Ćwiczenie nr 1: Sterowniki GE-FANUC
Ćwiczenie nr 2: Programowanie linii

Martyna Żukowska 252877
Maciej Barna 252931
Szymon Kordek 252935

grupa E13-01a, Pn 8:00 TN

1 Cel pracy

Nabycie podstawowych umiejętności w programowaniu Sterowników GE Fanuc oraz zapoznanie się z podstawowymi funkcjami i możliwościami programu CODESYS.

2 Opis wykonania ćwiczeń

Ćwiczenie było wykonywane w programie CODESYS, w którym początkowo należało stworzyć nowy plik, a następnie dodatkowo utworzyć i podpiąć do niego Vizualizacje, aby w późniejszych etapach można było sprawdzić poprawność napisanego kodu na fikcyjnych obiektach.

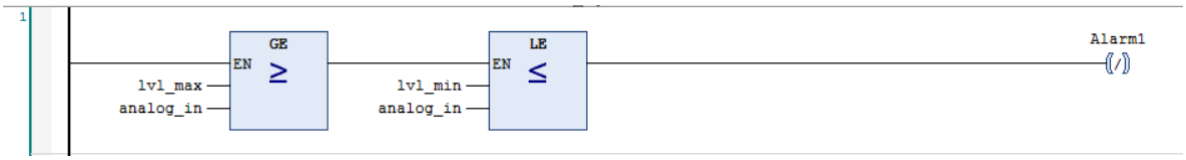
3 Laboratorium 1

3.1 Ćwiczenie 4.1

Pierwszym zadaniem było napisanie programu sygnalizującego przekroczenie ustalonej wartości minimalnej i maksymalnej analogowego sygnału wejściowego.

Do wykonania zadania wykorzystaliśmy bloki GE (greater than or equal - większy lub równy) i LE (less than or equal - mniejszy lub równy). Posłużyły one do stworzenia

swego rodzaju przedziałów i sprawdzenia czy sygnał "analog_in" do nich nie należy. Zostało to skonstruowane w taki sposób aby zapalić diode Alarmu 1 gdyby tak właśnie było, dlatego też na końcu tej linii jest cewka negująca (która zapisuje stan wartości 0 po dojściu sygnału).



Rys. 3.1. Schemat programu

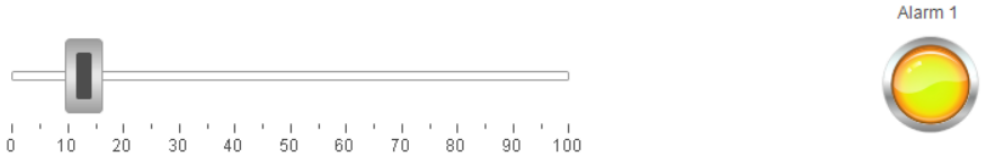
Na potrzebę zadania ustaliliśmy przykladowe wartości naszego przedzialu (20,80), a następnie sprawdziliśmy czy odpowiednio nasza cewka negująca jest typem BOOL, a wartości przy blokach GE i LE to INT'y.

	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**lvl_max**		INT	80		
2	VAR	**lvl_min**		INT	20		
3	VAR	**Alarm1**		BOOL			
4	VAR	**analog_in**		INT			

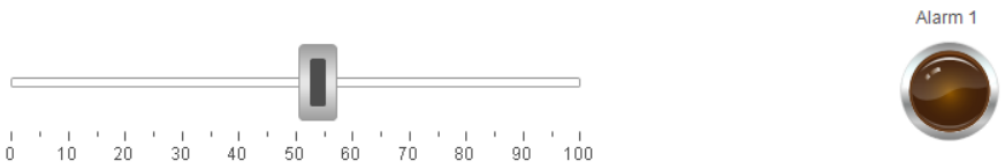
Rys. 3.2. Odpowiednie wartości schematu

Dodatkowo sygnał "analog_in" zasymulowaliśmy w wizualizacji, tak aby zmieniał odpowiednio wartość w zależności od przesuwania umieszczonego w niej suwaka.

W dalszej części sprawdziliśmy poprawność naszego kodu, odpowiednio ustawiając w programie status Symulacji. Następnie logowaliśmy sie i uruchamialiśmy program.



Rys. 3.3. Wizualizacja kiedy sygnał wychodzi poza zakres



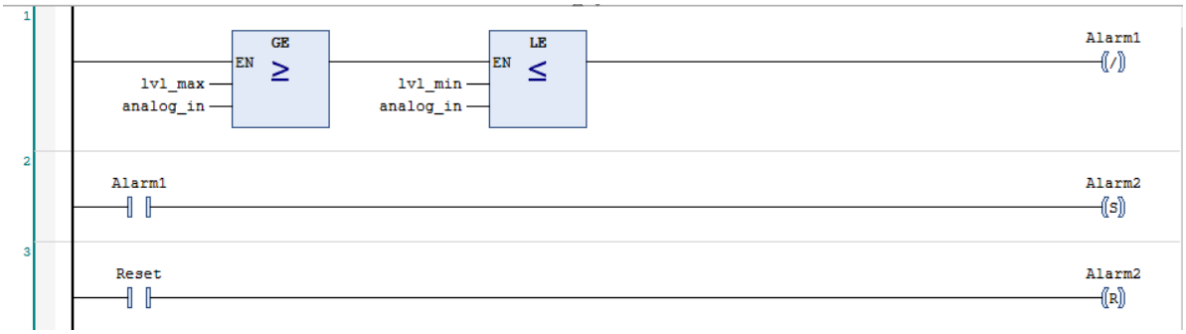
Rys. 3.4. Wizualizacja kiedy sygnał jest z zakresu

Jak można zauważyć zadanie zostało wykonane poprawnie. Dioda alarmu zaczyna świecić po przekroczeniu wyznaczonego zakresu i gaśnie, gdy do niego wracamy.

3.2 Ćwiczenie 4.2

W drugim zadaniu chodziło o zmianę poprzedniego programu w sposób, aby po załączeniu wskaźnik przekroczenia wartości alarmowej świecił aż nie wykasuje go operator.

Aby to zrobić dodaliśmy dodatkowe dwie linie kodu. Pierwszą zależną od wartości Alarmu 1 z pierwszego zadania, kończącą się cewką typu SET, która zapamiętywała stan 1 do momentu zresetowania go cewką resetującą, którą dodaliśmy w kolejnej linii kodu. Cewka resetująca zaczyna działać w zależności od stanu styku Reset.



Rys. 3.5. Schemat programu

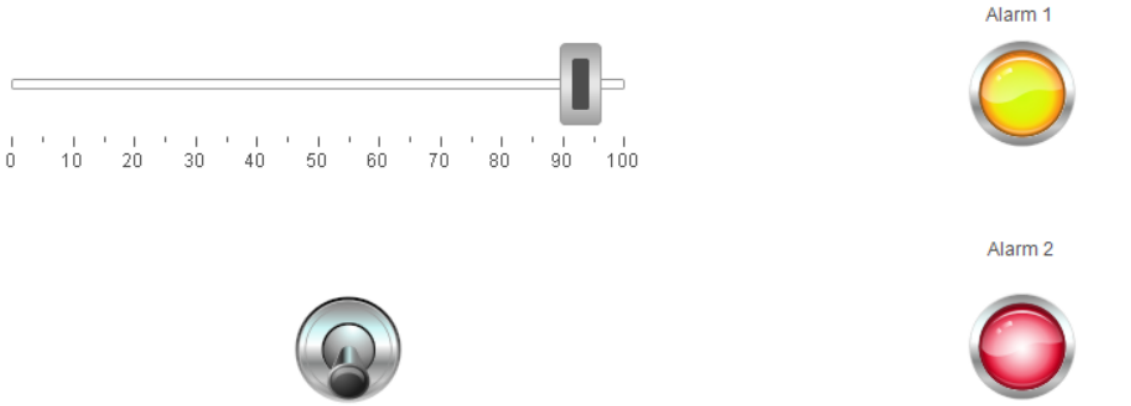
Następnie sprawdziliśmy czy zmienne Alarm2 i Reset przypisane do cewki oraz styku są typu BOOL.

