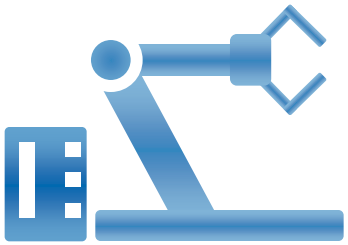
Automatyka i robotyka przemysłowa



Wprowadzenie do automatyki

Programowalne układy logiczne

PLC – programowane sterowniki logiczne

Programowalny sterownik logiczny (*ang. Programmable Logic Controller, PLC*) to wyspecjalizowane urządzenie mikroprocesorowe wyposażone w programowalną pamięć, które realizuje zadanie sterowania wykonując cyklicznie program zapisany w pamięci.

Standard PLC opisuje **norma IEC 61131** opracowana w latach dziewięćdziesiątych XX w. przez Międzynarodową Komisję Elektrotechniki (*International Electrotechnical Commision, IEC*) w odpowiedzi na potrzebę wprowadzenia standardów zwłaszcza w zakresie metod programowania sterowników (ze względu na dużą różnorodność rozwiązań stosowanych przez producentów).

Norma definiuje 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”

Klasyfikacja ze względu na budowę

- sterowniki kompaktowe – mają sztywną architekturę w jednej obudowie umieszczone są zasilacz, jednostka centralna CPU (mikroprocesor + pamięć), moduły wejść i wyjść (zwykle cyfrowych),
- sterowniki modułowe – mają architekturę elastyczną, składają się z modułów, które mogą być dowolnie łączone i konfigurowane,
- sterowniki kompaktowo–modułowe – sterowniki kompaktowe, które mogą być rozbudowywane.

Klasyfikacja ze względu na możliwości

- przekaźniki inteligentne – najmniejsze sterowniki, które mogą obsługiwać do kilkunastu wejść cyfrowych (rzadziej analogowych),
- sterowniki małe – stosowane w przypadku prostych zadań, do sterowania pojedynczymi urządzeniami, mają zwykle kilkadziesiąt wejść/wyjść, na ogół są sterownikami kompaktowymi,
- sterowniki średnie i duże – wykorzystywane w przypadku złożonych zadań, są sterownikami modułowymi, mają kilkaset (sterowniki średnie), kilka tysięcy (sterowniki duże) wejść/wyjść.

Rodziny sterowników największych producentów

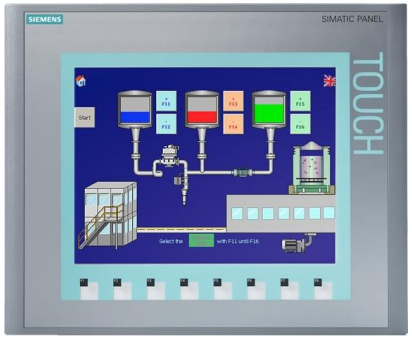
Producent	Przekaźniki inteligentne	Sterowniki		
		małe	średnie	duże
Siemens	Logo	SIMATIC S7-200	SIMATIC S7-300	SIMATIC S7-400
Schneider Electric	Zelio	Nano Micro Twido	Premium Compact Momentum	Quantum
GE Fanuc	VersaMax-Nano	VersaMax-Micro	90-30 VersaMax PACSystems RX3i	90-70 PACSystems RX7i
Mitsubishi Electric	ALPHA	MELSEC FX1 FX2	MELSEC QnAS	MELSEC QnA MELSEC System Q
Omron		CPM1, CPM2, CQM1H	C200H-alpha CJ1, CS1	CVM1
Rockwell Automation (Allen-Bradley)	Pico	MicroLogix	SLC500 FlexLogix ControlLogix	PLC-5

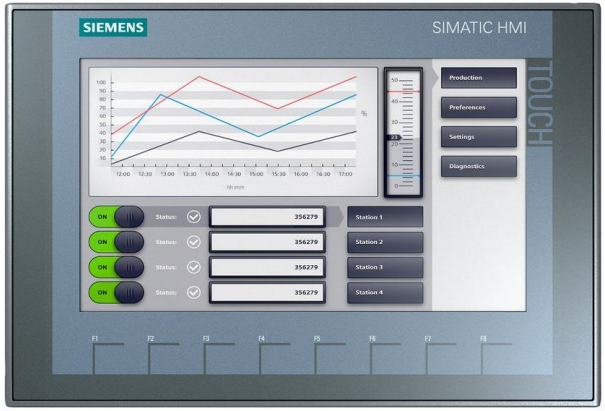
Przykłady sterowników PLC

LOGO!

