

## **Spis treści**

<b>1 WPROWADZENIE .....</b>	<b>5</b>
1.1 System DCS .....	5
1.2 System SCADA.....	6
<b>2 INFRASTRUKTURA PROGRAMOWA W LABORATORIUM .....</b>	<b>8</b>
2.1 System OVATION .....	8
2.1.1 Warstwa aplikacyjna - hardware.....	8
2.1.2 Warstwa obiektowa systemu OVATION .....	9
2.2 System MAPS .....	10
2.2.1 Warstwa aplikacyjna – hardware.....	10
2.2.2 Warstwa obiektowa systemu MAPS.....	11
<b>3 INFRASTRUKTURA SPRZĘTOWA W LABORATORIUM .....</b>	<b>13</b>
3.1 Obiekt termiczny .....	13
3.1.1 Bezpieczeństwo i wymagania .....	13
3.1.2 Charakterystyka obiektu .....	14
3.1.3 Konfiguracja procesu.....	15
3.2 INTECO – Dźwig (TCRANE).....	17
3.3 INTECO – Serwomechanizm (SERVO).....	20
3.4 INTECO – Helikopter (TRAS) .....	21
3.5 INTECO - Zbiorniki wodne (TANKS) .....	23
3.6 Podłączenie zestawów INTECO .....	24
3.7 Podłączenie zestawów INTECO .....	24
<b>4 PROGRAMOWANIE W SYSTEMIE OVATION.....</b>	<b>26</b>
4.1 Wprowadzenie.....	26
4.2 Podstawowe aplikacje systemu OVATION .....	26
4.2.1 Narzędzia projektowe .....	26
4.2.2 Narzędzia do pracy online .....	27
4.3 Przykład 1 – logika sterowania binarna .....	28
4.3.1 Etap projektowania logiki.....	28

4.3.2	Wykorzystanie narzędzi do pracy online .....	33
4.4	Przykład 2 – logika sterowania ciągłego.....	37
4.4.1	Etap projektowania logiki.....	37
4.4.2	Wykorzystanie narzędzi do pracy online .....	42
<b>5</b>	<b>PROGRAMOWANIE W SYSTEMIE SCADA .....</b>	<b>46</b>
5.1	Wprowadzenie.....	46
5.2	Oprogramowanie sterownika PLC – GX Works 3.....	46
5.2.1	Tworzenie nowego projektu .....	47
5.2.2	Elementy języka FBD/LD .....	48
5.2.3	Definicja zmiennych.....	53
5.2.4	Tworzenie kodu sterującego .....	54
5.2.5	Kompilacja kodu.....	57
5.2.6	Konfiguracja sterownika.....	57
5.2.7	Programowanie sterownika.....	60
5.2.8	Diagnostyka, monitorowanie działania programu .....	61
5.2.9	Pierwszy program PLC.....	62
5.2.10	Wejścia analogowe – FX5-4AD-ADP .....	66
5.2.11	Wejście licznikowe – pomiar pozycji z enkodera inkrementalnego .....	69
5.2.12	Wejście licznikowe – pomiar częstotliwości .....	72
5.2.13	Wyjście cyfrowe PWM.....	74
5.2.14	Socket Communication – wysyłanie danych do MATLAB.....	76
5.2.15	Definicja tablicy typu float .....	78
5.3	Oprogramowanie MAPS Server.....	82
5.4	Oprogramowanie MAPS Config Editor .....	82
5.5	Oprogramowanie MAPS Designer.....	83
5.6	Tworzenie grafik operatorskich w środowisku MAPS .....	83
5.6.1	Opis programu MAPS Designer .....	83
5.6.2	Tworzenie nowego projektu .....	83
5.6.3	Ustawienie komunikacji ze sterownikiem PLC .....	84
5.6.4	Tworzenie grafik operatorskich .....	85
5.6.5	Definicja nowych agentów .....	85
5.6.6	Skanowanie punktów ze sterownika PLC.....	86
5.6.7	Definiowanie zachowań.....	88
5.7	Oprogramowanie MAPS Operator .....	94
5.8	Przykład 1 – logika sterowania binarna .....	96

5.8.1	Etap projektowania PLC .....	96
5.8.2	Etap projektowania MAPS .....	100
5.9	Przykład 2 – logika sterowania ciągłego .....	103
5.9.1	Etap projektowania PLC .....	103
5.9.2	Etap projektowania MAPS .....	112
<b>6</b>	<b>ĆWICZENIA W SYSTEMIE OVATION.....</b>	<b>113</b>
6.1	Ćwiczenia 1 – programowanie binarne .....	113
6.1.1	Cel ćwiczenia.....	113
6.1.2	Pomocne informacje - wskazówki .....	113
6.2	Ćwiczenia 2 – programowanie ciągłe .....	116
6.2.1	Cel ćwiczenia.....	116
6.2.2	Pomocne informacje - wskazówki .....	116
6.3	Ćwiczenia 3 4 5 – projekt struktury regulacji .....	119
<b>7</b>	<b>ĆWICZENIA W SYSTEMIE SCADA .....</b>	<b>120</b>
7.1	Ćwiczenia 1 – programowanie binarne .....	120
7.2	Ćwiczenia 2 – programowanie ciągłe .....	121
7.3	Ćwiczenia 3 4 5 – projekt struktury regulacji .....	122

# 1 Wprowadzenie

Systemy DCS (Distributed Control System) i SCADA (Supervisory Control And Data Acquisition) należą do klasy systemów odpowiedzialnych za nadzorowanie pracy procesu technologicznego lub produkcyjnego. W podstawowym wydaniu systemy DCS i SCADA odpowiadają za:

- zbieranie aktualnych danych;
- wizualizację procesu;
- sterowanie procesem;
- alarmowanie ;
- archiwizację;
- raportowanie.

W branży automatyki przyjęło się stwierdzenie, że funkcjonalność współczesnych systemów DCS i SCADA pokrywa się i często terminy DCS i SCADA używane są zamiennie. Stwierdzenie to nie jest jednak do końca prawdziwe. Warto bowiem zauważyć, że system SCADA stanowi tylko oprogramowanie a warstwa sprzętowa realizowana jest przez sterowniki programowalne PLC, które mogą być (i często są) dostarczone przez innego producenta i formalnie nie stanowią części systemu SCADA. Przy takiej definicji system SCADA nie pokrywa funkcjonalność systemu DCS, gdzie kontrolery przemysłowe stanowią integralną część systemu.

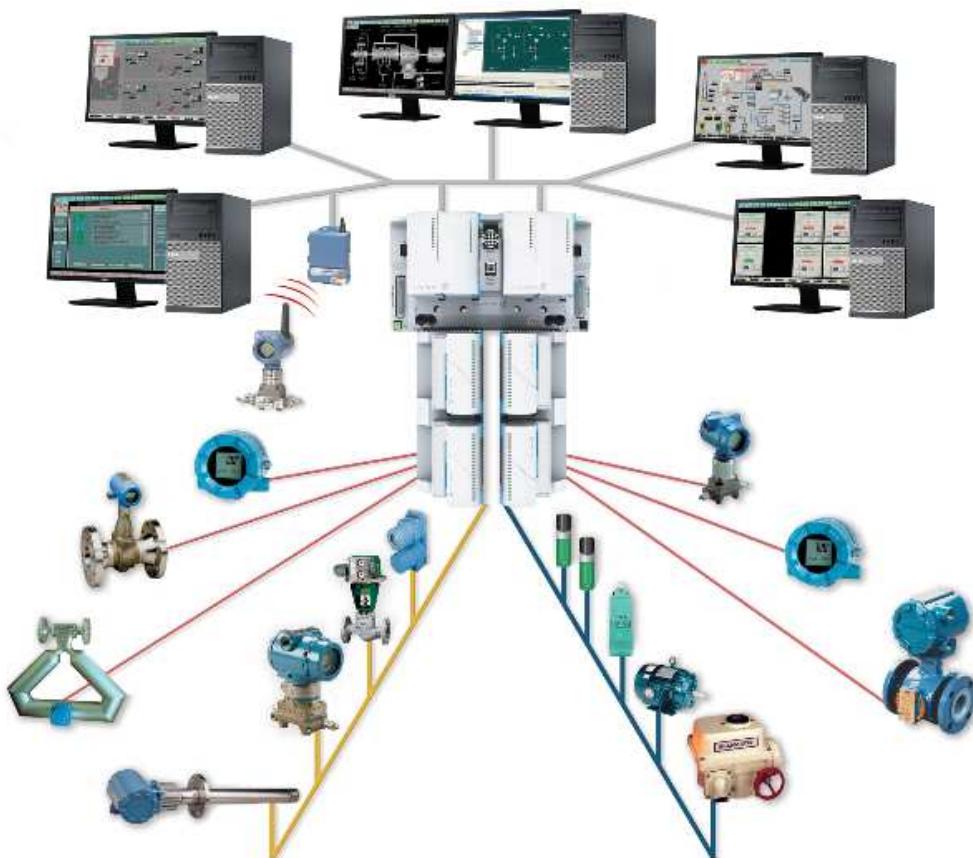
## 1.1 System DCS

Systemy DCS z racje pełnej spójności sprzętowo-systemowej są lepiej zintegrowane i pozwalają na tworzenie bardzo dużych aplikacji, dlatego znajdują głównie zastosowanie w dużych instalacjach przemysłowych, gdzie liczba punktów procesowych liczona jest w tysiącach (limit systemu OVATION to 200 tys. punktów – stan na 2017r ). Punkty procesowe używane w warstwie sprzętowej i w warstwie wizualizacji w przypadku systemów DCS są tymi samymi punktami a obie warstwy oparte są o wspólną bazę danych. Dodatkowo, systemy DCS rozbudowane są o narzędzia programistyczne pozwalające na konfigurację wejść i wyjść oraz projektowanie i programowanie struktur regulacji, dzięki czemu zapewniona jest jednorodność na poziomie projektu.

Na schemacie architektury systemu DCS przedstawionym na Rys. 1.1 ewidentnie można wyodrębnić warstwę wyższą aplikacyjną łączącą ze sobą komputery pełniące rolę: stacji procesowych operatorskich, stacji inżynierskich, stacji archiwizujących dane i serwera systemu. Warstwa aplikacyjna skupia stacje, na których zainstalowane są aplikacje pozwalające na bezpieczne prowadzenie procesu, analizę danych bieżących i historycznych oraz administrację systemem. Warstwa aplikacyjna jest też interfejsem pomiędzy procesem a wyższymi warstwami informatycznymi przedsiębiorstwa (takimi jak systemy MES [?], czy SAP [?]). Warstwa niższa to warstwa obiektowa zapewniająca komunikację urządzeń z kontrolerami przemysłowymi, które odpowiadają za wykonywanie logik sterowania.

Zastosowanie systemów DCS w dużych instalacjach przemysłowych wynika z historii, kiedy w latach 50 ubiegłego wieku rozpoczęto pracę nad systemami umożliwiającymi sterowanie procesami ciągłymi. Inspiracją do stworzenia systemów DCS były złożone systemy sterowania ciągłego używane w przemyśle, głównie w rafineriach i obiektach

chemicznych. Pierwsze takie systemy powstały w latach 40-50 ubiegłego wieku, jako systemy analogowe lub pneumatyczne i były używane jako systemy do "zarządzania" pracą i nastawami regulatorów analogowych.



Rys. 1.1 Schemat architektury systemu DCS

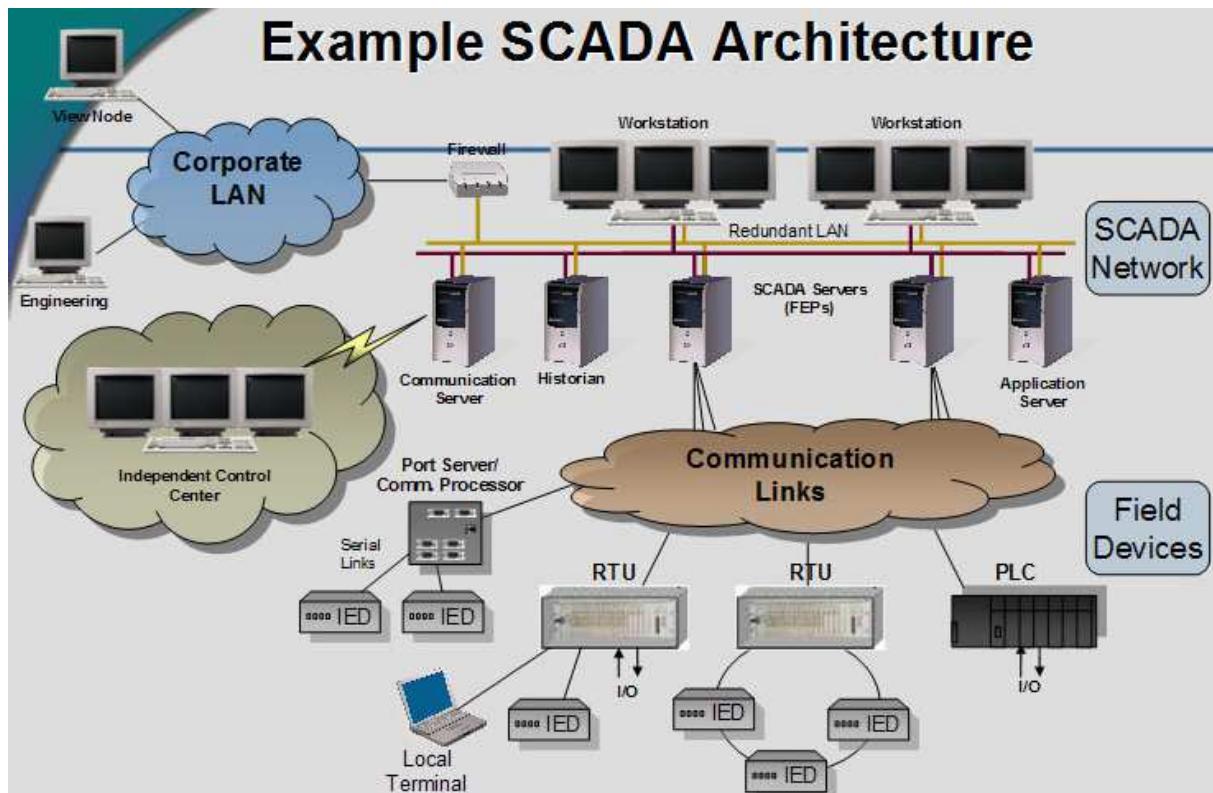
Prace nad systemami, które koncepcyjnie bardziej przypominały współczesne systemy DCS rozpoczęto w latach 60-tych. Firmy: IBM, Honeywell, Yokogawa, Taylor Instrument Company (obecnie część ABB), Westinghouse i inni stworzyli wówczas i wprowadzili na rynek pierwsze komputery dedykowane do sterowania procesami. Były to pojedyncze maszyny nadzorujące poszczególne pętle regulacji. Przełom lat 70-80 i towarzyszący im rozwój: komputeryzacji, sieci i standardów komunikacyjnych rozpoczął erę prawdziwych systemów DCS. Za pierwszy system DCS uważa się system Centrum stworzony przez firmę Yokogawa (1975r).

Praca dużych instalacji przemysłowych musi odbywać się w warunkach ciągłej pracy, gdzie przerwy (tzw. odstawienia instalacji) odbywają się zgodnie z planowanymi przeglądami i remontami. Praca w takich warunkach wymaga architektury (często redundantnej), która zapewni odpowiedni poziom niezawodności. W konsekwencji systemy DCS są zazwyczaj bardziej złożone i tym samym droższe niż rozwiązania SCADA, przez co pozycjonowane są na rynku jako rozwiązania do dużych aplikacji, z dużą liczbą punktów procesowych.

## 1.2 System SCADA

W przypadku systemów SCADA (gdzie formalnie warstwa sprzętowa nie jest częścią systemu), wyraźne jest rozgraniczenie na sprzęt (sterowniki PLC) i oprogramowanie (funkcje SCADA) a połączenie pomiędzy obiema warstwami odbywa się poprzez wymianę danych w

przestrzeni adresowej. Również od strony programistycznej, najczęściej część sprzętowa (sterowniki PLC) i część wizualizacyjna (zasadnicza SCADA) są rozdzielne i programowane niezależnymi narzędziami. Podejście takie prowadzi w konsekwencji do dwóch bytów i problemu w przypadku tworzenia dużych aplikacji. Szczególnie problemy te nasilają się w fazie utrzymania i modyfikacji aplikacji, kiedy wprowadzane zmiany muszą konsekwentnie odbywać się w obu środowiskach, co jak wiadomo często jest źródłem powstawania błędów. Pojawiły się na rynku rozwiązania, w których producenci dostarczają narzędzie potrafiące objąć warstwę SCADA i warstwę PLC, przykładem może być TIAportal firmy Siemens [<https://support.industry.siemens.com/cs/document/65601780/tia-portal-an-overview-of-the-most-important-documents-and-links-controller?dti=0&lc=en-WW>]. Rozwiązanie to w pewnym sensie stanowi nakładkę na aplikację do projektowania SCADA i programowania sterowników PLC, które w swoim zamyśle jest zbliżone do rozwiązania DCS, ale nie w pełni tożsame.



Rys. 1.2 Architektura połączenia systemu SCADA i warstwy sprzętowej

Systemy SCADA, w dużej mierze przez niezależność warstwy programowej od warstwy sprzętowej, wydają się być lepiej skalowalne – istnieje możliwość stosowania ich w bardzo prostych aplikacjach. Szczególnie, że producenci oferują model licencjonowania „per punkt”, w którym klient może dobrać system idealnie do swoich potrzeb. Podobnie jak w przypadku systemów DCS obecna pozycja systemów SCADA wynika z historii – SCADA była stosowana tam gdzie była potrzeba monitorowania procesu a nie był zainstalowany system DCS. Pierwsze wdrożenia systemów SCADA przypadają na lata 60 zeszłego wieku i były stosowane do nadzorowania i wizualizacji procesów sterowanych przez sterowniki PLC.

## **2 Infrastruktura programowa w laboratorium**

Laboratorium wyposażone jest w system klasy DCS o nazwie OVATION [<http://www.emerson.com/en-us/automation/ovation>] produkowany przez firmę Emerson oraz system SCADA marki MAPS (Mitsubishi Adroit Process Suite) [<http://www.mapsscada.com/>] produkowany przez firmę Mitsubishi Electric. Oba systemy wykorzystywane są do sterowania wybranych obiektów laboratoryjnych. Ze względu na specyfikę obiektów oraz doposażenie systemów obiekty zostały podzielone na te, które są sterowane niezależnie z systemu OVATION oraz z systemu MAPS.

### **2.1 System OVATION**

#### **2.1.1 Warstwa aplikacyjna - hardware**

System OVATION w laboratorium automatyki na Wydziale Elektroniki i Technik Informacyjnych zainstalowany jest na dedykowanym serwerze Dell PowerEdge T320. System pełni rolę edukacyjno-demonstracyjną, dlatego sprzęt w warstwie aplikacyjnej ograniczony został do jednego serwera, który jest jednocześnie: stacją procesową operatora, stacją inżynierską, stacją archiwizacji danych i serwerem systemu. W rzeczywistych aplikacjach role poszczególnych serwerów rozbite są na niezależne maszyny.

#### **Stacja procesowa operatora**

Stacja procesowa operatora dedykowana jest dla operatorów procesów, zwykle (w zależności od złożoności infrastruktury) na obiekcie zainstalowanych jest kilka stacji operatorskich. Stacje te wyróżniają się silnie ograniczonymi prawami dostępu, operator ma możliwość uruchomienia tylko wybranych aplikacji. Dodatkowo, często zablokowane są zewnętrzne napędy i wejścia USB. Zabiegi te prowadzone są z uwagi na bezpieczeństwo systemu.

Stacje operatorskie używane są przez operatorów procesu oraz kierowników zmian operatorów. Stacje operatorskie, w przypadku procesów ciągłych, wykorzystywane są 24h na dobę, 7 dni w tygodniu a praca odbywa się w trybie trójzmianowym.

#### **Stacja inżynierska**

Stacje inżynierskie dedykowane są dla osób odpowiedzialnych za utrzymanie i dostosowanie aplikacji do bieżących potrzeb procesu. W dużych instalacjach przemysłowych konieczne jest wykonywanie modyfikacji wynikających ze zmian przepisów (np. takich jak wprowadzenie limitów na emisję gazów CO<sub>2</sub>) czy zmian w opomiarowaniu (np. dołożenie nowych pomiarów w celu lepszej kontroli procesu). Zmiany w technologii muszą zostać odzwierciedlone w systemie DCS. System OVATION zaprojektowany został jako system otwarty, w którym klient ma dostęp do wszystkich narzędzi oraz zazwyczaj pełny dostęp do aplikacji. Dostęp zapewniony jest z poziomu stacji inżynierskiej. Stacje inżynierskie, jak sama nazwa wskazuje, używane są przez inżynierów, których praca zazwyczaj ma charakter projektowy, dlatego też zazwyczaj są pracownikami firm zewnętrznych świadczących usługi na rzecz właściciela procesu. Ze względu na okresowe wykorzystywanie stacji inżynierskich, jako stacje inżynierskie wykorzystywane są zazwyczaj stacje operatorskie, na które inżynierowie aplikacyjni logują się z innymi prawami dostępu. Prawa inżynierów aplikacyjnych z oczywistych powodów są znacznie szersze niż operatorów.

## **Serwer archiwizacji danych**

Serwer archiwizacji danych jest dedykowany do rejestracji wybranych (ustawionych przez inżyniera projektowego) punktów procesowych. Zazwyczaj dla każdego punktu określona jest strefa nieczułości (tzw. „dead band”) przekroczenie której skutkuje zapamiętaniem wartości na serwerze danych. Mechanizm taki pozwala na redukcję potrzebnej pamięci na przechowywanie danych historycznych. Dane historyczne pamiętane są tak długo na ile pozwala przeznaczone miejsce na dyskach serwerów. Dostęp do danych historycznych mają zarówno operatorzy, ich kierownicy oraz inżynierowie aplikacyjni, dostęp do konfiguracji parametrów archiwizacji mają tylko inżynierowie aplikacyjni.

## **Serwer systemu**

Serwer systemu jest najważniejszym elementem systemu DCS i jest jedyną stacją w systemie, która jest absolutnie wymagana i bez której system DCS nie może technicznie istnieć - wszystkie pozostałe stacje teoretyczne mogą nie występować. Na serwerze systemu uruchomiona jest baza danych (ORACLE), w której umieszczone są wszystkie informacje konfigurujące system; architektura systemu, adresy hardwareowe, punkty procesowe, limity alarmowe, struktury logik, grafiki procesowe – wszystkie byty które wykorzystywane są w aplikacji.

Serwer systemu zarządzany jest przez administratora systemu, którego najbardziej istotnymi zadaniami są: utrzymanie systemu w sprawności, dbanie o odpowiedni poziom uprawnień użytkowników, instalacje aktualnych poprawek do systemu operacyjnego i systemu DCS oraz zapewnienie kopi bezpieczeństwa.

Praca na systemie OVATION w sali laboratoryjnej możliwa jest poprzez bezpośrednie logowanie na serwer lub poprzez logowanie na serwer przez pulpit zdalny, z komputerów umieszczonych w sali laboratoryjnej.

### **2.1.2 Warstwa obiektowa systemu OVATION**

Warstwa obiektowa systemu OVATION w laboratorium automatyki na Wydziale Elektroniki i Technik Informacyjnych podobnie jak warstwa aplikacyjna również została mocno ograniczona. W systemie zainstalowany został redundantny kontroler z kilkoma kartami wejść i wyjść. Warstwa obiektowa została zobrazowana na poniżej fotografii.

(TO DO: WSTAWIĆ ZDJĘCIE)

System przystosowano do komunikacji z pięcioma obiektami laboratoryjnymi termicznymi i stanowiskiem wentylacji. Komunikacja z jednym obiektem laboratoryjnym termicznym odbywa się w standardzie 0-10V, pozostałe urządzenia komunikują się poprzez Modbus. Infrastruktura sprzętowa została przedstawiona na

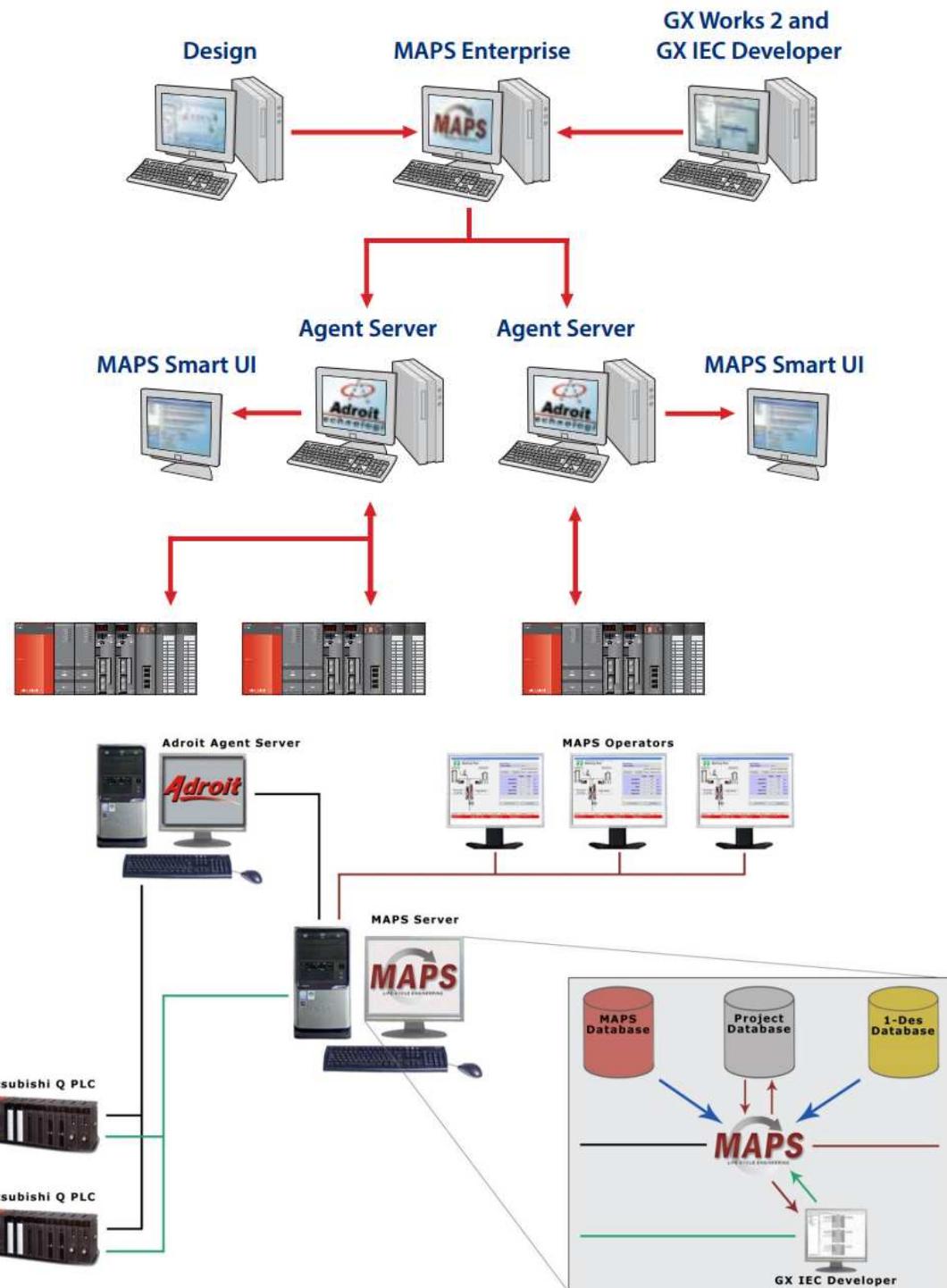
(TO DO: WSTAWIĆ RYSUNEK)

Praca w systemie odbywa się poprzez zdalny dostęp do serwera poprzez komputery stacjonarne znajdujące się w laboratorium.

## 2.2 System MAPS

### 2.2.1 Warstwa aplikacyjna – hardware

Przykładowe struktury systemu MAPS przedstawiono na Rys. 2.1.



Rys. 2.1 Przykładowe struktury systemu MAPS

System MAPS w laboratorium automatyki na Wydziale Elektroniki i Technik Informacyjnych zainstalowany jest na każdym stanowisku laboratoryjnym. Każde stanowisko pozwala na uruchomienie niezależnego serwera MAPS. Dodatkowo używany będzie

konfigurator systemu MAPS Config Editor oraz edytor grafik MAPS Designer. W rzeczywistych aplikacjach na jednym komputerze jest uruchomiony MAPS Server, gdzie możliwe jest również logowanie, jako operator. Niezależnie od tego można logować się z innego komputera lub wykorzystać do tego mechanizm Web Client, który pozwala na logowanie się z dowolnego komputera przez przeglądarkę internetową.

### **MAPS Server**

Serwer MAPS jest połączony z bazą danych MAPS SQL, która jest biblioteką szablonów. Dzięki temu rozwiązaniu każdy użyty element z biblioteki może być wielokrotnie połączony z różnymi elementami elektrycznymi automatyki wykonawczej. Każdy szablon współpracuje z grafiką SCADA i blokami funkcyjnymi PLC tworząc dodatkowo specjalne tagi, które służą łatwej wymianie danych między PLC i SCADĄ. Każdy projekt posiada własną bazę danych, która zawiera całą konfigurację systemu.

### **Agent Server**

Agent serwer zajmuje się odpisywaniem urządzeń automatyki. Wyniki są umieszczane w bazie danych. Jest to pomost łączący świat wizualizacji i świat rzeczywistych sterowników. Każdy obiekt jest swego rodzaju inteligentnym punktem, który działa w reżimie maszyny czasu rzeczywistego. W praktycznym wykorzystaniu agent zawiera wartości binarne czy numeryczne, które mogą być później wyświetcone na grafikach. Dodatkowo możliwe jest ustawienie automatycznego skalowania wartości i przedziałów alarmowych. Są to najprostsze wersje agentów, które będą używane w trakcie pracy w laboratorium. Dodatkowo można wyróżnić bardziej zaawansowane wersje agentów jak Alarm agent czy DataLog agent, które mogą wywoływać ustalone reakcje na alarm czy zapisywać dane w wyznaczonej przez nas bazie danych.

### **MAPS Config Editor**

Oprogramowanie służy do zarządzania i konfigurowania systemu. Z punktu widzenia zajęć laboratoryjnych potrzebne będzie ustawienie parametrów sterownika PLC. Pozostałe opcje będą zawierały ustawienia domyślne.

### **MAPS Designer**

Narzędzie to służy do realizacji grafik, konfiguracji agentów oraz wszelkich interakcji między grafikami a agentami. Oprogramowanie zawiera bazę grafik predefiniowanych obiektów graficznych.

### **MAPS Operator**

Dzięki temu oprogramowaniu można wyświetlać grafiki projektowane w narzędziu MAPS Designer. Wyświetlane ekranы mają tylko charakter interakcji z użytkownikiem i niemożliwe są jakiekolwiek ich zmiany.

## **2.2.2 Warstwa obiektowa systemu MAPS**

Warstwa obiektowa systemu MAPS w laboratorium automatyki na Wydziale Elektroniki i Technik Informacyjnych obejmuje sterownik PLC **FX5** firmy Mitsubishi, który będzie programowany przy pomocy aplikacji **GX Works 3**. Jest to sterownik kompaktowy zawierający wejścia, wyjścia cyfrowe oraz wejścia, wyjścia analogowe. Dodatkowo ze sterownikiem komunikuje się panel **HMI GOT** firmy Mitsubishi, na którym możliwe

wyświetlenie potrzebnych informacji. Warstwa obiektowa została zobrazowana na poniższej fotografii.



**Rys. 2.2 Warstwa obiektowa systemu MAPS**

System na każdym stanowisku przystosowano do pracy z urządzeniami firmy INTECO. Są to dedykowane zestawy laboratoryjne do komunikacji z sterownikami PLC. Aplikacja wykonawcza będzie realizowana w sterowniku, natomiast jej wizualizacja i interakcja z użytkownikiem będzie realizowana w środowisku MAPS.

Praca na każdym stanowisku odbywa się dwuetapowo. Na początku należy wykonać oprogramowanie na sterownik PLC a następnie należy wykonać wizualizację w środowisku MAPS. Do odtworzenia wizualizacji wykorzystuje się bezpośrednio środowisko MAPS Designer lub też można uruchomić aplikację operatorską MAPS Operator.

### **3 Infrastruktura sprzętowa w laboratorium**

#### **3.1 Obiekt termiczny**

**UWAGA:** Przed przystąpieniem do pracy ze stanowiskiem konieczne jest odbycie szkolenia oraz zapoznanie się z treścią podrozdziału 3.1.1

##### **3.1.1 Bezpieczeństwo i wymagania**

Korzystając ze stanowiska należy pamiętać, co następuje:

- W urządzeniu występuje napięcie sieciowe 120/230V.
- W celu wyłączenia stanowiska należy odłączyć wtyczkę zasilającą 120/230V od sieci.
- Zastosowany wyłącznik główny jest dwubiegowy, rozłącza zarówno linie L jak i N.
- Urządzenie przeznaczone jest do użytku wewnętrz pomieszczeń. Nie może pracować w warunkach wystąpienia kondensacji pary wodnej.
- Urządzenie wykonano w Klasie I ochronności i wymaga ono podłączenia uziemienia. Uziemienie wykonywane jest przewodem zasilającym.
- Utrzymywać w czystości urządzenie i jego otoczenie.

Czynności zabronione:

- Korzystanie z uszkodzonych przewodów zasilających.
- Pozostawienie bez nadzoru stanowiska będącego pod napięciem 120/230V, bez zabezpieczenia przed dostępem osób niepowołanych.
- Dotykanie wentylatorów z uwagi na możliwe wysokie obroty mogące powodować skaleczenie.
- Dotykanie rezystorów mocy i struktury obiektu w ich okolicy z uwagi na możliwość wystąpienia wysokiej temperatury na ich powierzchni, która może spowodować poparzenie.

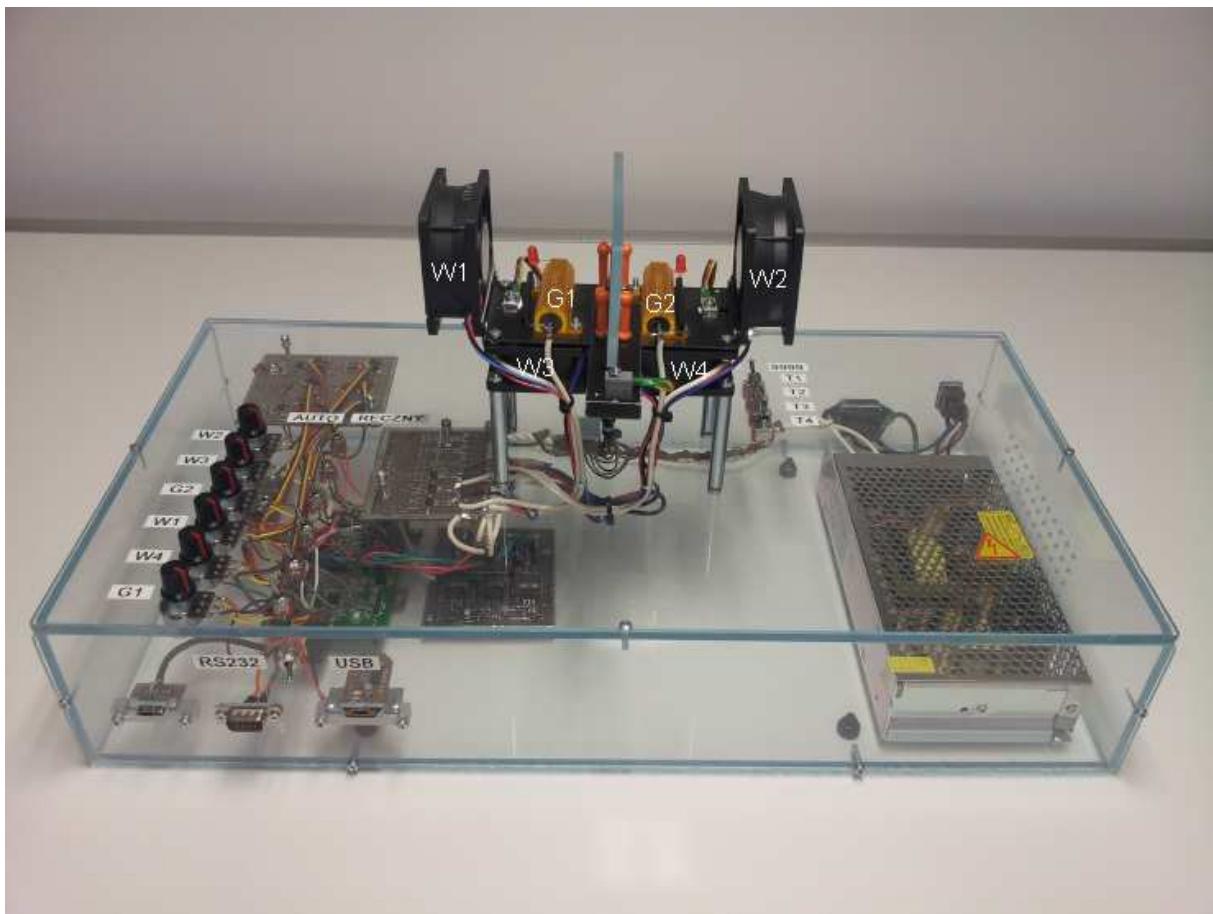
Konieczne czynności przygotowujące przed rozpoczęciem pracy:

- Sprawdzenie stanu wtyczek, gniazd i przewodów sieci 120/230V.
- Usunięcie zbędnych przedmiotów ze stanowiska pracy.
- Upewnienie się, że rozpoczęcie pracy nie spowoduje zagrożeń dla osób na stanowisku pracy oraz innych osób mogących się znaleźć w pobliżu.

Przed przystąpieniem do pracy ze stanowiskiem należy sprawdzić czy kabel zasilający jest włożony do odpowiedniego gniazda na tylnej ścianie obudowy. Jeżeli istnieją jakiekolwiek obawy, że kabel może być uszkodzony, należy zaprzestać pracy do czasu wymiany kabla. Włączenie zasilania następuje poprzez przełączenie przełącznika w pozycję „1”. Po tej czynności na zasilaczu powinna zapalić się zielona dioda. Mikrokontroler zarządzający posiada 4 LEDy: zieloną, pomarańczową, czerwoną i niebieską. Oznaczają one kolejno: inicjalizację sprzętu, dokonywanie pomiarów, błąd krytyczny, inicjalizację programową. Poprawna inicjalizacja powinna skutkować zgaszeniem wszystkich diod poza pomarańczową, która powinna zapalać się na chwilę co sekundę (oznacza to, że pomiary są okresowo wykonywane).

### 3.1.2 Charakterystyka obiektu

Obiektem regulacji jest stanowisko laboratoryjne (Rys. 3.1 i Rys. 3.2), którego struktura jest przedstawiona na Rys. 3.3. Jest to obiekt cieplny, w którym jako elementy grzewcze wykorzystano rezistory mocy, w specjalnych obudowach dobrze odprowadzających wytwarzane ciepło, oznaczone jako G1 i G2. Do chłodzenia wykorzystano wysokoobrotowe wentylatory (W1, W2, W3, W4). Zastosowano czujniki temperatury z magistralą danych OneWire – oznaczenia T1, T2, T3, T4, T5. Dodatkowo wykorzystano płytę pomiarową służącą do odczytu wartości prądu oraz napięcia – oznaczenia P1, P2.



Rys. 3.1 Stanowisko laboratoryjne z zaznaczonymi elementami wykonawczymi: elementami grzewczymi G1 i G2 oraz wentylatorami W1, W2, W3 i W4

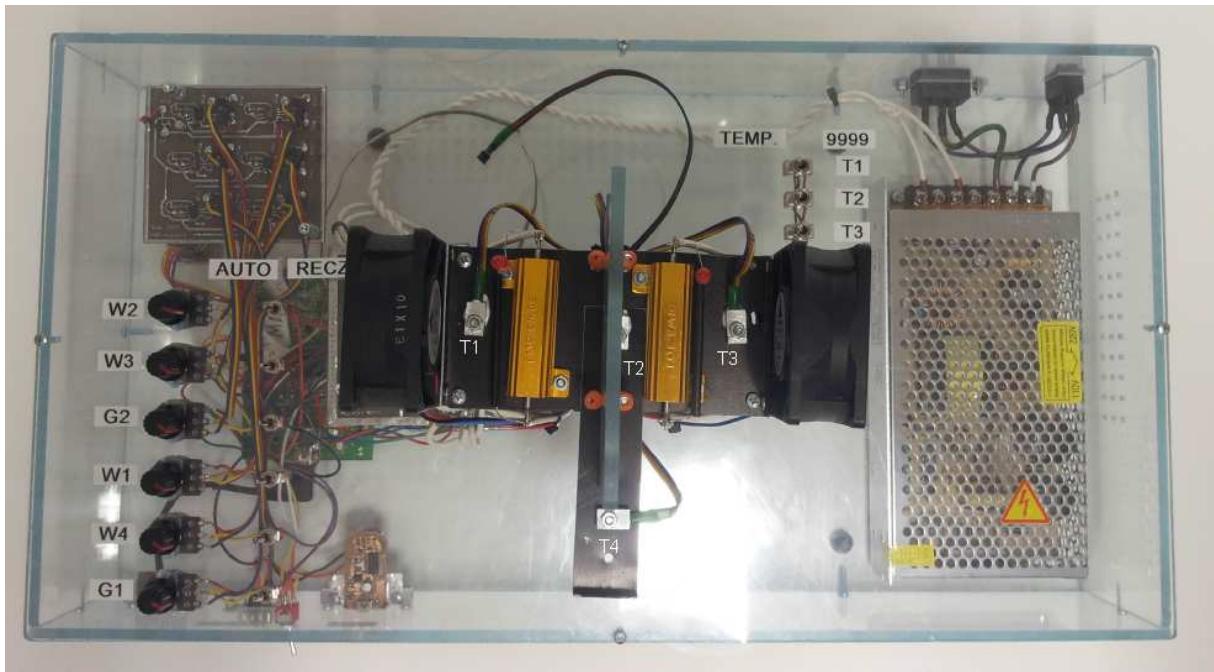
Urządzenie może pracować w trzech trybach komunikacji, poprzez:

- dedykowany protokół komunikacyjny przy użyciu standardu USB (przełącznik na frontowej ścianie musi być ustawiony w pozycji USB);

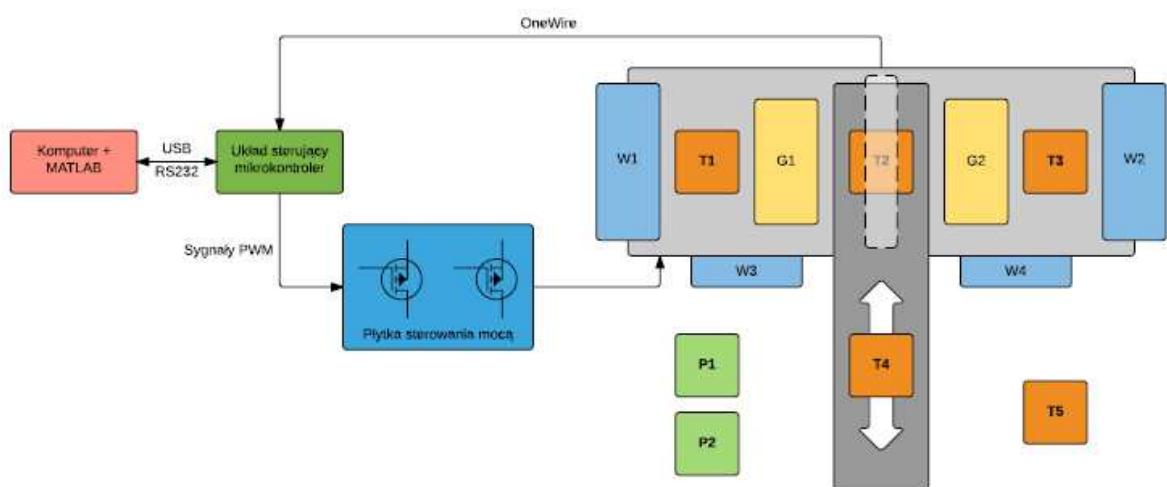
**protokół MODBUS RTU oraz standard napięciowy RS485 (frontowy przełącznik ustawiony na MODBUS/ANALOG, a przełącznik na tylnej ścianie obudowy ustawiony na MODBUS);**

- standard sygnałów analogowych 0-10V podłączając regulator przy użyciu złączy śrubowych (frontowy przełącznik ustawiony w pozycji MODBUS/ANALOG, tylny przełącznik w pozycji ANALOG).

Użytkownik ponadto ma możliwość zmiany charakteru obiektu, w tym celu zamontowana została fizyczna przegroda. Wykorzystana jest ona do oddzielenia dwóch strumieni powietrza wentylatora lewego i prawego, dzięki czemu można zredukować zakłócenia wynikające z mieszania się strumieni.



Rys. 3.2: Stanowisko laboratoryjne z zaznaczonymi czujnikami temperatury T1, T2, T3 i T4



Rys. 3.3: Schemat stanowiska [1]

### 3.1.3 Konfiguracja procesu

Omawiany obiekt należy do klasy obiektów wielowymiarowych, możliwe jest oddziaływanie 6 sygnałami wejściowymi (traktowanymi jako sygnały sterujące lub zakłócające) na 6 sygnałów wyjściowych (traktowanych jako sygnały regulowane lub procesowe) (tabl. 1), co pozwala na prowadzenie prac na stanowisku w wielu różnych konfiguracjach.

**Tabela 3.1**

<b>SYGNAŁY WEJŚCIOWE</b>	<b>Sygnały wyjściowe</b>
intensywność grzania grzałki G1	temperatura T1
intensywność grzania grzałki G2	temperatura T2
wydatek wentylatora W1	temperatura T3
wydatek wentylatora W2	temperatura T4
wydatek wentylatora W3	prąd C
wydatek wentylatora W4	napięcie V

Istnieje bardzo wiele możliwości konfiguracji procesu, w najprostszej wersji rozważa się obiekt o jednym sygnale wejściowym (np. W1) i jednym wyjściowym (np. T1), przy stałej pracy grzałki (np. G1). Znacznie bardziej elastyczny jest obiekt uwzględniający dwa sygnały wejściowe (np. G1 i W1, gdzie zmiany obciążenia wentylatora traktowane mogą być jako zakłócenie) oraz jeden sygnał wyjściowy (np. T1). Możliwe jest dalsze zwiększanie poziomu skomplikowania obiektu poprzez dodawanie kolejnych wejść i wyjść zyskując jednocześnie większą kontrolę nad rozkładem temperatur w punktach T1, do T5.

