Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej

Systemy automatyki DCS i SCADA

Projekt układu sterowania stanowiska INTECO TCRANE

Zdający:

Krystian Guliński
Jakub Sikora
Konrad Winnicki

Prowadzący:

mgr. inż. Andrzej Wojtulewicz

Warszawa, 26 maja 2019

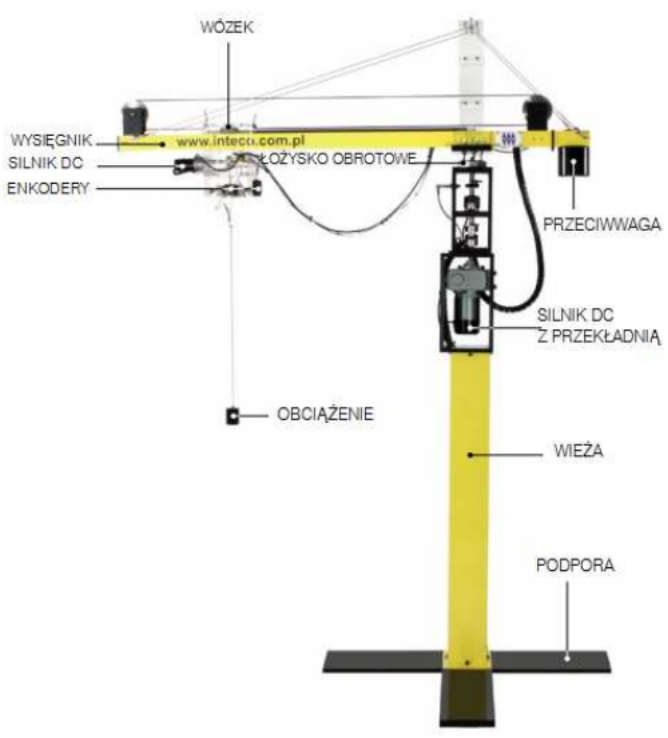
Spis treści

1. Opis stanowiska

1.1. Stanowisko TCRANE

Trójwymiarowy model laboratoryjnego modelu dźwigu ilustruje strukturę współczesnego żurawia, skutecznie odwzorowuje stosunek wielkości do maksymalnego podnoszonego ładunku. Obiekt jest wielowejściowym i wielowyjściowym systemem wyposażonym w dedykowane czujniki do mierzenia przemieszczeń i kątów.

Stanowisko laboratoryjne T-Crane posiada 5 enkoderów inkrementalnych. Trzy z nich mierzą położenie elementów napędzanych przez silniki. Dwa z nich znajdują się na karetce dźwigu i przedstawiają aktualne wychylenie obciążenia od pionu.



Rysunek 1.1. Stanowisko laboratoryjne TCRANE

W ramach projektu laboratoryjnego, mieliśmy wysterować ramię dźwigu w dwóch płaszczyznach:

— obrót kolumny dźwigu (wieży)
— ruch wózka wzdłuż ramienia

1.2. Enkodery inkrementalne

Enkoder (przetwornik położenia) służy do pomiaru położenia. W powyższej wersji mamy do czynienia z przetwornikiem obrotowym. Zatem możemy dzięki niemu określić położenie kątowe wokół osi. Jeżeli podłączymy go do liniowego układu przeniesienia napędu możemy określić położenie liniowe wyrażane w odległości.

Do określenia kierunku potrzebujemy dwóch sygnałów (tzw. fazy A i B).Do określenia pozycji wykorzystujemy dwa wejścia do zliczania impulsów z fazy A i B. Wykrywanie kierunku jest wykonywane automatycznie w sterowniku. Przy pomocy mechanizmu sprzętowych liczników możemy w dowolnym momencie odczytać aktualne położenie enkodera. W pamięci sterownika pozycja będzie przedstawiona w odpowiednim rejestrze 32 bitowym.

Zliczanie impulsów odbywa się za pomocą liczników *High Speed Counter*. Pozycja zadawana w procentach jest programowo zamieniana na impulsy enkodera według następującego wzoru:

$$I = \frac{STPT * MAX}{100\%}$$

Dla wózka jeżdżącego wzdłuż ramienia

$$MAX_{wozek} = 9000,$$

natomiast dla wieży

$$MAX_{wieza} = 2300$$

1.3. Opis wejść i wyjść obiektu

1.3.1. Wejścia cyfrowe

Wejście	Opis
X0	Enkoder inkrementalny, fala A, oś X
X1	Enkoder inkrementalny, fala B, oś X
X2	Enkoder inkrementalny, fala A, oś Y
X3	Enkoder inkrementalny, fala B, oś Y
X4	Enkoder inkrementalny, fala A, oś AX
X5	Enkoder inkrementalny, fala B, oś AX
X6	Enkoder inkrementalny, fala A, oś AY
X7	Enkoder inkrementalny, fala B, oś AY
X10	Enkoder inkrementalny, fala A, oś Z
X11	Enkoder inkrementalny, fala B, oś Z
X12	Wyłącznik krańcowy, oś Z
X13	Wyłącznik krańcowy, oś X
X14	Wyłącznik krańcowy, oś Y
X15	Flaga limitu temperatury, oś Z
X16	Flaga limitu temperatury, oś Y
X17	Flaga limitu temperatury, oś X

