Bezpieczeństwo infrastruktury krytycznej i systemów sterowania przemysłowego IoT - Laboratorium 8 – Programowanie PLC

Forma: praca w samodzielna / małe zespoły (2 - 4 osoby)

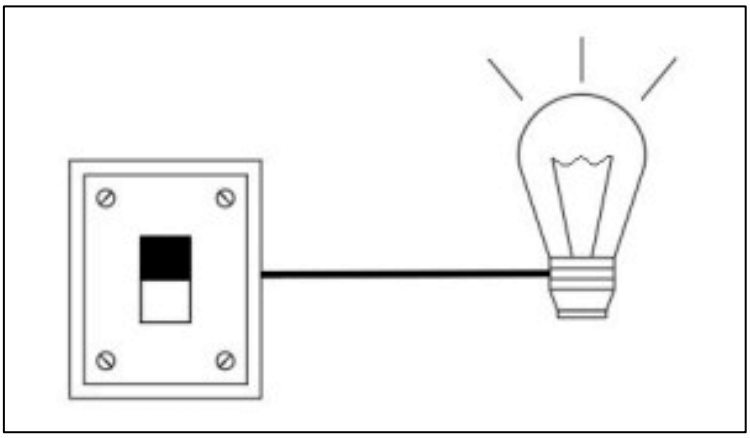
Wstęp teoretyczny

PLC

https://www.electrical4u.com/programmable-logic-controllers/

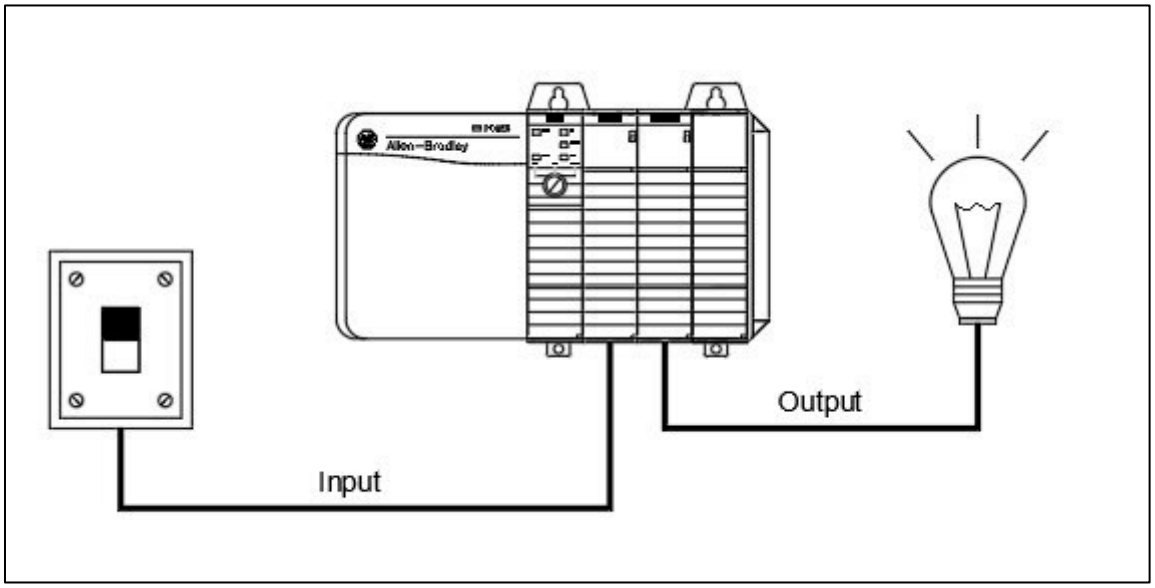
PLC, czyli sterownik programowalny (Programmable Logic Controller), to komputer zaprojektowany tak, aby działał niezawodnie w trudnych warunkach środowiska przemysłowego, takich jak ekstremalne temperatury czy zapylone otoczenie. Automatyzuje on procesy w przemyśle, w tym w produkcji oraz oczyszczaniu ścieków. Sterowniki PLC mają wiele cech wspólnych z komputerem osobistym, który masz w domu. Oba posiadają zasilacz, jednostkę centralną CPU (Central Processing Unit), wejścia i wyjścia (I/O), pamięć oraz oprogramowanie operacyjne (choć jest to inne oprogramowanie operacyjne). Największe różnice polegają na tym, że PLC może wykonywać dyskretne i ciągłe funkcje, których komputer PC nie jest w stanie realizować, a także na tym, że PLC jest znacznie lepiej przystosowany do pracy w trudnych warunkach przemysłowych. PLC można traktować jako „wzmocniony" (ruggedized) komputer cyfrowy, który zarządza procesami elektromechanicznymi w środowisku przemysłowym. Sterowniki PLC odgrywają kluczową rolę w dziedzinie automatyki, będąc częścią większego systemu SCADA. PLC można zaprogramować zgodnie z wymaganiami operacyjnymi danego procesu. W przemyśle wytwórczym konieczne jest przeprogramowywanie ze względu na zmieniający się charakter produkcji. Aby pokonać tę trudność, wprowadzono systemy sterowania oparte na PLC. Zanim przyjrzymy się różnym zastosowaniom sterowników PLC, omówimy najpierw podstawy działania PLC. Sterowniki PLC są powszechnie używane w praktycznie każdej gałęzi przemysłu – ze względu na uniwersalność tego urządzenia. Sterownik ma też wiele innych zastosowań, nawet w życiu codziennym – na przykład do sterowania sygnalizatorami świetlnymi, ruchomymi schodami, windami czy rozmaitymi systemami, w jakie wyposażone są nowoczesne inteligentne domy.

Sterowniki PLC, stworzone przez Dicka Morleya w 1964 roku, zrewolucjonizowały sektor przemysłowy i produkcyjny dzięki funkcjom takim jak odmierzanie czasu, zliczanie oraz przetwarzanie sygnałów. Główną zaletą PLC w porównaniu z „twardo okablowanym" systemem sterowania (hard-wired) jest to, że po zaprogramowaniu można łatwo wrócić i zmienić program PLC przy niewielkich kosztach (praktycznie tylko koszt czasu programisty). W tradycyjnym układzie opartym na okablowaniu, w razie konieczności zmian trzeba właściwie wyrwać przewody i zacząć od początku, co jest droższe i zajmuje więcej czasu. Przyjrzyjmy się przykładzie, aby lepiej zrozumieć tę zaletę. Wyobraź sobie, że masz lampę podłączoną do przełącznika. Zasadniczo lampa działa w dwóch stanach – WŁĄCZONA i WYŁĄCZONA. Teraz otrzymujesz zadanie: po włączeniu przełącznika lampa ma się zapalić dopiero po 30 sekundach. W przypadku tego twardo okablowanego układu – jesteśmy w kropce. Jedynym sposobem, aby to osiągnąć, jest całkowite przeokablowanie naszego obwodu i dodanie przekaźnika czasowego (timing relay). To dużo zachodu jak na tak drobną zmianę.