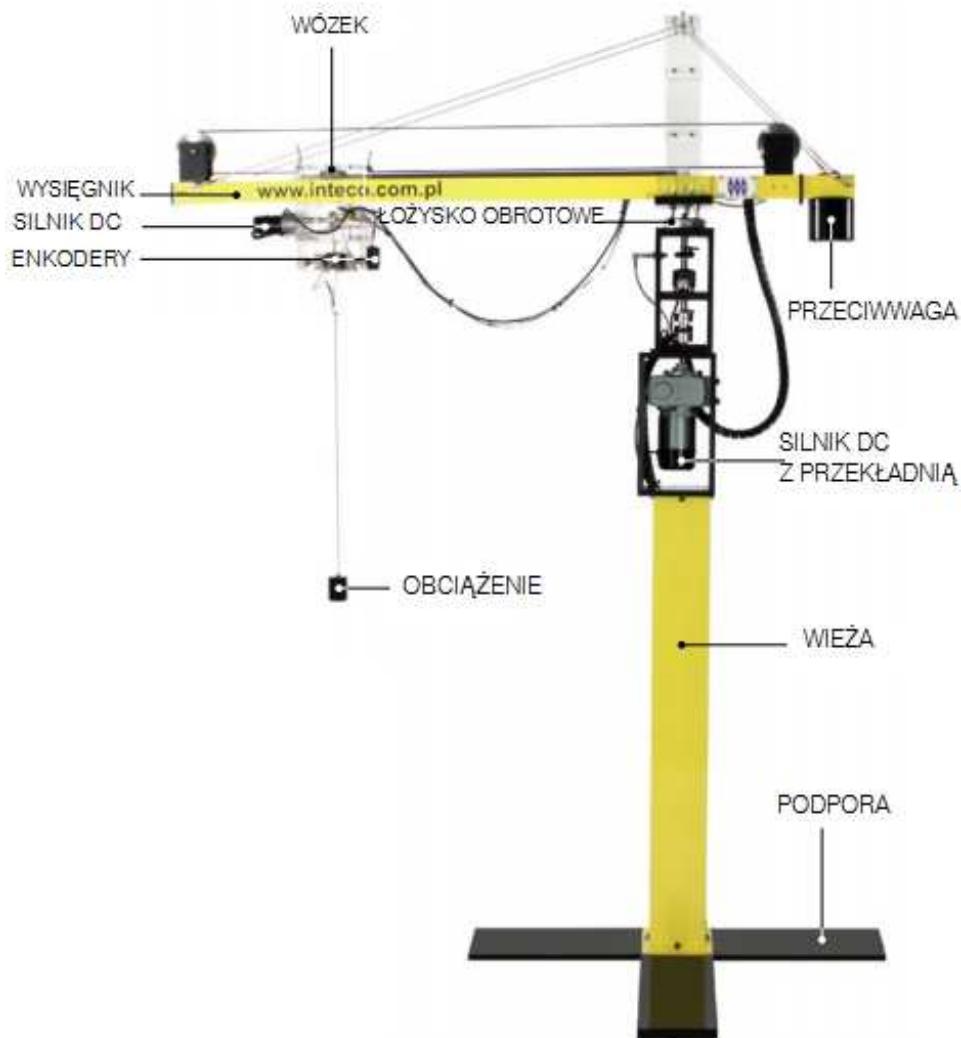
Wartymi uwagi są następujące kombinacje:

1. wejścia (W1, W3), 1 wyjście (T1) – regulacja temperatury T1 stale grzejącego się elementu (symulowane przez stałe włączoną grzałkę G1),
2. wejścia (G1, W1), 1 wyjście (T1) – regulacja temperatury T1 przy użyciu grzałki i wentylatora z uwzględnieniem zakłócenia w postaci zmieniającej się temperatury otoczenia T5,
3. wejścia (G1, W1), 2 wyjścia (T1, P1) – regulacja temperatury T1 przy użyciu grzałki i wentylatora z jednoczesną minimalizacją pobieranego prądu (a co za tym idzie mocy)
4. wejścia (G1, G2, W1, W2), 1 wyjście (T2) – regulacja temperatury T2 przy użyciu redundantnych elementów wykonawczych, które mogą pracować wymiennie,
5. wejścia (G1, G2, W1, W2), 2 wyjścia (T1, T3) – regulacja temperatur T1 i T3 z uwzględnieniem słabych zakłóceń (przegroda zamontowana) lub mocnych zakłóceń (przegroda zdjęta), głównie wynikających ze zderzających się strumieni powietrza wytwarzanych przez W1 i W2,
6. wejścia (G1, G2, W3, W4), 2 wyjścia (T2, T4) – regulacja temperatur T2 i T4, z czego zmiana temperatury odczytanej przez T4 następuje z wyraźnym opóźnieniem w stosunku do T2.

Dodatkowo każda z omawianych konfiguracji może być prowadzona w warunkach optymalizacji, tzn. w sposób zapewniający minimalizację sygnałów C i V (prądu i napięcia).

Podczas wykonywania ćwiczenia każdy zespół laboratoryjny otrzyma opis konfiguracji od prowadzącego zajęcia.

### 3.2 INTECO – Dźwig (TCRANE)

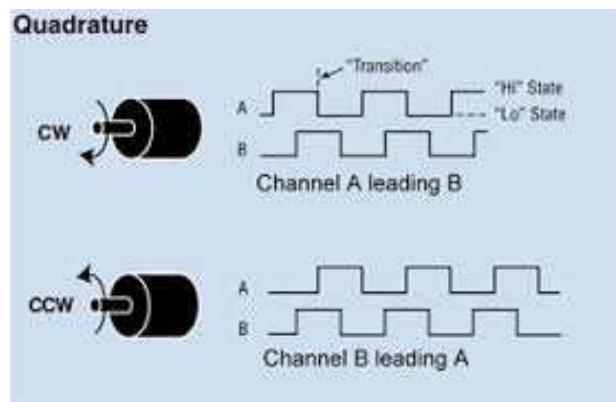


Rys. 3.4: Stanowisko laboratoryjne T-Crane

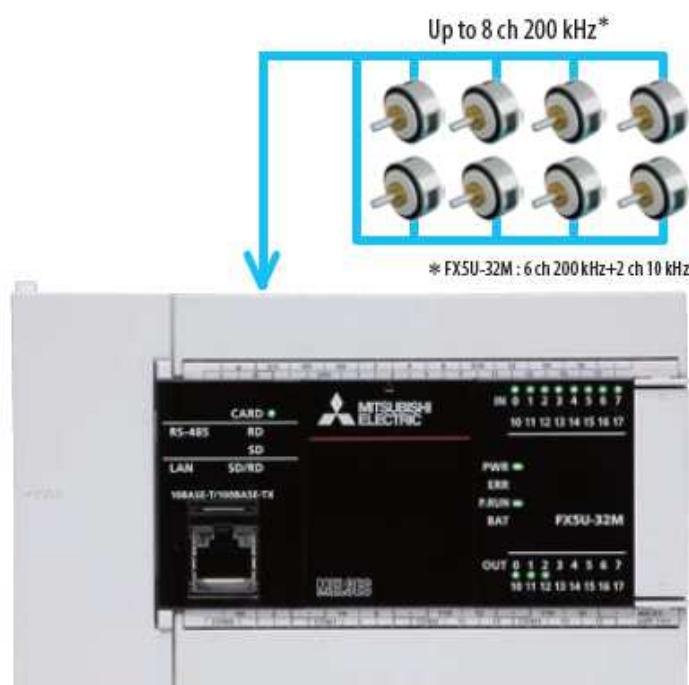
Stanowisko laboratoryjne T-Crane posiada 5 enkoderów inkrementalnych. Trzy z nich mierzą położenie elementów napędzanych przez silniki. Dwa z nich znajdują się na karetce dźwigu i przedstawiają aktualne wychylenie obciążenia od pionu.

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.

**UWAGA:** Pamiętamy, że do określenie kierunku potrzebujemy dwóch sygnałów (tzw. fazy A i B). W sterowniku wykorzystujemy dwa wejścia do zliczania impulsów z fazy A i B. Omawianą sytuację przedstawia rysunek 3.5. Wykrywanie kierunku jest wykonywane automatycznie w sterowniku. Przy pomocy mechanizmu sprzętowych liczników możemy w dowolnym momencie odczytać aktualne położenie enkodera. Rysunek 3.6 przedstawia możliwość podłączenia 8 enkoderów inkrementalnych do sterownika FX5. W pamięci sterownika pozycja będzie przedstawiona w odpowiednim rejestrze 32 bitowym. Reprezentacja liczby może być **UINT** lub **INT(U2)**.

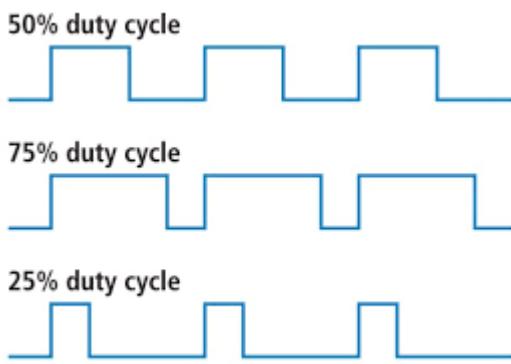


Rys. 3.5: Sygnał wyjściowy (kwadraturowy) enkodera inkrementalnego



Rys. 3.6: Wykorzystanie sygnałów kwadraturowych w sterowniku PLC

Regulacja pozycji - sterowanie silnikiem w naszym przypadku będzie sterowaniem jego prędkością obrotową. Silnik porusza elementem wykonawczym a jego położenie jest określane przez enkoder. Zmiana prędkości będzie proporcjonalna do wypełnienia sygnału PWM podanego na silnik. Sygnał PWM generowany będzie na wyjściu sterownika PLC. Rysunek 3.7 przedstawia przykładowe wypełnienie sygnału PWM.



Rys. 3.7: Prezentacja wypełnienia sygnału PWM

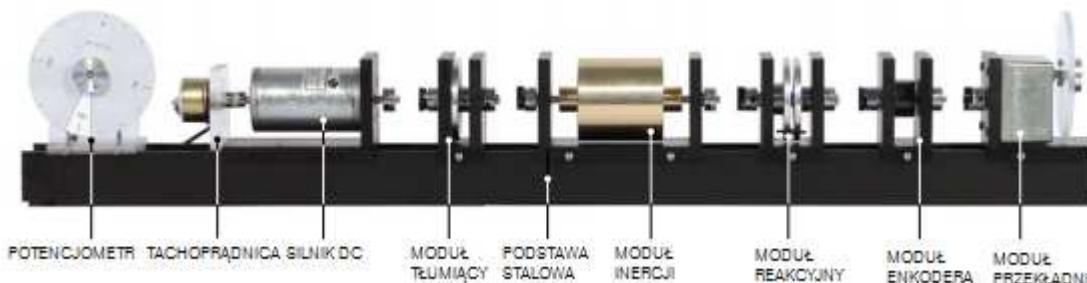
Rozdzielcość fizyczna enkoderów inkrementalnych użytych w osiach X i Z wynosi 1024 impulsy/obrót, natomiast w osi Th wynosi 512 impulsów/obrót.

Poniższa tabela przedstawia podłączenie wejść i wyjść fizycznych sterownika PLC FX5. Na jej podstawie należy utworzyć odpowiednie grupy etykiet.

	WEJŚCIA CYFROWE
X0	EncA1_X Enkoder inkrementalny, fala A, oś X
X1	EncB1_X Enkoder inkrementalny, fala B, oś X
X2	EncA2_Y Enkoder inkrementalny, fala A, oś Y
X3	EncB2_Y Enkoder inkrementalny, fala B, oś Y
X4	EncA4_AX Enkoder inkrementalny, fala A, kąt AX
X5	EncB4_AX Enkoder inkrementalny, fala B, kąt AX
X6	EncA5_AY Enkoder inkrementalny, fala A, kąt AY
X7	EncB5_AY Enkoder inkrementalny, fala B, kąt AY
X10	EncA3_Z Enkoder inkrementalny, fala A, oś Z
X11	EncB3_Z Enkoder inkrementalny, fala B, oś Z
X12	Switch1_Z Wyłącznik krańcowy, oś Z
X13	Switch3_X Wyłącznik krańcowy, oś X
X14	Switch2_Y Wyłącznik krańcowy, oś Y
X15	Therm_Z Flaga limitu temperatury, oś Z
X16	Therm_Y Flaga limitu temperatury, oś Y
X17	Therm_X Flaga limitu temperatury, oś X
	WYJŚCIA CYFROWE
Y1	PWM_Z Sygnał sterujący typu PWM dla silnika DC, oś Z. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y3	Brake_Z Sygnał zatrzymujący pracę silnika DC, oś Z
Y4	Dir_Z Sygnał zmiany kierunku obrotów silnika DC, oś Z

Y2	PWM_Y Sygnał sterujący typu PWM dla silnika DC, oś Y. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y5	Brake_Y Sygnał zatrzymujący pracę silnika DC, oś Y
Y6	Dir_Y Sygnał zmiany kierunku obrotów silnika DC, oś Y
Y0	PWM_X Sygnał sterujący typu PWM dla silnika DC, oś X. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y7	Brake_X Sygnał zatrzymujący pracę silnika DC, oś X
Y10	Dir_X Sygnał zmiany kierunku obrotów silnika DC, oś X

### 3.3 INTECO – Serwomechanizm (SERVO)



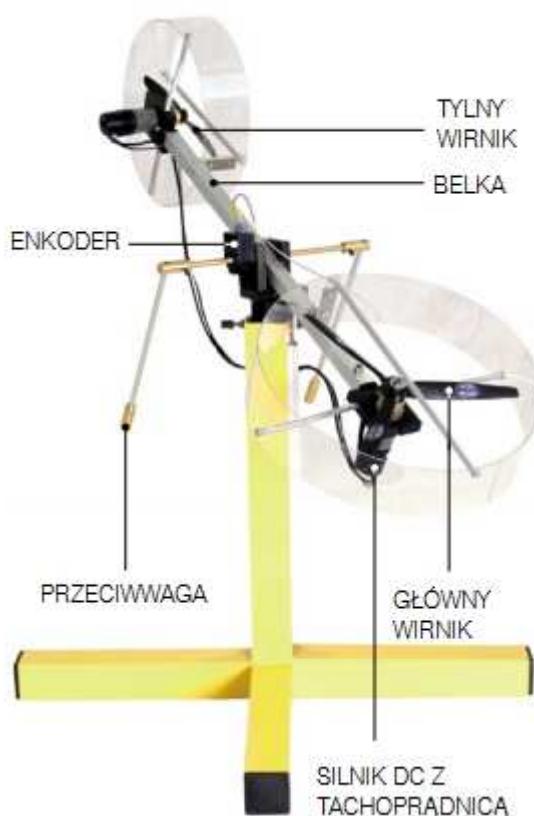
Rys. 3.8: Stanowisko laboratoryjne Modular Servo

Rozdzielcość fizyczna użytego enkodera inkrementalnego wynosi 1024 impulsy/obrót. Zakres wartości napięcia pojawiający się na wyprowadzeniach „Potentiometer” i „Tacho” wynosi  $\pm 10$  V. Sygnały analogowe przekazane są do sterownika przez moduł wejść analogowych FX5-4ADP. Możliwości regulacji w tym przypadku mogą obejmować pomiar prędkości obrotowej przy pomocy tachoprądnicy (generator – napięcie stałe na wyjściu rośnie proporcjonalnie do prędkości obrotowej - analogia dynamo w rowerze) i sterowanie prędkością obrotową silnika przez zadawanie wypełnienia sygnału PWM. Druga opcja sterowania to pomiar pozycji z enkodera inkrementalnego i sterowanie położenia silnika.

	WEJŚCIA CYFROWE
AIN1	Potentiometer Analogowy sygnał pomiarowy z potencjometru (zadajnik położenia, prędkości kątowej itp.).
AIN2	Tacho Analogowy sygnał pomiarowy prędkości obrotowej z tachoprądnicy.
X2	Therm Flaga limitu temperatury.
X1	EncB1 Enkoder inkrementalny, fala B, oś pozioma.
X0	EncA1 Enkoder inkrementalny, fala A, oś pozioma.

WYJŚCIA CYFROWE	
Y0	PWM Sygnał sterujący typu PWM dla silnika DC. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y1	Brake Sygnał zatrzymujący pracę silnika DC.
Y2	Dir0 Sygnał zmiany kierunku obrotów silnika DC.

### 3.4 INTECO – Helikopter (TRAS)



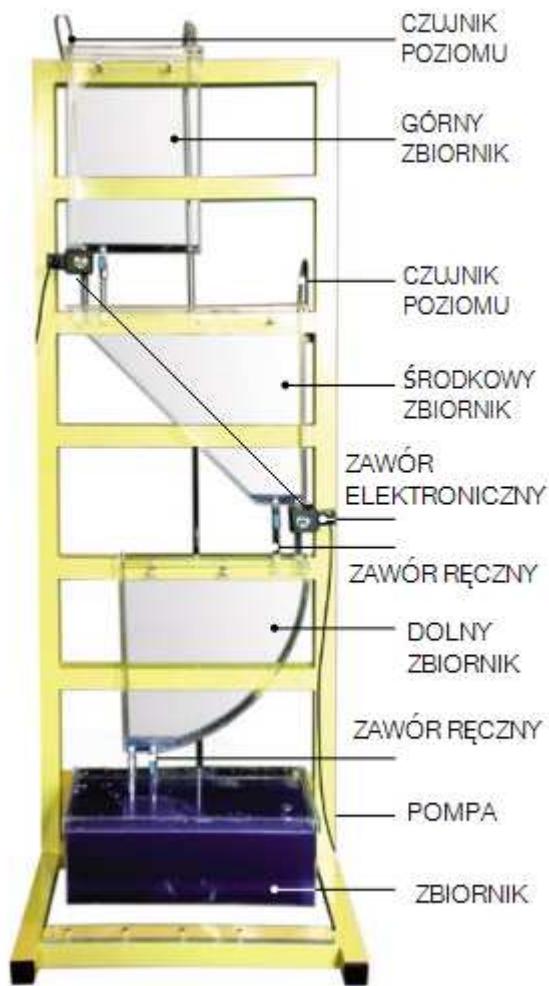
Rys. 3.9: Stanowisko laboratoryjne TRAS

Rozdzielczość fizyczna wszystkich użytych enkoderów inkrementalnych wynosi 1024 impulsy/obrót. Zakres wartości napięcia pojawiający się na wyprowadzeniach „Tacho\_Pt” i „Tacho\_Az” wynosi  $\pm 10$  V. Sygnały analogowe przekazane są do sterownika przez moduł wejść analogowych FX5-4ADP. Możliwości regulacji w tym przypadku obejmują pomiary położenia dwóch osi w przestrzeni i odpowiednie sterowanie silnikami przez zmianę wypełnienia PWM.

WEJŚCIA CYFROWE	

AIN1	Tacho+B1_Pt Analogowy sygnał pomiarowy z tachoprądnicy, śmiegle - oś pionowa (Pitch). Współczynnik przetwarzania tachoprądnicy wynosi 0.5 [V]/1000 [obr/min] przy założeniu, że napięcie mierzone jest bezpośrednio na tachoprądnicy.
AIN2	Tacho_Az Analogowy sygnał pomiarowy z tachoprądnicy, śmiegle - oś pozioma (Azimuth) . Współczynnik przetwarzania tachoprądnicy wynosi 0.5 [V]/1000 [obr/min] przy założeniu, że napięcie mierzone jest bezpośrednio na tachoprądnicy.
X0	EncA2_Pt Enkoder inkrementalny, fala A, oś pionowa
X2	EncB1_Az Enkoder inkrementalny, fala B, oś pozioma
X1	EncB2_Pt Enkoder inkrementalny, fala B, oś pionowa
X3	EncA1_Az Enkoder inkrementalny, fala A, oś pozioma
X4	Therm1_Az Flaga limitu temperatury, oś pozioma
X5	Therm0_Pt Flaga limitu temperatury, oś pionowa
	<b>WYJŚCIA CYFROWE</b>
Y2	Brake1_Az Sygnał zatrzymujący pracę silnika DC, oś pozioma
Y3	Dir1_Az Sygnał zmiany kierunku obrotów silnika DC, oś pozioma
Y0	PWM1_Az Sygnał sterujący typu PWM dla silnika DC, oś pozioma. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y1	PWM0_Pt Sygnał sterujący typu PWM dla silnika DC, oś pionowa. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y4	Brake0_Pt Sygnał zatrzymujący pracę silnika DC, oś pionowa. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y5	Dir0_Pt Sygnał zmiany kierunku obrotów silnika DC, oś pionowa

### 3.5 INTECO - Zbiorniki wodne (TANKS)



Rys. 3.10: Stanowisko laboratoryjne 3D-Tanks

Zgodnie z zaleceniami producenta zaworów proporcjonalnych, zalecana częstotliwość sygnału sterującego typu PWM (wyprowadzenia „Valve1”, „Valve2” i „Valve3”) powinna wynosić między 5 a 15 kHz. Do budowy czujnika poziomu cieczy w systemie 3-Tanks (wyjścia „Level1”, „Level2” i „Level3”) wykorzystano czujnik ciśnienia wraz z półprzewodnikowym układem do konwersji U/f (napięcie na częstotliwość). Zakres zmian częstotliwości to przedział od 80 kHz dla wartości poziomu cieczy 0.0 cm do około 180 kHz dla wartości poziomu cieczy 25 cm. Warto zweryfikować czy zakres ten jest prawidłowy.

Stanowisko pozwala na regulację poziomu wody w trzech zbiornikach, gdzie każdy jest wyposażony w jeden czujnik poziomu i jeden zawór. Każdy zawór może być sterowany wypełnieniem sygnału PWM. Dodatkowo pompa, który dostarcza wodę do górnego zbiornika jest sterowana wypełnieniem sygnału PWM.

	WEJŚCIA CYFROWE
X0	Level1 Cyfrowy sygnał pomiarowy z czujnika poziomu cieczy, typu częstotliwościowego, zbiornik górny.

X1	Level2 Cyfrowy sygnał pomiarowy z czujnika poziomu cieczy, typu częstotliwościowego, zbiornik środkowy.
X2	Level3 Cyfrowy sygnał pomiarowy z czujnika poziomu cieczy, typu częstotliwościowego, zbiornik dolny.
<b>WYJŚCIA CYFROWE</b>	
Y0	Pump Sygnał sterujący typu PWM dla pompy wody. Zalecana częstotliwość tego sygnału mieści się w przedziale (5-15) kHz.
Y1	Valve1 Sygnał sterujący typu PWM dla zaworu proporcjonalnego, zbiornik górny.
Y2	Valve2 Sygnał sterujący typu PWM dla zaworu proporcjonalnego, zbiornik środkowy.
Y3	Valve3 Sygnał sterujący typu PWM dla zaworu proporcjonalnego, zbiornik dolny.

### 3.6 Podłączenie zestawów INTECO

Każdy z interfejsów PLC musi zostać prawidłowo podłączony do interfejsu mocy wybranego zestawu firmy INTECO. Połączenie realizowane jest poprzez płaskie taśmy 20- lub 40- przewodowe z wykorzystaniem złącz typu IDC znajdujących się na tylnych panelach interfejsów PLC oraz panelach frontowych poszczególnych sterowników mocy (szczegółowy opis można znaleźć w dokumentacji technicznej dotyczącej danego zestawu). Przy realizacji połączenia należy zwrócić uwagę na oznaczenia poszczególnych gniazd IDC oraz dostarczonych płaskich kabli. Szczególnie dotyczy to przypadków, w których dokonuje się połączenia zarówno sygnałów cyfrowych jak i analogowych. Pomyłka w połączeniu może doprowadzić do uszkodzenia interfejsu PLC lub kanałów I/O sterownika mocy. Na poniższych rysunkach przedstawiono wygląd tylnych paneli interfejsów PLC dla poszczególnych zestawów. Szczególną uwagę należy zwrócić na ostatni rysunek, który przedstawia tylny panel dla zestawu TRAS i SERVO. Złącze 40- końcówkowe jest podzielone na dwie części. Pierwsza z nich (wyprowadzenia 1-20) stanowi interfejs analogowy, natomiast druga (wyprowadzenia 21-40) interfejs cyfrowy.

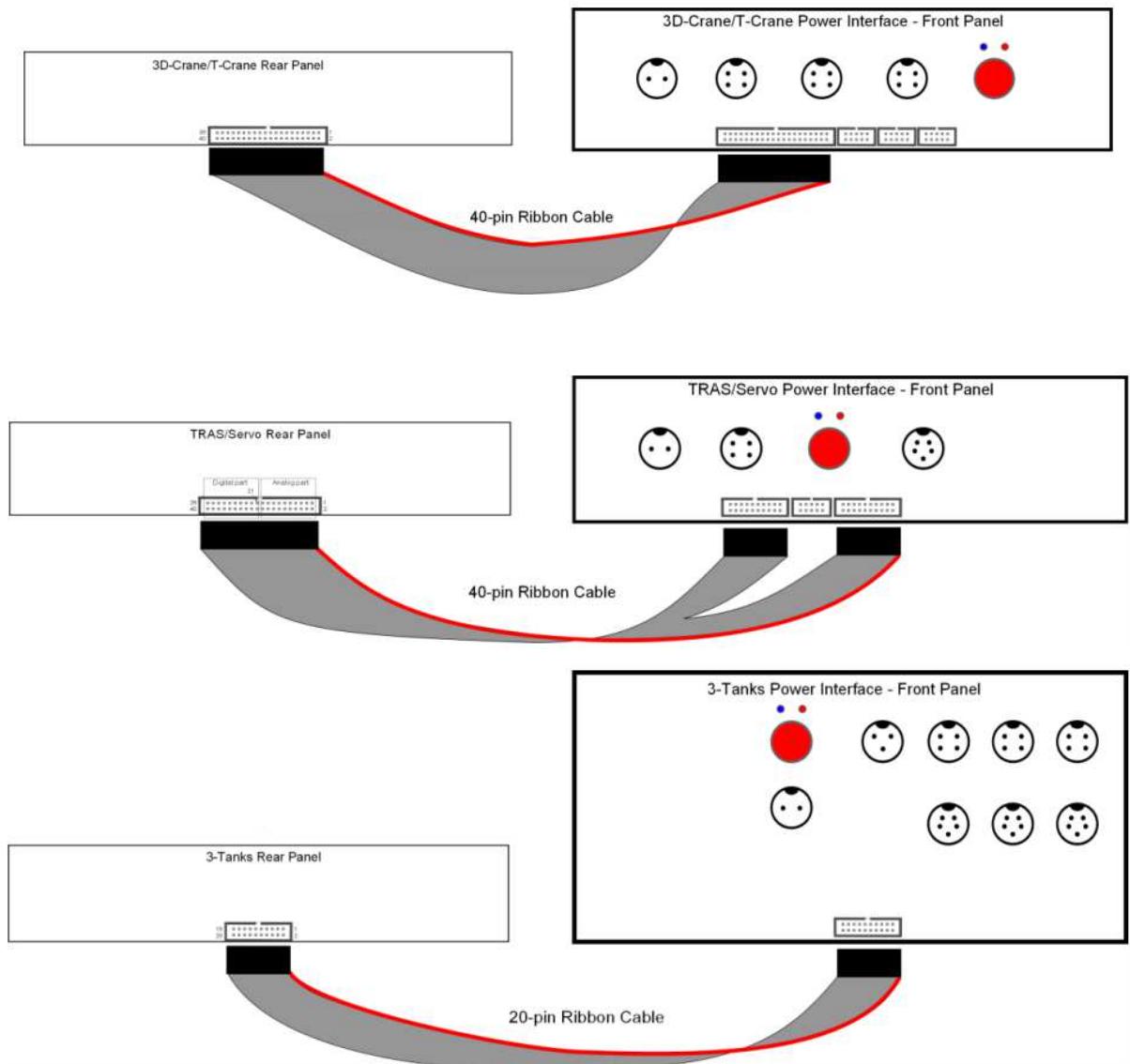
Przy łączeniu należy zwrócić uwagę na czerwony marker płaskiej taśmy łączeniowej, który oznacza wyprowadzenie nr 1. Jest to szczególnie istotne dla systemów TRAS i SERVO, gdzie płaska 40-kablowa taśma rozdziela się na dwie 20-kablowe części (analogową i cyfrową).

### 3.7 Podłączenie zestawów INTECO

Każdy z interfejsów PLC musi zostać prawidłowo podłączony do interfejsu mocy wybranego zestawu firmy Inteco. Połączenie realizowane jest poprzez płaskie taśmy 20- lub 40- przewodowe z wykorzystaniem złącz typu IDC znajdujących się na tylnych panelach interfejsów PLC oraz panelach frontowych poszczególnych sterowników mocy (szczegółowy opis można znaleźć w dokumentacji technicznej dotyczącej danego zestawu). Przy realizacji połączenia należy zwrócić uwagę na oznaczenia poszczególnych gniazd IDC oraz dostarczonych płaskich kabli. Szczególnie dotyczy to przypadków, w których dokonuje się połączenia zarówno sygnałów cyfrowych jak i analogowych. Pomyłka w połączeniu może doprowadzić do uszkodzenia interfejsu PLC lub kanałów I/O sterownika mocy. Na poniższych rysunkach przedstawiono wygląd tylnych paneli interfejsów PLC dla

poszczególnych zestawów. Szczególną uwagę należy zwrócić na ostatni rysunek, który przedstawia tylny panel dla zestawu TRAS i SERVO. Złącze 40-końcowkowe jest podzielone na dwie części. Pierwsza z nich (wyprowadzenia 1-20) stanowi interfejs analogowy, natomiast druga (wyprowadzenia 21-40) interfejs cyfrowy.

Przy łączeniu należy zwrócić uwagę na czerwony marker płaskiej taśmy łączeniowej, który oznacza wyprowadzenie nr 1. Jest to szczególnie istotne dla systemów TRAS i SERVO, gdzie płaska 40-kablowa taśma rozdziela się na dwie 20-kablowe części (analogową i cyfrową).



## **4 Programowanie w systemie OVATION**

### **4.1 Wprowadzenie**

Celem niniejszego rozdziału jest zapoznanie studentów z narzędziami systemu OVATION. Po krótkiej prezentacji narzędzi przedstawione są przykłady praktyczne, pokazujące krok po kroku sposób użycia aplikacji. Zaznajomienie się z materiałem z niniejszego rozdziału jest warunkiem koniecznym do wykonania zadań laboratoryjnych w systemie OVATION.

### **4.2 Podstawowe aplikacje systemu OVATION**

W systemie OVATION można wyróżnić dwa typy narzędzi programistycznych. Programy służące do projektowania oraz programy służące do pracy w trybie online z procesem. Do podstawowych programów do projektowania można zaliczyć:

- OVATION Developer Studio
- Control Builder
- Graphic Builder.

Do programów do pracy online można zaliczyć:

- Signal Diagram
- Trend
- Graphics
- Point Information
- Alarms
- Controller Diagnostics
- System Viewer

oraz wiele innych aplikacji do specjalizowanych zadań, które wykraczają poza ramy przedmiotu.

#### **4.2.1 Narzędzia projektowe**

Poniżej krótko przedstawiono podstawowe narzędzi do projektowania aplikacji w systemie OVATION. Funkcjonalność oprogramowania oraz sposób nawigacji przedstawiono (w sposób krok po kroku) na przykładzie mini projektu w podrozdziale 4.3.

##### **OVATION Developer studio**

OVATION Developer studio jest podstawową aplikacją systemu OVATION. Poprzez aplikacje można dokonać konfiguracji całego projektu począwszy od konfiguracji hardware, poprzez konfigurację kontrolerów, punkty procesowe, logiki i grafiki. W ramach przedmiotu DCS i SCADA środowisko nie będzie używane tak szeroko. Ograniczymy się do tworzenia

punktów, budowy logik (binarnych i ciągłych) oraz programowania prostych grafik. W tym celu istotne będzie zaznajomienie się z aplikacjami: Control Builder oraz Graphic Builder.

### **Control Builder**

Control Builder służy do budowania programu logik. Logiki budowane są w systemie OVATION przy pomocy języka FBD.

### **Graphic Builder**

Control Builder służy do budowania grafik procesowych. Grafiki budowane są w systemie OVATION w technice drag and drop wspomaganej możliwością edycji skryptów.

## **4.2.2 Narzędzia do pracy online**

Poniżej krótko przedstawiono podstawowe narzędzi do pracy on-line w systemie OVATION. Funkcjonalność oraz sposób nawigacji przedstawiono (w sposób krok po kroku) na przykładzie mini projektu w podrozdziale 4.3.

### **Signal Diagram**

Aplikacja prezentująca logikę w identyczny sposób jak Control Builder z tą różnicą, że wyświetlane są aktualne wartości liczone przez algorytmy. Działania aplikacji Signal Diagram w zamyśle odpowiada trybowi Monitoring dostępnemu w narzędziach projektowych sterowników PLC.

### **Trend**

Aplikacja wyświetlająca wartości bieżące punktów systemowych (wyjścia i wejścia do algorytmów) w postaci trendów lub w postaci trendu tabularycznego. Trend pozwala na zapis wartości punktów do pliku tekstowego.

### **Graphics**

Aplikacja prezentująca grafikę w identyczny sposób jak ona została zaprojektowana w Graphic Builderze z tą różnicą, ze wyświetlane są aktualne wartości punktów procesowych oraz że z grafiki można sterować procesem.

### **Point Information**

Aplikacja wyświetlająca informacje związane z dowolnym punktem w systemie, takie jak wartość, limity, pochodzenie, jakość, stan.

### **Alarm**

Aplikacja wyświetlająca alarmy występujące w systemie.

### **Controller Diagnostics**

Aplikacja monitorująca stan pracy kontrolerów oraz dostarczająca informacje o zasobach, czasach obliczeń i liczbie punktów.

### **System Viewer**

Aplikacja monitorująca stan pracy systemu: serwery, stacje procesowe i kontrolery.

## **4.3 Przykład 1 – logika sterowania binarna**

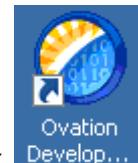
Przykład 1 pokazuje krok po korku jak zrealizować w systemie OVATION logikę która ma zapamiętać że wystąpiło zdarzenie B1 lub B2 i pamiętać te informację dopóki nie wystąpiło zdarzenie B3. Przedstawione zadanie jest typowym zadaniem z wykorzystaniem bramki OR i przerzutnika.

### **4.3.1 Etap projektowania logiki**

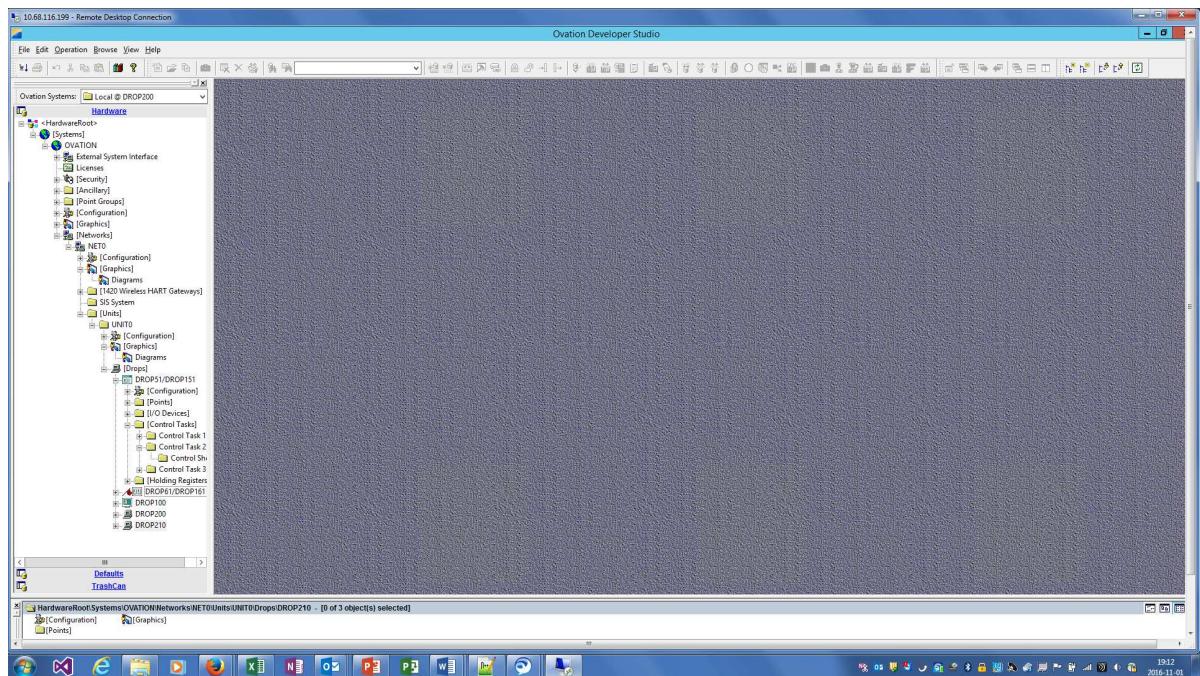
W celu realizacji zadania należy:

- otworzyć OVATION Developer Studio;
- utworzyć w Control Builder logikę;
- załadować logikę na kontroler;
- sprawdzić działanie za pomocą narzędzi online.

### **OVATION Developer studio**



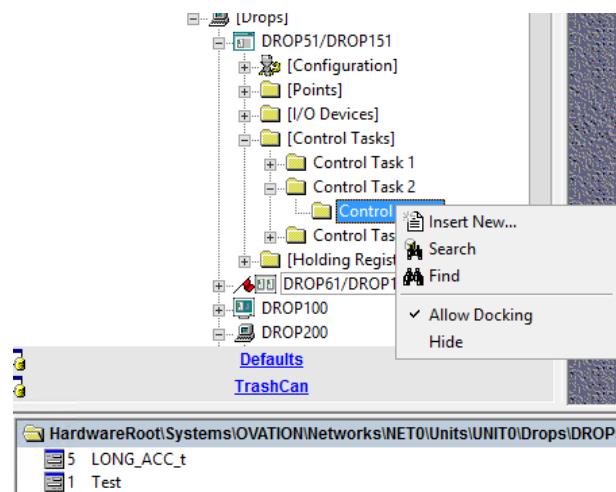
OVATION Developer studio uruchamiany jest poprzez skrót.



Rys. 4.1 OVATOIN Developer Studio.

**Na Błąd! Nie można odnaleźć źródła odwołania.** na drzewie katalogów (po lewej stronie) przedstawiona jest struktura w jakiej rozwijane są aplikacje w systemie DCS. Z punktu widzenia programowania logik najważniejszy jest poziom, gdzie umieszczone są DROPy. Pod nazwą DROP ukryte są stacje operatorskie, serwery i kontrolery. Kontrolery umieszczone są zazwyczaj pod podwójną nazwą DROP51/DROP151 wskazującą, że maszyny pracują w konfiguracji redundantnej.

Logiki tworzone są na poziomie „Control Sheet” (Rys. 4.2).

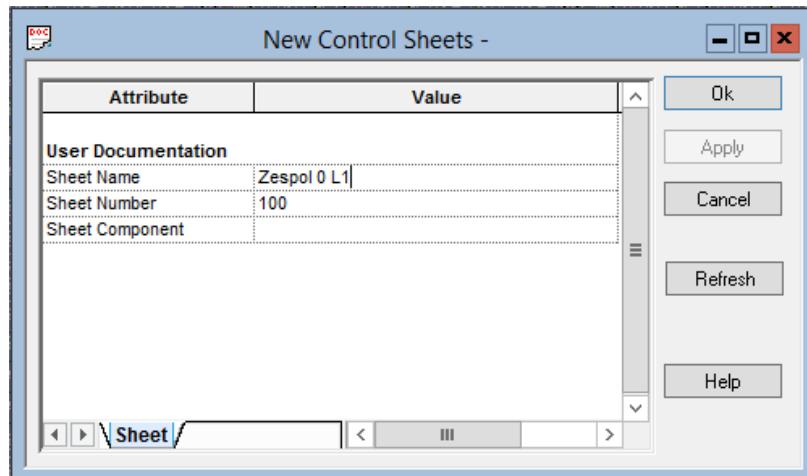


Rys. 4.2: Utworzenie nowej logiki – wywołanie aplikacji Control Builder.

Logiki mogą być utworzone w Control Task 1 (100ms), Control Task 2 (1000ms) i Control Task 3,4,5 konfigurowanych od 10ms do 30sek wg. potrzeb w projekcie. Opcja „Insert New” powoduje utworzenie nowej logiki i automatyczne wywołanie Control Buildera.

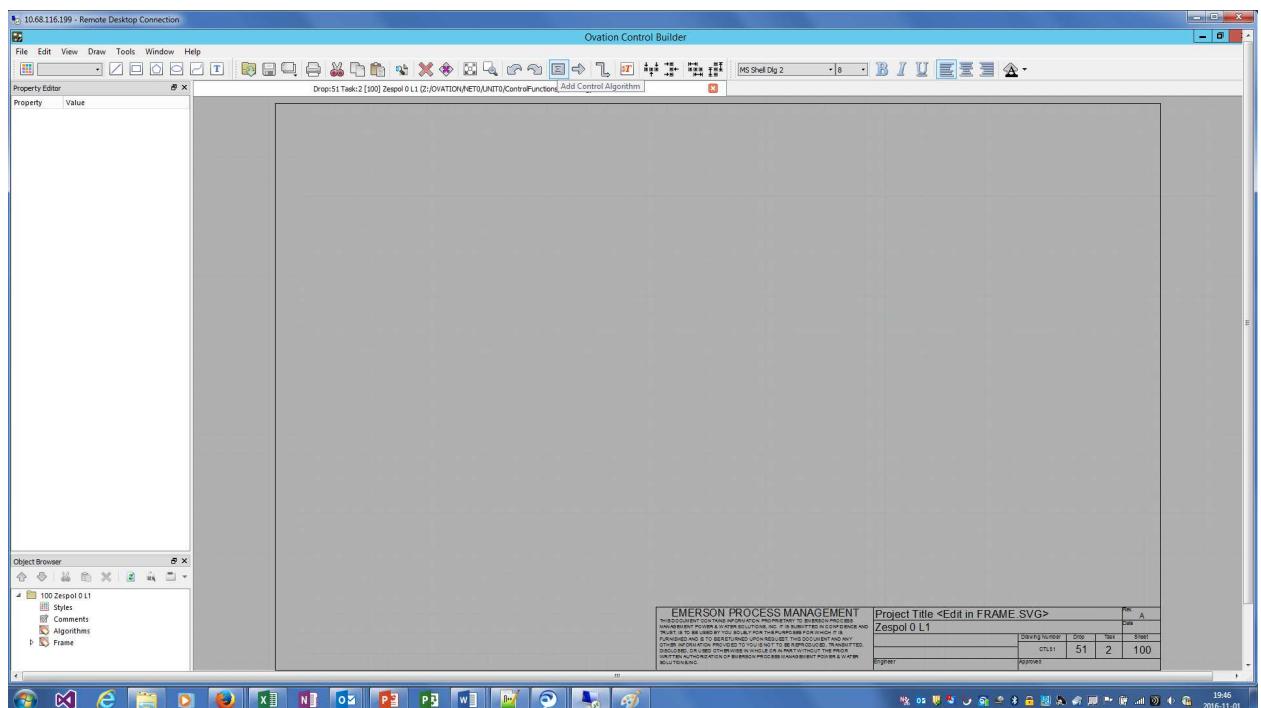
## Control Builder

Opcja „Insert New” powoduje utworzenie nowego logiki.



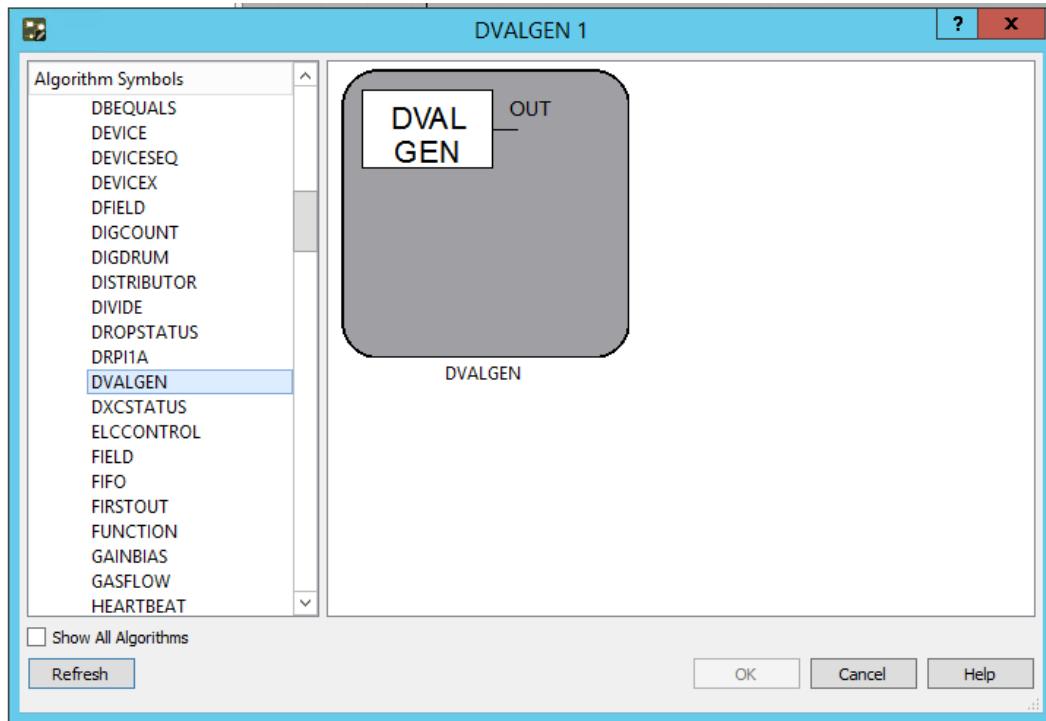
Rys. 4.3: Atrybuty nowej logiki.

**Uwaga:** w celu łatwej identyfikacji proponuje się utrzymywanie pewnego porządku numeracji. Zespół 1 powinien utworzyć logikę pod numerem 110, zespół 2 pod numerem 120 a zespół X pod numerem 1X0 (Rys. 4.3).



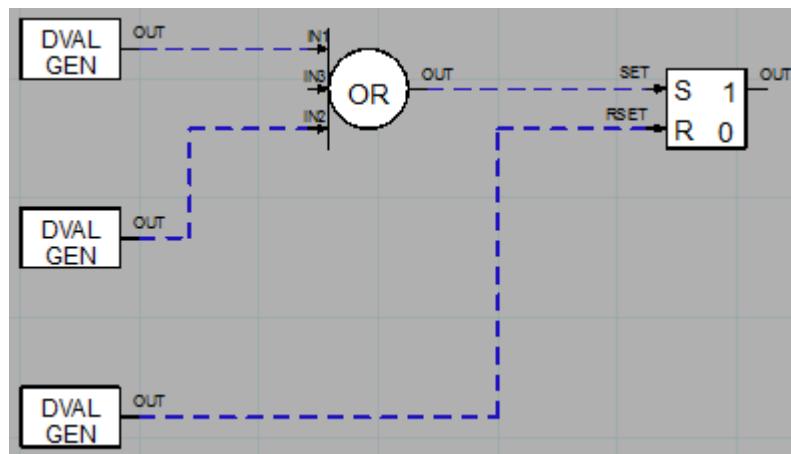
Rys. 4.4: Nowa logika.

Wraz z utworzeniem nowej logiki Rys. 4.4 automatycznie otwierane jest środowisko Control Buildera. Ikona “Add Control Algorithm” lub skrót “Ctrl+a” otwiera listę dostępnych algorytmów (Rys. 4.5).



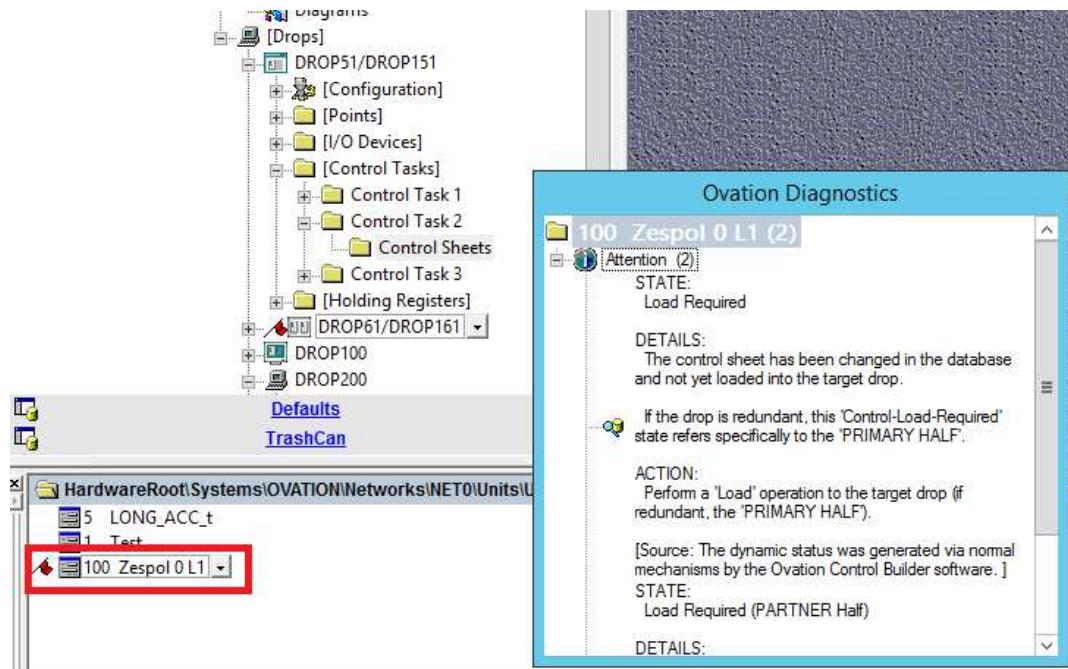
Rys. 4.5: List algorytmów.

