Marcin Tomasik, Henryk Juszka, Stanisław Lis

STEROWANIE I WIZUALIZACJA ROLNICZYCH PROCESÓW PRODUKCYJNYCH

MATERIAŁY NAUKOWO-DYDAKTYCZNE

Kraków 2013

Recenzenci:
Prof. dr hab. inż. Janusz Piechocki – UWM Olsztyn
Dr hab. inż. Deta Łuczycka – UP Wrocław

„Praca naukowa finansowana ze środków na naukę w latach 2008-2013

jako projekt badawczy N N313-154435" oraz z „Dotacji projakościowej dla WIPiE/06097"

ISBN 978-83-935020-6-6

Projekt okładki: dr inż. Marcin Tomasik

Opracowanie redakcyjne: dr hab. inż. Maciej Kuboń

Redaktor językowy: mgr Mirosław Grzegórzek

Wydanie I
Druk i oprawa:
DRUKROL S C., Kraków, Ujastek 9
Tel./fax. (12) 412 46 50, e-mail: drukarnia@drukrol.pl; www.drukrol.pl
Ark. wyd. 16,6, ark. druk. 15,0
Nakład: 200 egz.

SPIS TREŚCI

PRZEDMOWA

Współczesne systemy sterowania realizowane są przy pomocy programowalnych sterowników logicznych i komputerowych stacji operatorskich, pełniących rolę systemów kontrolno-nadzorczych. Systemy takie wykorzystują oprogramowanie SCADA do realizacji gromadzenia i przetwarzania danych pochodzących z procesu, wizualizacji jego stanu, sterowania nadrzędnego, alarmowania i rejestracji zdarzeń, archiwizacji danych oraz udostępniania informacji o procesie w sieciach komputerowych.

Adresatem niniejszej książki są głównie Studenci zdobywający wiedzę na temat programowalnych sterowników logicznych oraz systemów wizualizacji i kontroli procesów produkcyjnych. Obecnie na kierunkach studiów technika rolnicza i leśna (specjalność mechatronika oraz zarządzanie i inżynieria produkcji) istnieją moduły kształcące w zagadnieniach projektowania, wdrażania i użytkowania tych urządzeń.

Książka podzielona została na cztery części. Pierwsza część zawiera podstawowe informacje dotyczące budowy, działania i programowania sterowników logicznych. Druga część zawiera praktyczne informacje dotyczące montażu, użytkowania oraz programów narzędziowych, umożliwiających ich programowanie. Trzecią część poświęcono na przedstawienie przykładów opisujących możliwości stosowania tych urządzeń w inżynierii rolniczej. W czwartej części omówiono systemy wizualizacji i kontroli procesów produkcyjnych wraz z kilkoma przykładami.

Zagadnienia dyscypliny naukowej, jaką jest inżynieria rolnicza, obejmują wiele obszarów badawczych, często łączących tematykę rolniczą z techniczną. W zarządzaniu i sterowaniu procesami produkcyjnymi sterowniki są umieszczane najbliżej linii, dostarczają informacji do systemów zarządzających produkcją. Pełnią one rolę warstwy komunikującej pomiędzy linią produkcyjną, a oprogramowaniem wspomagającym procesy zarządzania.

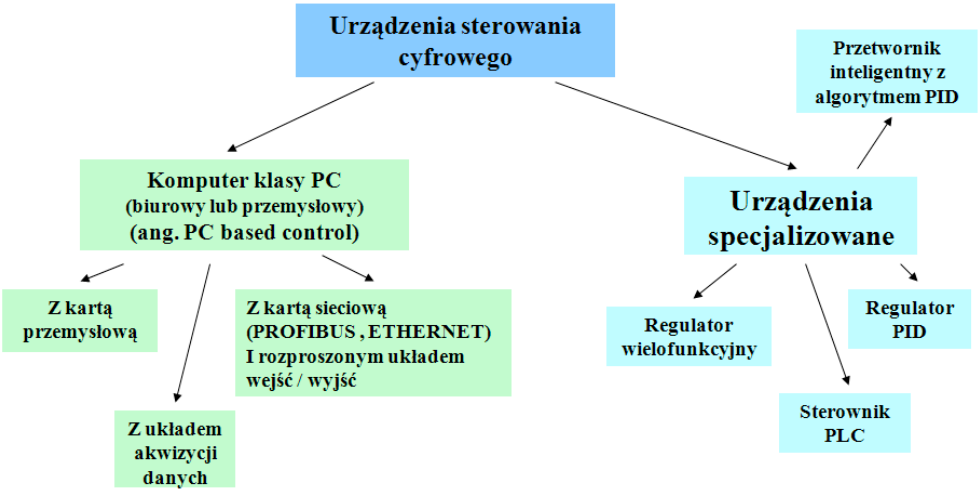
Przykładowo sterowniki PLC są stosowane w następujących aplikacjach sterujących:

- zautomatyzowanym procesem doju krów;
- mikroklimatem pomieszczeń inwentarskich;
- zautomatyzowanym systemem pojenia i zadawania pasz;
- mikroklimatem w obiektach produkcyjnych (oborach, fermach, szklarniach);
- systemami nawadniania upraw polowych i szklarniowych;
- liniami produkcyjnymi w przemyśle spożywczym;
- systemami zautomatyzowanego pakowania;
- systemami energetycznymi: kolektory słoneczne, fotoogniwa, produkcja biogazu i biopaliw, hybrydowe systemy energetyczne;
- wybranymi funkcjami w ciągnikach i maszynach rolniczych itd.

CZĘŚĆ PIERWSZA
– PODSTAWY TEORETYCZNE STEROWNIKÓW PLC

1. WPROWADZENIE

Sterowniki programowalne PLC *(ang. Programmable Logic Controllers)* są układami mikroprocesorowymi z programowalną pamięcią, powszechnie stosowanymi w automatyce do sterowania maszynami, urządzeniami w procesach przemysłowych. W swojej historii przeszły bardzo głęboką ewolucję – od programowalnych układów sterowania binarnego, zastępujących „przekaźnikowe szafy sterownicze" – do złożonych systemów mikrokomputerowych, realizujących oprócz zadań sterowania logicznego, skomplikowane zadania regulacji cyfrowej, obliczeń, diagnostyki i komunikacji w zdecentralizowanym systemie kompleksowej automatyzacji (Juszka, 2006). Na rysunku 1 zamieszczono diagram prezentujący klasyfikację cyfrowych systemów sterowania.



Rysunek 1. Klasyfikacja urządzeń sterowania cyfrowego

Początki historii programowalnych sterowników logicznych PLC sięgają do 1968 r. Wówczas grupa inżynierów zatrudnionych w koncernie General Motors opracowała innowacyjne rozwiązanie zastępujące ówczesne układy sterowania sekwencyjnego zawierające przekaźniki (Łukasik i Seta, 2001). Zastosowanie praktyczne tych urządzeń związane jest z ich cechami eksploatacyjnymi, jak:

– skalowalność systemów sterowania;
– łatwość programowania i przeprogramowywania do nowych zadań na liniach produkcyjnych;
– łatwość utrzymania w ruchu produkcyjnym z możliwością napraw przez wymianę modułów (budowa modułowa) (rys. 2).

Rysunek 2. *Modułowy sterownik PLC firmy GE-Fanuc*

Źródło: Peszyński i Siemieniako, 2002

Przed pojawieniem się PLC wiele zadań kontrolnych było rozwiązywanych przez łączone ze sobą styczniki lub przekaźniki (Ruda i Olesiński, 2003). Taki sposób sterowania nazywany jest często *sterowaniem konwencjonalnym*. Zasada pracy konwencjonalnego układu sterowania jest określona przez trwałe połączenie aparatury stycznikowo-przekaźnikowej i elementów obiektowych. Okablowanie układu jednoznacznie i trwale określa sposób jego funkcjonowania. Jakiekolwiek zmiany lub rozbudowa układu sterowania wymagają uzupełnienia aparatury kontrolnej i ponownego okablowywania. Takie same, a także bardziej skomplikowane zadania mogą być wykonane za pomocą PLC. Okablowanie połączeń logicznych pomiędzy urządzeniami i stykami przekaźników wykonywane jest w postaci programu zapisanego w pamięci PLC. Na zewnątrz wymagane jest jedynie proste podłączenie aparatury obiektowej do wejścia i wyjścia sterownika. Opracowanie aplikacji sterującej i usuwanie w niej błędów jest znacznie łatwiejsze niż w sterowaniu konwencjonalnym. Znacznie łatwiej tworzy się i modyfikuje program w PLC, niż zmienia okablowanie układu (Broel-Plater, 2000).

W 1976 roku wprowadzono sterowniki wyposażone w kasety sterowania zdalnego, które umożliwiły monitorowanie i uaktualnianie dużej liczby wejść oraz wyjść za pomocą połączeń komunikacyjnych przy odległościach nawet do kilkuset metrów od jednostki centralnej. Rok później po raz pierwszy zastosowano mikroprocesor 8080 z wykorzystaniem dodatkowego koprocesora dla operacji binarnych (Seta, 2002).

Dynamiczny rozwój elektroniki w latach osiemdziesiątych ubiegłego stulecia spowodował popularyzację programowalnych sterowników logicznych w systemach automatyki. Wprowadzono w sterownikach inteligentne moduły I/O, które wyposażono we własne procesory, dzięki czemu realizowały one znacznie bardziej złożone funkcje obliczeniowe. Przez kolejne lata sterowniki PLC rozwijały się, obecnie niemal całkowicie wyparły układy przekaźnikowe i przejmują funkcje regulatorów analogowych. Te ostatnie stosuje się tylko wtedy, gdy ekonomicznie nieuzasadnione jest stosowanie PLC. Duże oddziaływanie na rozwój sterowników PLC miały następujące fakty:

– schematy drabinkowe używane w programowaniu PLC są analogiczne do stosowanych w schematach stykowo-przekaźnikowych;
– zwiększenie niezawodności komputerów przemysłowych w zanieczyszczonym środowisku;
– możliwość kontrolowania poprawności obwodów wejściowych i wyjściowych;
– posiadanie w oprogramowaniu narzędziowym specjalnego zbioru instrukcji uwzględniających warunki przemysłowe;
– możliwość komunikacji operatorów procesu z systemami sterowania poprzez urządzenia typu HMI (Human Machine Interface) (Kasprzyk, 2011).

Wraz z pojawieniem się systemów operacyjnych bazujących na „oknach" ruszyły prace nad programami wspierającymi sterowniki PLC. Do takich programów można zaliczyć system sterowania nadrzędnego i archiwizacji danych tzw. SCADA *(ang. Supervisory Control and Data Aquisition)* (Jakuszewski, 2006). System ten uzupełnia i poszerza możliwości sterowników poprzez następujące funkcje:
– zbieranie, przetwarzanie oraz archiwizację danych;
– sporządzanie raportów aktualnego stanu procesu;
– wizualizację bieżących i historycznych wartości zmiennych procesowych;
– sygnalizowanie przekroczenia wartości granicznych;
– generowanie danych dla warstw operatywnego sterowania produkcją i zarządzania.

Obecnie sterowniki PLC stanowią najbardziej rozpowszechnioną jednostkę przetwarzającą w systemach sterowania. Jednocześnie zaciera się granica w możliwościach funkcjonalnych i mocach obliczeniowych pomiędzy sterownikami PLC, komputerami przemysłowymi i komputerami klasy PC (Ścibisz, 2009). Można zauważyć postępujący proces unifikacji sterowników z akcentowaniem takich cech, jak: niezawodność, uniwersalność, otwartość i kompatybilność z innymi sterownikami oraz duże możliwości komunikacyjne. Realizacja złożonych procesów sterowania nie stanowi już wielkiego problemu w dobie dynamicznego rozwoju mikroprocesorowych systemów sterowania. Chociaż architektura sterowników PLC została zaczerpnięta z komputerów PC, to współczesne rozwiązania są w istocie komputerami, ale nastawione są na realizację określonych zadań (sterowanie procesem) przy założeniu jak najkrótszego czasu realizacji programu sterowania. Oczywiście nieskomplikowane w aplikacji sprzętowej oraz programowej systemy oparte na układach procesorowych mogą podważać zasadność stosowania sterowników programowalnych. Pewne nieskomplikowane procesy mogą korzystać z zasobów np. komputerów jednoukładowych, jednakże stosowanie układów typu PLC umożliwia:
– skalowanie systemu;
– dynamiczną kontrolę poprawności sterowania;
– szybką lokalizację i usuwanie usterek (autodiagnostyka);
– zastosowanie standardowych czujników i elementów wykonawczych;
– łatwą modyfikację programu sterowania bez konieczności ingerencji w platformę sprzętową;
– rozbudowę systemu sterowania do postaci systemu rozproszonego itp. (Sałat i in., 2010).

Programowalne sterowniki logiczne jako komputery przemysłowe pracują pod kontrolą systemu operacyjnego. Rodzaj pracy wymusza, aby był to system czasu rzeczywistego (Kwaśniewski, 2008).

Zasada działania PLC polega na reagowaniu na zmiany stanu sygnałów wejściowych i obliczaniu aktualnych sygnałów wyjściowych zgodnie z przyjętym algorytmem programu zawierającym reguły sterowania lub regulacji. Wartości pomiarów zmiennych procesowych są odbierane przez wejścia sterownika PLC, natomiast obliczone zmienne sterujące trafiają na wyjścia sterownika. Praca jest cykliczna, sterownik wykonuje kolejno po sobie pojedyncze rozkazy programu w takiej kolejności, w jakiej są one zapisane w programie. Na początku każdego cyklu program odczytuje „obraz" stanu wejść sterownika i zapisuje ich stany (obraz wejść procesu). Po wykonaniu wszystkich rozkazów i określeniu (wyliczeniu) aktualnego dla danej sytuacji stanu wyjść, sterownik wpisuje stany wyjść do pamięci będącej obrazem wyjść procesu, a system operacyjny zmienia stan logiczny odpowiedniego wyjścia sterującego elementami wykonawczymi. Zatem wszystkie połączenia sygnałowe spotykają się w układach (modułach) wejściowych sterownika, a program śledzi ich obraz i reaguje zmianą stanów wyjść w zależności od algorytmu. Na algorytm programu sterującego składają się głównie: operacje arytmetyczno-logiczne oraz relacje czasowe pomiędzy zmiennymi procesowymi (Mikulczyński, 2006).

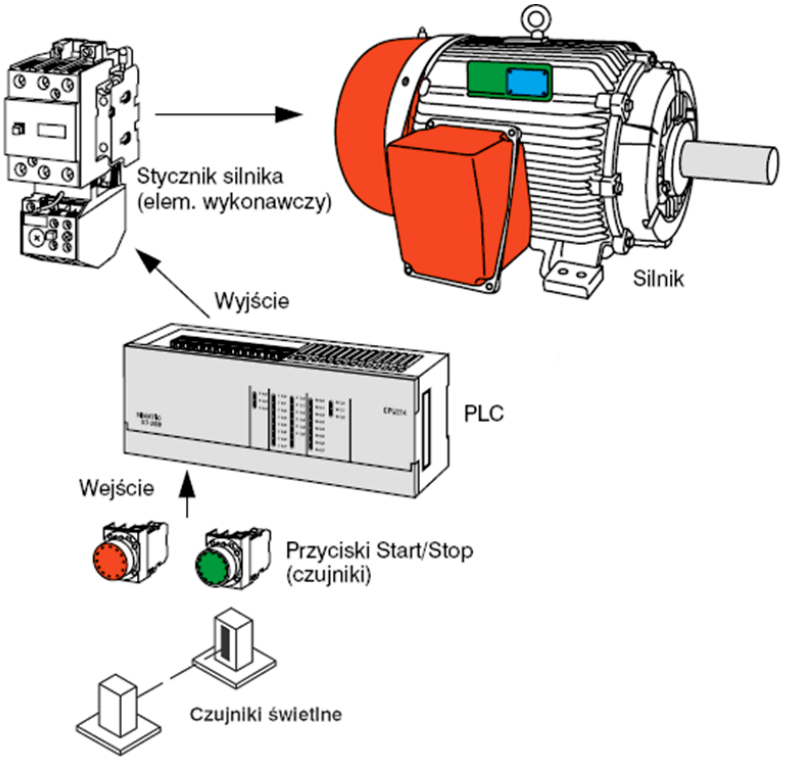
PLC znajdują zastosowanie praktycznie niemal we wszystkich gałęziach przemysłu w tym w szeroko pojmowanym rolnictwie oraz rolno-spożywczych procesach produkcyjnych. Dzięki dużej konkurencji na rynku sterowniki PLC stają się coraz tańsze i bardziej dostępne.

W prezentowanym na rysunku 3 przykładzie przyciski (czujniki) podłączone do wejścia PLC mogą być użyte do uruchomienia lub zatrzymania silnika dołączonego do PLC poprzez stycznik silnika, który spełnia rolę urządzenia wykonawczego. Stąd rolą tych urządzeń jest pośredniczenie w przekazywaniu informacji pomiędzy przyciskami, czujnikami, a uruchamianymi elementami wykonawczymi.

Oferta PLC obejmuje wiele typów różnej „wielkości" modeli sterowników obejmujących zarówno „małe" (mikro, mini) zintegrowane systemy typu kompakt (o liczbie we/wy rzędu kilkunastu), jak i „duże" systemy modułowe (konfigurowane w zależności od potrzeb użytkownika), mogące realizować złożone zadania sterowania binarnego, zadania regulacyjne, komunikacyjne (praca w sieci), jak i złożone obliczenia optymalizacyjne.

Wiodącą rolę wśród producentów sterowników PLC pełnią takie firmy jak: Siemens, Allen-Bradley, GE Intelligent Platforms, Moeller, Mitsubishi, AEG - Modicon, Omron.

Producenci dla zwiększenia niezawodności systemów sterowania PLC opracowali *systemy redundancji* polegające na podwojeniu elementów systemu sterowania oraz skomunikowaniu ich wzajemnie. Występują one w wersji sprzętowej i programowej. Redundancja jest to duplikacja elementów – podsystemów procesu technologicznego, mająca na celu podniesienie niezawodności (Grzywak, 1994).



Rysunek 3. PLC w sterowaniu elementami procesów technologicznych

2. OPIS CZĘŚCI SKŁADOWYCH MIĘDZYNARODOWEJ NORMY IEC 61131

Powszechne stosowanie sterowników PLC w skomplikowanych procesach produkcyjnych wymusiło konieczność znormalizowania (standaryzacji) ich budowy, języków programowania, zasilania, łączenia w sieć itp.

Norma IEC w rozdziale pierwszym 61131-1 określa sterownik programowalny jako: „*cyfrowy system elektroniczny do stosowania w środowisku przemysłowym, który posługuje się pamięcią programowalną do przechowywania zorientowanych na użytkownika instrukcji w celu sterowania przez cyfrowe lub analogowe wejścia i wyjścia szeroką gamą maszyn i procesów*" (Kacprzak, 2011).

Przedmiotem normy jest sprecyzowanie definicji i głównych właściwości ważnych przy wyborze i stosowaniu sterowników programowalnych oraz związanych z nimi urządzeń peryferyjnych, jak np. narzędzia programujące i ułatwiające rozruch, wyposażenie testujące, interfejs człowiek-maszyna. Ponadto określenie wymagań dotyczących własności konstrukcyjnych i funkcjonalnych dla sterowników programowalnych, ich warunków serwisowania, bezpieczeństwa i testów. W normie zdefiniowano syntaktykę i semantykę języków programowania, pogrupowanych jako: tekstowe i graficzne. Znajdują się tam również informacje ogólne i wskazówki dla użytkownika wraz z zasadami komunikacji między sterownikami, a innymi systemami elektronicznymi (Miłosiewicz, 1997).

W 1993 roku Międzynarodowa Komisja Elektrotechniki *(ang. International Electrotechnical Commission – IEC)* wprowadziła normę IEC 1131 zatytułowaną „Programmable Controllers". Norma ta składała się z pięciu części, które w późniejszym czasie rozszerzono do ośmiu. Pięć lat później norma zmieniła swoje oznaczenie na IEC 61131 oraz poszerzono jej zawartość. Obecnie norma IEC 61131 stanowi kontynuację poprzedniej i wykorzystuje szereg innych standardów. Odwołuje się ona do innych norm (IEC 50, IEC 559, IEC 617-12, IEC 617-13, IEC 848, ISO/AFNOR, ISO/IEC 646, ISO 8601, ISO 7185, ISO 7498). W Europie została zatwierdzona jako norma EN 61131, natomiast w Polsce została przeredagowana i wdrożono jej pięć rozdziałów jako PN-EN 61131. Poniżej przedstawiono ogólna charakterystykę poszczególnych części omawianej normy.

Część 1. Postanowienia ogólne *(ang. General Information)*

Część ta zawiera ogólne definicje fundamentalnych pojęć związanych ze sterownikami, opisuje typowe własności funkcjonalne, które odróżniają sterowniki programowalne PLC od innych systemów. Obejmuje ona standardowe własności sterowników PLC, jak np. cykliczne przetwarzanie programu aplikacyjnego, korzystającego z przechowywanego w pamięci obrazu stanu wejść i wyjść sterownika lub przydział czasu pracy na komunikację z programatorem czy urządzeniami interfejsu operatora MMI (PN-EN 61131-1:2004E).

Część 2. Wymagania i badania dotyczące sprzętu *(ang. Equipment Requirements and Tests)*

Znajdują się w niej dane dotyczące sprzętu, zdefiniowano wymagania funkcjonalne, mechaniczne oraz elektryczne dotyczące sterowników i urządzeń peryferyjnych w zakresie ich użytkowania, konserwacji oraz przechowywania. Sklasyfikowano sterowniki i narzędzia służące do ich programowania. Ponadto rozdział zawiera metodykę testów, jakim powinny być poddawane produkowane sterowniki, a także charakterystykę środowiska pracy (temperatura, wilgotność powietrza itp.). Można znaleźć w niej informacje o rodzajach obudowy, odporności na zakłócenia, izolacji przewodów i samego urządzenia, sposobach zasilania i montażu (PN-EN 61131-2:2008E).

Część 3. Języki programowania *(ang. Programing Languages)*

W tym rozdziale norma ujednolica stosowane dotychczas języki programowania w zharmonizowany i zorientowany przyszłościowo system. Pojęcia podstawowe, zasady ogólne, model programowy i model komunikacyjny zostały opisane za pomocą formalnych definicji. Ponadto ta część normy specyfikuje syntaktykę, semantykę tekstowych i graficznych języków programowania oraz elementy konfiguracji wspomagające instalację oprogramowania w sterownikach. Ze względu na duże znaczenie ten rozdział został szerzej przedstawiony w dalszej części niniejszej książki (PN-EN 61131-3:2004E).

Część 4. Wytyczne dla użytkownika *(ang. User Guidelines)*

Stanowi ona pewnego rodzaju przewodnik, który przedstawia cały cykl eksploatacji, zaczynając od analizy systemu, przechodząc przez wybór odpowiedniego sterownika, kończąc na jego zastosowaniu. Wspomaga użytkownika we wszystkich fazach projektowania systemu automatyki. Znaleźć w niej można praktyczne informacje, poczynając od analizy systemu i wyboru sprzętu, aż po zastosowania (IEC/TR 61131-4:2004).

Część 5. Wymiana informacji *(ang. Messaging Service Specifications)*

Ta część normy dotyczy zasad komunikacji między poszczególnymi sterownikami różnych typów oraz z innymi urządzeniami. W połączeniu z normą ISO 9506 (MMS) specyfikującą zasady komunikacji w procesie produkcji określa ona funkcje adresowania urządzeń, wymiany danych, przetwarzania alarmów, sterowanie dostępem i administrowanie siecią (PN-EN 61131-5:2002E).

Część 6. Komunikacja przez tzw. sieci polowe *(ang. Communications via fieldbus)*

W zakresie tej części niniejsza norma odwołuje się do IEC 61158 dotyczącej sieci lokalnych (polowych), pracujących w standardzie systemu czasu rzeczywistego, realizujących sterowanie rozproszone, tj.: Foundation Fieldbus H1, ControlNet, PROFIBUS, P-Net, FOUNDATION HSE (High Speed Ethernet), SWIFTNet (protokół opracowany dla Boeinga), WorldFIP, Interbus, Ethernet, Modbus, LonWorks. Znajdują się tam informacje na temat sieci oraz protokołów komunikacyjnych (IEC 61131-6:2012).

Część 7. Programowanie sterowania PLC z zastosowaniem zbiorów rozmytych *(ang. Fuzzy Control Programmings)*

Norma definiuje nowy język tzw. sterowania rozmytego FCL *(ang. Fuzzy Control Language)* umożliwiający aplikację teorii zbiorów rozmytych do programowania PLC. Analizując ofertę rynkową produktów z rodziny PLC trudno znaleźć sterowniki oferujące takie rozwiązania, do nielicznych należą sterowniki Eaton-Moeller serii XC (PN-EN 61131-7:2004E).

Część 8. Wytyczne dotyczące stosowania i wdrażania języków programowania *(ang. Guidelines for the Application and Implementation of Programming Languages)*

Rozdział ten odwołuje się do informacji zawartych w części 3 normy, przedstawia wymagania dotyczące sprzętu i oprogramowania koniecznego do rozwijania, konserwacji oraz modyfikacji programów użytkownika. Wprowadzenie tych wytycznych skutkuje podobieństwem pomiędzy programami oferowanymi przez różnych producentów, przeznaczonych do programowania sterowników PLC. Dzięki takiemu rozwiązaniu inżynier nabywający umiejętności programowania PLC jednego producenta podoła zadaniu zaprogramowania PLC innego producenta – elastyczność programistów (IEC/TR 61131-8:2003).

3. BUDOWA I FUNKCJONOWANIE STEROWNIKÓW PLC

Sterowniki programowalne są komputerami, które przechowują informacje w postaci dwóch stanów logicznych: 1 lub 0, nazywanych cyframi binarnymi (bitami). Cyfry binarne są używane indywidualnie lub wykorzystywane do przedstawiania wartości numerycznych (liczbowych), stąd pierwsze sterowniki wzorując się na przekaźnikach mogły rozróżnić tylko dwa sygnały: włączone lub wyłączone. Zastosowanie przetworników analogowo-cyfrowych (A/D) oraz cyfrowo-analogowych (D/A) umożliwia również sterowanie ciągłe ze zmianą wartości sygnału. Z powyższego wynika, że do najważniejszych elementów sterownika należą mikroprocesor i przetworniki A/D oraz D/A.

Omawianie zagadnień związanych z budową PLC należy rozpocząć od wyodrębnienia podstawowych dwóch ich typów wynikających z architektury. W dalszej części rozdziału zamieszczono szczegółowy opis przedstawicieli poszczególnych typów:

– kompaktowe (rys. 4) – są to zwykle małe sterowniki, stanowiące zwartą zabudowę o małych gabarytach, zawierające niezbędne elementy do ich funkcjonowania; posiadają ograniczoną możliwość modernizacji, moduły są łączone i występują pod wspólną obudową. W jednej obudowie umieszczone są: jednostka CPU, zasilacz oraz kilka gniazd wejść i wyjść, najczęściej cyfrowych. W sterowniki kompaktowe często wbudowane są wejścia licznika, wyświetlacz LCD i prosta klawiatura. Sterowniki tego typu ze względu na małą liczbę wejść i wyjść używane są do sterowania jedną maszyną bądź prostym procesem);
– modułowe (rys. 5) – zabudowa składająca się z elementów łączonych na szynie DIN lub podstawce stanowiącej magistralę zasilającą i przesyłającą sygnały pomiędzy modułami; mogą być rozproszone na powierzchni kilku metrów i skomunikowane za pomocą odpowiedniej magistrali przemysłowej; duże możliwości rozbudowy – skalowalność systemu sterowania, poszczególne podzespoły umieszczane są w osobnych modułach, dzięki czemu można lepiej dopasować sterownik do procesu, którym ma sterować. Do tej grupy należą głównie sterowniki średnie i duże).

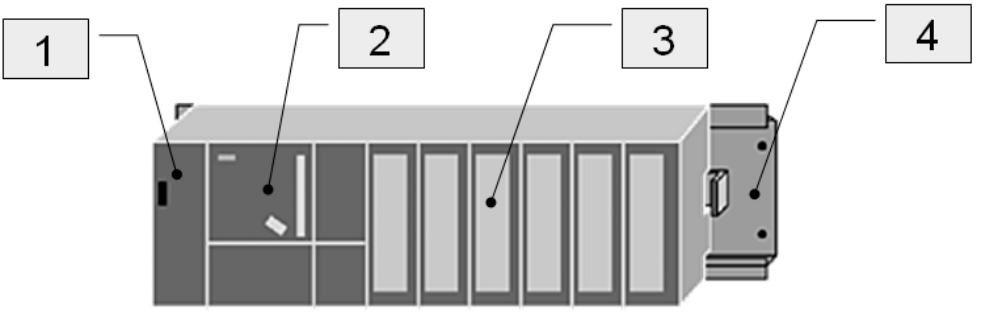
Ponadto w przypadku sterowników modułowych wyróżnia się następujące elementy budowy (rys. 6):

– płyta łączeniowa – kaseta *(ang. Rack)*, do której poprzez gniazda *(ang. Slots)* podłącza się moduły podstawowe i rozszerzające;
– moduły podstawowe, tj.: moduł jednostki centralnej CPU *(ang. Central Proccessing Unit)* oraz zasilacz PS *(ang. Power Supply)*;
– moduły rozszerzające możliwości PLC, umożliwiające jego konfigurację wg zapotrzebowania, tj.:
  - moduły wejść i wyjść cyfrowych *(ang. Digital Input, Digital Output)*;
  - moduły wejść i wyjść analogowych *(ang. Analog Input, Analog Output)*;
  - moduły szybkich liczników *(HSC, ang. High-Speed Counter)*;
  - moduły pozycjonowania osi *(APM, ang. Axis Positioning Module)*;

- moduły do sterowania napędami;
- moduły regulatorów, tj. PID;
- moduły rozszerzające pamięć oraz moduł pamięci Flash;
- moduły komunikacyjne, realizujące połączenie sterownika z siecią lokalną w określonym standardzie, np. Modbus, Profibus, ControlNet, Genius itp. lub do sieci Ethernet;
- moduły komunikacji bezprzewodowej w standardach: WIFI, GPRS;
- moduły wejściowe z przetwornikiem do standaryzowanych czujników, np. temperatury PT100.

Rysunek 4. Sterownik kompaktowy
Źródło: Moeller, 2008

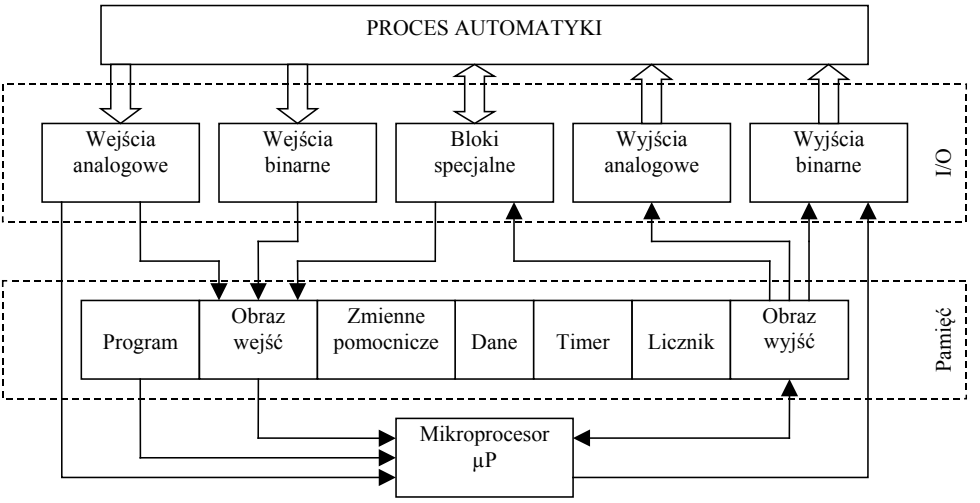
Rysunek 5. Sterownik modułowy
Źródło: Moeller, 2008



Rysunek 6. Sterownik modułowy: (1) – zasilacz, (2) – jednostka centralna zawierająca mikroprocesor, (3) – moduły, (4) – płyta łączeniowa

Na rysunku 7 przedstawiono ogólny schemat blokowy sterownika PLC. Zasadniczą częścią sterownika jest jednostka centralna **CPU**; jest to układ elektroniczny, w którym program jest ładowany, przechowywany i wykonywany – inna nazwa to: Główna Jednostka Przetwarzania *(MPU – ang. Main Processing Unit)*. Najczęściej projektowana jako układ wieloprocesorowy. Liczba oraz typ mikroprocesorów pracujących w jednostce cen-

tralnej ma wpływ przede wszystkim na szybkość działania sterownika, liczbę obsługiwanych obwodów wejściowo-wyjściowych, jak również pojemność pamięci. Do CPU zalicza się również pamięć oraz magistrale: adresowe, danych, sterowań i systemową we/wy.



Rysunek 7. Schemat blokowy sterownika PLC

Parametry jednostki centralnej można scharakteryzować poprzez: wielkość pamięci, czas typowego cyklu programowego, napięcie zasilania, możliwość pracy w systemie *Master-Slave*, rodzaj oprogramowania, liczbę dostępnych procedur i bloków funkcyjnych w oprogramowaniu, możliwość pracy w sieci i zaimplementowane protokoły komunikacyjne, liczbę dozwolonych modułów rozszerzających, możliwość rozszerzania pamięci, możliwość wykonywania obliczeń zmiennoprzecinkowych.

Pamięć w sterowniku jest potrzebna do zapamiętywania programu, który ma być wykonywany oraz do przechowywania informacji pośrednich, powstających w trakcie wykonywania programu. Do przechowywania programu służą pamięci RAM, EPROM i EEPROM. Pamięć ROM – przechowuje system operacyjny sterownika obsługujący zarządzanie zadaniami sterownika i komunikację z urządzeniami zewnętrznymi, Do przechowywania informacji pośrednich mogą być używane tylko pamięci RAM. Przechowuje ona programy i dane użytkownika, dane systemowe oraz obrazy wejść/wyjść. Pamięć RAM posiada podtrzymujące awaryjne źródło zasilania. Pamięć zasadniczo dzieli się na: operacyjną i programu. W większości sterowników pamięć RAM ma pojemność większą niż potrzebna pamięć operacyjna. Nadwyżkę wykorzystuje się do zapisywania programów. Większość PLC posiada możliwość rozszerzania pamięci za pomocą kości bądź płytki z zamontowanymi układami pamięci RAM.

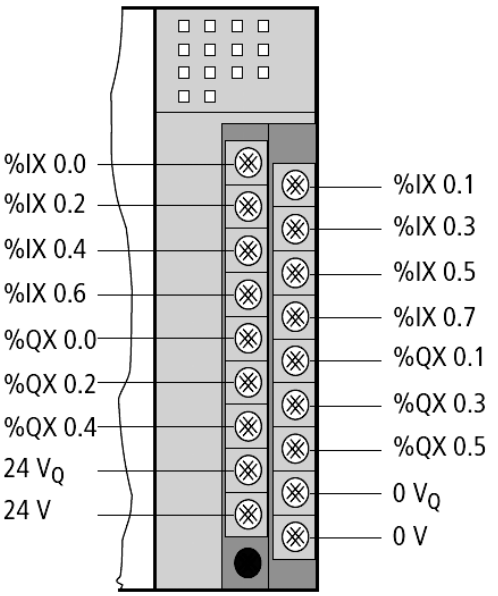
Programy użytkownika mogą być przechowywane również w pamięci zewnętrznej (E)EPROM. Dane użytkownika dzielą się na trzy grupy:

– dane wejściowe obrazujące stan wejść, adresowane prefiksem %I;
– dane wyjściowe obrazujące stan wyjść, adresowane prefiksem %Q;
– dane użytkownika, pozostałe dane niezwiązane z obrazami danych wejściowych i wyjściowych, adresowane prefiksem %M.

Adresy danych zalecane w normie nie zostały przyjęte przez wszystkich producentów sterowników. Na przykład w sterownikach z typu Modicon firmy Scheinder Electric problem adresów został rozwiązany przez dodanie kolejnej cyfry do adresu. Cyfra rozpoczynająca adresy cyfrowe to 1 dla wejść i 0 dla wyjść, natomiast dla adresów analogowych 3 dla wejść i 4 dla wyjść. Firma Allen-Bradley do adresowania danych zastosowała przedrostki literowe – I oznacza wejścia, O – wyjścia oraz numer gniazda, z którego pochodzi obraz sygnału. W sterownikach firmy Mitsubishi Electric wejścia oznaczono literą X, wyjścia – Y. Kolejnym podzespołem sterownika są moduły wejściowe, ich zadaniem jest doprowadzenie do sterownika danych z czujników, zadajników i urządzeń pomiarowych systemu. Zbiór tych sygnałów tworzy obraz wejść.

Ostatnim z integralnych modułów sterownika są moduły wyjściowe, które odpowiadają za przekazanie wyliczonych sygnałów wyjściowych do elementów i urządzeń wykonawczych.

Układy wejść-wyjść analogowych i cyfrowych stanowią połączenia sterownika ze sterowanym obiektem. Sygnały cyfrowe dwustanowe przed wprowadzeniem do jednostki centralnej muszą zostać uporządkowane, tzn. każdemu sygnałowi przypisywany jest numer zwany adresem (rys. 8).



Rysunek 8. Wejścia i wyjścia cyfrowe na terminalu przyłączeniowym

Wyróżnia się następujące typu modułów I/O:

– wejścia dyskretne, nazywane również wejściami cyfrowymi *(ang. Digital Inputs)* zamieniają pochodzące z urządzeń (przyciski, przełączniki, wyłączniki krańcowe etc.) sygnały prądu stałego lub przemiennego na sygnały logiczne (dwustanowe) akceptowane przez sterownik. W produkowanych obecnie sterownikach do takiej zamiany wykorzystywany jest zazwyczaj przetwornik optyczny, zapewniający dodatkowo optoizolację pomiędzy obwodami wejściowymi a magistralą sterownika. Obwody te zasilane są najczęściej prądem stałym o napięciu 24 VDC;
– wyjścia dyskretne, nazywane również wyjściami cyfrowymi *(ang. Digital Outputs)* zamieniają sygnały binarne sterownika na sygnały prądu stałego lub przemiennego potrzebne do wysterowania urządzeń wyjściowych (cewki styczników, lampki kontrolne etc.). Zamiany tych sygnałów dokonuje się poprzez zamykanie lub otwieranie zasilanych z zewnętrznego źródła obwodów wyjściowych za pomocą przekaźników (wyjścia przekaźnikowe, *ang. Relay Output*) lub łączników tranzystorowych (wyjście „napięciowe").

Układy wejść/wyjść analogowych ze względu na swoją bardziej złożoną budowę (konieczność przetwarzania sygnału analogowego na cyfrowy i odwrotnie) są opcjonalnym wyposażeniem sterowników. Moduły wejść i wyjść analogowych pracują na sygnałach ciągłych reprezentowanych przez zmieniające się napięcie lub natężenie prądu elektrycznego w określonym zakresie:

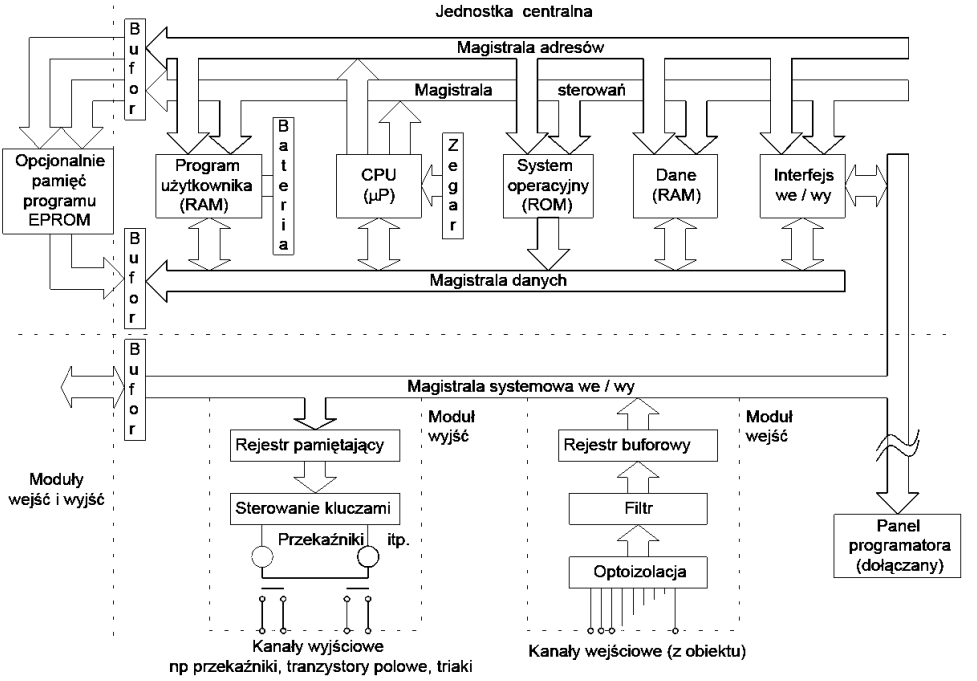
– wejścia analogowe, *(ang. Analog Input)* zamieniają pochodzące z czujników sygnały analogowe (ciągłe) na sygnały cyfrowe. Konwersja tych sygnałów realizowana jest za pomocą przetworników analogowo-cyfrowych A/D *(ang. Analog to Digital Converter)*. Zwykle jest to przetwornik 12-bitowy o rozdzielczości 4096 przedziałów, co oznacza, że np. dla zakresu napięcia pomiarowego 0 – 10VDC przy wartości napięcia 0 otrzymujemy cyfrową wartość 0 – natomiast dla wartości napięcia 10V otrzymujemy cyfrową wartość 4095. W przypadku sygnału prądowego najpopularniejszym jest zakres od 0 lub 4 mA do 20 mA. Sygnał prądowy jest bardziej odporny na zewnętrzne zakłócenia. Parametrami charakterystycznymi dla wejść analogowych są: rozdzielczość wynikająca z zastosowanego przetwornika A/D, dokładność, liniowość;
– wyjścia analogowe, *(ang. Analog Output)* zamieniają sygnały cyfrowe na sygnały ciągłe, sterujące urządzeniami wykonawczymi. Konwersja tych sygnałów realizowana jest za pomocą przetworników cyfrowo-analogowych D/C *(ang. Digital to Analog Converter)*. Sposób kodowania sygnału oraz zakresy są podobne jak w przypadku wejść analogowych.

Dla odzwierciedlenia działania sterownika PLC na rysunku 9 przedstawiono schemat z przepływem sygnałów pomiędzy blokami funkcjonalnymi. Natomiast na rysunku 10 przedstawiono poszczególne funkcje sterownika.

Analizę schematu należy rozpocząć od maszyny bądź procesu, które dostarczają informacji i zarazem są sterowane. Odbieranie i przesyłanie informacji do odbiorników zewnętrznych realizują *funkcje interfejsu z czujnikami i elementami wykonawczymi*. Podłączane do sterownika czujniki i elementy wykonawcze muszą spełniać standard dla danego typu sterownika, w innym przypadku nie będzie zachodzić komunikacja lub może dojść do uszkodzenia urządzenia, dlatego szczególnie ważne jest, aby przed montażem dobrze prześledzić schemat podłączania i warunki montażu tych odbiorników.

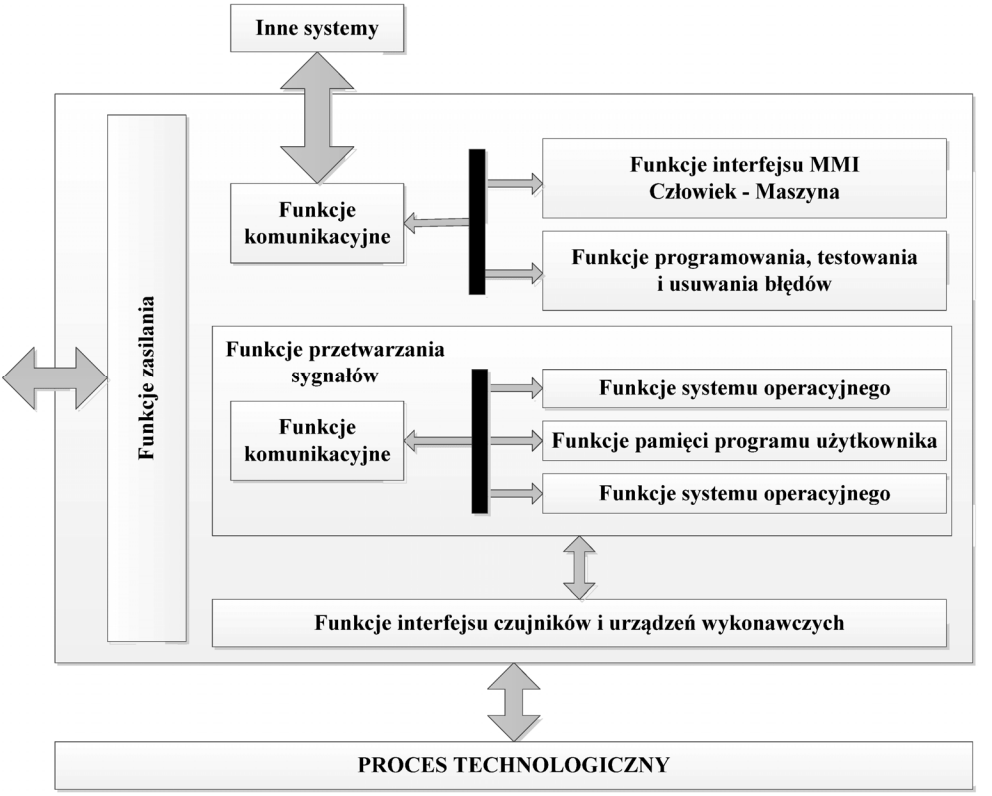
Funkcje przetwarzania sygnałów realizowane są przez system operacyjny przy udziale programu użytkownika. Obejmuje on zarówno przetwarzanie sygnałów pochodzących

z czujników, jak również zapisanych w pamięci danych. *Funkcje pamięci danych użytkownika* służą do zapewnienia miejsca w pamięci dla danych pochodzących z wejść/wyjść oraz danych wewnętrznych. *Funkcje pamięci programu użytkownika* odpowiedzialne są za przechowywanie algorytmu programu, który stanowi sekwencję rozkazów ustaloną w logicznej kolejności, a więc są odpowiedzialne za sposób działania sterowanego procesu lub maszyny. Zarządzaniem wewnętrznymi funkcjami PLC (konfiguracją, diagnostyką, pamięcią, komunikacją z urządzeniami peryferyjnymi) zajmują sie *funkcje systemu operacyjnego*. *Funkcje komunikacyjne* zajmują się wymianą danych pomiędzy urządzeniami zewnętrznymi a jednostką centralną sterownika. Do takich urządzeń można zaliczyć inne PLC, serwery danych, komputery z systemami wizualizacji typu SCADA. Najczęściej komunikacja odbywa się poprzez transmisję szeregową. *Funkcje programowania, testowania i usuwania błędów* realizują szeroko pojętą diagnostykę. Odpowiadają m.in. za poprawność językową składni programu, testowanie czujników i elementów wykonawczych, ustawianie lub zerowanie stałych (liczniki, timery). *Funkcje interfejsu CZŁOWIEK – MASZYNA (MMI – ang. Man-Maschine Interface),* jak sama nazwa wskazuje, ułatwiają komunikację operatorowi z PLC przez możliwość oddziaływania na system, dostarczają operatorowi informacji niezbędnych do monitorowania działania sterowanej maszyny lub procesu.



Rysunek 9. Architektura sterownika PLC

Źródło: Kasprzyk, 2011



Rysunek 10. Schemat funkcjonalny PLC

Coraz częściej te funkcje realizowane są za pośrednictwem graficznych paneli dotykowych. Sterownik posiada odpowiednie łącze zapewniające komunikację z programatorem, z reguły jest to port RS-232 lub RS-488, ale opcjonalnie można zamówić możliwość komunikacji przez Ethernet lub USB. Złącze to umożliwia programowanie i testowanie sterownika. Programatorem jest zazwyczaj komputer PC z odpowiednim oprogramowaniem lub specjalnie dedykowane urządzenie – programator klawiaturowy. Oprócz łącza do programowania sterownik może posiadać inne łącza komunikacyjne, pozwalające na komunikację z urządzeniami MMI (wyświetlacze ciekłokrystaliczne, panele operatorskie). *Funkcje zasilania* realizowane są przez zasilacz, którego zadaniem jest wytworzenie napięć niezbędnych do działania sytemu PLC. Stosowane są różne typy zasilaczy, najczęściej jednak obwody wejść i wyjść zasilane są prądem stałym 24 V, natomiast sterownik może być zasilany prądem przemiennym o napięciu 230 V.

3.1. Klasyfikacja sterowników PLC

Oprócz kryterium podziału sterowników na kompaktowe oraz modułowe przyjęto inne podziały wynikające z cech charakterystycznych tych urządzeń.

Kryterium podziału sterowników jest ich wielkość określana liczbą modułów wejść/wyjść oraz liczbą przekaźników. Według tej cechy sterowniki dzieli się na:

- małe o liczbie wejść/wyjść do 256 i 50-150 przekaźników;
- średnie o liczbie wejść/wyjść między 256 a 1024 i 150-500 przekaźników;
- duże o liczbie wejść/wyjść powyżej 1024 i 500-3000 przekaźników.

Moc sterowników programowalnych wyznaczają:

- wielkość pamięci programu;
- czas realizacji 1 kilorozkazu;
- funkcjonalność:
  - liczba wejść/wyjść binarnych i analogowych;
  - komunikacja z otoczeniem cyfrowym.

Małe sterowniki

Obszary zastosowań:

- małe maszyny i urządzenia;
- technika instalacyjna;
- technika domowa;
- pozycjonowanie osi;
- regulacyjne funkcje programowe i sprzętowe.

Cechy:

- komunikacja i wizualizacja;
- zwarta konstrukcja;
- mała moc obliczeniowa;
- możliwość decentralizacji zadania sterowania;
- programowanie za pomocą programatorów i komputerów PC;
- języki programowania kompatybilne „w górę”;
- ograniczone możliwości komunikacyjne.

Średnie sterowniki

Obszary zastosowań:

- szybkie regulatory PID;
- regulatory temperatury z czujnikiem PT100;
- operacje indeksowe i przeliczniki BCD;
- komunikacja i wizualizacja.

Cechy:

- budowa modułowa, pozwala uzupełniać jednostkę centralną o wiele modułów specjalizowanych;
- procesor standardowy lub specjalizowany, często wzmacniany koprocesorem;
- duża moc obliczeniowa, duża szybkość obliczeń;
- programowanie wyłącznie za pomocą komputerów PC;

– języki programowania typowe dla PLC oraz języki wyższego poziomu, np. C, MODULA;
– znaczne możliwości komunikacyjne, sieci PROFIBUS (złącze RS-485), Ethernet, CAN.

Duże sterowniki

Obszary zastosowań:
– wyższe poziomy sterowania;
– monitorowanie złożonych procesów przemysłowych;
– systemy z redundancją;
– sterowanie nadrzędne wielowymiarowe;
– komunikacja i wizualizacja;
– niemal nieograniczone obszary zastosowań.

Cechy:
– budowa modułowa, pozwala uzupełniać jednostkę centralną o wiele modułów specjalizowanych;
– efektywne procesory np. transputery (obliczenia równoległe), koprocesory;
– duży wybór inteligentnych urządzeń peryferyjnych, duża liczba wejść/wyjść;
– programowanie wyłącznie za pomocą komputerów PC;
– języki programowania typowe dla PLC oraz języki wyższego poziomu, np. C, MODULA;
– znaczne możliwości komunikacyjne, sieci PROFIBUS (złącze RS485), Ethernet, CAN, na wyższych poziomach sterowania stosowany jest protokół MAP30.

3.2. Przykładowe rozwiązania techniczne

Istotne znaczenie eksploatacyjne ma fakt, iż sterowniki programowalne są układami umożliwiającymi pracę w szczególnie trudnych warunkach, tj. podwyższonym poziomie drgań, zakłóceń elektromagnetycznych, szerokim zakresie temperatur itp. Warunki eksploatacji urządzeń technicznych w procesach rolno-spożywczych są zróżnicowane. W zależności od wymagań istnieje możliwość implementacji systemu sterującego opartego o sterowniki kompaktowe, aż do systemów rozproszonych o ogromnych możliwościach skalowania. Przedstawicielem sterownika kompaktowego jest seria *„easy”* firmy EATON-Moeller, która z racji swojej prostoty jest nazywana przekaźnikiem programowalnym. Producent podzielił je na cztery serie: easy 500, easy 700, easy 800, easy MFD-Titan. Najbardziej rozbudowane są sterowniki serii 800 posiadające możliwość podłączenia 96Wej/64Wyj dyskretnych. Ponadto posiadają 4 wejścia analogowe i możliwość dołączenia poprzez rozszerzenie wyjścia analogowego (Moeller, 2012). Wymienione układy różnią się miedzy sobą możliwościami sterowania oraz skalowania systemu. Easy zostały zaprojektowane z myślą o maksymalnym uproszczeniu konstrukcji i programowania urządzenia, dzięki czemu aplikacje są proste w implementacji. Duża różnorodność modeli każdej z ww. serii umożliwia wybór odpowiedniego urządzenia do sterowanego procesu. Sterowniki programowalne easy 800 wzbogacono o człon proporcjonalno-całkująco-różniczkujący (regulator PID), funkcje modulacji szerokości impulsów (PWM) itp.

W zależności od modelu i serii wyposażone są m.in. w:
– wejścia cyfrowe (binarne) umożliwiające sterowanie dwustanowe;
– wyjścia cyfrowe w zależności od modelu: przekaźnikowe lub tranzystorowe;

– wejścia analogowe, umożliwiające sterowanie uzależnione od analogowego sygnału, np. przez komparatory;
– wyjścia analogowe umożliwiające sterowanie ciągłe np. prędkością obrotową;
– zegar czasu rzeczywistego – programowanie procesów uzależnionych od daty i godziny;
– klawiaturę i wyświetlacz ciekłokrystaliczny do programowania bez użycia komputera PC;
– moduł komunikacji sieciowej Master-Slave; dzięki tej opcji możliwe jest implementowanie systemów rozproszonych (sieci przekaźników);
– gniazdo rozszerzeń umożliwiające zainstalowanie dodatkowych modułów we/wy.

Ponadto w programie istnieje możliwość umieszczania do czterech styków i jednej cewki w każdej z 256 linii programowych, do 32 komparatorów, 32 przekaźników czasowych, 32 liczników, 32 komunikatów tekstowych, 32 modułów funkcji arytmetycznych, 32 modułów funkcji logicznych. Sterowniki te pracują przy sygnałach o zmieniających się wartościach z wysoką częstotliwością.

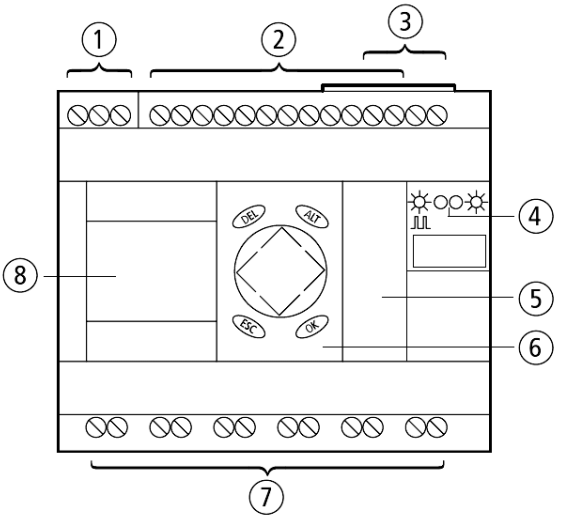
Sterowniki serii 800 są urządzeniami zarówno sterującymi, jak i programującymi. Wykorzystywane są m.in. do sterowania budynkiem w tzw. systemie „inteligentnego domu" oraz maszyn i nieskomplikowanych procesów produkcyjnych. Za pomocą wbudowanej sieci NET istnieje możliwość rozbudowy systemu do 8 sterowników. Każdy ze sterowników może mieć zapisany własny program, dzięki temu można zbudować rozproszony system sterowania. Sterowniki programuje się językiem drabinkowym. Schemat programu można wprowadzić bezpośrednio za pomocą klawiatury i wyświetlacza sterownika lub można opracować program offline za pomocą aplikacji *EasySoftPro,* a potem zainstalować go w sterowniku. Aplikacja ta pozwala również na przetestowanie i wydrukowanie programu w formacie DIN, ANSI lub easy. Przy programowaniu bezpośrednim można wykorzystywać następujące elementy schematu drabinkowego:
– styki zwierne i styki rozwierne łączone szeregowo i równolegle;
– przekaźniki wyjściowe i znaczniki;
– wyjścia jako cewki przekaźnika: zwykłe, bistabilne, z samopodtrzymaniem, reagujące na zbocza narastające i opadające;
– przekaźniki czasowe z funkcjami:
  o opóźnionego zadziałania;
  o losowej zmiany czasu opóźnionego zadziałania;
  o opóźnionego odpadania;
  o losowej zmiany czasu opóźnionego odpadania;
  o opóźnionego zadziałania i odpadania;
  o losowej zmiany czasu opóźnionego zadziałania i odpadania;
  o impulsowymi;
  o migającymi synchronicznie;
  o migającymi asynchronicznie;
– liczniki zliczające do przodu i wstecz;
– liczniki szybkich impulsów;
– liczniki liczące w przód i wstecz z dolną i górną wartością progową z funkcjami:
  o ustawiania wartości (Preset);
  o licznika częstotliwości;
  o szybkości licznika;
  o licznika impulsów z przetwornika obrotowo-impulsowego.

Ponadto za pomocą klawiatury i wyświetlacza sterownika można korzystać z funkcji:

- porównywania wartości;
- wyświetlania tekstów i zmiennych;
- operowania analogowymi wielkościami wejściowymi i wyjściowymi (urządzenia DC);
- korzystania z zegara tygodniowego i rocznego;
- liczenia czasu pracy (licznik godzin pracy);
- komunikowania za pomocą wbudowanej sieci NET;
- przeprowadzania operacji arytmetycznych:
  - dodawania;
  - odejmowania;
  - mnożenia;
  - dzielenia;
- śledzenia stanu linii programu (przepływ prądu);
- wpisywania, zapamiętywania i zabezpieczania programu hasłem (Moeller, 2006).

Przykładowy sterownik serii easy 819-DC-RC posiada dwanaście wejść cyfrowych, z czego cztery mogą być użyte jako wejścia analogowe i kolejne cztery, które mogą być użyte jako „wejścia szybkie" do zastosowań jako liczniki. Sterownik posiada 6 wyjść przekaźnikowych o natężeniu do 8A (dla obciążenia rezystancyjnego, 2A dla obciążenia wyjścia indukcyjnego). Rozmieszczenie wejść i wyjść oraz ogólny wygląd sterownika przedstawiono na rysunku 11. Sterownik jest zasilany poprzez złącze zasilania (1), może to być napięcie 24VDC lub 230VAC. Dwanaście wejść cyfrowych i cztery analogowe (2) pozwalają na podłączenie czujników i elementów stykowych. Poprzez moduły rozszerzeń istnieje możliwość powiększenia ich liczby. Złącze sieci EASY-NET (3) pozwala na łączenie sterowników w sieć lokalną typu Master-Slave, opcjonalnie istnieje możliwość dołączenia modułu zawierającego port sieci Ethernet lub Profibus.

Diody LED (4) sygnalizują pracę sterownika oraz sieci EASY-NET. Złączem (5) możliwe jest podłączenie sterownika do komputera PC lub karty pamięci umożliwia ono programowanie sterownika oraz obserwowanie jego pracy w trybie online. Istnieje możliwość programowania przez klawiaturę (6), ale jest to niewygodne. Sterownik posiada wyświetlacz (8), na którym widoczny jest program oraz stany logiczne poszczególnych wejść i wyjść. Wyjścia sterujące rozmieszczone są na dolnej listwie (8). Sterownik może pracować w zakresie temperatur od -25^{o}C do 55^{o}C.



Rysunek 11. Płyta czołowa sterownika easy800

Innym przykładem popularnych typów sterowników programowalnych są VersaMax, których producentem jest GE – Intelligent Platform. VersaMax-Nano i VersaMax-Micro występują w wersji kompaktowej, natomiast VersaMax jest modułowym sterownikiem. Sterowniki Nano to sterowniki kompaktowe dziesięciopunktowe (posiadają 10 punktów dyskretnych wejść/wyjść), sterowniki Micro mają maksymalnie do 140 punktów wejść/wyjść z zastosowaniem opcjonalnych modułów rozszerzających. Poprzez swoją zwartą budowę, dużą funkcjonalność oraz łatwość obsługi znajdują szerokie zastosowanie, np. jako elementy składowe maszyn oraz rozproszonych systemów sterowania (rys. 12 i 13).

Rysunek 12. Sterownik VersaMax-Nano

Źródło: Peszyński i Siemieniako, 2002

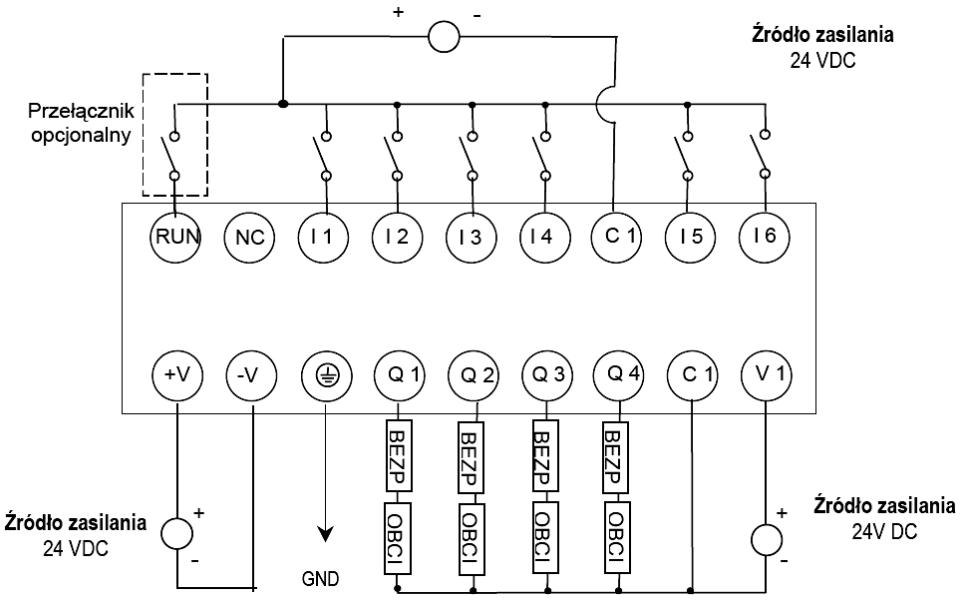
Rysunek 13. Sterownik VersaMax-Mikro

Źródło: Peszyński i Siemieniako, 2002

Sterowniki VersaMax Nano znajdują zastosowanie dla prostych aplikacji (np. mniejsze maszyny pakujące i rozpakowujące) z racji ograniczonej liczby punktów komunikacji (6 wejść i 4 wyjścia w wersji cyfrowej). Pomimo małych rozmiarów są wyposażone w takie funkcje, jak: licznik impulsów wysokiej częstotliwości, obsługa bloków funkcyjnych zmiennoprzecinkowych i podprogramów oraz możliwość przyporządkowania haseł i poziomów dostępu do funkcji sterujących, programowanie pętli regulatora PID. Programowane są narzędziem Proficy Machine Edition w języku drabinkowym LD lub listą instrukcji IL. Oprogramowanie to wykorzystywane jest we wszystkich sterownikach GE – Intelligent Platforms, stąd programując małe sterowniki nabywa się umiejętności wykorzystywane przy większych.

Przykładowy sterownik VersaMax-Nano oznaczony IC200NDD101 posiada sześć wejść z zasilaniem 24 VCD i cztery wyjścia tranzystorowe z nominalnym napięciem zasilającym 24 VCD. Wejścia mogą być standardowe, zaś wyjścia pracują w logice dodatniej lub jako liczniki impulsów wysokiej częstotliwości. Doprowadzenie sygnału prądu do wejścia powoduje zapisanie wartości logicznej 1 w tabeli stanu wejść. Podczas pracy w trybie standardowych wejść ich charakterystyki pozwalają na podłączenie szeregu po-

wszechnie stosowanych urządzeń wejściowych takich jak: wyłączniki przyciskowe, wyłączniki krańcowe, elektroniczne wyłączniki zbliżeniowe. Wyjścia mogą być używane do przełączania urządzeń takich jak: zawory, źródła światła, styczniki. Ponadto mogą zostać skonfigurowane jako wyjścia sterowane przez liczniki impulsów wysokiej częstotliwości. W tym wypadku mogą być one również używane jako wyjścia PWM. Wszystkie wyjścia są izolowane pomiędzy obwodami wejściowymi a obwodami logicznymi i przełączane za pomocą napięcia dodatniego. Wyjścia posiadają jedno wspólne uziemienie oraz jedno wspólne zasilanie. Wyjścia te mogą pracować przy wysokich natężeniach prądów rozruchowych i są zabezpieczone przed ujemnymi impulsami napięciowymi, co daje możliwość przełączania źródeł światła i obciążeń indukcyjnych. Sterownik zawiera dwie listwy zaciskowe, może być używany z zewnętrznym przełącznikiem trybu pracy Run/Stop. Przełącznik ten można skonfigurować jako przełącznik trybów pracy, przełącznik do blokowania pamięci oraz może zostać wykorzystany do zerowania tabeli błędów przy wystąpieniu błędu krytycznego (rys. 14).



Rysunek 14. Schematy podłączenia urządzeń do wejść i wyjść sterownika Nano

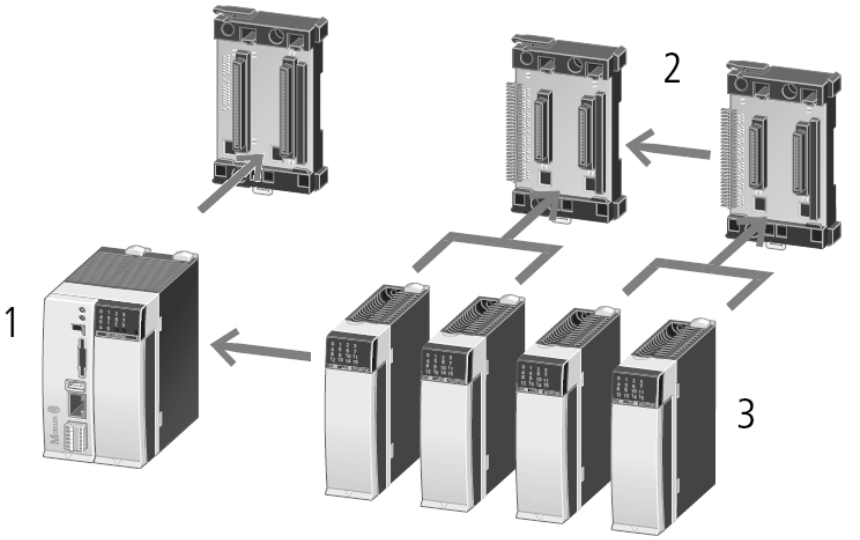
Sterownik jest wyposażony w kondensator podtrzymujący (przez krótki okres czasu) zawartość pamięci RAM. Zestaw instrukcji do programowania zawiera pełny zakres funkcji z matematyką liczb zmiennoprzecinkowych oraz 4 kilobajty słowa pamięci programu i 256 słów rejestrów bitowych. Tabela 1 przedstawia parametry techniczne sterownika IC200NDD101 VersaMax Nano.

Tabela 1. Parametry techniczne sterownika IC200NDD101 VersaMax Nano

Waga	150 gramów
Wymiary modułu	Wysokość: 80 mm Głębokość: 47 mm Szerokość 75 mm
Typowy czas trwania cyklu	1.3 ms dla 1kB operacji logicznych
Wejścia	6 obwodów wejściowych działających w logice dodatniej/ujemnej 24 VDC
Wyjścia	4 wyjścia tranzystorowe
Wyjścia zasilania	+5VCD na styku 7 portu szeregowego, maksymalnie 100 mA
Dokładność wskazań zegara czasu rzeczywistego (dla funkcji czasomierza)	+/- 0.5%

Źródło: Kwaśniewski, 2008

Jako przykładowy – modułowy sterownik PLC zostanie przedstawiona jednostka Eaton-Moeller XC-CPU101 (rys. 16). Przeznaczona jest ona do małych i średnich aplikacji. Istnieje możliwość rozszerzenia jej architektury 15 modułami XIOC (Moeller, 2012). Sposób podłączania modułów zamieszczono na rysunku 15. Jednostka podstawowa zawierająca CPU (1) montowana jest do podstawki bazowej, posiadającej slot umożliwiający podłączenie kolejnej podstawki z jej prawej strony (2). Podstawka posiada zdolność przyłączenia do jednostki centralnej dwóch lub trzech modułów (3) w zależności od jej wersji.



Rysunek 15. Montaż dodatkowych modułów do jednostki podstawowej

Panel czołowy tego sterownika (rys. 16) zawiera diodę sygnalizującą stan pracy sterownika (1), światło ciągłe oznacza tryb Run. Druga dioda koloru czerwonego (2) zapala się w sytuacji, kiedy wystąpi błąd w sterowniku.

Rysunek 16. Moduł podstawowy (CPU) sterownika PLC XC-CPU 101

Przełącznikiem (3) można zmienić tryb pracy sterownika Run/Stop, poniżej umieszczony jest slot kart pamięci typu MMC. Gniazdo (5) umożliwia podłączenie w standardzie RS 232 komputera PC, będącego programatorem lub systemem nadrzędnym dla PLC (skomunikowanym protokołem DDE). Na potrzeby komunikacji sterownik ten jest wyposażony również w port typu CAN (6). Diody oznaczone od 0 do 7 rozmieszczone w dwóch pierwszych rzędach sygnalizują stany wejść sterownika (7), natomiast od 0 do 5 w dwóch kolejnych rzędach sygnalizują stany wyjść sterownika (8). Pokrywa (9) zasłania terminale zaciskowe do podłączania wejść, wyjść oraz zasilania sterownika.

W tabeli 2 umieszczono informację o cechach charakterystycznych różnych wersji sterownika z jednostką centralną XC100. Sterownik może mieć podłączone moduły rozszerzeń zawierające wejścia i wyjścia cyfrowe, wejścia i wyjścia analogowe, moduły komunikacyjne, moduły technologiczne z szybkimi licznikami HSC, moduły do pomiaru temperatury PT100/1000. Rozdzielczość cyfrowa przetworników A/D i D/A wynosi 12 bit.

Tabela 2. Podstawowe parametry sterowników modułowych Moeller XC-CPU 101

Wejścia/Wyjścia	8 wejść cyfrowych, 6 wyjść cyfrowych
Karta pamięci	MMC
Możliwości rozbudowy	do 15 modułów XIOC
Zintegrowana magistrala sieciowa	CANopen (500 kBaud)
Kolejne złącza	RS-232
XC-CPU101-C64K-8DI-6DO	
Pamięć programu	64 kB
Pamięć danych	64 kB
XC-CPU101-C128K-8DI-6DO	
Pamięć programu	128 kB
Pamięć danych	128 kB
XC-CPU101-C256K-8DI-6DO	
Pamięć programu	256 kB
Pamięć danych	256 kB

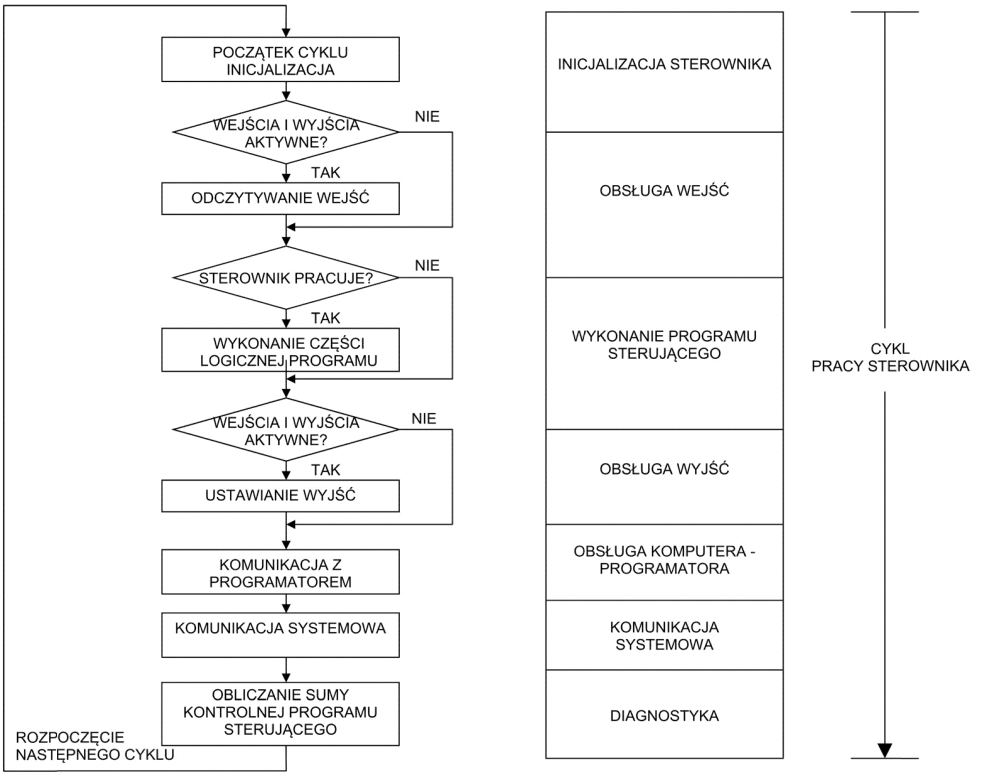
Źródło: Moeller, 2012

3.3. Algorytm działania sterownika PLC

Praca sterownika przebiega wg algorytmu zamieszczonego na rysunku 17. W zależności od stopnia zaawansowania sprzętu algorytmy mogą ulegać rozbudowie, powiększać się o kolejne operacje. Każdy sterownik PLC odznacza się cykliczną pracą, gdzie program (algorytm) wprowadzony przez użytkownika jest wykonywany krokowo od pierwszej do ostatniej instrukcji. Polega ona na obliczeniu i ustawianiu stanów sygnałów wyjściowych na podstawie odczytanych stanów sygnałów wejściowych (odczytanych przed rozpoczęciem wykonywania tego programu). Zmiany sygnałów wejściowych, które nastąpiły po rozpoczęciu cyklu, będą uwzględnione dopiero w cyklu następnym. Odstępstwem od tej reguły jest mechanizm przerwań. Cechą charakterystyczną sterowników PLC jest cykliczny obieg pamięci programu. Algorytm jest zapisywany w dedykowanym sterownikowi języku programowania. Istnieje możliwość zmiany algorytmu przez zmianę zawartości pamięci programu. Sterownik wyposaża się w odpowiednią liczbę układów wejściowych, zbierających informacje o stanie obiektu i żądaniach obsługi, oraz odpowiednią liczbę i rodzaj układów wyjściowych, połączonych z elementami wykonawczymi, sygnalizacyjnymi lub transmisji danych. Jeden cykl pracy składa się z następujących po sobie elementów:

– inicjalizacja sterownika, w tej fazie następuje m.in. obliczanie czasu cyklu, planowanie startu do następnego cyklu i określanie jego trybu. Czas trwania tej fazy jest stały i zależy ściśle od typu CPU;
– obsługa wejść, rozpoczyna się od przepisania stanów z buforów w modułach wejściowych do odpowiadających im obszarów danych w pamięci CPU (tworzony jest tzw. obraz wejść). Dla modułów wejść cyfrowych jest to obszar zawierający dane binarne, natomiast dla wejść analogowych dane zapisywane są w postaci 16-bitowych słów. Czas trwania tej fazy zależy od ilości wejść ustawionych w konfiguracji sterownika;

- wykonanie programu sterującego, realizowany jest pojedynczy przebieg całego programu od pierwszej instrukcji do instrukcji kończącej END. Czas wykonywania tego etapu zależy od: możliwości obliczeniowych CPU i liczby zastosowanych instrukcji;
- obsługa wyjść, zapisywane są dane w buforach modułów wyjściowych (tworzony jest tzw. obraz wyjść). W zależności od modułów: dla cyfrowych dane binarne, dla analogowych dane w postaci 16-bitowych słów;
- obsługa komputera-programatora, jeżeli komputer jest w trybie online następuje wymiana danych;
- komunikacja systemowa, obejmuje usługę wymiany danych pomiędzy modułami wykonującymi samodzielne obliczenia (pozycjonowanie osi, regulatory, przetworniki PT100/1000) oraz modułami komunikującymi się z innymi systemami np. SCADA;
- diagnostyka, realizowane jest sprawdzanie konfiguracji oraz poprawności kolejnej pętli programu.



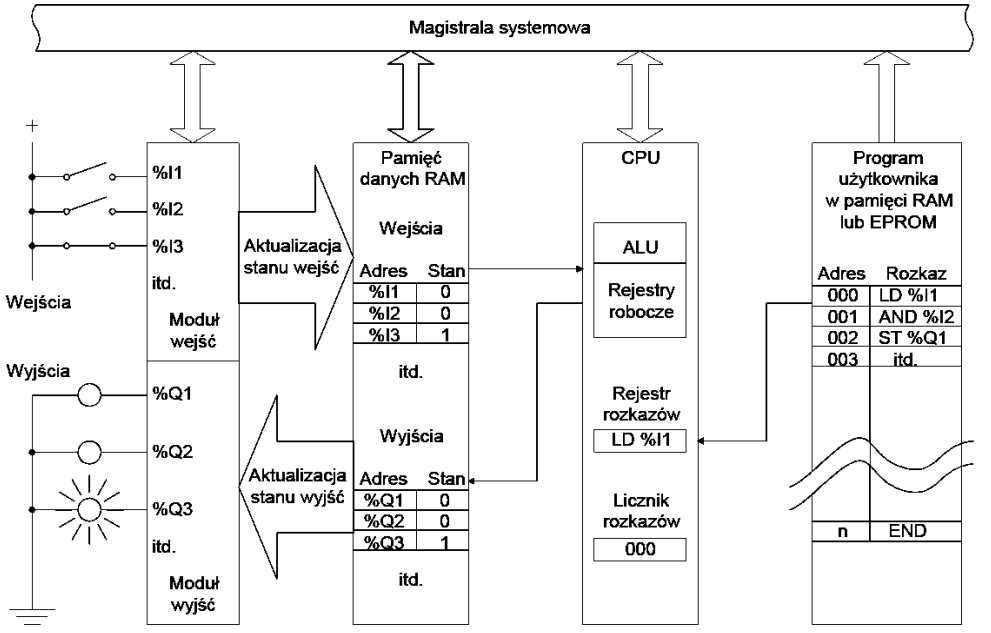
Rysunek 17. Cykl pracy sterownika PLC

Możliwe jest pomijanie niektórych faz w kolejnym cyklu pracy sterownika, co wydatnie przyspiesza jego pracę. Rozpatrując działanie tego urządzenia wyróżnia się dwa podstawowe tryby pracy:

– tryb RUN – wykonywanie programu (właściwy tryb jego pracy, realizowany jest powyższy cykl);
– tryb STOP – sterownik nie pracuje.

Zmiana trybu pracy wymuszana jest przez programator na wyraźne polecenie programisty lub przy pomocy przełącznika umieszczanego opcjonalnie na obudowie sterownika.

Przykład realizacji programu umieszczono na rysunku 18. W pamięci sterownika umieszczono krótki program. Sterownik aktualizuje stan wejść z modułów wejściowych przypisując w pamięci do adresów fizycznych stan logiczny np.: %I3 ma stan 1. CPU pobiera te informacje oraz program jako rejestr rozkazów, po wykonaniu operacji arytmetyczno-logicznych wylicza wartości wyjść, które następnie zapisywane są w pamięci RAM jako obraz wyjść (adresom fizycznym przypisywane są stany logiczne). W kolejnym kroku następuje aktualizacja stanu wyjść w modułach załączających elementy wykonawcze, na rysunku widoczna jest aktywacja wyjścia %Q3.



Rysunek 18. Schemat blokowy ilustrujący działanie sterownika

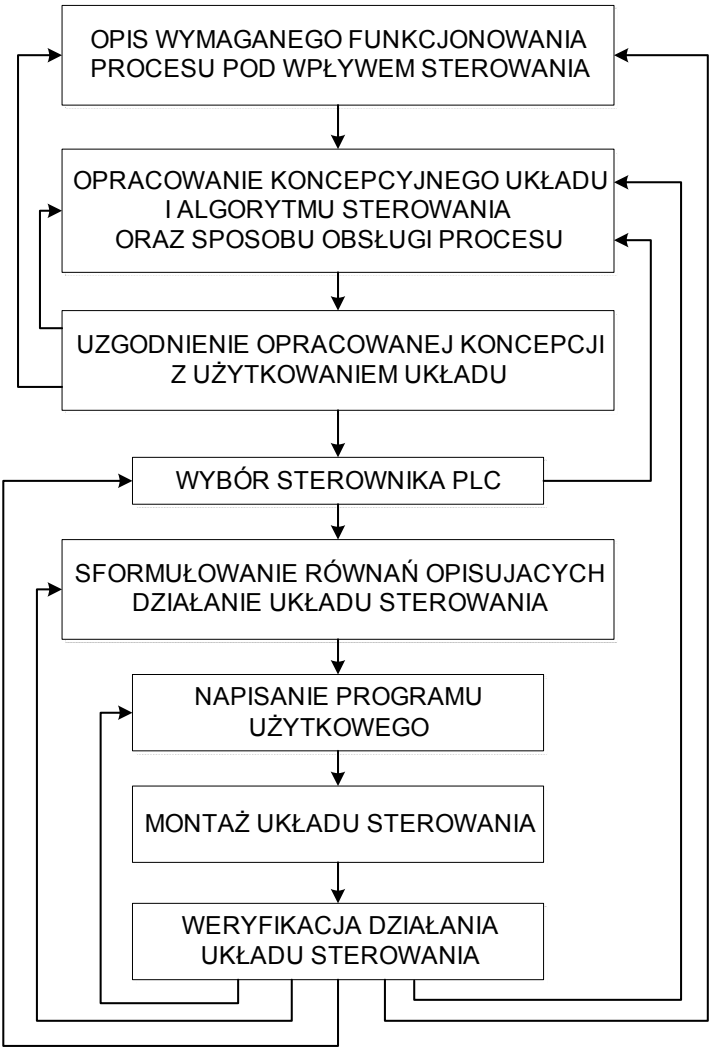
Źródło: Kwaśniewski, 1999

4. METODYKA WDRAŻANIA PLC DO PROCESÓW TECHNOLOGICZNYCH

Przygotowanie systemu automatyki wykorzystującego sterownik programowalny składa się z kilku etapów (rys. 19), przy czym wydzielić można dwie główne części: *projektowanie systemu* i *realizacja projektu*. Pierwszy z etapów, należący do części *projektowanie systemu,* to *opis wymaganego funkcjonowania procesu pod wpływem sterowania.* Na tym etapie konieczne jest dokładne rozpoznanie sterowanego procesu poprzez informacje pochodzące od technologa, zaczerpnięte z literatury, tak aby proces był prowadzony prawidłowo i zgodnie z oczekiwaniami. W dalszej kolejności konieczne jest *opracowanie koncepcyjnego układu i algorytmu sterowania oraz sposobu obsługi procesu*, należy przeanalizować, czy proces ma charakter sekwencyjny, ustalić kolejność sekwencji, dobrać rodzaj sterowania, określić strukturę programu. Struktura może mieć charakter liniowy, ewentualnie blokowy. W blokach należy opisać zadania poszczególnych fragmentów procesu. Najkorzystniejsze jest zamknięcie procesu koncepcyjnego w postaci tablic stanów, grafów stanu, schematu blokowego (algorytmu), sekwencyjnego schematu funkcjonalnego. Będą one zrozumiałe i czytelne dla inżyniera programisty odpowiedzialnego za napisanie kodu programu sterującego. Na końcu części projektowej konieczne jest oszacowanie wielkości obiektu poprzez zestawienie liczbowe i parametryczne wejść i wyjść koniecznych do sterowania danego procesu. W sytuacji kiedy gotowa jest pierwsza wersja projektu, należy *uzgodnić opracowaną koncepcję z użytkownikiem układu,* jeżeli nie odpowiada oczekiwaniom, można ją ponownie zmodernizować aż do akceptacji przez użytkownika.

Pierwszym etapem wchodzącym w skład *realizacji projektu* jest *wybór sterownika PLC.* Kierując się wyborem sterownika należy mieć na uwadze rodzaj urządzeń peryferyjnych, znajdujących się w istniejącym systemie sterowania, z którymi będzie współpracował. Liczba wejść/wyjść decyduje o liczbie modułów rozszerzeń. Dalsze prace obejmują konfigurację sterownika, która stanowi rodzaj interfejsu między oprogramowaniem a sprzętem. W ramach konfiguracji konieczne jest przygotowanie opisu urządzeń fizycznych (tj. czujniki, przyciski, przetworniki, styczniki, elementy wykonawcze), najlepiej sporządzić go w formie tablic zawierających: adres logiczny, fizyczny i symboliczny, oznaczenia na schemacie, opis punktu.

Etapem poprzedzającym programowanie jest *sformułowanie równań opisujących działanie układu sterowania*. Są to przeważnie równania logiczne opisujące działanie układu sterowania. Ponadto działanie PLC można opisać przy pomocy tablicy stanów, metodą minimalizacji funkcji logicznych (metoda Karnaugha) itp. Celem ułatwienia programowania, zaleca się sporządzenie algorytmu sterowania w postaci grafu stanów lub schematu blokowego. *Napisanie programu użytkowego* jest możliwe z wykorzystaniem kilku języków objętych normą takich jak: język drabinkowy, tekst strukturalny, lista instrukcji, funkcjonalny schemat blokowy. Często zachodzi konieczność łączenia programu napisanego różnymi językami, wówczas powstaje tzw. program hybrydowy.



Rysunek 19. Etapy automatyzacji procesu technologicznego z zastosowaniem PLC

Zalecane jest tworzenie komentarzy zawartych w kodzie programu, ułatwiających interpretację jego zapisu oraz tworzenie dokumentacji (notatek) przy użyciu *znaczników* zwanych też *flagami*. Po napisaniu programu konieczna jest weryfikacja jego działania, wielu producentów sterowników oferuje oprogramowanie umożliwiające symulację programu sterującego. Pozwoli ona wyeliminować błędy, które mogłyby skutkować uszkodzeniem urządzeń wykonawczych. Często możliwa jest również symulacja programu na sterowniku w trybie *online*, ale bez aktywacji jego wyjść (załączania urządzeń wykonawczych), warto korzystać z każdej możliwości weryfikacji programu przed pierwszym roz-

ruchem sterowanej linii technologicznej, gdyż ewentualne błędy mogą generować duże koszty usuwania uszkodzeń. Znajdując się w fazie *montażu układu sterowania* konieczne jest przestrzeganie wszelkich norm dotyczących instalowanych urządzeń, należy robić to zgodnie z dokumentacją techniczną i rzetelnie. W sytuacji gdy dochodzi do wypadku, analizowane są wszystkie nieścisłości, które mogłyby mieć na niego wpływ. Ostatnim etapem kończącym proces automatyzacji odcinka technologicznego jest *weryfikacja działania układu sterowania*. Polega ona na sprawdzeniu działania wszystkich torów sygnałowych, wykorzystaniu symulatora obiektu dla celów weryfikacji algorytmu sterowania lub jego poszczególnych sekcji. Większość sterowników posiada możliwość sterowania wykonywaniem programu, która może być zastosowana do weryfikowania jego działania, tj.:

– wykonanie tylko *pojedynczego cyklu programu*, po którym sterownik zatrzymuje się;
– punkty kontrolne, w których następuje zatrzymanie wykonywanego programu (może ono zostać określone warunkiem);
– wykonanie programu *krok po kroku*, a więc po każdej instrukcji następuje zatrzymanie sterownika.