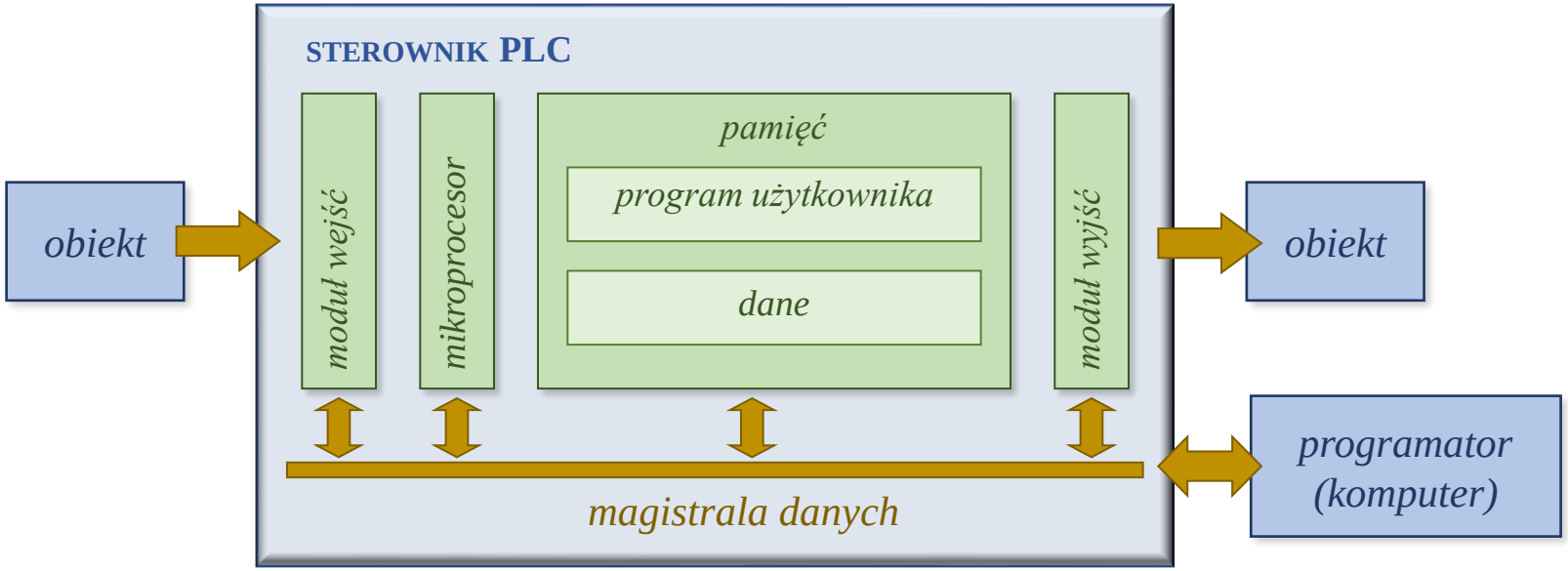
S7-1500

Panele HMI





S7-1200



Zasada działania sterownika

- sygnały z czujników kontrolowanego obiektu (procesu) zamieniane są w modułach wejściowych na sygnały cyfrowe akceptowane przez sterownik i zapamiętywane w obszarze pamięci danych wejściowych,
- mikroprocesor przetwarza program obliczając na podstawie danych wejściowych wartości sygnałów wyjściowych i zapisuje je w pamięci danych wyjściowych,
- moduły wyjściowe korzystając z danych wyjściowych generują sygnały dla urządzeń wykonawczych.

Cykl programowy PLC

Sterownik wykonuje działania w sposób cykliczny realizując tzw. cykl programowy sterownika. Na cykl pracy składają się cztery podstawowe fazy:

1. Odczyt danych wejściowych

   dane wejściowe są pobierane z modułów wejściowych i wprowadzane do pamięci sterownika.

2. Wykonanie programu użytkownika

   realizowany jest jeden przebieg programu.

3. Zapisu danych wyjściowych

   dane wyjściowe są wysyłane do modułów wyjściowych.

4. Wykonanie funkcji systemu operacyjnego sterownika

   wykonywane są czynności niezbędne do prawidłowego funkcjonowania sterownika: kontrola konfiguracji, diagnostyka, zarządzanie pamięcią i wykonaniem programu, komunikacja z urządzeniami zewnętrznymi.

Cykl programowy zwykle trwa od ułamka do kilkudziesięciu milisekund. W przypadku przekroczenia dopuszczalnego czasu (zwykle od 100 do 500 ms) cykl jest przerywany i rejestrowany jest błąd przekroczenia czasu.

Część 3 normy IEC 61131 wprowadza metody programowania sterowników PLC, definiując 2 grupy języków programowania:

Języki tekstowe

- język listy instrukcji IL (*ang. Instruction List*)
- język tekstu strukturalnego ST (*ang. Structured Text*)

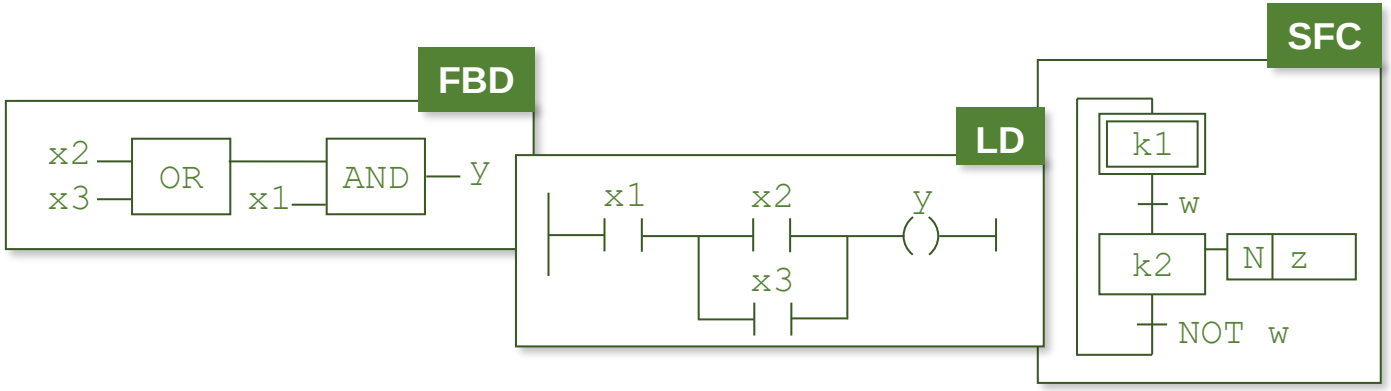
Języki graficzne

- język schematów drabinkowych LD (*ang. Ladder Diagram*)
- język funkcjonalnych schematów blokowych FBD (*ang. Function Block Diagram*)

```
IL
LD x1
AND( x2
OR x3
)
ST y
```

```
ST
y := x1 AND x2
IF y THEN
  z := z AND y;
END_IF;
```

Dodatkowo norma definiuje graficzną *metodę organizacji programu* w postaci sekwencyjnego schemat funkcjonalnego SFC (*ang. Sequential Function Chart*).



Język listy instrukcji IL (*ang. Instruction List*)

odpowiednik języka niskiego poziomu (asemblera), jego instrukcje bazują na liście rozkazów procesora (operacje logiczne, arytmetyczne, relacji, funkcje przerzutników, timerów, itp.).

Język tekstu strukturalnego ST (*ang. Structured Text*)

odpowiednik języka wysokiego poziomu, zbliżony do języków Pascal lub C.

Język schematów drabinkowych LD (*ang. Ladder Diagram*)

odpowiednik schematów układów cyfrowych realizowanych w technologii stykowo – przekaźnikowej, dodatkowo pozwala na wykorzystywanie funkcji realizujących operacje arytmetyczne, logiczne, porównania oraz bloków funkcjonalnych.