Tabela 1.1. Wejścia instalacji INTECO TCRANE

1.3.2. Wyjścia cyfrowe

Wejście	Opis
Y0	Sygnał PWM dla silnika DC, oś X
Y1	Sygnał PWM dla silnika DC, oś Z
Y2	Sygnał PWM dla silnika DC, oś Y
Y3	Hamulec silnika DC, oś Z
Y4	Wybór kierunku obrotów silnika DC, oś Z
Y5	Hamulec silnika DC, oś Y
Y6	Wybór kierunku obrotów silnika DC, oś Y
Y7	Hamulec silnika DC, oś X
Y10	Wybór kierunku obrotów silnika DC, oś X

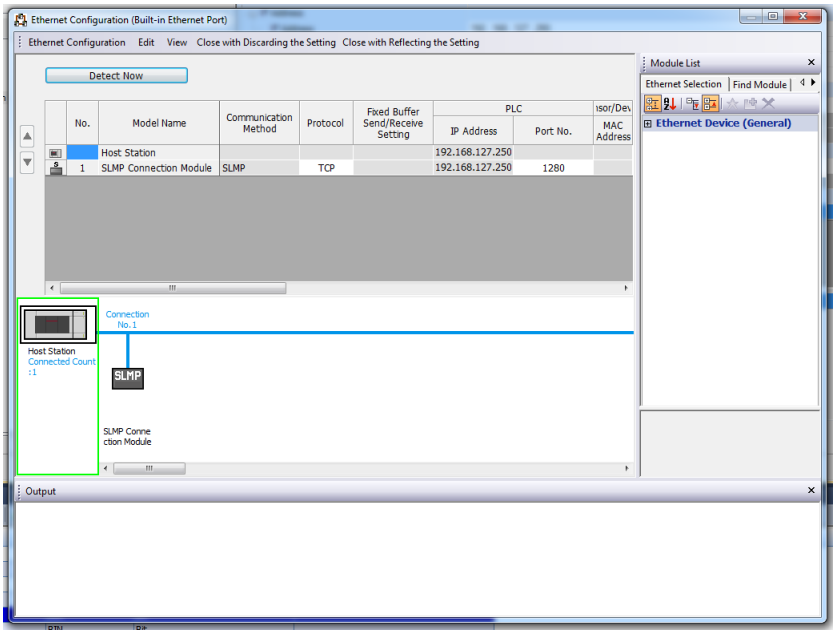
Tabela 1.2. Wyjścia instalacji INTECO TCRANE

2. Sterownik PLC

2.1. Konfiguracja sprzętowa

2.1.1. Ethernet

W celu umożliwienia komunikacji sterownika z komputerem PC, odpowiednio skonfigurowaliśmy połączenie w sieci Ethernet. Komunikacja odbywa się za pomocą protokołu SLMP (SeamLess Message Protocol) na porcie 1280.



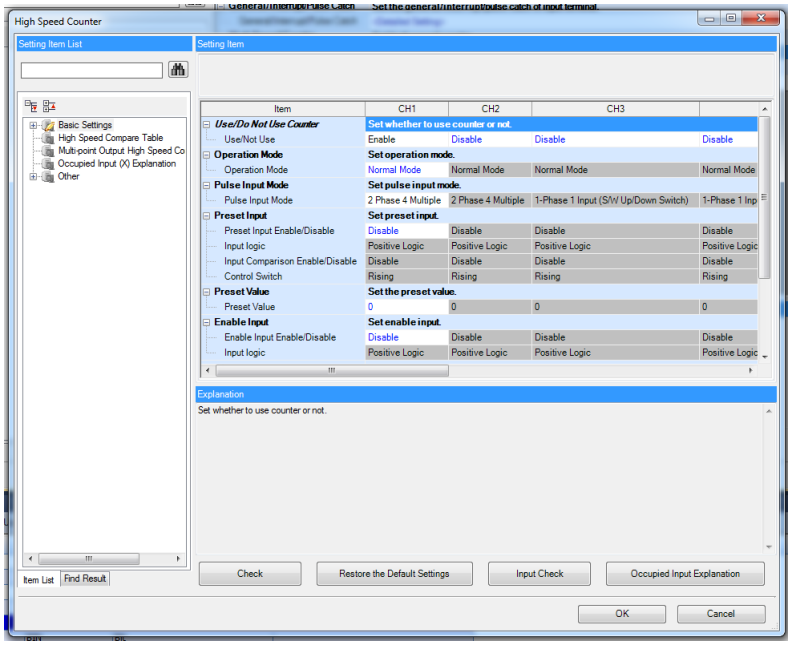
Rysunek 2.1. Konfiguracja komunikacji w sieci Ethernet w systemie GX Works3

