym, aby zarejestrować przebieg jego odpowiedzi skokowej. Na tej podstawie określa się czas opóźnienia t_d i czas narastania $t_{90\%}$. Współczynnik tłumienia równy ζ=0.2, który odpowiada przeregulowaniu $\varepsilon_{\%}$=25% powinien zostać osiągnięty przy doborze parametrów według zależności podanych w tabeli:

Tabela 5.1 Dobór nastaw regulatora PID na podstawie odpowiedzi skokowej.[6]

Typ regulatora	**kp**	**T_i**	**T_d**
P	$\frac{t_{90\%}}{t_d}$	∞	0
PI	$0{,}9 \cdot \frac{t_{90\%}}{t_d}$	$\frac{t_d}{0{,}3}$	0
PID	$1{,}2 \cdot \frac{t_{90\%}}{t_d}$	$2 \cdot t_d$	$0{,}5 \cdot t_d$

Czas narastania $t_{90\%}$ to czas wymagany do tego, aby przy wymuszeniu skokiem jednostkowym odpowiedź $y(t)$ obiektu wzrosła od 10% do 90% wartości końcowej $y(\infty)$. Czas opóźnienia t_d to czas wymagany do tego, aby przy wymuszeniu skokiem jednostkowym odpowiedź $y(t)$ obiektu osiągnęła 10% wartości końcowej $y(\infty)$. Przeregulowanie ε to stosunek maksymalnego uchybu przejściowego o znaku przeciwnym do maksymalnego uchybu początkowego *e1* do wartości maksymalnego uchybu początkowego *e0*. Przeregulowanie w procentach wyraża wzór [6]:

$$\varepsilon_{\%} = \frac{e_1}{e_0} \cdot 100\% \tag{5.3}$$

5.3 Jakość regulacji.

Zadaniem układu automatycznej regulacji jest eliminacja lub zmniejszenie uchybu regulacji spowodowanego zmianą wartości zadanej, czy też zakłóceniami. Spełnienie tylko tego zadania może być niewystarczające jeżeli czas regulacji jest bardzo długi, albo osiągnięcie wartości zadanej jest poprzedzone wieloma przeregulowaniami o znacznych amplitudach. Z tego powodu od układów regulacji wymaga się odpowiedniego zachowania w stanach przejściowych (dynamicznych) i stanach ustalonych. Są to wymagania dotyczące jakości regulacji. Z potrzeby liczbowego jej wyrażenia powstało wiele kryteriów (wskaźników) jakości. Najbardziej kompleksowe wydają się wskaźniki całkowe, ponieważ charakteryzują one w określony sposób całość przebiegu przejściowego. Stosowanie ich pozwala określić miarę jakości przez wielkości tzw. pola regulacji. Jego minimalizacja równa się maksymalizacji jakości dynamicznej układu. [2]

„Najprostszym wskaźnikiem jest całka względem czasu z wartości bezwzględnej uchybu (IAE – integraf of the absolute magnitude of the terror), równa :

$$J_1 = \int_0^{\infty} |e(t)|\, dt \tag{5.4}$$

Wyrażenie określające wskaźnik J_1 rośnie z upływem czasu t niezależnie od znaku uchybu, natomiast obranie go jako kryterium projektowego skutkuje uzyskaniem układu o słabym tłumieniu.

Kolejnym wskaźnikiem jest całka względem czasu z kwadratu uchybu (ISE – integraf of the squared terror), równa:

$$J_2 = \int_0^{\infty} e^2(t)\, dt \tag{5.5}$$

Wskaźnik J_2, podobnie jak J_1, rośnie z upływem czasu niezależnie od znaku uchybu, jednak szybciej dla uchybów większych od jedności.

Pozostałe wskaźniki w kolejności: J_3 (ITAE – *integraf of the time multiplied by the absolute value of the error*), J_4 (ITSE – *integral of the time multiplied by the squared error*), J_5 (ISTAE – *integral of the squared time multiplied by the absolute value of the error*), J_6 (ISTSE – *integral of the squared time multiplied by the squared error*)"[6], są określone wzorami [6]:

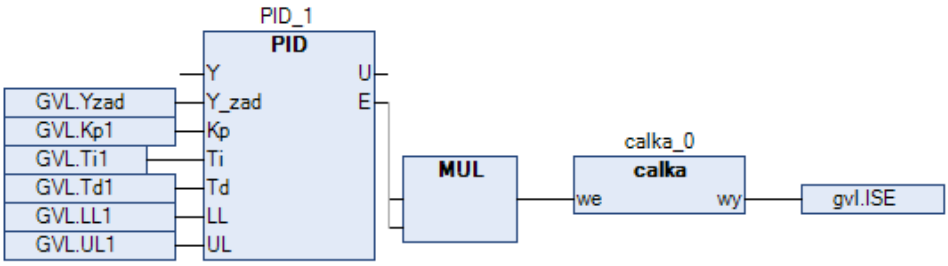
$$J_3 = \int_0^\infty t \cdot |e(t)|\, dt \tag{5.6}$$

$$J_4 = \int_0^\infty t \cdot e^2(t)\, dt \tag{5.7}$$

$$J_5 = \int_0^\infty t^2 \cdot |e(t)|\, dt \tag{5.8}$$

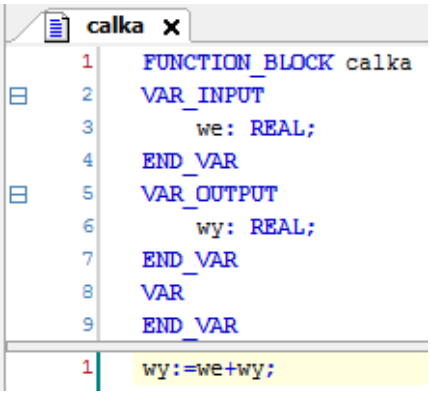
$$J_6 = \int_0^\infty t^2 \cdot e^2(t)\, dt \tag{5.9}$$

W projekcie zdecydowano się na pomiar jakości regulacji przy użyciu wskaźnika ISE opisanego wzorem (5.5). Jego realizacja w CODESYS wygląda następująco:



Rys. 5.4 Realizacja pomiaru wskaźnika ISE w CODESYS.

Gdzie wyjście E z bloku *PID_1* jest aktualna wartością uchybu sterowania, bloczek *MU*L odpowiada za mnożenie, a blok *calka* jest zrealizowany tak:



Rys. 5.5 Implementacja całkowania w CODESYS.

6 PLC.

„Sterowniki programowalne PLC (Programmable Logic Controllers) są komputerami przemysłowymi, które pod kontrolą systemu operacyjnego czasu rzeczywistego:

• zbierają pomiary za pośrednictwem modułów wejściowych z analogowych i dyskretnych czujników oraz urządzeń pomiarowych,

• transmitują dane za pomocą modułów i łącz komunikacyjnych,

• wykonują programy aplikacyjne na podstawie przyjętych parametrów i uzyskanych danych o sterowanym procesie lub maszynie,

• generują sygnały sterujące zgodnie z wynikami obliczeń tych programów i przekazują je poprzez moduły wyjściowe do elementów i urządzeń wykonawczych,

• realizują funkcje diagnostyki programowej i sprzętowej.

Wartości pomiarów zmiennych procesowych są wejściami sterownika, zaś obliczone zmienne sterujące stanowią wyjścia sterownika.