Język funkcjonalnych schematów blokowych FBD (*ang. Function Block Diagram*)

odpowiednik schematów opisujących układy logiczne przy pomocy bramek logicznych.

Metoda sekwencyjnego schematu funkcjonalnego SFC
(*ang. Sequential Function Chart*)

graficzna metoda projektowania zadań sterowania sekwencyjnego w oparciu o tzw. graf sekwencji.

Struktura programu

Każde zadanie sterowania stanowi tzw. projekt. Program rozwiązujący określone zadanie projektowe może składać się z oddzielnych modułów oprogramowania. Moduły te nazywane jednostkami organizacyjnymi oprogramowania (*ang. Program Organization Units, POU*). Rozbicie programu na jednostki organizacyjne poprawia czytelność i ułatwia przenoszenie jednostek organizacyjnych pomiędzy różnymi projektami.

POU wprowadzane przez normę IEC 61131

- funkcje – wyznaczają wartość argumentu wyjściowego na podstawie argumentów wejściowych (zależność wyjścia od wejścia jest jednoznaczna),
- bloki funkcjonalne – posiadają pamięć, wartość argumentu wyjściowego nie wynika wyłącznie z wartości argumentów wejściowych, zależy również od stanu pamięci,
- programy – podstawowe jednostki organizacyjne programu użytkowego, odpowiadają „programowi głównemu” w klasycznych językach programowania, programy mają dostęp do danych z modułów wejściowych i wyjściowych sterownika.

Norma zakłada, że pojedyncza jednostka organizacyjna musi być zaprogramowana tylko w jednym języku programowania.

Zmienne

Zmienne są używane przez jednostki organizacyjne programu (POU) do przechowywania i przetwarzania danych. Każda zmienna odpowiada pewnej informacji umieszczonej w strukturze wejść, wyjść lub w pamięci sterownika.

Zmienna prosta (skalarna, jednoelementowa) to pojedyncza wartość należąca do typu elementarnego (bit, liczba, znacznik czasu, tekst). Zmienne proste mogą być przedstawione symbolicznie (poprzez nazwę) lub reprezentowane bezpośrednio poprzez określenie ich położenia (odwołanie do wejścia/wyjścia lub komórki pamięci).

Deklaracja zmiennej symbolicznej

```
VAR
  <nazwa> AT<lokal>: <typ> := <wartość>;
END_VAR
```

`<nazwa>` unikalny ciąg znaków identyfikujący zmienną (litery alfabetu angielskiego, cyfry, podkreślenie, nie może zaczynać się od cyfry),

`<lokal>` określa położenie zmiennej zgodnie z zasadami adresowania bezpośredniego (opcjonalna),

`<typ>` rodzaj i zakres wartości, które może przyjmować zmienna,

`<wartość>` początkowa wartość zmiennej (opcjonalna).

Typy elementarne

BOOL wartości logiczne o rozmiarze 1 bitu, dane należy podawać jako `TRUE` lub `FALSE` albo `1` lub `0`

TIME czas, wartości należy poprzedzać symbolem `T#` lub `TIME#` i określić jednostkę: `d` lub `D` dni, `h` lub `H` godziny, `m` lub `M` minuty, `s` lub `S` sekundy, `ms` lub `MS` milisekundy, kolejne jednostki czasu podaje się w kolejności malejącej, dopuszczalne użycie symbolu podkreślenia jako separatora, np.: `t#5s`, `t#5d14h_12m_18s_3.5ms`

BYTE, WORD, DWORD, LWORD

ciągi (grupy) bitowe o rozmiarze odpowiednio: 8, 16, 32, 64 bitów.

SINT, INT, DINT, LINT

liczby całkowite o różnym zakresie (odpowiednio 1, 2, 4 i 8 bajtów).

REAL liczby rzeczywiste

DATE, DATE_AND_TIME

daty oraz daty połączone z czasem

STRING ciąg znaków

Lokalizacja zmiennej

Schemat opisu lokalizacji zmiennej (adresowanie zmiennej)

`%<położenie><rozmiar><adres>`

`<położenie>` litera, obszar pamięci w którym jest umieszczona zmienna,

`<rozmiar>` litera, liczba bitów, które zmienna zajmuje,

`<adres>` ciąg liczb całkowitych rozdzielonych kropkami, położenie zmiennej w danym obszarze pamięci sterownika, kolejne pola są interpretowane jako hierarchiczny sposób adresacji, pierwsze pole od lewej oznacza poziom najwyższy (adres zależy od urządzenia i jego konfiguracji).

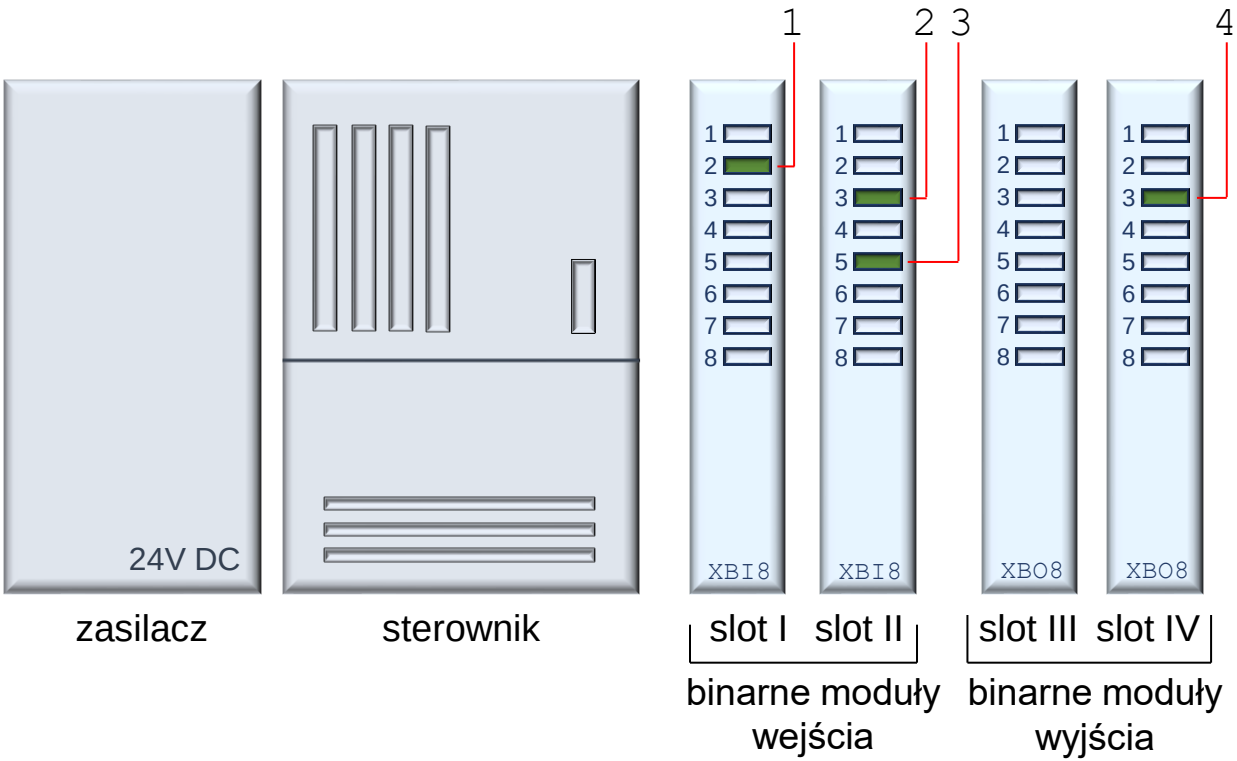
Położenie zmiennej	
Symbol	Znaczenie
I	pamięć danych wejściowych
Q	pamięć danych wyjściowych
M	pamięć danych użytkownika