Do zadań inżyniera wdrażającego system sterowania należy również wykonanie kopii programu sterującego na zewnętrzny nośnik oraz wykonanie przejrzystej i zrozumiałej dokumentacji technicznej programu dla ewentualnych innych uczestników projektu, użytkowników oraz konserwatorów.

5. PROGRAMOWANIE STEROWNIKÓW PLC

Najważniejszym zadaniem sterownika jest generowanie sygnałów sterujących w odpowiedzi na zmiany sygnałów wejściowych, zgodnie z przyjętym algorytmem sterowania. Najbardziej interesującym elementem systemu sterowania jest możliwość ich samodzielnego programowania. Zasoby języków programowania sterowników PLC są dostosowane do wymagań inżynierii automatyki.

Wspólną cechą wszystkich języków są rozkazy, wyrażenia lub bloki operacji logicznych, ponadto zachodzi podobieństwo sposobu ich przedstawiania do form stosowanych w technice przekaźnikowej. Wraz z rozwojem technik informatycznych należy oczekiwać, że również języki programowania PLC będą ewoluować. Obecnie producenci oferują możliwość programowania wybranych bloków funkcyjnych językami nieformalnymi. Najpopularniejszym językiem programowania jest język schematów drabinkowych LD.

Rozpoczynając opis metodyki programowania PLC należy zwrócić uwagę na uporządkowaną strukturę tej czynności. W pierwszym etapie omówione zostaną elementy składające się na model oprogramowania.

5.1. Model oprogramowania

W skład języków programowania PLC wchodzą następujące elementy:
- typy danych (ang. Data types);
- jednostki organizacyjne oprogramowania (ang. Program Organization Units – POU);
- elementy schematu funkcji sekwencyjnej (ang. Sequential Function Chart);
- elementy konfiguracji (ang. Configuration elements).

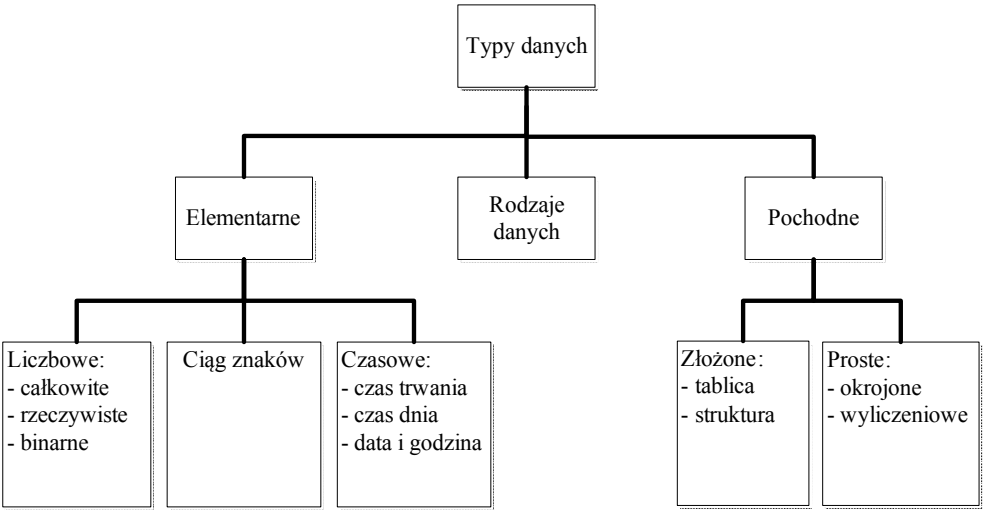
Typy danych służą określeniu struktury danych w sterowniku takich jak zmienne procesowe, stałe (współczynniki) i zakresu ich wartości, ponadto określeniu obszaru pamięci potrzebnego do ich przechowywania. Wyznaczają zbiór operacji, które mogą być na nich wykonywane. Klasyfikację przedstawiono na rysunku 20.

Zdefiniowane typy danych, jakie mogą być używane w systemach sterownikowych, noszą nazwę *elementarnych* (tab. 3). W grupie danych elementarnych najmniejszą jednostką danych jest BOOL (typ danych logicznych przyjmujący dwie wartości 0 lub1).

Korzystając z elementarnych typów danych programista może definiować swoje typy danych określane przez normę *pochodnymi typami*. Takie rozwiązanie umożliwia tworzenie modelu danych najbardziej odpowiedniego dla aplikacji użytkownika. W oknie deklaracji zmiennych należy używać następującego słowa klucz: TYPE … END_TYPE. Przykładowo deklaracja danych okrojonych (zdefiniowany zakres) powinna wyglądać następująco:

```
TYPE
ANALOG_DATA : INT (-4095…4095)
```

END_TYPE
Deklaracja danych może zawierać przyporządkowanie wartości początkowej.



Rysunek 20. Klasyfikacja typów danych używanych w PLC

Źródło: Legierski i in., 1998

Tabela 3. Elementarne typy danych

Oznaczenie	Typ danej	Ilość bitów	Zakres wartości	Opis
BOOL	Boolean	1	0-1	Zmienne tego typu mogą przyjmować tylko dwie wartości czyli „0” lub „1”
SINT	short integer	8	-128 : 127	„Krótka” liczba całkowita
INT	integer	16	-32 768 : 32 767	Liczba całkowita
DINT	double integer	32	$-2^{31} : (2^{31}-1)$	„Podwójna” liczba całkowita
LINT	long integer	64	$-2^{63} : (2^{63}-1)$	„Długa” liczba całkowita
USINT	unsigned short integer	8	0 : 255	„Krótka” liczba całkowita nieznakowana (przyjmuje tylko wartości dodatnie)
UINT	unsigned integer	16	0 : 65535	Liczba całkowita nieznakowana (przyjmuje tylko wartości dodatnie)
UDINT	unsigned double integer	32	$0 : (2^{32}-1)$	„Podwójna” liczba całkowita nieznakowana (przyjmuje tylko wartości dodatnie)
ULINT	unsigned long	64	$0 : (2^{64}-1)$	„Długa” liczba całkowita nieznakowana

Oznaczenie	Typ danej	Ilość bitów	Zakres wartości	Opis
	integer			(przyjmuje tylko wartości dodatnie)
REAL	real	32		Liczba rzeczywista (zmiennoprzecinkowa)
LREAL	long real	64		Długa liczba rzeczywista
STRING				Dowolne znaki (litery i cyfry), jednak cyfry będą traktowane jako znak a nie określona wartość
TIME	czas			Typ danej występujący w timerach
DATE	data			Format zapisu daty
BYTE	ciąg 8 bitów	8	brak	
WORD	ciąg 16 bitów	16	brak	
DWORD	ciąg 32 bitów	32	brak	
LWORD	ciąg 64 bitów	64	brak	

Jednostki organizacyjne oprogramowania stanowią najmniejsze niezależne jednostki oprogramowania aplikacji użytkownika. Stanowią także niezależne moduły oprogramowania, kompilowane oddzielnie. Wyróżniono trzy typy POU:
- funkcje (ang. Functions);
- bloki funkcjonalne (ang. Function Blocks);
- programy (ang. Programs).

Każda jednostka POU zawiera następujące elementy:
- typ i nazwę, które są określane w momencie wyboru nowego POU, a w przypadku funkcji określany jest także typ funkcji;
- deklarację zmiennych używanych w danym POU;
- ciało (body), czyli algorytm działania opisany w odpowiednim języku programowania.

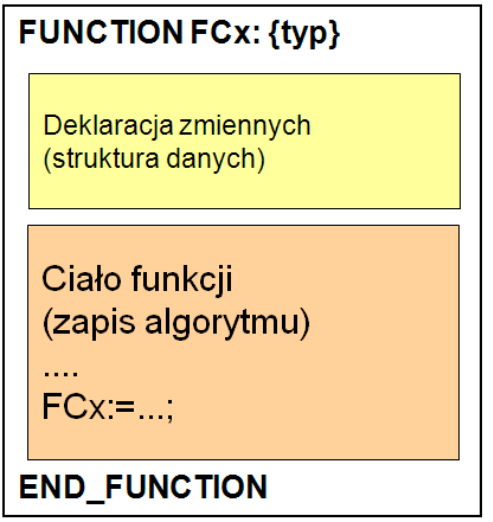
POU określony jest przez odpowiednie słowo kluczowe: ***PROGRAM***, ***FUNCTION_BLOCK*** lub ***FUNCTION***, po którym występuje nazwa własna POU. Dodatkowo w przypadku funkcji określa się także typ funkcji, np.: ***FUNCTION ALARM :BOOL***, oznacza deklarację funkcji o nazwie *ALARM*, która na wyjściu daje wartość typu *BOOL*.

Podstawową zasadą tworzenia *funkcji* jest, aby instrukcje zawarte w ciele funkcji, zastosowane do zmiennych wejściowych, zawsze dostarczały jednoznacznego wyniku w postaci wartości zwracanej funkcji, bez względu jak często i w jakiej chwili jest wywoływana. Reasumując jest to element statyczny o wielu wejściach i zwracający jedną wartość określonego typu (rys. 21).

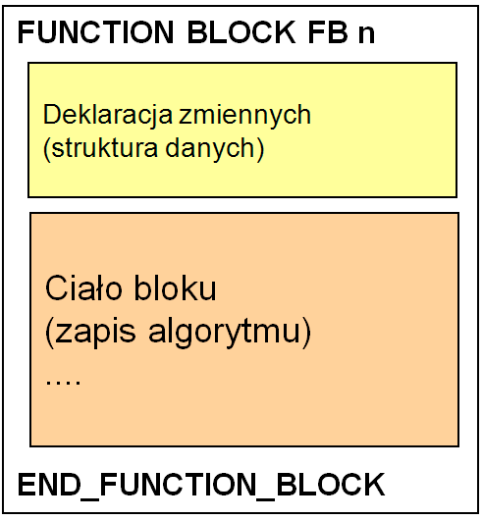
W językach graficznych funkcje są reprezentowane w postaci prostokątów o wymiarach zależnych od liczby wejść. Nazwę funkcji lub symbol umieszcza się wewnątrz prostokąta, a kierunek przetwarzania realizowany jest od lewej do prawej strony. Negacja sygnałów logicznych jest wyróżniana kółkiem.

Wyróżnia się następujące rodzaje funkcji: logiczne, arytmetyczne i matematyczne, konwersji typu, operujące na ciągach znaków, porównania.

Blok funkcjonalny jest jednostką POU, która z chwilą wykonania może dawać na wyjściu jedną wartość lub wiele wartości w przeciwieństwie do funkcji. Definicja FB to inaczej określenie pewnej struktury danych i związanego z nią algorytmu (rys. 22).



Rysunek 21. Formalizm zapisu funkcji



Rysunek 22. Formalizm zapisu bloku funkcjonalnego

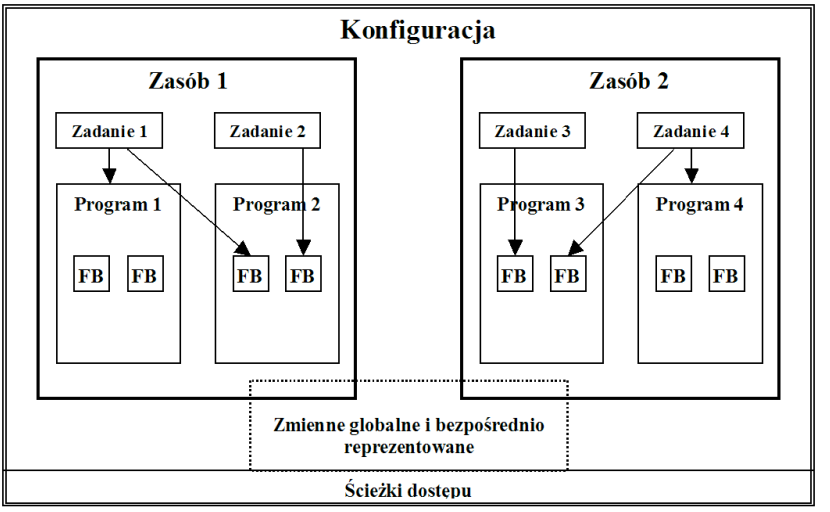
Ukonkretnienie inaczej *instancja* (inny egzemplarz) jest to lokalna kopia bloku tworzona w programie. Poszczególne instancje tego samego FB mają tę samą strukturę i realizują ten sam algorytm, ale działają w pełni niezależnie od siebie. Dodatkowo – każda ze zdefiniowanych w programie instancji może być wielokrotnie wywoływana w programie (np. impulsy z różnych wejść mogą być zliczane przez ten sam licznik). Bloki FB mogą być wywoływane przez programy lub inne FB, same też mogą wywoływać funkcje. Przykładem bloków funkcjonalnych są: liczniki, timery, elementy bistabilne, elementy detekcji zbocza. W językach tekstowych egzemplarz bloku tworzy się w obrębie deklaracji VAR … END_VAR.

Przez *programy* należy rozumieć zbiór logicznie powiązanych elementów języka programowania, koniecznych do zamierzonego przetwarzania sygnałów w celu sterowania urządzeniem lub procesem. W klasycznym ujęciu programowania *funkcje* i *bloki funkcjonalne* należałoby traktować jako podprogramy, natomiast *program* stanowiłby główny szkielet całości. Jednak norma PN-EN 61131 nie zezwala na formalne stosowanie podprogramów (ich rolę pełnią funkcje i bloki funkcjonalne). Stąd jednostka POU o nazwie program stanowi nadrzędny program zarządzający całym algorytmem. W jednostkach z wielozadaniowymi systemami operacyjnymi możliwe jest stosowanie kilku programów

jednocześnie. Deklaracja programu odbywa się za pomocą słów kluczy PROGRAM … END_PROGRAM. Programy mogą być realizowane tylko w ramach zasobów oraz tylko przez skojarzenie z konkretnym zadaniem. W programie możliwe jest deklarowanie zmiennych bezpośrednio reprezentowanych, tj. %I, %Q, %M, odwołujących się do adresów fizycznych, jednak ze względu na gorszą czytelność kodu programu nie jest to zalecane. Program może zawierać deklarację zmiennych globalnych oraz ścieżek dostępu do innych zasobów: VAR_GLOBAL, VAR_ACCESS.

Elementy konfiguracji odpowiadają za instalowanie i uruchamianie programów w systemach sterowania. Zależności pomiędzy elementami składającymi się na konfigurację przedstawia rysunek 23. Wyróżniane są następujące składowe:

- *konfiguracja (ang. Configurations)*, jest elementem języka, który odpowiada *systemowi* sterowników programowalnych rozumianemu jako całość, obejmującą wszystkie pozostałe elementy oprogramowania;
- *zasoby (ang. Resources)*, są programowym odpowiednikiem sprzętu realizującego funkcje przetwarzania sygnałów, łącznie z funkcjami określonymi przez podłączone czujniki i elementy wykonawcze oraz urządzenia operatorskie MMI (ang. Man-Machine Interface);
- zadania *(ang. Tasks)*, elementy nadzorujące wykonanie programu lub pewnej jego logicznej części;
- zmienne globalne *(ang. Global Variables)*, dostępne dla wszystkich elementów programu;
- ścieżki dostępu *(ang. Access Paths)*, elementy oprogramowania zapewniające wymianę danych z innymi konfiguracjami lub konfiguracją a innym elementem (np. systemem SCADA);
- zmienna bezpośrednio reprezentowana *(ang. Directly Represented Variable)* taka, której nazwą jest adres fizyczny (wejścia, wyjścia lub komórki pamięci wewnętrznej).



Rysunek 23. Schemat blokowy modelu oprogramowania

Źródło: Kamiński, 2009

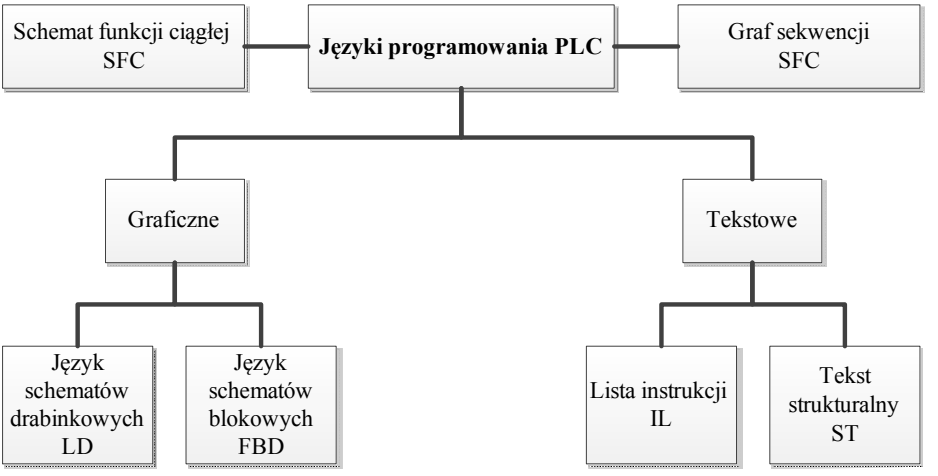
Konfiguracja może zawierać jeden lub więcej *zasobów*, z kolei każdy *zasób* może zawierać kilka *programów* wykonywanych pod kontrolą *zadania*. Programy zawierają funkcje, bloki funkcjonalne oraz zmienne lokalne. Konfiguracje i zasoby mogą być uruchamiane i zatrzymywane za pomocą funkcji realizowanych przez interfejs operatora, funkcji systemu operacyjnego lub funkcji programowania, testowania i monitorowania. Uruchomienie konfiguracji powinno spowodować zainicjowanie zdefiniowanych w niej *zmiennych globalnych,* a następnie uruchomienie wszystkich należących do niej *zasobów,* tak aby natychmiastowo były dostępne dla innych aplikacji. Uruchomienie zasobu powinno z kolei spowodować inicjowanie wszystkich występujących w nim *zmiennych*, a następnie umożliwienie działania wszystkim zadaniom zdefiniowanym w zasobie.

Przedstawiony model komunikacji ma duże znaczenie w perspektywie prac nad systemami wieloprocesorowymi, które wymagają doskonałej organizacji zadań i zarządzania równoległego systemami pracującymi w rygorze czasu rzeczywistego.

5.2. Języki programowania

Programowanie sterowników PLC realizuje się za pomocą specjalnych urządzeń mikrokomputerowych zwanych programatorami lub komputerów PC z zainstalowanym oprogramowaniem narzędziowym zawierającym język (-i) programowania.

W związku z dynamicznym rozwojem sterowników PLC oraz rosnącą liczbą producentów za pomocą normy PN-EN 61131 ujednolicono stosowane dotychczas języki programowania. Pojęcia podstawowe, zasady ogólne, model programowy i model komunikacyjny zostały opisane za pomocą formalnych definicji. Specyfikują one syntaktykę i semantykę tekstowych i graficznych języków programowania oraz elementy konfiguracji wspomagające instalację oprogramowania w sterownikach. Języki podzielone zostały na dwie grupy obejmujące: *języki tekstowe* i *graficzne*, ponadto norma doprecyzowała semantykę dwóch innych niewchodzących w skład tych grup (rys. 24).



Rysunek 24. Klasyfikacja języków programowania PLC

W grupie ***języków tekstowych*** zdefiniowane zostały następujące języki:

- *lista rozkazów (instrukcji) IL (ang. Instruction List)*, będąca odpowiednikiem języka typu *asembler*, którego zbiór instrukcji obejmuje operacje logiczne, arytmetyczne, operacje relacji, jak również funkcje przerzutników, czasomierzy, liczników itp. Program wykonywany jest za pomocą rozkazów wykonywanych od góry do dołu i od lewej do prawej strony. Każdy nowy rozkaz powinien zaczynać się od nowej linii. Poszczególnym funkcjom przypisano określenia – operatory;
- *tekst strukturalny ST (ang. Structured Text)* jest odpowiednikiem języka algorytmicznego *wysokiego poziomu*, zawierającego struktury programowe i polecenia podobne do występujących w językach typu PASCAL lub C. Jest on przeznaczony głównie do opisu złożonych wyrażeń, których nie można zrealizować w językach graficznych (lub jest to bardzo utrudnione). Język ST jest podstawowym językiem używanym do opisywania akcji w poszczególnych krokach i warunkach struktur wyrażonych w językach SFC i CFC. Program wykonywany jest przez szereg instrukcji i obliczanie wartości wyrażeń. Funkcje nie mają przypisanych symboli, określane są za pomocą abstrakcyjnych poleceń.

Do grupy języków graficznych należą:

- *schemat drabinkowy LD (ang. Ladder Diagram)*, w swojej strukturze podobny do stykowych obwodów przekaźnikowych. Oprócz symboli styków, cewek i połączeń między nimi, dopuszcza się także użycie funkcji (np. arytmetycznych, logicznych, porównań, relacji) oraz bloków funkcjonalnych (np. przerzutniki, czasomierze, liczniki). Wykonanie programu polega na umownym przepływie prądu między szynami prądowymi przez szczeble. Symbole poszczególnych funkcji umieszcza się na szczeblach zamykających symboliczny obwód elektryczny;
- *funkcjonalny schemat blokowy FBD (ang. Function Block Diagram)* stanowi odpowiednik schematu przepływu sygnału dla obwodów logicznych przedstawionych w formie połączonych bramek logicznych, funkcji i bloków funkcjonalnych, takich jak w języku LD oraz połączeń między nimi. Wykonanie programu odbywa się na zasadzie przepływu sygnału między elementami programu. Symbole poszczególnych funkcji oznaczone są prostokątnymi blokami, natomiast połączenia między nimi to poziome lub pionowe linie.

Ponadto w normie przedstawiono sposób tworzenia struktury wewnętrznej programu w postaci *schematu funkcji sekwencyjnej SFC – graf sekwencji* (ang. *Sequential Function Chart*), który pozwala na opisywanie zadań sterowania sekwencyjnego za pomocą grafów zawierających kroki *(etapy)* i warunki przejścia *(tranzycji)* między tymi krokami. W celu otrzymania odpowiedniej struktury programu można wykorzystać SFC, w którym definicje akcji dla poszczególnych kroków oraz warunki przejścia programuje się w jednym z czterech wymienionych wcześniej języków.

W graficznym edytorze *schematów funkcji ciągłej CFC* w odróżnieniu od SFC elementy nie są zestawiane w sieci, lecz mogą być dowolnie rozmieszczane. Elementami na liście do przetwarzania mogą być moduły, wejście/wyjście, skok, etykieta, powrót i komentarz. Wejścia i wyjścia tych elementów można łączyć poprzez przeciąganie połączenia za pomocą myszy. Linia połączenia jest rysowana automatycznie. Przy uwzględnieniu istniejących połączeń program rysuje najkrótszą linię połączenia. Podczas przesuwania elementów linie połączenia są automatycznie optymalizowane. Jeśli linia połączenia z powodu braku miejsca nie może być narysowana, wówczas między wejściem, a przyna-

leżącym wyjściem wyświetlana jest linia czerwona. Gdy tylko zwolni się miejsce, linia ta zostanie przekształcona w linię połączenia. Edytor graficzny CFS posiada przewagę nad edytorem bloków funkcji FBD, polegającą na możliwości bezpośredniego dodawania sprzężeń zwrotnych.

Każdy z języków programowania posiada pewne cechy, które powodują, że do zaprogramowania niektórych zagadnień nadaje się lepiej niż inne języki. Język LD jest najodpowiedniejszy do programowania operacji logicznych (algebra Boole'a). Języki tekstowe są wygodniejsze w procedurach zarządzania pamięcią lub przy programowaniu obliczeń iteracyjnych. Programy w językach graficznych są stosunkowo łatwe do analizy, czego nie można powiedzieć o języku IL, który z kolei jest najbardziej elastyczny itd. Niektóre pakiety programowania umożliwiają wykonanie takiego przetłumaczenia, aczkolwiek norma nie stawia takich wymagań. Podstawowy problem stanowi tu niepełna kompatybilność poszczególnych języków programowania. Szczególna trudność pojawia się przy przechodzeniu z języków graficznych na tekstowe i odwrotnie. Wynika to z różnego sposobu przetwarzania w obu grupach języków. Te pierwsze są przede wszystkim językami proceduralnymi, tzn. instrukcje są wykonywane w nich jedne po drugich, podczas gdy podstawą w językach graficznych jest przepływ sygnału („prądu" w języku LD), który może być realizowany równolegle.

Mając do dyspozycji kilka różnych języków programowania, przed wyborem konkretnego z nich należałoby odpowiedzieć sobie na kilka kluczowych pytań. Oczywiście naturalną tendencją jest pozostawanie przy tym, który znamy. Jednak zawsze warto wziąć pod uwagę umieszczoną poniżej listę cech całej piątki:

- łatwość nadzoru przez użytkownika końcowego: SFC;
- powszechność i akceptacja języka: LD,
- znajomość i akceptacja w Europie: IL lub ST;
- prędkość wykonywania przez PLC: IL lub ST;
- aplikacje wykorzystujące głównie cyfrowe We/Wy oraz prosta regulacja ciągła: LD lub FBD;
- łatwość dokonywania zmian w kodzie: LD;
- łatwość i umiejętność obsługi: ST;
- łatwość w implementacji skomplikowanych operacji matematycznych: ST;
- aplikacje, które cechują powtarzające się operacje lub procesy wymagające łączenia i jednoczesności operacji: SFC.

We wszystkich językach występują pewne elementy wspólne, należą do nich *ograniczniki (ang. Delimiters)* są to znaki specjalne, takie jak + - *$* = := *# ; ( )* * oraz *spacja*. Użytkownik może wstawić spację wszędzie w tekstach programów, za wyjątkiem słów kluczowych, literałów oraz identyfikatorów. *Słowa kluczowe (ang. Keywords)* są identyfikatorami (nazwami) standardowymi, zdefiniowanymi w normie jako elementy danego języka. Słowa kluczowe nie mogą być używane jako identyfikatory wprowadzane przez użytkownika (nazwy własne). Zwyczajowo przyjmuje się, że słowa kluczowe powinny być pisane z użyciem dużych liter. Do słów kluczowych należą:

- nazwy deklaracji, np. PROGRAM, FUNCTION, VAR_INPUT, END_PROGRAM;
- nazwy elementarnych typów danych, np. *BOOL*, *INT*, *REAL*, *TIME*;

– nazwy standardowych funkcji i bloków funkcjonalnych, np. *AND*, *ADD*, *MOVE*, *SHL*, *TON*, *CTU*;
– nazwy parametrów wejściowych i wyjściowych standardowych funkcji i bloków funkcjonalnych;
– operatory języka IL;
– operatory i instrukcje języka ST;
– elementy SFC;
– zmienne *EN* i *ENO* w językach graficznych.

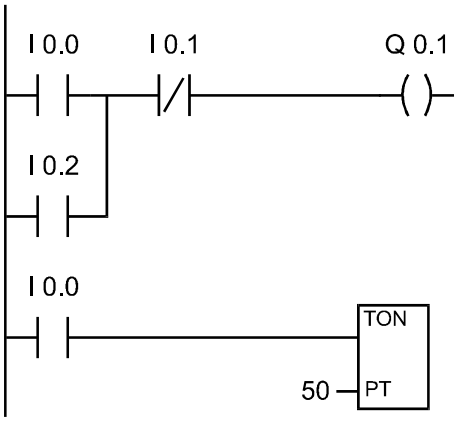
Literały (ang. *literals*) służą przedstawianiu wartości danych (zmiennych lub stałych). Ich format zależy od typu danej, który jednocześnie określa zakres zmienności.

Identyfikatory (ang. *identifiers*) są ciągami znaków alfanumerycznych, których programista może użyć do nazwania własnych: zmiennych, jednostek POU itp. Nazwa taka musi zaczynać się od litery lub pojedynczego znaku podkreślenia _ , po których może występować dowolna liczba liter, cyfr lub znaków podkreślenia. Jednak dopuszczalna długość nazwy może zależeć od konkretnego systemu programującego. Norma IEC 61131-3 wymaga tylko, aby przynajmniej sześć pierwszych znaków nazwy było znaczących. W programowaniu nie rozróżnia się między dużymi a małymi literami, np.: *ALA, Ala* czy *ala* powinny być traktowane przez system PLC jednakowo.

Programista może także wstawić do tekstu programu swoje *komentarze* w dowolnym miejscu, w którym może używać spacji. Jedynie w języku IL komentarze powinny być wstawione na końcu. Komentarze ograniczone są przez kombinację znaków *(** oraz **)*. Komentarze nie mają żadnego znaczenia składniowego ani semantycznego, służą jedynie do przedstawienia dodatkowej informacji użytkownikowi. Nie mogą być one zagnieżdżane, tzn. nie można wstawiać komentarza do komentarza.

5.2.1. Schematy drabinkowe – LD

Język ten jest graficznym odzwierciedleniem rozmieszczenia elementów stykowych, tj. przekaźników i styczników w układach sterowania. Program składa się z obwodów *(ang. Network)* – zbioru logicznie połączonych symboli graficznych (Flaga, 2010). Symbole te umieszcza się w obwodach w sposób podobny do szczebli. Do każdego obwodu można dołączyć etykietę z nazwą lub liczbą. Obwody LD z lewej i prawej strony zamknięte są szynami prądowymi. Zazwyczaj w schematach pomija się rysownie prawej szyny, może być rysowana w sposób jawny lub pozostawać w domyśle. Na rysunku 25 występuje ona jako domyślna.



Rysunek 25. Schemat drabinkowy z prawą szyną domyślną

Przyjmuje się, że lewa szyna niesie ładunek elektryczny (tzw. szyna prądowa), natomiast stan prawej szyny jest nieokreślony. W języku drabinkowym program

wykonuje się przez przepływ prądu z lewej szyny poprzez szczeble na prawą szynę, jak w schemacie drabinkowym systemów przekaźników elektromechanicznych. Podczas programowania w tym języku należy pamiętać o czterech podstawowych zasadach:

– wartości elementów obwodu wyznaczane są dopiero po wyznaczeniu wartości wszystkich wejść;
– wartość elementu nie może być wyznaczona przed wyznaczeniem wartości wszystkich jego wyjść;
– wykonywanie programu nie może być zakończone do czasu wyznaczenia wartości wyjść dla wszystkich elementów obwodu;
– obwody numerowane są z góry na dół zgodnie z kolejnością pojawienia się na schemacie, z wyjątkiem pojawienia się elementów kontrolnych zmieniających kolejność.

Elementy mogą przyjmować dwa stany logiczne ON (podczas przepływu prądu) oraz OFF (podczas braku przepływu). Stany ON i OFF odpowiadają boolowskim stanom 1 i 0. Połączenia w diagramach drabinkowych są dwóch typów: pionowe i poziome. Połączenie poziome – przedstawiane za pomocą poziomej linii, przekazuje sygnał z elementu po lewej stronie do elementu po prawej. Połączenia pionowe – przedstawiane za pomocą pionowej linii łączącej połączenia poziome, przekazuje sygnał z elementów po lewej stronie do elementów po prawej, realizując równocześnie funkcje OR; oznacza to, że jeśli po lewej stronie połączenia wszystkie sygnały są w stanie OFF, sygnał po prawej również jest w stanie OFF. Jeśli jednak przynajmniej jeden sygnał po lewej stronie połączenia ma wartość ON, sygnał po prawej stronie również jest w stanie ON. Sygnał w połączeniach pionowych przekazywany jest do elementów po prawej stronie i nie może być przekazywany do elementów po lewej stronie połączenia.

Wiele sterowników ma ograniczoną maksymalną długość obwodu – jeśli program nie mieści się w jednej linii kodu, można ją przedłużyć za pomocą łączników. Łączniki nie są elementami sterującymi, sygnalizują jedynie, że w następnej linii kontynuowany jest program z poprzedniej. Na schemacie występują również elementy umożliwiające skoki i powroty programu. Komentarz do programu może być umieszczony w dowolnym miejscu programu, musi być jednak zawarty między znakami „*”, chyba że producent udostępnia stosowanie obiektów tzw. notatek tekstowych.

Podstawowymi elementami sterującymi są *cewki* i *styki*. *Styki* są elementami programu spełniającymi funkcje bramki logicznej AND stanu po lewej stronie i stanu przypisanego do styku. *Styki* umożliwiają przesyłanie sygnału z lewej szyny (prądowej) do cewek umieszczonych na drugiej szynie lub końcu szczebla. *Cewki* są drugim typem elementów sterujących, realizują przenoszenie stanu sygnałów z lewej strony na prawą, nie zmieniając ich wartości i przypisując odpowiednią wartość ze względu na stan połączeń i zasadę działania cewki. Istnieje kilka typów cewek:

– cewki zwykłe, których zadaniem jest zapamiętanie obrazu stanu połączeń;
– cewki zatrzaskowe, które realizują funkcje przerzutnika SR lub RS;
– cewki podtrzymujące, z pamięcią, które zapamiętują zmienną przypisaną do cewki i odtwarzają po ponownym uruchomieniu programu;
– cewki impulsowe, służą do testowania zmian sygnału połączenia po lewej stronie.

Jeżeli parametry sygnału zgadzają się z zadanymi parametrami cewki przyjmuje ona wartość 1. W tabeli 4 i 5 przedstawiono standardowe elementy schematu drabinkowego. Przedstawione w tabeli 4 skoki bezwarunkowe mają miejsce w sytuacji, gdy żaden element

nie poprzedza polecenia skoku, możliwe są jednak skoki warunkowe opisane połączeniem dowolnych styków funkcją AND lub OR.

Tabela 4. Podstawowe elementy języka schematów drabinkowych – łączniki

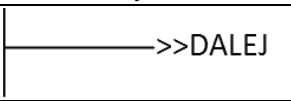
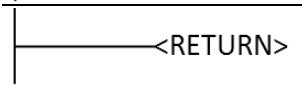
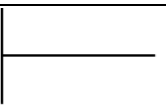
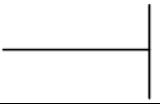
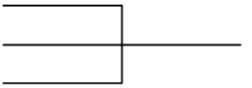
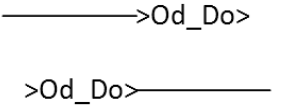
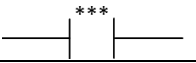
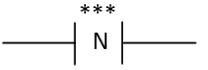
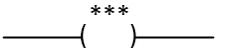
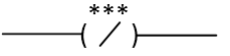
Typ elementu	Symbol	Nazwa elementu
skoki i powroty programu		skok bezwarunkowy do etykiety DALEJ
		powrót bezwarunkowy do początku programu
szyny prądowe i elementy łączące		lewa szyna prądowa (z połączeniem poziomym)
		prawa szyna prądowa (z połączeniem poziomym)
		połączenie poziome
		połączenie pionowe (z połączeniem poziomym)
		łącznik (górna część to zakończenie linii, część dolna to początek nowej linii)

Tabela 5. Podstawowe elementy schematów drabinkowych – symbole styków i cewek

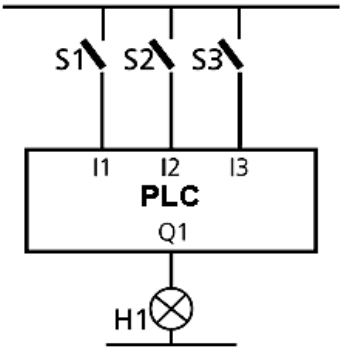
Typ elementu	Symbol	Nazwa elementu
styki statyczne		styk zwierny
		styk rozwierny
styki impulsowe (wrażliwe na zbocze)		styk wrażliwy na zbocze narastające
		styk wrażliwy na zbocze opadające
cewki zwykłe		cewka
		cewka negująca

Typ elementu	Symbol	Nazwa elementu
cewki zatrzaskiwane	*** (S)	cewka ustawiająca SET
	*** (R)	cewka kasująca RESET
cewki podtrzymywane (z pamięcią)	*** (M)	cewka z pamięcią stanu
	*** (SM)	cewka ustawiająca z pamięcią stanu
	*** (RM)	cewka kasująca z pamięcią stanu
cewki impulsowe (wrażliwe na zbocze)	*** (P)	cewka wrażliwa na zbocze rosnące
	*** (N)	cewka wrażliwa na zbocze malejące

Język schematów drabinkowych pozwala na realizację funkcji logicznych, ale programista musi je zbudować samodzielnie, korzystając z dostępnych elementów i własnej wiedzy. Poniżej przedstawiono realizację podstawowych funkcji logicznych na sterownikach PLC.

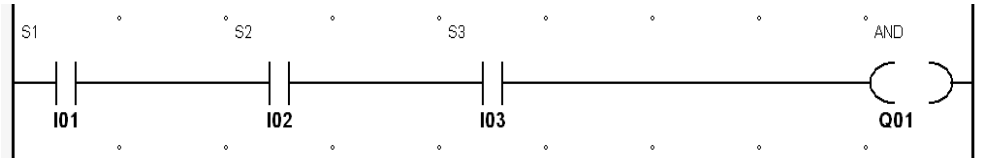
Iloczyn logiczny „AND"

Trzy łączniki mające realizować funkcję logiczną *AND* przedstawia rysunek 26.



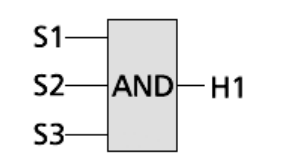
Rysunek 26. Schemat realizacji funkcji logicznej AND

Sygnał wyjściowy (H1) funkcji logicznej AND ma stan logiczny „1", kiedy wszystkie sygnały wejściowe (S) mają stan „1". Stan ten na wejściu odpowiada naciśniętemu stykowi zwiernemu lub niewciśniętemu stykowi rozwiernemu. Reprezentację funkcji AND w języku schematów drabinkowych przedstawiono na rysunku 27.



Rysunek 27. Zapis funkcji AND w języku schematów drabinkowych

Schemat bloków funkcyjnych, tabelę logiczną oraz zapis funkcji logicznej AND zamieszczono na rysunku 28.



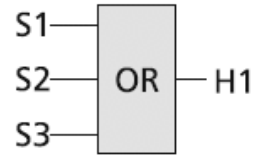
S1	S2	S3	S4
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	1

Rysunek 28. Funkcja AND jako blok, jej zapis oraz tabela logiczna

Suma logiczna „OR”

Trzy łączniki połączone analogicznie jak dla funkcji AND mogą również realizować funkcję logiczną OR, pod warunkiem że reprezentujące je w programie styki będą rozmieszczone równolegle (rys. 26). Sygnał wyjściowy (H1) funkcji logicznej OR ma stan logiczny „1”, kiedy przynajmniej jeden sygnał wejściowy (S) ma stan „1”. Stan „1” na wejściu odpowiada naciśniętemu stykowi zwiernemu lub nienaciśniętemu stykowi rozwiernemu (rys. 29). Schemat bloków funkcyjnych, tabelę logiczną oraz zapis funkcji logicznej OR zamieszczono na rysunku 30.

Rysunek 29. Zapis funkcji OR w języku schematów drabinkowych



S1	S2	S3	H1
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

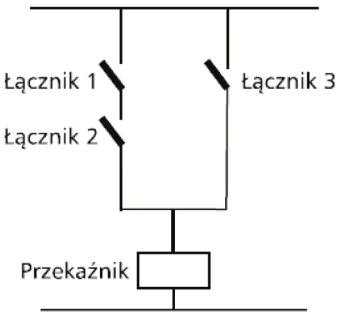
Rysunek 30. Funkcja OR jako blok, jej zapis oraz tabela logiczna

Iloczyn logiczny „AND" przed sumą logiczną „OR"

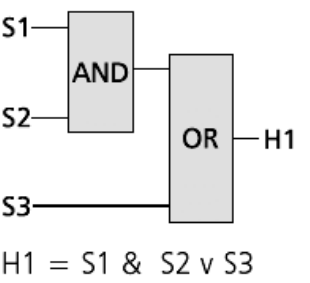
Trzy łączniki mogą tworzyć funkcję logiczną *AND* i *OR*. Wynik tej funkcji logicznej wynika z podstawowej reguły algebry Boola: *operację mnożenia wykonuje się przed operacją dodawania.*

W odniesieniu do analogicznego schematu połączeń zrealizowanego na obiektach stykowych oznacza to, że: *funkcja połączenia szeregowego jest realizowana przed funkcją połączenia równoległego* (rys. 31).

Schemat bloków funkcyjnych, tabelę logiczną oraz zapis funkcji iloczynu logicznego AND przed sumą logiczną OR zamieszczono na rysunku 32. Reprezentację funkcji OR w języku schematów drabinkowych przedstawiono na rysunku 33.

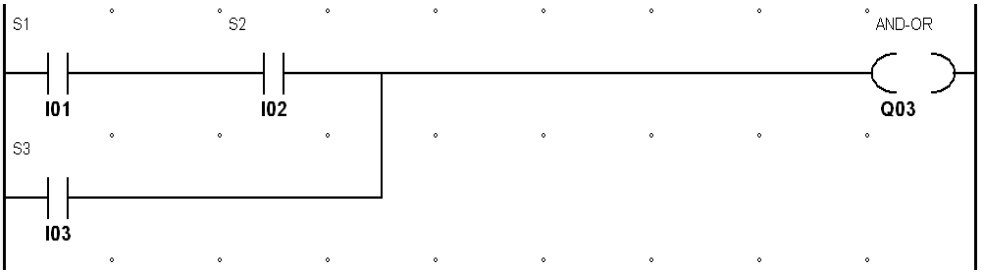


Rysunek 31. Schemat realizacji funkcji iloczynu logicznego AND przed sumą logiczną OR



S1	S2	S3	H1
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

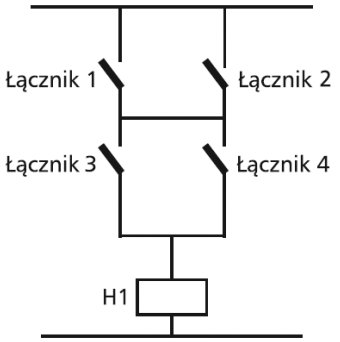
Rysunek 32. Funkcja iloczynu logicznego AND przed sumą logiczną OR jako blok, jej zapis oraz tabela logiczna



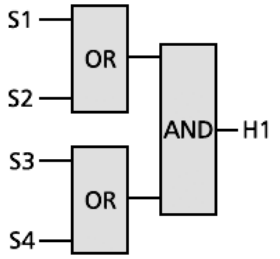
Rysunek 33. Zapis funkcji iloczynu logicznego AND przed sumą logiczną OR w języku schematów drabinkowych

Suma logiczna „OR" przed iloczynem logicznym „AND"

Cztery łączniki mogą tworzyć funkcję logiczną *OR* i *AND* w programie wg analogicznego schematu połączeń stykowych (rys. 34). Taki układ logiczny nazywa się także *koniunkcją*. Ponieważ reguły algebry Boola obowiązują również w przypadku realizacji programu, trzeba zastosować nawiasy (rys. 35). Z tego wynika, że funkcja *OR* realizowana jest jako pierwsza, następnie *AND* i po spełnieniu obu funkcji następuje załączenie przekaźnika H1.



Rysunek 34. Schemat realizacji funkcji sumy logicznej OR przed iloczynem logicznym AND



S1	S2	S3	S4	H1
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	1
0	1	1	0	1
1	1	1	0	1
0	0	0	1	0
1	0	0	1	1
0	1	0	1	1
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

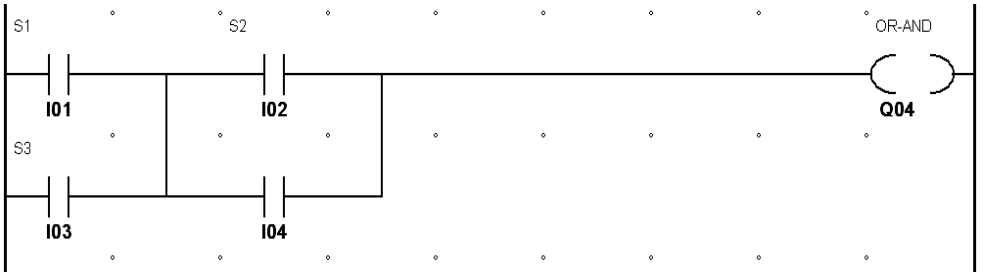
Rysunek 35. Funkcja OR przed AND jako blok, jej zapis oraz tabela logiczna

Zapis funkcji w języku schematów drabinkowych przedstawiono na rysunku 36.

Ponadto język ten pozwala również na programowanie bardziej skomplikowanych aplikacji, jest to możliwe dzięki wywołaniu bloków funkcyjnych lub funkcji; mogą być one domyślne lub scharakteryzowane według preferencji użytkownika. W normie IEC 61131 określono siedem typów standardowych funkcji:

– przekształcania typów;
– liczbowe;

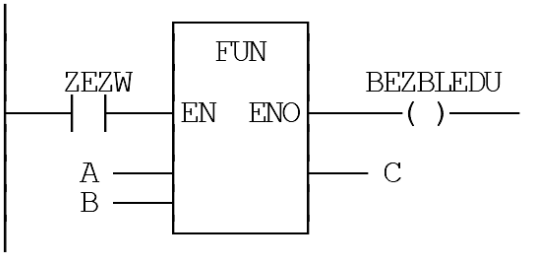
– na sekwencjach bitów;
– preferencji i porównania;
– na sekwencjach znaków;
– na typach danych powiązanych z czasem;
– na obliczeniowych typach danych.



Rysunek 36. Zapis sumy logicznej OR przed iloczynem logicznym AND jako blok w języku schematów drabinkowych

Podczas wywoływania funkcji lub bloków funkcyjnych stosowane są następujące zasady:
– wywołana funkcja lub blok symbolizowana jest przez prostokąt, funkcje i bloki funkcyjne posiadają przynajmniej dwa wyjścia (wyjście ENO oraz wyjście funkcyjne). Bloki funkcyjne mogą posiadać kilka wyjść, natomiast funkcje dwa. Nazwy parametrów formalnych wpisywane są w prostokąt, a nazwy parametrów aktualnych mogą być zapisane nad wejściami/wyjściami na zewnątrz prostokąta. Bloki mogą być łączone bezpośrednio ze sobą (wyjście pierwszego z wejściem drugiego);
– każda przywołana funkcja musi posiadać parę wejścia/wyjścia – EN/ENO. Para ta determinuje wykonanie funkcji. Jeżeli sygnał na wejściu EN przyjmuje wartość 1 (TRUE), funkcja jest wykonywana i na wyjście ENO przenoszony jest sygnał 1, jeżeli sygnał na wejściu EN przyjmuje wartość 0 (FALSE), funkcja nie jest wykonywana i na wyjście ENO przenoszony jest sygnał 0. Para EN/ENO jest wykorzystywana do określenia warunków wykonania funkcji oraz do kontroli błędów;
– blok funkcyjny może posiadać kilka wejść różnego typu, jednak powinien posiadać przynajmniej jedno wyjście cyfrowe, aby zapewnić przesyłanie sygnału przez blok do kolejnych elementów obwodu. Wyjście cyfrowe zazwyczaj oznaczone jest symbolem Q. W blokach funkcyjnych można stosować również parę EN/ENO;
– przy wprowadzaniu bloku funkcyjnego do aplikacji należy określić jego deklaracje – umieszczając nad blokiem jego nazwę.

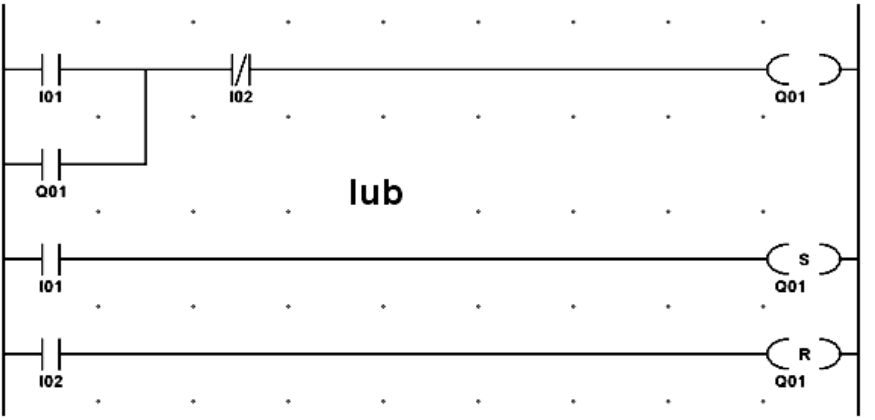
Na rysunku 37 przedstawiono przykład wywołania dowolnej funkcji FUN. Funkcja zostanie spełniona, kiedy sygnał z wejścia ZEZW będzie wynosił 1, wtedy sygnał na wejściu EN będzie miała ten sam stan. Po spełnieniu warunków zawartych w bloku FUN dotyczących relacji pomiędzy A i B, sygnał na wyjściu ENO również przyjmie wartość 1. Wyjście ENO może zostać połączone z kolejnymi elementami układu lub może pozostać wolne. Analogicznie działa większość bloków realizujących funkcje specjalne w programach drabinkowych, wśród nich są timery, liczniki, komparatory wielkości analogowych itd.



Rysunek 37. Przykład wywołania dowolnej funkcji – FUN

Program wykonuje operacje zapisane w obwodach od lewej do prawej, z góry na dół. Można zmienić tę kolejność dzięki użyciu skoków, jednak nie jest to zalecane, gdyż istnieją korzystniejsze rozwiązania warunkowego realizowania fragmentów aplikacji. Możliwe jest przeniesienie akcji cewką z pamięcią stanu (ozn. M). Szczególną uwagę należy zwracać przy programowaniu pętli, korzystając ze skoków wstecz, ponieważ jeśli nie zaistnieją warunki do jej wykonania, może zostać przekroczony maksymalny czas wykonania cyklu programu, co spowoduje zatrzymanie sterownika. W miarę możliwości warunki wykonania pętli powinny być określone w wejściach EN funkcji lub bloków funkcyjnych.

Często przy programowaniu językiem drabinkowym wykorzystuje się sprzężenie zwrotne. Występuje ono wówczas, gdy w jednym cyklu wykonania programu zmienna jest zapamiętywana, a w kolejnym odczytywana z tego samego obwodu. Sprzężenie to często służy do tworzenia zatrzasku (samopodtrzymania), którego schemat przedstawiono na rysunku 38. Samopodtrzymanie może zostać zrealizowanie również za pomocą pary cewek S (set) i R (reset) skojarzonych z jedną zmienną. Wtedy cewka S ustawia zmienną, cewka R kasuje ją. Obie cewki posiadają ten sam adres fizyczny.

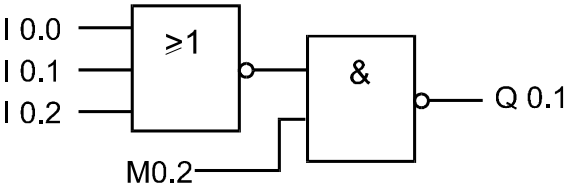


Rysunek 38. Przykład zatrzasku – zastosowania sprzężenia zwrotnego

Ze względu na graficzny typ języka drabinkowego łatwo stracić czytelność i jednoznaczność programu, dlatego też zaleca się tworzenie prostych, przejrzystych schematów, co poprawia łatwość zrozumienia programu oraz usuwa wieloznaczności. W celu zwiększenia czytelności programu zaleca się wyraźne rozdzielenie części obliczeniowej od części odpowiedzialnej za zapamiętywanie.

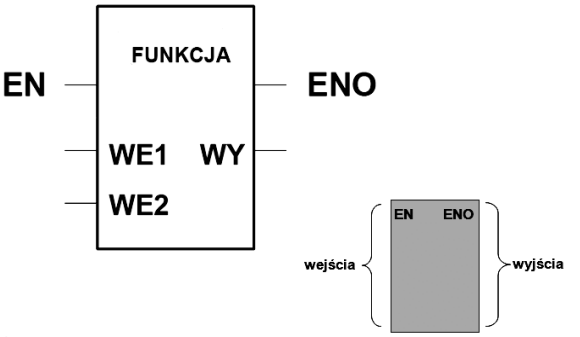
5.2.2. Funkcjonalne schematy blokowe – FBD

FBD należy do grupy języków graficznych, budowa programu w FBD jest zbliżona do budowy diagramu w SIMULINK-u, program wzorowany jest na schematach blokowych układów scalonych (Masi, 2008). Każda instrukcja języka jest reprezentowana przez *blok*. Język ten podobnie jak język schematów drabinkowych jest odpowiednikiem schematu przepływu sygnału dla obwodów logicznych, ale przedstawionych w formie połączonych bramek logicznych oraz funkcji i bloków funkcjonalnych. Interpretacja programu jest identyczna, jak interpretacja programu w LD. Realizacja programu w języku FBD opiera sie na przepływie sygnału (analogicznie do przepływu sygnału pomiędzy elementami systemu przetwarzania sygnału) z wyjścia (z prawej strony) funkcji lub bloku funkcyjnego do przyłączonego wejścia (z lewej strony) następnej funkcji lub bloku funkcyjnego. Fragment programu realizowanego w języku FBD przedstawia rysunek 39. Istnieje możliwość sterowania i monitorowania wykonania bloku z użyciem wejść EN i wyjść ENO, których interpretacja jest taka sama, jak w LD; program może być również podzielony na obwody.



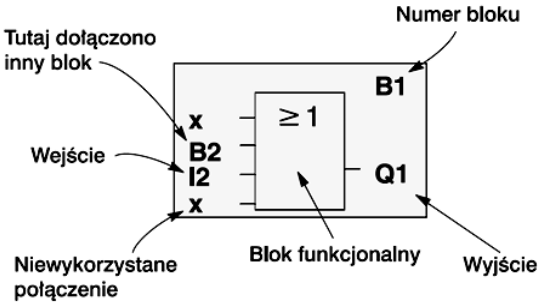
Rysunek 39. Przykładowa aplikacja zrealizowana w języku FBD

Podstawowym elementem programu jest blok (rys. 40). Blok posiada listwę wejść umieszczoną z lewej strony oraz listwę zawierającą wyjścia po prawej stronie. Na wejście mogą trafiać sygnały z wejść sterownika, innych bloków funkcyjnych lub elementów stanowiących sprzężenia zwrotne (rys. 41). Każdy blok stanowi *instancję* określoną przez nazwę lub numer bloku (Masi, 2008).



EN (*ang. Enable*)
ENO (*ang. Enable Output*)

Rysunek 40. Blok jako podstawowa jednostka języka FBD



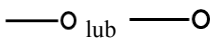
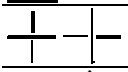
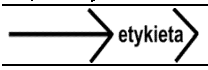
Rysunek 41. Blok jako podstawowa jednostka języka FBD

Wejścia bloków mogą mieć następujące oznaczenia i właściwości:

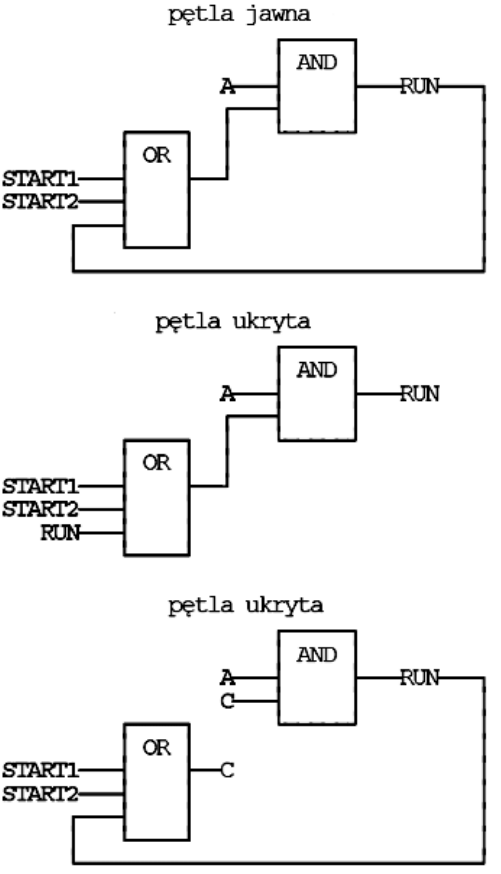
- S (set – ustaw) – wejście S służy do nadania wyjściu stanu;
- R (reset – wyzeruj) – wejście zerujące R ma charakter nadrzędny w stosunku do wszystkich pozostałych wejść i powoduje wyzerowanie wyjścia;
- Trg (trigger – uruchom) – wejście to służy do uruchomienia bloku;
- Cnt (count – zliczaj) – wejście służy do zliczania impulsów;
- Fre (frequency) – wejście to służy do pomiaru częstotliwości sygnału;
- Dir (direction) – wejście służy np. do określania kierunku zliczania;
- En (enable – włącz) – sygnał na tym wejściu uaktywnia funkcję bloku (jeśli wejście to ma stan 0, blok ignoruje wszelkie inne sygnały);
- Inv (invert) – sygnał na tym wejściu powoduje negowanie stanu sygnału wyjściowego;
- Ral (reset all – wyzeruj wszystko) – zeruje wszystkie wartości robocze bloku.

W tabeli 6 przedstawiono podstawowe elementy sterujące wykonywaniem programu.

Tabela 6. Elementy języka FBD sterujące wykonywaniem programu

Symbol	Znaczenie
	przepływ sygnału
	połączenie przepływów
 lub	negacja wejścia lub wyjścia
	narożnik przepływu
	skrzyżowanie przepływów bez połączenia
	konektor
1 ——>>DALEJ	skok bezwarunkowy do „Dalej”
X ——>>DALEJ	skok warunkowy do „Dalej” (gdy x = 1)
X ——<RETURN>	powrót warunkowy z funkcji lub bloku

Innym rozwiązaniem sterującym kolejnością wykonywania bloków funkcyjnych są pętle (rys. 42). Oprócz tego, że sterują wykonywaniem programu, umożliwiają ukrycie połączeń, co czyni schematy czytelniejszymi. W przykładzie ciężko jest zinterpretować kolejność działań, wyznaczona zostanie po numerach bloków. W sytuacji kiedy wprowadzimy pętlę ukrytą, jak w przykładzie drugim – najpierw będzie wykonana funkcja OR z racji tej, że jest na lewo od AND. Wynik zostanie zapisany w zmiennej RUN. Trzeci przykład obrazuje pętlę, której położenie wyznacza kolejność pierwszego działania, jako AND, a wynik zostanie zapisany w zmiennej C.



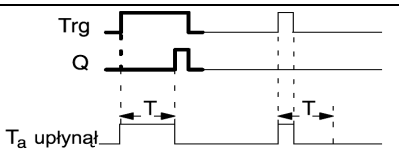
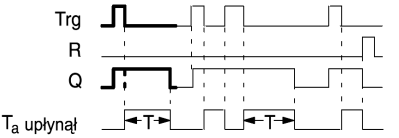
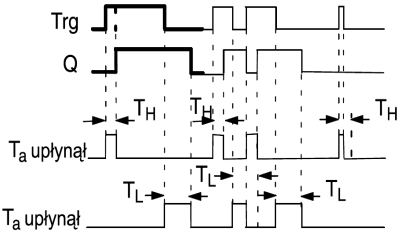
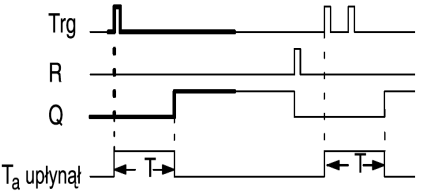
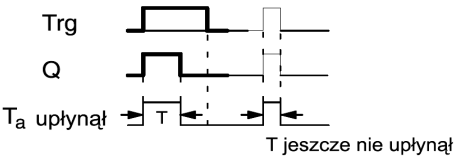
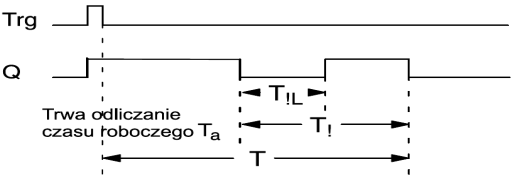
Rysunek 42. Sprzężenia zwrotne: pętla jawna i ukryta

Do bloków podstawowych należą:
- funkcje logiczne AND, OR i pochodne;
- przerzutnik bistabilny;
- licznik góra-dół;
- przekaźnik czasowy;

– komparator wielkości analogowych;
– moduły arytmetyczne itp.

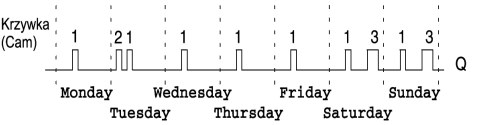
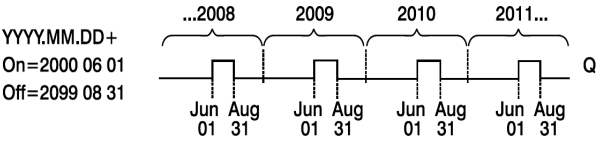
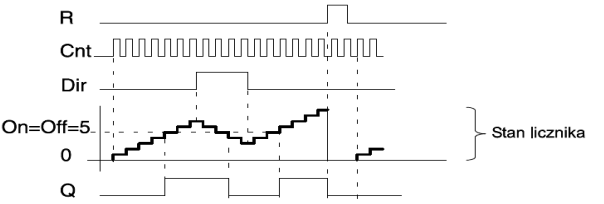
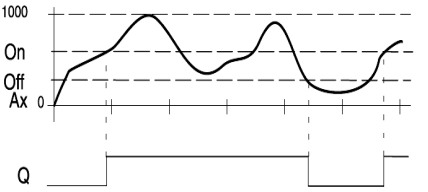
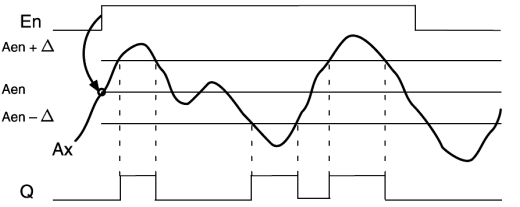
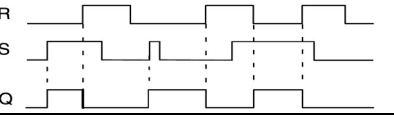
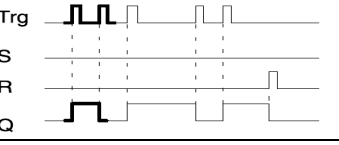
W tabeli 7 i 8 zamieszczono przykładowe bloki funkcyjne z graficznym opisem ich działania.

Tabela 7. Elementy języka FBD sterujące wykonywaniem programu

Nazwa bloku	Sposób działania
Opóźnione załączanie	
Opóźnione wyłączanie	
Opóźnione zał/wył	
Opóźnione załączanie z podtrzymaniem	
Przekaźnik czasowy z wyj. impulsowym	
Wyłącznik oświetlenia	

Źródło: Nowakowski, 2006

Tabela 8. Elementy języka FBD sterujące wykonywaniem programu

Nazwa bloku	Sposób działania
Timer tygodniowy	
Timer roczny	
Licznik góra/dół	
Komparator wielkości analogowych	
Analogowy watchdog	
Przekaźnik zatrzaskowy	
Przekaźnik impulsowy (bistabilny)	

Źródło: Nowakowski, 2006

5.2.3. Lista rozkazów (instrukcji) – IL

Pierwszym z omawianych języków tekstowych jest *lista rozkazów* (instrukcji) zbliżona budową składni do języków typu *asembler*. Jest językiem symbolicznym niskopoziomowym – stanowi przedostatni poziom przed językiem maszynowym. Symbolika ułatwia programowanie, zapamiętanie; np. podstawowe polecenie Load (LD) jest łatwiejsze niż odpowiadającemu mu kodowi operacji maszynowej wraz z trybami adresowania i argumentami. Składa się z sekwencji rozkazów. Mnemoniki (krótkie rozkazy wykonywane przez mikroprocesory) odpowiadają kodowi maszynowemu. Postrzegany jest jako język pośredni, przed kompilacją programu z innych języków programowania na postać maszynową, ponieważ jest najbardziej podobny do języka maszynowego.

Każdy rozkaz powinien rozpoczynać się w nowej linii, między rozkazami można wprowadzać puste linie. Rozkaz powinien zawierać nazwę operatora z możliwymi modyfikatorami oraz operand – jeden lub więcej, oddzielone przecinkami. Operandami mogą być stałe lub zmienne.

Przewagą języka IL nad pozostałymi językami programowania jest szybkość wykonywania, jak również duża szybkość wprowadzania kodu. Program napisany w języku IL składa się z wielu linii kodu, z których każda reprezentuje dokładnie jedną operację, co sprawia, że wprowadzanie szeregu funkcji matematycznych jest bardzo łatwe. Jednakże IL jest językiem mniej czytelnym, np. w stosunku do języka drabinkowego. Podstawową wadą tego języka jest brak diagnostyki błędów.

W języku IL nie używa się znaku (;) tak, jak to ma miejsce w języku ST do zakończenia instrukcji czy zakończenia deklaracji pojedynczej zmiennej.

Operatory służą do określenia działania, jakie ma być wykonane. Wymagane jest, by operator i operand były rozdzielone co najmniej jednym znakiem spacji, żeby można było je rozróżnić. Podstawą działania tego języka jest *akumulator* stanowiący rejestr w pamięci; wykorzystywany jest do:

- wczytywania do niego wartości z komórek pamięci PLC;
- wykonywania operacji matematycznych;
- przechowywania tymczasowych wyników;
- kopiowania stanu akumulatora do wybranych komórek pamięci.

W celu zwiększenia czytelności pisanego programu można odpowiednio formatować tekst, używając np. odpowiednich wcięć lub pustych linii oddzielających fragmenty programu. Tabela 9 prezentuje przykład sekwencji rozkazów: w pierwszym wierszu ładowany jest do akumulatora stan wejścia fizycznego I1 (X oznacza, że jest to wartość Boolowska), następnie realizowana jest funkcja logiczna AND z zanegowaną pamięcią M5, ostatnim krokiem jest zapisanie stanu akumulatora do wyjścia fizycznego Q2.

Rozkaz może być poprzedzony etykietą zakończoną dwukropkiem. Etykiety służą do oznaczenia rozkazów, są wykorzystywane do zaznaczenia miejsca docelowego dla rozkazów skoku z dowolnego miejsca programu w czasie jego wykonywania. Komentarz powinien być ostatnim elementem linii, stanowi nieformalny opis zawartości linii programu oraz ograniczony jest kombinacjami znaków.

Jak już wcześniej wspomniano, język IL to język zorientowany akumulatorowo. Klasyczne asemblery oparte są przede wszystkim na działaniach na akumulatorze procesora, do którego wprowadzana jest wartość, a następnie dodawane, odejmowane itp. są inne

wartości, natomiast wynik przepisywany jest z akumulatora do pamięci. W języku IL akumulator nosi nazwę *wynik bieżący* CR *(ang. Current Result)*, lecz nie ma określonej liczby bitów, ponieważ nie jest akumulatorem sprzętowym. Kompilatory języka IL używają wirtualnego akumulatora o dowolnej długości. Liczba bitów w uniwersalnym akumulatorze zależna jest od typu danych, do którego należy przetwarzany operand, przy czym jednocześnie typ danej w akumulatorze dopasowuje się do typu ostatnio przetwarzanego operandu.

Tabela 9. Przykład sekwencji rozkazów

Etykieta	Operator	Operand	Komentarz
START :	*LD*	*%IX1*	*(* WCIŚNIJ PRZYCISK*)*
	ANDN	*%MX5*	*(*NIE WSTRZYMANY*)*
	ST	*%QX2*	*(*ZAŁĄCZ*)*

W klasycznych asemblerach występuje słowo stanu procesora *(ang. Processor Status Word – PSW)* zawierające bity, które określają stan procesora wynikający z wykonania rozkazu. W języku IL wynik operacji staje się wynikiem bieżącym. Dlatego też skoki warunkowe w programie lub też wywołania bloków funkcjonalnych (FB) odwołują się do wartości boolowskiej CR, a nie do bitów PSW. Wynik bieżący może należeć do elementarnego typu danych, pochodnego typu danych lub też może to być typ FB. Konieczna jest natomiast kompatybilność kolejnych operatorów, czyli do następnego typu rozkazu musi być przypisany odpowiedni typ danych CR. Tabela 10 przedstawia listę standardowych operatorów oraz możliwych modyfikatorów.

Tabela 10. Lista operatorów, modyfikatorów i operandów języka IL

Operator	Modyfikatory	Operand	Opis
LD	N	*	Ustawia bieżący wynik równy argumentowi, CR przyjmuje wartość operandu (LoaD)
ST	N	*	Zapisuje bieżący wynik w miejscu wstawienia argumentu, przesłanie CR do operandu (STore)
S		BOOL	Ustawia argument boolean na TRUE, jeśli bieżący wynik jest TRUE, jeśli CR = 1, to operand ustaw na 1 (Set)
R		BOOL	Ustawia argument boolowski na FALSE, jeśli bieżący wynik jest FALSE, jeśli CR = 1, to zeruj operand (Reset)
AND	N, (, N(	BOOL	Boolowskie AND operandu i CR
OR	N, (, N(	BOOL	Boolowskie OR operandu i CR
XOR	N, (, N(	BOOL	Boolowskie (eXclusive OR) operandu i CR
ADD	(	*	Dodawanie (ADDition) operandu i CR
SUB	(	*	Odejmowanie (SUBtraction) operandu i CR
MUL	(	*	Mnożenie (MULtiplication) operandu i CR
DIV	(	*	Dzielenie (DIVision) CR przez operand
GT	(	*	Porównanie: CR > operand (Greater Than)

Operator	Modyfikatory	Operand	Opis
GE	(	*	Porównanie: CR >= operand (Greater than or Equal)
EQ	(	*	Porównanie: CR = operand (Equal)
NE	(	*	Porównanie: CR <> operand (Not Equal)
LE	(	*	Porównanie: CR <= operand (Less than Or Equal)
LT	(	*	Porównanie: CR < operand (Less Than)
JMP	C, CN	LABEL	Skok do etykiety (JuMP to label)
CAL	C, CN	NAME	Wywołanie (CaLl) FB o nazwie NAME
RET	C, CN		Powrót (RETurn) z wywołanej funkcji lub FB
)			Operator ograniczający dla modyfikatora (

Przykład programu IL wykorzystującego kilka modyfikatorów:
```
LD TRUE              (* ładuje TRUE do akumulatora *)
ANDN BOOL1   (* wykonuje AND z zanegowaną wartością zmiennej BOOL1 *)
JMPC znacznik   (* jeśli wynik był TRUE, przeskakuje do znacznika „znacznik” *)

LDN BOOL2            (* zachowuje zanegowaną wartość dla *)
ST ERG               (* zapisuje BOOL2 w ERG *)
Znacznik:
LD BOOL2 (* zachowuje wartość dla *)
ST ERG (* BOOL2 w ERG *)
```

W języku IL można również wstawiać nawiasy po operacji. Jako argument będzie traktowana wtedy wartość nawiasów.
Przykład:
```
LD 2
MUL 2
ADD 3
Erg
```
W tym przykładzie wartość Erg wynosi 7. Jeśli jednak zostaną wstawione nawiasy:
```
LD 2
MUL (2
ADD 3
)
ST Erg
```
wartość Erg wynosi już 10, gdyż operacja MUL zostanie wykonana dopiero wtedy, gdy osiągnięty zostanie nawias zamykający „)”, z obliczenia argumentem dla MUL będzie wtedy 5.

Wywoływanie funkcji i bloków funkcjonalnych

Wywołanie funkcji realizowane jest przez umieszczenie jej nazwy w polu operatora. Pierwszym parametrem wejściowym funkcji jest wynik bieżący CR, natomiast w przypadku, gdy wymagana jest większa liczba parametrów, umieszcza się je w polu operandu

i oddziela przecinkami. Po domyślnym wykonaniu rozkazu RET lub też po osiągnięciu fizycznego końca funkcji, wartość przez nią zwracana staje się wynikiem bieżącym CR, dla którego typ danych jest określony przez deklarację funkcji. Przykład deklaracji funkcji w języku IL:

```
(*deklaracja ziemnych*)
FUNCTION NowaF : INT;   (*deklaracja funkcji NowaF*, typ danych: INT)
    VAR_INPUT
    We1, We2, We3 : INT;   (*parametry wejściowe*)
END_VAR
(*ciało funkcji*)
    LD We1
    ADD We2
    ADD We3
    ST NowaF (*wartość funkcji na wyjściu*)
END_FUNCTION
```

Wywołanie tej funkcji powinno mieć następujacą składnię:

```
(*zapis POU wywołującego funkcję NowaF*)
    VAR
    Par1, Par2, Par3, Wynik : INT; (*deklaracja zmiennych*)
    END_VAR

    LD          Par1          (*wprowadzenie pierwszego parametru wejściowego*)
    NowaF       Par2, Par3    (*wywołanie funkcji z pozostałymi wartościami*)
    ST          Wynik         (*zapisanie wartości NowaF w zmiennej Wynik*)
```

Bloki funkcjonalne mogą być wywoływane bezwarunkowo za pomocą operatora CAL lub warunkowo przy użyciu modyfikatora C (wywołanie, jeśli wynik bieżący CR jest TRUE) bądź też modyfikatorów CN (wywołanie, jeśli wynik bieżący jest FALSE). Trzy przykłady wywołania bloku funkcjonalnego zamieszczono w tabeli 11.

Tabela 11. Wywołanie bloków funkcjonalnych w języku listy instrukcji

Lp.	Przykładowy program
1	*CAL* z listą wejść: *CAL C10(CU :=%IX10, PV := 15*
2	Przesyłanie wejść za pomocą operatorów LD i ST: *LD 15* *ST C10,PV* *LD %IX10* *ST C10.CU* *CAL C10*
3	Użycie operatorów wejściowych *LD 15* *PV C10* *LD %IX10* *CU C10* *CAL C10*

W pierwszym przykładzie przedstawione jest wywołanie bloku standardowego licznika, oznaczonego jako C10 z wypisaną listą aktualnych parametrów wejściowych, przyporządkowanych parametrom formalnym. Polecenie wskazuje wejście, z którego będą zliczane sygnały logiczne, PV stanowi wartość graniczną zliczania. Drugi przykład opisuje wprowadzenie i zapamiętywanie wartości wejść w strukturze danych egzemplarza FB, który jest wywoływany. W trzecim przykładzie użyte są operatory wejściowe dla FB standardowych. W przedstawionych przykładach wywołanie licznika CTU odbywa się z niepełną listą wejściową – brak wejścia zerującego R.

Pozostałe elementy języka IL

IL wykorzystuje elementy wspólne dla innych języków programowania, szczególnie takie, które dotyczą deklaracji typów danych i zmiennych, deklaracji jednostek organizacyjnych oprogramowania oraz tworzące strukturę programu w postaci sekwencyjnego schematu funkcjonalnego, m.in.:

- VAR … END_VAR;
- VAR_INPUT … END_VAR;
- VAR_OUTPUT … END_VAR;
- VAR_IN_OUT … END_VAR;
- VAR_EXTERNAL … END_VAR;
- TYPE … END_TYPE;
- STEP … END_STEP;
- ACTION … END_ACTION;
- TRANSITION … END_TRANSITION;
- FUNCTION … END_FUNCTION;
- FUNCTION_BLOCK … END_FUNCTION_BLOCK;
- PROGRAM … END_PROGRAM.

5.2.4. Tekst strukturalny – ST

Język ten jest odpowiednikiem języka algorytmicznego *wysokiego poziomu*, zawierającego struktury programowe i polecenia podobne do występujących w językach typu PASCAL lub C które, jak w językach wyższego rzędu, mogą być wykonywane warunkowo:
If...then...else...end_if
For...to...do...end_for
While...do...end_while

Podstawowymi elementami tego języka są *wyrażenia* i *instrukcje*. Wyrażenia składają się z operatorów i argumentów. Argument może być stałą, zmienną, wywołaniem funkcji lub innym wyrażeniem. Wyrażenie jest konstrukcją, która po wykonaniu zwraca obliczoną wartość. Obliczanie wyrażenia odbywa się poprzez działania na operatorach wg określonych *zasad ważności*. Operator z największym priorytetem jest przetwarzany w pierwszej kolejności, dalej operator z kolejnym co do ważności priorytetem itd., aż wszystkie operatory zostaną przetworzone. Instrukcje oddziela się średnikami, w jednej linii programu może się znajdować więcej niż jedna instrukcja. Jednak zaleca się dla większej czytelności

kodu rozdzielanie instrukcji w poszczególnych wierszach, nie ma to wpływu na szybkość wykonywania kodu.

Przy przetwarzaniu operatorów z takim samym priorytetem obowiązuje kolejność z lewej do prawej. W tabeli 12 przedstawiono wykaz operatorów języka ST uporządkowanych wg priorytetu.

Tabela 12. Operatory języka ST z wyznaczeniem priorytetu

Operacja	Symbol	Priorytet
ujęcie w nawiasy	(wyrażenie w nawiasach)	najwyższy priorytet
wywołanie funkcji	nazwa funkcji (lista parametrów, obliczanie)	
potęgowanie	EXPT	
negowanie arytmetyczne	–	
negowanie Boolowskie	NOT	
iloczyn	*	
iloraz	/	
modulo	MOD	
dodawanie	+	
odejmowanie	–	
porównanie	<,>,<=,>=	
znak równości	=	
znak większości lub mniejszości	> <	
iloczyn logiczny AND	AND	
suma modulo 2 XOR	XOR	
suma logiczna OR	OR	najniższy priorytet

Język tego typu może być używany do obliczania złożonych wyrażeń, zawierających wielkości analogowe i binarne.

Przykład:

IF value < 7 THEN
WHILE value < 8 DO
value := value + 1;
END_WHILE;
END_IF;

Oprócz wyrażeń i instrukcji używane są typowe elementy wspólne dla wszystkich języków programowania, dotyczące deklaracji zmiennych, typów danych, jednostek organizacyjnych oprogramowania.

Element oprogramowania napisany w ST zawsze musi być kompilowany, na PLC jest ładowany wyłącznie zbiór wynikowy (nie jest ładowany plik źródłowy).

Podstawowe grupy instrukcji:

Instrukcje przypisania zastępują wartość bieżącą zmiennej, są stosowane do nadawania wartości funkcji w ciele funkcji przez umieszczenie nazwy funkcji po lewej stronie operatora „:=”. Używając instrukcji przypisania, należy zwrócić uwagę na zgodność typów danych.

Wyrażenie jest budowane z operatorów i/lub wywołań funkcji lub instancji FB. Kolejność wykonywania obliczeń jest zgodna z priorytetami operatorów (najwyższy – nawiasy, najniższy – OR logiczny). Wartość wyrażenia jest obliczana tylko do momentu, gdy można jednoznacznie określić wynik.

```
                {stała}
<zmienna> :  =  {zmienna} ;
                {wyrażenie}
```

Instrukcje warunkowe IF pozwalają na wykonanie jednej lub grupy instrukcji w zależności od spełnienia określonego warunku logicznego. Jeśli {warunek logiczny} daje w wyniku TRUE, wówczas wykonywane będą tylko <instrukcje, jeśli prawda…> i żadne z pozostałych instrukcji. Jeśli warunek nie jest spełniony, wykonywane będą po kolei wyrażenia (może ich być więcej) umieszczone po ELSIF tak długo, dopóki któreś z nich da wynik TRUE. Następnie wykonywane będą tylko instrukcje zawarte między tym wyrażeniem (spełniającym warunek), a następną instrukcją ELSE lub ELSIF. Jeśli żadne z wyrażeń nie daje wyniku TRUE, wówczas będą wykonywane wyłącznie <instrukcje, jeśli nieprawda>.

```
IF {warunek logiczny} THEN
        < instrukcje jeśli prawda...> ;
        ELSIF { warunek logiczny }
        < instrukcje, jeśli spełniony warunek...>;
        ELSE
        <instrukcje, jeśli nieprawda...>;
END_IF;
```

Przykład:

```
IF temp<20
        THEN ogrzew_wl := TRUE;
        ELSE ogrzew_wl := FALSE;
END_IF;
```

W tym przykładzie program włącza ogrzewanie, gdy temperatura spadnie poniżej 20 stopni, w innym razie ogrzewanie jest wyłączone.

Instrukcja wyboru CASE zawiera wyrażenie typu INT stanowiące wybierak (selektor) oraz listę bloków instrukcji, z których każdy ma swoją etykietę (jedna lub więcej liczb całkowitych INT). Wykonana będzie ta instrukcja lub blok instrukcji, która ma etykietę równą wartości wybieraka. Jeżeli wartość wybieraka nie odpowiada żadnemu z zakresów, to będzie wykonana sekwencja instrukcji umieszczonych po słowie kluczowym ELSE; w sytuacji kiedy nie zdefiniowano tam instrukcji, nie będzie wykonana żadna instrukcja.

```
CASE {wyrażenie lub zmienna typu: INT lub wyliczeniowa}
        <wartość (jedna, kilka lub zakres)>: <instrukcje...>;
        <wartość (jedna, kilka lub zakres)>: <instrukcje...>;
        .....
        ELSE
        <instrukcje...>
END_CASE;
```

Przykład:

```
CASE INT1 OF
        1, 5: BOOL1 := TRUE;
        BOOL3 := FALSE;
        2: BOOL2 := FALSE;
        BOOL3 := TRUE;
        10..20: BOOL1 := TRUE;
        BOOL3:= TRUE;
ELSE
        BOOL1 := NOT BOOL1;
        BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

Instrukcje iteracji realizują powtarzanie pewnych sekwencji instrukcji w pętli. Jeżeli liczba powtórzeń jest znana, wykorzystuje się pętlę *FOR*, w przeciwnym wypadku należy skorzystać z pętli *WHILE* lub *REPEAT*. Wprowadzenie instrukcji *EXIT* umożliwia wcześniejsze opuszczenie pętli, zanim zostanie spełniony warunek pętli. Jeżeli instrukcja znajduje się w pętlach zagnieżdżonych, wtedy wyjście następuje tylko z tej pętli, w której instrukcja EXIT się znajduje.

Za pomocą pętli FOR można zaprogramować powtarzanie przebiegów.

```
INT_Var :INT;
FOR <INT_Var> := <INIT_WART>
        TO <END_WART>
        {BY <wielkość kroku>}
        DO
        <instrukcje>
END_FOR;
```

Element w nawiasach klamrowych {} jest opcjonalny. <Instrukcje> są wykonywane tak długo, jak długo licznik <INT_Var> nie będzie większy niż <END_WART>. Warunek ten jest sprawdzany przed wykonaniem się <instrukcji>; w ten sposób <instrukcje> nigdy nie są wykonywane, jeśli <INIT_WART> jest większa niż <END_WART>. Zawsze, gdy <instrukcje> zostaną wykonane, wartość <INT_Var> zwiększa się o <wielkość kroku>. Wielkość kroku może przyjmować każdą wartość typu integer. Jeśli nie została ona podana, wtedy wynosi ona domyślnie 1. Pętla musi określać warunek zakończenia dla powtarzania, gdyż <INT_Var> za każdym razem zwiększa się. Wartość <END_WART> nie może być równa wartości granicznej licznika <INT_VAR>. Jeśli np. zmienna Licznik jest typu SINT, wartość <END_WART> nie może być 127, gdyż oznacza to pętlę nieskończoną.

Przykład:

```
FOR licznik:=1 TO 5 BY 1 DO
        Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Przyjmując, że zmienna Var1 ma wartość początkową 1, wtedy po wykonaniu pętli FOR będzie miała wartość 32.

Pętli WHILE można używać tak, jak pętli FOR z tą różnicą, że warunkiem kontrolującym jej zakończenie może być dowolne wyrażenie boolean. Oznacza to, że jeśli podany warunek jest spełniony, wówczas pętla zostanie wykonana.

WHILE <wyrażenie boolean>
DO
<instrukcje>
END_WHILE;

<Instrukcje> są powtarzane tak długo w pętli, jak długo <wyrażenie boolean> daje wynik TRUE. Jeśli <wyrażenie boolean> już przy pierwszym wykonaniu daje wynik FALSE, wówczas <instrukcje> nie zostaną ani razu wykonane. Jeśli <wyrażenie_boolean> nigdy nie przyjmuje wartości FALSE, wówczas <instrukcje> są wykonywane bez końca, przez co powstaje błąd czasu przebiegu.

Przykład:

```
WHILE licznik<>0 DO
        Var1 := Var1*2;
        licznik := licznik-1;
END_WHILE
```

Pętla REPEAT odróżnia się od pętli WHILE tym, że warunek zakończenia wykonywania pętli sprawdzany jest dopiero po jej wykonaniu. Skutek tego jest taki, że pętla musi zostać co najmniej jeden raz wykonana niezależnie od warunku jej zakończenia.

REPEAT
<instrukcje>
UNTIL <wyrażenie boolean>
END_REPEAT;

<Instrukcje> są wykonywane tak długo, jak długo <wyrażenie boolean> daje wynik TRUE. Jeśli <wyrażenie boolean> już przy pierwszym wykonaniu daje wynik TRUE, wówczas <instrukcje> zostaną wykonane tylko jeden raz. Jeśli <wyrażenie_boolean> nigdy nie przyjmuje wartości TRUE, wówczas <instrukcje> są wykonywane bez końca, przez co powstaje błąd czasu przebiegu.

Przykład:

```
REPEAT
        Var1 := Var1*2;
        Licznik := licznik-1;
UNTIL
        licznik=0
END_REPEAT
```

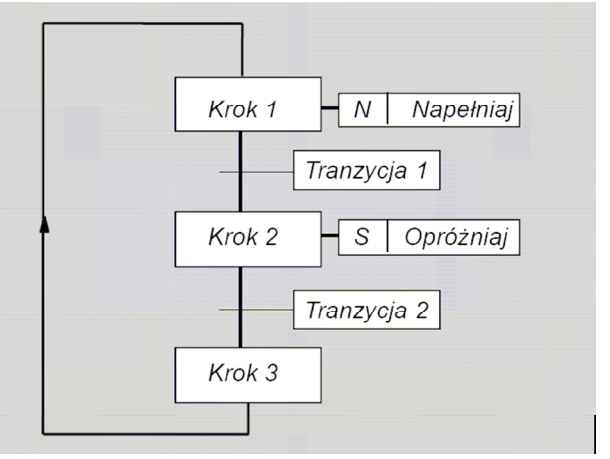
Pętle WHILE i REPEAT mają w pewnym sensie tę przewagę nad pętlą FOR, że przed wykonaniem pętli nie trzeba znać liczby jej przebiegów. Najczęściej w praktyce wykorzystywane są tylko te dwa rodzaje pętli. Jeśli jednak znana jest liczba przebiegów pętli, należy korzystać z pętli FOR, gdyż ta nie powoduje powstawania pętli nieskończonych.

Podsumowując język ST należy zauważyć, że jest on przeznaczony do realizacji złożonych procedur obliczeniowych, wymagających stosowania pętli i instrukcji wyboru, które są trudne do realizacji w innych językach (Kwaśniewski, 2008).

5.2.5. Sekwencyjny schemat funkcjonalny – SFC

Język przebiegu sekwencyjnego schematu funkcjonalnego jest jednym z języków zorientowanych graficznie, opisujących czasowy przebieg różnych czynności wewnątrz programu. W tym celu język wykorzystuje *grafy* zawierające *kroki*, dla których przypisane są określone *akcje* i *tranzycje*, które sterują przebiegiem kroków. Jest to graficzne narzędzie programowe, bazujące na metodologii sieci Petriego typu P/T, umożliwiające realizację układów *sterowania sekwencyjnego* (tj. takich, w których da się wyróżnić poszczególne etapy i wykonywane w nich działania oraz zdefiniować warunki przejścia od jednego etapu do następnego). Jest on też przydatny do realizacji algorytmów zawierających dużo skoków lub alternatywnych ścieżek realizacji. Z użyciem SFC są budowane wyłącznie *bloki funkcyjne*. Podczas budowy projektu z użyciem SFC są używane inne języki programowania. Projekt zbudowany z użyciem SFC musi być kompilowany przed załadowaniem na CPU.