2.1.2. Analog

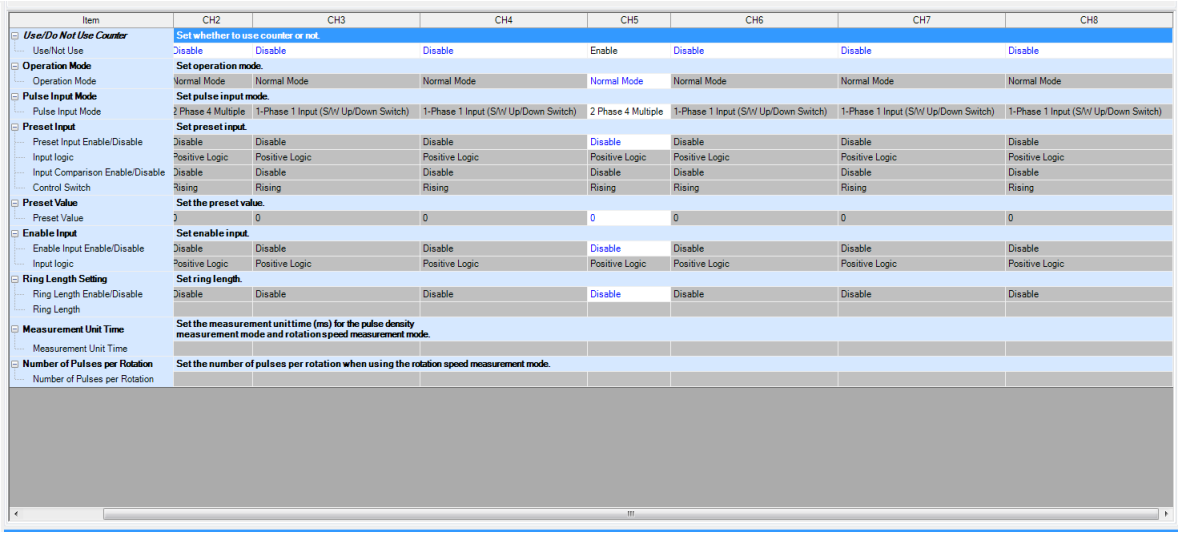
Obsługa wejść analogowych została przedstawiona jako jedno z kryterium oceny projektu. Niestety, stanowisko INTECO TCRANE nie zawiera żadnych wejść i wyjść analogowych.

2.1.3. High Speed Counter

Odczyt z enkoderów inkrementalnych odbywał się za pomocą specjalnych liczników *High Speed Counter*. Skonfigurowaliśmy dwa kanały CH1 oraz CH5 do odczytu pozycji wózka oraz obrotu wieży. Wartość pozycji wózka odczytywaliśmy spod adresu `SD4500` a wartość pozycji kątowej wieży spod adresu `SD4620`.



Rysunek 2.2. Okno konfiguracji kanału *CH1* High Speed Counters w systemie GX Works3



Rysunek 2.3. Okno konfiguracji kanału *CH5* High Speed Counters w systemie GX Works3

2.1.4. Wyjścia PWM

Wyjścia analogowe zostały zastąpione wyjściami cyfrowymi PWM. PWM czyli *Pulse Width Modifiaction* to technika przybliżania sygnału analogowego poprzez sygnał prostokątny o zmiennym wypełnieniu. W projekcie, w ten sposób sterowaliśmy silnikami prądu stałego obiektu. W programie GX Works3 odpowiednio skonfigurowaliśmy 3 kanały PWM do współpracy z trzema silnikami obiektu. Do dalszej pracy wykorzystaliśmy tylko dwa z nich. Kanał *CH1* służył do sterowania wyciągnikiem, kanał *CH2* sterował silnikiem wózka a kanał *CH3* zajmował się sterowaniem silnika obracającym wieżą.