Algorytm DVALGEN, dostępny w grupie STANDARD pozwala ustawać wartość punktu binarnego. Operacje logiczne na punktach binarnych wykonywane są poprzez algorytmy z grupy FAST BOOLEAN. Zgodnie z treścią zadania zbudowano logikę jak na Rys. 4.6.



Rys. 4.6 Logika realizująca zadanie.

Po zapisaniu logiki możliwe jest załadowanie jej na kontroler. Czerwona flaga sygnalizuje, że powstała nowa logika i wymagany jest Load (Rys. 4.7).



Rys. 4.7: Nowa logika gotowa do ladowania.

### Ladowanie logiki na kontroler

W celu załadowania kontrolera należy rozwinąć menu prawego przycisku myszy i wybrać operację LOAD.

**Uwaga:** wszystkie grupy pracują na wspólnym kontrolerze, dlatego operację ładowania należy skoordynować z innymi grupami, w celu uniknięcia równoczesnego ładowania.



Rys. 4.8: Operacja ładowania aplikacji na kontroler.

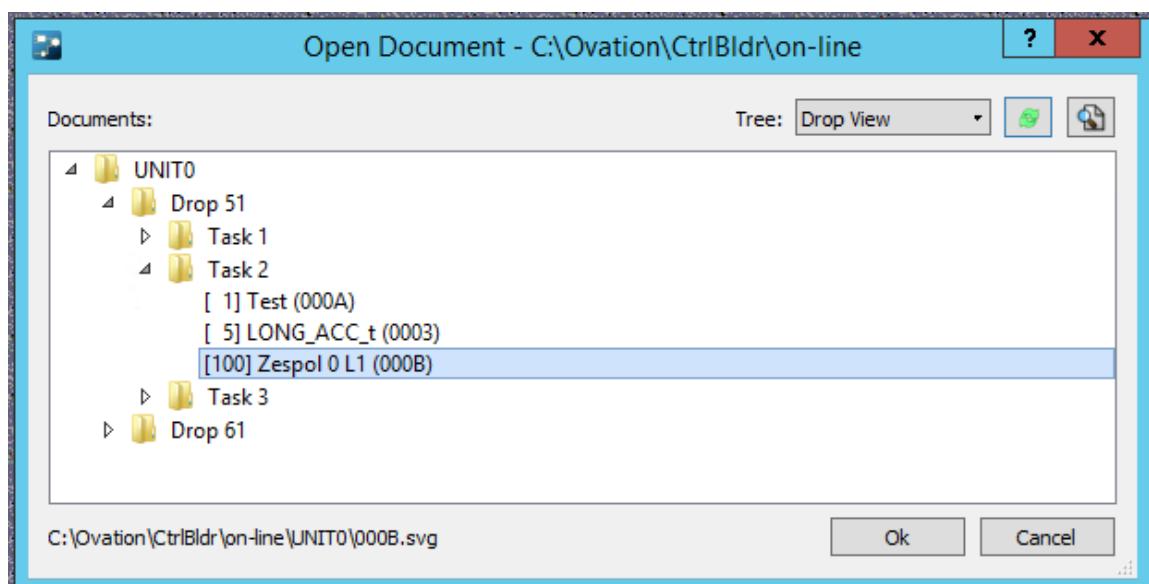
W pierwszej kolejności powinien być ładowany kontroler, który właśnie pracuje (Rys. 4.8) – Primary Drop nie oznacza że ten kontroler właśnie pracuje. Status kontrolerów można sprawdzić w programie Diagnostics (opisanym w następnym podrozdziale).

Po załadowaniu kontrolera możliwe jest sprawdzenie działania aplikacji przy użyciu narzędzi do pracy online.

#### 4.3.2 Wykorzystanie narzędzi do pracy online

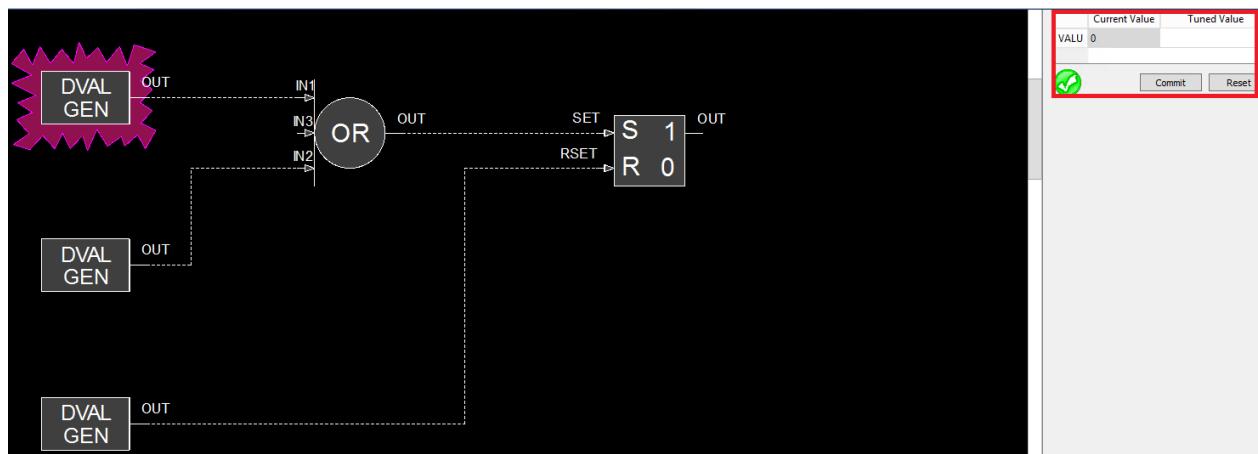
##### Signal Diagram

W celu wyświetlenia działającej logiki należy wywołać program „Signal Diagram” (w Okna startu Windows) i odnaleźć stosowną logikę na drzewie projektowym (Rys. 4.9).



Rys. 4.9: Widok logik w programie Signal Diagram.

Przycisk OK wyświetla wybraną logikę.



Rys. 4.10: Logika w programie Signal Diagram.

W prawym górnym oknie możliwe jest ustawianie wartości algorytmu DVALGEN i sprawdzanie działania logiki (Rys. 4.10).

## Trend i Point Information

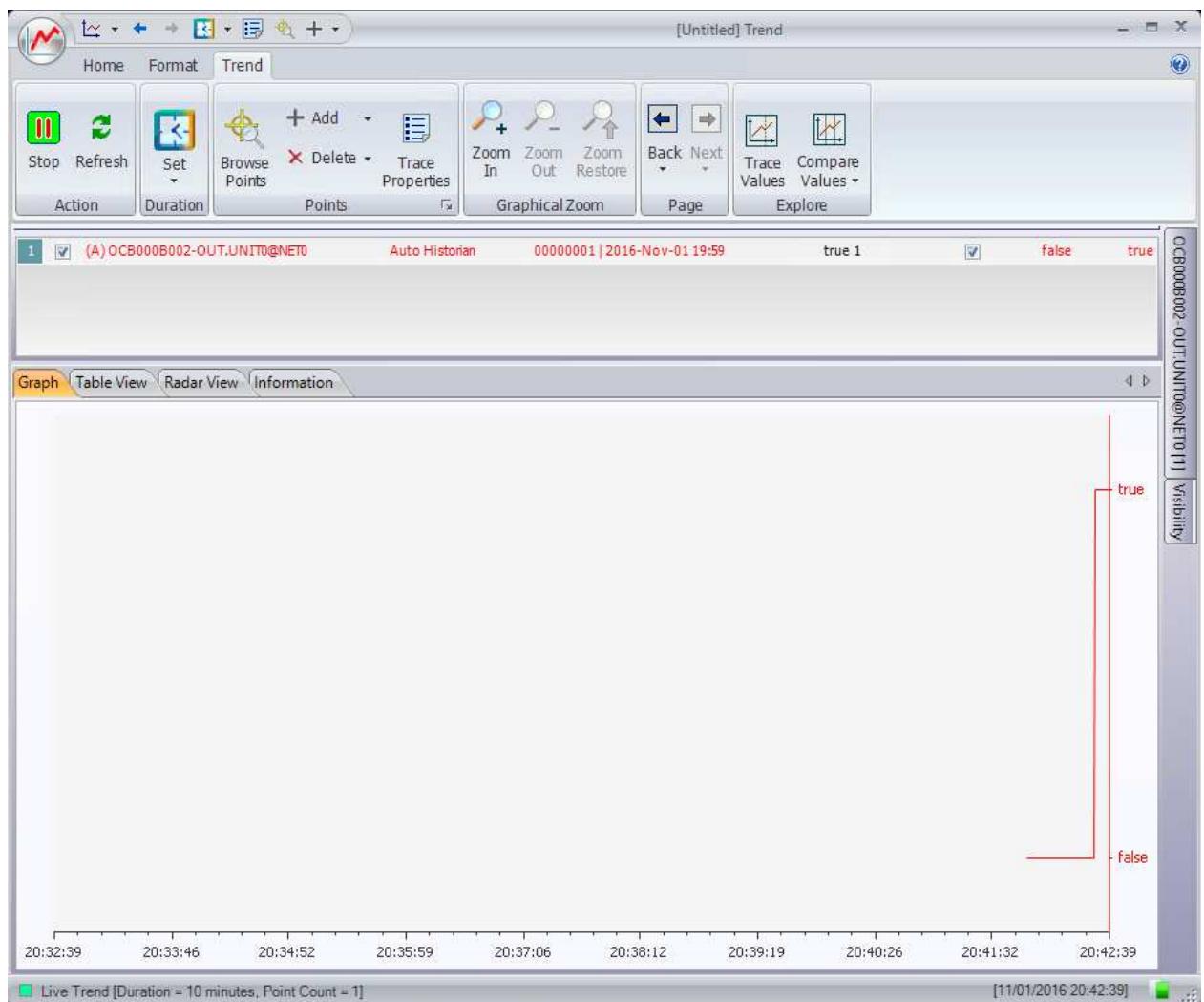
Z okna Signal Diagram można bezpośrednio wywołać aplikację Trend i Point Information. W tym celu należy na liście „Algorithm Summary” wybrać punkt i rozwinąć menu prawego klawisza myszy (Rys. 4.11).

Name	Parameter Description	Point	Point Description	Type	Bottom	Top	
IN1	Input 1	OCB000B002-OUT.UNIT0@NET0	00000001   2016-Nov-01 19:59	LD	false	true	false (0)
IN	OCB000B002-OUT.UNIT0@NET0	-OUT.UNIT0@NET0	00000001   2016-Nov-01 19:59	LD	false	true	false (0)

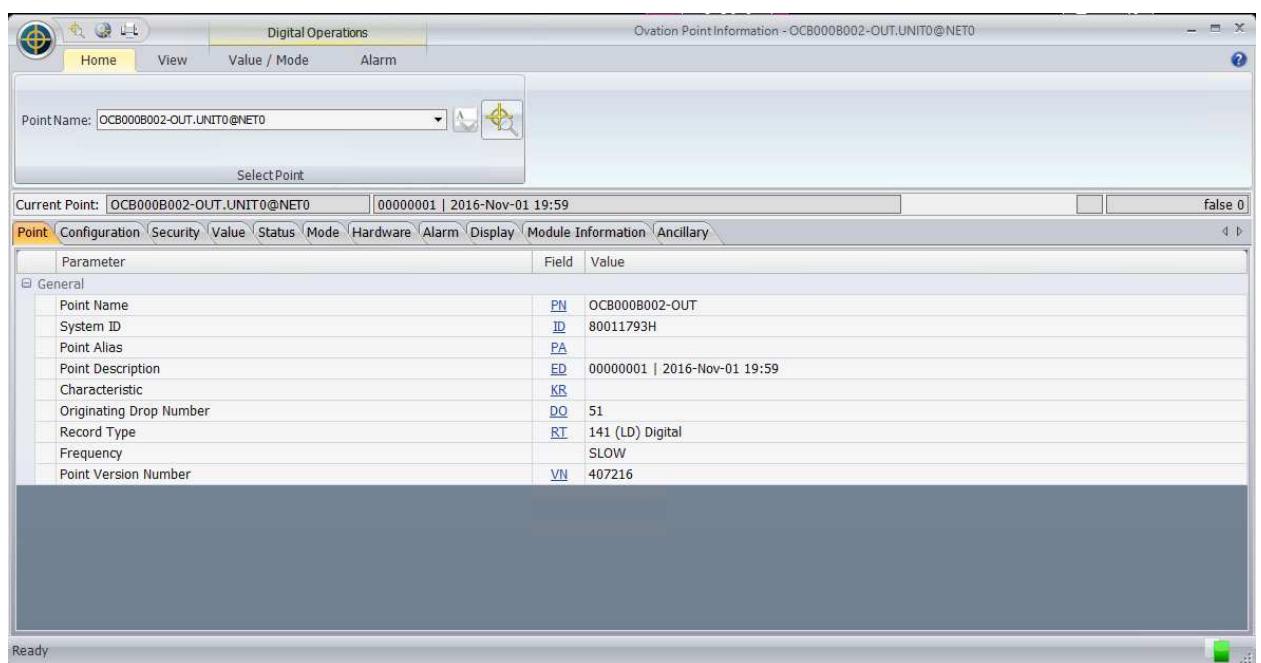
The context menu, which is open over the second row of the table, contains two items: "Point Info" and "Trend".

Rys. 4.11: Wywołanie programów Trend i Point Information z okna Signal Diagram.

Point Info wyświetli informację o wybranym punkcie (Rys. 4.13) a Trend wyświetli trend wartości punktu (Rys. 4.12).



Rys. 4.12: Okno programu Trend.

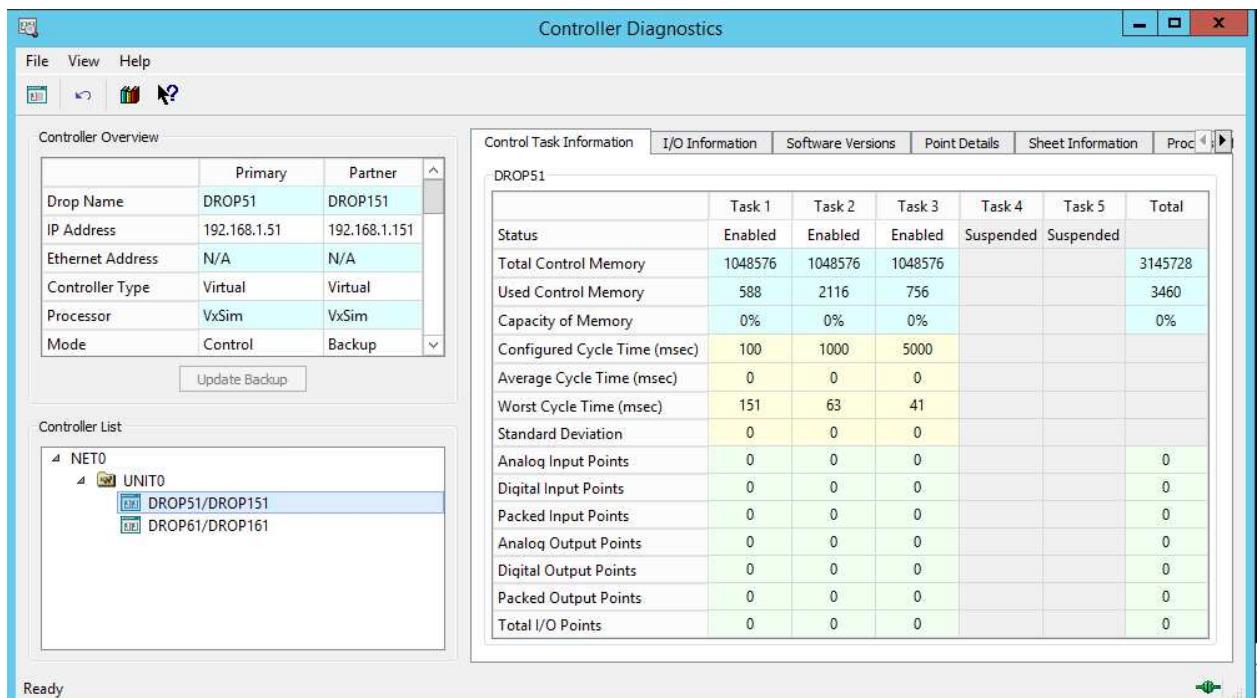


Rys. 4.13: Okno programu Point Information.

Te same aplikacje mogą być wywołanie z okna systemowego Windows ale wówczas wywołanie takie jest bez kontekstu i należy ręcznie podać nazwę punktu, który chce się obserwować.

### Controller Diagnostics

Aplikacja może być wywołana poleceniem „Controller Diagnostics” i wyświetla informacje związane z kontrolerami (Rys. 4.14).



Rys. 4.14: Okno programu Controller Diagnostics.

## 4.4 Przykład 2 – logika sterowania ciągłego

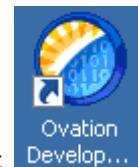
Przykład 2 pokazuje krok po korku jak zrealizować w systemie OVATION logikę symulującą sterowanie obiektem o transmitancji  $1/(Ts + 30)$ .

### 4.4.1 Etap projektowania logiki

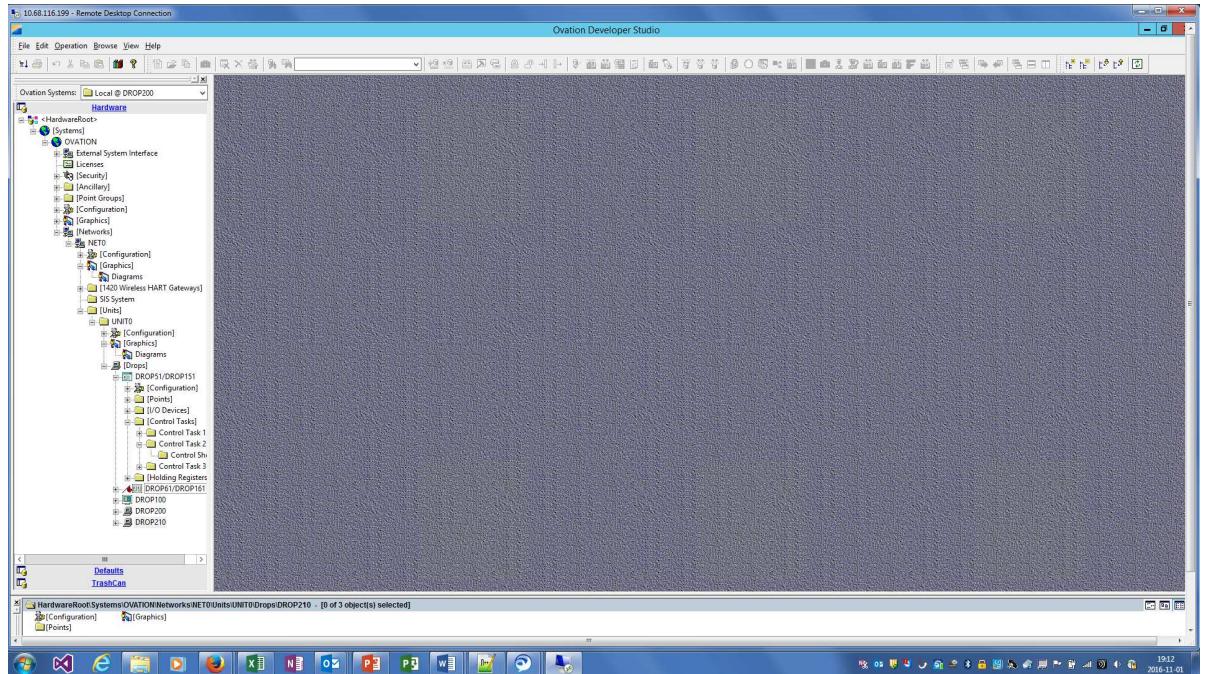
W celu realizacji zadania należy:

- otworzyć OVATION Developer Studio;
- utworzyć w Control Builder logikę;
- załadować logikę na kontroler;
- sprawdzić działanie za pomocą narzędzi online.

# OVATION Developer studio



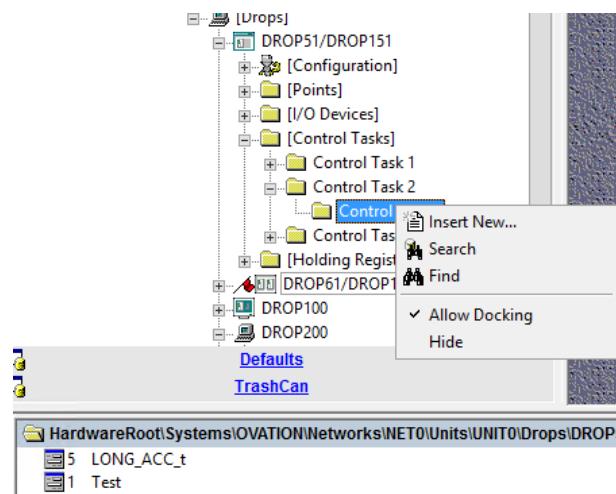
OVATION Developer studio uruchamiany jest poprzez [Develop...](#) skrót.



Rys. 4.15: OVATOIN Developer Studio.

**Na Błęd! Nie można odnaleźć źródła odwołania.** na drzewie katalogów (po lewej stronie) przedstawiona jest struktura w jakiej rozwijane są aplikacje w systemie DCS. Z punktu widzenia programowania logik najważniejszy jest poziom, gdzie umieszczone są DROPy. Pod nazwą DROP ukryte są stacje operatorskie, serwery i kontrolery. Kontrolery umieszczone są zazwyczaj pod podwójną nazwą DROP51/DROP151 wskazującą, że maszyny pracują w konfiguracji redundantnej.

Logiki tworzone są na poziomie „Control Sheet” (Rys. 4.2).

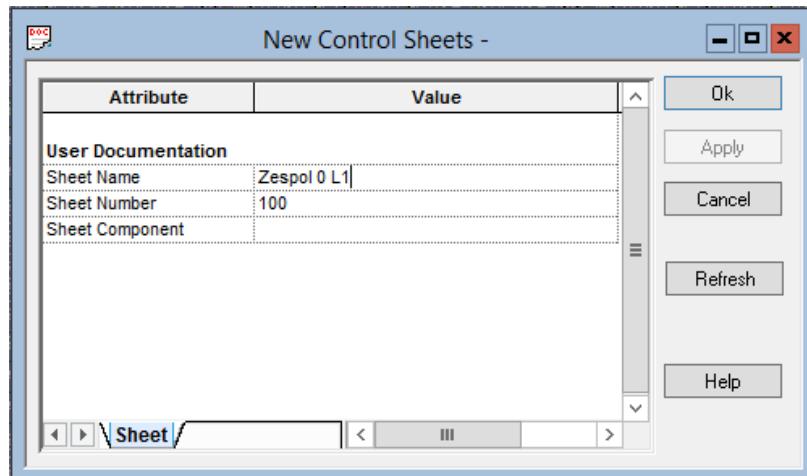


Rys. 4.16: Utworzenie nowej logiki – wywołanie aplikacji Control Builder.

Logiki mogą być utworzone w Control Task 1 (100ms), Control Task 2 (1000ms) i Control Task 3,4,5 konfigurowanych od 10ms do 30sek wg. potrzeb w projekcie. Opcja „Insert New” powoduje utworzenie nowej logiki i automatyczne wywołanie Control Buildera.

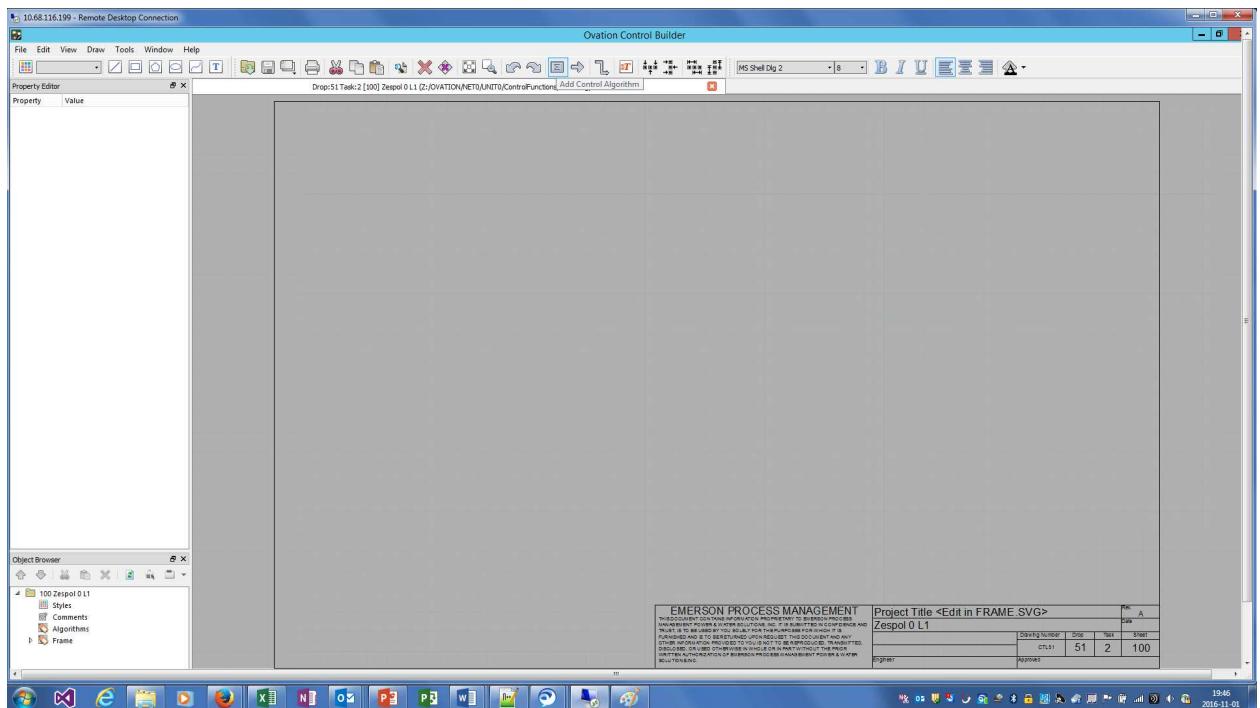
### Control Builder

Opcja „Insert New” powoduje utworzenie nowego logiki.



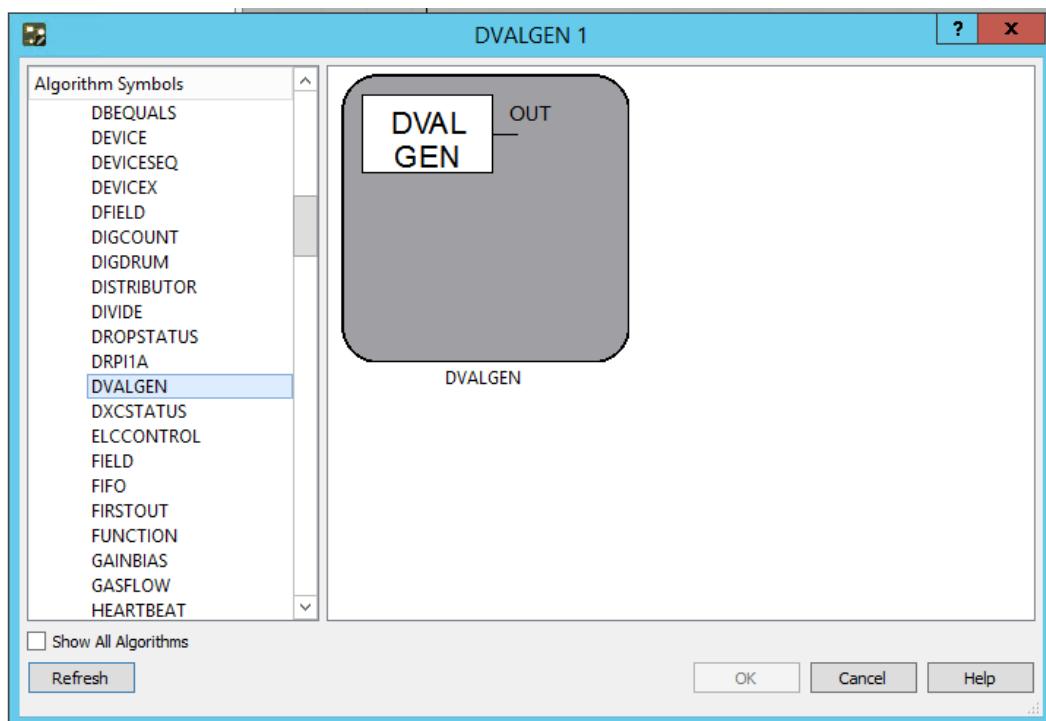
Rys. 4.17: Atrybuty nowej logiki.

**Uwaga:** w celu łatwej identyfikacji proponuje się utrzymywanie pewnego porządku numeracji. Zespół 1 powinien utworzyć logikę pod numerem 210, zespół 2 pod numerem 220 a zespół X pod numerem 2X0 (Rys. 4.3).



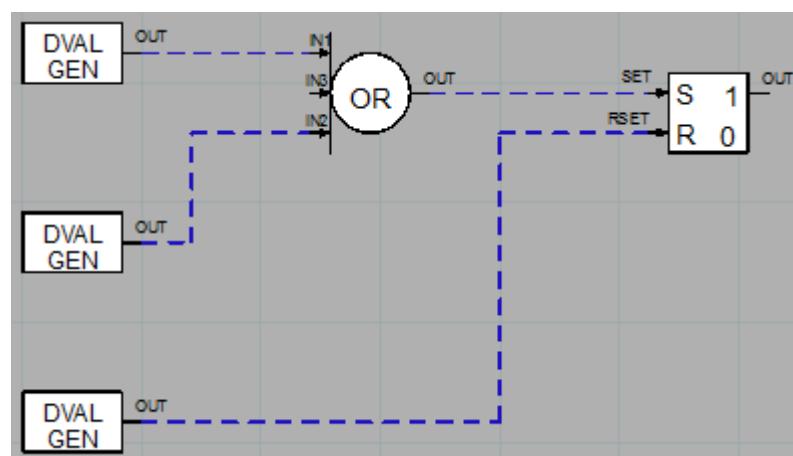
Rys. 4.18: Nowa logika.

Wraz z utworzeniem nowej logiki Rys. 4.4 automatycznie otwierane jest środowisko Control Buildera. Ikona “Add Control Algorithm” lub skrót “Ctrl+a” otwiera listę dostępnych algorytmów (Rys. 4.5).



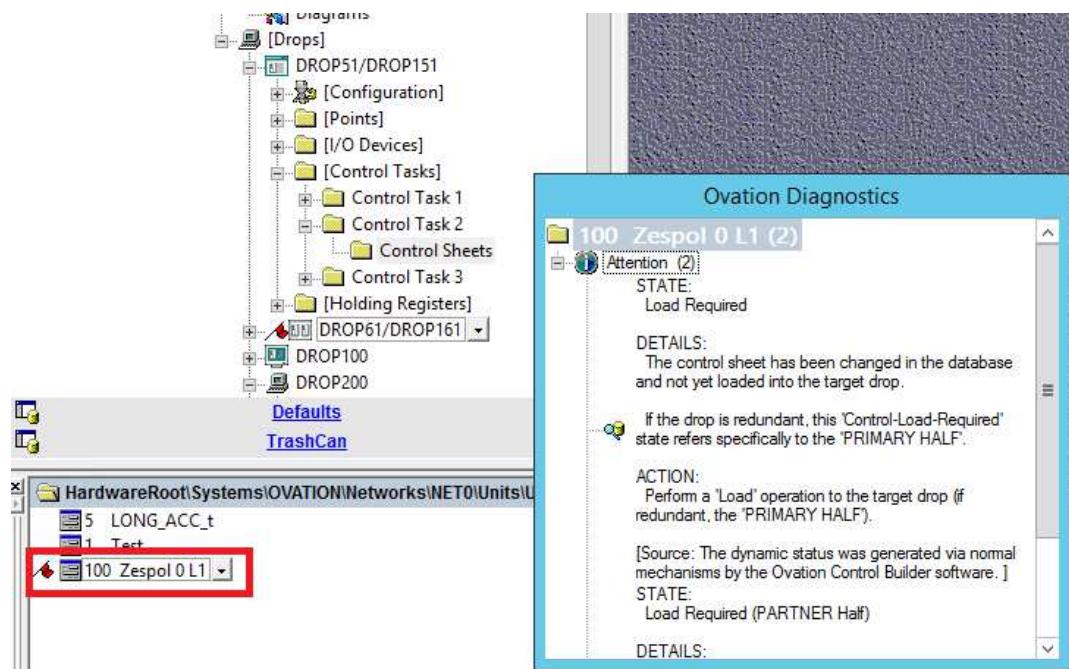
Rys. 4.19: List algorytmów.

Algorytm DVALGEN, dostępny w grupie STANDARD pozwala ustawać wartość punktu binarnego. Operacje logiczne na punktach binarnych wykonywane są poprzez algorytmy z grupy FAST BOOLEAN. Zgodnie z treścią zadania zbudowano logikę jak na Rys. 4.6.



Rys. 4.20: Logika realizująca zadanie.

Po zapisaniu logiki możliwe jest załadowanie jej na kontroler. Czerwona flaga sygnalizuje, że powstała nowa logika i wymagany jest Load (Rys. 4.7).

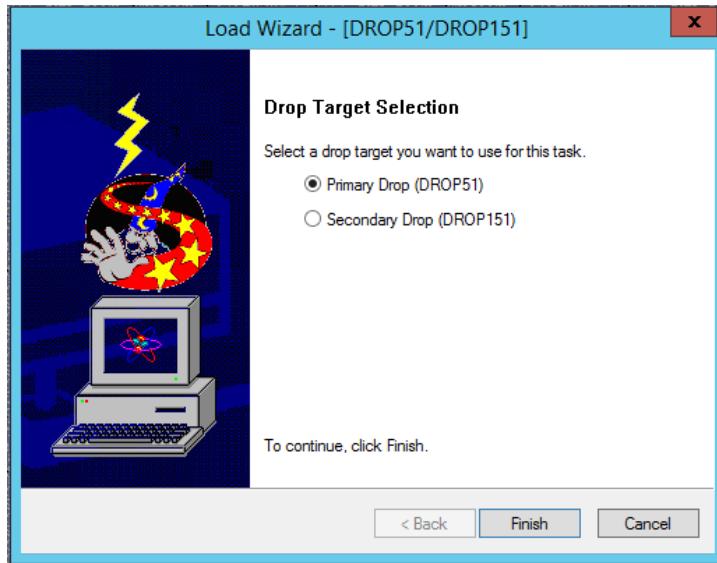


Rys. 4.21: Nowa logika gotowa do ładowania.

### Ladowanie logiki na kontroler

W celu załadowania kontrolera należy rozwinać menu prawego przycisku myszy i wybrać operację LOAD.

**Uwaga:** wszystkie grupy pracują na wspólnym kontrolerze, dlatego operację ładowania należy skoordynować z innymi grupami, w celu uniknięcia równoczesnego ładowania.



Rys. 4.22: Operacja ładowania aplikacji na kontroler.

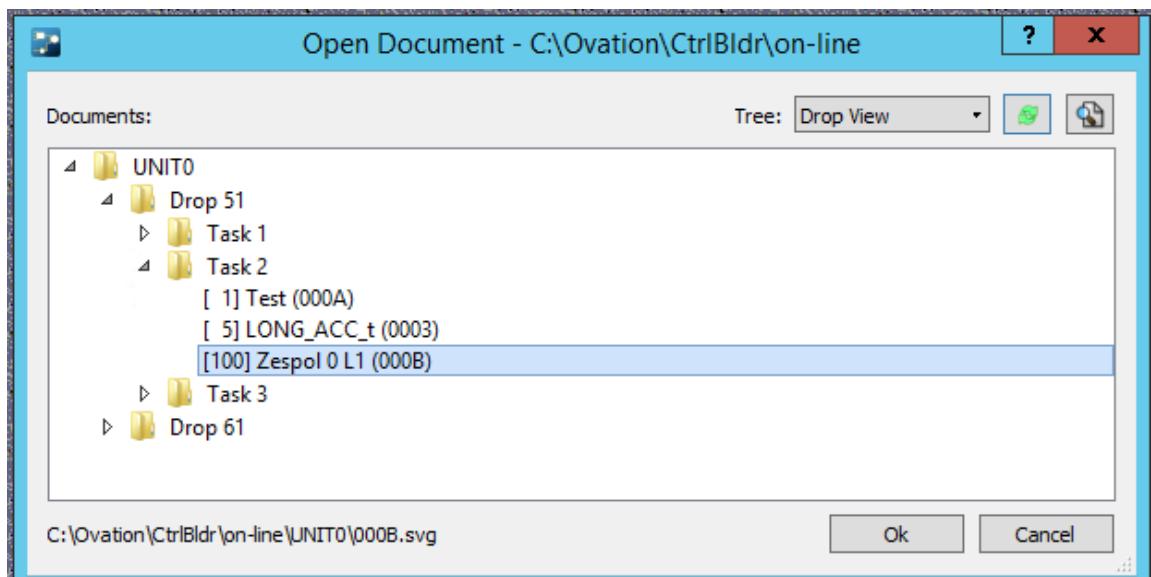
W pierwszej kolejności powinien być ładowany kontroler, który właśnie pracuje (Rys. 4.8) – Primary Drop nie oznacza że ten kontroler właśnie pracuje. Status kontrolerów można sprawdzić w programie Diagnostics (opisanym w następnym podrozdziale).

Po załadowaniu kontrolera możliwe jest sprawdzenie działania aplikacji przy użyciu narzędzi do pracy online.

#### 4.4.2 Wykorzystanie narzędzi do pracy online

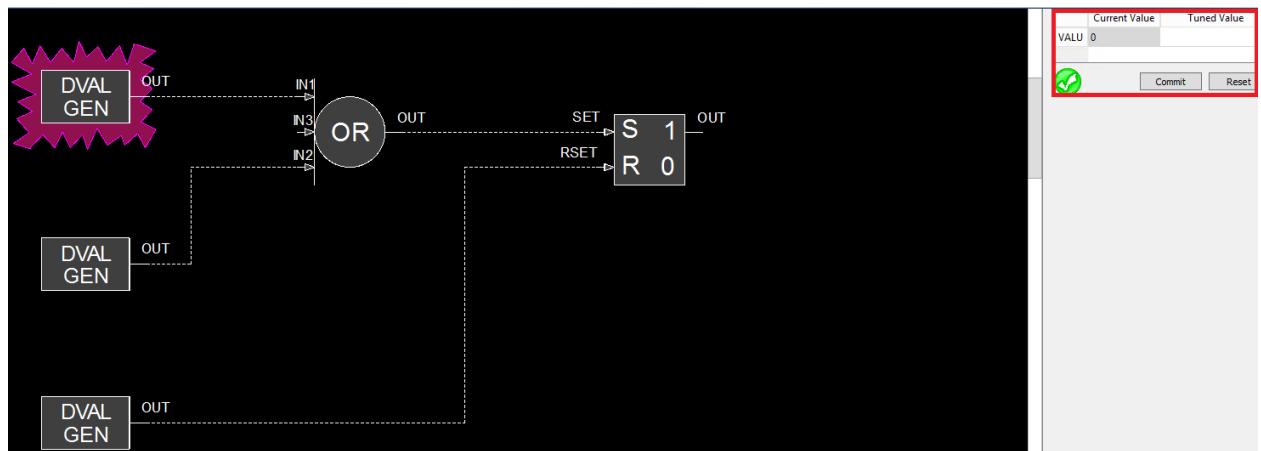
##### Signal Diagram

W celu wyświetlenia działającej logiki należy wywołać program „Signal Diagram” (w Okna startu Windows) i odnaleźć stosowną logikę na drzewie projektowym (Rys. 4.9).



Rys. 4.23: Widok logik w programie Signal Diagram.

Przycisk OK wyświetla wybraną logikę.



Rys. 4.24: Logika w programie Signal Diagram.

W prawym górnym oknie możliwe jest ustawianie wartości algorytmu DVALGEN i sprawdzanie działania logiki (Rys. 4.10).

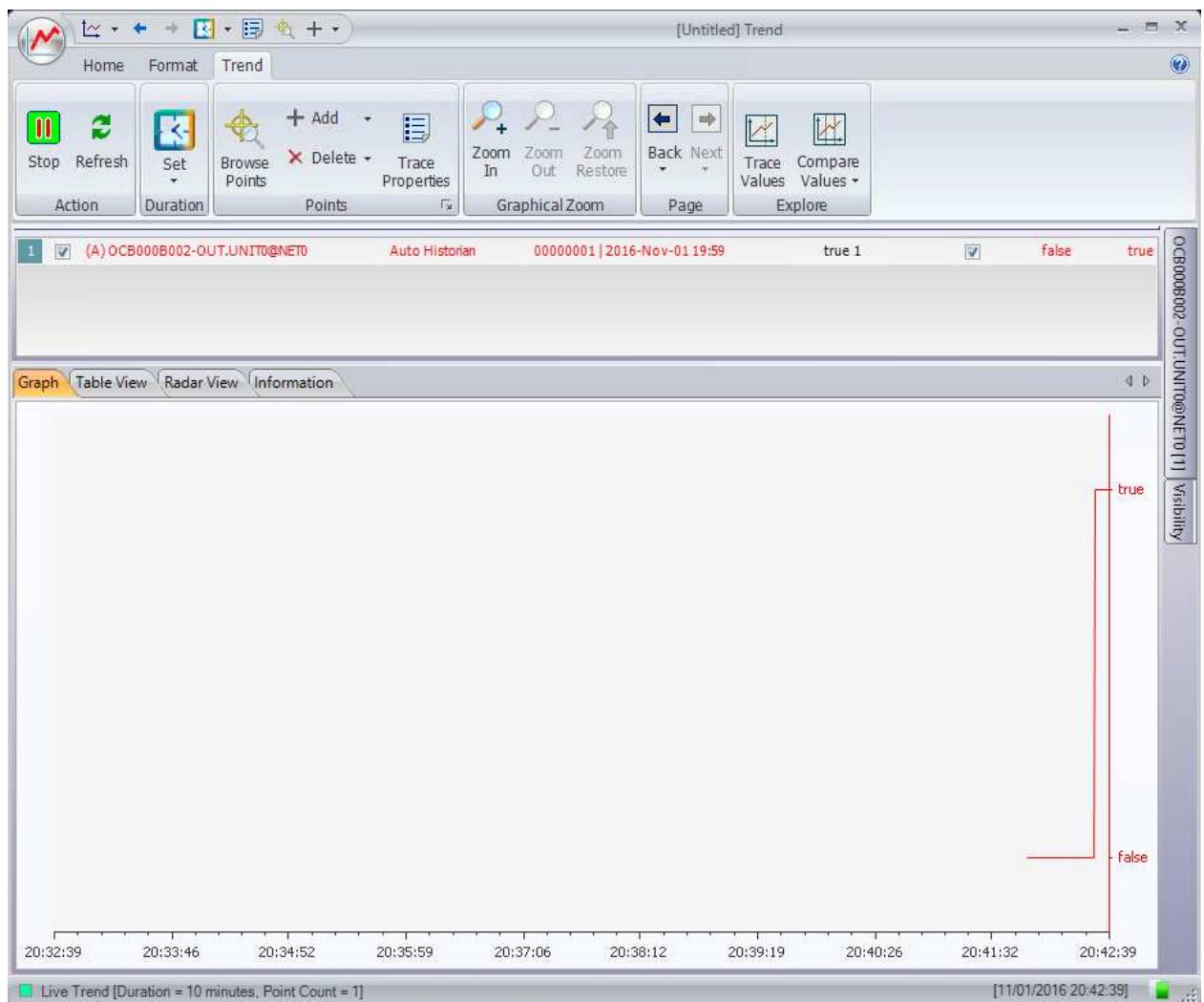
## Trend i Point Information

Z okna Signal Diagram można bezpośrednio wywołać aplikację Trend i Point Information. W tym celu należy na liście „Algorithm Summary” wybrać punkt i rozwinąć menu prawego klawisza myszy (Rys. 4.11).

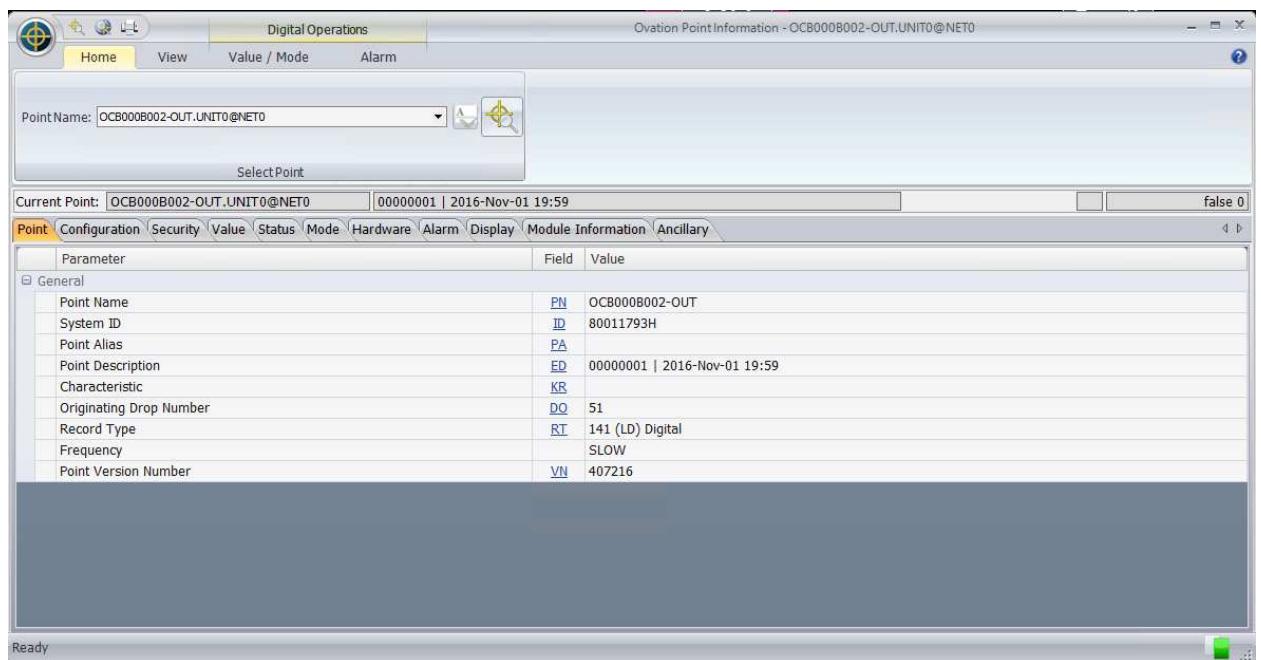
Name	Parameter Description	Point	Point Description	Type	Bottom	Top	
I1	Input 1	OCB000B002-OUT.UNIT0@NET0	00000001   2016-Nov-01 19:59	LD	false	true	false (0)
IN	OCB000B002-OUT.UNIT0@NET0	-OUT.UNIT0@NET0	00000001   2016-Nov-01 19:59	LD	false	true	false (0)

Rys. 4.25: Wywołanie programów Trend i Point Information z okna Signal Diagram.

Point Info wyświetli informację o wybranym punkcie (Rys. 4.13) a Trend wyświetli trend wartości punktu (Rys. 4.12).



Rys. 4.26: Okno programu Trend.

