МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА
Институт кибернетики
Кафедра проблем управления

Лабораторная работа №2

по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

Тема: Программное обеспечение системы управления мехатронного модуля управления защитной дверью

Работу выполнил
студент КРБО-01-17:
Денисов Д.С.

Преподаватель:
Морозов А.А.

Москва 2020

Цель работы: получение навыков построения программного обеспечения промышленных систем управления на базе функциональных блоков и конечных автоматов

Задание: разработать программное обеспечение системы автоматического управления приводом защитной двери. Схема механизма представлена на рисунке 1. Дверь оснащена асинхронным двигателем и четырьмя датчиками положения ($S0$ – $S3$), которые реагируют на пластину, обозначенную крестиком. Открытие и закрытие двери управляется тумблером.

При включении системы управления дверь должна двигаться в заданную сторону на небольшой скорости для определения своего местоположения. В этом режиме необходимо, чтобы индикаторы мерцали с частотой 1 Гц. После чего происходит переход в рабочий режим.

В рабочем режиме обеспечить максимально возможную скорость движения на отрезке $s1s2$, небольшую скорость на отрезках $s0s1$ и $s2s3$ (для обеспечения безопасности движения). Индикаторы должны показывать местоположение двери.

Система управления ворот должна быть выполнена в виде функционального блока, предполагающего повторное использование.

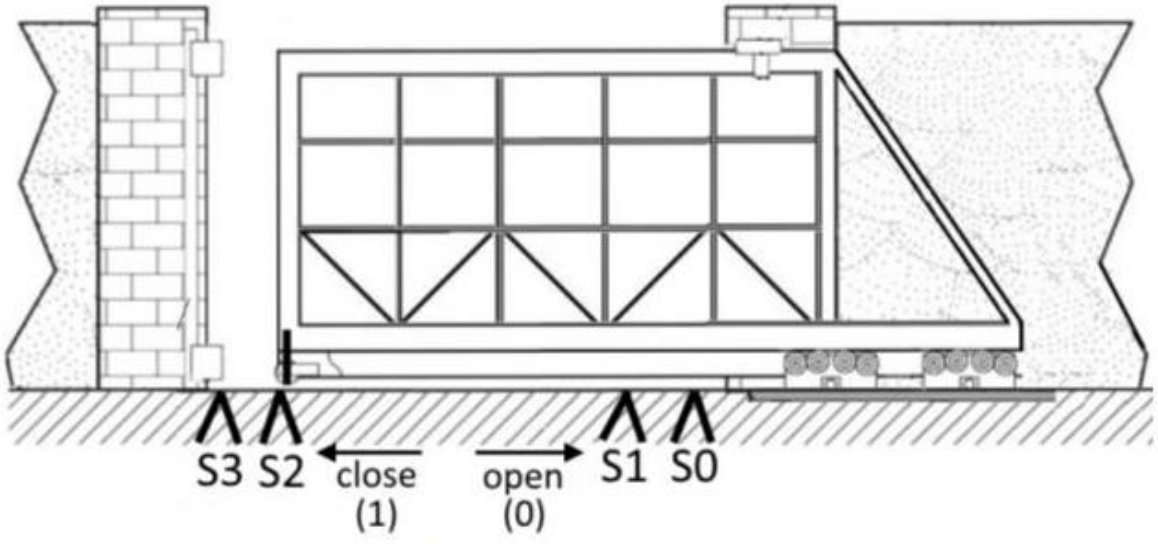


Рисунок 1. Устройство ворот

Ход работы:

Был создан новый проект в среде Automation Studio, конфигурация оборудования представлена на рисунках 2 и 3.

