LABORATORIUM: Sterowanie obiektem cieplnym z wykorzystaniem sterowników oraz biblioteki Controller Toolbox firmy Beckhoff

Wprowadzenie

Celem ćwiczenia jest realizacja i przetestowanie układu regulacji stałowartościowej z użyciem regulatora PI oraz obiektu cieplnego, który zidentyfikowano w poprzednim ćwiczeniu dla zadanego punktu pracy. Ćwiczenie składa się z następujących etapów:

1. Przygotowanie teoretyczne do ćwiczenia:

- pobranie, zainstalowanie i zapoznanie się z pakietami **TwinCAT** oraz **Beckhoff Information System** z lokalizacji www.beckhoff.pl (wprowadzając dane na stronie www należy podać informację, że osoba pobierająca jest studentem PRz),
- zapoznanie się z biblioteką **Controller Toolbox** (krótki opis **w instrukcji w punkcie 2**, wyczerpujący w **Beckhoff Information System**),
- przypomnienie metody doboru nastaw regulatora PI dla obiektu o transmitancji „inercja + opóźnienie" [L. Trybus, *Teoria sterowania. Materiały pomocnicze*, Rzeszów 2005, s. 164–169],
- zapoznanie się z zasadą działania i celem użycia układu ***anti-windup*** w strukturach regulacyjnych,
- przypomnienie podstaw programowania w języku ST,
- przeanalizowanie programu zamieszczonego w **punkcie 4 instrukcji**.

2. Zrealizowanie przed laboratorium następujących zadań (są one warunkiem dopuszczenia do laboratorium → punkt 1 i 2 instrukcji):

- Obliczenie doboru nastaw dla wybranego przeregulowania z tabeli 1 zgodnie z wyprowadzonymi wzorami,
- Przeprowadzenie symulacji z użyciem pakietu Simulink dla układu regulacji, który będzie badany na laboratoriach **(punkt 2 instrukcji).**

3. Realizacja praktyczna ćwiczenia (zgodnie z punktem 5 instrukcji):

- Badanie odpowiedzi skokowej dla układu sterowania obiektem cieplnym z wykorzystaniem regulatora PI dla zadanego punktu pracy oraz wcześniej obliczonych nastaw regulatora PI,
- jw. ale dla okresowo zmiennej wartości zadanej.

4. Wykonania sprawozdania, w którym należy zamieścić:

- wzór transmitancji obiektu cieplnego (zidentyfikowanej w poprzednim ćwiczeniu),
- wartości nastaw oraz obliczenia prowadzące do ich uzyskania,
- przebiegi symulacyjne uzyskane w pakiecie Matlab,
- obrazy ekranu z **TwinCAT Scope View**, ukazujące wymagane przebiegi:
  - zbiór odpowiedzi skokowych,
  - przybliżenie odpowiedzi skokowej, pozwalające wyznaczyć przeregulowanie,
  - przebieg śledzenia zmiennej wartości zadanej,
  - przybliżenie przebiegu śledzenia, pozwalające wyznaczyć błąd ustalony,
- odpowiedzi na pytania, które zawarte są w treści instrukcji,
- wnioski i spostrzeżenia.

Literatura

[1] L. Trybus, *Teoria sterowania. Materiały pomocnicze*, Rzeszów 2005
[2] J. Kasprzyk, *Programowanie sterowników przemysłowych*, ISBN 83-204-3109-3, WNT 2005
[3] T.Legierski, J.Kasprzyk, J.Wyrwał, J.Hajda, *Programowanie sterowników PLC*, Pracownia Komputerowa Jacka Skalmierskiego
[4] materiały pomocnicze na stronie http://www.automatyka.kia.prz.edu.pl/
[5] *Beckhoff Information System oraz dokumentacje ze strony* – do pobrania ze strony www.beckhoff.pl

1. Dobór nastaw regulatora PI dla obiektu inercyjnego z opóźnieniem

Obiekt modelowany jest transmitancją postaci:

$$G(s)=\frac{k_o}{Ts+1}e^{-s\tau},$$

której współczynniki k_o, T, τ zostały zidentyfikowane w poprzednim ćwiczeniu.

Transmitancję regulatora można sprowadzić do postaci:

$$G_{PI}(s)=k_p\left(1+\frac{1}{T_i s}\right)=\frac{k_p}{T_i}\cdot\frac{T_i s+1}{s}.$$

Stałą czasową całkowania dobrać według równości $\boldsymbol{T_i = T}$, dla zredukowania bieguna transmitancji obiektu. Przy takim doborze T_i, transmitancja układu otwartego przyjmuje postać:

$$G_{otw}(s)=G_{PI}(s)\cdot G(s)=\frac{k_p}{T_i}\cdot\frac{T_i s+1}{s}\frac{k_o}{Ts+1}e^{-s\tau}=\frac{k_p k_o}{T_i}\cdot\frac{e^{-s\tau}}{s}.$$

Po zastosowaniu aproksymacji Pade' I rzędu dla czynnika opóźniającego [1, s. 163], otrzymuje się:

$$G_{otw}(s)=\frac{k_p k_o}{T_i}\cdot\frac{e^{-s\tau}}{s}=\frac{k_p k_o}{T_i}\frac{\tau}{2}\cdot\frac{-s'+1}{s'(s'+1)}=-k\cdot\frac{s'-1}{s'(s'+1)},$$

gdzie:

$$s'=\frac{\tau}{2}s, \qquad k=\frac{k_p k_o}{T_i}\frac{\tau}{2}.$$

Zatem poszukiwane wzmocnienie regulatora, dane zależnością:

$$k_p=2k\frac{T_i}{k_o\tau}$$

zależy od parametrów obiektu (T_i, k_o, τ) oraz od współczynnika k. Stosując metodę linii pierwiastkowych [1, s. 164-167] można powiązać wartość współczynnika k z projektową wartością przeregulowania. Z metody linii pierwiastkowych wynika także, że spodziewany czas regulacji będzie proporcjonalny do czasu opóźnienia obiektu:

$$t_r=b\tau,$$

gdzie współczynnik b zależy od projektowej wartości przeregulowania.

Tabela 1 zestawia wybrane wartości przeregulowania z odpowiadającymi im współczynnikami k i b.

Tab. 1. Zestawienie wartości $p\%$, k i b

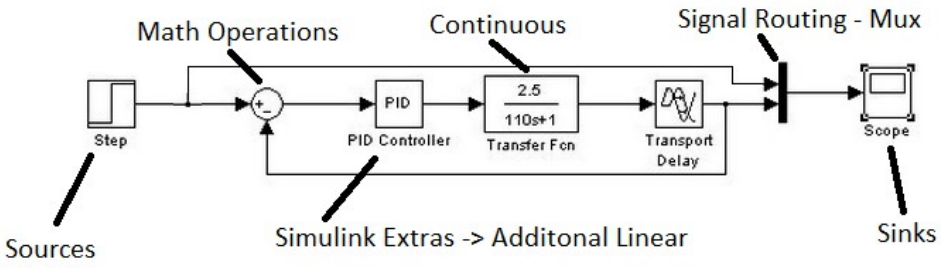
$\boldsymbol{p\%}$	0 *)	4,6	9,5	16,3	25,4	37,2
$\boldsymbol{k}$	0,17	0,27	0,32	0,38	0,46	0,55
$\boldsymbol{b}$	4,8	5,5	5,9	6,5	7,4	9,0

*) przebiegi aperiodyczne krytyczne

Na podstawie wartości zidentyfikowanych parametrów obiektu, przeregulowania podanego przez prowadzącego oraz zamieszczonych powyżej zależności, należy wyznaczyć nastawy regulatora PI i spodziewany czas regulacji. **W sprawozdaniu należy zamieścić obliczenia i uzyskane wartości k_p, T_i oraz t_r.**