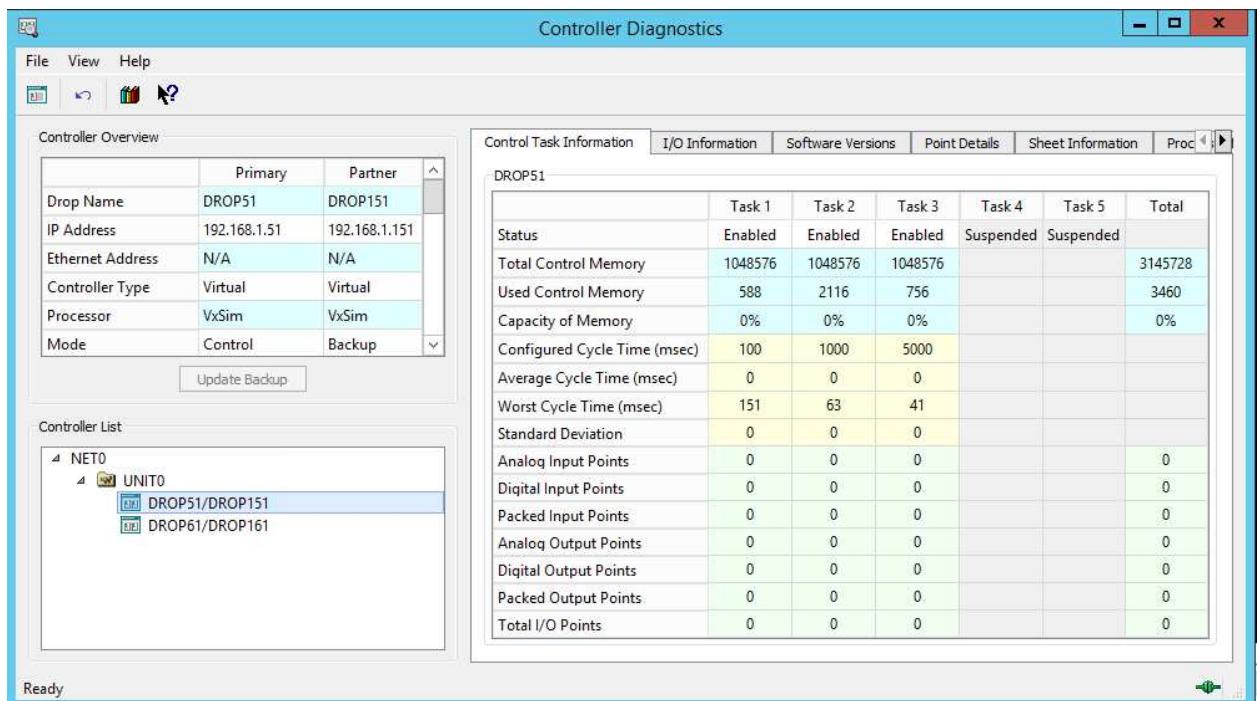


Rys. 4.27: Okno programu Point Information.

Te same aplikacje mogą być wywołanie z okna systemowego Windows ale wówczas wywołanie takie jest bez kontekstu i należy ręcznie podać nazwę punktu, który chce się obserwować.

### Controller Diagnostics

Aplikacja może być wywołana poleciem „Controller Diagnostics” i wyświetla informacje związane z kontrolerami (Rys. 4.14).



Rys. 4.28: Okno programu Controller Diagnostics.

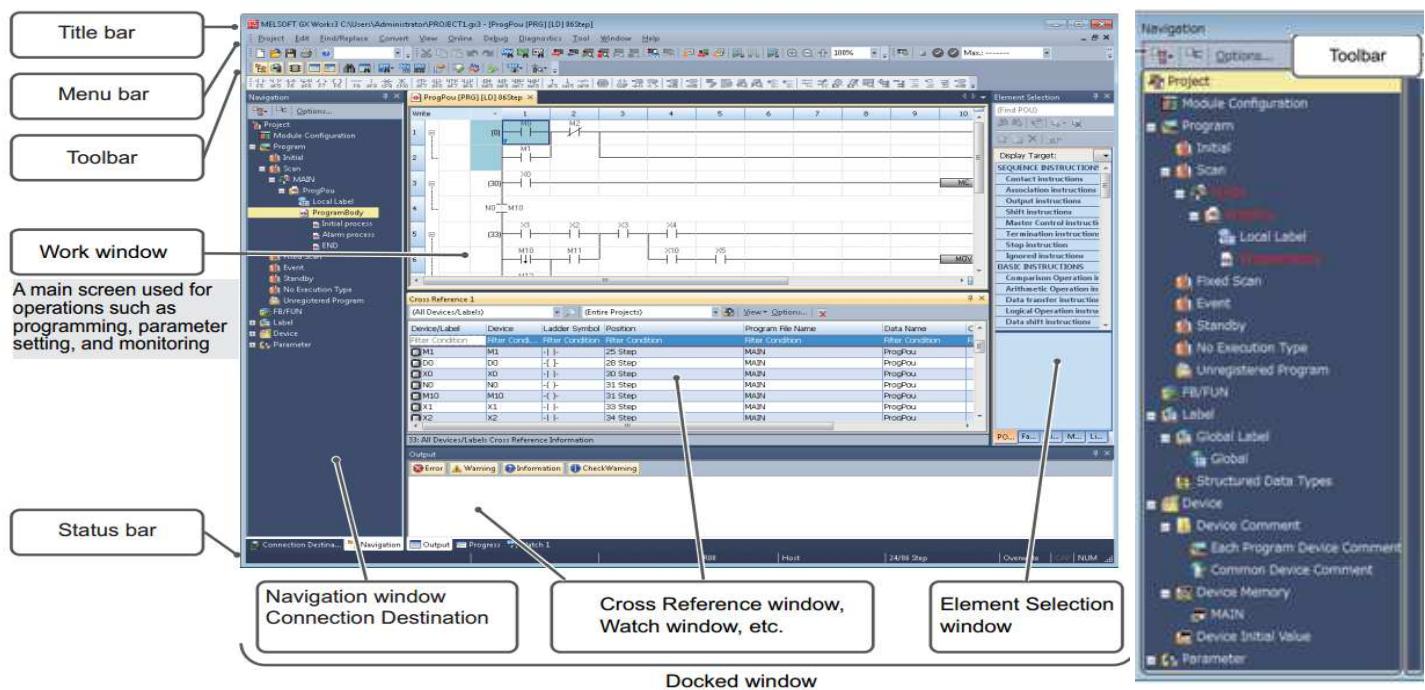
## 5 Programowanie w systemie SCADA

### 5.1 Wprowadzenie

Celem niniejszego rozdziału jest zapoznanie studentów z narzędziami systemu SCADA. Po krótkiej prezentacji narzędzi przedstawione są przykłady praktyczne, pokazujące krok po kroku sposób użycia aplikacji. Zaznajomienie się z materiałem z niniejszego rozdziału jest warunkiem koniecznym do wykonania zadań laboratoryjnych w systemie SCADA.

### 5.2 Oprogramowanie sterownika PLC – GX Works 3

Rozdział ten dotyczy tworzenia kodu sterującego sterownika PLC w środowisku GxWorks3. Na rysunku poniżej przedstawiono najważniejsze okna do zarządzania projektem, edycję programów, podglądem zmiennych.



**Rysunek** Okno główne programu GxWorks3 (z lewej); Pasek nawigacji projektu (z prawej)

### 5.2.1 Tworzenie nowego projektu

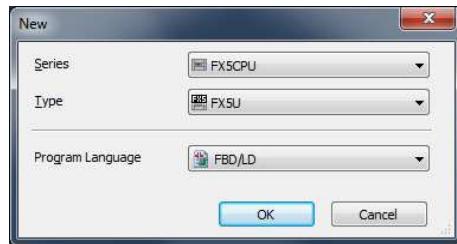
Proszę utworzyć nowy projekt ([Project] → [New] ( )

Proszę zidentyfikować model oraz serię sterownika PLC znajdującego się na stanowisku. Na poniższym rysunku oznaczono czerwonym prostokątem miejsce, w którym znajduje się wymagana informacja.



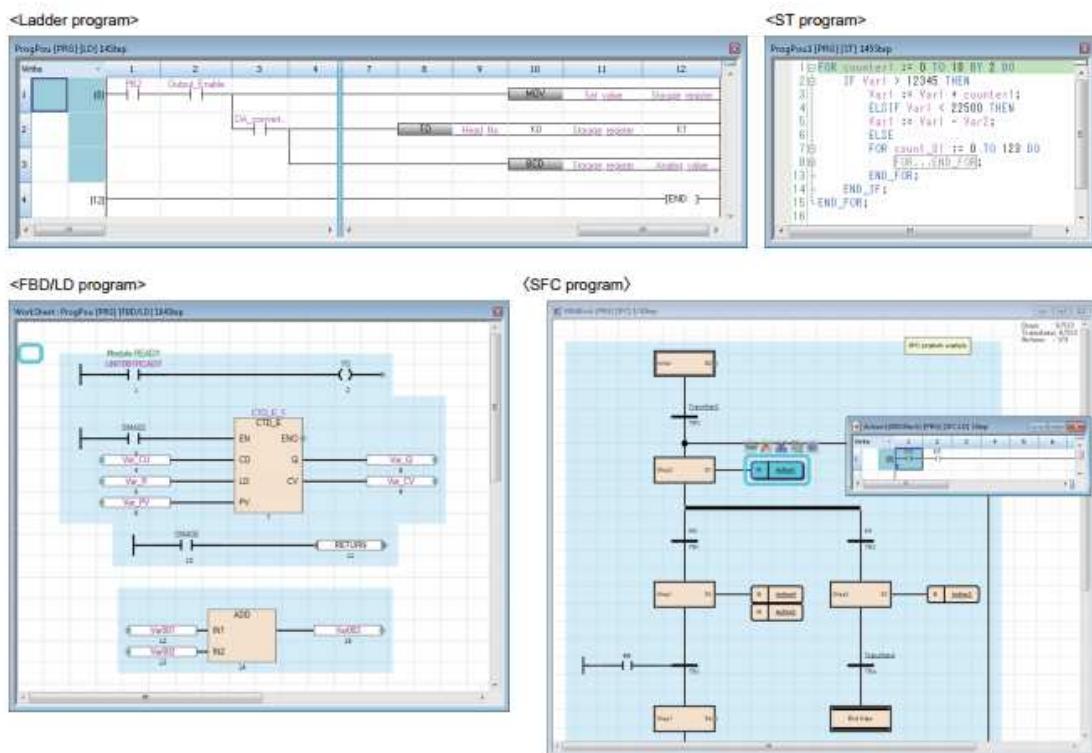
**Rysunek 1** Oznaczenie modelu i serii sterownika.

Następnie w programie GxWorks3 proszę wprowadzić dane sterownika oraz wybrać język programowa **FBD/LD**, po czym zatwierdzić ustawienia klikając przycisk OK.



**Rysunek 2 Parametry wstępne projektu – NIE POMYLIĆ SERII STEROWNIKA**

Środowisko GXWorks3 wspiera programowanie zgodnie z normą IEC61131-3 (wsparcie dla: FBD/LD, Ladder Diagram, ST i SFC). Przykłady programów we wspomnianych językach zaprezentowano na poniższym rysunku.



**Rysunek 3 Języki programowania wspierane przez aplikację GxWorks3.**

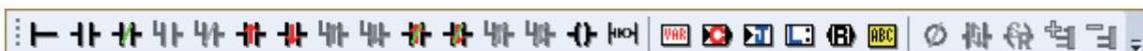
---

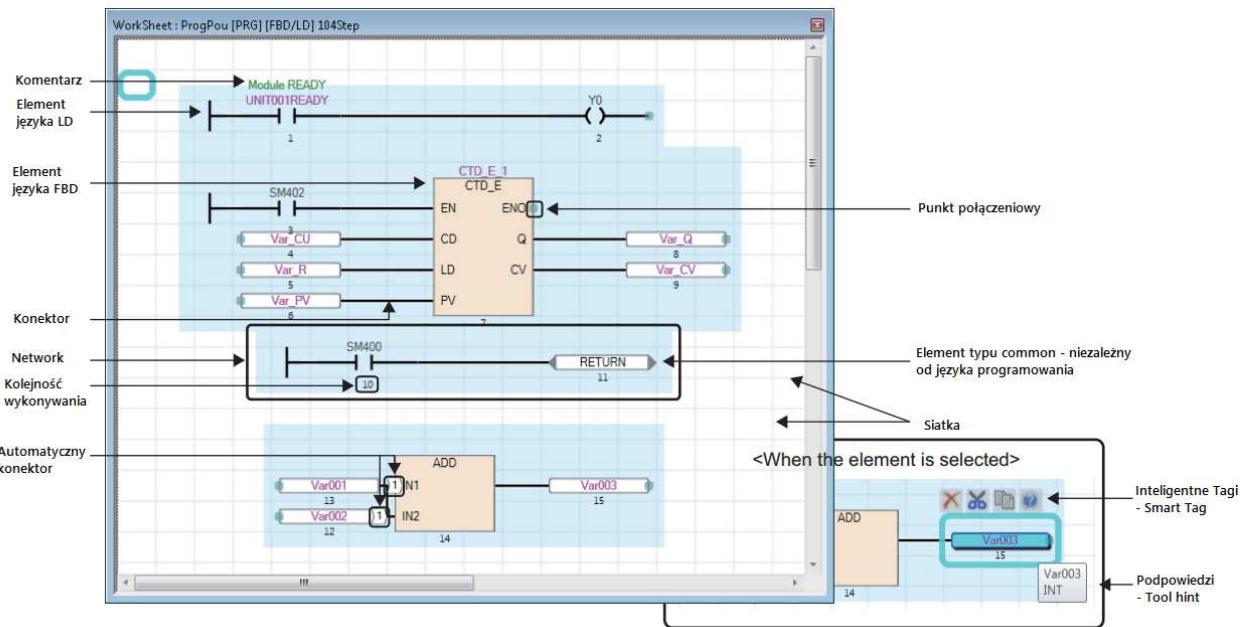
**Uwaga:** Proszę upewnić się, że projekt został zapisany [Project] → [Save] (💾)

---

### 5.2.2 Elementy języka FBD/LD

Pasek narzędziowy zawiera wszystkie elementy strukturalne języka FBD/LD:





Rysunek 4 Edytor języka FBD/LD.

Element	Opis
Komentarz	Komentarz etykiety lub bloku funkcjonalnego. Nie podlega komplikacji.
Element języka LD	Element pochodzący z języka programowania Ladder Diagram
Element języka FBD	Element pochodzący z języka Function Block Diagram (FBD)
Element typu common	Element wbudowany, niezależny od języka programowania
Konektor	Linia łącząca punkty pomiędzy elementami programu. Możliwe jest automatyczne łączenie punktów poprzez zbliżenie bloków.
Network	Pojedyncza sieć zbudowana ze wszystkich elementów podłączonych razem. Program może zawierać maksymalnie 4096 networków.
Kolejność wykonywania (execution order)	Liczba określająca kolejność wykonania danego elementu programu.
Automatyczny konektor	Jeśli konektor nie może być wyświetlony w danym miejscu, wtedy zostaje zastąpiony liczbą.
Punkt podłączeniowy	Terminal (punkt) pozwalający na połączenie bloków/elementów programu poprzez konektor. Punkty powinny być łączone z uwzględnieniem typów danych.
Siatka	Linie siatki arkusza na którym umieszczane są elementy programu
Smart tag	Przyciski wyświetlane nad wybranym elementem, pozwalające na wykonanie operacji t.j. np.

	usuwanie lub kopiowanie elementu.
Tool hint	Informacja o elementach programu wyświetlana po najechaniu kursorem myszki

## Elementy języka LD

Element	Opis
Lewa szyna zasilająca	(1) Wyjściowy punkt połączeniowy (2) Lewa szyna zasilająca
Element stykowy	(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Komentarz urządzenia/etykiety (4) Urządzenie/etykieta
Cewka	(1) Wyjściowy punkt połączeniowy (2) Wejściowy punkt połączeniowy (3) Komentarz urządzenia/etykiety (4) Urządzenie/etykieta

## Elementy języka FBD

Element	Opis
Zmienna (lokalna/globalna)	(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Komentarz urządzenia/etykiety (4) Urządzenie/etykieta
Stała	(1) Wyjściowy punkt połączeniowy (2) Stała wartość
Blok funkcyjny	(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Nazwa instancji FB (4) Komentarz etykiety (5) Typ danych (6) Etykieta wejścia/wyjścia

	<p><b>Uwaga:</b> wyjścia z bloku mogą być umieszczone również z lewej strony symbolu bloku funkcyjnego</p>
<p>Funkcja</p>	<p>(1) Wejściowy punkt połączeniowy  (2) Wyjściowy punkt połączeniowy  (3) Typ danych  (4) Etykieta wejścia/wyjścia</p> <p><b>Uwaga:</b> wyjścia z bloku mogą być umieszczone również z lewej strony symbolu bloku funkcyjnego</p>

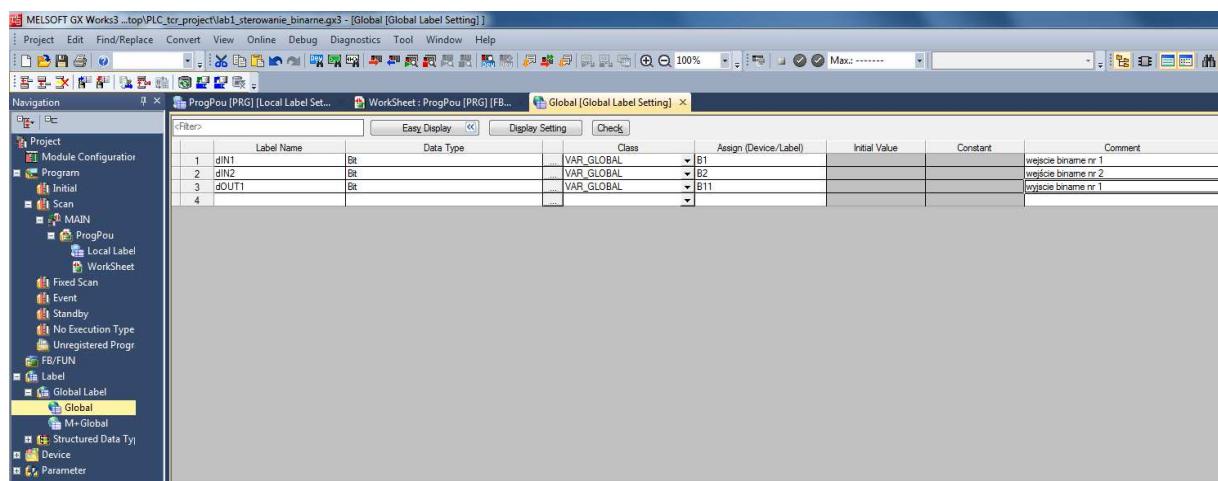
### Elementy wspólne (common element)

Element	Opis
Instrukcja skoku (Jump element) 	(1) Wejściowy punkt połączeniowy (2) Etykieta
Etykieta instrukcji skoku 	(1) Wyjściowy punkt połączeniowy
Konektor 	(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy (3) Komentarz etykiety
Instrukcja return 	(1) Wejściowy punkt połączeniowy (2) Wyjściowy punkt połączeniowy
Blok komentarza 	(1) Powierzchnia na której zostanie treść komentarza



### 5.2.3 Definicja zmiennych

W celu tworzenia czytelnego kodu zalecane jest wykorzystanie zmiennych symbolicznych zamiast adresowania bezpośredniego pamięci sterownika. W tym celu należy wykorzystać mechanizm etykiet (label), które mogą mieć zakres lokalny (widoczne tylko w danym komponencie lub podprogramie) lub globalny (widoczne w całym systemie i propagowane po sieci – to rozwiązywanie jest zalecane z uwagi na możliwość przypisania fizycznych urządzeń, które później będą skanowane w systemie SCADA, przez fizyczne urządzenia rozumiemy tutaj odpowiedniego adresy pamięci sterownika PLC). Definicja etykiet możliwa jest poprzez wypełnienie tabeli (patrz rysunek poniżej) lub poprzez bezpośrednie definiowanie w trakcie tworzenia kodu sterującego - wpisanie nazwy nowej zmiennej symbolicznej w miejscu jej użycia powoduje otwarcie okna dialogowego, gdzie możliwa jest konfiguracja zmiennej. Jeżeli posługujemy się językiem skryptowym ST po wpisaniu tekstu nowej nazwy klikamy na niej prawym przyciskiem i wybieramy opcję „Register Label”. Jeżeli etykieta nie ma koloru „magenta” znaczy to, że nie została zadeklarowana bądź wpisano złą jej nazwę w kodzie.



Rysunek 5 Deklaracja lokalnych/globalnych etykiet

Możliwe jest również tworzenie grup zmiennych tworząc pomocnicze kontenery (np. Wejścia, Wyjścia, Sygnały\_analogowe, Sygnały\_dyskretne itp.). Aby to zrobić należy w oknie Navigation przejść do zakładki Label -> Global Label, następnie kliknąć prawym przyciskiem myszy i wybrać opcję Add New Data.

W trakcie laboratorium przydatne będą urządzenia typu Bit, Rejestr. Zakres urządzeń typu Bit zawiera się w zakresie od M0 do M7680 - numerowane co jeden. Zakres urządzeń typu Rejestr(16 bit) zawiera się w zakresie D0 do D7999 - numerowane co jeden. Proszę zwrócić szczególną uwagę, że zmienne typu Double lub Float zajmują dwa rejesty D. Rysunek 7a przedstawia przykładową konfigurację. Przy pisaniu programu PLC można używać etykiet, ale do skanowania zmiennych w systemie MAPS należy używać bezpośrednich adresów pamięci zadeklarowanych w kolumnie Assign. Przykładowo jeżeli

byędziemy chcieli wyświetlić wartość „zmienna6” w systemie MAPS trzeba zeskanować rejestr D8 sterownika PLC.

	Label Name	Data Type	Class	Assign (Device/Label)
1	zmienna1	Word [Unsigned]/Bit String [16-bit]	... VAR_GLOBAL	D0
2	zmienna2	Double Word [Unsigned]/Bit String [32-bit]	... VAR_GLOBAL	D1
3	zmienna3	Word [Signed]	... VAR_GLOBAL	D3
4	zmienna4	Double Word [Signed]	... VAR_GLOBAL	D4
5	zmienna5	FLOAT [Single Precision]	... VAR_GLOBAL	D6
6	zmienna6	Word [Unsigned]/Bit String [16-bit]	... VAR_GLOBAL	D8
7	start_auto	Bit	... VAR_GLOBAL	M0
8	stop	Bit	... VAR_GLOBAL	M1
9			...	...

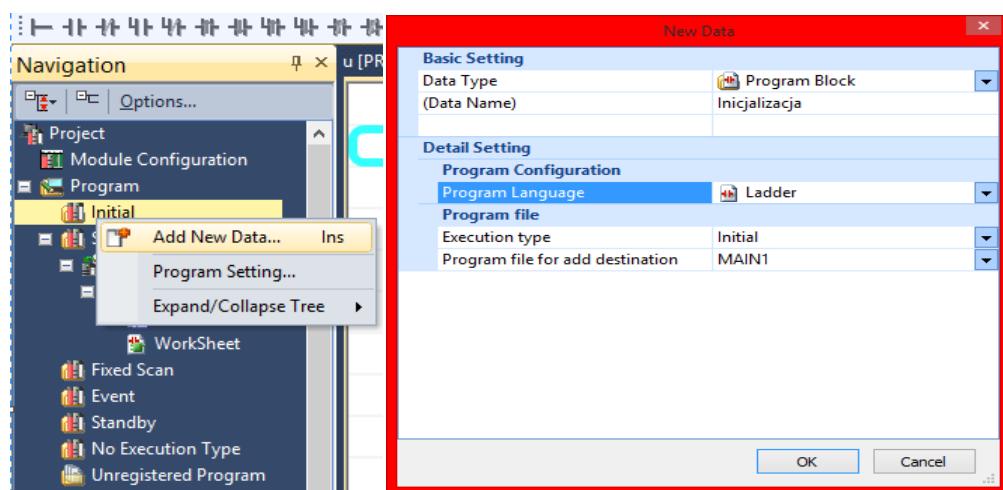
Rysunek 6a Deklaracja globalnych „Labeli”.

#### 5.2.4 Tworzenie kodu sterującego

W zależności od sposobu wykonywania programu sterującego należy określić jego lokalizację w drzewie projektu. Wspierane są następujące sekcje:

- Initial - instrukcje wykonywane tylko w pierwszym cyklu sterownika,
- Scan - główny skan procesora, czas cyklu zależny obciążenia,
- Fixed scan - skan z narzuconym okresem wykonania (czas konfigurowalny),
- Event - obsługa zdarzeń,
- No execution Type - magazyn kodu, który nie jest wykonywany.

W każdej z sekcji można stworzyć kilka podprogramów klikając w sekcję prawym przyciskiem myszy a następnie wybierając z menu opcję „Add New Data” (Patrz poniżej).



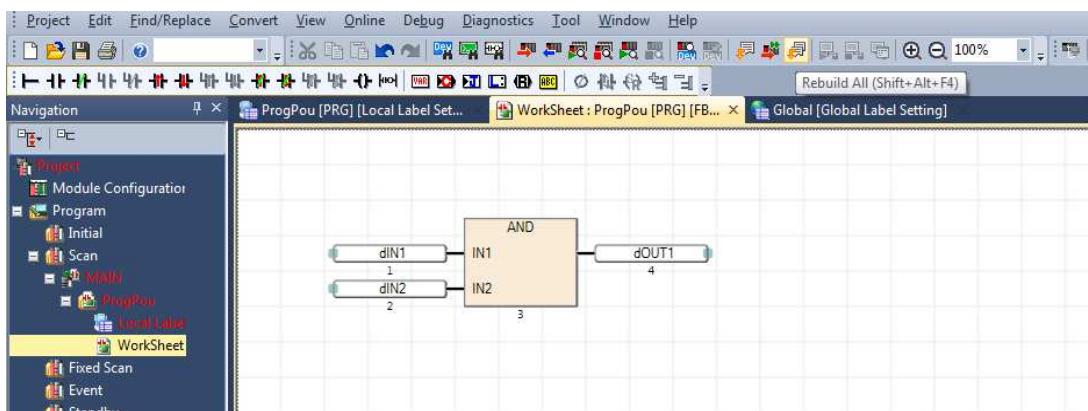
Rysunek 7 Dodawanie nowych podprogramów

Wstawianie elementów języka FBD na arkusz roboczy może odbywać się dzięki technice „drag and drop” z biblioteki (znajduje się po prawej stronie), lub poprzez bezpośrednie wpisywanie z klawiatury nazwy bloku funkcyjnego. W trakcie wpisywania kolejnych znaków podpowiadzi o dostępnych blokach są wyświetlane pod tworzonym blokiem.

---

**Uwaga:** Należy zwrócić szczególną uwagę na kolejność wykonywania algorytmów. Kolejność wykonywania bloków jest zaznaczona małymi cyferekami pod każdym blokiem. Ułożenie bloków na karcie edycji może zmienić kolejność wykonania poszczególnych instrukcji co bezpośrednio ma wpływ na działania.

---

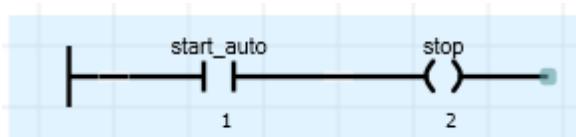


Rysunek 8 Przykład tworzenia kodu sterującego.

W czasie laboratorium najbardziej użyteczne będą następujące instrukcje:

1. Styk normalnie otwarty
2. Cewka wyjściowa (należy pamiętać, że w programie powinna być tylko jeden raz do jednej zmiennej)

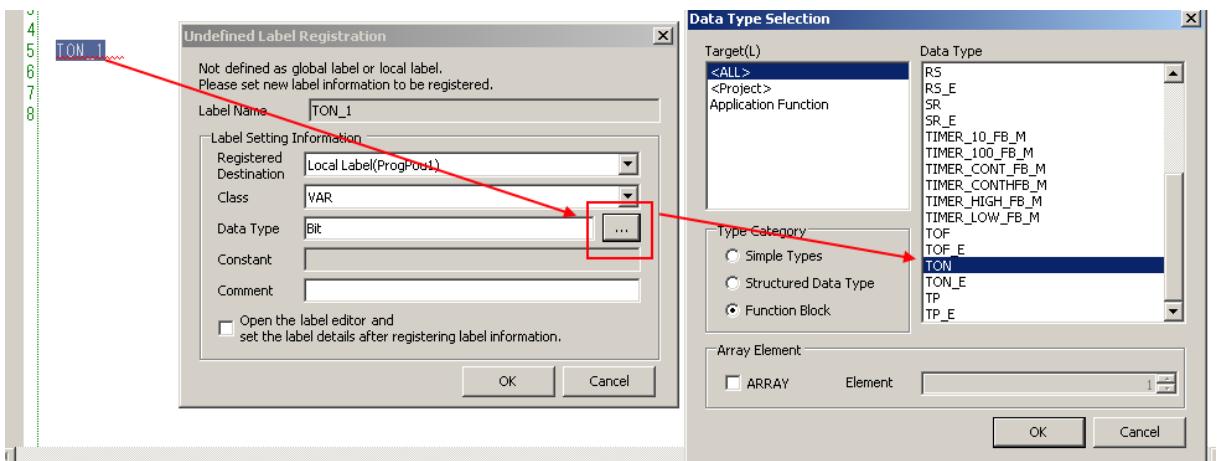
#### FBD:



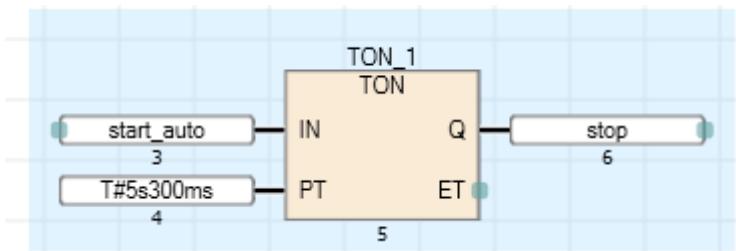
#### ST:

OUT(start\_auto, stop);

3. Opóźnienie załączenia TON, opóźnienie wyłączenia TOF, impuls o zadanym czasie TP. Wszystkie te funkcje potrzebują deklaracji instancji w zmiennych lokalnych. Wprowadzenie nowej instancji można wykonać wpisując jej nazwę, następnie klikamy prawym przyciskiem myszy na wprowadzonej nazwie, wybieramy „Register Label” i wybieramy kolejne opcje jak na rysunku poniżej.



### FBD:



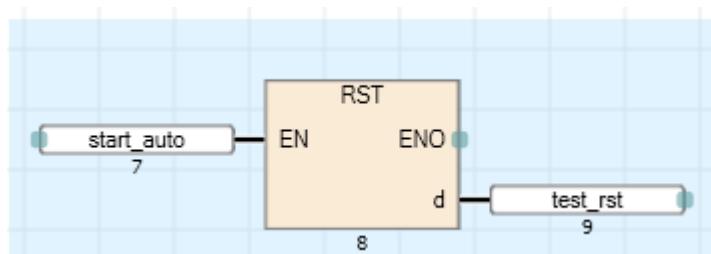
### ST:

```

TON_1(IN:= start_auto ,PT:= T#5s300ms ,Q=> stop );
TON_2(PT:= T#5s300ms);
    
```

### 4. Instrukcja ustawienia SET, kasowania RST

### FBD:



### ST:

```

SET(start_auto, test_set);
RST(start_auto, test_RST);
    
```

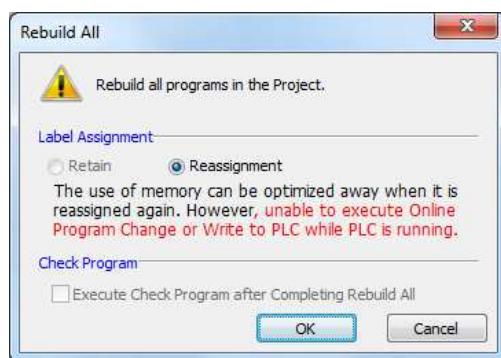
Należy unikać stosowania nazw zmiennych, które mogą być nazwami własnymi zastrzeżonymi w programie np. STOP, ST. Mogą one oznaczać nazwy instrukcji lub nazwy urządzeń fizycznych.

### 5.2.5 Kompilacja kodu



- 1 – Kompilacja (po małych modyfikacjach)
- 2 – Rekompilacja ( po zmianach konfiguracyjnych)

Rekompilacja wymaga potwierdzenia komunikatu:



Rysunek 9 Rekompilacja – okno dialogowe

Po rekompilacji projektu nie możliwe będzie ładowanie sterownika w trybie Online. Z tego powodu drobne zmiany w programie należy zatwierdzać bezpośrednio zapisując projekt i wywołując komendę „*Online Program Change*” – przycisk między 1 a 2.

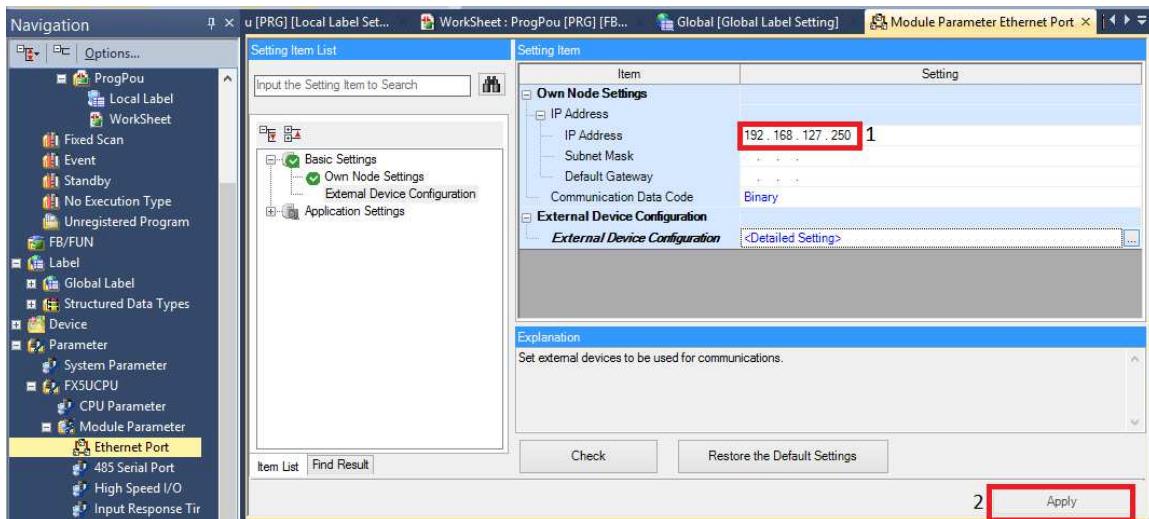
### 5.2.6 Konfiguracja sterownika

W celu umożliwienia komunikacji sterownika z panelem GOT oraz serwerem SCADA należy skonfigurować jego ustawienia sieciowe. Poniższe instrukcję przeprowadzają przez wymagane operacje. Dodatkowo wprowadzona zostanie komunikacja ze środowiskiem MATLAB przy pomocy Socket Communication, dzięki czemu możliwe będzie szybkie i wygodne przesłanie danych z przykładowych scenariuszy regulacji procesów laboratoryjnych.



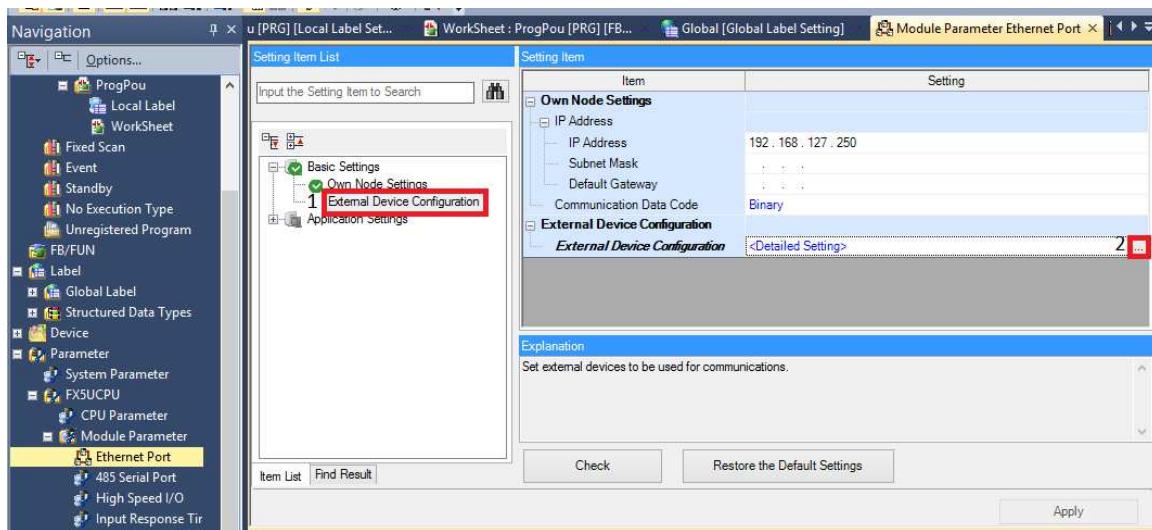
**Rysunek 10** Adresacja urządzeń w sieci lokalnej stanowiska

### Ustawienie adresu IP sterownika:

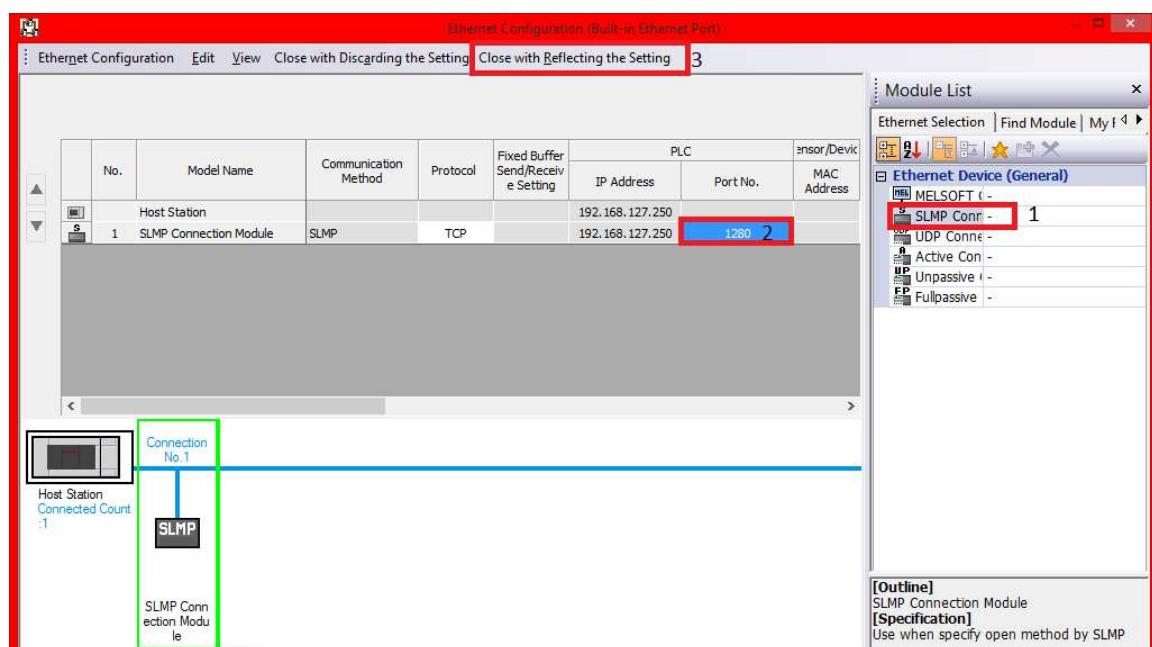


**Rysunek 11** Ustawienie adresu IP portu Ethernet

### Ustawienie komunikacji z MAPS:

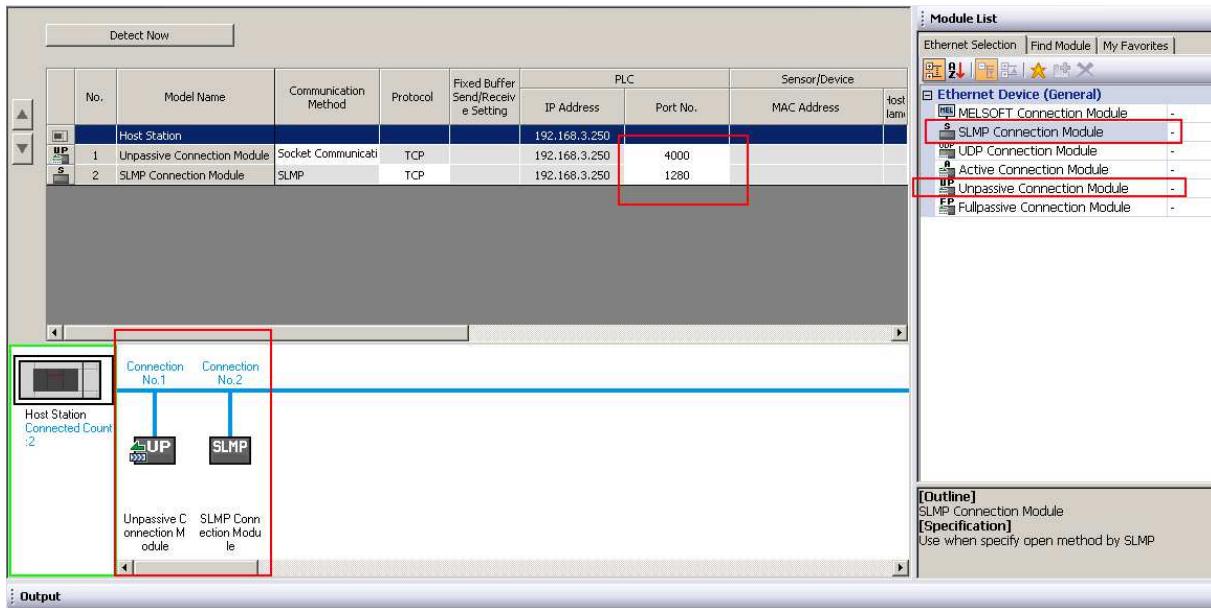


Rysunek 12 Wywołania okna konfiguracji zewnętrznej komunikacji



Rysunek 13 Dodanie komunikacji SLMP (port 1280) i Socket Communication (port 4000)

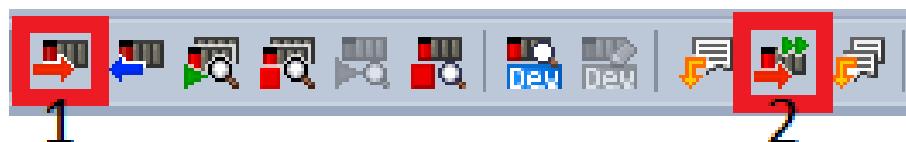
Poniżej przedstawiono ostateczne wymagane ustawienie komunikacji poprzez Ethernet.



**Rysunek 16 Dodanie komunikacji Unpassive (port 4000)**

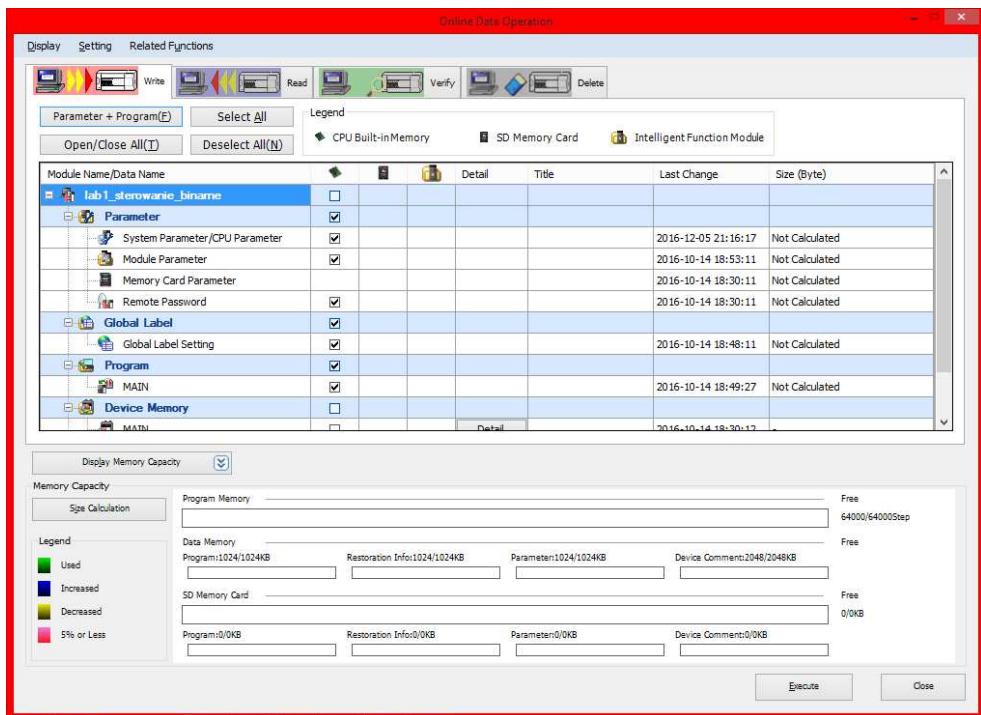
### 5.2.7 Programowanie sterownika

Zmiany konfiguracyjne wymagają pełnego ładowania sterownika z ręcznym restartem. W tym celu należy skompilować projekt i wybrać opcję „Write to PLC” (opcja 1 z poniższego rysunku). Do wprowadzenia szybkich zmian na sterowniku (np. modyfikacja logiki kontrolera) bez restartu kontrolera należy wykorzystać opcję „Online Program Change” (opcja 2 z poniższego rysunku). Operacja ta nie może być poprzedzona komplikacją, gdyż ta odbywa się automatycznie przed aktualizacją programu sterującego.



**Rysunek 14 Operacja ładowania sterownika.**

Wybór opcji „Write to PLC” przekierowuje do okna „Online Data Operation”, gdzie można przeprowadzić operacje: zapisu, odczytu, weryfikacji oraz czyszczenia pamięci kontrolera.



**Rysunek 15 Okno Online Data Operation – Zapis/Odczyt/Veryfikacja/Czyszczenie sterownika.**

---

**Uwaga 1:** Przed operacją ładowania kontrolera należy upewnić się, że projekt nie zawiera błędów(w oknie Output nie powinno być żadnych błędów – Error)

---



---

**Uwaga 2:** Jeżeli przy próbie wgrywania programu do sterownika otrzymamy komunikat błędu „Inconsistency.....” należy wówczas przejść do zakładki Delete, wybrać wszystkie elementy przez Select All i wcisnąć Execute (nastąpi usunięcie starych parametrów i programów ze sterownika). Następnie należy powrócić do zakładki Write i przez Select All a następnie Execute wgrać nowy program i parametry do sterownika.

---



---

**Uwaga 3:** Po wykonaniu operacji wgrania parametrów i programu należy wykonać sprzętowy RESET sterownika PLC. Wykonuje się to przez otworzenie pokrywki po lewej stronie sterownika, przełączenie dźwigienki z pozycji RUN do RESET, przytrzymanie dźwigienki do momentu pojawiennia się diody ERR na sterowniku a następnie powrót do pozycji RUN. W tym momencie sterownik został zresetowany i można kontynuować pracę.

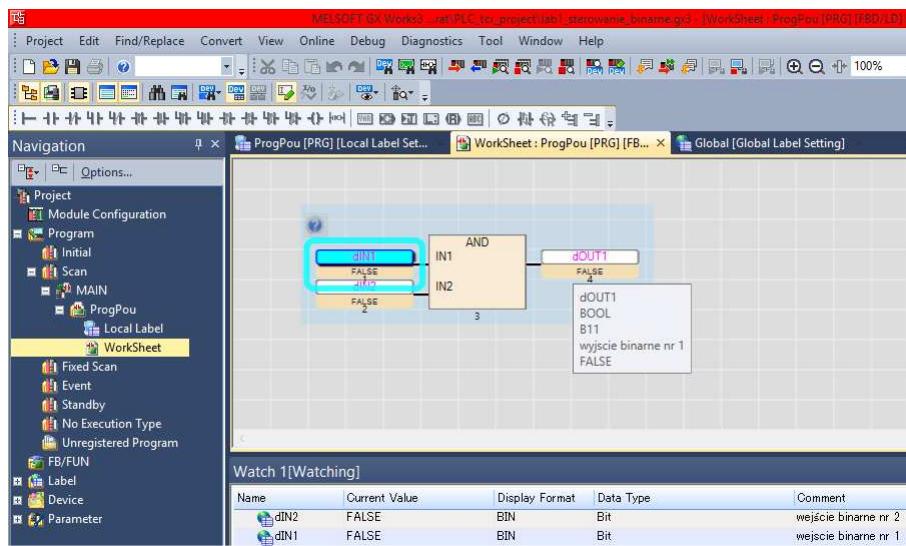
---

### 5.2.8 Diagnostyka, monitorowanie działania programu

Po załadowaniu kontrolera możliwy jest podgląd wykonywania programu za pomocą opcji „Start Monitoring”.