Głównym zadaniem sterownika jest więc reagowanie na zmiany wejść przez obliczanie wyjść według zaprogramowanych reguł sterowania lub regulacji. Reakcja ta może być zależna od wyników operacji arytmetyczno-logicznych wykonanych dla aktualnych wartości wejść sterownika, jego zmiennych wewnętrznych oraz od zaprogramowanych warunków czasowych. Może ona także zależeć od operacji wykonanych na danych transmitowanych w sieciach łączących wiele elementów pomiarowych, sterowników, regulatorów czy też komputerów.” [19]

Swoją popularność sterowniki PLC zawdzięczają między innymi możliwości ich programowania, a więc realizowanie systemu sterowania tylko przez wprowadzenie odpowiedniego algorytmu. Robi się to zwykle za pomocą komputera wyposażonego w odpowiednie oprogramowanie lub rzadziej programatora klawiaturowego. Najbardziej znanym i powszechnym językiem programowania jest język drabinkowy, ze względu na jego podobieństwo do projektowania klasycznych systemów sterowania stykowo-przekaźnikowego wykorzystywanego wcześniej przez inżynierów elektryków. Na przestrzeni lat powstawało wiele innych sposobów opisywania algorytmów sterowania, które różniły się od siebie w zależności od producenta. Rozpowszechnienie sterowników PLC spowodowało konieczność standaryzacji metod ich programowania. W 1993 roku powstała więc norma IEC 1131. Aktualnie obowiązuje jej trzecia wersja IEC 61131 z 2013 roku. Jej trzecia część dotyczy języków programowania i jest

ujednoliceniem tej koncepcji, aby można było programować różne systemy PLC w oparciu o jednakowe zasady. Zdefiniowane zostały pojęcia podstawowe, zasady ogólne, model programowy i model komunikacyjny realizujący wymianę danych między elementami oprogramowania, oraz podstawowe typy i struktury danych.

Języki tekstowe:

- IL (Instructoin List) – język listy instrukcji, odpowiednik języka typu assembler, jego zbiór instrukcji obejmuje operacje logiczne, arytmetyczne, operacje relacji, funkcje przerzutników, czasomierzy, liczników itp.

- ST (Structured Text) – język strukturalny, odpowiednik języka algorytmicznego wysokiego poziomu, zawiera struktury programowe takie jak: if…then…else…end_if, case…of…end_case, for…to…do…end_for, while…do…end_while, repeat…end_repeat. (Ten język został użyty w projekcie.)

Języki graficzne:

- LD (Ladder Diagram) – język schematów drabinkowych, podobny do obwodów przekaźnikowych, dopuszcza użycie funkcji: arytmetycznych, logicznych, porównań i relacji, bloków funkcyjnych: przerzutników, czasomierzy, liczników, regulatora PID, bloków programowych.

- FBD (Function Block Diagram) – język schematów blokowych, odpowiednik schematów przepływu sygnału dla obwodów logicznych przedstawionych formie połączonych bramek logicznych i bloków funkcyjnych takich jak w języku LD.

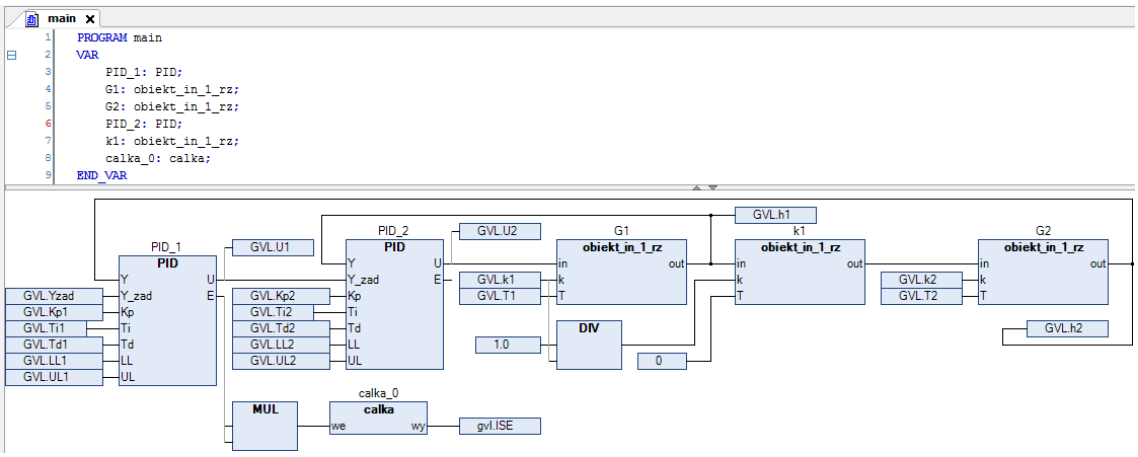
- SFC (Sequential Function Chart) – pozwala na opisywanie zadań sterowania sekwencyjnego za pomocą grafów zawierających etapy (step) i warunki przejścia (transition) miedzy etapami, obrazuje strukturę programu, poszczególne elementy programu programowane są w innych językach. [19]

- CFC (Continous Function Chart) – język najwyższego poziomu, algorytm sterowania tworzony jest w funkcji głównej z elementów umieszczanych w dowolnych miejscach ekranu, elementy to wejścia, wyjścia do których można odwoływać się wielokrotnie, bloki wbudowane, wewnątrz których wykonywane są operacje na zmiennych i bloki funkcyjne programowane w dowolnym języku.

7 Realizacja.

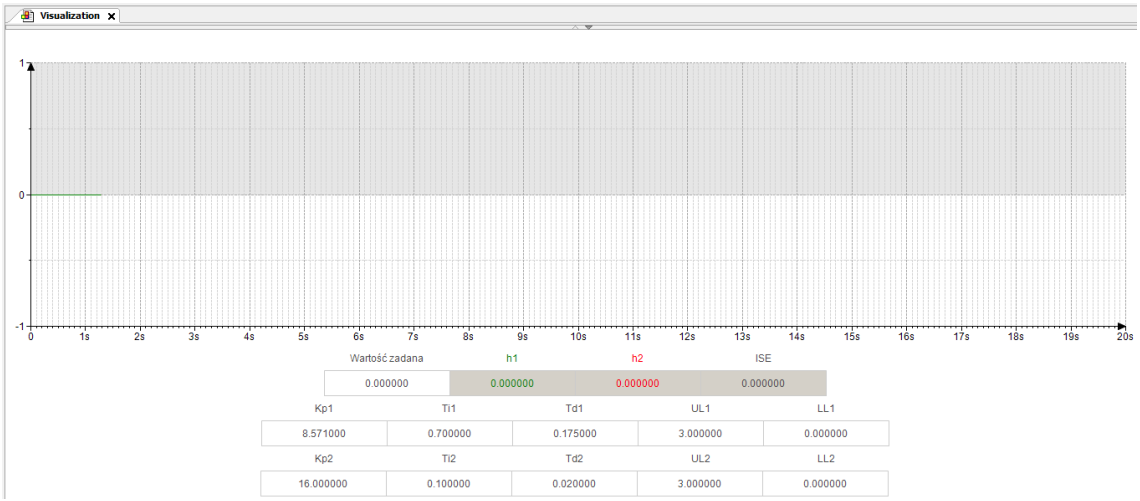
7.1 Finalny wygląd programu.

Funkcja główna programu (*main*) w całości wygląda następująco:



Rys. 7.1 Funkcja główna programu (*main*).

Zmienne globalne GVL pozwolą na wykorzystanie pożądanych wielkości w czasie wizualizacji, której okno prezentuje się tak:

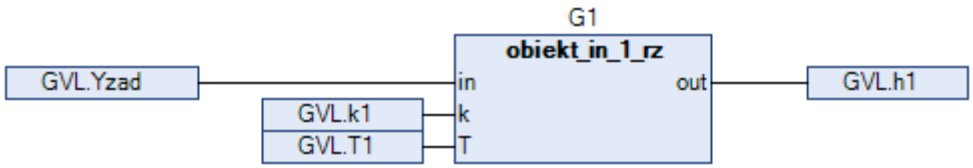


Rys. 7.2 Część programu odpowiadająca za wizualizację (*Visualization*).

Górne ograniczenia sygnałów sterujących przyjęto na poziomie 3 zakładając maksymalne natężenia przepływu równe 3 m^3/s. Dolne ograniczenia są równe 0, ponieważ natężenie przepływu w modelu (rys. 3.3) nie może zmienić znaku (woda może płynąć tylko z góry na dół).

