Sterowniki PLC

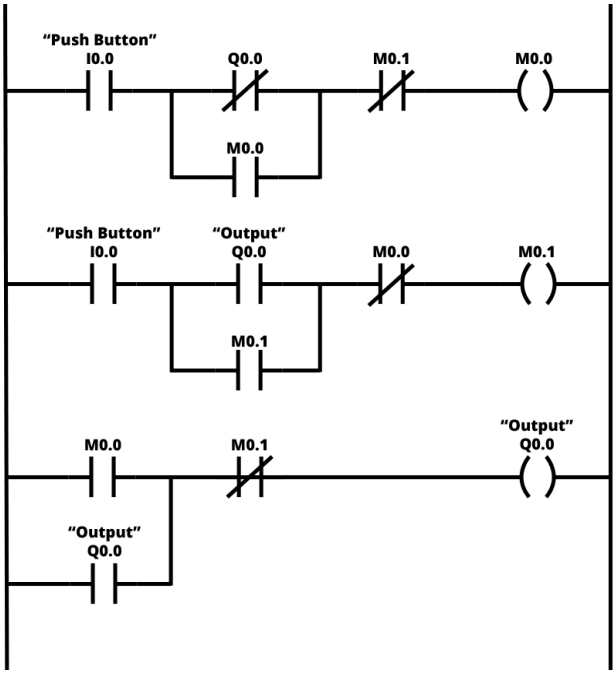
Programowalny sterownik logiczny, PLC (od ang. programmable logic controller) – uniwersalne urządzenie mikroprocesorowe przeznaczone do sterowania pracą maszyny lub urządzenia technologicznego. PLC musi zostać dopasowany do określonego obiektu sterowania poprzez wprowadzenie do jego pamięci żądanego algorytmu działania obiektu. Cechą charakterystyczną programowalnych sterowników logicznych, odróżniającą je od innych sterowników komputerowych, jest cykliczny obieg pamięci programu.

Algorytm jest zapisywany w przeznaczonym dla sterownika 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.

Sposoby (środowiska) programowania PLC:

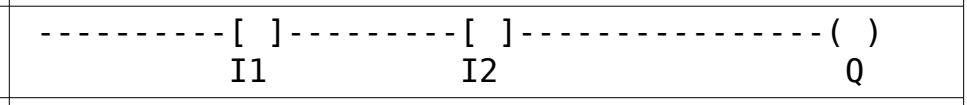
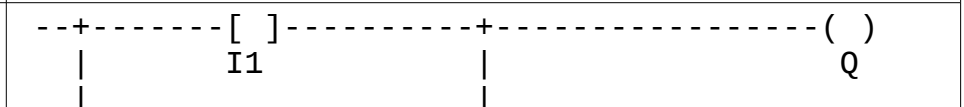
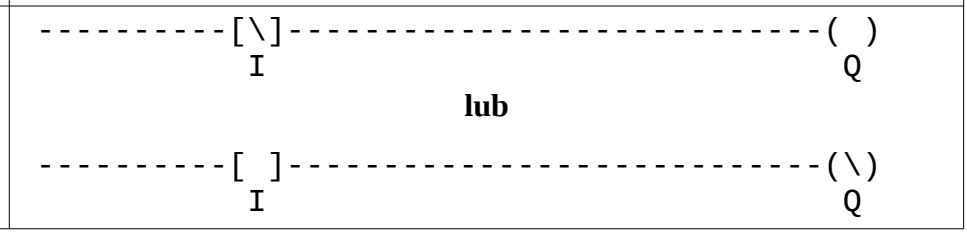
- **LD (ladder diagram) logika drabinkowa** – schemat zbliżony do klasycznego rysunku technicznego elektrycznego
- FBD (function block diagram) – diagram bloków funkcyjnych, sekwencja linii zawierających bloki funkcyjne
- ST (structured text) tekst strukturalny – język zbliżony do Pascala
- IL (instruction list) lista instrukcji – rodzaj asemblera
- SFC (sequential function chart) sekwencyjny ciąg bloków – sekwencja bloków programowych z warunkami przejścia.

Przykładowy program (pojedynczy przycisk włącz/wyłącz)



Logika drabinkowa

Oznaczenie	Rodzaj	Opis
I	**I**nput (wejście)	W każdym sterowniku PLC mają takie samo oznaczenie, mogą być przypisywane **tylko** do symboli styków informują o stanie wejść na sterowniku. `--[ ]--` Wejście normalne (NO) `--[/]--` Wejście odwrócone (NC)
Q	Output (wyjście)	W każdym sterowniku PLC mają takie samo oznaczenie, mogą być przypisywane **zarówno** do symboli cewek (wtedy ustawiają konkretne wyjście sterownika) jak i styków gdzie informują o stanie wyjść. `--( )--` Wyjście normalne (NO) `--(/)--` Wyjście odwrócone (NC)
M	**M**arker	Inaczej zmienna wewnętrzna. Tym symbolem określa się zmienne wewnętrzne sterownika, wykorzystywane są jako cewki i styki. elementy pośrednie programu.