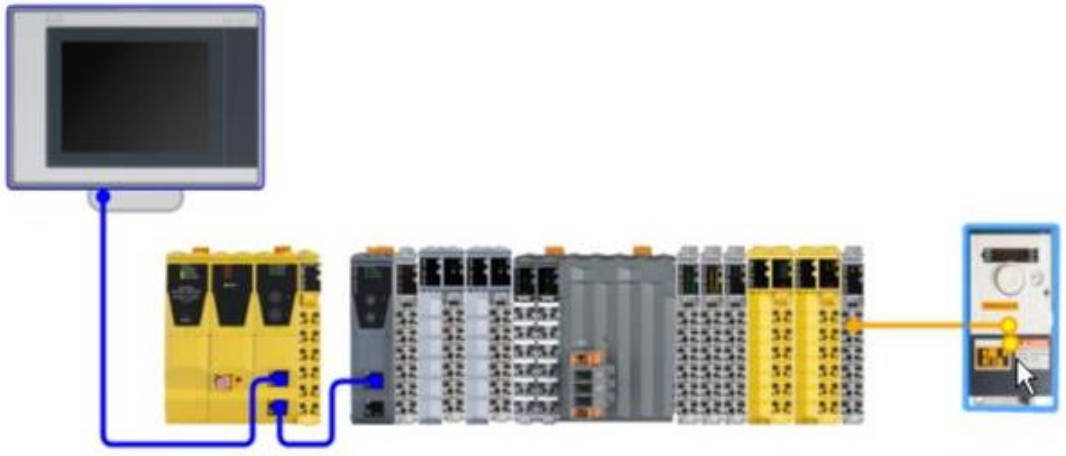


Рисунок 2. Конфигурация оборудования

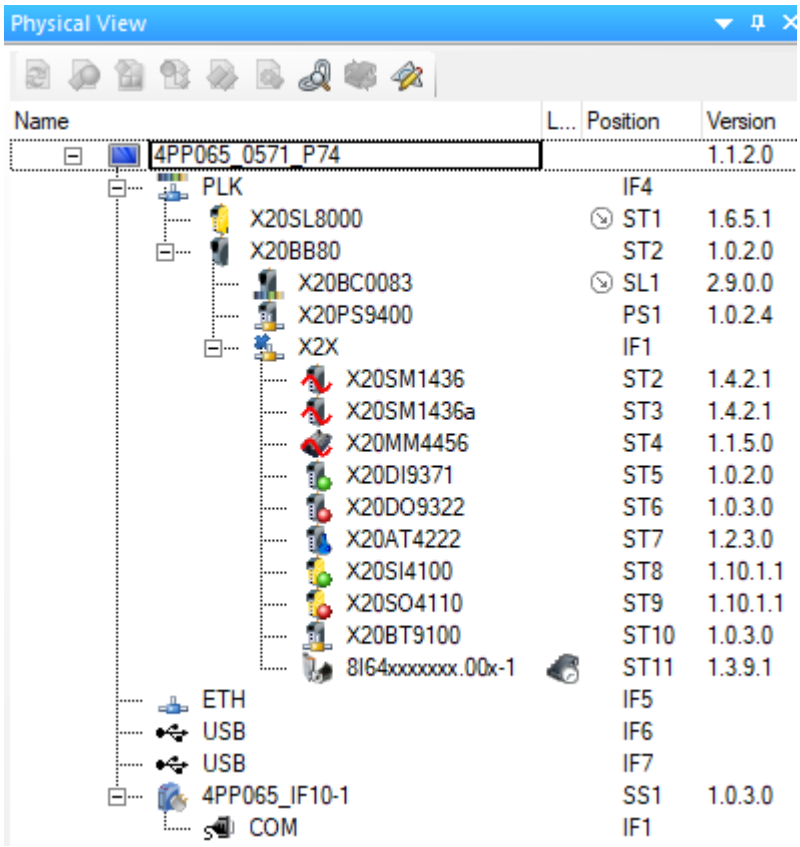


Рисунок 3. Список используемых компонентов

Затем в пункте Configuration меню Physical View были заданы параметры асинхронного двигателя, указанные в таблице 1.

Таблица 1. Параметры асинхронного двигателя

Наименование параметра	Название в конфигурационной таблице [eng.]	Заданное значение
Номинальное питающее напряжение двигателя [В]	UNS Rated Motor volt [V]	220
Номинальная питающая частота [0.1 Гц]	FRS Rated motor freq [0.1 Hz]	500
Сопротивление статора [мОм]	RSC Cold Stator resist [mOhm]	33000
Коэффициент мощности	COS Motor 1 Cosinus Phi [0.01]	64
Номинальная скорость вращения двигателя [об/мин]	NSP Rated motor speed [rpm]	890
Номинальный ток [0.1 А]	NCR Rated motor current [0.1 A]	10

В библиотеке «DriveLib» были созданы 3 функциональных блока:

1) «DriveStateMachine» –функциональный блока управления частотным преобразователем. Описание структуры приведено в таблице 2.