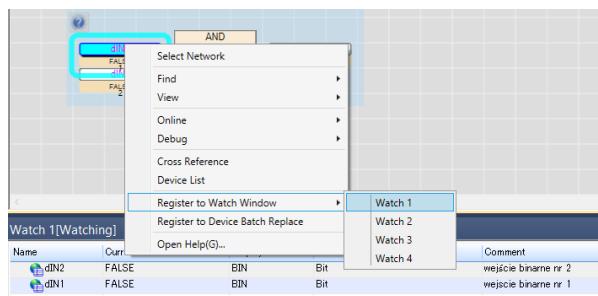


**Rysunek 16 Uruchamianie Monitora.**



**Rysunek 17 Podgląd wykonywania programu**

W celu zmiany/wyświetlania wartości zmiennych należy dodać je do podglądu za pomocą mechanizmu Watch'a. Należy najechać kursem na nazwę zmiennej a następnie prawym przyciskiem myszy wybrać otworzyć menu i wybrać Register to Watch Window -> Watch 1. Można również w kolejnych wierszach wpisywać bezpośrednio nazwy zmiennych lub rejesty pamięci sterownika (np. D100, M22, X10, Y2). Z poziomu okienka Watch można zmieniać wartości zmiennych w celu testowania działania programu. Oczywiście w czasie pracy sterownika niektóre zmienne mogą być natychmiast nadpisywane przez program. Dodatkowo warto zwrócić uwagę na kolumnę typu danych. W szczególności jest to istotne, kiedy operujemy na 32 bitowych zmiennych, aby wybrać typ np. Double Word.

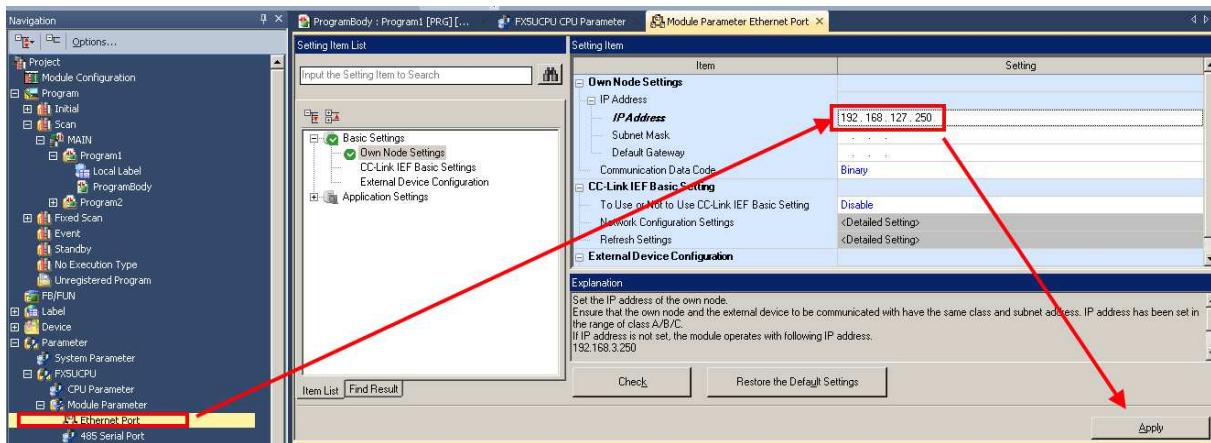


**Rysunek 18 Uruchomienie Watch'a.**

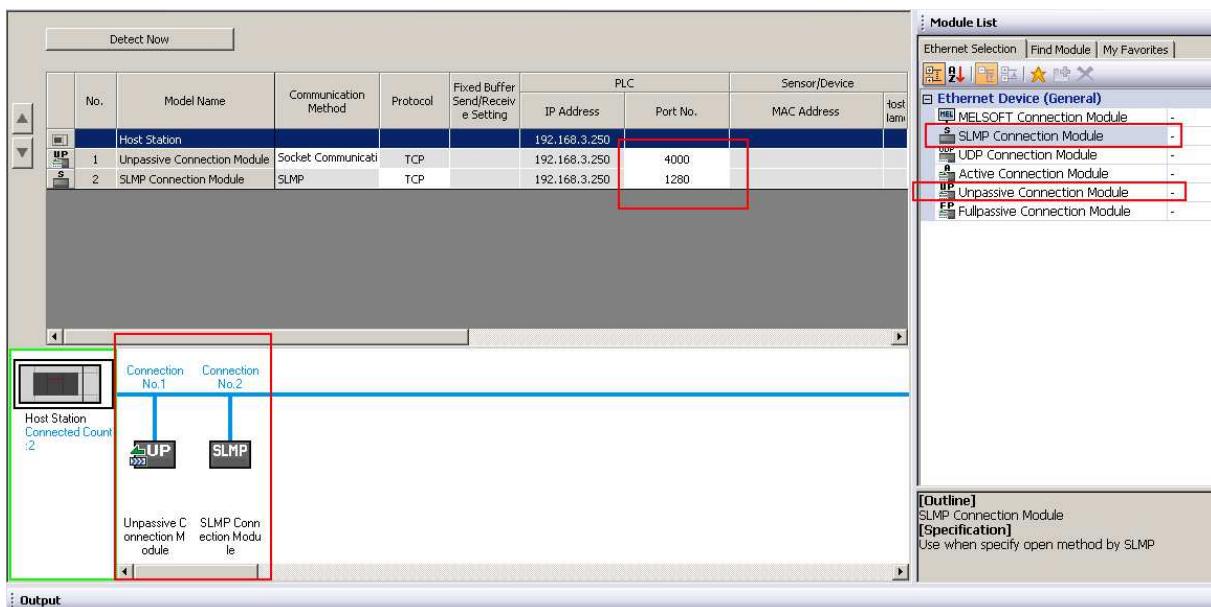
## 5.2.9 Pierwszy program PLC

Pierwszy program wgrywany do sterownika powinien obejmować ustawienia wymagane przy realizacji ćwiczeń laboratoryjnych. Poniżej przedstawiono kolejno konfigurację wymaganych opcji.

## Ustawienie adresu IP sterownika PLC na 192.168.127.250



Konfigurację komunikacji dla MATLAB poprzez dodanie Unpassive TCP connection i ustawienie portu 4000. Jednocześnie konfigurację komunikacji dla MAPS poprzez dodanie SLMP connection i ustawienie portu 1280.



Dodanie programu w sekcji INIT – inicjalizacja zmiennych dla symulacji procesów i regulatorów

```

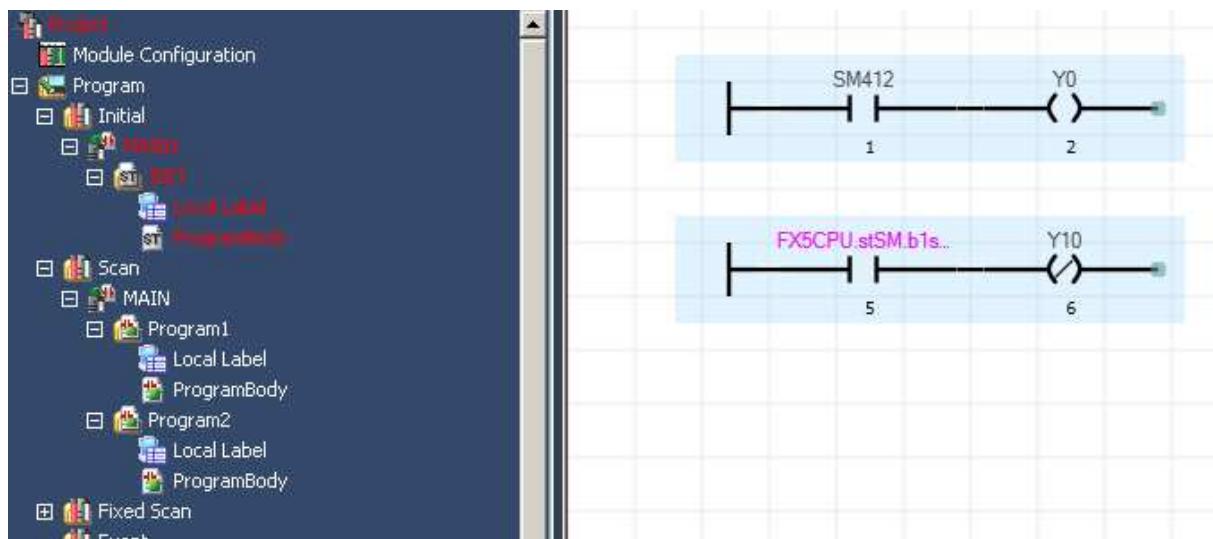
Navigation
Project
Module Configuration
Program
Initial
MAIN1
INIT
Scan
MAIN
Program1

ProgramBody : Program1 [PRG] [...]
FX5UCPU

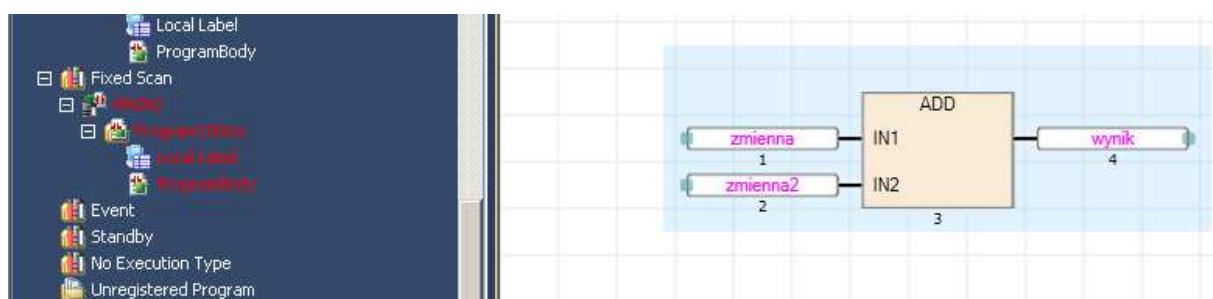
1 //Program inicjujacy zmienne
2
3 RST(TRUE, regulacja);
4 SET(TRUE, testowy);
5 MOV(TRUE, zmienna);
6 zmienna2 := 0.0;

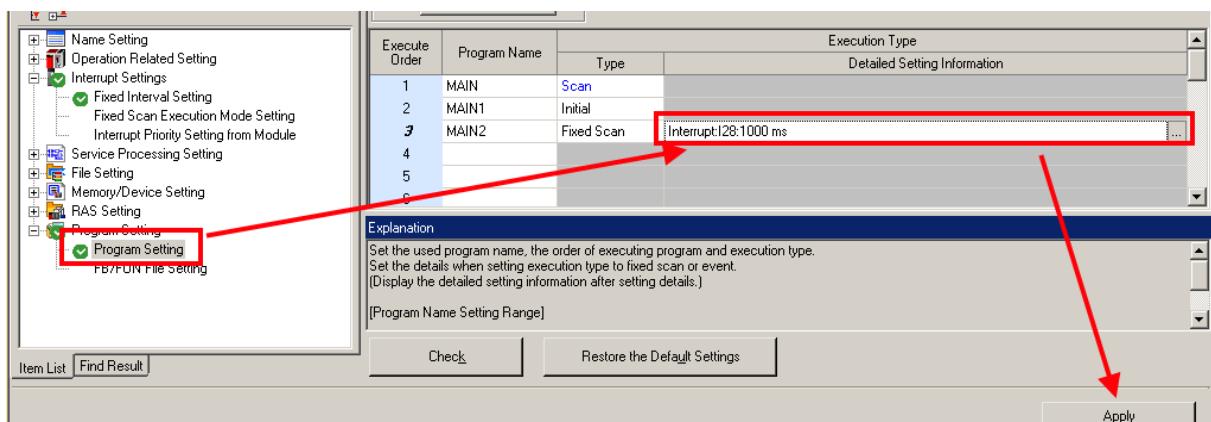
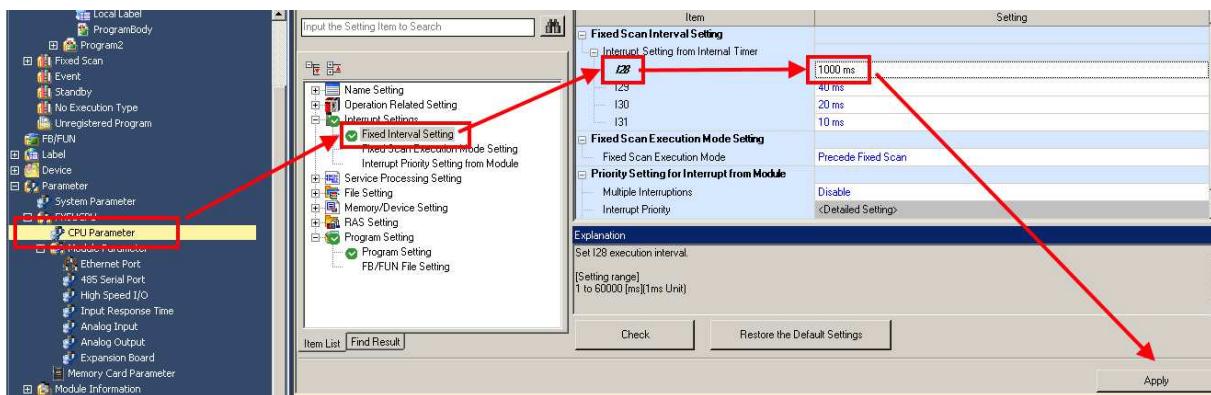
```

Dodanie programu w sekcji SCAN – operacje wykonywane cyklicznie ze skanem procesora. W tej grupie może znajdować się kilka programów, które są odpowiedzialne za różne procesy. Należy pamiętać, że programy w czasie jednego skanu PLC są wykonywane jeden po drugim.



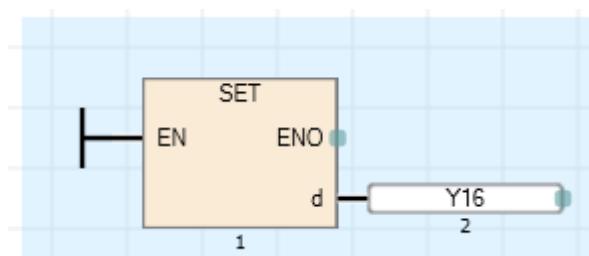
Dodanie programu FIXED SCAN – operacje wykonywane cyklicznie ze skanem 1000ms (czas dyskretyzacji procesów regulacji i regulatorów). Okres próbkowania programu z tej grupy jest ustalony w parametrach sterownika, co zostało przedstawione poniżej.





Program przykładowy w sekcji INIT:

### FBD:

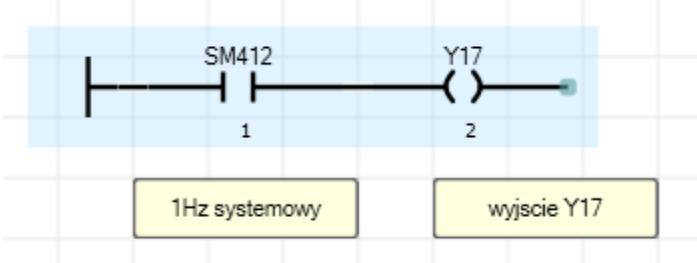


### ST:

```
SET(TRUE, Y16);
```

Po uruchomieniu sterownika lub jego resecie po wgraniu programu powinno aktywować się wyjście Y16, co można zaobserwować na zielonych diodach na sterowniku lub na odpowiednim ekranie na panelu GOT.

Program przykładowy w sekcji SCAN:

**FBD:****ST:**

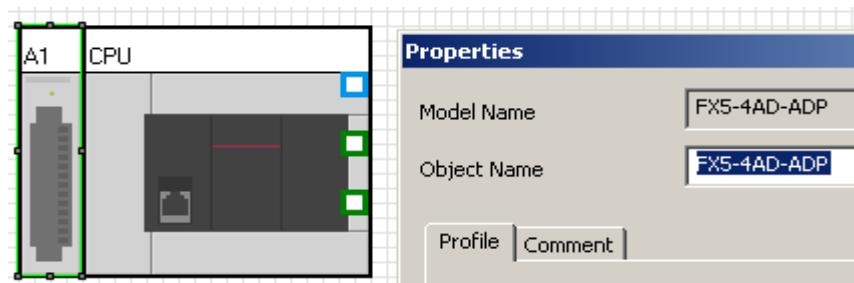
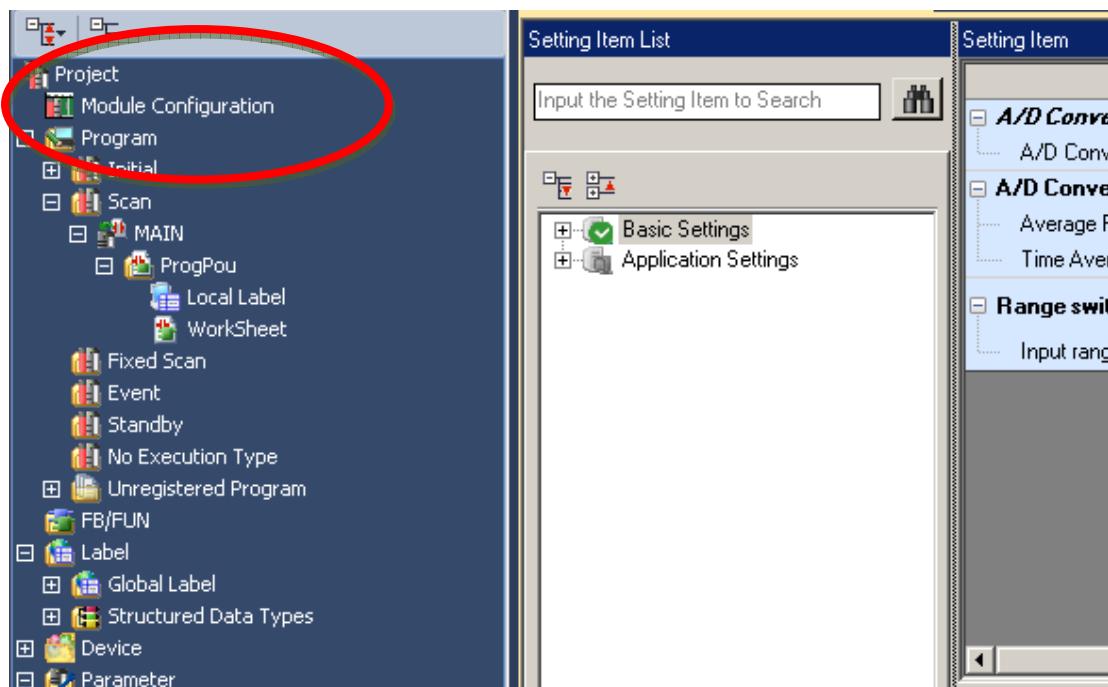
```
OUT(SM412, Y17);
```

Program powinien mrugać wyjściem Y17 zgodnie z zegarem wewnętrznym 1Hz.

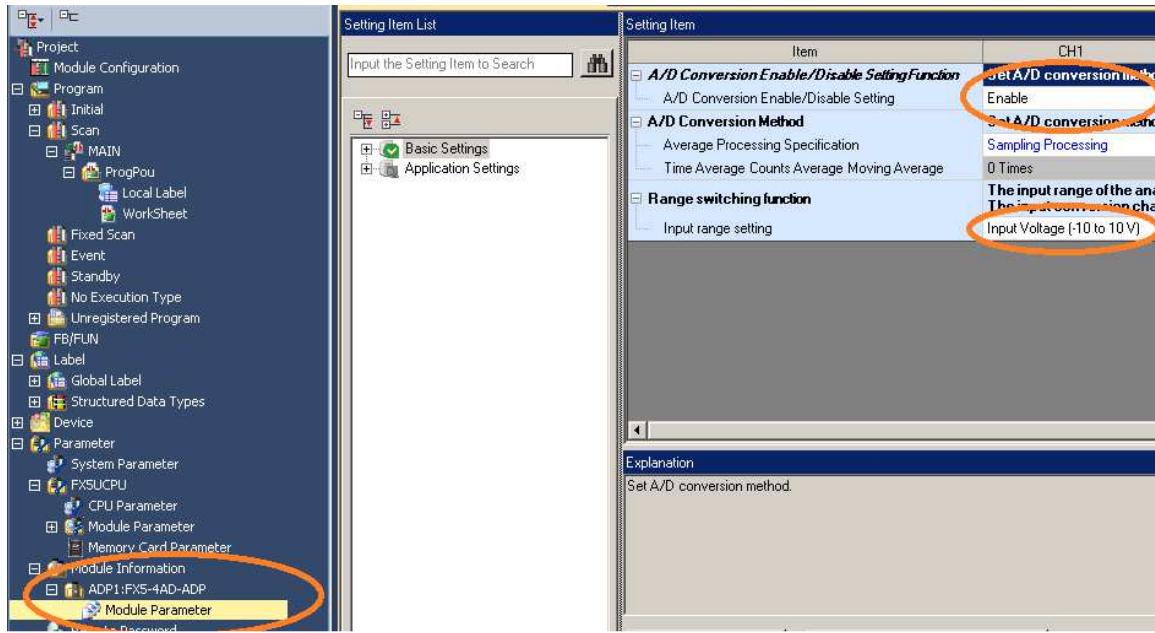
Później ten program posłuży do stworzenia pierwszego powiązania systemu SCADA ze sterownikiem PLC. Po utworzeniu programu wstępnego należy go skompilować opcją Rebuild All a następnie wgrać do sterownika. Po wykonaniu restartu sterownika wyjście Y16 powinno się zapalić a wyjście Y17 powinno cyklicznie się zmieniać.

### 5.2.10 Wejścia analogowe – FX5-4AD-ADP

Konfigurację modułu analogowe rozpoczynamy od dodania go w sekcji Project -> Module Configuration. Po otworzeniu się karty należy z prawej strony z okienka Element Selection przeciągnąć moduł na jego właściwe miejsce. Okienko Element Selection można aktywować z górnego menu View -> Docking Window -> Element Selection. Po dodaniu modułu należy kliknąć prawym przyciskiem na jednostce CPU (okienko Module Configuration) i wybrać Parameter -> Fix. Po tej operacji moduł jest już prawidłowo dodany do projektu a jego właściwości można modyfikować z sekcji Navigation -> Project -> Parameter -> Module Information.

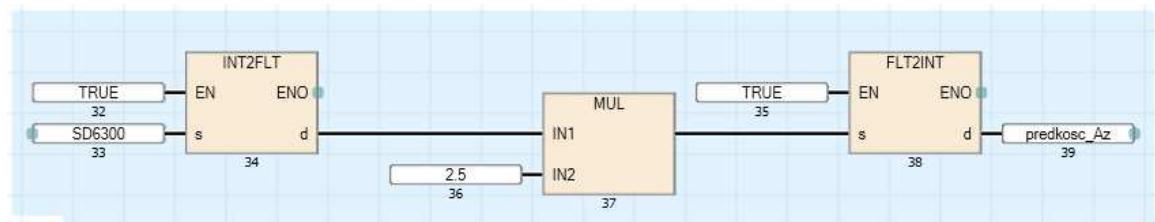


Następnie należy ustawić odpowiednie parametry danego wejścia. Prawidłowa konfiguracja dla kanału pierwszego została przedstawiona poniżej. Należy włączyć odpowiedni kanał przetwarzania analogowo cyfrowego i właściwy zakres napięcia wejściowego -10V do +10V.



Przykład programu PLC do odczytu wartości cyfrowej z przetwornika C/A został przedstawiony poniżej. Rejestr specjalny sterownika PLC o nazwie SD6300 jest przypisany do kanału pierwszego, modułu pierwszego, po lewej stronie sterownika PLC. Dokładniejsze opisy rejestrów można odszukać w dokumentacji lub w programie E-Manual Viewer wpisując np. frazę SD6300.

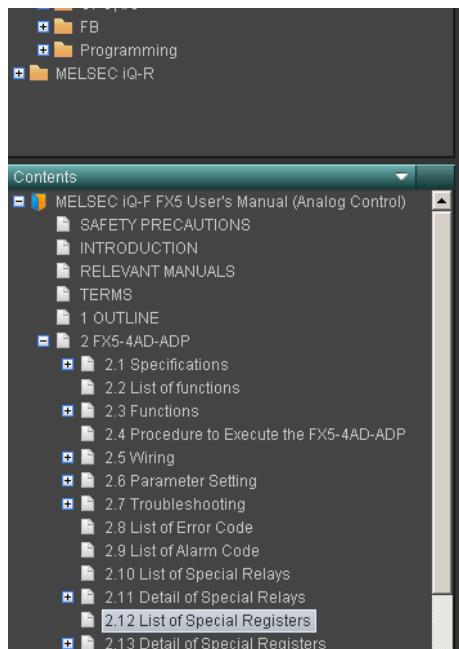
### FBD:



### ST:

```
predkosc_Az := REAL_TO_INT( 2.5 * INT_TO_REAL(SD6300) );
```

## Wyciąg z programu E-Manual Viewer.



The special registers list for the 1st FX5-4AD-ADP module is shown below.

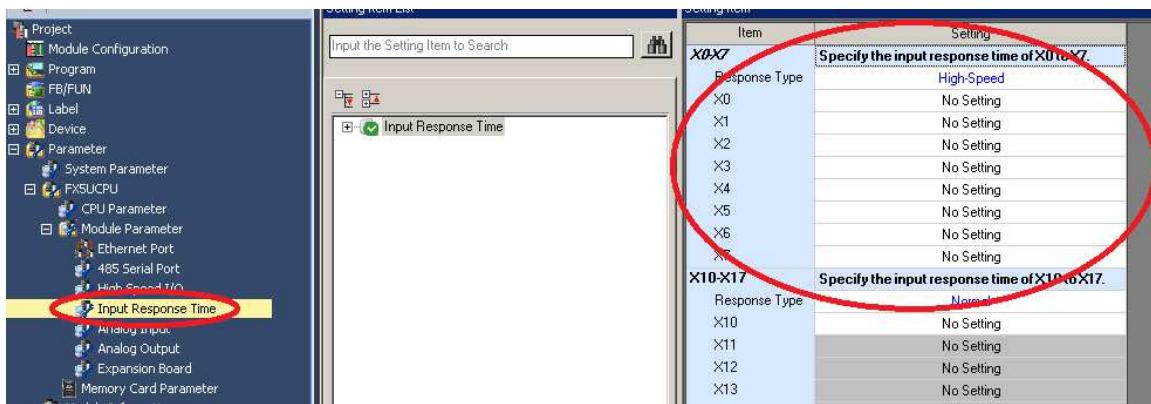
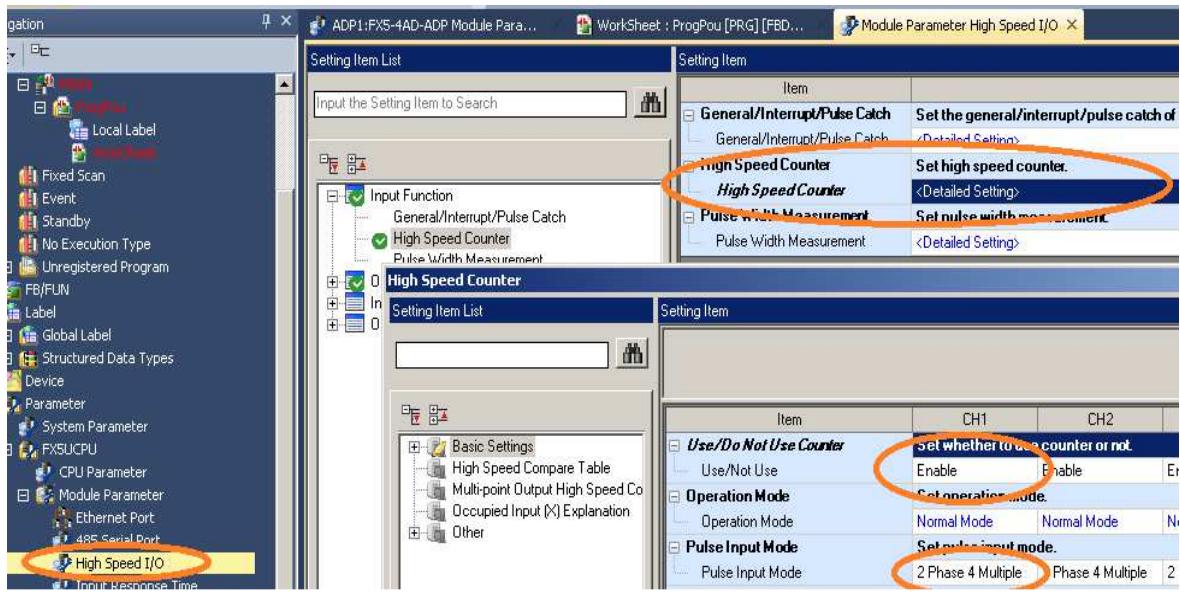
Special registers				Name
CH1	CH2	CH3	CH4	
SD6300	SD6340	SD6380	SD6420	Digital output value
SD6301	SD6341	SD6381	SD6421	Digital operation value
SD6302	SD6342	SD6382	SD6422	Analog input value monitor
SD6303	SD6343	SD6383	SD6423	Average processing specify
SD6304	SD6344	SD6384	SD6424	Time Average/Count Average/Moving Average setting
SD6305	SD6345	SD6385	SD6425	Input range setting
SD6306	SD6346	SD6386	SD6426	Maximum value
SD6307	SD6347	SD6387	SD6427	Minimum value
SD6308	SD6348	SD6388	SD6428	Scaling upper limit value
SD6309	SD6349	SD6389	SD6429	Scaling lower limit value
SD6310	SD6350	SD6390	SD6430	Conversion value shift amount
SD6311	SD6351	SD6391	SD6431	Process alarm upper upper limit value
SD6312	SD6352	SD6392	SD6432	Process alarm upper lower limit value
SD6313	SD6353	SD6393	SD6433	Process alarm lower upper limit value
SD6314	SD6354	SD6394	SD6434	Process alarm lower lower limit value
SD6315	SD6355	SD6395	SD6435	Rate alarm upper limit value
SD6316	SD6356	SD6396	SD6436	Rate alarm lower limit value
SD6317	SD6357	SD6397	SD6437	Rate alarm warning detection period setting
SD6322	SD6362	SD6402	SD6442	Convergence detection upper limit value

### 5.2.11 Wejście licznikowe – pomiar pozycji z enkodera inkrementalnego

Wejście licznikowe jest fizycznym wejściem cyfrowym sterownika. Aby dane wejście pracowało, jako szybki licznik należy przeprowadzić konfigurację parametrów przedstawioną poniżej. Enkodery inkrementalne podłączone przy pomocy fazy A i B muszą zostać ustawione, jako „2 phase”. Jeżeli podłączona jest tylko jedna faza to wykorzystuje się opcję „1 phase”. Oczywiście w pierwszym przypadku możliwe jest sprzętowe zliczanie pozycji w obie strony, natomiast w drugim przypadku zliczanie sprzętowe możliwe jest tylko w jednym kierunku.

Istotnym parametrem jest wyłączenie filtrów wejściowych, aby uzyskać prawidłowe pomiary. Wynika to z szybkości wysyłanych impulsów przez enkoder. Przy zbyt dużych częstotliwościach filtry mogą wyciąć prawidłowe informacje o pozycji.

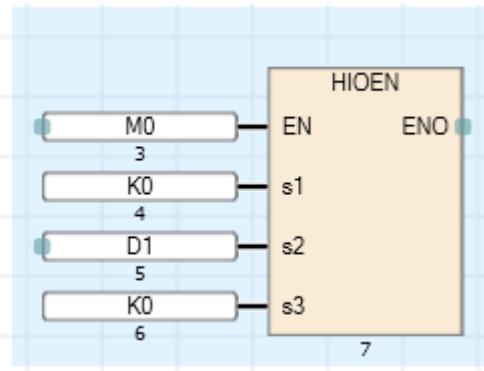
Do aktywacji zliczania należy w programie PLC umieścić odpowiednio skonfigurowaną instrukcję HIOEN. Wejście EN podłączone zostało do bitu aktywującego M0. Argument s1 wybiera funkcję licznika szybkiego. Argument s2 wybiera bitowo, które kanały mają zostać uruchomione: przykładowo, jeżeli chcemy użyć tylko kanału pierwszego to do rejestru D0 należy wpisać wartość 1; jeżeli chcemy użyć dwóch pierwszych kanałów to należy wpisać  $2+1 = 3$ . Argument s3 służy do deaktywacji kanałów licznika. Argument ten może pozostać z wartością zerową.



The special registers list for the 1st FX5-4AD-ADP module is shown below.

Special registers				Name
CH1	CH2	CH3	CH4	
SD6300	SD6340	SD6380	SD6420	Digital output value
SD6301	SD6341	SD6381	SD6421	Digital operation value
SD6302	SD6342	SD6382	SD6422	Analog input value monitor
SD6303	SD6343	SD6383	SD6423	Average processing specify
SD6304	SD6344	SD6384	SD6424	Time Average/Count Average/Moving Average setting
SD6305	SD6345	SD6385	SD6425	Input range setting
SD6306	SD6346	SD6386	SD6426	Maximum value
SD6307	SD6347	SD6387	SD6427	Minimum value
SD6308	SD6348	SD6388	SD6428	Scaling upper limit value
SD6309	SD6349	SD6389	SD6429	Scaling lower limit value
SD6310	SD6350	SD6390	SD6430	Conversion value shift amount
SD6311	SD6351	SD6391	SD6431	Process alarm upper upper limit value
SD6312	SD6352	SD6392	SD6432	Process alarm upper lower limit value
SD6313	SD6353	SD6393	SD6433	Process alarm lower upper limit value
SD6314	SD6354	SD6394	SD6434	Process alarm lower lower limit value
SD6315	SD6355	SD6395	SD6435	Rate alarm upper limit value
SD6316	SD6356	SD6396	SD6436	Rate alarm lower limit value
SD6317	SD6357	SD6397	SD6437	Rate alarm warning detection period setting
SD6322	SD6362	SD6402	SD6442	Convergence detection upper limit value

## FBD:



## ST:

```
MOV(TRUE, K1, D0);  
HIOEN(M0, K0, D0, K0);  
HIOEN(TRUE, K0, K1, K0);
```

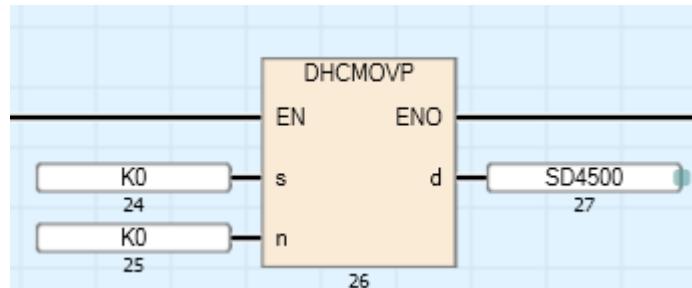
Aby odczytywać wartość aktualną licznika należy odszukać w dokumentacji, który rejestr specjalny jest powiązany z kanałem pierwszym – w naszym przypadku będzie to SD4500.

Category	
Engineering Software	
MELSEC iQ-F	
Analog input/output	
FX5U	
FX5UC	
CPU, I/O	
FB	
Programming	
MELSEC iQ-R	
Contents	
16 DEVICE/LABEL ACCESS SERVICE PROCEDURE	
17 RAS FUNCTIONS	
18 SECURITY FUNCTIONS	
19 HIGH-SPEED INPUT/OUTPUT FUNCTION	
19.1 High-speed Counter Function	
High-speed counter function overview	
High-speed counter function execution parameters	
High-speed counter specifications	
Assignment for high-speed counters	
High-speed counter parameters	
High-speed counter (normal mode)	
SD4500	High-speed counter current value (CH1)
SD4501	
SD4502	High-speed counter maximum value (CH1)
SD4503	
SD4504	High-speed counter minimum value (CH1)
SD4505	
SD4506	High-speed counter pulse density (CH1)
SD4507	
SD4508	High-speed counter rotational speed (CH1)
SD4509	
SD4510	High-speed counter preset control switch (CH1)
SD4511	Not used
SD4512	High-speed counter preset value (CH1)
SD4513	
SD4514	High-speed counter ring length (CH1)
SD4515	
SD4516	High-speed counter measurement unit time (CH1)
SD4517	
SD4518	High-speed counter number of pulses per rotation (CH1)
SD4519	
SD4520 to SD4529	Not used

Poniższy przykład programu pokazuje, w jaki sposób można wykonać referencję (bazowanie) liczniki – procedura wpisuje zadaną wartość do rejestru licznika – w omawianym przypadku będzie to wpisanie wartości 0 do rejestru kanału 1 licznika SD4500. Procedurę tą należy wykonać po ustawieniu elementu wykonawczego w miejscu bazowym. W przypadku zestawu TCRANE należy wykorzystać krańcówki sprzętowe do wykonania bazowania. W tym celu należy zachować szczególną ostrożność w pisaniu aplikacji, ponieważ element

jeżdżący powinien bezzwłocznie się zatrzymać po najechaniu na taką krańcówkę i dopiero później powinna być wykonana procedura zerowania licznika.

**FBD:**



**ST:**

```
DHCMOV P(M55, 0, 0, SD4500);
```

### 5.2.12 Wejście licznikowe – pomiar częstotliwości

Konfiguracja wejścia licznikowego w trybie pomiaru częstotliwości jest analogiczna do powyższego ustawienia w trybie pomiaru pozycji. Poniżej przedstawiono odpowiednie ustawienie parametrów. Istotnym parametrem jest wyłączenie filtrów wejściowych, aby uzyskać prawidłowe pomiary.

Item	CH1	CH2
<b>Use/Do Not Use Counter</b>	<b>Set whether to use counter or not.</b>	
Use/Not Use	Enable	Enable
<b>Operation Mode</b>	<b>Set operation mode.</b>	
Operation Mode	Pulse Density Measurement Mode	Pulse Density Measurement Mode
<b>Pulse Input Mode</b>	<b>Set pulse input mode.</b>	
Pulse Input Mode	1-Phase 1 Input (S/W Up/Down Switch)	1-Phase 1 Input (S/W Up/Down Switch)
<b>Preset Input</b>	<b>Set preset input.</b>	
Preset Input Enable/Disable	Disable	Disable
Input logic	Positive Logic	Positive Logic
Input Comparison Enable/Disable	Disable	Disable
Control Switch	Rising	Rising
<b>Preset Value</b>	<b>Set preset value.</b>	
Preset Value	0	0
<b>Enable Input</b>	<b>Set enable input.</b>	
Enable Input Enable/Disable	Disable	Disable
Input logic	Positive Logic	Positive Logic
<b>Ring Length Setting</b>	<b>Set ring length.</b>	
Ring Length Enable/Disable	Disable	Disable
Ring Length		
<b>Measurement Unit Time</b>	<b>Set the measurement unit time (ms) for the pulse density measurement mode and rotation speed measurement mode.</b>	
Measurement Unit Time	100	100
<b>No. of Pulse per Rotation</b>	<b>Set the number of pulses per rotation when using the rotation</b>	
No. of Pulse per Rotation	1000	1000

Ważne jest ustawienie czasu pomiaru – to ustawienie jest związane z rozdzielcością pomiaru częstotliwości.

W programie sterownika należy umieścić instrukcję aktywacji pomiaru. Podobnie jak poprzednio wykorzystujemy funkcję HIOEN. Tym razem w argumencie s1 podajemy wartość 10 – funkcja pomiaru częstotliwości.

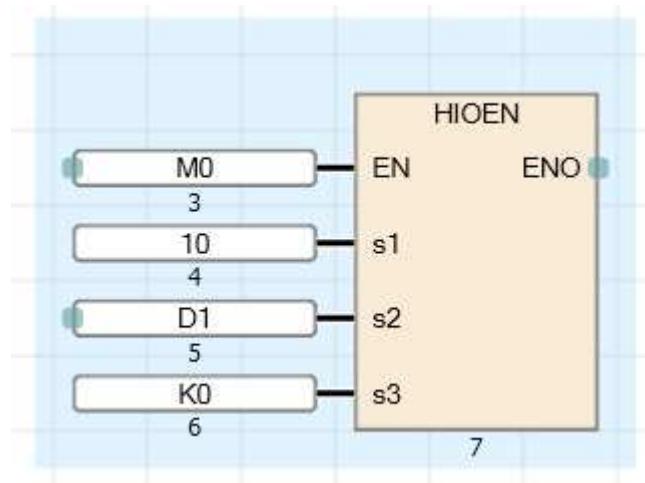
The screenshot shows the FX3U-4AD-4DA configuration software interface. On the left, the project tree is visible with 'Input Response Time' highlighted. In the center, a search bar is present. On the right, two tables show the configuration for X0-X7 and X10-X17. Both tables have columns for 'Item' and 'Setting'. The 'Setting' column contains the text 'Specify the input response time of X0 to X7.' and 'Specify the input response time of X10 to X17.' respectively. The 'Response Type' column for X0-X7 lists X0 through X7 with 'No Setting' under 'Setting'. The 'Response Type' column for X10-X17 lists X10 through X13 with 'No Setting' under 'Setting'.

Item	Setting
X0-X7	Specify the input response time of X0 to X7.
X0	No Setting
X1	No Setting
X2	No Setting
X3	No Setting
X4	No Setting
X5	No Setting
X6	No Setting
X7	No Setting

Item	Setting
X10-X17	Specify the input response time of X10 to X17.
X10	No Setting
X11	No Setting
X12	No Setting
X13	No Setting

**FBD:**



**ST:**

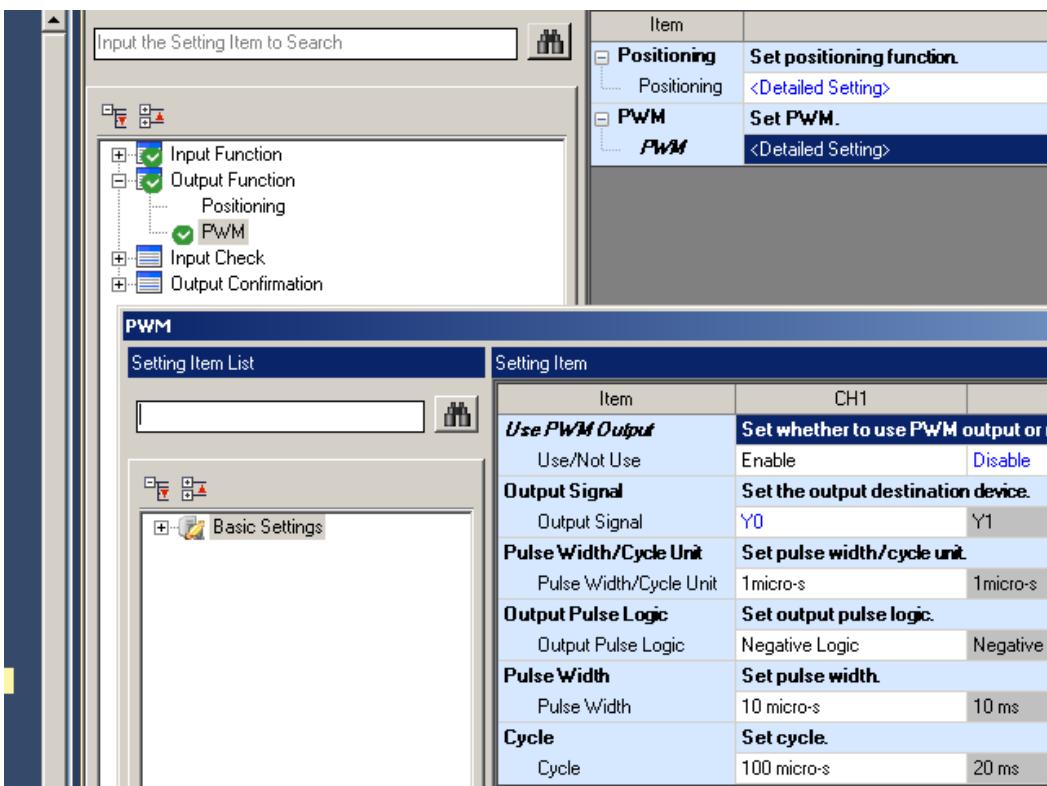
HIOEN(M0, 10, D1, K0);

Wartość mierzonej częstotliwości będzie w rejestrze SD4506 (i SD4507 – należy uważać, które wartości są zwracane jako 16 bit czy 32 bit).

SD4506	High-speed counter pulse density (CH1)	0 to 2147483647	0	R/W
SD4507				
SD4508	High-speed counter rotational speed (CH1)	0 to 2147483647	0	R/W
SD4509				
SD4510	High-speed counter preset control switch (CH1)	0: Rising edge 1: Falling edge 2: Both edges 3: Constant when ON	Parameter set value	R/W

### 5.2.13 Wyjście cyfrowe PWM

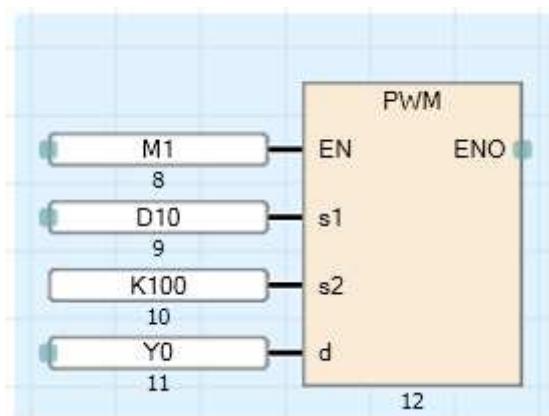
Ustawienie wyjścia PWM zaczynamy od parametrów.



Powyższe ustawienie dotyczy wyjścia Y0. Okres sygnału PWM wynosi 100us, czyli odpowiada to częstotliwości 10kHz. Przy takim ustawieniu możliwe wartości do sterowania wypełnieniem znajdują się od 1 do 100. Wpisanie wartości 0 do rejestru sterowania wypełnienie wprowadzi sterownik w błąd. Istotne jest również ustawienie prawidłowej logiki wyjściowej, aby później zwiększenie wypełnienia powodowało np. zwiększenie prędkości obrotowej silnika – będzie to naturalne podejście do dalszej regulacji. Jeżeli reakcja silnika będzie odwrotna należy zmienić opcję „Output Pulse Logic” na przeciwną.

W programie PLC należy dodać instrukcję PWM. Wejście EN aktywuje wyjście PWM. Rejestr D0(arg. S1) odpowiada za wartość wypełnienia. Argument s2 odpowiada za okres sygnału PWM i powinien być tak samo ustawiony jak parametr „Cycle”. Argument d powinien mieć przypisane właściwe wyjście sterownika z którego będzie generowany sygnał PWM.

### FBD:



**ST:**

PWM(M1, D10, K100, Y0) ;

### 5.2.14 Socket Communication – wysyłanie danych do MATLAB

Konfiguracja parametrów komunikacji została przedstawiona w rozdziale 5.2.9. Proszę zwrócić szczególną uwagę na pozycję, w której została dodana komunikacja typu „Socket Communication”. W przedstawionym przykładzie jest to Connection No. 1. Ma to bezpośredni związek z trzecim parametrem funkcji SP\_SOCSND. Jego wartość musi odpowiadać wartości Connection No.

Kolejnym etapem jest dodanie programu w grupie SCAN o nazwie np. „Socket”. Następnie należy dodać w zmiennych lokalnych tego programu następujące zmienne.

#### Lokalne

	Label Name	Data Type	
1	Sent	Bit	...
2	Not_Sent	Bit	...
3	Auto_Send	Bit	...

W zmiennych globalnych należy dodać poniższe zmienne do przechowywania łańcucha znaków. Proszę zwrócić uwagę, czy podane w przykładzie rejesty sterownika nie będą się pokrywały z już używanymi rejestrami w programie. W takiej sytuacji należy oczywiście zmienić przypisane urządzenia z tabelki poniżej.

#### Globalne

	Label Name	Data Type	Class	Assign (Device/Label)
1	Send_String	String(32)	VAR_GLOBAL	D3001
2	Value_String	String(32)	VAR_GLOBAL	D3500
3	Temp_String	String(32)	VAR_GLOBAL	D3700
4	String_Len	Word [Signed]	VAR_GLOBAL	D3000
5	Send_START	Bit	VAR_GLOBAL	
6	Trigger_Send	Bit	VAR_GLOBAL	

W programie wykonywanym cyklicznie (SCAN) należy wpisać program, który przygotowuje ramkę łańcucha znaków do wysłania przez Ethernet do MATLAB. Flaga Send\_START ustawiona na „1” spowoduje wysłanie łańcucha znaków do MATLAB. WAŻNE, aby w pierwszej kolejności uruchomić skrypt MATLAB, który inicjuje połączenie ze sterownikiem i potwierdza to obecnością bitu SD10680.1 (dla urządzenia z Connection No. = 2, odpowiednio bit SD10680.0 dla urządzenia z Connection No. =1).