Sekwencyjny schemat funkcjonalny umożliwia podział jednostki POU na mniejsze elementy składowe, między którymi następuje przepływ sygnałów sterujących. Napisany w języku przebiegu moduł, składa się z ciągu kroków połączonych ze sobą za pomocą ukierunkowanych warunków (tranzycji) (rys. 43). Tworzą one sieć (network), która stanowi podstawową strukturę w języku SFC. Jednostka POU może zawierać jedną lub więcej sieci. Z każdym krokiem jest skojarzony odpowiedni zbiór instrukcji, które nazywa się *akcjami*, a każdemu przejściu między krokami towarzyszy warunek przejścia – *tranzycja*.



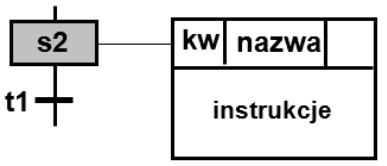
Rysunek 43. Przykład programu sterującego poziomem cieczy zrealizowany w języku SFC

Istnieją dwa rodzaje kroków. Najprostsza konstrukcja składa się z jednej akcji i jednej flagi, która pokazuje, czy krok jest *aktywny*. Flaga jest reprezentowana przez zmienną boolowską, przyjmuje wartość 1, gdy jest *aktywny* i 0 – *nieaktywny*. Jeśli dla danego kroku została zaimplementowana akcja, wówczas w prawym górnym rogu prostokąta przedstawiającego krok pojawi się mały trójkąt. Powiązane akcje pojawiają się z prawej strony kroku. Ponieważ określa on, w którym miejscu realizacji program się aktualnie znajduje,

powinien mieć unikalną nazwę skojarzeniową. Oprócz zwykłego kroku wyróżniamy krok początkowy, który jest rysowany podwójną linią, a w przypadku reprezentacji tekstowej, jako INITIAL_STEP (*ciało kroku*) END_STEP.

W kroku umieszczona jest *akcja* stanowiąca ciało kroku, może zawierać ciąg instrukcji w IL lub w ST, dowolną ilość sieci w FBD lub w LD, albo znowu strukturę przebiegu (SFC). W krokach uproszczonych akcja jest powiązana zawsze z należącym do niej krokiem. Aby edytować akcję, najczęściej należy kliknąć dwukrotnie krok, który do niej należy lub zaznaczyć krok i wybrać polecenie menu. Ponadto dla każdego kroku można przypisać jedną akcję wejściową i/lub wyjściową.

Dla akcji zawartej w danym kroku można dodatkowo utworzyć akcję wejściową i akcję wyjściową. Akcja wejściowa wykonywana jest tylko jeden raz, bezpośrednio po aktywowaniu kroku. Akcja wyjściowa wykonywana jest również tylko jeden raz, zanim krok zostanie dezaktywowany. Akcja wejściowa i wyjściowa może być implementowana w dowolnym języku. Każda akcja zawiera *kwalifikator działania* (kw) określa on korelację czasową pomiędzy czasem aktywności etapu i czasem aktywności działania. W najprostszym przypadku działanie jest wykonywane przez cały czas aktywności etapu (rys. 44). W tabeli 13 zestawiono najczęściej występujące kwalifikatory działania akcji.



Rysunek 44. Powiązanie kroku z akcją

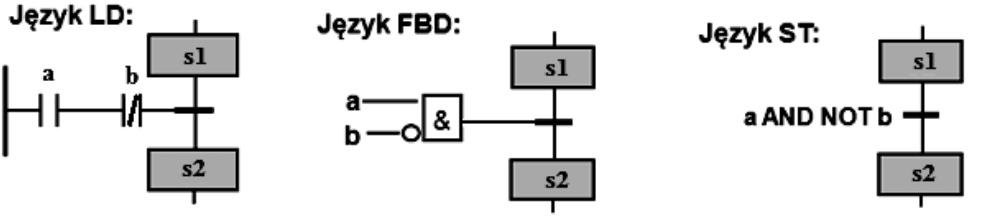
Tabela 13. Rodzaje kwalifikatorów działania

Kwalifikator	Nazwa	Opis
N	Non-stored	akcja jest tak długo aktywna, jak długo aktywny jest krok
R	overriding Reset	akcja dezaktywowana
S	Set (Stored)	akcja jest aktywowana i pozostaje aktywna do następnego resetu
L*	time Limited	akcja jest aktywna w określonym czasie, jednak maksymalnie dopóki aktywny jest krok
D*	time Delayed	akcja jest aktywna po upływie określonego czasu, o ile krok jest nadal jeszcze aktywny i dalej dopóki krok jest aktywny
P	Pulse	akcja jest wykonywana dokładnie jeden raz, jeśli krok jest aktywny
SD*	Stored and time Delayed	akcja jest aktywowana po upływie określonego czasu i pozostaje aktywna do następnego resetu
DS.*	Delayed and Stored	akcja jest aktywowana po upływie określonego czasu, o ile krok jest jeszcze aktywny i pozostaje aktywna do następnego resetu
SL*	Stored and time Limited	akcja jest aktywna w określonym czasie

* kwalifikatory L, D, SD, DS i SL wymagają podania czasu w formacie stałych czasowych TIME, np. L T#5s

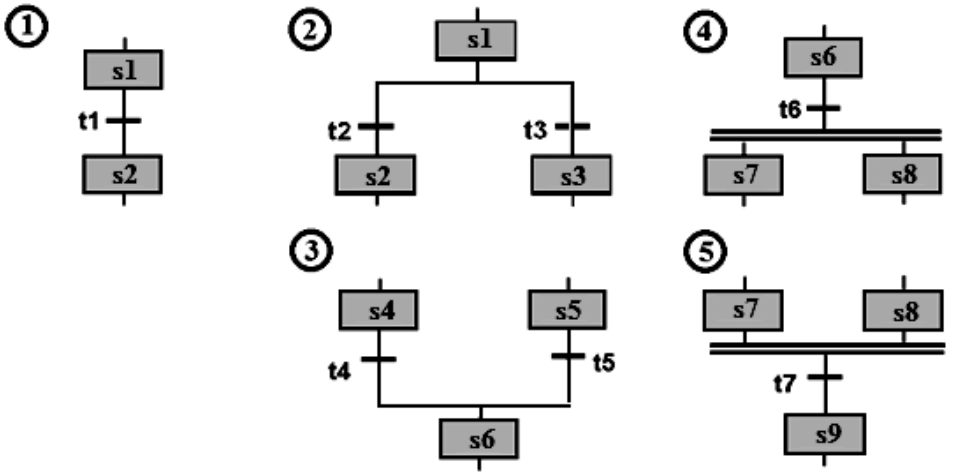
Źródło: Mikulczyński i Samsonowicz, 1997

Między krokami znajdują się *tranzycje*. Warunek tranzycji musi przyjmować wartość TRUE lub FALSE. Zatem może on zawierać zmienną booloean, adres boolean lub stałą boolean. Może on również zawierać ciąg instrukcji dających wynik boolean, napisanych w składni ST (np. (i <= 100) AND b) lub w dowolnym języku (rys. 45). Tranzycja nie może jednak zawierać programów, bloków funkcji lub przyporządkowań.



Rysunek 45. Przykłady tranzycji warunkujących przejście pomiędzy krokami

W języku tym wyróżnia się następujące podstawowe typy sekwencji przejścia (rys. 46):
- pojedyncza (1);
- wyboru (2);
- zakończenie sekwencji wyboru (3);
- współbieżna (4);
- zakończenie sekwencji współbieżnych (5).



Rysunek 46. Podstawowe typy sekwencji

Można też stosować odmiany tych sekwencji, umożliwiające np. wykonanie pętli lub ominięcie pewnej grupy instrukcji.

Podstawowe reguły budowy i wykonania SFC:

– każde 2 kroki muszą być rozdzielone przejściem i każde 2 przejścia musza być rozdzielone krokiem;
– aby nastąpiło przejście do następnego etapu, poprzedni etap musi być aktywny i musi być spełniony warunek logiczny na przejście.

Metoda SFC nadaje się szczególnie do programowania algorytmu sterowania procesów, w których występują pewne sekwencje działań, krok po kroku. Typowym przykładem może być proces wsadowy, na który składają się kolejne fazy: napełnianie, przebieg procesu (mieszanie, reakcja chemiczna), opróżnianie. Dla każdej fazy opracowany jest właściwy algorytm sterowania, a przejścia pomiędzy fazami odbywają się po spełnieniu określonych warunków. W inżynierii produkcji rolno-spożywczej można znaleźć wiele procesów, dla których idealnym rozwiązaniem jest opracowywanie programów sterujących w języku SFC.

6. KOMUNIKACJA W SYSTEMACH STEROWANIA PLC

Postęp w technologii sterowania PLC oraz wykorzystanie w przemyśle rozproszonych systemów sterowania (DCS, Distributed Control Systems) w znaczny sposób wpłynęło na rozwój architektury sterowników PLC. Coraz bardziej złożone procesy produkcyjne zwiększyły stopień skomplikowania układów automatyki, często wymuszając połączenia większej liczby sterowników PLC – tworząc sieć o wzajemnej komunikacji. Analizując nowoczesne systemy sterowania, można wyartykułować argumenty przemawiające za koniecznością rozwoju systemów sieciowych:

- komunikacja pomiędzy modułami sterowników;
- komunikacja z czujnikami pomiarowymi;
- komunikacja z inteligentnymi czujnikami i oprzyrządowaniem pracującymi na pętli prądowej 4…20mA;
- zdalne programowanie i nadzór sterowników i innych urządzeń;
- automatyzacja linii produkcyjnych;
- automatyzacja w systemach przesyłu mediów takich jak: woda, gaz, prąd;
- automatyzacja budynków – systemy „inteligentnych budynków";
- systemy kontroli ruchu;
- szybkie przesyłanie danych na nieduże odległości;
- monitorowanie i rejestrowanie danych pomiarowych;
- komunikacja z systemami SCADA (Werewka, 1997).

Obecnie sporadycznie można spotkać systemy składające się tylko z jednego sterownika. Jednak nawet w przypadku wykorzystania tylko jednego sterownika, musi on posiadać wbudowany port, umożliwiający komunikację z urządzeniami zewnętrznymi tj. z panelami operatorskimi, komputerami, modemami, programatorami itp. Port komunikacyjny najczęściej wykonany jest w standardzie RS-232 lub RS-485, ale można także spotkać łącza Ethernet z protokołem TCP/IP. Komunikacja przez port odbywa się przy użyciu odpowiedniego protokołu, określonego zwykle przez producenta sterownika, np. Modbus – protokół stworzony przez firmę Modicon, SNP dla GE – Intelligent Platforms. Służy do komunikacji z programowalnymi kontrolerami tej firmy, a także innych producentów. Inny przykład to Hostlink firmy Omron (Grzywak, 1994).

Coraz powszechniejsze staje się projektowanie systemów sterowania jako systemów o wejściach i wyjściach rozproszonych, z wykorzystaniem tzw. sieci polowych (Field Network lub Fieldbus). Rozproszenie systemów sterowania obniża koszty okablowania, a ponadto stwarza możliwość lokalnej diagnostyki, zapewniając tym samym obniżenie kosztów wykrywania i usuwania awarii. Jednocześnie dzięki zaawansowanej diagnostyce sieciowej jest zapewniona kontrola statusów wszystkich elementów sieci oraz obwodów

wejściowo-wyjściowych. Możliwe jest również rozproszenie obliczeń przez stosowanie lokalnych procesów w systemie Field Control (Dorobczyński, 1991).

W sieci takiej są wymieniane dane (zmienne), będące elementami listy wszystkich zmiennych systemowych. Zmiennymi tymi mogą być np.:

- wartości zmierzone pochodzące z procesu (dane tworzone przez układy wejścia);
- rozkazy do urządzeń wykonawczych;
- wartości przesyłane do sterowników, będące ich wewnętrznymi zmiennymi.

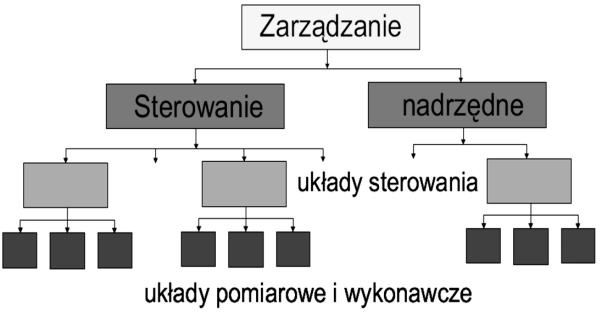
Jednostka danych podlegająca wymianie i przekazywana do sieci przez swego „producenta" jest identyfikowana przez unikalną w systemie nazwę (globalną). „Producent" jest informowany o żądaniu wysłania jednostki danych przez odbiór kodu odpowiadającego żądanym danym. Mechanizm ten nazywa się adresowaniem źródła. Lista wszystkich zmiennych istniejących w systemie jest umieszczona w pamięci arbitra magistrali (kontrolera sieci). Natomiast abonenci sieci mają swoje lokalne listy zmiennych, na których umieszcza się nazwy zmiennych, których abonent jest bądź „producentem", bądź „konsumentem". Listy takie są tworzone na etapie konfiguracji sieci. W sieci krążą transmitowane przez arbitra nazwy zmiennych i ich wartości produkowane przez abonentów. Transmisje w sieci są typu rozgłoszeniowego, trafiają jednocześnie do wszystkich abonentów i każdy może z nich dowolnie korzystać. „Konsumenci" czuwają jedynie nad detekcją błędów transmisji.

Sieci polowe ułatwiają tworzenie rozproszonych systemów sterowania na drodze zamiany konwencjonalnych połączeń typu *point-to-point* na połączenia typu *multi-point*. Jest to rozwiązanie tańsze i prostsze w realizacji, przy zachowaniu takich własności, jak krótki czas odpowiedzi czy duża pewność transmisji. Do jego zalet należy zaliczyć także łatwość rozbudowy i modyfikacji, z możliwością przyłączenia do sieci elementów wykonawczych i zadajników bez względu na ich położenie w obiekcie. Dzięki temu urządzenia te można traktować jako bezpośrednie układy wejścia/wyjścia (urządzenia zdalne), bez pośrednictwa sterowników PLC. Możliwe jest także tworzenie systemów otwartych, jak np. w standardzie FIP, który jest rozwiązaniem dostępnym wszystkim producentom i użytkownikom.

Obecnie bardzo dobrze rozwinęła się infrastruktura sieci GSM. Szczególnie wprowadzenie przez operatorów usługi transferu danych w technologii GPRS pozwoliło na wykorzystanie tej platformy do tworzenia systemów monitorowania, charakteryzujących się dużą funkcjonalnością przy zachowaniu niskich kosztów eksploatacji. Rozwiązanie to polega na wyposażeniu sterownika w moduł do komunikacji GSM lub w odpowiednią przystawkę połączoną za pomocą standardowego portu szeregowego. Dane są transmitowane przez GPRS zgodnie z regułami określonymi przez użytkownika: jako odpowiedź na zapytanie, samodzielnie w określonych momentach czasu, samodzielnie w wyniku zaistnienia określonego zdarzenia (alarm, zmiana stanu, znacząca zmiana wartości analogowej, spełnione wyrażenie logiczne itp.). Za pomocą SMS-ów sterownik może informować operatora o stanie procesu. Jego główną zaletą jest możliwość praktycznie nieograniczonego kontaktu na odległość, jedyne opłaty wiążą się z wykupieniem abonamentu na usługi GSM.

6.1. Architektura sieci przemysłowych dla PLC

Sieci komputerowe są obecnie stosowane powszechnie w przemyśle na wszystkich poziomach sterowania, zarówno na niższych (sterowanie bezpośrednie), jak i na wyższych (sterowanie decyzyjne, zarządzanie) (rys. 47).

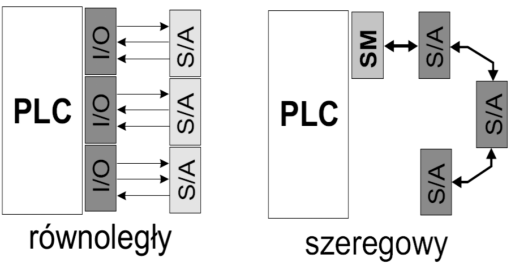


Rysunek 47. Hierarchiczny system sieci przemysłowych

W komunikacji między różnymi poziomami sterowania i zarządzania podstawą układów komputerowo zintegrowanego wytwarzania są:
- sieć miejscowa (polowa – przemysłowa) SM;
- sieć lokalna SL (biurowa).

Wymagania stawiane wobec sieci przemysłowej:
- łatwość dołączania prostych urządzeń;
- możliwość zasilania elementów sieci przez magistralę;
- łatwość lokalizacji uszkodzeń;
- odporność na błędy w montażu i konfiguracji;
- odporność na zakłócenia.

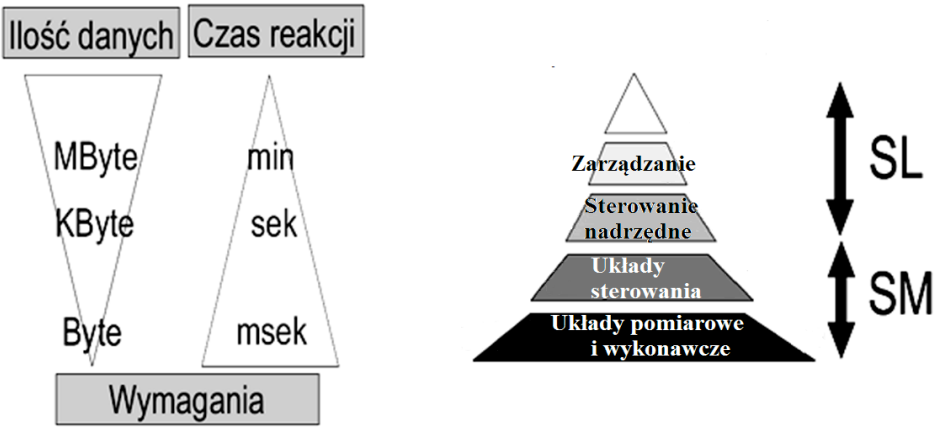
Sieci miejscowe wykorzystywane są do łączenia: sterownika PLC i programatora, PLC z regulatorem lub robotem przemysłowym, kilku sterowników. Sieci miejscowe nastawione są na przesyłanie mniejszej ilości danych, na krótkich odcinkach, z jak największą szybkością. Stosowany jest w nich uproszczony model przesyłania danych, wymagana jest od nich duża niezawodność. Sieci miejscowe korzystają z dwóch metod przesyłania danych: równoległej i szeregowej (rys. 48).



Rysunek 48. Metody przesyłania danych w sieciach miejscowych

Źródło: Grzywak, 1994

Na rysunku 49 przedstawiono istotne z punktu widzenia odbiorcy różnice pomiędzy rodzajami sieci. W odróżnieniu od sieci miejscowej, sieci lokalne przesyłają większe pakiety danych, co wymaga również wzrostu czasu na transfer.



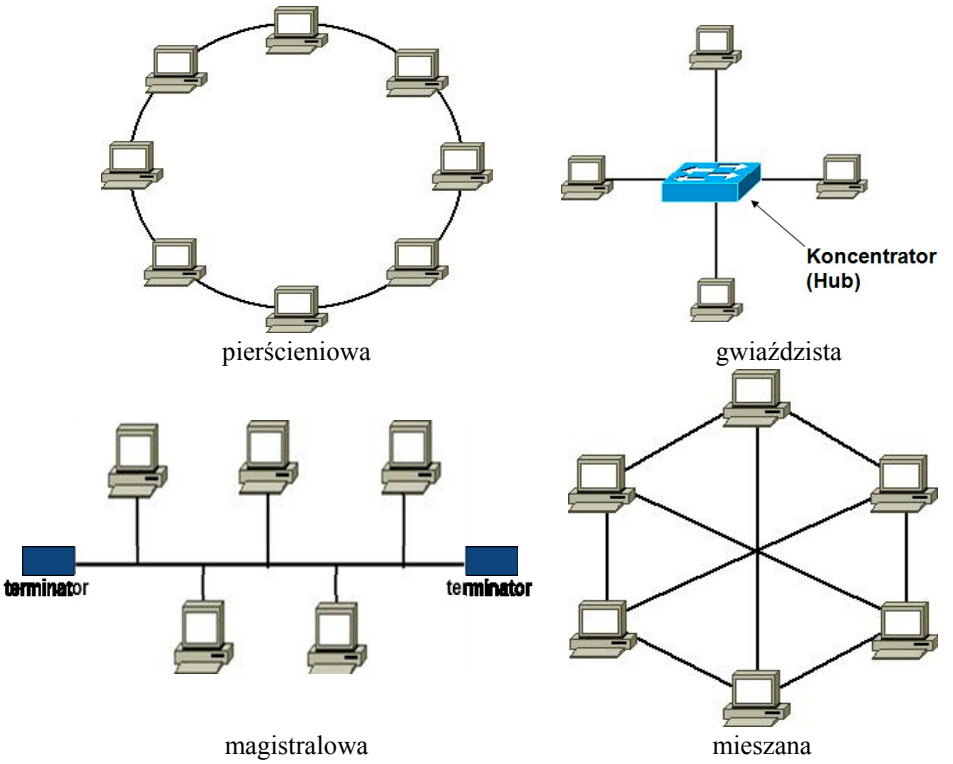
Rysunek 49. Porównanie sieci miejscowych i lokalnych

Źródło: Grzywak, 1994

Rozwój sieci miejscowych powodował powstawanie protokółów komunikacyjnych takich jak: Modbus, Profibus, CAN, Sercos, Interbus. Do najpopularniejszych należą trzy pierwsze, przy czym obecnie prym wiedzie Profibus, który posiada wersje najlepiej dopasowane do zastosowań. Dla prostych zastosowań z pojedynczym węzłem nadrzędnym na zasadzie odpytywania przeznaczony jest Profibus-DP. Profibus-FMS definiuje siódmą warstwę modeli OSI (aplikacyjną), stąd umożliwia współpracę na poziomie programów i przekazywania danych między węzłami. Profibus-PA przeznaczony jest do automatyki procesowej, stanowi łącze fizyczne standardu IEC1158-2 z najniższym poziomem, znajdującym się najbliżej sterowanego obiektu; jest to poziom związany z elementami wykonawczymi i sensorami wymagającymi pokazania informacji 1 bitu. Ta część sieci jest najczęściej związana z systemem sterowników i z dostarczaniem danych na temat sterowanego procesu technologicznego.

W przypadku połączenia między sobą większej liczby sterowników, komputerów lub innych urządzeń powstaje *sieć lokalna* LAN (Local Area Network). W zależności od struktury połączenia poszczególnych węzłów sieci (stacji), wyróżnia się kilka rodzajów topologii sieci (rys. 50). Podstawowe topologie sieci to: pierścieniowa, gwiaździsta, magistralowa oraz mieszana.

Topologia sieci ma zasadniczy wpływ na jej cechy funkcjonalne, takie jak niezawodność, złożoność interfejsu, modularność czy koszty. Najczęściej stosowanymi w warunkach przemysłowych są: sieci magistralowa oraz pierścieniowa. W tabeli 14 zestawiono wady i zalety sieci lokalnych.



Rysunek 50. Topologia sieci przemysłowych

Centralnym punktem sieci gwiaździstej jest koncentrator (hub), który ma oddzielne łącze do każdego węzła w sieci. W sieci takiej można korzystać nawet ze zwykłego interfejsu RS-232 lub łączy dalekopisowych TTY.

W sieci o strukturze pierścieniowej wszystkie węzły są równouprawnione i zbędny jest kontroler sieci. Dane są przesyłane w jednym kierunku, a kolizji zapobiega cykliczny sygnał zezwalający, tzw. żeton (token), wysyłany przez stację, która przetwarzała dane jako ostatnia. Inną możliwością zapobiegania kolizji jest reguła, która zapewnia podział czasu dostępu do sieci dla poszczególnych węzłów.

W sieciach magistralowych jest stosowana przede wszystkim reguła wymiany danych typu „nadrzędny-podrzędny" (Master-Slave), która uprawnia do inicjowania transmisji (zapytania) jedynie węzeł sieci z tytułem master. W sieci magistralowej dane mogą być również wymieniane przez przekazywanie dostępu do sieci. Polega to na tym, że kolejnym stacjom w sieci przydziela się tzw. żeton, który stanowi specjalny rodzaj sygnału zapewniający dostęp do sieci i możliwość wymiany danych.

Najbardziej niezawodną, ale zarazem najkosztowniejszą, jest sieć siatkowa (mieszana). W tej sieci awaria jednego z węzłów nie tamuje przesyłu danych, brak jest urządzeń pośrednich, jednak wymaga ona największego okablowania.

Tabela 14. Sieci lokalne – wady i zalety

Sieć	Zalety	Wady
Magistralowa	prosta budowa, mała ilość przewodów, awaria jednego węzła nie powoduje awarii sieci, łatwość rozbudowy,	trudna diagnostyka błędów, awaria terminatora lub kabla uniemożliwia działanie, możliwe opóźnienia,
Pierścieniowa	proste okablowanie, łatwa diagnostyka błędów,	awaria jednego węzła lub kabla uniemożliwia działanie całej sieci, złożona konstrukcja węzłów (nadawanie w jedną stronę)
Gwiaździsta	awaria jednego węzła nie powoduje awarii sieci, łatwość rozbudowy, łatwa diagnostyka błędów i awarii	wymagany dodatkowy element (Hub), awaria Huba powoduje awarię całej sieci, duża ilość kabli,
Siatkowa (mieszana)	typowa dla sieci rozległych wysoka niezawodność (każdy węzeł można osiągnąć różnymi drogami, awaria węzła nie powoduje awarii całości)	duża ilość kabli, złożona komunikacja (wiadomości muszą przechodzić przez węzły pośrednie)

Źródło: Werewka, 1997

Transmisja danych cyfrowych w sieciach lokalnych i przemysłowych odbywa się przede wszystkim z wykorzystaniem następujących mediów:
- przewód typu skrętka (najtańszy typ kabla, szybkość transmisji: 2 $MB \cdot s^{-1}$ do kilku km, ponad 10 $MB \cdot s^{-1}$ do 100 m, łącze 2-punktowe i wielopunktowe);
- kabel koncentryczny (wychodzi obecnie z użycia w sieciach przemysłowych, typowy kabel TV 75 Ohm oraz kabel 50 Ohm, łącze 2 punktowe i wielopunktowe);
- światłowód (wysokie koszty zarówno okablowania, jak i osprzętu – rozgałęźniki pryzmatyczne, przetworniki elektrooptyczne, bardzo małe tłumienie sygnału i całkowita niewrażliwość na zakłócenia, izolacja galwaniczna węzłów sieci, możliwość budowy rozległych sieci o dużych szybkościach transmisji, łącza 2-punktowe);
- łącza bezprzewodowe (GPRS, Bluetooth i inne).

Podstawowym warunkiem poprawnego przekazania wiadomości od nadawcy do odbiorcy jest uniknięcie kolizji z innymi wiadomościami. Zadanie to jest realizowane przez warstwę 2 (łącza danych) modelu OSI.

Na poziomie podwarstwy dostępu do kabla najczęściej stosowane są protokoły:
- *metoda przypadkowego dostępu do łącza* powoduje, że czasy przesyłania są wielkościami przypadkowymi, zależnymi od obciążenia sieci – nie nadaje się więc do sterowania w sieciach przemysłowych. Sieci tego rodzaju są przede wszystkim stosowane do komunikacji na wyższych poziomach hierarchii systemów sterowania, tzn. w systemach sterowania nadrzędnego SCADA oraz na poziomie zarządzania. W sieciach tych dużą popularność zdobyła niedeterministyczna, równoprawna reguła wielodostępu przez rozpoznawanie nośnej i wykrywanie

kolizji CSMA/CD (*ang. Carrier Sense Multiple Access with Collision Detection)*. Polega ona na nasłuchu łącza przed transmisją i na wykrywaniu oraz rozstrzyganiu sytuacji kolizyjnych, które występują w trakcie jednoczesnego zgłaszania potrzeby dostępu do sieci przez wielu jej użytkowników. W sytuacjach kolizyjnych każda ze stacji musi przerwać transmisję i odczekać losowo wybrany okres, po czym ponawia próbę dostępu do sieci. Reguła ta jest stosowana w sieciach Ethernet z wykorzystaniem protokołu TCP/IP (*ang. Transfer Control Protocol/Internet Protocol)*;

– *protokoły znacznikowe*: z przekazywaniem znacznika *(ang. Token Passing)*, pierścień znacznikowy *(ang. Token Ring)*. Pomiędzy węzłami sieci jest przekazywany znacznik (token), prawo do nadawania w danej chwili ma tylko ten węzeł, który aktualnie posiada znacznik. Maksymalny czas posiadania znacznika i kolejność jego przekazywania są zdefiniowane. Do zalet zaliczyć można determinizm czasowy, niezależność szybkości działania od obciążenia oraz brak ograniczeń na minimalną długość wiadomości. Wadami są skomplikowana obsługa błędów związanych z zgubieniem lub powieleniem znacznika oraz konieczność ciągłego przekazywania znacznika pomiędzy węzłami;
– *odpytywanie (ang. Polling)*, system w naturalny sposób dostosowany do pracy w sieciach typu Master-Slave. Master ma uprawnienia do wysyłania danych i zapytań do stacji Slave, może inicjalizować komunikację. Slave nie może inicjalizować komunikacji, musi odpowiedzieć na zapytanie. Komunikacja odbywa się wyłącznie pomiędzy Masterem i Slave na zasadzie transakcji pytanie – odpowiedź. Zaletami są prostota oraz to, że elementy Slave nie muszą być komputerami. Do najważniejszych wad należą całkowity rozpad sieci w razie awarii Mastera (ale w razie takiej awarii sieć i tak nie jest potrzebna – Masterem może być np. CPU sterownika).

W dwóch ostatnich metodach czasy przesyłania informacji w sieci są zdeterminowane, a obciążenie sieci nie ma wpływu na czas dostępu do łącza. Sieci tego rodzaju nadają się więc do systemów, w których czas reakcji musi być ściśle określony, jak np. w sieciach przemysłowych.

6.2. Model sieci lokalnej wg OSI

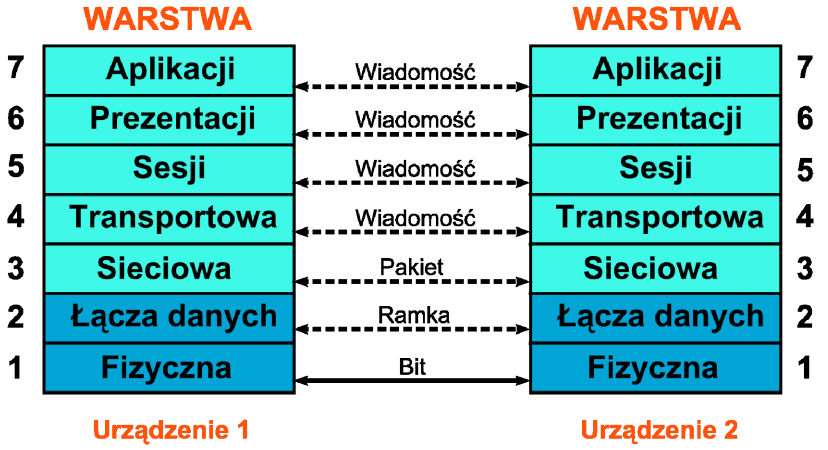
Celem umożliwienia współpracy urządzeń pochodzących od różnych dostawców, konieczne stało się opracowanie zasad opisujących sposoby ich komunikowania się. Standardy takie tworzą międzynarodowe organizacje finansowane przez producentów sprzętu sieciowego. Do najbardziej znanych należą ISO (*International Standard Organization*) i IEEE (*Institute of Electrical and Electronic Engineers*) (Miłosiewicz, 1997).

Model Referencyjny Połączonych Systemów Otwartych (czyli model OSI) został opracowany 1984 roku celem ułatwienia realizacji połączeń w otwartych systemach komputerowych. Połączenia otwarte to takie połączenia, które mogą być realizowane wewnątrz lub między wieloma systemami. OSI jest zbiorem zasad komunikowania się urządzeń sieciowych. Podzielony jest na siedem warstw, z których każda zbudowana jest na bazie warstwy poprzedniej. Rozróżnia się podział ze względu na sposób realizacji danej warstwy. Warstwy 1 i 2 są tworzone w sposób sprzętowy (hardwareowy), a warstwy od 3 do 7 w sposób softwarowy (programowy). Model ten nie określa fizycznej budowy poszczególnych warstw, a koncentruje się na sposobach ich współpracy. Takie podejście do problemu sprawia, że każda warstwa może być implementowana przez producenta na swój sposób, a urządzenia sieciowe od różnych dostawców będą poprawnie współpracować. Poszcze-

gólne warstwy sieci stanowią niezależne całości i chociaż nie potrafią wykonywać żadnych widocznych zadań w odosobnieniu od pozostałych warstw, to z programistycznego punktu widzenia są one odrębnymi poziomami (Mielczarek, 1994).

Komunikacja pomiędzy komputerami odbywa się na poziomie odpowiadających sobie warstw i dla każdej z nich powinien zostać stworzony własny protokół komunikacyjny.

W rzeczywistej sieci komputerowej komunikacja odbywa się wyłącznie na poziomie warstwy fizycznej (linia ciągła na rysunku 51). W tym celu informacja każdorazowo przekazywana jest do sąsiedniej niższej warstwy, aż do dotarcia do warstwy fizycznej. Tak więc pomiędzy wszystkimi warstwami z wyjątkiem fizycznej istnieje komunikacja wirtualna (linie przerywane na rysunku), możliwa dzięki istnieniu połączenia fizycznego.



Rysunek 51. Model warstwy ISO-OSI sieci komputerowych

Źródło: Mielczarek, 1994

Warstwa fizyczna odpowiada za transmisję sygnałów w sieci. Realizuje ona konwersję bitów informacji na sygnały, które będą przesyłane w kanale z uwzględnieniem maksymalizacji niezawodności przesyłu. W warstwie tej określa się: parametry amplitudowe i czasowe przesyłanego sygnału, fizyczny kształt i rozmiar łączy, znaczenie ich poszczególnych zestyków i wartości napięć na nich występujących, sposoby nawiązywania połączenia i jego rozłączania po zakończeniu transmisji.

Warstwa łącza danych odpowiedzialna jest za odbiór i konwersję strumienia bitów pochodzących z urządzeń transmisyjnych w taki sposób, aby nie zawierały one błędów. Warstwa ta postrzega dane jako grupy bitów zwane ramkami, posiada zdolność tworzenia i rozpoznawania granic ramki. Ramka tworzona jest przez dołączenie do jej początku i końca grupy specjalnych bitów. Kolejnym zadaniem warstwy jest eliminacja zakłóceń, powstałych w trakcie transmisji informacji po kanale łączności. Ramki, które zostały przekazane niepoprawnie, są przesyłane ponownie. Ponadto warstwa łącza danych zapewnia synchronizację szybkości przesyłania danych oraz umożliwia ich przesyłanie w obu kierunkach.

Warstwa sieciowa steruje działaniem podsieci transportowej. Jej podstawowe zadania to przesyłanie danych pomiędzy węzłami sieci wraz z wyznaczaniem trasy przesyłu, określanie charakterystyk sprzęgu węzeł–komputer obliczeniowy, łączenie bloków informacji w ramki na czas ich przesyłania, a następnie stosowny ich podział. W najprostszym przypadku określanie drogi transmisji pakietu informacji odbywa się w oparciu o stałe tablice opisane w sieci. Istnieje również możliwość dynamicznego określania trasy na bazie bieżących obciążeń linii łączności. Stosując drugie rozwiązanie, mamy możliwość uniknięcia przeciążeń sieci na trasach, na których pokrywają się drogi wielu pakietów.

Podstawową funkcją *warstwy transportowej* jest obsługa danych przyjmowanych z warstwy sesji. Obejmuje ona opcjonalne dzielenie danych na mniejsze jednostki, przekazywanie zblokowanych danych warstwie sieciowej, otwieranie połączenia stosownego typu i prędkości, realizację przesyłania danych, zamykanie połączenia. Ponadto mechanizmy wbudowane w warstwę transportową pozwalają rozdzielać logicznie szybkie kanały łączności pomiędzy kilka połączeń sieciowych. Możliwe jest także udostępnianie jednego połączenia kilku warstwom sieciowym, co może obniżyć koszty eksploatacji sieci. Celem postawionym przy projektowaniu warstwy transportowej jest zapewnienie pełnej jej niezależności od zmian konstrukcyjnych sprzętu.

Warstwa sesji określa parametry sprzężenia użytkowników. Po nawiązaniu stosownego połączenia warstwa sesji pełni szereg funkcji zarządzających, związanych m. in. z taryfikacją usług w sieci. W celu otwarcia połączenia pomiędzy komputerami (sesji łączności) poza podaniem stosownych adresów warstwa sprawdza, czy obie warstwy (nadawcy i odbiorcy) mogą otworzyć połączenie. Następnie obie komunikujące się strony muszą wybrać opcje obowiązujące w czasie trwania sesji. Dotyczy to na przykład rodzaju połączenia (simpleks, dupleks) i reakcji warstwy na zerwanie połączenia (rezygnacja, ponowne odtworzenie). Przy projektowaniu warstwy zwraca się uwagę na zapewnienie bezpieczeństwa przesyłanych danych. Przykładowo, jeżeli zostanie przerwane połączenie, którego zadaniem była aktualizacja bazy danych, to w rezultacie tego zawartość bazy może okazać się niespójna. Warstwa sesji musi przeciwdziałać takim sytuacjom.

Zadaniem *warstwy prezentacji* jest obsługa formatów danych. Odpowiada ona za kodowanie i dekodowanie zestawów znaków oraz wybór algorytmów, które do tego będą użyte. Przykładową funkcją realizowaną przez warstwę jest kompresja przesyłanych danych, pozwalająca na zwiększenie szybkości transmisji informacji. Ponadto warstwa udostępnia mechanizmy kodowania danych w celu ich utajniania oraz konwersję kodów w celu zapewnienia ich mobilności.

Warstwa aplikacji zapewnia programom użytkowym usługi komunikacyjne. Określa ona formaty wymienianych danych i opisuje reakcje systemu na podstawowe operacje komunikacyjne. Warstwa stara się stworzyć wrażenie przezroczystości sieci. Jest to szczególnie ważne w przypadku obsługi rozproszonych baz danych, w których użytkownik nie powinien wiedzieć, gdzie zlokalizowane są wykorzystywane przez niego dane lub gdzie realizowany jest jego proces obliczeniowy.

6.3. Łącza szeregowe wykorzystywane w sterownikach PLC

Standard RS-232 (Recommended Standard) opisuje sposób podłączenia urządzenia DTE (ang. Data Terminal Equipment), czyli *końcowych urządzeń danych,* takich jak np. komputer, oraz urządzeń DCE (and. Data Communication Equipment), do których można

zaklasyfikować np. modem. Został opracowany w 1962 roku przez stowarzyszenie EIA (Electronic Industries Association) w celu ujednolicenia parametrów sygnału i konstrukcji urządzeń i zrewidowany w 1969 roku. Standard RS-232 umożliwia transmisję danych przez niesymetryczne (jednoprzewodowe) łącze bit po bicie, pozwala na transfer na odległość nieprzekraczającą 15 m z szybkością maksymalną 20 kbit·s^{-1}. Specyfikacja napięć i odpowiadających im stanów logicznych została przedstawiona w tabeli 15. Należy zaznaczyć przy tym, że zwarcie dwóch styków RS-232 teoretycznie nie powoduje jego uszkodzenia (Mielczarek, 1994).

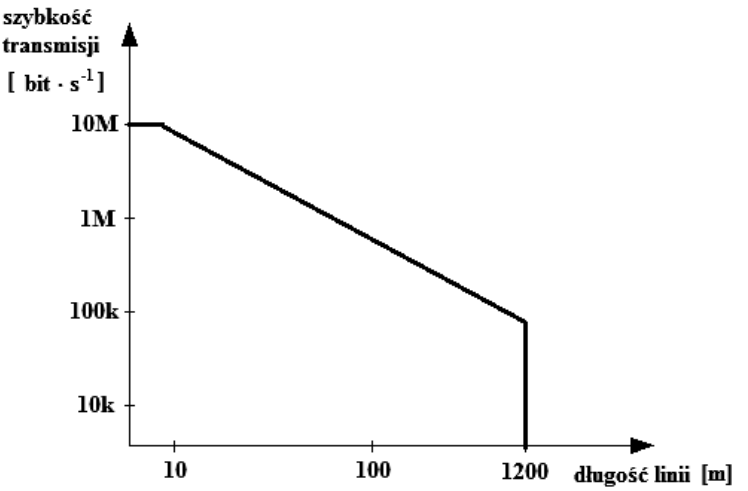
Tabela 15. Standardy napięć portu RS-232

Stan logiczny	Napięcie dopuszczalne	Napięcie maksymalne
0	-3V do -15V	-25V
1	+3V do +15V	+25V

Źródło: Mielczarek, 1994

Rozwinięciem ww. standardu są obwody dwuprzewodowe, zdefiniowane jako standardy RS-422 i RS-485, dzięki czemu zwiększa się zasięg, prędkość i odporność na zakłócenia.

Standard RS-422 określa obwody transmisyjne, które są symetryczne i zrównoważone. Składają się one z nadajnika, zrównoważonego dwuprzewodowego toru przesyłowego oraz odbiornika z wejściem różnicowym. Prędkość transmisji wynika z ograniczenia, jakie w tym standardzie nakłada się na szybkość zmian sygnału i czas trwania bitu, może wynosić od 0,1 do 10 Mb·s^{-1}. Odległość przesyłu danych może sięgać 1200 m, jednak wraz ze wzrostem odległości maleje szybkość transmisji (rys. 52).



Rysunek 52. Zależność szybkości transmisji od długości przewodu

Źródło: Mielczarek, 1994

Właściwości nadajnika:
- różnicowy obwód wyjściowy o napięciu zrównoważonym od 2 do 6 V;
- rezystancja nie większa niż 100 Ω;
- prąd zwarciowy źródła nie może być większy od 0,15 A.

Właściwości odbiornika:
- impedancja wejściowa musi być nie mniejsza niż 4 kΩ;
- parametry elektryczne określa progowe napięcie wejścia, które wynosi 200 mV dla napięć wspólnych od -7 do +7 V;
- odporność odbiornika – przyjmuje się, że nie może ona ulec uszkodzeniu po wyłączeniu lub załączeniu zasilania, w warunkach gdy nastąpi zwarcie na linii lub wystąpi zwarcie przewodu do masy, lub gdy nadajnik zostanie wyłączony (Mielczarek, 1994).

Standard RS-485 został wprowadzony w 1983 roku jako rozszerzenie standardu RS-422 i do dnia dzisiejszego stał się bazą wielu protokołów komunikacyjnych. Jest łączem szeregowym asynchronicznym przeznaczonym do realizacji szybkiej transmisji danych na duże odległości. System transmisji złożony z różnicowych nadajników, dwuprzewodowego zrównoważonego toru przesyłowego oraz odbiorników o różnicowym obwodzie wejściowym. Zastosowanie w omawianym standardzie trójstanowych nadajników pozwala na dołączenie do wspólnej linii wielu stacji nadawczo-odbiorczych. Warunkiem poprawnej pracy jest nadawanie w danym czasie tylko jednego z nich, pozostałe powinny w tym czasie znajdować się w stanie wysokiej impedancji. W zastosowaniach przemysłowych standard RS-485 wykorzystuje kable o impedancji charakterystycznej 120 Ω i przekroju 22AWG (ok. 0,6mm). Wymagane jest zastosowanie rezystorów dopasowujących, o wartości impedancji charakterystycznej przewodu.

Właściwości nadajników:
- jeden nadajnik może być obciążony przez maksymalnie 32 odbiorniki (obwody wprowadzające obciążenie do 1 mA przy napięciu wspólnym 12 V) oraz dwie zastępcze rezystancje dopasowujące: $R_T = 60\ \Omega$ lub większe;
- prąd upływu nadajnika w stanie wyłączenia nie może przekraczać 100 μA;
- rezystancja wyjściowa nadajnika 120 kΩ – wysoka impedancja (wyłączone zasilanie);
- nadajnik powinien zapewnić różnicowe napięcie wyjściowe z przedziału (-1.5 V do 5 V) przy obecności wspólnego napięcia (-7 V do 12 V);
- nadajnik musi mieć zabezpieczenie przed kolizją przy nadawaniu jednocześnie przez kilka nadajników (nadajnik nie może zostać uszkodzony).

Właściwości odbiorników:
- impedancja wejściowa powinna wynosić nie mniej niż 12 kΩ;
- zakres napięcia wspólnego na wejściu odbiornika (-7 V do 12 V);
- progowe napięcie wejścia wynosi 200 mV dla napięć wspólnych (Mielczarek, 1994).

W omawianych łączach transmisja danych może być realizowana jako asynchroniczna lub synchroniczna transmisja znakowa. W transmisji asynchronicznej przesyła się za pomocą pojedynczej linii danych pojedyncze znaki w określonym formacie. Linia w stanie spoczynkowym znajduje się na poziomie boolowskiej jedynki. Pierwszy znak jest bitem startu o wartości zera boolowskiego, a następne bity tworzą znak w kodzie ASCII, począwszy od bitu najmniej znaczącego *(ang. Least Significant Bit – LSB)*. Po nich występuje bit kontrolny zawartości bitów znaku, a na końcu 1 lub 2 bity stopu. Przesyłanie bitów w ra-

mach jednego znaku jest synchronizowane taktowaniem nadajnika, znaki są zaś przesyłane asynchronicznie.

W transmisji synchronicznej dane są przesyłane w postaci ramek ze znacznikami początku i końca ramki, które umożliwiają synchronizację po stronie odbiornika. Odbiór danych w takiej transmisji wymaga rozpoznawania kolejnych bitów (synchronizacji bitowej) i łączenia bitów w znaki (synchronizacji znakowej).

6.4. Protokoły komunikacyjne

Przesyłanie informacji jest realizowane za pośrednictwem *protokołów komunikacyjnych.* Stanowią one zbiór ścisłych reguł i kroków postępowania, które są automatycznie wykonywane przez urządzenia komunikacyjne w celu nawiązania łączności i wymiany danych, korzystają z opracowanego 7- warstwowego systemu ISO-OSI. Ze względu na dużą złożoność oraz różnorodność możliwych rozwiązań istnieje konieczność pewnej standaryzacji protokołów komunikacyjnych. Organizacje normalizacyjne, takie jak: ISO (International Standards Organisation), EIA (Electronic Industries Association), PNO (Profibus Nutzer Organization), IEEE (Institute of Electrical and Electronic Engineers), CCITT (Consultive Committee on International Telephone and Telegraph) i inne, opracowały różne wielowarstwowe protokoły komunikacyjne (Miłosiewicz, 1997).

Protokoły klasyczne, których pierwowzorem był protokół teleksu, składają się z trzech części:

- procedury powitalnej (ang.) handshake, która polega na przesłaniu wzajemnej podstawowej informacji o łączących się urządzeniach, ich adresu (np. nr telefonu), szybkości i rodzaju transmisji itd.;
- właściwego przekazu danych, procedury analizy poprawności przekazu (np. sprawdzania sum kontrolnych) połączonej z procedurą pożegnania, żądaniem powtórzenia transmisji lub powrotem do procedury powitalnej.

Przesyłana informacja może być porcjowana – protokół musi umieć odtworzyć informację w postaci pierwotnej.

Reguły opisujące protokół dotyczą:

- mechanicznej, elektrycznej i funkcjonalnej charakterystyki łącza komunikacyjnego;
- sposobów sterowania umożliwiających przesłanie danych pomiędzy nadajnikiem, a odbiornikiem.

W systemach sterowników programowalnych do komunikacji stosowane są przede wszystkim sieci przemysłowe. Sieci takie powinny spełniać wymagania pracy w czasie rzeczywistym (czas potrzebny na przesył informacji od stacji do stacji jest zdeterminowany) oraz zwiększonej odporności na występujące w warunkach przemysłowych zakłócenia.

6.4.1. Modbus

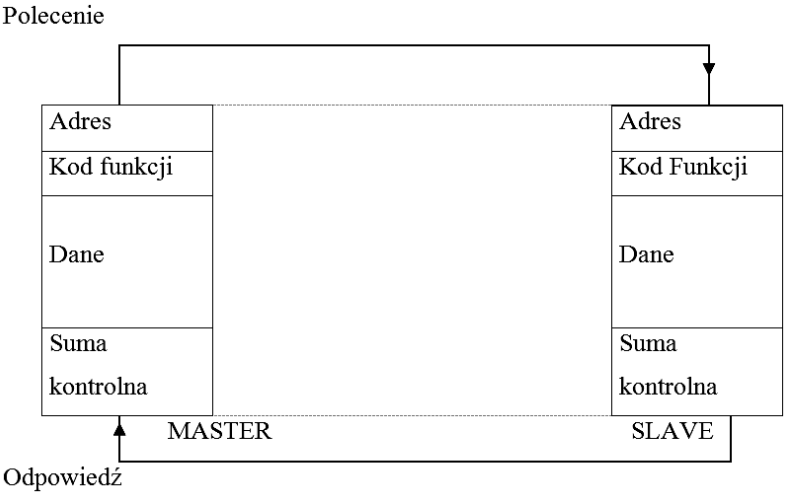
Protokół Modbus został opracowany w firmie Modicon (obecnie Schneider Electric) w 1980 roku. Mimo upływu dość znacznego czasu od chwili wprowadzenia jest on nadal szeroko stosowany w aplikacjach automatyki przemysłowej o niskich wymaganiach dotyczących szybkości i częstości transmisji danych, w szczególności w systemach z wydzielonym centrum, do którego przesyłane są dane z urządzeń peryferyjnych. Jest standardem

przyjętym przez większość producentów sterowników przemysłowych dla asynchronicznej komunikacji pomiędzy urządzeniami wyposażonymi w interfejs zgodny z RS-232 takich jak: RS-422, RS-485. W procedury komunikacyjne realizujące ten protokół są wyposażone niemal wszystkie dostępne na rynku pakiety SCADA.

Modbus swą popularność zyskał dzięki prostocie zastosowanych w nim rozwiązań, jawności specyfikacji protokołu, a ponadto takim cechom, jak: dostęp do łącza na zasadzie Master-Slave, zabezpieczenie komunikatów przed przekłamaniami, potwierdzenie wykonania rozkazów i sygnalizacja błędów oraz mechanizmy unikające zawieszania się systemu. Pozwala to na łatwą implementację w dowolnym urządzeniu posiadającym mikrokontroler i w znacznym stopniu wpływa na obniżenie kosztów. W modelu ISO/OSI protokół Modbus zajmuje trzy warstwy: 1 – fizyczną, 2 – łącza danych oraz 7 – aplikacji (Werewka, 1997).

Modbus może pracować w dwóch trybach transmisji: RTU i ASCII. Istnieje również implementacja na stosie TCP/IP w sieci Ethernet.

Najczęściej stosowaną i używaną konfiguracją w automatyce przemysłowej jest protokół Modbus współpracujący z interfejsem RS-485, gdzie występuje jedno urządzenie nadrzędne (Master) inicjalizujące transakcje (wysyłające polecenie – *ang. Query*), natomiast pozostałe urządzenia są podrzędne (Slaves), wykonują polecenia Master'a i odsyłają odpowiedź *(ang. Response)* (rys. 53). W danej chwili tylko jeden Slave może odpowiadać na zdalne zapytanie Mastera, natomiast nie ma możliwości komunikacji pomiędzy urządzeniami podrzędnymi. Typowym Masterem jest urządzenie z procesorem głównym *(ang. Host Procesor)*, zawierające programowalny panel, na przykład komputer PC lub nadrzędny sterownik logiczny, a typowy Slave to programowalny sterownik logiczny. Węzły podrzędne (Slaves) są wykorzystywane do sterowania oraz zbierania danych z urządzeń peryferyjnych takich jak: mierników, liczników, przetworników A/C i C/A, czujników, przekaźników, sygnalizatorów itp.



Rysunek 53. Transakcja w systemie Modbus

W protokole Modbus możliwe są dwa typy transakcji:

- *rozgłoszenie* (ang. *Broadcast*) – jednostka Master wysyła jedynie ramkę rozgłoszenia o adresie 0, który nie jest przypisany do konkretnego urządzenia Slave, ale wszystkie urządzenia ją odbierają i wykonują zawartą w niej funkcję, nie odsyłając odpowiedzi;
- *zapytanie – odpowiedź* (ang. *Query-Response*) – jednostka Master wysyła zapytanie i oczekuje na odpowiedź od wybranego urządzenia Slave.

Master może adresować indywidualnych odbiorców (jednostki Slave) lub też przesyłać wiadomości rozgłoszeniowe przeznaczone dla wszystkich urządzeń podrzędnych w systemie. Na polecenia rozgłoszeniowe jednostki Slave nie odpowiadają (nie wysyłają odpowiedzi).

Istnieje również odmiana protokołu Modbus z więcej niż jednym masterem – Modus Multi-Master, w którym jednostki nadrzędne przekazują sobie wzajemnie prawo do nadawania. Protokół pracuje z niewielkimi prędkościami transmisji danych (typowe: 9,6 Kb $\cdot s^{-1}$, 19,2 Kb$\cdot s^{-1}$) na ograniczonym dystansie wynikającym z typu zastosowanego łącza komunikacyjnego (RS-232, RS-422, RS-485, Modem). Pozwala ono pracować w warunkach silnych zakłóceń (np. w przemyśle). Główne zalety protokołu to:

– prostota zastosowanych w nim rozwiązań;
– jawność specyfikacji protokołu;
– zabezpieczenie przesyłanych komunikatów przed błędami;
– potwierdzanie wykonania rozkazów zdalnych i sygnalizacja błędów;
– stały format ramki i zestaw standardowych funkcji służących wymianie danych;
– mechanizmy zabezpieczające przed zawieszeniem systemu (Werewka, 1997).

W systemie MODBUS wiadomości są zorganizowane w ramki o określonym początku i końcu. Pozwala to urządzeniu odbierającemu na odrzucenie ramek niekompletnych i sygnalizację związanych z tym błędów. Definiuje się dwie ramki ze względu na możliwość pracy w jednym z dwóch trybów transmisji (ASCII lub RTU).

Tryb ASCII

W trybie pracy ASCII każdy bajt wiadomości przesyłany jest w postaci dwóch znaków ASCII (tryb znakowy). Podstawową zaletą tego trybu transmisji jest to, że pozwala on na długie odstępy między znakami bez podawania błędów. Format znaku przesyłanego w trybie transmisji ASCII jest następujący:

– system kodowania: heksadecymalny, znaki ASCII 0-9, A-F. Jeden znak heksadecymalny zawarty jest w każdym znaku ASCII wiadomości;
– jednostka organizacyjna jest 10-bitowa, z tym że ograniczona jest bitem startu (na początku) i bitem stopu (na końcu).

Znacznikiem początku w ramce w trybie ASCII jest dwukropek (‘:’ odpowiada w ASCII 3A hex). Oprócz znacznika końca ramki (CRLF), dopuszczalnymi znakami dla pozostałych pól są 0 – 9, A – F. Urządzenie po wykryciu znacznika początku ramki sprawdza, czy pole adresowe zawiera jego adres własny. Jeżeli tak, to odczytuje zawartość pola funkcji i związaną z nią zawartość pola danych. Pole kontrolne LRC zabezpiecza część informacyjną ramki (czyli bez znaku ‘:’). Odstęp między znakami tworzącymi ramkę nie

może przekraczać 1 sek. W przeciwnym wypadku urządzenie odbierające dane zasygnalizuje błąd.

Tryb RTU

W trybie RTU (tryb binarny) wiadomości rozpoczynają się odstępem czasowym trwającym min. 3,5 x T_z (czas trwania pojedynczego znaku), w którym to czasie panuje cisza na łączu. Odmierza się wielokrotnie czas trwania znaku przy zadanej szybkości bodowej przyjętej na łączu. Pierwszym polem informacyjnym ramki jest adres urządzenia.

Dalej występują dane, jako ciąg 8-bitowych liczb w zakresie od 16#00 do 16#FF. Ramka kończy się 16-bitowym słowem kontrolnym, obliczanym według algorytmu CRC (Cyclical Redundancy Check). Ramkę kończy przerwa czasowa trwająca co najmniej 3,5 x T_z. Przerwę tą można traktować jako początek następnej ramki. Cała ramka musi zostać przesłana w sposób ciągły, tzn. odstęp między kolejnymi znakami w ramce nie może być większy niż 1.5 x T_z. Jeżeli odstęp ten będzie dłuższy, urządzenie odbierające uzna ramkę za niekompletną i następny znak przyjmie za bajt pola adresowego kolejnej wiadomości. Natomiast jeżeli nowa wiadomość pojawi się na magistrali przed upływem ciszy (3.5 x T_z) , to urządzenie odbierające potraktuje ją jako kontynuację poprzedniej wiadomości. To doprowadzi do błędu sumy kontrolnej, ponieważ urządzenie odbierające będzie obliczać CRC dwóch wiadomości i porównywać go z CRC drugiej.

W tym przypadku jednostka informacyjna zawiera 11 bitów: bit startu, 8 bitów informacyjnych, bit parzystości i stopu. Zaletą trybu RTU jest większa przepustowość łącza.

6.4.2. Profibus

Sieć Profibus jest otwartą i standardową technologią komunikacyjną, która stwarza liczne możliwości aplikacyjne w automatyce przemysłowej i procesowej. Komunikacja Profibus oparta jest na międzynarodowym standardzie IEC 61158 oraz IEC 61784. Standaryzacja spełnia oczekiwania użytkowników, ich niezależność i otwartość, jak również zapewnia komunikację pomiędzy poszczególnymi stacjami i urządzeniami różnych producentów. Historia sieci Profibus rozpoczyna się przez złożenie projektu utworzenia organizacji w roku 1987 w Niemczech, kiedy powołało ją 21 przedsiębiorstw i instytucji skupionych w celu utworzenia i wspierania strategii projektu sieci polowej. Cel, jaki sobie postawiono, to utworzenie sieci cyfrowej, która byłaby standardem dla rozproszenia stacji polowych. Pierwszym krokiem była specyfikacja protokołu komunikacyjnego Profibus FMS (ang. *Fieldbus Message Specification*). Kolejnym w 1993 r. było opracowanie protokołu Profibus-DP (ang. *Decentralized Periphery*), który w założeniach miał być prostszy i szybszy. Obecnie protokół ten dostępny jest w trzech wersjach DP-V0, DP-V1 i DP-V2 (Boroń i Wyłupek, 1996).

Węzłami sieci mogą być zarówno proste urządzenia: wejścia/wyjścia analogowe i cyfrowe, czujniki lub elementy wykonawcze, jak i bardziej zaawansowane: komputery, sterowniki PLC, falowniki czy też terminale operatorskie. Zadaniem sieci jest efektywne przekazywanie dużej ilości krótkich informacji przy zachowaniu deterministycznego czasu przesyłania danych.

Protokół komunikacyjny sieci Profibus definiuje norma DIN 19245, która opisuje warstwę fizyczną, liniową i aplikacyjną siedmiowarstwowego modelu ISO/OSI. Przy czym

warstwa aplikacyjna jest opcjonalna. Użytkownicy (wykonywane programy) korzystają z sieci, wywołując usługi warstwy aplikacyjnej lub liniowej. Warstwa liniowa odpowiada za niezawodne przekazywanie komunikatu z odpowiedzią lub potwierdzeniem odbioru oraz przekazywanie komunikatu bez potwierdzenia, w tym rozgłaszanie (ang. *broadcast*). Usługi warstwy aplikacyjnej udostępniają obiekty programowe zdefiniowane w innych węzłach sieci (zmienne, zdarzenia, programy) oraz umożliwiają bezpołączeniowe przekazywanie wartości zmiennych i zdarzeń do odbiorców wykonywanych w wielu węzłach. Profibus wykorzystuje warstwy 1, 2 oraz 7 modelu ISO/OSI.

Profibus-FMS przeznaczony jest do wzajemnej komunikacji na poziomie sterowania jednostek centralnych PLC oraz komputerów PC. Pozwala na wymianę informacji z prędkością 0,5 – 1,5 $Mb \cdot s^{-1}$. Był on poprzednikiem protokołu Profibus DP.

Profibus-DP jest przeznaczony do szybkiej wymiany danych (1,5 – 12 $Mb \cdot s^{-1}$) na poziomie urządzeń procesowych, takich jak urządzenia we/wy, sterowniki, napędy i przetworniki.

Pierwotna wersja DP-V0 i późniejsza rozszerzona wersja DP-V1 pozwalają na cykliczną wymianę danych pomiędzy stacjami Master i Slave. Wersja DP-V2 pozwala dodatkowo na komunikację *Slave-to-Slave* z *trybem pracy izochronicznym* (synchronizacja cyklu przetwarzania programu z obsługą sygnałów wej/wyj). Zmiany w poszczególnych wersjach są następujące:

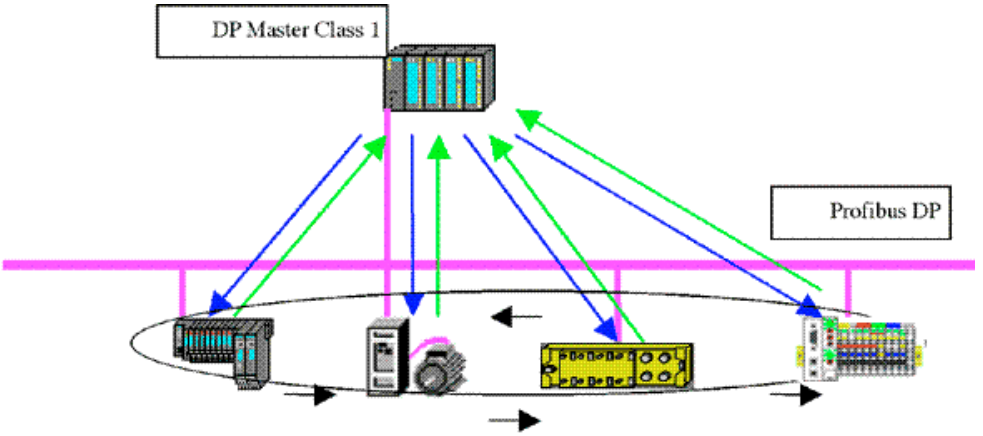
- **DP-V0** posiada podstawowe funkcje DP, włączając cykliczną wymianę danych, jak również diagnostykę stacji, diagnostykę modułu i poszczególnych kanałów;
- **DP-V1** zawiera rozszerzenia związane z automatyką procesu, w szczególności acykliczną komunikację danych w celu parametryzacji, obsługi wizualizacji i alarmów inteligentnych urządzeń polowych, działającą wraz z cykliczną komunikacją danych użytkowych. Pozwala to na bezpośredni dostęp do stacji, wykorzystując odpowiednie narzędzia inżynierskie. Dodatkowo DP-V1 definiuje alarmy. Przykładem różnych typów alarmów są alarmy statusowe, alarmy odświeżania oraz alarmy określone przez użytkownika;
- **DP-V2** wprowadza dalsze rozszerzenia, które są związane przede wszystkim z napędami. Dzięki dodatkowym funkcjom, takim jak izochroniczny tryb pracy oraz komunikacji Slave-to-Slave (DXB, Data eXchange Broadcast) itp. DP-V2 implementowany jest do obsługi napędów przy sterowaniu osiami.

Topologia sieci może być liniowa lub drzewiasta i łączyć 126 uczestników w czterech segmentach, przy czym w jednym segmencie może ich być do 32. Zaleca się jednak, by nie przekraczać 50 węzłów w sieci.

Szybkość transmisji danych zależy od rozległości i topologii sieci oraz liczby uczestników i wynosi od 9,6 $kb \cdot s^{-1}$ do 12 $Mb \cdot s^{-1}$. Wartość maksymalną można uzyskać w przypadku bezpośredniej transmisji z udziałem dwóch stacji i odległości między nimi mniejszej od 100 m. Maksymalny zasięg transmisji pomiędzy dwoma uczestnikami wynosi 9600 m przy użyciu wzmacniaczy sygnałowych.

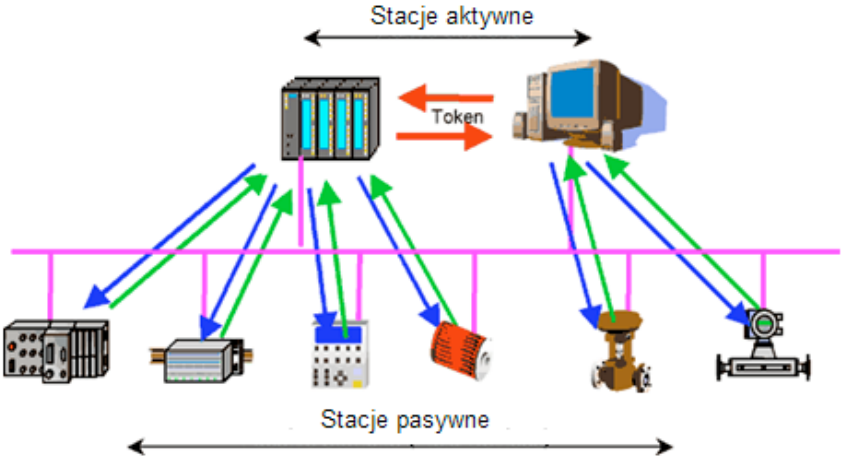
Rozwinięte funkcje diagnostyczne magistrali Profibus-DP umożliwiają bardzo szybką lokalizację usterek. Komunikaty diagnostyczne są przesyłane za pośrednictwem magistrali i interpretowane przez urządzenie zewnętrzne.

W strukturze *Mono-Master* aktywna jest tylko jedna stacja Master w sieci. PLC stanowi centralną jednostkę sterującą. Stacje Slave połączone są zdalnie do PLC poprzez odpowiednie medium transmisyjne. Tego typu konfiguracja systemu pozwala na uzyskanie najkrótszych czasów cyklu (rys. 54).



Rysunek 54. Sieć Profibus-DP typu mono-master

W systemie *Multi-Master* kilka stacji Master podłączonych jest do sieci. Stanowią one albo niezależne podsystemy, albo traktowane są jako dodatkowe stacje skonfigurowane w systemie i stacje diagnostyczne. Stan wejść i wyjść stacji Slave może być odczytywany przez każdą ze stacji DP Master, natomiast tylko jedna stacja DP Master (skonfigurowana stacja DPM1) może ustawiać wyjścia (rys. 55).

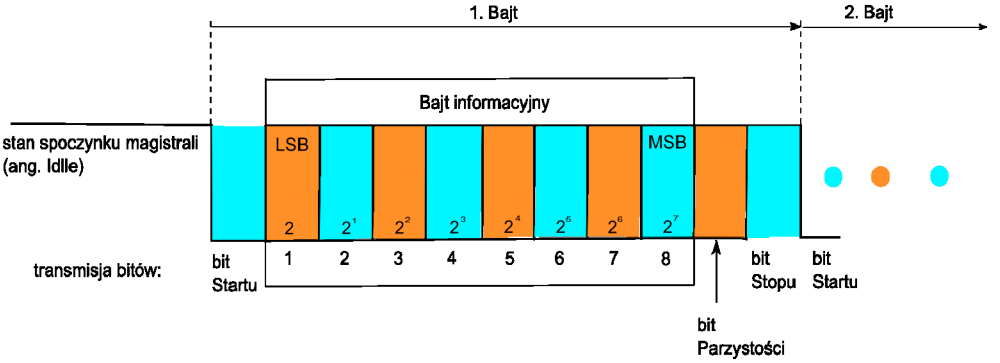


Rysunek 55. Sieć Profibus-DP typu multi-master

System DP określa zasadniczo trzy różne typy stacji (typy urządzeń):

– DP Master klasy 1 (DPM1), jest to jednostka centralna, która cyklicznie wymienia informacje ze stacjami rozproszonymi (Slave). Typowo stację DPM1 stanowi sterownik programowalny (PLC) lub komputer PC. DPM1 posiada aktywny dostęp do sieci z możliwością odczytu danych wejściowych (input) stacji polowych oraz z możliwością zapisu wartości wyjściowych (outputs);
– DP Master klasy 2 (DPM2), urządzenia tego typu stanowią stacje inżynierskie i systemy konfiguracyjne. Wykorzystuje się je podczas uruchamiania i do obsługi i diagnostyki skonfigurowanych stacji, odczytu wartości wejściowych i parametrów oraz statusu urządzenia. Master DPM2 nie musi być podpięty ciągle do sieci systemowej. DPM2 również posiada aktywny dostęp do sieci;
– stacje Slave to urządzenia peryferyjne, takie jak: moduły I/O, napędy, panele, zawory, przetworniki, które przekazują informacje o procesie i do procesu. Są stacjami o pasywnym dostępie do sieci (stacje pasywne); oznacza to, że odpowiadają one na bezpośrednie zapytania. Tego typu zachowanie jest bardzo proste i efektywne (w wersji DP-V0 całkowicie obsługiwany jest przez hardware).

Wszystkie komunikaty w sieci Profibus składają się z 11 bitowych znaków, zawierających: bit startu (stan logiczny niski – 0), 8 bitów danych (nadawanych od najmniej znaczącego bitu), bit parzystości i bit stopu (stan logiczny wysoki – 1). Kolejne znaki komunikatu nadawane są jeden po drugim, bez żadnych przerw między znakami (rys. 56).



Rysunek 56. Pojedynczy znak komunikatu w sieci Profibus DP

Źródło: Werewka, 1997

6.4.3. Ethernet

Popularność klasycznych sieci *Ethernet* sprawiła, że rozwiązania tego typu szybko upowszechniły się również w przypadku systemów wykorzystywanych w przemyśle. Obecnie dostępne są różne implementacje sieci określanych zbiorczo mianem *Ethernetu przemysłowego*, które stopniowo wypierają z wielu aplikacji sieci własnościowe.

Głównymi zaletami, dla których warto wdrażać Ethernet przemysłowy, są jego kompatybilność, efektywność, uniwersalność i elastyczność konfiguracji, a także minimalizacja

kosztów. Dzięki zastosowaniu odpowiednich bram możliwe jest także dołączenie do sieci urządzeń wyprodukowanych przez różnych producentów, a przystosowanych do pracy z konkretnymi standardami sieci polowych. Ethernet jak żaden inny standard sieci przemysłowych oferuje możliwość podłączenia wielu tysięcy urządzeń za pomocą tej samej sieci. Ethernet oferuje także nieporównywalnie większe, stale rosnące prędkości transferu, jednocześnie stając się coraz tańszym w utrzymaniu. Oprócz standardowego dziś Ethernetu 100Base-TX, pozwalającego na transfer 100 $Mb \cdot s^{-1}$, niewiele droższe są rozwiązania oparte na technologii gigabitowej.

Obecnie najpopularniejszym protokołem sieci Ethernet jest TCP/IP. Istnieje szereg firmowych rozwiązań opartych o TCP/IP do zastosowań w systemach przemysłowych:
- Ethernet/IP firmy Allen-Bradley;
- Modbus/TCP firmy Schneider Electric;
- Profinet firmy Siemens i wiele innych.

Obecnie istnieje wiele różnych odmian Ethernetu, jego rodzina zawiera przynajmniej pięć różnych struktur szkieletowych, trzy różne techniki arbitrażu dostępu do medium oraz wciąż powiększającą się grupę interfejsów zależnych od nośnika *(ang. Medium Dependent Interface – MDI)*. MDI jest najbardziej widocznym aspektem fizycznej warstwy 802.3, definiuje i opisuje zarówno oczekiwany typ nośnika transmisyjnego, jak i charakterystyki transmisyjne i impedancyjne. Obecnie w przemyśle wykorzystywane są 3 technologie Ethernetu:
- *Ethernet 10* $Mb \cdot s^{-1}$ – jest to kilka specyfikacji, z których każda pracuje z taką przepustowością;
- *Ethernet 100* $Mb \cdot s^{-1}$ – jest to pojedyncza specyfikacja znana również jako *Fast Ethernet*;
- *Ethernet 1000* $Mb \cdot s^{-1}$ – jest to pojedyncza specyfikacja znana również jako *Gigabit Ethernet* ($1Gb \cdot s^{-1}$).

Ethernet 10 $\mathbf{Mb \cdot s^{-1}}$ **– zwykły Ethernet 10Base-T**

Technologia 10Base-T została wprowadzona w roku 1990. Zamiast kabla koncentrycznego jest w niej najczęściej używana tańsza i łatwiejsza w instalacji skrętka nieekranowana (UTP) kategorii 3 lub wyższej. Kabel jest podłączany do centralnego urządzenia zawierającego wspólną szynę. Technologia 10Base-T była początkowo półdupleksowa. Później dodano możliwość pracy w pełnym dupleksie (jednoczesne nadawanie i odbieranie). W technologii 10Base-T wykorzystywane jest kodowanie typu Manchester. Sieć 10Base-T przenosi dane z prędkością 10 $Mb \cdot s^{-1}$ w trybie półdupleksu i prędkością 20 $Mb \cdot s^{-1}$ w trybie pełnego dupleksu.

Ethernet 100 $\mathbf{Mb \cdot s^{-1}}$ **– Fast Ethernet**

Jest to technologia bardzo podobna do 10Base-T, zachowana bowiem została metoda zarządzania dostępem do wspólnego medium transmisyjnego – CSMA/CD. Spowodowało to dość znaczne ograniczenie dopuszczalnej rozpiętości sieci przy dziesięciokrotnym zwiększeniu szybkości transmisji. Zmianie natomiast nie uległ format ramek, ich długość oraz metoda kontroli błędów. Zmieniły się jednak techniki kodowania sygnałów oraz rodzaje mediów, z którymi standard współpracuje. Jeśli chodzi o kodowanie, wyróżniono dwa jego etapy. W pierwszym kroku sygnał jest kodowany za pomocą techniki 4B/5B, czyli czterobitowe ciągi z podwarstwy MAC kodowane są na pięciu bitach. W drugim

etapie wykorzystuje się kodowanie linii, zależne od używanego medium, np. NRZI *(ang. Nonreturn to Zero Inverted)* dla 100Base-FX lub MLT-3 *(ang. Multi Level-Three Levels)* dla 100Base-TX. Standard 100Base-T przewiduje możliwość współpracy z dwoma rodzajami medium transmisyjnego – kabel miedziany typu skrętka lub światłowód.

Ethernet 1000 Mb·s^{-1} – Gigabit Ethernet

Technologia Gigabit Ethernet zapewnia przepływność 1Gb·s^{-1} oraz kompatybilność z urządzeniami sieciowymi Ethernetu i Fast Ethernetu. Gigabitowy Ethernet używa ramki IEEE 802.3/Ethernet oraz metody dostępu do medium CSMA/CD.

Ponadto specyfikacja ta jest w pełni kompatybilna z wcześniejszymi 10Base-T i 100Base-T. Standard zapewnia transmisje w trybie pełnego dupleksu między przełącznikami oraz między przełącznikami a stacjami sieciowymi, a także pracę w trybie półdupleksu dla połączeń współdzielonych przy użyciu regeneratorów. W Gigabit Ethernecie stosuje się najczęściej wielomodowy kabel światłowodowy o maksymalnej długości do 500 metrów, jednomodowy kabel światłowodowy o maksymalnej długości do dwóch kilometrów oraz kabel miedziany o długości do 25 metrów. Opracowana przez IEEE specyfikacja 1000Base-T wspiera użycie kabli 5 kategorii lub kategorii 5a dla sieci gigabitowych. Jest to możliwe dzięki pewnym zmianom w sygnalizacji. Kabel 5 kategorii jest najczęściej nieekranowaną skrętką, zawierającą cztery pary przewodów. Gigabit Ethernet wykorzystuje wszystkie z tych przewodów. Pod wieloma względami Gigabit Ethernet na kablu 5 kategorii jest prostszy w działaniu niż Ethernet 10 lub 100 Mb·s^{-1}. Specyfikacja 1000Base-T zapewnia automatyczną negocjację charakterystyk łącza, w tym automatyczną korekcję przesłuchów w kablu.

Ramka – z pominięciem preambuły i SFD – może mieć rozmiar od 64 (6 + 6 + 2 +46 + 4) do l518 bajtów (6 + 6 + 2 + l 500 + 4) (tab. 16).

Tabela 16. Ramka danych w sieci Ethernet

Preambuła	SFD	Adres docelowy DA	Adres źródłowy SA	Długość/ Typ	Dane	FCS
7 B	1 B	6 B	6 B	2 B	46 – 1500 B	4 B

Źródło: Kalista, 2012

Preambuła – naprzemienny ciąg bitów l i 0, informujący o nadchodzącej ramce. Najczęściej nie jest on włączany do wielkości ramki. Uznawany jest za część procesu komunikacji. Składa się z 7 bajtów:

- SFD – pole startu ramki *(ang. Start of Frame Delimiter)* – bajt kończący preambułę o postaci: „10101011", zawsze jest zakończony dwoma bitami 1;
- Adres docelowy DA *(ang. Destination Address)* – pole adresu docelowego składa się z 6 bajtów. Informuje o tym, która stacja powinna odebrać ramkę;
- Adres źródłowy SA *(ang. Source Address)* – identyfikuje stację nadawczą, składa się z 6 bajtów;

– Długość/Typ *(ang. Type/Length Value)* – określa długość pola danych oraz typ ramki, składa się z 2 bajtów.

Dane – minimalna długość ramki (bez preambuły i pola startu) musi wynosić 46 bajty, zatem jeśli pole danych jest mniejsze niż 46 bajty, zostaje ono wydłużone przez dodanie w polu rozszerzenia odpowiedniej liczby oktetów.

FCS (ang. Frame Check Sequence) – zawiera 4 bajty kontrolne *(ang. Cyclic Redundancy Check – CRC)* wygenerowane przez interfejs nadający i sprawdzane przez odbierający. Na ich podstawie protokół sprawdza, czy dane nie zostały uszkodzone.

Wykorzystywany w sieci Ethernet protokół TCP/IP jest pakietem najbardziej rozpowszechnionych protokołów komunikacyjnych współczesnych sieci. Następca protokołu NCP. Najczęściej obecnie wykorzystywany standard sieciowy, stanowiący podstawę współczesnego Internetu. Nazwa pochodzi od dwóch najważniejszych jego protokołów: TCP oraz IP. TCP/IP jest standardem komunikacji otwartej. Otwartość oznacza tu możliwości komunikacji między dowolnymi typami urządzeń, bez względu na ich fizyczną różnorodność. Protokoły te mają następujące cechy charakterystyczne:

– dobrą odtwarzalność po awarii;
– możliwość dodawania nowych sieci bez przerywania pracy istniejących;
– wysoki współczynnik korekcji błędów;
– niezależność od platformy;
– mały stopień obciążenia danych własnymi strukturami;
– dużą wydajność.

Protokół TCP *(ang. Transmission Control Protocol)* jest zorientowany połączeniowo *(ang. Connection-Oriented)*, czyli wymaga potwierdzenia odbioru każdej ramki. Ma to istotne znaczenie, jeśli łącza są słabej jakości, ponieważ w przypadku błędów złe ramki są retransmitowane. Warstwa transportowa zapewnia dostarczenie danych między dwoma punktami, w tej warstwie następuje analiza danych oraz sprawdzenie, czy nie ma potrzeby retransmitować danych ponownie. TCP zapewnia przekazanie danych do odpowiedniego procesu pracującego w warstwie aplikacji w takiej samej postaci, w jakiej zostały one wysłane. Protokół TCP identyfikuje połączenia procesów warstwy aplikacji za pomocą numeru portu – 16 bitowej liczby całkowitej. Proces może transportować dane poprzez specjalny interfejs zwany *gniazdem*. W celu nawiązania komunikacji, dwa procesy – nadający i odbierający dane – muszą otworzyć gniazda. Zestawienie adresu źródłowego i numeru portu oraz adresu docelowego wraz z numerem portu nazywa się *asocjacją*.

IP *(ang. Internet Protocol)* – protokół internetowy zawiera informacje na temat adresów internetowych oraz informacje kontrolne, umożliwiające przesyłanie pakietów. IP jest odpowiedzialny za następujące funkcje: organizowanie bezpołączeniowej transmisji danych najlepszą trasą, podział danych na mniejsze części *(ang. Minimum Transmission Unit – MTU)* z ponownym ich składaniem. Istnieją dwie wersje protokołu IP: IPv4 i IPv6. Różnica pomiędzy tymi wersjami polega na adresowaniu stacji. W wersji 4 przeznaczono 32 bity na adres stacji, w nowszej – wersji 6 przewidziano 128 bitów.

6.4.4. CAN

Technologia CAN *(ang. Controller Area Network)* została opracowana w latach osiemdziesiątych przez firmę Bosch na potrzeby komunikacji w pojazdach i jest standardem

multipleksowanej magistrali szeregowej. Pierwotnym celem jej istnienia było poprawienie funkcjonalności i uproszczenie złożonego okablowania stosowanego do przesyłania danych pomiędzy urządzeniami elektronicznymi w samochodach. Działanie CAN, w odróżnieniu od innych systemów magistralowych, opiera się na mechanizmie identyfikacji wysyłanych wiadomości zgodnie z ich zawartością oraz adresem węzła nadawczego lub odbiorczego. Komunikacja ma charakter rozgłoszeniowy, a węzły sieci odbierają i przetwarzają wiadomości, bazując na ich ważności oraz priorytecie. Tego typu adresowanie zwiększa elastyczność systemu z CAN, pozwalając na łatwe dodawanie nowych urządzeń do istniejących sieci bez konieczności stosowania dodatkowych urządzeń lub modyfikacji oprogramowania. Powyższe cechy sprawiły, że magistrala przeznaczona dla pojazdów znalazła zastosowanie w przemyśle, a specyfikacja została objęta normą ISO 11898.

Za pomocą magistrali CAN łączy się urządzenia sterujące z wykonawczymi, jak też stosuje się ją do zestawiania komunikacji pomiędzy czujnikami, sterownikami i układami rejestrującymi. Rosnąca popularność CAN sprawia, że prowadzone są liczne prace mające na celu zwiększenie wydajności tej magistrali i ułatwienie jej implementacji w przemyśle.

CAN wykorzystywana jest m.in. w zakładach przemysłowych, w automatyce budynkowej oraz magazynowaniu. Sieci tego typu znajdują zastosowanie w automatyzacji ciągów produkcyjnych, systemów maszyn, drukarek, pieców, chłodni i in.

Analizując standard CAN zgodnie z siedmiopoziomowym modelem odniesienia OSI, można w nim wyróżnić warstwę łącza danych oraz warstwę fizyczną. Ostatnia jest jednym z krytycznych elementów sieci CAN, musi bowiem charakteryzować się dużą odpornością na zakłócenia występujące w warunkach przemysłowych i szybkością działania. Standardem określającym tę warstwę dla CAN (o dużej szybkości przesyłania danych do 1 $Mb \cdot s^{-1}$) jest ISO 11898-2. Zostały w niej określone funkcje dostępu do nośnika *(ang. Medium Access Unit)* i niektóre własności interfejsów od niego zależnych (ang. *Medium Dependent Interface*). Warstwa fizyczna określona została dla urządzeń komunikujących się magistralą dwuprzewodową (skrętka miedziana).

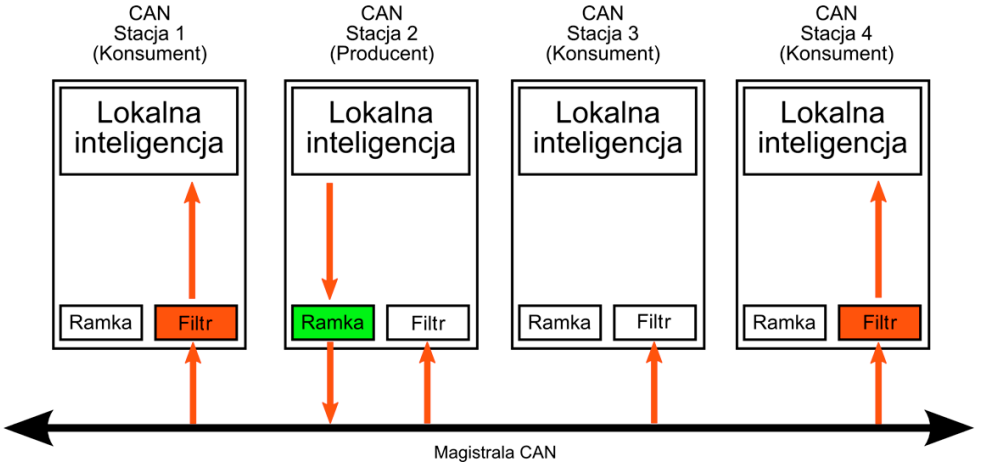
Na poziomie warstwy łącza danych CAN zdefiniowano sposób transmisji danych, którym jest asynchroniczna transmisja szeregowa. W tym przypadku każde urządzenie jest synchronizowane przez wiadomość z innego urządzenia. Elementami, które wpływają na synchronizację, są początek i czas trwania bitu, struktura i czas trwania wiadomości oraz potwierdzenie odbioru *(ang. Handshaking)*. Jeżeli chodzi o strukturę ramki danych, najczęściej stosowanymi obecnie wersjami protokołu CAN są 2.0A (z 11-bitowym identyfikatorem) oraz rozszerzona 2.0B (identyfikator o długości 29-bitów). CAN jest siecią multimaster. Używa CSMA/CD+AMP (dostęp rywalizacyjny z wykrywaniem kolizji z arbitrażem priorytetów wiadomości). Przed wysłaniem wiadomości węzeł CAN sprawdza, czy magistrala jest zajęta, używa również wykrywania kolizji. Pod tym względem jest podobny do Ethernet, jednak kiedy sieć Ethernet wykryje kolizję, oba wysyłające węzły zaprzestają transmisji. Wtedy odczekują losowy odcinek czasu przed ponowną próbą wysłania. To czyni Ethernet bardzo wrażliwym na szybkie załadowanie magistrali. CAN rozwiązuje ten problem bardzo zręcznie na podstawie arbitrażu.

Dowolny kontroler CAN może rozpocząć transmisję, kiedy wykryje stan bezczynności magistrali. To może doprowadzić do sytuacji, że dwa lub więcej kontrolerów zaczną transmitować wiadomość w tym samym czasie. Konflikt jest rozwiązywany w następujący sposób. Transmitujący węzeł monitoruje magistralę w chwili wysyłania. Jeżeli wykryje

poziom dominujący, kiedy sam wysyła poziom niski, to natychmiast opuszcza proces arbitrażu i staje się odbiornikiem. Arbitraż jest dokonywany na całym polu do tego przeznaczonym i kiedy to pole zostało wysłane, dokładnie jeden nadajnik zostaje na magistrali. Węzeł ten kontynuuje transmisję, jak gdyby nic się nie stało. Inne potencjalne nadajniki będą mogły próbować retransmitować swoje wiadomości, kiedy magistrala stanie się dostępna, nie ma strat czasowych na proces arbitrażu.

W CAN nie ma wyraźnie określonego adresu w wiadomości. Każdy kontroler CAN może śledzić cały ruch na magistrali i używając kombinacji sprzętowych, filtrów, oprogramowania decydować, czy wiadomość jest ważna, czy nie. Zamiast pojęcia adresu, zawartość wiadomości jest określana przez identyfikator obecny gdzieś w wiadomości. Mówi się, że wiadomości CAN są adresowane zawartością (rys. 57).

Zawartość pola arbitrażu jest używana do określenia priorytetu wiadomości. Wszystkie kontrolery CAN powinny również używać całego pola arbitrażu jako klucza w procesie sprzętowej filtracji. Standard nie określa, czy pole arbitrażu musi być używane jako identyfikator wiadomości, tym niemniej jest bardzo powszechnie używane.