Rozmiar zmiennej	
Symbol	Znaczenie
X lub brak	bit
B	bajt (8 bitów)
W	słowo (16 bitów)
D	podwójne słowo (32 bity)
L	poczwórne słowo (64 bity)

Jeżeli położenie nie zostało określone zmienna jest lokowana w pamięci sterownika.

W programie można odwołać się do zmiennej bezpośrednio poprzez wskazanie jej lokalizacji (metoda niepolecana, może prowadzić do trudno wykrywalnych błędów).

Lokalizacja zmiennych – przykłady



1. `%IX1.2` – drugi bit modułu wejściowego umieszczonego w pierwszym gnieździe
2. `%IX2.3` – trzeci bit modułu wejściowego umieszczonego w drugim gnieździe
3. `%IX2.5` – piąty bit modułu wejściowego umieszczonego w drugim gnieździe
4. `%QX4.3` – trzeci bit modułu wyjściowego umieszczonego w czwartym gnieździe.

Język schematów drabinkowych LD

Język schematów drabinkowych LD jest odpowiednikiem schematów obwodowych, (drabinkowych) układów realizowanych w technologii stykowo – przekaźnikowej.

Wybrane elementy języka LD

szyny prądowe ograniczają obwód języka LD, prawa opcjonalna

połączenie poziome przesyła stan elementu znajdującego się po lewej stronie połączenia na jego prawą stronę

połączenie pionowe realizuje logiczną alternatywę (OR) sygnałów poziomych znajdujących się po lewej stronie i ustawia jej wartość w połączeniach poziomych po prawej stronie

zestyk zwierny (normalnie otwarty) przyjmuje wartość 1 gdy skojarzona z nim zmienna ma wartość 1 (w przeciwnym wypadku ma wartość 0)

zestyk rozwierny (normalnie zamknięty) przyjmuje wartość 1 gdy skojarzona z nim zmienna ma wartość 0 (w przeciwnym wypadku ma wartość 0)

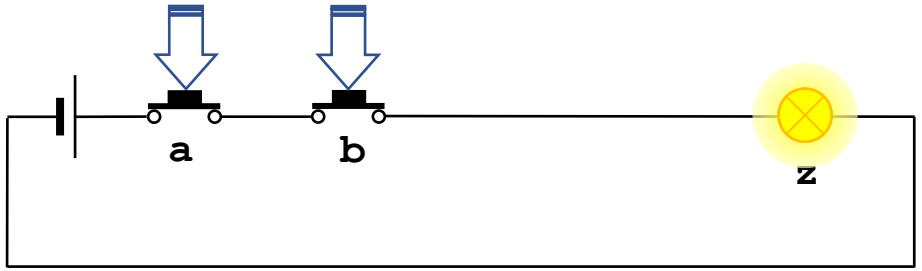
cewka ustawia wartość skojarzonej z nią zmiennej na podstawie wartości połączenia z lewej strony

Kolejne szczeble drabiny języka LD przetwarzane są od góry do dołu, a pojedynczy obwód od strony lewej do prawej.

Język LD – koniunkcja

```
VAR
   a AT%IX1.1: BOOL;   (*pierwszy sygnał wejściowy*)
   b AT%IX1.2: BOOL;   (*drugi sygnał wejściowy*)
   z AT%QX2.1: BOOL;   (*sygnał wyjściowy*)
END_VAR
```

$$z = a\ b$$

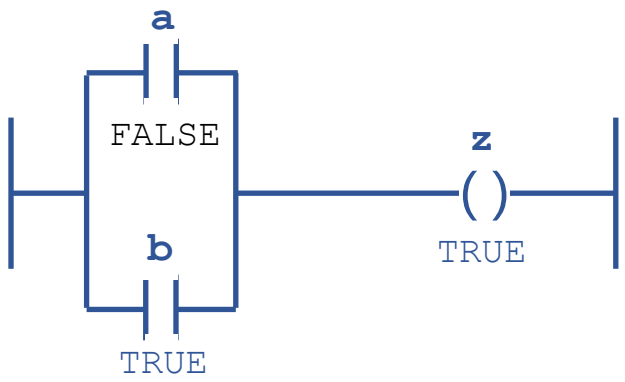


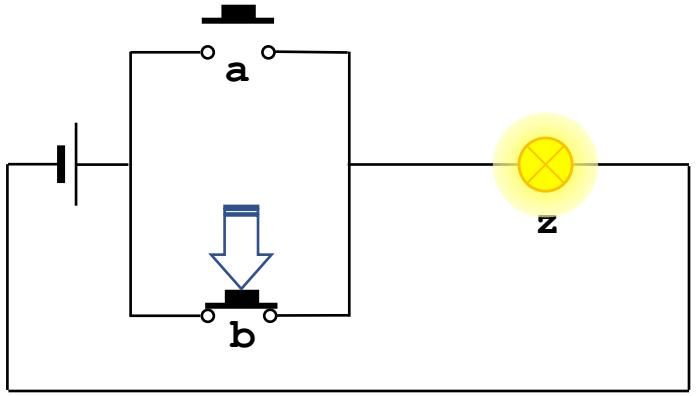
Żarówka zapala się po naciśnięciu przycisków a i b (koniunkcja)

