KATEDRA AUTOMATYKI, BIOMECHANIKI
I MECHATRONIKI

LABORATORIUM AUTOMATYKI

TEMAT:

Sterowanie sekwencyjne silnika z krzywką – układ Moore'a

Aktualizacja: 05.04.2024 r.

Opracował: dr inż. Krystian Polczyński
dr inż. Adam Wijata

wersja: 1.0

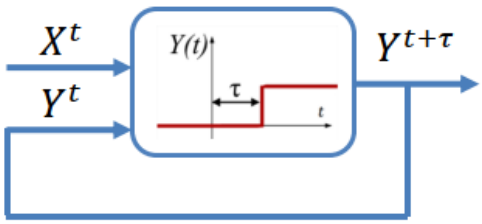
Cel ćwiczenia

Celem ćwiczenia jest opracowanie programu napisanego w języku LAD sterującego silnikiem DC z krzywką według założonego scenariusza. Program sterujący opracowany zostanie przy wykorzystaniu teorii automatu Moore'a.

1 Automaty skończone

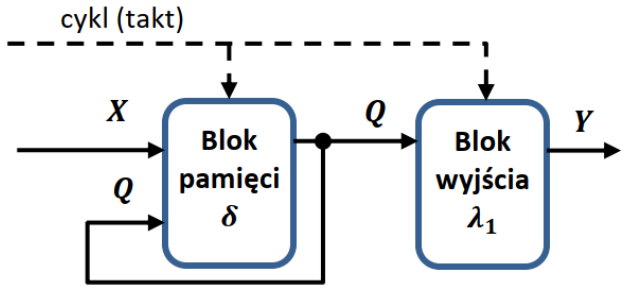
Układy logiczne, w których stan wyjść układu $\boldsymbol{Y} = (y_1, y_2, \ldots)$ zależy jedynie od aktualnego stanu (chwilowej **kombinacji**) wejść układu $\boldsymbol{X} = (x_1, x_2, \ldots)$ nazywane są układami **kombinacyjnymi**. W technice występują sytuacje, w których znajomość kombinacji wejść może być niewystarczająca do określenia wyjścia. Za przykład niech posłuży nam automat do napojów i przekąsek[1]. Jeżeli użytkownik ma ochotę na przekąskę umieszczoną pod numerem „12", to wciśnie na klawiaturze przyciski z numerami „1" i „2". Jednak mając ochotę na napój spod numeru „21" użytkownik wciska te same dwa przyciski tylko w odwrotnej kolejności – najpierw „2", a później „1". Oznacza to, że do wyboru odpowiedniego produktu (określenia wyjścia układu logicznego) potrzebna jest znajomość nie tylko kombinacji przycisków (wejść układu) ale też **sekwencja** ich zmiany. Układy logiczne realizujące takie zadania nazywane są **sekwencyjnymi**. Uwzględnienie sekwencji (historii zdarzeń) zachodzącej na wejściach układu jest możliwe dzięki wprowadzeniu do układu **pamięci**. W tej pamięci nie przechowuje się wszystkich zmian zachodzących na wejściach układu, ale jedynie tzw. **wewnętrzny stan** układu logicznego.

Aby zrozumieć podstawowy mechanizm działania pamięci, w analizie działania układów logicznym musimy uwzględnić **czas**. Fizycznie wykonany układ logiczny (nie zależnie od technologii wykonania) będzie „wykonywać obliczenia" przez jakiś skończony czas, tzn. od momentu pojawienia się nowej kombinacji wejść do pojawienia się nowego stanu wyjść upłynie czas τ. Dlatego, każdy element układu logicznego możemy traktować jako człon wprowadzający opóźnienie. Podstawowym sposobem na wprowadzenie pamięci do układu jest wykorzystanie sprzężenia zwrotnego sygnału wyjściowego, jak pokazano to na schemacie na rys. 1. Jeżeli w takim układzie w chwili t, pojawi się nowa kombinacja sygnałów wejściowych $\boldsymbol{X}^t$, to nowy stan wyjść $\boldsymbol{Y}^{t+\tau}$ pojawi się dopiero po upływie czasu opóźnienia (czasu obliczeń) w chwili czasowej $t + \tau$. Do tego momentu układ „pamięta" stan wyjść z chwili początkowej $\boldsymbol{Y}^t$ (i zostaje on wykorzystany w obliczeniach).



Rys. 1. Elementarny układ sekwencyjny

Przy projektowaniu układów logicznych interesuje nas głównie ich stan ustalony, tzn. po upływnie opóźnienia τ. Dlatego układy sekwencyjne analizuje się w **czasie dyskretnym**, czyli w chwilach czasowych po kolejnych opóźnieniach nazywanych **taktami**. Czas opóźnienia upływający pomiędzy kolejnymi taktami może mieć różne wartości i wynikać ze skumulowanego opóźnienia jego elementów składowych (układy asynchroniczne) lub być regulowany przez zewnętrzy układ zegarowy układy synchroniczne). Wartość tego czasu nie jest istotna w naszych rozważaniach, dlatego stan układu w danym takcie będziemy oznaczać indeksem t ($\boldsymbol{X}^t$, $\boldsymbol{Y}^t$), w takcie następnym indeksem $t + 1$ ($\boldsymbol{X}^{t+1}$, $\boldsymbol{Y}^{t+1}$), a w poprzednim $t - 1$ ($\boldsymbol{X}^{t-1}$, $\boldsymbol{Y}^{t-1}$). W najprostszej wersji, funkcje lo-



Rys. 2. Struktury układu sekwencyjnego automatu Moore'a.

[1] Współczesne automaty do przekąsek mają układy sterowania programowane na uniwersalnych sterownikach lub innych układach mikroprocesorowych, prawdopodobnie z wykorzystaniem programowej instrukcji wielokrotnego wyboru „switch". Przykład ten proszę traktować jako uproszczony, o dużym stopniu ogólności.

giczne układów sekwencyjnych można syntezować (wyprowadzać) stosując metody dla układów kombinacyjnych, przyjmując zmienne wyjściowe (lub określające wewnętrzny stan układu) $\boldsymbol{Y}^{t+1}$ jako wyjście układu, a $\boldsymbol{Y}^{t}$ jako wejście. W ten sposób tworząc niezbędne sprzężenie zwrotne. Jednak sprzężenie zwrotne sprawia, że stany przejściowe układu (które były pomijane przy analizie układów kombinacyjnych) mogą wywoływać nieporządne efekty, powodujące nieprawidłowe działanie układu. Dlatego synteza układów sekwencyjnych wymaga większej uważności. Podstawowe struktury wg. jakich projektuje się układy sekwencyjne to tzw. automaty. Podstawowym automatem jest automat Moore'a, a jego schemat blokowy jest przedstawiony na rys. 2.

Na powyższych schematach wektor wejść układu jest opisany jako $\boldsymbol{X}$ = (x_1, x_2, …,), wektor wyjść $\boldsymbol{Y}$ = (y_1, y_2, …), a wektor reprezentujący wewnętrzy stan układu (stan pamięci) to $\boldsymbol{Q}$ = (q_1, q_2, …). To właśnie zmienne (q_1, q_2, …) są przekazywane w sprzężeniu zwrotnym realizując pamięć układu. Wracając do przykładu automatu z przekąskami, w stanie pamięci może być przechowywana informacja o tym ile i które przyciski zostały już wybrane, a stan wyjść będzie uruchamiał silnik na odpowiedniej półce.