Rysunek 57. Adresowanie i identyfikacja wiadomości

Źródło: Werewka, 1997

Obsługa błędów jest wbudowana w protokół CAN i odgrywa w nim ogromną rolę, ma na celu wykrywanie błędów w wiadomościach na magistrali CAN, aby nadajnik mógł retransmitować błędną wiadomość. Każdy kontroler CAN może próbować wykryć błędy w wiadomości. Jeżeli błąd zostanie wykryty, węzeł wykrywający wyśle flagę błędu, która zakłóci ruch na magistrali. Pozostałe węzły wykryją błąd spowodowany przez flagę błędu i podejmą odpowiednie działania.

Stosowanie technologii CAN w przemyśle wiąże się ze spełnieniem innych wymogów niż w przypadku pozostałych typowych aplikacji. W zastosowaniach w pojazdach, które nie są związane z bezpieczeństwem, wykorzystuje się zwykle sieci CAN o niskiej prędkości przesyłania danych (poniżej 500 $kb \cdot s^{-1}$). Jest to często zbyt mało w przypadku sieci

przemysłowych, gdzie zwykle wymagane są prędkości w granicach 1 $Mb \cdot s^{-1}$, a nawet więcej. W zastosowaniach przemysłowych dane przesyłane są dodatkowo w trudnych warunkach środowiskowych z racji występujących zaburzeń elekromagnetycznych, których źródłem są maszyny elektryczne. Dodatkowo samo okablowanie urządzeń przemysłowych komunikujących się w standardzie CAN, jest w przypadku linii produkcyjnych stosunkowo długie, co ogranicza prędkości przesyłania danych.

Według standardu ISO 11898 długości magistrali CAN może wynosić maksymalnie 40 metrów (oraz po 30 cm dla rozgałęzień). Wartość ta może być zwiększona poprzez odpowiednią rozbudowę sieci; przykładowo dwie magistrale mogą zostać połączone w celu ich przedłużenia urządzeniem powtarzającym (tzw. *repeater*). Składają się na niego dwa urządzenia nadawczo-odbiorcze, posiada on wbudowane układy do obsługi protokołu CAN oraz zwykle układy zabezpieczające *repeater* przed uszkodzeniem.

6.4.5. DeviceNet

Protokół komunikacyjny DeviceNet został opracowany przez firmę Allen-Bradley (obecnie Rockwell Automation). Bazuje on na specyfikacji sieci CAN 2.0A, z której została zaczerpnięta warstwa fizyczna oraz warstwa łącza danych. DeviceNet jest dedykowany do łączenia w strukturę sieciową kontrolerów przemysłowych z urządzeniami wejścia/wyjścia (zdalnymi stacjami wejść/wyjść cyfrowych, analogowych), które stanowią interfejs pomiędzy systemem, a obiektem sterowania. Mogą to być także pojedyncze czujniki podłączone bezpośrednio do magistrali komunikacyjnej sieci DeviceNet i dostarczające binarnej informacji o obiekcie, jak również urządzenia wykonawcze pozwalające na realizację sterowania obiektem.

Możliwość zasilania urządzeń sieciowych bezpośrednio z magistrali komunikacyjnej upraszcza w znaczący sposób budowę systemu rozproszonego. Dzięki temu proste urządzenia, jak sensory o niskim poborze prądu (maksymalnie 8 lub 16A dla wszystkich abonentów sieci w zależności od zastosowanego okablowania), nie wymagają dodatkowego źródła zasilania.

Sieć DeviceNet jest bardzo popularnym standardem regulowanym przez organizację ODVA *(ang. Open DeviceNet Vendors Association)*. Urządzenia z obsługą sieci DeviceNet produkuje ponad 700 firm na całym świecie. Na popularność standardu istotny wpływ ma jego „otwartość", co oznacza, że sprzedawcy nie są zmuszeni do zakupu sprzętu ani oprogramowania, czy też do uiszczania opłat licencyjnych, aby podłączać urządzenia do systemu.

Przy użyciu sieci DeviceNet może być realizowana komunikacja „peer to peer", Multi-Master oraz Master-Slave. Sieć ta jest siecią typu Producent-Konsument. W tego typu sieciach występują urządzenia będące „producentami" wartości zmiennych, transmitowanych z użyciem magistrali komunikacyjnej (na przykład położenie zaworu, przepływ itp.), oraz „konsumentami" tych wartości. Każda zmienna musi mieć dokładnie jednego producenta oraz może mieć wielu konsumentów. Do magistrali tej sieci może być podłączonych do 64 abonentów.

W sieci DeviceNet wyróżnia się ponadto co najmniej jednego abonenta, często nazywanego Masterem sieci. Abonent ten jest odpowiedzialny za nadzorowanie procesu wy-

miany informacji poprzez odpowiednie odpytywanie pozostałych abonentów sieci i odsługiwanie żądań komunikacji przez nich wysyłanych.

Magistralę komunikacyjną sieci DeviceNet stanowią dwie skręcone w ekranie pary przewodów. Jedna z par przewodów jest wykorzystywana do komunikacji pomiędzy urządzeniami. Druga para stanowi źródło zasilania abonentów sieci. Maksymalna długość magistrali komunikacyjnej jest uzależniona od częstotliwości jej pracy i wynosi do:

– 500 m przy prędkości transmisji 125 $kb \cdot s^{-1}$;
– 250 m przy prędkości transmisji 250 $kb \cdot s^{-1}$;
– 125 m przy prędkości transmisji 500 $kb \cdot s^{-1}$.

Na magistrali komunikacyjnej mogą występować odgałęzienia. Maksymalna długość pojedynczego odgałęzienia wynosi 6 m. Suma długości wszystkich odgałęzień jest uzależniona od częstotliwości pracy magistrali. Na końcach magistrali umieszcza się terminatory.

Komunikacja w sieci DeviceNet opiera się na transmisji komunikatów. Komunikaty te dzielą się na dwie grupy: *komunikaty explicit* oraz *komunikaty implicite (ang. Explicit Messages oraz Implicit Messages)*. Komunikaty *explicit* są komunikatami typu zapytanie-odpowiedź i służą do przesyłania informacji dotyczących konfiguracji poszczególnych abonentów sieci bądź też danych niekrytycznych czasowo. Nie wszystkie urządzenia obsługują ten typ komunikatów. Komunikaty *implicit* – nazywane często komunikatami I/O, wykorzystywane są do komunikacji w czasie rzeczywistym. Komunikaty I/O mogą być przesyłane w jednym z 4 trybów, przez co można wyróżnić 4 typy komunikatów:

– polled (odpytywanie);
– strobed (strobowane);
– cyclic (cykliczne);
– COS (ang. Change of State).

W zależności od zastosowanych urządzeń sprzętowych jest możliwa komunikacja z danym abonentem sieci w jednym lub większej liczbie trybów. I tak na przykład część informacji danego abonenta może być odczytywana komunikatami typu cyclic, a pozostała część komunikatami typu polled. Należy zaznaczyć, iż nie wszystkie urządzenia sieci DeviceNet obsługują wszystkie 4 typy komunikatów.

6.4.6. DDE

DDE *(ang. Dynamic Data Exchange)* – jest protokołem wprowadzonym w Microsoft Windows 3.x (dostępny też w OS/2 i Mac OS), stanowi formą komunikowania się, wykorzystującą pamięć współdzieloną do wymiany danych pomiędzy aplikacjami. Pozwalał on aplikacjom komunikować się ze sobą w taki sposób, że jeżeli zawartość dokumentu utworzonego w jakiejś aplikacji (np. edytorze tekstów lub arkuszu kalkulacyjnym) została zmodyfikowana, automatycznie ulegał modyfikacji inny dokument, do którego był dołączony ten pierwszy. Typowym zastosowaniem DDE była aktualizacja dokumentu tekstowego, gdy zmieniła się zawartość arkusza kalkulacyjnego dołączonego do tego dokumentu. DDE przeznaczony jest do wymiany informacji pomiędzy różnymi aplikacjami znajdującymi się na tym samym komputerze. Aplikacje te mogą być różne, a przesył pomiędzy nimi odbywa się na zasadzie serwer-klient.

Mechanizm DDE został potem zastąpiony przez sprawniejszy i bogatszy funkcjonalnie mechanizm OLE *(ang. Object Linking and Embedding)*.

Oprogramowanie wykorzystujące system wymiany danych DDE umożliwia użytkownikowi wykonywanie następujących zadań:

– przekazywanie informacji dotyczących procesu m.in. aplikacjom, takim jak MRP (Planowanie wymagań materiałowych, ang. Materials Requirements Planning) czy SPC (Statystyczne Sterowanie Procesem, ang. Statistical Process Control);
– wyświetlanie danych pochodzących z innych aplikacji na ekranach synoptycznych oraz wprowadzanie danych do bazy danych procesu w celu obsługi alarmowania i kreślenia przebiegów czasowych (Tomasik i in., 2012c).

DDE różni się od metody wykorzystującej schowek do przekazywania danych tym, że dane te są uaktualniane bez ingerencji użytkownika. Przy współdzieleniu danych aplikacja, która otrzymuje informacje, jest nazywana klientem, a aplikacja, która dostarcza dane, jest nazywana serwerem. Aplikacja może być klientem DDE, serwerem DDE lub zarówno klientem i serwerem DDE. Składnia wykorzystywana przy przesyłaniu danych jest nazywana adresowaniem DDE.

Każdy program, który wykorzystuje DDE jako formę przesyłania danych, wykorzystuje specyficzne adresy DDE do uzyskania dostępu do danych. Obsługiwanie serwera DDE umożliwia użytkownikowi przesyłanie danych o procesie z jednego programu do innych aplikacji, natomiast obsługiwanie klienta DDE umożliwia użytkownikowi przesyłanie danych z innych aplikacji do aktualnie wykorzystywanego programu. System DDE wspiera wielu producentów oprogramowania umożliwiającego programowanie PLC, ale również pakietów zarządzających bazami danych, oprogramowania naukowo-inżynierskiego, m.in. Matlab, i przede wszystkim producenci oprogramowania SCADA.

Obsługiwanie klienta DDE w systemie umożliwia użytkownikowi wykorzystanie danych z innych aplikacji w bazie danych procesu, warunkiem jest jednak współistnienie obu w systemie operacyjnym, np. MS Windows. Umożliwia to czytanie i zapisywanie informacji do/z adresów DDE. Wykorzystując drivery We/Wy dla DDE oraz adresy DDE przy konfigurowaniu bloków, użytkownik może wprowadzać informację do bazy danych procesu z innej aplikacji, sterownika urządzenia zewnętrznego, wykorzystującego DDE lub innego węzła SCADA. Ponieważ dane te są dostępne w bazie danych, to użytkownik może wykorzystywać je w łańcuchach oraz wykonywać zadania alarmowania i kreślenia przebiegów czasowych.

Wymianę danych pomiędzy narzędziami inicjuje klient. Dzieje się to poprzez rozesłanie komunikatu „DDE-connect" do wszystkich programów działających w środowisku Windows. Komunikat ten jest informacją o rodzaju danych potrzebnych klientowi. Odpowiednio zdefiniowany serwer danych, który dysponuje potrzebnymi informacjami, może odpowiedzieć na nadesłany komunikat i zwrócić potrzebne informacje klientowi. Dostęp do lokalnego serwera DDE jest uzyskiwany za pomocą trzyczęściowego adresu, który nazywany jest: ATI *(ang. Application, Topic, Item)* lub formułą zdalnego dostępu. Jeśli użytkownik korzysta z systemu Excel, adres ten składa się z pól: *aplikacja*, *temat danych* oraz *element danych*. Poniżej zamieszczono ogólny opis poszczególnych części adresu DDE:

– *aplikacja* – nazwa serwera DDE, na którym rezydują dane. W wielu przypadkach programy używają swojej nazwy jako nazwy *aplikacji;*
– *temat* – nazwa grupy danych na serwerze DDE. Dane te mogą być arkuszem kalkulacyjnym lub nazwą pliku zawierającego dane, do których użytkownik chce uzyskać dostęp;

– *element* – konkretne dane, które należy przesłać. Nazwa *elementu* zależy od metody, za pomocą której dane są składowane przez aplikację serwera.

Aby uzyskać dostęp do danych ze skoroszytu programu Excel, należy zastosować następującą składnię:

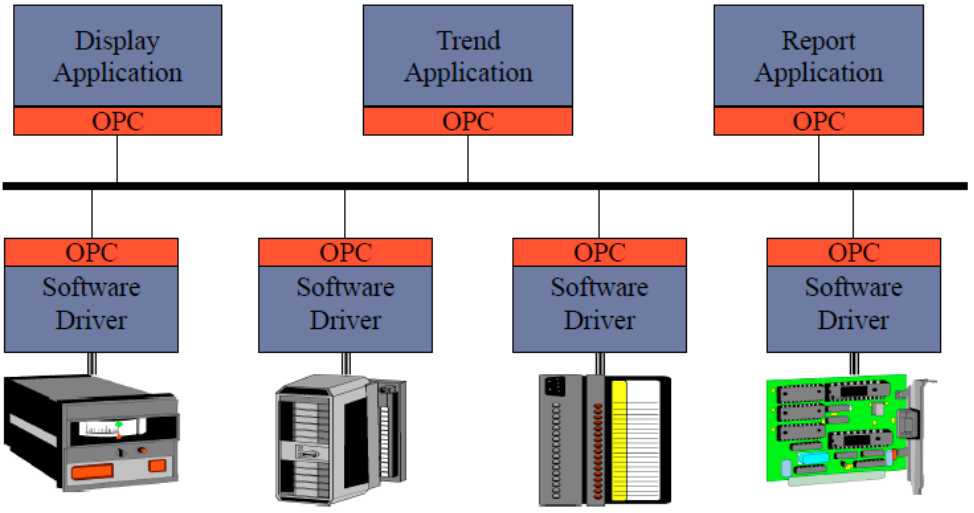
=EXCEL|[PLIK.XLS]ARKUSZ1!R1C1

Program Excel umożliwia tworzenie wielu arkuszy w skoroszycie. W omawianym przykładzie adres DDE udostępnia wartość z pierwszego wiersza i kolumny arkusza o nazwie ARKUSZ1 w skoroszycie PLIK.XLS.

Warunkiem poprawnej pracy DDE jest, aby programy uczestniczące w wymianie danych były cały czas uruchomione. Nie ma specjalnych ograniczeń co do liczby równocześnie prowadzonych konwersacji, tzn. dana aplikacja może być równocześnie klientem dla jednej konwersacji i serwerem dla innej, serwer może dostarczać dane dla wielu klientów, klient może otrzymywać dane od wielu serwerów.

6.4.7. OPC

Serwer OPC *(ang. OLE for Process Control)* jest to standard przemysłowy utworzony przy współpracy wielu wiodących producentów sprzętu i oprogramowania z firmą Microsoft. Standard ten tworzy typowe połączenie dla komunikowania się pomiędzy różnymi urządzeniami kontrolującymi procesy technologiczne. Ich celem jest uniezależnienie oprogramowania monitorującego lub kontrolującego od producenta sprzętu i oprogramowania (rys. 58). Dzięki temu można połączyć w jeden system sterowniki różnych producentów.



Rysunek 58. Komunikacja w standardzie OPC pomiędzy sprzętem i oprogramowaniem

Źródło: Zbrzeżny, 2009

Jego głównym zadaniem jest niezawodna, szybka i efektywna wymiana danych pomiędzy aplikacjami. Server OPC zawiera zwykle wszystkie lub wybrane elementy zgodne z OPC. Standardowo funkcjonalność serwera zapewnia zdolność komunikowania się ze sterownikiem PLC, używa jego zastrzeżonego protokołu oraz udostępnia zbiór standardowych funkcji zgodnych ze specyfikacją OPC. Serwery są dedykowane do protokołów komunikacyjnych sterowników PLC, ale aplikacje wykorzystujące OPC-Serwery są hardwarowo i protokółowo niezależne (Zbrzeżny, 2009).

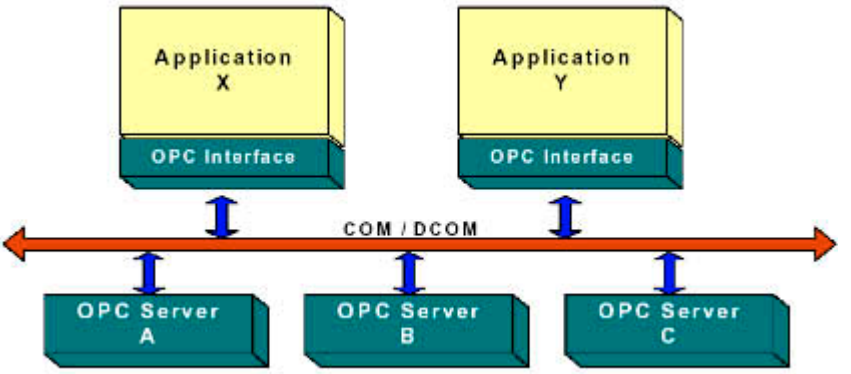
Różne firmy oferują serwery OPC, jednakże komunikacja pomiędzy tymi programami korzystającymi ze standardu OPC nie nastręcza żadnych problemów. OPC klienci korzystają z usług różnych OPC serwerów, które udostępniają im własne dane. Nad całością sprawuje kontrolę organizacja *OPC-Foundation*. Powstała w celu zapewnienia współdziałania, interoperatywności przez opracowywanie i utrzymywanie otwartych specyfikacji, które standaryzują komunikację z wykorzystaniem:

- danych o charakterze aktualnych danych procesowych;
- alarmów i systemów zdarzeniowych, danych historycznych;
- danych wsadowych;
- urządzeń różnych producentów, takich jak czujniki, urządzenia wykonawcze, PLC, RTU (*ang. RemoteTerminal Unit*), DCS (*ang. Distribiuted Control System*), HMI, systemy obsługi trendów, systemy alarmów i wielu innych – w wielu gałęziach przemysłu.

Klasyczny system sterowania składa się z obiektu i sterownika PLC. Sterownik czyta zmienne procesowe i steruje obiektem poprzez zmienne sterujące. Jest odpowiedzialny za kontrolowanie i monitorowanie prawidłowego przebiegu czynności procesów technologicznych. Bardzo często w wielu systemach sterowania zachodzi potrzeba udostępnienia części danych procesowych czy sterujących zawartych w sterowniku PLC na zewnątrz do operatora lub przekazania nowych nastaw do sterownika. Sterownik PLC realizuje sterowanie obiektem we współpracy z komputerem przemysłowym. Połączony jest poprzez łącze transmisyjne, któremu udostępnia swoje dane i otrzymuje polecenia. W komputerze może być uruchomiony program wizualizacyjny typu SCADA, realizujący wszystkie funkcje potrzebne operatorowi.

Powstanie OPC-Serwerów wynikło m.in. z potrzeby uproszczenia sposobu obsługi i komunikowania się sterowników PLC różnych firm z programami wizualizacyjnymi typu SCADA pracującymi na PC.

Standard OPC wykorzystuje mechanizm wymiany informacji DCOM (DCOM – Distributed Component Object Model). OPC klient może być zainstalowany na tym samym PC, co OPC Server (local), lub mogą być zainstalowane na oddzielnych PC (remote). W lokalnej klienckiej aplikacji DCOM jest używany jako mechanizm komunikacyjny pomiędzy klientem a serwerem (rys. 59).



Rysunek 59. Mechanizm komunikacji pomiędzy OPC klient i OPC Serwer

Źródło: Zbrzeżny, 2009

Chociaż OPC zostało zaprojektowane dla udostępniania danych ze sterowników PLC, to dzięki swojej uniwersalności aktualne zastosowania są bardzo szerokie. Wykorzystując mechanizmy sieciowe, które udostępniają dane z serwera sieciowego, można budować systemy hierarchiczne czy kaskadowe.

Inna zaleta OPC pojawia się, kiedy system SCADA jest OPC klientem i OPC serwerem. Może wtedy dostarczać dane z PLC lub dystrybuować je innemu systemowi, np. dużemu systemowi informacyjnemu czy do systemu kompletnej kontroli produkcji.

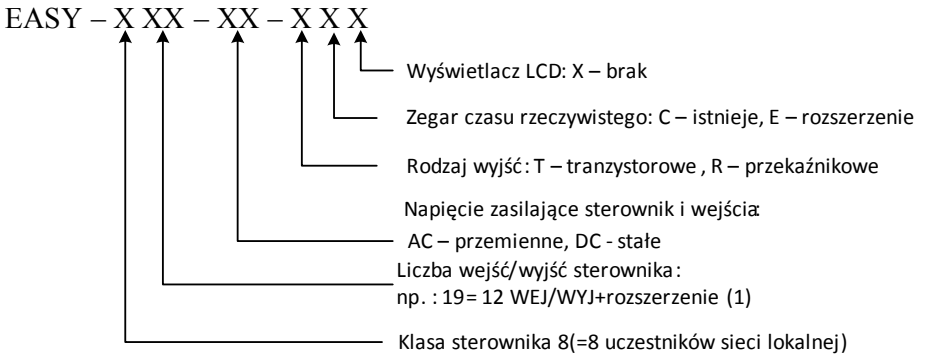
Ponadto do wielu zalet technologii OPC można zaliczyć m.in.:

– standaryzację komunikacji i wymiany danych przemysłowych;
– dużą uniwersalność i skalowalność rozwiązań;
– znaczne obniżenie kosztów integracji dużych systemów przemysłowych.

CZĘŚĆ DRUGA
– EKSPLOATACJA STEROWNIKÓW PLC

1. INSTALOWANIE STEROWNIKÓW PLC

Producenci oferują bogatą gamę sterowników kompaktowych i modułowych. Każdy z nich cechuje unikalna nazwa zawierająca istotne informacje na temat zasilania, rodzaju wejść/wyjść, pamięci itp., w dokumentacji można znaleźć klucz do rozkodowywania tej nazwy. Przykładowy algorytm kodowania dla sterowników Eaton-Moeller zamieszczono na rysunku 60.



Rysunek 60. Algorytm kodowania właściwości PLC na przykładzie sterowników Eaton-Moeller

1.1. Wytyczne ogólne – montaż

W czasie montażu i wykonywania oprzewodowania należy kierować się następującymi zasadami:

- przestrzegać wszystkich ważnych i obowiązujących norm;
- stosować przewody o właściwym przekroju dla danej wartości prądu;
- nie dokręcać zbyt mocno zacisków (przeważnie maksymalny moment dokręcania to 0.6 Nm);
- przewody należy prowadzić najkrótszą drogą, jeżeli sytuacja wymaga dłuższych przewodów – stosować przewody ekranowane;
- przewody należy prowadzić parami: przewód neutralny lub zerowy razem z przewodem fazowym lub sygnałowym;
- upewnić się, czy przewody posiadają odpowiednie zabezpieczenie przed wyrwaniem;
- oddzielać przewody prądu przemiennego od przewodów niskiego napięcia – sygnałowych.

Do podłączania zasilania oraz sygnałów wej/wyj najczęściej należy używać następujących przewodów:

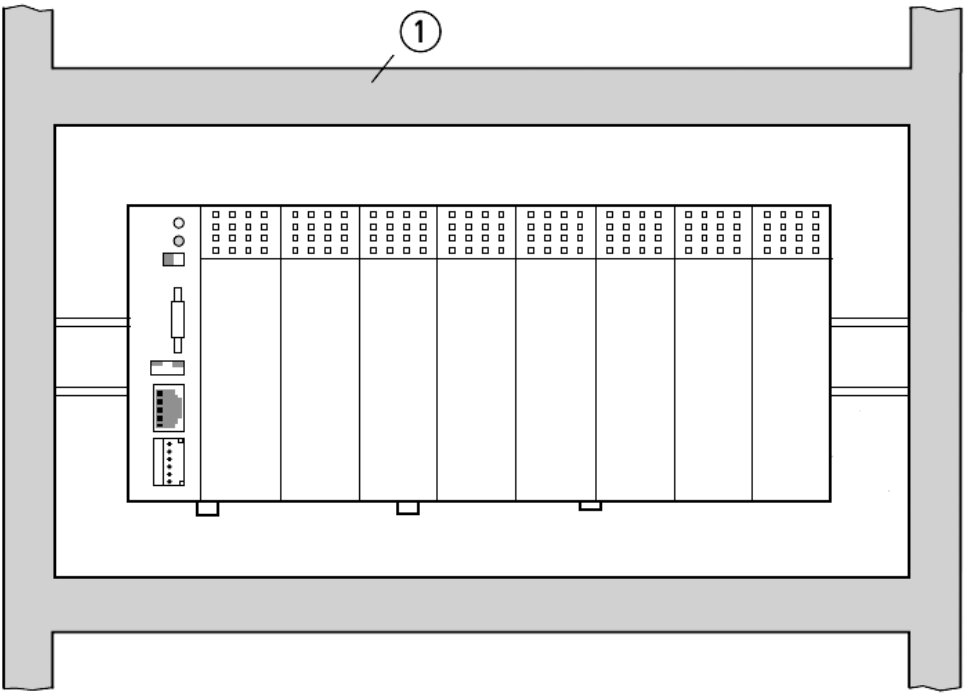
- 1 x 2,5 mm^2;
- 2 x 1.5 mm^2 w przypadku dołączania do jednego zacisku dwóch przewodów.

Należy sprawdzić, czy izolacja dołączonych do systemu przewodów jest prawidłowa. Przestrzegać określonych krajowych standardów, aby zabezpieczyć sterownik przed kontaktem z elementami znajdującymi się pod napięciem.

Instalację należy przeprowadzać w następującej kolejności:
– montaż;
– podłączenie wejść;
– podłączenie wyjść;
– wykonanie połączeń sieci NET (jeżeli konieczne);
– podłączenie napięcia zasilającego.

Sterowniki mikroprocesorowe należy tak zamontować w szafie sterowniczej, rozdzielnicy instalacyjnej lub w obudowie, aby podłączenia napięcia zasilającego i zaciski przyłączeniowe były podczas pracy chronione przed bezpośrednim dotykiem.

Większość sterowników posiada możliwość założenia na szynę montażową zgodną z normą DIN EN 50022 lub mocowanych do niej za pomocą dodatkowych uchwytów (rys. 61). Jeśli nie ma zastrzeżeń, można je montować w położeniu pionowym lub poziomym.



Rysunek 61. Montaż sterownika modułowego na szynie DIN: (1) – korytka przewodowe

Przy włączonym zasilaniu nie wolno prowadzić żadnych prac elektrycznych przy sterowniku mikroprocesorowym.

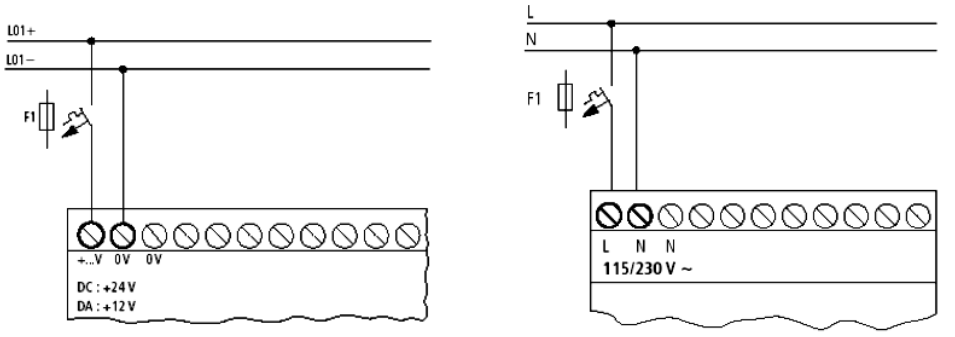
Należy przestrzegać zasad bezpieczeństwa:
– odłączyć instalację;
– stwierdzić brak napięcia;

– zabezpieczyć przed ponownym załączeniem;
– zewrzeć fazy i uziemić;
– osłonić sąsiednie części pozostające pod napięciem.

Jeżeli sterownik będzie zastosowany z dodatkowym rozszerzeniem, przed montażem należy najpierw przyłączyć rozszerzenie (PN-EN 61131-2:2008E).

Aby uniknąć problemu z podłączaniem przewodów do sterownika, należy po stronie zacisków zachować odstęp minimum 3 cm od ścianek lub od sąsiednich urządzeń.

Zalecane jest zabezpieczenie urządzenia w obwodzie zasilającym bezpiecznikiem topikowym (rys. 62). Skoki napięcia zasilającego można wyeliminować poprzez użycie warystora tlenkowego. Warystor należy umieścić między zaciskami L i N przekaźnika. Napięcie pracy warystora musi przekraczać napięcie znamionowe o przynajmniej 20%. Obwody wejściowe najczęściej zabezpieczone są przed podaniem odwrotnej polaryzacji, więc nie ma niebezpieczeństwa uszkodzenia urządzenia, jednakże przed przystąpieniem do pierwszego uruchomienia należy bezwzględnie sprawdzić poprawność polaryzacji.

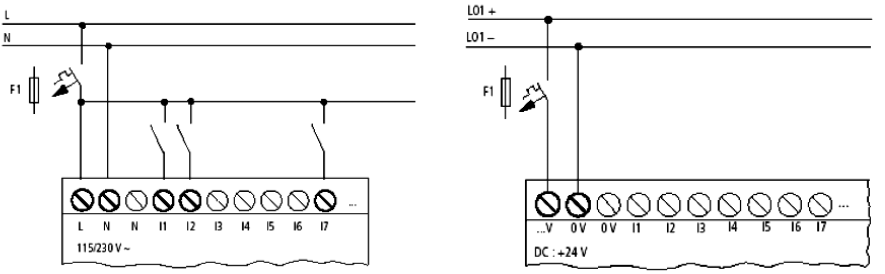


Rysunek 62. Schemat podłączenia zasilania prądem stałym 24 V i zmiennym 230 V

1.2. Podłączanie wejść i wyjść

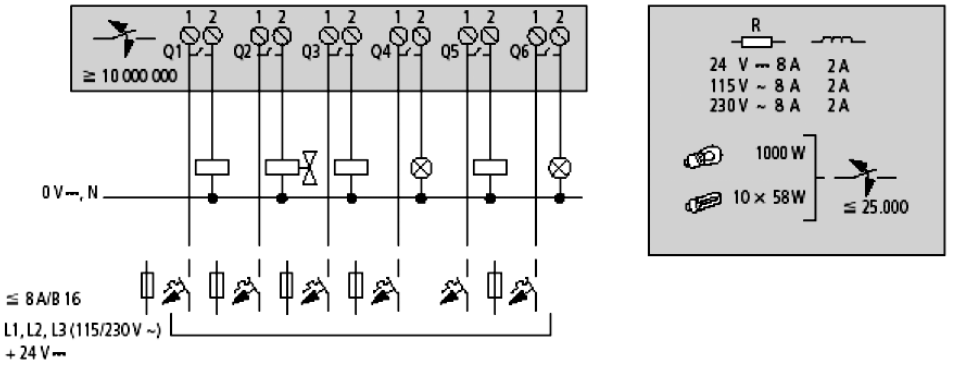
Do zacisków wejściowych sterownika można dołączać styki takie jak przycisk lub łącznik, jak również różnego rodzaju czujniki, detektory, sensory itp. Wejścia sterownika i urządzeń pomocniczych typu AC należy łączyć zgodnie z normami bezpieczeństwa do tych samych faz, do których jest podłączone napięcie zasilające PLC (rys. 63). W przeciwnym razie sterownik może nie rozpoznać poziomu przełączenia lub, z powodu zbyt dużego napięcia, może zostać uszkodzony. Dla modeli zasilanych napięciem stałym nie ma formalnego wymogu zgodności bieguna napięcia wejściowego z zasilaniem. W tych urządzeniach stosować można różnorodne źródła sygnału wejściowego, jednakże należy pamiętać, by ich poziom był zgodny ze specyfikacją techniczną danego sterownika mikroprocesorowego. W celu zapewnienia stabilnej pracy należy stosować zalecenia dotyczące urządzeń i przewodów wejściowych opisane szczegółowo w instrukcji obsługi tych urządzeń.

a) b)



Rysunek 63. Schemat podłączenia wejść dla sterownika: a) AC, b) DC

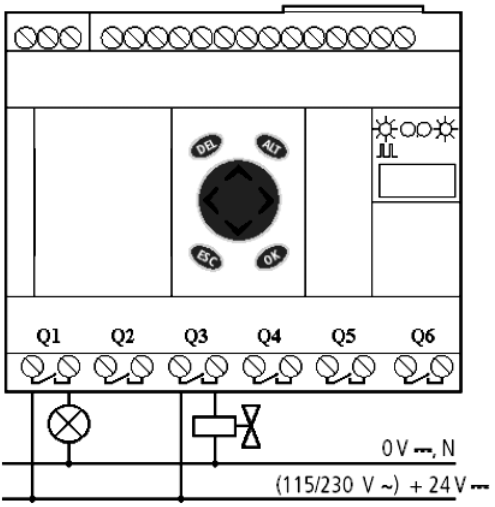
Sterowniki mikroprocesorowe w zależności od rodzaju posiadają wyjścia w postaci *przekaźnikowej* lub *tranzystorowej*. Zarówno jedna jak i druga grupa zaprojektowana została z myślą o różnych zastosowaniach. Urządzenia podstawowe z wyjściami przekaźnikowymi bez potrzeby stosowania dodatkowego osprzętu gotowe są do sterowania urządzeniami prądu stałego oraz przemiennego (rys. 64). Prąd stały powoduje szereg niekorzystnych zjawisk w procesach przełączania, dlatego też, aby nie uszkodzić wyjść przekaźnikowych, producenci stosują napięcie DC do 24 V.



Rysunek 64. *Podłączanie wyjść przekaźnikowych*

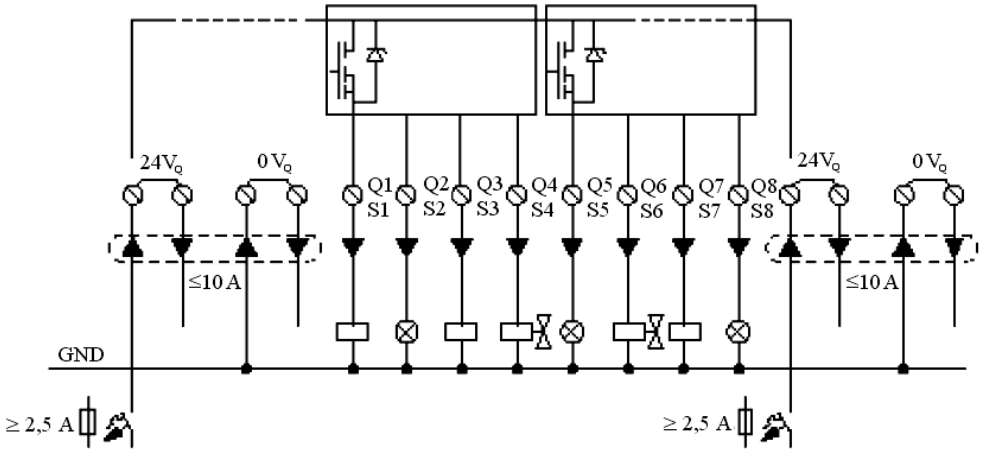
Źródło: Moeller, 2012

Na rysunku 65 zamieszczono przykładowe podłączenie dwóch odbiorników, faza zasilania sterowanych urządzeń nie musi być zgodna z fazą zasilania sterownika (styki przekaźnika zamykają obwód, doprowadzając zasilanie do urządzeń).



Rysunek 65. Przykład podłączenia odbiorników do wyjść przekaźnikowych

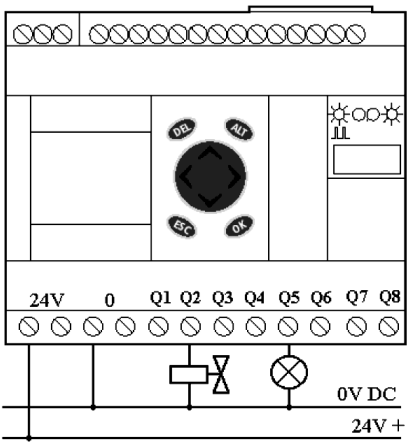
Wyjścia tranzystorowe przystosowane są do współpracy z napięciem 24 VDC, charakteryzują się mniejszą obciążalnością. Można do nich podłączać bezpośrednio urządzenia niepobierające więcej niż 500 mA prądu. Taka konfiguracja nie oznacza wcale ograniczenia zastosowań urządzenia, ponieważ wyjścia tranzystorowe w bardzo prosty sposób mogą sterować zewnętrznymi przekaźnikami czy stycznikami, stąd pośrednio mogą sterować urządzeniami dużej mocy (rys. 66).



Rysunek 66. Podłączanie wyjść tranzystorowych

Źródło: Moeller, 2012

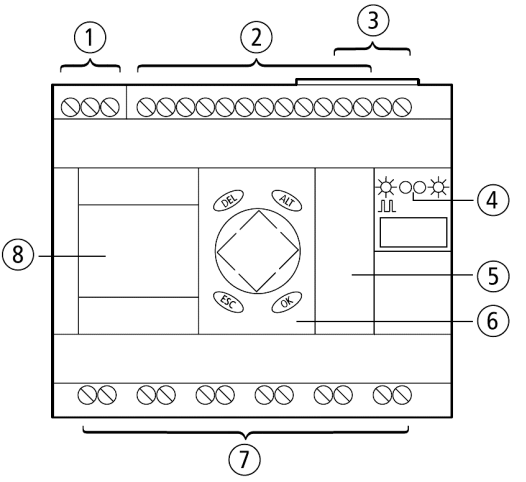
Na rysunku 67 widoczny jest sterownik PLC, do którego wyjść podłączone są odbiorniki. W odróżnieniu od wyjść przekaźnikowych wyjścia tranzystorowe posiadają jeden zacisk przyłączeniowy. W przypadku wyjść przekaźnikowych (rys. 65) lewy zacisk z każdej pary służy do doprowadzenia prądu.



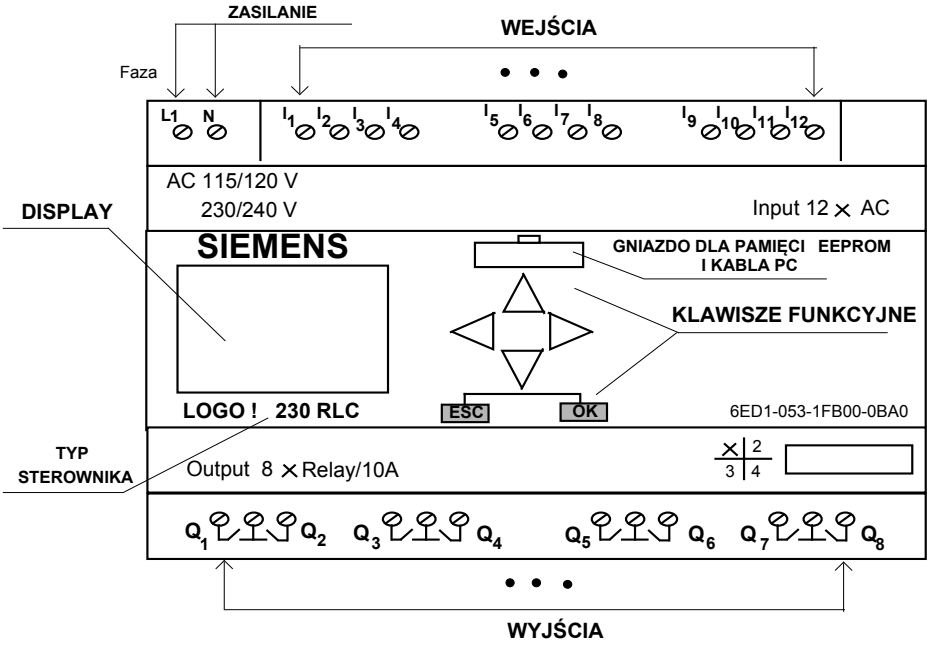
Rysunek 67. Przykład podłączenia odbiorników do wyjść tranzystorowych

Realizacja programu polega na programowym zamykaniu styku danego przekaźnika (wyjścia) i przesyłaniu prądu z lewego do prawego zacisku danego wyjścia. Sterowanie wyjść tranzystorowych polega na programowanym wyzwalaniu potencjału 24 VDC pomiędzy zaciskiem 0 VDC, a jednym z zacisków oznaczonych jako Q.

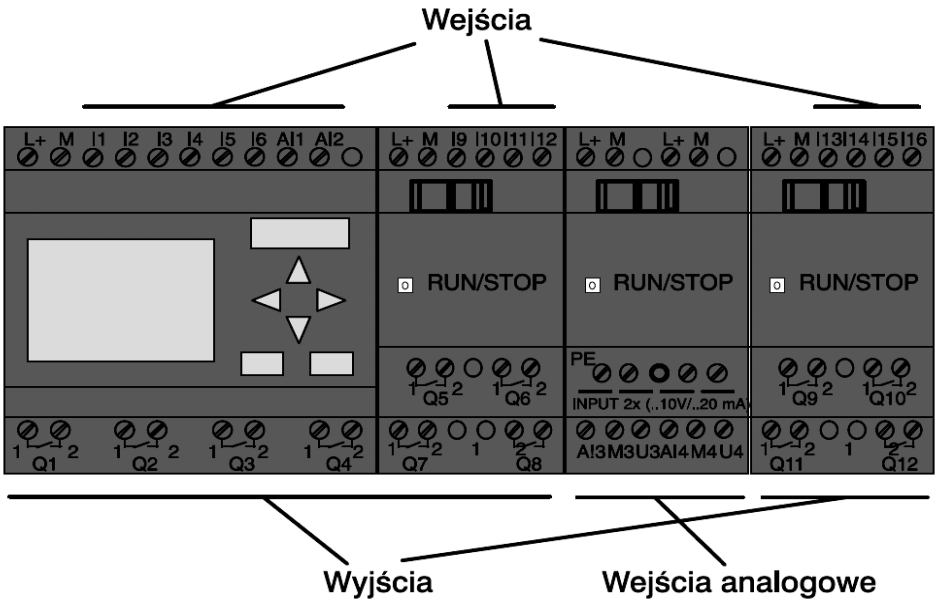
Na rysunkach 68–70 zamieszczono płyty czołowe sterowników kompaktowych z opisem wejść/wyjść, przycisków klawiatury itd. Przykład budowy sterownika modułowego można znaleźć na rysunku 16.



Rysunek 68. Płyta czołowa kompaktowego sterownika PLC: (1) zasilanie, (2) wejścia, (3) podłączenie sieci NET, (4) diody LED (wskaźniki stanu), (5) złącze programatora (PC) lub karty pamięci, (6) klawiatura, (7) wyjścia, (8) opcjonalny wyświetlacz LCD



Rysunek 69. Płyta czołowa sterownika Siemens Logo



Rysunek 70. Płyta czołowa sterownika Siemens Logo – moduły dodatkowe

2. PRZYKŁADY APLIKACJI PROGRAMUJĄCYCH STEROWNIKI

W tym rozdziale przedstawiono wybrane programy z omówieniem metodyki programowania sterowników PLC. Programy dobrano tak, aby omówić trzy najczęściej stosowane języki programowania: *język schematów drabinkowych (LD), język schematów blokowych (FBD)* oraz *język tekstowy – tekst strukturalny (ST)*.

Przez programowanie rozumie się odwzorowanie logiki połączeń układu sterowania w pamięci sterownika. Taki program jest w istocie innym sposobem przedstawienia funkcji działania układu sterowania.

2.1. Program EasySoft – język schematów drabinowych LD

Do omówienia *języka schematów drabinkowych* wybrano program *EasySoft Pro* umożliwiający obsługę i konfigurowanie urządzeń serii easy (Eaton-Moeller), są to urządzenia klasy kompaktowej. Jest to jeden z najlepszych i najprostszych programów umożliwiających tzw. naukę od podstaw programowania sterowników PLC języku LD. W pomocy programu można znaleźć dużo praktycznych wskazówek (przewodnik programowania), przykłady rozwiązań oraz zadania do samodzielnej pracy. Wadą tego programu jest to, że nie oferuje innych języków programowania (Bauerfeind, 2002).

Pierwotnym zadaniem sterowników PLC jest realizacja funkcji wielu styków zwiernych i rozwiernych zgodnie z określoną logiką. Taka logika odpowiada połączeniom szeregowym lub równoległym znanym z układów sterowania przekaźnikowo-stycznikowego.

Dla prawidłowego zaprogramowania sterownika istotne jest rozumienie dwóch terminów: *styk* i *blok funkcjonalny*.

Stykami są wszystkie wejścia (oznaczenie I) i wyjścia (oznaczenie Q) zgodnie z płytą czołową używanej wersji modułu. Terminem tym określa się również stany wejść i wyjść oznaczane przez:

– *lo*, sygnał o poziomie niskim „0” (OFF);
– *hi*, sygnał o poziomie wysokim „1” (ON).

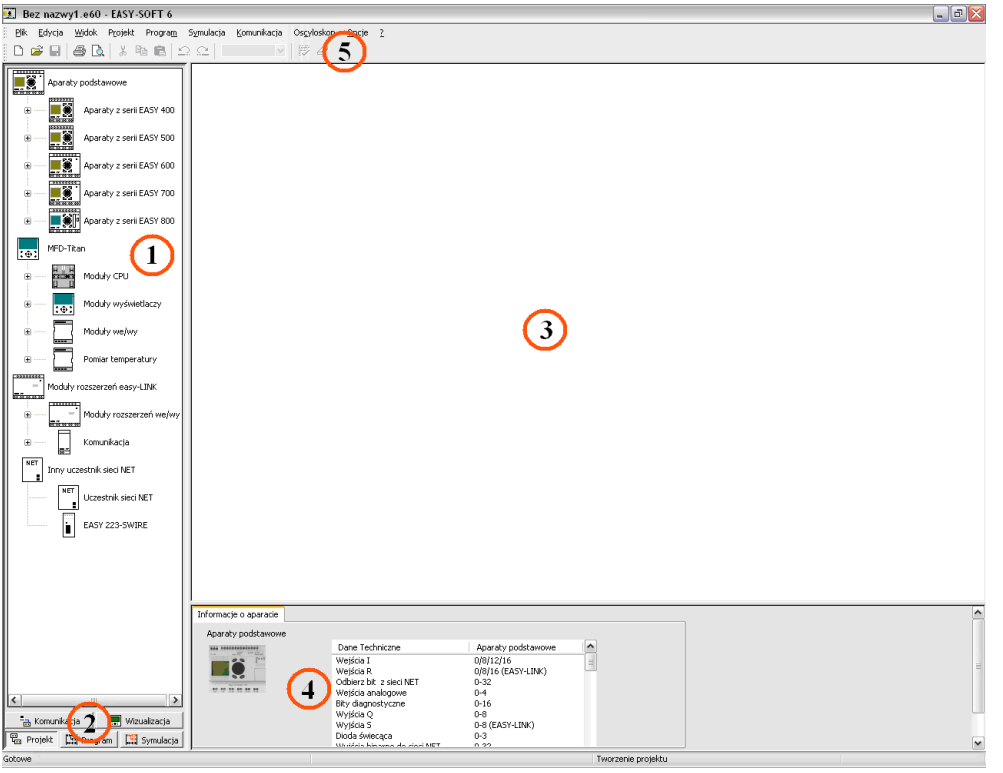
Blok funkcjonalny w easy jest elementem programowym, który przetwarza informację wejściową na informację wyjściową zgodnie ze swoją funkcją działania. Do zestawu funkcji specjalnych należą: przekaźniki czasowe, liczniki, zegary sterujące, komparatory itp.

Najbardziej efektywnym sposobem programowania easy jest programowanie za pośrednictwem komputera PC (istnieje możliwość programowania tych urządzeń z poziomu jego wyświetlacza i przy udziale klawiatury umieszczonej na płycie czołowej). W programie tym można opracować program do dowolnego sterownika easy z uwzględnieniem modułów rozszerzeń i sieci. Programy można również zainstalować w PLC przy pomocy

specjalnych kart pamięci. Kompletny program może być przechowywany w pamięci RAM modułu, karcie pamięci EEPROM (opcja) lub twardym dysku komputera (Moeller, 2006).

Po uruchomieniu programu na ekranie komputera otwiera się okno widoczne na rysunku 71. W domyśle jest to nowy projekt, który w pierwszej kolejności należy nazwać i zapisać. Okno to składa się z następujących elementów:

– menu urządzeń i komponentów sterownika (1), stąd wybieramy model PLC i przenosimy go do okna w pole oznaczone (3);
– zakładek poszczególnych etapów projektu (2), kliknięcie powoduje wybranie zakładki (zakładka aktualnie otwarta nosi nazwę „*Projekt*”);
– okna konfiguracji projektu (4), wyświetlane są tam naprzemiennie informacje o urządzeniu i parametry konfiguracji, które można edytować (nazwa, hasło itp.);
– menu narzędzi (5).



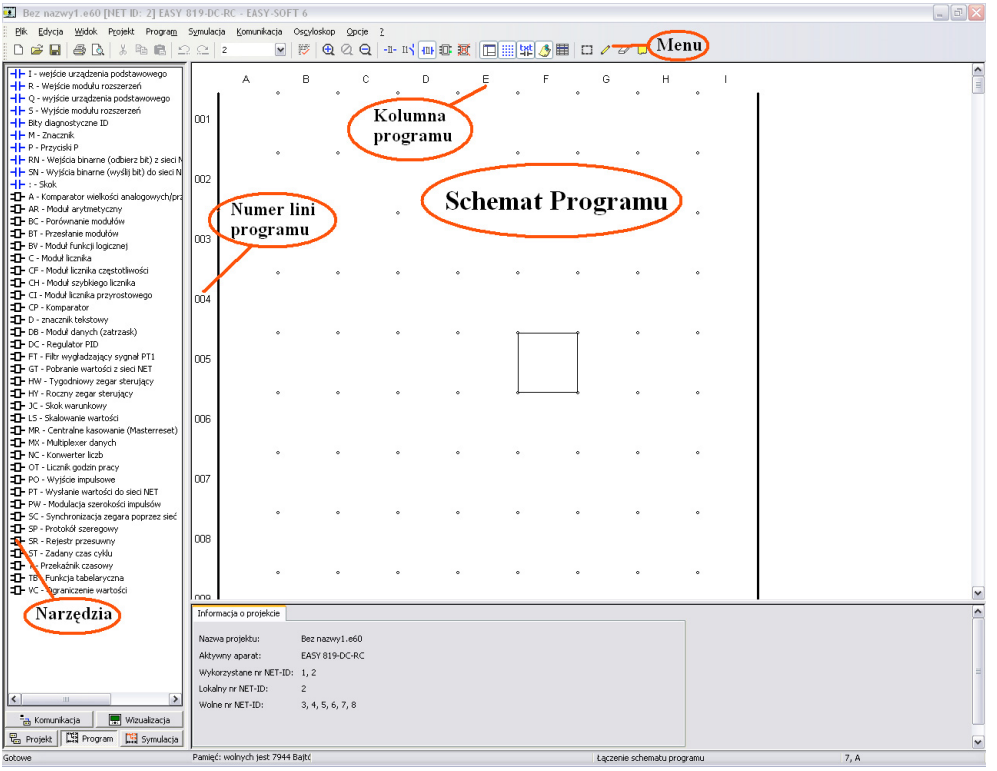
Rysunek 71. Główne okno programu EasySoft Pro

Wraz z otwarciem programu standardowo otwiera się zakładka *Projekt.* Oprócz niej znajdują się tam: *Program* – służy do tworzenia programu sterowania, *Symulacja* – umożliwia testowania programu bez konieczności przesyłania go do sterownika programowalnego (bardzo wygodne rozwiązanie), *Komunikacja* – w tym miejscu można skonfigurować obsługę komunikacji między sterownikiem a komputerem PC oraz pomiędzy połączonymi

w sieć sterownikami. Ostatnią zakładką jest *Wizualizacja* – jeśli sterownik posiada wyświetlacz (seria MFD-Titan), to istnieje możliwość zaprogramowania prostych animacji wyświetlanych na nim.

Kolejnym krokiem po zaprojektowaniu sytemu sterującego (Projekt) jest jego zaprogramowanie, stąd należy przejść do zakładki *Program*.

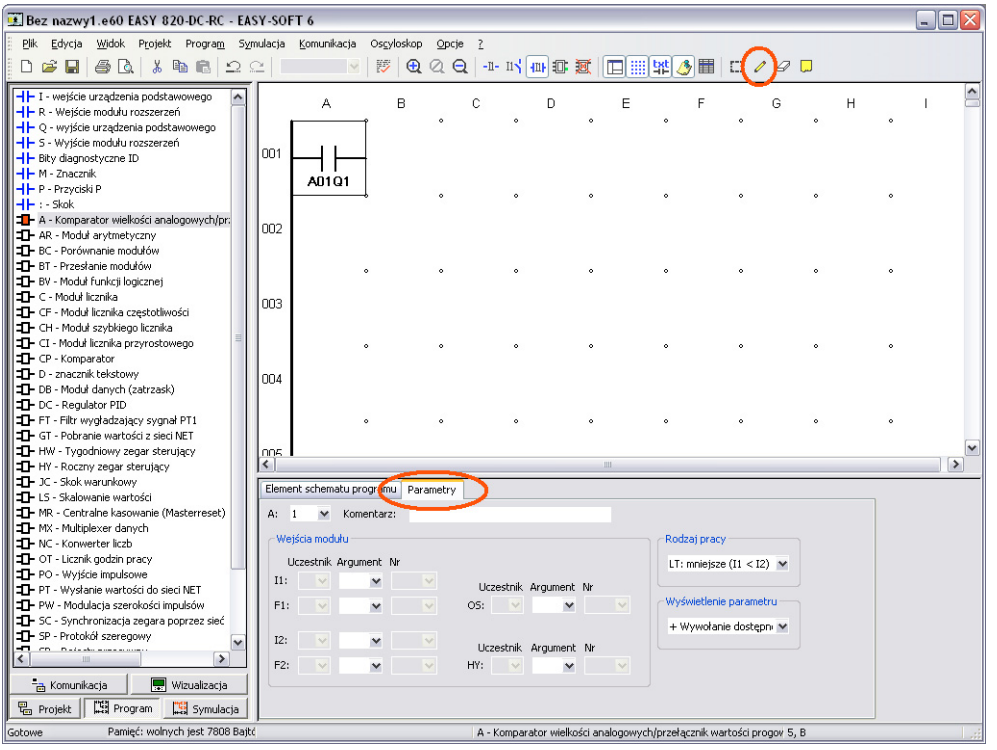
Programowanie w języku schematów drabinkowych polega na przenoszeniu odpowiednich elementów z *Okna Narzędzi* (rys. 72) i umieszczaniu ich w oknie *Schemat Programu* – w odpowiedniej kolumnie oraz linii. Idea programowania drabinkowego polega na tworzeniu szczebli drabinki, przez które popłynie prąd (sygnał) z lewej strony na prawą stronę drabinki. Poszczególne szczeble są więc pojedynczymi algorytmami sterowania, składającymi się na jeden duży algorytm jako program. Kolumny B, D, F i H są zarezerwowane na połączenia poprzeczne (alternatywne).



Rysunek 72. Zakładka „Program”

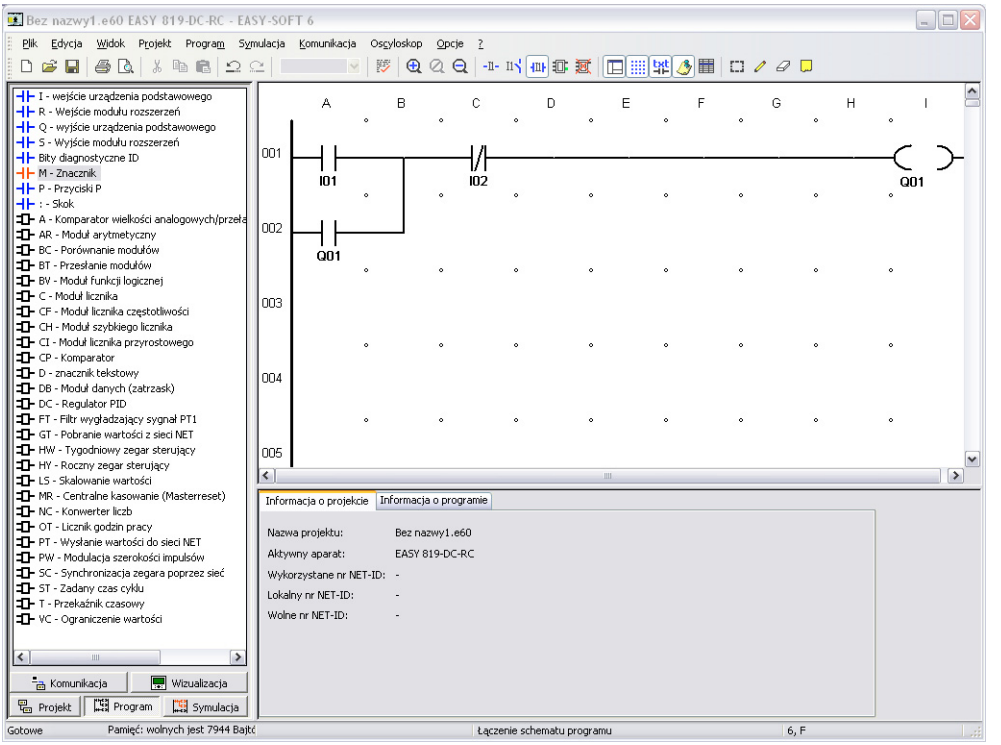
Tworzenie szczebla rozpoczyna się od przeniesienia określonego elementu z biblioteki narzędzi i umieszczeniu go np. w polu A001, tak umieszczony przyjmuje symbol styku. Kolejnym krokiem jest ewentualna zmiana nazwy, ustalenie numeru argumentu oraz ustawienie parametrów, właściwości (po wcześniejszym zaznaczeniu tego elementu) itp. (rys. 73).

Połączenia w jednej linii programu są tworzone automatycznie. Jeśli potrzebne jest poprzeczne połączenie między dwoma lub więcej liniami programu, należy skorzystać z funkcji *Narysuj połączenie*. Po kliknięciu na przycisk na płaszczyźnie roboczej pojawi się pisak. Po wykonaniu poprzecznych połączeń należy deaktywować funkcję rysowania połączeń.



Rysunek 73. Parametryzacja komparatora wielkości analogowych

Na rysunku 74 przedstawiono prosty program realizujący układ uruchamiania z podtrzymaniem pamięci i niezależnymi przyciskami „załącz” I01 oraz „wyłącz” I02. Istota działania jest następująca: przyciskiem I01 (styk zwierny) załączamy układ, prąd płynie z lewej szyny (poprzez styk rozwierny I02) na prawą do cewki Q01. Układ podtrzymania realizuje cewka Q01 umieszczona jako styk w polu A002 (alternatywa dla styku I01). Rozłączenie następuje po wciśnięciu I02.

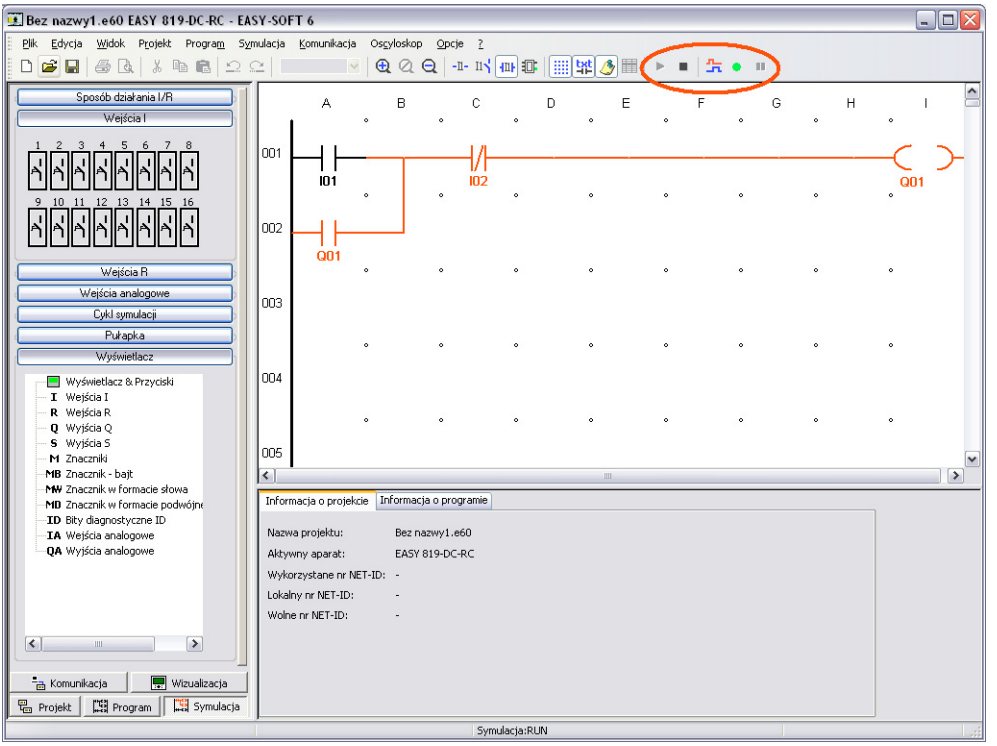


Rysunek 74. Układ uruchamiania z podtrzymaniem pamięci

Tak przygotowany program należy przetestować pod kontem poprawności działania, do tego służy zakładka *Symulacja*. W polu zaznaczonym na rysunku 75 znajdują się przyciski obsługi symulacji, tj.: start, stop, oscyloskop, start rejestracji na oscyloskopie oraz pauza w rejestracji. W oknie narzędzi (po lewej) do dyspozycji znajdują się następujące opcje:

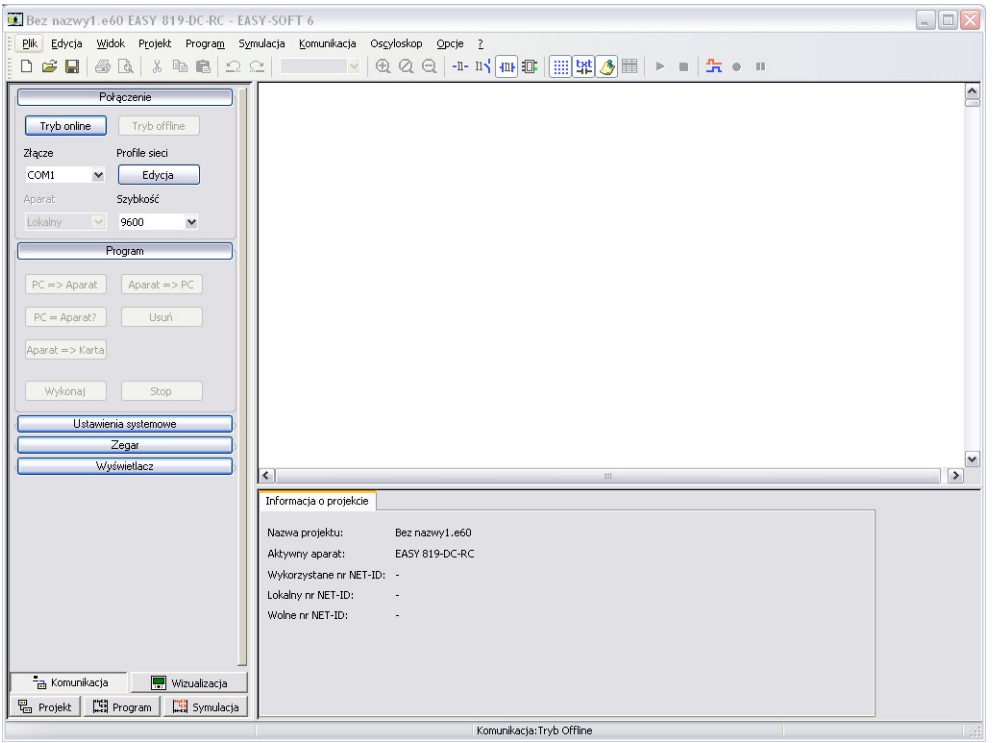
– ustawienie sposobu działania wejść (m.in. kontaktowe – monostabilne, przełączane – bistabilne);
– przełączniki do zadawania dwustanowych binarnych sygnałów wejściowych;
– symulacja wejść analogowych przy pomocy suwaków z regulacją napięcia w zakresie 0 – 10 VDC;
– parametryzacja procesu symulacji;
– zadawanie pułapek w celu wykrycia niestabilności programu;
– kilka wariantów wyświetlania sygnałów wyjściowych łącznie z widokiem ekranu wyświetlacza.

Przepływ sygnału (prądu) przez szczeble odzwierciedla w programie kolor czerwony. Po wciśnięciu przycisku I01 prąd popłynął poprzez I02 do Q01, następnie po jego zwolnieniu prąd płynie już drogą styk Q01, I02 i Q01 do czasu, aż obwód zostanie przerwany stykiem rozwiernym I02.



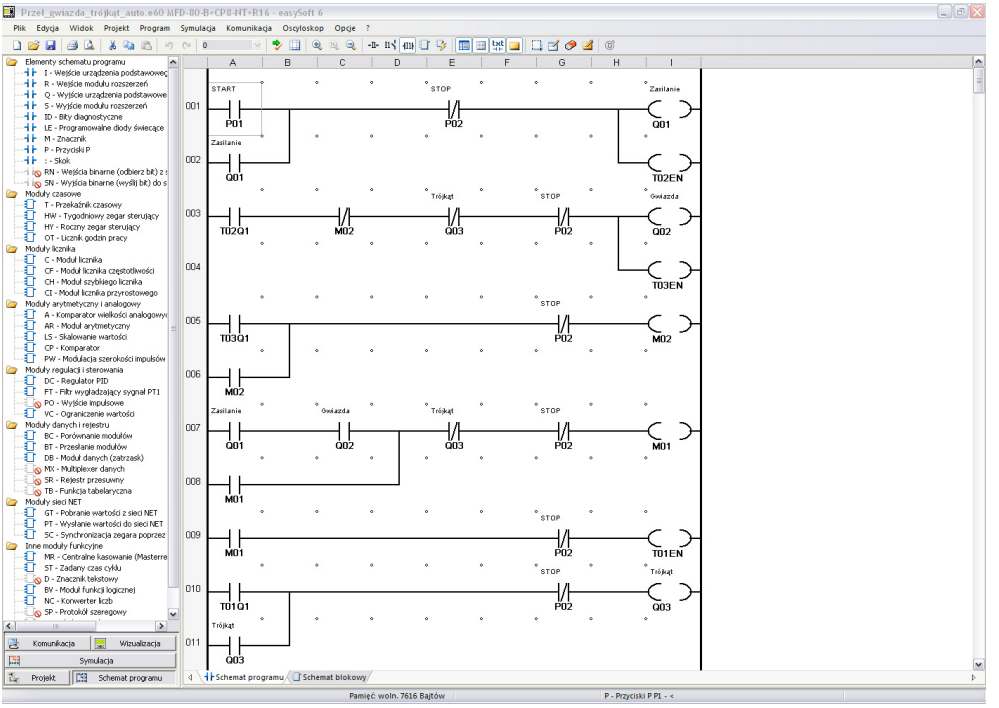
Rysunek 75. *Okno zakładki Symulacja*

Ostatnim etapem jest realizacja komunikacji pomiędzy programatorem (komputer PC), a sterownikiem PLC – wszystkie czynności w tym zakresie realizowane są w oknie *Komunikacja*. Najczęściej w tego typu urządzeniach do komunikacji wykorzystuje się port RS-232 (COM), na rysunku 76 widoczny jest aktywny port. Podstawowym warunkiem komunikacji jest połączenie kablem komputera PC i PLC. Po zrealizowaniu tego kroku należy kliknąć podświetlony *Tryb online,* jeżeli z jakiegoś powodu (uszkodzony kabel, błędna wersja sterownika zdeklarowanego w programie w stosunku do podłączonego) nie nastąpi połączenie – wyświetlony zostanie komunikat błędu. Dalsza praca nie jest możliwa. Z chwilą jednak, kiedy możliwa jest komunikacja, pojawiają się podświetlone przyciski w poniższej zakładce *Program*. Celem zainstalowania (aktualizacji) programu w PLC należy kliknąć „PC=>Aparat". W trybie online możliwa jest aktualizacja zegara czasu rzeczywistego. Z poziomu okna *Komunikacja* możliwa jest również tak zwana *praca w trybie online*. Po wciśnięciu przycisku *Play* (piąty z prawej w górnym pasku narzędzi) na schemacie drabinkowym widoczna jest realizacja programu (pojawiający się kolor czerwony szczebli i elementów na nich umieszczonych sygnalizuje ich załączanie).



Rysunek 76. Zakładka „Komunikacja”

Na rysunku 77 przedstawiono przykładowy program realizujący algorytm sterowania rozruchem silnika z wykorzystaniem programu jako przełącznika gwiazda/trójkąt. Przycisk P01 służy do uruchamiania startu silnika, wyjście Q01 załącza stycznik doprowadzający zasilanie do silnika, przycisk P02 umożliwia zatrzymanie silnika. Sekwencja załączania obwodu gwiazdy (wyjście Q02) na określony czas, a następnie jego wyłączania i z 0,5 sek. opóźnieniem załączania stycznika (wyjście Q03) odpowiedzialnego za połączenie obwodów zasilających silnik w trójkąt, realizowane są automatycznie przez widoczny program.



Rysunek 77. Przykładowy program przełącznika gwiazda/trójkąt

2.2. Program LogoSoft Comfort – język bloków funkcjonalnych FBD

Programowanie sterowników PLC językiem bloków funkcjonalnych opisane zostanie na przykładzie oprogramowania Siemens LogoSoft Comfort. Jest to uniwersalny moduł logiczny, który znajduje zastosowanie w gospodarstwach domowych oraz w technice instalacyjnej (np. oświetlenie, zasłony słoneczne, żaluzje), w szafach sterujących, maszynach i urządzeniach (np. systemy sterowania bramą, systemy wentylacyjne, systemy sterowania pompami wody deszczowej, zadawanie pasz dla zwierząt itp.). Można stosować go do specjalistycznych systemów sterowania ogrodami zimowymi lub szklarniami, do obróbki sygnałów w układach sterowania oraz do miejscowych sterowań maszynami lub procesami w rozproszonych systemach sterowania, przy wykorzystaniu sieci ASI (Actuator Sensor Interface). ASI jest otwartym standardem przemysłowym magistrali obejmującej czujniki i elementy wykonawcze, charakteryzującej się bardzo krótkim czasem odpowiedzi pozwalającym na bardzo szybką reakcję elementów wykonawczych.

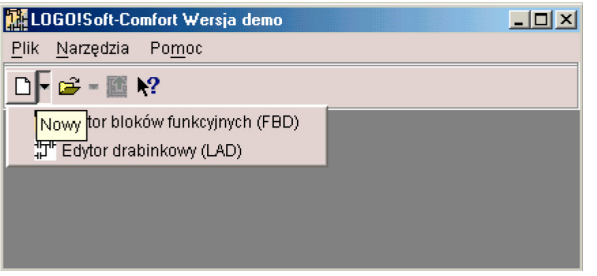
Oprogramowanie to zawiera następujące funkcje:

– tworzenie programu aplikacji bez połączenia z sterownikiem (offline);
– symulacja programu na komputerze;
– tworzenie i drukowanie schematu ideowego układu;

– przesłanie programu i komunikacja:
  - z Logo do PC;
  - z PC do Logo.

Programowanie na PC odbywa się na zasadzie „przeciągnij i upuść", dzięki czemu jest efektywne i przejrzyste. Najpierw powstaje program, a dopiero później wybiera się wersję sterownika wymaganą dla gotowego programu. Rozwiązanie pozwala optymalnie dobrać sterownik do zastosowania. Szczególnie przyjazna dla użytkownika jest symulacja programu Offline, która pokazuje jednocześnie stan większości funkcji specjalnych. Cały układ programu drukowany jest jako schemat blokowy lub kilka schematów blokowych posortowanych względem wyjść.

Po włączeniu programu pojawia się na ekran początkowy, w którym należy wybrać opcje *Edytor bloków funkcyjnych* (rys. 78). W Logo dostępne są dwa języki programowania: język bloków funkcyjnych (FBD) oraz język schematów drabinkowych (LAD).

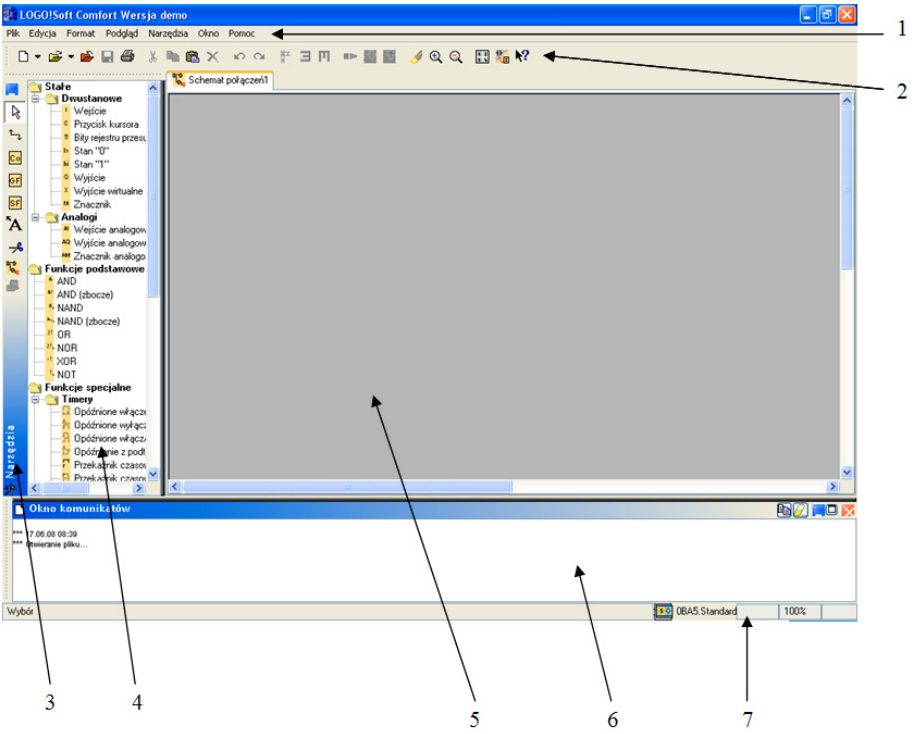


Rysunek 78. *Okno zakładania nowych projektów*

Następnie pojawia się interfejs użytkownika (rys. 79), największy obszar ekranu jest zajęty przez przestrzeń do tworzenia schematów diagramów. Jest to obszar, gdzie rozmieszcza się i łączy pobrane z biblioteki elementy programu sterującego. Pod nim znajduje się *Okno komunikatów*, w którym wyświetlane są informacje o projekcie (zawierające także informacje, jak: ilość użytych bloków funkcyjnych, zajętej pamięci, ilość wejść/wyjść, znaczników oraz maksymalną głębokość zagnieżdżenia).

Okno programu LogoSoft składa się z następujących części:

– *menu główne programu (1),* umożliwia dostęp opcji, funkcji i narzędzi programu. Po naciśnięciu lewym przyciskiem myszy na pozycjach z menu otwierają się okna kolejnych menu. Po menu można się poruszać się myszą lub za pomocą klawiszy kursora. Wejście do głównego menu umożliwia również przycisk F10;
– *lista narzędzi standardowych (2),* przyciski znajdujące się na pasku narzędzi standardowych umożliwiaj szybki dostęp do podstawowych operacji na programie, takich jak: odczyt, zapis, drukowanie, ogólnych narzędzi edycyjnych systemu Windows oraz opcji komunikacyjnych Logo;
– *lista narzędzi programowych (3),* przyciski znajdujące się na pasku narzędzi programowych umożliwiają szybki dostęp do narzędzi edycyjnych programu oraz symulatora i testowania *online.* Wygląd i możliwości umieszczonych na pasku przycisków zależą od edytora, w którym zachodzi programowanie;



Rysunek 79. *Okno główne programu LogoSoft*

- *katalog – lista elementów programu (4),* zawiera wykaz wszystkich elementów, które można wykorzystać w programie: wejść, wyjść oraz funkcji podstawowych i specjalnych;
- *okno edytowanej aplikacji (5),* w tym miejscu tworzy się program aplikacji;
- *okno komunikatów (6),* położone jest na dole okna programu LogoSoft. Po wybraniu z menu programu *Narzędzia-> Ustalenia Logo* (lub naciśnięciu F2) w oknie wyświetlane są informacje o wykorzystanych zasobach, maksymalnej głębokości zagnieżdżania programu oraz możliwościach zastosowania sprzętu do realizacji stworzonego programu. Z kolei po wybraniu z menu *Narzędzia-> wybór urządzenia,* w oknie podawane są informacje o wersjach sterownika, których nie można zastosować do realizacji projektu, wymieniane są niedostępne dla danej wersji Logo, a zastosowane w projekcie, bloki funkcjonalne. W oknie komunikatów wyświetlane są również informacje o błędach powstałych na początku symulacji, przy próbach połączenia LogoSoft ze sterownikiem Logo oraz podczas ładowania i ściągania programu. Oprócz właściwej treści każdy komunikat zawiera również nazwę projektu oraz informacje o dacie i czasie powstania. Okno komunikatora zamyka się i otwiera przez wybieranie z menu *Podgląd → Okno informacyjne.* Po ustawieniu wskaźnika w oknie komunikatów i kliknięciu lewym przyciskiem myszy można wprowadzać do okna tekst z klawiatury;
- *pasek statusu (7),* zawiera użyteczne informacje dotyczące aplikacji użytkownika i realizowania działań.

Pasek *Menu* zawiera polecenia przeznaczone do redagowania i zarządzania programem. Na dole okna aplikacji znajduje się pasek stanu zawierający informację o aktywnym narzędziu, stanie programu i aktualnym powiększeniu.

Przyciski na *Standardowym pasku narzędzi* reprezentują polecenia, które są też dostępne na pasku menu (rys. 80). Umieszczanie przycisków dla najważniejszych poleceń na *Standardowym pasku narzędzi* umożliwia szybki dostęp do takich funkcji, jak: *Nowy projekt, Otwórz projekt, Zamknij projekt, Zapisz projekt, Wytnij zaznaczony obiekt, Skopiuj zaznaczony obiekt, Wstaw, Cofnij, Formatuj Automatycznie, Rozłóż Pionowo, Rozłóż Poziomo, Załaduj program do Logo, Wczytaj program z Logo, Powiększ, Pomniejsz, Podział strony, Konwersja LAD/FBD.*

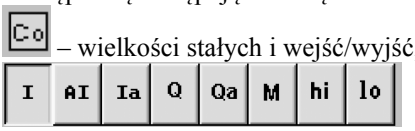
Rysunek 80. Standardowy pasek narzędzi

Pasek Programowania zawiera przyciski do tworzenia projektu i jego redagowania (rys. 81). Każde z narzędzi reprezentuje pewien tryb programowania. Znajdują się tam następujące polecenia: *Zaznacz, Wstaw tekst, Przetnij połączenie* (połączenie ukryte), *Połącz* (połączenie jawne), oraz grupy bloków funkcjonalnych: Co, GF i SF.

Rysunek 81. Pasek programowania

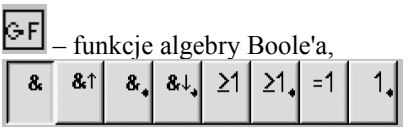
Pierwszym krokiem w tworzeniu diagramu programu jest wybór schematu bloku funkcji, które są wymagane do wykonania zamierzonego obwodu. Nie ma znaczenia, czy rozpocznie się projektowanie od wejść, podstawowych funkcji czy funkcji specjalnych.

Dostępne są następujące narzędzia:



– wielkości stałych i wejść/wyjść,

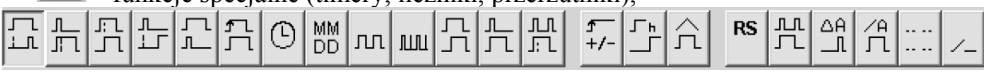
Zakładka ta zawiera wejścia/wyjścia cyfrowe i analogowe, pamięć (M) oraz bloki umożliwiające zadanie wartości początkowej: wysoki stan logiczny – hi oraz niski – lo.



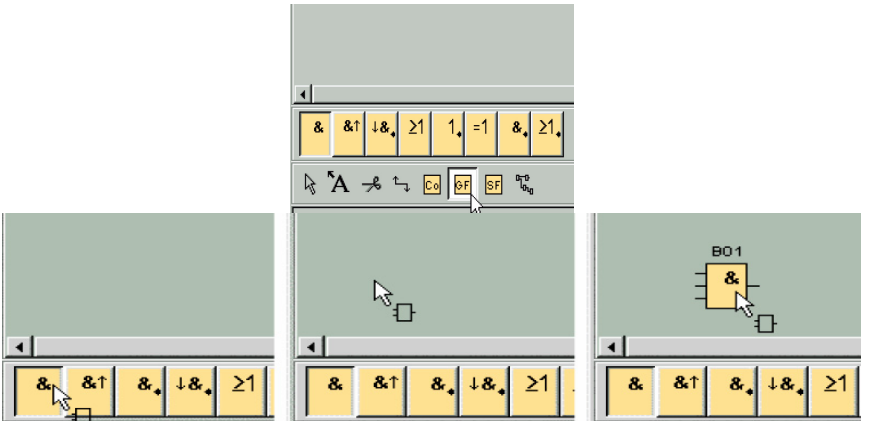
– funkcje algebry Boole'a,

Funkcje algebry Boole’a zawierają wszystkie możliwe wariancje funkcji podstawowych AND, OR i XOR.

– funkcje specjalne (timery, liczniki, przerzutniki),

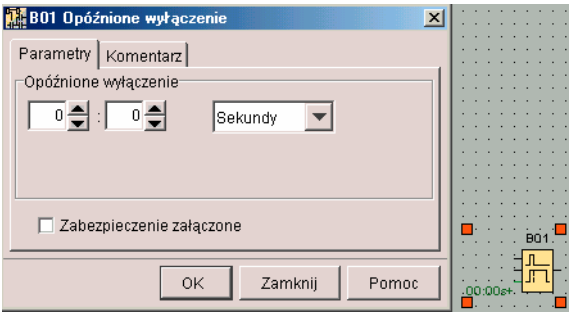


Stanowią one w istocie bloki funkcjonalne, które realizują określone funkcje, należy je parametryzować. Wstawianie elementów funkcji do schematu blokowego polega na zaznaczeniu odpowiedniego narzędzia, wybraniu bloku funkcyjnego, a następnie na kliknięciu na obszar kreślenia (rys. 82). Blok funkcjonalny zostanie umieszczony pod kursorem myszki.



Rysunek 82. Procedura wyboru i umieszczenia bloku funkcyjnego w programie

Nie jest wymagane precyzyjne rozmieszczanie bloków w obszarze kreślenia, gdyż po połączeniu między sobą i nadaniu im etykiet można je przesuwać w celu osiągnięcia najbardziej czytelnego układu programu sterującego. Numery bloków dla funkcji podstawowych i funkcji specjalnych nadawane są automatycznie, w kolejności wstawiania ich do programu i nie mogą być zmienione. Numery przypisane wejściom, wyjściom i znacznikom można zmienić – wybierając z dostępnej listy, natomiast bloki reprezentujące stałe poziomy logiczne nie posiadają numeracji. Po dwukrotnym kliknięciu na bloku funkcjonalnym otwiera się okienko dialogowe, udostępniające wprowadzanie parametrów bloku niezbędnych do prawidłowego działania w projektowanym obwodzie (rys. 83).

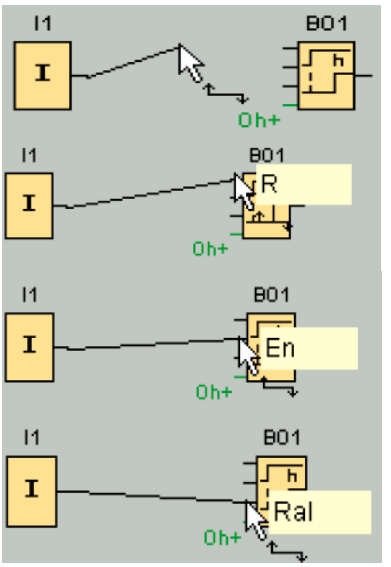


Rysunek 83. Parametryzacja bloku funkcyjnego

Aby uzupełnić obwód, należy połączyć bloki funkcyjne pomiędzy sobą. W tym celu należy wybrać przycisk *Połącz* na pasku narzędzi i połączyć wejście i wyjście dwóch wybranych bloków.

 **– Połącz**

Jeśli linia łącząca jest prowadzona od wyjścia do wejścia, wówczas otwiera się okno opisu połączenia. Kiedy nastąpi zwolnienie przycisku myszki, linia jest przyciągana do wskazanego wejścia (rys. 84).



Rysunek 84. *Łączenie bloków funkcyjnych*

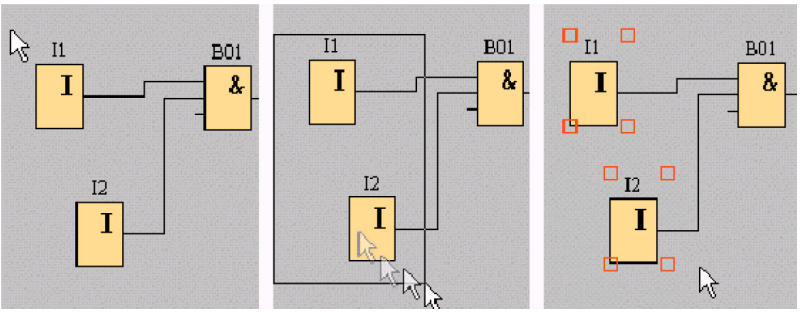
Przy łączeniu bloków funkcji obowiązują następujące zasady:
- połączenia mogą zostać zrobione tylko między wejściami bloku i wyjściem bloku;
- wyjście może zostać połączone do więcej niż jednego wejścia, ale wejście nie może zostać połączone do więcej niż jednego wyjścia;
- wejście i wejście tego samego bloku nie mogą byś połączone z sobą, jeśli takie połączenie jest wymagane, pomiędzy nimi musi być włączony znacznik lub inne wyprowadzenie;
- w funkcjach specjalnych występują tzw. *zielone piny* (rys. 84 – pin 0h+), wejścia te nie są łączone z pozostałymi blokami, lecz służą do przypisania parametrów bloku funkcyjnego.

Przed przesunięciem, wyrównaniem lub usunięciem obiektu trzeba go najpierw zaznaczyć. Aby zaznaczyć blok, najpierw musimy wybrać narzędzie *Wybór* na pasku *Narzędzi***.**

 **– Wybór**

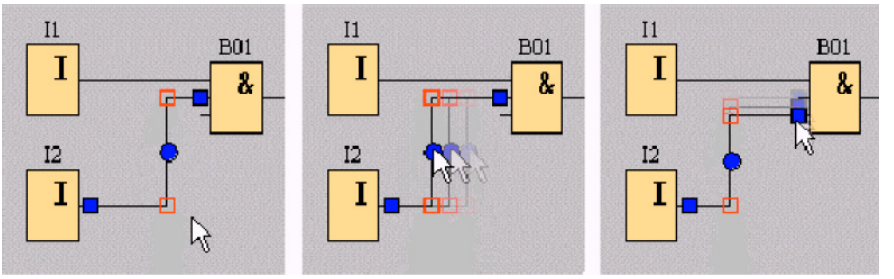
W przypadku konieczności zaznaczenia pojedynczego bloku lub linii połączenia wystarczy na nim kliknąć. Grupy bloków lub linie połączeń są wybierane przez obejmowanie i złączanie ich z prostokątem. Polecenie to wykonywane jest standardowo, tj. należy nacisnąć i przytrzymać lewy przycisk myszy, a następnie wykonać przeciągnięcie nad obszarem zawierającym elementy przeznaczone do edycji. Po zwolnieniu przycisku myszy wybrane obiekty są wskazane przez małe, czerwone kwadraty (rys. 85). Zaznaczone obiekty mogą być usunięte lub przesunięte.

Wybrane obiekty mogą też zostać wycięte, skopiowane lub wklejone przy pomocy przycisków dostępnych w *Standardowym pasku narzędzi*.