Таблица 2. Параметры функционального блока «DriveStateMachine»

Конфигурация	**Имя**	**Тип данных**	**Описание**
вход	u	REAL	входное напряжение [В]
выход	w	REAL	частота вращения [об/мин]
выход	phi	REAL	положение [рад]
внутреннее состояние	integrator	FB_Integrator	интегратор
внутреннее состояние	Tm	REAL	электромеханическая постоянная времени [с]
внутреннее состояние	ke	REAL	постоянная ЭДС двигателя [В•мин/об]
внутреннее состояние	dt	REAL	шаг расчета [с]

Блок-схема данного ФБ представлена на рисунке 4.

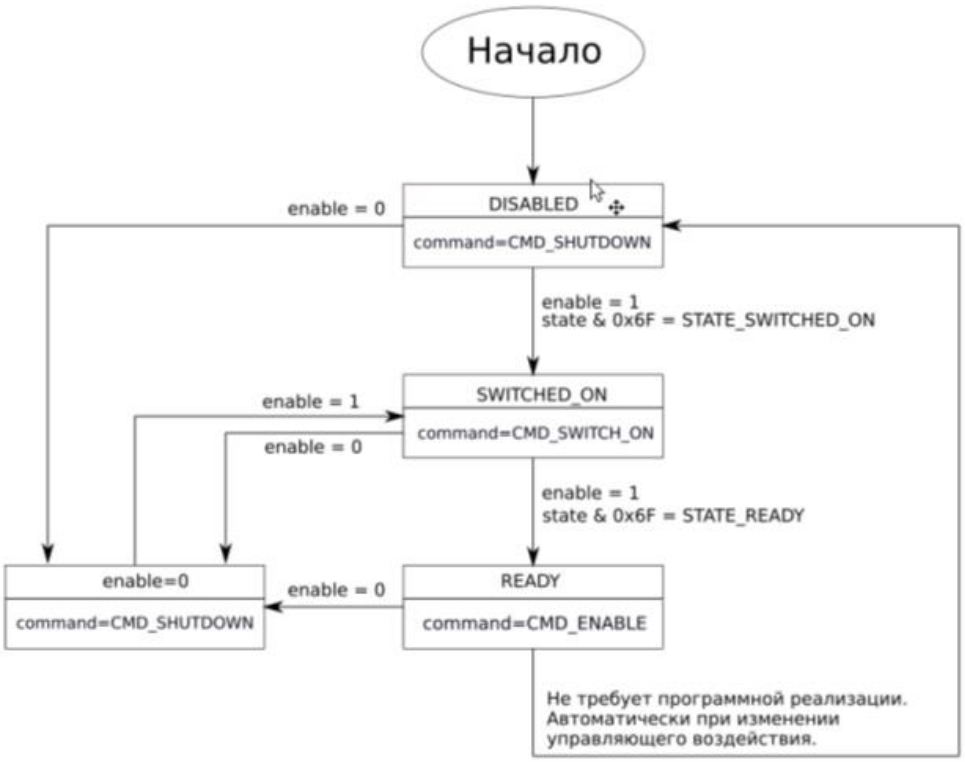


Рисунок 4. Блок схема «DriveStateMachine»

2) «DoorStateMachine» - функциональный блок, в котором реализована основная логика программы – задание направления вращения и скорости движения воротами в зависимости от их положения и требуемого направления вращения. Был создан новый тип данных: перечисление «DoorStates». Состояния ворот приведены в таблице 3.

Таблица 3. Состояния ворот

Состояние	Описание
ST_INIT	Инициализация параметров и ожидание включения частотного преобразователя
ST_UNKNOWN	Ворота в неизвестном положении
ST_OPEN	Ворота открыты
ST_CLOSE	Ворота закрыты
ST_ACC_POS	Ускорение ворот в сторону открытия
ST_ACC_NEG	Ускорение ворот в сторону закрытия
ST_POS	Движение к открытию
ST_NEG	Движение к закрытию
ST_DEC_POS	Замедление ворот в сторону открытия
ST_DEC_NEG	Замедление ворот в сторону закрытия

Блок-схема функционального блока «DoorStateMachine» имеет следующий вид (рисунок 5):