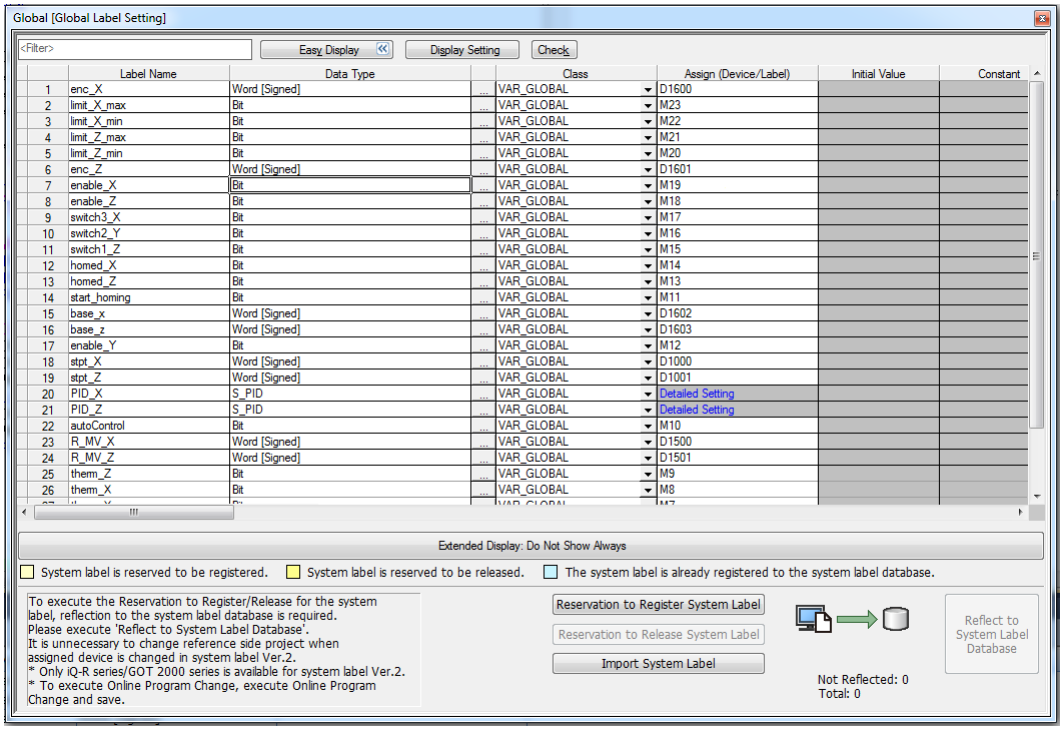
Setting Item

Item	CH1	CH2	CH3
Use PWM Output	**Set whether to use PWM output or not.**		
Use/Not Use	Enable	Enable	Enable
Output Signal	**Set the output destination device.**		
Output Signal	Y1	Y0	Y2
Pulse Width/Cycle Unit	**Set pulse width/cycle unit.**		
Pulse Width/Cycle Unit	1micro-s	1micro-s	1micro-s
Output Pulse Logic	**Set output pulse logic.**		
Output Pulse Logic	Negative Logic	Negative Logic	Negative Logic
Pulse Width	**Set pulse width.**		
Pulse Width	50 micro-s	50 micro-s	50 micro-s
Cycle	**Set cycle.**		
Cycle	100 micro-s	100 micro-s	100 micro-s

Rysunek 2.4. Okno konfiguracji kanałów PWM w systemie GX Works3

2.2. Mechanizm labeli

Zarejestrowaliśmy szereg labeli globalnych zastępujących m.in. wejścia X, ponieważ własne nazwy zwiększają uniwersalność i znacznie zwiększają czytelność kodu.



Rysunek 2.5. Ustawienia labeli globalnych

2.3. Skalowanie i bazowanie

2.3.1. Skalowanie osi X

Odczytując wartości enkodera osi X w dwóch skrajnych pozycjach i dzieląc ich różnicę(9421) przez maksymalną założoną wartość pozycji zadanej(100) otrzymaliśmy zaokrąglony współczynnik skalowania(94) Efektem jest operacja: MOV(TRUE, stpt_X*K94, PID_X.SV);

2.3.2. Skalowanie osi Z

Odczytując wartości enkodera osi Z w dwóch skrajnych pozycjach i dzieląc ich różnicę(2300) przez maksymalną założoną wartość pozycji zadanej(100) otrzymaliśmy zaokrąglony współczynnik skalowania(23) Efektem jest operacja: MOV(TRUE, stpt_Z*K23, PID_Z.SV);

2.3.3. Bazowanie osi

Bazowanie wszystkich osi inicjowane jest stanem wysokim zmiennej start_homing. Po spełnieniu tego warunku rozpoczynany jest ruch osi w kierunku krańcówek. Osie poruszają się do chwili gdy osiągnął pozycje swoich krańcówek bazujących. W następnym kroku osie są zatrzymywane, ustawiane są bity informujące o zakończeniu bazowania konkretnej osi i wykonywane jest zerowanie liczników enkoderów osi. Zbazowanie każdej z osi kończy proces bazowania dzwigu. Możliwe staje się sterowanie za pomocą pozycji zadanych poprzez regulatory PID.

```
IF start_homing = TRUE THEN
  IF homed_X = FALSE THEN
    IF switch3_X = FALSE THEN
      Y7:=TRUE; // DEAKTYWACJA HAMULCA
      PWM(TRUE, K30, K100, Y0); // PWM OSI X - KARETKA
      ELSE
      PWM(FALSE, K1, K100, Y0);
      Y7:=FALSE; // AKTYWACJA HAMULCA
      DHCMOVP(TRUE, 0, 0, SD4500); // WYZEROWANIE ENKODERA OSI X
      homed_X:=TRUE;
    END_IF;
  END_IF;
  IF homed_Z = FALSE THEN
    IF switch1_Z = FALSE THEN
      Y5:=TRUE; // DEAKTYWACJA HAMULCA
      PWM(TRUE, K30, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU
      ELSE
      PWM(FALSE, K1, K100, Y2);
      DHCMOVP(TRUE, 0, 0, SD4620); // WYZEROWANIE ENKODERA OSI Z
      Y5:=FALSE; // AKTYWACJA HAMULCA
      homed_Z:=TRUE;
    END_IF;
  END_IF;
  IF homed_X AND homed_Z THEN
    start_homing := FALSE;
  END_IF;
END_IF;
```

2.4. Obsługa I/O cyfrowych

2.4.1. Odczyt wejść cyfrowych

Stany wejść cyfrowych w celu zwiększenia czytelności kodu są przepisywane do uprzednio zdefiniowanych labeli, które następnie są wykorzystywane w programie.