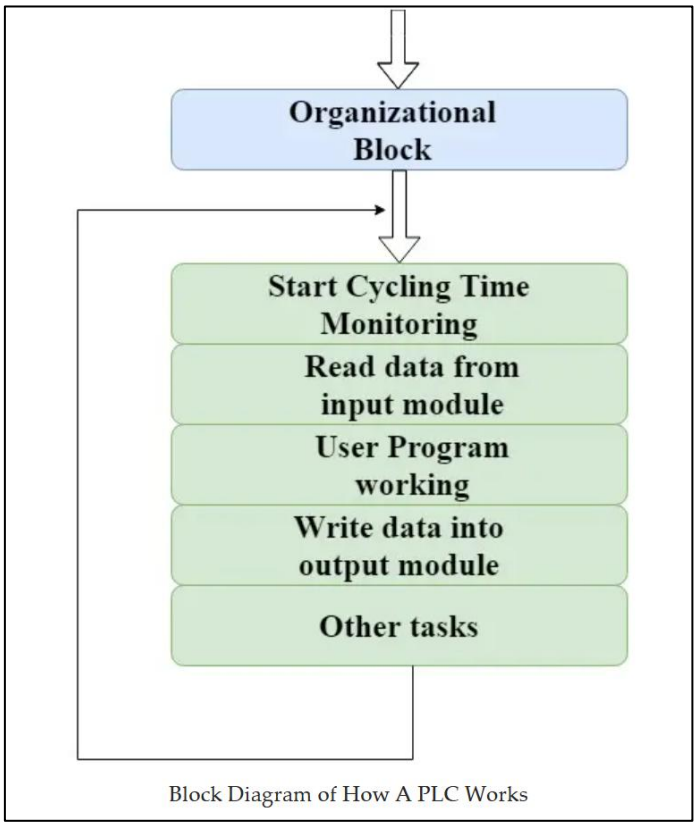


Tutaj właśnie pojawia się sterownik PLC, który nie wymaga żadnego dodatkowego okablowania ani sprzętu, aby wprowadzić taką zmianę. Zamiast tego potrzebna jest jedynie prosta modyfikacja kodu zaprogramowanie PLC tak, by włączał lampę dopiero 30 sekund po przełączeniu przełącznika w pozycję ON. Dzięki użyciu PLC łatwo jest obsługiwać wiele wejść i wyjść jednocześnie.

To tylko prosty przykład PLC ma możliwość sterowania znacznie większymi i bardziej złożonymi procesami. Sterownik PLC może być dostosowany do konkretnej aplikacji oraz potrzeb użytkownika.



Działanie sterownika PLC można łatwo zrozumieć jako cykliczną metodę skanowania, znaną jako cykl skanowania (scan cycle).

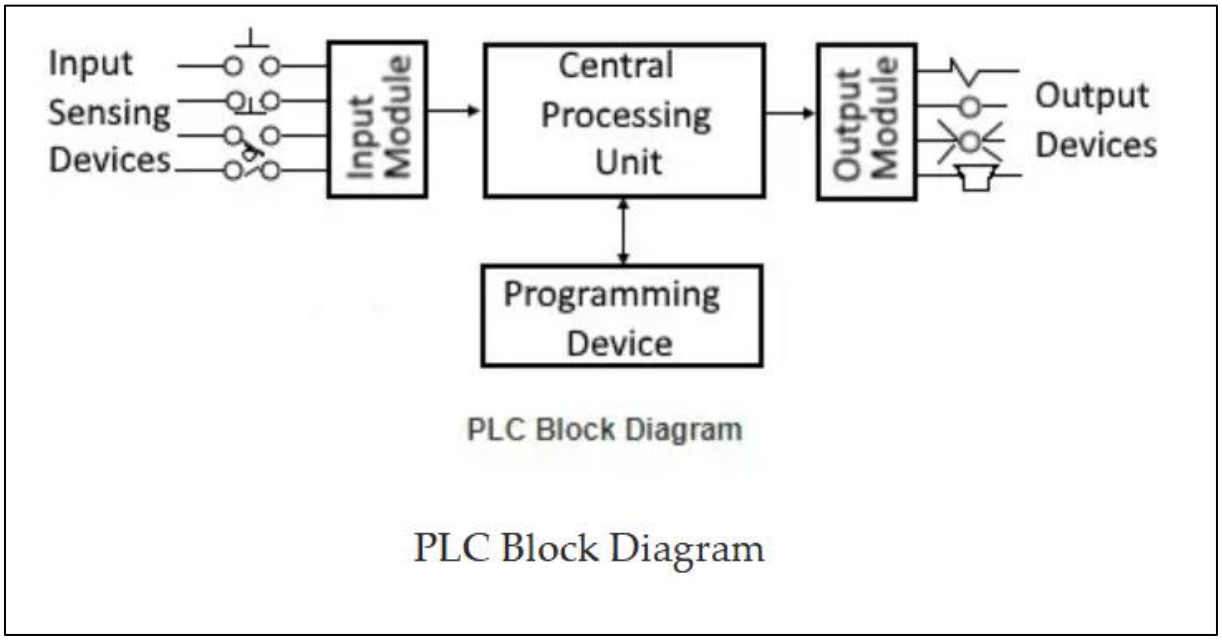


Block Diagram of How A PLC Works

Proces skanowania PLC obejmuje następujące kroki:

1. System operacyjny rozpoczyna cykl pracy i monitorowanie czasu.
2. Jednostka CPU zaczyna odczytywać dane z modułu wejściowego i sprawdza status wszystkich wejść.
3. CPU rozpoczyna wykonywanie programu użytkownika lub programu aplikacyjnego napisanego w języku drabinkowym (relay-ladder logic) lub innym języku programowania PLC.
4. Następnie CPU wykonuje wszystkie wewnętrzne zadania diagnostyczne i komunikacyjne.
5. Zgodnie z wynikami programu CPU zapisuje dane do modułu wyjściowego, aby wszystkie wyjścia zostały zaktualizowane.
6. Proces ten trwa nieprzerwanie tak długo, jak PLC znajduje się w trybie pracy (run mode).

Fizyczna struktura PLC
Struktura sterownika PLC jest niemal podobna do architektury komputera.



PLC Block Diagram

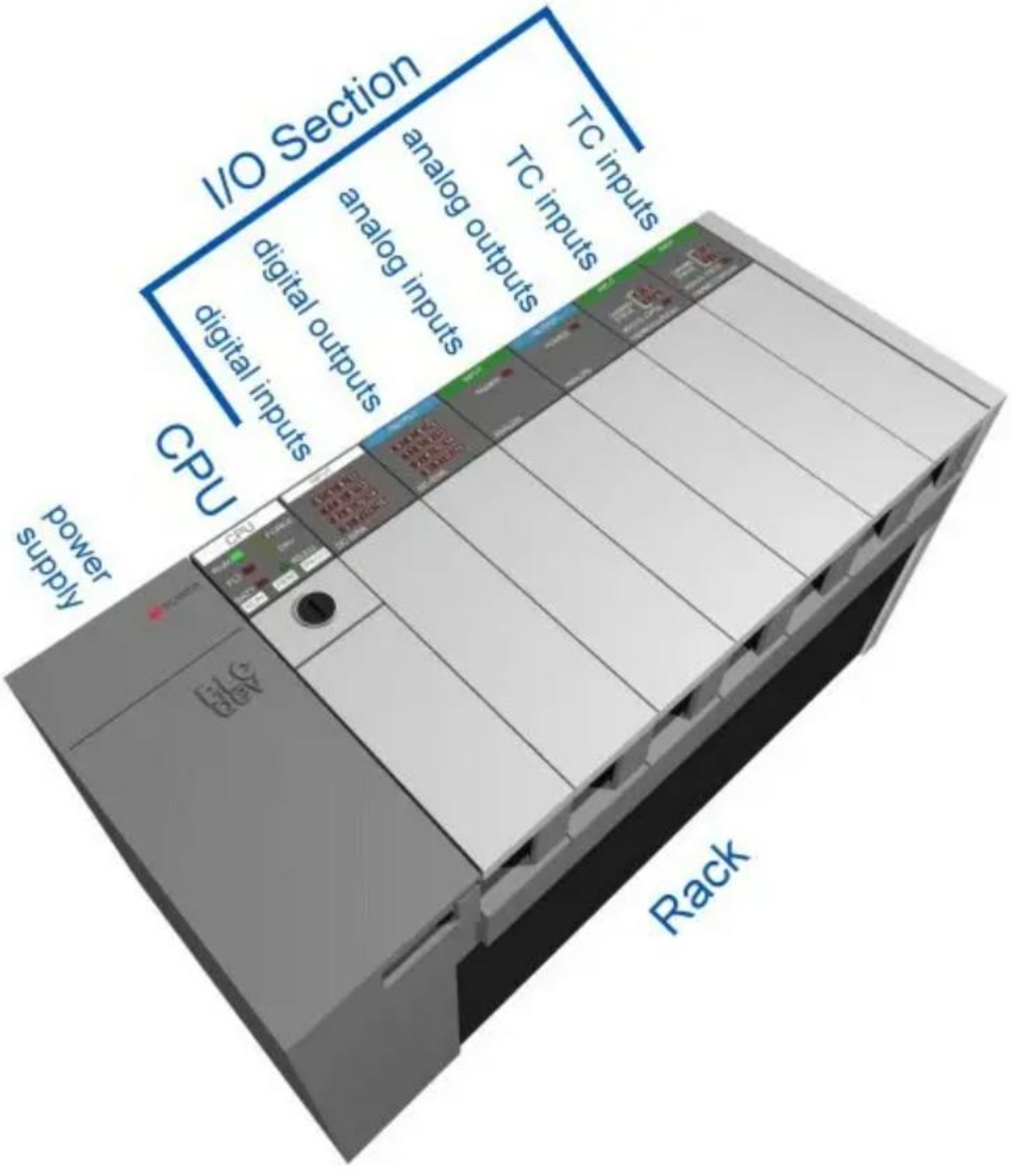
Sterowniki programowalne (PLC) nieustannie monitorują wartości sygnałów wejściowych pochodzących z różnych urządzeń pomiarowych (np. akcelerometr, waga przemysłowa, sygnały twardo okablowane itp.) i generują odpowiadające im sygnały wyjściowe w zależności od charakteru procesu produkcyjnego oraz specyfiki branży. Typowy schemat blokowy PLC składa się z pięciu części:

- Rack lub chassis (konstrukcja nośna)
- Moduł zasilający (Power Supply Module)
- Jednostka centralna (CPU – Central Processing Unit)
- Moduł wejść i wyjść (Input & Output Module)
- Moduł komunikacyjny (Communication Interface Module)

Rack lub Chassis

We wszystkich systemach PLC rack lub chassis stanowi najważniejszy moduł i pełni rolę szkieletu całej konstrukcji. Sterowniki PLC występują w różnych kształtach i rozmiarach. Gdy w grę wchodzą bardziej złożone systemy sterowania, wymagane są większe i bardziej rozbudowane racki PLC.

Mały PLC jest wyposażony w stałą konfigurację pinów wejściowych i wyjściowych. Z tego względu opracowano modułowe racki PLC, które umożliwiają montaż różnych typów modułów I/O w formie wsuwanej (sliding) i zatrzaskowej (fit-in). Wszystkie moduły wejść/wyjść instalowane są wewnątrz tego racka/chassis.



Breakdown of PLC Rack

Moduł zasilający

Moduł ten dostarcza wymagane zasilanie dla całego systemu PLC. Przekształca dostępne zasilanie prądem przemiennym (AC) na zasilanie prądem stałym (DC), które jest potrzebne jednostce CPU oraz modułom wejścia/wyjścia. PLC zazwyczaj pracują przy zasilaniu 24V DC, lecz niektóre używają izolowanego źródła zasilania.

Moduł CPU i pamięć

Moduł CPU zawiera procesor centralny oraz pamięci ROM i RAM. Pamięć ROM obejmuje system operacyjny, sterowniki oraz programy aplikacyjne. RAM służy do przechowywania programów i danych. CPU jest „mózgiem" sterownika PLC i zawiera mikroprocesor oktalny lub heksadecymalny.

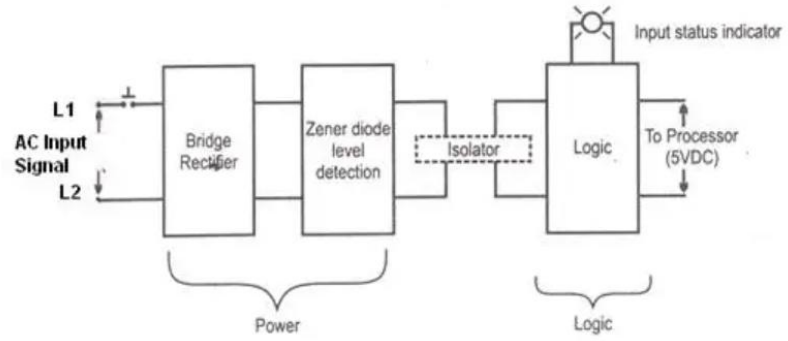
Jako procesor oparty na mikroprocesorze, CPU zastępuje przekaźniki, liczniki i timery. W sterowniku PLC mogą być zastosowane dwa typy procesorów: procesor jednobitowy lub słowowy. Procesor bitowy jest używany do wykonywania funkcji logicznych. Procesory słowowe stosuje się do przetwarzania danych tekstowych i numerycznych, a także do kontrolowania i rejestrowania danych.

CPU odczytuje dane wejściowe z czujników, przetwarza je i wysyła polecenia do urządzeń wykonawczych. Jak wspomniano wcześniej, źródło zasilania DC wymaga sygnałów napięciowych. CPU zawiera również inne elementy elektryczne służące do podłączenia kabli używanych przez pozostałe jednostki.

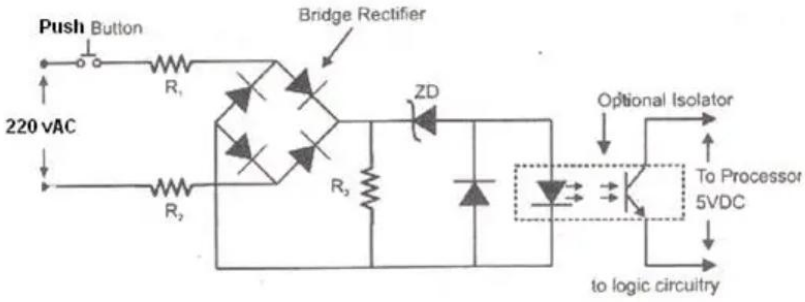
Input and Output Module

Moduły wejść i wyjść sterownika PLC są kluczowe przy wykrywaniu parametrów fizycznych, takich jak temperatura, ciśnienie czy przepływ, umożliwiając interakcję z różnymi procesami przemysłowymi.

Urządzeniami wejściowymi mogą być np. przyciski start i stop, przełączniki itp., natomiast urządzeniami wyjściowymi mogą być grzałki elektryczne, zawory, przekaźniki itp. Moduł I/O pomaga w interfejsowaniu urządzeń wejściowych i wyjściowych z mikroprocesorem. Moduł wejściowy PLC pokazano na poniższym rysunku.