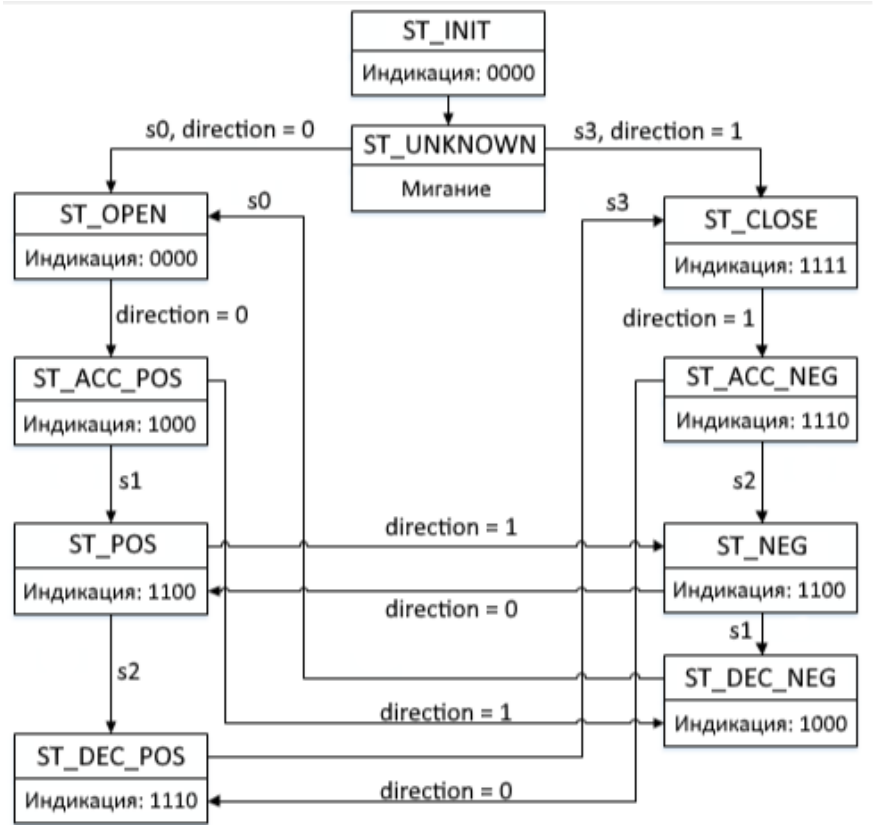


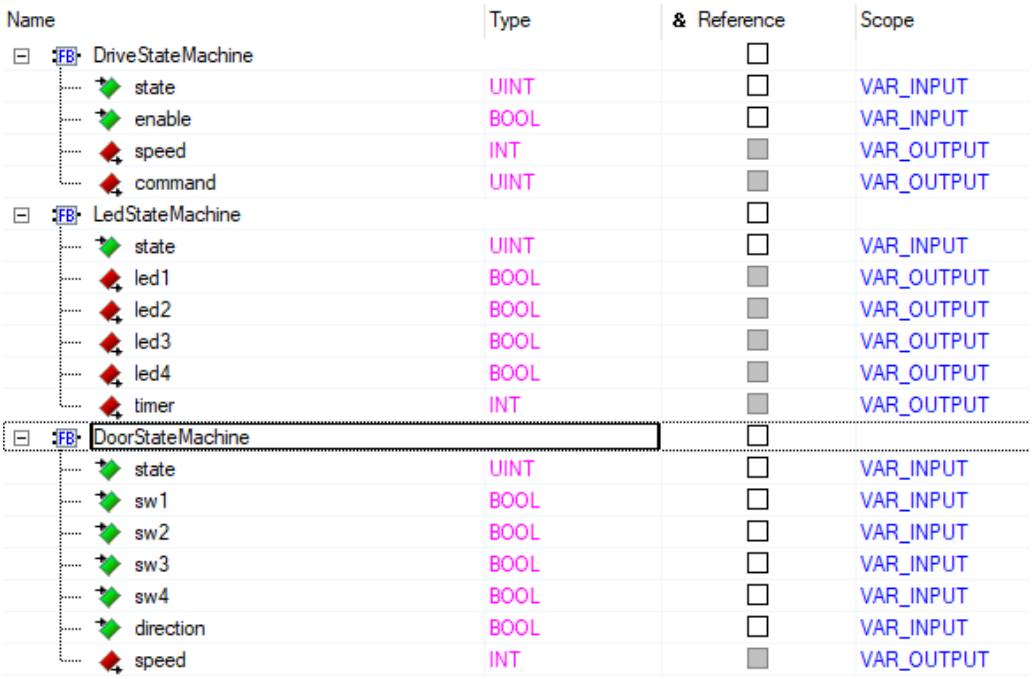
Рисунок 5. Блок-схема функционального блока «DoorStateMachine»