Język LD – alternatywa

```
VAR
   a AT%IX1.1: BOOL;   (*pierwszy sygnał wejściowy*)
   b AT%IX1.2: BOOL;   (*drugi sygnał wejściowy*)
   z AT%QX2.1: BOOL;   (*sygnał wyjściowy*)
END_VAR
```

$$z = a + b$$



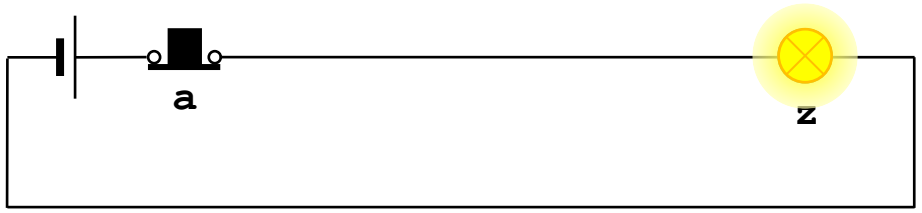


Żarówka zapala się po naciśnięciu przycisku a lub b (alternatywa)

Język LD – negacja

```
VAR
   a AT%IX1.1: BOOL;   (*pierwszy sygnał wejściowy*)
   z AT%QX2.1: BOOL;   (*sygnał wyjściowy*)
END_VAR
```

$$z = \bar{a}$$



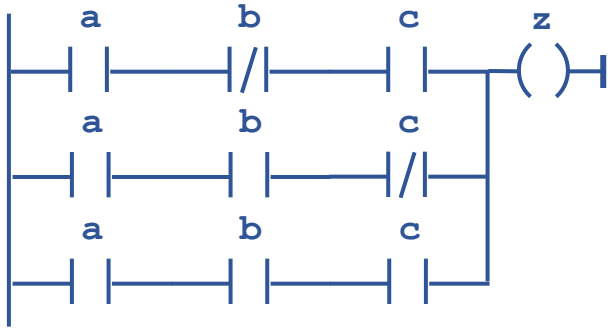
Żarówka zgaśnie po naciśnięciu przycisku
zapali się po jego zwolnieniu (negacja)

Przykład 1. – program w języku LD

```
VAR
    a AT%IX1.1: BOOL;      b AT%IX1.2: BOOL;      c AT%IX1.3: BOOL;
    z AT%QX2.1: BOOL;
END_VAR
```

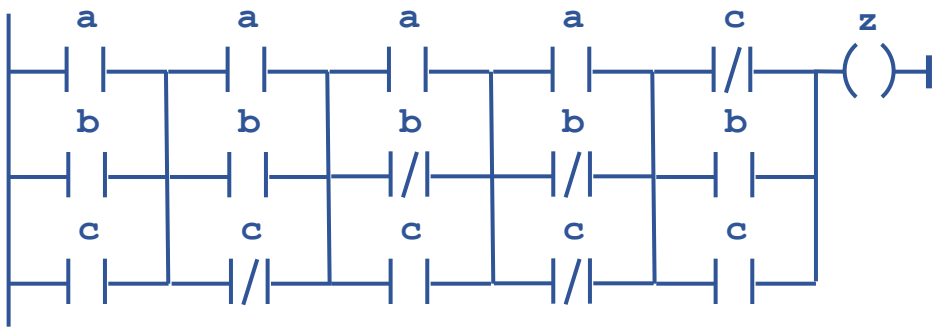
Kanoniczna postać dysjunkcyjna

$z = a\bar{b}c + ab\bar{c} + abc$



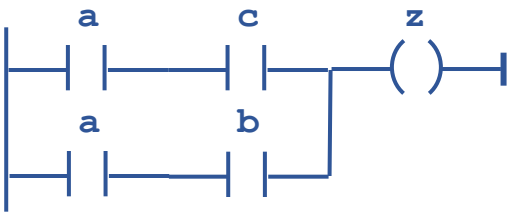
Kanoniczna postać koniunkcyjna

$z = (a + b + c)(a + b + \bar{c})(a + \bar{b} + c)(a + \bar{b} + \bar{c})(\bar{a} + b + c)$



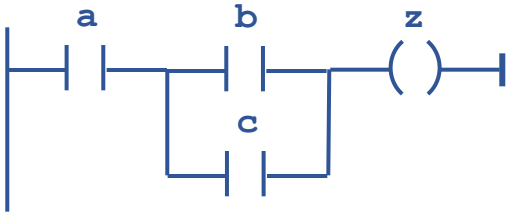
Minimalna postać dysjunkcyjna

$z = ac + ab$



Minimalna postać koniunkcyjna

$z = a(b + c)$

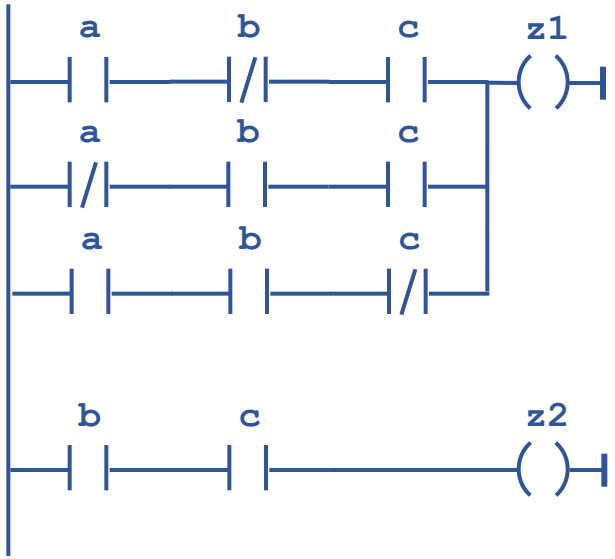


Przykład 2. – program w języku LD

```
VAR
    a AT%IX1.1: BOOL;      b AT%IX1.2: BOOL;      c AT%IX1.3: BOOL;
    z1 AT%QX2.1: BOOL;     z2 AT%QX2.1: BOOL;
END_VAR
```