PLC Input Module



Circuit Diagram of PLC Input Module

Moduł wejściowy PLC – schemat blokowy i działanie

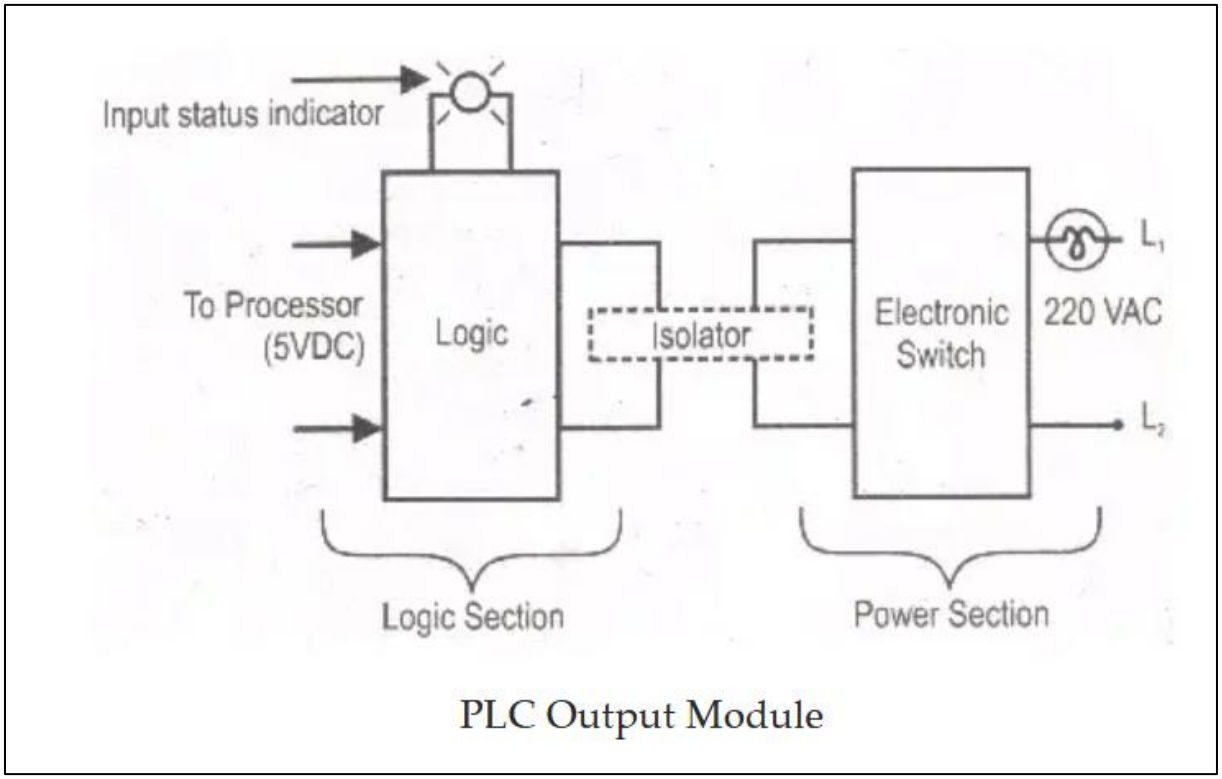
Moduł wejściowy PLC posiada cztery główne funkcje:

1. Interfejs wejściowy odbiera sygnały z urządzeń procesowych o napięciu 220 V AC.
2. Konwertuje sygnał wejściowy na 5 V DC, które mogą być użyte przez PLC.
3. Blok izolatora służy do izolowania PLC i chroni go przed wahaniami napięcia.
4. Po tym sygnał jest wysyłany do końca wyjściowego, czyli do PLC.

W module wejściowym wyróżnia się dwa główne obwody: sekcję zasilania i sekcję logiczną. Obie sekcje są od siebie elektrycznie odizolowane. Początkowo przycisk jest zamknięty, więc zasilanie 220 V AC doprowadzane jest do układu mostkowego przez rezystory R1 i R2.

Mostek prostowniczy (np. mostek diodowy) służy do zamiany sygnału AC na DC, a dioda Zenera zapewnia niskie napięcie zasilania dla diody LED. Gdy światło z LED pada na fototranzystor, ten przechodzi w stan przewodzenia. Ostatecznie dostarczane jest 5V DC do procesora.

Moduł wyjściowy PLC działa podobnie do modułu wejściowego, lecz w odwrotnym kierunku. Łączy obciążenie wyjściowe z procesorem. W tym przypadku pierwszą sekcją jest sekcja logiczna, a dopiero później sekcja zasilania.



PLC Output Module

Kiedy sygnał logiczny wysoki (HIGH) zostaje wygenerowany przez procesor, dioda LED zapala się, oświetlając fototranzystor. Gdy tranzystor przechodzi w stan przewodzenia, generuje impuls do bramki triaka. Blok izolatora służy do oddzielenia sekcji logicznej od sekcji sterującej.

Moduł komunikacyjny (Communication Interface Module)

Inteligentne moduły wejścia/wyjścia są wykorzystywane do przesyłania informacji pomiędzy CPU a sieciami komunikacyjnymi. Moduły te pozwalają na łączenie z innymi sterownikami PLC i komputerami znajdującymi się w odległych lokalizacjach.

Rodzaje sterowników PLC

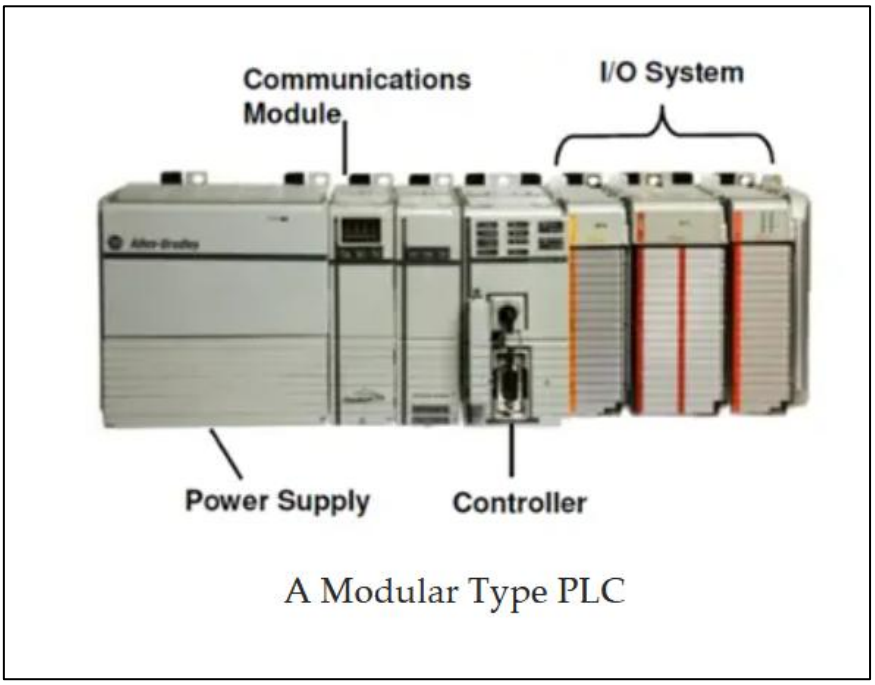
Dwa główne typy sterowników PLC to:

Sterownik kompaktowy (Compact PLC)

W jednej obudowie znajduje się wiele modułów. Posiada ustaloną liczbę modułów wejściowych i wyjściowych oraz zewnętrzne karty I/O. Nie można dodawać kolejnych modułów. Producent określa liczbę wejść i wyjść.

Sterownik modułowy (Modular PLC)

Ten typ PLC umożliwia rozbudowę poprzez dodatkowe moduły, stąd nazwa PLC modułowy. Komponenty I/O mogą być zwiększane. Jest prostszy w użyciu, ponieważ każdy element działa niezależnie.



A Modular Type PLC

Typy PLC na podstawie rodzaju wyjść

- Wyjścia przekaźnikowe (Relay output)
- Wyjścia tranzystorowe (Transistor output)
- Wyjścia triakowe (Triac output)

Wyjścia przekaźnikowe najlepiej współpracują z urządzeniami AC i DC. Wyjścia tranzystorowe używane są do operacji przełączania w mikroprocesorach.