Rysunek 85. Wybieranie – zaznaczanie elementów

Zaznaczone połączenia można edytować, posługując się punktami pomocniczymi, powstającymi po ich zaznaczeniu (rys. 86).

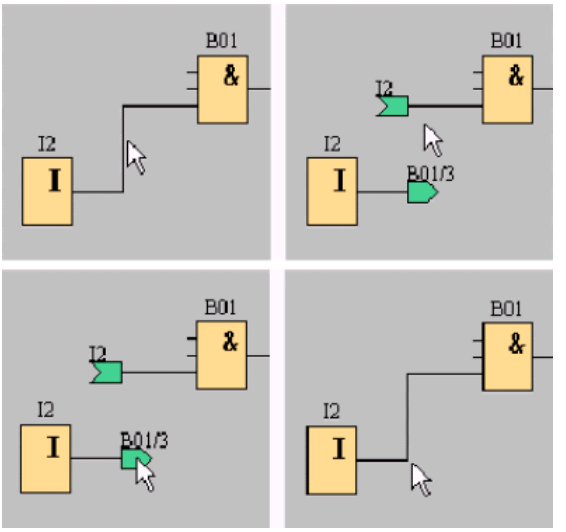


Rysunek 86. *Edycja połączeń*

Rozbudowane schematy mogą stawać się nieczytelne w programie Logo, gdyż czasami trudno jest rozróżnić przecięcia się linii z ich skrzyżowaniami. Aby poprawić czytelność schematów, można użyć opcji *Rozłącz/Scalaj Połączenia* dostępnej jako przycisk. Powstają wtedy tzw. *pętle ukryte (niejawne)*.

– Rozłącz/Scalaj Połączenia

Jeżeli aktywna jest ta opcja, każde kliknięcie za wybrane połączenie zostanie przecięte (połączenie pomiędzy blokami pozostaje dalej aktywne), a na końcach połączeń utworzone zostaną zielone strzałki, których groty wskazują kierunek przepływu sygnału. Nad strzałkami umieszczony zostaje opis, do którego bloku i pinu odnosi się połączenie. Po kliknięciu na grot przeciętego połączenia jest ono z powrotem przywracane (rys. 87).



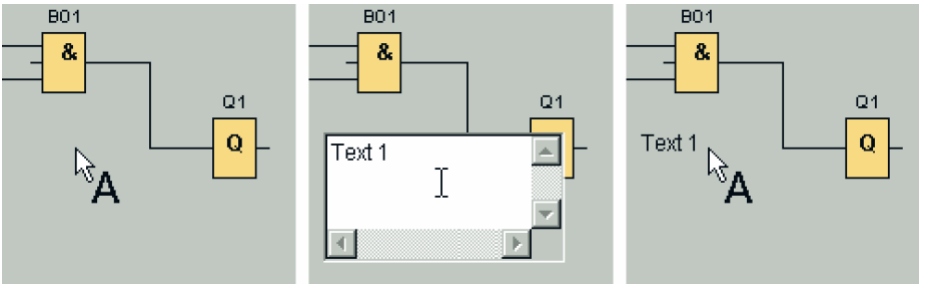
Rysunek 87. *Rozłączanie i scalanie połączeń*

Jeśli po wybraniu przycisku *Wstaw Komentarz* nastąpi kliknięcie na obszarze kreślenia lub na wybranym obiekcie, wówczas otworzy się okno, w którym jest możliwe wpisywanie tekstu będącego komentarzem lub opisem do schematu. Wpisywanie należy zakończyć, klikając poza obszarem okienka tekstowego lub wciskając klawisz *ESC*.

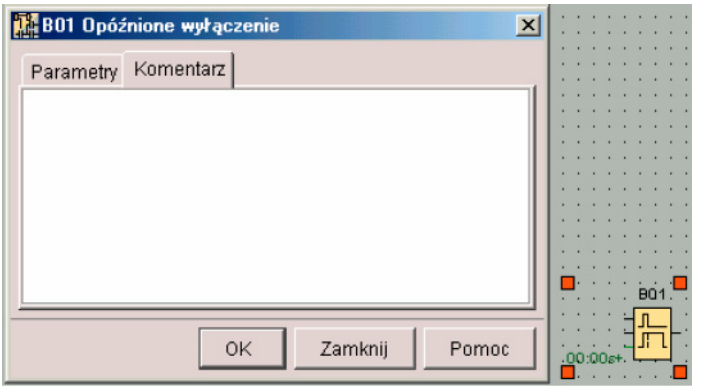
– Wstaw Komentarz

Okno zostanie zamknięte, a wpisany tekst pojawi się na schemacie. Tekst ten może być zaznaczany, przenoszony, wyrównywany lub kasowany jak każdy obiekt (rys. 88). Jego zadaniem jest poprawić czytelność diagramu programu.

W okienku dialogowym własności bloku można wprowadzić komentarz, za pomocą którego można opisać nazwę bloku oraz uwagi co do zadania, jakie on spełnia w obwodzie (rys. 89). Komentarze są wyświetlane na schemacie jako tekst powiązany, a ich pozycja może być zmieniana, jednakże jeżeli nastąpi przesunięcie bloku – komentarz też zostanie przesunięty, jeśli blok zostanie usunięty, to samo stanie się z komentarzem.



Rysunek 88. *Umieszczanie komentarza*



Rysunek 89. *Komentarze bloków*

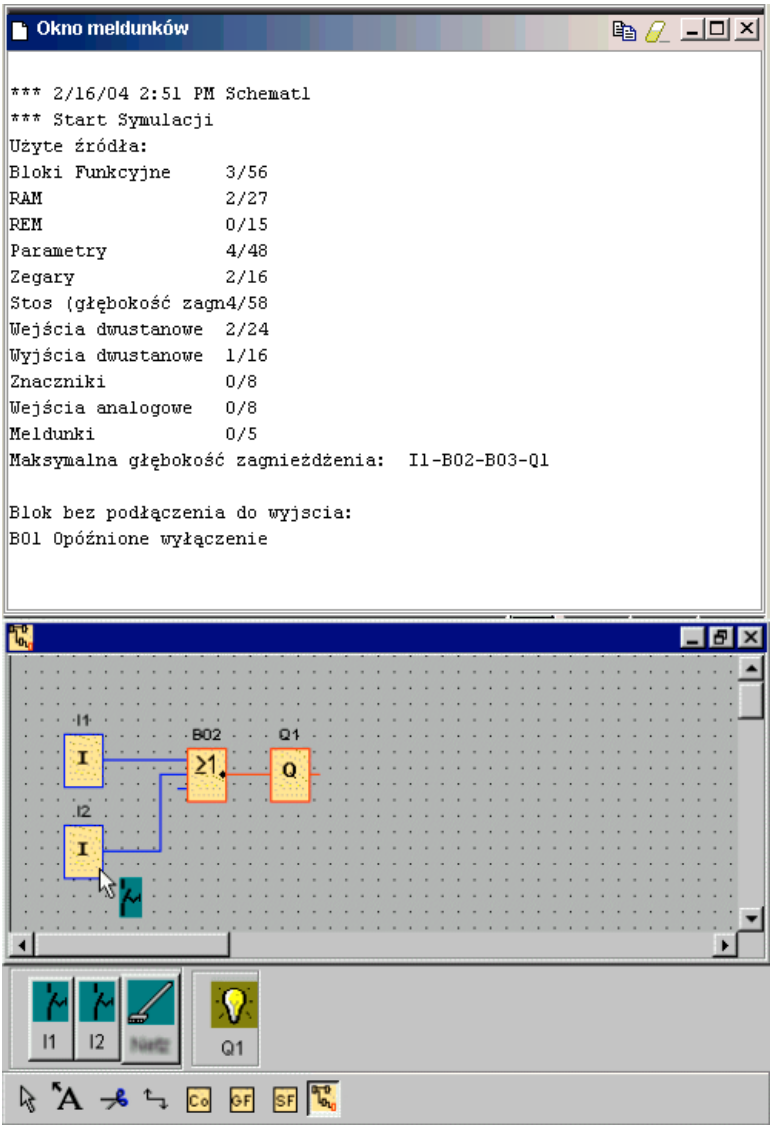
Po rozpoczęciu symulacji analizowany jest cały schemat w celu stwierdzenia, jakie zasoby modułu sterownika mikroprocesorowego zostaną użyte.

 **– Symulacja**

Informacja o nich wraz z komunikatami błędów jest wyświetlana w *Oknie Meldunków* (rys. 90).

Wciśnięcie jakiegokolwiek przycisku z paska Narzędziowego powoduje wyłączenie symulacji.

W trybie symulacji sygnalizacja stanu urządzeń wejściowych, wyprowadzeń i zasilania umieszczona jest przeważnie pod obszarem kreślenia.



Rysunek 90. Okno raportowania procesu symulacji

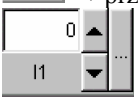
Wejścia reprezentowane są jako przyciski z symbolem klucza lub przełącznika, pod którym znajduje się opis wejścia. Otwarte wejście odpowiada przełącznikowi, który nie został obsłużony:

→ przycisk klucza I1 nieobsłużony → wejście otwarte,

→ przycisk klucza I1 obsłużony → wejście zamknięte,

→ przycisk włącznika I2 nieobsłużony → wejście otwarte,

→ przycisk włącznika I2 obsłużony → wejście zamknięte.



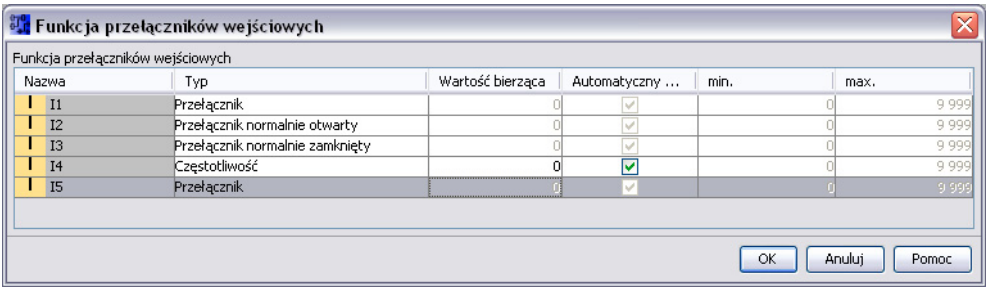
→ okno statusu modułu wejściowego,

→ wejście częstotliwościowe I3,

Wejścia mogą być zdefiniowane na kilka sposobów. Aby zmienić parametry wejść, należy wybrać sekwencję: *Narzędzia → Symulacja Parametrów* z paska narzędzi.

W oknie dialogowym, które powinno się otworzyć można zdefiniować, w jaki sposób będzie pracować wejście. Istnieją trzy możliwości:

– *przełącznik* zostaje włączony po pierwszym naciśnięciu, zaś wyłączony po kolejnym naciśnięciu;
– *przycisk monostabilny* jest aktywny wyłącznie, kiedy jest naciskany, można wybrać, czy przycisk monostabilny ma być *normalnie otwarty*, czy też *normalnie zamknięty*;
– częstotliwość wejścia może być wstępnie wyznaczona lub zmienia podczas przebiegu symulacji – podawana jest w Hz. Wejście częstotliwości jest specjalnym przypadkiem wejścia i zaleca się stosować je z poziomu bloku funkcji specjalnej (rys. 91).



Rysunek 91. *Parametryzacja wejść*

Kliknięcie na przycisk *Power* powoduje rozłączenie zasilania dla wszystkich urządzeń wejściowych, tym samym zachodzi stan symulujący awarię zasilania. W ten sposób można przetestować zachowanie się układu w przypadku awarii zasilania oraz ponownego restartu zaprojektowanego układu sterowania.

→ przycisk *Zasilanie* niewciśnięty

→ przycisk *Zasilanie* wciśnięty → symulacja awarii zasilania

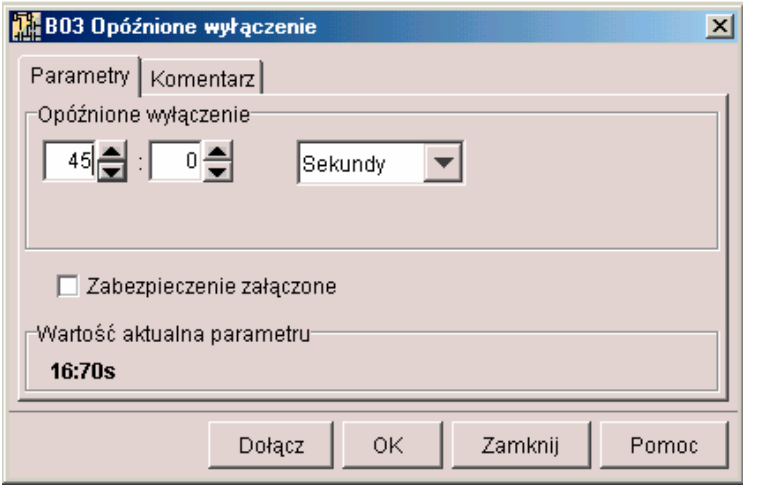
Stan wyjść i znaczników jest sygnalizowany przez symbol zapalonej lub zgaszonej żarówki. Pod symbolem żarówki znajduje się (podobnie jak w przypadku wejść) opis wyjścia.

→ ikona statusu wyjścia Ql—> wyjście nieaktywne

→ ikona statusu wyjścia Ql—> wyjście aktywne

Ikony statusu wyjść mają charakter czysto informacyjny i nie można ich użyć do zmiany stanu wyjść. Wyjścia są opisane literą Q, zaś znaczniki (pamięć) literą M. Niektóre wyjścia mają oznaczenia Q_a, są one wyjściami linii Asi (dostępne tylko modułach sterownika Logo).

Dwukrotne kliknięcie na bloku funkcjonalnym podczas przebiegu symulacji otworzy okienko dialogowe właściwości bloku. Możliwe są zmiany parametrów bloku (podobnie jak w trybie programowania), oprócz tego można również wstawiać i edytować komentarze przypisane do poszczególnych bloków (rys. 92).

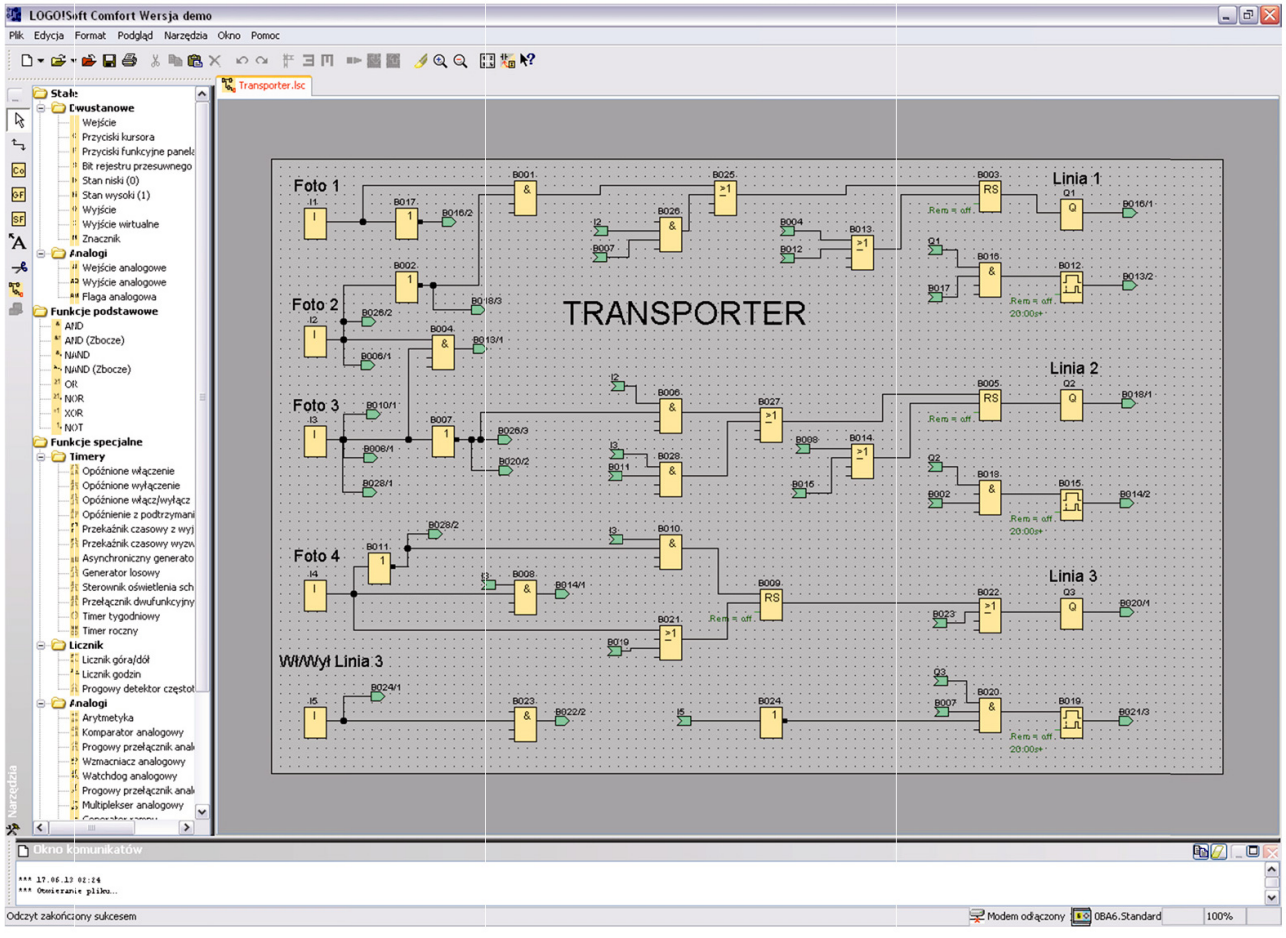


Rysunek 92. Okno ustawiania parametrów symulacji

Podczas przebiegu symulacji w oknie tym pokazywane są aktualne parametry bloku sterującego. Udogodnienie to pozwala prześledzić zachowanie się programu sterującego. Możliwe jest otworzenie wielu okien informacyjnych, jednego bloku funkcyjnego i zredukowanie ich do takiej wielkości, żeby pokazywały parametry potrzebne użytkownikowi.

Przykładowy program realizujący sekwencyjne załączanie trzech taśmociągów (tzw. ciąg transportowy) przesyłających gotowy wyrób (np. skrzynki, palety, pudełka) do stanowiska zrobotyzowanej paletyzacji z opcją automatycznego zatrzymywania poszczególnych transporterów zamieszczono na rysunku 93. System sterowania ma za zadanie przetransportować towar umieszczony na początku taśmociągu nr 1 na koniec taśmociągu nr 3. W przykładzie założono, że każdy z taśmociągów jest napędzany jednym silnikiem elektrycznym. Opakowanie umieszczone na pierwszym taśmociągu wykrywane jest przez fotokomórkę podającą sygnał logiczny „1" na wejście I1 sterownika Program sterownika sprawdza, czy taśmociąg nr 1 nie jest zajęty przez wcześniej umieszczony towar (czy fotokomórka I2 znajdująca się na końcu taśmociągu nr 1 nie jest przysłonięta) i przez wyjście Q1 uruchamiany jest silnik napędzający taśmociąg nr 1. Przemieszczające się opakowanie po dotarciu do końca taśmociągu nr 1 zostaje wykryte przez fotokomórkę, która wysyła sygnał na wejście I2. Następuje sprawdzenie, czy taśmociąg nr 2 nie jest zajęty. Jeśli nie ma tam kartonu, wówczas poprzez wyjście Q2 uruchamiany jest silnik napędzający taśmociąg nr 2 i opakowanie kartonowe przemieszcza się z taśmociągu nr 1 na taśmociąg nr 2. Analogicznie realizowane jest przejście opakowania z taśmociągu nr 2 na taśmociąg nr 3. Karton po dotarciu do końca transportera nr 3 zostaje wykryty przez fotokomórkę (przesłanie sygnału logicznego „1" na wejście I4), co powoduje zatrzymanie silnika (wyjście Q3) napędzającego ten transporter. Ładunek może zostać zdjęty z transportera nr 3 ręcznie przez człowieka lub mechanicznie przez robota. Podanie sygnału „1" na wejście I5 np. z przycisku uruchamia bezwarunkowo linię 3. Jeśli któryś z transporterów jest zajęty przez opakowanie, program to wykrywa i nie dopuszcza do kolizji – ładunek jest odpowiednio kolejkowany. Transportery są uruchamiane i zatrzymywane sygnałami z fotokomórek, dodatkowo w program wbudowany został mechanizm zatrzymywania taśmociągów po określonym czasie, zależnym od długości taśmociągu. Ten mechanizm czasowego zatrzymywania ruchu transportera jest przydatny, w momencie gdy jeden z taśmociągów zostanie uruchomiony przypadkowo, np. jedna z fotokomórek zostanie na moment przysłonięta przez przechodzącego człowieka, ścinek papieru itp.

Funkcje wyprowadzeń:
I1 – wejście dla pierwszego taśmociągu;
I2 – wejście dla drugiego taśmociągu;
I3 – wejście dla trzeciego taśmociągu;
I4 – wejście wyłączające trzeci taśmociąg;
I5 – wejście ponownie włączające trzeci taśmociąg;
Q1 – wyjście pierwszego taśmociągu;
Q2 – wyjście drugiego taśmociągu;
Q3 – wyjście pierwszego taśmociągu.



Rysunek 93. Program sterujący transportem opakowań

2.3. Program CoDeSys – język tekstu strukturalnego ST

CoDeSys jest środowiskiem programistycznym, służącym do programowania sterowników mikroprocesorowych zgodnych z IEC 61131-3. Pozwala użytkownikowi na tworzenie programów za pomocą: schematu drabinkowego (LD), bloków funkcyjnych (FDB), listy rozkazów (IL), tekstu strukturalnego (ST) i sekwencyjnego schematu funkcjonalnego (SFC). CoDeSys posiada również narzędzie do wizualizacji, które umożliwia tworzenie grafiki z obiektami oraz ich animację sterowaną zmiennymi. Zmienne posiadają format zgodny z IEC 61131-3. W niniejszym rozdziale zostanie omówione programowanie sterownika PLC w języku tekstu strukturalnego ST (Moeller, 2012).

Kod źródłowy, który zawiera się w programie sterownika, jest przechowywany w pliku z nazwą projektu. Projekt zawiera różnego rodzaju obiekty: moduły, definicje typów danych, elementy prezentacji (wizualizacja) i zasoby. Pierwszy moduł, utworzony w nowym projekcie, automatycznie nosi nazwę *PLC_PRG*. Stamtąd uruchamiane są procesy (analogicznie jak funkcja *main* w programie C) oraz wywołania innych modułów (programów, bloków funkcji i funkcji).

Elementy programu:

- operatory arytmetyczne;
- operatory bitstring;
- operatory bit-shift;
- operatory wyboru;
- operatory porównawcze;
- operatory adresu;
- konwersja typów;
- operatory numeryczne;
- typy danych;
- argumenty – stałe, zmienne, adresy.

Biblioteki standardowe:

- standard.lib (funkcje bistabilne);
- standard.lib (funkcje liczenia);
- standard.lib (funkcje string);
- standard.lib (funkcje czasowe);
- standard.lib (wykrywanie zbocza);
- util.lib (funkcje analogowe);
- util.lib (konwersja BCD);
- util.lib (funkcje bit/bajt);
- util.lib (regulatory);
- util.lib (manipulatory funkcji);
- util.lib (funkcje matematyczne);
- util.lib (generatory sygnału).

Funkcje, bloki funkcji i programy są to moduły, dla których można wykonywać określone działania. Każdy moduł składa się z części deklaracyjnej i kodu. Kod jest pisany w jednym z języków programowania IEC: IL, ST, SFC, FBD, LD lub CFC.

CoDeSys obsługuje wszystkie standardowe moduły IEC. Chcąc używać tych modułów w projekcie, należy podłączyć do projektu bibliotekę standard.lib.

Funkcją jest moduł, który po wykonaniu zwraca dokładnie jedną wartość (może być także wieloskładnikowa, jak np. pola lub struktury). Wywołanie funkcji w językach tekstowych może występować w wyrażeniach jako operator. Podczas deklarowania funkcji należy pamiętać o tym, że funkcja musi zawierać typ (wpisując nazwę funkcji należy po dwukropku wstawić jej typ).

Deklaracja funkcji rozpoczyna się słowem kluczowym *FUNCTION,* dla funkcji należy przypisać rodzaj typu danych.

Prawidłowa deklaracja funkcji przedstawia się następująco:

FUNCTION Fct: INT

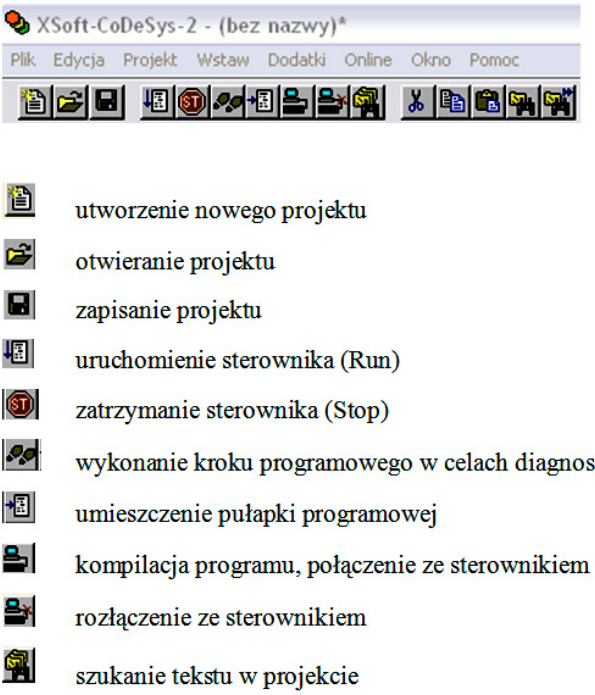
Moduł funkcji (blok funkcjonalny) – nazywany także blokiem funkcji – jest to moduł, który podczas wykonywania dostarcza jedną lub kilka wartości. Blok funkcji nie dostarcza wartości zwracanych w przeciwieństwie do funkcji. Deklaracja bloku funkcji rozpoczyna się słowem kluczowym *FUNCTION_BLOCK*. Można tworzyć duplikacje, inaczej zwane instancjami (kopie) bloku funkcji. Każda instancja posiada osobną identyfikację (nazwę instancji) oraz strukturę danych, która zawiera wejścia, wyjścia i wewnętrzne zmienne. Instancje deklaruje się podobnie jak zmienne lokalnie lub globalnie, podając jako typ identyfikatora nazwę bloku funkcji.

Program jest to moduł, który podczas wykonywania dostarcza jedną lub kilka wartości. Programy są jawne globalnie w całym projekcie. Wszystkie wartości są zachowywane pomiędzy kolejnymi cyklami wykonywania programu. Deklaracja programu rozpoczyna się słowem kluczowym *PROGRAM* i kończy słowem kluczowym *END_PROGRAM*. Programy mogą wywoływać programy lub bloki funkcji. Wywołanie programu w funkcji jest niedozwolone. Nie można również tworzyć instancji (lokalnych kopii) programów.

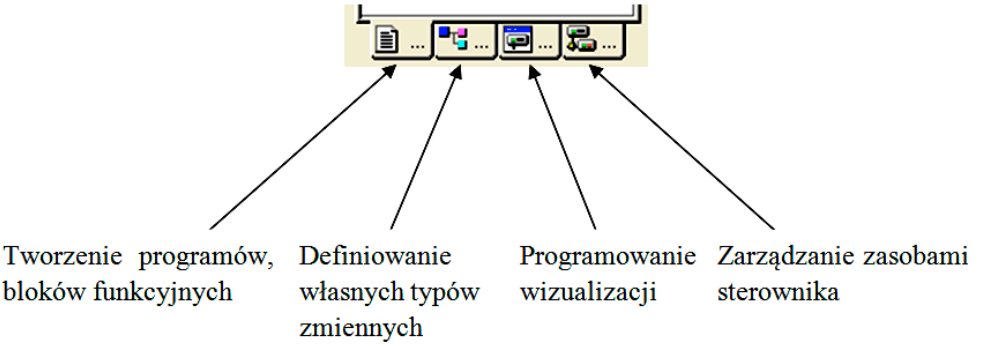
Program składa się z głównego paska zadań oraz z czterech zakładek. W górnej części ekranu umieszczone są podstawowe ikony funkcyjne (rys. 94).

W dolnej części ekranu umieszczone są zakładki (rys. 95), pełnią one funkcję menadżera projektu, a najważniejszą są z nich są *zasoby PLC*. Używa się ich do konfiguracji i organizacji projektu oraz do śledzenia wartości zmiennych:

- *Zmienne globalne,* można zastosować w całym projekcie albo sieci;
- *Biblioteki,* można podłączać do projektu za pomocą menedżera bibliotek;
- *Dziennik,* służy do rejestrowania aktywności w trybie online;
- *Konfiguracja alarmów,* wykorzystywana jest do konfigurowania obsługi alarmów w projekcie;
- *Konfiguracja sterownika* odpowiada za parametry konfiguracji urządzenia (m.in. wprowadza się moduły i adresuje w pamięci);
- *Konfiguracja zadań* przeznaczona jest do sterowania programem za pomocą zadań;
- *Menadżer podglądu zmiennych i receptur* służy do wyświetlania i ustawiania wartości domyślnych dla zmiennych;
- *Ustawienia systemu docelowego* pozwalają na wybieranie i konfigurację systemu docelowego;
- *Obszar roboczy* zawiera obraz opcji projektu.



Rysunek 94. Widok głównego paska zadań oraz głównych przycisków poleceń



Rysunek 95. Widok zakładek w dolnej części okna projektu

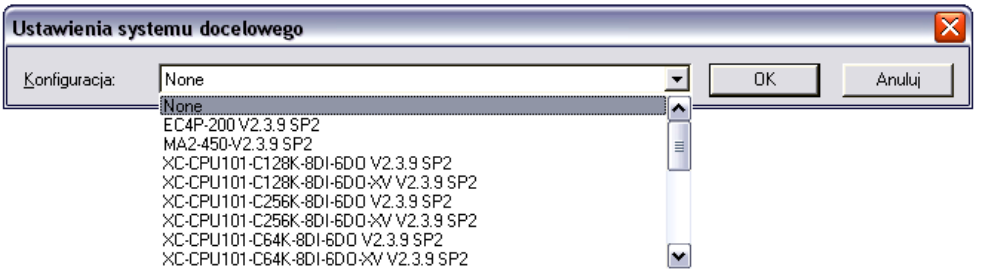
Program CoDeSys umożliwia tworzenie wizualizacji przedstawiających zmienne projektowe na obrazie graficznym. Za pomocą wizualizacji można rysować w trybie offline elementy geometryczne, które w trybie online zmieniają, w zależności od wartości przypisanych zmiennych, swoje kształty/kolory/napisy. Wizualizacja CoDeSys może być również

używana w systemie CoDeSys HMI jako: jedyna platforma obsługi systemu sterowania, wizualizacja sieciowa lub wizualizacja elementu docelowego obsługiwana przez Internet, lub na systemie docelowym.

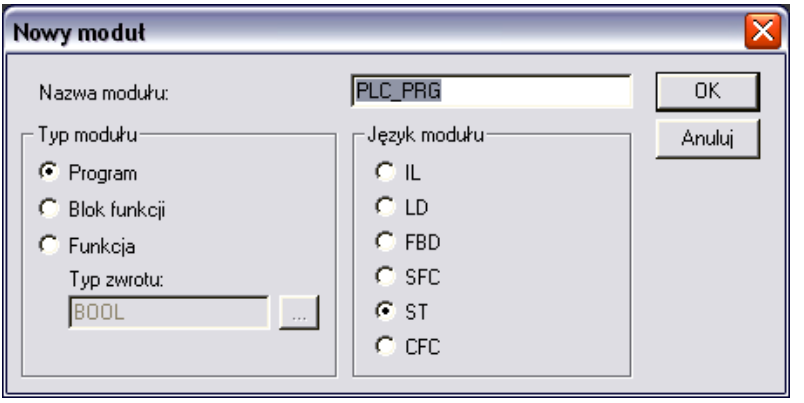
2.3.1. Struktura projektu

Po uruchomieniu programu należy wybrać z paska zadań *File* następnie *New* lub kliknąć na ikonę  . Po wybraniu jednej z tych opcji wyświetli się okno wyboru sterownika (jednostki bazowej CPU, moduły rozszerzeń dodawane są w zakładce *Zasoby*) (rys. 96).

Po wybraniu sterownika pojawia się okno z możliwością wyboru modułu, a więc *program*, *blok funkcjonalny* i *funkcja*. Po wybraniu typu należy zaznaczyć język (spośród dostępnych), w jakim będzie programowany ten moduł (rys. 97).



Rysunek 96. Widok zakładki wyboru jednostki podstawowej sterownika



Rysunek 97. *Okno „Nowy moduł”*

Po dokonaniu wyboru nazwy programu, rodzaju modułu oraz języka, program przechodzi do okna, które jest sformatowane w zależności od dokonanego wcześniej wyboru (rys. 98). Okno to przeznaczone jest do wprowadzania kodu programu, podzielono go na dwie części, górną i dolną.

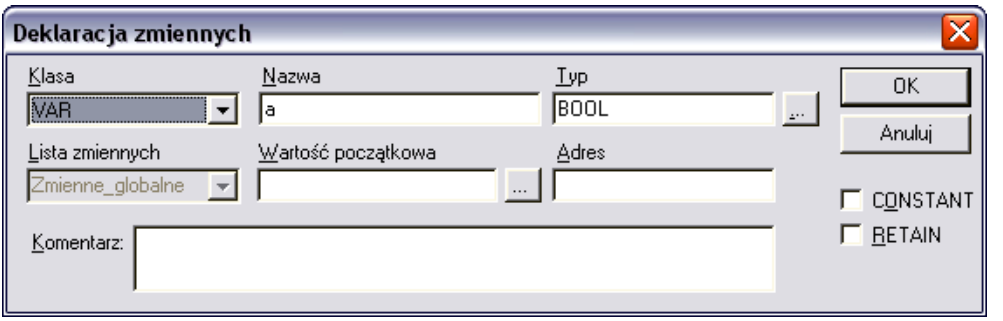
Rysunek 98. Widok okna wprowadzania programu i kompilacji

W górnej części należy zdeklarować zmienne, które będą później wykorzystane w pisaniu programu, natomiast dolna cześć jest główną częścią, w której powstaje kod programu. Powstaje on, jak wszystkie języki tekstowe, jako ciąg odpowiednich komend i poleceń. Poprawność pisania programu można w każdej chwili sprawdzić poprzez jego kompilację za pomocą klawisza funkcyjnego F11. Jeśli nie występują żadne błędy w pisanym programie, zostanie wyświetlona informacja o skompilowaniu się programu, czyli braku jakichkolwiek błędów, w przeciwnym wypadku program zwróci nam komunikat w postaci informacji o liczbie występujących błędów i czego one dotyczą. Na rysunku 98 widoczny jest komunikat błędu mówiący o tym, że do wykonania programu konieczna jest przynajmniej jedna instrukcja.

W trakcie pisania programu należy deklarować zmienne lokalne. Przy wprowadzaniu każdej nowej zmiennej pojawia się okno deklaracji zmiennych (rys. 99).

W oknie tym należy określić wszystkie niezbędne parametry dotyczące danej zmiennej, m.in. klasę zmiennej, np. VAR, nazwę zmiennej czy typ zmiennej.

Okno *Deklaracja zmiennych* wyposażone jest w następujące sekcje:
Klasa – klasa zmiennej. Dostępne po rozwinięciu listy opcje to:
– *VAR* – zmienna lokalna (niewidoczna poza programem lub blokiem, w którym została zdefiniowana);
– VAR_INPUT – zmienna wejściowa bloku funkcyjnego;
– VAR_OUTPUT – zmienna wyjściowa bloku funkcyjnego;
– VAR_IN_OUT – zmienna wejściowo-wyjściowa bloku funkcyjnego;
VAR_GLOBAL – zmienna globalna (widoczna w całym projekcie).

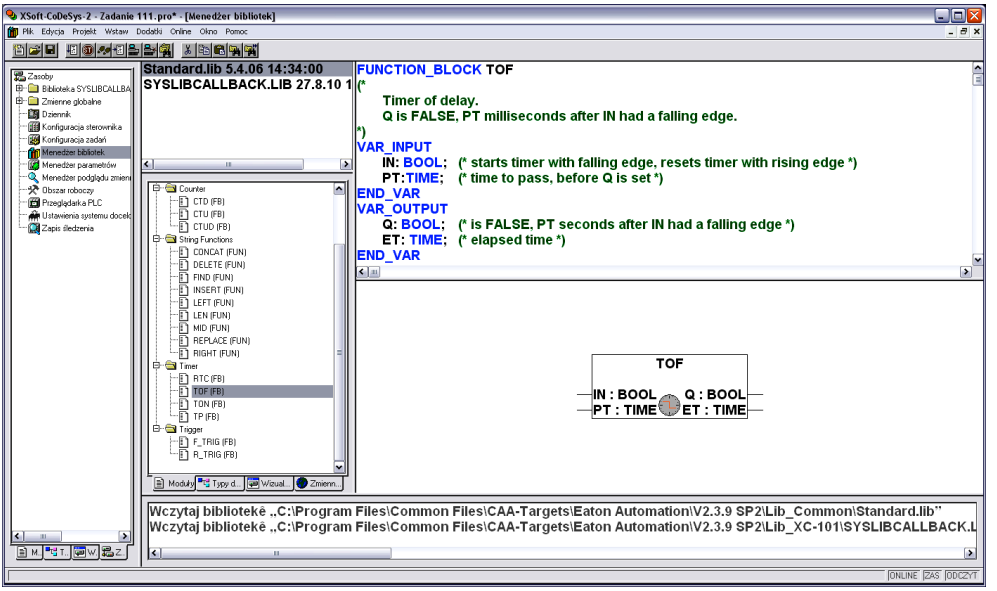


Rysunek 99. *Widok okna deklarowania zmiennych*

Wybranie rodzaju zmiennej uaktywni okno *Symbol list*. Należy w nim wybrać grupę zmiennych lokalnych;
– *Nazwa* – nazwa deklarowanej zmiennej;
– *Typ* – typ zmiennej np. BOOL, BYTE, WORD, INT itp.;
– *Wartość początkowa,* nadana zmiennej przy pierwszym cyklu programu;
– *Adres* - adres wejścia, wyjścia bądź markera, do którego zmienna ma być przypisana np. %QX0.0 (wyjście zerowe sterownika – typ BOOL), %MW100 (marker – słowo typu INT);
– *Constant,* zmienna zostaje zapisana jako stała;
– *Retain,* wartość zmiennej jest pamiętana po restarcie sterownika.

Nieodłączną częścią programu pisanego dla PLC są biblioteki. Zarządzanie takimi bibliotekami jest wykonywane za pomocą odpowiedniego managera (rys. 100).

W celu skorzystania z bibliotek należy przejść do zakładki *Resources* (zarządzanie zasobami sterownika). W tej zakładce istnieje możliwość skorzystania z biblioteki standardowej, która jest przypisana do danego programu, lub można też wczytać inną bibliotekę niezbędną dla potrzeb bieżącego programu. Nie zaleca się dodawania bibliotek niewykorzystywanych w programie, gdyż zajmują zasoby pamięci PLC oraz wydłużają jego cykl pracy.



Rysunek 100. *Widok okna biblioteki programu*

Aby zaprogramować przykładową aplikację licznika cykli pracy sterownika PLC, należy w oknie edycji programu wpisać polecenie:

a:=a+1;

Po wciśnięciu klawisza ENTER powinno uruchomić sie okno deklaracji zmiennych identyczne z przedstawionym na rysunku 99. Jako typ zmiennej wpisać UINT (oznacza dane typu: liczby całkowite bez znaku), pozostałe pola można pozostawić puste. Po zatwierdzeniu w oknie deklaracji zmiennych lokalnych pojawi sie nowa zmienna:

a: UINT;

Ten prosty przykład obrazuje semantykę języka ST. Powyższe wyrażenie tłumaczy się następująco:

zmiennej *a* przypisz (:=) wartość zmiennej *a* zwiększoną o *1*, zakończ to polecenie (;)

Polecenie wywoływane jest z każdym cyklem programu, efektem czego w zmiennej *a* zapisywana jest liczba cykli od uruchomienia programu. Do prostej funkcji licznika można dopisać sterowanie: jeżeli wartość zmiennej *a* jest równa *100*, to zmiennej *a* przypisz *0*.

Całość programu wygląda następująco:

a:=a+1;

IF a=100 THEN a:=0; END_IF

Funkcja *IF THEN* jest warunkiem, wyrażenie END_IF jest wymagane, gdy po THEN wystąpić może dowolnie wiele innych poleceń – każde zakończone oczywiście znakiem średnika. Po END_IF średnik nie jest wymagany. Wewnątrz IF można stosować również wyrażenia ELSE, ELSIF oraz zagnieżdżać wyrażenia IF.

Program może odwoływać się do adresów fizycznych wejść, wyjść i markerów:

a:=a+1;

IF a=100 THEN

```
        a:=0;
        %MB10:=%MB10+1;
END_IF

IF a<50 THEN
        %QX0.0:=TRUE;
ELSE
        %QX0.0:=FALSE;
END_IF
```

W powyższym programie każde zresetowanie wartości zmiennej *a* spowoduje również zwiększenie wartości w markerze 10. Ponadto jeżeli wartość zmiennej *a* będzie mniejsza od 50, załączone zostanie wyjście Q0.0 sterownika (w programie widocznym powyżej % jest znakiem umownym, a X oznacza zmienną Bool), w przeciwnym razie wyjście będzie wyłączone.

Zastosowane w przykładzie adresy bezpośrednio reprezentowane wygodniej jest zastąpić zmiennymi. Rozwiązanie takie pozwala w przyszłości na łatwą modernizację programu. Przykładowo zmiana adresu wyjścia %QX0.0 na %QX0.1 w pierwotnym programie zmuszałaby nas do przeszukania całego kodu programu i wprowadzania zmian w wielu miejscach – nietrudno wtedy o pomyłkę. W poniższej wersji przykładowego programu należy jedynie zmodyfikować deklaracje zmiennej xWyjscieAlarm. Jedna zmiana oddziałuje na cały program bez względu na to, jak często odwołujemy sie do tego wyjścia.

Deklaracja takich przykładowych zmiennych wyglądałaby następująco:

```
PROGRAM PLC_PRG
VAR
a: UINT;
bPamiecLicznika AT %MB10: BYTE;
xWyjscieAlarm AT %QX0.0: BOOL;
END_VAR
```

Zmiany są również konieczne w programie sterującym:

```
a:=a+1;
IF a=100 THEN
a:=0;
bPamiecLicznika:=bPamiecLicznika+1;
END_IF
IF a<50 THEN
xWyjscieAlarm:=TRUE;
ELSE
xWyjscieAlarm:=FALSE;
END_IF
```

Dla czytelności programu warto nadawać zmiennym takie nazwy, które będą bezpośrednio wskazywały na zadanie i typ; w przedstawionych poniżej przykładach kluczem do kodowania jest prefiks w nazwie zmiennej:

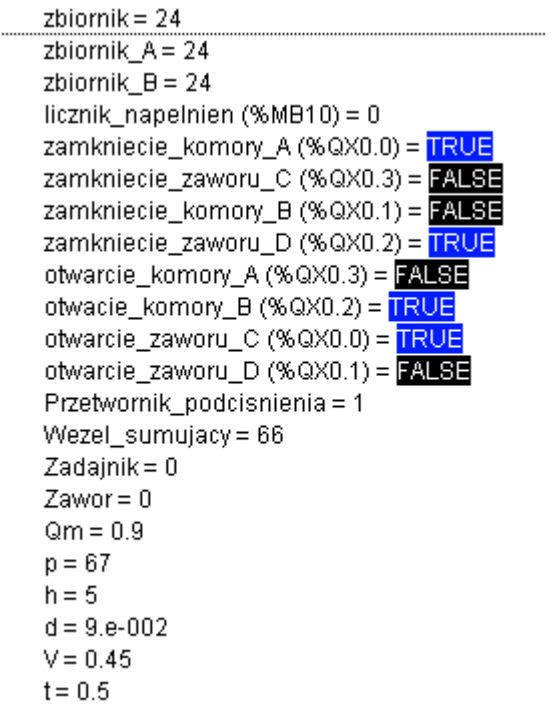
xZmienna1 – zmienna typu BOOL;

bZmienna2 – zmienna typu BYTE;
wZmienna3 – zmienna typu WORD;
dwZmienna4 – zmienna typu DWORD;
usiZmienna5 – zmienna typu USINT, itp.

Istnieje również możliwość bezpośredniego sterowania poszczególnymi bitami danej zmiennej. Przykładowo:

bZmienna2.0:=TRUE;
bZmienna2.1:=FALSE;
bZmienna2.3:=xZmienna1;

Po zadeklarowaniu wszystkich zmiennych i poprawnej kompilacji programu, można przystąpić do komunikacji ze sterownikiem za pomocą przycisku  , następnie należy użyć przycisku uruchamiającego sterownik  . Po uruchomieniu sterownika program wykonuje się automatycznie (rys. 101). Jeżeli w parametrach zakładki *Online* będzie zaznaczona opcja *Symulacja* – sterownik nie zostanie uruchomiony, będzie realizowana symulacja programu na komputerze (programatorze).



Rysunek 101. Widok przykładowego programu podczas pracy

Po zakończeniu pracy należy przerwać komunikację programu ze sterownikiem za pomocą przycisku *rozłączenie ze sterownikiem*  .

Przedstawiony przykład programowania PLC w języku tekstu strukturalnego ujawnił jego najważniejsze zasady, pozwolił zapoznać się z instrukcją warunkową (najczęściej używaną), sterowaniem wyjściami, programem realizującym samokasujący się licznik.

2.3.2. Programowanie algorytmu regulacji PID

W układach regulacji automatycznej regulator zastąpił operatora, który w układzie sterowania ręcznego kontrolował przebieg procesu regulowanego. Poprawne działanie układu regulacji zależy od doboru odpowiedniego typu regulatora do obiektu regulacji. Przyjęty typ regulatora określa zasadę regulacji, tzn. zależność wiążącą sygnał odchyłki e z sygnałem sterującym u. W regulatorach ciągłych ta zależność opiera się na proporcjonalności, całkowaniu i różniczkowaniu odchyłki e. Stąd oznaczenie typów regulatorów pochodzi od pierwszych liter angielskich nazw poszczególnych operacji realizowanych na nim (*P – proportional, I – integration, D – differentiation*). Istnieją również inne kombinacje regulatorów, jednak są one rzadziej wykorzystywane w aplikacjach sterujących. Wartość sterowania wypracowana przez regulator PID zależy proporcjonalnie od:

- uchybu;
- uchybu scałkowanego (zsumowanego);
- pochodnej (szybkości zmiany) uchybu.

PID stosowany jest do regulacji ciągłej, w programach dla PLC stanowi blok funkcjonalny.

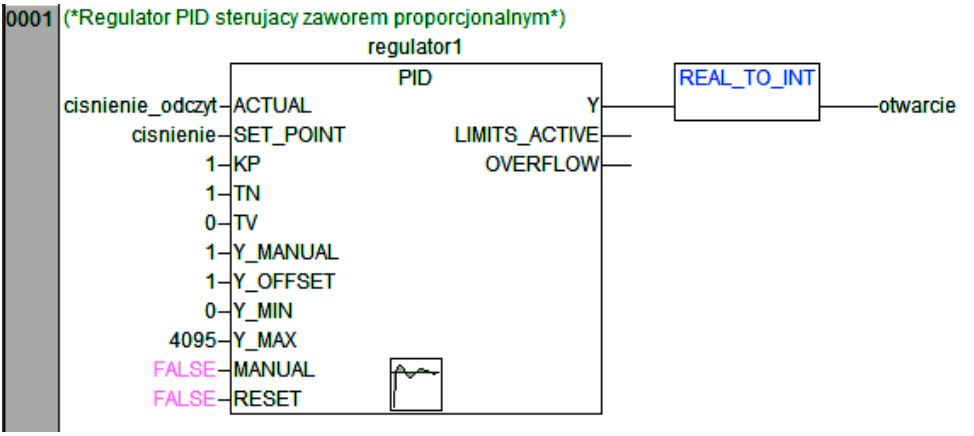
W programie CoDeSys regulator PID realizowany jest poprzez blok funkcyjny PID dostarczony przez bibliotekę Util.lib. Blok PID musi zostać sparametryzowany następującymi zmiennymi:

- wartość rzeczywista wielkości regulowanej;
- wartość zadana;
- współczynnik wzmocnienia części P;
- czas całkowania, odwrotny współczynnik wzmocnienia części I;
- czas różniczkowania, współczynnik wzmocnienia części D;
- offset wielkości nastawczej;
- dolna i górna granica wielkości nastawczej.

Blok funkcyjny regulatora PID przedstawiono na rys. 102. Opis wszystkich wejść/wyjść tego bloku przedstawiono w tabeli 17.

Wartość wyjściowa bloku PID jest typu REAL, dlatego na wyjściu regulatora zastosowano dodatkowo blok funkcyjny REAL_TO_INT przekształcający wartość typu REAL na wartość całkowitą typu INT. Tego typu rozwiązanie zalecane jest dla sterowania urządzeń wykonawczych za pomocą sygnału analogowego.

Y_OFFSET, Y_MIN i Y_MAX są stosowane do zmiany wielkości nastawczej w zadany przedział. Przy pomocy MANUAL można uzyskać przełączenie na tryb ręczny, natomiast za pomocą RESET można ponownie zainicjalizować regulator.



Rysunek 102. Blok funkcyjny regulatora PID

Tabela 17. Wejścia i wyjścia modułu

Zmienna	Typ danych	Opis
ACTUAL	REAL	Wartość rzeczywista wielkości regulowanej
SET_POINT	REAL	Wartość zadana, wielkość przewodnia
KP	REAL	Współczynnik proporcjonalności (wzmocnienia części P)
TN	REAL	Czas całkowania, współczynnik wzmocnienia części I; dane w [s], np. (0.5) dla 500 (ms)
TV	REAL	Czas różniczkowania, wzmocnienie części D; dane w (s)
Y_MANUAL	REAL	Wartość wydawana manualnie; wydawana do Y, jeśli MANUAL = TRUE
Y_OFFSET	REAL	Offset wartości nastawczej
Y_MIN, Y_MAX	REAL	Dolna lub górna granica wartości nastawczej Y, po przekroczeniu w dół lub w górę, wyjście LIMITS_ACTIVE jest ustawiane na TRUE i Y utrzymywane w obrębie granic. Nadzorowanie jest aktywne tylko, jeśli Y_MIN<Y_MAX
MANUAL	BOOL	Włącza ręczne zadawanie wartości nastawczej za pomocą Y_MANUAL (TRUE) lub wyłącza (FALSE)
RESET	BOOL	TRUE ponownie inicjalizuje regulator. W trakcie inicjalizacji Y = Y_OFFSET
Y	REAL	Wartość nastawcza obliczana przez moduł
LIMITS_ACTIVE	BOOL	Ukazuje za pomocą TRUE, że Y przekroczyło dozwolony zakres (Y_MIN, Y_MAX)
OVERFLOW	BOOL	Wskazuje za pomocą TRUE przepełnienie

W normalnym trybie (MANUAL = RESET = LIMITS_ACTIVE = FALSE) regulator wylicza błąd regulacji *e* jako różnicę pomiędzy SET_POINT a ACTUAL, tworzy jego

pochodną po czasie δe/δt i następnie zapisuje te wartości. Wartość nastawcza (Y) obliczana jest w następujący sposób:

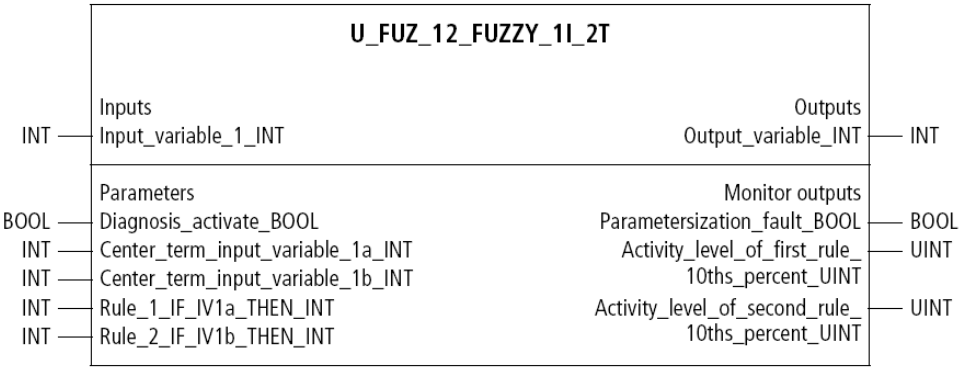
$$Y = KP \cdot \left(\Delta + \frac{1}{TN} \cdot \int edt + TV \frac{\delta}{\delta t}\right) + Y\_OFFSET \quad (1)$$

Oprócz części P na wielkość nastawczą mają wpływ również aktualna zmiana błędu regulacji (część D) i historia błędu regulacji (część I). Regulator PID można zamienić na regulator PI zmieniając ustawienia TV=0. Przy pomocy dodatkowej części I, gdy wystąpi niewłaściwa parametryzacja regulatora oraz całka błędu Δ jest za duża, może dojść do przepełnienia. Dla zabezpieczenia się przed taką sytuacją powstało wyjście BOOLean OVERFLOW, które w przedstawionym przypadku ma wartość TRUE. Taka sytuacja ma miejsce, gdy system regulacji jest niestabilny na skutek nieodpowiedniej parametryzacji.

2.3.3. Programowanie algorytmu sterowania Fuzzy Logic

Oprogramowanie CoDeSys oferuje funkcję sterowania za pomocą logiki rozmytej przez grupę bloków funkcyjnych Fuzzy Systems, która należy do biblioteki Closed-Loop Control Toolbox. Bloki funkcyjne sterowania rozmytego są skonstruowane tak, aby możliwe było ich użycie bez specjalistycznej wiedzy na temat logiki rozmytej. Wszystkie operacje składowe sterowania rozmytego (rozmywanie, wnioskowanie, wyostrzanie oraz dobór odpowiedniego kształtu funkcji przyporządkowania) wykonywane są automatycznie. Programista zobowiązany jest jedynie do określenia reguł sterowania rozmytego (Juszka i Tomasik, 2006).

Podstawowym blokiem funkcyjnym grupy Fuzzy Systems jest blok U_FUZ_12_FUZZY_1I_2T (rys. 103). Posiada on jedno wejście, jedno wyjście oraz dwie reguły sterowania.



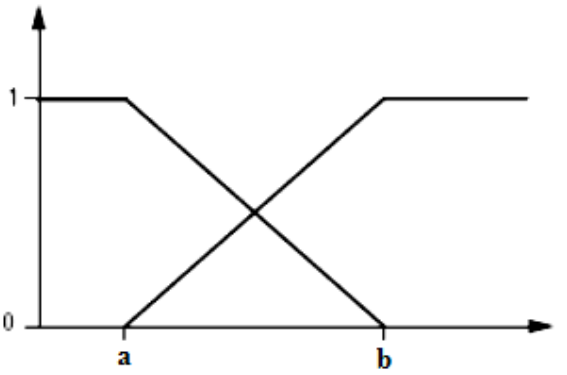
Rysunek 103. *Blok funkcyjny FuzzyLogic*

W tabeli 18 przedstawiono opis wejść oraz wyjść tego bloku funkcjonalnego.

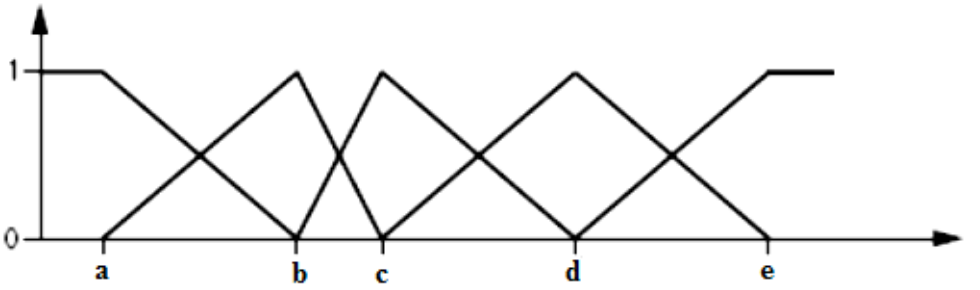
Tabela 18. Wejścia i wyjścia moduły FuzzyLogic

Oznaczenie	Znaczenie	Zakres wartości
	Wejścia bloku	
Input_variable_1_INT	Zmienna wejściowa	–32768 do 32767
	Parametry bloku	
Diagnosis_activate_BOOL	Aktywacja trybu diagnostycznego	0/1
Center_term_input_variable_1a_INT	Wartość „1a” zmiennej wejściowej	–32768 do 32767
Center_term_input_variable_1b_INT	Wartość „1b” zmiennej wejściowej	–32768 do 32767
Rule_1_IF_IV1a_THEN_INT	Reguła 1 określa wartość zmiennej wyjściowej przy wartości „1a” zmiennej wejściowej	–32768 do 32767
Rule_2_IF_IV1b_THEN_INT	Reguła 2 określa wartość zmiennej wyjściowej przy wartości „1b” zmiennej wejściowej	–32768 do 32767
	Wyjścia	
Output_variable_INT	Zmienna wyjściowa	–32768 do 32767
	Wyjścia diagnostyczne	
Parameterization_fault_BOOL	Błąd konfiguracji	0/1
Activity_level_of_first_rule_ 10ths_percent_UINT	Wskaźnik aktywacji pierwszej reguły	0 do 1000
Activity_level_of_second_rule_ 10ths_percent_UINT	Wskaźnik aktywacji drugiej reguły	0 do 1000

Bloki funkcyjne Fuzzy Systems Automatycznie tworzą funkcję przyporządkowania wartości wejściowych. Na rysunkach 104 oraz 105 przedstawiono przykładowe funkcje przyporządkowania. Zastosowanie większej liczby reguł poprawia dokładność pracy układu (lepsze dopasowanie), jednak skutkuje wydłużeniem jego czasu reakcji ze względu na wzrost liczby obliczeń koniecznych do wyznaczenia sygnału sterującego.



Rysunek 104. Funkcja przyporządkowania dla bloku z dwoma regułami

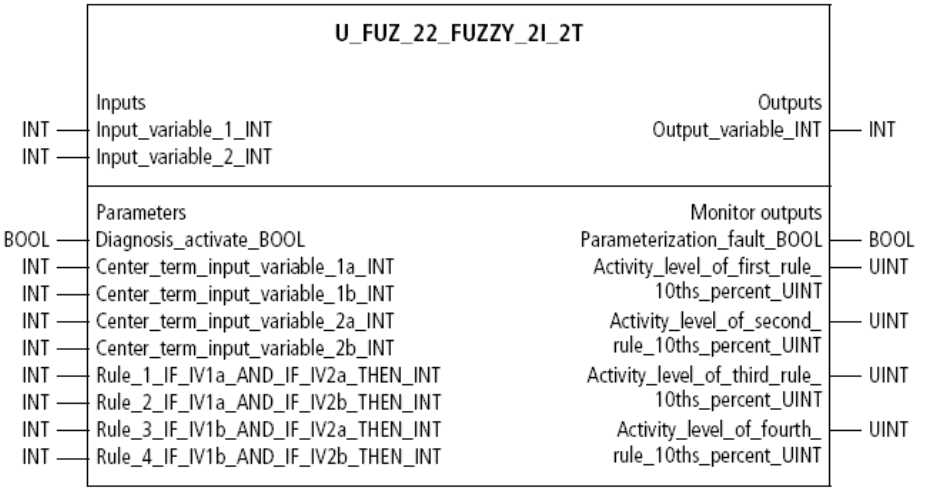


Rysunek 105. Funkcja przyporządkowania dla bloku z pięcioma regułami

Środowisko CoDeSys poza podstawową funkcją Fuzzy Logic oferuje także bardziej rozbudowane wersje. Przykładem takiej funkcji jest blok posiadający dwie zmienne wejściowe (rys. 106). Dzięki takiej funkcji możliwe jest zbudowanie sterowania rozmytego, opartego na dwóch wartościach wejściowych.

Funkcja z dwoma wejściami posiada analogiczne parametry jak funkcja z jednym wejściem. Różni się jednak regułami:

- Rule_1_IF_IV1a_AND_IF_IV2a – Reguła 1 określa wartość zmiennej wyjściowej przy wartości „1a" zmiennej wejściowej 1 oraz wartości „2a" zmiennej wejściowej 2;
- Rule_2_IF_IV1a_AND_IF_IV2b – Reguła 2 określa wartość zmiennej wyjściowej przy wartości „1a" zmiennej wejściowej 1 oraz wartości „2b" zmiennej wejściowej 2;
- Rule_3_IF_IV1b_AND_IF_IV2a – Reguła 3 określa wartość zmiennej wyjściowej przy wartości „1b" zmiennej wejściowej 1 oraz wartości „2a" zmiennej wejściowej 2;
- Rule_4_IF_IV1b_AND_IF_IV2b – Reguła 4 określa wartość zmiennej wyjściowej przy wartości „1b" zmiennej wejściowej 1 oraz wartości „2b" zmiennej wejściowej 2.



Rysunek 106. Blok funkcyjny FuzzyLogic z dwoma zmiennymi wejściowymi

Aby przedstawić możliwości sterowania rozmytego, zaprojektowano przykładowy program, który ma za zadanie utrzymywać odpowiednią temperaturę centralnego ogrzewania w zależności od temperatury na zewnątrz oraz temperatury zadanej wewnątrz budynku:

- zmienną wejściową 1 jest temperatura zewnętrzna;
- zmienną wejściową 2 jest zadana temperatura wewnętrzna;
- zmienną wyjściową jest temperatura czynnika grzewczego w bojlerze;
- wartość „1a" temperatury zewnętrznej wynosi 0°C;
- wartość „1b" temperatury zewnętrznej wynosi 10°C;
- wartość „2a" zadanej temperatury wewnętrznej wynosi 15°C;
- wartość „2b" zadanej temperatury wewnętrznej wynosi 20°C.

Dobrano następujące reguły sterowania:

- jeśli temperatura zewnętrzna przyjmuje wartość „1a" oraz zadana temperatura wewnętrzna przyjmuje wartość „2a", to temperatura bojlera jest równa 75°C;
- jeśli temperatura zewnętrzna przyjmuje wartość „1a" oraz zadana temperatura wewnętrzna przyjmuje wartość „2b", to temperatura bojlera jest równa 80°C;
- jeśli temperatura zewnętrzna przyjmuje wartość „1b" oraz zadana temperatura wewnętrzna przyjmuje wartość „2a", to temperatura bojlera jest równa 60°C,
- jeśli temperatura zewnętrzna przyjmuje wartość „1b" oraz zadana temperatura wewnętrzna przyjmuje wartość „2b", to temperatura bojlera jest równa 75°C.

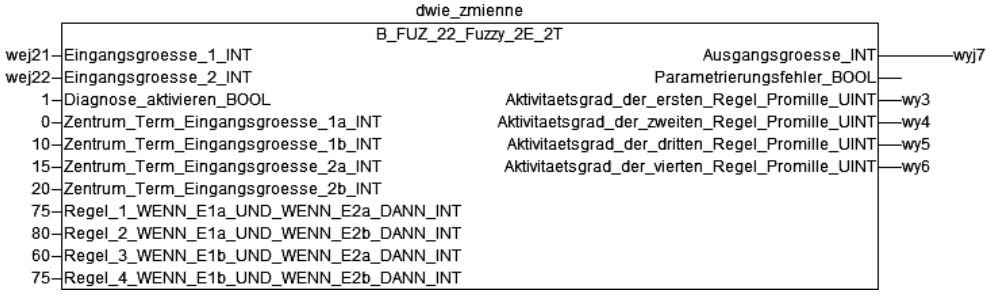
W tabeli 19 przedstawiono założone wartości wyjściowe funkcji (temperaturę czynnika grzewczego w bojlerze) w zależności od wartości wejściowych (temperatury zewnętrznej oraz zadanej temperatury wewnętrznej) (Tomasik i in., 2012d).

Tabela 19. Reguły sterowania temperaturą bojlera

Zadana temperatura wewnętrzna (°C)	Temperatura zewnętrzna (°C)	
	0	10
15	TB = 75 (Reguła 1)	TB = 60 (Reguła 3)
20	TB = 80 (Reguła 2)	TB = 75 (Reguła 4)

TB – temperatura bojlera (°C)

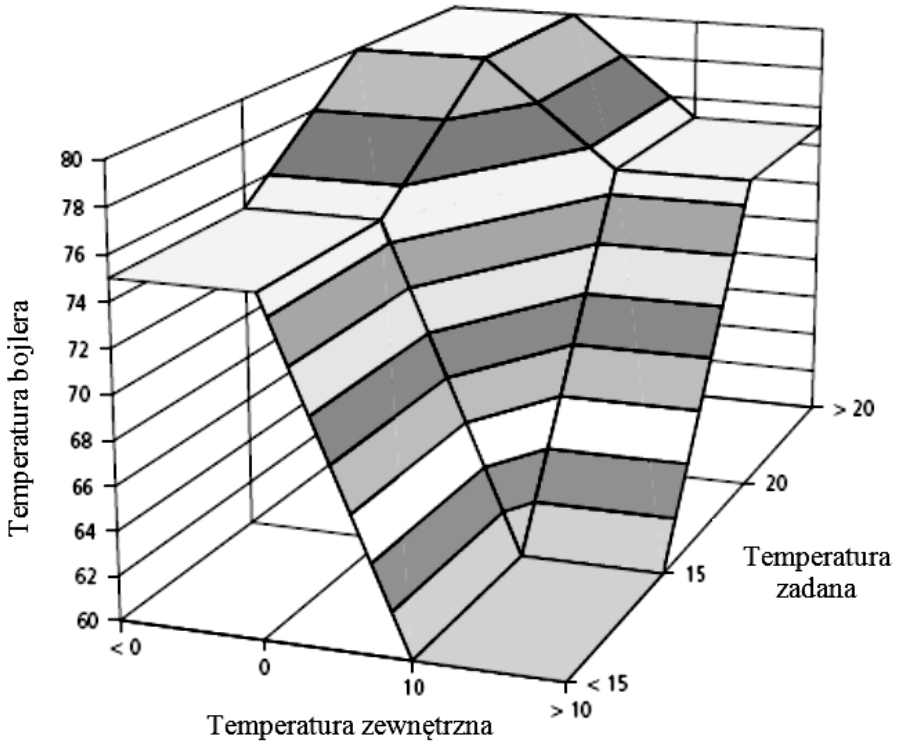
Sparametryzowany blok funkcyjny dla powyższego przykładu przedstawiono na rysunku 107.



Rysunek 107. Przykładowa realizacja sterowania rozmytego

Na rysunku 108 przedstawiono wykres obrazujący logikę sterowania za pomocą bloku Fuzzy.

W celu zobrazowania działania bloku funkcjonalnego opracowano wizualizację widoczną na rysunku 109. Z poziomu wizualizacji można ustawiać temperaturę zewnętrzną oraz wartość zadaną temperatury wewnętrznej w przyjętych zakresach, a także obserwować pożądaną temperaturę bojlera, będącą wynikiem działania funkcji i symulacji dla modelu matematycznego tego obiektu. Dodatkowo widoczne są funkcje diagnostyczne bloku FuzzyLogic, dzięki czemu w łatwy sposób można określić poprawność doboru parametrów sterowania (Tomasik i in., 2012). Poziom aktywności danej reguły rozmytej odzwierciedla stopień, w jakim reguła jest zaangażowana w generowanie zmiennej wyjściowej. Podgląd online na te wartości rozmyte pozwala na modyfikację reguł i procesu rozmywania (przyporządkowania wartości rozmytych) celem uzyskania jak najlepszego efektu.

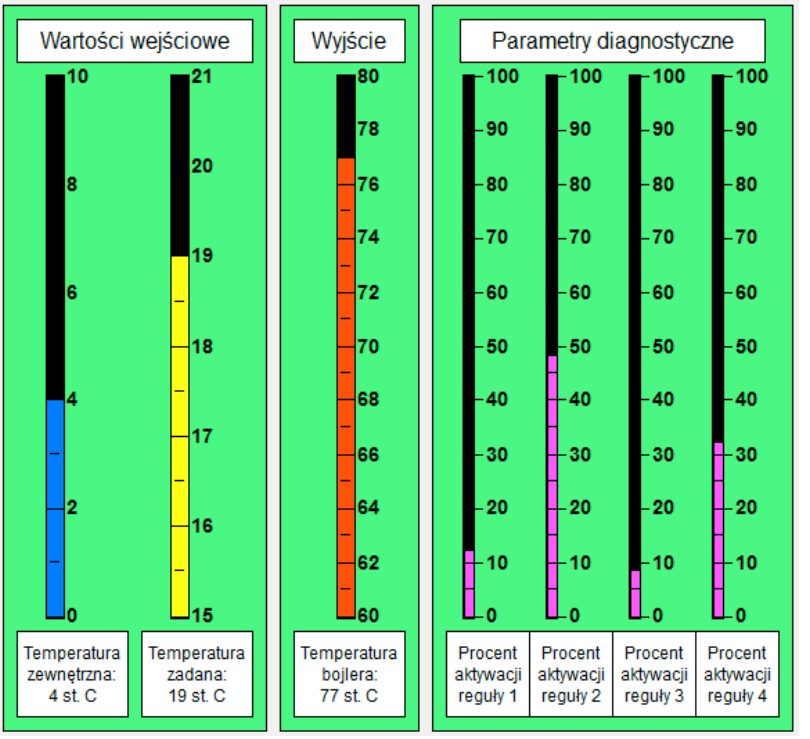


Rysunek 108. Wykres zależności temperatury zadanej w funkcji dwóch zmiennych

Oprogramowanie CoDeSys posiada także zintegrowany edytor wizualizacji, umożliwiający użytkownikowi tworzenie gotowych okien wizualizacji w kontekście programowania aplikacji w jednym i tym samym interfejsie. Umożliwia ono tworzenie grafiki z obiektami oraz ich animację sterowaną zmiennymi. Wizualizacja zintegrowana w CoDeSys nie potrzebuje listy tagów i może odwoływać się bezpośrednio do zmiennych ze sterowni-

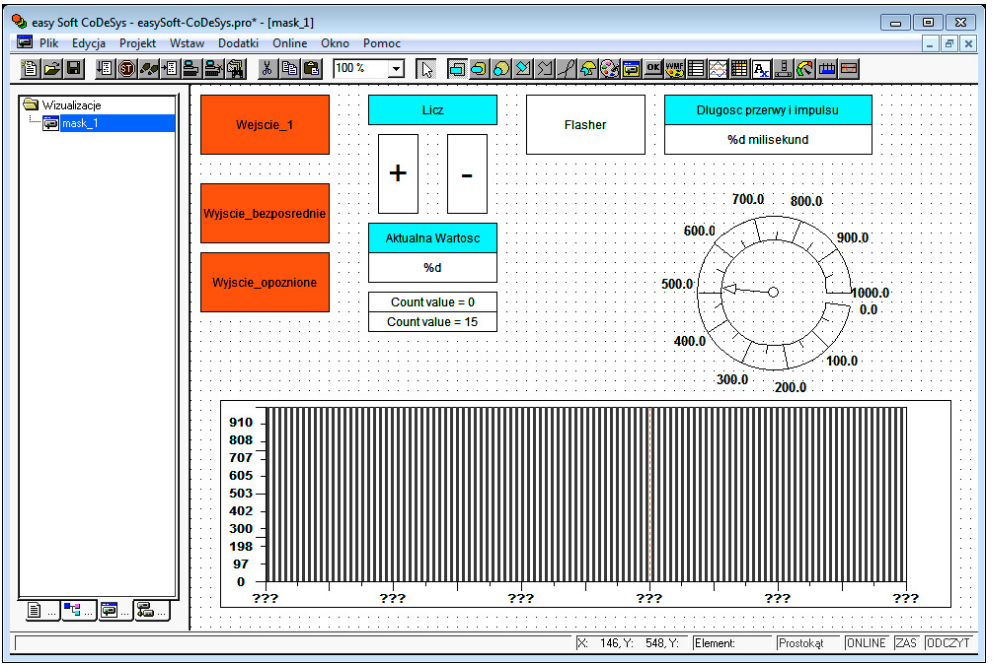
ka. System nie posiada trudnej w konfiguracji warstwy OPC lub DDE, gdyż komunikacja odbywa się poprzez ten sam mechanizm, jaki wykorzystuje system programowania.

Powiązanie sterownika i wizualizacji umożliwia ponadto tworzenie wariantów przebiegu, w których można zrezygnować z tradycyjnej prezentacji na komputerze z monitorem.



Rysunek 109. Wizualizacja przykładu sterowania rozmytego

Program CoDeSys-HMI, będący aplikacją Win32, umożliwia wyświetlanie masek wizualizacji na komputerze bez kompletnego interfejsu programistycznego. CoDeSys może opcjonalnie, na podstawie informacji o wizualizacjach, generować opis XML, który razem z apletem Java (aplet – program utworzony w języku JAVA, przetwarzany przez przeglądarki) jest przechowywany w pamięci sterownika. Po wysłaniu przez protokół TCP/IP może zostać wyświetlony w oknie przeglądarki. Dla sterowników z wbudowanym wyświetlaczem można z systemu programowania załadować do systemu docelowego wizualizację łącznie z programem sterujacym. Na rysunku 110 przedstawiono oprogramowanie CoDeSys w trybie tworzenia wizualizacji, widoczne są przykładowe elementy możliwe do zastosowania w programowaniu masek wizualizacji.



Rysunek 110. Okno środowiska CoDeSys – edytor wizualizacji

CZĘŚĆ TRZECIA
– PRZYKŁADY ZASTOSOWAŃ STEROWNIKÓW PLC W WYBRANYCH PROCESACH PRODUKCYJNYCH

1. STEROWNIKI PLC W INŻYNIERII PRODUKCJI

W rozdziale przedstawionych zostanie kilka przykładów programów sterujących procesami inżynierii rolniczej. Programy wykonano dla różnych sterowników PLC, wykorzystywane są w trakcie realizacji zajęć dydaktycznych.

1.1. Inżynieria produkcji roślinnej

Celem sterowania w obiektach produkcyjnych może być utrzymywanie optymalnych w danym etapie i fazie rozwoju roślin: przebiegów temperatury i wilgotności względnej powietrza w obrębie szklarni lub tunelu foliowego, stężenia dwutlenku węgla w powietrzu, nawodnienia i nawożenia podłoża uprawowego oraz natężenia światła (Kurpaska, 2007). Głównymi celami automatyzacji procesów w uprawach pod osłonami są:
- zwiększenie plonowania roślin;
- poprawa jakości produktów finalnych;
- minimalizacja zużycia energii;
- zmniejszenie nakładów pracy ludzkiej;
- poprawa bezpieczeństwa pracy;
- eliminacja lub zminimalizowanie prac monotonnych i nużących, wykonywanych przez obsługę procesu.

1.1.1. Zastosowanie sterownika PLC w uprawach pod osłonami

Automatyczne sterowanie urządzeniami odpowiedzialnymi za parametry mikroklimatu w szklarni staje się powoli standardem. Sterownie mikroklimatem szklarni ogranicza się głównie do utrzymywania zadanych wartości parametrów mikroklimatu oraz włączania urządzeń bądź systemów w określonym czasie, dlatego też programy do sterownia uprawami pod osłonami nie są skomplikowane, stąd nie korzystają z zaawansowanych funkcji czy bloków funkcyjnych. Ze względu na specyfikę aplikacji oraz dla zapewniania przejrzystości aplikacji, koncepcyjny system sterowania szklarniową uprawą pomidora przedstawiony poniżej zaprogramowano w języku schematów drabinkowych (LD).

Sterownik PLC jest głównym elementem systemu sterowania mikroklimatem, pozostałe to czujniki pomiarowe oraz mechanizmy wykonawcze. Czujniki odpowiadają za pomiary następujących wartości:
- temperatury powietrza wewnątrz szklarni;
- wilgotności względnej powietrza wewnątrz szklarni;
- kierunku i prędkości wiatru;
- opadów atmosferycznych;
- natężenia oświetlenia;
- stężenia CO_2 w powietrzu wewnątrz szklarni (Kurpaska, 2007).

Do pomiaru temperatury powietrza wewnątrz szklarni najczęściej używa się rezystancyjnych czujników temperatury PT100 klasy A. Wartość temperatury jest wyznaczana na podstawie zmiany rezystancji. Wartości opadów atmosferycznych, nasłonecznienia oraz

kierunku i prędkości wiatru pozyskiwane są ze stacji pogodowej zamontowanej na szczycie szklarni. W skład stacji wchodzą:
- pluwiograf (mierzący ilość, czas trwania oraz intensywność opadów);
- anemometr (mierzący prędkość i kierunek wiatru);
- zestaw termometrów (mierzących temperaturę zewnętrzną).

Dane o natężeniu oświetlenia i promieniowaniu słonecznym odczytywane są za pomocą fotorezystorów, w czujnikach tych wykorzystano zjawisko zmiany rezystancji pod wpływem promieniowania. Wartość wilgotności mierzona jest za pomocą higrometru. Podczas zmiany wilgotności powietrza zmienia się pojemność elektryczna sondy. Na podstawie tej zmiany obliczana jest wilgotność. Ostatnim zestawem czujników zamontowanych w szklarni jest zestaw czujników stężenia dwutlenku węgla, określają one stężenie CO_2 na podstawie zjawiska pochłaniania promieniowania podczerwonego przez gazy.

Mechanizmy wykonawcze sterują:
- wietrzeniem dachowym, z uwzględnieniem siły wiatru;
- zasłoną cieniująco-termoizolacyjną;
- systemem ogrzewania powietrza;
- systemem zamgławiaczy;
- stężeniem CO_2 w powietrzu;
- sygnalizacją wystąpienia stanów alarmowych.

Sterowanie systemem grzewczym następuje w oparciu o temperaturę powietrza w szklarni. Zmiana temperatury możliwa jest dzięki odpowiedniemu ustawieniu prędkości pracy pomp obiegowych oraz ustawieniu zaworów mieszających, które dodają niezbędną ilość gorącej wody z kotła. Jeżeli system bazuje na elektrycznym ogrzewaniu powietrza, sterownik włącza lub wyłącza odpowiednią część nagrzewnic. Do chłodzenia szklarni stosuje się wentylatory i wietrzniki dachowe, dzięki czemu możliwe jest usunięcie zbyt ciepłego powietrza ze szklarni. Zwiększenie wilgotności powietrza możliwe jest poprzez system nawilżaczy i zamgławiaczy, aby obniżyć wilgotność stosuje się wietrzenie szklarni. Zwiększanie i zmniejszanie zaciemnienia możliwe jest dzięki silnikom rozsuwającym ekrany, które posiadają dodatkowe funkcje izolacyjne. W przypadku niedostatecznego nasłonecznia włączany jest system lamp doświetlających. Sygnały o stanach alarmowych wyświetlane są na tablicach synoptycznych lub na ekranach komputerów dzięki programom do wizualizacji procesu (Kurpaska, 2007; Krzesiński, 2005).

W przedstawianym przykładzie programu sterującego uprawą szklarniową przyjęto pewne wymagania i uproszczenia dotyczące warunków produkcyjnych. Założono, że w szklarni utrzymana jest stała temperatura powietrza – 21°C. Jeśli temperatura w szklarni wzrośnie, opuszane są rolety cieniujące; jeżeli temperatura w szklarni spadnie, rolety zostają podniesione, aby umożliwić ogrzanie szklarni przez promienie słoneczne. Jeżeli po upływie 10 minut temperatura nie osiągnie poziomu zadanego, włączany jest system ogrzewania. Zarówno chłodzenie jak i ogrzewanie szklarni prowadzone jest do momentu osiągnięcia zadanej temperatury. Przy włączonym trybie pracy automatycznej rolety mogą być podniesione jedynie w godzinach od 8 do 18. To rozwiązanie zmniejsza straty ciepła w nocy. Rolety posiadają zabezpieczenie przed silnym wiatrem. W trybie ręcznym praca rolet sterowana jest przy użyciu klawiatury sterownika. System wentylacyjny składa się z zespołu wietrzników szczytowych i okien bocznych. Możliwy jest tryb pracy automatycznej oraz ręcznej. W trybie automatycznym okna otwierane są co trzy godziny na 15 minut oraz na 30 minut o godzinie 6.00, 12.00, 18.00. Okna nie są otwierane podczas sil-

nego wiatru oraz po 10 sekundach od wystąpieniu opadów atmosferycznych. Ponowne otwarcie okien możliwe jest po ustaniu opadów. Okna wyposażone są w automatyczne wyłączniki krańcowe. System nawadniania włącza się na 3 minuty co 3 godziny, zaczynając od północy; dodatkowo jeśli zawartość wody w podłożu spadnie poniżej 35%, włączy się system nawadniający, który wyłączy się przy osiągnięciu wilgotności na poziomie 50%.

W pierwszym programie przyjęto trzy tryby pracy dla 3 typów roślin:

- pierwszy tryb nawadniania bazuje na wskazaniach czujnika pływakowego, włączanie i wyłączanie sterowane jest na podstawie wskazań minimalnego i maksymalnego poziomu;
- drugi tryb nawadniania umożliwia nawadnianie roślin przez 3 minuty o godzinie 6.00 i 20.00;
- trzeci tryb nawadniania bazuje na czujniku zmierzchowym i nawadnia rośliny co drugi dzień przez 2 min.

W aplikacji wykorzystano:

- cztery wejścia cyfrowe (łącznik pływakowy (max.), łącznik pływakowy (min.), wyłącznik zmierzchowy i wyłącznik automatyczny);
- trzy wyjścia cyfrowe (włączniki pomp dla każdej sekcji);
- dwa parametry (czas nawadniania roślin typu drugiego i trzeciego).

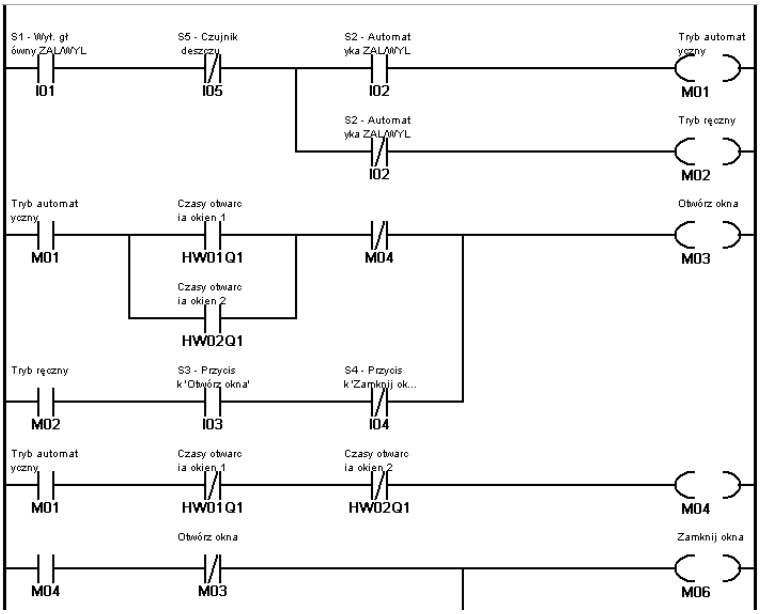
W drugiej części programu odpowiedzialnego za sterowanie temperaturą przyjęto dwa tryby pracy:

- ręczny – przeznaczony do prac systemowych, pozwala ręcznie opuszczać i podnosić rolety za pomocą przycisków sterownika;
- automatyczny, w którym zadaniem sterownika jest utrzymywanie temperatury na poziomie 17°C z buforem wynoszącym 2°C. Przy wzroście temperatury do 19°C – rolety zostają opuszczone, przy spadku temperatury do 15°C – rolety zostają podniesione oraz jeśli po 10 minutach temperatura nie podniesie się do żądanej wartości włącza się system ogrzewania. Wyłącza się on bezzwłocznie przy osiągnięciu 17°C. Dodatkowo rolety zabezpieczone są przed silnym wiatrem poprzez ich zwinięcie.

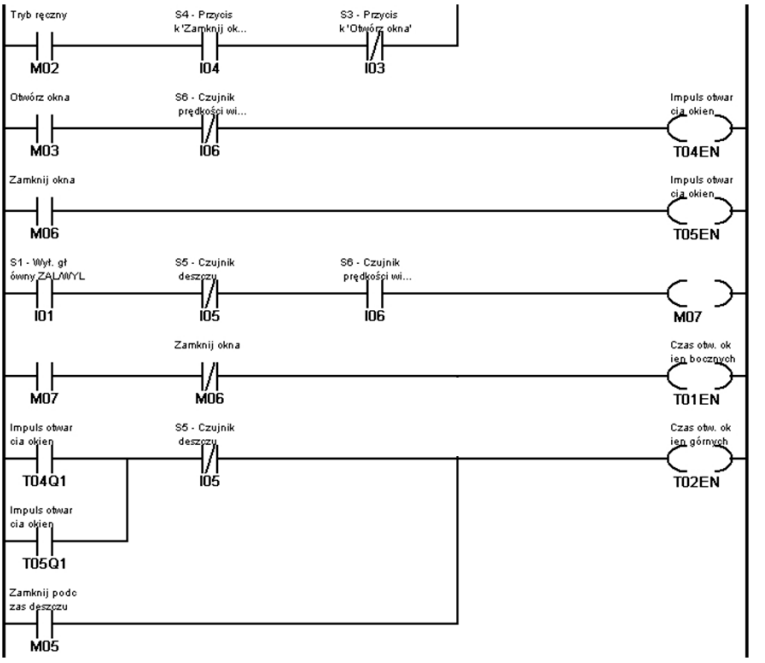
W aplikacji wykorzystano:

- dwa wejścia analogowe (czujnik temperatury, czujnik wiatru);
- trzy wyjścia cyfrowe (dwa silniki do podnoszenia i opuszczenia rolet, włącznik ogrzewania elektrycznego);
- osiem parametrów (trzy zaprogramowane czasy – opóźnienia włączenia ogrzewania, interwał między pomiarami prędkości wiatru, czas trwania pomiaru prędkości wiatru, licznik impulsów z czujnika pomiaru prędkości wiatru, trzy wartości temperatury – 15, 17, 19°C oraz czas otwierania rolet.

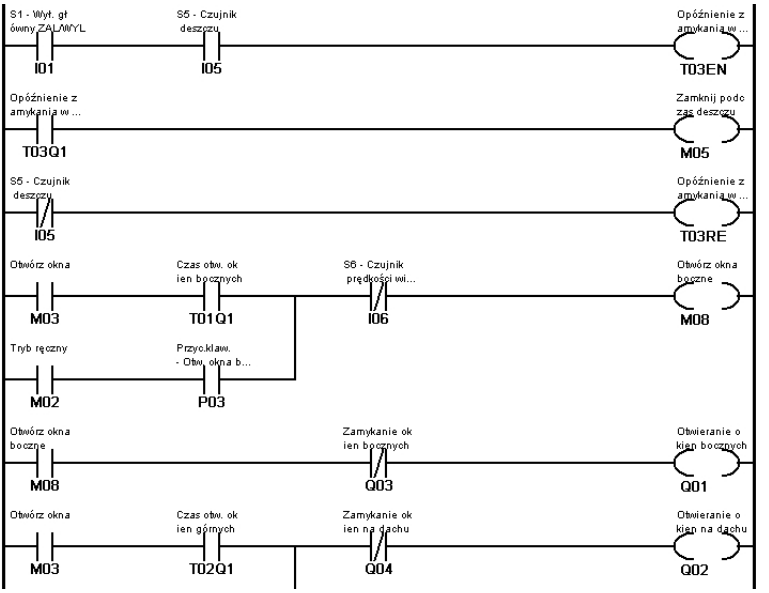
Aplikacja sterująca uprawą została zaprogramowana w języku drabinkowym za pomocą programu EasySoft-Moeller. Podzielono ją na trzy części według przydzielonych funkcji. Na rysunkach 111–114 przedstawiono fragment aplikacji odpowiedzialny za sterowanie wentylacją szklarni. Poprzez wejście I01 załączany jest cały system sterowania, logiczna „1" w I02 uruchamia tryb automatyczny. Tygodniowe zegary sterujące HW01 i HW02 sterują cyklicznym otwieraniem okien.