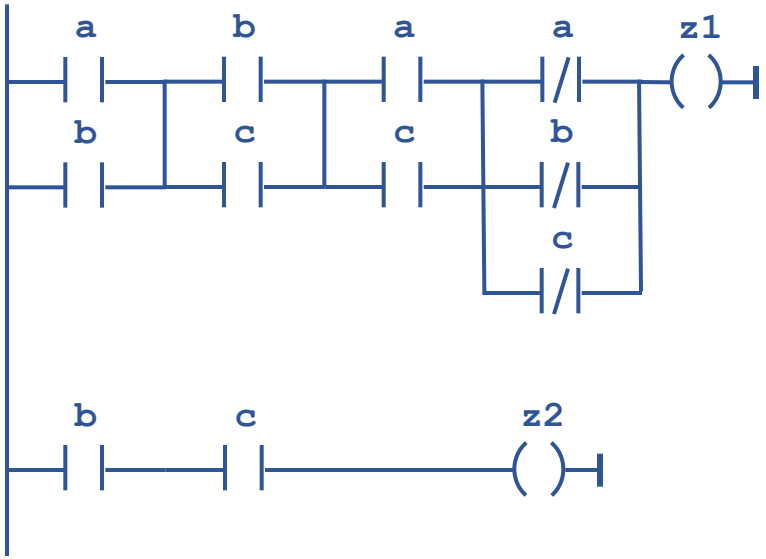
Minimalna postać dysjunkcyjna

$z_1 = a\bar{b}c + \bar{a}bc + ab\bar{c}$
$z_2 = bc$



Minimalna postać koniunkcyjna

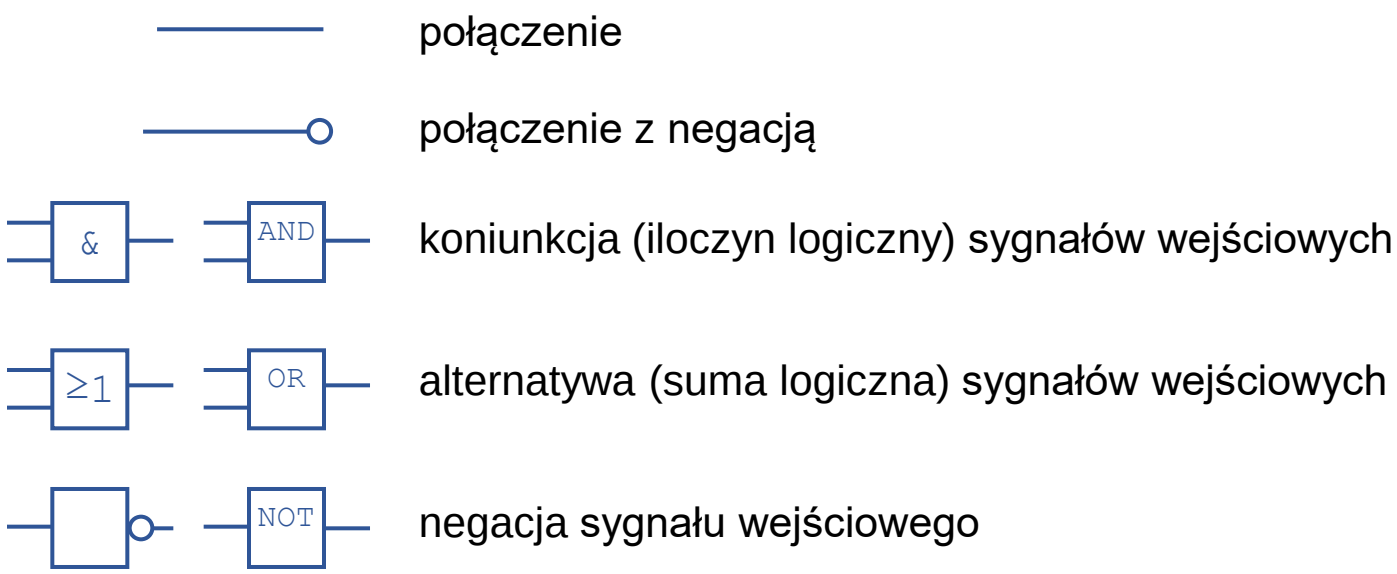
$z_1 = (a + b)(b + c)(a + c)(\bar{a} + \bar{b} + \bar{c})$
$z_2 = bc$



Język funkcjonalnych schematów blokowych FBD

Język funkcjonalnych schematów blokowych FBD pozwala na zapis programu w postaci zbioru bloków funkcyjnych połączonych w obwody. Wykonanie programu polega na przepływie sygnału przez kolejne bloki analogicznie do przepływu prądu (gazu, cieczy) w układzie zbudowanym z elektronicznych (pneumatycznych, hydraulicznych) elementów logicznych.

Wybrane elementy języka FBD

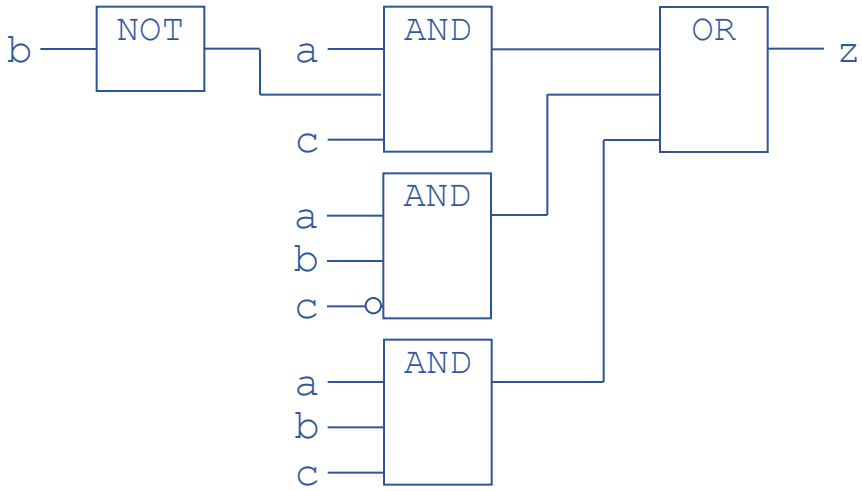


Przykład 1. – program w języku FBD

```
VAR
    a AT%IX1.1: BOOL;      b AT%IX1.2: BOOL;      c AT%IX1.3: BOOL;
    z AT%QX2.1: BOOL;
END_VAR
```