Rodzaje PLC według wielkości

- Mini PLC
- Micro PLC
- Nano PLC

Najpopularniejsi producenci PLC

- Allen Bradley
- ABB
- Siemens
- Mitsubishi PLC
- Hitachi PLC
- Delta PLC
- General Electric (GE) PLC
- Honeywell PLC

Każdy z nich posiada własne, specyficzne funkcje, zalety oraz dedykowane oprogramowanie programistyczne.

Zastosowania PLC (PLC Applications)

PLC znajdują zastosowanie m.in. w:

- Automatyce procesowej (np. górnictwo, ropa i gaz)
- Przemyśle szklarskim
- Przemyśle papierniczym
- Produkcji cementu
- Kotłowniach – elektrownie cieplne

Programowanie PLC (PLC Programming)

Podczas pracy ze sterownikiem PLC ważne jest projektowanie i implementacja koncepcji opartych na konkretnym zastosowaniu. Aby to zrobić, musimy poznać szczegóły dotyczące programowania PLC.

Program PLC składa się z zestawu instrukcji tekstowych lub graficznych, które reprezentują logikę sterującą procesem.

Główne klasyfikacje języków programowania PLC

Języki tekstowe

- Instruction List (lista instrukcji)
- Structured Text (tekst strukturalny)

Forma graficzna

- Ladder Diagrams (LD) – logika drabinkowa
- Function Block Diagram (FBD)
- Sequential Function Chart (SFC)

Chociaż wszystkie te języki mogą być stosowane do programowania PLC, to najczęściej używane są graficzne (np. ladder logic), ponieważ są łatwiejsze do zrozumienia niż tekstowe.

Ladder Logic

https://www.controlbyte.pl/jak-zaczac-nauke-programowania-plc-zacznij-od-jezyka-lad/

https://home.agh.edu.pl/~aprzem/pliki/plc_2.pdf

https://www.astor.com.pl/poradnikautomatyka/jak-korzystac-ze-stykow-cewek-i-blokow-operacji-arytmetycznych-w-programie-sterujacym-kurs-plc-4/

Zapoznaj się z następującymi artykułami:

- https://www.plcacademy.com/ladder-logic-tutorial/
- https://www.plcacademy.com/ladder-logic-tutorial-part-2/

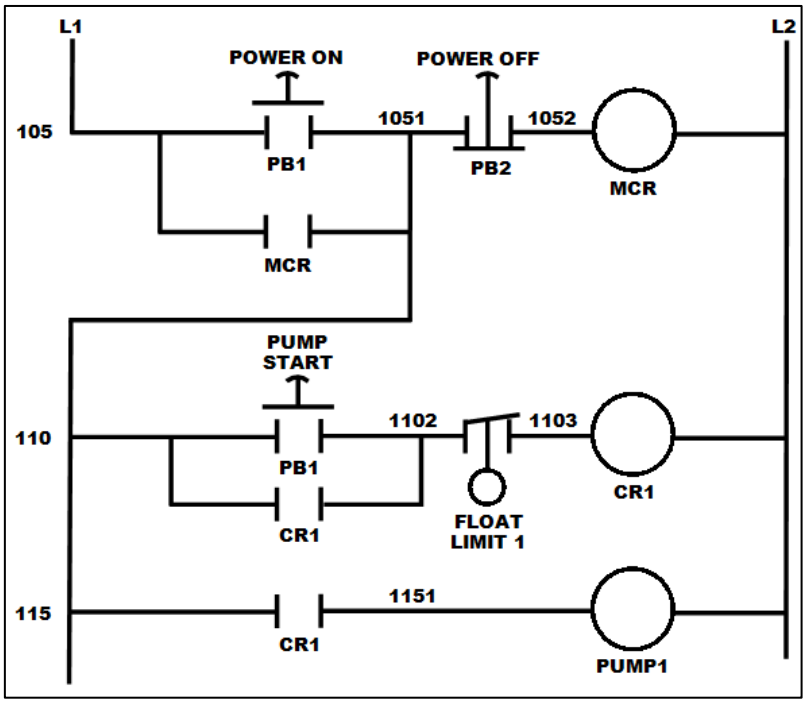
Język Drabinkowy/Logika Drabinkowa (LD lub LAD) (ang. Ladder logic (LD or LAD)) – Graficzny język programowania sterowników PLC, pierwotnie był pisemną metodą dokumentacji sterowania przekaźnikowego, stosowanego w produkcji i kontroli procesu. Każde urządzenie w szafie przekaźnikowej było reprezentowane przez symbol na schemacie drabinkowym razem z połączeniami między pokazanymi urządzeniami. Jako język graficzny jest on dla przeciętnego programisty prostszy oraz łatwiejszy w opanowaniu niż przeciętny klasyczny język programowania sterowników.

Ponadto na schemacie drabinkowym zostaną również pokazane inne elementy znajdujące się poza układem przekaźnikowym, takie jak:

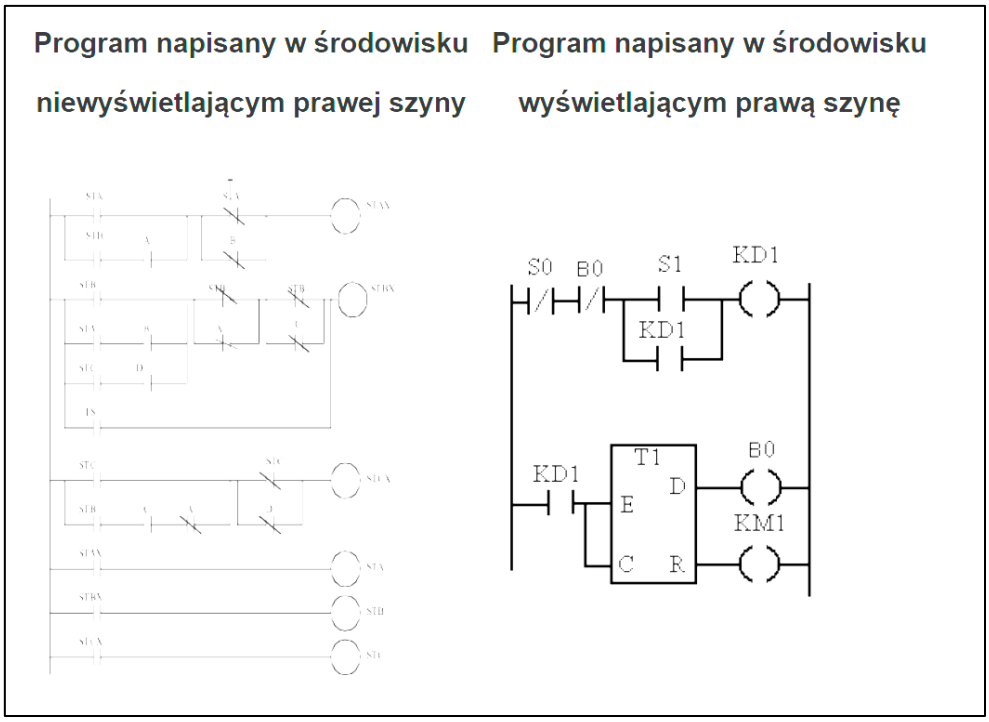
- Elementy wykonawcze: pompy, grzejniki, silniki itp.
- Detektory najczęściej: czujniki laserowe, czujniki ciśnienia, czujniki światła oraz czujniki chemiczne itp.

Logika drabinkowa przekształciła się w język programowania, który reprezentuje zapisany program za pomocą symboli logiki przekaźnikowej. Zasady tworzenia programów w języku LD są zawarte w normie IEC 61131-3.

Nazwa wzięła się od wyglądu programu, który dwiema pionowymi szynami i szeregiem poziomych szczebli pomiędzy przypomina drabinę.