2. Symulacja w pakiecie Matlab dla obliczonych nastaw

Przed przystąpieniem do laboratoriów należy przeprowadzić w pakiecie Simulink symulację układu regulacji dla obliczonych nastaw oraz parametrów zidentyfikowanego obiektu. Schemat powinien wyglądać jak na rysunku poniżej:



3. Wprowadzenie do Biblioteki Controller Toolbox

3.1. Biblioteka Controller Toolbox

Biblioteka **Controller Toolbox** stanowi bogaty zestaw bloków funkcjonalnych, zgodnych z normą IEC 61331, przeznaczonych do realizacji układów regulacji. Zbiór bloków podzielony jest na 8 sekcji:

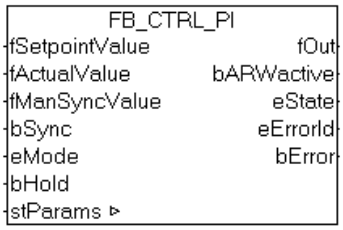
- **Auxiliary** – bloki pomocnicze, przeznaczone do wsparcia współpracy układu regulacji z systemem sterownika,
- **Base** – podstawowe, jednotorowe bloki regulacyjne, np. proporcjonalny (FB_CTRL_P), całkujący (FB_CTRL_I), histereza (FB_CTRL_HYSTERESIS) itp.,
- **Controller** – złożone, kompletne struktury regulacyjne, np. regulator dwupołożeniowy (FB_CTRL_2POINT), regulator PID (FB_CTRL_PID), regulator kaskadowy PI-PID (FB_CTRL_PI_PID) itp.,
- **Filter / Controlled System Simulation** – bloki filtrowania i uśredniania sygnałów, generacji szumów oraz symulacji typowych członów dynamicznych obiektów regulacji, np. filtr wartości bieżącej (FB_CTRL_ACTUAL_VALUE_FILTER), korektor (FB_CTRL_LEAD_LAG), człon inercyjny II rzędu (FB_CTRL_PT2), człon opóźniający (FB_CTRL_PTt) itp.,
- **Interpolation** – blok interpolacji liniowej (FB_CTRL_LIN_INTERPOLATION), blok normalizacji (FB_CTRL_NORMALIZE),
- **Monitoring / Alarming** – bloki kontroli i logowania wartości sygnałów, np. kontrola zakresu wartości (FB_CTRL_CHECK_IF_IN_BAND), rejestracja wartości zmiennych procesowych do pliku (FB_CTRL_LOG_DATA) itp.,
- **Output To Controlling** – bloki dostosowujące wyjście sygnału do rodzaju podłączonego obiektu, np. skalowanie wartości (FB_CTRL_SCALE), dodanie strefy

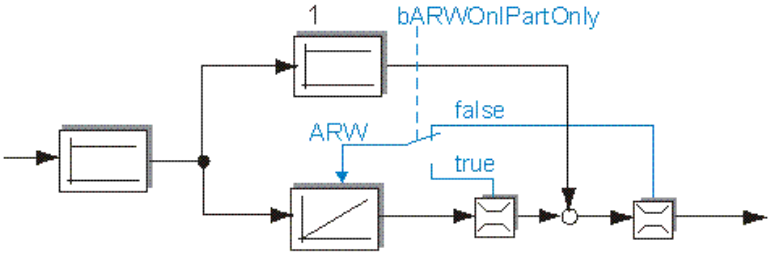
martwej (FB_CTRL_DEADBAND), wyjście typu PWM (FB_CTRL_PWM_OUT) itp.,

- **Setpoint generation** – bloki generacji wartości zadanej, np. generator sygnału trójkąt-sinus-piła (FB_CTRL_SIGNAL_GENERATOR), generator przebiegu tablicowanego (FB_CTRL_SETPOINT_GENERATOR) itp.

3.2. Blok regulatora PI

Blok regulatora PI o nazwie **FB_CTRL_PI** znajduje się w sekcji **Controller** biblioteki **Controller Toolbox**. Poniżej ukazano symbol bloku regulatora oraz jego strukturę wewnętrzną, zawierającą składnik ***anti-windup***.





Działanie układu *anti-windup*:

W strukturach regulacyjnych z całkowaniem dochodzi do efektu nasycania elementu całkującego, które występuje przy ograniczeniu sygnału sterowania. Zjawisko to, zwane *windup*, pogarsza jakość pracy układu regulacji. *Anti-windup* jest mechanizmem przeciwdziałającym temu efektowi. Jego funkcjonowanie polega na ograniczeniu lub zatrzymaniu działania integratora w sytuacji ograniczenia sygnału sterowania.

W przypadku bloku PI z biblioteki **Controller Tollbox**, mechanizm *anti-windup* wykorzystuje rozwiązanie całkowitego zatrzymania całkowania. Zależnie od wartości bitu konfiguracyjnego **`bARWOnIPartOnly`**, całkowanie zostaje wstrzymane, gdy ograniczeniu ulega główne wyjście bloku (`false`) lub tylko wyjście toru całkującego (`true`).

Spotykane są też inne realizacje struktury *anti-windup*, np. odjęcie od wejścia integratora sygnału proporcjonalnego do różnicy pomiędzy wejściem i wyjściem bloku ograniczającego, czy też odjęcie od wejścia integratora sygnału ukształtowanego z jego wyjścia lub wyjścia całego regulatora przez strefę nieczułości.

Zbiór wartości wejściowych bloku zdefiniowany jest następująco:

```
VAR_INPUT
        fSetpointValue  : FLOAT;
        fActualValue    : FLOAT;
        fManSyncValue   : FLOAT;
        bSync           : BOOL;
        eMode           : E_CTRL_MODE;
        bHold           : BOOL;
END_VAR
```

gdzie:

- **`fSetpointValue`** – wartość zadana wielkości regulowanej,
- **`fActualValue`** – wartość bieżąca wielkości regulowanej, odczytywana z wyjścia obiektu,

- **fManSyncValue** – wartość inicjująca wyjście bloku PI,
- **bSync** – zbocze narastające na tym (logicznym) wejściu powoduje załadowanie do bloku PI wartości inicjującej `fManSyncValue`,
- **eMode** – zmienna typu wyliczeniowego `E_CTRL_MODE`, określająca tryb pracy regulatora PI, możliwe wartości tej zmiennej wyszczególnione są poniżej, w definicji typu `E_CTRL_MODE`,
- **bHold** – wartość stanu wewnętrznego i wyjścia bloku zastają „zatrzaśnięte" na czas utrzymywania stanu wysokiego (`TRUE`) na tym wejściu.

Definicja typu wyliczeniowego **E_CTRL_MODE**:

```
TYPE E_CTRL_MODE:
(
   eCTRL_MODE_IDLE          :=  0,    (*mode idle *)
   eCTRL_MODE_PASSIVE       :=  1,    (*mode passive *)
   eCTRL_MODE_ACTIVE        :=  2,    (*mode active *)
   eCTRL_MODE_RESET         :=  3,    (*mode reset *)
   eCTRL_MODE_MANUAL        :=  4,    (*mode manual *)
   eCTRL_MODE_TUNE          :=  5,    (*mode tuning *)
   eCTRL_MODE_SELFTEST      :=  6,    (*mode selftest *)
   eCTRL_MODE_SYNC_MOVEMENT :=  7     (*mode synchronize *)
);
END_TYPE
```

Zmienne wyjściowe zdefiniowane są następująco:

```
VAR_OUTPUT
       fOut          : FLOAT;
       bARWactive    : BOOL;
       eState        : E_CTRL_STATE;
       eErrorId      : E_CTRL_ERRORCODES;
       bError        : BOOL;
END_VAR
```

gdzie:

- **fOut** – wyjście regulatora (sterowanie),
- **bARWactive** – stan wysoki tego wyjścia sygnalizuje aktywność układu *anti-windup*,
- **eState** – wartość typu wyliczeniowego `E_CTRL_STATE`, określająca ogólny stan pracy bloku,
- **eErrorId** – zmienna typu wyliczeniowego `E_CTRL_ERRORCODES`, określająca rodzaj błędu w przypadku błędnej pracy bloku,
- **Terror** – wartość stanu wysokiego (`TRUE`) tej zmiennej sygnalizuje wystąpienie błędu w pracy bloku.