Przykładowy odczyt stanu krańcówek osi:

```
MOVB(TRUE, X13, switch3_X);
MOVB(TRUE, X14, switch2_Y);
MOVB(TRUE, X12, switch1_Z);
```

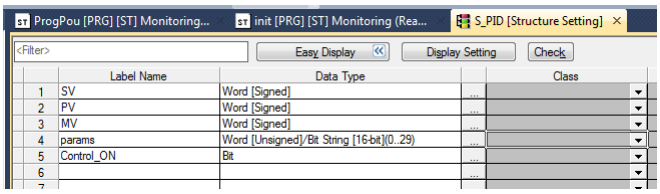
Dalsze przykładowe wykorzystanie labeli do sterowania:
IF switch3_X = FALSE THEN
Y7:=TRUE; // DEAKTYWACJA HAMULCA
PWM(TRUE, K30, K100, Y0); // PWM OSI X - KARETKA
ELSE
PWM(FALSE, K1, K100, Y0);
Y7:=FALSE; // AKTYWACJA HAMULCA
DHCMOVP(TRUE, 0, 0, SD4500); // WYZEROWANIE ENKODERA OSI X
homed_X:=TRUE;
END_IF;

2.4.2. Zapis wyjść cyfrowych

Zapis wyjść cyfrowych odbywa się poprzez bezpośrednie przypisanie stanu wyjścia. Nie zdecydowaliśmy się na zastosowanie dedykowanych labeli, ponieważ wyjść było stosunkowo niewiele. Przykładowy zapis wyjścia cyfrowego: Y5:=TRUE; // DEAKTYWACJA HAMULCA OSI Z

2.5. PID

Zastosowaliśmy wbudowaną strukturę regulatora PID. Do jej wykorzystania potrzebne było stworzenie struktury S_PID zawierającej SV, PV, MV, parametry regulatora oraz bit aktywujący regulator.



Rysunek 2.6. Struktura S_PID

W programie inicjującym znajduje się inicjalizacja parametrów regulatorów PID

//Parametry regulatora wbudowanego PID osi X
PID_X.params[0] := K100; //okres regulacji w milisekundach
PID_X.params[3] := K3; //wzmocnienie regulatora P
PID_X.params[4] := K5; //TI = 0 oznacza nieskonczony czas calkowania - inaczej mowiac calkowanie wylaczone
PID_X.params[5] := K0; //KD = 0 oznacza zerowe wzmocnienie roznicz-kowania
PID_X.params[6] := K0; //TD = 0 oznacza wylaczone rozniczkowanie
PID_X.params[22] := K100; //gorny limit wartosci wyjsciowej z regulatora - zapobiega rowniez efektowi wind-up
PID_X.params[23] := K1; //dolny limit wartosci wyjsciowej z regulatora - -||-
SET(TRUE, PID_X.params[1].5); //aktywacja limitow na wyjsciu regula-tora
SET(TRUE, PID_X.params[1].0); //trzeba odwrocic kierunek dzialania PID

//Parametry regulatora wbudowanego PID osi Z
PID_Z.params[0] := K100; //okres regulacji w milisekundach

PID_Z.params[3] := K1; //wzmocnienie regulatora P

PID_Z.params[4] := K2; //TI = 0 oznacza nieskonczony czas calkowania - inaczej mowiac calkowanie wylaczone

PID_Z.params[5] := K0; //KD = 0 oznacza zerowe wzmocnienie rozniczkowania

PID_Z.params[6] := K0; //TD = 0 oznacza wylaczone rozniczkowanie

PID_Z.params[22] := K100; //gorny limit wartosci wyjsciowej z regulatora - zapobiega rowniez efektowi wind-up

PID_Z.params[23] := K0; //dolny limit wartosci wyjsciowej z regulatora - -||-

SET(TRUE, PID_Z.params[1].5); //aktywacja limitow na wyjsciu regulatora

SET(TRUE, PID_Z.params[1].0); //trzeba odwrocic kierunek dzialania PID

Regulatory PID pracują w głównym programie. Kolejne wartości sterowania MV wyznaczane są poprzez wywołania funkcji PID(). Jeśli wyznaczone MV jest większe od 50 wykonywany jest ruch w przód z zadanym wypełnieniem PWM, a w przeciwnym wypadku wykonywany jest ruch w tył z zadanym wypełnieniem PWM.

Sekcja regulatora PID osi X:

```
MOVB(TRUE, TRUE, PID_X.Control_ON);
MOV(TRUE, enc_X, PID_X.PV);
MOV(TRUE, stpt_X*K94, PID_X.SV);
PID(PID_X.Control_ON, PID_X.SV , PID_X.PV , PID_X.params[0] , PID_X.MV);
PWM(TRUE, PID_X.MV, K100, Y0); // PWM OSI X - KARETKA
IF PID_X.MV < K50 THEN
  MOVB(TRUE, FALSE, Y10);
  MOVB(TRUE, TRUE, Y7);
  PWM(TRUE, K51-PID_X.MV, K100, Y0); // PWM OSI X - KARETKA
ELSE
  IF PID_X.MV > K50 THEN
    MOVB(TRUE, TRUE, Y10);
    MOVB(TRUE, TRUE, Y7);
    PWM(TRUE, PID_X.MV-K49, K100, Y0); // PWM OSI X - KARETKA
    ELSE
    MOVB(TRUE, FALSE, Y7);
    PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA
  END_IF;
END_IF;
```

