WOJSKOWA AKADEMIA TECHNICZNA

im. Jarosława Dąbrowskiego

Wprowadzenie do Automatyki

Sprawozdanie z pracy laboratoryjnej nr 3

Programowanie sterowników logicznych PLC.

Schematy drabinkowe LD. Schematy blokowe FBD.

KAROL DYBOWSKI

I5X1N1

Prowadzący:

dr inż. Andrzej Wiśniewski

1. Wstęp teoretyczny.

Sterownik PLC (**PLC** z angielskiego **P**rogrammable **L**ogic **C**ontroller, lub z niemieckiego **SPS** **S**peicher**p**rogrammierbare **S**teuerung) - jest urządzeniem służącym do sterowania urządzeniem (np. linią produkcyjną), zgodnie z programem, stworzonym przez użytkownika. Sterownik taki posiada system operacyjny czasu rzeczywistego, zaprogramowany przez producenta urządzenia, oraz program napisany przez użytkownika, według którego przebiega algorytm sterowania procesem i polega to na wypracowaniu odpowiedzi (w postaci sygnałów np. binarnych, ciągłych(analogowych)), na podstawie danych wejściowych (również w formie sygnałów binarnych, ciągłych), lub innej interakcji z otoczeniem,np. poprzez komunikację z innym systemem.Dzięki zastosowaniu systemu operacyjnego czasu rzeczywistego mamy pewność, że wypracowanie odpowiedzi zajmie sterownikowi ściśle określony czas, nie większy niż maksymalny (najdłuższy) założony czas. Czas ten będzie zależał od mocy obliczeniowej sterownika, jego peryferii oraz od długości napisanego programu. Wykorzystanie sterowników PLC w dużym stopniu ułatwia wprowadzanie zmian w sterowaniu, analizowanie algorytmu i wprowadzanie zmian, ponieważ każdy sterownik posiada oprogramowanie, (edytor), w którym programista pisze program sterowania, ładuje program do sterownika, ale ma również możliwość jego podglądania. Nawet przy małych maszynach i urządzeniach, gdzie sterownik jest zawsze alternatywą, dla techniki przekaźnikowej, stosowanie sterownika jest lepszym rozwiązaniem, gdyż praktyka pokazuje, że z upływem czasu zawsze użytkownik widzi konieczność wprowadzenia zmian - co w technice przekaźnikowej wymaga dołożenia kolejnych przekaźników, dodatkowych połączeń (na co nie zawsze jest miejsce). W rozwiązaniu ze sterownikiem PLC wymaga to przeważnie zmiany sekwencji programu.

2. Cel ćwiczenia.

Zaprojektować układ sterowania silnikiem windy. Winda porusza się pomiędzy dwoma piętrami (kondygnacjami). Winda powinna kursować między 1 i 3 kondygnacją. Zadany jest stan początkowy (zachowanie windy w momencie uruchomienia programu na sterowniku). Stan początkowy - zjazd na 1 kondygnację.

3. Wykonanie ćwiczenia.

Pierwszym krokiem było wykonanie funkcji przejść na podstawie tabeli stanów:

Funkcja przejść

I_2	I_4	I_6	I_8	M_1	M_2	M_3	M_4	M_1'	M_2'	M_3'	M_4'
*	*	0	*	0	0	0	0	0	0	0	0
*	*	1	*	0	0	0	0	1	0	0	0
*	0	*	*	1	*	*	*	1	0	0	0
*	1	*	*	1	*	*	*	0	1	0	0
*	*	*	0	*	1	*	*	0	1	0	0
*	*	*	1	*	1	*	*	0	0	1	0
0	*	*	*	*	*	1	*	0	0	1	0
1	*	*	*	*	*	1	*	0	0	0	1
*	*	0	*	*	*	*	1	0	0	0	1
*	*	1	*	*	*	*	1	1	0	0	0