7.2 Nastawy według metody Zieglera-Nicholsa.

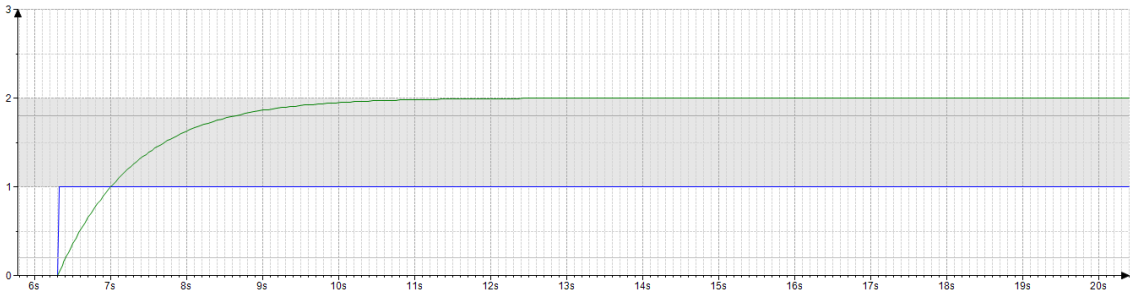
W pierwszej kolejności wyliczono wartości nastaw dla regulatora pomocniczego PID_2. W tym celu zbadano odpowiedź pierwszej części układu na wymuszenie skokiem jednostkowym.



Rys. 7.3 Układ do wyznaczenia odpowiedzi skokowej pierwszej części obiektu.

```
Yzad: REAL :=0;
//obiekty
k1: REAL :=2;
T1: REAL :=1;
//wyk
h1: REAL;
```

Rys. 7.4 Zmienne globalne (pierwsza część obiektu).

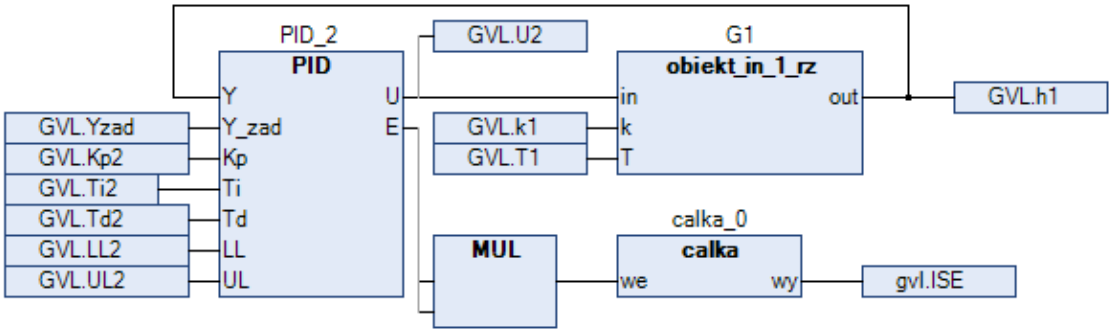


Rys. 7.5 Odpowiedź skokowa pierwszej części obiektu (metoda Z-N).

Wyniki obliczeń na podstawie tabeli 5.1 i rys. 6.4:

Tabela 7.1 Parametry nastawy PID_2 według metody Z-N.

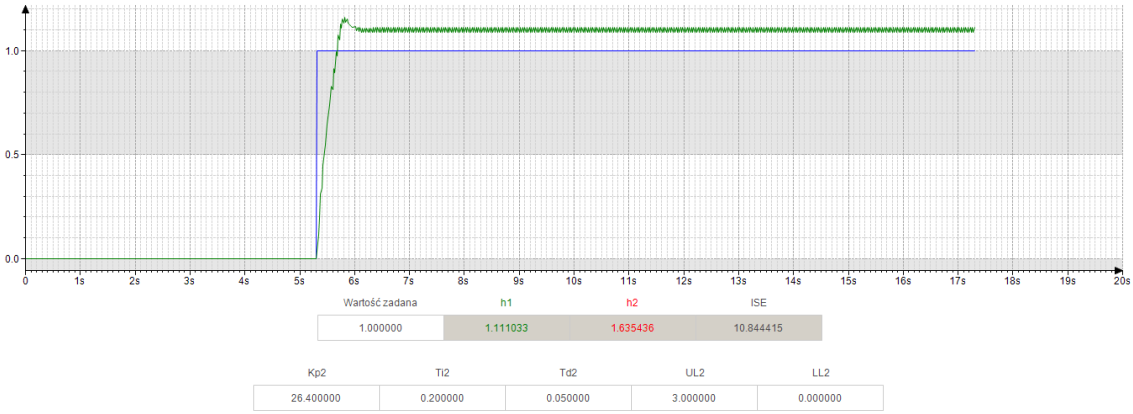
t_d	$t_{90\%}$	k_p	T_i	T_d
0,1	2,2	26,4	0,2	0,05



Rys. 7.6 Układ regulacji pierwszej części obiektu.

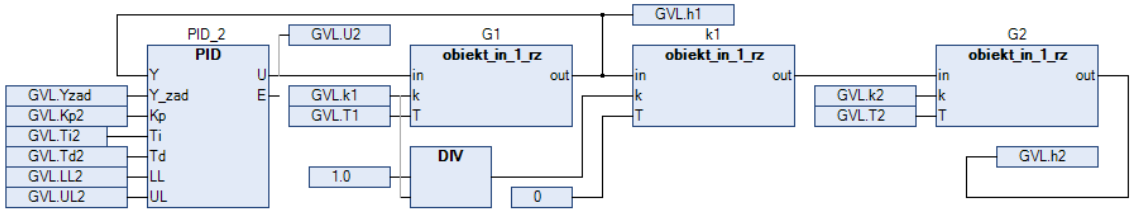
```
Yzad: REAL :=0;
//pid2
Kp2: REAL :=26.4;
Ti2: REAL :=0.2;
Td2: REAL :=0.05;
UL2: REAL :=3;
LL2: REAL :=0;
//obiekty
k1: REAL :=2;
T1: REAL :=1;
//wyk
U2: REAL;
h1: REAL;
ISE: REAL;
```

Rys. 7.7 Zmienne globalne (układ regulacji pierwszej części obiektu) (metoda Z-N).



Rys. 7.8 Odpowiedź układu regulacji pierwszej części obiektu (metoda Z-N).

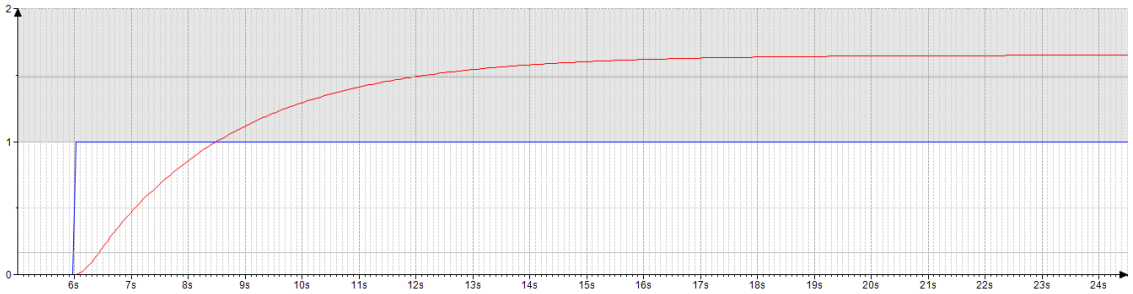
Następnie wyliczono wartości nastaw dla regulatora głównego PID_2. W tym celu potraktowano regulator pomocniczy i obiekt sterowania jako obiekt zastępczy. Zbadano odpowiedź takiego obiektu na wymuszenie skokiem jednostkowym.



Rys. 7.9 Układ do wyznaczenia odpowiedzi obiektu zastępczego.

```
Yzad: REAL :=0;
//pid2
Kp2: REAL :=26.4;
Ti2: REAL :=0.2;
Td2: REAL :=0.05;
UL2: REAL :=3;
LL2: REAL :=0;
//obiekty
k1: REAL :=2;
T1: REAL :=1;
k2: REAL :=3;
T2: REAL :=2.5;
//wyk
U1: REAL;
U2: REAL;
h1: REAL;
h2: REAL;
ISE: REAL;
```

Rys. 7.10 Zmienne globalne (obiekt zastępczy)(metoda Z-N).