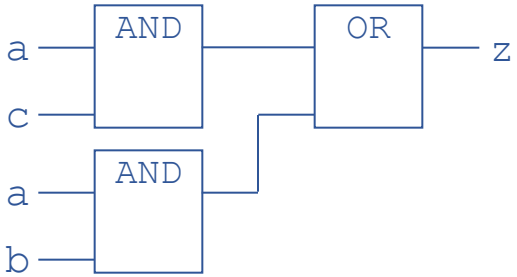
Kanoniczna postać dysjunkcyjna

$z = a\bar{b}c + ab\bar{c} + abc$



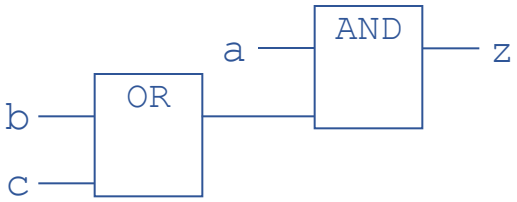
Minimalna postać dysjunkcyjna

$z = ac + ab$



Minimalna postać koniunkcyjna

$z = a(b + c)$

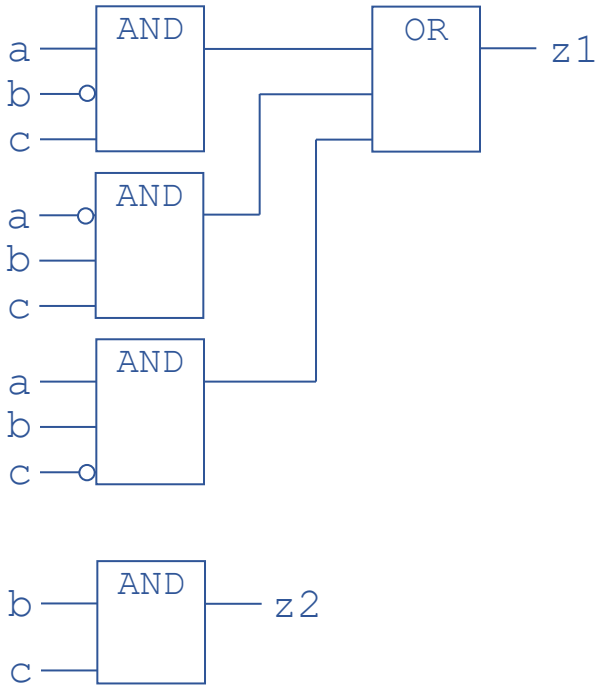


Przykład 2. – program w języku FBD

```
VAR
    a AT%IX1.1: BOOL;      b AT%IX1.2: BOOL;      c AT%IX1.3: BOOL;
    z1 AT%QX2.1: BOOL;     z2 AT%QX2.1: BOOL;
END_VAR
```