Parametry konfiguracyjne (wejściowo-wyjściowe) bloku PI określone są za pomocą jednej zmiennej **stParams** typu strukturalnego:

```
VAR_IN_OUT
       stParams      : ST_CTRL_PI_PARAMS;
END_VAR
```

Typ strukturalny **ST_CTRL_PI_PARAMS** zdefiniowany jest następująco:

```
TYPE ST_CTRL_PI_PARAMS :
STRUCT
   tCtrlCycleTime   : TIME     := T#0ms; (* controller cycle time [TIME] *)
   tTaskCycleTime   : TIME     := T#0ms; (* task cycle time [TIME] *)
   tTn              : TIME     := T#0ms; (* integral action time Tn *)
   fKp              : FLOAT    := 0;     (* proportional gain *)
   fOutMaxLimit     : FLOAT    := 1E38;  (* maximum output limit *)
   fOutMinLimit     : FLOAT    := -1E38; (* minimum output limit *)
   bARWOnIPartOnly  : BOOL     := FALSE;
END_STRUCT
END_TYPE
```

gdzie:

- **tCtrlCycleTime** – czas cyklu (aktualizacji stanu i wyjścia) algorytmu regulacyjnego,
- **tTaskCycleTime** – czas cyklu zadaniowego (tj. czas pomiędzy kolejnymi wywołaniami bloku),
- **tTn** – stała czasowa całkowania,
- **fKp** – wzmocnienie bloku PI,
- **fOutMaxLimit** – górna wartość ograniczenia *anti-windup*,
- **fOutMinLimit** – dolna wartość ograniczenia *anti-windup*,
- **bARWOnIPartOnly** – tryb pracy układu *anti-windup*; jeżeli zmienna ta ustawiona jest na `FALSE` (domyślnie), wtedy ograniczenie *anti-windup* działa w oparciu o wyjście całego bloku (patrz schemat bloku), w przeciwnym wypadku ograniczenie *anti-windup* działa w oparciu o wyjście samego toru całkującego.

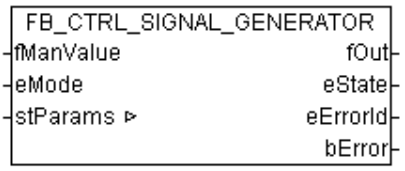
Transmitancja rozważanego bloku PI posiada postać:

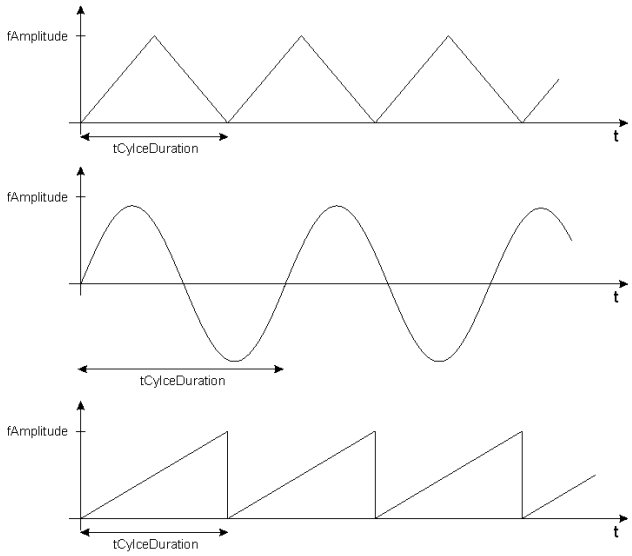
$$G(s)=K_p\left(1+\frac{1}{T_n s}\right),$$

gdzie współczynniki K_p i T_n są równe wartościom konfiguracyjnym, odpowiednio **fKp** i **tTn**.

3.3. Blok generatora sygnałów

Blok generatora sygnałów **FB_CTRL_SIGNAL_GENERATOR** umieszczony jest w sekcji **Setpoint generation** biblioteki **Controller Toolbox**. Jest on źródłem sygnału sinusoidalnego, trójkątnego lub piłokształtnego o konfigurowanych parametrach. Symbol bloku regulatora i kształty wytwarzanych przebiegów ukazano poniżej.





Zbiór wartości wejściowych bloku zdefiniowany jest następująco:

```
VAR_INPUT
       fManValue      : FLOAT;
       eMode          : E_CTRL_MODE;
END_VAR
```

gdzie:

- **`fManValue`** – wartość trybu manualnego, przekazywana bezpośrednio na wyjście, gdy tryb pracy (`eMode`) ustawiony jest na `eCTRL_MODE_MANUAL`,
- **`eMode`** – zmienna typu wyliczeniowego `E_CTRL_MODE` (definicja typu przytoczona została przy opisie bloku PI), określająca tryb pracy generatora.

Zmienne wyjściowe zdefiniowane są następująco:

```
VAR_OUTPUT
       fOut           : FLOAT;
       eState         : E_CTRL_STATE;
       eErrorId       : E_CTRL_ERRORCODES;
       bError         : BOOL;
END_VAR
```

gdzie:

- **`fOut`** – wartość wyjściowa sygnału generatora,
- **`eState, eErrorId, Terror`** – znaczenie identyczne jak w przypadku bloku `FB_CTRL_PI`.

Parametry konfiguracyjne (wejściowo-wyjściowe) generatora określone są za pomocą jednej zmiennej **`stParams`** typu strukturalnego:

```
VAR_IN_OUT
       stParams       : ST_CTRL_SIGNAL_GENERATOR_PARAMS;
END_VAR
```

Typ strukturalny **ST_CTRL_SIGNAL_GENERATOR_PARAMS** zdefiniowany jest następująco:

```
TYPE ST_CTRL_I_PARAMS :
STRUCT
   tCtrlCycleTime   : TIME  :=  T#0ms; (* controller cycle time *)
   tTaskCycleTime   : TIME  :=  T#0ms; (* task cycle time *)
   eSignalType      : E_CTRL_SIGNAL_TYPE;
   tCylceDuration   : TIME;
   fAmplitude       : FLOAT;
   fOffset          : FLOAT := 0.0;
   tStart           : TIME  := T#0s;
END_STRUCT
END_TYPE
```

gdzie:

- **tCtrlCycleTime** – czas cyklu (aktualizacji stanu i wyjścia) bloku,
- **tTaskCycleTime** – czas cyklu zadaniowego,
- **eSignalType** – rodzaj sygnału wyjściowego, określony zmienną typu wyliczeniowego (definicja typu `E_CTRL_SIGNAL_TYPE` podana jest niżej),
- **tCylceDuration** – okres przebiegu generowanego sygnału,
- **fAmplitude** – amplituda wytwarzanego sygnału,
- **Offset** – stałe przesunięcie, dodawane do przebiegu,
- **Start** – faza początkowa sygnału, wyrażona przez moment czasowy odniesiony do okresu przebiegu.