	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**lvl_max**		INT	80		
2	VAR	**lvl_min**		INT	20		
3	VAR	**Alarm1**		BOOL			
4	VAR	**analog_in**		INT			
5	VAR	**Alarm2**		BOOL			
6	VAR	**Reset**		BOOL			

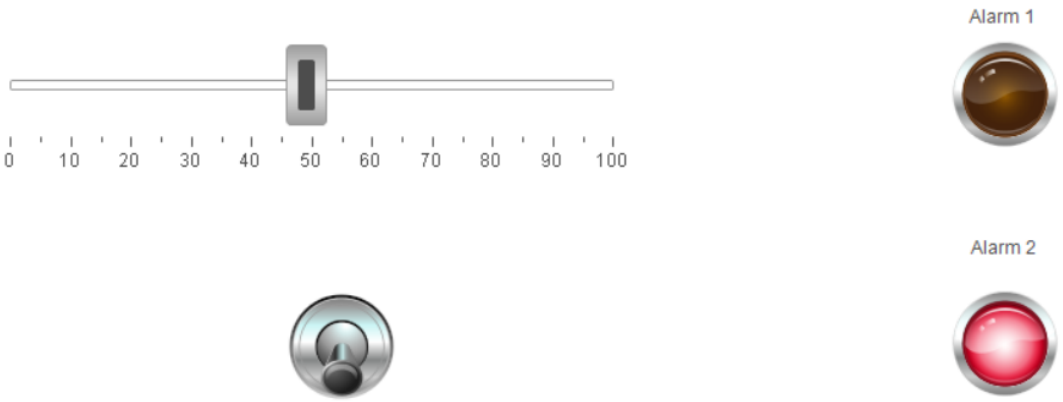
Rys. 3.6. Odpowiednie wartości schematu

W wizualizacji dodalismy dodatkową diodę zależną od wyjścia Alarmu 2 i przełącznik który będzie uaktywniał styk Reset.

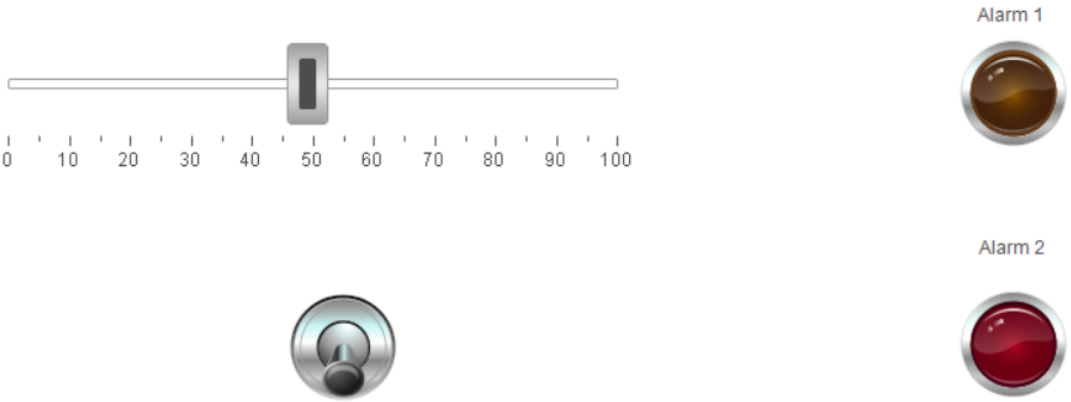
Następnie zasymulowaliśmy program, aby zweryfikować czy nasz kod jest poprawny. Po zadaniu błędnego zakresu sprawdzamy czy zapalą się oba alarmy. Po czym patrzymy jak zachowa się Alarm2, gdy zmienimy sygnał na mieszczący się w zakresie, a następnie zresetujemy go przyciskiem Reset.



Rys. 3.7. Wizualizacja gdy sygnał nie jest z zakresu, co powoduje zapalenie obu Alarmów



Rys. 3.8. Wizualizacja gdy sygnał z powrotem jest z zakresu



Rys. 3.9. Wizualizacja po wciśnieciu Reset gdy sygnał z powrotem jest z zakresu

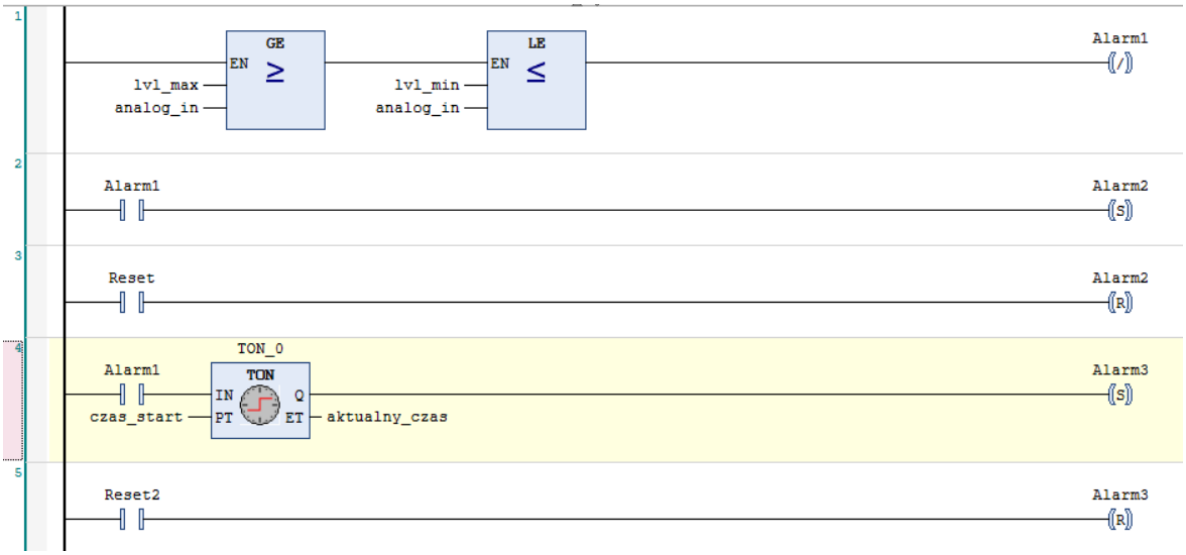
Łatwo zauważyć, że program działa poprawnie, obie diody zapalają się w opowiednim momencie, a kiedy Alarm 1 gaśnie Alarm 2 pozostaje zapalony do momentu, aż zresetujemy go ręcznie za pomocą przycisku.

3.3 Ćwiczenie 4.3

Trzecie zadanie polegało na rozszerzeniu programu tak, aby ustawiany był kolejny znacznik alarmu w momencie gdy przekroczenie poziomu alarmowego trwa dłużej niż 10 sekund, zresetowanie stanu alarmu miało być realizowane przez operatora.

Do realizacji tego podpunktu dodaliśmy kolejne 2 linie. Na poczatku linii 4 dodaliśmy wejście zależne po raz kolejny od Alarmu 1. Natępnie podłaczylismy timer TON 0,

który za zadanie mial przepuszczać sygnał dalej po upływie "czas start" równy 10 sekund od momentu otrzymania sygnału na wejście (włączenie alarmu). Na koniec została dołączona cewka typu "SET" Alarm 3, która następnie będzie resetowana w linijce 5 analogicznie jak w Ćwwiczeniu 4.2.



Rys. 3.10. Schemat programu

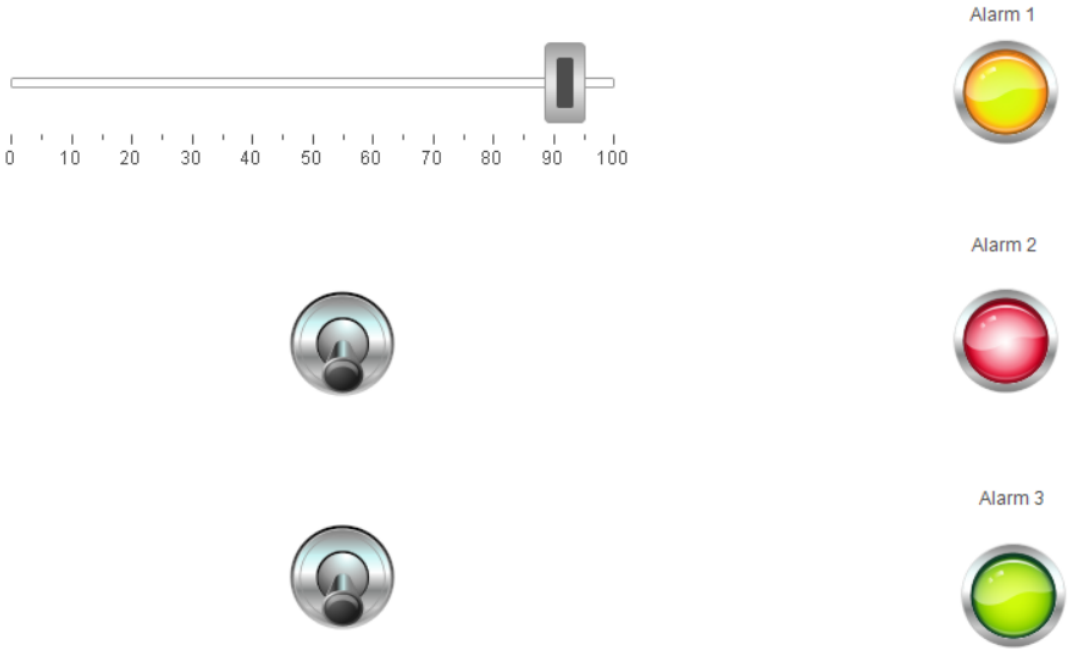
Po stworzeniu schematu inicjujemy wartość zmiennej "czas start" (T10s) i sprawdzamy czy zmienne Alarm 3 oraz Reset 2 są typu BOOL, a także czy "aktulany czas" i "czas start" są typu TIME.