W logice działania obu automatów możemy wyróżnić dwa etapy: określenie nowego stanu automatu oraz określenie na jego podstawie nowego stanu wyjść. Każdy z tych etapów jest realizowany przez układ kombinacyjny, odpowiednio w bloku pamięci lub bloku wyjścia. Nowy stan pamięci układu $\boldsymbol{Q}^{t+1}$ jest obliczany w bloku pamięci, a obliczenia te przebiegają zgodnie z funkcją przejść δ, na podstawie aktualnego stanu wejść $\boldsymbol{X}^{t}$ oraz aktualnego stanu pamięci $\boldsymbol{Q}^{t}$

$$\boldsymbol{Q}^{t+1} = \delta(\boldsymbol{Q}^{t}, \boldsymbol{X}^{t}). \tag{1}$$

Następnie w bloku wyjścia, obliczany jest stan zmiennych wyjściowych $\boldsymbol{Y}^{t}$ zgodnie z funkcją wyjść λ. Tutaj pojawia się różnica pomiędzy opisywanymi automatami. W automacie Moore'a, argumentem funkcji wyjść jest tylko stan wewnętrzny układu

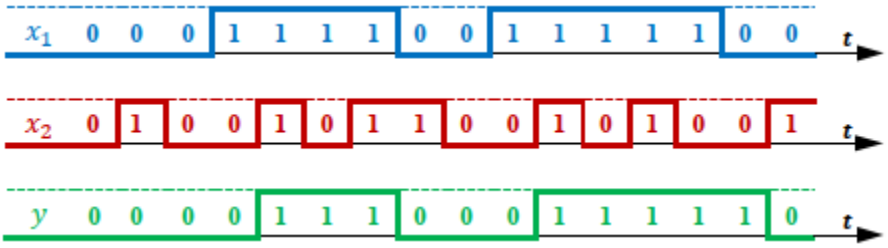
$$\boldsymbol{Y}^{t} = \lambda_1(\boldsymbol{Q}^{t}). \tag{2}$$

Na powyższym schemacie (rys. 2) występuje jeszcze sygnał cyklu (taktu). Nie jest on elementem koniecznym. Jest to sygnał zegarowy (ciąg impulsów prostokątnych) wyznaczający chwile czasowe, w których do układu wpuszczane są nowe stany wejść oraz w których wystawiane są nowe stany wyjść. Układy posiadające taki sygnał nazywane są **synchronicznymi**, natomiast układy bez tego sygnału to układy **asynchroniczne**. Podstawową różnicą pomiędzy tymi układami jest to, że w układach asynchronicznych sygnał o stanie wysokim (1) lub niskim (0) uważany jest za jeden sygnał, bez względu na czas trwania. Natomiast w układach synchronicznych sygnał trwający n taktów sygnału zegarowego, jest uważany za n kolejnych sygnałów.

Najpowszechniejszym sposobem opisu układów sekwencyjnych jest **opis słowny**, przyporządkowujący stany zmiennych wyjściowych do sekwencji stanów wejściowych. Opis ten, w zależności od skomplikowania układu może być zwięzły lub bardzo rozbudowany. Trudno jednak na podstawie samego opisu słownego napisać funkcję logiczną, dlatego opis słowny zazwyczaj wykorzystuje się do opracowania kolejnych opisów układu, bardziej użytecznych przy syntezie funkcji logicznych.

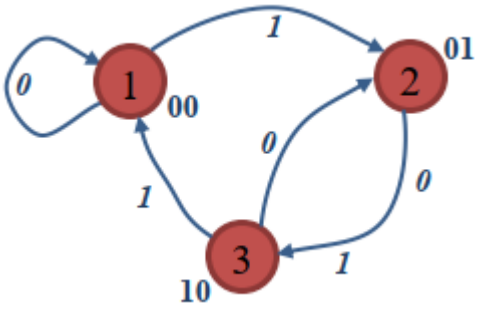
Kolejnym sposobem przedstawienia sposobu działania układu są **ciągi zero-jedynkowe** i **wykresy czasowe**. Przedstawiają one typową sekwencję działania układu, przypisując kolejnym stanom wejściowym, odpowiednie stany wyjść. Przykład takiego opisu przedstawia rys. 3. Jak widać, na takich wykresach oś czasu zazwyczaj nie jest wyskalowana, a sygnały przyjmują tylko wartości 1 lub 0, z pominięciem wszelkich stanów przejściowych.

Opis słowny i wykresy czasowe opisują zależności pomiędzy wejściami $\boldsymbol{X}$ i wyjściami układu $\boldsymbol{Y}$. Jednak aby zaprojektować układ w strukturze Moore'a potrzebujemy znać informacje o stanie pamięci $\boldsymbol{Q}$ oraz o funkcjach przejść δ i wyjść λ. Stosowane są dwie podstawowe metody opisów układów uwzględniające te informacje – graficzna i tabelaryczna.



Rys. 3. Wykres czasowy i ciągi zero-jedynkowe opisujące układ sekwencyjny.

Metoda graficzna polega na wykreśleniu grafu układu. Wierzchołki takiego grafu reprezentują stany pamięci $\boldsymbol{Q}$, natomiast gałęzie grafu reprezentują stan wejść $\boldsymbol{X}$ powodujące przejście pomiędzy dwoma stanami. Gałąź grafu jest skierowana od jednego stanu, do stanu w którym układ ma się znaleźć po pojawieniu się stanu wejść $\boldsymbol{X}$, co opisuje funkcję przejść δ. W automacie Moore'a stan wyjść $\boldsymbol{Y}$ zależy tylko od stanu pamięci układu, dlatego wpisuje się je przy wierzchołkach grafu. Przykładowy graf dla automatu Moore'a przedstawiony jest na rys. 4. Stany wyjść określają tam dwie zmienne binarne, tzn. $\boldsymbol{Y} = (y_1, y_2)$, a stan wejść jedna - $\boldsymbol{X} = (x)$. W układzie przyjęto trzy stany pamięci: $\boldsymbol{Q}_1$, $\boldsymbol{Q}_2$, $\boldsymbol{Q}_3$, które na grafie zaznaczono po prostu jako 1, 2, 3.



Rys. 4. Graf dla układu Moore'a.

Tabelarycznym odpowiednikiem grafu układu są tablice przejść i wyjść. Są one mniej obrazowe, ale za to zdecydowanie wygodniejsze przy przekształceniach i wyprowadzaniu zminimalizowanych funkcji logicznych. Zgodnie z nazwą, tablica przejść opisuje funkcję przejść δ. Zgodnie z równaniem (1), każdej parze ($\boldsymbol{Q}^t$, $\boldsymbol{X}^t$), przypisuje nowy stan pamięci $\boldsymbol{Q}^{t+1}$. Taką tablice dla automatu Moore'a zadanego grafem z rys. 4 przedstawia tablica z rys. 5.

Q^t \ X^t	0	1	Y^t
1	(1)	2	00
2	3	3	01
3	2	1	10
	Q^{t+1}		

Rys. 5. Tablica przejść i wyjść automatu Moore'a.

W układach asynchronicznych każda zmiana stanu wejściowego powoduje bezpośrednio zmiany stanu pamięci oraz stanu wyjść.

Zgodnie z grafem, jeżeli układ znajduje się w stanie 1, a na wejściu jest 0, to układ pozostanie w stanie 1. Jednak jeżeli na wejściu pojawi się 1, to układ przejdzie do stanu 2, następnie do stanu 3, do stanu 1 i ponownie do 2. Jak długo na wejściu będzie 1, tak długo układ będzie się ciągle przełączał pomiędzy stanami wewnętrznymi. Oznacza to, że są to **stany niestabilne**. Poprawne działanie sekwencyjnego układu asynchronicznego wymaga obecności **stanów stabilnych**, tzn. takich które nie zmieniają się przy stałym sygnale wejściowym $\boldsymbol{Q}^{t+1} = \boldsymbol{Q}^t$. Taka sytuacja dla powyższego grafu zachodzi w stanie 1 przy wejściu 0. Stany stabilne zaznaczamy w tablicy przejść otaczając je okręgiem, tak jak to przedstawiono na rys. 5.

Zmienne wejściowe i wyjściowe u układach logicznych są opisane zmiennymi dwuwartościowymi (binarnymi 0-1). W układach technicznych zmiennymi wejściowymi mogą być stany przycisków (wciśnięty lub nie) czy stany wszelkich czujników krańcowych wykrywających kontakt lub obecność przedmiotu (jest lub nie ma). Zmienne wyjściowe to będą z kolei zazwyczaj sygnały sterujące układem wykonawczym np.: silnik włączony lub nie, zawór zamknięty lub otwarty, lampa zapalona lub zgaszona, itd. Kwestia wewnętrznych zmiennych stanu układu (pamięci) nie jest już tak oczywista, ponieważ oznaczenia stanów mają charakter umowny. Teoria związana z układami logicznymi była rozwijana razem z rozwojem techniki cyfrowej. Dlatego klasycznie przyjmuje się, że wewnętrzne stany układu koduje się z wykorzystaniem zmiennych binarnych. Przykładowo, do zakodowania czterech stanów pamięci, wystarczą dwie zmienne binarne, które zapewniają cztery możliwe kombinacje (00, 01, 11, 10). Taki sposób

kodowania powoduje, że do zaprojektowania poprawnego przełączania pomiędzy stanami, trzeba rozważyć również zależności pomiędzy poszczególnymi zmiennymi. A wybór sposobu zakodowania poszczególnych stanów może mieć znaczący wpływ na złożoność znalezionych funkcji logicznych.