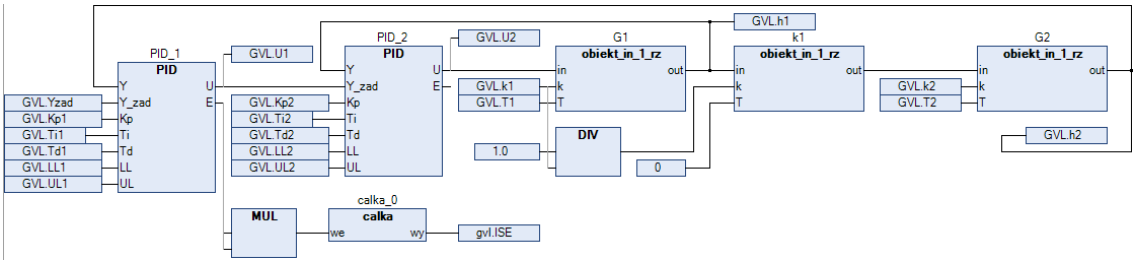


Rys. 7.11 Odpowiedź skokowa obiektu zastępczego.

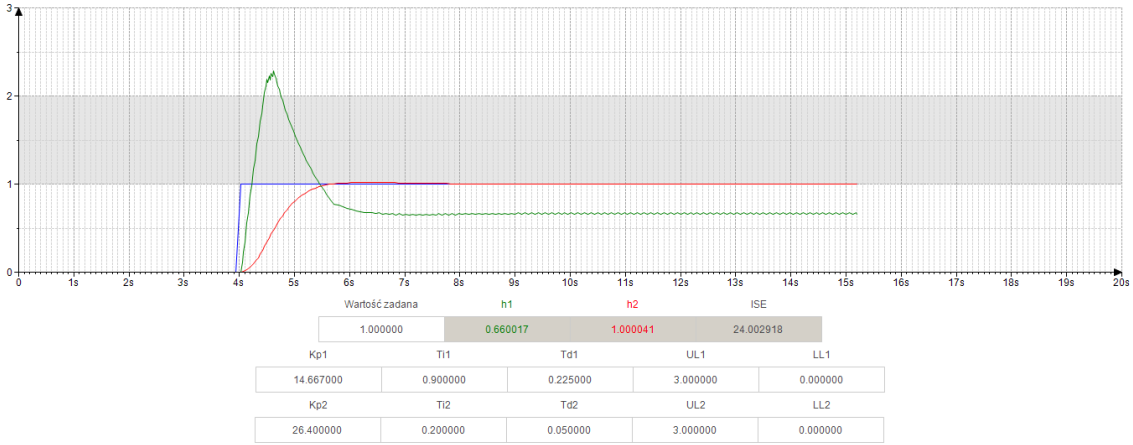
Wyniki obliczeń na podstawie tabeli 5.1 i rys. 6.11:

Tabela 7.2 Parametry nastawy PID_1 według metody Z-N.

t_d	$t_{90\%}$	k_p	T_i	T_d
0,45	5,5	14,667	0,9	0,225



Rys. 7.12 Układ regulacji kaskadowej.



Rys. 7.13 Odpowiedź układu regulacji kaskadowej (metoda Z-N).

7.3 Modyfikacja nastaw.

Postanowiono spróbować uzyskać lepszą jakość regulacji poprzez ręczną modyfikację wartości nastaw regulatorów. Poniżej przedstawiono rezultaty.

Układ do wyznaczenia odpowiedzi skokowej, zmienne globalne pierwszej części obiektu i odpowiedź skokowa pierwszej części obiektu jak na rys. 6.3, rys. 6.4 i rys. 6.5.

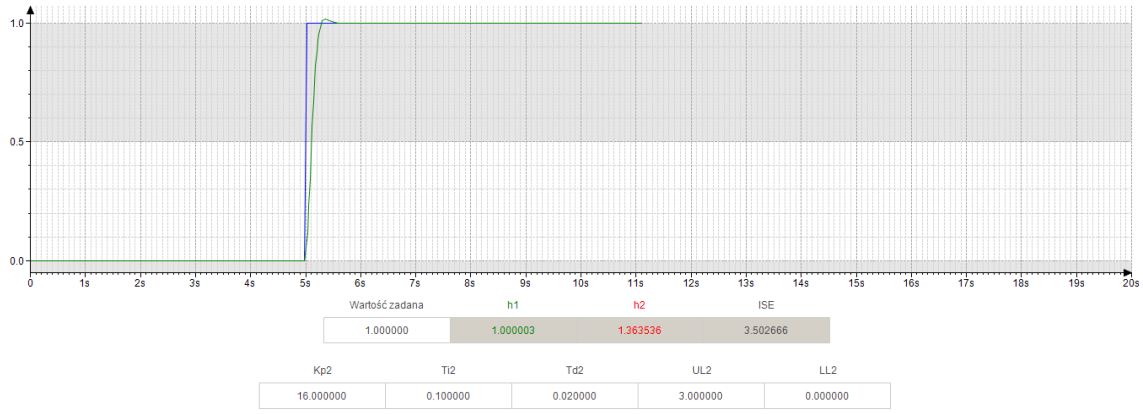
Tabela 7.3 Zmodyfikowane parametry nastawy PID_2.

k_p	T_i	T_d
16	0,1	0,02

Układ regulacji pierwszej części obiektu jak na rys. 6.6.

```
Yzad: REAL :=0;
//pid2
Kp2: REAL :=16;
Ti2: REAL :=0.1;
Td2: REAL :=0.02;
UL2: REAL :=3;
LL2: REAL :=0;
//obiekty
k1: REAL :=2;
T1: REAL :=1;
//wyk
U2: REAL;
h1: REAL;
ISE: REAL;
```

Rys. 7.14 Zmienne globalne (układ regulacji pierwszej części obiektu) (zmod. metoda Z-N).

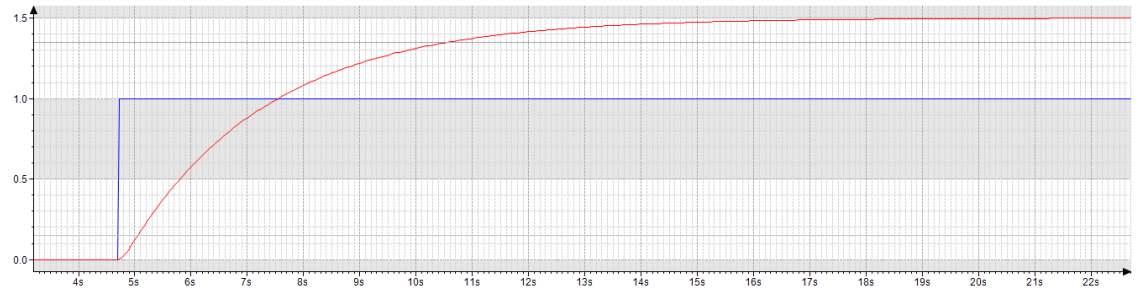


Rys. 7.15 Odpowiedź układu regulacji pierwszej części obiektu (zmod. metoda Z-N).

Następnie wyliczono wartości nastaw dla regulatora głównego PID_2. W tym celu potraktowano regulator pomocniczy i obiekt sterowania jako obiekt zastępczy. Zbadano odpowiedź takiego obiektu na wymuszenie skokiem jednostkowym. Układ do wyznaczenia odpowiedzi obiektu zastępczego jak na rys. 6.9.

```
Yzad: REAL :=0;
//pid2
Kp2: REAL :=16;
Ti2: REAL :=0.1;
Td2: REAL :=0.02;
UL2: REAL :=3;
LL2: REAL :=0;
//obiekty
k1: REAL :=2;
T1: REAL :=1;
k2: REAL :=3;
T2: REAL :=2.5;
//wyk
U1: REAL;
U2: REAL;
h1: REAL;
h2: REAL;
ISE: REAL;
```

Rys. 7.16 Zmienne globalne (obiekt zastępczy)(zmod. metoda Z-N).