	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**lvl_max**		INT	80		
2	VAR	**lvl_min**		INT	20		
3	VAR	**Alarm1**		BOOL			
4	VAR	**analog_in**		INT			
5	VAR	**Alarm2**		BOOL			
6	VAR	**Reset**		BOOL			
7	VAR	**TON_0**		TON			
8	VAR	**czas_start**		TIME	T#10s		
9	VAR	**aktualny_czas**		TIME			
10	VAR	**Reset2**		BOOL			
11	VAR	**Alarm3**		BOOL			

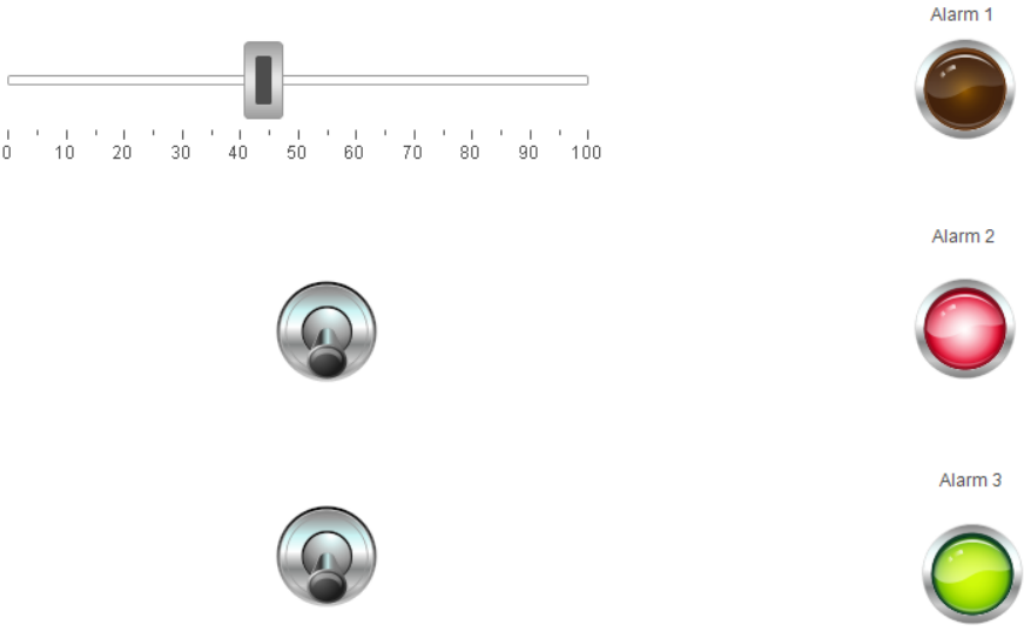
Rys. 3.11. Odpowiednie wartości schematu

Wizualizację uzupełniamy o zieloną diodę, która powinna zapalac się po 10 sekundach od zapalenia się Alarmu 1, oraz odpowiedni przycisk resetujący Alarm 3. Jak we wcze-

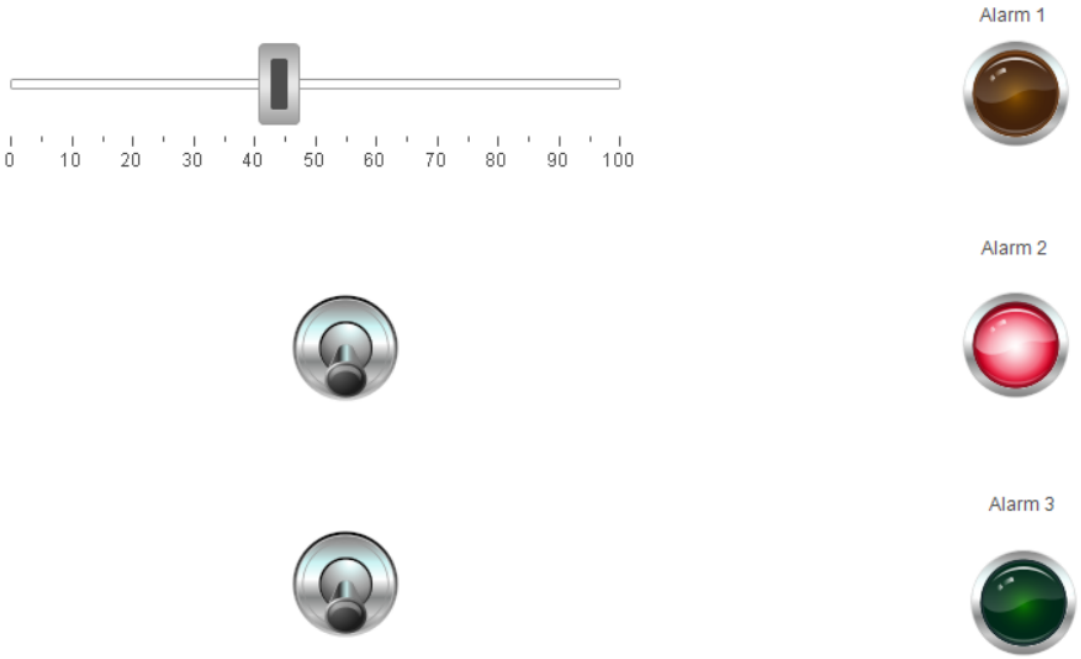
śniejszych etapach również za pomocą symulacji sprawdzamy poprawność wykonanego kodu.



Rys. 3.12. Wizualizacja programu po 10 sekund Alarmu 1 po wyjściu poza zakres



Rys. 3.13. Wizualizacja programu po powrocie do zadanego zakresu



Rys. 3.14. Wizualizacja programu po zresetowaniu Alarmu 3

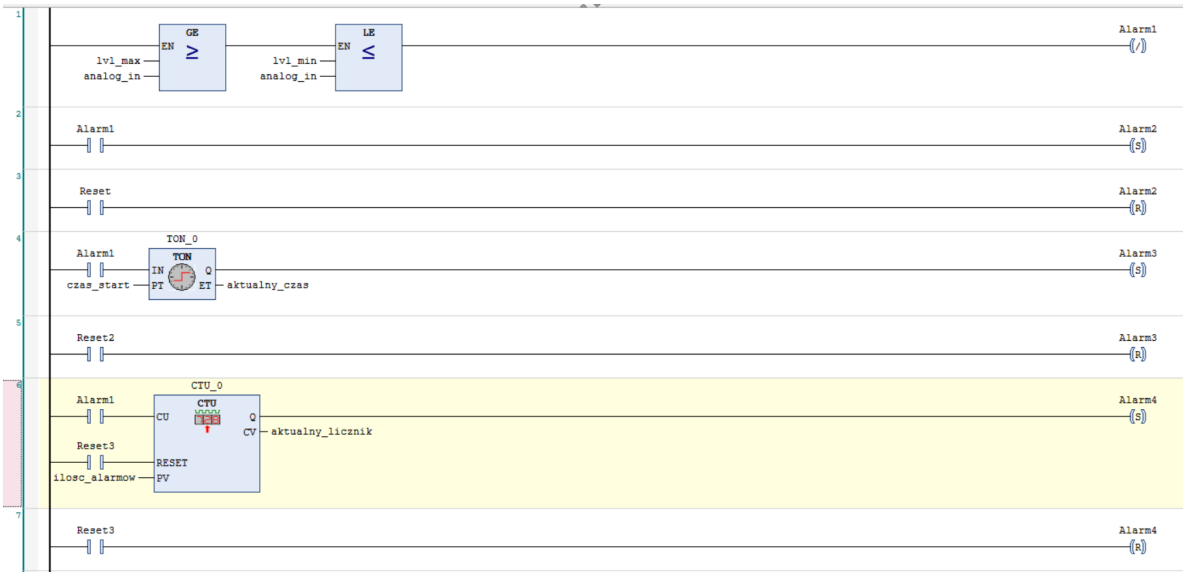
Zauważamy, że zadanie zostało zrealizowane poprawnie. Dioda zielona zapaliła się po 10 sekundach od zaświecenia się diody żółtej. Mimo powrotu sygnału do zakresu i zgaśnięcia Alarmu 1, Alarm 3 pozostał zapalony. Dopiero po zresetowaniu go przyciskiem Reset 2 gaśnie. Natomiast Alarm 2 od momentu wyjścia poza zakres sygnału nie gaśnie, ponieważ przycisk do jego resetu nie został aktywowany, co z kolei świadczy o tym, że nie zepsuliśmy działania wcześniej już napisanego kodu.