//Generacja tekstu do wysłania przez socket communication

// Otwarcie portu wykonuje się automatycznie po uruchomieniu skryptu MATLAB

```

// Pojawia sie wowczas flaga SD10680.1 – dla drugiego połączenia z ustawien
Ethernet

// Wlaczenie automatycznego wysylania
SET(Auto_Send AND SD10680.1 AND LDP(TRUE, SM412), Send_START);

IF Send_START THEN

    Temp_String := 'Y=';

    Temp_String := CONCAT(Temp_String, REAL_TO_STRING(y_proces));

    Temp_String := CONCAT(Temp_String, ';U=');

    Temp_String := CONCAT(Temp_String, REAL_TO_STRING(u_proces));

    Temp_String := CONCAT(Temp_String, ';Y_ZAD=');

    Temp_String := CONCAT(Temp_String, REAL_TO_STRING(yz_proces));

    Temp_String := CONCAT(Temp_String, '$L');

    Send_String := Temp_String;

    //Dlugosc tekstu
    String_Len := LEN(Send_String);

    Trigger_Send := 1;
    Send_START := 0;

END_IF;

```

Komunikacja odbywa się przy użyciu funkcji SP\_SOCSND, której przykładową implementację przedstawiono poniżej. Fragment programu musi być umieszczony w grupie programów typu SCAN. Na początku resetowane są znaczniki sukcesu wysłania lub porażki. Następnie właściwa funkcja, która jest uruchomiona pod wpływem zbocza narastającego sygnału `Trigger_Send` i obecności sygnału SD10680.1, który mówi o prawidłowym połączeniu z serwerem (inicjowane po uruchomieniu skryptu w MATLAB). Wynik operacji wysłania jest umieszczony w bitach pamięci M300 i M301.

```

RST(LDP(TRUE, Trigger_Send), sent);
RST(LDP(TRUE, Trigger_Send), not_sent);

                                            //bit 1 Conn 2  //Conn
2
SP_SOCSND(LDP(TRUE, Trigger_Send) AND SD10680.1, 'U0', K2, D3200, String_Len,
M300 );

```

```

SET( LDP(TRUE, M300) AND NOT M301, sent);
SET( LDP(TRUE, M300) AND M301, not_sent);

IF M300 AND NOT M301 THEN
    Trigger_Send := 0;
END_IF;

```

W programie PLC wysyłanie będzie realizowane cyklicznie zgodnie z taktowaniem zegara na bicie SM412 – można to oczywiście dostosować do swoich potrzeb przyśpieszając lub opóźniając wysyłanie. Aby uruchomić cykliczne wysyłanie należy ustawić na „1” flagę **Auto\_Send**. Wysyłanie będzie możliwe dopiero po nawiązaniu komunikacji z serwerem, która musi zostać zainicjowana po stronie środowiska MATLAB. Przykładowa realizacja skryptu MATLAB do nawiązania połączenia i następnie cyklicznego rysowania danych na wykresie została przedstawiona poniżej.

### Skrypt MATLAB 2017

```

%%Socket Communication demo script
delete(instrfindall)
pause(2);

close all;
clear all;

t = tcpip('192.168.127.250', 4000, 'NetworkRole', 'client');

t.OutputBufferSize = 3000;
t.InputBufferSize = 3000;

fopen(t);
u = [];
y = [];
yz = [];

figure(1);
while (length(y) < 100)
    if (t.BytesAvailable ~= 0)
        temp = fscanf(t);
        %disp(temp);
        eval(temp);
        u = [u; U];
        y = [y; Y];
        yz = [yz; Y_ZAD];

        subplot(2,1,1);
        plot(y);
    end
end

```

```

title('Wyjście i zadana');
xlabel('iteracja');
hold on;
plot(yz);
hold off;

subplot(2,1,2);
stairs(u);
title('Sterowanie');
xlabel('iteracja');

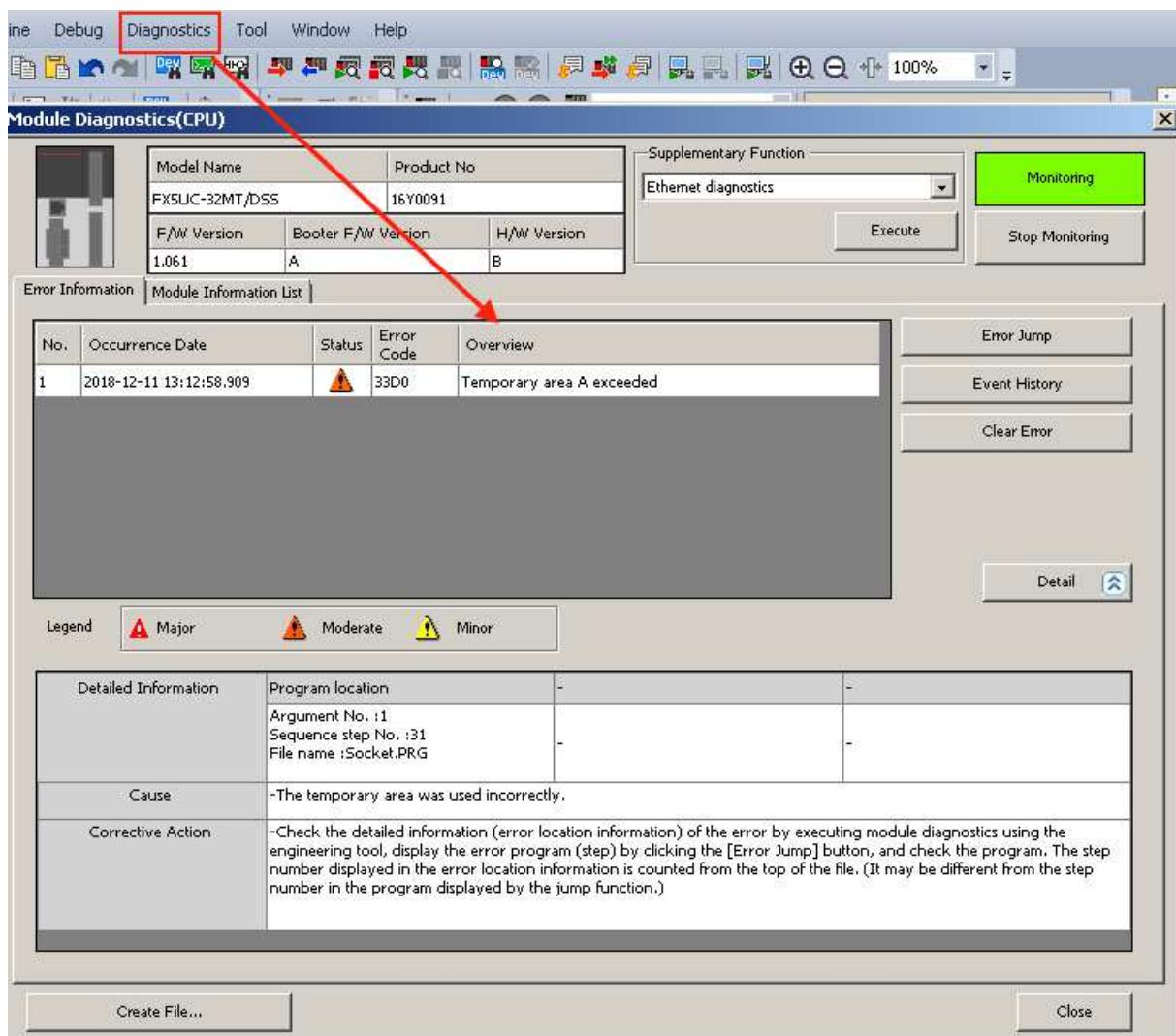
drawnow
end
pause(0.05);
end

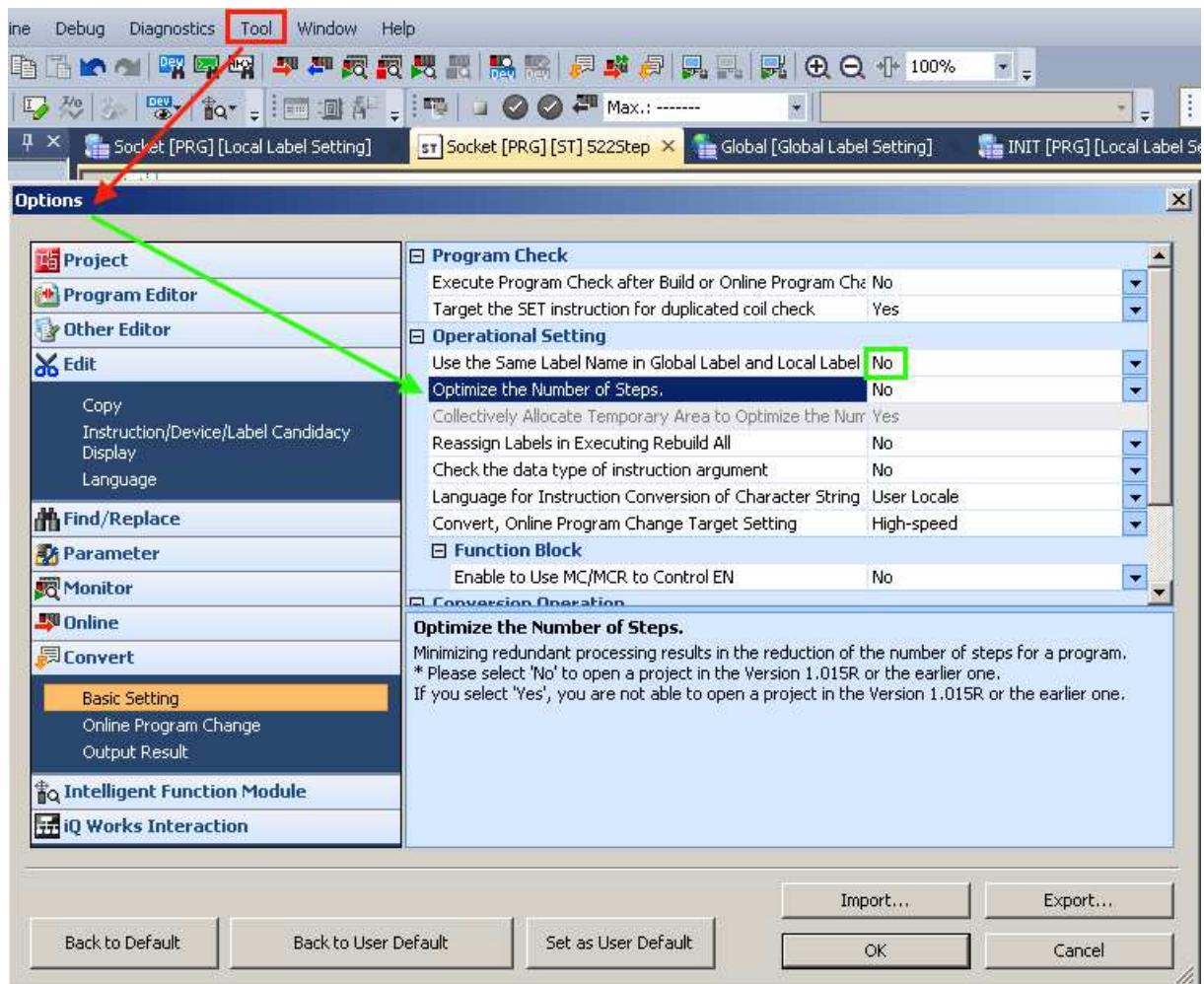
fclose(t);
delete(t);
clear t;

```

#### UWAGA:

Jeżeli po dodaniu fragmentu programu z instrukcją „CONCAT” pojawi się błąd sterownika jak na rysunku poniżej to należy przejść do ustawień kompilatora Tool -> Options i następnie zgodnie z kolejnym rysunkiem wybrać opcję bez optymalizacji kodu wynikowego.





### 5.2.15 Definicja tablicy typu float

W pierwszej kolejności definiujemy zmienną dwuwymiarową tablica\_MP. Pole Data Type uzupełniamy zgodnie z naszymi wymaganiami, co do wielkości tablicy. Możemy przydzielić odpowiedni obszar pamięci na przechowywanie danych w polu Assign.

	Label Name	Data Type	Class	Assign (Device/Label)
1	tablica_MP	FLOAT [Single Precision](0..49,0..49)	VAR_GLOBAL	D2500
2				

Poniżej przedstawiono przykładowe użycie zmiennej tablicowej w kodzie języka ST. Podajemy nazwę zmiennej a następnie w nawiasie kwadratowym musimy podać kolejno numer wiersza i numer kolumny oddzielone przecinkiem.

```

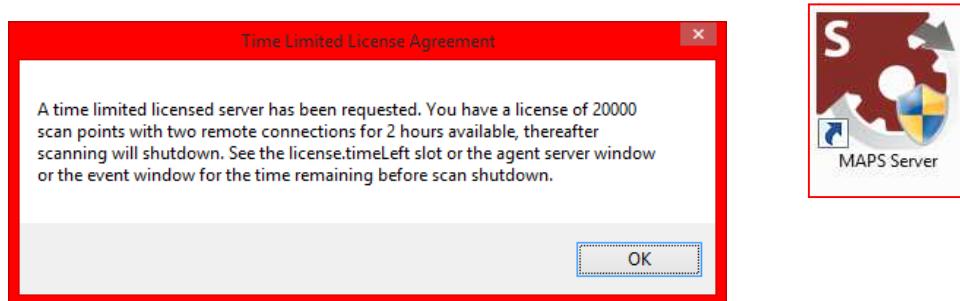
tablica_MP[0, 0] := 1.23;

FOR licznik := 0 TO 49 BY 1 DO
    FOR licznik2 := 0 TO 49 BY 1 DO
        tablica_MP[licznik, licznik2] := tablica_MP[0, 0];
    
```

```
END_FOR;  
END_FOR;
```

### 5.3 Oprogramowanie MAPS Server

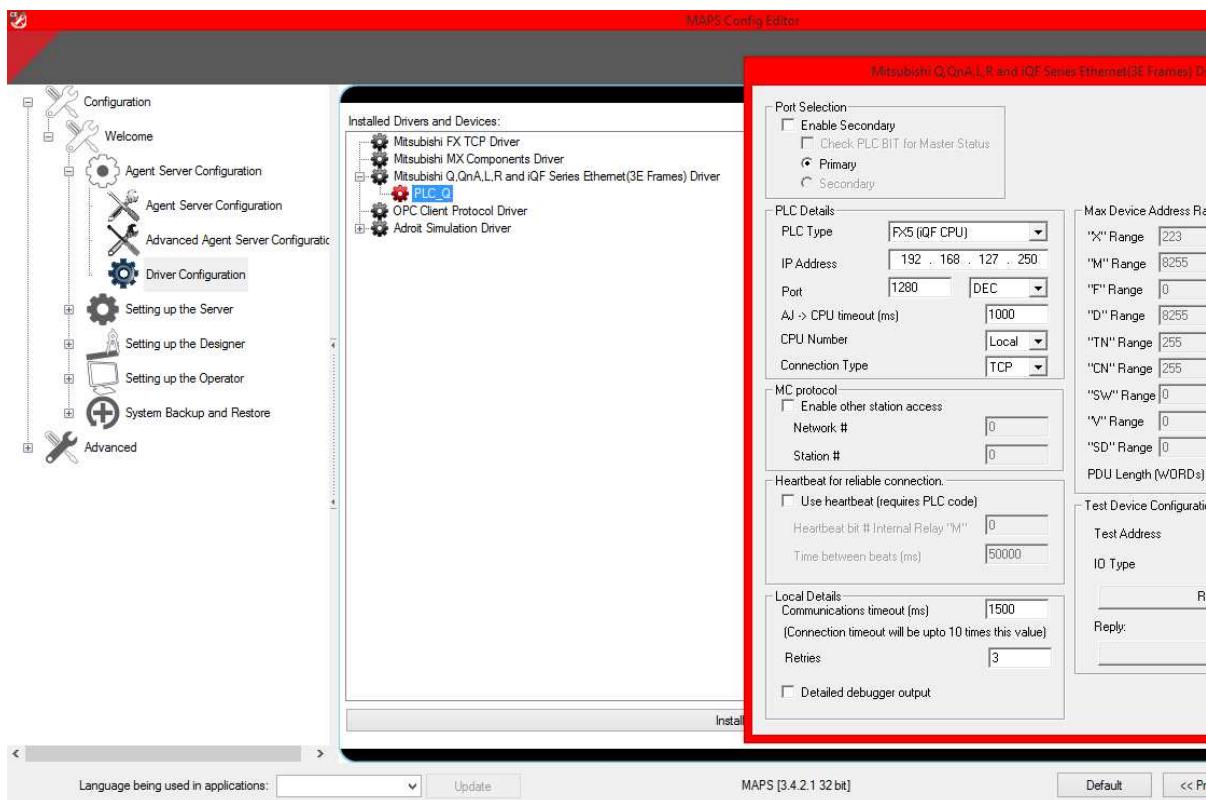
W celu uruchomienia środowiska MAPS, proszę uruchomić aplikację *MAPS Server* oraz poczekać na pełnąinicjalizację serwerów: Agent oraz MAPS. Proszę zatwierdzić komunikat o ograniczeniach licencyjnych:



Rys. 5.1: *Potwierdzenie komunikatu o warunkach licencyjnych*

### 5.4 Oprogramowanie MAPS Config Editor

W celu umożliwienia komunikacji ze sterownikiem należy skonfigurować Agent Server. Proszę uruchomić MAPS Config Editor a następnie skonfigurować driver wg podanych poniżej ustawień:



Rys. 5.2: Ustawienie drivera to połączenia ze sterownikiem FX5 na porcie 1280

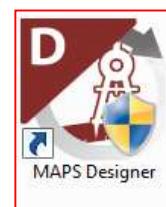
Aby potwierdzić prawidłowe skonfigurowanie połączenia można skorzystać z opcji PING lub wykonać testowy odczyt wartości rejestru (Read Register). Jeżeli operacje zakończą się sukcesem można przejść do kolejnych kroków tworzenia systemu SCADA.

## 5.5 Oprogramowanie MAPS Designer

## 5.6 Tworzenie grafik operatorskich w środowisku MAPS

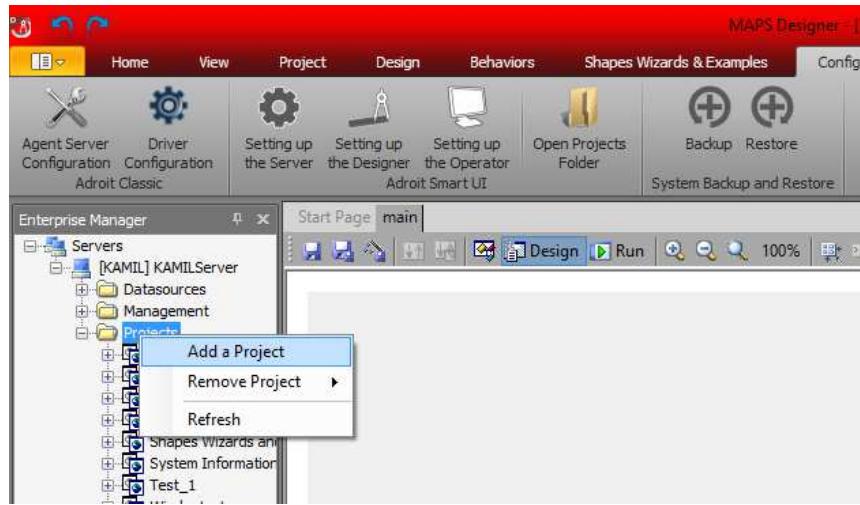
### 5.6.1 Opis programu MAPS Designer

Proszę uruchomić oprogramowanie MAPS Designer oraz zalogować się wg podanych informacji przez prowadzącego zajęcia. Login: User, Password: admin.



### 5.6.2 Tworzenie nowego projektu

Proszę utworzyć nowy projekt wybierając opcję „Add a Project” z zakładki Enterprise Manager:

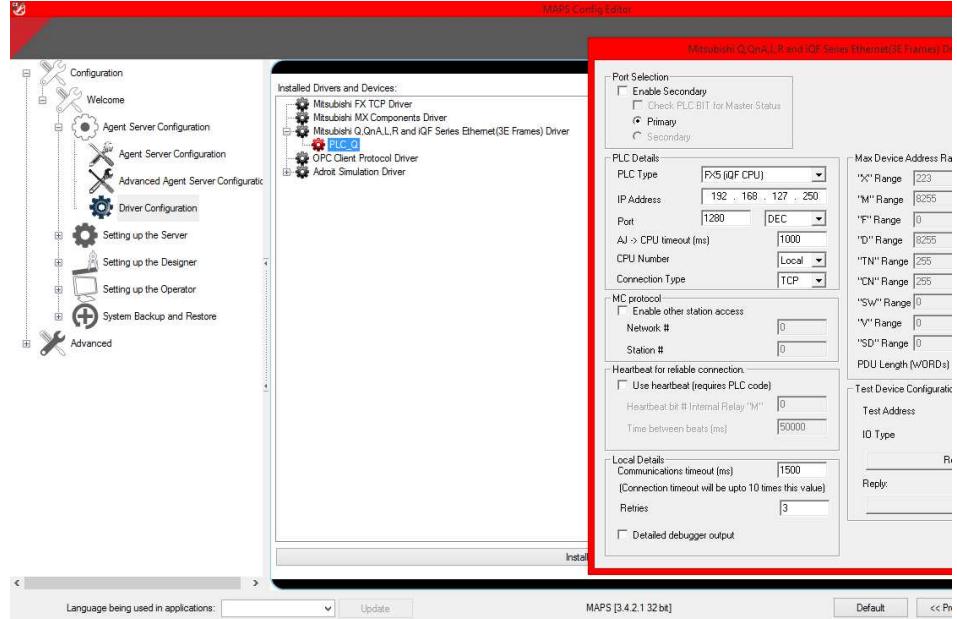
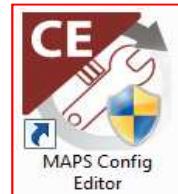


Rysunek 19 Utworzenie nowego projektu

Następnie proszę dodać nową formę graficzną korzystając z opcji „Create graphic form”.

### 5.6.3 Ustawienie komunikacji ze sterownikiem PLC

W celu umożliwienia komunikacji ze sterownikiem należy skonfigurować Agent Server. Proszę uruchomić MAPS Config Editor a następnie skonfigurować driver wg podanych poniżej ustawień:

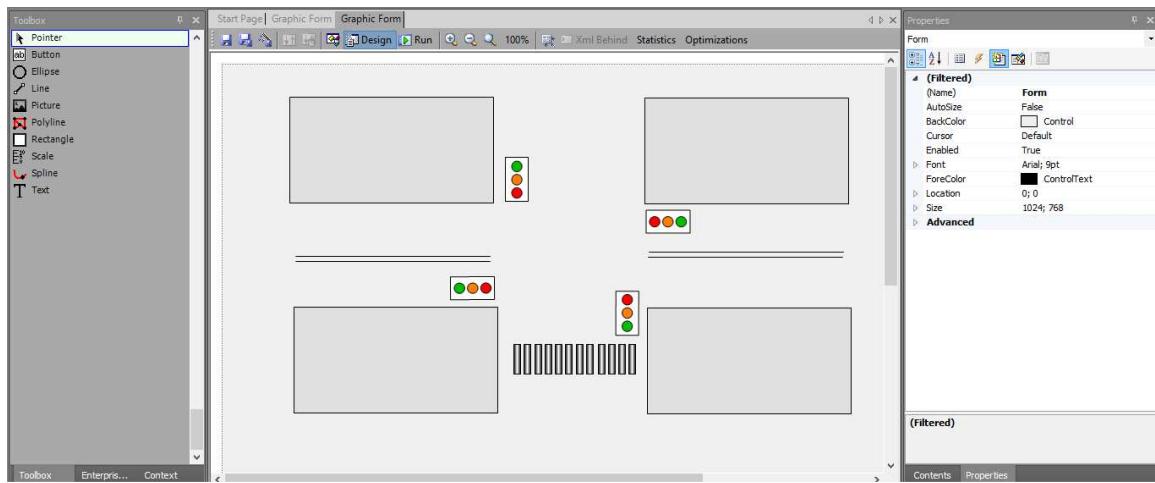


Rysunek 20 Ustawienie drivera do połączenia ze sterownikiem FX5 na porcie 1280

Aby potwierdzić prawidłowe skonfigurowanie połączenia można skorzystać z opcji PING lub wykonać testowy odczyt wartości rejestru (Read Register). Jeżeli operacje zakończą się sukcesem można przejść do kolejnych kroków tworzenia systemu SCADA.

## 5.6.4 Tworzenie grafik operatorskich

MAPS Designer dostarcza bibliotekę obiektów, które można wykorzystać do budowy wizualizacji procesu. Obiekty mogą być pasywne lub aktywne, w zależności od definicji zachowań. Domyslnie obiekty nie mają skonfigurowanych żadnych zachowań. Właściwości dodanego obiektu konfiguruje się w oknie „Properties”.



Rysunek 21 Przykład wizualizacji skrzyżowania, grafika zbudowana wyłącznie z podstawowych elementów graficznych t.j prostokąt, linia, okrąg.

---

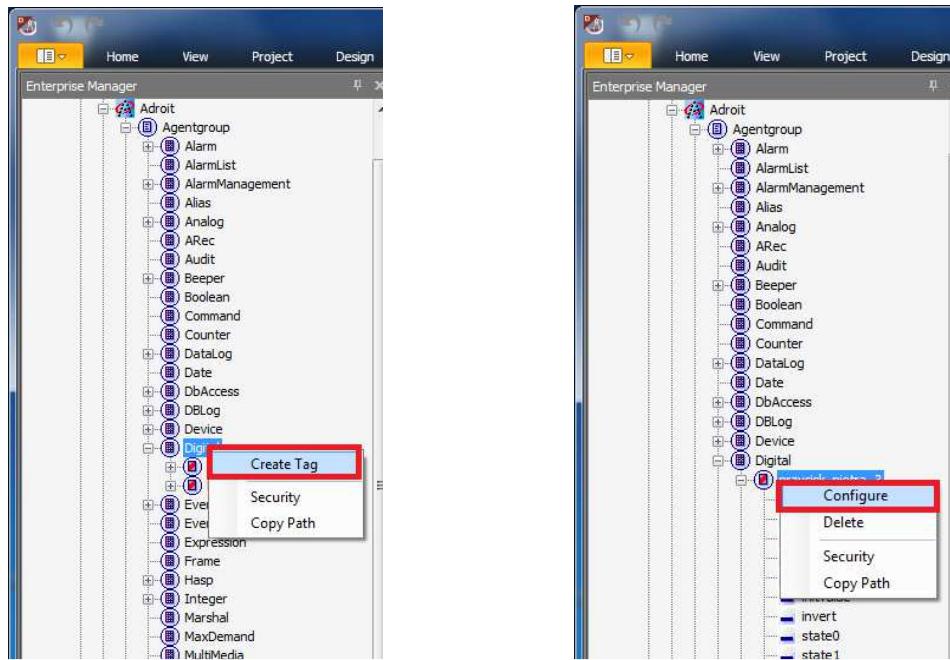
**Uwaga:** W celu przemieszczenia kilku elementów, należy nacisnąć przycisk Shift a następnie zaznaczyć obszar obiektów wykorzystując lewy przycisk myszy.

---

## 5.6.5 Definicja nowych agentów

Definicja zmiennych w grafikach opiera się na utworzeniu i konfiguracji Agentów odpowiednich typów. Mogą one być łącznikiem do obszaru pamięci sterownika dzięki możliwości skanowania punktu z sieci.

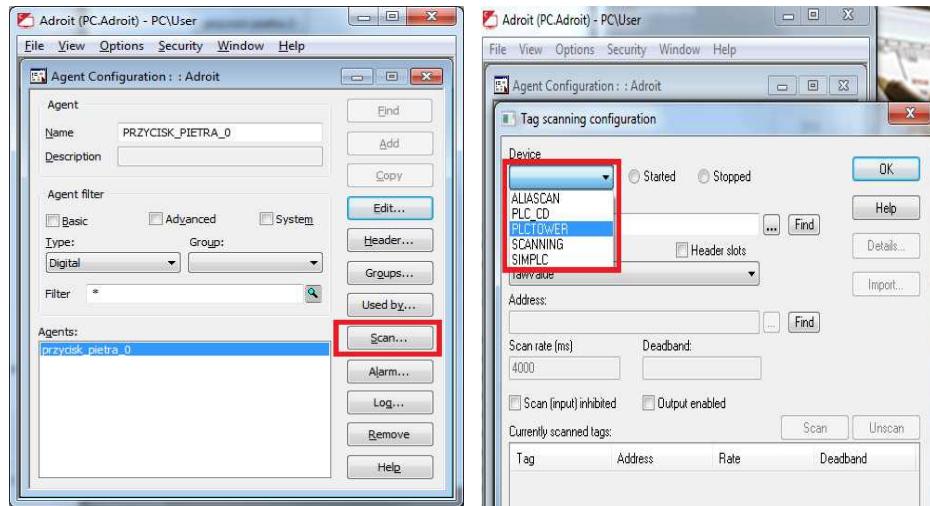
Skanowane powinny być tylko zmienne istotne w sterowaniu czy wizualizacji procesu.

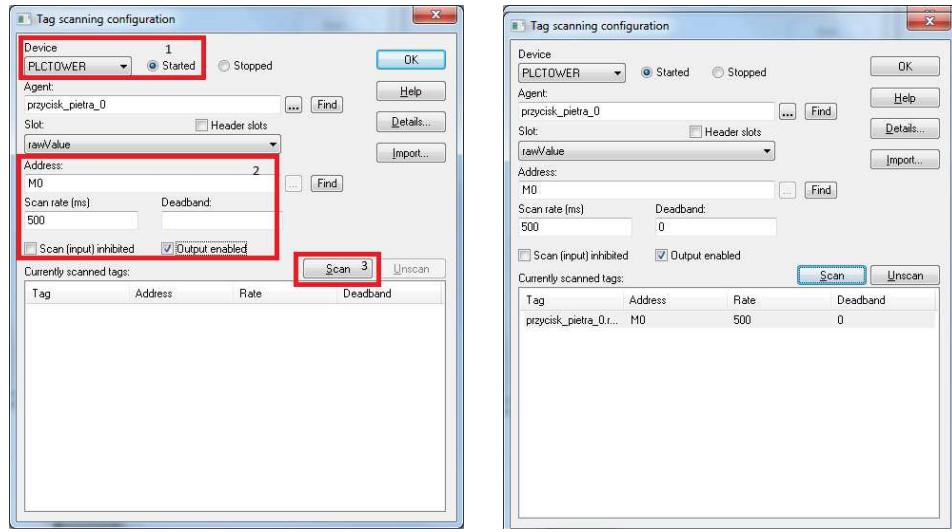


Rysunek 22 Tworzenie oraz wywołanie okna konfiguracji agentów

### 5.6.6 Skanowanie punktów ze sterownika PLC

Sekwencja do konfiguracji skanowania punktu binarnego:





Rys. 5.3: Włącznie skanowania punktu binarnego

Adresację zmiennych w pamięci sterownika, można sprawdzić w tabeli zmiennych globalnych (która została wcześniej wypełniona):

	Data Type	Class	Assign (Device/Label)	Initial Value
1	Bit	VAR_GLOBAL		
2	Bit	VAR_GLOBAL	M3	
3	Word [Signed]	VAR_GLOBAL	D1	
4	Word [Signed]	VAR_GLOBAL	D2	
5	Bit	VAR_GLOBAL	M2	
6	Bit	VAR_GLOBAL		
7	Bit	VAR_GLOBAL	M1	
8	Bit	VAR_GLOBAL		
9	Bit	VAR_GLOBAL	M0	
10	Bit	VAR_GLOBAL		
11	Bit	VAR_GLOBAL		
12	Bit	VAR_GLOBAL		
13				

Rys. 5.4: Adresacja wykorzystanych zmiennych globalnych w pamięci sterownika

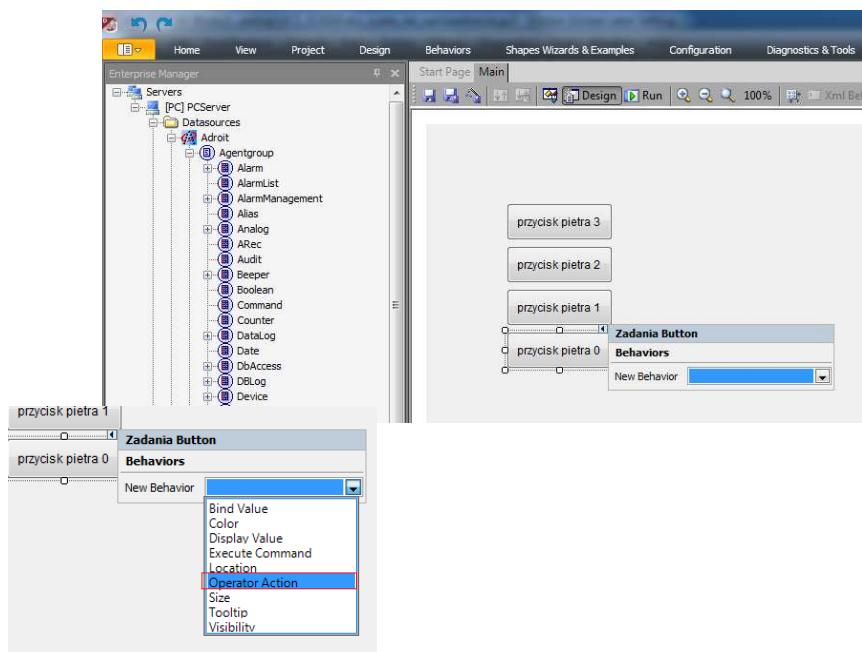
W tym miejscu należy użyć odpowiedniego sterownika, który został skonfigurowany. Następnie należy dodać do skanowania bit pamięci M0. Później do tego agenta dołączona zostanie przykładowa grafika zmieniająca kolor w zależności od stanu wyjścia. Opcja Output Enabled pozwala na zezwolenie sterowania zmienną z poziomu systemu SCADA. Jeśli zmienna skanowana będzie tylko w trybie odczytu, wtedy opcja Output Enabled może zostać niewłączona.

W trakcie ustawienia skanowania należy dobrać odpowiedni typ agenta dla skanowanej zmiennej sterownika. Do skanowania zmiennych typu Int, Word, Dword, Long, Float należy wykorzystać agent typu Analog. Adres musi zawierać odpowiedni Tag, który oznacza typ skanowanej zmiennej. Poniższa tabela zawiera zestawienie wspieranych typów danych skanowanych dla sterownika serii FX. **Przykładowo jeżeli chcemy skanować zmienną typu Float należy wpisać w polu adres: D500 F.**

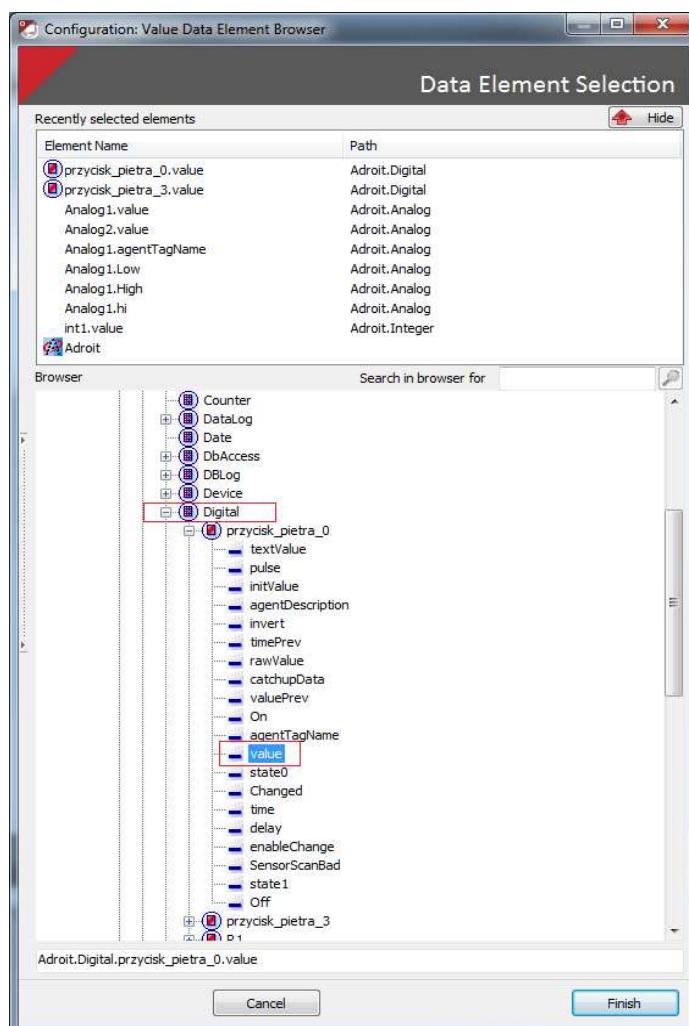
Device	prefix	Max Range	Notation	PLC Data Type	BOOLEAN	INTEGER	REAL	STRING
Inputs	X	337	OCT	discrete	X 0			
				Int 16		X0	X0	
				Word 16		X0 W	X0 W	
				dword 32		X0 D	X0 D	
				Long 32		X0 L	X0 L	
				Float 32		X0 F	X0 F	
Output relay	Y	337	OCT	discrete	Y 0			
Internal relay	M	8255	DEC	discrete	M 0			
Latch relay	L	0	DEC	discrete	L 0			
Annunciator	F	0	DEC	discrete	F 0			
Link relay	B	0	HEX	discrete	B 0			
Special link relay	SB	0	HEX	discrete	SB 0			
Edge relay	V	0	DEC	discrete	V 0			
Data register	D	8255	DEC	bit	D0.0			D 0 Sn
		8255	DEC	Int 16		D0	D0	
				Word 16		D0 W	D0 W	
				dword 32		D0 D	D0 D	
				Long 32		D0 L	D0 L	
				Float 32		D0 F	D0 F	
Link register	W	0	OCT	Int 16		W 0	W 0	
Timer (current value)	T or TN*	255	DEC	Int 16		TN 0	TN 0	
Retentive timer (current value)	ST	0	DEC	Int 16		ST 0	ST 0	
Counter (current value)	C or CN*	255	DEC	Int 16		CT 0	CN 0	
Special link reg.	SW	0	OCT	Int 16		SW 0	SW 0	
File register	R	0	DEC	bit	R0.0			R 0 Sn
		0	DEC	Int 16		R0	R0	
				Word 16		R0 W	R0 W	
				dword 32		R0 D	R0 D	
				Long 32		R0 L	R0 L	
				Float 32		R0 F	R0 F	
Extended File register	ZR	0	OCT	bit	ZR0.0			
		0	OCT	Int 16		ZR0	ZR0	
				Word 16		ZR0 W	ZR0 W	
				dword 32		ZR0 D	ZR0 D	
				Long 32		ZR0 L	ZR0 L	
				Float 32		ZR0 F	ZR0 F	

### 5.6.7 Definiowanie zachowań

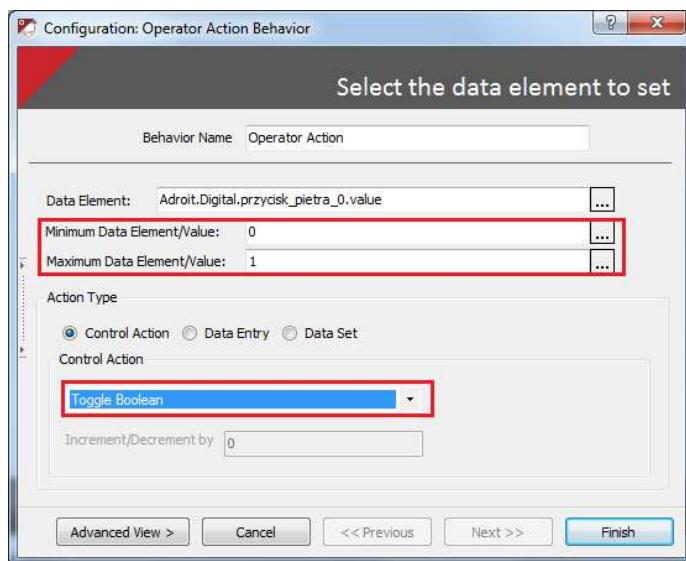
Po dodaniu elementu graficznego, możliwe jest zdefiniowanie jego zachowania (np. zmiana koloru diody jako reakcja na zmianę stanu krańcówki; zmiana wartości zmiennej binarnej po naciśnięciu przycisku przez operatora). Aby dodać zachowanie należy zaznaczyć element graficzny a następnie kliknąć lewym przyciskiem myszy w symbol strzałki. Następnie wybieramy typ zachowania z menu.



Rysunek 23 Dodawanie akcji/zachowania dla przycisku

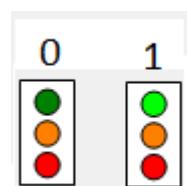
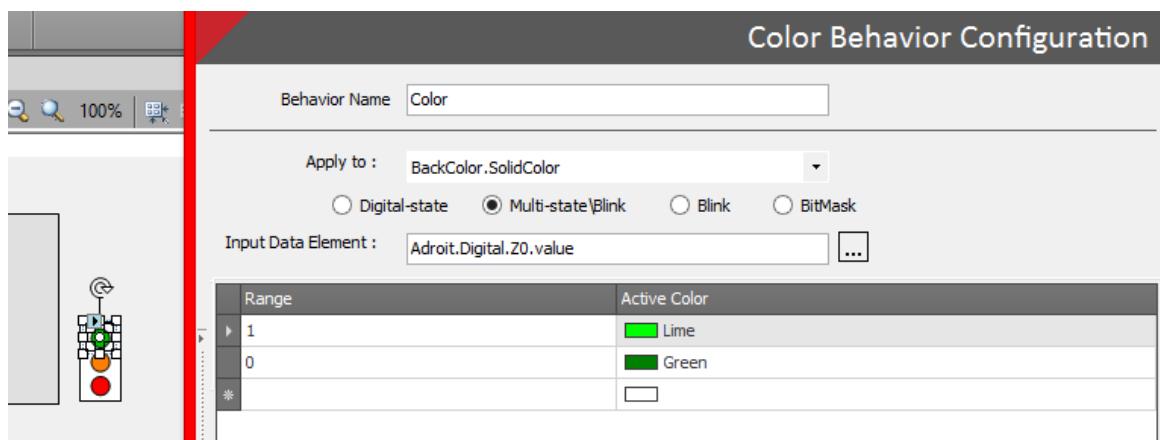


Rysunek 24 Wybór zmiennej do obsługi przycisku.



Rysunek 25 Konfiguracja zachowania przycisku po jego naciśnięciu.

Do wizualizacji stanu zapalonej/zgaszonej żarówki można wykorzystać element graficzny (okrąg) posiadający zachowanie typu „Color” ( kolor jasnozielony - stan aktywny, ciemnozielony - stan nieaktywny).



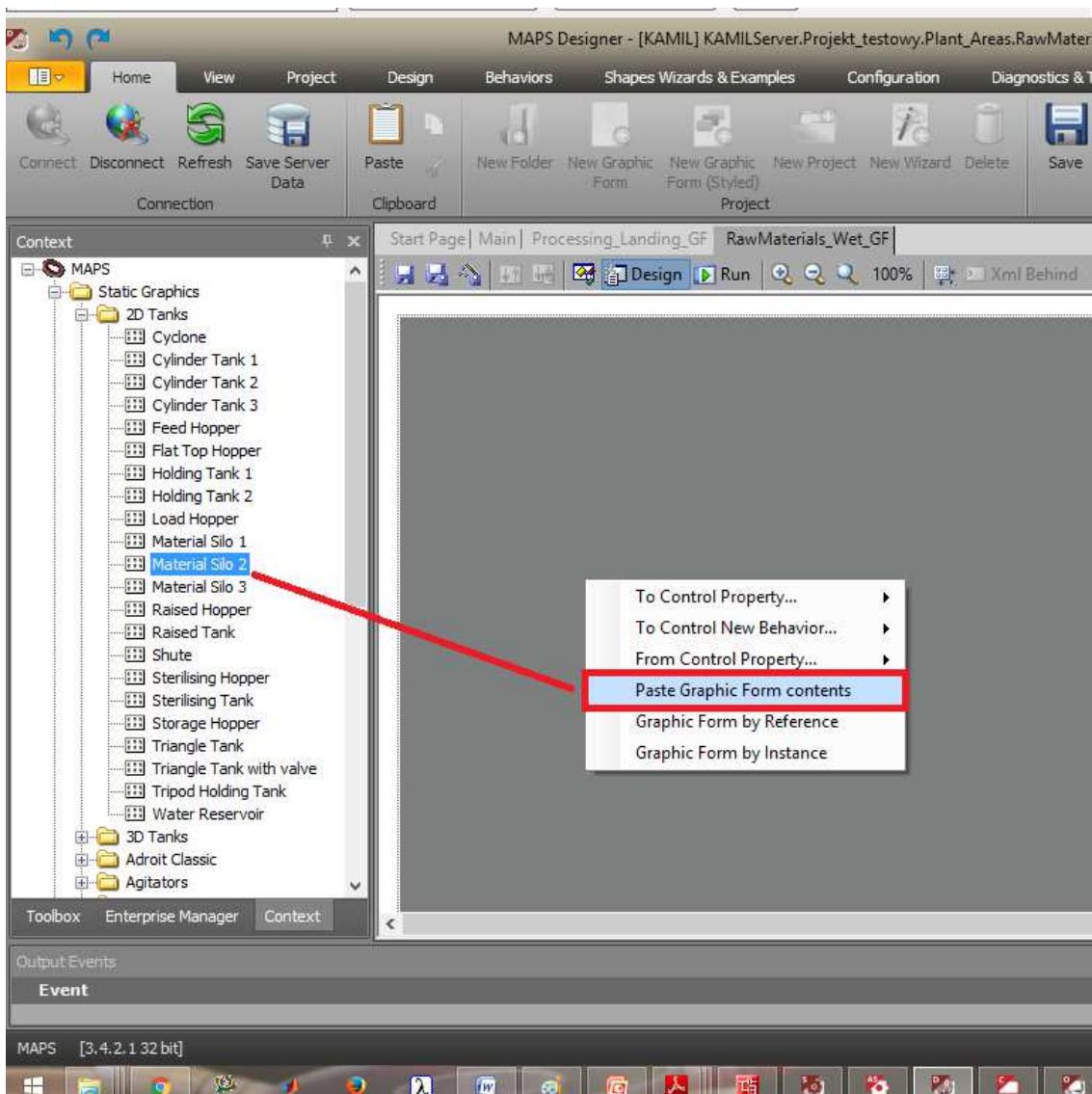
Rysunek 26 Sygnalizacja włączonego/wyłączonego światła zielonego (0,1 - wartość zmiennej sterującej )

## Przykład realizacji przycisku typu Momentary

Dzięki zaawansowanym ustawieniom z pola Advanced Setting danego przycisku możemy zrealizować dwie podstawowe funkcje przycisku: tryb Momentary – wciśnięcie przycisku wygeneruje tylko impuls stanu wysokiego i następnie wróci do stanu początkowego; tryb Alternate – każde wciśnięcie przycisku przełączy jego stan na przeciwny.