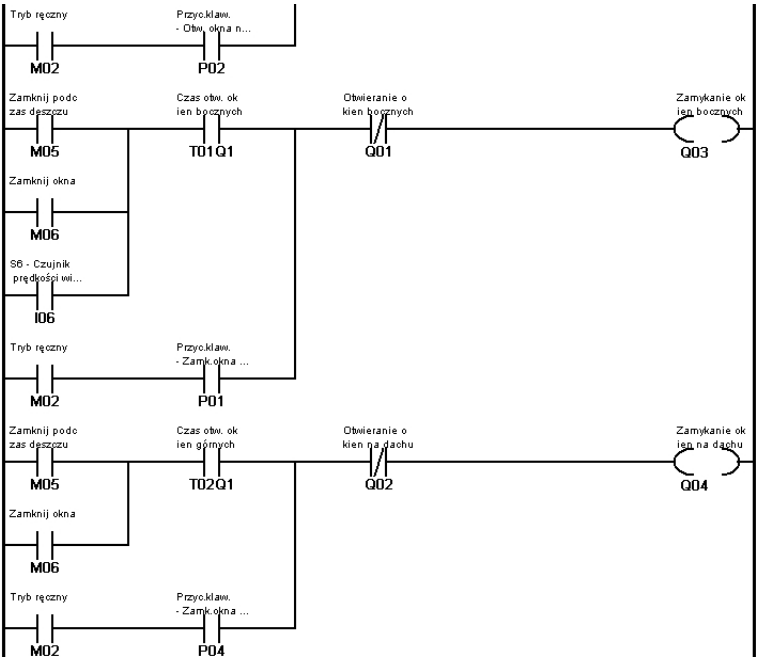
Rysunek 111. Program sterujący wentylacją szklarni – część 1



Rysunek 112. Program sterujący wentylacją szklarni – część 2



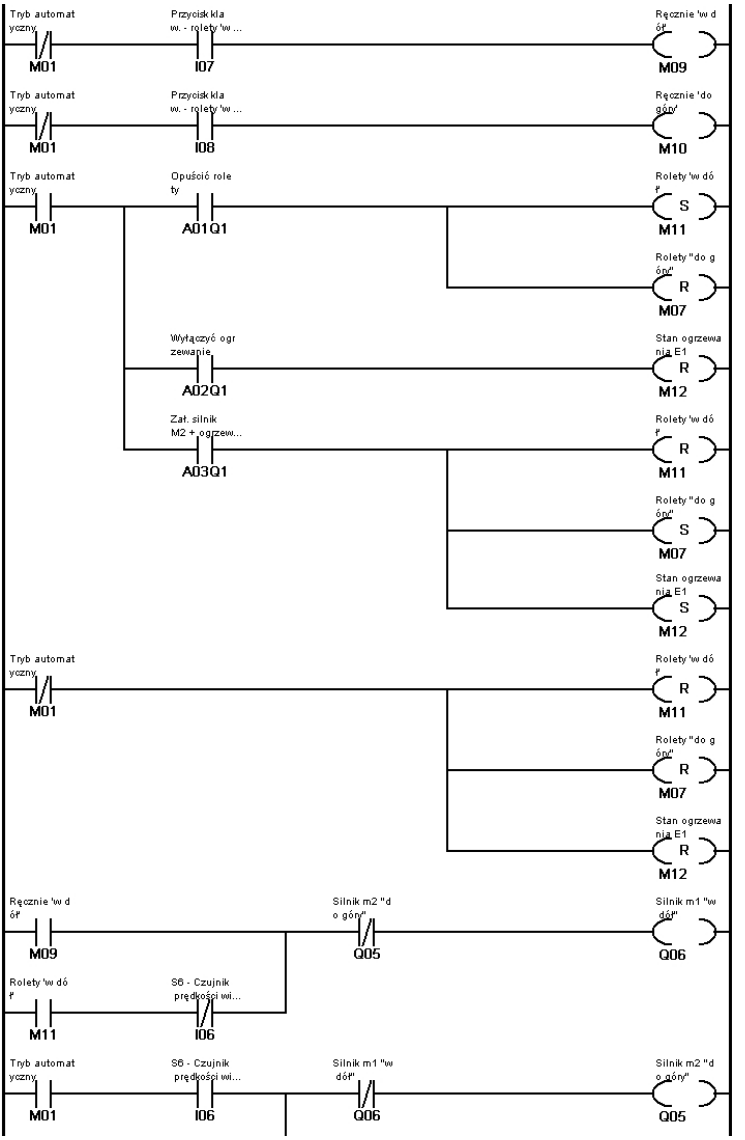
Rysunek 113. Program sterujący wentylacją szklarni – część 3



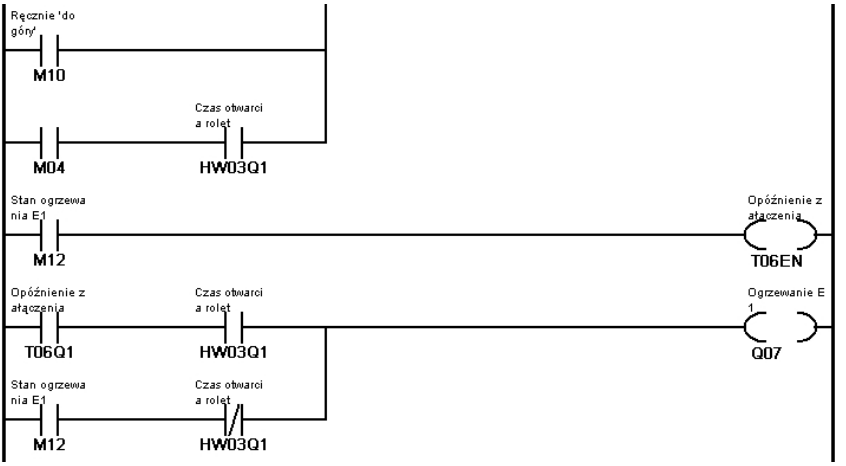
Rysunek 114. Program sterujący wentylacją szklarni – część 4

W przypadku wystąpienia deszczu zamykane są okna górne (czujnik I05 – rys. 113), natomiast jeżeli pojawi się wiatr, zamykane są również okna boczne (czujnik I06 – rys. 112). Przyciskami P01, P02, P03 oraz P04 można ręcznie otwierać i zamykać okna przy założeniu, że warunki atmosferyczne pozwalają na ich otwarcie.

Na rysunkach 115 i 116 przedstawiono fragment aplikacji odpowiedzialny za regulację temperatury w szklarni.



Rysunek 115. Program odpowiedzialny za regulację temperaturą w szklarni – część 1



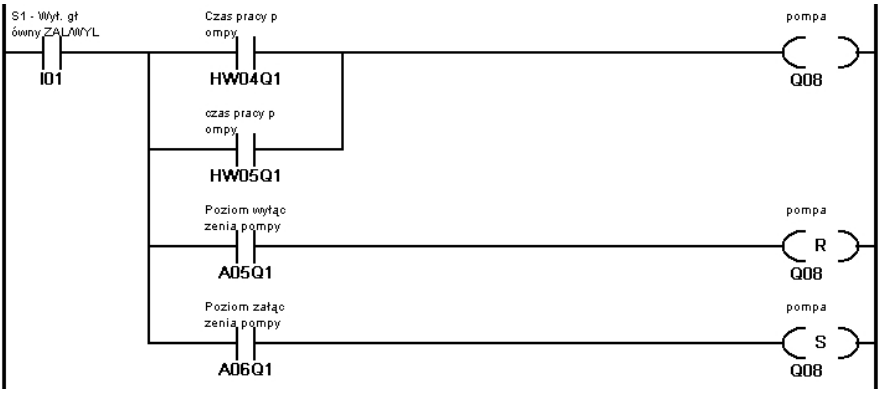
Rysunek 116. Program odpowiedzialny za regulację temperatury w szklarni – część 2

Elementami wykonawczymi są rolety oraz system ogrzewania. Jeżeli wybrany zostanie tryb automatyczny – sterownik uruchamia zwijanie lub rozwijanie rolet oraz zał/wył ogrzewanie. Dodatkowo w trybie ręcznym poprzez wejścia I07 oraz I08 można zmieniać ich położenie. W trybie pracy automatycznej sterownik porównuje aktualną temperaturę z zadaną (komparator wielkości analogowych A01, A02, A03). Jeżeli temperatura jest niższa od założonej – rolety są podnoszone, aby promienie słoneczne ogrzały powietrze wewnątrz szklarni. Natomiast po upływie 10 minut jeśli temperatura nie osiągnie zadanej wartości – włączany jest system ogrzewania. W przypadku kiedy temperatura przekroczy zadaną – opuszane są rolety. Sterownik sprawdza również prędkość wiatru, aby chronić rolety przed uszkodzeniem (tylko przy otwartych oknach); w sytuacji kiedy zostanie przekroczona dopuszczalna prędkość, rolety są zwijane.

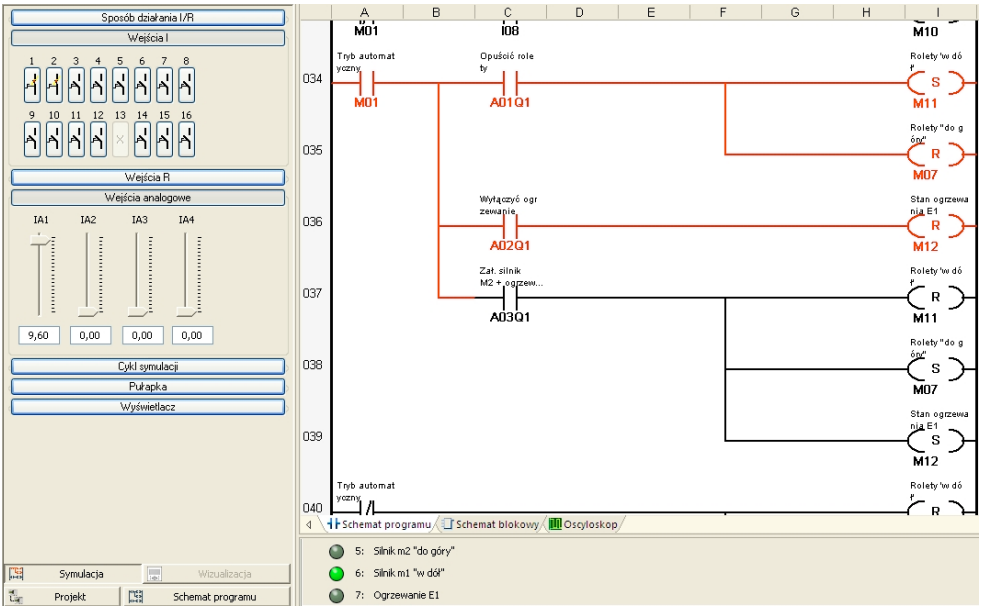
Na rysunku 117 przedstawiono fragment aplikacji sterujący cyklicznym nawadnianiem upraw z pomiarem wilgotności gleby (A05 i A06). Nawadnianie włączane jest tygodniowym zegarem sterującym na zadany czas, wyłączenie może nastąpić przez wykrycie nadmiernej wilgotności podłoża (Reset Q08), ponadto system nawadniania włączany jest również przy spadku wilgotności podłoża poniżej zadanej wartości.

Działanie programu w trybie symulacji przedstawiają dwa rysunki 118 i 119. Pierwszy przykład obrazuje reakcje sterownika na temperaturę wyższą niż zadaną, przy włączonym trybie automatycznym podczas bezwietrznej pogody. Komparator A01Q1 uruchomił opuszczanie rolet (marker M11-S), natomiast A02Q1 zablokował (wyłączył) możliwość uruchomienia ogrzewania (marker M12-R). Na rysunku 119 widoczne jest aktywne wyjście Q06 informujące o uruchomieniu trybu opuszczania rolet w szklarni. Czerwony kolor linii obrazuje przepływający sygnał w programie (prąd na drabince).

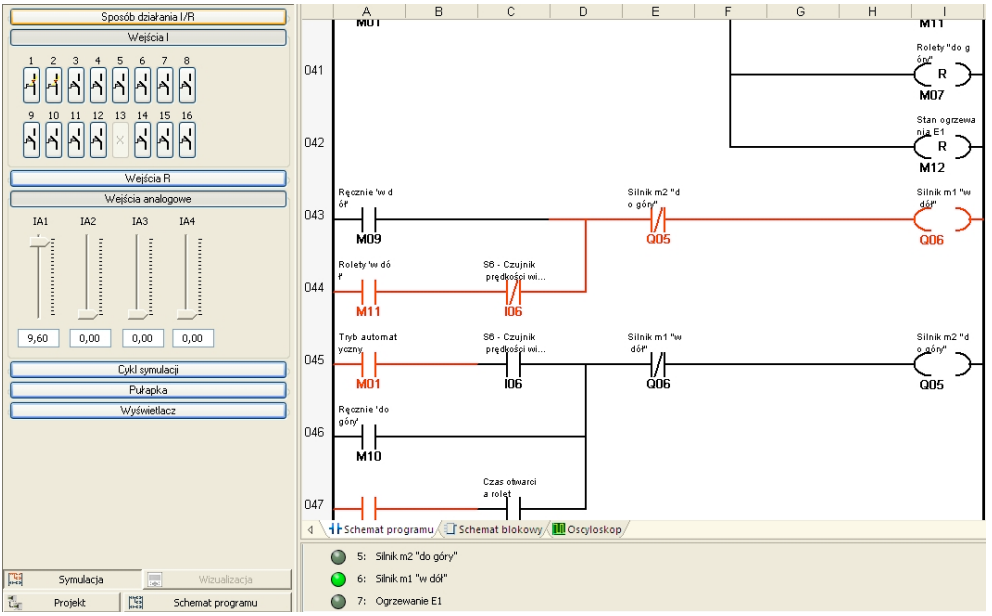
Kolejny przykład przedstawiony na rysunku 120 ilustruje realizację wietrzenia szklarni sterowanego ręcznie. Wciśnięcie przycisku podłączonego do wejścia sterownika oznaczonego jako I03 powoduje otwieranie okien bocznych i dachowych (zapalone zielone lampki). Również w trybie ręcznym funkcjonuje blokada możliwości otwarcia okien w przypadku wystąpienia silnego wiatru I06 (rys. 121).



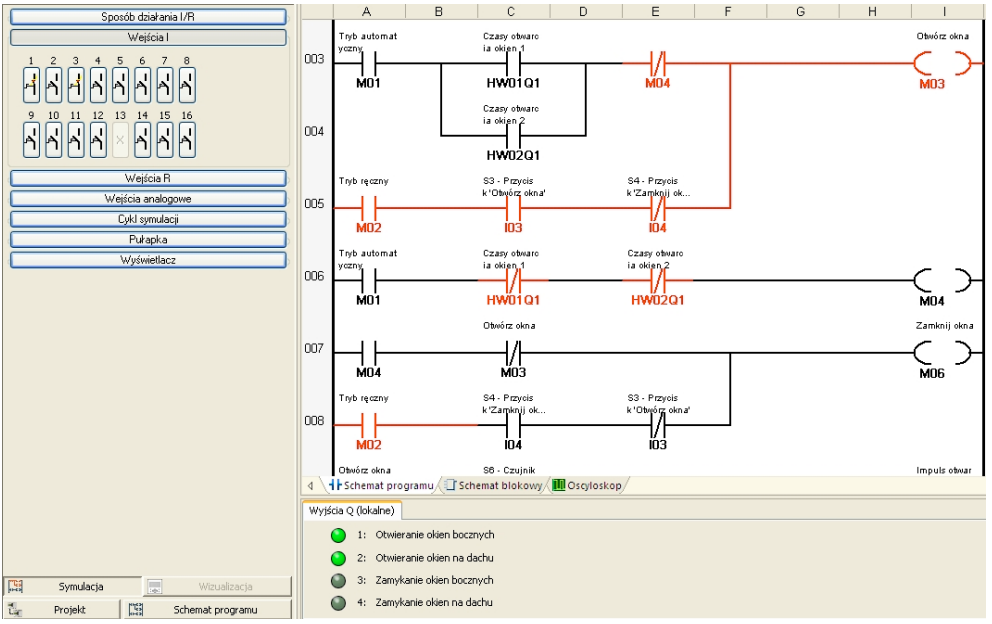
Rysunek 117. Program sterujący nawadnianiem upraw w szklarni



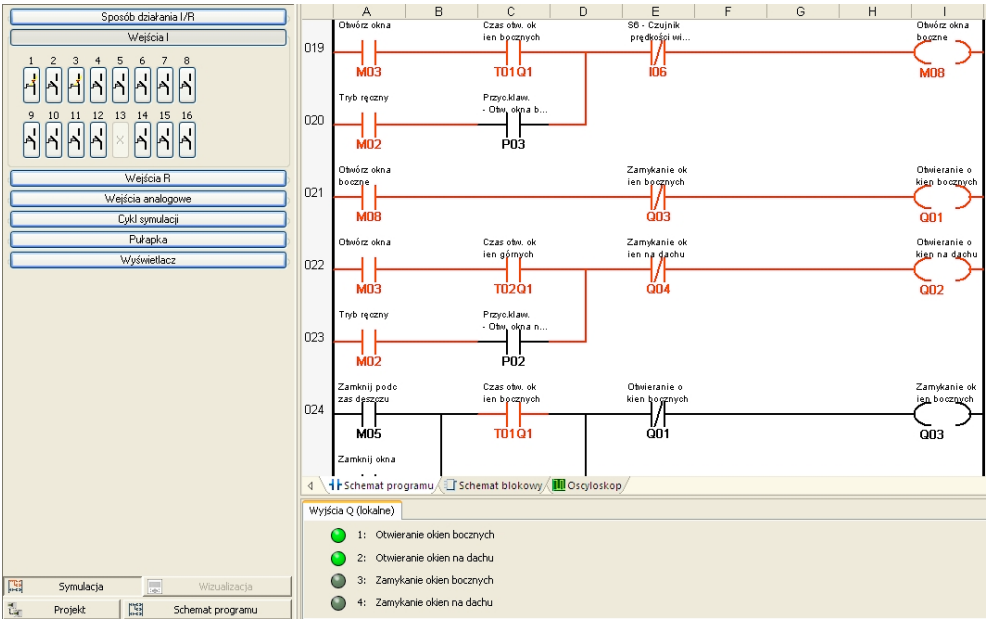
Rysunek 118. Symulacja działania programu sterującego temperaturą w szklarni – część 1



Rysunek 119. Symulacja działania programu sterującego temperaturą w szklarni – część 2



Rysunek 120. Symulacja działania trybu „ręcznego” otwierania okien – część 1



Rysunek 121. Symulacja działania trybu „ręcznego" otwierania okien – część 2

1.2. Inżynieria produkcji zwierzęcej

W produkcji zwierzęcej sterowniki PLC znajdują zastosowanie w sterowaniu:
- procesem przygotowywania pasz;
- procesem zadawania pasz;
- systemem pojenia zwierząt;
- systemem usuwania odchodów;
- procesem doju;
- mikroklimatem budynków inwentarskich (Juszka, 2006; Juszka i in., 2007).

Specyfika konstrukcji budynków inwentarskich w wielu przypadkach zakłada możliwość szybkiej adaptacji do nowych profili produkcyjnych, uwzględniających rentowność produkcji. Wprowadzanie systemów reprogramowalnych zmniejsza problematykę adaptacji pomieszczenia. Zarówno w systemach sterujących mikroklimatem, jak i zadawaniem pasz treściwych można znaleźć elementy wykorzystywane w podobnych profilach produkcyjnych (Juszka i in., 2008a).

1.2.1. Sterowanie mikroklimatem w budynku inwentarskim

W przedstawionym przykładzie programu sterującego mikroklimatem budynku inwentarskiego zaprogramowano również prostą wizualizację dostępną dla sterowników zintegrowanych z wyświetlaczem HMI. Program sterujący został wykonany w języku drabin-

kowym. W programie sterującym dla celów wizualizacyjnych zostało zdefiniowanych 6 masek, które umożliwiają zmianę sterowania zmiennymi oraz informują o wartościach sygnałów wejściowych (Moeller, 2009).

Maska 1 „Start systemu"

Na rysunku 122 została przedstawiona pierwsza maska wizualizacyjna, która pojawia się na ekranie wyświetlacza podczas włączania sterownika. Przyciśnięcie przycisku OK. na panelu czołowym sterownika pozwala na przejście do kolejnego ekranu.

Maska 2 „Uruchamianie/Zatrzymywanie"

Na drugiej masce (rys. 123) umożliwiono uruchomienie lub zatrzymanie systemu sterowania. Po wybraniu opcji „Start" lub „Stop" zostaje ona zaznaczona. Zaznaczenie realizowane jest strzałkami góra/dół (panel sterownika), wciśniecie przycisku po zaznaczeniu opcji powoduje jej aktywację.

Maska 3 „Temperatura"

Przyciskami w lewo/w prawo (panel sterownika) można przemieszczać się pomiędzy maskami. Trzecia maska (rys. 124) przedstawia wartość temperatury mierzonej dwoma niezależnymi czujnikami. Umieszczono w niej obiekty typu tekst statyczny np. *Temperatura T1* oraz obiekty wyświetlające liczby całkowite *99999*. Po przeskalowaniu wartości z przetwornika A/C widoczne są stopnie Celsjusza.

Maska 4 „Temperatura - Zadawanie"

Czwarta maska umożliwia zadawanie pożądanej temperatury dla obiektu inwentarskiego – należy skorzystać z modułu *zadawanie wartości liczbowej* (rys. 125). Rolą systemu automatycznego sterowania będzie utrzymanie zadanej temperatury w obiekcie.

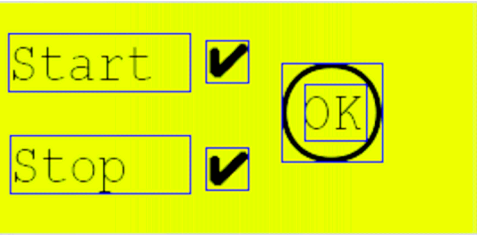
Maska 5 „Wilgotność"

Poprzez tą maskę wyświetlana jest aktualna wilgotność względna powietrza znajdującego się w budynku inwentarskim (rys. 126).

Maska 6 „Urządzenia wykonawcze"

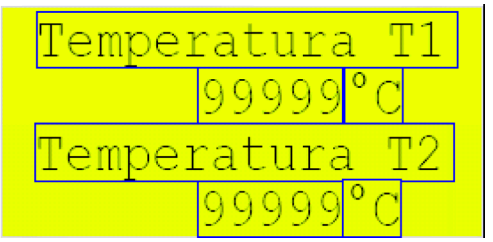
Ostatnia maska pozwala obserwować stan urządzeń wykonawczych (praca/stop) (rys. 127).

Rysunek 122. Maska 1



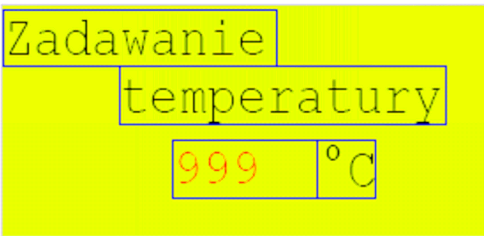
Rysunek 123. Maska 2

„Start sytemu”

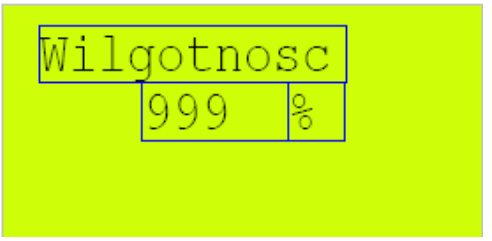


Rysunek 124. Maska 3
„Temperatura”

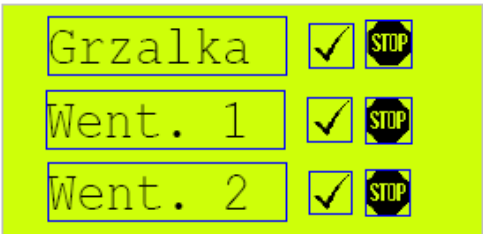
„Uruchamianie/Zatrzymywanie”



Rysunek 125. Maska 4
„Temperatura – Zadawanie



Rysunek 126. Maska 5
„Wilgotność”



Rysunek 127. Maska 6
„Urządzenia wykonawcze”

Na rysunku 128 przedstawiono kompletny schemat programu sterowania mikroklimatem z pomiarem temperatury i wilgotności. Urządzeniami wykonawczymi są wentylatory oraz system ogrzewania. W pierwszym szczeblu umieszczono wyjście Q01 jako styk uruchamiający cały program. Załączenie tego styku jest realizowane, z chwilą kiedy strzałką zaznaczamy opcję start i wciśniemy przycisk OK (rys. 123). Do wyjścia Q01 sterownika można podłączyć kontrolkę, która światłem poinformuje o stanie systemu sterowania. Bloki funkcyjne AR są modułami arytmetycznymi, umożliwiają m.in. skalowanie wartości w programie.

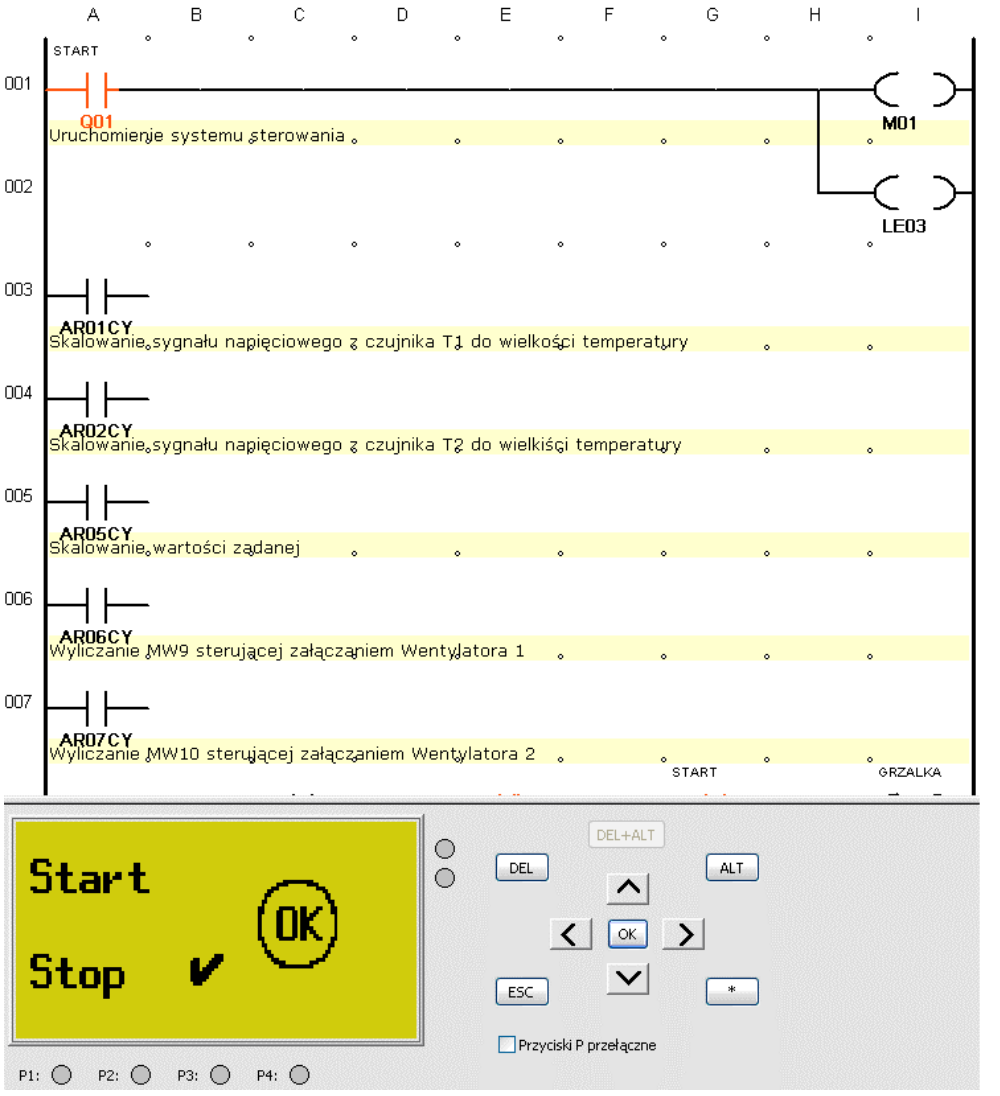
Stan programu po uruchomieniu systemu sterowania przedstawia rysunek 129. Cechą charakterystyczną jest świecąca się dioda LED z prawej strony ekranu wyświetlacza sterowana cewką LE03.

Dla celów testowych wprowadzono temperaturę zadaną, która wynosi 25 °C (rys. 130). Jeżeli temperatura w pomieszczeniu jest niższa niż temperatura zadana – włączona jest nagrzewnica, co uwidoczniają na rysunku 130 linie czerwone (wiersz 008 i 009). Istnieje możliwość ręcznego włączania (wejście I01) nagrzewnicy w celach diagnostycznych. Podobnie można sprawdzić wentylatory (wejścia I02 oraz I03).

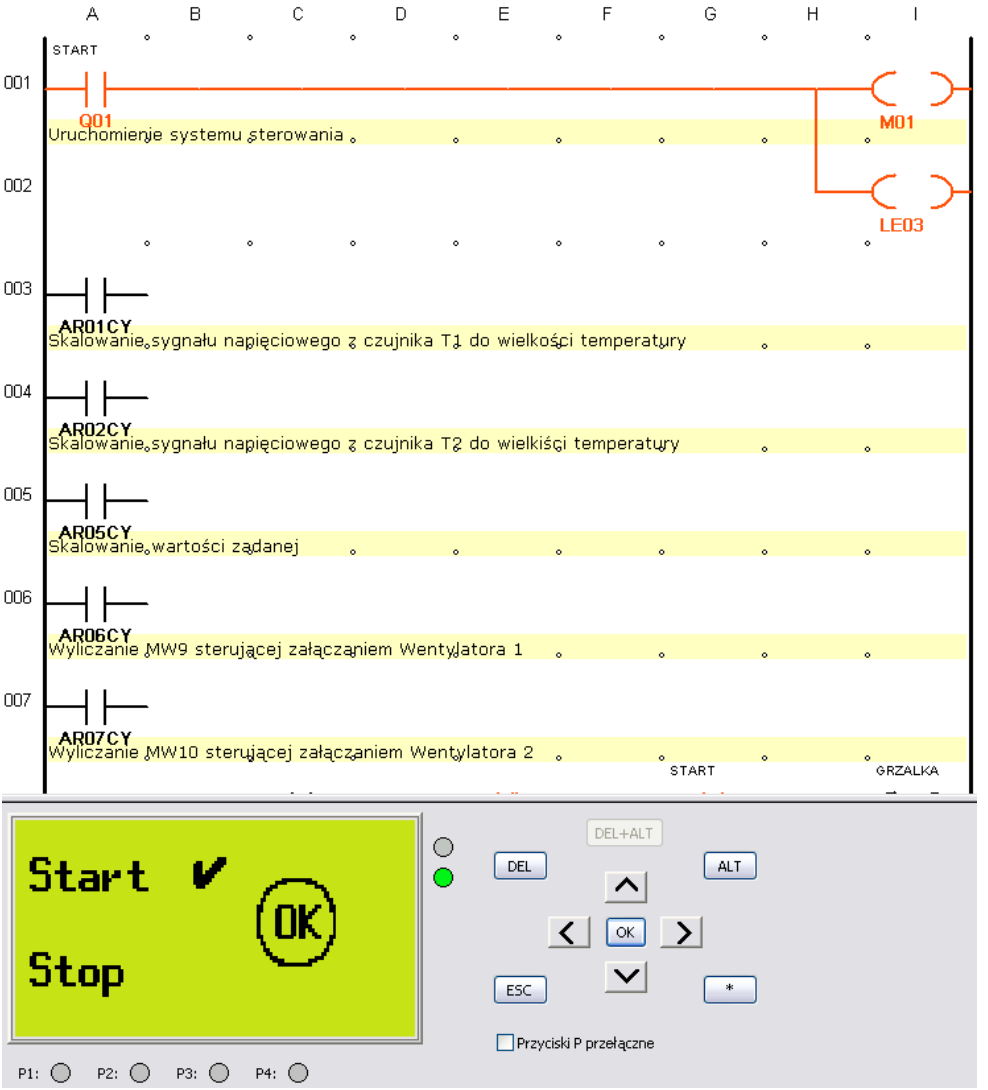
Gdy temperatura w pomieszczeniu wzrośnie do zadanych 25°C (widoczne na wyświetlaczu), system ogrzewania jest wyłączany (rys. 131). Proces sterowania przechodzi w stan czuwania, pobierane i przetwarzane są tylko sygnały wejściowe z czujników.

Wzrost temperatury (przekroczenie o 4°C temperatury zadanej) spowoduje włączenie pierwszego wentylatora wyciągowego (rys. 132).

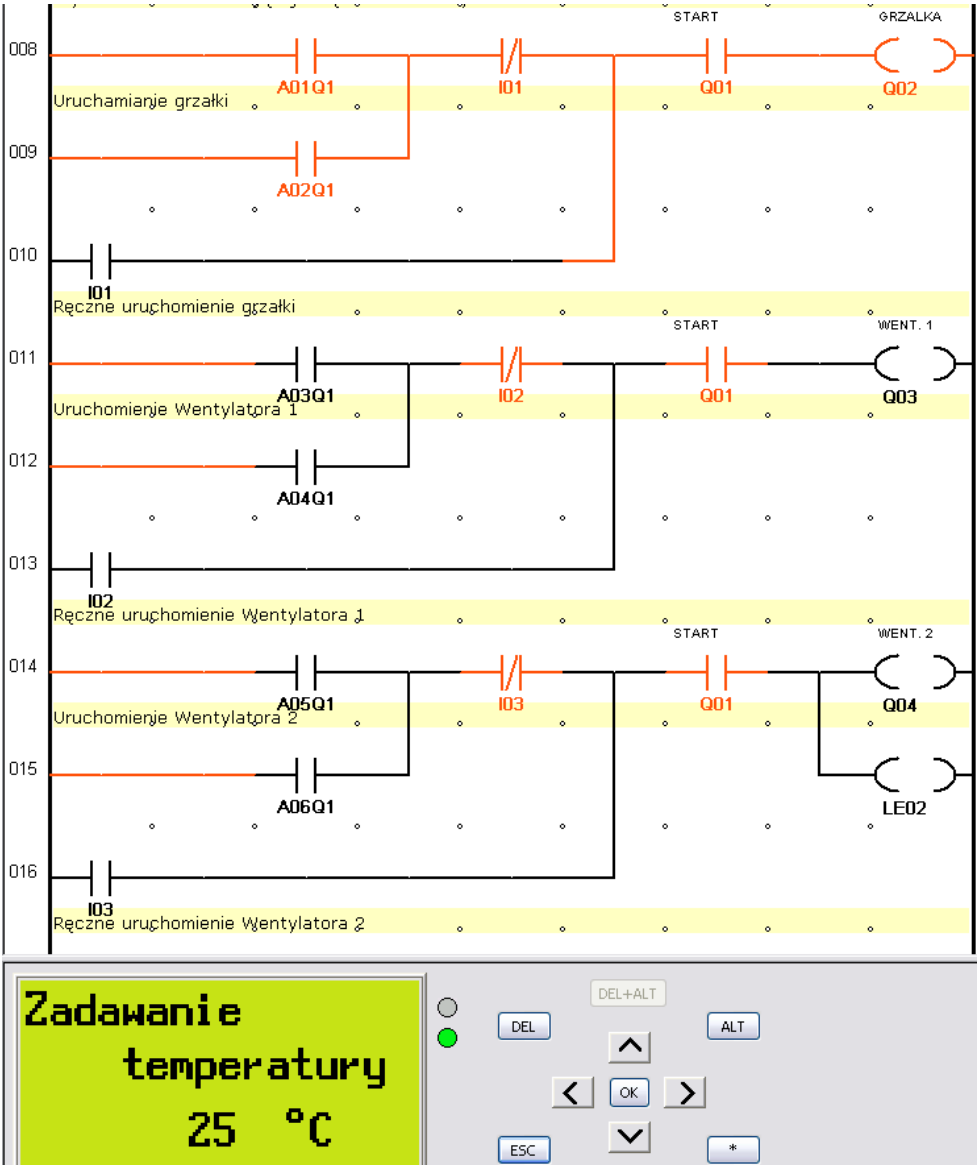
Dalszy wzrost temperatury o kolejne 7°C uruchomi kolejny wentylator oraz spowoduje zaświecenie się czerwonej diody LED (rys. 133). System pracuje z maksymalną wydajnością wentylacyjną.



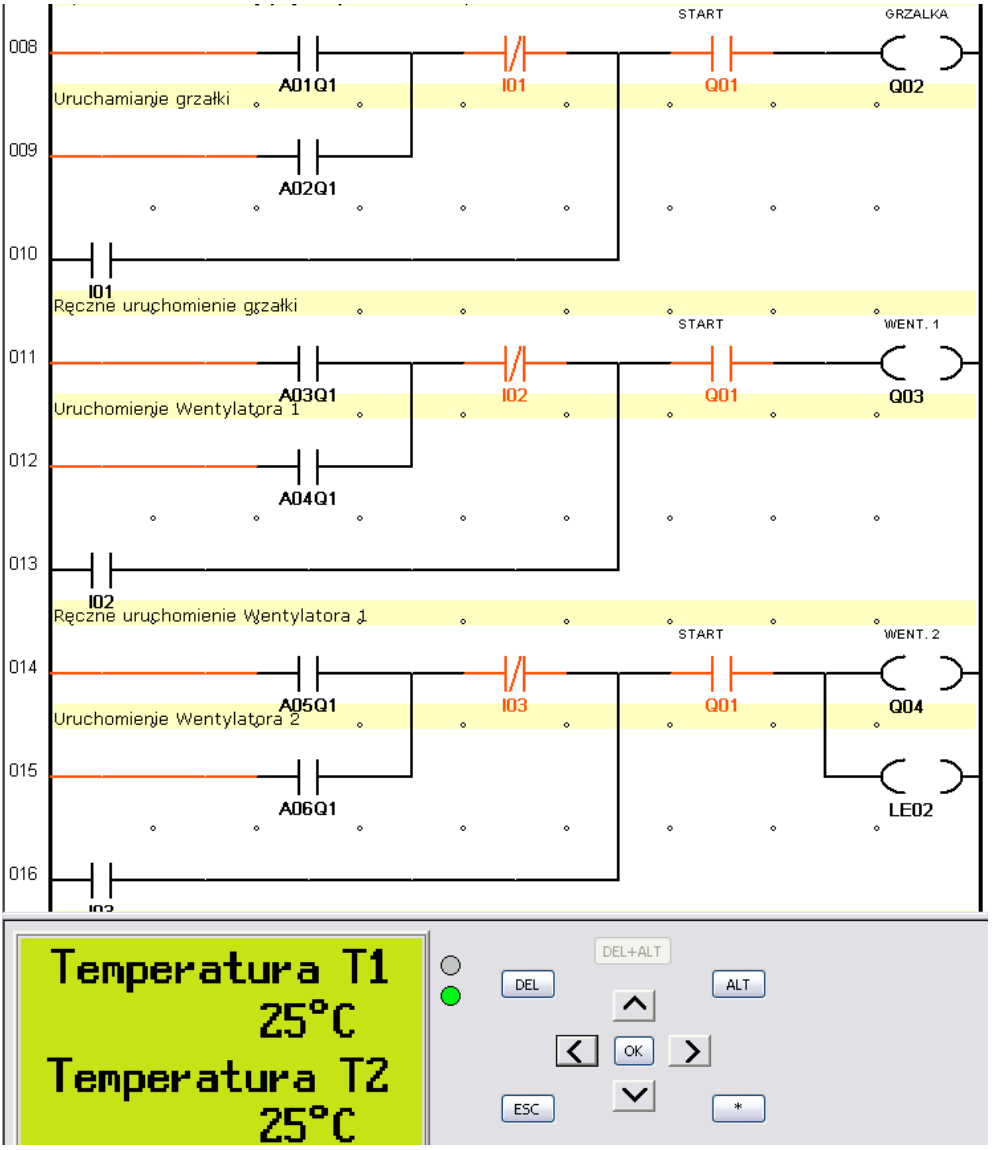
Rysunek 128. Schemat programu sterowania mikroklimatem



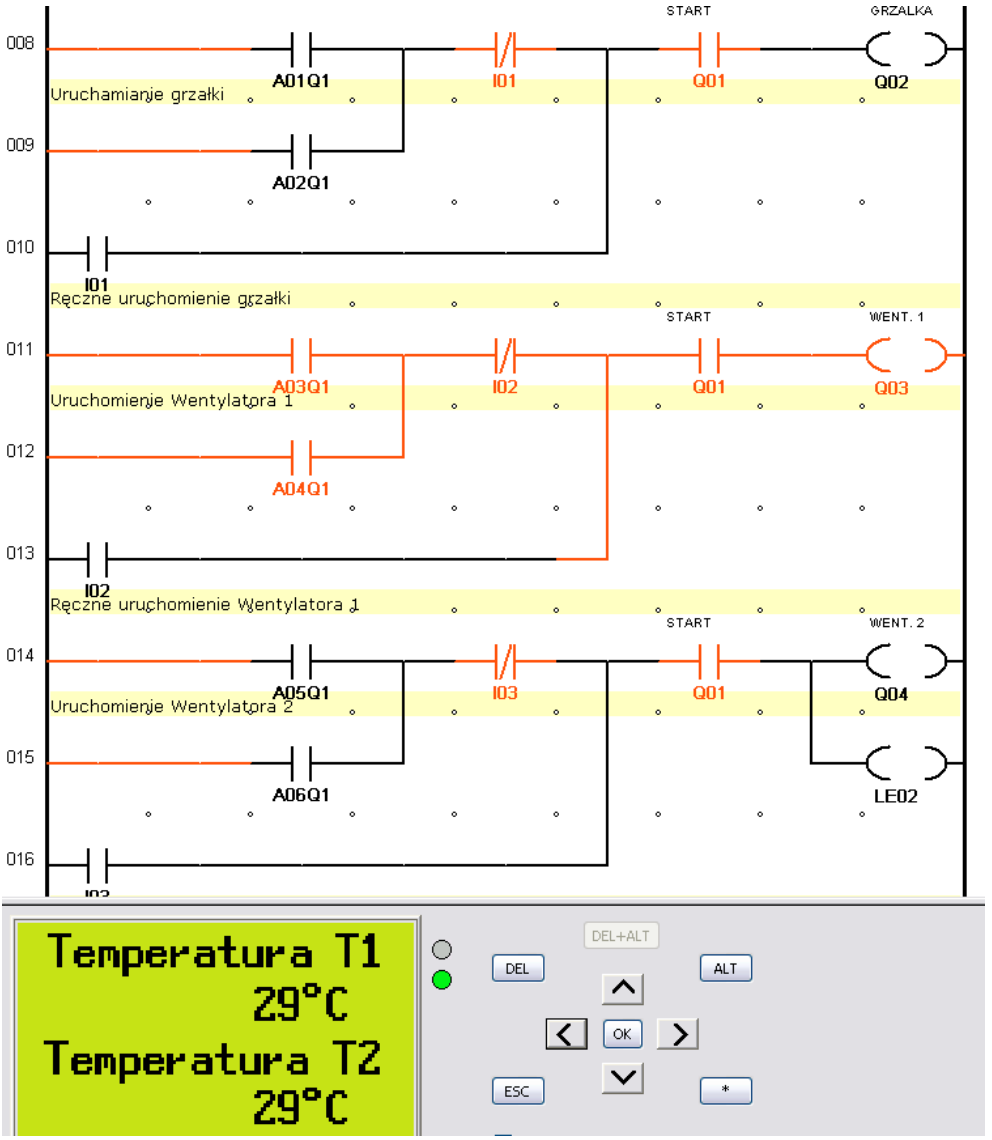
Rysunek 129. Symulacja – uruchomiony program sterujący



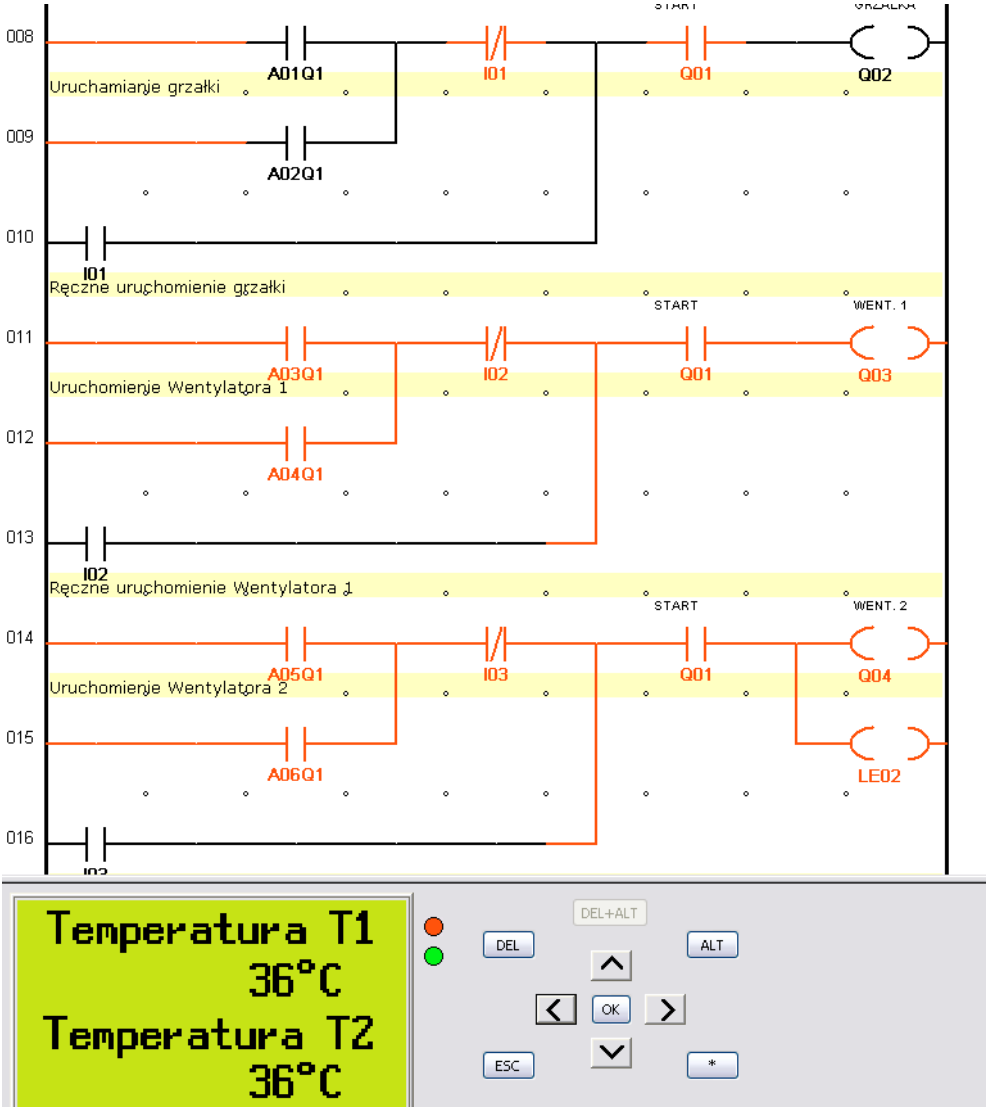
Rysunek 130. Symulacja – wprowadzanie zadanej temperatury



Rysunek 131. Symulacja – tryb oczekiwania



Rysunek 132. Symulacja – praca wentylatora nr 1



Rysunek 133. Symulacja – praca dwóch wentylatorów

Zaprezentowany powyżej przykładowy program sterujący można rozwinąć (w przypadku wyposażenia budynku inwentarskiego) o możliwość sterowania urządzeniem zamgławiającym, odpowiedzialnym za zmiany wilgotności powietrza.

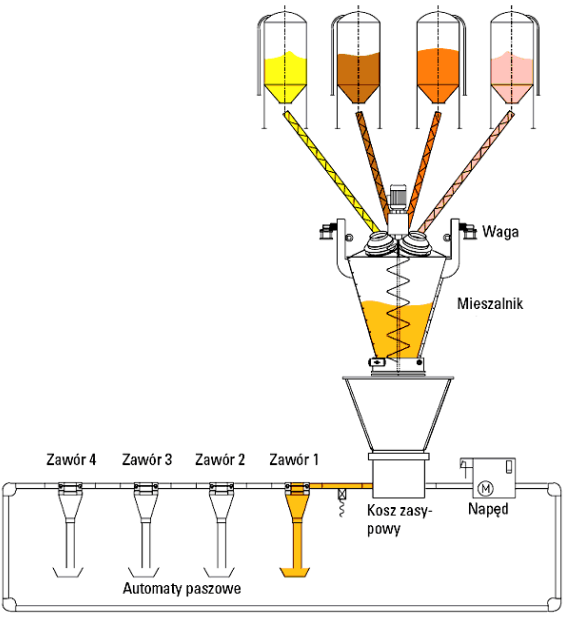
1.2.2. Sterowanie PLC mieszaniem i zadawaniem pasz treściwych

W procesie chowu zwierząt gospodarskich istnieje wiele zadań podlegających automatyzacji, a odpowiedzialnych pośrednio i bezpośrednio za żywienie zwierząt. Pasze najczęściej przechowuje się w silosach. Z tego miejsca rozprowadza się je bezpośrednio do karmideł. Do niedawna silosy można było zobaczyć tylko przy kurnikach. Obecnie takie rozwiązania znajdują zastosowanie w chowie trzody chlewnej oraz stopniowo wdrażane są w chowie bydła szczególnie tam, gdzie stosuje się metodę uwięziową (Juszka i Tomasik, 2009; Juszka i in., 2007).

Decydując się na automatyczny system karmienia redukuje się dotychczasowy czas potrzebny do wykonywania tej czynności. Zapewnia on precyzyjne sterowanie dawkami żywieniowymi według indywidualnych potrzeb zwierząt. Na rynku oferowanych jest kilka rozwiązań automatycznego żywienia, takich jak tzw. „robot Pellon" do zadawania pasz treściwych czy automat paszowy TMR do komponowania i zadawania pasz treściwych.

System mieszania pasz

Na rysunku 134 przedstawiono przykładowy system mieszania i zadawania pasz składający się: z czterech silosów dostarczających komponenty, układu mieszalnika, urządzeń transportowych oraz automatów paszowych. Poszczególne składniki trafiają sekwencyjnie do głównego zbiornika mieszalnika, gdzie podlegają ważeniu w czasie rzeczywistym. Masa zamieniona zostaje na analogowy sygnał ciągły (AI4), wysyłany do sterownika PLC. Masy tych składników są sumowane.



Rysunek 134. Poglądowy schemat systemu mieszania pasz

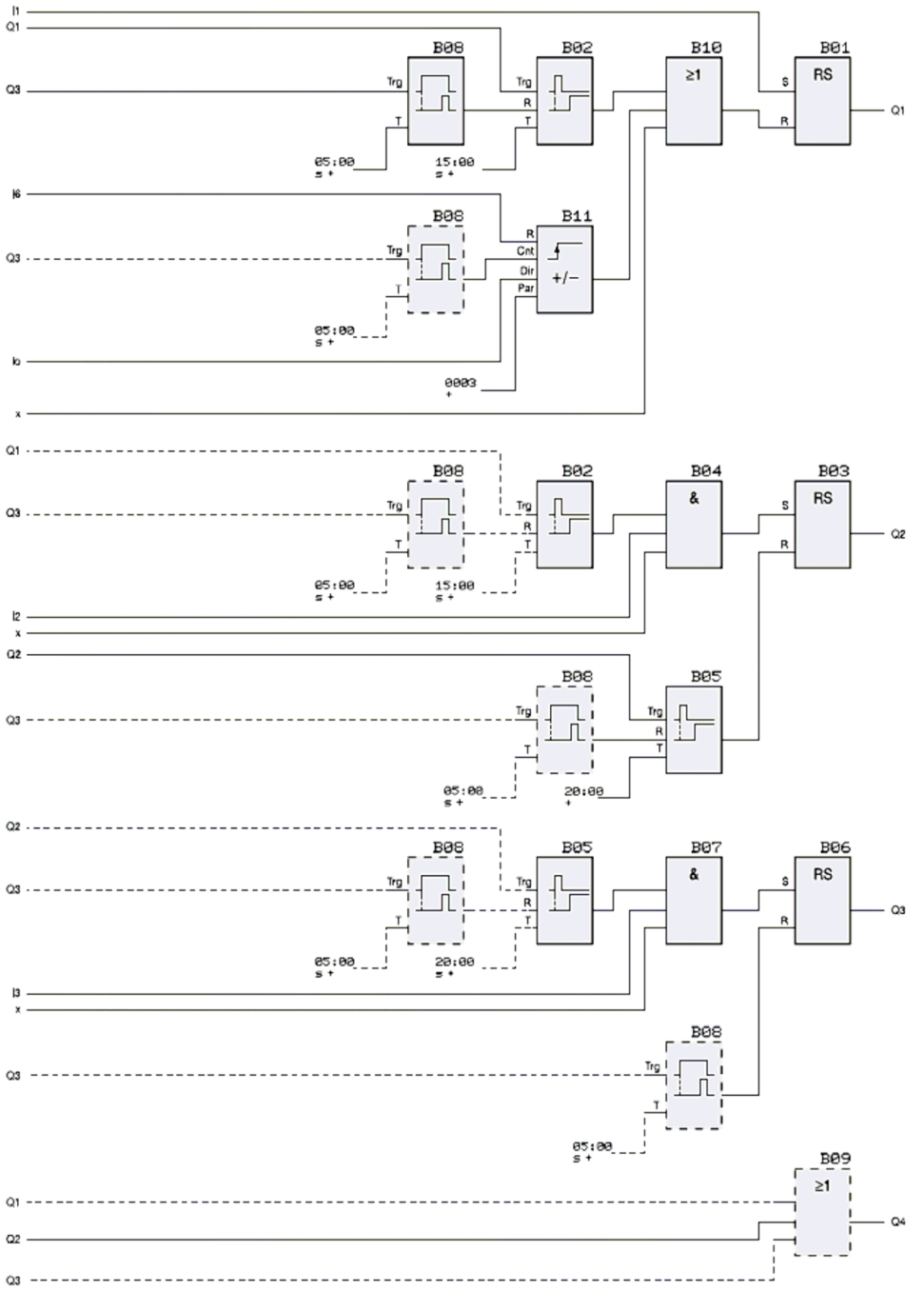
Poszczególne wartości można określić z różnicy pomiędzy całkowitą masą wsadową przed i po dodaniu kolejnego składnika. Składniki poddawane zostają procesowi mieszania, który musi zostać skonfigurowany tak, aby urządzenie zapewniało jak najlepsze ich wymieszanie oraz minimalizowało zużycie energii elektrycznej. Inteligentne sterowanie systemem mieszania wykorzystywanym do przygotowywania pasz dla zwierząt umożliwia załączanie zaprogramowanych czasów (np. trzech czasów, jak w prezentowanym programie) pracy mieszadła z wymuszoną kolejnością oraz daje możliwość zliczania wykonanych pełnych, np. potrójnych, cykli mieszania (ponadto pozwala również na określenie liczby cykli mieszania). Prezentowany program wykonany w języku schematów bloków funkcjonalnych FBD dotyczy samego mieszania pasz.

Wykaz bloków funkcyjnych oraz wejść/wyjść (rys. 135):

B02 – czas pierwszego mieszania;
B05 – czas drugiego mieszania;
B08 – czas trzeciego mieszania;
B11 – zadawanie ilości pełnych cykli;
I1 – start pierwszego czasu mieszania;
I2 – start drugiego czasu mieszania;
I3 – start trzeciego czasu mieszania;
I6 – kasowanie zadanej ilości pełnych cykli mieszania;
Q1 – sygnalizacja trwania czasu pierwszego;
Q2 – sygnalizacja trwania czasu drugiego;
Q3 – sygnalizacja trwania czasu trzeciego;
Q4 – sterowanie stycznikiem załączającym mieszadło.

Założono nastawy czasów mieszania niezależnie od siebie. Po podaniu sygnału na wejście sterownika I1 przerzutnik RS w bloku B01 wysterowuje wyjście Q1. W bloku B02 (podtrzymanie opóźnione załączenie) startuje licznik czasu, po wyzerowaniu którego za pośrednictwem B10 (bramki OR) wyzwolone zostaje wejście zerujące B01, w wyniku czego wyjście pierwsze zostaje wyłączone. W bloku B04 (bramka AND) na wejściu pojawia się stan wysoki z bloku B02. Po podaniu drugiego sygnału z wejścia I2 zostaje wysterowany przerzutnik RS – B03 załączający wyjście Q2. W bloku B05 (podtrzymane opóźnione załączenie) startuje czas, po upłynięciu nastawionego czasu B03 zostaje wyzerowany, a wyjście Q2 wyłączone. Na wejściu bloku B07 (bramka AND) pojawia się stan wysoki z bloku B05.

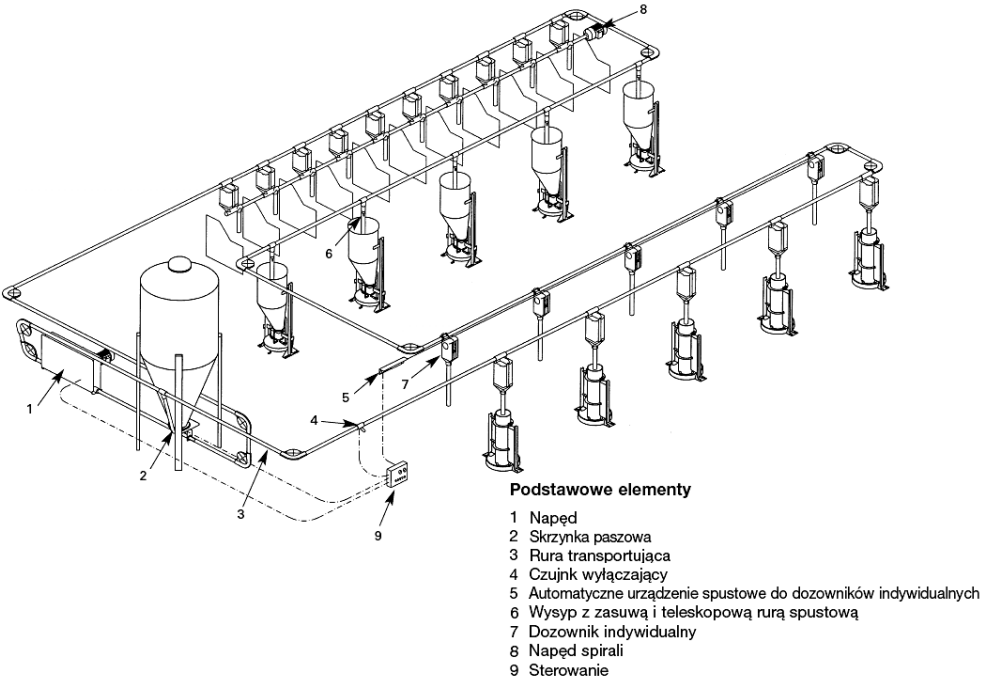
Po podaniu drugiego sygnału wejściowego z wejścia I3 zostaje wysterowany (przerzutnik RS) w bloku B06 i wysterowane wyjście Q3. W bloku B08 (opóźnione załączenie) startuje licznik czasu. Po odmierzeniu zadanego czasu wyzerowany zostaje blok B06 i wyjście Q3 wyłącza się. Jednocześnie wyzerowane zostają przekaźniki czasowe bloków B02 i B05. W przypadku gdy wartość zliczona osiągnie zadaną wartość, wyjście licznika wysteruje B10 i wyzeruje B01, uniemożliwiając startowanie nowych cykli, aż do momentu skasowania licznika B11 za pośrednictwem I6. Załączanie wyjścia Q4 odbywa się za pośrednictwem bramki AND – B09, którą sterują wyjścia Q1, Q2, Q3.



Rysunek 135. Schemat programu mieszania pasz

System automatycznego sterowania dystrybucją paszy w budynku inwentarskim

Najczęściej nowoczesne systemy transportujące pasze w budynku inwentarskim stanowią układy połączonych rur z umieszczoną w środku linką transportującą (rys. 136). Analizując systemy sterowania w takich instalacjach, można wyróżnić: zespół napędowy, układ sterujący i czujniki. Informacja z czujników oraz wprowadzone wielkości nastaw będą decydowały o procesie zadawania paszy. Stąd konieczność instalowania czujników w korytach paszowych, wykrywających obecność paszy (zaleganie jej) i pozyskiwanie informacji zwrotnej jako alarmu, z jednoczesnym zablokowaniem uzupełniania paszy w tym korycie (Juszka i Tomasik 2009).

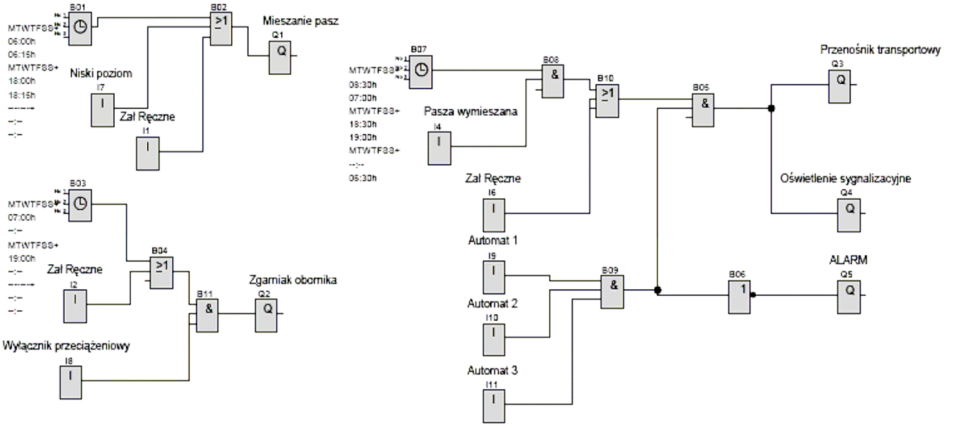


Rysunek 136. System transportowo-rozdzielający pasze

Przykład programu sterującego z jednego sterownika mikroprocesorowego procesem zadawania paszy, karmienia oraz zgarniania obornika zgarniaczem linowym zamieszczono na rysunku 137. Zaletą takiego rozwiązania jest możliwość wprowadzenia bezpośredniej programowej zależności pomiędzy procesami, natomiast niebezpieczną wadą (w przypadku awarii sterownika) jest to, że nie pracuje żaden system. Program wykonany został podobnie jak dla poprzedniego przykładu w języku FBD.

Algorytm sterowania przedstawia się następująco. Każdy z systemów można uruchomić niezależnie ręcznie po spełnieniu podstawowych warunków. Pierwszą czynnością pracy automatycznej jest uruchamianie procesu mieszania paszy wyjściem Q1. Jeżeli system mieszania będzie wydzielony, tak jak to ma miejsce w tym przykładzie, wyjście to powinno

go uruchomić. Aby pojawił się na nim sygnał logicznej jedynki, musi zostać spełniony jeden z warunków: na przekaźniku czasowym B01 zadano czas aktywacji tego systemu (dni tygodnia oraz godziny), możliwe jest ręczne załączenie I1 lub w przypadku wystąpienia niskiego poziomu paszy (I7). Pasza trafi do automatów paszowych instalacją transportową (Q3) o określonej porze dnia (B07), pod warunkiem że został zakończony proces mieszania (I04). Warunkiem niezbędnym do uruchomienia taśmociągu są również puste koryta paszowe (I9, I10, I11). Jeśli w którymś zalega pasza, zaświeci się lampka kontrolna (Q5). Prawidłowy przebieg transportu sygnalizuje lampka Q4. Podobnie korzystając z bloków funkcyjnych, zbudowano algorytm programu sterowania zgarniakiem obornika (Q2). Uruchamiany jest codziennie o określonej godzinie (możliwe jest uruchomienie ręczne), posiada zabezpieczenie przeciążeniowe.



Rysunek 137. Program sterujący mieszaniem i zadawaniem pasz oraz usuwaniem obornika

1.2.3. Zastosowanie sterowników PLC do sterowania ciśnieniem bezwzględnym (ssącym) w maszynowym doju krów

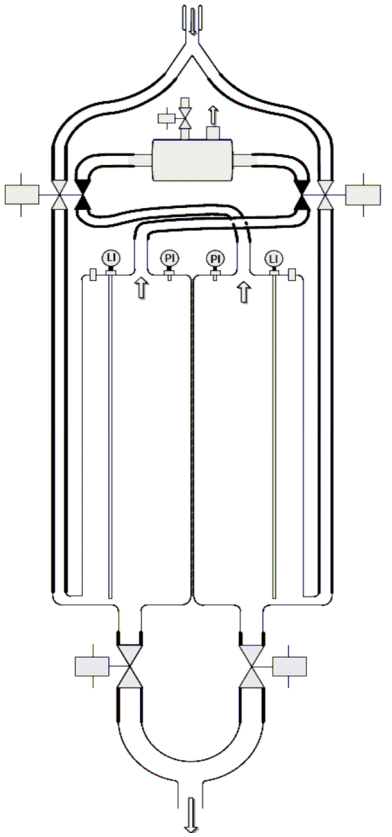
Współczesne konstrukcje kolektorów aparatów udojowych dla krów dostarczają do wymienia ciśnienie ssące o takich samych wartościach. Stąd powstał pomysł opracowania innowacyjnej konstrukcji aparatu udojowego, składającego się z niezależnych kolumn dla każdej ćwiartki wymienia. Do zadań kolumny należy: odbieranie mleka z kubka udojowego, pomiar udojonego mleka z ćwiartki wymienia krowy, dostarczanie ciśnienia ssącego do komory podstrzykowej kubka udojowego, rozdzielanie ciśnienia ssącego mleko ze strzyka od ciśnienia transportowego. Zbiorniki są wyposażone w czujniki poziomu oraz ciśnienia bezwzględnego. Nad zbiornikami znajduje się układ przygotowujący ciśnienie bezwzględne dla AAU. Przepływem mleka i powietrza sterują zawory nad i pod zbiornikami (rys. 138) (Juszka i in., 2011a).

Z uwagi na złożoną konstrukcję tego systemu, zawierającą dużą liczbę czujników i elementów wykonawczych, układ sterujący pracą kolumny autonomicznego aparatu udojowego (AAU) zbudowano na bazie sterownika PLC (Tomasik i in., 2012c).

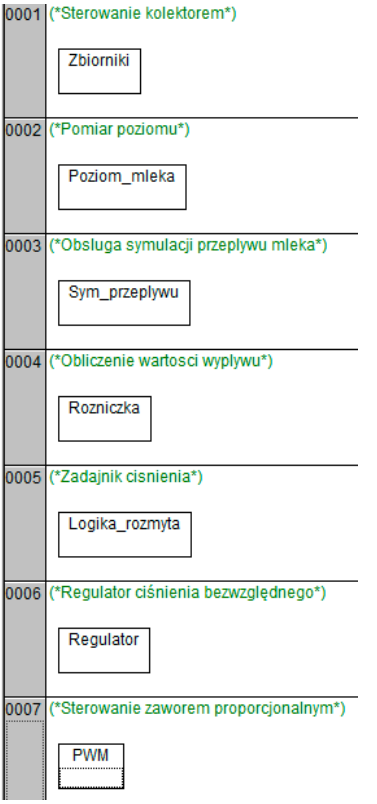
Program sterujący napisany dla sterownika PLC został podzielony na części odpowiadające za konkretne funkcje:

– sterowanie kolektorem;
– pomiar poziomu;
– symulacja przepływu;
– obliczanie natężenia wypływu;
– zadajnik ciśnienia ssącego;
– regulator ww. ciśnienia;
– sterowanie zaworem regulacyjnym.

Części programu odpowiedzialne za poszczególne funkcje zostały umieszczone w osobnych modułach (blokach funkcjonalnych), które są kolejno wykonywane przez sterownik (rys. 139).



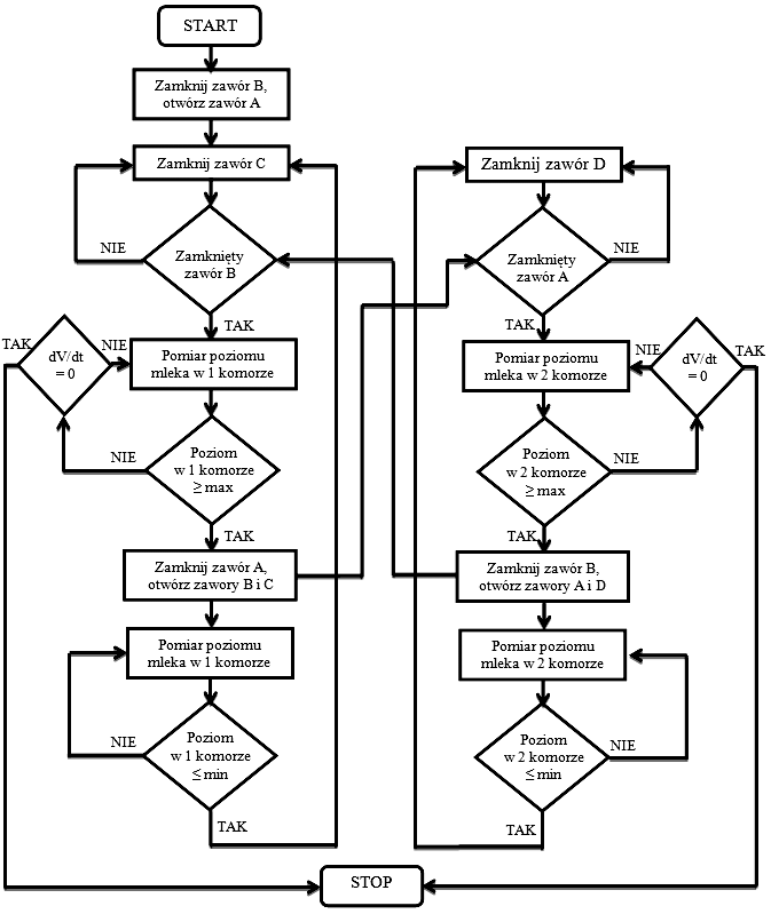
Rysunek 138. Schemat kolumny kolektora AAU



Rysunek 139. Schemat wywołań kolejnych podprogramów

Sterowanie kolumną kolektora

Sterowanie przepływem mleka i powietrza przez kolumnę kolektora odbywa się według algorytmu przedstawionego na rysunku 140. Jeżeli zawór napełniający (A) komory pierwszej jest zamknięty i zawór napełniający (B) komory drugiej jest zamknięty, to następuje otwarcie zaworu napełniającego (A) pierwszą komorę z jednoczesnym zamknięciem wszystkich pozostałych zaworów (C, D). Jeżeli poziom w pierwszej komorze osiągnie wartość maksymalną to zamykany jest zawór napełniający (A) tę komorę, następnie otwierany jest zawór (B) napełniający drugą komorę. W następnym kroku programu otwierany jest zawór spustowy (C) pierwszej komory. Jeśli poziom w pierwszej komorze osiągnie wartość minimalną, to zamykany jest zawór spustowy (C) w tej komorze. Jeżeli poziom w drugiej komorze osiągnie wartość maksymalną, otwierany jest zawór spustowy (D) drugiej komory. Program przystępuje do napełniania pierwszej komory i dalsza jego praca jest cykliczna.



Rysunek 140. Algorytm sterowania kolumną kolektora

Program wykonujący powyższy algorytm zamieszczono na rysunku 141. Został on napisany językiem tekstu strukturalnego ST. Głównym elementem programu są instrukcje warunkowe sterujące elementami wykonawczymi po spełnieniu określonego warunku. Pierwsza instrukcja warunkowa, obejmująca wszystkie pozostałe, to warunek, że program może być wykonywany, jeżeli zostanie wybrana opcja *auto*. Pozostałe pięć instrukcji należące do niej steruje zaworami. Na uwagę zasługuje jeszcze instrukcja *licznik_napelnien,* która zlicza ilość cykli napełnienia poszczególnych komór, na jej podstawie można wyliczyć całkowitą objętość udojonego mleka z danej ćwiartki wymienia krowy.

```
0052 (* Sterowanie kolektorem w trybie automatycznym*)
0053 IF auto = TRUE THEN
0054
0055 IF zawor_napeln1 = FALSE AND zawor_napeln2 = FALSE THEN
0056     zawor_napeln1:= TRUE;
0057     zawor_spust1:= TRUE;
0058     zawor_napeln2:= FALSE;
0059     zawor_spust2:= TRUE;
0060 END_IF
0061
0062 IF poziom1 <= min1 THEN
0063     zawor_spust1:=TRUE;
0064 END_IF
0065
0066 IF poziom2 <= min2 THEN
0067     zawor_spust2:=TRUE;
0068 END_IF
0069
0070 IF poziom1 >= max1 AND zawor_napeln1 = TRUE THEN
0071     zawor_napeln1:= FALSE;
0072     zawor_spust1:= FALSE;
0073     zawor_napeln2:= TRUE;
0074     zawor_spust2:= TRUE;
0075     licznik_napelnien1:= licznik_napelnien1 + 1;
0076 END_IF
0077
0078 IF poziom2 >= max2 AND zawor_napeln2 = TRUE THEN
0079     zawor_napeln1:= TRUE;
0080     zawor_spust1:= TRUE;
0081     zawor_napeln2:= FALSE;
0082     zawor_spust2:= FALSE;
0083     licznik_napelnien2:= licznik_napelnien2 + 1;
0084 END_IF
0085
0086 END_IF
```

Rysunek 141. Program sterujący kolumną kolektora

Pomiar poziomu mleka w komorze

Precyzyjny pomiar poziomu mleka w komorze kolumny kolektora jest niezbędny do wyznaczenia chwilowego objętościowego natężenia przepływu mleka przez kolektor odpowiadającego wypływowi ze strzyka wymienia krowy (Tomasik i in., 2012a). Pomiar odbywa się w kilku krokach. Najpierw wartość 12-bitowa odczytana z wejścia analogowego (podłączony jest czujnik poziomu) przeliczana jest na napięcie. Następnie napięcie przeliczane jest na poziom w centymetrach. Dla czujnika w pierwszej komorze realizowane jest przeskalowanie liniowe, obliczane za pomocą funkcji transformacji liniowej (linie od 5 do 11). Natomiast dla koncepcyjnego czujnika w drugiej komorze obliczana jest wartość funkcji wielomianowej 4-tego stopnia (linie od 15 do 19). Taka forma obliczeń była wynikiem zastosowanych czujników pomiarowych własnej konstrukcji. W linii 12 i 22 realizowane jest przeskalowywanie zmiennych na potrzeby wizualizacji. W linii od 26 do 31 realizowany jest wybór komory, z której wartość poziomu posłuży do obliczania natężenia wypływu mleka. Wybierana jest ta komora, która ma otwarty zawór napełniania. Fragment programu odpowiadający za pomiar przedstawiono na rysunku 142.

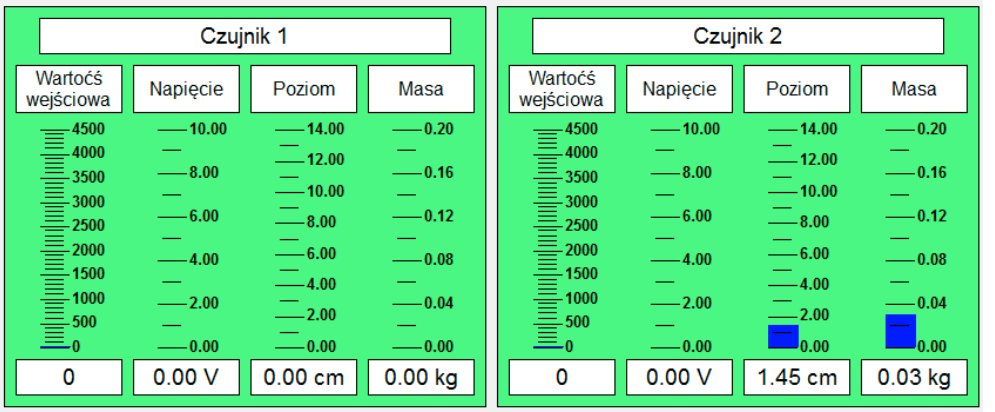
```
0001 wolty1:=czujnik1/409.6;                    (* Odczyt napięcia z czujnika poziomu 1 komory *)
0002 wolty:=czujnik2/409.6;                     (* Odczyt napięcia z czujnika poziomu 2 komory *)
0003
0004                    (* Przeliczenie liniowe napięcia na poziom w centymetrach, dla 1 komory *)
0005 transformacja(
0006     IN:= wolty1,
0007     IN_MIN:= 0.85875,
0008     IN_MAX:= 5.31375,
0009     OUT_MIN:= 0,
0010     OUT_MAX:= 12,
0011     OUT=> poziom1_cm);
0012 poziom1:=poziom1_cm/50;
0013
0014                                            (* Wielomian 4 stopnia dla czujnika rownoleglego*)
0015 poziom_cm2:=(EXPT(wolty,4))*(0.0098);
0016 poziom_cm3:=(EXPT(wolty,3))*(-0.1761);
0017 poziom_cm4:=(EXPT(wolty,2))*(1.1408);
0018 poziom_cm5:=wolty*(-2.2898);
0019 poziom_cm:=poziom_cm2+poziom_cm3+poziom_cm4+poziom_cm5+1.4465;
0020
0021 IF poziom_cm<0 THEN poziom_cm:=0; END_IF
0022 poziom2:= poziom_cm/50;
0023
0024                                (* Wybór odczytu poziomu z aktualnie napełnianego zbiornika
0025                                                dla potrzeb obliczenia natężenia wypływu *)
0026 IF zawor_napeln1 = TRUE THEN
0027     poziom:=poziom1;
0028 END_IF
0029 IF zawor_napeln2 = TRUE THEN
0030     poziom:=poziom2;
0031 END_IF
```

Rysunek 142. Schemat programu odpowiedzialnego za pomiary poziomu mleka w komorach

Do zobrazowania wartości odczytanych z czujników poziomu zaprogramowano wizualizację. Oprogramowanie CoDeSys umożliwia tworzenie wizualizacji na dedykowane dla sterownika panele graficzne lub na komputer PC. Dzięki temu w prosty sposób można:

– sporządzić charakterystyki dynamiczne i statyczne czujników;
– dobrać wartości współczynników do przeliczania wartości odczytanych z wejść analogowych na poziom i masę cieczy w zbiorniku;
– ocenić poprawność pomiarów.

Wizualizację pracy czujników poziomu przedstawiono na rysunku 143.

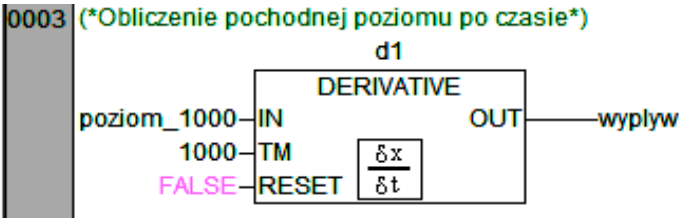


Rysunek 143. Wizualizacja wartości odczytanych z czujników poziomu

Obliczanie natężenia wypływu mleka przez komorę kolektora

Oprogramowanie CoDeSys zawiera bibliotekę Util.lib, która jest zbiorem dodatkowych bloków funkcyjnych. Biblioteka ta została podzielona na grupy, jedna z nich to funkcje matematyczne, do których należy blok funkcyjny DERIVATIVE, czyli pochodna.

Chwilowe natężenie strumienia masowego mleka obliczono przez zróżniczkowanie poziomu mleka znajdującego się w komorze kolektora po czasie (rys. 144).



Rysunek 144. Blok funkcyjny obliczający przepływ mleka

Funkcja DERIVATIVE ma trzy parametry wejściowe i jedno wyjście. Na wejścia dostarczana jest aktualna wartość poziomu mleka znajdującego się w zbiorniku, czas w milisekundach, po którym następują kolejne pomiary wartości wejściowej, a także informacja RESET, która zeruje wartość wyjściową. Na wyjściu funkcji otrzymujemy obliczoną wartość strumienia objętościowego wypływającego mleka.

Symulacja przepływu mleka przez kolumnę kolektora

Dla celów testowych napisano program symulujący przyrost poziomu cieczy w zbiornikach kolektora. Dzięki symulacji możliwe jest sprawdzenie działania wszystkich części programu sterującego bez konieczności uruchamiania całego układu sterowania. Na rysunku 145 przedstawiono fragment programu odpowiedzialny za symulację odczytu wartości z wejść analogowych sterownika. W liniach od 3 do 9 realizowany jest przyrost poziomu cieczy w zbiorniku, w którym otwarty jest zawór napełniający. Przyrost wartości jest inny dla każdej komory, aby symulować różne wartości natężenia przepływu podczas doju. Natomiast w liniach od 11 do 17 realizowane jest opróżnianie zbiorników przy większym natężeniu wypływu cieczy niż przy ich napełnianiu.

```
0001 IF symulacja = TRUE THEN
0002
0003     IF zawor_napeln1 = TRUE AND poziom1 < 0.2 THEN
0004         czujnik1:=czujnik1-10;
0005     END_IF
0006
0007     IF zawor_napeln2 = TRUE AND poziom2 < 0.2 THEN
0008         czujnik2:=czujnik2-15;
0009     END_IF
0010
0011     IF zawor_spust1 = FALSE AND poziom1 > 0.001  THEN
0012         czujnik1:=czujnik1+20;
0013     END_IF
0014
0015     IF zawor_spust2 = FALSE AND poziom2 > 0.001 THEN
0016         czujnik2:=czujnik2+20;
0017     END_IF
0018
0019 END_IF
```

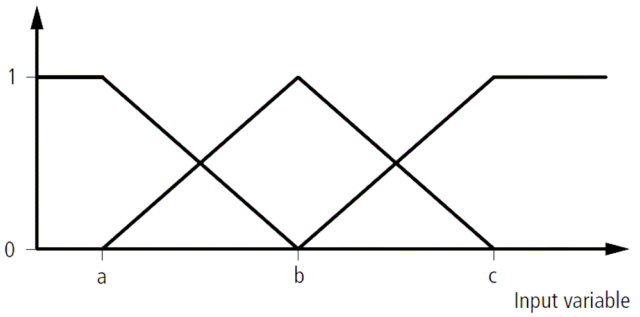
Rysunek 145. Schemat programu symulującego poziom cieczy w zbiornikach

Zadajnik ssącego ciśnienia bezwzględnego

Zadajnik jest elementem mającym za zadanie wyznaczyć wartość ciśnienia bezwzględnego w kubku udojowym dla aktualnej wartości natężenia strumienia masowego mleka wypływającego ze strzyka wymienia krowy. Programowy zadajnik zrealizowano, bazując na logice rozmytej. Wykorzystano do tego celu blok funkcyjny FuzzyLogic z jednym wejściem, jednym wyjściem oraz trzema regułami sterowania. „Rozmycie” wielko-

ści wejściowej (Input variable) w bloku funkcyjnym zadajnika FLC przeprowadzono wg następującej procedury (rys. 146):

– zmienną wejściową jest natężenie strumienia masowego mleka wypływającego ze strzyka krowy;
– zmienną wyjściową jest ciśnienie bezwzględne w komorze podstrzykowej;
– wartość „1a" natężenia wypływu mleka wynosi 0,8 ($g \cdot s^{-1}$);
– wartość „1b" natężenia wypływu mleka wynosi 2 ($g \cdot s^{-1}$);
– wartość „1c" natężenia wypływu mleka wynosi 25 ($g \cdot s^{-1}$) (Tomasik i in., 2011).



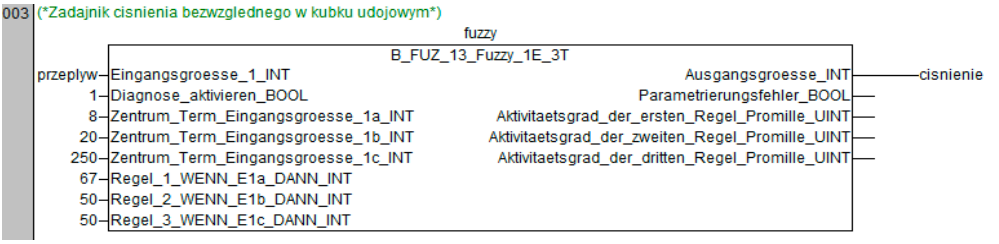
Rysunek 146. Rozmywanie wielkości wejściowej na 3 przedziały w sterowniku PLC

Jeżeli zachodzi konieczność dokładniejszego dopasowania zależności wejściowych i wyjściowych zalecane jest rozmycie wartości w większej ilości przedziałów.

Dobrano następujące reguły sterowania:

– jeśli natężenie wypływu mleka przyjmuje wartość „1a", to ciśnienie bezwzględne jest równe 50 kPa;
– jeśli natężenie wypływu mleka przyjmuje wartość „1b", to ciśnienie bezwzględne jest równe 60 kPa;
– jeśli natężenie wypływu mleka przyjmuje wartość „1c", to ciśnienie bezwzględne jest równe 67 kPa.

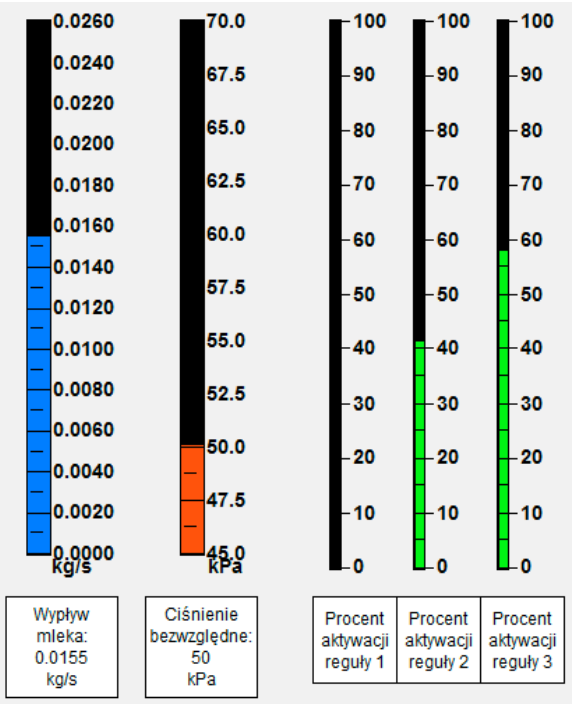
Blok funkcyjny zadajnika FLC z dobranymi parametrami przedstawiono na rysunku 147.



Rysunek 147. Blok funkcyjny zadajnika FLC

Do weryfikacji pracy układu sterowania z zadajnikiem FLC zaprogramowano wizualizację przedstawiającą jego działanie (rys. 148). W oknie widoczne są: chwilowe masowe natężenie wypływu mleka i odpowiadające mu ciśnienie bezwzględne (wyliczone przez zadajnik) oraz procent aktywacji poszczególnych reguł sterowania.

Za pomocą powyższej wizualizacji można odczytać wartość wejściową zadajnika FLC, czyli wartość strumienia masowego wypływającego mleka. Kolejnym parametrem przedstawionym na wizualizacji jest wartość wyjściowa działania funkcji, czyli wartość ciśnienia bezwzględnego, jakie powinno panować w komorze podstrzykowej w danym momencie. Na wizualizacji przedstawiono także parametry diagnostyczne, generowane przez funkcję FuzzyLogic.