3.4 Ćwiczenie 4.4

Kolejne zadanie polegało na zrealizowaniu kolejnego znacznika alarmu, który po dziesięciokrotnym pojawieniu się stanu alarmowego zapalał się. Kasowanie znacznika alarmu miało być realizowane przez operatora.

Aby to wykonać, w lini 8 dodaliśmy blok CTU 0, który był odpowiedzialny za zasygnalizowanie dziesięciokrotnego włączenie się Alarmu 1. Blok CTU potrzebuje na swoim wejsciu dwóch styków: styku odpowiedzialnego za sygnał, który należy zliczać (w naszym przypadku styku o wartości stanu Alarmu 1) i styku Reset 3, który jest odpowiedzialny za zresetowanie licznika. Na koniec podłączyliśmy cewkę typu SET służącą zapaleniu się alarmu, a w następnej linii przycisk reset, oraz cewkę typu RESET

aby móc odświeżyć licznik i wyłączyć Alarm 4.



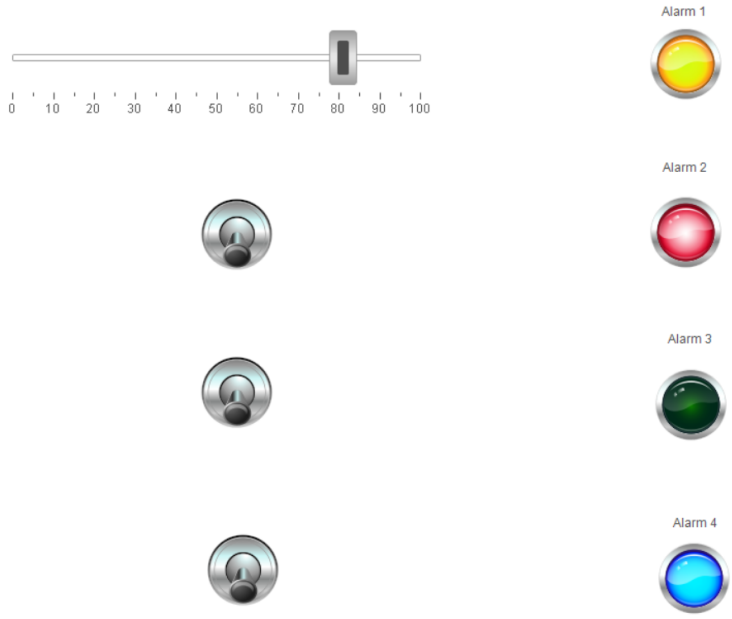
Rys. 3.15. Schemat programu

Po stworzeniu schematu ustawiamy wartość zmiennej "ilość alarmów" na 10 i sprawdzamy czy typ zmiennej "aktualny licznik" oraz "ilość alarmów" jest równy WORD, czy CTU 0 to CTU, i czy "Reset 4" i "Alarm 4" są typu BOOL.

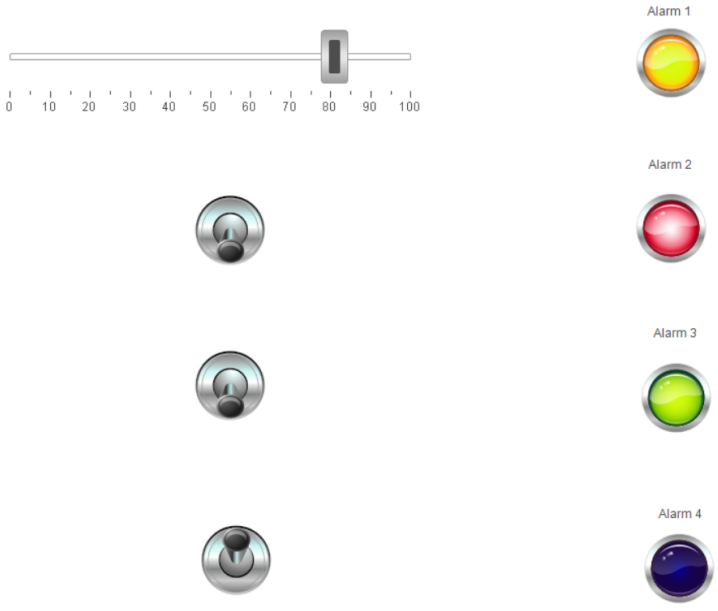
	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**lvl_max**		INT	80		
2	VAR	**lvl_min**		INT	20		
3	VAR	**Alarm1**		BOOL			
4	VAR	**analog_in**		INT			
5	VAR	**Alarm2**		BOOL			
6	VAR	**Reset**		BOOL			
7	VAR	**TON_0**		TON			
8	VAR	**czas_start**		TIME	T#10s		
9	VAR	**aktualny_czas**		TIME			
10	VAR	**Reset2**		BOOL			
11	VAR	**Alarm3**		BOOL			
12	VAR	**Alarm4**		BOOL			
13	VAR	**CTU_0**		CTU			
14	VAR	**Reset3**		BOOL			
15	VAR	**ilosc_alarmow**		WORD	10		
16	VAR	**aktualny_licznik**		WORD			
17	VAR	**Reset4**		BOOL			

Rys. 3.16. Odpowiednie wartości sygnału

Na wizualizacji dodajemy niebieską diodę jako Alarm 4 i przycisk Reset 4 wyzerowujący ją. Następnie sprawdzamy poprawność zapisanego kodu.



Rys. 3.17. Wizualizacja po dziesieciokrotnym wyjściu poza zakres



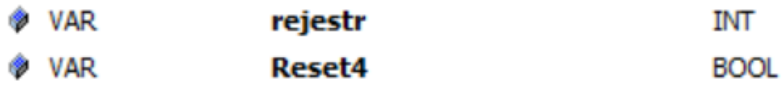
Rys. 3.18. Wizualizacja po odczekaniu 10 sekund zresetowaniu licznika

Zauważamy, że zadanie zostało wykonane poprawnie. Dioda niebieska zapaliła się po

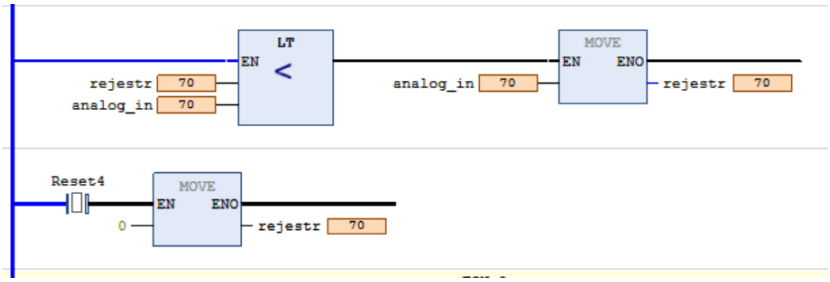
dziesięciokrotnym zapaleniu się diody żółtej. Alarm 4 świeci się do momentu zresetowania go przyciskiem Reset 4. Dodatkowo sprawdzona była też opcja czy licznik się resetuje w trakcie naliczania warości, przed uzyskaniem 10, co za tym idzie przed zapaleniem się Alarmu 4. Ta opcja również jest wykonywana poprawnie. Alarm 2 i Alarm 3 działają tak samo jak w zadaniach wcześniejszych, co świadczy o tym, że nie zepsuliśmy działania wcześniej już napisanego kodu.