Sekcja regulatora PID osi Z:

```
MOVB(TRUE, TRUE, PID_Z.Control_ON);
MOV(TRUE, enc_Z, PID_Z.PV);
MOV(TRUE, stpt_Z*K23, PID_Z.SV);
PID(PID_Z.Control_ON, PID_Z.SV , PID_Z.PV , PID_Z.params[0] , PID_Z.MV);
PWM(TRUE, PID_Z.MV, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU
IF PID_Z.MV < K50 THEN
  MOVB(TRUE, FALSE, Y6);
  MOVB(TRUE, TRUE, Y5);
```

```
      PWM(TRUE, K51-PID_Z.MV, K100, Y2); // PWM OSI Z - OBRÓT
DZWIGU
      ELSE
      IF PID_Z.MV > K50 THEN
        MOVB(TRUE, TRUE, Y6);
        MOVB(TRUE, TRUE, Y5);
        PWM(TRUE, PID_Z.MV-K49, K100, Y2); // PWM OSI Z - OB-
RÓT DZWIGU
        ELSE
        MOVB(TRUE, FALSE, Y5);
        PWM(FALSE, K1, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU
      END_IF;
    END_IF;
```

2.6. Tryb sterowania ręcznego

Sterowanie ręczne możliwe jest po zresetowaniu flagi AUTO_CONTROL(flaga domyślnie ustawiona). Jeśli flaga jest zresetowana program przechodzi do sekcji kodu w której odbywa się bezpośrednie wysterowanie wyjść PWM wartościami zmiennych R_MV_Z dla osi Z i R_MV_X dla osi X

Sekcja kodu sterowania ręcznego osią Z:

```
    IF enc_Z > K2350 OR enc_Z < -K50 THEN
      // PRZEKROCZENIE ZAKRESÓW OSI Z
      MOVB(TRUE, FALSE, Y5);
      PWM(FALSE, K1, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU
      MOVB(TRUE, FALSE, homed_Z);
      ELSE
      //STEROWANIE RĘCZNE OSI Z
      PWM(TRUE, R_MV_Z, K100, Y2); // PWM OSI Z - OBRÓT DZWI-
GU
      IF R_MV_Z < K50 THEN
        MOVB(TRUE, FALSE, Y6);
        MOVB(TRUE, TRUE, Y5);
        PWM(TRUE, K51-R_MV_Z, K100, Y2); // PWM OSI Z - OBRÓT
DZWIGU
        ELSE
        IF R_MV_Z > K50 THEN
          MOVB(TRUE, TRUE, Y6);
          MOVB(TRUE, TRUE, Y5);
          PWM(TRUE, R_MV_Z-K49, K100, Y2); // PWM OSI Z - OB-
RÓT DZWIGU
          ELSE
          MOVB(TRUE, FALSE, Y5);
          PWM(FALSE, K1, K100, Y2);
        END_IF;
      END_IF;
    END_IF;
```

Sekcja kodu sterowania ręcznego osią X:

```
    IF enc_X > K9450 OR enc_X < -K50 THEN
      // PRZEKROCZENIE ZAKRESÓW OSI X
      MOVB(TRUE, FALSE, Y7);
      PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA
```

```
      MOVB(TRUE, FALSE, homed_X);
      ELSE
      // STEROWANIE RĘCZNE OSI X
      PWM(TRUE, R_MV_X, K100, Y0); // PWM OSI X - KARETKA

      IF R_MV_X < K50 THEN
      MOVB(TRUE, FALSE, Y10);
        MOVB(TRUE, TRUE, Y7);
        PWM(TRUE, K51-R_MV_X, K100, Y0); // PWM OSI X - KA-
RETKA
        ELSE
        IF R_MV_X > K50 THEN
          MOVB(TRUE, TRUE, Y10);
          MOVB(TRUE, TRUE, Y7);
          PWM(TRUE, R_MV_X-K49, K100, Y0); // PWM OSI X - KA-
RETKA
          ELSE
          MOVB(TRUE, FALSE, Y7);
          PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA
        END_IF;
      END_IF;
    END_IF;
```

2.7. Zabezpieczenia ruchów krańcowych

W projekcie założyliśmy, że sytuacja w której któraś z osi wyjdzie poza założony dopuszczalny zakres ruchów jest ona natychmiast zatrzymywana i gaszona jest flaga zbazowania danej osi. Działaniem naprawczym w takiej sytuacji jest przejście na tryb sterowania automatycznego i ponowne zainicjowanie bazowania osi ustawiając flagę start_homing sekcja kodu zabezpieczająca ruchy krańcowe osi X:

```
    IF homed_X = TRUE THEN
      IF enc_X > K9450 OR enc_X < -K50 THEN
        MOVB(TRUE, FALSE, Y7);
        PWM(FALSE, K1, K100, Y0); // PWM OSI X - KARETKA
        MOVB(TRUE, FALSE, homed_X);
      ELSE
        //automatyczne lub ręczne sterowanie osi X
        ...
      END_IF;
    END_IF;
```

Sekcja kodu zabezpieczająca ruchy krańcowe osi Z:

```
    IF homed_Z = TRUE THEN
      IF enc_Z > K2350 OR enc_Z < -K50 THEN
        MOVB(TRUE, FALSE, Y5);
        PWM(FALSE, K1, K100, Y2); // PWM OSI Z - OBRÓT DZWIGU
        MOVB(TRUE, FALSE, homed_Z);
      ELSE
        //automatyczne lub ręczne sterowanie osi Z
        ...
      END_IF;
    END_IF;
```