Definicja typu wyliczeniowego **E_CTRL_SIGNAL_TYPE**:

```
TYPE E_CTRL_SIGNAL_TYPE :
(
    eCTRL_TRIANGLE  := 0,
    eCTRL_SINUS     := 1,
    eCTRL_SAWTOOTH  := 2
);
END_TYPE
```

4. Algorytm regulacji w języku ST

Proponowana postać algorytmu regulacji zawarta jest w pliku **reg_PI.pro** (dostępny na stronie WWW wraz z instrukcją). Program wykorzystuje dwa opisane wcześniej bloki funkcjonalne z biblioteki **Controller Toolbox**: blok regulatora PI **FB_CTRL_PI** oraz blok generatora sygnałowego **FB_CTRL_SIGNAL_GENERATOR**. Kod programu, z pominięciem części deklaracyjnej, przytoczony jest poniżej.

```
IF bInit
THEN
      (*  konfiguracja bloku regulatora FB_CTRL_PI  *)
      stCTRL_PI_PARAMS.tCtrlCycleTime  := T#10ms;
      stCTRL_PI_PARAMS.tTaskCycleTime  := T#10ms;
      stCTRL_PI_PARAMS.tTn             := T#130s; (* TUTAJ WSTAWIC wyliczone Ti *)
      stCTRL_PI_PARAMS.fKp             := 11;     (* TUTAJ WSTAWIC wyliczone Kp *)
      stCTRL_PI_PARAMS.fOutMaxLimit    := 100;    (* gorne ograniczenie wyjscia *)
      stCTRL_PI_PARAMS.fOutMinLimit    := 0;      (* dolne ograniczenie wyjscia *)

      (* konfiguracja generatora sygnalowego FB_CTRL_SIGNAL_GENERATOR *)
      stCTRL_GEN_PARAMS.tCtrlCycleTime := T#10ms;
      stCTRL_GEN_PARAMS.tTaskCycleTime := T#10ms;
      stCTRL_GEN_PARAMS.eSignalType    := eCTRL_SINUS;
      stCTRL_GEN_PARAMS.fAmplitude     := 15;       (* amplituda *)
      stCTRL_GEN_PARAMS.fOffset        := fSetpointConst;  (* offset - skladowa
                                                              stala *)
      stCTRL_GEN_PARAMS.tCylceDuration := T#300s;   (* okres przebiegu *)
      stCTRL_GEN_PARAMS.tStart         := T#0s;     (* faza początkowa, odniesiona
                                                      do okresu *)

      bInit  := FALSE;
END_IF

(* wybor trybu sterowania AUTO/MAN *)
IF bMode THEN
      fSetpointValue      := fSetpointGen;
      eGenMode            := eCTRL_MODE_ACTIVE;
ELSE
      fSetpointValue      := fSetpointConst;
      eGenMode            := eCTRL_MODE_MANUAL;
END_IF

(* wywołanie bloku generatora sygnalu wartosci zadanej  *)
fbCTRL_GEN   (
                        fManValue    := fSetpointConst,
                        eMode        := eGenMode,
                        stParams     := stCTRL_GEN_PARAMS,
                        fOut         => fSetpointGen,
                        eState       => eSinState,
                        eErrorId     => eSinErrorId,
                        bError       => bSinError
                  );

hwInFix16.value     := hwIn;
hwInFix16.n         := 0;
fActualValue        := 100.0 / 16#7FFF * FIX16_TO_LREAL(hwInFix16);

(* wywolanie bloku regulatora PI *)
fbCTRL_PI(          fSetpointValue      := fSetpointValue,
                    fActualValue        := fActualValue,
                    fManSyncValue       := fManSyncValue,
                    bSync               := bSync,
                    eMode               := eCTRL_MODE_ACTIVE,
                    bHold               := bHold,
                    stParams            := stCTRL_PI_PARAMS,
```

```
                    fOut                => fOut_PI,
                    bARWactive          => bARWactive,
                    eErrorId            => eErrorId,
                    bError              => bError
        );

hwOutFix16 := LREAL_TO_FIX16(fOut_PI * 16#7FFF / 100.0, 0);
hwOut := hwOutFix16.value;
```

Inicjalizacja struktur konfiguracyjnych dla bloków regulatora i generatora odbywa się jednorazowo, w pierwszym cyklu przetwarzania sterownika. Po jednokrotnej inicjalizacji zerowana jest flaga **bInit**, blokując kolejne wejścia do bloku przypisania wartości początkowych.

W każdym cyklu przetwarzania program wykonuje kolejno następujące czynności:

1. Na podstawie flagi **bMode** przypisuje jako wartość zadaną dla regulatora PI wartość z wyjścia generatora **fSetpointGen** (bMode=TRUE) lub wartość stałą określoną manualnie **fSetpointConst** (bMode=FALSE).
2. Wywołuje blok generatora sygnałowego **fbCTRL_GEN**, który wyznacza kolejną wartość przebiegu funkcyjnego, zapisując ją do zmiennej **fSetpointGen**.
3. Konwertuje wartość całkowitoliczbową z wejścia analogowego **hwIn** z zakresu 0..7FFFh na liczbę zmiennoprzecinkową **fActualValue** z przedziału 0..100, która stanowi zmienną procesową dla regulatora.
4. Wywołuje blok regulatora PI **fbCTRL_PI**, który wylicza nową wartość sterowania, zapisywaną do zmiennej **fOut_PI**.
5. Przelicza zmiennoprzecinkową wartość wyjścia regulatora **fOut_PI** o zakrasie 0..100 do liczby całkowitej **hwOut** z przedziału 0..7FFFh, stanowiącej wartość wyjściową dla obiektu.

W kodzie programu zwrócić uwagę na znaczenie wartości przypisanych poszczególnym polom struktur konfiguracyjnych bloku generatora i regulatora (**stCTRL_PI_PARAMS**, **stCTRL_GEN_PARAMS**) oraz na wartości parametrów wywołania tych bloków. W szczególności opanować czynności konfiguracyjne:

- wprowadzania nastaw **regulatora PI**: wzmocnienia k_p – **fKp** oraz stałej czasowej całkowania T_i – **tTn**,
- konfiguracji składnika **anti-windup**: wybór aktywny/nieaktywny, ustawianie wartości dolnego i górnego ograniczenia, wybór trybu aktywacji ograniczenia (tylko tor całkujący, pełny sygnał wyjściowy),
- konfiguracji **generatora sygnałowego**: wybór rodzaju przebiegu, ustawienie amplitudy, okresu/częstotliwości, składowej stałej, wartości/fazy początkowej.

5. Realizacja ćwiczenia

UWAGA:
Podobnie jak w ćwiczeniu z identyfikacją, używane będą dwa programy z pakietu **TwinCAT System**, połączone *on-line* ze sterownikiem: **TwinCAT PLC Control** oraz **TwinCAT Scope View**. **Przechodząc pomiędzy programami nie należy zamykać żadnego z nich, aż do zakończenia eksperymentu**.

5.1. Otwarcie szablonów projektowych

Otworzyć pliki:

- **`<Pulpit>\SzablonyBeckhoff\CX\Control\reg_PI.pro`**

  który jest szablonem projektowym dla programu **TwinCAT PLC Control.**
- **`<Pulpit>\SzablonyBeckhoff\CX\Control\config.tsm`**

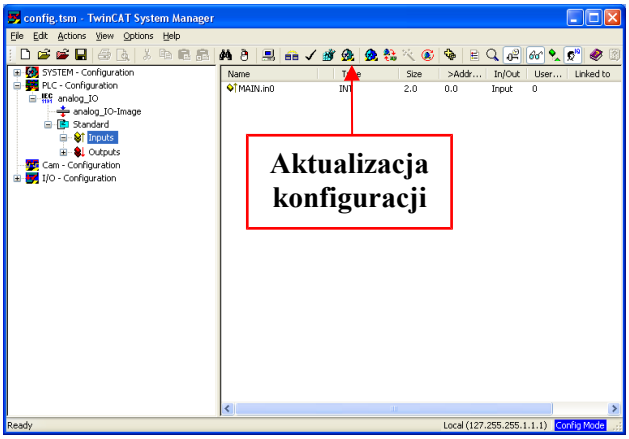
  który jest szablonem projektowym dla programu **TwinCAT PLC System Manager.**
- **`<Pulpit>\SzablonyBeckhoff\CX\Control\scope.scp`**

  który jest szablonem projektowym dla programu **TwinCAT PLC Scope View.**

UWAGA: *Istnieje możliwość samodzielnego utworzenia projektów, zamiast użycia pliku szablonowych. Szczegółowe kroki opisane są w dodatku A i B.*

5.2. Uruchomienie programu PLC

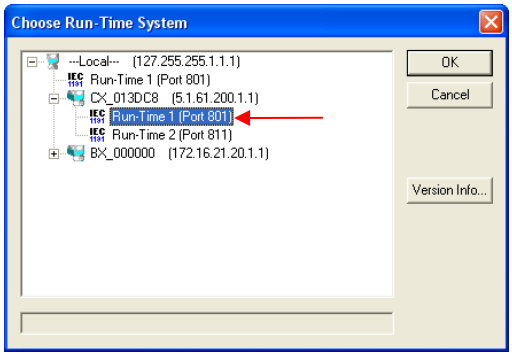
- Przygotowaną konfigurację należy przesłać do systemu PLC (**TwinCAT PLC System Manager**). System powinien zostać następnie przełączony w tryb pracy (RUN).