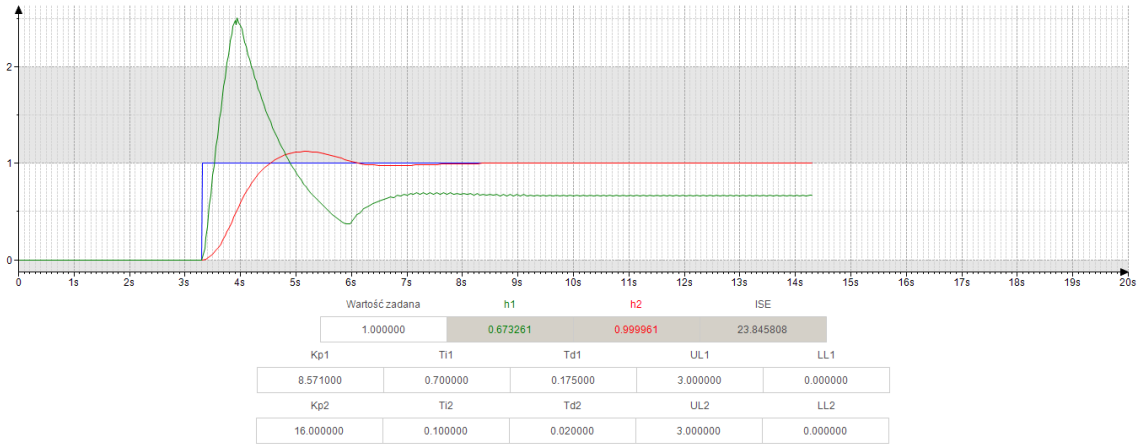
Rys. 7.17 Odpowiedź skokowa obiektu zastępczego.

Wyniki obliczeń na podstawie tabeli 5.1 i rys. 6.17:

Tabela 7.4 Parametry nastawy PID_1 według metody Z-N po modyfikacji nastaw PID_2.

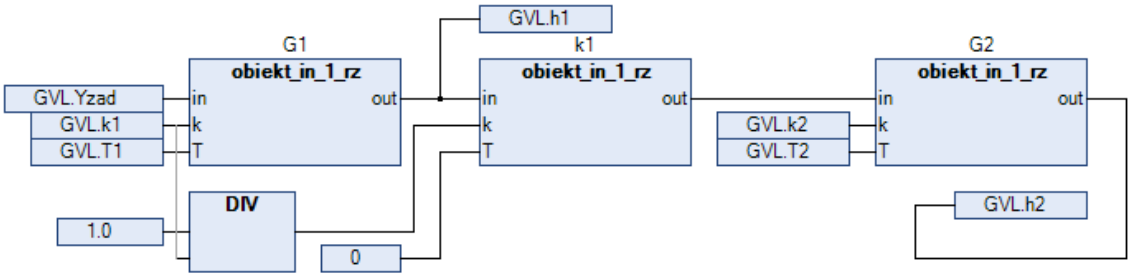
t_d	$t_{90\%}$	k_p	T_i	T_d
0,35	2,5	8,571	0,7	0,175

Układ regulacji jak na rys. 6.12.

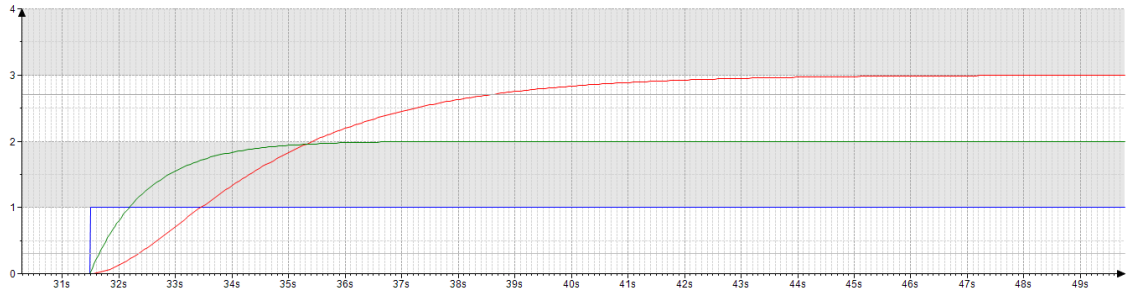


Rys. 7.18 Odpowiedź układu regulacji kaskadowej (zmod. metoda Z-N).

7.4 Struktura jednoobwodowa.



Rys. 7.19 Układ do wyznaczenia odpowiedzi skokowej całego obiektu.

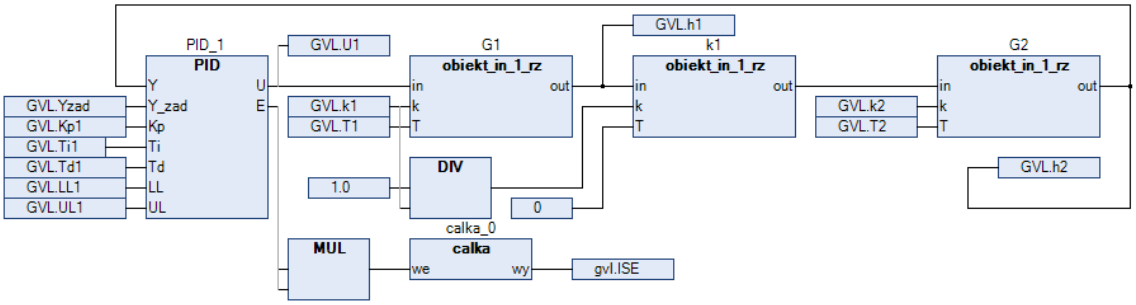


Rys. 7.20 Odpowiedź skokowa całego obiektu.

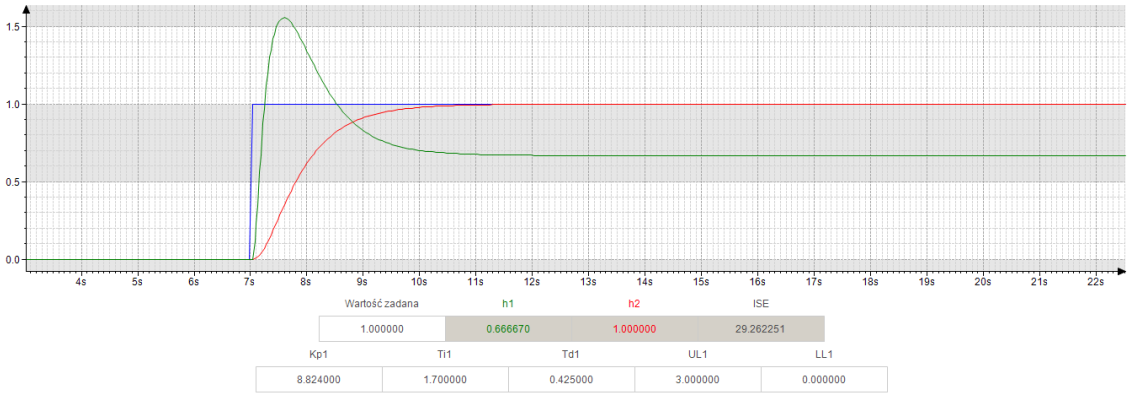
Wyniki obliczeń na podstawie tabeli 5.1 i rys. 6.20:

Tabela 7.5 Parametry nastawy PID_1 w strukturze jednoobwodowej.

t_d	$t_{90\%}$	k_p	T_i	T_d
0,85	6,25	8,824	1,7	0,425



Rys. 7.21 Układ regulacji jednoobwodowej.



Rys. 7.22 Odpowiedź układu regulacji jednoobwodowej.