Tabela stanów

Stan	M_1	M_2	M_3	M_4	Q_1	Q_2
Start	0	0	0	0	1	0
3	1	0	0	0	0	1
1 -> 3	0	1	0	0	0	0
3	0	0	1	0	0	1
3 ->1	0	0	0	1	1	0

Następnie utworzyłem wyrażenia boolowskie:

$M_1' = I_6 \sim(M_1) \sim(M_2) \sim(M_3) \sim(M_4) + \sim(I_4) M_1 + I_6 M_4$

$M_2' = I_4 M_1 + \sim(I_8) M_2$

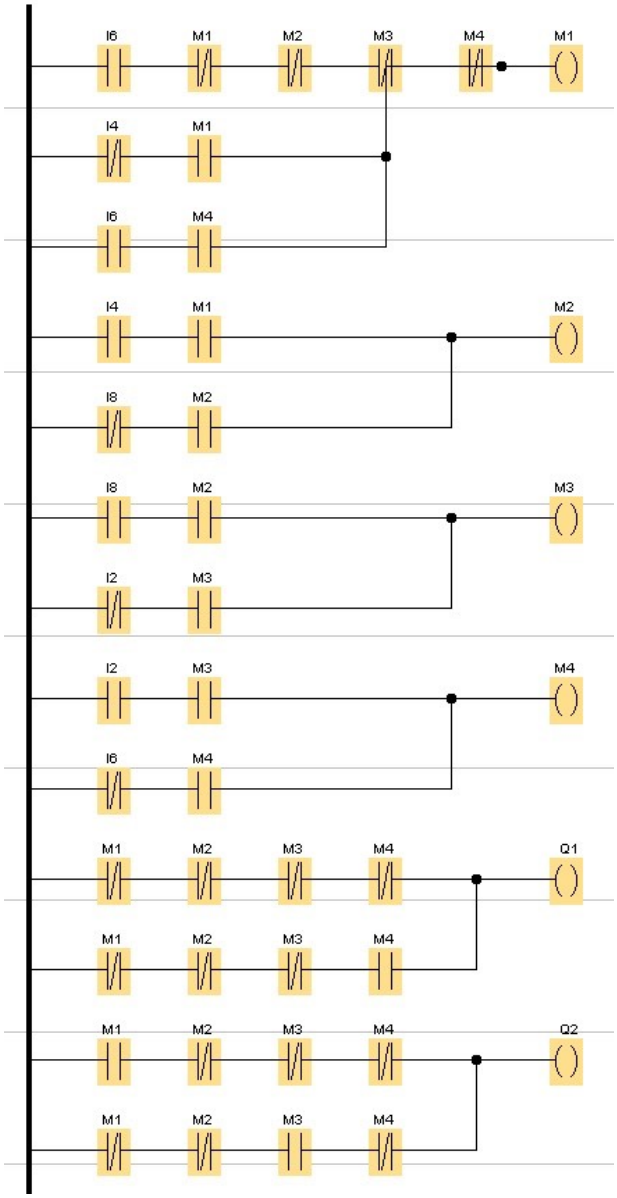
$M_3' = I_8 M_2 + \sim(I_2) M_3$

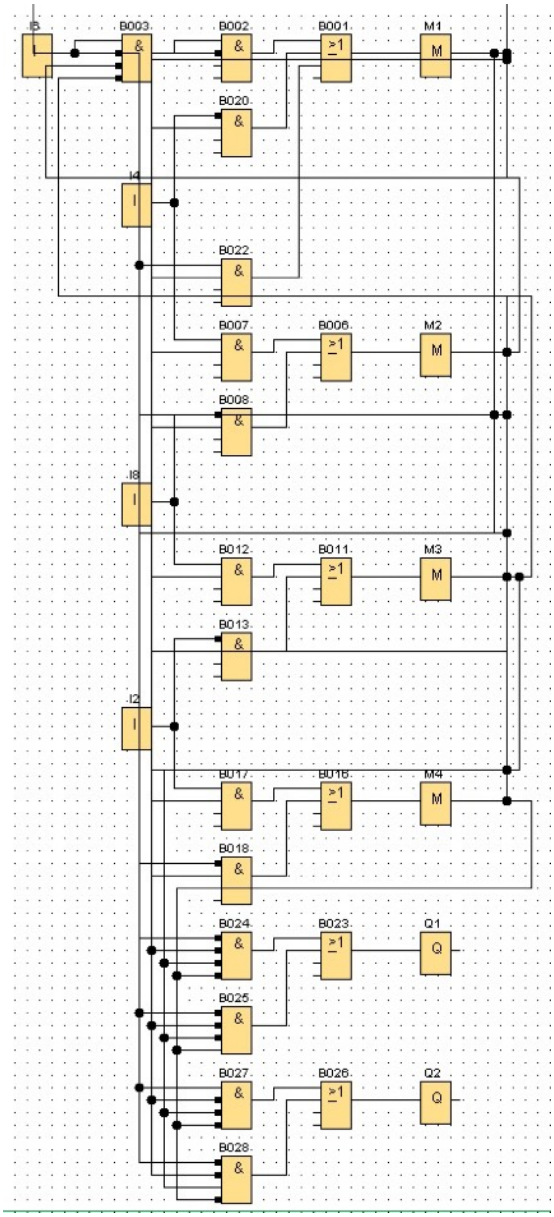
$M_4' = I_2 M_3 + \sim(I_6) M_4$

$Q_1 = \sim(M_1) \sim(M_2) \sim(M_3) \sim(M_4) + \sim(M_1) \sim(M_2) \sim(M_3) M_4 = \sim(M_1) \sim(M_2) \sim(M_3)$

$Q_2 = M_1 \sim(M_2) \sim(M_3) \sim(M_4) + \sim(M_1) \sim(M_2) M_3 \sim(M_4)$

Kolejnym krokiem była implementacja w programie LOGO! Soft Comfort w językach LD oraz FBD.





Po poprawnym wgraniu programów do sterownika PLC winda działała poprawnie i zgodnie z oczekiwanymi wynikami.

Ostatnim krokiem było zaimplementowanie tego samego działania windy w środowisku Arduino.

Oznaczenia czujników i przycisków została wprowadzona zgodnie ze schematem pobranym ze strony prof. Kwiatkowskiego.