3.5 Ćwiczenie 4.5

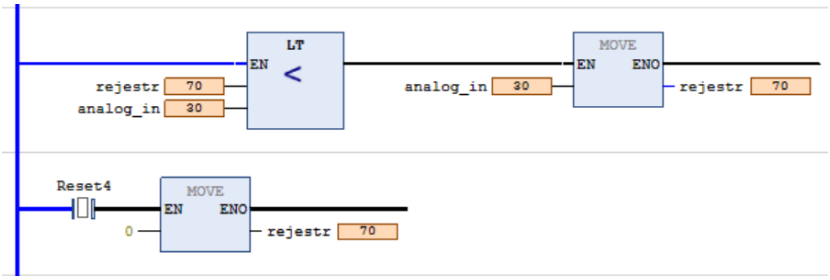
Kolejnym poleceniem była modyfikacja programu w taki sposób, aby maksymalna wartość alarmowego sygnału wejściowego była zapisywana do rejestru sterownika, z możliwością ręcznego wyzerowania rejestru, przez operatora.
Podpunkt ten został zrealizowany za pomocą dodania w kolejnej linii programu bloczka MOVE oraz LT. A opcja resetowania w następnej linijce, także poprzez użycie bloczka MOVE.



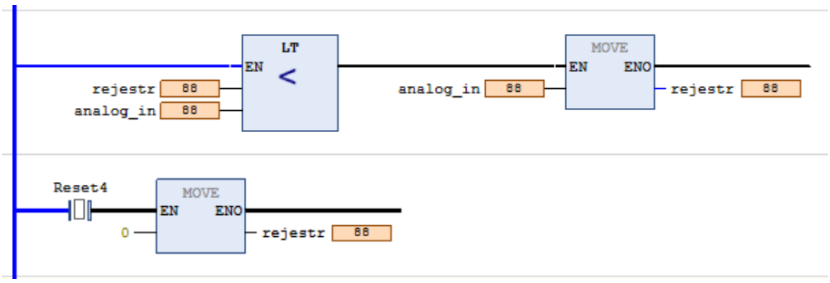
Rys. 3.19. Zmienne stworzone na potrzeby podpunktu z rejestrem



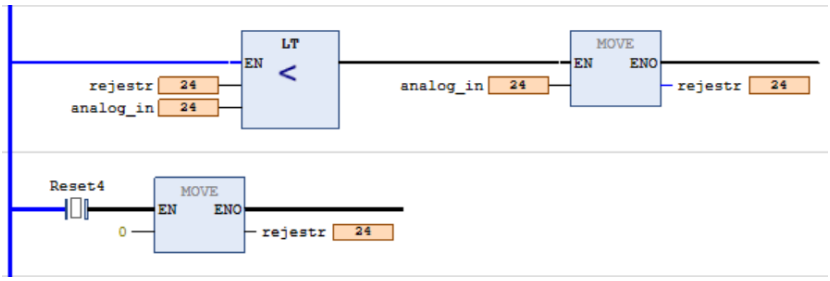
Rys. 3.20. Wizualizacja po ustawieniu wartości 70 na suwaku



Rys. 3.21. Wizualizacja po zmniejszeniu wartości suwakiem



Rys. 3.22. Wizualizacja po zwiększeniu wartości suwakiem powyżej 70



Rys. 3.23. Wizualizacja po wciśnięciu przycisku resetu oraz ustawieniu suwaka na niższą wartość

Jak widać na załączonych wizualizacjach. Program działa w ten sposób, że po ustawieniu wartości na suwaku, zostaje ona zapisana do rejestru. Jeśli zmienimy wartość na mniejszą, sygnał nie zostanie przepuszczony przez bloczek LT ponieważ wartosc będzie mniejsza od aktualnego rejestru. Więc ten się nie zaktualizuje. Jeśli zaś zwiększymy wartość suwaka, bloczek przepuści sygnał i zaktualizuje stan rejestru. Po wciśnięciu przycisku "reset" zmienna rejestr zostanie ustawiona na wartość 0, a z uwagi na wciąż ustawiony suwak zaraz po tej operacji zaktualizuje się do aktualnej wartości suwaka.

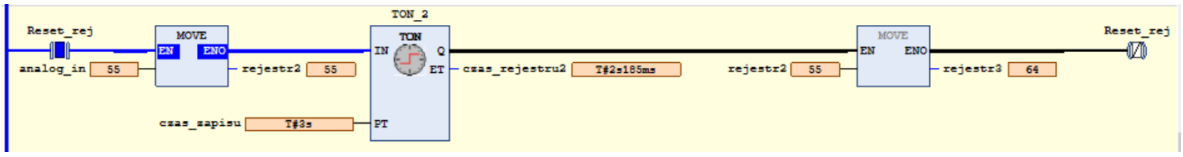
3.6 Ćwiczenie 4.6

Ostatnie zadanie polegało na zmodyfikowaniu programu, aby co ustaloną ilość sekund, w tym wypadku co 3 sekundy, wartośćc podawana na wejście analogowe była zapisywana kolejno do dwóch rejestrów wewnętrznych sterownika, aby znajdowały się tam wartości historyczne w odstępach 3-sekundowych.
Do realizacji tego zadania zostały użyta kombinacja bloczków MOVE, timerów, oraz cewek. Działanie tej linii polega na tym, że bazowo sygnał zostaje przepuszczony i wartość z wejścia "analog_in" zostaje skopiowana do zmiennej "rejestr2". Następnie uruchamia sie TIMER, po odmierzeniu 3 sekund, wartość zapisana wcześniej w zmiennej

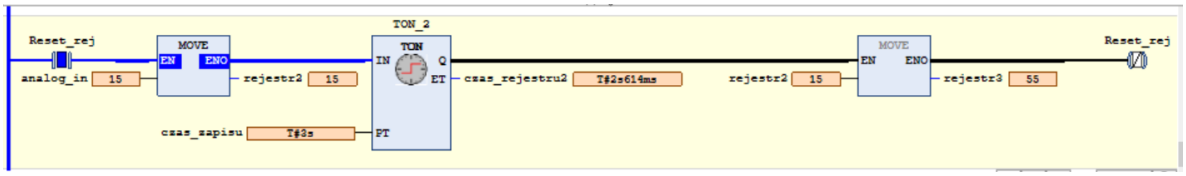
”rejestr2” zostaje skopiowana do zmiennej ”rejestr3”, a później sygnał aktywuje na chwilę zanegowaną cewkę, ustawiając na chwilę wartość ”Reset_rej” na 0, co blokuje sygnał na wejściu i resetuje timer. Dzięki temu zastosowaniu, tworzy się swego rodzaju pętla, magazynująca odpowiednio wartości odczytane z wejścia w dwóch rejestrach z 3-sekundowym odstępem.

Scope	Name	Address	Data type	Initialization	Comment	Attributes
VAR	**rejestr2**		INT			
VAR	**rejestr3**		INT			
VAR	**Reset_rej**		BOOL			
VAR	**TON_2**		TON			
VAR	**czas_rejestru2**		TIME			
VAR	**czas_zapisu**		TIME	T#3S		

Rys. 3.24. Zmienne użyte do realizacji podpunktu 4.6



Rys. 3.25. Wizualizacja przykładowego działania zapisu do 2 rejestrów co 3 sekundy



Rys. 3.26. Wizualizacja przykładowego działania zapisu do 2 rejestrów co 3 sekundy

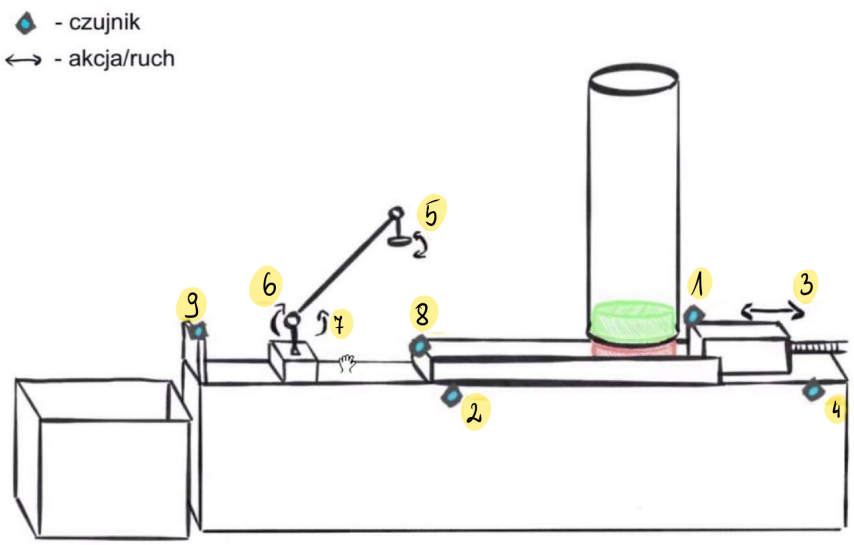
4 Laboratorium 2

4.1 Wstęp

Zadanie zaprezentowane na zajęciach polegało na napisaniu programu odpowiedzialnego za poprawne działanie mechanizmu przenoszącego klocek z podajnika do koszyka za pomocą ramienia. Zadanie zostanie zrealizowane w programie CODESYS.