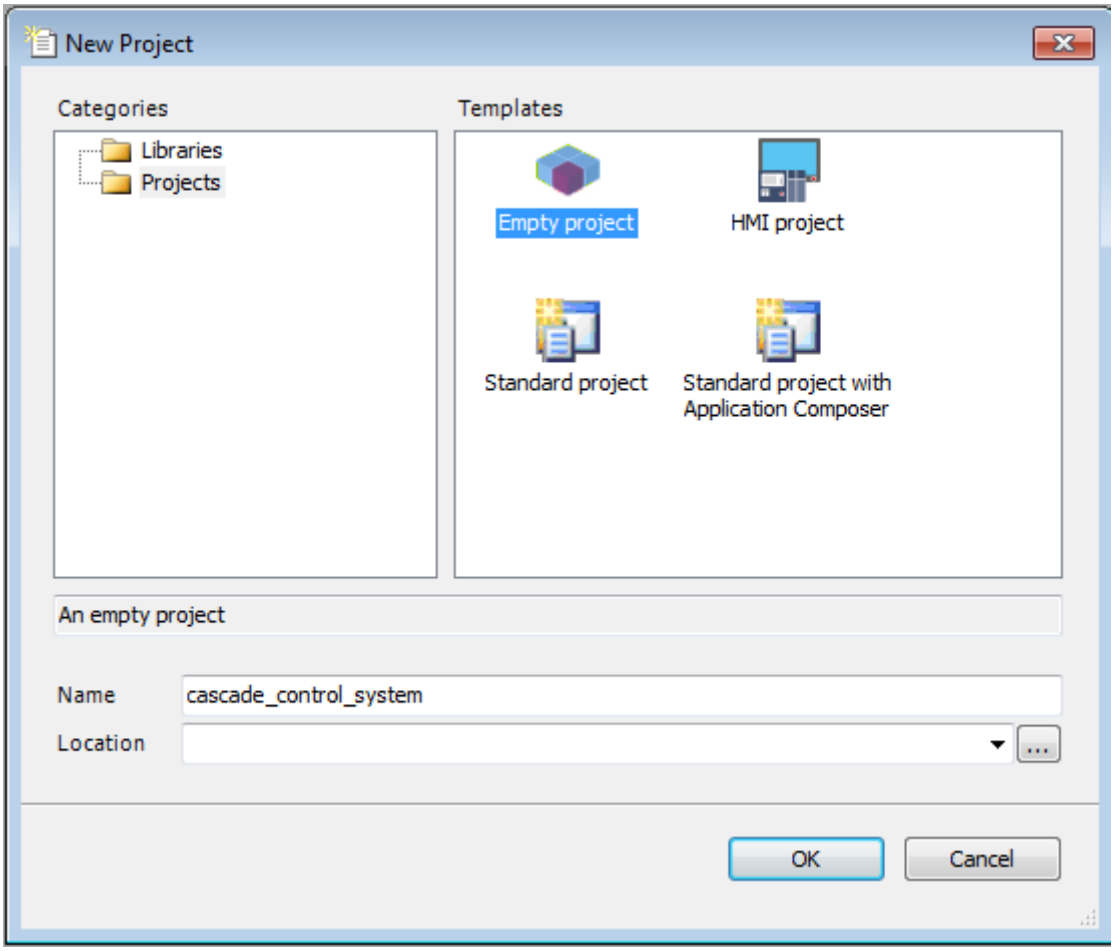
8 CODESYS.

Codesys jest wiodącym, niezależnym od producentów narzędziem programistycznym zgodnym z normą IEC 61131-3. Miliony kompatybilnych urządzeń, ponad 1000 różnych typów urządzeń od ponad 400 producentów i dziesiątki tysięcy użytkowników końcowych świadczą o jego praktyczności. Codesys to platforma oprogramowania dla technologii automatyki przemysłowej. Rdzeniem platformy jest narzędzie programistyczne „CODESYS Development System". Oferuje on użytkownikom praktyczne, zintegrowane rozwiązania do wygodnej konfiguracji aplikacji automatyki. Celem jest zapewnienie im praktycznego wsparcia w codziennych zadaniach. Otwarte interfejsy, kompleksowe funkcje bezpieczeństwa i wygodne połączenie z platformą administracyjną opartą na chmurze sprawiają, że CODESYS jest naturalną platformą Przemysłu 4.0. Producenci urządzeń używają CODESYS

do wdrażania własnych programowalnych lub konfigurowalnych komponentów automatyki. [18]

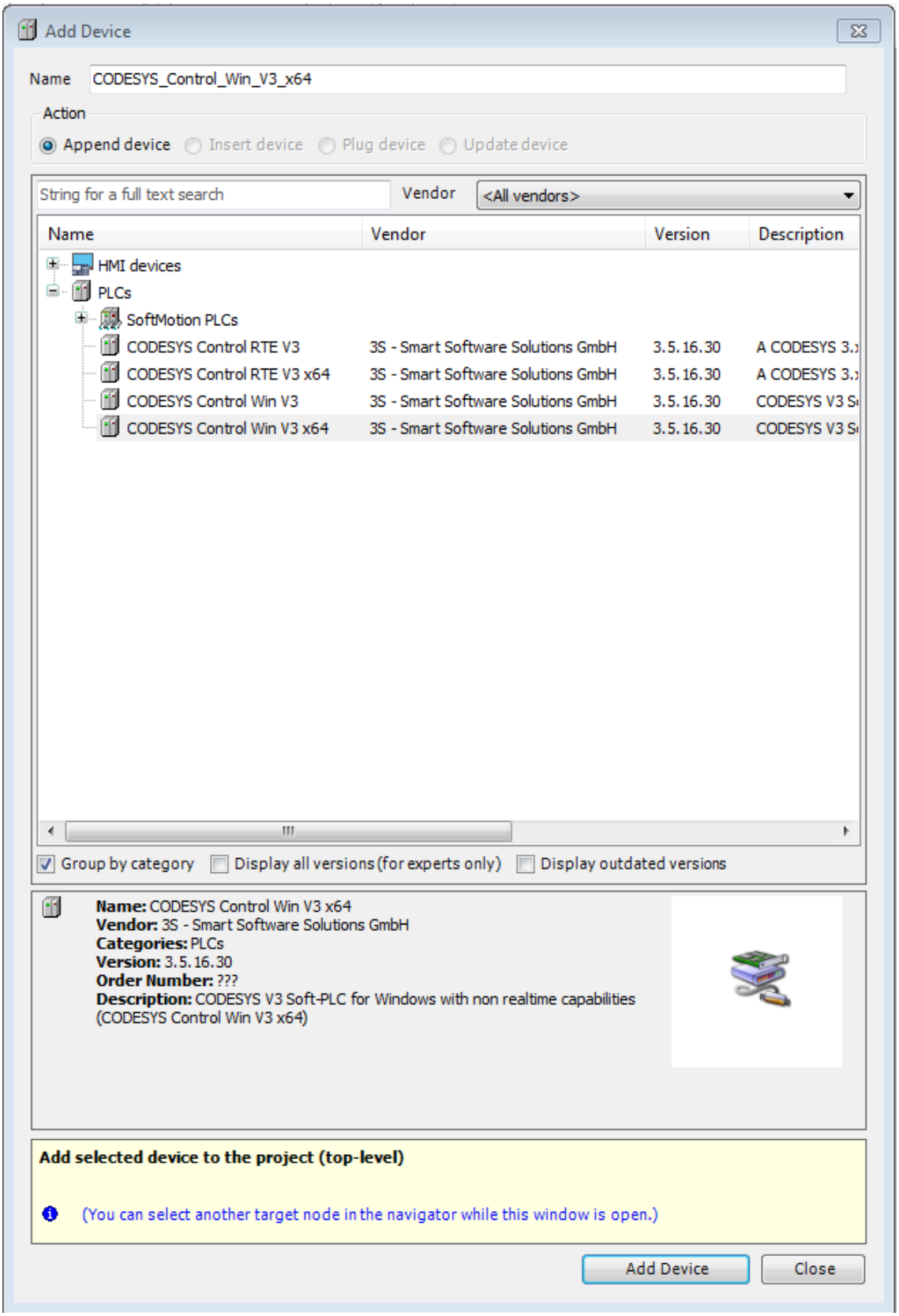
8.1 Tworzenie projektu.

File -> New Project…



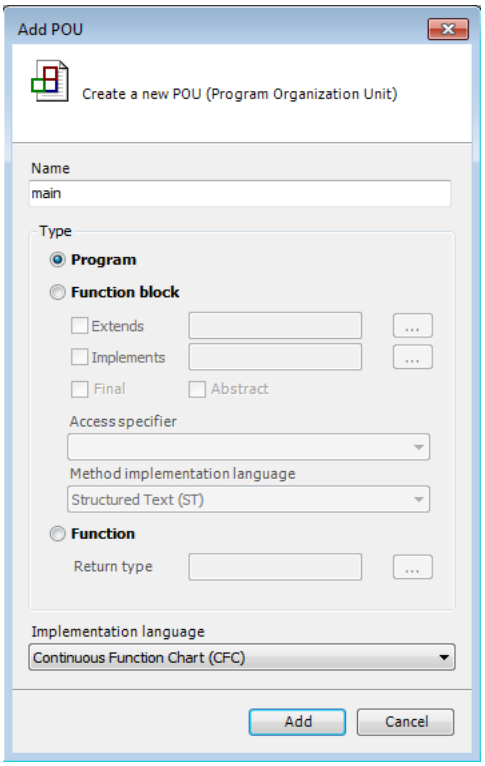
Rys. 8.1 Określenie nazwy i lokalizacji plików projektowych.