Q^t \ X^t	0	1
00	11	(00)
01	(01)	10
11	(11)	10
10	11	(10)
	Q^{t+1}	

Rys. 6. Tablica przejść z wyścigiem krytycznym.

Okazuje się, że opóźnienie wprowadzane przez poszczególne elementy wykonujące funkcje logiczne, ma też wpływ na przełączanie stanów układu. Bezpiecznym wydaje się być założenie, że żadne dwa elementy układu logicznego nie wykonują obliczeń dokładnie w tym samym czasie. Taka obserwacja prowadzi do wniosku, że bardzo mało prawdopodobne jest uzyskanie w układzie fizycznym przełączenia dwóch zmiennych w dokładnie tej samej chwili. Przykładowo, oznacza to że przejście ze stanu (00) do (11) zawsze będzie odbywać się ze stanem pośrednim: (00) → (10) → (11) lub (00) → (01) → (11). Biorąc pod uwagę powyższe, przenalizowano tablicę przejść z rys. 6. Jeżeli układ jest w stanie $\boldsymbol{Q}$ = (00), a wejścia wynoszą $\boldsymbol{X}$ = (0), to układ powinien przejść do stanu, $\boldsymbol{Q}$ = (11). Jak już zauważono wcześniej, takie przejście musi odbyć się w dwóch krokach. Poprawne przełączenie jest możliwe w sekwencji (00) → (10) →(11), ponieważ stan $\boldsymbol{Q}$ = (10) dalej kieruje układ do stabilnego stanu $\boldsymbol{Q}$ = (11). Jednak przejście ze stanem pośrednim (01) nie jest możliwe, ponieważ stan $\boldsymbol{Q}$ = (01) jest stanem stabilnym i układ już w nim pozostanie. Takie zjawisko uzyskania błędnego stanu stabilnego na skutek różnicy czasu działania elementów nazywa **się wyścigiem krytycznym**. Jak można zaobserwować, dla powyższej tablicy przy $\boldsymbol{X}$ = (1) zjawisko wyścigu nie występuje. Jeżeli w projektowanym układzie asynchronicznym występuje wyścig krytyczny, należny zmienić kodowanie poszczególnych stanów lub dodać stan pośredni, aby to zjawisko wyeliminować.

Jak napisano wcześniej, określenie stanów wewnętrznych układu może być kłopotliwe. A pierwsza opracowana tablica przejść (tablica pierwotna) może zawierać więcej stanów niż jest to potrzebne do poprawnego działania układu sekwencyjnego. Ogólnie można przyjąć, że im więcej stanów wewnętrznych, tym bardziej skomplikowany układ logiczny. Dlatego korzystnie jest zredukować nadmiarowe stany. Dwa (lub więcej) stany wewnętrze możemy zastąpić jednym, jeżeli oba stany uznamy za **zgodne**. Zgodność najłatwiej jest stwierdzić analizując tablicę przejść.

Dla automatu o strukturze Moore'a za zgodne możemy uznać stany, które spełniają wszystkie poniższe warunki:

- dla tych samych wejść ($\boldsymbol{X_i}$) przechodzą do tych samych (lub zgodnych) stanów (tzn. opisują je jednakowe wiersze w tablicy przejść i wyjść),
- odpowiada im ten sam stan wyjściowy ($\boldsymbol{Y}$),
- jeśli są stabilne przy tym samym $\boldsymbol{X}$, to odpowiadające im stany $\boldsymbol{Y}$ są jednakowe. Jeżeli w tablicy przejść są niewypełnione pola, (tzn. że stan automatu jest tam nieustalony) to możemy je uzupełnić tak, żeby uzyskać stany zgodne.

Spójrzmy na przykład z rys. 7. Analizując tablicę pierwotną, możemy zaobserwować, że stany 2 i 4 są zgodne, ponieważ dla każdego stanu wejść (X_1, X_2, X_3, X_4), przechodzą do tych samych stanów $\boldsymbol{Q}^{t+1}$ oraz mają te same stany wyjść $\boldsymbol{Y}$ = (0). W zredukowanej tablicy przejść połączeniu stanów 2 i 4 opowiada nowy stan 2.

Q^t \ X^t	X_1	X_2	X_3	X_4	Y^t
1	(1)	3	2	(1)	1
2	1	(2)	4	1	0
3	(3)	-	4	1	1
4	1	2	(4)	1	0
	Q^{t+1}				

Q^t \ X^t	X_1	X_2	X_3	X_4	Y^t
(1,3)→1	(1)	(1)	2	(1)	1
(2,4)→2	1	(2)	(2)	1	0
	Q^{t+1}				

Rys. 7. Redukcja tablicy przejść automatu Moore'a: a) tablica pierwotna, b) tablica zredukowana.

Zastąpić jednym nowym stanem możemy również stany 1 i 3, jednak ten przypadek nie jest taki oczywisty. Stany te są zgodne ponieważ:

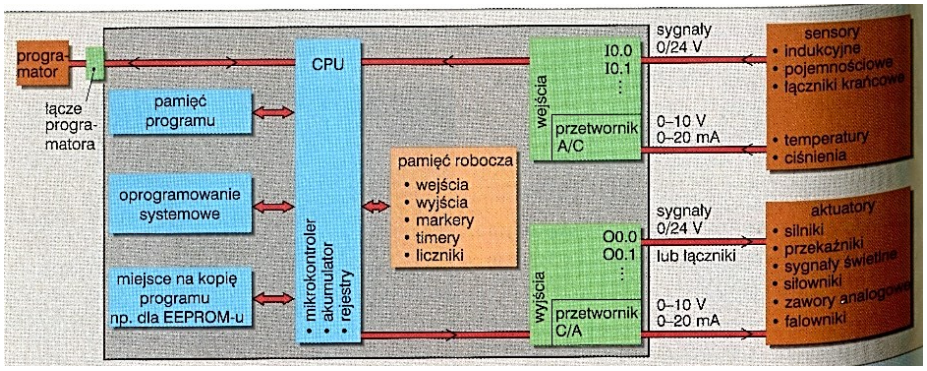
- dla $\boldsymbol{X_4}$ przechodzą do tego samego stanu 1,
- dla $\boldsymbol{X_3}$ przechodzą od stanów zgodnych 2 i 4,
- dla $\boldsymbol{X_2}$ przechodzą do tego samego stanu, jeżeli wykorzystamy stan nieokreślony (-) i wpiszemy w jego miejsce stan 3,
- dla $\boldsymbol{X_1}$ oba stany są stabilne i odpowiada im ten sam stan wyjść $\boldsymbol{Y}$ = (1).

W zredukowanej tablicy przejść połączone stany 1 i 3 tworzą nowy stan 1. Aby dwa stany uznać za zgodne w strukturze automatu Mealy'ego nie jest wymaga zgodność wyjść $\boldsymbol{Y}$. Podczas syntezy układu sekwencyjnego, pierwotna tablica przejść jest zazwyczaj przygotowana dla struktury Moore'a. Znajdując w niej stany zgodne dla automatu Mealy'ego, można zdecydować o zmianie typu automatu. Trudno jest z góry przewidzieć, która opcja okaże się mniej skomplikowana, dlatego trzeba opracować obie i dopiero dokonać wyboru.

2 Sterownik PLC

Sterowniki PLC (Programmable Logic Controller) to układy mikroprocesorowe podobnie jak komputery. Składają się z zasilacza, modułu sygnałów wejść i wyjść (I/O), jednostki centralnej z mikroprocesorem (CPU) oraz pamięci programu. Sterownik PLC jest więc układem programowalnym w jednym z kilku języków: FBD, LAD, STL, SFC, IL, których komendy wprowadzane są do urządzenia pod kontrolą specjalizowanego systemu operacyjnego. W przemyśle sterowniki PLC służą do sterowania urządzeniami mechatronicznymi, takimi jak: siłowniki pneumatyczne/hydrauliczne, przekaźniki, styczniki, silniki elektryczne, lampki kontrolne, itd., które pracują w mniej lub bardziej rozbudowanych systemach mechatronicznych. Ponadto, sterowniki mogą pełnić rolę regulatorów parametrów procesów technologicznych takich jak: określona zmiana temperatury, ciśnienia czy natężenia przepływu płynu w czasie. W związku z tym wejścia i wyjścia sterowników PLC są dostosowywane pod względem zastosowanych elementów i ich parametrów do parametrów sterowanych obiektów. Budowę sterownika PLC można przedstawić w następujący sposób, patrz rys. 8.