2.8. Język ST

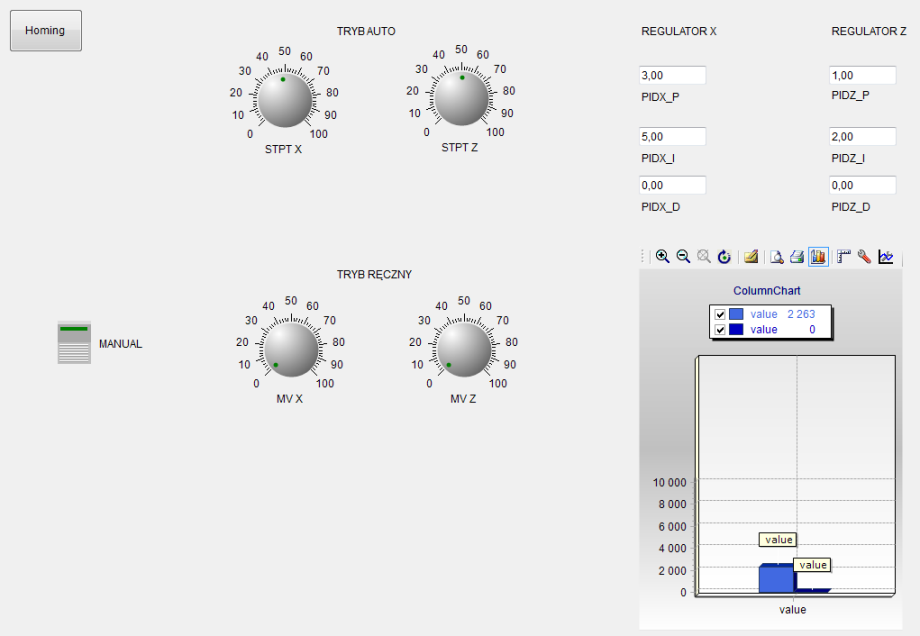
Listingi kodu zawarte powyżej jednoznacznie wskazują na pomyślne wykorzystanie języka ST.

3. Realizacja w systemie MAPS

3.1. Panel operatorski

Na podstawie odpowiednio wykonanej wcześniej konfiguracji sprzętowej sterownika PLC jesteśmy w stanie bezpośrednio połączyć działanie stanowiska z systemem MAPS. Po uruchomieniu serwera (oprogramowania *MAPS Server* i *Agent Server*) i środowiska do projektowania interfejsów operatorskich (*MAPS Designer*) rozpoczęsliśmy od utworzenia projektu.

Kolejnym krokiem jest dodanie do systemu MAPS odpowiednich agentów. Akwizycją danych poprzez uchwyty do agentów zajmuje się *Agent Server*. Aby nasze środowisko graficzne miało dostęp do tych danych należy zdefiniować zbiór agentów dla naszego stanowiska. W zadaniu wykorzystujemy podstawowe rodzaje agentów służące do pozyskiwania danych.



Rysunek 3.1. Panel operatorski stanowiska w systemie MAPS

Przy dodawaniu nowych agentów należało zwrócić uwagę na funkcjonalność jaką będziemy chcieli uzyskać. Dane, które będziemy chcieli tylko i wyłącznie wyświetlać nie powinny być oznaczone jako edytowalne z poziomu interfejsu graficznego (pole *Output enabled*). Zapewnia to większe zabezpieczenie na błędy w trakcie projektowania środowiska graficznego, ponieważ dane wyświetlane mogą być krytyczne z punktu widzenia stanowiska, a ich zewnętrzna zmiana może spowodować awarie. Dane, na które operator będzie mógł wpływać z poziomu środowiska graficznego analogicznie muszą być oznaczone jako edytowalne. Ogólnie przy definiowaniu agentów należy podwójnie sprawdzać poprawność wpisywanych adresów, szczególnie w przypadku struktur, do których trzeba wyznaczać offset od adresu początkowego struktury przy okrelnianiu agenta dla jej pola.

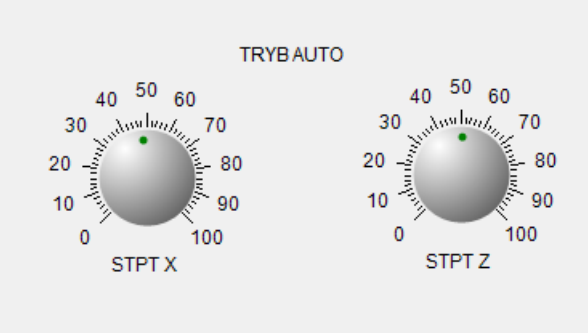
Otrzymany panel operatorski realizuje określoną w zadaniu funkcjonalność, informuje operatora o stanie obiektu oraz pozwala na wpływ na sposób i parametry sterowania obiektem.