Prawy przycisk myszy na nazwę projektu w drzewie projektu -> Add Device…



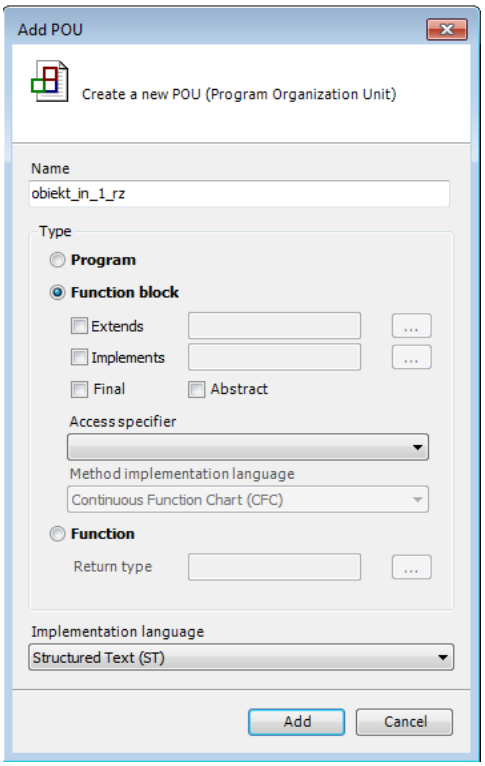
Rys. 8.2 Wybór urządzenia.

Prawy przycisk myszy na Application w drzewie projektu -> Add Object -> POU



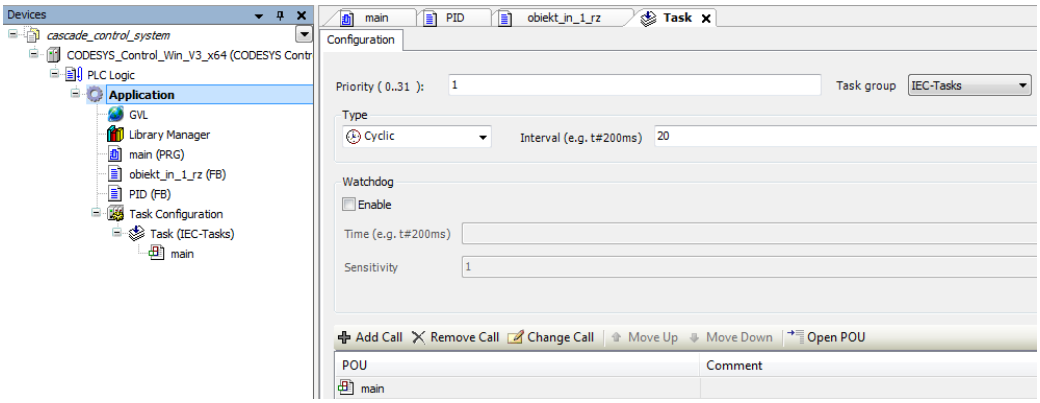
Rys. 8.3 Tworzenie głównej funkcji programowej.

Prawy przycisk myszy na Application w drzewie projektu -> Add Object -> POU



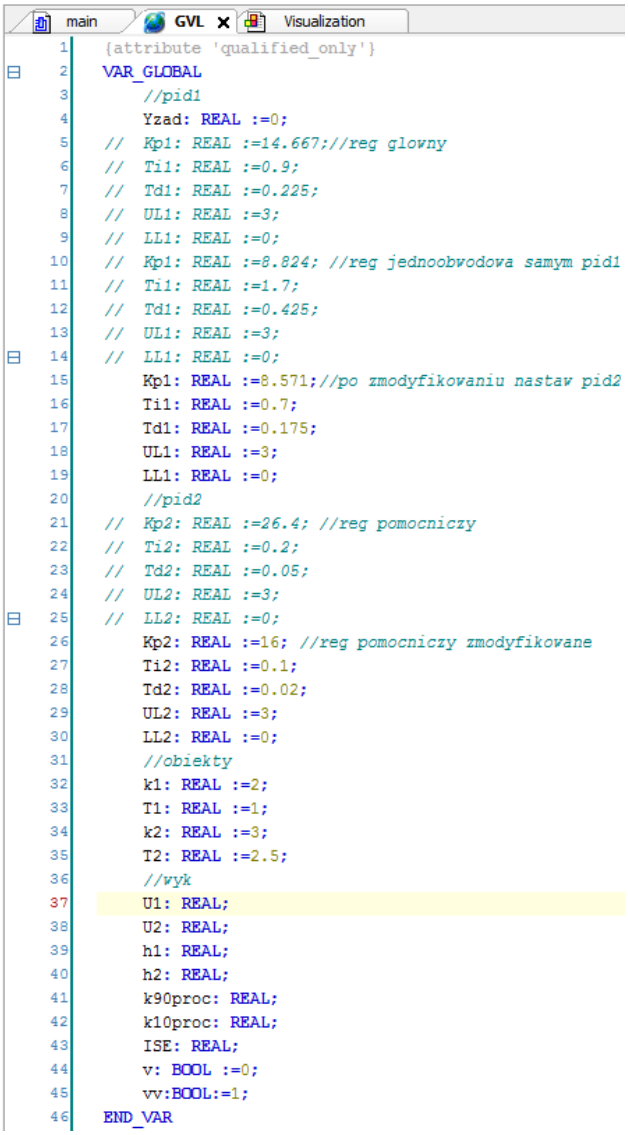
Rys. 8.4 Tworzenie bloku funkcyjnego.

Prawy przycisk myszy na Application w drzewie projektu -> Add Object -> Task Configuration… -> Add



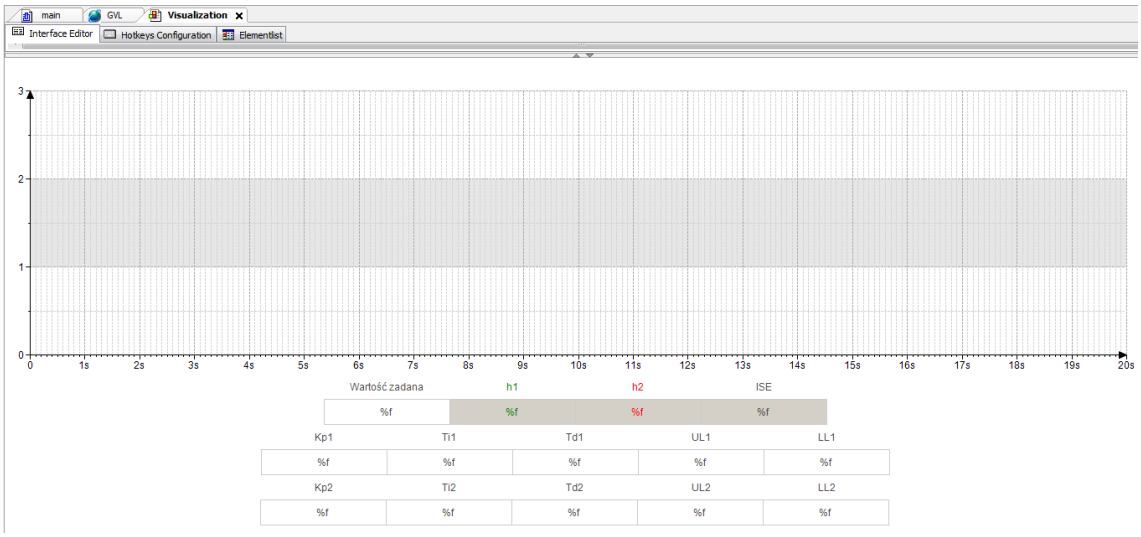
Rys. 8.5 Zarządzanie zadaniami.

Prawy przycisk myszy na Application w drzewie projektu -> Add Object -> Global Variable List… -> Add

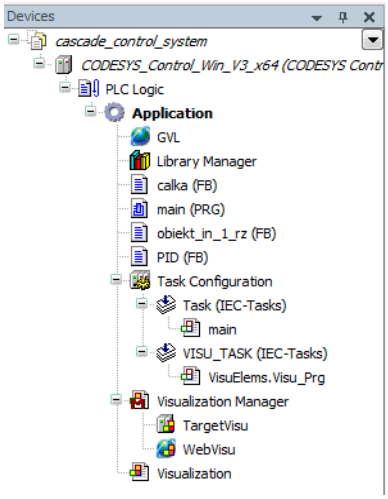


Rys. 8.6 Końcowy wygląd listy zmiennych globalnych.