Sterownik PLC w czasie swojej pracy pobiera dane z urządzeń wejściowych i na podstawie algorytmu zapisanego w programie sterowniczym wykonuje odpowiednie operację arytmetyczo-logiczne. Wyniki tych operacji są wysyłane jednocześnie do odpowiednich wyjść sterownika. Sterownik może znajdować się w jednym z dwóch trybów pracy: RUN lub STOP. Jeżeli sterownik jest w stanie RUN to program wykonywany jest cyklicznie. Natomiast w trybie STOP realizacja programu jest wstrzymana i istnieje możliwość wprowadzenia nowego programu do pamięci sterownika. Nowy program może być wprowadzony z komputera PC poprzez specjalny adapter. Na ekranie komputera tworzony jest program w tzw. edytorze, a następnie po sprawdzeniu go dokonywana jest jego kompilacja i przesłanie do sterownika kodu binarnego.



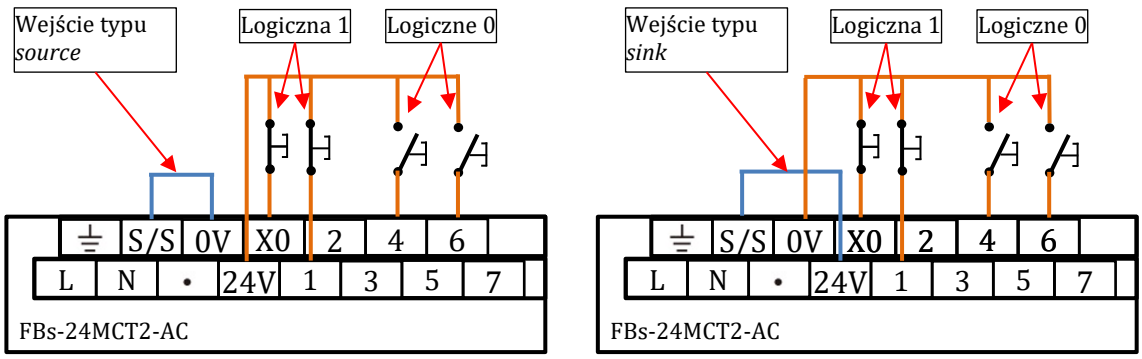
Rys. 8. Budowa sterownika PLC.

Pamięć w sterowniku służy do przechowywania danych, obliczeń i programu. Fizycznie w sterownikach stosowana jest pamięć RAM (odczyt-zapis) oraz EEPROM lub FLASH, która jest odczytywana i zapisywana za pomocą specjalnego programatora. Pod względem funkcjonalnym pamięć sterownika dzielimy na 3 rodzaje: roboczą, danych i systemową. Robocza pamięć operacyjna przechowuje bieżące argumenty oraz wyniki obliczeń. W pamięci danych przechowywany jest program sterowniczy. W pamięci systemowej przechowywany jest system sterownika, program obsługi awaryjnej oraz tzw. obrazy, czyli skany wejść i wyjść sterownika. Sterowniki PLC pracują cyklicznie co oznacza, że po włączeniu zasilania i przełączeniu w tryb RUN, sterownik zaczyna realizację programu, a po jej skończeniu zaczyna skanować wejścia i ponownie realizować program. Czas takiego cyklu zależy od wielkości programu, liczby obsługiwanych wejść i wyjść oraz od szybkości pracy CPU. Praktyczny czas cyklu prostych programów sterowniczych wynosi od kilku do kilkudziesięciu milisekund. Cykl pracy sterownika składa się z następujących faz:

i). Inicjacja cyklu, w tej fazie sterownik sprawdza stan pracy (RUN/STOP) oraz stan komunikatów o błędach.
ii). Czytanie sygnałów wejściowych, faza ta polega na tym, że odczytane stany wejść są zapisywane w specjalnym obszarze pamięci systemowej nazywanej obrazem wejść. Po tym zapisie, sterownik staje się „ślepy" na zmiany stanów logicznych na wejściach, aż do końca cyklu.
iii). Wykonywanie programu sterowniczego, w czasie tej fazy wyliczane są stany logiczne wyjść sterownika, ale nie sią one jeszcze wysyłane do wyjść fizycznych tylko są zapisywane w obrazie wyjść, który jest również specjalnym obszarem pamięci systemowej.
iv). Aktualizacja wyjść fizycznych, w tej fazie przesyłane są jednocześnie stany logiczne z obrazu wyjść na wyjścia fizyczne i zostają zatrzaśnięte, aż do następnej aktualizacji.
v). Diagnostyka sterownika, w tej fazie sterownik dokonuje autodiagnostyki zarówno pod względem sprzętowym jak i błędów systemowych.
vi). Komunikacja, ta faza cyku jest opcjonalna i występuje wtedy, kiedy sterownik musi skomunikować się z czujnikami i odległymi od niego (np. drogą radiową) lub innymi sterownikami pracującymi w sieci przemysłowej.

Do wejść sterowników można doprowadzić zarówno sygnały analogowe jak i binarne. Sygnały binarne to sygnały dwustanowe, najczęściej 0 V – „0" lub +24 VDC – „1".

Wejścia cyfrowe sterownika (rys. 9) mogą być typu *sink* (ang. zlew) lub typu *source* (ang. *źródło*). W przypadku wejścia typu *sink,* zewnętrzne urządzenie (np. czujnik, przycisk, czy też inny sterownik) jest zasilane przez źródło zewnętrzne, a prąd płynący przez obwód wejściowy PLC jest "połykany" do wewnątrz sterownika PLC.

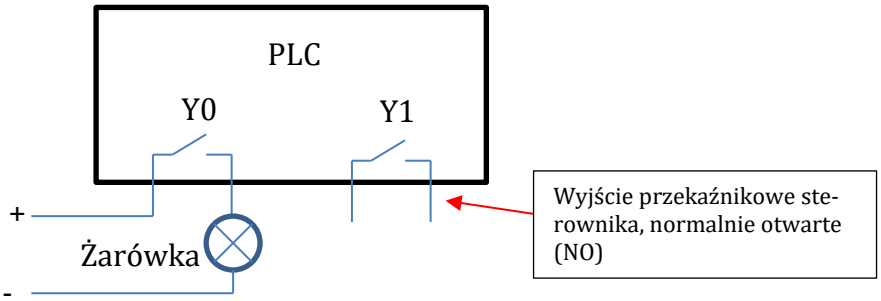


Rys. 9 Wejścia sterownika PLC typu sink i source

Węzeł wspólny (COM) wejścia PLC jest podłączony do masy (GND), a pozytywne napięcie jest dostarczane z zewnętrznego źródła. Dla wejście typu *source* zewnętrzne urządzenie jest zasilane przez PLC, a prąd płynie z wewnątrz PLC do zewnętrznego urządzenia. W tym przypadku PLC "pompuje" prąd na

zewnętrzne urządzenie. Węzeł wspólny (COM) wejścia PLC jest podłączony do źródła zasilania, a ujemne napięcie jest podłączone do zewnętrznego urządzenia.