3) «LedStateMachine» – машина состояний обработки светодиодных индикаторов. Изменение состояния индикаторов представлено на блок-схеме (рисунок 5).
Структура ФБ имеет следующий набор переменных (таблица 4):

Таблица 4. Структура функционального блока управления воротами

Конфигурация	Имя	Тип данных	Описание
вход	state	UINT	Состояние частотного преобразователя
выход	led1	BOOL	Сигнал работы функционального блока
выход	led2	BOOL	Сигнал работы функционального блока
выход	led3	BOOL	Сигнал работы функционального блока
выход	led4	BOOL	Сигнал работы функционального блока
выход	timer	INT	Заданная скорость

Таким образом, в проект были добавлены функциональные блоки, параметры которых отображены на рисунке 6.



Name	Type	& Reference	Scope
DriveStateMachine		☐	
state	UINT	☐	VAR_INPUT
enable	BOOL	☐	VAR_INPUT
speed	INT	☐	VAR_OUTPUT
command	UINT	☐	VAR_OUTPUT
LedStateMachine		☐	
state	UINT	☐	VAR_INPUT
led1	BOOL	☐	VAR_OUTPUT
led2	BOOL	☐	VAR_OUTPUT
led3	BOOL	☐	VAR_OUTPUT
led4	BOOL	☐	VAR_OUTPUT
timer	INT	☐	VAR_OUTPUT
DoorStateMachine		☐	
state	UINT	☐	VAR_INPUT
sw1	BOOL	☐	VAR_INPUT
sw2	BOOL	☐	VAR_INPUT
sw3	BOOL	☐	VAR_INPUT
sw4	BOOL	☐	VAR_INPUT
direction	BOOL	☐	VAR_INPUT
speed	INT	☐	VAR_OUTPUT

Рисунок 6. Описание функциональных блоков

Далее была разработана программа Program, которая осуществляет цикличный вызов функциональных блоков. Скорость и команда на «DriveStateMachine» устанавливается в соответствии со скоростью «DoorStateMachine». Текущее состояние ворот также передается в «LedStateMachine».

Затем был проведен эксперимент, подтверждающий работоспособность системы управления, который представляет из себя полный цикл управления дверью: от подачи питания до отработки команд на открытие и закрытие. В доказательство успешной работы были сняты показаний Trace (рисунки 7 и 8) по следующим величинам:

- Скорость вращения приводного мотора;

- Состояние датчиков (sw 1-4);

- Состояние светодиодных индикаторов;