Treść programu:

```
#define czyRaport 1
#define lag 0
//Przyłacza Arduino
int Button0Pin =A0;//numer pinu dla "Przycisk żądanie 0-go piętra"
int Button1Pin =A1;//numer pinu dla "Przycisk żądanie 1-go piętra"
int Button2Pin =A2;//numer pinu dla "Przycisk żądanie 2-go piętra"
int Button3Pin =A3;//numer pinu dla "Przycisk żądanie 3-go piętra"
int Sensor0Pin =A4;//numer pinu dla "Sensor obecności kabiny na 0-szym piętrze"
int Sensor1Pin =A5;//numer pinu dla "Sensor obecności kabiny na 1-szym piętrze"
int Sensor2Pin =12;//numer pinu dla "Sensor obecności kabiny na 2-gim piętrze"
int Sensor3Pin =13;//numer pinu dla "Sensor obecności kabiny na 3-im piętrze"
int Relay4Pin=4;//numer pinu dla Control Signal of Relay 4 / Output Signal Q2 / Start/Stop Signal
int Relay3Pin=5;//numer pinu dla Control Signal of Relay 3 / Output Signal Q1 / Forward/ Reverse Signal
//Zmienne globalne
boolean Q1 = 1;//Forward/Reverse Control Signal Relay 3
boolean Q2 = 0;//Start/Stop Control Signal Relay 4
boolean IP0 = 0;//Odczyt z "Przycisk żądanie 0-go poziomu"
boolean IP1 = 0;//Odczyt z "Przycisk żądanie 1-go poziomu"
boolean IP2 = 0;//Odczyt z "Przycisk żądanie 2-go poziomu"
boolean IP3 = 0;//Odczyt z "Przycisk żądanie 3-go poziomu"
boolean IS0 = 0;//Odczyt z "Sensor obecności kabiny na 0. poziomie"
boolean IS1 = 0;//Odczyt z "Sensor obecności kabiny na 1. poziomie"
boolean IS2 = 0;//Odczyt z "Sensor obecności kabiny na 2. poziomie"
boolean IS3 = 0;//Odczyt z "Sensor obecności kabiny na 3. poziomie"

boolean M1 = 0;//Flaga stanu
boolean M2 = 0;
boolean M3 = 0;
boolean M4 = 0;

void raport()
{
  Serial.println("Odczyt przyciskow  ");
  Serial.print(IP0);
  Serial.print(IP1);
  Serial.print(IP2);
  Serial.print(IP3);
  Serial.println();
  Serial.println("Odczyt sensorow  ");
  Serial.print(IS0);
  Serial.print(IS1);
  Serial.print(IS2);
  Serial.print(IS3);
  Serial.println();
```

```
}

void odczytWejsc()
{
  IP0 = digitalRead(Button0Pin);
  IP1 = digitalRead(Button1Pin);
  IP2 = digitalRead(Button2Pin);
  IP3 = digitalRead(Button3Pin);
  IS0 = digitalRead(Sensor0Pin);
  IS1 = digitalRead(Sensor1Pin);
  IS2 = digitalRead(Sensor2Pin);
  IS3 = digitalRead(Sensor3Pin);
}

void funkcjaPrzejscia()
{
  boolean M1prim = 0;//Obliczana, nowa wartość flagi
  boolean M2prim = 0;
  boolean M3prim = 0;
  boolean M4prim = 0;
// ..............................
  //Obliczenie wartości funkcji przejśc stanów (flag)
  //Np.
  M1prim = IS1&!M1&!M2&!M3&!M4 | !IP3&M1 | IS1&M4;
  M2prim = IP3&M1 | !IS3&M2;
  M3prim = IS3&M2 | !IP1&M3;
  M4prim = IP1&M3 | !IS1&M4;
//  ............................
  //Przepisanie "nowych" wartości do "starych"
  M1 = M1prim;
  M2 = M2prim;
  M3 = M3prim;
  M4 = M4prim;
}

void funkcjaWyjscia()
{
  Q1 = !M1&!M2&!M3&!M4 | !M1&!M2&!M3&M4;
  Q2 = M1&!M2&!M3&!M4 | !M1&!M2&M3&!M4;
 }

void zapisWyjscia()
{
  digitalWrite(Relay3Pin,Q1);
  digitalWrite(Relay4Pin,Q2);
}

void setup()
{
  pinMode(Button0Pin, INPUT);
  pinMode(Button1Pin, INPUT);
  pinMode(Button2Pin, INPUT);
  pinMode(Button3Pin, INPUT);
  pinMode(Sensor0Pin, INPUT);
  pinMode(Sensor1Pin, INPUT);
  pinMode(Sensor2Pin, INPUT);
  pinMode(Sensor3Pin, INPUT);
  pinMode(Relay3Pin, OUTPUT);
```

```
  pinMode(Relay4Pin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Koniec wykonywania Setup");
 }

void loop()
{

  odczytWejsc();
  funkcjaPrzejscia();
  funkcjaWyjscia();
  zapisWyjscia();
  raport();
}
```

Winda w przypadku mikrokontrolera Arduino również działała poprawnie i zgodnie z założeniami.

4. Wnioski.

Układ, który został zamodelowany na laboratorium działał poprawnie. To laboratorium pozwoliło nam utrwalić wiedzę na temat tworzenia funkcji przejść, wyjść oraz implementacji zadanego problemu w językach LD oraz FBD i Arduino. Dzięki takim ćwiczeniom można sobie uświadomić, iż znajomość działania na formułach boolowskich pozwala tworzyć proste projekty, z których korzystamy niemal codziennie.