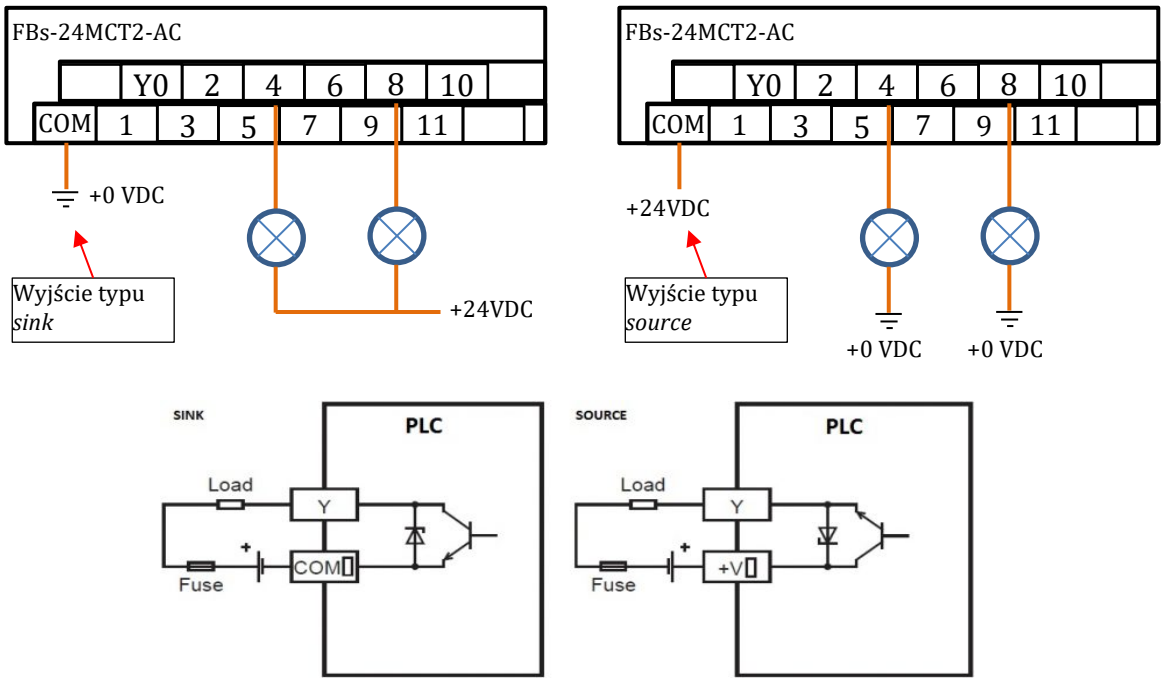
Wyjścia sterownika mogą być również cyfrowe bądź analogowe. Wyjścia cyfrowe mogą być przekaźnikowe lub tranzystorowe. Do wyjść cyfrowych binarnych należą wyjścia przekaźnikowe, gdzie wyjściem fizycznym sterownika jest zestyk normalnie otwarty, patrz rys. 10.



Rys. 10. Przykładowy schemat połączenia zasilania żarówki z wykorzystaniem wyjścia przekaźnikowego sterownika PLC.

Zaletami wyjść przekaźnikowych jest ich duża odporność na zakłócenia i duża obciążalność styków. Wadami natomiast jest wolne działanie w porównaniu do wyjść tranzystorowych.

Wyjścia tranzystorowe stosowane są tylko w przypadku zasilania odbiorników napięcia stałego, zwykle +24 VDC. Dopuszczalny prąd tranzystorów jest wystarczająco duży, by wysterować silnik małej mocy, cewki zaworów elektromagnetycznych lub lampę. Wyjścia tranzystorowe również mogą być typu *sink* lub typu *source* (rys. 11).

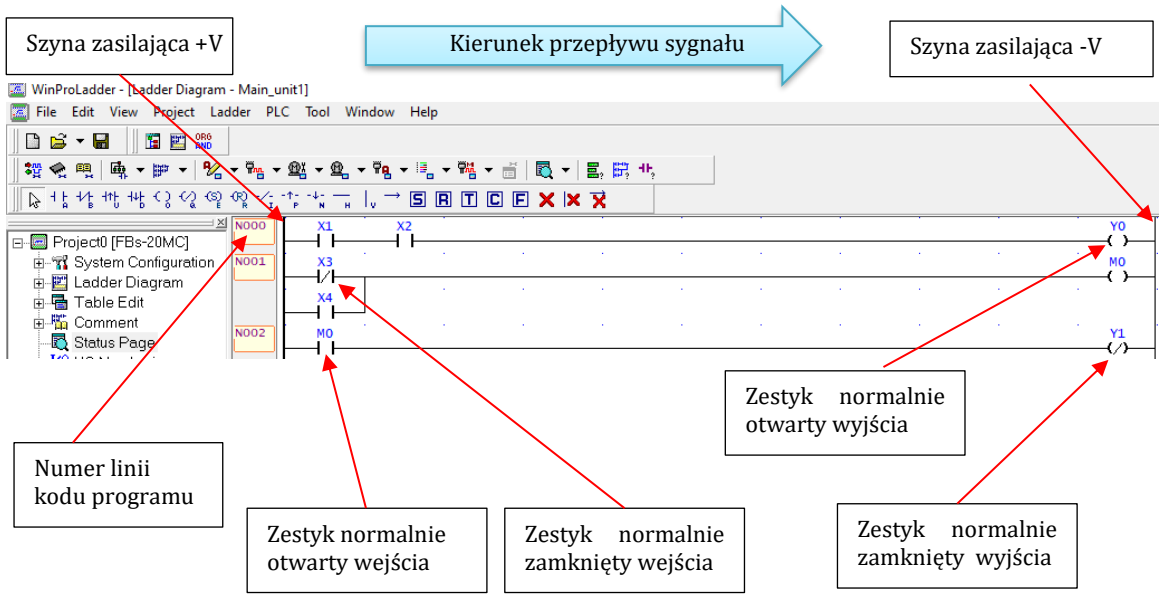


Rys. 11. Tranzystorowe wyjścia sterownika PLC typu sink i source.

Różnica w tych połączeniach jest taka sama jak w przypadku wejść, gdzie typ *sink* umożliwia sterowanie urządzeń wyjściowych potencjałem minusowym, natomiast typ *source* zapewnia sterowanie potencjałem dodatnim. W skrócie, dla wyjść typu *source* stan logiczny 1 będzie odpowiadał napięciu 24 VDC (podłączonego do zestyku wspólnego COM), a stan logiczny 0 będzie odpowiadał 0 V. Natomiast dla wyjścia typu *sink* stan logiczny 1 będzie odpowiadał 0 V (podłączonym do zestyku wspólnego COM), a stanu logicznego 0 będzie odpowiadał +24 VDC.

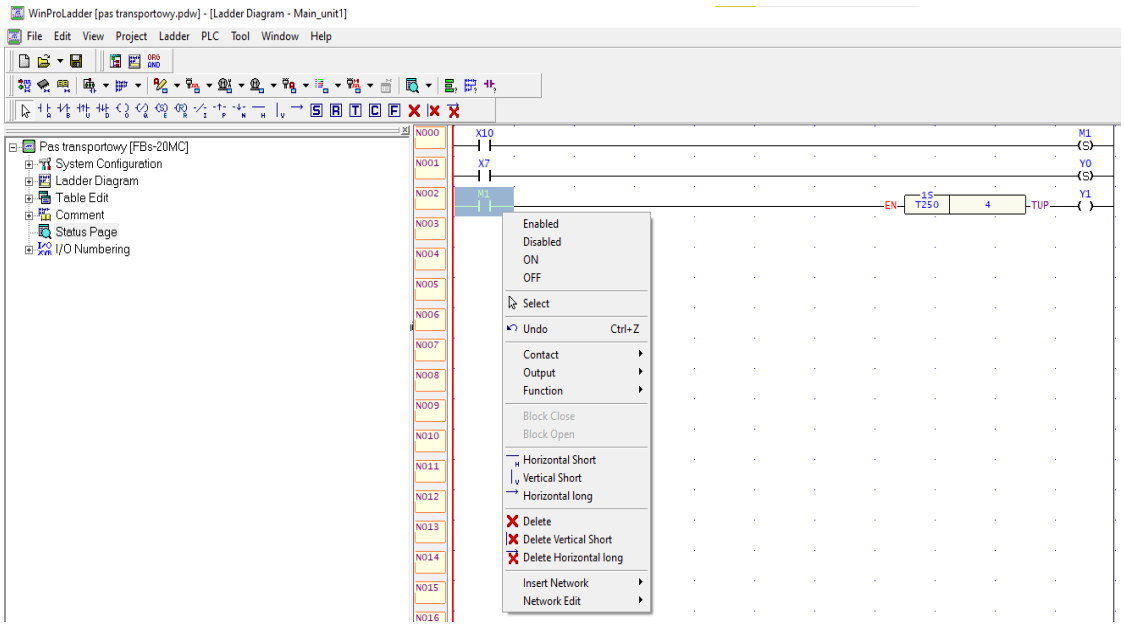
3 Język programowania LAD i sterownik Fatek FBs-24MCT2-AC.

W tym rozdziale na przykładzie sterownika PLC Fatek FBs-24MCT2-AC i dedykowanego do niego programu WinProladder.exe zostanie wytłumaczony sposób programowania w języku drabinkowym LAD (ang. *Ladder Diagram*). Program napisany w tym języku jest analogiczny do układów elektrycznych ze stykami, które są normalnie otwarte (NO) i normalnie zamknięte (NC). Przeanalizujmy, linijka po linijce program napisany w języku LAD znajdujący się na rys. 12.



Rys. 12. Przykładowy program LAD napisany w programie WinProladder.exe.