- Po wykonaniu opisanych wyżej operacji, przejść do **TwinCAT PLC Control** i w kodzie programu **zmodyfikować wartości nastaw regulatora**, zgodnie z rezultatami własnych obliczeń, opartych na wcześniej zidentyfikowanej transmitancji obiektu.

- Projekt należy skompilować (**Project → Build**).
- Po udanej kompilacji uruchomić program sterownika w trybie *on-line*, tj:

  1. Zweryfikować wybór adresu sterownika w oknie **Online → Choose Run-Time System…**



  2. Zalogować się do sterownika: **Online → Login.**

  3. Uruchomić program: **Online → Run** (w czasie wykonywania programu kontrolki LED w modułach KL3448 i KL4418 powinny świecić na zielono).

5.3. Badanie odpowiedzi skokowej

5.3.1 Czynności wstępne:

- Dla domyślnych wartości parametrów konfiguracyjnych, program przystosowany jest do realizacji eksperymentu odpowiedzi skokowej. Wybrane jest źródło wartości stałej (`bMode = FALSE`) o początkowej wartości 50 (`fSetpointConst = 50`). **Wartość zmiennej należy poprawić na wartość punktu pracy dla którego zidentyfikowano obiekt.**
- Uruchomić rejestrację przebiegu w **TwinCAT Scope View** (**Scope → Start Scope** lub przycisk ▶). Rejestracja powinna ukazać stan ustalony z ustabilizowaną wartością sterowania lub proces osiągania stanu ustalonego – zależnie od czasu, który upłynął od momentu uruchomienia programu sterownika z algorytmem regulacji (w pkt. 4).
- Należy odczekać do chwili osiągnięcia stanu ustalonego, jeżeli nie będzie on uzyskany w chwili rozpoczęcia obserwacji.

5.3.2 Realizacja eksperymentu:

- Po osiągnięciu stanu ustalonego, dokonać skokowej zmiany wartości zadanej. Aby zrealizować skok wartości zadanej, w **TwinCAT PLC Control** kliknąć na wartość zmiennej `fSetpointConst` i podać nową wartość. Następnie zapisać zmianę, wykorzystując opcję **Online → Write Values** (lub klawisze **[ctrl + F7]**).
- Odpowiedź układu regulacji powinna mieć charakter oscylacyjny. Stosując kilkakrotnie wymuszenie zmian wartości zadanej, wykonać wymuszenia skokowe w górę i w dół (przynajmniej po jednym w górę i w dół), o różnych amplitudach. **Zmiany nie powinny wychodzić poza przedział 10% zakresu wartości zadanej od punktu pracy, dla którego dokonano identyfikacji obiektu cieplnego na poprzednich zajęciach**, a amplituda skoku powinna wynosić ok. 5-10% zakresu.

- Przykład sekwencji testowych skoków wartości zadanej ukazuje rysunek poniżej. Po wykonaniu rejestracji zatrzymać program sterownika i wylogować się z trybu **Online**.

- Rejestracja pozwala zaobserwować, jak amplituda skoku, jego znak oraz początkowa wartość odniesienia wpływa na postać przebiegu regulacyjnego. Na przykładowym wykresie powyżej są to skoki zaznaczone okręgami. Dla takich skoków spełniony jest warunek pracy liniowej i postać przebiegu powinna być zbliżona do założeń projektowych.

W sprawozdaniu umieścić:

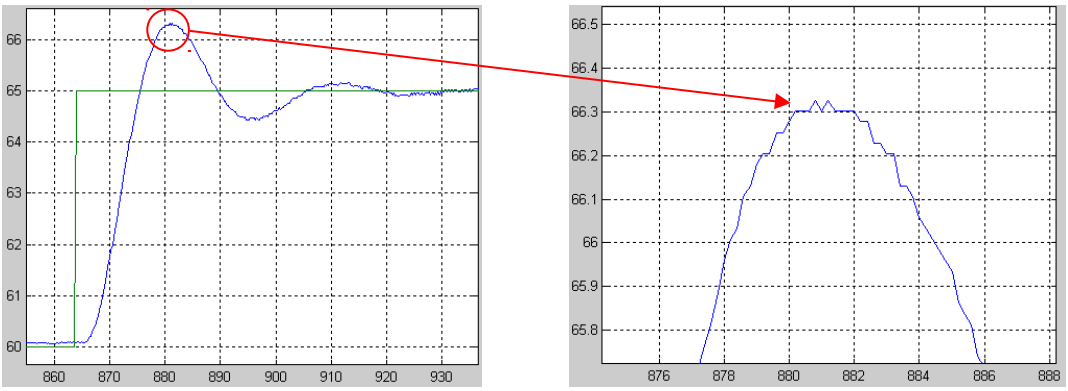
1. **Obraz pełnego okna programu TwinCAT Scope View z zarejestrowanymi skokami. Zaznaczyć przypadki wystąpienia ograniczeń sygnału sterowania.**
2. **Krótki komentarz, stwierdzający czy kształt uzyskanej odpowiedzi jest analogiczny przy skoku w górę i w dół, jeżeli zgodność nie występuje, podać uzasadnienie.**

5.3.3 Dokładne określenie przeregulowania

Jeden z wybranych skoków należy powiększyć tak, aby w dokładny sposób określić przeregulowanie. Powiększenie można wykonać w programie **TwinCAT Scope View** wykorzystując opcję **Zoom**. W tym celu należy kolejno:

- kliknąć w obszarze wykresu,
- kliknąć ikonę **Zoom**  ,
- wskazać kursorem powiększany element wykresu i kliknąć, klikając wielokrotnie uzyskuje się kolejne powiększenia,
- w celu cofnięcia powiększenia, należy kliknąć ikonę **Zoom-out**  .

Przykładowe powiększenie przebiegu odpowiedzi skokowej, zaznaczone okręgiem niebieskim na poprzednim wykresie, ukazane jest poniżej.



Wartość przeregulowania $p = \frac{66.3 - 65}{5} \cdot 100\% = 26\%$.