4.2 Krótki opis działania mechanizmu

Aby uprościć sobie realizację działania, cały mechanizm dzielimy na etapy. Możemy wyróżnić kilka kluczowych punktów.



Rys. 4.1. Schemat zadania

- Uruchomienie programu, czego warunkiem jest odpowiednie położenie tłoka, a także obecność klocka w podajniku [Czujnik_cofniecia_tloka (4), Obecnosc_klocka (1)]
- Moment dopchania klocka przez tłok do końca linii [Czujnik_dopchniecia_tloka (2)]
- Ustawienie się ramienia bezpośrednio nad dopchanym klockiem [Czujnik_cofniecia_ramienia (8)]
- Dojechanie ramienia do pozycji, kiedy znajdować będzie się w punkcie docelowym - nad koszykiem [Ramie_dojechalo (9)]

Wymienione powyżej etapy będą reprezentowane w kodzie przez styki, a w wizualizacji przez przyciski, które na potrzeby późniejszej symulacji będziemy aktywować ręcznie. Podczas działania mechanizmu możemy wyróżnić jeszcze etapy pośrednie. Będą to chociażby:

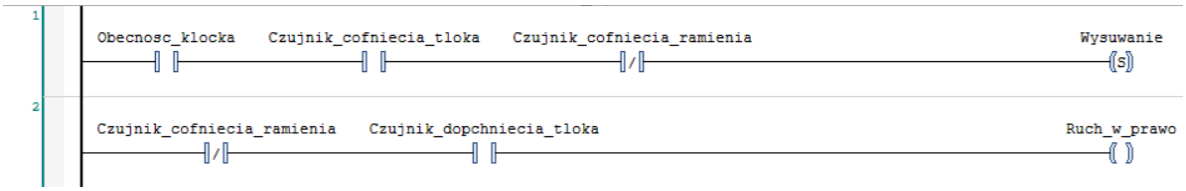
- Czas, w którym tłok będzie wypychany/chowany [Wysuwanie (3)]
- Czas, w którym ramię będzie wykonywało ruch w prawą stronę - nad klocek [Ruch_w_prawo (6)]
- Czas, w którym ramię będzie wykonywało ruch w lewą stronę - do koszyka [Ruch_w_lewo (7)]

- Czas zasysania klocka [ssanie (5)]

Wymienione powyżej etapy pośrednie będą w programie reprezentowane poprzez cewki, a w wizualizacji przez diody.
Dodatkowo w programie zawrzemy też cewki oraz diody alarmowe, służące poinformowaniu operatora o zbyt długim czasie wykonywania się ruchu w prawo/w lewo ramienia, a także o zbyt długim czasie wysuwania się tłoka. Wszystkie alarmy są resetowane ręcznie przez operatora.

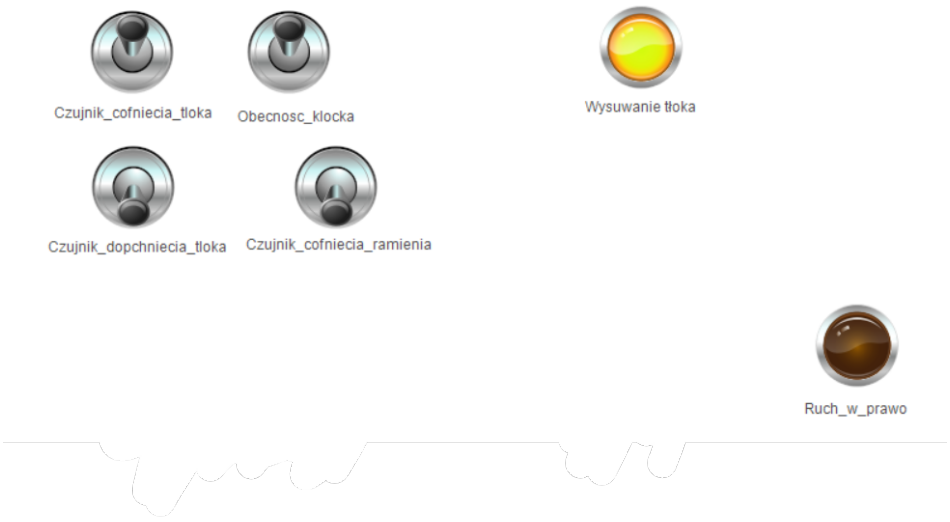
4.3 Realizacja zadania oraz symulacje

Na początek zajmijmy się pierwszym etapem programu, czyli warunkami rozpoczęcia jego działania. Aby program zaczął działać i tłok zaczął się wysuwać muszą być spełnione oba warunki - obecność klocka, a także obecność tłoka w pozycji cofniętej. Co w programie prezentuje się tak:



Rys. 4.2. Fragment kodu 1

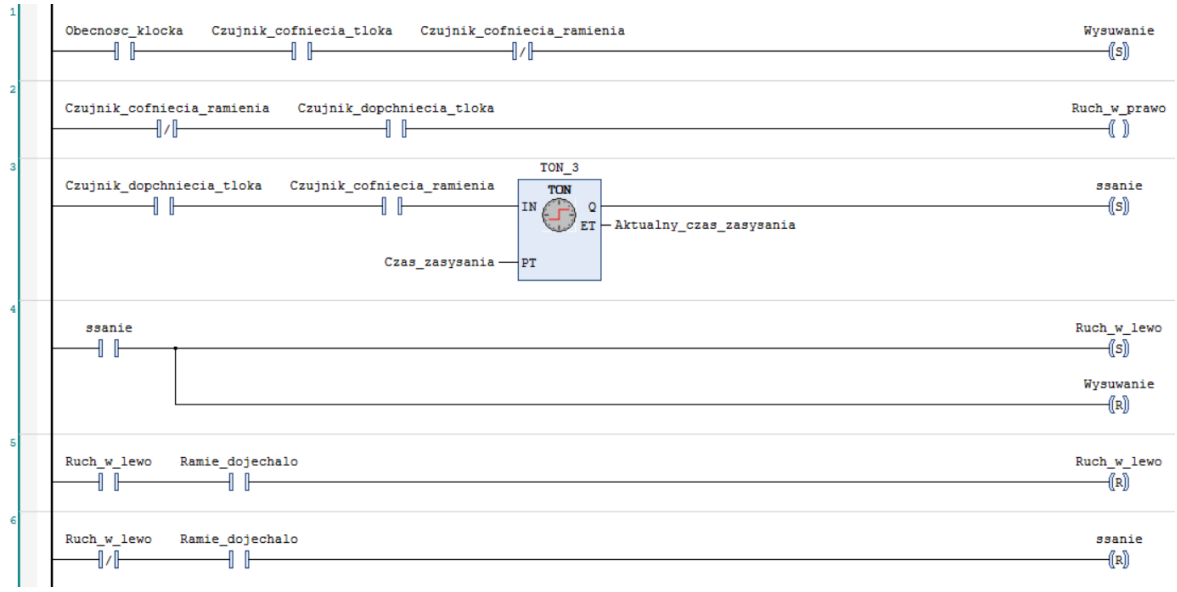
Aby sprawdzić poprawność działania programu aktywujemy symulację.



Rys. 4.3. Symulacja wysuwania się tłoka

Jak widac na powyższej symulacji, po aktywowaniu obydwu warunków, dioda zaczyna świecić, co oznacza wysuwanie się tłoka.