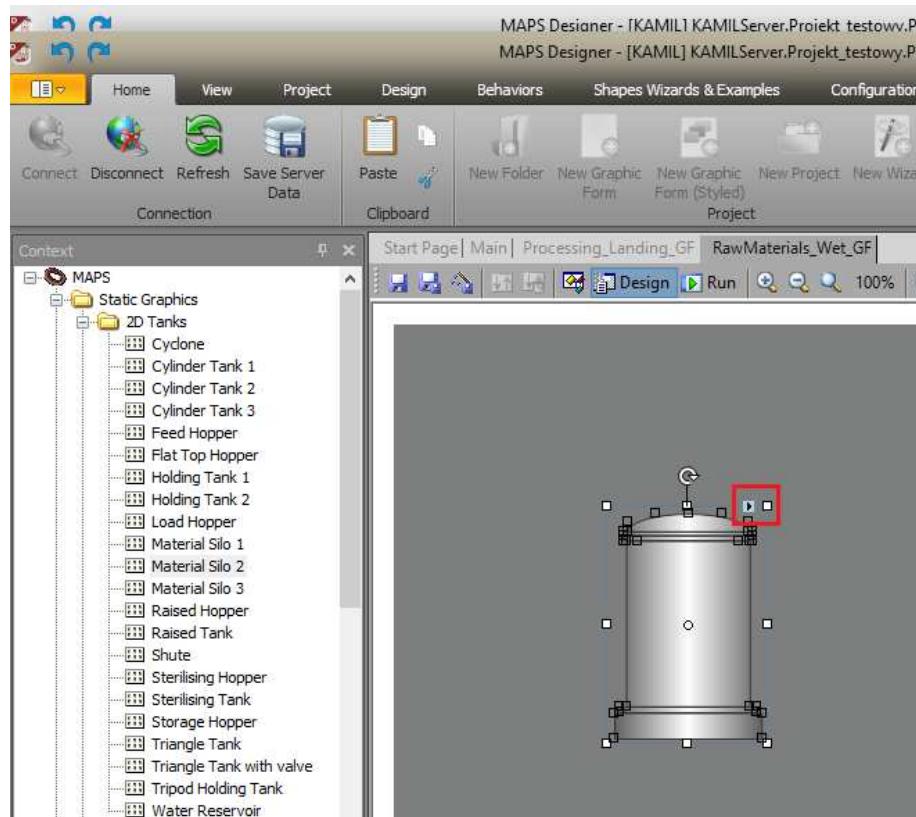
## Przykład wizualizacji zmiany stanu poziomu w zbiorniku

Wizualizacja stanu napełnienia zbiornika jest jednym z najczęściej wykorzystywanych animacji stosowanych w systemach SCADA. W celu wizualizacji stanu napełnienia zbiornika należy utworzyć nowy obiekt ze statycznej biblioteki:

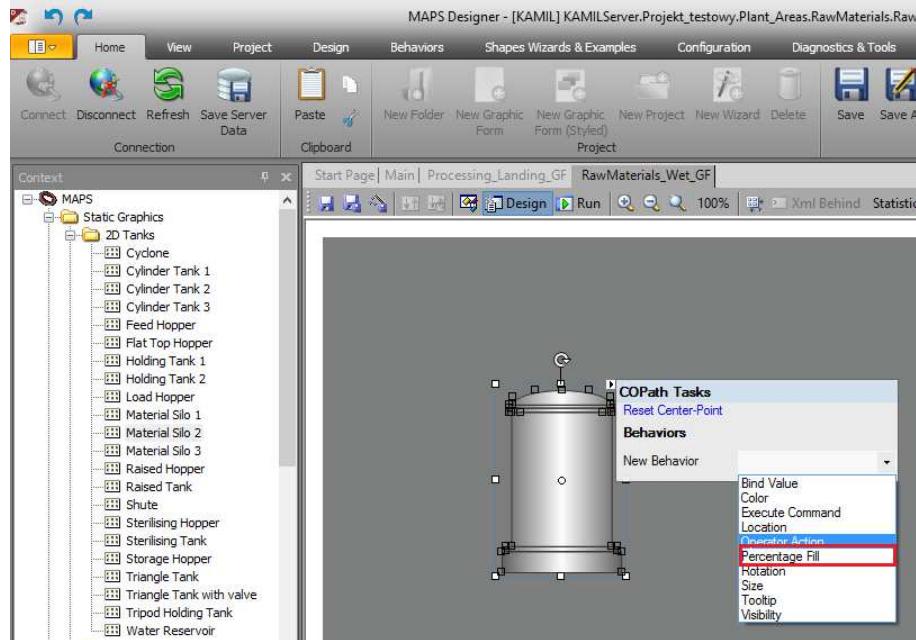


Rys. 5.5: Utworzenie nowego obiektu statycznego typu Material Silo 2

Po dodaniu nowego obiektu można utworzyć nowe zachowanie, które będzie prezentowało stan napełnienia zbiornika w zależności od wartości zmiennej procesowej. Zachowanie to dodawane z menu dostępnego po zaznaczeniu danego obiektu (rysunek poniżej).



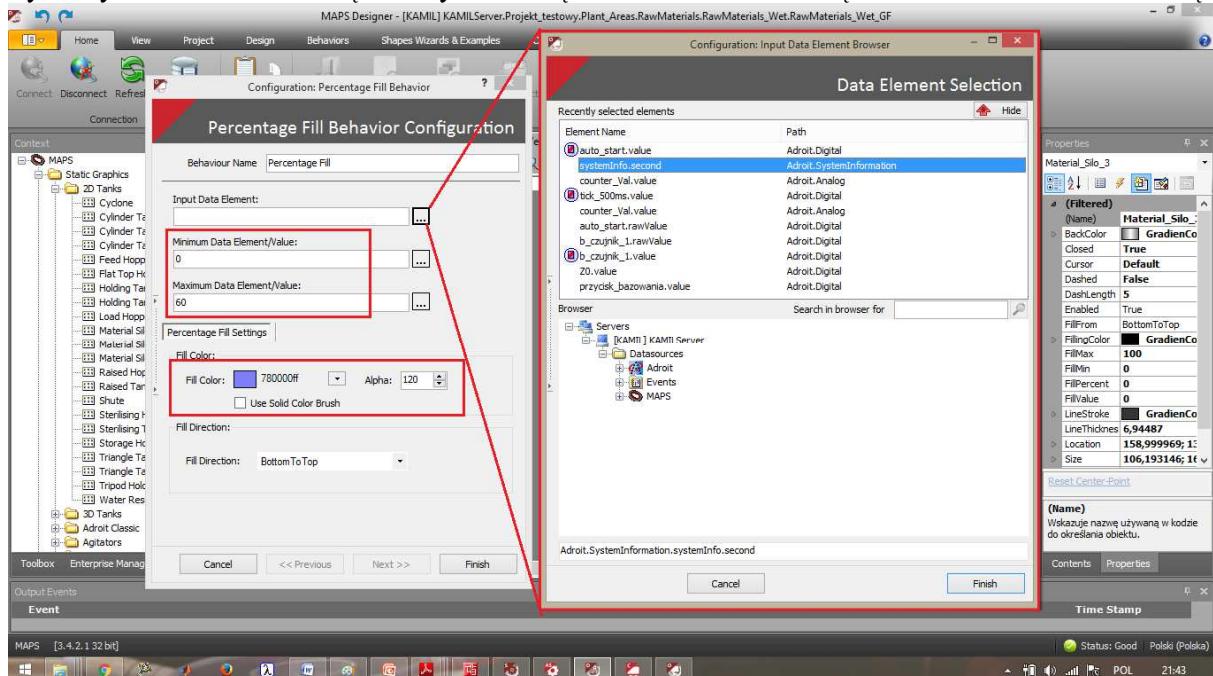
**Rys. 5.6: Lokalizacja okna do definicji zachowań obiektu**



**Rys. 5.7: Dodanie nowego zachowania typu Percentage Fill**

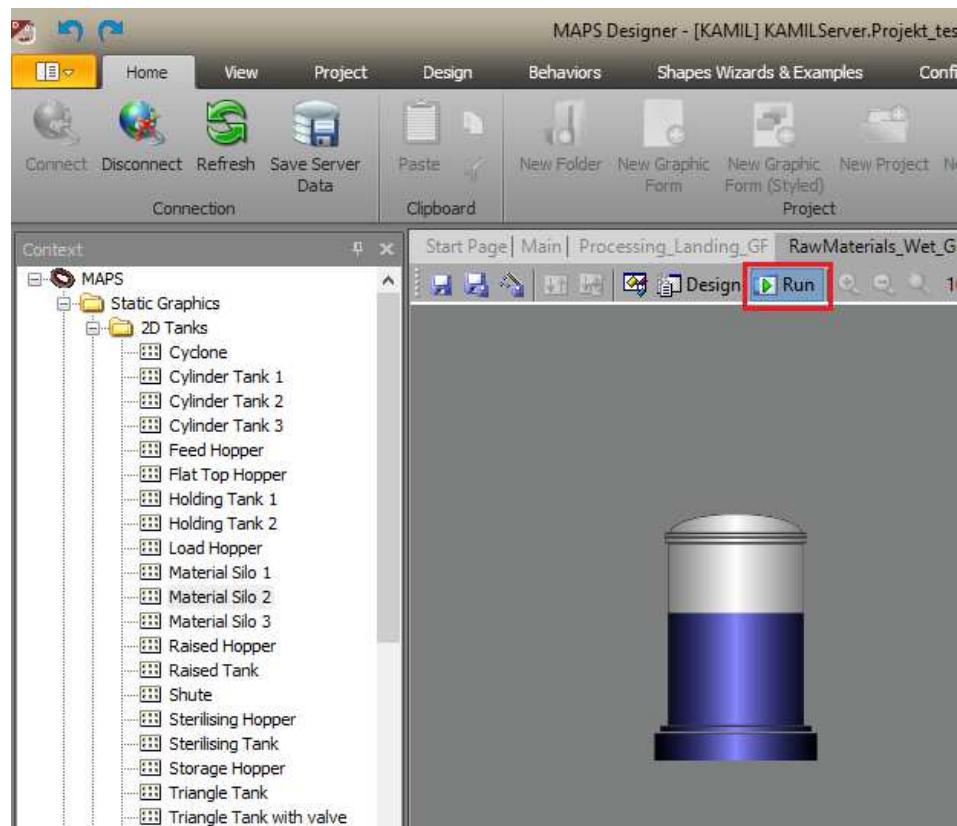
Kolejnym krokiem jest połączenie wizualizacji ze zmienną dostępną w bazie. Może to być zmienna skanowana z pamięci sterownika, jak również zmienna systemowa. W teście

wykorzystano zmienną systemową inkrementowaną co 1 sekundę.



Rys. 5.8: Przypisanie wartości odpowiedzialnej za wypełnienie zbiornika

Po zatwierdzeniu zmian należy wykonać test logiki. Dla zmiennej systemowej „*systemInfo.second*” z limitami 0-60 wypełnienie zbiornika odpowiada sekundzie czasu komputera na którym uruchomiony jest system MAPS.

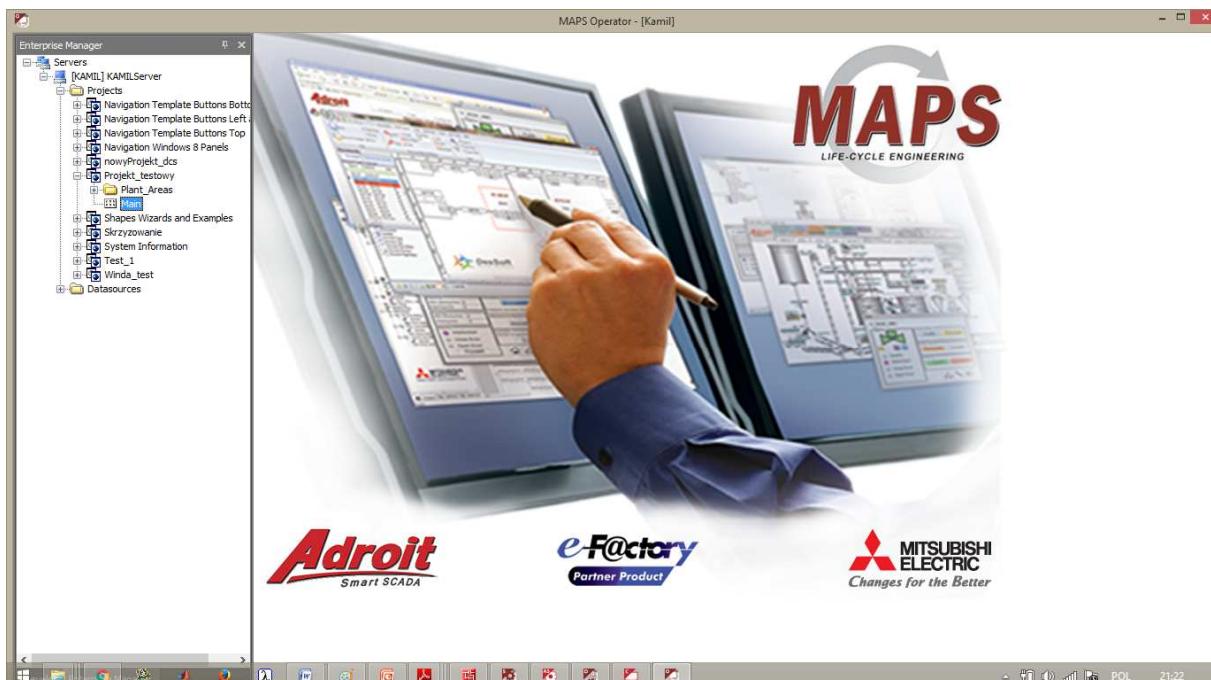


Rys. 5.9: Test działania wypełnienia zbiornika

## 5.7 Oprogramowanie MAPS Operator

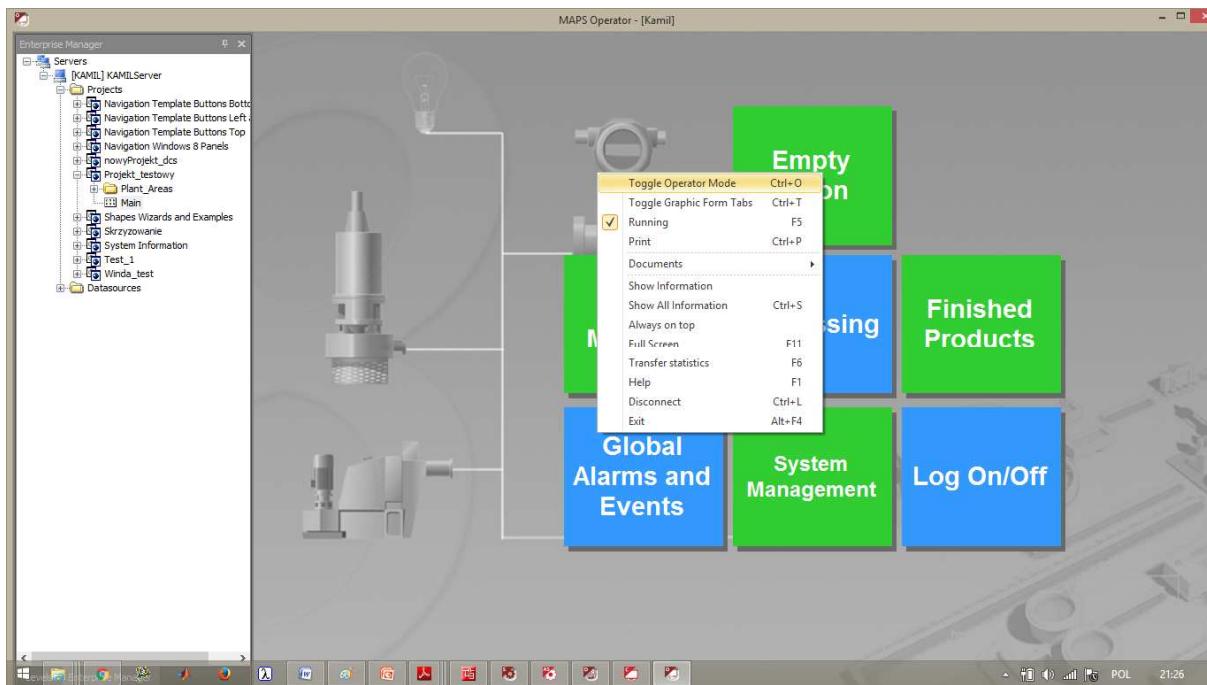
Aplikacja MAPS Operator przeznaczona jest do obsługi systemu z punktu widzenia operatora procesu. W trybie widoku operator nie ma możliwości modyfikacji logik, ponadto istnieje możliwość ustawienia uprawnień określających prawa dostępu do wybranych grafik.

Do korzystania z aplikacji MAPS Operator należy uruchomić nowąinstancję oraz zalogować się na konto operatora (w ramach laboratorium, można wykorzystać takie same dane logowania jak dla MAPS Designer).



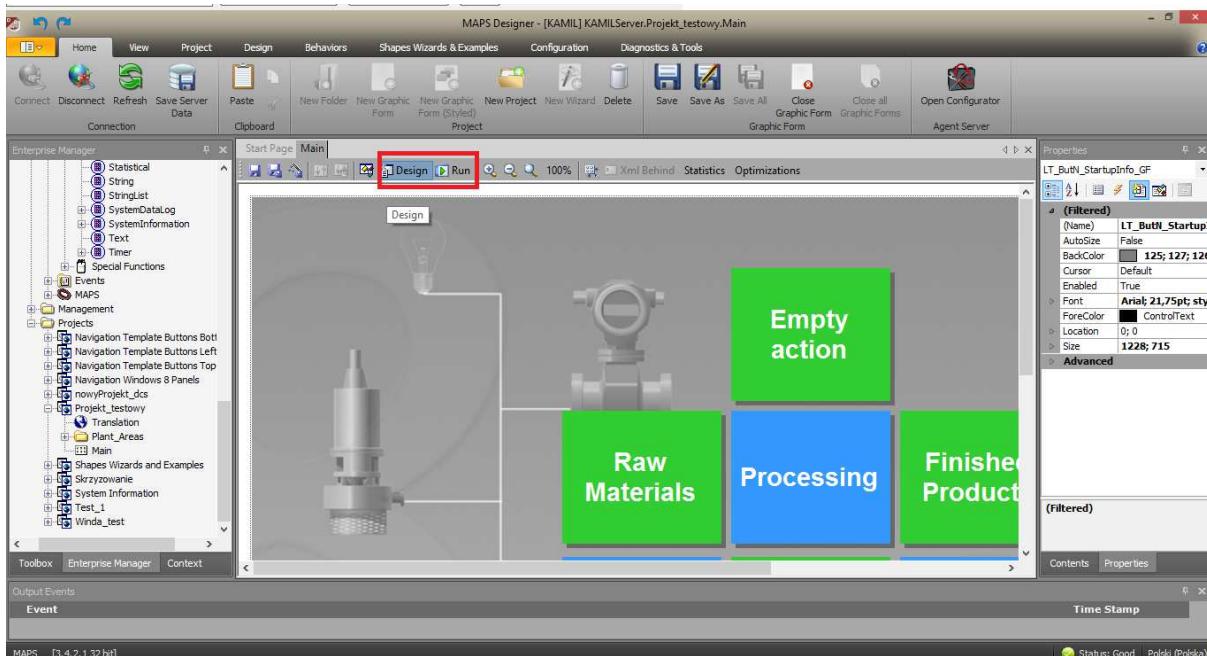
Rys. 5.10: Widok aplikacji MAPS Operator po zalogowaniu

W lewej części aplikacji widoczne jest drzewo projektów. Należy wybrać swój projekt i otworzyć główny arkusz. Można skorzystać z opcji dostępnych po naciśnięciu prawego przycisku myszy, np. w celu włączenia Widoku Operatora



Rys. 5.11: Włączenie widoku operatora

Podczas projektowania i implementacji arkuszy z wizualizacją można wykorzystać podgląd działania aplikacji z poziomu MAPS Designera, przełączając się pomiędzy stanem *Design a Run* (poniższy rysunek):



Rys. 5.12: Przełączanie widoku z poziomu MAPS Designer

## 5.8 Przykład 1 – logika sterowania binarna

### 5.8.1 Etap projektowania PLC

W rozdziale tym przedstawiona zostanie realizacja prostego automatu sekwencyjnego. Jego zastosowanie ma szerokie horyzonty z uwagi na łatwą implementację i naturalne odwzorowanie zadanego cyklu maszyny czy procesu. Każdy stan posiada warunki przejścia do kolejnego stanu a także operacje realizowane w danym stanie. Przechodzenie między kolejnymi stanami może być wykonane automatycznie lub można wybrać tryb krokowy działania automatu i wówczas przejście do kolejnych stanów są realizowane przez użytkownika. Naturalnie wybór kolejnego stanu jest realizowany w sposób programowy na podstawie warunków przejścia a użytkownik wyznacza tylko moment, w którym to przejście następuje.

Pierwszym krokiem do realizacji przykładu jest stworzenie nowych zmiennych i przydzielenie do nich odpowiednich obszarów pamięci. Zmienna Stan\_aktualny będzie pokazywała, w jakim aktualnie stanie się znajdujemy, zmienna Stan\_nastepny będzie pokazywać do jakiego stanu nastąpi przejście, bit Automat\_krokowy służy do wyboru normalnego trybu pracy lub trybu krokowego i ostatni bit Automat\_krok będzie służył do wyzwolenia przejścia automatu do kolejnego stanu.

The screenshot shows a software interface for PLC variable declaration. On the left, there is a navigation pane with a tree view under 'Label'. The 'Global Label' node is expanded, showing 'Wejścia', 'Wyjścia', 'Rejestry', 'Falowniki', 'M+Global', and 'Automat'. The 'Automat' node is highlighted with a yellow background. On the right, there is a table titled 'Label' with columns: 'Label Name', 'Data Type', and 'Assign (Device/Label)'. The table contains five rows:

	Label Name	Data Type	Assign (Device/Label)
1	Stan_aktualny	Word [Signed]	D100
2	Stan_nastepny	Word [Signed]	D101
3	Automat_krokowy	Bit	M50
4	Automat_krok	Bit	M51
5			...

Po deklaracji zmiennych należy przygotować program, który będzie oparty na instrukcji warunkowej CASE. Początkowy fragment pozwala na wybór automatu krokowego i obsługę przejść do kolejnych stanów. Następnie kolejne stany zostały przedstawione w postaci cyfr od 0 do n, wybieranych przy pomocy instrukcji CASE. Warunki przejść zostały zapisane jako instrukcje IF ... THEN. Każdy stan może zawierać instrukcje przypisania MOV, ustawienia SET, skasowania RST oraz inne. Należy tylko pamiętać, że instrukcje te będą realizowane tylko podczas wybranego danego stanu aktualnego. W szczególności należy pamiętać o tym, że jeżeli automat będzie w stanie 1 z poniższego przykładu to instrukcja MOV(**TRUE**,2,D610) będzie realizowana w każdym skanie procesora – oznacza to innymi słowami, że w każdym skanie do rejestru D610 będzie wpisywana wartość 2. Idąc dalej oznacza to, że wartości rejestrów D610 nie będziemy mogli zmienić z innego miejsca np. okienko Watch. Rozważmy jeszcze stan numer 2, gdzie instrukcja SET(**MEP(TRUE)**,M650) wykona się tylko jeden raz w momencie wejścia do stanu 2. Dzięki temu możemy ustawić bit M650 na 1 tylko w chwili przejścia do stanu 2 a w kolejnych skanach procesora można już zmienić bit na stan 0 pozostając dalej w stanie 2.

```
IF Automat_krokowy AND LDP(TRUE, Automat_krok) THEN  
    Stan_aktualny := Stan_nastepny;
```

```

ELSE
    IF NOT Automat_krokowy THEN
        Stan_aktualny := Stan_nastepny;
    END_IF;
END_IF;

IF Stan_aktualny = Stan_nastepny THEN
    CASE Stan_aktualny OF

        0 :
            IF M610 THEN
                Stan_nastepny:= 1;
                RST(TRUE, M610);
            END_IF;
            IF M618 THEN
                Stan_nastepny:= 3;
                RST(TRUE, M618);
            END_IF;

        1:
            IF M611 THEN
                Stan_nastepny:= 2;
                RST(TRUE, M611);
            END_IF;
            MOV(TRUE,2,D610);

        2:
            SET(MEP(TRUE),M650);
            IF M612 THEN
                Stan_nastepny:= 3;
                RST(TRUE, M612);
                RST(TRUE,M650);
            END_IF;

            IF M615 THEN

```

```

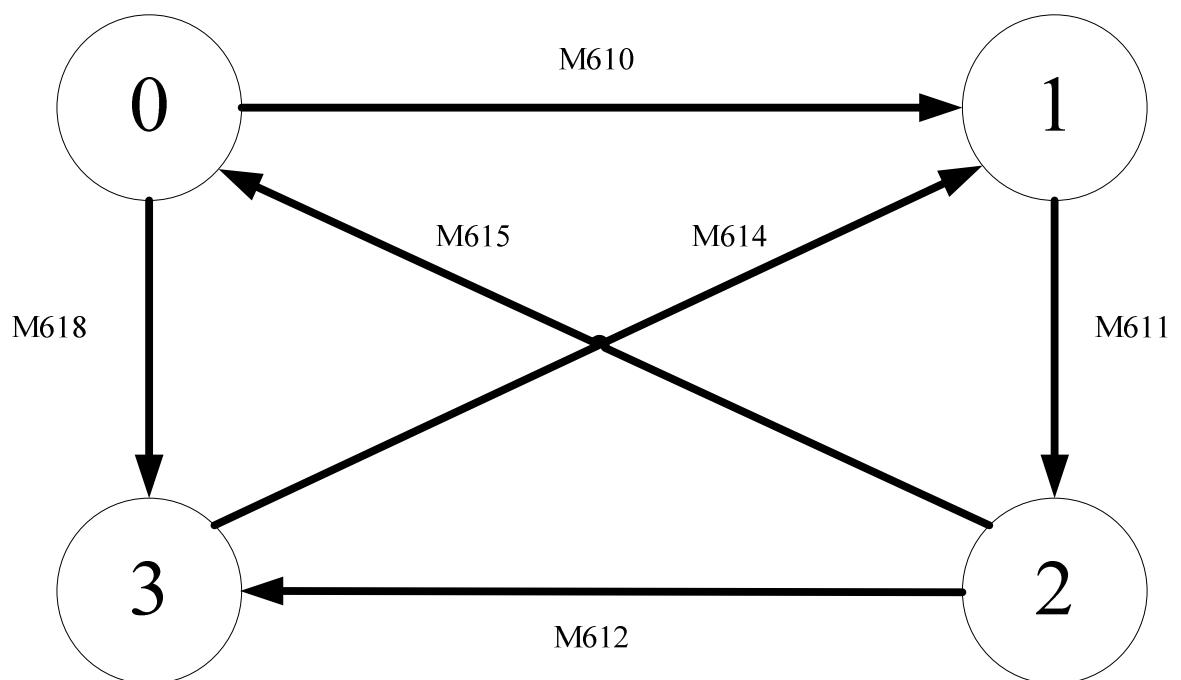
Stan_nastepny:= 0;
RST(TRUE, M615);
RST(TRUE,M650);
END_IF;
MOV(TRUE,1,D610);

3:
IF M614 THEN
    Stan_nastepny:= 1;
    RST(TRUE, M614);
END_IF;
MOV(SM412,2,D610);
MOV(NOT SM412, 0, D610);
OUT(SM412,M652);

END_CASE;
END_IF;

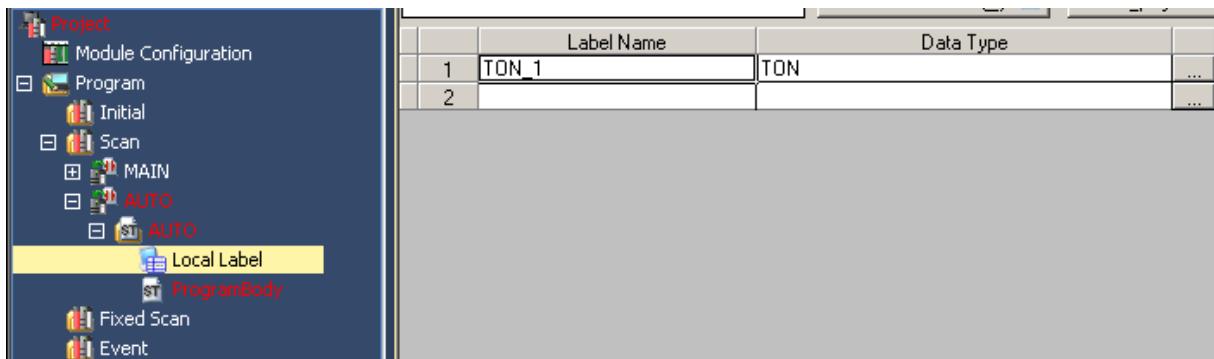
```

Przedstawiony program będzie realizował automat stanów przedstawiony na rysunku poniżej.



### Przykład automatu stanów z wykorzystaniem czasówki:

W pierwszej kolejności należy zdefiniować zmienną lokalną typu TON – opóźnienie załączenia. Nazwa zmiennej to TON\_1. Definicje lokalną wykonujemy w obrębie wybranego programu w zakładce Local Label. Można wpisać dane bezpośrednio do tabelki jak na rysunku XX. Alternatywnie w programie można wpisać nazwę zmiennej TON\_1 (będzie ona podkreślona czerwonym szlaczkiem a jej kolor będzie czarny) a następnie kliknąć prawym przyciskiem myszy na nowej nazwie i wybrać opcję Register Label.



Poniższy listing przedstawia definicję czasówki z zadany czasem odliczania. Zmienna TON\_1 posiada pola IN do aktywacji czasówki, Q do sygnalizacji, że czas został odliczony.

```
//Zdefiniowanie czasówki i czasu liczenia
TON_1(PT:= T#2s500ms);

CASE Stan_aktualny OF
    0:
        IF Start THEN
            Stan_aktualny := 1;
        END_IF;

    1:
        // Aktywacja odliczania czasówki
        TON_1. IN := 1;

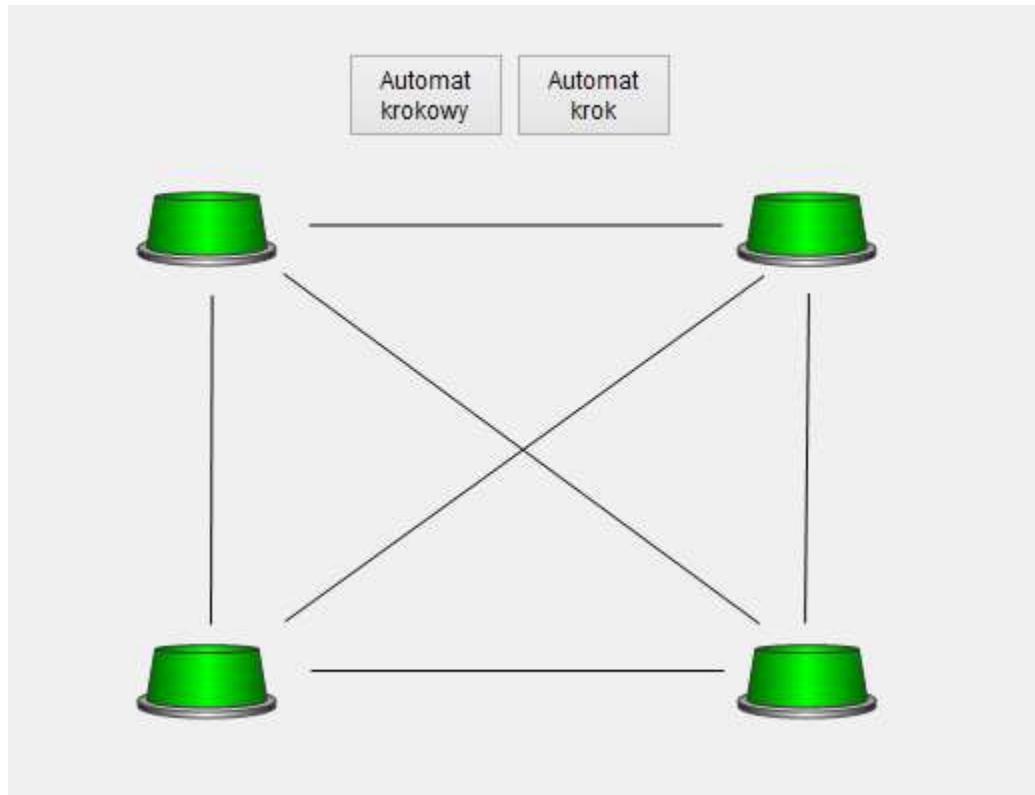
        // Sprawdzenie czy czas odliczony
        IF TON_1.Q THEN
            Stan_aktualny := 2;
            // Deaktywacja odliczania czasówki
            TON_1. IN := 0;
        END_IF;

    2:
        Start := 0;
        Stan_aktualny := 0;

END_CASE;
```

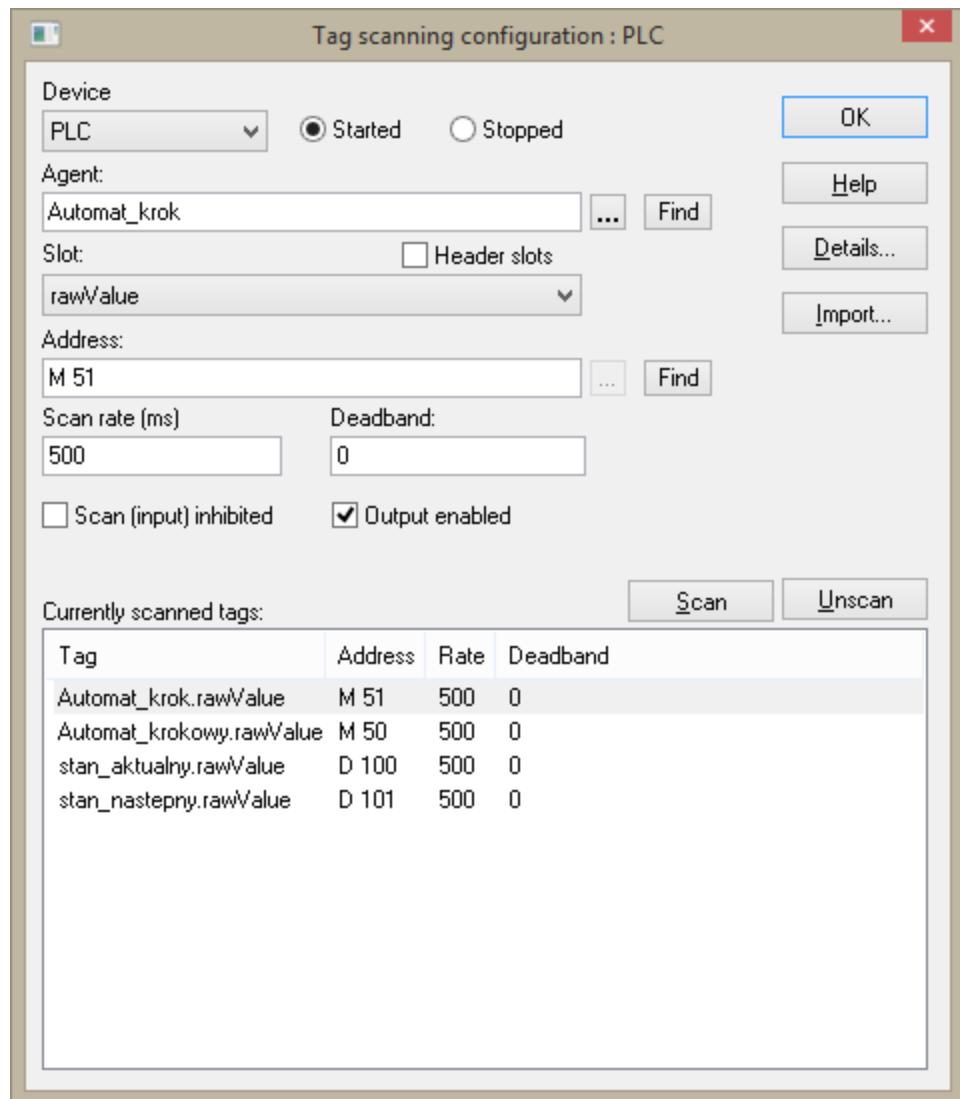
### 5.8.2 Etap projektowania MAPS

Wizualizacja powyższej aplikacji w środowisku MAPS wymaga utworzenia nowych tagów typu Digital oraz podłączenia kontrolek do użytych adresów pamięci. Przykładowa wizualizacja graficzna może wyglądać jak na rysunku poniżej



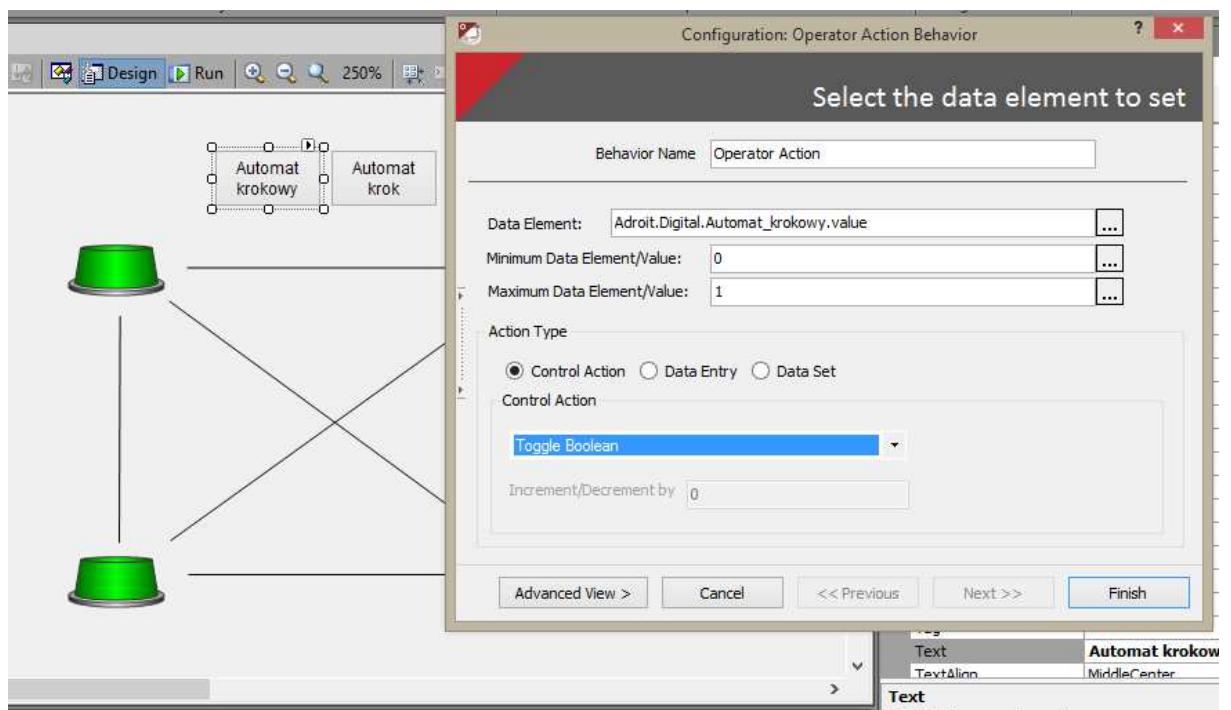
Rys. 5.13: Wizualizacja grafu przejść

Konfiguracja zmiennych skanowanych z PLC:



Rys. 5.14: Ustawienie skanowania zmiennych z PLC

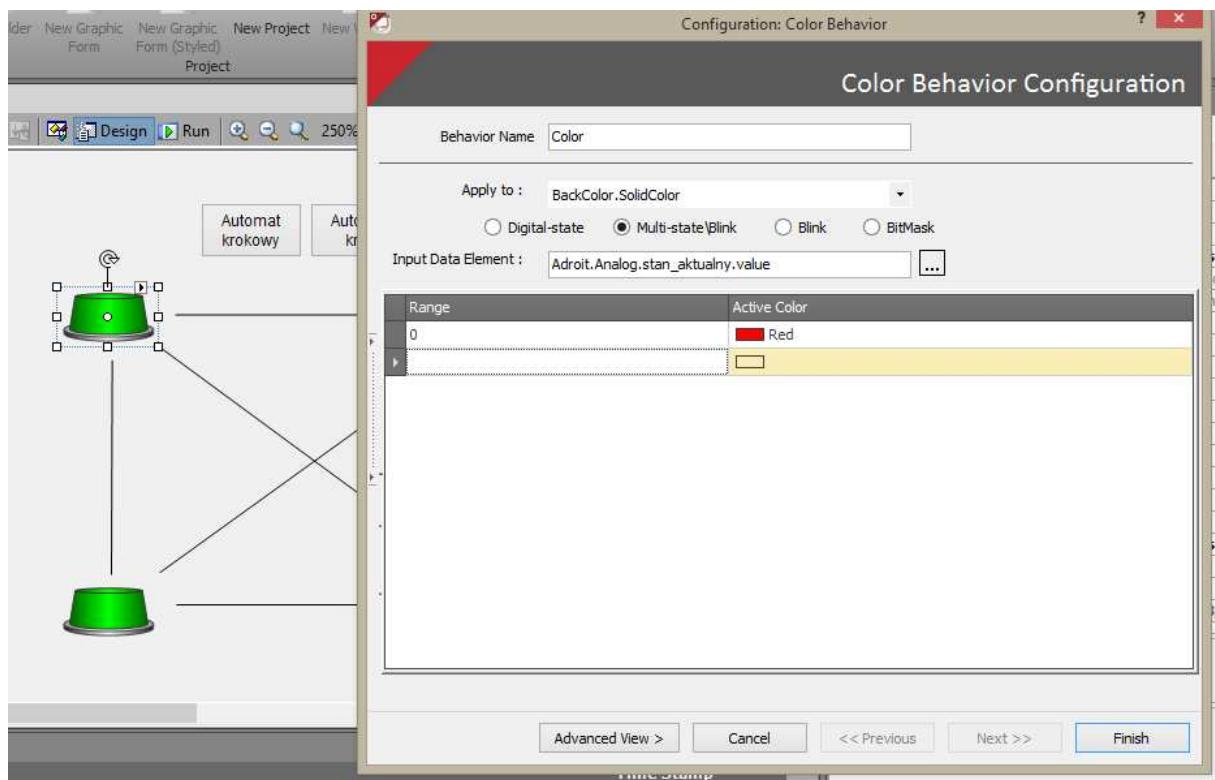
UWAGA: Jeżeli dana zmienna ma być tylko monitorowana z poziomu MAPS to opcja Output enabled może być odznaczona. Jeżeli chcemy oprócz monitorowania również sterować zmienną opcja Output enabled musi być obowiązkowo zaznaczona.



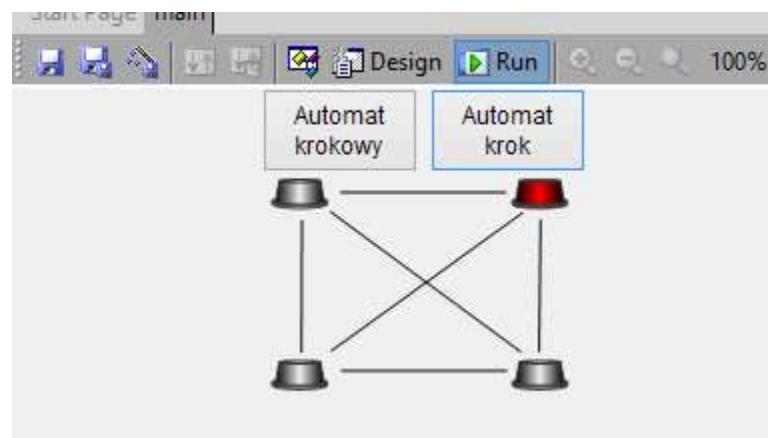
**Rys. 5.15: Zachowanie dla przycisku „Automat krokowy”**

Analogicznie należy połączyć zmienną M51 do przycisku „Automat krok”.

Zmianę stanu można zaprezentować poprzez zmianę koloru kontrolek. Np. dla stanu 0 poniższe ustawienie sprawi, że dla wartości `stan_aktualny = 0` kontrolka przyjmie kolor czerwony. Analogicznie należy ustawić zachowania dla pozostałych stanów.



Rys. 5.16: Zmiana koloru dla stanu 0



Rys. 5.17: Wizualizacja działania wizualizacji. Stan aktualny = 1

## 5.9 Przykład 2 – logika sterowania ciągłego

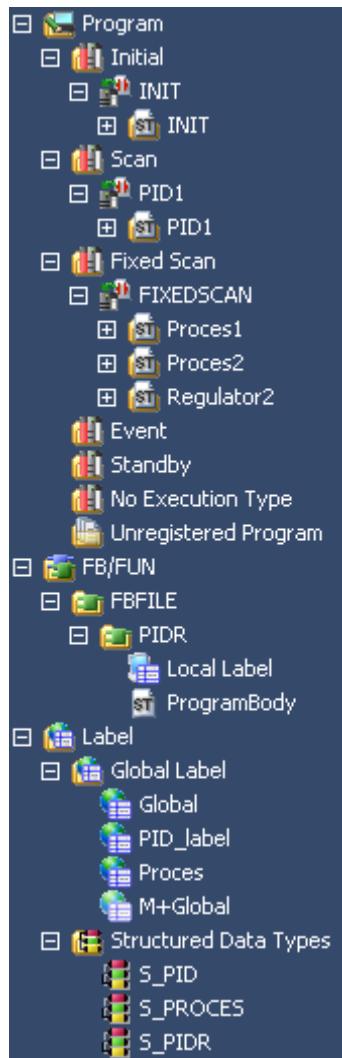
### 5.9.1 Etap projektowania PLC

W tej części zostanie dokładnie omówiona część programowa sterownika PLC, która pozwoli na symulację działania dwóch procesów inercji pierwszego rzędu. Następnie zostaną przygotowane regulatory (PID wbudowany w PLC, PID z równania różnicowego).

Pierwsze kroki pracy z programem GXWorks3 zostały wykonane w powyższych rozdziałach. Należy zweryfikować poprawność parametrów z powyższych rozdziałów, dodać

diagnostyczne mruganie diodą wyjściową oraz uzupełnić do końca programy symulacji procesów, regulatorów i przestrzeni zmiennych globalnych.

Poniżej przedstawiona została struktura programów, grup zmiennych oraz struktur potrzebnych do realizacji przykładu. W sekcji programów INIT znajduje się jeden program, który ustawia parametry procesów i regulatorów. W sekcji programów SCAN znajduje się program realizujący zadanie wbudowanego w PLC regulatora PID. Sekcja programów FIXED SCAN zawiera programy odpowiedzialne za cykliczne wykonywanie symulacji procesu 1 i 2 oraz regulację PID przy użyciu równania różnicowego. W sekcji FB/FUN widnieje biblioteka PIDR, która wykonuje obliczenia w zaimplementowanym regulatorze PID. W sekcji zmiennych znajdziemy trzy grupy potrzebne do realizacji przykładu: Global, PID\_label, Proces. Dla wygody pisania programu zastosowano również struktury danych S\_PID, S\_PROCES, S\_PIDR.



Poniższy rysunek przedstawia elementy i ich typy danych dla struktury S\_PID.

	Label Name	Data Type
1	SV	Word [Signed]
2	PV	Word [Signed]
3	MV	Word [Signed]
4	params	Word [Unsigned]/Bit String [16-bit](0..29)
5	Control_ON	Bit

Poniższy rysunek przedstawia elementy i ich typy danych dla struktury S\_PROCES.

	Label Name	Data Type
1	sampling_time	FLOAT [Single Precision]
2	Time_const	FLOAT [Single Precision]
3	Alfa	FLOAT [Single Precision]
4	Gain	FLOAT [Single Precision]
5	u_k	FLOAT [Single Precision]
6	u_k_1	FLOAT [Single Precision]
7	y_k	FLOAT [Single Precision]
8	y_k_1	FLOAT [Single Precision]
9	A_p	FLOAT [Single Precision]

Poniższy rysunek przedstawia elementy i ich typy danych dla struktury S\_PIDR.

	Label Name	Data Type
1	SV	FLOAT [Single Precision]
2	PV	FLOAT [Single Precision]
3	MV	FLOAT [Single Precision]
4	K_gain	FLOAT [Single Precision]
5	TI	FLOAT [Single Precision]
6	TD	FLOAT [Single Precision]
7	Rp0	FLOAT [Single Precision]
8	Rp1	FLOAT [Single Precision]
9	Rp2	FLOAT [Single Precision]
10	Ep0	FLOAT [Single Precision]
11	Ep1	FLOAT [Single Precision]
12	Ep2	FLOAT [Single Precision]
13	Control_ON	Bit
14	sampling_time	FLOAT [Single Precision]

Poniższy rysunek przedstawia deklaracje zmiennej Proces1 w grupie Proces.

Structure Device Setting Window

	Label Name	Data Type	Class	Assign (Device/Label)
1	Proces1	S_PROCES	...	VAR_GLOBAL Detailed Setting
2	Proces2	S_PROCES	...	VAR_GLOBAL Detailed Setting
3			...	

Proces1 (S\_PROCES)

	Label Name	Data Type	Device
1	sampling_time	FLOAT [Single Precision]	D2000
2	Time_const	FLOAT [Single Precision]	D2002
3	Alfa	FLOAT [Single Precision]	D2004
4	Gain	FLOAT [Single Precision]	D2006
5	u_k	FLOAT [Single Precision]	D2008
6	u_k_1	FLOAT [Single Precision]	D2010
7	y_k	FLOAT [Single Precision]	D2012
8	y_k_1	FLOAT [Single Precision]	D2014
9	A_p	FLOAT [Single Precision]	D2016

Poniższy rysunek przedstawia deklaracje zmiennej Proces2 w grupie Proces.