W sprawozdaniu:

1. **Zamieścić powiększony wykres odpowiedzi skokowej oraz obliczenia i wynik dla przeregulowania.**
2. **Określić czy wartość przeregulowania jest zgodna z założeniami projektowymi i uzasadnić zgodność bądź niezgodność.**
3. **Podać rzeczywisty czas regulacji i porównać go z oszacowaniem dokonanym w punkcie 1.**

5.4. Badanie śledzenia okresowo zmiennej wartości zadanej

5.4.1 Czynności wstępne

- Jako przebieg wartości zadanej wybrany zostanie jeden z sygnałów, możliwych do wytworzenia przez generator **`fbCTRL_GEN`**. Zwrócić się do prowadzącego z zapytaniem o parametry przebiegu. W przypadku braku wskazań prowadzącego, samodzielnie ustalić postać przebiegu, respektując następujące zalecenia:
  1. Wartość zadana nie powinna wychodzić poza przedział 40…80% zakresu, gdyż może to wiązać się z ograniczeniami sygnału sterującego, a przez to utratą liniowości.
  2. Wartość zadana nie powinna wychodzić poza 10% zakresu od punktu pracy dla którego zidentyfikowano obiekt.
  3. Okres przebiegu powinien wynosić co najmniej 300 s.
  4. Wybrać przebieg o kształcie sinusoidalnym lub trójkątnym.
- Po określeniu wartości parametrów konfiguracyjnych, które zapewnią generację pożądanego przebiegu (por. pkt. 3), wprowadzić je do kodu programu **`reg_PI.pro`**. W programie tym należy zmienić inicjalizację wartości flagi `bMode` na `TRUE`, co oznacza wybór generatora sygnałowego jako bieżącego źródła wartości zadanej.
- Zmodyfikować odpowiednio program.

UWAGA:
Jeśli w programie na etapie kompilacji pojawi się problem w związku z nazwą określającą typ przegiegu z typu wyliczeniowego **`E_CTRL_SIGNAL_TYPE`**, to należy użyć odpowiedniej wartości liczbowej przy ustawianiu rodzaju przebiegu sterowania.

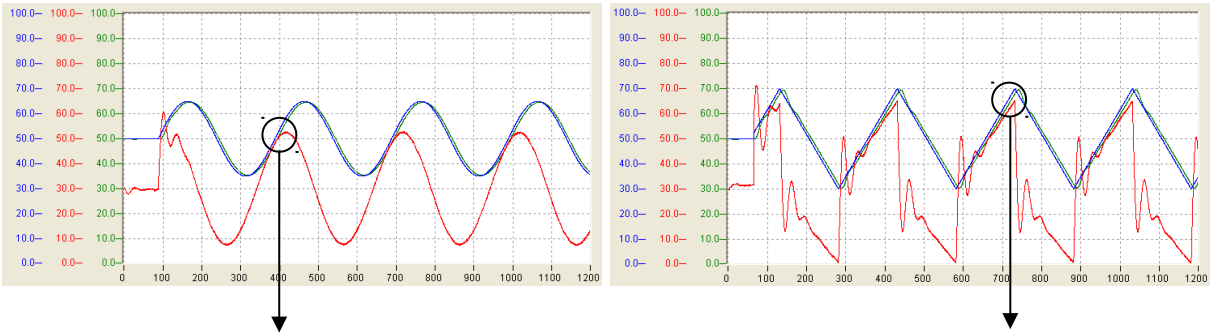
5.4.2 Realizacja eksperymentu

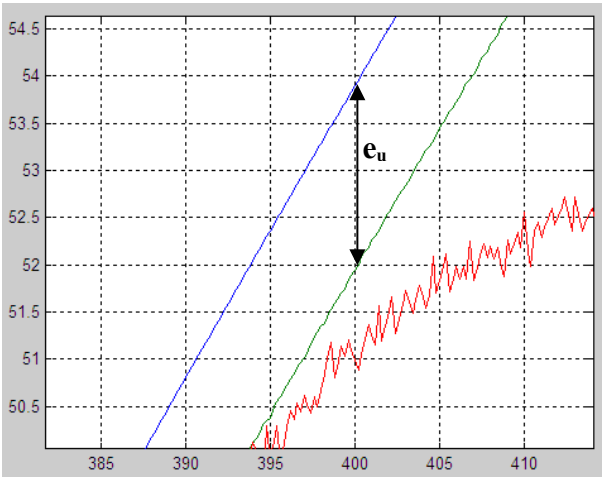
- Zmodyfikowany program skompilować i uruchomić, korzystając z trybu **Online**. Przed właściwym uruchomieniem programu zaleca się wykonanie restartu programowego sterownika. Wywołać kolejno następujące opcje:
  1. **Project → Build**
  2. **Online → Login**
  3. **Online → Reset All**
  4. **Online → Run**
- Po uruchomieniu programu uaktywnić rejestrację przebiegu w **TwinCAT Scope View** (**Scope → Start Scope** lub przycisk ▶).
- Obserwowany przebieg wartości zadanej powinien być zgodny z zaplanowanym. W przypadku niezgodności, przeanalizować jakie parametry konfiguracyjne programu wymagają korekty. W razie potrzeby zatrzymać program i wylogować się z trybu **Online**, zmodyfikować program, skompilować go i ponownie uruchomić.
- Po stwierdzeniu poprawnej postaci przebiegu wartości zadanej i skutecznego jej śledzenia przez układ regulacji, zarejestrować przebieg. Rejestracja powinna objąć przynajmniej dwa okresy zmian wartości zadanej.
- Należy zapisać obraz ekranu z otrzymanym przebiegiem, który powinien być podobny do jednego z przedstawionych w kolejnym.

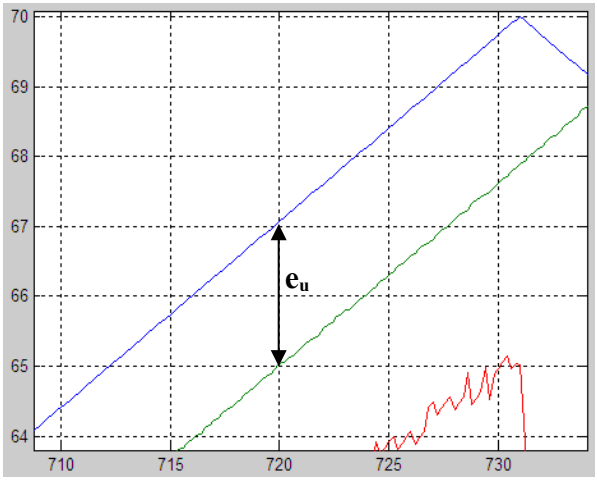
<u>Do sprawozdania</u> dołączyć uzyskany obraz ekranu.

5.4.3 Analiza otrzymanych przebiegów

Dokonać przybliżenia wykresu w obszarze, w którym zachodzi ustabilizowane śledzenie wartości zadanej narastającej liniowo. W przypadku przebiegu sinusoidalnego, obszar najlepszej liniowości przypada na otoczenie punktu przejścia przez wartość „zero”. W przypadku przebiegu trójkątnego, najlepiej wykorzystać końcowy zakres przedziału narastania. Zasadę realizacji przybliżenia ukazują poniższe rysunki:







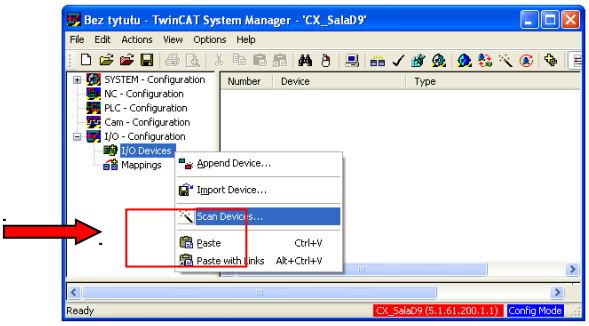
W sprawozdaniu:

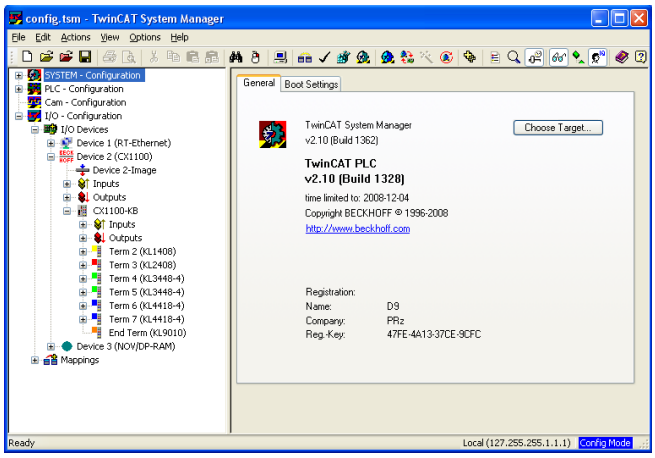
1. **Wyjaśnić na podstawie otrzymanych przebiegów bądź wykresów z poprzedniego punktu, dlaczego przebieg sterowania w przypadku śledzenia sygnału sinusoidalnego ma kształt zgodny z tym sygnałem (także sinusoidalny), natomiast dla trójkątnego przebiegu wartości zadanej nie uzyskuje się trójkątnego przebiegu sterowania**
2. **Zamieścić obraz ekranu z wykresem po przybliżeniu, pozwalającym na odczytanie wartości błędu ustalonego śledzenia. Podać odczytaną wartość błędu ustalonego e_u.**
3. **Wyjaśnić czy i jak wartość odczytanego błędu ustalonego śledzenia wynika z parametrów obiektu i nastaw regulatora. Wyznaczyć teoretyczną wartość błędu ustalonego. Czy teoretyczna wartość zgadza się z wartością odczytaną z wykresu?**

6. Dodatki

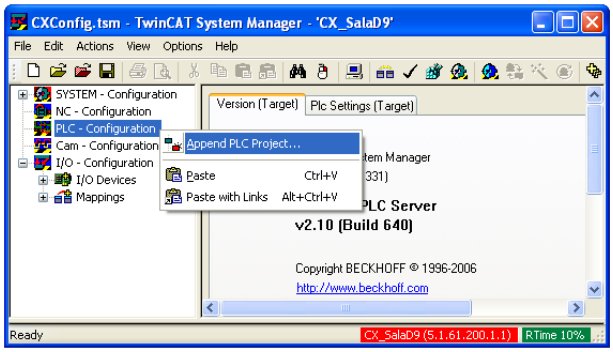
Dodatek A
Przygotowanie projektu w TwinCAT System Manager

Otworzyć **TwinCAT System Manager** i utworzyć w nim nowy projekt. Połączyć się ze sterownikiem CX1000 (**SYSTEM-Coniguration → Choose Target….**). Następnie uruchomić skanowanie konfiguracji sprzętowej sterownika i upewnić się, że nastąpiło wykrycie modułów analogowych we/wy, odnotowane przez podwójne wpisy KL3448 oraz KL4418.

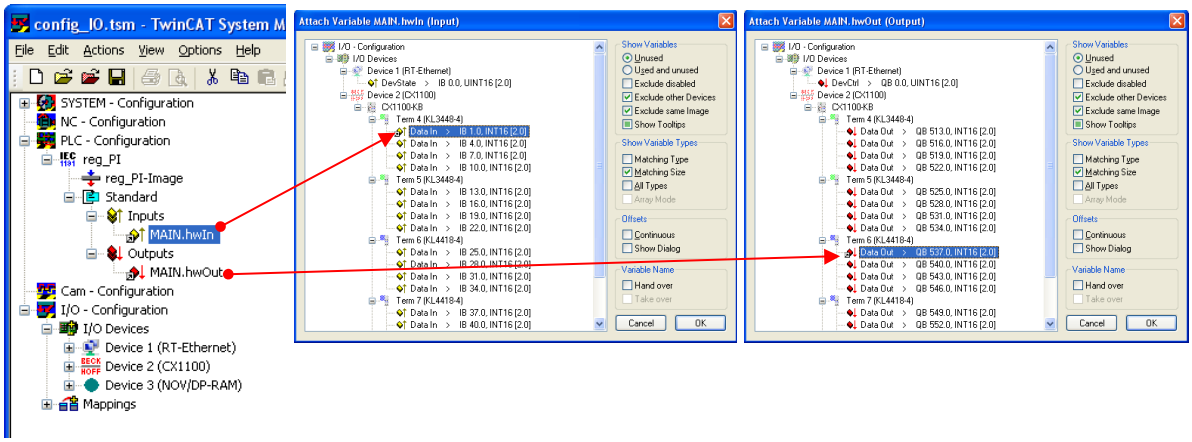




Wczytać konfigurację (**PLC-configuration**) wygenerowaną dla programu PLC, wybierając odpowiedni plik ***.tpy**.

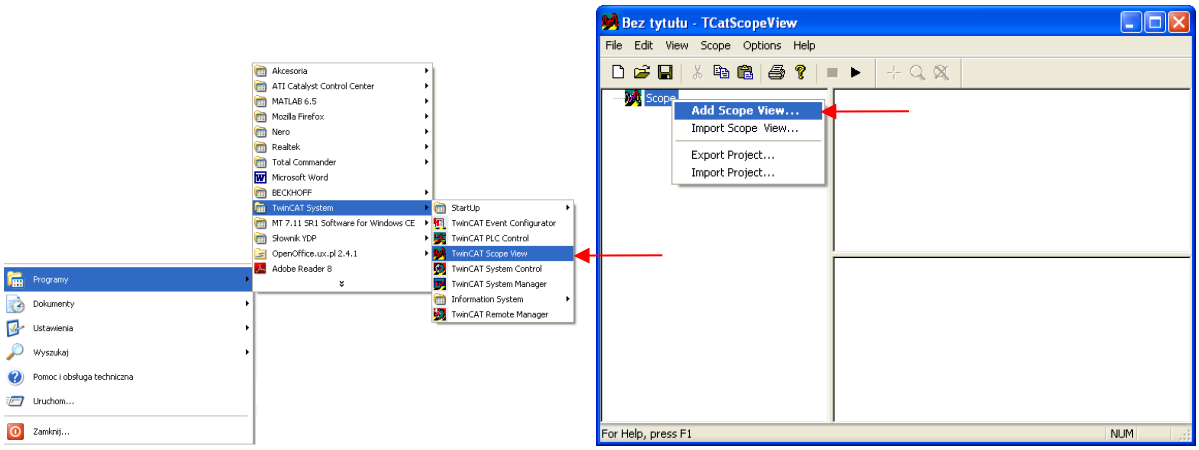


Następnie wykonać odpowiednie powiązanie adresów zmiennych zadeklarowanych w programie z fizycznymi lokalizacjami przestrzeni obrazu procesu (**PLC-configuration → Standard → Inputs/Outputs**).