Program tworzy się poprzez umieszczanie symboli na schemacie oraz łączenie ich. Sygnał można interpretować jako prąd płynący od lewej szyny do prawej, w niektórych programach niewystępującej. Różnice wynikają z przyjętych przez programistów sposobów działania środowisk programistycznych. Programowanie w języku LAD polega na umieszczaniu pomiędzy „nogami drabiny" ikon symbolizujących rzeczywiste lub istniejące w pamięci sterownika.



Podstawowymi symbolami są styki oraz cewki:

Styki - służą jako wejścia w programie, sprawdzają czy zmienna ma wartość 0 lub 1.

-[ ]- Styk normalnie otwarty, bez sygnału sterującego tworzy przerwę w obwodzie, gdy pojawi się sygnał łączy swoje krańce

-[/]- Styk normalnie zamknięty, bez sygnału łączy swoje krańce, gdy pojawi się sygnał tworzy przerwę w obwodzie.

Cewki - Służą do manipulowania zmiennymi,

-( )- Cewka normalnie otwarta, gdy pojawi się na niej sygnał nadaje przypisanej zmiennej wartość 1; gdy nie pojawia się na niej sygnał, cewka przestaje nadawać wartość zmiennej.

-(/)- Cewka normalnie zamknięta, gdy nie pojawia się na niej sygnał, nadaje przypisanej zmiennej wartość 1; gdy dociera do niej sygnał, cewka przestaje nadawać wartość zmiennej.

Logiczne NOT [edytuj | edytuj kod]

```
   Przycisk01               Lampka01
------[/]---------------------( )
```

Powyższy program realizuje funkcję logiczna NOT poprzez negację sygnału wejścia.

```
   Przycisk01               Lampka01
------[ ]---------------------(/)
```

Powyższy program realizuje funkcję logiczna NOT poprzez negację sygnału wyjścia.

Oba powyższe programy działają w ten sam sposób.

Gdy Przycisk01 jest wyłączony, lampka01 świeci.

Gdy Przycisk01 jest włączony, lampka01 nie świeci.

Logiczne AND [edytuj | edytuj kod]

```
|   Przycisk01      Przycisk02          Lampka01
|------[ ]--------------[ ]----------------( )
|
|
```

Powyższy program realizuje funkcję:

Gdy Przycisk01 jest włączony oraz przycisk02 jest włączony, to wtedy Lampka01 jest włączona.

Logiczne OR [edytuj | edytuj kod]

```
|        Przycisk01               Lampka01
|-----+------[ ]------+-------------( )
|     |  Przycisk02   |
|     +------[ ]------+
|
```

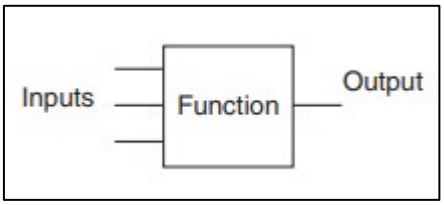
Powyższy program realizuje funkcję:

Gdy Przycisk01 jest włączony lub przycisk02 jest włączony, to wtedy Lampka01 jest włączona.

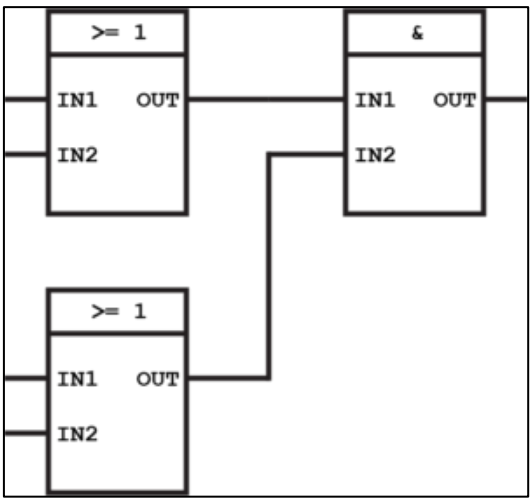
Functional Block Diagrams

FBD to prosta i graficzna metoda programowania wielu funkcji w PLC. PLCOpen opisało użycie FBD w normie IEC 61131-3. Blok funkcji to instrukcja programu, która podczas wykonania generuje jedną lub więcej wartości wyjściowych.

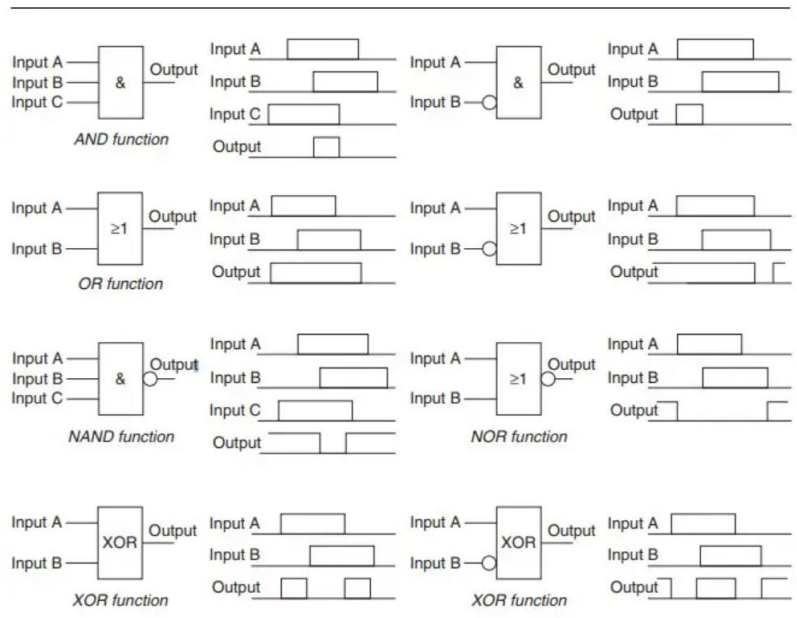
Blok przedstawia się jako prostokąt z wejściami po lewej stronie i wyjściami po prawej. Pokazuje zależność między stanem wejść i wyjść.



Zaletą użycia FBD jest możliwość stosowania dowolnej liczby wejść i wyjść. W przypadku wielu bloków można podłączać wyjście jednego bloku do wejścia kolejnego, tworząc pełny schemat FBD.

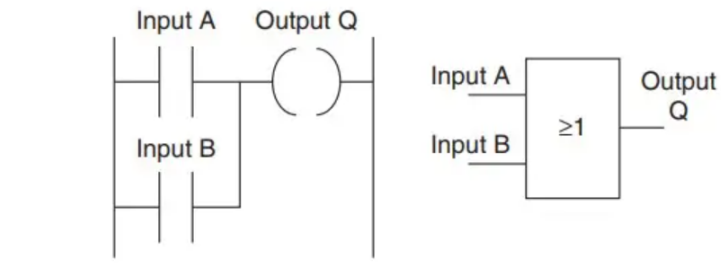


The figure below shows various function blocks used in FBD programming.

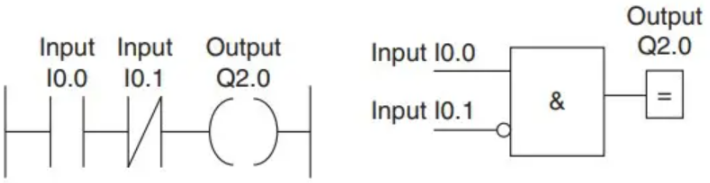


Functional Block Programming

The figure below shows a ladder diagram and its function block equivalent in Siemens notation.



Ladder to functional block [Source]



Ladder to functional block diagram [Source]

Structured Text Programming

Structured Text to język programowania PLC wykorzystujący proste instrukcje do określania operacji. Jest podobny do konwencjonalnego programowania, lecz nie rozróżnia wielkości liter. Wykorzystuje operatory do wyrażania logiki i relacji.