Rysunek 148. Wizualizacja parametrów zadajnika

Regulator ciśnienia bezwzględnego

Aby skutecznie sterować ciśnieniem bezwzględnym, zaprogramowano regulator PID. Zadaniem regulatora jest sterowanie zaworem proporcjonalnym tak, aby w zbiorniku panowało ciśnienie wyznaczone przez zadajnik FuzzyLogic. Regulator PID pracuje w układzie sprzężenia zwrotnego. Konieczny dla działania regulatora PID błąd regulacji *e* wyznaczany jest jako różnica pomiędzy wartością z FLC, a wartością zmierzoną przez czujnik ciśnienia absolutnego umieszczony w zbiorniku. Wartością zadaną regulatora jest wynik działania zadajnika FuzzyLogic. Regulator PID potrzebuje do działania także aktualną (zmierzoną) wartość ciśnienia bezwzględnego oraz element wykonawczy. Ciśnienie bez-

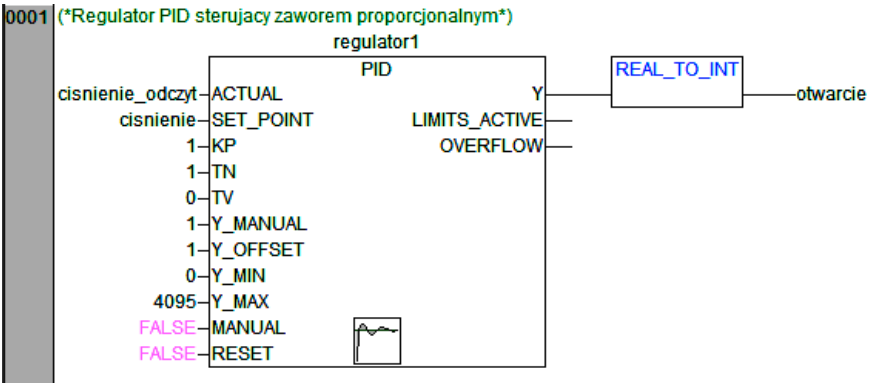
względne mierzone jest za pomocą czujnika ciśnienia, a elementem wykonawczym jest zawór proporcjonalny (Juszka i in., 2011b).

Oprogramowanie CoDeSys zastosowane do zaprogramowania sterownika PLC posiada bibliotekę Util.lib., w której znajduje się blok funkcyjny regulatora PID. W bloku tym konieczne jest zaprogramowanie następujących parametrów niezbędnych do jego poprawnej pracy:

– wartość rzeczywista wielkości regulowanej;
– wartość zadana;
– współczynnik wzmocnienia części P;
– czas całkowania, odwrotny współczynnik wzmocnienia części I;
– czas różniczkowania, współczynnik wzmocnienia części D;
– offset wielkości nastawczej;
– dolna i górna granica wielkości nastawczej (Brzózka, 2004).

Blok funkcyjny regulatora PID widoczny z poziomu programatora sterownika PLC zamieszczono na rysunku 149.

Wartość wyjściowa bloku PID jest typu REAL, dlatego na wyjściu regulatora zastosowano dodatkowo blok funkcyjny REAL_TO_INT, przekształcający wartość typu REAL na wartość całkowitą typu INT. Zmiana typu danych wynika z wejścia bloku PWM (modulacja szerokości impulsu) sterującego zaworem, które oczekuje danych 12-bitowych typu INT.

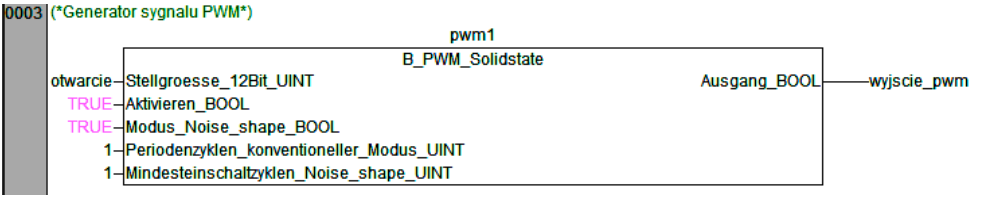


Rysunek 149. Blok funkcyjny regulatora PID

Sterowanie zaworami

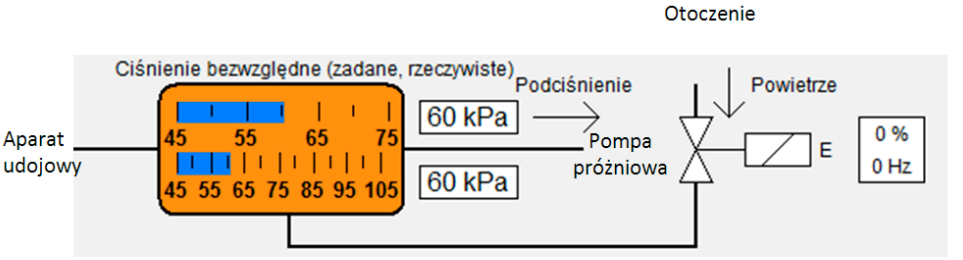
Program umożliwia sterowanie zaworami za pomocą sygnału PWM (Pulse-width modulation) o regulowanej szerokości impulsów. Sterowanie PWM pozwala sterować w sposób ciągły za pomocą wartości dyskretnych (0, 1). Dzięki temu za pomocą wyjścia cyfrowego można sterować np. zaworem proporcjonalnym, doprowadzającym powietrze atmosferyczne do zbiornika z ciśnieniem bezwzględnym dla komory podstrzykowej kubka udojowego. Częstotliwość sygnału PWM jest stała i musi być na tyle wysoka, by element wykonawczy nie oscylował z tą częstotliwością, czyli okres musi być krótszy niż czas bezwładności elementu wykonawczego. Wartością regulowaną sygnału PWM jest szerokość impulsu, czyli jego wypełnienie, może się ono wahać od 0 do 100%.

W programie CoDeSys sygnał PWM uzyskuje się za pomocą bloku funkcyjnego B_PWM_Solidstate (rys. 150), należącego do biblioteki Closed-Loop Control Toolbox. Zastosowanie tego sygnału w sterownikach PLC wiąże się z pewnymi ograniczeniami. Okres sygnału nie może być krótszy niż czas trwania cyklu programu.



Rysunek 150. Blok funkcyjny generujący sygnał PWM

Opisany powyżej układ sterowania ciśnieniem bezwzględnym, zasilający komory kolumny kolektora, odzwierciedlono w formie wizualizacji, która umożliwiała obserwację jego pracy oraz rejestrację danych do analizy (rys. 151).



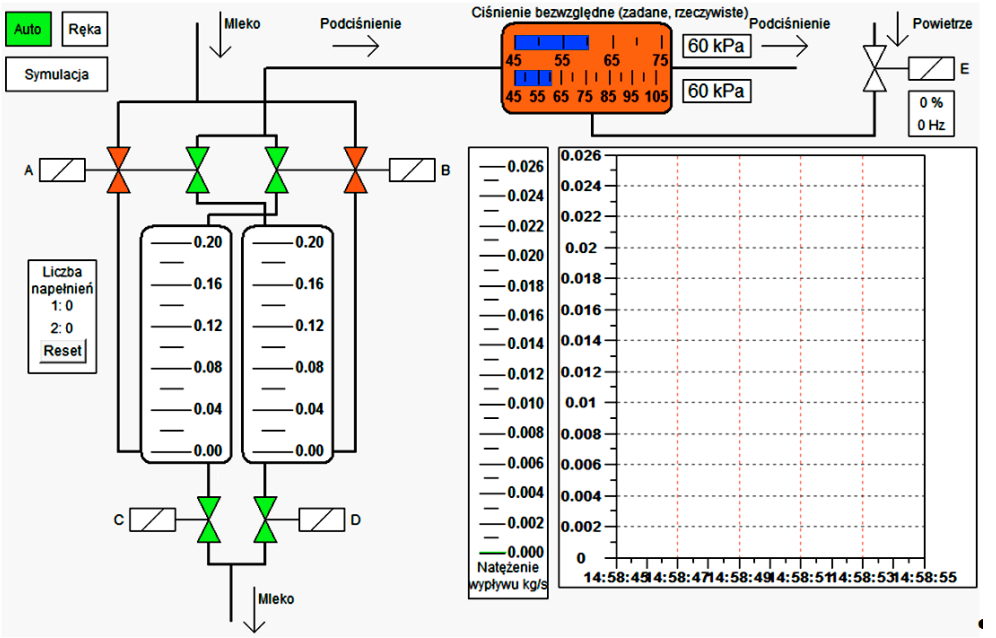
Rysunek 151. Wizualizacja układu podciśnieniowego

Wizualizacja układu sterowania

Działanie całego układu sterowania zobrazowano w formie wizualizacji. Za pośrednictwem komputera PC podłączonego do sterownika można odczytać wszystkie najważniejsze parametry pracy układu sterowania. Należą do nich:

- poziom mleka w obu zbiornikach kolektora;
- stan zaworów doprowadzających i odprowadzających mleko z kolektora;
- obliczona wartość natężenia strumienia masowego (objętościowego) mleka;
- wartość ciśnienia bezwzględnego (zadana i rzeczywista);
- stopień otwarcia zaworu regulującego ciśnienie bezwzględne w kolektorze;
- tryb pracy układu (automatyczny, ręczny, symulacja).

Po uruchomieniu układ sterowania pracuje w trybie automatycznym. Za pomocą wizualizacji układu sterowania możliwe jest także sterowanie ręczne kolektorem. Korzystając z myszy lub klawiatury komputerowej, można włączyć tryb ręczny i sterować zaworami kolektora. Aby zmienić pozycję zaworu, należy nacisnąć lewym przyciskiem myszy na symbol danego zaworu lub wybrać na klawiaturze literę odpowiadającą oznaczeniu danego zaworu (A, B, C lub D). Okno wizualizacji przedstawiono na rysunku 152.



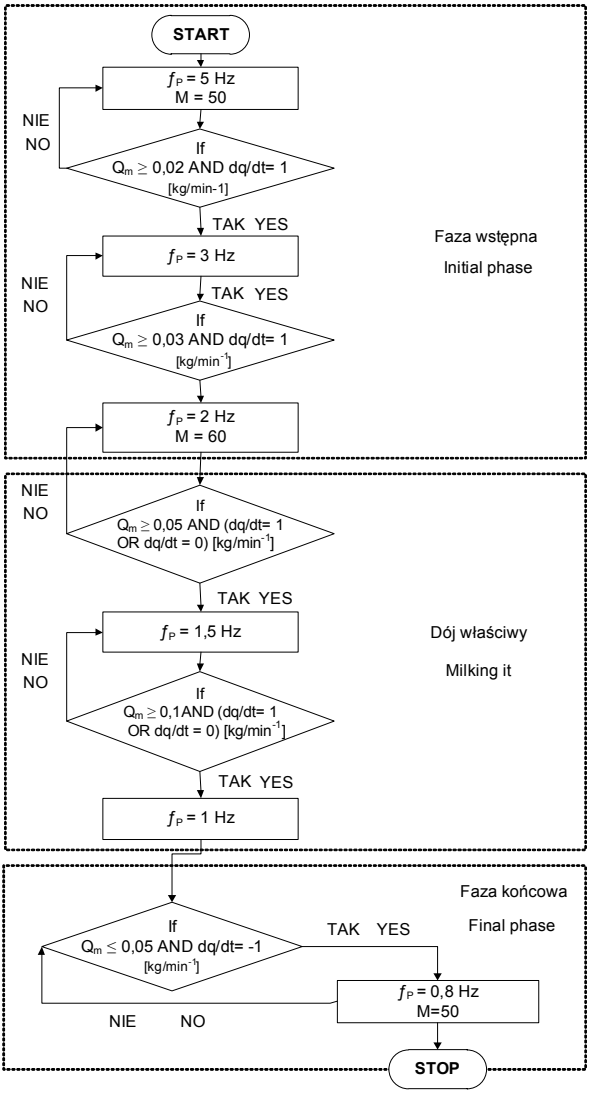
Rysunek 152. Okno wizualizacji pracy całego systemu sterowania kolumny kolektora AAU

1.2.4. Zastosowanie sterowników PLC do sterowania pulsacją w doju maszynowym krów

Jednym z ważniejszych urządzeń w zmechanizowanym systemie udojowym jest pulsator. Wytwarza on pulsację, czyli przemiennie podciśnienie i ciśnienie atmosferyczne w komorze międzyściennej kubka udojowego w celu masażu i stymulacji strzyków wymienia krowy, poprawiając m.in. ich ukrwienie. Proces doju dzieli się na trzy podstawowe fazy: wstępną, dój właściwy, końcową. W każdej z tych faz zarówno częstotliwość pulsacji, jak i podciśnienie powinny posiadać inne wartości. Bardzo ważnym zagadnieniem jest moment przejścia pomiędzy poszczególnymi fazami. Powinien on wynikać z pomiaru natężenia wypływu mleka ze strzyka wymienia krowy. W przedstawionym poniżej przykładzie opisano program realizujący funkcję pulsatora na sterowniku PLC. Parametry takiego pulsatora mogą być dynamicznie modyfikowane w zależności od natężenia wypływu mleka ze strzyka wymienia krowy (Juszka i in., 2012b).

Każda z faz doju charakteryzuje się innymi warunkami i sposobem prowadzenia. Konieczne jest, aby w czasie rzeczywistym mierzyć natężenie wypływu mleka ze strzyka i obserwować jego trend zmian. Klasyczne pulsatory rozdzielają fazy doju ze względu na zadany czas i często przechodzą pomiędzy nimi w nieodpowiednich momentach. Ponadto przejścia pomiędzy fazami są często gwałtowne, gdyż parametry pulsacji nie zmieniają się w ich obrębie. W pulsatorze zaprogramowanym na sterowniku PLC zmiany parametrów pulsacji ciśnienia powietrza w komorze międzyściennej sterowane są na podstawie pomiaru

natężenia wypływu mleka ze strzyka wymienia krowy oraz drugiego parametru, którym jest pochodna chwilowego natężenia wypływu po czasie. Dzięki temu pulsator w czasie rzeczywistym, poprzez analizę powyższych parametrów, dobiera optymalne wartości parametrów pulsacji. W algorytmie przedstawionym na rysunku 153 uwzględniono zarówno pomiar wypływu mleka Q_m ze strzyka, jak i trend jego zmian w czasie oznaczony *dq/dt*. W badaniach wstępnych uproszczono wartość trendu, przyjęto trend równy „1", odpowiadający wzrostowi natężenia wypływu mleka, natomiast „0" dla wartości stałej w czasie oraz „-1" dla spadku tej wartości (Juszka i in., 2010).



Rysunek 153. Algorytm sterowania pulsacją

Na rysunku 154 przedstawiono okno deklaracji zmiennych programu sterującego pulsacją. Przyjęto założenie, że zmiany parametrów pulsacji, a więc częstotliwość i współczynnik pulsacji, powinny przebiegać łagodnie, stąd w programie sterującym stopniowano zmiany wartości parametrów pulsacji. W zaprogramowanym pulsatorze zmiany zachodzą w funkcji natężenia wypływu mleka oraz w funkcji trendu zmian, a więc pochodnej natężenia wypływu mleka po czasie. W nawiasach zamieszczono komentarze opisujące zmienne programu. Ostatnich kilka wierszy deklaracji zmiennych opisuje zmienne konieczne do wyznaczenia wartości trendu zmian natężenia wypływu mleka ze strzyka wymienia krowy.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     Q: REAL := 0.001;      (*Dane z czujnika natężenia wypływu*)
0004
0005
0006     BLINKpulsacja: BLINK;    (*Deklaracja bloku BLINK*)
0007     CTUpuls: CTU;
0008     reset: BOOL;
0009     zliczone: BOOL;
0010     war_licz: WORD;
0011                                  (*b,d,f,h,j,l  wyjścia można zamienić na jedno*)
0012     a:BOOL;
0013         b:BOOL;                  (*Częstotliwość f= 5HZ, Stosunek M=50*)
0014     c: BOOL;
0015         d: BOOL;                 (*Częstotliwość f= 3HZ, Stosunek M=50*)
0016     e: BOOL;
0017         f: BOOL;                 (*Częstotliwość f= 2HZ, Stosunek M=60*)
0018     g: BOOL;
0019         h: BOOL;                 (*Częstotliwość f= 1.5, Stosunek M=60*)
0020     i: BOOL;
0021         j: BOOL;                 (*Częstotliwość f= 1HZ, Stosunek M=60*)
0022     k: BOOL;
0023         l: BOOL;                 (*Częstotliwość f= 0.8, Stosunek M=60*)
0024
0025     GENpuls: GEN;                (*GEnerowanie funkcji dla chwilowego natężenia wypływu "cze*)
0026     cze: REAL;
0027     DERIVATIVEpuls: DERIVATIVE;
0028     qout: REAL;
0029
0030     GENQ: GEN;
0031
0032     u: BOOL ;
0033
0034 END_VAR
```

Rysunek 154. Deklaracja zmiennych dla programu pulsatora

Polecenie „VAR” poprzedza deklaracje poniższych zmiennych:

- zmienna „Q” typu Real reprezentuje natężenie wypływu mleka ze strzyka;
- zmienna „BLINKpulsacja” bloku BLINK wytwarza sygnał sterujący pulsacją;
- zmienna „CTUpuls” bloku CTU zlicza pulsy;
- zmienna „reset” typu BOOL resetuje bloki;

– zmienna „zliczone” typu BOOL w zależności od stanu – załącza lub wyłącza licznik;
– zmienna „war_licz” typu WORD przedstawia liczbę wykonanych pulsów;
– zmienne „a, c, e, g, i, k” typu BOOL przełączają poszczególne etapy doju;
– zmienne „b, d, f, h, j, l” typu BOOL załączają wyjścia PLC danych częstotliwości;
– zmienna „GENpuls” bloku GEN wywołuje generator funkcji;
– zmienna „cze” typu REAL przedstawia wartość wygenerowanej częstotliwości;
– zmienna „DERIVATIVEpuls” bloku DERIVATIVE uruchamia obliczanie pochodnej;
– zmienna „qout” typu REAL – oznacza wartość obliczonej pochodnej w bloku DERIVATIVE;
– zmienna „u” typu BOOL jest pomocniczą dla stymulacji natężenia wypływu.

Poleceniem END_VAR zakończono proces deklaracji zmiennych.

Program odzwierciedlający założenia przedstawione w powyższym algorytmie przedstawiają rysunku 155 – 157. Faza wstępna rozpoczyna się w 24 wierszu programu (rys. 155), a kończy w 42 linii programu (rys. 156). Sterownik porównuje wartość zmierzoną natężenia wypływu mleka z wartością graniczną, przypisaną do poszczególnych faz, i wykonuje pętle warunkowe programu. Faza doju właściwego zawarta jest pomiędzy 42 a 64 linią programu, natomiast faza końcowa (podój) od 64 do 85 linii. Elementem odpowiadającym za dobór współczynnika pulsacji i częstotliwości jest pętla warunkowa IF.

Program umieszczony pomiędzy 10 a 19 wierszem (rys. 155) jest symulatorem wypływu mleka ze strzyka wymienia krowy, w trybie symulacji komputerowej jest konieczne jego działanie.

```
0001 GENpuls(MODE:=TRIANGLE_POS ,BASE:=FALSE ,PERIOD:=T#3s ,CYCLES:=200 ,AMPLITUDE:=100 ,RESET:=FALSE ,OUT=>cze );
0002
0003
0004
0005        DERIVATIVEpuls(IN:=Q,TM:=10,RESET:= , OUT=>qoul );  |  (*Obliczanie pochodniej chwilowego natężenia po czasie*)
0006
0007
0008
0009
0010 IF
0011     Q<0.12 AND Q>0.0001
0012 THEN
0013         u:=TRUE;
0014         Q:=Q+0.0001;                          (*Symulowanie wypływu*)
0015 ELSE
0016         u:=FALSE;
0017         Q:=0.0;
0018         qoul:=-1;
0019 END_IF;
0020
0021
0022
0023
0024 IF
0025     Q< 0.02 AND qoul>0 AND qoul<=1
0026 THEN
0027     a:=TRUE;
0028     BLINKpulsacja(ENABLE:=a , TIMELOW:=T#0.1S , TIMEHIGH:=T#0.1S , OUT=>b );
0029     b:=BLINKpulsacja.OUT;
0030     CTUpuls(CU:=b , RESET:=reset , PV:=100 , Q=>zliczone , CV=>war_licz );
0031     zliczone:=CTUpuls.Q;
0032     war_licz:=CTUpuls.CV;
0033 ELSIF
0034     Q>= 0.02 AND Q<0.03 AND qoul>0 AND qoul<=1
```

Rysunek 155. Program sterujący pulsacją – część 1

Program składa się z kilku takich pętli, im większa ich liczba, tym lepsze dopasowanie pulsacji do wypływu mleka, jednak wraz ze wzrostem liczby pętli zwiększa się czas wykonywania programu. Każda pętla posiada pulsator z ustawionymi przerwami dla zegara taktującego, odpowiedzialnego za częstotliwość pulsacji oraz za współczynnik pulsacji.

```
0035 THEN
0036     c:=TRUE;
0037     BLINKpulsacja(ENABLE:=c , TIMELOW:=T#0.165S , TIMEHIGH:=T#0.165S , OUT=>d );
0038     d:=BLINKpulsacja.OUT;
0039     CTUpuls(CU:=d , RESET:=reset , PV:=100 , Q=>zliczone , CV=>war_licz );
0040     zliczone:=CTUpuls.Q;
0041     war_licz:=CTUpuls.CV;
0042 ELSIF
0043     Q>= 0.03 AND Q<0.05 AND qout>0 AND qout<=1
0044 THEN
0045     e:=TRUE;
0046     BLINKpulsacja(ENABLE:=e , TIMELOW:=T#0.3S , TIMEHIGH:=T#0.2S , OUT=>f
0047 );
0048     f:=BLINKpulsacja.OUT;
0049     CTUpuls(CU:=f , RESET:=reset , PV:=100 , Q=>zliczone , CV=>war_licz );
0050     zliczone:=CTUpuls.Q;
0051     war_licz:=CTUpuls.CV;
0052 ELSIF
0053     Q>= 0.05 AND Q<0.1 AND qout>=0 AND qout<=1
0054 THEN
0055     g:=TRUE;
0056     BLINKpulsacja(ENABLE:=g , TIMELOW:=T#0.4S , TIMEHIGH:=T#0.27S , OUT=>h
0057 );
0058     h:=BLINKpulsacja.OUT;
0059     CTUpuls(CU:=g , RESET:=reset , PV:=100 , Q=>zliczone , CV=>war_licz );
0060     zliczone:=CTUpuls.Q;
0061     war_licz:=CTUpuls.CV;
0062
```

Rysunek 156. Program sterujący pulsacją – część 2

```
0063 ELSIF
0064     Q>= 0.1 AND qout>=0 AND qout<=1
0065 THEN
0066     i:=TRUE;
0067     BLINKpulsacja(ENABLE:=i , TIMELOW:=T#0.6S , TIMEHIGH:=T#0.4S , OUT=>j
0068 );
0069     j:=BLINKpulsacja.OUT;
0070     CTUpuls(CU:=j , RESET:=reset , PV:=100 , Q=>zliczone , CV=>war_licz );
0071     zliczone:=CTUpuls.Q;
0072     war_licz:=CTUpuls.CV;
0073
0074 ELSIF
0075     Q<= 0.02 AND qout<0 AND qout>= -1
0076 THEN
0077     k:=TRUE;
0078     BLINKpulsacja(ENABLE:=k , TIMELOW:=T#0.625S , TIMEHIGH:=T#0.625S , OUT=>l
0079 );
0080     l:=BLINKpulsacja.OUT;
0081     CTUpuls(CU:=l , RESET:=reset , PV:=100 , Q=>zliczone , CV=>war_licz );
0082     zliczone:=CTUpuls.Q;
0083     war_licz:=CTUpuls.CV;
0084
0085 END_IF;
```

Rysunek 157. Program sterujący pulsacją – część 3

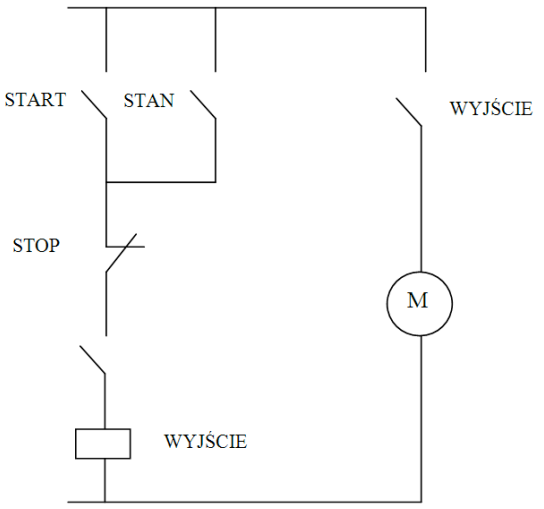
1.3. Inżynieria produkcji spożywczej

W kilku przykładach związanych tematycznie z inżynierią produkcji przybliżony zostanie język tekstowy lista instrukcji IL. Wyróżnia się on spośród pozostałych języków programowania szybkością wprowadzania kodu oraz szybkością wykonywania programu (norma zaleca krótki kod programu), stąd przedstawiony zostanie w strategicznych zastosowaniach, gdzie te cechy są szczególnie pożądane. Przykładami będą: tryb awaryjnego włączania/wyłączania przenośników transportowych oraz generator fali prostokątnej, który jest możliwy do zaprogramowania we wszystkich językach, jednak tylko *lista instrukcji* umożliwia jego poprawne działanie (Juszka i Tomasik, 2005).

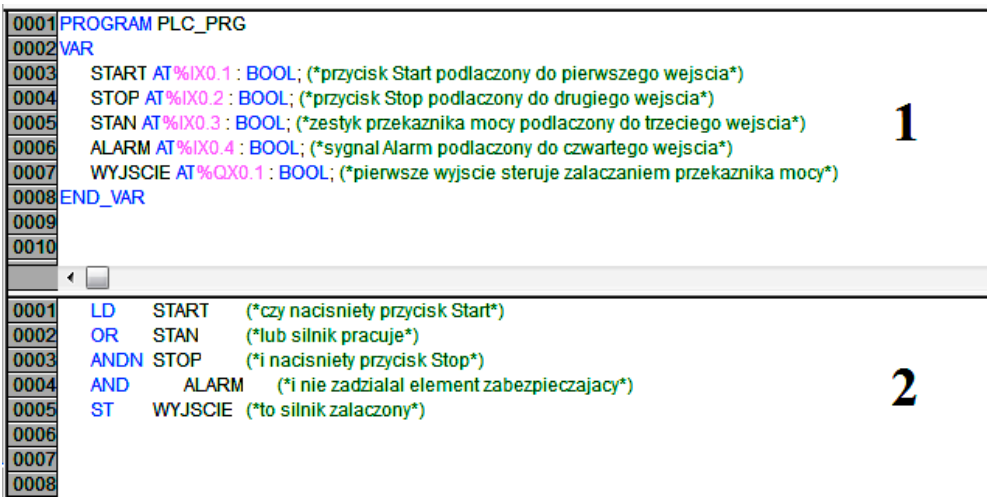
Program startu i zatrzymywania przenośnika transportowego

Program polega na uruchamianiu i zatrzymaniu silnika napędu urządzenia transportującego. Schemat elektryczny uruchamiania silnika przenośnika transportowego pokazany jest na rysunku 158. Do dyspozycji są dwa przyciski: *Start* (styk zwierny) oraz *Stop* (styk rozwierny), za pomocą których napęd jest uruchamiany/zatrzymywany. Do sterownika wprowadzony jest sygnał ze zestyków (zwiernych) przekaźnika włączającego silnik (informuje o stanie napędu) oraz sygnał alarmowy z elementu zabezpieczającego rozwiernego (np. wyłącznika krańcowego, termicznego) (Juszka i Tomasik, 2005).

Na rysunku 159 przedstawiono okno zawierające kod programu sterującego napędem przenośnika transportowego. W górnej części (1) zamieszczono deklarację zmiennych, która rozpoczyna się słowem kluczem „VAR” i kończy „END_VAR”. Pomiędzy tymi słowami zdeklarowano 5 zmiennych, wszystkie są typu BOOL. Deklaracja każdej zmiennej rozpoczyna się od wprowadzenia jej nazwy, następnie kojarzona jest z konkretnym adresem fizycznym (wejścia, wyjścia lub pamięci). W dolnej części okna (2) zamieszczono kod działania programu. Program rozpoczyna się od polecenia LD, jest to pierwsze obowiązkowe polecenie dla języka lista instrukcji, oznaczające załadowanie do akumulatora zmiennej *Start*. Jeśli *Start* ma wartość logiczną „1” lub *Stan* ma wartość logiczną „1” (samopodtrzymanie) oraz *Stop* ma wartość logiczną „0” i nie zadziałał element zabezpieczający (sygnały *Stop* i *Alarm* są dominujące w stosunku do sygnałów *Start* i *Stan*), to *Wyjście* sterownika powinno być w stanie *ON*, czyli mieć wartość logiczną „1”.



Rysunek 158. Schemat elektryczny uruchamiania silnika przenośnika transportowego



Rysunek 159. Deklaracja zmiennych i kod programu

Program sterowania przenośnikiem z opóźnieniem

Program polega na sterowaniu załączaniem/wyłączaniem dwóch napędów taśmociągów. Do dyspozycji są trzy przyciski: *Start*, *Wyłącz* (w trybie normalnym) i *Awaria* (wyłącz w trybie awaryjnym). Przycisk *Start* jest zwierny, natomiast przycisk *Wyłącz* i *Awaria* są rozwierne.

Na rysunku 160 przedstawiono okno zawierające kod programu sterującego napędem przenośnika transportowego z opóźnieniem. W górnej części (1) zamieszczono deklarację zmiennych do sterowania napędami, rozpoczynającą się słowem kluczem „VAR" i kończącą „*END_VAR*". Pomiędzy nimi zdeklarowano 6 zmiennych typu BOOL. Deklaracja każdej zmiennej rozpoczyna się od wprowadzenia jej nazwy, następnie skojarzenia z konkretnym adresem fizycznym. W dolnej części okna (2) zamieszczono kod działania programu. Program rozpoczyna się poleceniem LD, załadowanie zmiennej *Start*, co spowoduje załączenie obu napędów. Polecenie LDN, oznaczające załadowanie zmiennej *Wyłącz,* spowoduje wyłączenie w trybie normalnym najpierw pierwszego napędu, natomiast po czasie 2 sekund drugiego napędu. Działanie zegara odliczającego czas 2 sek. opisane jest poleceniem ST ZEGAR.IN, co oznacza uruchomienie zegara przy wyłączonym napędzie 1 i załączonym napędzie 2, ustawiany jest czas 2 sek., a następnie wywołanie bloku funkcjonalnego zegara oraz zapamiętanie wyjścia zegara w zmiennej pomocniczej. Naciśnięcie przycisku Awaria spowoduje natychmiastowe wyłączenie obu napędów.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     START     AT%IX0.1     :BOOL; (*przycisk Start podlaczony do pierwszego wejscia*)
0004     WYLACZ    AT%IX0.2     :BOOL; (*przycisk Wylacz podlaczony do drugiego wejscia*)
0005     AWARIA    AT%IX0.3     :BOOL; (*przycisk Awaria podlaczony do trzeciego wejscia*)
0006     NAPED1    AT%QX0.1     :BOOL; (*pierwsze wyjscie steruje zalaczaniem pierwszego napedu*)
0007     NAPED2    AT%QX0.2     :BOOL; (*drugie wyjscie steruje zalaczaniem drugiego napedu*)
0008     MARKER                 :BOOL; (*zmienna pomocnicza do wylaczenia drugiego napedu*)
0009     ZEGAR : TON;                  (*deklaracja egzemplarza czasomierza TON*)
0010 END_VAR
```

1

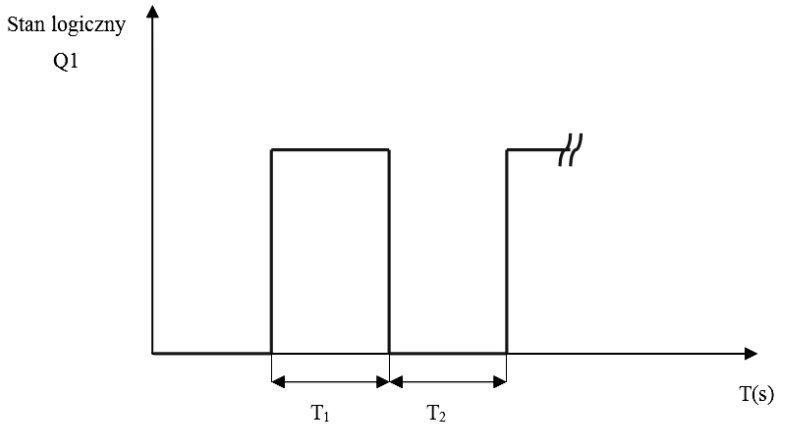
```
0001     (*zalaczanie napedow z uzyciem operatorow S*)
0002     LD      START     (*czy nacisniety przycisk Start*)
0003     S       NAPED1    (*zalacz Naped 1*)
0004     S       NAPED2    (*zalacz Naped2*)
0005
0006
0007     (*wylaczanie Napedu 1*)
0008     LDN       WYLACZ       (*czy nacisniety przycisk Wylacz*)
0009     ORN       AWARIA       (*lub nacisniety przycisk Awaria*)
0010     R       NAPED1       (*to wylacz naped 1*)
0011
0012     (*pomiar czasu 2s do wylaczenia Napedu 2*)
0013     LDN       NAPED1
0014     AND       NAPED2
0015     ST      ZEGAR.IN     (*jesli Naped 1 wylaczony i Naped 2 zalaczony, to uruchom Zegar*)
0016     LD      T#2S
0017     ST      ZEGAR.PT     (*ustawienie czasu 2s*)
0018     CAL       ZEGAR        (*wykonanie bloku funkcjonalnego Zegara*)
0019     LD      ZEGAR.Q
0020     ST      MARKER       (*zapamietanie wyjscia Zegara w zmiennej pomocniczej*)
0021
0022     (*wylaczenie Napedu 2*)
0023     LD      MARKER       (*jesli ustawiony Marker*)
0024     ORN       AWARIA       (*lub nacisniety przycisk Awaria*)
0025     R       NAPED2       (*to wylacz Naped 2*)
0026
0027
```

2

Rysunek 160. Deklaracja zmiennych oraz program do sterowania dwoma systemami napędowymi

Program modulujący falę prostokątną

Program polega na wygenerowaniu na wyjściu cyfrowym sygnału prostokątnego o zadanym okresie i stopniu wypełnienia. Sygnał można wykorzystać do sterowania urządzeniami elektrycznymi, zasilanymi prądem stałym, sterowanie polega na modulacji szerokości impulsu tzw. PWM (Jabłoński, 1998). Ze względu na czas wykonywania programu tylko język lista instrukcji umożliwia zaprogramowanie i poprawne działanie takiego generatora, co wynika z tego, że język ten jest najszybciej wykonywany ze wszystkich. Cykl wykonywania pętli programu zawiera się w przedziale od 0,5 ms do 10 ms, przy czym te mniejsze wartości dotyczą właśnie języka lista instrukcji. Programując generator fali prostokątnej, należy mieć na uwadze cykl pętli programu danego sterownika, który może skutecznie uniemożliwić zaprogramowanie takiego generatora. Przebieg fali prostokątnej pokazany jest na rysunku 161 – T_1, T_2 to okresy fali.



Rysunek 161. Przebieg fali prostokątnej

Na rysunku 162 przedstawiono program generatora PWM. Tradycyjnie rozpoczyna się on deklaracją zmiennych, zamieszczoną pomiędzy słowem kluczem „*VAR*" i „*END_VAR*", zdeklarowano 8 zmiennych. Zmienne: *Wyjście* i *Marker* są typu BOOL, *Okres* i *Wypeln* są typu INT, czyli stanowią liczby całkowite, T_1 oraz T_2 to zmienne typu TIME, natomiast *Pomoc* to zmienna typu DINT, co oznacza podwójną liczbę całkowitą. Kod programu rozpoczyna się od obliczenia czasu T_1 poleceniem LD, następnie zachodzi konwersja wartości okresu i wypełnienia na liczby typu DINT, po czym po przemnożeniu okresu przez wypełnienie, wartość jest jeszcze dzielona przez 100. Konwersja wartości na liczby typu DINT pozwoliła uniknąć wystąpienia błędu przepełnienia w trakcie mnożenia liczb typu INT. Po obliczeniu czasu T_2 następuje wywołanie zegara. Wykorzystano dwa czasomierze wzajemnie sprzężone (wyjście jednego czasomierza uruchamia drugi czasomierz i odwrotnie). Czasomierz *Zegar1* po czasie T2 załącza wyjście, a czasomierz *Zegar2* po czasie T1 wyłącza wyjście.

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     WYJSCIE AT %QX0.1 :BOOL; (*wyjscie, na ktorym pojawia sie sygnal fali prostokatnej*)
0004     OKRES AT %MW1 :INT; (*okres sygnalu prostokatnego jako liczba milisekunda*)
0005     WYPELN AT %MW2 :INT; (*stopien wypelnienia sygnalu w procentach*)
0006     T1 :TIME; (*czas, gdy sygnal wyjsciowy jest w stanie ON*)
0007     T2 :TIME; (*czas, gdy sygnal wyjsciowy jest w stanie OFF*)
0008     POMOC :DINT; (*zmienna pomocnicza di konwersji na typ DINT*)
0009     MARKER :BOOL; (*zmienna pomocnicza do zerowania wyjœcia po czasie T1*)
0010     MS: BOOL;
0011 END_VAR
0012
0013 VAR
0014     ZEGAR1, ZEGAR2 : TON;      (*deklaracja egzemplarzy czasomierzy TON*)
0015 END_VAR
0016
0017
```