Wejścia sterownika oznacza się jako X0, X1, X2..., natomiast wyjścia jako Y0, Y1, Y2... Linia kodu oznaczona, jako N000 wykonuje działanie logicznej koniunkcji sygnałów X1 i X2, a wynik pokazuje na wyjściu Y0 sterownika. Po zaniku choćby jednego z sygnałów X1, X2, wyjście sterownika Y0 natychmiast zostaje wyłączone. W zapisie logicznym linia ta realizuje następującą funkcje logiczną $Y0 = X1 \cdot X2$. Linia kodu oznaczona jako, N001 wykonuje działanie logicznej alternatywy sygnałów X3 i X4, przy czym sygnał X3

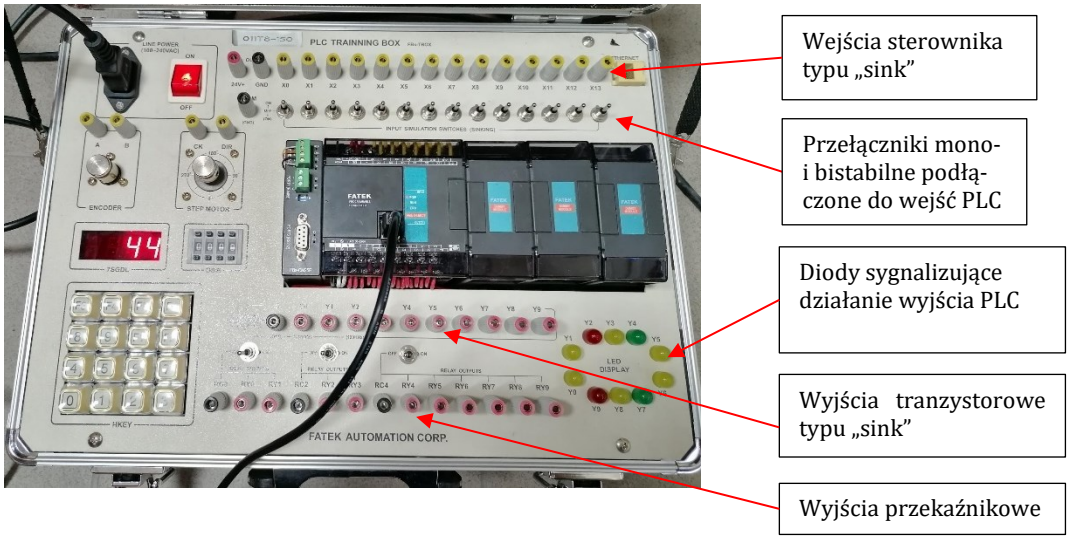


Rys. 13. Symulacja działania programu w WinProLadder.exe

jest zanegowany. Wynik tego działania jest zapisywany w pamięci sterownika jako tzw. zmienna markująca M0. Zmienna markująca jest zmienną wewnętrzną, która nie jest wysyłana na fizyczne wyjścia sterownika, ale może być używana w dalszych częściach programu i brać udział w algorytmie sterującym. W zapisie logicznym linia ta realizuje następującą funkcje logiczną $M0 = \overline{X3} + X4$. Linia kodu oznaczona, jako N002 przypisuje stan zmiennej markującej M0 zanegowanemu wyjściu sterownika Y1. W zapisie logicznym linia ta realizuje następującą funkcje logiczną $M0 = \overline{Y1}$.

Po napisaniu programu, można zasymulować jego działanie w programie. W tym celu należy kliknąć w zakładkę *PLC*, a następnie *Simulation.* Wtedy pojawiają się dostępne opcję *RUN* i *STOP*. Opcja *RUN* uruchamia wirtualnie sterownik PLC, szyna zasilająca programu robi się czerwona. Wtedy poprzez kliknięcie prawym przyciskiem myszy na dany zestyk można go załączyć *ON* lub wyłączyć *OFF,* patrz rys. 13. W ten sposób zadaje się sygnały na wejściach, a następnie obserwuje co się dzieje z wyjściami, bądź poszczególnymi liniami kodu. Po zakończeniu symulacji możemy wirtualnie wyłączyć sterownik *STOP* i wyłączyć symulowanie programu poprzez kliknięcie *End Simulation* w zakładce *PLC*.

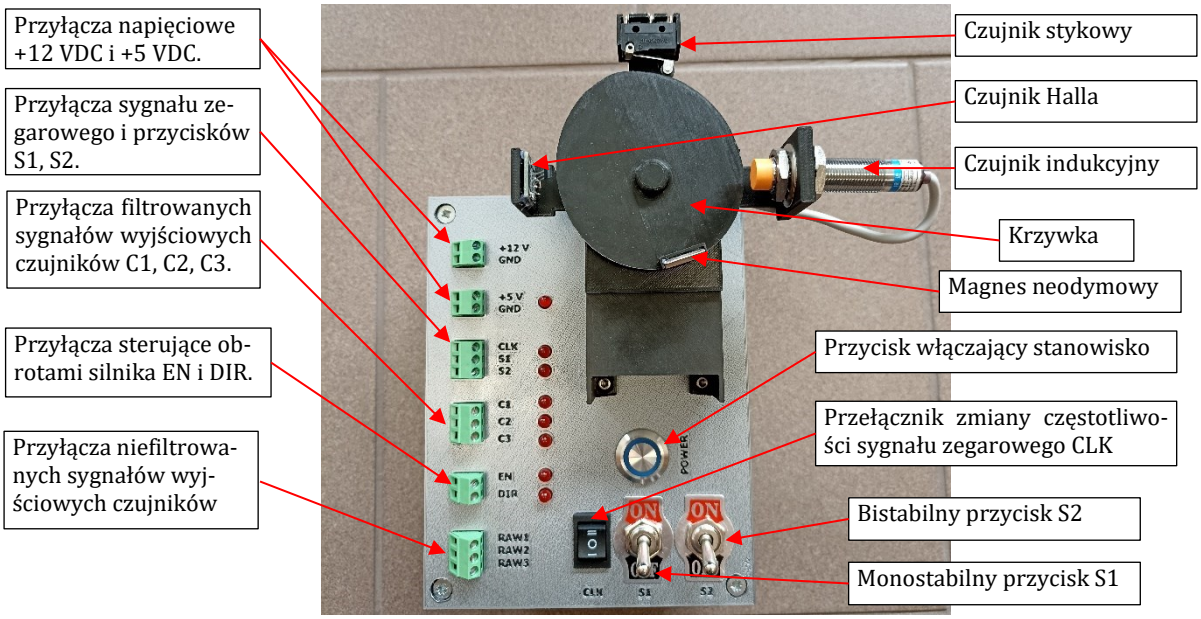
W czasie symulacji na stanowisku ćwiczeniowym (rys. 14), napisany przez nas program zostanie wgrany na sterownik PLC, a następnie jego działanie będzie można podglądać jednocześnie w programie WinProllader.exe i na trenażerze. W celu wgrania programu na sterownik należy podłączyć go do komputera poprzez kabel i kliknąć zakładkę *PLC*, a następnie *On-line*. W oknie, które zostanie otwarte należy kliknąć *Auto Check* i potwierdzić wybór w kolejnym oknie klikając *OK.* W ten sposób program wyszuka, do którego portu jest podłączony sterownik PLC. Następnie program spyta się nas, czy chcemy połączyć się z wyszukanym sterownikiem PLC, co należy potwierdzić klikając *TAK.* Wchodząc ponownie w zakładkę *PLC* klikamy *RUN,* spowoduje to wgranie do sterownika napisanego programu i rozpocznie proces jego symulacji zarówno na trenażerze, jak i na komputerze. Jeśli w czasie symulacji kliknęlibyśmy *STOP* w zakładce *PLC*, symulacja zostanie zatrzymana, a wgrany na sterownik program zostanie usunięty z jego pamięci. Aby pozostawić na sterowniku wgrany program w pamięci urządzenia, tak żeby po odłączeniu go od komputera wciąż był przez niego wykonywany, należy kliknąć *Off-line*, zamiast *STOP*. Wtedy program zostanie w pamięci sterownika i będzie można go symulować po odłączeniu od komputera.



Rys. 14. Stanowisko ćwiczeniowe sterownika PLC – Fatek FBs-24MCT.