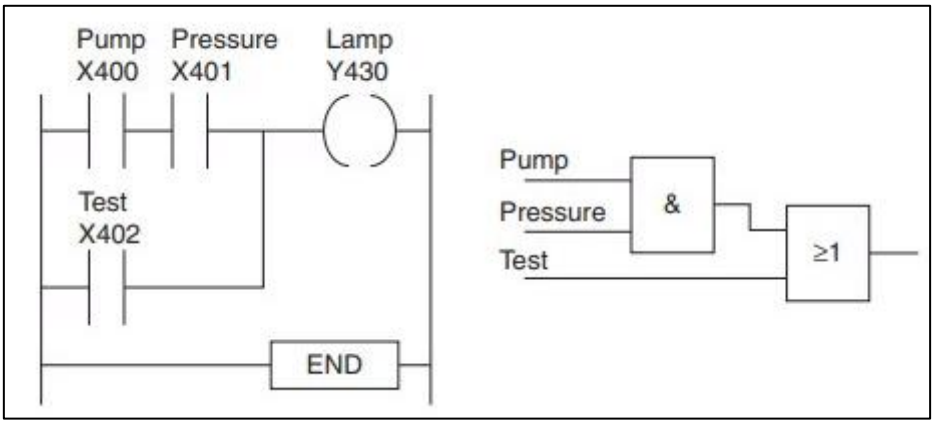
Order	Operation
1.	( )
2.	function (...)
3.	**
4.	- (negate)
5.	NOT
6.	*, /, MOD
7.	+, - (subtract)
8.	<, <=, >, >=
9.	=, <>
10	&, AND
11.	XOR
12.	OR

Przykłady programów PLC

Sterowanie lampą

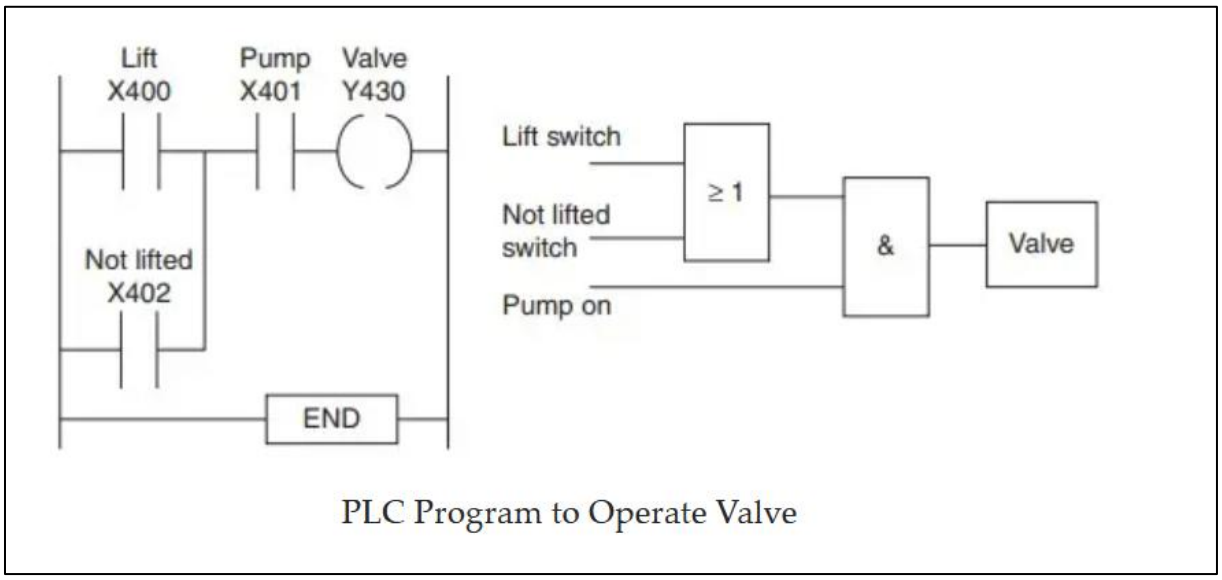
Lampa sygnalizacyjna ma zostać włączona, jeśli pompa pracuje i ciśnienie jest prawidłowe, lub jeśli przełącznik testowy jest zamknięty. W tej aplikacji, aby uzyskać wyjście lampy, wymagane są wejścia zarówno z pompy, jak i z czujnika ciśnienia, dlatego stosuje się bramkę AND.

Logika OR jest używana dla wejścia testowego lampa ma się włączyć, niezależnie od sygnału z systemu AND. Za pomocą instrukcji END lub RET w logice drabinkowej informujemy PLC o końcu programu.



Sterowanie zaworem

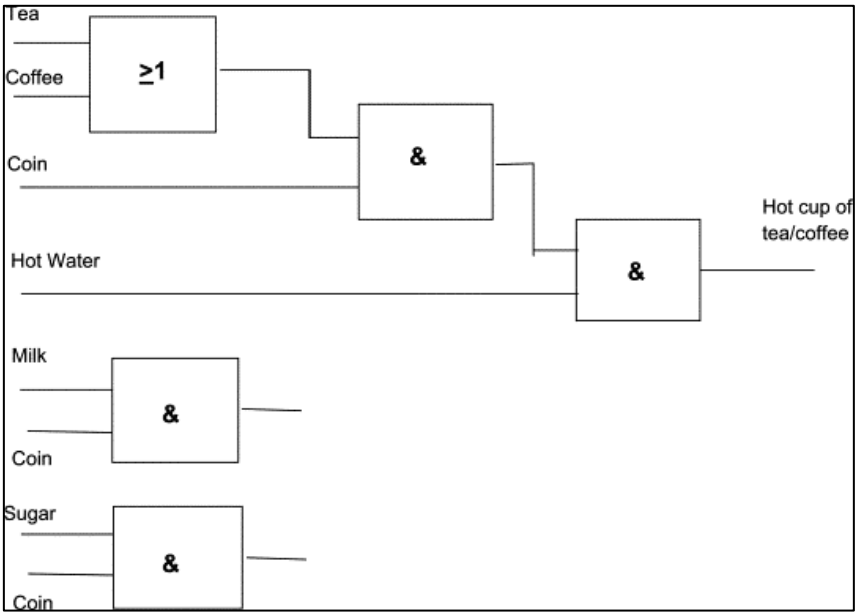
Rozważmy zawór, który ma być uruchomiony w celu podniesienia ładunku, gdy pompa pracuje i albo przełącznik podnoszenia jest aktywowany, albo przełącznik wskazuje, że ładunek nie został jeszcze podniesiony i znajduje się na dole kanału.