```
0001  (*obliczenie czasu T1*)
0002  LD    OKRES
0003  INT_TO_DINT
0004  ST    POMOC (*zapamietana wartosc okresu jako DINT*)
0005  LD    WYPELN
0006  INT_TO_DINT (*wartosc wypelnienia jako liczba DINT*)
0007  MULPOMOC (*okres razy wypelnienie*)
0008  DIV   100 (*podzielenie przez 100*)
0009  ST    POMOC (*zapamietana liczba ms dla czasu T1 jakoDINT*)
0010  LD    T#1MS (*jednostka czasu 1 ms*)
0011  TIME_MUL_DINT POMOC (*mnozenie przez liczbe ms*)
0012  ST    T1
0013
0014  (*obliczanieczasuT2*)
0015  LD    T#1MS (*jednostka czasu 1 ms*)
0016  TIME_MUL_INT OKRES (*mnozenie przez wartosc Okresu*)
0017  SUB       T1 (*odjecie czasu T2*)
0018  ST    T2 (*obliczony czas T2*)
0019
0020  (*wywolanie Zegara dla czasu T2*)
0021  LDN       MARKER
0022  ST    ZEGAR1.IN
0023  LD    T2
0024  ST    ZEGAR1.PT
0025  CAL       ZEGAR1
0026  LD    ZEGAR1.Q
0027  ST    WYJSCIE (*ustawienie wyjscia po czasieT2*)
0028
0029  (*wywolanie Zegara dla czasu T1*)
0030  LD    WYJSCIE
0031  ST    ZEGAR2.IN
0032  LD    T1
0033  ST    ZEGAR2.PT
0034  CAL       ZEGAR2
0035  LD    ZEGAR2.Q
0036  ST    MARKER (*ustawienie Markera zerujacego wyjscie po czasie T1*)
0037
```

Rysunek 162. Program w języku IL dla generatora fali prostokątnej

CZĘŚĆ CZWARTA
– SYSTEMY WIZUALIZACJI (SCADA)
W ROLNICZYCH PROCESACH PRODUKCYJNYCH

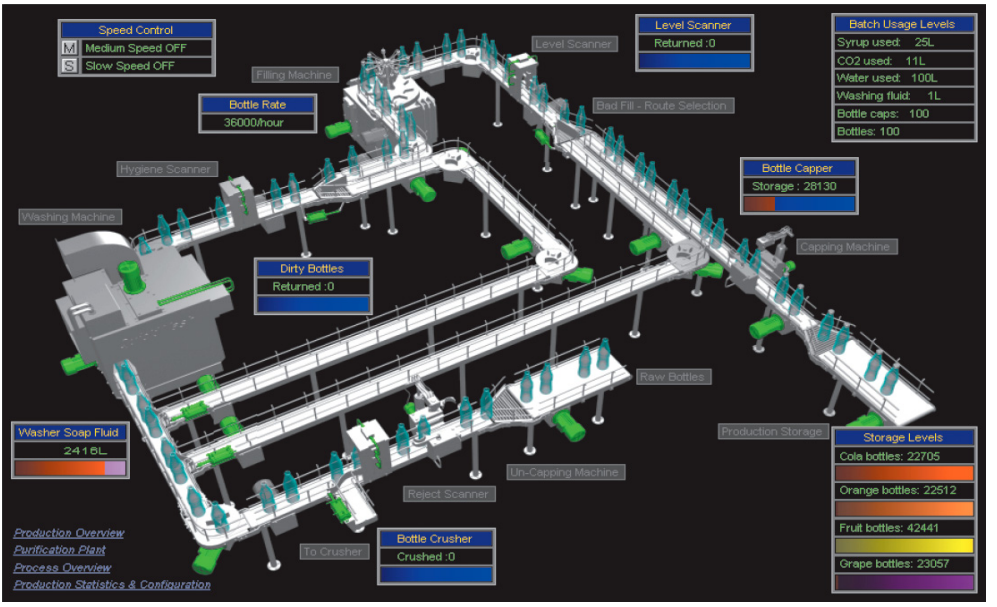
1. SYSTEMY NADZORUJĄCE PRZEBIEG PROCESU PRODUKCYJNEGO

1.1. Charakterystyka systemów wizualizacji i kontroli produkcji

Wizualizacja komputerowa stanu procesu jest zobrazowaniem w postaci animowanych obiektów tekstowych i graficznych zmian w nim zachodzących, których spójną całość komponuje się za pośrednictwem akcji sterujących ich wyświetlaniem, usuwaniem itp. Coraz częściej istnieje potrzeba obserwacji procesu, a także kontaktu z aparaturą kontrolno-pomiarową. SCADA (ang. Supervisory Control And Data Acquisition) oznacza system nadzorujący przebieg procesu technologicznego (Michta, 1998). Jego główne funkcje obejmują zbieranie aktualnych danych, ich wizualizację, sterowanie procesem, alarmowanie oraz archiwizację danych. System ten jest systemem komputerowym, który pełni rolę nadrzędną w stosunku do sterowników PLC i innych urządzeń. Na ogół to sterowniki PLC połączone są bezpośrednio z urządzeniami wykonawczymi (zawory, pompy itp.) i pomiarowymi (czujniki temperatury, poziomu itp.), i zbierają aktualne dane z obiektu oraz wykonują automatyczne algorytmy sterowania i regulacji. Za pośrednictwem sterowników PLC dane trafiają do systemu komputerowego i tam są archiwizowane oraz przetwarzane na formę bardziej przyjazną dla użytkownika. Operatorzy systemu zadają generalne parametry procesu lub prowadzą proces w trybie ręcznym. Przykład wizualizacji dla linii rozlewniczej z myjnią butelek zamieszczono na rysunku 163.

W procesach produkcyjnych, wizualizacja może wspomagać kontrolę oraz zarządzanie poszczególnymi etapami produkcji. W każdym rodzaju produkcji wymagana jest jak największa wydajność oraz wysoka jakość produktu. Aby spełnić wymagania norm oraz oczekiwania klientów, niezbędna staje się dokładna kontrola przebiegu procesu w celu szybkiego wykrycia i wyeliminowania błędów. Wykorzystanie systemów wizualizacji w procesach produkcji pozwala na efektywne i szybkie zlokalizowanie usterki, co daje możliwość skrócenia przestojów (Bailey, 2003).

Pierwszym – najprostszym sposobem wizualizacji były *tablice synoptyczne,* wyposażone w lampki, wskaźniki pomiarowe, natomiast schematy całych ciągów technologicznych były odwzorowane w postaci rysunków na obudowach szaf sterowniczych. Choć były to systemy proste, to jednak awaryjne, przepalenie lampki sygnalizacyjnej mogło błędnie oddawać aktualny stan procesu, stąd stany awaryjne wspomagane były dodatkowymi sygnałami dźwiękowymi. Wadą takiej wizualizacji było także to, iż jakiekolwiek zmiany na linii produkcyjnej skutkowały przebudową szafy, a nawet jej wymianą, co było czasochłonne i kosztowne. Na początku lat dziewięćdziesiątych pojawiły się komputerowe systemy wizualizacyjne jako tzw. rozwiązania semigraficzne. Polegały one na odwzorowaniu grafiki w trybie tekstowym przy użyciu odpowiedniej kompozycji znaków.



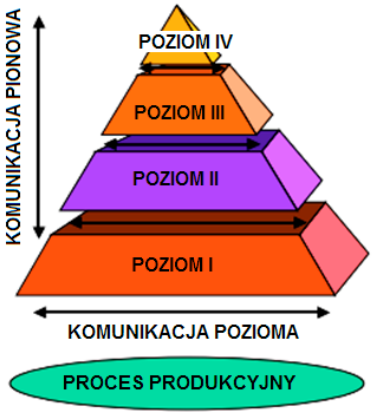
Rysunek 163. Przykładowa wizualizacja linii rozlewniczej

Źródło: Pawełczyk i in., 2009

Przy pomocy semigrafiki możliwe jest rysowanie zarówno obiektów prostych jak ramki i linie, jak i bardziej złożonych, dających złudzenie cieniowania. Semigrafika była szczególnie popularna w czasach, gdy komputery nie posiadały interfejsu graficznego. Dynamiczny rozwój tych systemów nastąpił z chwilą, kiedy firma Microsoft opracowała systemy operacyjne bazujące na oknach z możliwością pracy w sieci (Jakuszewski, 2006). Prekursorem oprogramowania HMI *(ang. Human Machine Interface*) na platformie Windows była firma Wonderware; obecnie na rynku jest wielu producentów, ale prym wiedzie właśnie ta firma z systemem o nazwie InTouch® (Astor, 2011). Wśród innych dostawców systemów SCADA wyróżnić można:

– **Asix** (producent: **Askom**) – powszechnie stosowany ze względu na korzystną cenę, posiada możliwość indywidualizacji interfejsu użytkownika końcowego bez ingerencji inżyniera systemu (wygląd i funkcjonalność aplikacji, sposób przeszukiwania zasobów), redundacja wszystkich stacji systemu;
– **Citect** (producent: **Schneider Citect**) – zawiera bezpłatne narzędzia projektowe oraz 250 driverów w standardzie;
– **iFIX, Cimplicity** (producent: GE Fanuc Automation) – dwa produkty tego samego producenta, będące elementem większego systemu informatycznego Proficy do kontroli produkcji (inne jego elementy: aplikacje do tworzenia i zrządzania przemysłowymi bazami danych, MES, portal raportowy), posiada wbudowany w pełnym zakresie język Visual Basic, umożliwiający tworzenie bardzo zindywidualizowanych aplikacji;

- **Industrial IT** (producent: **ABB**) – IndustrialIT to cała gama produktów, od programowalnych sterowników, do kompleksowych rozproszonych systemów sterowania, wspomagania i zarządzania produkcją dla instalacji i przedsiębiorstw;
- **InTouch** (producent: **Wonderware**) – popularny ze względu na łatwość tworzenia aplikacji, przez co prace programistyczne trwają krótko, bardzo wszechstronny, stosowany w różnych sektorach gospodarki; w przypadku oprogramowania SCADA firma Wonderware oprócz InToucha oferuje także Platformę Systemową Wonderware stosowaną w rozbudowanych aplikacjach SCADA;
- **PlantScape** (producent: **Honeywell**) – dedykowany do małych i średnich instalacji produkcyjnych;
- **Plant View** (producent: **FAST Automation**) – narzędzie służące do tworzenia systemów sterowania, zarządzania i kontroli zarówno dużym zakładem, jak i pojedynczymi maszynami;
- **SCADA Pro2000** (producent: **MikroB**) – elastyczny system przygotowania i realizacji komputerowych systemów wizualizacji i monitorowania procesów przemysłowych, linii technologicznych, maszyn i urządzeń, a także nadzoru i sterowania w oparciu o programowalne sterowniki i urządzenia kontrolno-pomiarowe;
- **RSView** (producent: **Rockwell Automation**) – przeznaczony do obsługi bazy zmiennych procesowych o rozmiarach 150, 300, 1500, 32 000 elementów, zapewnia możliwość upgrade'u z bazy o niższej na wyższą liczbę punktów;
- **TwinCat** (producent: **Beckhoff**) – tworzenie programu użytkowego usprawniają rozbudowane funkcje debuggowania oraz symulacja i diagnostyka;
- **WinCC** (producent: **Siemens**) – służy do wizualizacji i sterowania procesami produkcyjnymi, dedykowany do obsługi sterowników produkcji Siemens;
- **Wizcon** (producent: **Wizcon Systems**) – rozwiązanie dla sterowania i kontroli systemami w kluczowych gałęziach przemysłu i automatyce budynkowej (Lewandowska, 2007; MikroB S.A., 2008; Fijałka, 2007; Systemy Scada, 2009).



Rysunek 164. Hierarchiczny system wizualizacji

Jedną z najpopularniejszych metod przedstawiania form automatyzacji procesów produkcyjnych jest hierarchiczna czterowarstwowa struktura, stanowiąca całościowy system informatyczny (rys. 164).

Każda z warstw przedstawia oddzielny poziom administrowania i kontrolowania procesu produkcji:

– poziom pierwszy – sterowanie przebiegiem produkcji przy wykorzystaniu: CNC (ang. *Computerized Numerical Control* – skomputeryzowane sterowanie numeryczne), układów sterowania PLC, urządzeń wykonawczych i kontrolno-pomiarowych;

– poziom drugi stanowi wizualizacja procesu realizowana za pomocą interfejsu człowiek-maszyna *HMI* (ang. *Human Machine Interfa-*

ce), czyli urządzeń prezentujących operatorowi stan procesu, umożliwiających zadawanie parametrów pracy lub system nadzoru, kontroli i zbierania danych tworzących system SCADA;
- poziom trzeci to systemy zarządzania i śledzenia przebiegu produkcji tzw. *MES* (ang. *Manufacturing Execution System*), składają się na niego operacje, takie jak planowanie produkcji, gospodarka materiałami, dokumentacja procesu, ewidencja produktu ikomponentów, raportowanie produkcji;
- poziom czwarty jest systemem zarządzania i planowania zasobów całego przedsiębiorstwa *ERP* (ang. *Enterprise Resource Planning*), należą do niego: zarządzanie dostawami, rozliczenia finansowe, kosztorysowanie, obsługa zamówień (Kościelny, 1998).

Hierarchiczna struktura stawia wysokie wymagania systemom SCADA, użytkownicy w zależności od konkretnego przeznaczenia, specyfiki obiektu i procesu produkcyjnego oczekują:
- *wielozadaniowości systemu,* która jest możliwa tylko przy dużej niezawodności działania systemu SCADA jako całości. Błąd, jaki może wystąpić w jednym zadaniu, nie może wpłynąć na pracę całego systemu sterowania i wizualizacji. Ponadto oczekuje się od takich systemów możliwości modyfikacji danych konfiguracyjnych w trybie „on-line". Praca w tym trybie umożliwia wprowadzanie zmian bez konieczności wyłączania systemu SCADA. Zmiany te mogą dotyczyć np.: zmiany wartości granicznych alarmów, dodawanie nowych lub likwidowanie istniejących procedur, konfigurowanie istniejących obrazów synoptycznych oraz wprowadzanie nowych itp.;
- *skalowalności systemu* – pozwala ona na ciągłą modyfikację struktury sieciowej systemu od pojedynczych stanowisk operatorskich, do kompleksowego systemu sterowania i nadzoru bez potrzeby zamiany aplikacji we wcześniej zainstalowanych stacjach;
- *otwartości systemu* dla innych aplikacji poprzez system standardowego środowiska pracy aplikacji, np. Windows, oraz standardowych sieci komunikacyjnych zarówno na poziomie procesowym, jak i operatorskim. Pozwala ona na przyłączanie do niego programów użytkownika, umożliwia współpracę z innymi aplikacjami (np. bazy danych, arkusze kalkulacyjne) oraz z inteligentnymi urządzeniami różnych producentów (szereg driverów komunikacyjnych). Ponadto cecha ta umożliwia łatwe uzupełnienie systemu o nowe moduły programowe. Pakiet wizualizacyjny może pracować na komputerach sprzężonych siecią lokalną ETHERNET w celu udostępniania informacji o procesie z komputerów posiadających bezpośredni sprzęg ze sterownikami do zdalnych stacji roboczych. Komputery nadzoru sprzężone bezpośrednio ze sterownikami pełnią rolę serwerów danych o procesie – pomiarów, sterowań, trendów, alarmów, raportów – dla sieciowych stacji roboczych. Sieciowy tryb pracy, możliwość przyłączenia do sieci zdalnych stanowisk za pomocą łączy modemowych (dzierżawionych, GSM), możliwość stosowania struktur nadmiarowych (redundancyjnych), praca w technice klient-serwer, dwukierunkowy dostęp do relacyjnych baz danych oraz przydzielanie użytkownikom uprawnień dostępu do procesu;
- *akwizycji i przetwarzania zmiennych procesowych*, dostępność wielu sterowników komunikacyjnych oraz możliwość jednoczesnej obsługi przez każdą stację operatorską kilku sterowników komunikacyjnych zapewnia równoległą współpracę z różnymi rodzajami sterowników i regulatorów. System nadzoru udostępnia mechanizmy programowe zarówno do przetwarzania cyklicznego – analogowe i binarne sygnały pomiaro-

we, jak i sporadycznego – zmienne wprowadzane przez obsługę. Zmienne procesowe można indywidualnie przetwarzać według zależności wprowadzonych w fazie konfiguracji: filtrować, linearyzować, przeliczać na jednostki fizyczne, kontrolować przekroczenie wartości alarmowych, zliczać liczbę załączeń itp.

- *niezawodności*, szczególnie tam, gdzie awaria może grozić życiu lub zdrowiu lub też spowodować poważne straty materialne. W takich przypadkach buduje się struktury równoległe (zapasowe – redundancyjne podwójne, a nawet potrójne) w odniesieniu do stacji operatorskich, serwerów baz danych oraz połączeń sieciowych;
- *chronionego dostępu,* co jest szczególnie istotne w połączeniu z otwartością systemu – system powinien zapewniać możliwość przydzielenia poszczególnym użytkownikom uprawnień do: administracji systemu, sterowania procesem, modyfikacji parametrów, dostępu do danych itp.;
- *funkcji oddziaływanie na proces,* nielimitowana liczba obiektów (w tym obiektów dynamicznych) na masce technologicznej stanowi o funkcjonalnym zobrazowaniu stanu procesu. Stacje operatorskie umożliwiają sterowanie procesem, przewyższając wygodą obsługi różnorodne panele operatorskie do sterowników i regulatorów. Operator może wprowadzać zmiany wartości zadanej, nastaw regulatorów i innych parametrów regulacji realizowanych w sterownikach, przełączać tryb pracy obwodu (sterowanie ręczne-automatyczne), włączać i wyłączać urządzenia (np.: pompy, wentylatory);
- *obsługi alarmów* poprzez ich sygnalizację wykrytych alarmów. System powinien posiadać możliwość generacji alarmów systemowych (o błędach programowych, błędach transmisji) oraz alarmów i ostrzeżeń technologicznych (informujących o przekroczeniach granic alarmowych, nieprawidłowych stanach zmiennych binarnych). Alarmy są sygnalizowane w specjalnych oknach alarmów bieżących i historycznych oraz są rejestrowane w dzienniku alarmów. Rozbudowany mechanizm filtracji alarmów bieżących wg typu, priorytetu oraz grupy alarmowej umożliwia ich przeglądanie oraz dogłębną analizę;
- *informowania o aktualnych i historycznych danych procesowych*, obiektami o szczególnym znaczeniu są wykresy czasowe przebiegów zmiennych procesowych. Systemy SCADA pozwalają na wyświetlanie dwóch typów trendów: bieżących i historycznych. Podczas pracy na wykresie wyświetlany jest horyzont czasowy, zakres wartości, ilość pisaków i kolory mogą być dynamicznie zmieniane przez operatora;
- *raportowania*, każdy proces ma swoją specyfikę i wymaga opracowania dla niego odpowiednich raportów. Łatwość opracowania i uruchomienia nowych raportów ma bardzo istotne znaczenie, ponieważ zbiór raportów potrzebnych obsłudze wzrasta w trakcie eksploatacji systemu. Raporty mogą być obliczane na żądanie operatora lub automatycznie według zdefiniowanego terminarza;
- *archiwizacji danych procesowych*, pojemność archiwum dostępnego on-line jest ograniczona jedynie pojemnością dysku. Narzędzia przetwarzania danych mogą wyliczać różne wielkości charakterystyczne na podstawie zarejestrowanych danych. Efektywne metody archiwizacji pozwalają na tej samej stacji komputerowej prowadzić wizualizację, sterować procesem i generować raporty nawet dla dużej liczby zmiennych (Astor, 2006; Mizga, 2002).

Cechą wspólną dla prawie wszystkich wymienionych wcześniej systemów SCADA jest rozdzielenie funkcji bezpośrednio wykorzystywanych do obsługi procesu technologicznego

od funkcji odpowiedzialnych za tworzenie i modyfikację całej aplikacji. Funkcje obsługi procesów technologicznych realizowane są przez tak zwaną operatorską część systemu wizualizacji (ang. runtime). Część ta wykorzystywana jest do prowadzenia wizualizacji procesu technologicznego i sterowania obiektem. Użytkownikiem operatorskiej części systemu wizualizacji są osoby odpowiadające za kontrolę i nadzór obsługiwanego przez ten system procesu technologicznego. Oferowane przez poszczególne programy menu dostępnych operacji zależy oczywiście od konkretnych możliwości danego pakietu wykorzystywanego do tworzenia systemu wizualizacji, ale sposób rozwiązania poszczególnych funkcji jest silnie uwarunkowany poprzez specyfikę wymagań związanych z konkretną technologią, sposobem prezentacji stanu obiektu określanym przez liczne normy branżowe, przyzwyczajeniami obsługi wynikającymi z dotychczasowych metod prowadzenia instalacji czy też wreszcie – ograniczeniami technicznymi, dotyczącymi wykorzystywanych do realizacji systemu urządzeń (Williams, 1992).

Część inżynierska systemu wizualizacji (ang. dewelopement system) wykorzystywana jest zazwyczaj przez osoby przygotowujące projekt takiego systemu oraz osoby realizujące konkretną aplikację. To właśnie od dostępnej w tej części palety funkcji użytkowych, od sposobu realizacji przez system poszczególnych operacji oraz od inwencji projektanta tworzącego konkretną aplikację zależy efektywność wykonanej aplikacji. Proces jej tworzenia wymaga czynnego udziału zarówno technologów decydujących o sposobie obsługi procesu, jak i późniejszych użytkowników systemu, od których oczekiwać się będzie jego efektywnego wykorzystania (Więcek, 1995).

W przypadku procesów technologicznych, rozproszonych geograficznie na znacznym obszarze, kluczową rolę odgrywają systemy SCADA, rejestrujące dane pomiarowe pochodzące z oddalonych obiektów oraz pozwalające na pełną zdalną kontrolę nad obiektami przemysłowymi. Na ekranie komputera zlokalizowanego w centralnej dyspozytorni operator dysponuje podglądem stanu całej instalacji. W przypadku pojawienia się stanów alarmowych, otrzymuje na ekranie powiadomienie o zaistniałym zdarzeniu. Za pomocą prostych kliknięć myszą może się poruszać po kolejnych widokach ze szczegółowymi fragmentami instalacji, co pozwala na szybką reakcję i zdalną diagnozę przyczyn alarmu, bez ponoszenia kosztów związanych z wysłaniem ekipy do oddalonego obiektu (Bednarek, 2001; Wrzuszczak, 2010).

Producenci systemów kontrolowania i monitorowania starają się nadążyć za wciąż wzrastającą liczbą różnego typu sieci przemysłowych oraz współpracujących z nimi urządzeń. Rozwiązaniem pozwalającym na przystosowanie danego systemu wizualizacji do pracy z nowymi układami automatyki, pochodzącymi od różnych producentów, może być stworzenie mechanizmu pozwalającego na łatwą adaptację oferowanego oprogramowania bądź to przez producenta poszczególnych urządzeń, bądź też bezpośrednio przez użytkownika systemu.

2. STRUKTURA I PROGRAMOWANIE SYSTEMU SCADA INTOUCH®

Wonderware-InTouch® to przemysłowe oprogramowanie zaprojektowane do wizualizacji oraz kontroli procesów produkcyjnych, w pełni zgodne z wytycznymi dla systemów klasy SCADA (Supervisory Control And Data Acquisition) oraz HMI (Human-Machine-Interface). Stanowi środowisko programistyczne do projektowania, testowania oraz wdrażania systemów udostępniających użytkownikom dane bezpośrednio z produkcji. Programowanie polega na budowaniu aplikacji z gotowych, łatwo konfigurowalnych elementów: obiektów graficznych. System SCADA może zostać zintegrowany z aplikacją GIS, udostępniając wówczas, jako jeden z widoków, mapę cyfrową z zaznaczonymi lokalizacjami poszczególnych obiektów instalacji przemysłowej. InTouch może korzystać z technologii nazwanej ArchestrA®, stanowi ona zaawansowaną architekturę z zakresu automatyki przemysłowej i informatyki, zaprojektowaną w celu przedłużenia cyklu użytkowania systemów istniejących od lat w przedsiębiorstwie. Możliwe jest to dzięki zastosowaniu najnowszych technologii informatycznych i otwartości na inne, działające już w przedsiębiorstwie systemy automatyki. Należy podkreślić, że ArchestrA nie jest kolejnym oprogramowaniem czy też linią produktów, lecz sposobem na integrację aplikacji przemysłowych (działających na platformach Microsoft'u®), umożliwiającym skorzystanie ze wszystkich najnowszych technologii dostępnych pod systemami firmy Microsoft. ArchestrA pozwala także na łatwiejsze tworzenie aplikacji, które dzięki niej mogą być składane (budowane) z istniejących elementów bez konieczności programowania od podstaw. Nowo powstające aplikacje mogą być tworzone poprzez proste przekonfigurowanie już istniejących (Astor, 2009; Fijałka, 2007; Masłowski i in., 2008).

Celem powstania ArchestrA była wizja dostarczenia jednolitej i ergonomicznej architektury, która byłaby podstawą tworzenia systemów zarządzania produkcją, stanowiących wsparcie dla przedsiębiorstw przemysłowych (Astor, 2010; Garbacki, 2005).

Platforma Wonderware System Platform® składa się z zestawu usług oraz aplikacji opartych o technologię ArchestrA, która pozwala w efektywny sposób tworzyć oraz rozwijać rozproszone aplikacje przemysłowe na wielu poziomach zarządzania informacją. Ten modułowy system obejmuje zarówno komunikację z urządzeniami produkcyjnymi, warstwy pośrednie zarządzania informacją z produkcji, jak i warstwę integracji z systemami ERP (Czmich, 2007).

Swobodny dostęp do kluczowych wskaźników i raportów, a także drążenie szczegółowych danych oraz analizy możliwe są dzięki wygodnemu w obsłudze portalowi produkcyjnemu Wonderware-Information Server®, który udostępnia informacje z poziomu zwykłej przeglądarki internetowej. Na stronach produkcyjnego portalu Wonderware-Information Server® informacje pochodzące z różnych źródeł prezentowane są w czytelnej formie ekra-

nów synoptycznych, tabel lub wykresów, dając użytkownikowi dostęp online do wszelkich potrzebnych danych (Czmich, 2002).

Uprawnienia i role przypisane dla użytkowników sprawiają, że dostęp mają tylko powołane do tego osoby, a prezentowane raporty i zestawienia są dokładnie takie, jakich potrzebuje dany użytkownik. Użytkownicy mogą także tworzyć na bieżąco własne analizy i raporty bez konieczności angażowania specjalistów z działu IT. Podstawowa konfiguracja portalu nie wymaga od użytkownika znajomości języków programowania ani umiejętności tworzenia stron www. Cechy oprogramowania Wonderware-InTouch®:

- jest częścią pakietu Wonderware-Development Studio® wykorzystującą technologię ArchestrA;
- posiada możliwość zintegrowania z Wonderware-Industrial Application Server®, umożliwiając użytkownikom oprogramowania Wonderware korzystanie z rozwiązań w dziedzinie aplikacji przemysłowych, zachowując przy tym korzyści z dokonanych inwestycji;
- stanowi systemem wizualizacyjnym HMI, który może być instalowany i uruchamiany w środowisku MS-Windows®;
- umożliwia połączenie prawie z każdym sterownikiem i urządzeniem stosowanym w przemyśle;
- może w tej samej chwili być wykorzystywany jako klient OPC, klient i serwer DDE oraz klient i serwer SuiteLink;
- każda aplikacja może bez żadnych ograniczeń komunikować się przez sieć z innymi programami i aplikacjami;
- jest kontenerem obiektów ActiveX oraz kontrolek NET zawartych w nowych obiektach graficznych ArchestrA Graphics, co dodatkowo wzbogaca środowisko projektowe przy projektowaniu rozbudowanych graficznie i funkcjonalnie aplikacji;
- wbudowany system redundancji połączeń z urządzeniami zapewnia wykorzystanie w odpowiedzialnych systemach;
- obsługa kart wielomonitorowych pozwala na projektowanie aplikacji rozbudowanych graficznie, gdzie wymagana jest duża ilość przekazywanych informacji.

Program InTouch może być uruchamiany na komputerach panelowych, umiejscowionych bezpośrednio przy maszynie, komputerze przemysłowym, zainstalowanym w szafie sterowniczej, lub PC zlokalizowanym w pomieszczeniu dla operatorów linii produkcyjnej. Program ten pobiera dane bezpośrednio z systemu sterowania linią produkcyjną, redukując tym samym ryzyko popełnienia błędów ręcznego wprowadzania danych oraz eliminując papierowy obieg dokumentów. Administrator sytemu na ekranie ma dostęp do aktualnego stanu pracy maszyny, otrzymuje natychmiastowe powiadomienia o stanach awaryjnych. Za pomocą ekranów graficznych kontroluje pracę linii produkcyjnej lub pojedynczej maszyny. Dane pomiarowe oraz stany maszyny rejestrowane są automatycznie w przemysłowej bazie danych, umożliwiając dokonywanie szczegółowych analiz przez dział produkcji, technologów, utrzymanie ruchu, dział jakości itp. Każdy operator jest jednoznacznie identyfikowany przez system, co podnosi bezpieczeństwo i pozwala na wykorzystanie danych o czasie pracy na stanowisku. Na ekranie monitora operator może otrzymywać dodatkowe informacje, które pozwalają mu ocenić efektywność w stosunku do założonego celu/normy.

Oprogramowanie InTouch jest stosowane od prostych aplikacji jednostanowiskowych, poprzez rozproszone systemy sieciowe o architekturze serwer/klient, aż po systemy korzy-

stające z możliwości Wonderware System Platform czy też Usług Terminalowych (Terminal Services).

Pakiet InTouch składa się z trzech głównych programów:

– *Application Manager*;
– *WindowMaker*;
– *WindowViewer*.

Funkcje *Application Manager*:

– ma za zadanie zarządzać zaprogramowanymi aplikacjami;
– wykorzystywany jest również do konfigurowania programu WindowViewer jako usługi NT;
– ułatwia konfigurację i tworzenie aplikacji sieciowych NAD (Network Application Dewelopment) zarówno dla architektur klienta, jak i serwera;
– umożliwia konfigurację dynamicznej konwersji rozdzielczości.

Funkcje *WindowMaker*:

– jest środowiskiem edycyjnym, w którym grafika wykorzystywana jest do tworzenia animowanych okien z elementami dotykowymi;
– okna te mogą być połączone z przemysłowymi systemami „wejścia/wyjścia", jak również z innymi aplikacjami MS-Windows.

Funkcje *WindowViewer*:

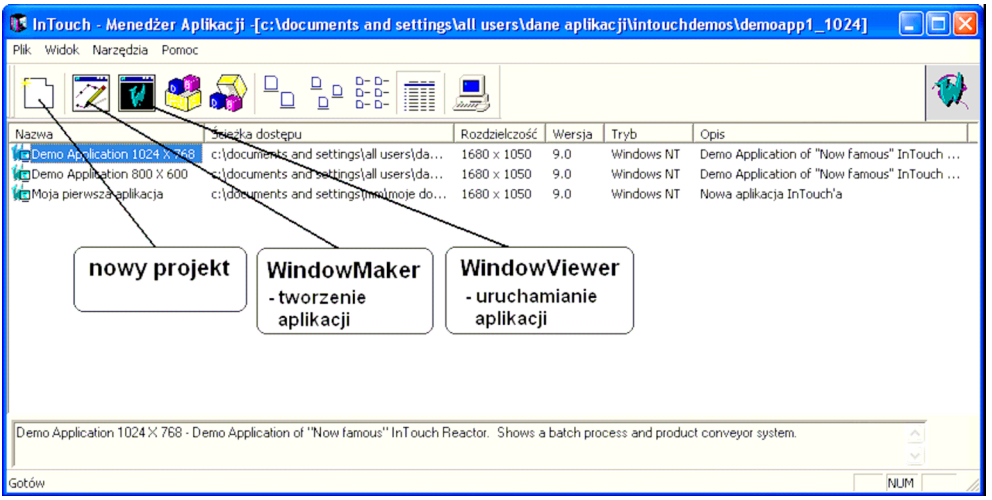
– jest środowiskiem eksploatacyjnym, w którym uruchamiane są aplikacje utworzone wcześniej za pomocą programu WindowMaker;
– wykonuje skrypty;
– sprawdza dane archiwalne;
– przetwarza sygnały o alarmach;
– tworzy sprawozdania.

2.1. Programowanie wizualizacji procesu technologicznego w InTouch®

Proces projektowania i programowania aplikacji realizującej wizualizację linii technologicznej można podzielić na kilka etapów:

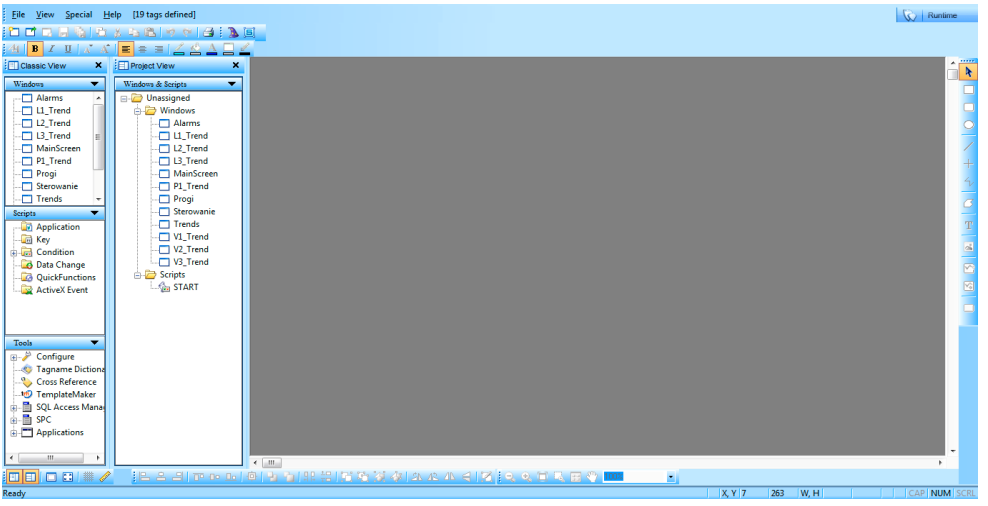
– opracowanie projektu wizualizacji (założenia, schematy, przegląd okien);
– zaprogramowanie okna głównego i okien pochodnych;
– opracowanie listy zmiennych systemowych;
– zaprogramowanie połączeń animacyjnych dla elementów graficznych;
– napisanie skryptów sterujących akcjami w oknach;
– zaprogramowanie systemu alarmowania i raportowania zmiennych;
– zaprogramowanie okien zawierających bieżące i historyczne trendy zmiennych (wykresy);
– organizacja systemu zabezpieczeń dla danej aplikacji,
– kompresowanie i upublicznianie aplikacji .

Po uruchomieniu programu InTouch jako pierwsze otwiera się okno Menedżera Aplikacji przedstawione na rysunku 165. Do głównych zadań Menedżera Aplikacji należą: zakładanie nowego projektu, edycja gotowych aplikacji celem modyfikacji (WindowMaker), uruchamianie gotowych systemów InTouch (WindowViewer).



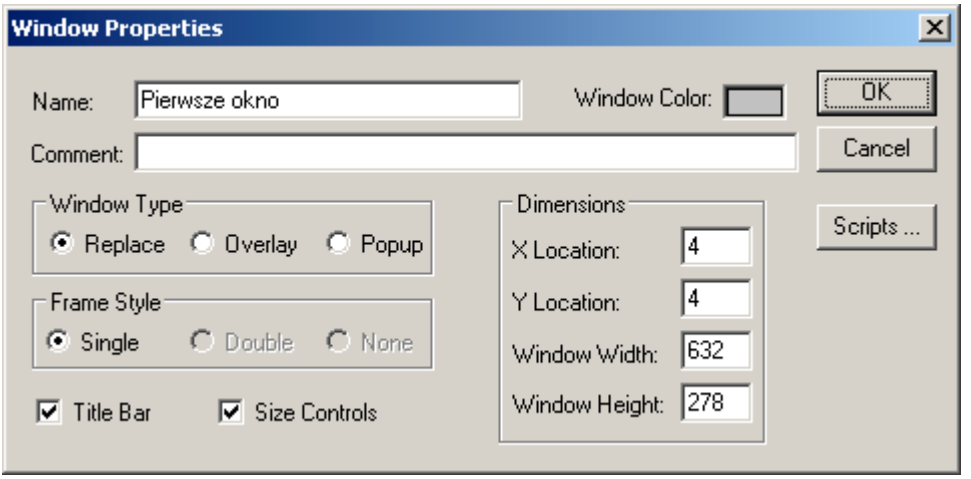
Rysunek 165. Okno Menadżer Aplikacji

W celu utworzenia nowej aplikacji należy wybierać polecenie *Nowy projekt* lub polecenie *Plik → Nowa aplikacja*, po czym na ekranie pojawia się okno, gdzie w kolejnych krokach należy wskazać katalog, w którym aplikacja ma być zapisana, nazwę aplikacji oraz jej opis. Utworzoną nową aplikację, znajdującą się w oknie Menadżera Aplikacji, należy zaznaczyć, po czym kliknąć na ikonę programu WindowMaker. Wygląd głównego okna programu WindowMaker przedstawiono na rysunku 166.



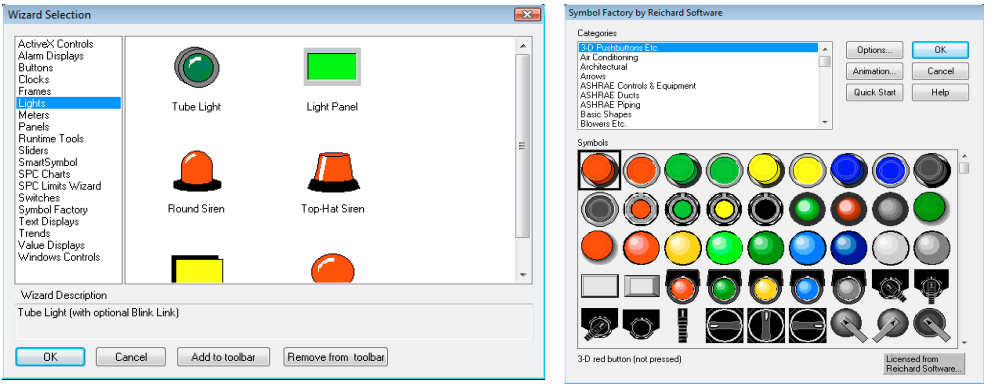
Rysunek 166. Okno główne programu – WindowMaker

Projektowanie rozpoczyna się od utworzenia okien, na których będą znajdować się poszczególne obiekty animacyjne (rys. 167).



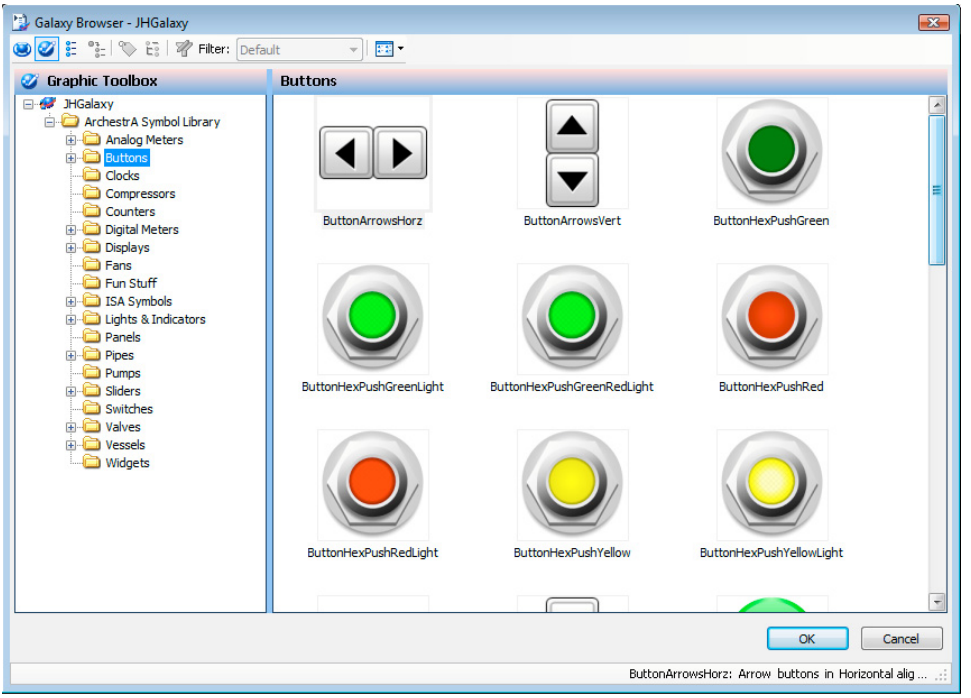
Rysunek 167. Właściwości tworzonego okna

W środowisku projektowym użytkownik znajdzie wiele narzędzi do rysowania obiektów, jak również możliwość importowania plików graficznych z innych programów graficznych (rys. 168).



Rysunek 168. Biblioteka obiektów graficznych Wizards i Symbol Factory

W celu przyspieszenia procesu projektowania graficznej strony aplikacji można korzystać z kontrolek ActiveX, kontrolek NET lub gotowej biblioteki symboli ArchestrA Graphics zawierającej ponad 500 gotowych obiektów graficznych (Kuter, 2011). Obiekty te można edytować, używać do tworzenia szablonów obiektów graficznych (rys. 169).

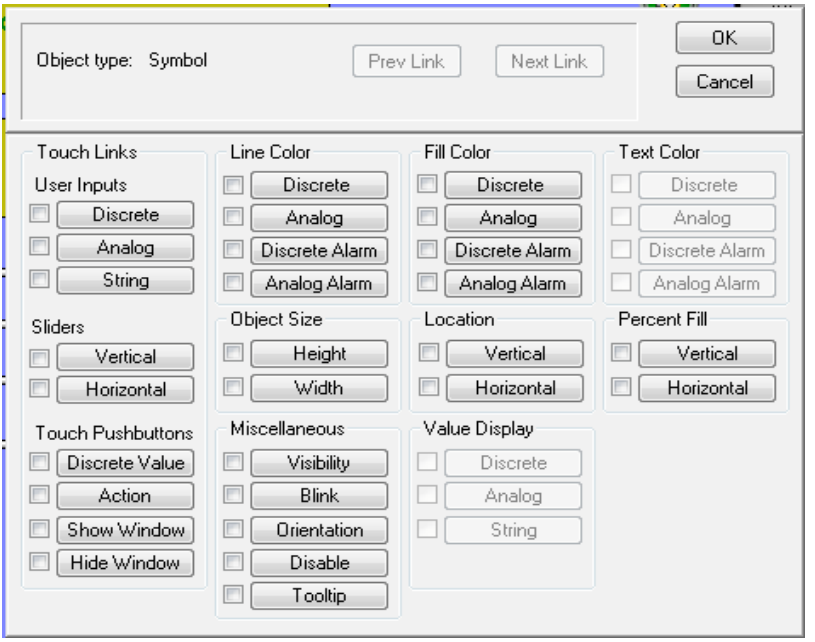


Rysunek 169. Biblioteka symboli Archestra Graphics

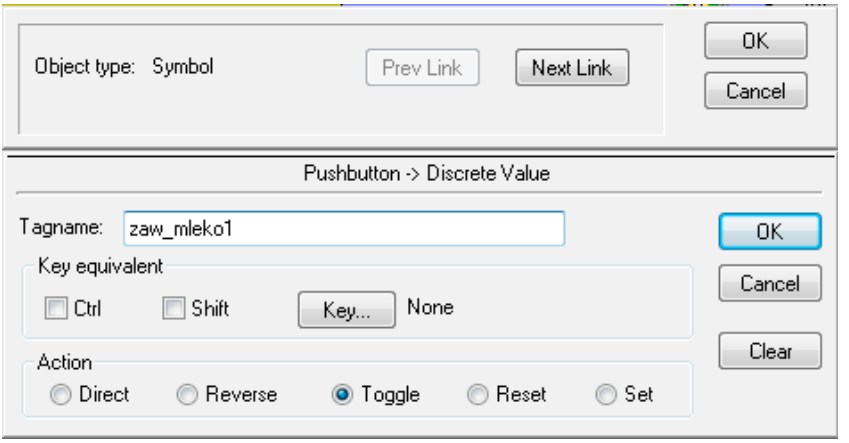
Następnym etapem po zaprojektowaniu wszystkich okien wizualizacji jest nadanie poszczególnym elementom połączeń animacyjnych, które będą odzwierciedlać zmiany wartości zmiennych systemowych. Istnieją dwie drogi realizacji tego etapu: można najpierw prowadzić i zdefiniować wszystkie zmienne procesowe, a później przyporządkować połączenia animacyjne obiektom, lub rozpocząć od przypisywania połączeń animacyjnych i przy każdym system InTouch poprosi o parametryzację zmiennej skojarzonej z tym połączeniem.

Animacje polegają na: zmianie koloru obiektów, przesuwaniu się ich, zmianie stopnia wypełnienia, zmianie wielkości obiektu, widoczności, wyświetlaniu wartości, nadaniu obiektom trybu przełączania zmiennej itp. Okno przedstawiające wszystkie możliwe opcje animacji dla obiektu zamieszczono na rysunku 170.

Każdemu obiektowi można nadać zmienną, która będzie sterowała jego animacją. Na rysunku 171 widoczny jest przykład okna odpowiedzialnego za parametryzację animacji polegającej na zmianie stanu zaworu z zamkniętego na otwarty, po kliknięciu lub przyciśnięciu (ekran dotykowy) tego elementu (*Pushbutton* → *Discrete Value*). Sterowanie odbywa się za pomocą zmiennej dyskretnej (typ Bool) o nazwie zaw_mleko1. Dostępne są różne akcje (działania): *Direct* (przełącznik bezpośredni – rozwierny), *Toggle* (przełącznik), *Set* (ustawianie – przełącznik z pamięcią), *Reset* (przełącznik kasujący zapamiętany stan). Należy zaznaczyć, że jeden obiekt może mieć kilka animacji działających jednocześnie (np. zmiana koloru zaworu z jednoczesną możliwością jego przełączania po kliknięciu).



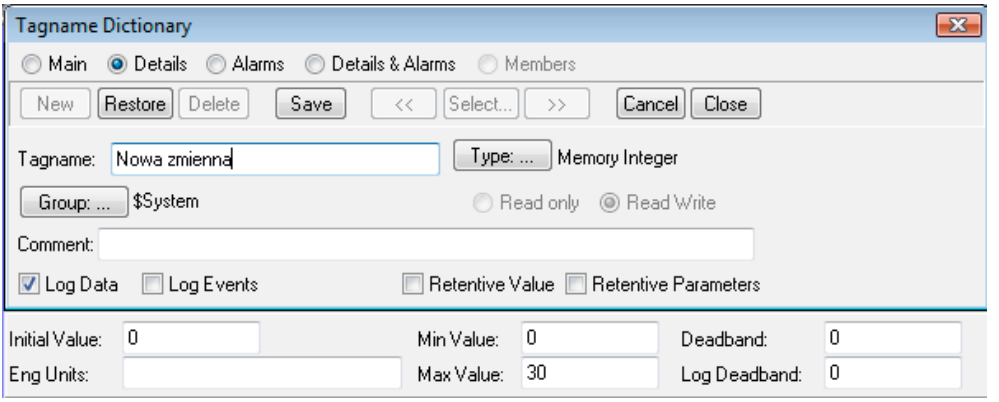
Rysunek 170. Biblioteka połączeń animacyjnych



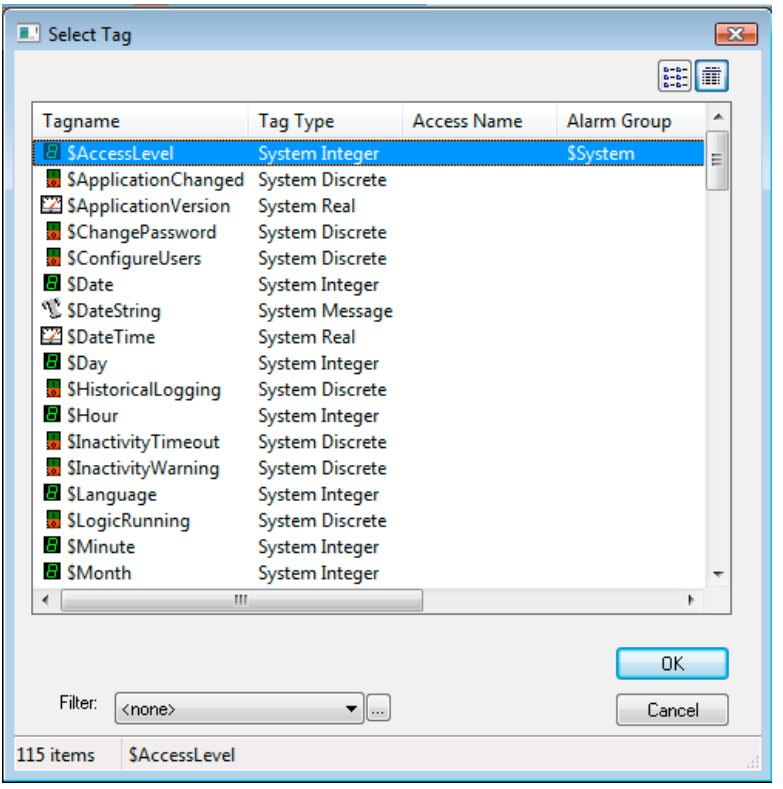
Rysunek 171. Przykładowe okno parametryzacji animacji obiektu

Czynnością obowiązkową przy wprowadzaniu animacji, jak wspomniano wcześniej, jest przypisanie zmiennej, realizuje się to w polu *Tagename*. Dwukrotne kliknięcie w tym polu powoduje wywołanie okna wprowadzania zmiennych, należy wskazać typ zmiennej, określić wartość początkową, zakres zmian wartości, następnie zapisać ją poleceniem Save, w przeciwnym wypadku nie zostanie ona wprowadzona do biblioteki zmiennych (rys.

172). Bibliotekę zmiennych (rys. 173) można otworzyć z menu *Special* → *Tagname Dictionary*.

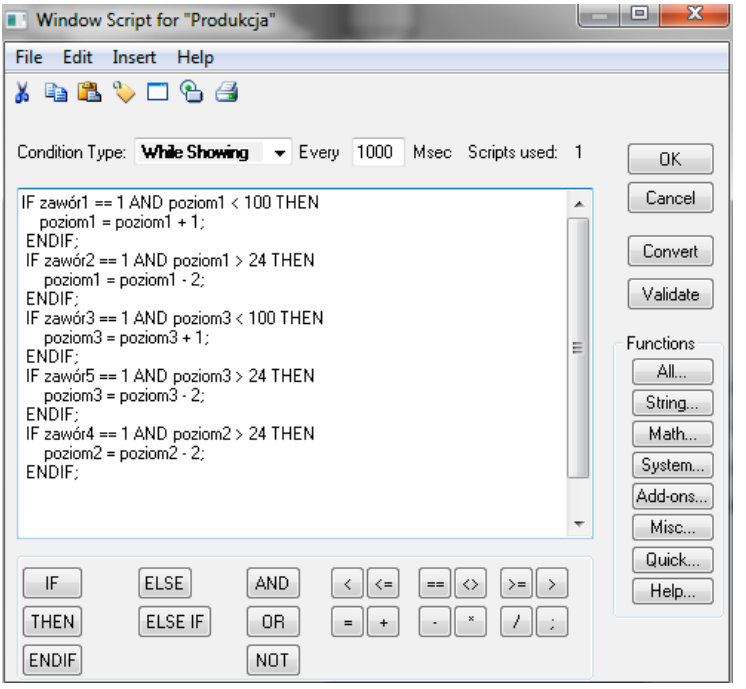


Rysunek 172. Okno wprowadzania zmiennych



Rysunek 173. Biblioteka zmiennych

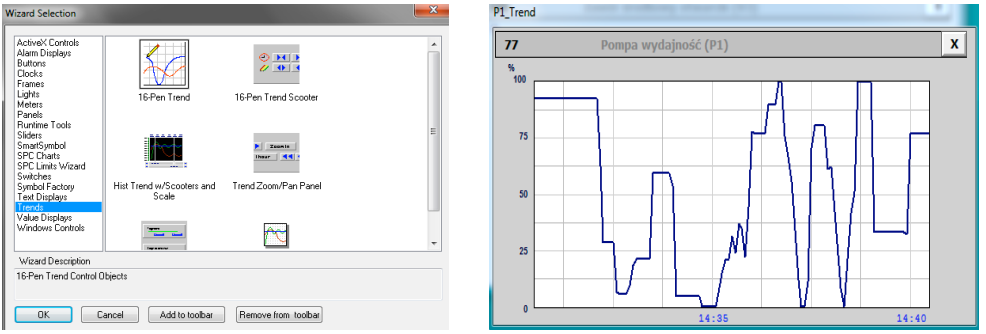
Sterowanie określonymi zdarzeniami w procesie sterowania i wizualizacji umożliwiają skrypty. Metodyka pracy ze skryptami, tj. składnia, sposób wywołania funkcji, jest podobna do sposobu pracy z językami programowania. Do poszczególnych okien można przypisać skrypty, które umożliwiają wykonywanie poleceń oraz operacji logicznych w zależności od zadanych kryteriów, zwiększają funkcjonalność aplikacji oraz lepsze jej dopasowanie do potrzeb użytkownika. Skrypty mogą być uzależnione od warunków specyficznych dla aplikacji, zmian wartości wizualizowanych parametrów, zdarzeń, pojawiających się okien, użycia klawiszy, warunków logicznych i wielu innych zdarzeń. Istnieje możliwość definiowania własnych funkcji – QuickFunctions, które w aplikacji mogą być używane wielokrotnie; upraszcza to proces projektowania aplikacji. Skrypty mogą sterować maszynami i urządzeniami na liniach produkcyjnych, jednak nie jest to zalecane, ponieważ sterownik PLC zrobi to szybciej. Do opracowywania skryptów służy edytor skryptów (rys. 174). Jest łatwy w użyciu, posiada wbudowany mechanizm diagnostyki definiowanych skryptów, czyli sprawdzania poprawności składni skryptów jeszcze przed uruchomieniem aplikacji.



Rysunek 174. Edytor skryptów

InTouch posiada dwa typy obiektów do wyświetlania trendów: jeden z nich przeznaczony jest do wyświetlania trendów bieżących, a drugi do wyświetlania trendów historycznych (dostępne są w bibliotece Wizard). Obiekty te pozwalają tworzyć wykresy zmian wartości zmiennych w czasie (rys. 175). Trend bieżący pozwala rysować wykresy z uży-

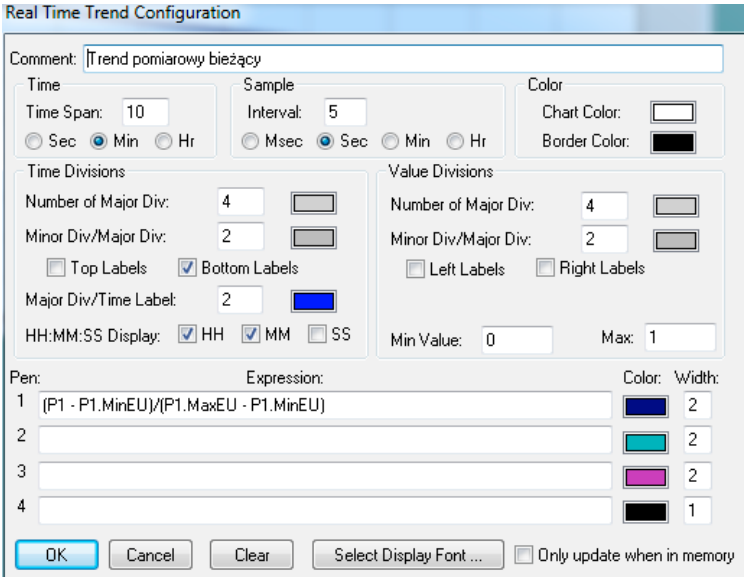
ciem do czterech pisaków (przebiegi czterech zmiennych), natomiast trend historyczny pozwala rysować wykresy z użyciem do ośmiu pisaków (Piotrowski, 2003).



Rysunek 175. Biblioteka z zestawem trendów oraz przykład trendu

Oba typy trendów tworzone są w programie WindowMaker przy użyciu specjalnych narzędzi. InTouch zapewnia też pełną kontrolę nad konfiguracją trendów. Przykładowo można zadawać przedział czasu, zakres wartości, rozdzielczość siatki, położenie oznaczeń na osi czasu, położenie oznaczeń na osi wartości, ilość pisaków, kolory itp.

W celu skonfigurowania zmiennych dla trendu bieżącego należy dwukrotnie kliknąć obiekt trendu w oknie. Wyświetlone zostanie okno dialogowe *Real Time Trend Configuration* (rys. 176).



Rysunek 176. Okno konfiguracji trendu bieżącego

W grupie *Expression* (Wyrażenie) wpisać należy nazwę zmiennej lokalnej lub wyrażenia, zawierającego jedną lub więcej zmiennych lokalnych. Dwukrotne kliknięcie pola *Pen* (Pisak) powoduje wyświetlenie okna dialogowego *Select Tag* (Wybierz zmienną) z listą zmiennych. W celu przypisania zmiennej do pisaka, należy ją wybrać w oknie dialogowym *Select Tag* (Wybierz zmienną). Po prawej stronie ustalić grubość oraz kolor linii trendu.

W grupie *Time* (Czas), w polu *Initial Time Span* (Początkowy przedział czasu) należy wpisać długość przedziału czasu, który ma być pokazywany na osi X trendu. W grupie *Colors* (Kolory) można skonfigurować wygląd trendu bieżącego, kolory obramowania, podziałki czasu, czcionkę itp.

InTouch umożliwia alarmowanie stanu zmiennych. Użytkownicy używają listy alarmów do śledzenia zdarzeń zaistniałych podczas całego procesu produkcji. Program oferuje cztery różne sposoby wglądu w system alarmowy. Każdy z nich może być użyty osobno bądź wszystkie razem.

Alarm jest ostrzeżeniem o wystąpieniu nieprawidłowości w procesie, które mogą prowadzić do zakłóceń, w związku z czym wymagana jest interwencja operatora. Typowy alarm jest uruchamiany, w momencie gdy wartość zmiennej wykroczy poza granice zdefiniowane przez użytkownika, na przykład wartość zmiennej analogowej przekroczy górną wartość graniczną. Zdarzenie to powoduje wygenerowanie alarmu, którego zadaniem jest poinformowanie operatora o nieprawidłowości. Po zatwierdzeniu alarmu przez operatora, interfejs InTouch przyjmuje, że alarm został zatwierdzony. Interfejs InTouch może zostać tak skonfigurowany, aby wymagał zatwierdzenia alarmu nawet wtedy, jeśli stan powodujący alarm został usunięty. Daje to pewność, że operator zostanie poinformowany o zdarzeniu, które spowodowało tymczasowy stan alarmowy, ale stan procesu powrócił do normy.

W celu skonfigurowania alarmu dla zmiennej należy określić typ alarmu oraz zdefiniować wartości progowe. Alarm jest generowany zawsze, jeżeli wartość zmiennej przekroczy zdefiniowaną wartość progową. Wszystkie przypadki wejścia lub wyjścia wartości zmiennej poza zakres alarmowania są zgłaszane do rozproszonego systemu alarmowania (Astor, 2004).

Do wyświetlania alarmów można wykorzystać kontrolki ActiveX:

- *Alarm Database View Control* (prezentuje alarmy historyczne zgromadzone w relacyjnej bazie danych i umożliwia ich łatwe i szybkie filtrowanie);
- *Alarm Viewer Control* (pozwala na pokazywanie alarmów bieżących. Przez elastyczną konfigurację kontrolki można szybko zmienić sposoby prezentacji oraz sortowanie informacji o alarmach);
- *Alarm Pareto View Control* (prezentuje informacje o alarmach w postaci wykresu Pareto, co znacznie ułatwia dostęp do bardzo ważnych informacji, np. o najczęściej pojawiających się alarmach),
- *Alarm TreeView Control* (wyświetla aktywne alarmy w postaci przejrzystej struktury drzewa).

Alarmy mogą być uaktywniane lub dezaktywowane bezpośrednio lub pośrednio przy użyciu zmiennych aplikacji InTouch (*Alarm Inhibitor Tags*). Sposób wyświetlania alarmów może być uzależniony od typu alarmu, zmiennej, grupy alarmowej. Wyświetlanie alarmów może być także uzależnione od nazwy stacji roboczej. Operator ma trzy sposoby potwierdzania alarmów: tradycyjny (warunkowy), zdarzeniowy, kompatybilny z modelem

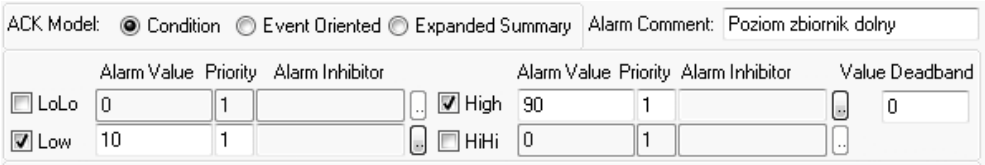
OPC, wymagający potwierdzenia najnowszych zdarzeń, rozszerzony, zezwalający na potwierdzanie każdej transakcji związanej z systemem alarmowym.

Aplikacje InTouch mogą generować alarmy i zdarzenia w celu poinformowania operatorów o stanie procesu:

– alarmy ostrzegają operatorów o sytuacjach, które są potencjalnym źródłem problemów. Zwykle alarm jest generowany, jeżeli jedna ze zmiennych ma wartość większą od skonfigurowanego poziomu granicznego. Alarm jest z reguły zatwierdzany przez operatora;
– zdarzenia są zwykłymi komunikatami o stanie systemu. Typowe zdarzenie jest generowane w momencie wystąpienia określonego warunku w systemie, przykładowo po zalogowaniu się operatora do aplikacji InTouch. Zdarzenia nie muszą być zatwierdzane przez operatora (Astor, 2004).

Program InTouch daje możliwość ustawienia podwójnego progu alarmowania dla wartości niskich i wysokich:

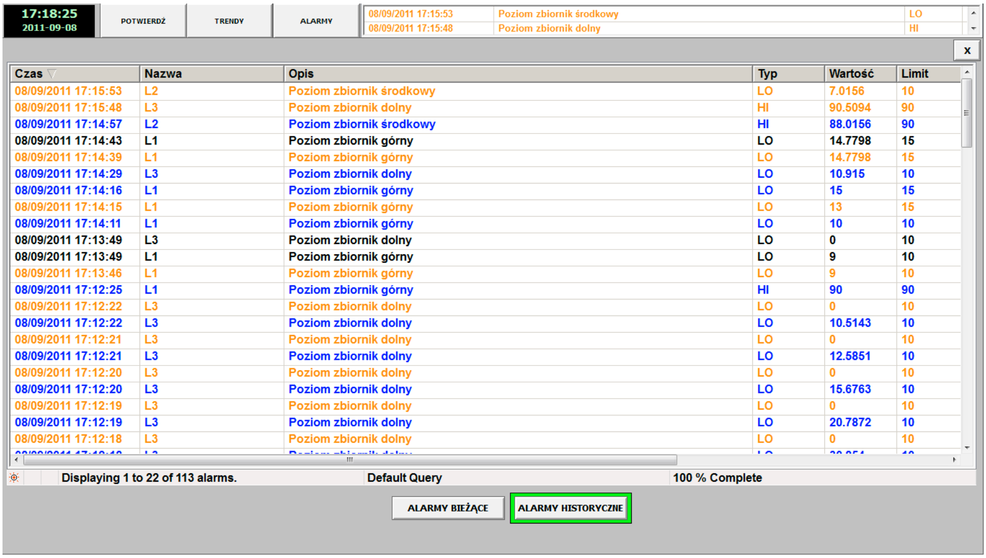
– Low – poziom niski;
– LOLO – poziom bardzo niski;
– High – poziom wysoki;
– HIHI – poziom bardzo wysoki (rys. 177).



Rysunek 177. Okno dialogowe do definiowania alarmów

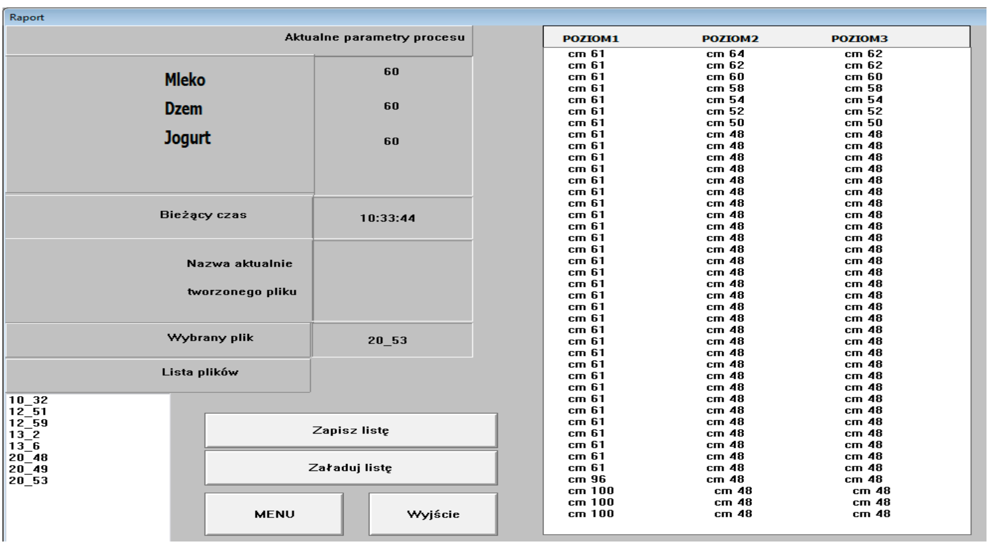
Przykładowe okno prezentujące listę zarejestrowanych alarmów przedstawiono na rysunku 178. Do każdego alarmu przypisywany jest priorytet. Przykładowo przekroczenie dopuszczalnego poziomu napełnienia zbiornika to alarm o dużym priorytecie, wymagający natychmiastowej interwencji operatora. Koniec zmiany to alarm o znacznie mniejszym priorytecie. Priorytet alarmu zwykle zależy od okoliczności – aplikacji, cech urządzeń, bezpieczeństwa, dostępności systemów rezerwowych, potencjalnych kosztów awarii itd. Priorytet alarmu jest określany w czasie definiowania zmiennej. Jest to wartość z zakresu 1 do 999, gdzie 1 oznacza alarm o największej wadze. Priorytety są również użytecznym narzędziem do filtrowania alarmów. Przykładowo można tak skonfigurować wyświetlanie alarmów, aby były pokazywane tylko alarmy o krytycznym znaczeniu. Od priorytetu alarmu można uzależniać tworzenie połączeń animacyjnych, skrypty zatwierdzania, filtrowane wyświetlanie informacji oraz drukowanie.

Wygenerowany w czasie pracy alarm musi być zatwierdzony przez operatora (lub przez system). Zatwierdzenie jest sygnałem, że przyjęto informację o alarmie. Jest to niezależne od podjęcia działań korygujących. Zatwierdzenie nie jest również powiązane z przyczyną alarmów – przyczyna może przestać występować bez zatwierdzenia alarmu. Alarm o wysokim lub średnim priorytecie zazwyczaj wymaga zatwierdzenia, podczas gdy alarm o bardzo niskim priorytecie może nie wymagać zatwierdzenia. Pomimo tego, że stan, który wygenerował alarm, może przeminąć (przykładowo, temperatura przekroczyła dopuszczalną wartość, a następnie opadła) – alarm jest aktywny do momentu zatwierdzenia.



Rysunek 178. Okno alarmów

Aplikacja InTouch wyposażona jest w system raportowania danych pochodzących z procesów technologicznych. Raportowanie polega na zapisywaniu danych do plików. Dane zapisywane są w funkcji czasu. Przykład edytora raportów przedstawiono na rysunku 179. W lewym dolnym rogu widoczna jest lista plików z danymi, po wybraniu jednego z nich automatycznie wyświetlana jest zawartość (okno z prawej strony). Pliki mają format tekstowy, można je łatwo edytować w arkuszach kalkulacyjnych (Astor, 2005).



Rysunek 179. Okno raportów

Od systemów SCADA wymaga się zaawansowanych systemów bezpieczeństwa, chroniących przed niepowołanym dostępem do samej aplikacji, jak i danych procesowych. W oprogramowaniu InTouch możliwe jest stosowanie następujących poziomów dostępu:

– *aplikacyjny poziom dostępu*, użytkownicy posiadają własne hasła, mają nadany poziom dostępu, który decyduje o tym, jakie informacje są dla nich widoczne lub które parametry mogą być przez nich modyfikowane;
– *systemowy poziom dostępu* pozwala przydzielać dostęp na poziomie kontrolera domeny lub lokalnego komputera, bazując na identyfikatorze użytkownika lub grupy użytkowników. Ten zintegrowany system bezpieczeństwa ułatwia zarządzanie prawami dostępu użytkowników i administrację hasłami (Astor, 2008).

  Liczba użytkowników aplikacji, jakich można zdefiniować jest nieograniczona.

3. PRZYKŁADY WIZUALIZACJI W PROCESACH ROLNO-SPOŻYWCZYCH

W rolniczych procesach produkcyjnych komputerowa wizualizacja może wspomagać kontrolę oraz zarządzanie poszczególnymi etapami produkcji. Zapewnia operatorowi obsługującemu dany proces technologiczny dostęp do informacji prezentującej działanie układów automatyki oraz dostarcza wiedzy o sterowaniu pracą poszczególnych elementów wykonawczych. Ponadto zapewnia rejestrację wskazań, prezentację wykresów bieżących i historycznych oraz raportów dokumentujących przebieg procesów, sygnalizację stanów alarmowych, a przede wszystkim – wizualizację procesu w formie czytelnych schematów synoptycznych z opisami i wartościami na ekranie monitora (Juszka i in., 2005a; Tomasik i in., 2009).

W kilku przykładach zostaną przedstawione przykłady rozwiązań systemów wizualizacji opracowanych dla procesów rolno-spożywczych. Wszystkie przykłady zrealizowano w programie Wonderware-InTouch.

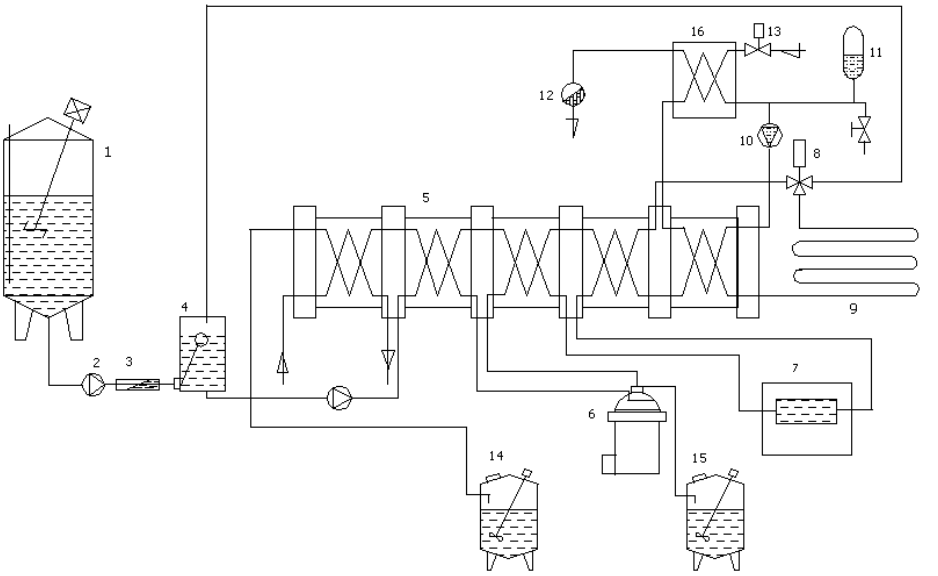
3.1. Wizualizacja pracy pasteryzatora przepływowego

Pierwszym przykładem jest system wizualizacji i nadzoru procesu przepływowej pasteryzacji mleka. Poprawa jakości żywności, zapewnienie bezpieczeństwa konsumentów oraz obniżenie kosztów produkcji, to najważniejsze cele do osiągnięcia w przetwarzaniu i utrwalaniu surowców rolniczych. Aby je osiągnąć, powszechnie stosuje się mikroprocesorowe sterowanie procesami cieplnego utrwalania żywności. Nowoczesne systemy pasteryzacji pozwalają uzyskać mleko o wartości odżywczej równej, a w przypadku modyfikacji jego składu – nawet większej od wartości mleka surowego. Zaprogramowana aplikacja InTouch może nadzorować wielkości związane z poprawnym przebiegiem procesu, w przypadku przekroczenia wartości krytycznych, uruchomi alarm i odpowiednie procedury uniemożliwiające wyprodukowanie wadliwego produktu. System automatycznego sterowania linią do pasteryzacji mleka w połączeniu z wizualizacją umożliwia kontrolę procesu technologicznego i archiwizację danych procesowych.

Schemat technologiczny linii pasteryzacji mleka przedstawiono na rysunku 180, głównym urządzeniem jest przepływowy pasteryzator, stanowiący wymiennik ciepła (5). Pozostałe urządzenia to:

1 – tank mleka surowego;
2 – pompa podająca;
3 – filtr siatkowy;
4 – zbiornik regulacyjny;
5 – wymiennik płytowy;
6 – wirówka;

7 – homogenizator;
8 – zawór trójdrożny;
9 – przytrzymywacz;
10 – pompa obiegu wody;
11 – naczynie wyrównawcze;
12 – odwadniacz;
13 – zawór regulacyjny;
14 – tank mleka odtłuszczonego;
15 – tank śmietanki;
16 – wymiennik para-woda (Juszka i Tomasik, 2005).

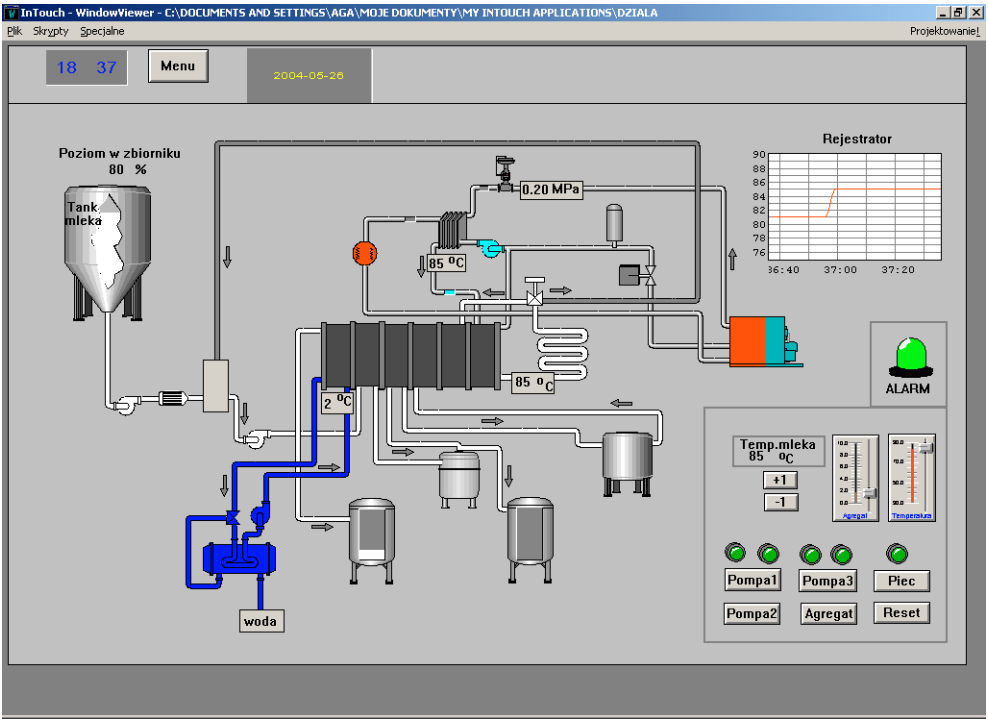


Rysunek 180. Schemat linii pasteryzacji mleka z homogenizacją i odwirowaniem

Po uruchomieniu pomp mleko transportowane jest rurociągiem do pasteryzatora poprzez zbiornik regulacyjny. W pasteryzatorze realizowane jest dodatkowo odwirowanie śmietany i homogenizacja. Następnie odbywa się właściwa pasteryzacja. Jeżeli parametry nie są spełnione, następuje skierowanie poprzez zawór obiegu mleka do powtórnej pasteryzacji. Po przeprowadzeniu procesu pasteryzacji trafia do sekcji wymiany (oddając ciepło, ogrzewa mleko wpływające do pasteryzatora), schłodzone trafia do zbiornika końcowego. Linia pasteryzacji pracuje w cyklu ciągłym. Prezentowany algorytm został wykorzystany do zaprogramowaniu symulacji działania opracowanej aplikacji wizualizacji pracy pasteryzatora. Systemem pasteryzacji steruje sterownik PLC.

Okno sytemu wizualizacji przedstawia rysunek 181. Napełnianie zbiornika mleka surowego z cysterny uwidoczniono w programie poprzez obiekt ilustrujący wypełnienie

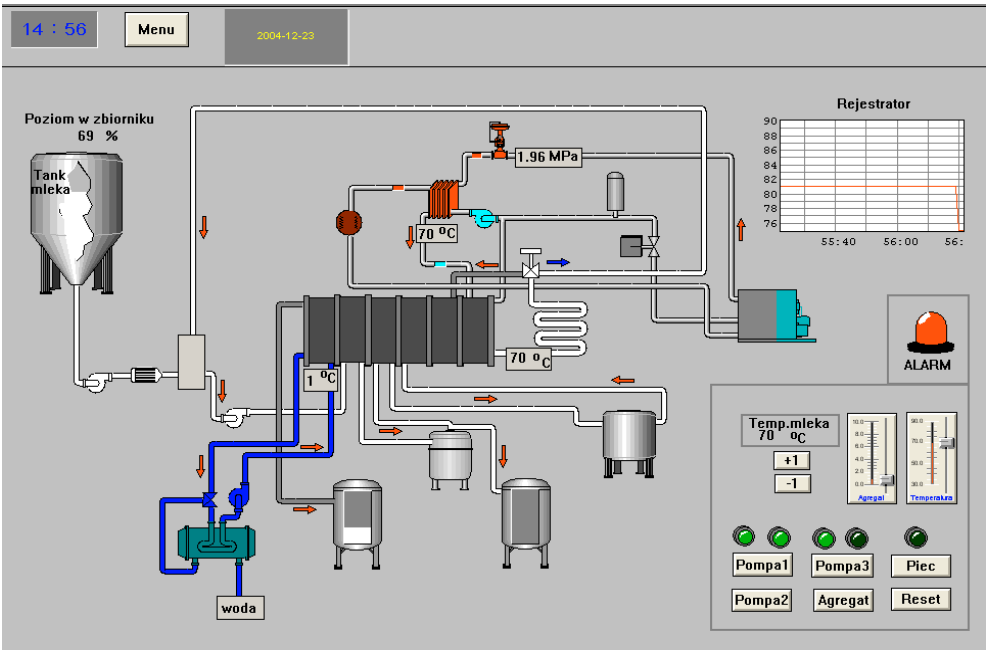
zbiornika (Tank mleka). Następnie za pomocą pomp podających jest transportowane poprzez filtr do zbiornika regulacyjnego, a stamtąd przepompowane do pasteryzatora. Przepływ surowca został zobrazowany poprzez zmianę atrybutu koloru statycznych elementów linii. Program kontroluje temperaturę procesu pasteryzacji i na bieżąco wyświetla aktualną temperaturę na ekranie. W przypadku spadku temperatury poniżej 80^{o}C program automatycznie uruchamia zawór obiegowy mleka, powodujący ponowne skierowanie mleka do wymiennika płytowego, ponadto włącza się kontrolka Alarm (kolor czerwony) sygnalizująca zbyt niską temperaturę. Równocześnie załączany jest czynnik grzewczy, co również jest przedstawione na ekranie za pomocą zmiany kolorów.



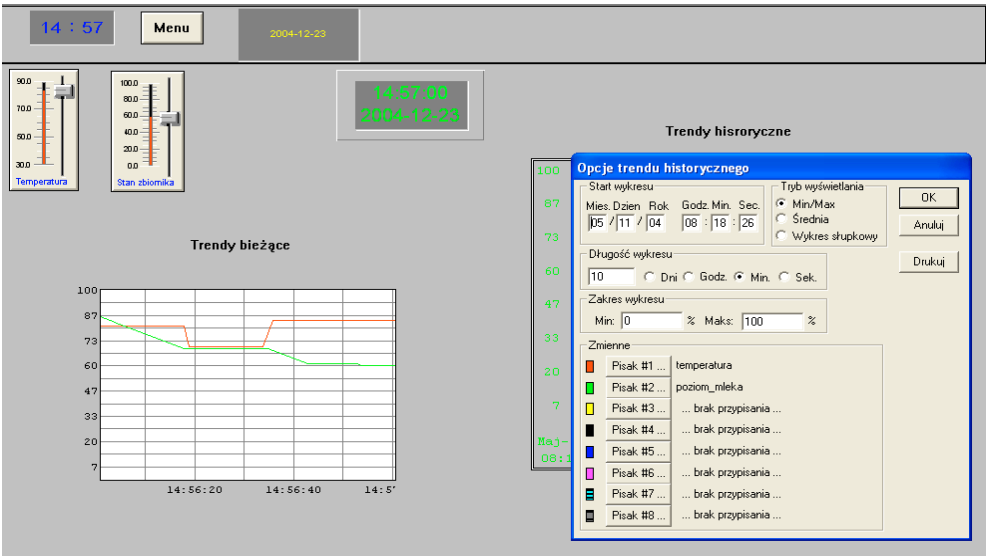
Rysunek 181. Wizualizacja procesu pasteryzacji mleka

Na rysunku 182 zamieszczono ekran wizualizacji przy uruchomionym alarmie. Włączenie sygnalizacji alarmowej wywołała za niska temperatura pasteryzacji. Mleko, które przepływało przez pasteryzator i uzyskiwało temperaturę niższą od wymaganej, powtórnie zostało skierowane do pasteryzacji.

Podczas pracy program rejestruje temperaturę i przechowuje ją w bazie danych. Wybranie w menu programu okna „Trendy" umożliwia odczytanie temperatury z dowolnego okresu czasu. Odczytu danych archiwalnych (rys. 183) (m.in. zarejestrowanej temperatury i poziomu w zbiorniku) można dokonać z poziomu okna „Trendy" po wejściu w zakładkę „Menu".



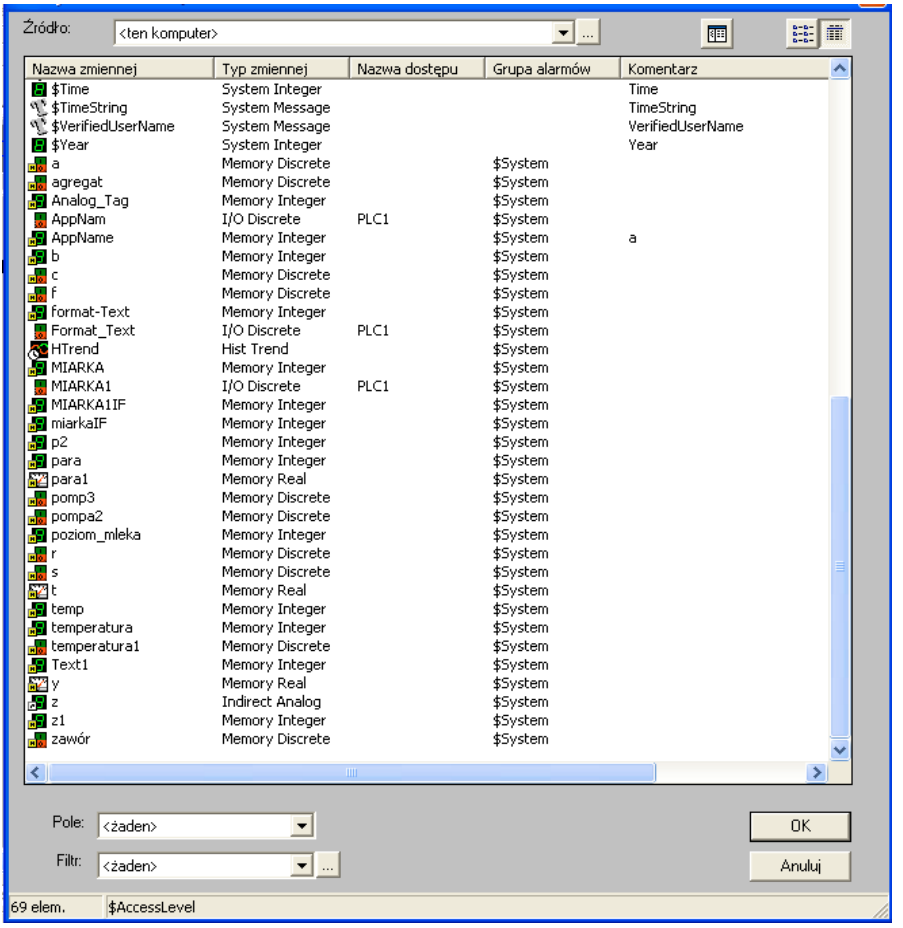
Rysunek 182. Wizualizacja procesu pasteryzacji mleka – alarm



Rysunek 183. Odczyt danych bieżących i wywoływanie danych archiwalnych

Program InTouch umożliwia wykorzystanie dwóch podstawowych typów zmiennych:
– własnych, inaczej pamięciowych (w oryginale nazywanych Memory tags);
– globalnych, wymiennych z innymi aplikacjami oraz sterownikiem PLC (I/O tags).

Program InTouch umożliwia zdefiniowanie zmiennych, których wartości mogą być na bieżąco wykorzystywane przez inne aplikacje zainstalowane pod systemem operacyjnym za pośrednictwem protokołów komunikacyjnych takich jak DDE lub OPC. Wartości zmiennych tego typu wykorzystywanych przez inne aplikacje, są uaktualniane automatycznie bezpośrednio po zmianie wartości zmiennej u źródła. W celu umożliwienia wymiany danych z programem komunikacyjnym zmienna musi być typu globalnego (I/O). Zastosowane zmienne w aplikacji przestawiającej pasteryzację mleka wraz z opisem zamieszczono na rysunku 184.



Rysunek 184. Zmienne zastosowane w wizualizacji pasteryzatora

3.2. System wizualizacji stanowiska dydaktycznego MultiTank

Stanowisko MultiTank jest przeznaczone do badania algorytmów sterowania w obiektach dynamicznych na przykładzie regulacji poziomu cieczy w zbiornikach, umożliwia praktyczne zaznajomienie się z podstawowymi algorytmami regulacji tj.:

- regulacja dwupołożeniowa;
- PID;
- regulator rozmyty (fuzzy logic controller);
- regulator LQR (ang. Linear Quadratic Regulator) – optymalno-kwadratowy.
- pętla otwarta (open loop);
- oraz umożliwia konstrukcję nowych algorytmów, które można sprawdzać praktycznie w sterowaniu przepływem cieczy lub poziomem cieczy w zbiornikach (Juszka i in., 2008).

Sterowanie odbywa się za pomocą karty procesorowej RT-DAC4/PCI, zainstalowanej w komputerze, dla której kod generowany jest w programie Matlab-Simulink. Stanowisko to zostało przedstawione na rysunku 185, składa się ono z:

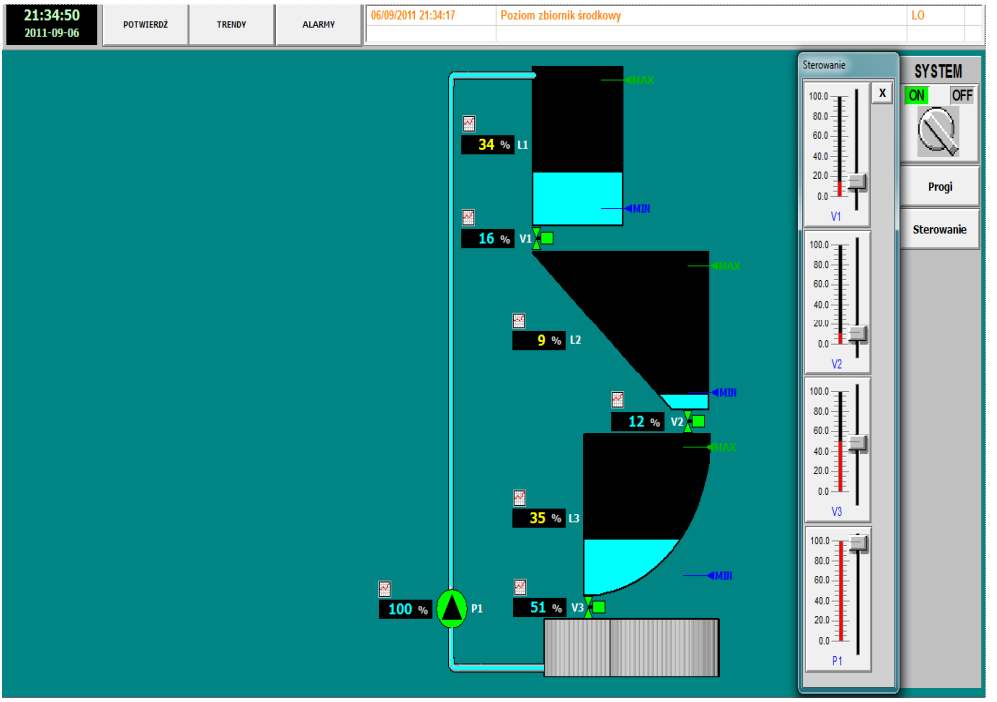
- trzech zbiorników ustawionych w formie kaskady, każdy o innym kształcie i innej pojemności oraz zbiornika zasilającego;
- trzech zaworów zamontowanych pod każdym ze zbiorników z regulowanym analogowo stopniem otwarcia zaworu;
- czujników poziomu w każdym zbiorniku, gdzie poziom mierzony jest w oparciu o pomiar ciśnienia hydraulicznego;
- pompy, która pompuje wodę do zbiornika znajdującego się najwyżej; również wydajność pompy może być regulowana;
- karty procesorowej RT-DAC/PCI, zamontowanej w komputerze, modułu zasilacza i przetworników oraz wyłącznika awaryjnego.

Główne okno główne aplikacji dla systemu MultiTank przedstawiono na rysunku 186. Każda zmienna komunikuje się z systemem Matlab-Simulink protokołem DDE w czasie rzeczywistym, tzn. zachodzące zmiany w systemie sterowania zbiorników są rejestrowane w aplikacji SCADA bez żadnych opóźnień. W głównym oknie programu odwzorowano układ trzech zbiorników.

Każdy ze zbiorników posiada taki kształt, jak obiekt doświadczalny. Zmiana poziomu napełnienia zbiornika cieczą reprezentowana jest kolorem niebieskim. Przy każdym ze zbiorników umieszczony jest wskaźnik pokazujący, w jakim stopniu zbiornik jest napełniony. Używając funkcji Percent Fill, uzyskano efekt proporcjonalnego do rzeczywistego obiektu wypełnienia płaskiej figury geometrycznej, stanowiącej uproszczony

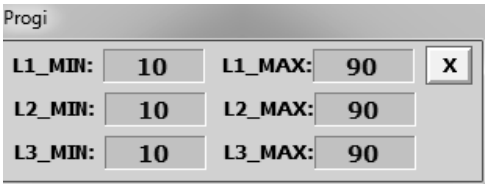
Rysunek 185. Stanowisko MultiTank

model zbiornika. W okienku przy każdym zbiorniku wyświetlana jest wartość poziomu lub procent wypełnienia, w zależności od ustawienia opcji. Przy pompie również znajduje się wskaźnik określający wydajność, z jaką ona pracuje. Na zbiornikach zaznaczone są zdefiniowane minimalne oraz maksymalne progi napełnienia zbiornika. Wartości tych progów wprowadzono jako parametry zmiennych, przy których uruchamiane są alarmy informujące o nieprawidłowościach pracy. W prawym górnym rogu znajduje się wyłącznik systemu, który umożliwia uruchomienie sterowania.



Rysunek 186. Wizualizacja systemu MultiTank

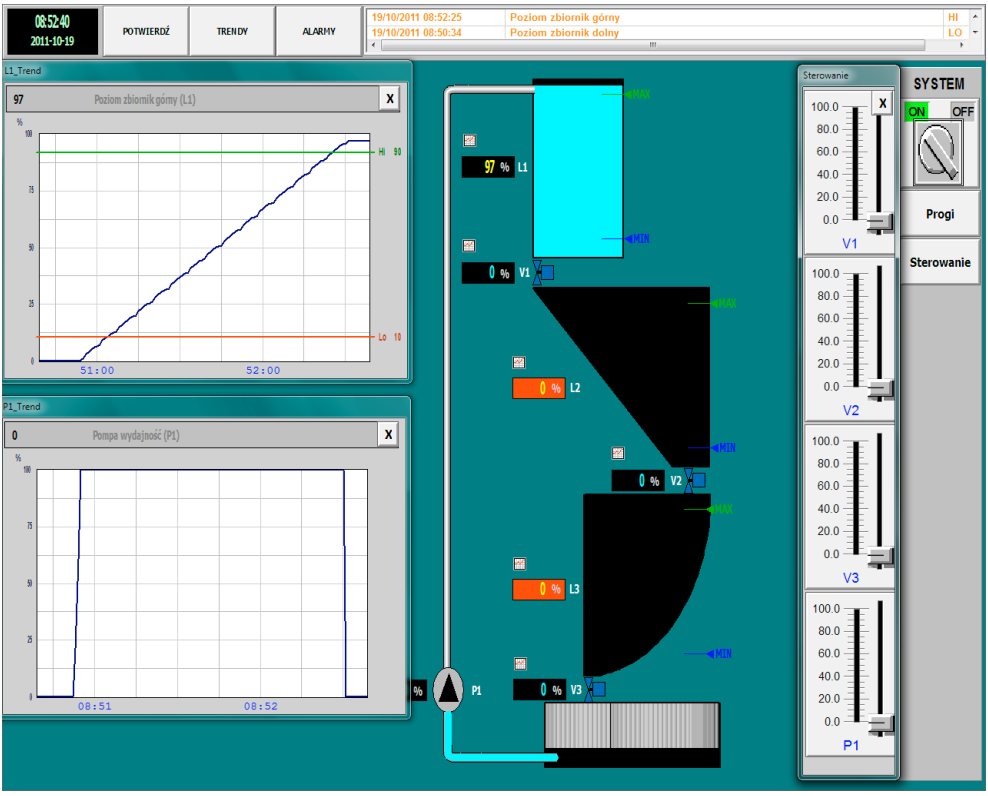
Poniżej umiejscowiony jest przycisk umożliwiający wyświetlenie zadanych progów napełnienia zbiorników oraz przycisk wyświetlający okno sterowania zaworami oraz pompą. Sterowanie ręczne odbywa się przy wykorzystaniu suwaków. Każdy element jest sterowany analogowo w zakresie 0 – 1, przez co możliwa jest regulacja stopnia otwarcia zaworu oraz wydajności pompy. Sterowanie ręczne jest możliwe tylko wtedy, gdy system nie pracuje w trybie automatycznej regulacji, przełącznik pracy w pozycji OFF. Gdy zawór jest zamknięty, to na schemacie jego kolor jest zielony, natomiast jeśli za pomocą suwaka nastąpi otwarcie zaworu o co najmniej 1%, wtedy jego kolor zmieni się na niebieski. W górnej części okna umieszczono przyciski uruchamiające okna trendów bieżących, a więc charakterystyki dynamiczne sygnałów oraz okno trendów archiwalnych. W oknie trendów archiwalnych studenci mogą generować dane na podstawie historycznych wartości sygnałów. Ponadto znajduje się okno alarmów informujące o za niskich lub za wysokich



Rysunek 187. Okno progów alarmowych

wartościach poziomu. W oknie tym możliwe jest dodawanie kolejnych alarmów, każdy alarm powinien być potwierdzony. Na rysunku 187 przedstawiono okno progów alarmowych, możliwa jest edycja wartości w zakresie od 0 do 100% napełnienia zbiornika.

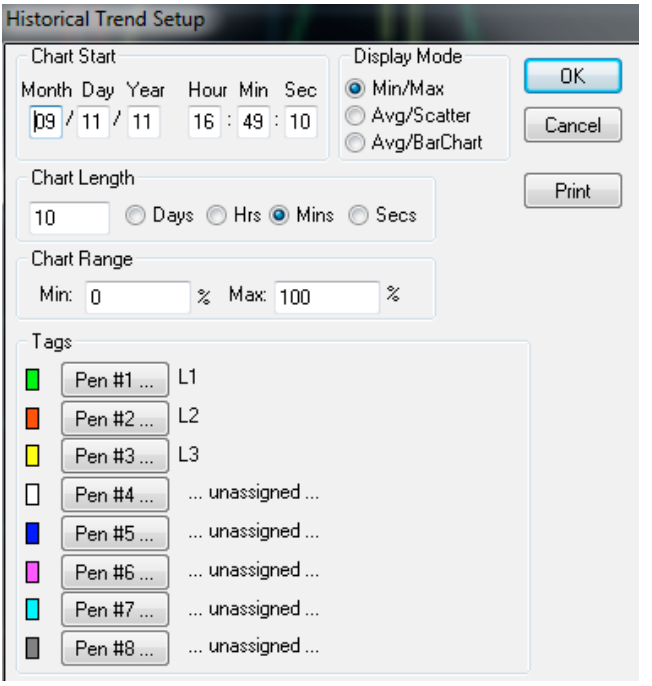
Okno zarejestrowanych alarmów podczas działania badań testowych aplikacji MultiTank przedstawione zostało wcześniej na rysunku 178. Natomiast rysunek 188 przedstawia wizualizację w trakcie działania, widoczne są niepotwierdzone dwa alarmy dla górnego zbiornika. W oknie trendu widoczna jest liniowa charakterystyka napełniania zbiornika górnego. W drugim oknie trendu zarejestrowano sygnał sterujący pompą. Na wykresie można zaobserwować, że w pewnym momencie poziom w zbiorniku górnym się zatrzymał, związane jest to z awaryjnym wyłączeniem pompy przez system sterujący stanowiskiem. Wyłącza on pompę i zamyka wszystkie zawory przy osiągnięciu maksymalnego dopuszczalnego poziomu w którymkolwiek ze zbiorników, tak aby nie doszło do przelania cieczy.



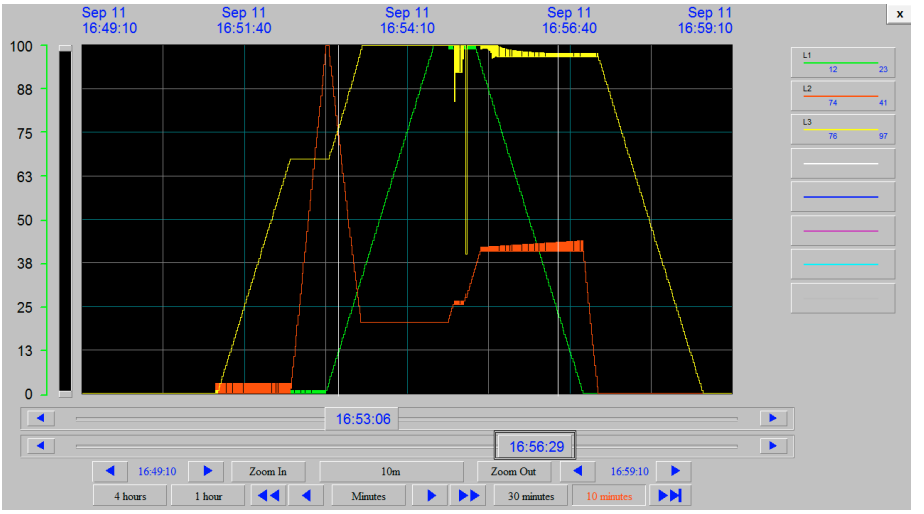
Rysunek 188. Napełnianie zbiornika górnego maksymalną wydajnością pompy

Trendy historyczne prezentują segment danych z przeszłości. W odróżnieniu od trendów bieżących, trendy historyczne mogą być aktualizowane wyłącznie przez skrypt lub działanie zrealizowane przez operatora. Trend historyczny pokazuje graficzną reprezentację danych z przeszłości w formie graficznej dla maksymalnie ośmiu zmiennych. Dane wyświetlane przez trend historyczny są przypisane do pisaków trendu. Kliknięcie przycisku *TRENDY* wyświetla okno dialogowe *Historical Trend Setup* (Konfiguracja wyświetlania trendu historycznego) (rys. 189). W oknie tym można skonfigurować kolory pisaków przypisanych do odpowiednich zmiennych po kliknięciu przycisku Pens oraz przedział czasu wyświetlanego w oknie wykresu trendu historycznego. Okno przedstawiające wybrane trendy historyczne dla trzech zbiorników zarejestrowane w trakcie prowadzenia prac doświadczalnych zamieszczono na rysunku 190. W dolnej części okna dostępny jest panel umożliwiający precyzyjne ustawienie przedziału czasowego analizowanych trendów. Za pomocą suwaków można sprawdzić wartości napełnienia zbiorników dowolnego dnia i czasu. Niezwykle istotna funkcja dla studentów wykonujących ćwiczenia laboratoryjne, którzy utracili dane do sprawozdania przechowywane na nośnikach zewnętrznych.

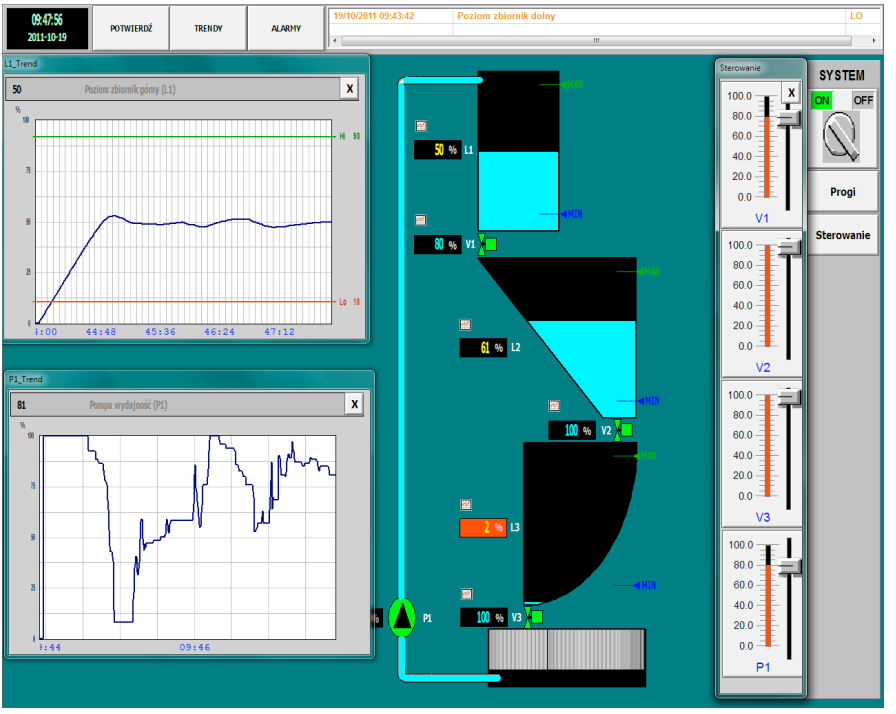
Przykład współdziałania dydaktycznego systemu zbiorników z programem Matlab-Simulink zamieszczono na rysunku 191. Widoczna jest realizacja regulacji poziomu cieczy z zastosowaniem algorytmu LQR (ang. Linear Quadratic Regulator) – optymalno-kwadratowy.



Rysunek 189. Konfiguracja wyświetlania trendów historycznych



Rysunek 190. Trendy historyczne



Rysunek 191. Wizualizacja pracy układu przy sterowaniu LQR poziomem cieczy w zbiorniku górnym

3.3. Wizualizacja systemu sterowania dojem maszynowym krów

W typowo towarowej produkcji mleka zastosowanie znajdują głównie mechaniczne urządzenia udojowe. Parametry ich pracy w zakresie podciśnienia występującego w komorze podstrzykowej kubka udojowego, charakterystyki i częstotliwości zmian pulsacji, masażu strzyka oraz prawidłowego funkcjonowania poszczególnych urządzeń stacji udojowej powinny być stale kontrolowane i monitorowane, ponieważ wpływają na jakość mleka i zdrowie krów. Stąd opracowano aplikację monitorującą i sterującą aparatem udojowym w procesie doju krów (Tomasik i in., 2010). System monitoringu i sterowania został zaprogramowany na platformie ArchestrA® w programie InTouch, oferuje:

- sterowanie (podstawowe funkcje resztę zrealizowano na sterowniku PLC) i wizualizację pracy aparatu udojowego;
- system raportowania i alarmowania stanów procesu doju krów;
- rejestrację danych procesowych w postaci bazy danych.

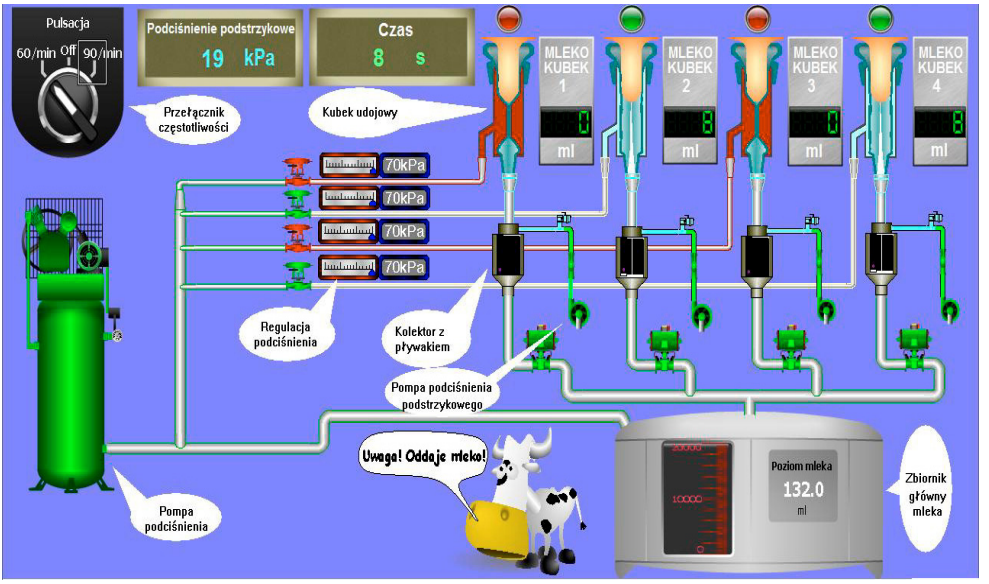
W zaprezentowanej aplikacji przedstawiono wizualizację pracy aparatu udojowego, sterowanego za pomocą sterownika PLC, szczegółowy opis tego systemu sterownia przedstawiono w trzeciej części niniejszej książki w rozdziale 1.2.3. System wizualizacji odpowiedzialny jest za kontrolowanie, monitorowanie, rejestrowanie oraz archiwizowanie parametrów pracy aparatu udojowego. Ponadto istnieje możliwość ręcznego zadawania sygnałów sterujących. Aplikacja umożliwia wymianę danych z pozostałymi użytkownikami programu poprzez sieć, jak również daje możliwość sterowania danym procesem przez urządzenia, takie jak np. sterownik PLC (Juszka i in., 2012a).

W nadrzędnym oknie prezentowanej aplikacji programu InTouch umieszczono następujące elementy graficzne (rys. 192):

- pompa podciśnieniowa;
- przewody podciśnieniowe;
- rurociąg mleczny;
- cztery niezależnie sterowane kubki udojowe ze strzykami;
- kolektory z wyświetlaczami poziomu;
- pompy podciśnienia podstrzykowego;
- suwaki ustalające odpowiednią wartość podciśnienia w kubku udojowym;
- przełącznik sterujący częstotliwością zmiany podciśnienia;
- zawory podciśnienia w kubku udojowym;
- główny zbiornik mleka;
- przyciski służące do ręcznego włączania, wyłączania systemu oraz wywoływania odpowiednich okien;
- lampki kontrolne.

Ponadto w aplikacji utworzono takie okna jak: animacja, menu, parametry, trendy mleka, trendy podciśnienia, alarmy.

Na pasku *Menu* umiejscowionym poniżej wizualizacji ulokowano przyciski typu Button, które umożliwiają przechodzenie pomiędzy oknami (rys. 193). Wszystkie przyciski wywołują okna z ich nazwy, natomiast Zakończenie – zamyka program InTouch.



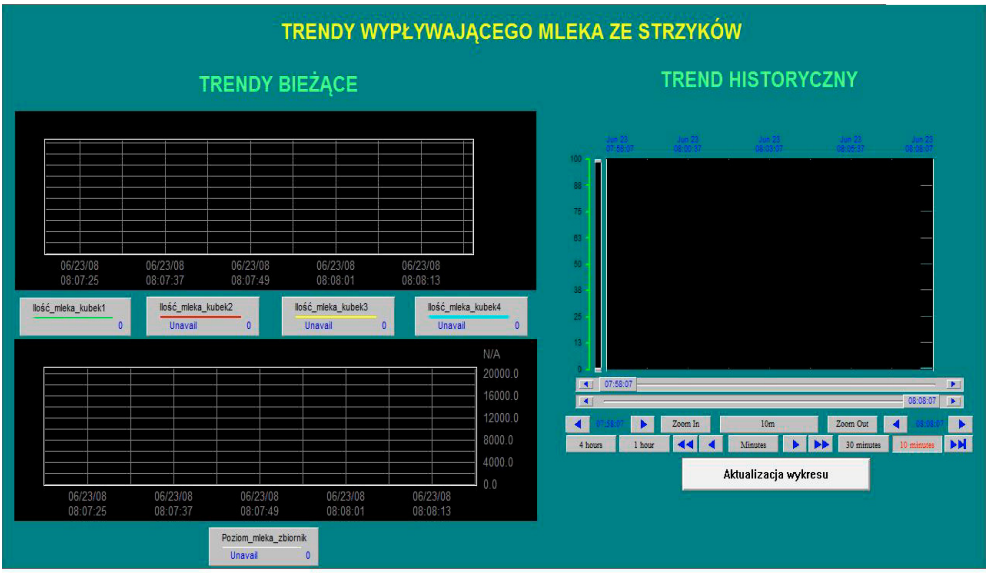
Rysunek 192. Wizualizacja sterowania aparatem udojowym

Rysunek 193. Pasek Menu

Nad wizualizacją znajduje się pasek *Parametry* posiadający przycisk włączania systemu udojowego, kontrolkę stanu przycisku, dynamiczny obiekt tekstowy oraz zegar z datownikiem (rys. 194). Przycisk włączania ma za zadanie uruchomić wszystkie urządzenia oraz otworzyć zawory. Kontrolka stanu świecąc na zielono, informuje o włączeniu; na czerwono – sterowanie wyłączone. Dynamiczny obiekt tekstowy w czasie działania procesu ma informować komunikatami *FAZA DOJU ROZPOCZĘTA* o rozpoczęciu wypływu mleka ze strzyków, a komunikatem *FAZA DOJU ZAKOŃCZONA* – o zaprzestaniu wypływu mleka.

Rysunek 194. Okno Parametry

Okno *Trendy mleka* wyposażono w obiekty przedstawiające wykresy przebiegów zmiennych (rys. 195). Zmienne reprezentowane w trendach to: ilość mleka wypływająca z poszczególnych strzyków oraz ilość mleka w zbiorniku głównym. W oknie przedstawiono dwa typy trendów: bieżące i historyczne.



Rysunek 195. Trendy bieżące i historyczne dla wypływu mleka ze strzyków

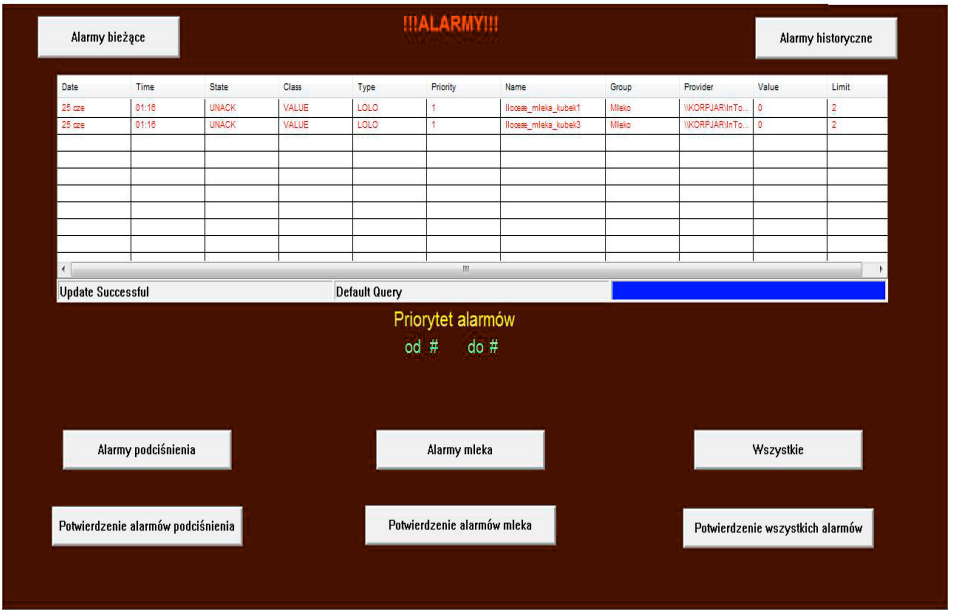
Trendy bieżące przedstawiają wykresy zmian wartości zmiennej w czasie rzeczywistym, co zapewnia bieżącą kontrolę przebiegu procesu. Trendy historyczne prezentują wykresy zmian wartości zmiennej w pewnym odstępie czasu. Pozwala to na sprawdzanie przebiegu procesu po jego zakończeniu. Analogicznie opracowano okno *Trendy podciśnienia*, które zawiera trendy bieżące i historyczne takich zmiennych jak: pulsacja oraz podciśnienie panujące w komorze podstrzykowej każdego kubka udojowego.

W oknie *Alarmy* umieszczono Wizard, który ma za zadanie prezentować alarmujące wartości zmiennych: podciśnienia w komorze podstrzykowej, podciśnienia w rurociągu mlecznym, ilości wypływającego mleka z danego kubka, poziomu mleka w zbiorniku głównym (rys. 196).

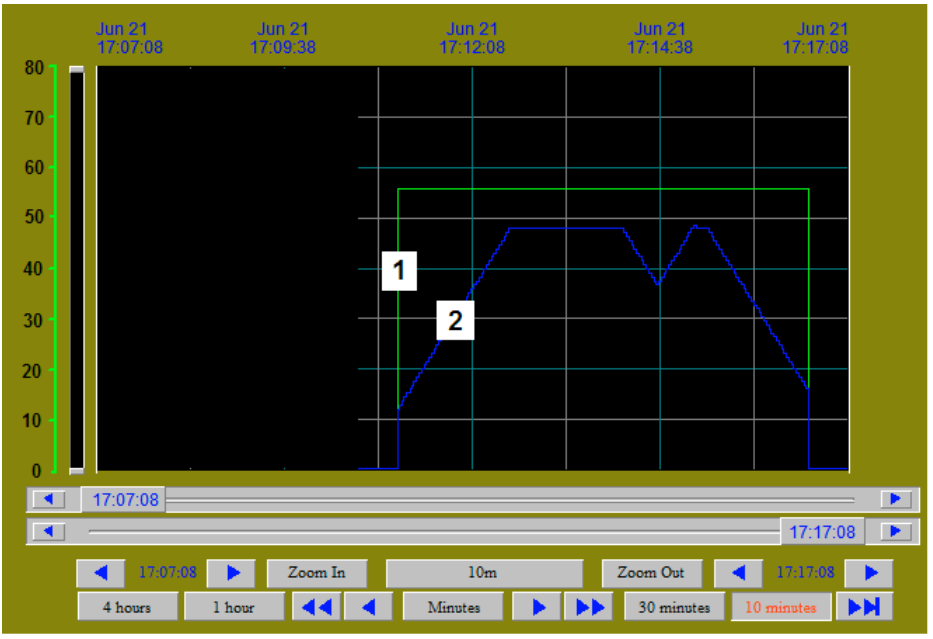
Następnym krokiem po zaprogramowaniu wszystkich obiektów było nadanie poszczególnym elementom połączeń animacyjnych, które będą odzwierciedlać zmiany wartości zmiennych. Konieczne również okazało się zaprogramowanie skryptów sterujących zdarzeniami. Skryptami posłużono się do zadawania i wyliczenia konkretnych parametrów sterowania zgodnie z rzeczywistym procesem doju krów.

Wyniki działania systemu sterowania i wizualizacji aparatu udojowego przedstawiono w obiektach typu *trend historyczny* na rysunkach 197 i 198.

Zilustrowano testy systemu sterowania wykonane na doświadczalnym stanowisku udojowym. Badano działanie aparatu udojowego dla krów. Używano modelu funkcjonującego jako sztuczny strzyk połączony ze zbiornikiem cieczy mlekozastępczej (Juszka i in., 2005b). Pierwszy z wykresów zawiera charakterystykę zmian podciśnienia podstrzykowego (2) oraz podciśnienia w rurociągu mlecznym (1). Ze zmianami podciśnienia wiąże się zmiana natężenia strumienia masowego wypływu mleka ze strzyków wymienia krowy (1), a także zmiana poziomu mleka w zbiorniku kolektora aparatu udojowego (2) (rys. 198).

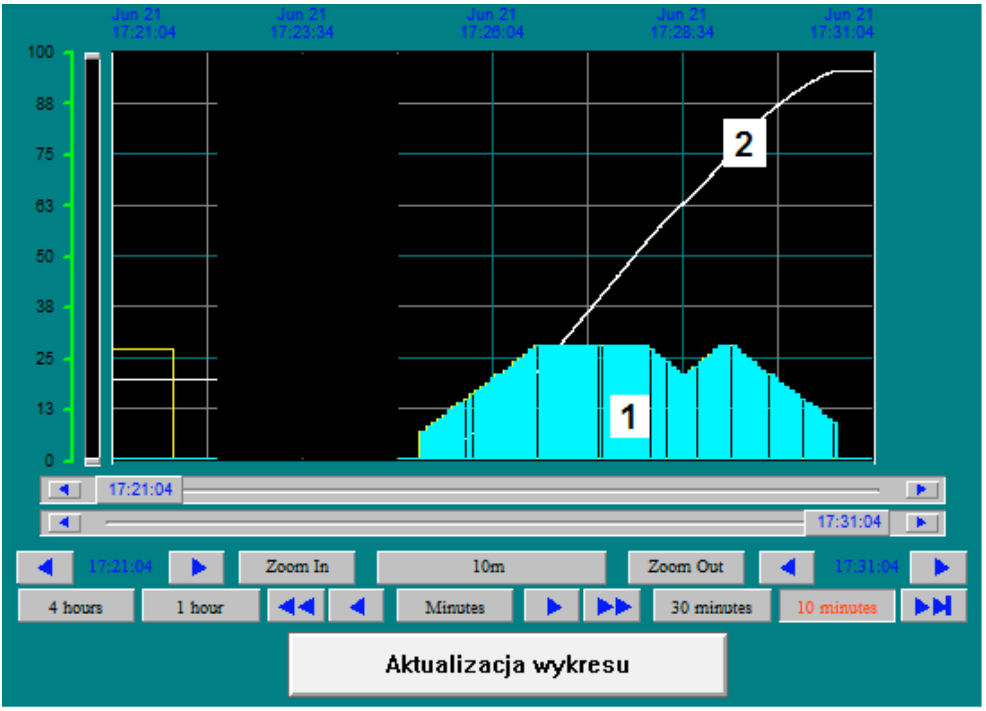


Rysunek 196. Okno alarmów dotyczących pracy aparatu udojowego



Rysunek 197. Wizualizacja przebiegu zmian podciśnienia

Widoczna skala zakresu wartości umieszczona z lewej strony dotyczy tylko procentowego wypełnienia kolektora, aby otrzymać skalę dla chwilowego masowego natężenia wypływu mleka ze strzyka wymienia krowy, należy wybrać tę zmienną do aktualizacji.



Rysunek 198. Wizualizacja przebiegu zmian wypływu mleka ze strzyka wymienia i procentowego wypełnienia mlekiem kolektora

LITERATURA

Astor. (2011). *Wonderware InTouch – Charakterystyka produktu*. Pozyskano z: http://www.astor.com.pl/wonderware/produkty/wonderware-intouch.html?start=1.

Astor. (2009). *Wizualizacja i raportowanie.Systemy Scada. Wonderware – InTouch.* Pozyskano z: http://www.astor.com.pl/produkty/wizualizacja-i-raportowanie/systemy-scada/wonderware-intouch.html.

Astor. (2010). *Technologia ArchestrA*. Pozyskano z: http://www.astor.com.pl/wonderware /technologia-archestra/informacje-ogolne.html.

Astor. (2004). *Alarmy i zdarzenia*. Podręcznik użytkownika. Materiały firmy.

Astor. (2005). *Wonderware InTouch.* Podręcznik użytkownika. Materiały firmy.

Astor. (2006). *Systemy informatyczne dla przemysłu.* Materiały firmy.

Astor. (2008). *Zarządzanie danymi.* Podręcznik użytkownika. Materiały firmy.

Bailey, D. (2003). Practical SCADA for industry. Londyn, Wyd. Elsevier, ISBN 0750658053.

Bauerfeind, D. (2002). Podręcznik użytkownika EASY 800. Bonn, Moeller GmbH.

Bednarek, M. (2001). Wizualizacja procesów: laboratorium. Rzeszów, Oficyna Wyd. Politechniki Rzeszowskiej, 12-15.

Boroń, W.; Wyłupek, J. (1996). PROFIBUS – standardy otwartych sieci przemysłowych. *Pomiary Automatyka Kontrola. ISS w Katowicach, 11*, 313-320.

Broel-Plater, B. (2000). *Sterowniki programowalne, właściwości i zasady stosowania*. Szczecin, Seria Tempus, ISBN 83-874233-4-3.

Brzózka, J. (2004). *Regulatory i układy automatyki*. Warszawa, MIKOM, ISBN 83-7279-380-8.

Czmich, W. (2002). SuiteVoyager 2.0 - bezpieczny przemysłowy portal internetowy. Biuletyn Automatyki, 34(4). Pozyskano z: http://www.astor.com.pl/page/index.php/biuletyn-automatyki /archiwum/rocznik2002indexhtml/754.html.

Czmich, W. (2007). Platforma Systemowa Wonderware 3.0. Biuletyn Automatyki, 54(4). Pozyskano z: http://www.astor.com.pl/page/downloads/biuletyn_automatyki/BA54.pdf.

Dorobczyński, L. (1991). *Cyfrowa symulacja elementów i układów automatycznej regulacji*. Szczecin, Wyd. Wyższej Szkoły Morskiej w Szczecinie, 44-47.

Fijałka, G. (2007). InTouch 10.0. Łatwiej, szybciej, ładniej. *Biuletyn Automatyki, 54*(4). Pozyskano z: http://www.astor.com.pl/page/downloads/biuletyn_automatyki/BA54.pdf.

Flaga, S. (2010). *Programowanie sterowników PLC w języku drabinkowym*. Legionowo, Wydawnictwo BTC, ISBN 978-83-60233-56-6.

Garbacki, A. (2005). Wonderware ArchestrA – architektura do tworzenia zintegrowanych systemów przemysłowych. Biuletyn Automatyki, 46(4). Pozyskano z: http://www.astor.com.pl/page /index.php/biuletyn-automatyki/archiwum/rocznik-2005/nr-46-42005-/504.html.

Grzywak, A. (1994). *Praca zbiorowa: Rozproszone systemy komputerowe*. Gliwice. Wyd. Promet, 31-35.

Jabłoński, W. (1998). *Automatyka i sterowanie*. Bydgoszcz, Wyd. Uczelniane Akademii Techniczno-Rolniczej w Bydgoszczy, 18-22.

Jakuszewski, R. (2006). *Programowanie systemów SCADA: Proficy HMI/SCADA - iFIX/.* Gliwice, Wyd. Pracowni Komputerowej Jacka Skalmierskiego, 38-44.

Juszka, H. (2006). *Automatyzacja i robotyzacja w inżynierii rolniczej*. Kraków, PTIR, ISBN 83-917053-3-1.

Juszka, H.; Lis, S.; Tomasik, M. (2011a). Koncepcja dwukomorowego kolektora autonomicznego aparatu udojowego. *Inżynieria Rolnicza, 8*(133), 161-165.
Juszka, H.; Tomasik, M. (2005). Sterowanie procesem pakowania produktów pochodzenia rolniczego. Acta Scientiarum Polonorum, *Technica Agraria, 4*(1), 69-75.
Juszka, H.; Tomasik, M. (2005). Wizualizacja procesu przepływowej pasteryzacji mleka. Acta Scientiarum Polonorum, *Technica Agraria 4*(1), 77-83.
Juszka, H.; Tomasik, M. (2006). Automatyzacja odcinka technologicznego na bazie logiki rozmytej. Acta Scientiarum Polonorum, *Technica Agraria 5*(1), 61-71.
Juszka, H.; Tomasik, M. (2009). Mikrosterowniki w procesach hodowlanych. *Inżynieria Rolnicza, 5*(114), 101-107.
Juszka, H.; Tomasik, M; Lis, S. (2010). Mikroprocesorowy system sterowania pulsacją doju maszynowego krów. *Inżynieria Rolnicza, 1*(119), 243-249.
Juszka, H.; Tomasik, M.; Bochnia, B. (2008a). Modelowanie mikroprocesorowego sterowania temperaturą i wilgotnością w oborze. *Problemy Inżynierii Rolniczej, 4*(62), 103-109.
Juszka, H.; Tomasik, M.; Lis, S. (2005a). Wizualizacja komputerowa narzędziem wspomagającym modelowanie procesów rolniczych. *Inżynieria Rolnicza, 10*(70), 133-141.
Juszka, H.; Tomasik, M.; Lis, S. (2005b). Wizualizacja komputerowa w prezentacji wyników modelowania doju krów. *Problemy Inżynierii Rolniczej, 4*(50), 65-70.
Juszka, H.; Tomasik, M.; Lis, S. (2007). Sterowanie mikroprocesorowe procesem zadawania paszy dla zwierząt. *Inżynieria Rolnicza, 7*(95), 53-60.
Juszka, H.; Tomasik, M.; Lis, S. (2012a). Dynamiczna wymiana danych DDE w sterowaniu aparatem udojowym. *Problemy Inżynierii Rolniczej, 2*(76), 119-125.
Juszka, H.; Tomasik, M.; Lis, S. (2012b). Pulsacja podciśnienia sterowana PLC w doju maszynowym krów. *Problemy Inżynierii Rolniczej, 2*(76), 111-118.
Juszka, H.; Tomasik, M.; Lis, S.(2008b). Modelowanie i symulacja funkcjonowania regulacji na stanowisku Multi Tank. *Inżynieria Rolnicza, 10*(108), 97-104.
Juszka, H.; Tomasik, M.; Lis, S.; Haczyk, G. (2011b). Sterowanie logiczne z regulacją PID podciśnieniem w aparacie udojowym. *Problemy Inżynierii Rolniczej, 1*(71), 77-86.
Kacprzak, S. (2011). *Programowanie sterowników PLC zgodnie z normą IEC 61131-3 w praktyce*. Legionowo, BTC, ISBN 978-83-60233-81-8.
Kalista, C. (2012). Ethernet w zastosowaniach przemysłowych. *Pomiary Automatyka Robotyka, 11*, 11-18.
Kamiński, K. (2009). *Podstawy sterowania z PLC*. Warszawa, Gryf, ISBN 97883-923756-0-9.
Kasprzyk, J. (2011). *Programowanie sterowników przemysłowych*. Warszawa, WNT, ISBN 978-83-204-3109-4.
Kościelny, J. (1998). Systemy nadzorowania i wizualizacji procesów przemysłowych – wymagania, kryteria oceny. *Pomiary Automatyka Kontrola, 3*, 69-76.
Krzesiński, W. (2005). Zasady sterowania klimatem z wykorzystaniem komputerów. Hasło Ogrodnicze, 4, 3-8.
Kurpaska, S. (2007). *Szklarnie i tunele. Inżynieria i procesy*. Warszawa, PWRiL, ISBN 978-83-09-01024-1.
Kuter, S. (2011). SCADA Wonderware InTouch - Opis. Pozyskano z: http://www.toffic.pl /scadarealizacje/71-intouch.html?showall=1.
Kwaśniewski, J. (1999). *Programowalne sterowniki przemysłowe w systemach sterowania*. Kraków, J. Kwaśniewski & Fundacja Dobrej Książki, ISBN 83-86320-45-1.
Kwaśniewski, J. (2008). *Sterowniki PLC w praktyce inżynierskiej*. Legionowo, BTC, ISBN 978-83-60233-35-1.
Legierski, T.; Wyrwał, J.; Kasprzyk, J.; Hajda, J. (1998). Progra*mowanie sterowników PLC. Gliwice*. Wyd. Pracowni Komputerowej Jacka Skalmierskiego, ISBN 838664416-8.

Lewandowska, M. (2007). Aplikacje HMI/SCADA – analiza rynku. Raporty branżowe. Pozyskano z: http://hmi-scada.raport.xtech.pl/artykul.aspx?id=HMI_SCADA_raport&pg=1.
Łukasik, Z.; Seta Z. (2001). *Programowalne sterowniki PLC w systemach sterowania przemysłowego*. Radom, Wyd. Politechniki Radomskiej, ISBN 83-88001-18-3.
Masi, C.G. (2008). *Tworzenie bloków funkcyjnych*. Control Engineering Polska. Artykuł pod redakcją Łukasza Urbańskiego, Wydział Elektryczny Politechniki Szczecińskiej. Pozyskano z: http://www.controlengineering.pl/menu-gorne/artykul/article/tworzenie-schematow-blokowych/.
Masłowski, A.; Garbacki, A.; Woźniczka, M. (2008). Wonderware InTouch i Platforma Systemowa. O czym warto wiedzieć. *Biuletyn Automatyki, 55*(1). Pozyskano z: http://www.astor.com.pl /page/downloads/biuletyn_automatyki/BA55.pdf.
Michta, E. (1998). Wizualizacja obiektów i procesów technologicznych. *Pomiary Automatyka Robotyka, 2*, 9-16.
Mielczarek, W. (1994). *Szeregowe interfejsy cyfrowe*. Gliwice, Wyd. Helion, ISBN 83-85701-23-0.
Mikulczyński, T. (2006). *Automatyzacja procesów produkcyjnych*. Warszawa, WNT, ISBN 83-204317-7-8.
Mikulczyński, T.; Samsonowicz, Z. (1997). *Automatyzacja dyskretnych procesów produkcyjnych*. Warszawa, WNT, ISBN 83-204221-1-6.
Miłosiewicz, M. (1997). Interfejsy dialogu człowiek-maszyna. *Pomiary Automatyka Robotyka, 4*, 19-25.
Miozga, A. (2002). Hierarchiczna budowa współczesnych systemów sterowania i wizualizacji - część I. Biuletyn Automatyki, 33(3). Pozyskano z: http://www.astor.com.pl/page/index.php/biuletyn-automatyki/archiwum/rocznik/-2002/nr-33-32002-/767.html.
Moeller. (2008). Easy i MFD-Titan w praktyce. Pozyskano z: http://www.moeller.pl /Programs/TBI/easy_mfd%20w%20praktyce.pdf.
Moeller. (2012). Easy, XV100 i MFD-Titan w praktyce. Pozyskano z: http://biuro-inzynierskie.com/easy%20w%20praktyce1.pdf.
Moeller. (2006). Instrukcja obsługi easy800. Pozyskano z: http://www.moeller.pl /Documentation/AWB/podr_uzytk_easy800_pl.pdf.
Moeller. (2009). Łączenie, sterowanie, wizualizacja. Poradnik układów elektrycznych, nr 2, Pozyskano z: http://www.moeller.pl/Documentation/Literatura/pfach_EASY.pdf.
Novak, R. (2007). Systemy SCADA podbijają branżę wod-kan. Pozyskano z: http://www.automatyka.pl/newsItem.aspx?pk=4366
Nowakowski, W. (2006). *LOGO! w praktyce*. Warszawa, BTC, ISBN: 83-60233-12-8.
Pawełczyk, W.; Wąsik, W.; Świstek, M.; Szulc, K.; Jabłoński, L. (2009). Systemy SCADA w sterowaniu procesami technologicznymi. Pozyskano z: http://www.utrzymanieruchu.pl/menu-gorne/artykul/article/systemy-scada-w-sterowaniu-procesami-technologicznymi/.
Peszyński, K.; Siemieniako, F. (2002). *Sterowanie procesów: podstawy i przykłady*. Bydgoszcz. Wyd. Uczelniane Akademii Techniczno-Rolniczej w Bydgoszczy, 36-42.
Piątek, Z. (2011). *Wiosna na rynku oprogramowania przemysłowego. Raport z polskiej branży SCADA/HMI*. Pozyskano z: http://automatykab2b.pl/raporty/3828-wiosna-na-rynku-oprogramowania-przemyslowego -raport-z-polskiej-branzy-scadahmi?showall=1.
Piotrowski. (2003). Trendy bieżące i historyczne. Pozyskano z: http://moby.piwko.pl /pliki/piotrowski/pio9.htm.
PN-EN 61131-1:2004E. Sterowniki programowalne Część1: Postanowienia ogólne.
PN-EN 61131-2:2008E. Sterowniki programowalne Część2: Wymagania i badania dotyczące sprzętu.
PN-EN 61131-3:2004E. Sterowniki programowalne Część3: Języki programowania.
IEC/TR 61131-4:2004. Programmable controllers – Part 4: User guidelines
PN-EN 61131-5:2002E. Sterowniki programowalne Część5: Komunikacja.
IEC 61131-6:2012. Programmable controllers – Part 6: Functional safety.
PN-EN 61131-7:2004E. Sterowniki programowalne Część7: Programowanie rozmyte.

IEC/TR 61131-8:2003. Programmable controllers – Part 8: Guidelines for the application and implementation of programming languages.
Ruda, A.; Olesiński, R. (2003). *Sterowniki programowalne PLC*. Warszawa, COOSiW SEP, ISBN 83-89008-17-3.
Sałat, R.; Korpysz, K.; Obstawski, P. (2010). *Wstęp do programowania sterowników PLC*. Wyd. WKŁ, Warszawa, ISBN 978-83-206-1754-2
Seta, Z. (2002). *Wprowadzenie do zagadnień sterowania: wykorzystanie programowalnych sterowników logicznych PLC*. Warszawa, Wyd. Mikom, ISBN 8372792550.
Systemy Scada. Producenci systemów SCADA – zestawienie i porównanie. (2009). Pozyskano z: http://systemyscada.pl/systemy-scada.html.
Ścibisz, M. (2009). Wykorzystanie sterowników PLC jako elementów kontrolnych w liniach technologicznych w przemyśle rolno-spożywczym. *Inżynieria Rolnicza, 8*(117), 231-236.
Tomasik, M.; Juszka, H.; Lis, S. (2010). Rozwój systemów wizualizacji w automatyzacji doju krów. *Inżynieria Rolnicza, 4*(122), 253-259.
Tomasik, M.; Juszka, H.; Hojna J. (2009). Sterowanie procesami rolniczymi wspomaganymi przez systemy informatyczne. *Inżynieria Rolnicza, 5*(114), 281-288.
Tomasik, M.; Juszka, H.; Lis, S. (2011). Blok funkcyjny „Fuzzy Logic" w sterowaniu PLC autonomicznym aparatem udojowym. *Inżynieria Rolnicza, 8*(133), 277-283.
Tomasik, M.; Juszka, H.; Lis, S. (2012a). Analiza porównawcza metod pomiaru przepływu mleka w kolumnie autonomicznego aparatu udojowego. *Inżynieria Rolnicza, 2*(136), 335-344.
Tomasik, M.; Juszka, H.; Lis, S. (2012b). Efficiency analysis of an automatic water treatment system. *Inżynieria Rolnicza, 4*(140), 165-173.
Tomasik, M.; Juszka, H.; Lis, S. (2012c). System komunikacji układu kontrolno-pomiarowego w autonomicznym aparacie udojowym. *Inżynieria Rolnicza, 2*(137), 303-310.
Tomasik, M.; Juszka, H.; Lis, S. (2012d). Zastosowanie logiki rozmytej do sterowania temperaturą wody na fermie bydła. Opracowania monograficzne pod redakcją naukową prof. dra hab. inż. Wacława Romaniuka. Problemy intensyfikacji produkcji zwierzęcej z uwzględnieniem struktury obszarowej gospodarstw rodzinnych, ochrony środowiska i standardów UE. Warszawa, 167-171.
Werewka, J. (1997). Protokoły dostępu i charakterystyki czasowe magistrali miejscowych. Pomiary *Automatyka Robotyka, 7*, 4-11.
Więcek, H.(1995). System sterowania i wizualizacji procesów technologicznych. *Pomiary Automatyka Kontrola, 8*, 231-239.
Williams R. I. (1992). *Handbook of SCADA systems for the oil and gas industry*. Oxford, Wyd. Elsevier Advanced Technology, 41-44.
Wrzuszczak, J. (2010). Projektowanie wymiany danych na przykładzie małego systemu sterowania i wizualizacji. Pozyskano z: http://www.par.pl/390-projektowanie-wymiany-danych-na-przykladzie-malego-systemu-sterowania-i-wizualizacji.html.
Zbrzeżny, M. (2009). Co to jest OPC. Pozyskano z: http://maciej-progtech.blogspot.com/2009/11/co-to-jest-opc-w-prostych-zonierskich.html.