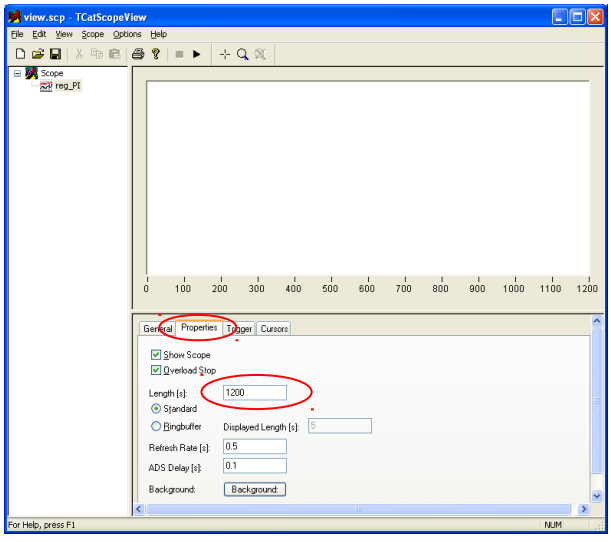


Dodatek B
Przygotowanie obserwacji i rejestracji przebiegów w TwinCAT Scope View

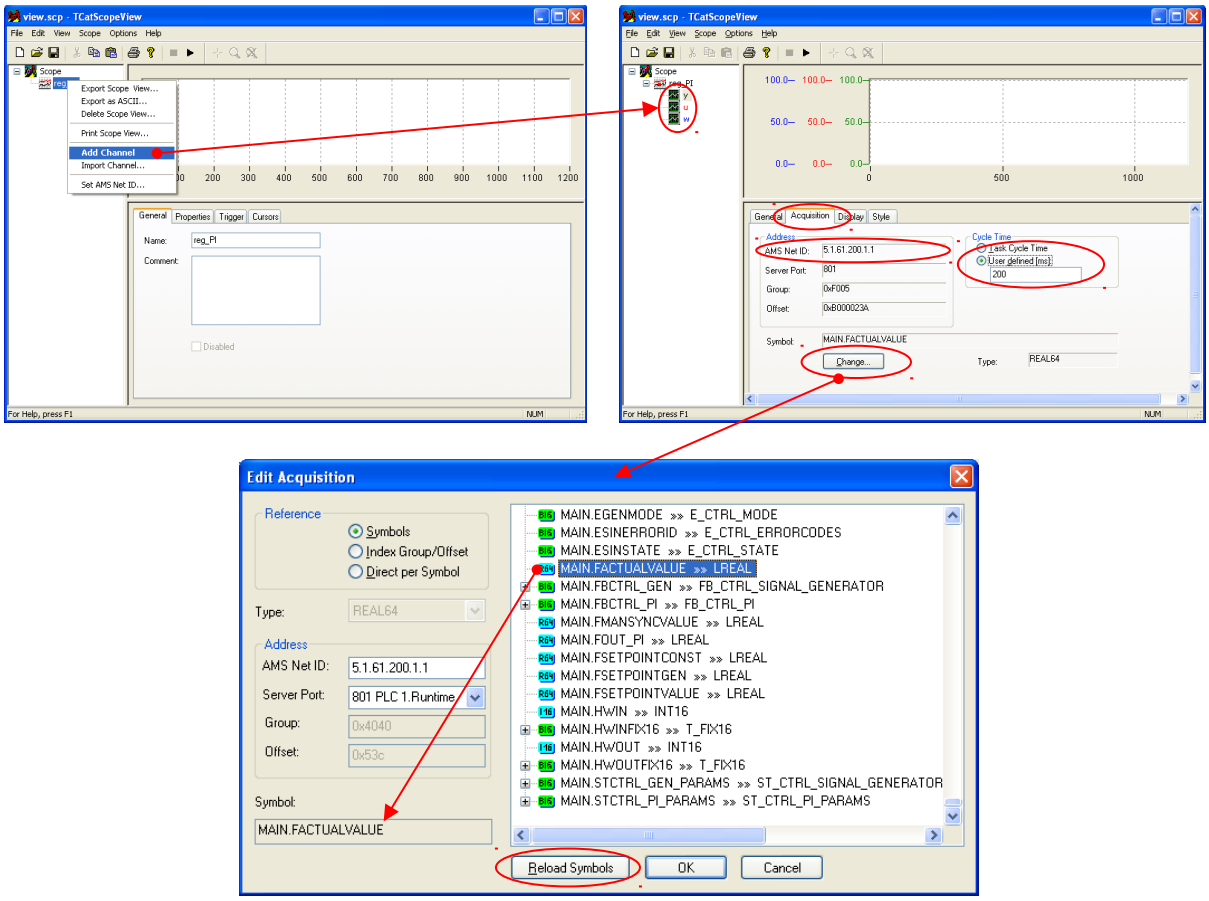
Wizualizację i rejestrację czasowego przebiegu wartości zmiennych procesowych należy zrealizować przy wykorzystaniu programu **TwinCAT Scope View** z pakietu **TwinCAT System**. Po otwarciu programu, z menu kontekstowego elementu **Scope (**prawy klawisz myszy**)**, wybrać **Add Scope View …**, a następnie podać tytuł tworzonego okna rejestracji, np. **reg_PI**.



Utworzony zostanie element okna rejestracji o podanej nazwie. W prawym dolnym panelu dostępne są zakładki, umożliwiające konfigurację utworzonego elementu. W zakładce **Properties** czas rejestracji **Length [s]** powinien zostać ustawiony na odpowiednio dużą wartość (np: 1200 s = 20 min.), ze względu na dużą stałą czasową obiektu cieplnego. Pozostałe ustawienia mogą pozostać domyślne.



Menu kontekstowe utworzonego elementu posiada pozycję dodania kanału wizualizacji **Add Channel**. Dodać trzy kanały o nazwach np. ***y***, ***u*** i ***w***, przeznaczone do rejestracji odpowiednio wyjścia obiektu, sterowania i wartości zadanej. Po wybraniu danego kanału (przez kliknięcie na odpowiadający mu element), w prawym dolnym panelu ukazują się zakładki konfiguracyjne.

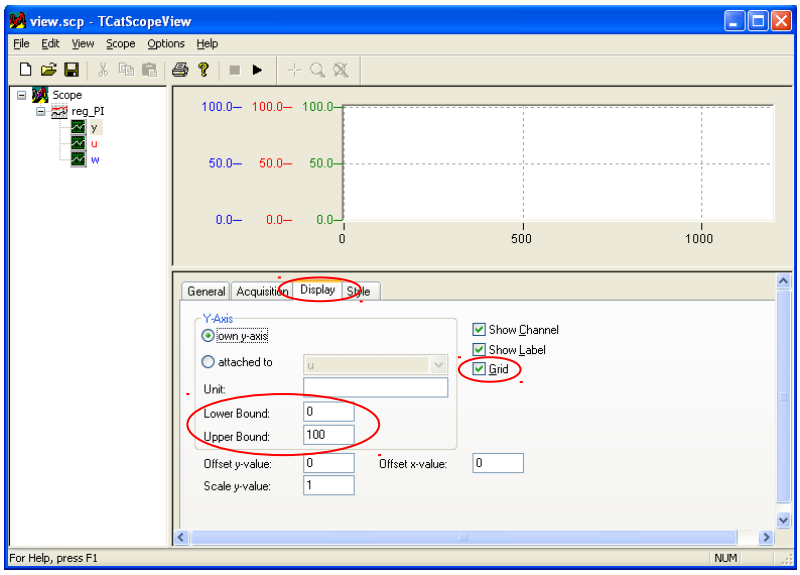


Dla każdego z kanałów zrealizować opisane niżej czynności konfiguracyjne.
W zakładce **Acquisition**:

- Jako adres **AMS Net ID** sterownika wpisać adres podłączonego sterownika CX1000 (dostępny na dolnym pasku stanu programu **TwinCAT System Manager** lub **TwinCAT PLC Control**),
- Nacisnąć przycisk **Change…**, a w otwartym następnie oknie dialogowym przycisk **Reload Symbols**. Załadowane zostaną nazwy symboliczne zmiennych z pamięci sterownika, które można wybrać do rejestracji w danym kanale, wybrać odpowiednio (np. MAIN.FACTUALVALUE dla kanału y, MAIN.FOUT dla kanału u oraz MAIN.FSETPOINTVALUE dla kanału w) i wcisnąć przycisk OK,
- W polu **Cycle Time** wybrać **User defined [ms]** i podać **200** – domyślnie proponowany czas cyklu zadaniowego jest za krótki dla badanego obiektu i spowodowałby rejestrację zbyt dużej liczby danych.

W zakładce **Display**:

- Ustawić odpowiedni zakres rejestrowanych wartości. Ze względu normalizację wartości sygnałów do przedziału 0..100, potrzeba ustawić: **Lower Bund: 0**, **Upper Bund: 100**,
- Załączyć wyświetlanie siatki wykresu (pole **Grid**).



W zakładce **Style**:

- Można zmienić kolory linii wykresu dla poszczególnych kanałów (**Pen Color → Change…**), przypisując np. kolor czerwony dla kanału sterowania (*u*), zielony dla wyjścia (*y*) i niebieski dla wartości zadanej (*w*).