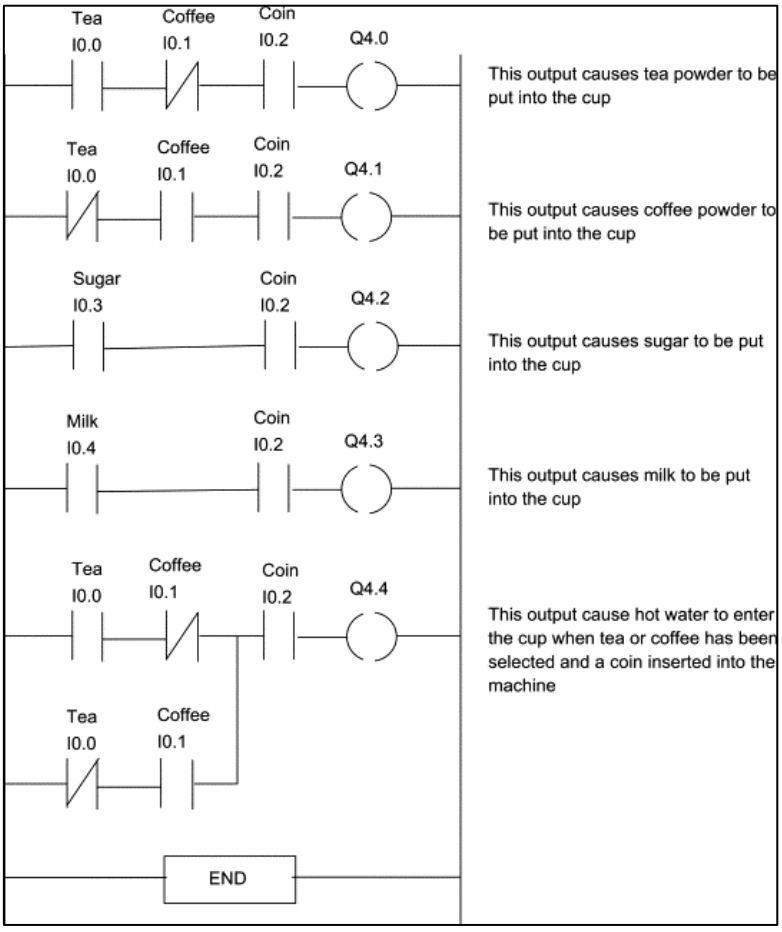


Automat vendingowy - napoje

Rozważmy automat do napojów, który umożliwia wybór herbaty lub kawy, mleka lub bez mleka, cukru lub bez cukru, i wyda napój po wrzuceniu monety.

Wyboru herbaty lub kawy dokonuje się za pomocą pierwszej bramki OR. Pierwsza bramka AND generuje wyjście, gdy wybrano herbatę lub kawę i wrzucono monetę. Jej wyjście przechodzi do drugiej bramki AND, która działa tylko gdy gorąca woda łączy się z herbatą. Mleko i cukier mogą być dodawane opcjonalnie po wrzuceniu monety.





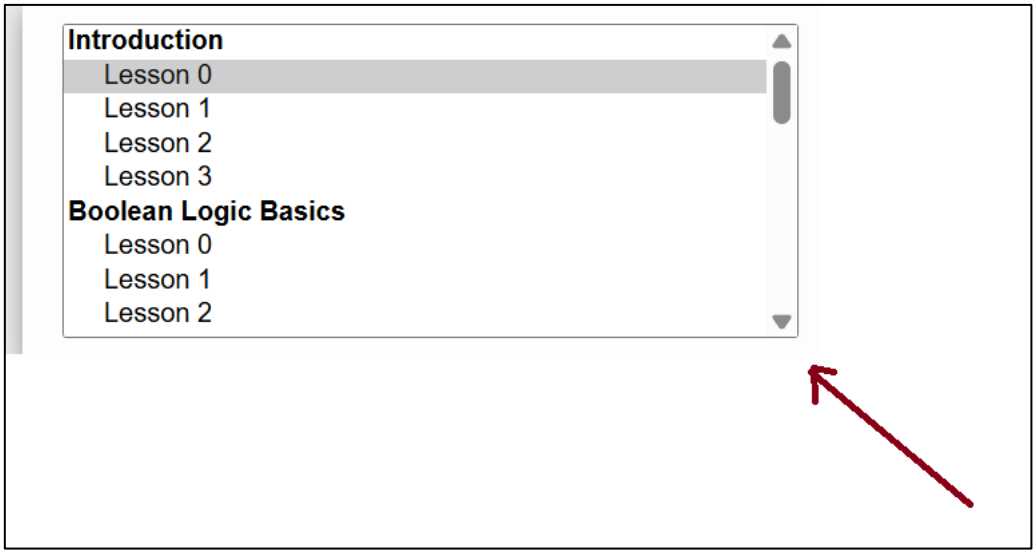
Materiały dodatkowe:

- https://www.plcacademy.com/function-block-diagram-programming/
- https://www.youtube.com/watch?v=-WIn84DKtRw&list=PLI78ZBihrkE1yvv_jpdsF24RpDF1Pye-z
- https://home.agh.edu.pl/~aprzem/pliki/plc_1.pdf
- https://home.agh.edu.pl/~aprzem/pliki/plc_2.pdf
- https://esezam.okno.pw.edu.pl/course/view.php?id=11#section-0

Zadania praktyczne

Zadanie 1

Wykonać zadania ze strony: https://www.plcfiddle.com/code-school



Zadanie 2 - Jedna lampka sterowana jednym przyciskiem

Uruchom sandbox plc na stronie: https://www.plcfiddle.com/

https://www.youtube.com/watch?v=Cgd_Or1Mcac

Napisz program, który:

- Gdy przycisk Start jest włączony (ON) → lampka Lampka świeci (ON).
- Gdy puścisz przycisk (OFF) → lampka gaśnie (OFF).

Czyli lampka świeci tylko wtedy, gdy trzymasz przycisk.

Zadanie 3 - Jedna lampka, dwa przyciski – wariant AND i OR

Pokazać różnicę między połączeniem szeregowym (AND) a równoległym (OR) kontaktów.

Dodaj:

- Wejścia: Przycisk1, Przycisk2 (BOOL)
- Wyjście: Lampka (BOOL)

Najpierw ustaw przyciski szeregowo a następnie równolegle i zaobserwuj różnice i zastanów się nad wykorzystaniem.

Zadanie 4 - Prosty układ Start/Stop z podtrzymaniem (zatrzask)

Zaprogramować:

- kontakt normalnie zamknięty (NC),
- samopodtrzymanie – klasyczny przerzutnik Start/Stop.

Wyobrażamy sobie silnik:

- przycisk Start – chwilowy, włącza silnik, ale po puszczeniu silnik ma dalej pracować;
- przycisk Stop – chwilowy, zatrzymuje silnik;
- wyjście Silnik – reprezentuje pracę silnika (na symulatorze będzie to zwykła lampka/cewka).

Działanie:

1. Naciśniesz Start - Silnik włącza się.
2. Puścisz Start - Silnik dalej pracuje (program go „trzyma").
3. Naciśniesz Stop - Silnik wyłącza się.

Zadanie 5 - Lampka świeci, gdy przycisk nie jest wciśnięty (kontakt NC)

Napisz program, który:

- Gdy przycisk Przycisk NIE jest wciśnięty (OFF) → Lampka świeci (ON).
- Gdy przycisk Przycisk jest wciśnięty (ON) → Lampka gaśnie (OFF).

Czyli odwrotnie niż w Zadaniu 1.
To typowe zachowanie np. dla „czujnika awarii”: świeci, gdy wszystko OK, gaśnie/gasi się gdy coś się dzieje.

Zadanie 6 - Lampka włącza się po czasie wciśnięcia przycisku (pierwszy timer TON)

https://www.youtube.com/watch?v=b4Kz_k7ZkLw

Napisz program, który:

- Gdy wciśniesz i trzymasz przycisk Start, po 3 sekundach zapala się Lampka.
- Jeśli puścisz Start przed upływem 3 sekund, Lampka w ogóle się nie zapali.
- Gdy puścisz Start po tym, jak lampka już się zapali, Lampka znów gaśnie.

Czyli: lampka zapala się z opóźnieniem, ale tylko jeśli przycisk jest trzymany odpowiednio długo.

Zadanie 7 – Dodatkowe

Zapoznaj się z zadaniem: https://arkusze.pl/zawodowy/e19-2025-czerwiec-egzamin-zawodowy-praktyczny.pdf

Zapoznaj się z zadaniami i materiałami ze strony: https://plc-coep.vlabs.ac.in/