Prawy przycisk myszy na Application w drzewie projektu -> Add Object -> Visualization… -> Add

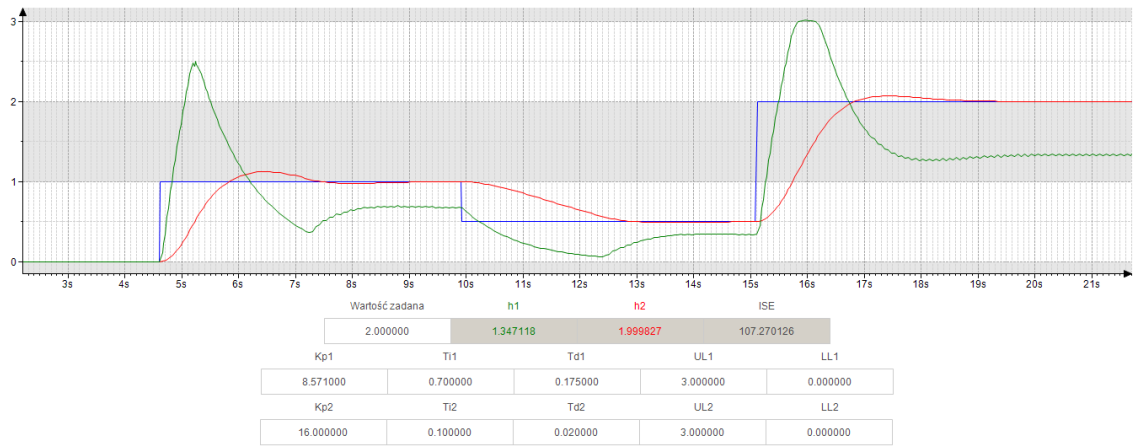


Rys. 8.7 Końcowy wygląd wizualizacji.



Rys. 8.8 Finalne drzewo projektu.

Online -> Simulation -> Login (alt+F8) -> Start (F5)



Rys. 8.9 Przykładowy przebieg symulacji.

9 Podsumowanie.

Celem pracy było przeprowadzenie symulacji regulacji kaskadowej wraz z doborem parametrów regulatorów. Osiągnięto go poprzez zaimplementowanie modelu obiektu i regulatora PID w programie CODESYS, zbudowanie systemu regulacji, przeprowadzenie symulacji, dobranie nastaw regulatorów, określenie jakości regulacji i porównanie jej w trzech przypadkach. Stworzono możliwość wizualizacji pozwalającej obserwować sygnały i zmieniać parametry układu podczas jego pracy.

Tabela 9.1 Porównanie jakości regulacji.

Rodzaj regulacji	**PID**	$\mathbf{k_p}$	$\mathbf{T_i}$	$\mathbf{T_d}$	**ISE**
jednoobwodowa	główny	8,824	1,7	0,425	29,262
kaskadowa[1]	pomocniczy	26,4	0,2	0,05	24
	główny	14,667	0,9	0,225	
kaskadowa[2]	pomocniczy	16	0,1	0,02	23,846
	główny	8,571	0,7	0,175	

1 – nastawy metodą Zieglera-Nicholsa, 2 – nastawy zmodyfikowane

Na podstawie przeprowadzonych symulacji i obserwacji, oraz otrzymanych wyników i wykresów można stwierdzić, że w analizowanym przypadku zastosowanie kaskadowej struktury sterowania poprawiło jakość regulacji, względem struktury jednoobwodowej. Z porównania regulacji kaskadowej[1] z regulacją kaskadową[2] wypływa wniosek, że regulator pomocniczy można zastąpić regulatorem typu P, bo uchyb ustalony występujący na jego wejściu nie ma istotnego znaczenia dla celu regulacji spełnianego przez regulator główny, a więc i dla jakości regulacji.

Istnieje możliwość rozwoju tego projektu w przyszłości. Ciekawym pomysłem wydaje się dodanie możliwości wprowadzania zakłóceń do obu części obiektu i analiza skuteczności regulacji kaskadowej pod tym kątem. Teoria mówi, że zakłócenia pierwszej części obiektu byłyby kompensowane szybciej niż w przypadku zastosowania struktury jednoobwodowej.

10 Bibliografia.

[1] Brzózka J.: *Regulatory i układy automatyki.* MIKOM, Warszawa 2004.

[2] Holejko D., Kościelny W.J.: *Automatyka procesów ciągłych.* Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2012.

[3] Rosołowski E.: *Podstawy regulacji automatycznej.* Notatki do wykładu, Wrocław 2019.

[4] Kaczorek T.: *Teoria układów regulacji automatycznej.* WNT, Warszawa 1974.

[5] Skup Z.: *Podstawy automatyki i sterowania.* Politechnika Warszawska Warszawa 2012.

[6] Horla D.: *Podstawy automatyki ćwiczenia laboratoryjne.* Wydawnictwo Politechniki Poznańskiej, Poznań 2003

[7] ÅSTRÖM K.J., HÄGGLUND T.: *Advanced PID Control. ISA—The Instrumentation, Systems, and Automation Society*, Research Triangle Park, NC, 2005.

[8] Chudy R.: *Regulator PID w środowisku Codesys.* 31.03.2020 online: https://iautomatyka.pl/regulator-pid-w-srodowisku-codesys/ [dostęp 14.01.2021].

[9] Robert: *Podstawy automatyki model zbiornika.* 24.04.2020 online: https://youtu.be/FMhDtYZFKoU [dostęp 14.01.2021].

[10] Tomera M., Talaśka M.: PORÓWNANIE JAKOŚCI PRACY REGULATORÓW STANU I PID W UKŁADZIE KASKADOWYM DWÓCH ZBIORNIKÓW. „Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej" Nr 32/2012, str.139-146.

[11] Tomera M., Kęska J., Kasprowicz A.: *STEROWANIE POZIOMEM WODY W KASKADZIE DWÓCH ZBIORNIKÓW PRZY UŻYCIU MIKROKONTROLERA SYGNAŁOWEGO* TMS320F28335. „Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej" Nr 30/2011, str.123-132.

[12] Gruk W., Habecki Sz., Piotrowski R.: *Implementacja niekonwencjonalnych regulatorów PID w sterowniku programowalnym.* „Pomiary Automatyka Robotyka", 21, Nr 1/2017, 31–39.

[13] Zhang Q., Ding Z., Zhang M.: *Adaptive self-regulation pid control of course-keeping for ships.* „Polish Maritime Research", (105) 2020 Vol. 27; pp. 39-45.

[14] Ryes-Ortiz O.J., Useche-Castelblanco J.S., Vargas-Fonseca G.L.: *Implementation of Fuzzy PID Controller in Cascade with Anti-Windup to Real-Scale Test Equipment for Pavements.* „Engineering Transactions", 68, 1, pp. 3–19, 2020.

[15] Skrobek D., Cekus D., Zając T.: *CONTROL OF THE MOBILE ROBOT USING CONTROLLERS OF TYPES P, PI, PID.* "Journal of Applied Mathematics and Computational Mechanics", 2018, 17(1), 69-78.

[16] Skoczowski S., Domek S., Pietrusewicz K., Broel-Plater B.: *ROBUST MODEL FOLLOWING PlD CONTROL AND IT SIMPLE MENTATION ON PLC.* „IFAC Proceedings Volumes", Volume 36, Issue 11, June 2003, Pages 293-298.

[17] Janiszowski K.: *Dynamic Modification of Reference Value for Reduction of Negative Effects of Actuator Constraints in Single-loop PID Control Systems.* "POMIARY AUTOMATYKA KONTROLA", vol.55, nr 3/2009, str.170-173.

[18] https://www.codesys.com/ [dostęp 14.01.2021].

[19] Lagierski T., Kasprzyk J., Hajda J., Wyrwał J.: *Programowanie sterowników PLC.* Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 1998.