4 Stanowisko silnika z krzywką

Stanowisko obcowzbudnego silnika prądu stałego z zamocowana na jego wale krzywką pokazana na rys. 15. Stanowisko zasilane jest napięciem +12 VDC pochodzącego z zasilacza stabilizowanego i uruchamia się je bistabilnym przełącznikiem POWER. Układ wyposażony jest w dwa przełączniki – monostabilny i bistabilny, których jeden z zestyków podłączony jest do napięcia +5 VDC, zaś drugi z zestyków



Rys. 15. Stanowisko silnika z krzywką.

wyprowadzony został do przyłączy nazwanych S1 i S2. Układ posiada dwa przyłącza, które dostarczają napięcia +12 VDC i +5 VDC. Przyłącza EN i DIR odpowiadają za sterowanie silnikiem. Po dostarczeniu +5 VDC do przyłącza EN (*enable*) silnik zaczyna obracać się zgodnie ze wskazówkami zegara.

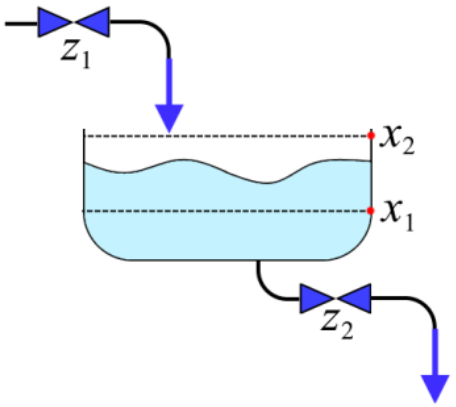
Dostarczenie +5 VDC do przyłącza DIR (*direction*) spowoduje zmianę kierunku obrotów silnika. Stanowisko wyposażone jest w trzy czujniki położenia krzywki. Pierwszy czujnik to zestyk z rolką, który po złączeniu wysyła +5 VDC na przyłącze C1. Drugi czujnik stanowi czujnik indukcyjny, który po wykryciu metalowej – niklowej powłoki ochronnej oraz żelaza znajdującego się w magnesie neodymowym zamocowanym na krzywce załącza się wysyłając na zestyk C2 napięcie +5 VDC. Trzeci czujnik to czujnik Halla reagujący na pole magnetyczne zamocowanego na krzywce magnesu neodymowego, po wykryciu magnesu czujnik wysyła na przyłącze C3 napięcie 0 V, ponieważ w czasie niedziałania czujnika wysyła on na swoje wyjście +5 VDC. Napięcia wyjściowe z czujników wysyłane na przyłącza C1, C2 i C3 są napięciami filtrowanymi, natomiast przyłącza RAW1, RAW2, RAW3 odpowiadają ich nieprzetworzonym wartościom. Przyłącze CLK jest to 5 VDC sygnał zegarowy, który w zależności od położenia trójstanowego przełącznika CLK może generować dwa sygnały prostokątne o małej i dużej częstotliwości.

5 Bibliografia

[1] W. Traczyk, Układy cyfrowe automatyki, Wydawnictwo Naukowo-Techniczne, Warszawa 1974.
[2] J. Kalisz, Podstawy elektroniki cyfrowej, Wydawnictwo Komunikacji i Łączności, Warszawa 2002.
[3] T. Łuba, Syneza układów logicznych, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2005.
[4] D. Schmid, Mechatronika, Warszawa: REA, 2008.
[5] M. Olszewski, Urządzenia i systemy mechatroniczne, Warszawa: REA, 2009.
[6] U. Fischer, Poradnik mechanika, Warszawa: REA, 2008.

Załącznik A - Przykład syntezy układu asynchronicznego w strukturze Moore'a

Przykładowa synteza układu asynchronicznego w strukturze Moore'a zostanie przedstawiona dla układu sterowania poziomem cieczy w zbiorniku. Schemat układu przedstawiono na rys. 6.11. Składa się on ze zbiornika, zaworu wlewowego z_1, zaworu spustowego z_2 oraz z dwóch czujników poziomu cieczy x_1 i x_2. Cieczy do zbiornika można dolewać otwierając zawór z_1 oraz upuszczać otwierając zawór z_2.



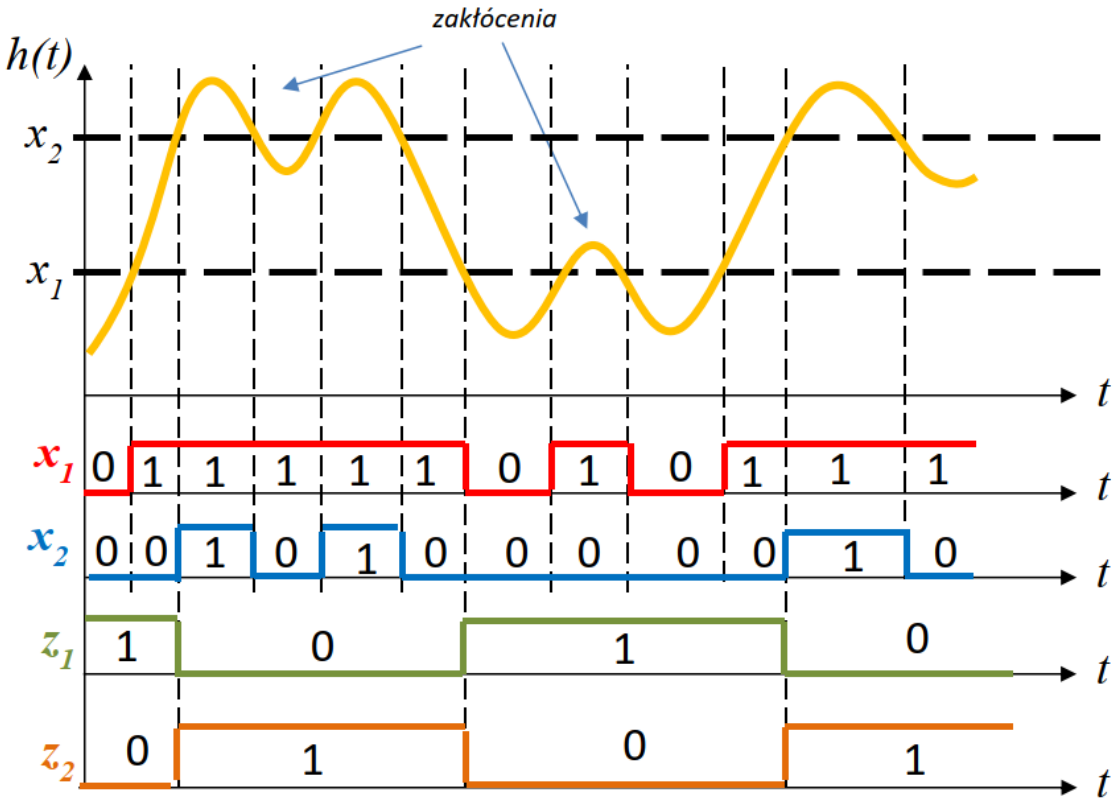
Rys. 6.11 Układ zbiornika z zaworami z_1 i z_2 oraz czujnikami poziomu cieczy x_1 i x_2.

Projekt układu logicznego rozpoczniemy od opisu słownego pożądanego działania układu: *Zadanie polega na utrzymaniu poziomu cieczy w zbiorniku pomiędzy poziomami x_1 i x_2. Działanie układu ma zostać zatrzymane na wypadek awarii czujników.*

W następnej kolejności przygotujemy wykresy czasowe odpowiadające typowemu działaniu (typowej sekwencji) układu. W tym celu najpierw przypiszemy stany logiczne odpowiadające fizycznym stanom układu:

- ciecz poniżej poziomu czujnika → stan logiczny 0,
- ciecz powyżej lub równo z poziomem czujnika → stan logiczny 1,
- zawór otwarty → stan logiczny 1,
- zawór zamknięty → stan logiczny 0.

Wykresy czasowe zostaną przygotowane z pomocą wykresu poziomu cieczy, uwzględniającego zakłócenia – falowanie cieczy, co przedstawia rys. 6.12

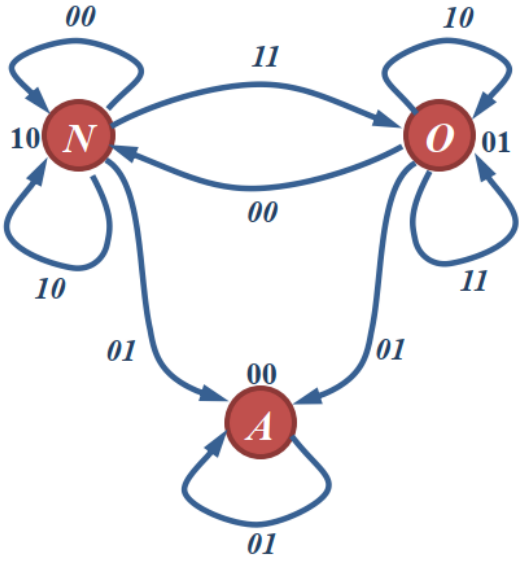


Rys. 6.12 Wykres czasowy typowej sekwencji działania układu.