Structure Device Setting Window

	Label Name	Data Type	Class	Assign (Device/Label)
1	Proces1	S_PROCES	...	VAR_GLOBAL Detailed Setting
2	Proces2	S_PROCES	...	VAR_GLOBAL Detailed Setting
3			...	

Proces2 (S\_PROCES)

	Label Name	Data Type	Device
1	sampling_time	FLOAT [Single Precision]	D2020
2	Time_const	FLOAT [Single Precision]	D2022
3	Alfa	FLOAT [Single Precision]	D2024
4	Gain	FLOAT [Single Precision]	D2026
5	u_k	FLOAT [Single Precision]	D2028
6	u_k_1	FLOAT [Single Precision]	D2030
7	y_k	FLOAT [Single Precision]	D2032
8	y_k_1	FLOAT [Single Precision]	D2034
9	A_p	FLOAT [Single Precision]	D2036

Poniższy rysunek przedstawia deklaracje zmiennej PID1 w grupie PID\_label.

Structure Device Setting Window

PID1 (S_PID)			
	Label Name	Data Type	Device
1	SV	Word [Signed]	D1000
2	PV	Word [Signed]	D1001
3	MV	Word [Signed]	D1002
4	params	Word [Unsigned]/Bit String [16-bit][0..29]	D1003
5	Control_ON	Bit	D1033.0

Poniższy rysunek przedstawia deklaracje zmiennej PID2 w grupie PID\_label.

Structure Device Setting Window

PID2 (S_PIDR)			
	Label Name	Data Type	Device
1	SV	FLOAT [Single Precision]	D1100
2	PV	FLOAT [Single Precision]	D1102
3	MV	FLOAT [Single Precision]	D1104
4	K_gain	FLOAT [Single Precision]	D1106
5	TI	FLOAT [Single Precision]	D1108
6	TD	FLOAT [Single Precision]	D1110
7	Rp0	FLOAT [Single Precision]	D1112
8	Rp1	FLOAT [Single Precision]	D1114
9	Rp2	FLOAT [Single Precision]	D1116
10	Ep0	FLOAT [Single Precision]	D1118
11	Ep1	FLOAT [Single Precision]	D1120
12	Fn2	FLOAT [Single Precision]	D1122

Deklaracje zmiennych w grupie Global pozostają do implementacji w ramach dalszych prac nad przykładowym projektem. W tej chwili grupa może pozostać pusta.

Poniżej przedstawiono konfigurację uruchamiania programów (CPU Parameter -> Program Setting)

Execute Order	Program Name		
		Type	
1	PID1	Scan	
2	FIXEDSCAN	Fixed Scan	Interrupt:I28:1000 ms
3	INIT	Initial	

Przedstawienie przykładowych rozwiązań programów zaczniemy od bloku funkcyjnego PIDR zdefiniowanego w grupie FB/FUN. Poniżej znajduje się deklaracja zmiennej lokalnej wymaganej do pracy ze strukturą danych regulatora.

	Label Name	Data Type	Class
1	PIDR_s	S_PIDR	...
2			VAR_IN_OUT

Kod źródłowy bloku funkcyjnego znajduje się poniżej.

```
//Regulator PID na podstawie równania różnicowego
IF PIDR_s.Control_ON THEN

    //Wyliczenie parametrow
    PIDR_s.Rp0:=PIDR_s.K_gain*(1.0+(PIDR_s.sampling_time/(2.0*PIDR_s.TI)
)+PIDR_s.TD/PIDR_s.sampling_time); //r0 = K*( 1+(Tp/(2*Ti))+Td/Tp );
    PIDR_s.Rp1 := PIDR_s.K_gain*((PIDR_s.sampling_time/(2.0*PIDR_s.TI))-
(2.0*PIDR_s.TD/PIDR_s.sampling_time)-1); //r1 = K*( (Tp/(2*Ti))-(2*Td/Tp)-1 );
    PIDR_s.Rp2:=PIDR_s.K_gain*PIDR_s.TD/PIDR_s.sampling_time; //K*Td/Tp

    //Wyliczenie uchybu regulacji i przesunięcie historii
    PIDR_s.Ep2 := PIDR_s.Ep1;
    PIDR_s.Ep1 := PIDR_s.Ep0;
    PIDR_s.Ep0 := PIDR_s.SV - PIDR_s.PV;

    //Obliczenie sterowania
    PIDR_s.MV := PIDR_s.Rp2*PIDR_s.Ep2 + PIDR_s.Rp1*PIDR_s.Ep1 +
PIDR_s.Rp0*PIDR_s.Ep0 + PIDR_s.MV; //u = R2*E2 + R1*E1 + R0*E0 + u;

    //ANTI WIND UP
    IF (PIDR_s.MV > 100.0) THEN
        PIDR_s.MV := 100.0;
    END_IF;

    IF (PIDR_s.MV < 0.0) THEN
        PIDR_s.MV := 0.0;
    END_IF;
```

END\_IF;

Blok funkcyjny regulatora został użyty w programie Regulator2 w grupie FIXED SCAN. Poniżej przedstawiono deklarację zmiennych lokalnych programu i jego kod źródłowy.

	Label Name	Data Type	
1	PIDR2	PIDR	...

EMOV(PID2.Control\_ON, Proces2.y\_k ,PID2.PV);

PIDR2(PIDR\_s:= PID2);

EMOV(PID2.Control\_ON, PID2.MV, Proces2.u\_k);

Regulator wbudowany w sterowniku PLC został wykorzystany w programie PID1 z grupy SCAN. Poniżej przedstawiono kod źródłowy.

MOV(PID1.Control\_ON, REAL\_TO\_INT( Proces1.y\_k ),PID1.PV);

PID( PID1.Control\_ON, PID1.SV , PID1.PV , PID1.params[0] , PID1.MV);

EMOV(PID1.Control\_ON, INT\_TO\_REAL(PID1.MV), Proces1.u\_k);

Symulacja dwóch procesów została umieszczona w programach Proces1 i Proces2 umieszczonych w grupie FIXED SCAN. Odpowiednie kody źródłowe przedstawiono poniżej.

### Proces 1

```
Proces1.Alfa:=EXP(-Proces1.sampling_time/Proces1.Time_const);
//Wspolczynnik Alfa
```

```
Proces1.A_p := Proces1.Gain * (1 - Proces1.Alfa);
//Wspolczynnik Ap
```

```
Proces1.y_k := (Proces1.A_p * Proces1.u_k_1) + (Proces1.Alfa * Proces1.y_k_1);
//Nowa wartosc wyjscia na podstawie u(k-1), y(k-1)
```

```
Proces1.y_k_1:= Proces1.y_k;
//Zapamietanie wyliczonej wartosci wyjscia
```

```
Proces1.u_k_1:= Proces1.u_k;
//Zapamietanie ostatniego sterowania
```

## Proces 2

```
Proces2.Alfa:=EXP(-Proces2.sampling_time/Proces2.Time_const);
//Wspolczynnik Alfa

Proces2.A_p := Proces2.Gain * (1 - Proces2.Alfa);
//Wspolczynnik Ap

Proces2.y_k := (Proces2.A_p * Proces2.u_k_1) + (Proces2.Alfa * Proces2.y_k_1);
//Nowa wartosc wyjscia na podstawie u(k-1), y(k-1)

Proces2.y_k_1:=Proces2.y_k;
//Zapamietanie wyliczonej wartosci wyjscia

Proces2.u_k_1:=Proces2.u_k;
//Zapamietanie ostatniego sterowania
```

Ostatnim z opisywanych programów będzie program inicjalizujący wszystkie potrzebne zmienne. Program znajduje się w grupie Initial a jego kod źródłowy przedstawiono poniżej.

Warto zwrócić uwagę na ostatnią instrukcję EI(TRUE), która uruchamia przerwania w sterowniku PLC. Dzięki niej uruchamiają się programy z grupy FIXED SCAN z określonym odstępem czasowym.

```
// Ustawienie procesu symulowanego
//Okres probkowania 1s=1000ms - parametr w programie FIXED SCAN
Proces1.sampling_time := 1.0;
// Ustawienie wartosci poczatkowych procesu 1
Proces1.Gain := 10.0;
Proces1.Time_const := 5.0;
//Okres probkowania 1s=1000ms - parametr w programie FIXED SCAN
Proces2.sampling_time := 1.0;
// Ustawienie wartosci poczatkowych procesu 2
Proces2.Gain := 5.0;
Proces2.Time_const := 10.0;

//Parametry regulatora wbudowanego PID1
PID1.params[0] := K1000; //okres regulacji w milisekundach
PID1.params[3] := K1; //wzmocnienie regulatora P
PID1.params[4] := K0; //TI = 0 oznacza nieskonczony czas całkowania - inaczej mowiąc całkowanie wyłączone
PID1.params[5] := K0; //KD = 0 oznacza zerowe wzmocnienie różniczkowania
PID1.params[6] := K0; //TD = 0 oznacza wyłączone różniczkowanie
```

```
PID1.params[22] := 100; //gorny limit wartosci wyjsciowej z regulatora - zapobiega rowniez efektowi wind-up
```

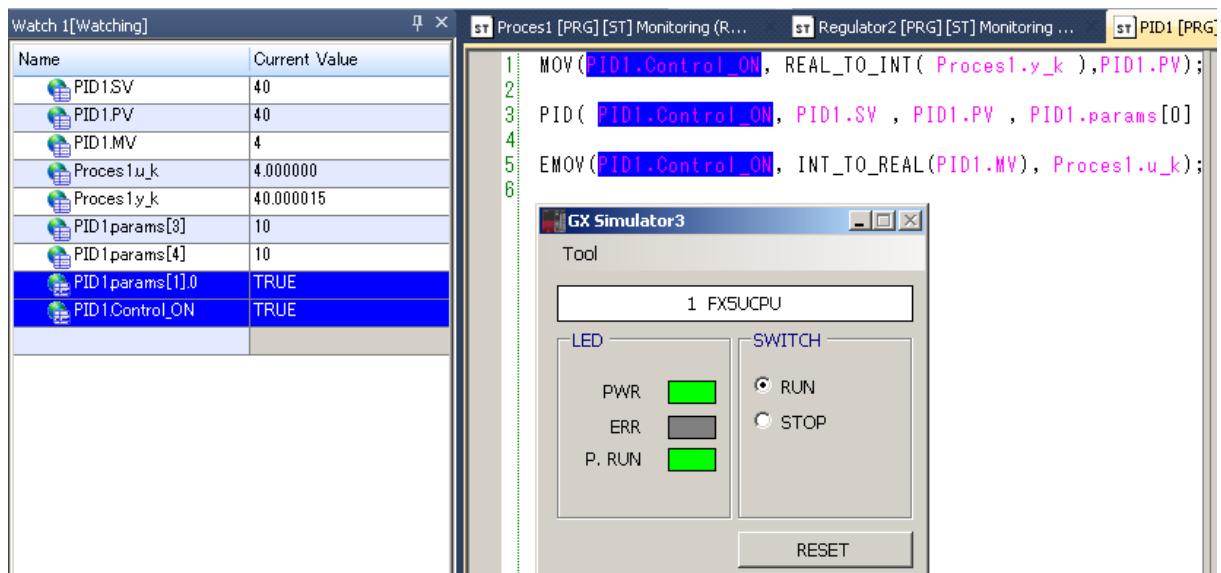
```
PID1.params[23] := 0; //dolny limit wartosci wyjsciowej z regulatora - -||-  
SET(TRUE, PID1.params[1].5); //aktywacja limitow na wyjsciu regulatora  
SET(TRUE, PID1.params[1].0); //trzeba odwrocic kierunek dzialania PID
```

```
//Parametry regulatora z dyskretnego rownania roznicowego PID2
```

```
PID2.K_gain := 1.0;  
PID2.TI := 99999.0;  
PID2.TD := 0.00001;  
PID2.Ep0 := 0.0;  
PID2.Ep1 := 0.0;  
PID2.Ep2 := 0.0;  
PID2.Rp0 := 0.0;  
PID2.Rp1 := 0.0;  
PID2.Rp2 := 0.0;  
PID2.sampling_time := 1.0;
```

```
EI(TRUE);
```

Poniżej przedstawiono możliwości symulacji i monitorowania napisanych programów. Do okna Watch można dodawać pojedyncze zmienne jak również całe struktury.



Screenshot of the Siemens SIMATIC Manager interface showing the Watch [Watching] window and the Source code editor.

**Watch [Watching]**

Name	Current Value
PID1.SV	0
PID1.PV	10
PID1.MV	1
Proces1.u_k	1.000000
Proces1.y_k	10.000004
PID1.params[3]	1
PID1.params[4]	0
PID1.params[10]	TRUE
PID1Control_ON	TRUE
PID2	
SV	30.000000
PV	30.296410
MV	6.058675
K_gain	1.500000
TI	1000.000000
TD	1.000000E-005
Rp0	1.500765
Rp1	-1.499280
Rp2	1.500000E-005
Ep0	-0.296410
Ep1	-0.296671
Ep2	-0.296930
Control_ON	TRUE
sampling_time	1.000000

**Source Code**

```

1 EMOV(PID2.Control_ON, Proces2.y_k ,PID2.PV);
2 |
3 PIDR2(PIDR_s:= PID2);
4
5 EMOV(PID2.Control_ON, PID2.MV, Proces2.u_k);

```

**Output**

Online Program Change: Error:0 Warning:0 Information

## 5.9.2 Etap projektowania MAPS

**Metoda postępowania projektowania w środowisku MAPS będzie analogiczna jak w przypadku zadań z rozdziału 4.6. Dodatkowo trzeba będzie wykonać wizualizację wartości analogowych dla odpowiednich regulatorów.**

## 6 Ćwiczenia w systemie OVATION

### 6.1 Ćwiczenia 1 – programowanie binarne

#### 6.1.1 Cel ćwiczenia

Celem ćwiczenia pierwszego jest poznanie środowiska programowania systemu OVATON oraz zapoznanie się z algorytmami operacji logicznych.

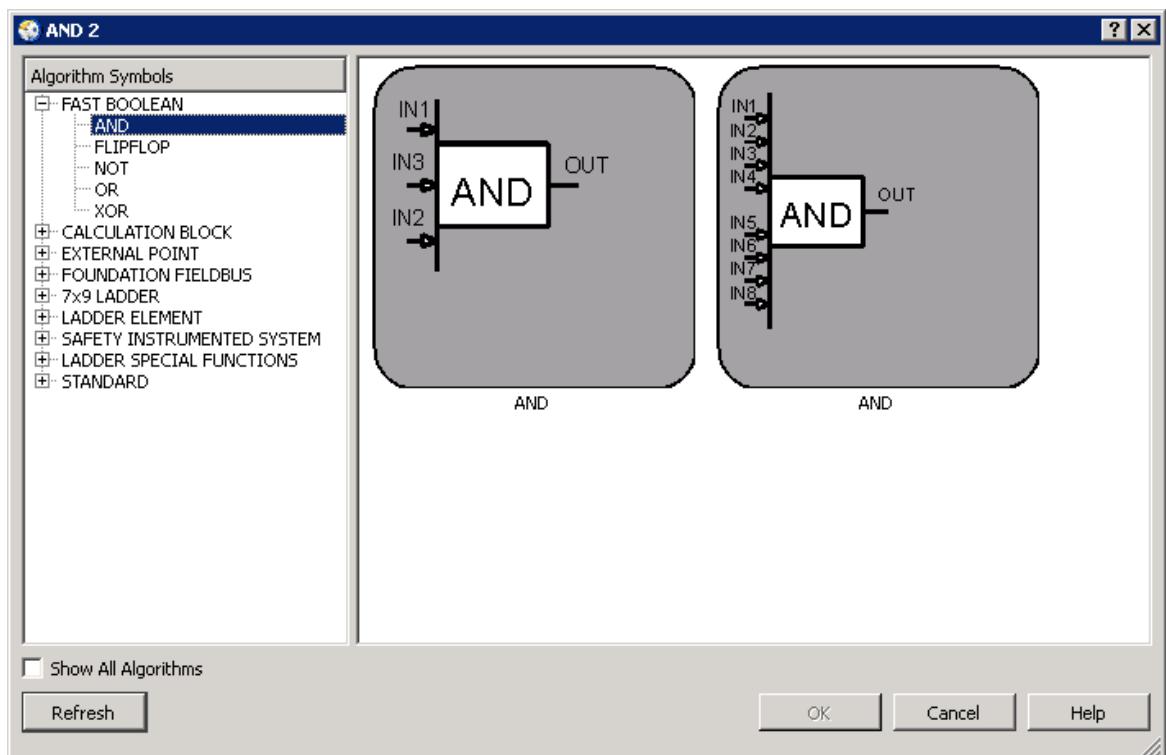
Podczas zajęć studenci będą mieli do wykonania zadanie kombinacyjne polegając na zaprogramowaniu logiki w programie Control Builder a następnie uruchomienie tej logiki w systemie i przedstawienie wyników prowadzącemu.

#### 6.1.2 Pomocne informacje - wskazówki

Przed realizacją zadania zaleca się zapoznanie z przykładem przedstawionym w rozdziale 4.3.

Algorytmy przetwarzające sygnały binarne wykorzystywane w trakcie budowania logiki znajdują się w następujących zbiorach:

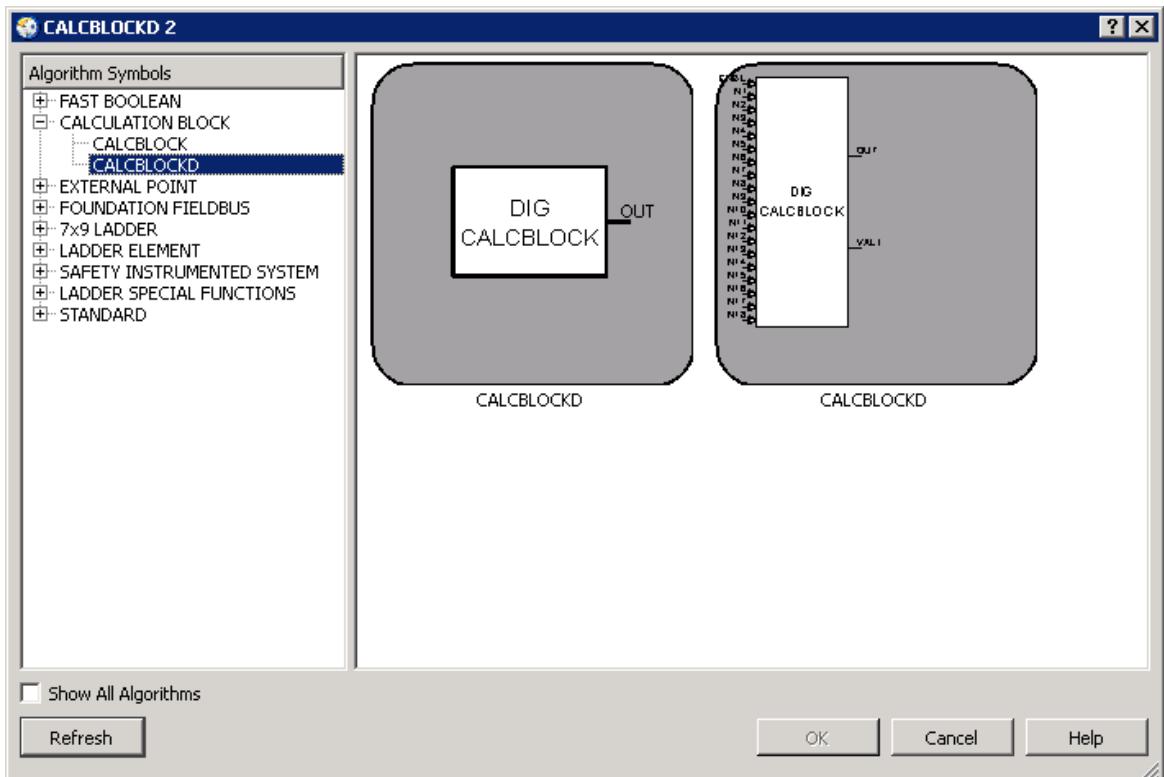
- a) FAST BOOLEAN



Rys. 6.1: Zbiór algorytmów FAST BOOLEAN.

Katalog ten zawiera algorytmy bramek: AND, OR, XOR oraz algorytm przerzutnika FLIPFLOP i negacji NOT. Szczegółowy opis dostepny jest w pomocy (przycisk Help).

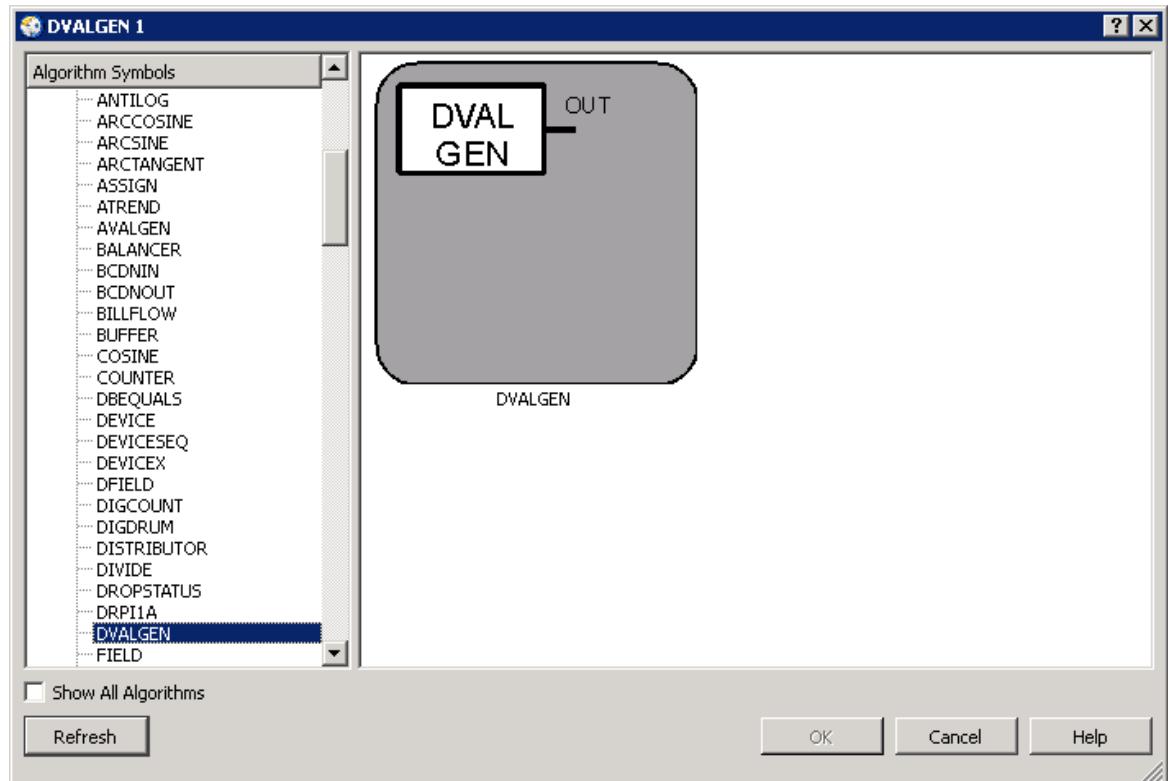
b) CALCULATION BLOCK



Rys. 6.2: Zbiór algorytmów CALCULATON BLOCK.

Katalog ten zawiera algorytm CALCBLOCKD pozwalający na zapis skryptowy logiki. Istotne są ograniczenia na liczbę zmiennych i operacji które można użyć w jednym bloku.

c) STANDARD



Rys. 6.3: Zbiór algorytmów STANDARD.

Katalog ten zawiera grupę standardowych algorytmów obliczeniowych systemu OVATION. Większość z tych algorytmów operuje na sygnałach ciągłych, ale też są pośród nich te, które przetwarzają sygnały dyskretne. Szczególne użyteczne z punktu widzenia wykonyanego ćwiczenia pierwszego są: AAFLIPFLOP, ONESHOT, ONDELAY, OFFDELAY, FIRSTOUT, LOWMON, HIGHMON. Szczegółowe informacje dostępne są w plikach pomocy każdego z algorytmów.

## 6.2 Ćwiczenia 2 – programowanie ciągłe

### 6.2.1 Cel ćwiczenia

Celem ćwiczenia drugiego jest zapoznanie się z algorytmami wykorzystywanymi w sterowaniu ciągłym.

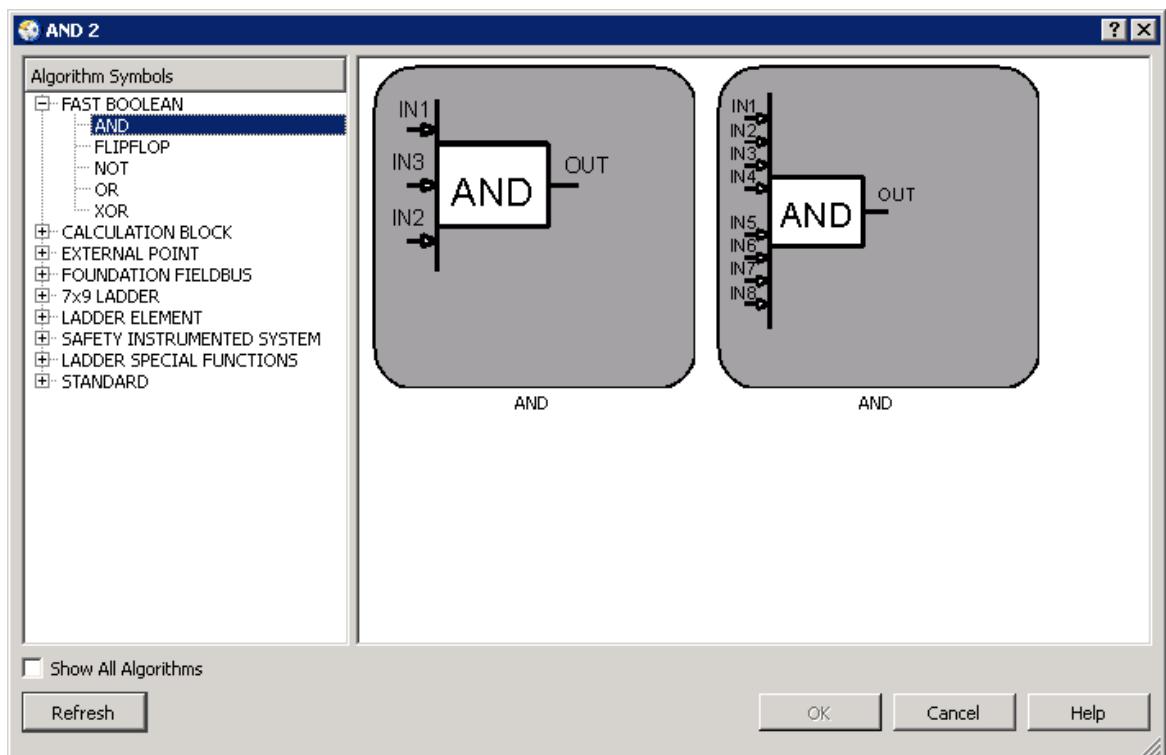
Podczas zajęć studenci będą mieli do wykonania zadanie polegając na zaprogramowaniu logiki w programie Control Builder a następnie uruchomienie tej logiki w systemie i przedstawienie wyników prowadzącemu.

### 6.2.2 Pomocne informacje - wskazówki

Przed realizacją zadania zaleca się zapoznanie z przykładem przedstawionym w rozdziale 4.3.

Algorytmy przetwarzające sygnały binarne wykorzystywane w trakcie budowania logiki znajdują się w następujących zbiorach:

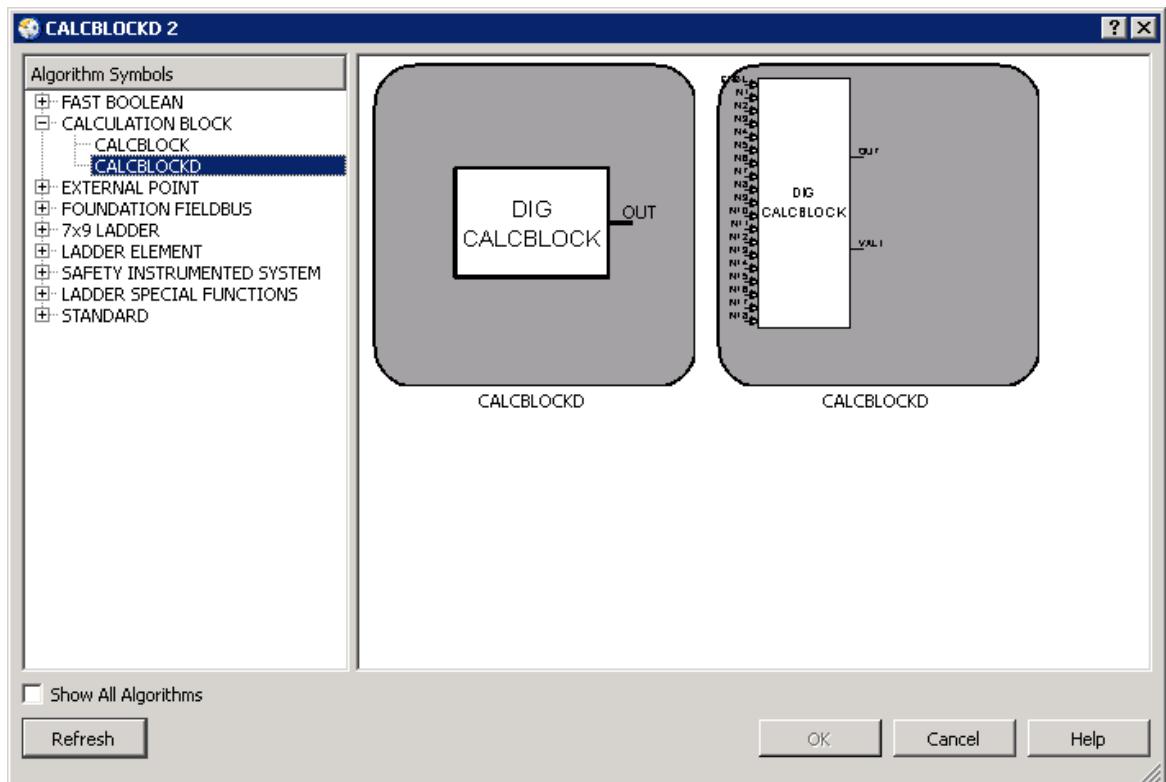
- d) FAST BOOLEAN



Rys. 6.4: Zbiór algorytmów FAST BOOLEAN.

Katalog ten zawiera algorytmy bramek: AND, OR, XOR oraz algorytm przerzutnika FLIPFLOP i negacji NOT. Szczegółowy opis dostępny jest w pomocy (przycisk Help).

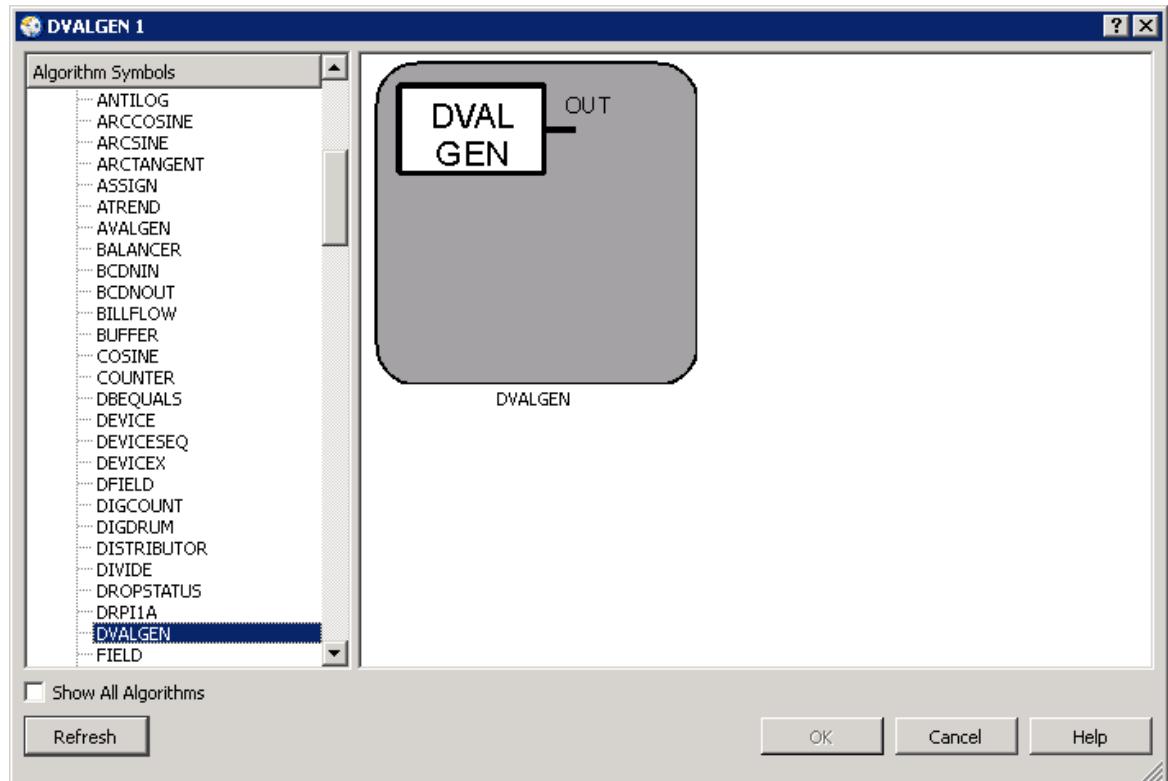
### e) CALCULATION BLOCK



Rys. 6.5: Zbiór algorytmów CALCULATON BLOCK.

Katalog ten zawiera algorytm CALCBLOCKD pozwalający na zapis skryptowy logiki. Istotne są ograniczenia na liczbę zmiennych i operacji które można użyć w jednym bloku.

### f) STANDARD



Rys. 6.6: Zbiór algorytmów STANDARD.

Katalog ten zawiera grupę standardowych algorytmów obliczeniowych systemu OVATION. Większość z tych algorytmów operuje na sygnałach ciągłych, ale też są pośród nich te, które przetwarzają sygnały dyskretne. Szczególne użyteczne z punktu widzenia wykonyanego ćwiczenia pierwszego są: AAFLIPFLOP, ONESHOT, ONDELAY, OFFDELAY, FIRSTOUT, LOWMON, HIGHMON. Szczegółowe informacje dostępne są w plikach pomocy każdego z algorytmów.

### **6.3 Ćwiczenia 3 4 5 – projekt struktury regulacji**

## 7 Ćwiczenia w systemie SCADA

### 7.1 Ćwiczenia 1 – programowanie binarne

#### Skrzyżowanie

##### Opis zadania:

Przedmiotem projektu jest sterowanie sygnalizacją świetlną na krzyżowaniu zawierającym pasy dedykowane do skrętu w lewo, do jazdy na wprost oraz warunkowe do skrętu w prawo, a także przejście dla pieszych na żądanie.

##### W projekcie należy:

1. Przygotować odpowiedni algorytm do sterowania sygnalizatorami wg poniższych wytycznych – sugerowane jest zastosowanie automatu stanów przedstawionego w poprzednich rozdziałach
2. Napisać program na sterownik PLC (GX Works)
3. Przygotować wizualizację projektu w środowisku SCADA MAPS

##### Wytyczne do sterowania sygnalizatorami:

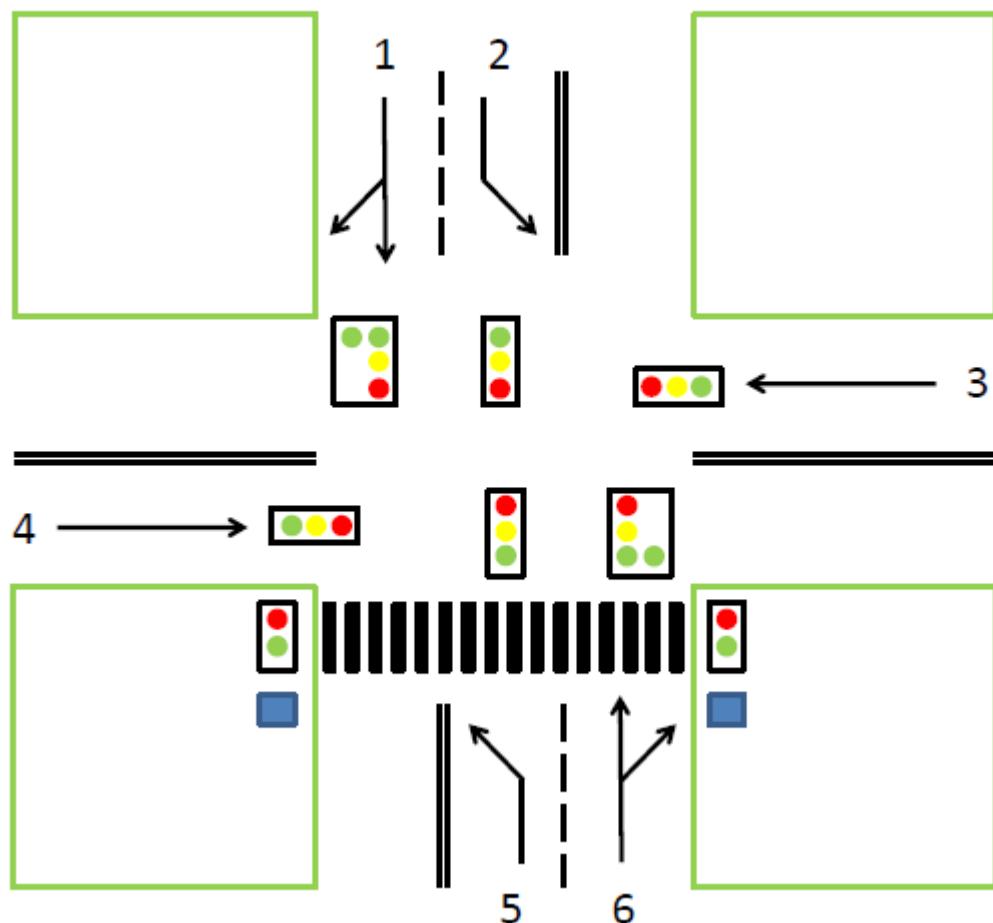
1. Kombinacja świateł (tory drogowe) zmieniają się, co 20 sekund.
2. Przez te 20 sekund: 2 sekundy świeci się żółte z czerwonym, następnie 16 sekund zielone, następnie 2 sekundy żółte i koniec sekwencji, poza sekwencją zmiany świateł świeci się światło czerwone.
3. Kolejność torów drogowych:
  - a) 2,5 (lewo skręty), reszta czerwone
  - b) 3,4 (jazda na wprost)
  - c) 1,6 (jazda na wprost i w prawo oczywiście, ale sygnalizator strzałka w prawo zgaszony)
  - d) strzałka w prawo dla toru 1 (tylko w czasie, gdy świeci się zielone na torze 3, 4),
  - e) warunkowo strzałka w prawo dla toru 6 (tylko w czasie, gdy świeci się zielone na torze 3, 4; warunek: przejście dla pieszych sygnał czerwony)
  - f) przejście dla pieszych na żądanie: zielone zapalone tylko, gdy zapalone zielone na torze 3,4 (czyli po zwłoce 2 sekund na czerwone + żółte). Zielone dla pieszych zapalone na 13 sekund, później miganie przez 5 sekund (miganie SM412, 0.5sek ON, 0.5sek OFF).

Czyli w cyklu:

- 2 sekundy czerwone
- 13 sekund zielone ciągłe
- 5 sekund zielone miganie

Pieszy poprzez naciśnięcie przycisku żąda przejścia, PLC powinno zapamiętać żądanie a następnie wykryć, w którym cyklu załączyć zielone, i jeżeli żądanie następuje, gdy aktywny jest tor 1,2 lub 5,6 sygnał zielony zapalany jest w następnej najbliższej sekwencji 3,4. Jeżeli natomiast żądanie nastąpiło, gdy aktywny jest tor 3,4 to sygnał zielony jest zapalany dopiero w kolejnej sekwencji 3,4.

Poniżej przedstawiony jest widok skrzyżowania do projektu.



Uwaga: Zadanie należy zrealizować w języku ST (Structured Text).

## 7.2 Ćwiczenia 2 – programowanie ciągłe

### Symulacja procesów ciągłych

#### Opis zadania:

Przedmiotem projektu jest sterowanie ciągłe procesów inercyjnych pierwszego rzędu. Sterowanie powinno odbywać się z poziomu systemu SCADA.

#### W projekcie należy:

1. Utworzyć 4 procesy inercyjne na bazie przykładów

2. Utworzyć 2 regulatory PID z użyciem wbudowanej funkcji PID
3. Utworzyć 2 regulatory PID z użyciem równania różnicowego
4. Przygotować wizualizację projektu w środowisku SCADA MAPS
5. Wizualizacja powinna obejmować pracę w trybie ręcznym i automatycznym dla każdego regulatora
6. Praca w trybie ręcznym powinna pozwalać na zmianę wartości sterowania przy pomocy dowolnego interfejsu użytkownika (pole do wpisu wartości, suwak, potencjometr...)
7. Praca w trybie automatycznym powinna pozwalać na zmianę wartości zadanej procesu, zmianę wartości parametrów regulatora
8. Należy użyć dostępnych form do tworzenia grafiki (oceniana będzie oryginalność i pomysłowość realizacji zadania wizualizacji)
9. Parametry symulowanych inercji można przyjąć wedle uznania (nie są sugerowane duże stałe czasowe)
10. Parametry regulatorów dobrać bardzo zgrubnie, aby uzyskać zadowalające wyniki regulacji (nie jest to głównym przedmiotem pracy)
11. Wysłanie przykładowych wyników regulacji do środowiska MATLAB przy pomocy Socket Communication w celu wykonania wykresów.

**Uwaga:** Zadanie należy zrealizować w języku ST (Structured Text).

**Sugerowane jest rozpoczęcie pracy od zapoznania się z przykładowym programem dla sterownika PLC, następnie jego uruchomienie i krótkie przetestowanie (zadawanie wartości wejściowych procesu, odczyt odpowiedzi procesu, uruchomienie regulatora PID wbudowanego, zmiana wartości zadanej). Następnie należy przejść do realizacji wizualizacji dla pierwszego procesu (grafiki dynamiczne od wartości zadanych, sterowań i wartości mierzonych, przełącznik AUTO/RĘKA....). Jeżeli ten etap będzie zrealizowany można przejść do uruchamiania kolejnych procesów i realizacji kolejnych grafik.**

### 7.3 Ćwiczenia 3 4 5 – projekt struktury regulacji

**UWAGA: URUCHOMIENIE STANOWISK ROZPOCZYNAMY OD SPRAWDZENIA POŁĄCZENIA MIĘDZY STANOWISKIEM (PUDEŁKO Z ELEKTRONIKĄ FIRMY INTECO) A STEROWNIKIEM PLC (ZESTAW PLC Z PŁYTĄ KONVERTUJĄCĄ SYGNAŁY) – SZARE TAŚMY KOMPUTEROWE. NASTĘPNIE WCISKAMY Czerwony PRZYCISK „GRZYB” AWARYJNY. WŁĄCZAMY ZASILANIE ZESTAWU PLC, NASTĘPNIE WŁĄCZAMY ZASILANIE STANOWISKA (PRZEŁĄCZNIK Z TYŁU OBUDOWY PUDEŁKA Z ELEKTRONIKĄ). JEŻELI PLC JEST WŁĄCZONE MOŻNA WYCIAĞNĄĆ PRZYCISK „GRZYB” AWARYJNY I WCISNĄĆ NA STANOWISKU Czerwony PRZYCISK WŁĄCZAJĄCY ZASILANIE W STANOWISKU. OD TEGO MOMENTU NALEŻY ZWRÓCIĆ SZCZEGÓLNAJ UWAGĘ NA PRACĘ W LABORATORIUM. NALEŻY ZACHOWAĆ OSTROŻNOŚĆ, ABY NIE USZKODZIĆ ZESTAWÓW A NAJWAŻNIEJSZE NIE ZROBIĆ KOMUŚ LUB SOBIE KRZYWDY.**

### **Cel projektu:**

Celem projektu jest zaprojektowanie oraz implementacja układu regulacji automatycznej dla wybranego stanowiska laboratoryjnego z wykorzystaniem systemu SCADA. Realizacja zadania odbywa się w formie pracy zespołowej (po trzy osoby na stanowisko).

### **Etap 1:**

#### **Wymagania ogólne:**

1. Kompletny projekt dla sterownika przemysłowego PLC:
  - a. Konfiguracja modułów peryferyjnych t.j:
    - i. Moduł komunikacyjny Ethernet (internal + external),
    - ii. Moduł analogowy,
    - iii. Moduł szybkich liczników HIOEN.
  - b. Właściwy podział na zmienne lokalne i globalne wraz z określeniem adresacji pamięci sterownika
  - c. Obsługa pomiarów z obiektu – konwersja odczytów do wielkości fizycznych (skalowanie – instrukcja SCL)
  - d. Obsługa sterowań (np. PWM)
  - e. Obsługa dodatkowych sygnałów (ciągłych, binarnych), umożliwiających sterowanie obiektem.
  - f. Realizacja sterowania w otwartej/zamkniętej pętli regulacji
    - i. Określenie prawidłowego zakresu wartości zadanej
    - ii. Wstępne dobranie nastaw regulatorów, metodą inżynierską
  - g. Implementacja zabezpieczeń zapewniających bezpieczną pracę obiektu:
    - i. Obsługa krańcówek (jeśli występują – szczególnie dźwig)
    - ii. Bezpieczna szybkość poruszania się elementów ruchomych
  - h. Implementacja możliwości pracy w trybie ręcznym i automatycznym
  - i. Zaproponowanie trajektorii testowej zawierającej kilka zmian wartości zadanych

**Uwaga:** Zadanie należy zrealizować w języku ST (Structured Text).

Każde stanowisko wyposażone jest w wyłącznik bezpieczeństwa, z którego należy korzystać zawsze, gdy istnieje ryzyko uszkodzenia sprzętu lub istnieje zagrożenie zdrowia osoby znajdującej się w przestrzeni roboczej obiektu.

#### **Wymagania dla poszczególnych stanowisk:**

## 1. TRAS



- Realizacja sterowania w zamkniętej pętli umożliwiająca utrzymywanie zadanego kąta obrotu i poziomu helikoptera

(Zadajnik wartości zadanej – programowy (PLC, SCADA))

Możliwość ręcznego sterowania poszczególnych silników, wizualizacja prędkości z pomiaru analogowego.

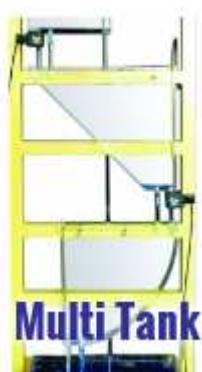
## 2. SERVO



Realizacja sterowania w zamkniętej pętli

(Zadajnik wartości zadanej – pokrętło ręczne); Dwa przełączane tryby pracy: tryb pozycjonowania – zadawanie pozycji pokrętłem ręcznym, pomiar pozycji enkoderem; tryb prędkościowy -> zadawanie prędkości pokrętłem ręcznym; użycie pomiaru prędkości z wejścia analogowego

## 3. MULTI TANK



Sterowanie poziomu cieczy w trzech zbiornikach (Zadajnik wartości zadanej – programowy (PLC, SCADA); Regulacja poziomu w trzech zbiornikach jednocześnie. Sterowanie wydajnością pompy i otwarciem zaworów.

#### 4. TOWER CRANE



Sterowanie położenia obciążenia w trzech osiach X, Z, Th. Należy przewidzieć obsługę krańcówek sprzętowych A także programowych!

Przygotowanie procedury bazowania – dopiero po jej wykonaniu powinno być możliwe sterowanie na zadane pozycje.

Procedura centrowania – po bazowaniu automatyczny przejazd na pozycje środkowe danych osi.

(Zadajniki wartości zadanej – programowe (PLC, SCADA))

##### *Etap 2:*

1. Kompletną wizualizację procesu w systemie MAPS SCADA:
  - a. Wizualizacja obiektu
  - b. Panel operatorski do sterowania ręcznego/automatycznego
  - c. Panel z nastawami regulatorów
  - d. Panel z wykreślaniem wartości zmiennych określających jakość regulacji: SP, PV, MV.

##### *Etap 3:*

Sprawozdanie dokumentujące stan aplikacji po ukończeniu ostatnich zajęć projektowych.

**UWAGA: Z uwagi na możliwość utraty projektów z ostatnich zajęć zaleca się wykonanie kopii projektu z GXWorks 3 oraz eksport projektu z MAPS i archiwizację.**