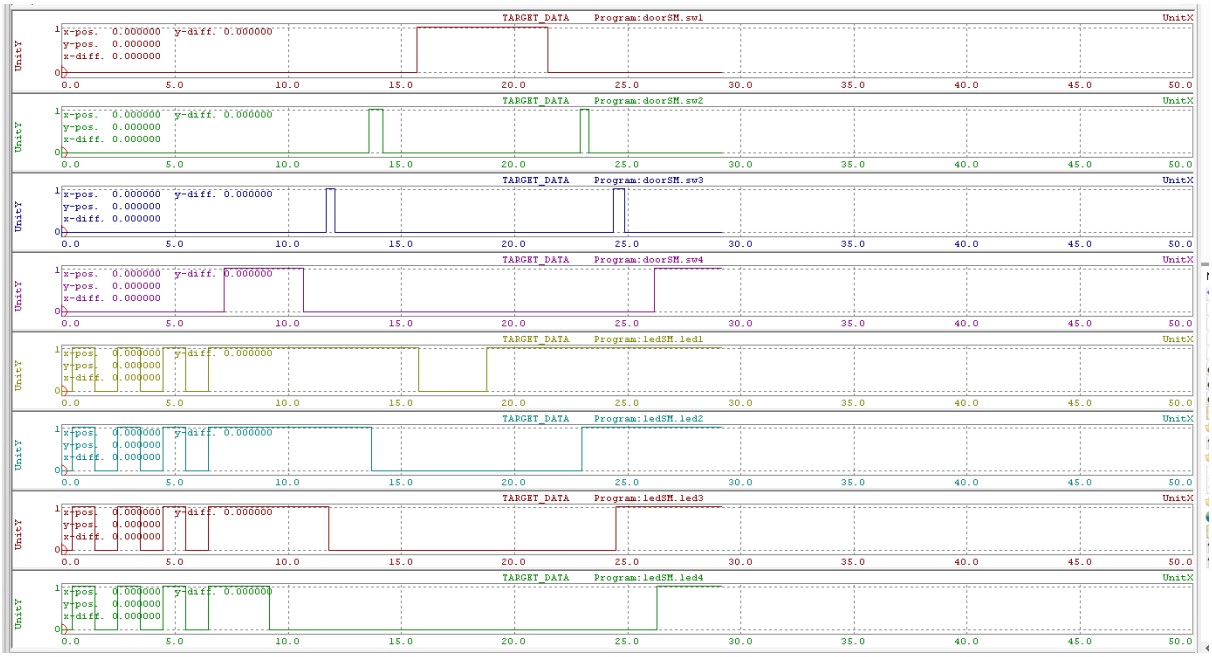


Рисунок 7. Зависимость состояний датчиков и светодиодных индикаторов от времени

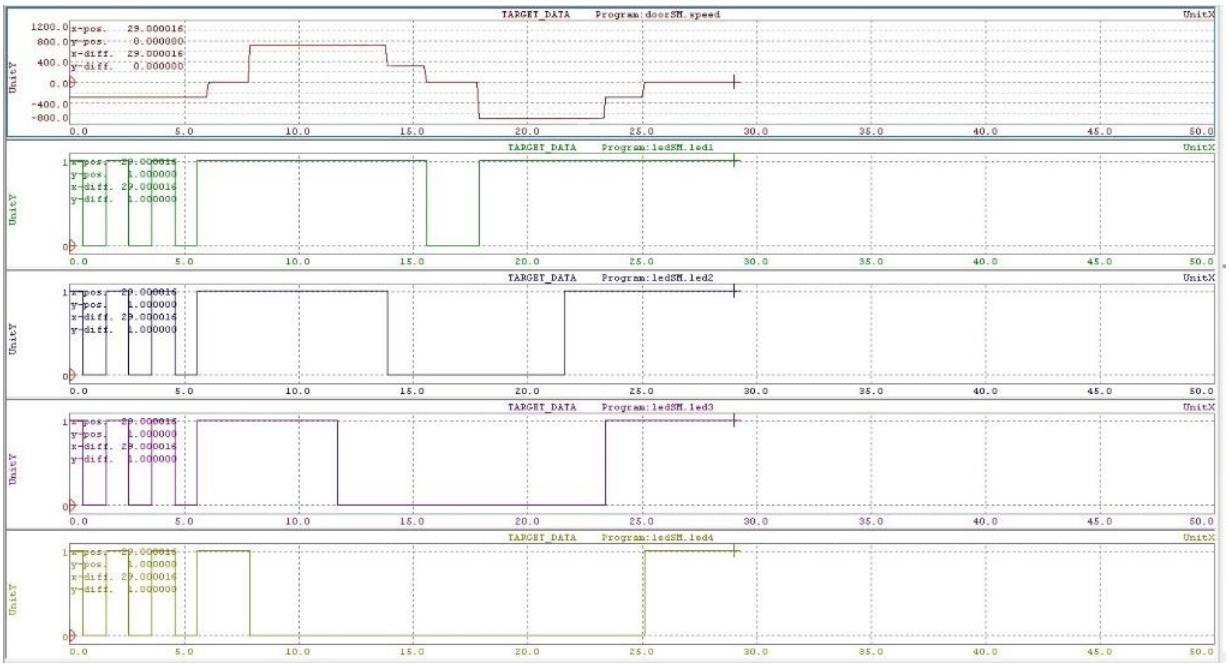


Рисунок 8. Изменение скорости вращения двигателя и состояний светодиодных индикаторов в ходе эксперимента

Вывод: получены навыки построения программного обеспечения промышленных систем управления на базе функциональных блоков и конечных автоматов. В ходе выполнения работы был проведен эксперимент, показывающий работоспособность системы управления воротами, оснащенными асинхронным двигателем.