Analizując wykresy czasowe (lub ciągi zero-jedynkowe) możemy zaobserwować, że realizacja zadania wymaga układu sekwencyjnego, ponieważ przy pośrednim poziome cieczy ($x_1 = 1$, $x_2 = 0$) raz zbiornik jest napełniany a raz opróżniany, tzn. temu samemu stanowi wejść odpowiadają różne stany wyjściowe.

Wektor wartości wejściowych obejmuje stany czujników poziomu cieczy, tj. $\boldsymbol{X} = (x_1, x_2)$. Natomiast wektor wartości wyjściowych będzie opisany przez stany zaworów, tj . $\boldsymbol{Y} = (z_1, z_2)$. Jeżeli chodzi o stany wewnętrzne układu, to intuicyjne można je określić jako stan napełniania zbiornika N i stan opróżniania O. Stanowi napełniania odpowiada stan wyjść $\boldsymbol{Y} = (1, 0)$, a stanowi opróżniania $\boldsymbol{Y} = (0,1)$. Przewidziana miała być również możliwość awaryjnego zatrzymania działania układu w wypadku awarii czujników. Na podstawie odczytu stanu czujników za awarię można uznać sytuację ($x_1 = 0,\ x_2 = 1$), która oznaczałaby poziom cieczy jednocześnie powyżej czujnika x_2 i poniżej czujnika x_1. Zdecydowano, że w takim przypadku należy zamknąć oba zawory. Na taki wypadek wprowadzono stan awarii A, któremu odpowiada stan wyjść $\boldsymbol{Y} = (0, 0)$.

Na podstawie przebiegów czasowych z rys. 6.12, sporządzono graf przejść, który przedstawia rys. 6.13a. Rozpoczęto od zaznaczenia wierzchołków grafu reprezentujących stany N, O i A. Następnie analizując kolejne stany wejściowe dodawano kolejne gałęzie grafu. Inaczej wyglądała analiza stanu awaryjnego. Jego zależności nie wynikają z typowego (poprawnego) działania układu, dla którego sporządzono wykresy czasowe. Tutaj przyjęto założenia, co do tego jak układ ma zadziałać w sytuacji awaryjnej. Przyjęto, że układ wchodzi w stan awarii A, kiedy pojawi się stan wejściowy $\boldsymbol{X} = (0, 1)$.



a)

Q^t \ X^t	00	01	11	10	Y^t
N	(N)	A	O	(N)	**10**
O	N	A	(O)	(O)	**01**
A	-	(A)	-	-	**00**
	Q^{t+1}				

b)

Rys. 6.13 Graf układu (a) oraz odpowiadająca mu tablica przejść i wyjść (b).

Na podstawie grafu układu opracowano pierwotną tablice przejść i wyjść (Rys. 6.13b) z zaznaczonymi stanami stabilnymi. Następnym etapem jest minimalizacja tablicy, przez redukcję stanów zgodnych. W uzyskanej tablicy jednak nie znajdujemy stanów zgodnych dla struktury Moore'a.

Następnie przyjęto sposób kodowania stanów. Do zakodowania trzech stanów potrzebne są dwie zmienne binarne q_1 i q_2. Sposób kodowania przedstawiono w tablicy z rys. 6.14a.

Q	q_1	q_2
N	0	0
O	0	1
A	1	1
-	1	0

a)

Q^t \ X^t	00	01	11	10	Y^t
00	(00)	11	(01)	(00)	**10**
01	00	11	(01)	(01)	**01**
11	-	(11)	-	-	**00**
10	-	-	-	-	-
	Q^{t+1}				

b)

Rys. 6.14 Kodowanie wewnętrznych stanów układu (a) oraz odpowiadająca mu tablica przejść i wyjść (b).

Zakodowaną tablice przejść należy przeanalizować pod kątem wyścigów krytycznych. Dla założonego kodowania, problematyczne może być przejście układu ze stanu napełniania $\boldsymbol{Q} = (0,0)$ do stanu awarii $\boldsymbol{Q} = (1,1)$, przy pojawieniu się na wejściu $\boldsymbol{X} = (0,1)$. Poprawne przełączenie jest możliwe w sekwencji (00) → (01) →(11). Aby umożliwić przełączenie w sekwencji (00) → (10) →(11) wykorzystano nieokreślone przejście układu dla $\boldsymbol{Q} = (1,0)$ i $\boldsymbol{X} = (0,1)$. Nie określone przejście zastąpiono przejściem do stabilnego stanu $\boldsymbol{Q} = (1,1)$.

Funkcję przejścia $\boldsymbol{Q}^{t+1} = \delta(\boldsymbol{Q}^t, \boldsymbol{X}^t)$ należy przygotować dla obu zmiennych wewnętrznych q_1 i q_2. W tym celu przygotowano oddzielną tablicą przejść dla każdej zmiennej (patrz Rys. 6.15a i Rys. 6.15b).

Q^t \ X^t	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	-	1	-	-
10	-	1	-	-
	q_1^{t+1}			

a)

Q^t \ X^t	00	01	11	10
00	0	1	1	0
01	0	1	1	1
11	-	1	-	-
10	-	1	-	-
	q_2^{t+1}			

b)

Q	q_1q_2	z_1	z_2
N	00	1	0
O	01	0	1
A	11	0	0
-	10	-	-

c)

Rys. 6.15 Tablice przejść dla wewnętrznych zmiennych stanu q_1 (a) i q_2 (b) oraz tablica wyjść (c)

Na podstawie tablic zapisano poszczególne funkcje przejść wykorzystując metodę minimalizacji jedynkowej:

$$q_1^{t+1} = \overline{x_1}x_2, \tag{6.6}$$

$$q_2^{t+1} = x_2 + q_2x_1. \tag{6.7}$$

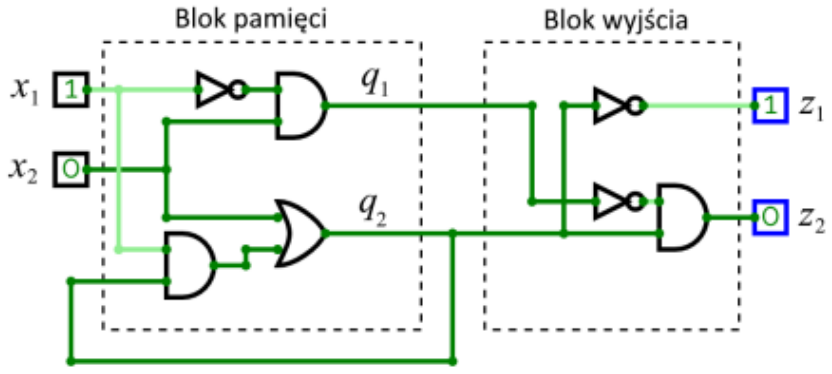
Podczas minimalizacji stany nieokreślone można wykorzystać jako „jedynki" lub zera w zależności od wybranej postaci funkcji logicznej. Należy przy tym pamiętać, że przez taką poszerzoną minimalizację przyjmujemy pośrednio sposób działania układu w stanach, które wcześniej były nieokreślone. Przed podjęciem takiego kroku należy przeanalizować jaki ten zabieg ma wpływ na działanie układu. Analizując wyprowadzone funkcje logiczne, można stwierdzić, że nie występuje w nich zagrożenie hazardem statycznym.

Do wyprowadzenia pozostała jeszcze funkcja wyjść $\boldsymbol{Y^t} = \lambda_1(\boldsymbol{Q^t})$ na podstawie tablicy z rys. 6.15c:

$$z_1 = \overline{q_2}, \tag{6.8}$$

$$z_2 = \overline{q_1}q_2. \tag{6.9}$$

Układ z bramek logicznych realizujący zaprojektowany układ sekwencyjny (6.6)-(6.9) przedstawia rys. 6.16.



Rys. 6.16 Realizacja zaprojektowanego logicznego układu sekwencyjnego (6.6)-(6.9) z wykorzystaniem bramek logicznych.