Następnie zajmiemy się kolejnymi etapami tj. momentemm w którym klocek zostanie dopchnięty do końca przez tłok. Przemieszczeniem się ramienia, zassaniem klocka, a także ruchem ramienia w stronę koszyka. Na końcu zajmiemy się stanem kiedy ramię znajdzie się nad koszykiem. Wszystkie te etapy w programie prezentują sie następująco:



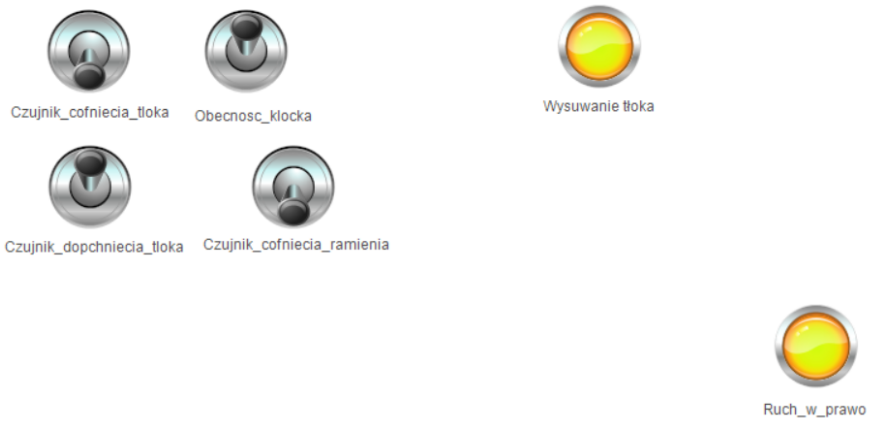
Rys. 4.4. Fragment kodu 2

W momencie kiedy klocek zostanie dopchnięty aktywuje się Czujnik_dopchniecia_tloka. Aktywacja tego czujnika spowoduje wysłanie sygnału do ramienia, że klocek jest w odpowiedniej pozycji, więc zacznie ono Ruch_w_prawo, aby ostatecznie znaleźć się nad klockiem. W momencie kiedy ramię znajdzie się nad klockiem aktywuje się Czujnik_cofniecia_ramienia, co spowoduje zakończenie Ruchu_w_prawo i uruchomienie Timera odliczającego 1 sekundę, potrzebną na zassanie się ramienia do klocka.

Po aktywacji ssania "Wysuwanie" zostaje ustawione na 0, co oznacza chowanie się tłoka do pozycji początkowej. Jednocześnie rozpoczyna się Ruch_w_lewo ramienia. Kiedy ramię znajdzie się nad koszykiem aktywuje się styk "Ramie_dojechalo", spowoduje to zakończenie Ruchu_w_lewo (ustawienie cewki na 0), a także zakończenie ssania, co oznacza opuszczenie klocka do koszyka.

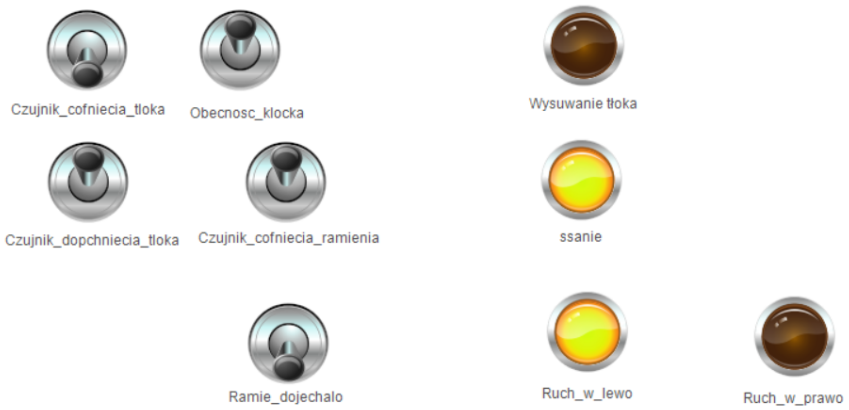
Po skończeniu cyklu ramię zostanie w pozycji nad koszykiem. Jeśli tłok wróci do położenia początkowego a w podajniku nadal będą znajdować się klocki (oba styki zostaną aktywowane) cały cykl zacznie się od początku.

Przejdźmy zatem do symulacji zaproramowanego zachowania.



Rys. 4.5. Symulacja dojechania tłoka

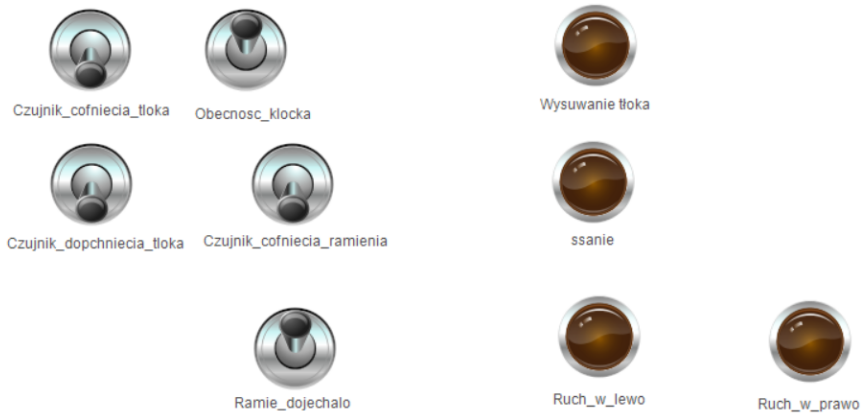
Jak widzimy powyżej, po aktywacji obydwu przycisków i rozpoczęciu wysuwania się tłoka dioda "Wysuwanie tłoka" była zapalona. Tłok ruszył, więc Czujnik_cofniecia_tloka został wyłączony. Po dojechaniu tłoka do końca (podniesieniu przełącznika Czujnik_dopchniecia_tloka) rozpoczął się ruch w prawą stronę (zapaliła się dioda Ruch_w_prawo



Rys. 4.6. Symulacja znalezienia się ramienia nad klockiem, zassania go, oraz rozpoczęcia ruchu w lewą stronę

Zdjęcie powyżej prezentuje moment dojechania ramienia nad klocek i chwile po nim. Po

aktywacji Czujnika_cofniecia_ramienia gaśnie dioda "Wysuwanie tłoka" co oznacza jego cofanie się. Po sekundzie zapala się dioda "ssanie" co oznacza, że klocek jest chwycony przez ramię, a także zapala się dioda "Ruch w lewo" oznaczająca rozpoczęcie ruchu ramienia w lewą stronę.



Rys. 4.7. Symulacja dojechania ramienia do końca i opuszczenia klocka do koszyka

Ostatnie zdjęcie przedstawia moment dojechania ramienia do końca, tj. znalezienia się go w pozycji nad koszykiem. Po rozpoczęciu poprzedniego etapu Czujnik_cofniecia_ramienia został wyłączony ponieważ to było już w trakcie ruchu. Po aktywacji czujnika "Ramie_dojechalo" gaśnie dioda "ssanie" a także "Ruch w lewo". Symbolizuje to, że ramię znalazło się nad koszykiem, skończyło swój ruch i opuściło klocek. Po przeprowadzonej symulacji widać, że przygotowany program działa zgodnie z pierwotnymi założeniami.

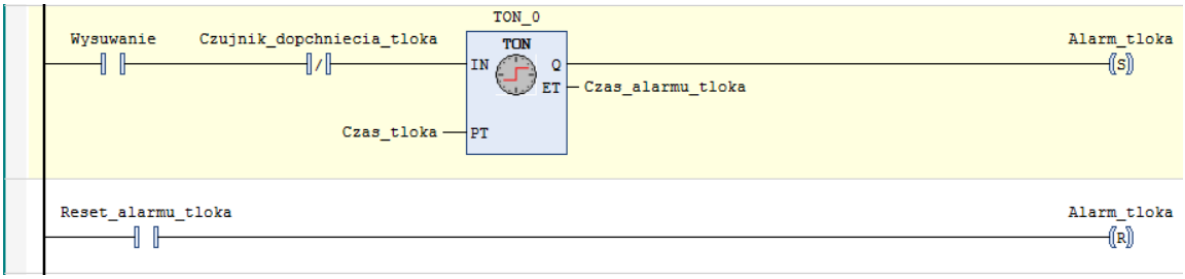
4.4 Alarmy

Dodatkowo w zadaniu należało dodać 3 alarmy:

4.4.1 Alarm tłoka

Miał włączać się w momencie gdy nasz tłok przesuwał klocek dłużej niż 4 sekundy, lub kiedy klocek w ogóle nie dociera do czujnika dopchnięcia tłoka. W celu zrealizowania tego alarmu dodaliśmy timer, który zaczynał odliczanie w momencie gdy zaczynało się "wysuwanie" (co świadczyło o rozpoczęciu przesuwania się tłoka) i gdy styk czujnika_dopchnięcia_tłoka był w stanie 0 (co świadczyło o tym, że klocek nie dotarł jeszcze na miejsce). Czas tłoka ustawiliśmy odpowiednio jako T4s a na wyjście dodaliśmy cewke

typu SET o nazwie Alarm_tłoka, aby zapamietywała jego stan do momentu zresetowania go w kolejnej linii kodu przez styk Reset_alarmu_tłoka.



Rys. 4.8. Schemat Alarmu tłoka

W wizualizacji dodalismy dodatkowo diodę odpowiadającą za Alarm_tłoka i przycisk odpowiadjący Reset_alarmu_tłoka, a następnie sprawdziliśmy poprawość działania kodu.