3.2. Kalibracja

Praca z obiektem rozpoczyna się od skalibrowania jego enkoderów absolutnych na podstawie krańcówek umieszczonych na osiach obiektu. Inicjalizacja tego procesu rozpoczyna się od naciśnięcia znajdującego się w lewej górnej części przycisku *Homing*. Po restarcie urządzenia nie jest możliwy żaden tryb pracy bez pomyślnego przejścia przez proces kalibracji, dlatego przycisk umieszczony jest intuicyjnie w lewej górnej części interfejsu w celu określenia odpowiedniej kolejności postępowania w trakcie obsługi obiektu.

3.3. Sterowanie auto/ręka

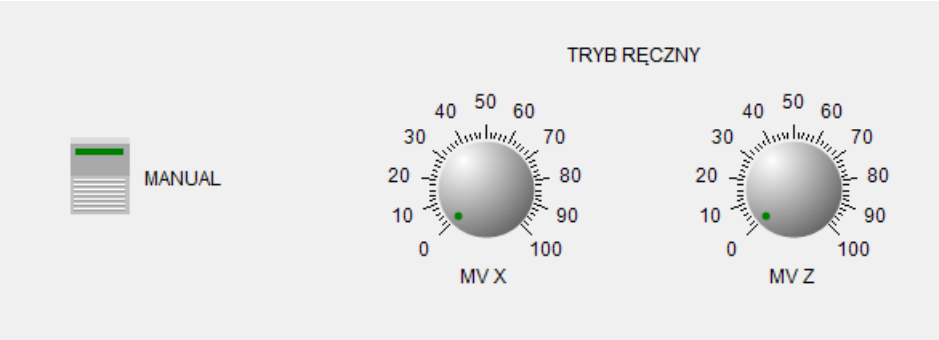
Sterowanie w trybie automatycznym pozwala na ustawianie wartości docelowych stanu obiektu (stopnia obrotu wieży i wysunięcia wózka). Zmiana tych wartości następuje poprzez ustawienie ich pokrętłem w środowisku graficznym operatora. Pokrętło *STPT_Z* odpowiada za wartość zadaną stopnia obrotu wieży w procentach. Pokrętło *STPT_X* odpowiada natomiast za wartość zadaną stopnia wysunięcia wózka w procentach.



Rysunek 3.2. Fragment panelu operatorskiego odpowiedzialny za sterowanie w trybie automatycznym

Sterowanie w trybie ręcznym jest domyślnie nieaktywne. Aby je aktywować należy wcisnąć przycisk *MANUAL* umieszczony po lewej stronie pokręteł odpowiedzialnych za sterowanie ręczne. Po aktywacji trybu ręcznego mamy możliwość zmiany prędkości obrotu wieży oraz wysunięcia wózka. Pokrętło *MV_X* odpowiada za zadawanie prędkości wózka, wartości poniżej 50 powodują cofanie się wózka, natomiast wartości powyżej 50 powodu ruch w przód. Pokrętło *MV_Z* odpowiada za nadawanie prędkości obrototowej wieży, analogicznie wartość poniżej 50 powoduje ruch w lewo, a wartość powyżej 50 powoduje ruch w prawo.

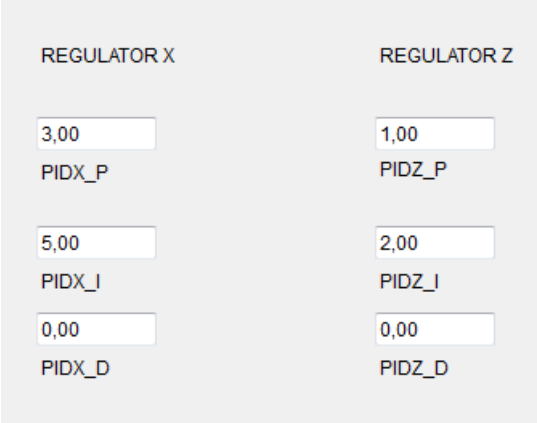
Deaktywacja trybu ręcznego zachodzi poprzez ponowne wciśnięcie przycisku *MANUAL*, po czym następuje przekazanie sterowania do regulatorów w trybie automatycznym.



Rysunek 3.3. Fragment panelu operatorskiego odpowiedzialny za sterowanie w trybie ręcznym

3.4. Nastawy regulatorów

Operator w ramach obsługi panelu operatorskiego ma również możliwość zmiany nastaw regulatorów PID odpowiedzialnych za nadążanie za wartościami zadanymi położenia w osiach. Na poniższej ilustracji przedstawiony jest fragment panelu operatorskiego, w którym widoczne są pola tekstowe pozwalające na edycje tych parametrów. Kolumna po lewej stronie pozwala na zmianę parametrów regulatora PID odpowiedzialnego za położenie wózka, natomiast kolumna po prawej stronie pozwala na zmianę parametrów regulatora PID odpowiedzialnego za obrót wieży.



Rysunek 3.4. Fragment panelu operatorskiego odpowiedzialny za zmianę wartości nastaw regulatorów w trybie automatycznym

3.5. Wykresy

Znajdujący się na panelu operatora wykres kolumnowy wizualizuje aktualny stan obiektu względem wartości zadanych dla regulatora w celu zapewnienia łatwego sposobu na weryfikacje poprawności stanu obiektu przez operatora nawet z pewnej odległości w sytuacjach opuszczenia stanowiska.