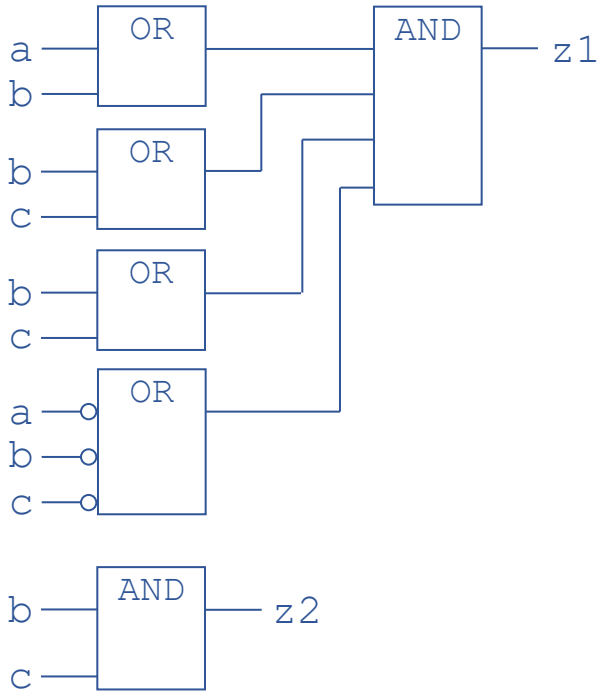
Minimalna postać dysjunkcyjna

$z_1 = a\bar{b}c + \bar{a}bc + ab\bar{c}$
$z_2 = bc$



Minimalna postać koniunkcyjna

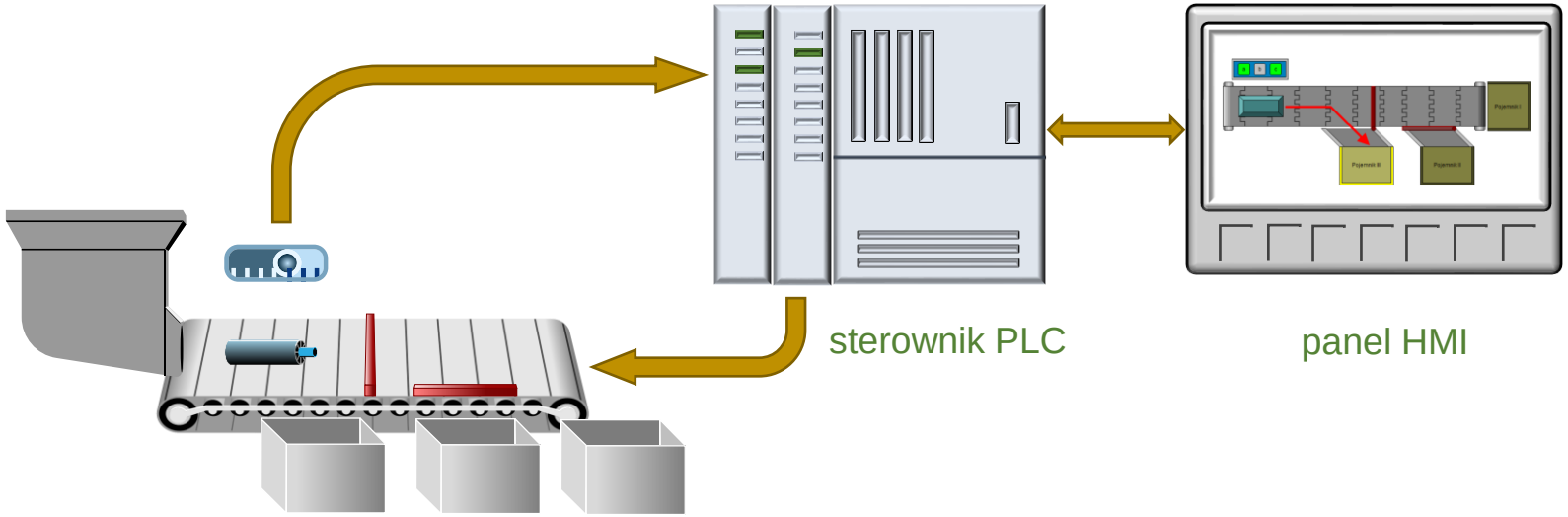
$z_1 = (a + b)(b + c)(a + c)(\bar{a} + \bar{b} + \bar{c})$
$z_2 = bc$



Wizualizacja

Wizualizacja jest elementem programu sterownika PLC, zazwyczaj jest wyświetlana na panelu operatorskim, pozwala na monitorowanie przebiegu procesu, wprowadzanie parametrów, kontrolowanie pracy sterownika, itp.

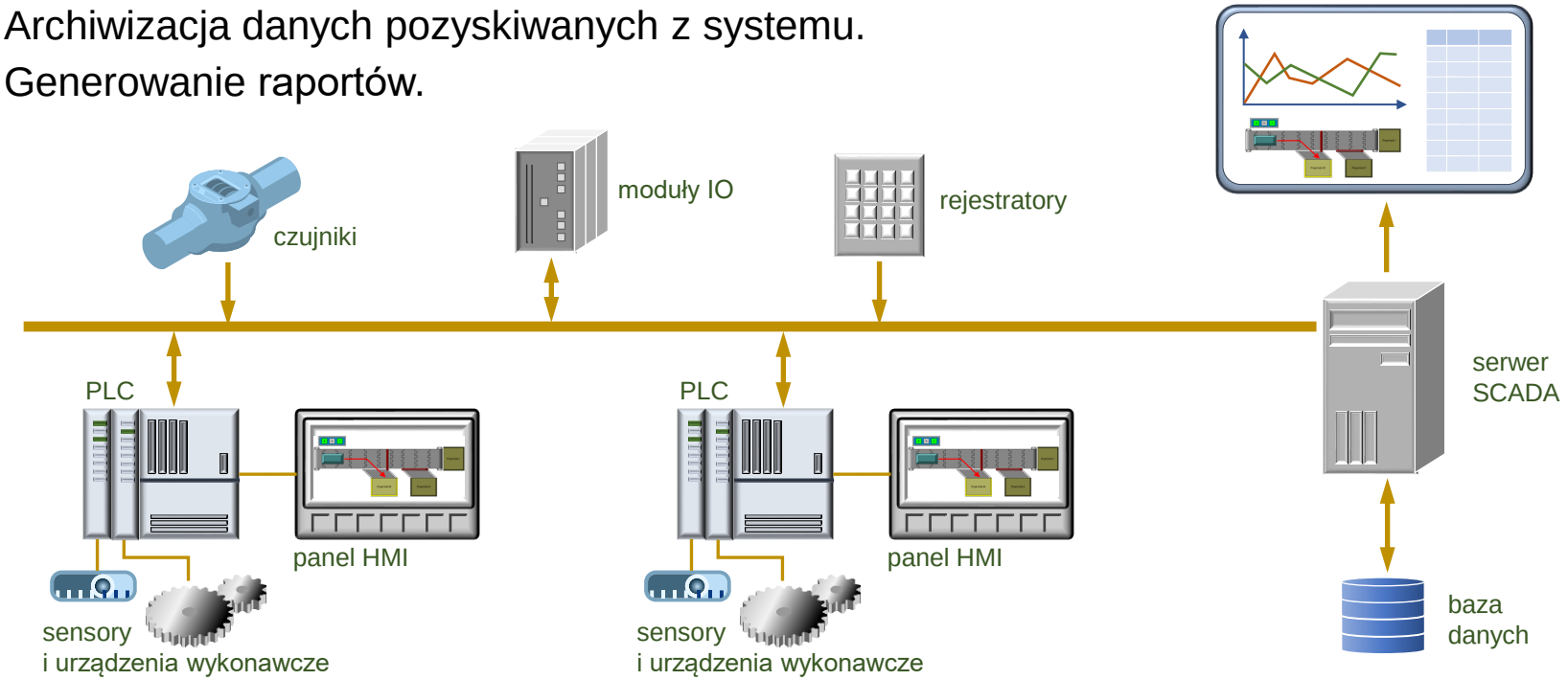
Panel operatorski, **panel HMI** (*ang. Human Machine Interface*) to specjalizowany wyświetlacz współpracujący ze sterownikiem PLC (połączenie przez interfejs RS-232, ETHERNET, itp.), może być wyposażony w przyciski i/lub ekran dotykowy, przeznaczony do wyświetlania wizualizacji przebiegu procesu.



SCADA, *ang. Supervisory Control And Data Acquisition* – nadzór i akwizycja danych, system informatyczny nadrzędny w stosunku do warstwy sprzętowej, nadzoruje przebieg całego procesu produkcyjnego.

Funkcje systemu SCADA

- Wymiana danych ze sterownikami PLC i innymi modułami systemu produkcyjnego.
- Wyświetlanie informacji o przebiegu całego procesu oraz jego składowych.
- Zarządzanie alarmami.
- Archiwizacja danych pozyskiwanych z systemu.
- Generowanie raportów.



Systemy SCADA