Rys. 4.9. Wizualizacja gdy wysuwanie trwa dłużej niż 4 sekundy, a czujnik_dopchnięcia_tłoka dalej nie jest włączony

Rys. 4.10. Wizualizacja po opuszczeniu klocka do koszyka

Rys. 4.11. Wizualizacja po wciśnięciu Reset_alarmu_tłoka

Jak można zauważyć Alarm_tłoka zgodnie z założeniami zapala się, w momencie gdy wysuwanie zaczyna się, a po upływie 4 sekund czunik_dopchnięcia_tłoka dalej nie jest w stanie 1. Po całkowitym zakończeniu działania programu, czyli w momencie kiedy klocek jest już w koszyku Alarm_tłoku dalej się świeci i wyzerowywuje się dopiero w momencie wciśnięcia przycisku Reset_alarmu_tłoku, co świadczy o dobrze wykonanym zadaniu. Dodatkowo każde z poprzednich zadań również działa, co wskazuje na to że nie zepsuliśmy dotychczas napisanego przez nas kodu.

4.4.2 Alarm podczas przesuwania się ramienia w prawo

Miał włączać się w momencie gdy nasze ramię przesuwało się z klockiem w prwo dłużej niż 6 sekund, lub kiedy ramię w ogóle nie docierało do czujnik_cofania_ramienia. Żeby tak się stało dodaliśmy timer, który zaczynał odliczanie w momencie gdy zaczynał się ruch w prwo. Czas_ramienia ustawiliśmy odpowiednio jako T6s a na wyjście dodaliśmy cewke typu SET o nazwie Alarm_ramienia_lewy, aby zapamietywała jego stan do momentu zresetowania go w kolejnej linii kodu przez styk Reset_alarmu_ramienia_prawy.

Rys. 4.12. Schemat Alarmu ruchu w prwo

W wizualizacji dodalismy dodatkowo diodę odpowiadającą za Alarm_ramienia_prawy i przycisk odpowiadjący Reset_alarmu_ramienie_prawy, a następnie sprawdziliśmy poprawość działania kodu.

Rys. 4.13. Wizualizacja gdy ruch w prawo trwa dłużej niż 6 sekund

Rys. 4.14. Wizualizacja po opuszczeniu klocka do koszyka

Rys. 4.15. Wizualizacja po wciśnięciu Reset_alarmu_ramienia$_p$*rawy*

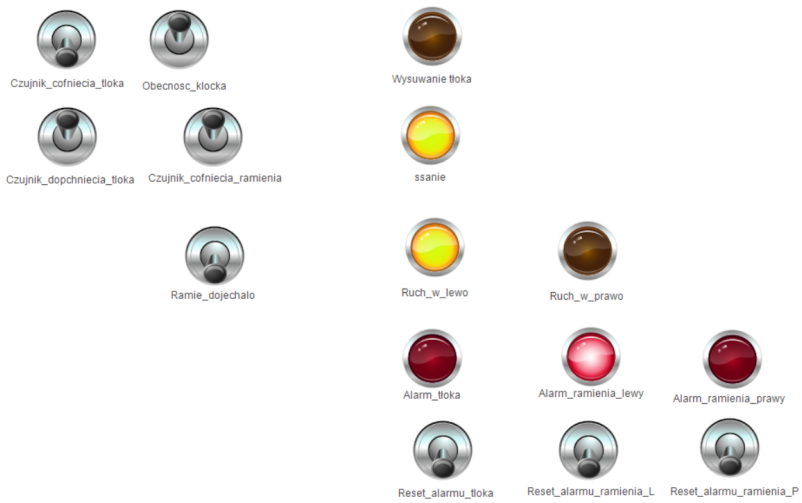
Jak można zauważyć Alarm_ramienia_prawy zgodnie z założeniami zapala się, w momencie gdy ruch w prawo zaczyna się i po upływie 6 sekund dalej trwa. Po całkowitym zakończeniu działania programu, czyli w momencie kiedy klocek jest już w koszyku Alarm_ramienia_prawy dalej się świeci i wyzerowywuje się dopiero w momencie wciśnięcia przyciusku Reset_alarmu_ramienia_P, co świadczy o dobrze zrealizowanym zadaniu. Dodatkowo każde z poprzednich zadań również działa, co wskazuje na to że nie zepsuliśmy dotychczas napisanego przez nas kodu.

4.4.3 Alarm podczas przesuwania się ramienia w lewo

Miał włączać się w momencie gdy nasze ramię przesuwało się z klockiem w lewo dłużej niż 6 sekund, lub kiedy ramię w ogóle nie docierało do czujnika Ramie_dojechalo. Żeby tak się stało dodaliśmy timer, który zaczynał odliczanie w momencie gdy zaczynał się ruch w lewo. Czas_ramienia ustawiliśmy odpowiednio jako T6s a na wyjście dodaliśmy cewke typu SET o nazwie Alarm_ramienia_lewy, aby zapamietywała jego stan do momentu zresetowania go w kolejnej linii kodu przez styk Reset_alarmu_ramienia_lewy.

Rys. 4.16. Schemat Alarmu ruchu w lewo

W wizualizacji dodalismy dodatkowo diodę odpowiadającą za Alarm_ramienia_lewy i przycisk odpowiadający Reset_alarmu_ramienie_lewy, a następnie sprawdziliśmy poprawość działania kodu.



Rys. 4.17. Wizualizacja gdy ruch w lewo trwa dłużej niż 6 sekund

Rys. 4.18. Wizualizacja po opuszczeniu klocka do koszyka

Rys. 4.19. Wizualizacja po wciśnięciu Reset_alarmu_ramienia*lewy*

Jak można zauważyć Alarm_ramienia_lewy zgodnie z założeniami zapala się, w momencie gdy ruch w lewo zaczyna się i po upływie 4 sekund dalej trwa. Po całkowitym zakończeniu działania programu, czyli w momencie kiedy klocek jest już w koszyku Alarm_ramienia_lewy dalej się świeci i wyzerowywuje się dopiero w momencie wciśnięcia przycisku Reset_alarmu_ramienia_L, co świadczy o dobrze zrealizowanym zadaniu. Dodatkowo każde z poprzednich zadań również działa, co wskazuje na to że nie zepsuliśmy dotychczas napisanego przez nas kodu.

4.5 Wartosci

Przy każdym działaniu sprawdzamy czy bloki, styki i cewki mają odpowiednie typy

	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**Obecnosc_klocka**		BOOL			
2	VAR	**Wysuwanie**		BOOL			
3	VAR	**Czujnik_cofniecia_tloka**		BOOL			
4	VAR	**Czujnik_dopchniecia_tloka**		BOOL			
5	VAR	**Czujnik_cofniecia_ramienia**		BOOL			
6	VAR	**ssanie**		BOOL			
7	VAR	**Ruch_w_lewo**		BOOL			
8	VAR	**Ramie_dojechalo**		BOOL			
9	VAR	**Ruch_w_prawo**		BOOL			
10	VAR	**TON_0**		TON			
11	VAR	**Czas_tloka**		TIME	T#4s		
12	VAR	**Czas_alarmu_tloka**		TIME			
13	VAR	**Alarm_tloka**		BOOL			
14	VAR	**TON_1**		TON			
15	VAR	**Czas_ramienia**		TIME	T#6s		
16	VAR	**Czas_ruchu_ramienia**		TIME			
17	VAR	**TON_2**		TON			
18	VAR	**Alarm_ramienia_prawy**		BOOL			
19	VAR	**Alarm_ramienia_lewy**		BOOL			
20	VAR	**Reset_alarmu_ramienia_prawy**		BOOL			
21	VAR	**Reset_alarmu_ramienia_lewy**		BOOL			
22	VAR	**Reset_alarmu_tloka**		BOOL			
23	VAR	**TON_3**		TON			
24	VAR	**Czas_zasysania**		TIME	T#1s		
25	VAR	**Aktualny_czas_zasysania**		TIME			

Rys. 4.20. Wartosci w zadaniu

5 Wnioski

Wszystkie zadania zostały wykonane poprawne, zapoznaliśmy się z działaniem bloków matematycznych (GE,LE,GT,LT), bloków kopiujących, timerów, counterów, różnych styków i cewek, można zatem powiedzieć ze dobrze posługujemy się funkcjami i możliwościami programu CODESYS oraz mamy podstawowe umiejętności w pragramowaniu Sterowników GE Fanuc.

6 Literatura

- Dokument z treścią zadania: http://pawel.dobrowolski.staff.iiar.pwr.wroc.pl/Urzadzenia%20obiektowe%20automatyki/5.%20GE%20Fanuc.pdf