Bramka logiczna	Przykład
AND	
OR	
NOT	

Pierwsze zadanie na dziś

Przy użyciu logiki drabinkowej napisać program obsługi wózka w taki sposób aby:
wózek po wciśnięciu przycisku I0.0 rozpoczynał podróż w prawo (Q0).
Po dojechaniu do krańcówki prawej (I1.0) wózek zmienia kierunek podróży w lewo (Q1).
Po dojechaniu do krańcówki lewej (I1.1) wózek kończy bieg.

Dodatkowe przyciski: I0.1 – zatrzymuje wózek. I0.2 – wymuszający powrót wózka.

Podstawy programowania mikroprocesorów (C++)

C++ – język programowania ogólnego przeznaczenia.

Język wysokiego poziomu (autokod) – typ języka programowania, którego składnia i słowa kluczowe mają maksymalnie ułatwić rozumienie kodu programu dla człowieka, tym samym zwiększając poziom abstrakcji i dystansując się od sprzętowych niuansów.

Pseudokodem nazywany jest taki sposób zapisu algorytmu, który zachowując strukturę charakterystyczną dla kodu zapisanego w języku programowania ale rezygnuje ze ścisłych reguł składniowych na rzecz prostoty i czytelności.

Pseudokod

```
Powtarzaj bez końca:
      Włącz diodę
      Poczekaj 1 s. (1000 ms.)
      Wyłącz diodę
      Poczekaj 1 s. (1000 ms.)
```

C++ (Arduino)

```
while(1){
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Arduino składa się z 8-bitowego mikrokontrolera Atmel AVR z uzupełniającymi elementami w celu ułatwienia programowania oraz włączenia innych układów. **Mikrokontroler**, mikrokomputer jednoukładowy to scalony system mikroprocesorowy, zrealizowany w postaci pojedynczego układu zawierającego procesor, pamięć RAM, układy wejścia-wyjścia i na ogół pamięć programu.

Przykładowa struktura programu Arduino:

```
                              // - komentarze
int dioda = 13;               // definicja zmiennej (globalnej)
void setup() {                //funkcja wykonywana raz, po włączeniu zasilania
  pinMode(dioda, OUTPUT);     //ustawia pin nr 13 jako wyjście
}

void loop() {                 //funkcja wykonywana w kółko
  digitalWrite(dioda, HIGH);  //ustaw na 13 wyjściu 5V (logiczne 1)
  delay(1000);                //czekaj 1000 ms. = 1 s.
  digitalWrite(dioda, LOW);   //ustaw na 13 wyjściu 0V (logiczne 0)
  delay(1000);                //czekaj 1 s.
}
```

Drugie zadanie na dziś

Napisać pseudokod sygnalizujący za pomocą diody czy zostały wciśnięte odpowiednie trzy przyciski z sześciu.

Wybrane informacje (programowanie w Arduino)

Typy zmiennych

bool	zmienna przyjmująca wartość prawda (true, high, 1) lub fałsz (false, low, 0)
int	zmienna przyjmująca wartość od -32768 do 32767

Funkcje wejścia i wyjścia

digitalRead(10) - sprawdza czy pin nr 10 podłączony jest do 5V czy do 0V.
digitalWrite(13, *stan*) – podaje na pin 13 *stan* (wysoki lub niski). *Stan* na wyjściu pozostaje podtrzymany do następnego wykonania funkcji na tym samym pinie

Pętle

- while(*warunek*){ *polecenie; }*
- for(deklaracja; warunek; deklaracja2){ *polecenie; }*

np. for(int i = 0; i < 10; i++){
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}

Instrukcja warunkowa *if*

if(*warunek*){
polecenie-jeśli-prawda
} //dalsza część jest opcjonalna
else{
polecenie-jeśli-nieprawda
}

Warunki

a > b lub b < a	*a* większe od *b*
a >= b	*a* większe bądź równe *b*
a == b	*a* równe *b (ważne, podwójne równa się!)*
a != b	*a* nie-równe b

Działania

+, -, *, /	jak w arytmetyce
i++ (i--), ++i (--i)	zwiększ(zmniejsz) zmienną *i* o jeden
i = 10	przypisz zmiennej *i* wartość 10

Tryby pinMode(*pin, tryb*)

INPUT	wejście
OUTPUT	wyjście

Trzecie zadanie na dziś

Zrealizować zadanie drugie, a następnie pierwsze (na kartce) w formie programu w C++.

Zdeklarować numery (od 2 do 12) przycisków oraz wyjść (diody, ruch silnika) jako zmienne globalne. W funkcji *setup()* ustawić je jako wejścia oraz wyjścia. W funkcji *loop()* umieścić odpowiednie funkcje (pętle, warunki, odczyt wartości).

Ruch silnika w lewo określić jako:

*digitalWrite(*silnik1, *LOW);*
*digitalWrite(*silnik2, *HIGH);*

Ruch silnika w prawo

*digitalWrite(*silnik1, *HIGH);*
*digitalWrite(*silnik2, *LOW);*

Zatrzymanie silnika

*digitalWrite(*silnik1, *LOW);*
*digitalWrite(*silnik2, *LOW);*

https://www.arduino.cc/reference/en/
https://pl.wikipedia.org/wiki/Programowalny_sterownik_logiczny
http://www.plcacademy.com/ladder-logic-examples/?epik=0LyiXE_IWX-V-
http://home.agh.edu.pl/~aprzem/pliki/plc_1.pdf