Приложение А.

Листинг программного кода функционального блока «DriveStateMachine».

```
#include <bur/plctypes.h>
#ifdef cplusplus
   extern "C"
   {
#endif
   #include "DriveLib.h"
#ifdef cplusplus
   };
#endif
void DriveStateMachine(struct DriveStateMachine inst)
{
   if(inst->enable) {
      UINT stateMask = inst->state & 0x6f;
      switch (stateMask) {
         case ST_READY:
            inst->command = CMD_ENABLE;
            break;
         case ST_DISABLED:
            inst->command = CMD_SHUTDOWN;
            break;
         case ST_SWITCHED_ON:
            inst->command = CMD_SWITCH_ON;
      }
   }
   else
      inst->command = CMD_SHUTDOWN;
}
```

Приложение Б.

Листинг программного кода функционального блока «DoorStateMachine».

```
#include <bur/plctypes.h>
#ifdef __cplusplus
	extern "C"
	{
#endif
	#include "DriveLib.h"
#ifdef __cplusplus
	};
#endif
void DoorStateMachine(struct DoorStateMachine* inst)
{
	switch(inst->state){
		case ST_INIT:
			break;
		case ST_UNKNOWN:
			if(inst->direction){
				inst->speed = -SLOW_SPEED;
				if(inst->sw4){
					setStateAndChangeSpeed(inst,ST_CLOSE);
				}
			}
			else{
				inst->speed = SLOW_SPEED;
				if(inst->sw1){
					setStateAndChangeSpeed(inst,ST_OPEN);
				}
			}
			break;
		case ST_OPEN:
			if(inst->direction){
				setStateAndChangeSpeed(inst,ST_ACC_NEG);
			}
			break;
		case ST_CLOSE:
			if(!inst->direction){
				setStateAndChangeSpeed(inst,ST_ACC_POS);
			}
			break;
		case ST_ACC_POS:
			if(inst->direction){
```

```
                setStateAndChangeSpeed(inst,ST_DEC_NEG);
        }
        else if(inst->sw3){
                setStateAndChangeSpeed(inst,ST_POS);
        }
        break;
case ST_ACC_NEG:
        if(!inst->direction){
                setStateAndChangeSpeed(inst,ST_DEC_POS);
        }
        else if(inst->sw2){
                setStateAndChangeSpeed(inst,ST_NEG);
        }
        break;
case ST_POS:
        if(inst->direction){
                setStateAndChangeSpeed(inst,ST_NEG);
        }
        else if(inst->sw2){
                setStateAndChangeSpeed(inst,ST_DEC_POS);
        }
        break;
case ST_NEG:
        if(!inst->direction){
                setStateAndChangeSpeed(inst,ST_POS);
        }
        else if(inst->sw3){
                setStateAndChangeSpeed(inst,ST_DEC_NEG);
        }
        break;
case ST_DEC_POS:
        if(inst->direction){
                setStateAndChangeSpeed(inst,ST_ACC_NEG);
        }
        else if(inst->sw1){
                setStateAndChangeSpeed(inst,ST_OPEN);
        }
        break;
case ST_DEC_NEG:
        if(!inst->direction){
                setStateAndChangeSpeed(inst,ST_ACC_POS);
        }
        else if(inst->sw4){
                setStateAndChangeSpeed(inst,ST_CLOSE);
```

```
                }
                break;
            default:
                break;
        }
    }

    void setStateAndChangeSpeed(struct DoorStateMachine* inst, UINT state)
{
        inst->state = state;
        switch(state){
            case ST_CLOSE:
                inst->speed = 0;
                break;
            case ST_OPEN:
                inst->speed = 0;
                break;
            case ST_ACC_POS:
                inst->speed = FAST_SPEED;
                break;
            case ST_DEC_POS:
                inst->speed = SLOW_SPEED;
                break;
            case ST_ACC_NEG:
                inst->speed = -FAST_SPEED;
                break;
            case ST_DEC_NEG:
                inst->speed = -SLOW_SPEED;
                break;
            default:
                break;
        }
    }
```

Приложение В.

Листинг программного кода функционального блока «LedStateMachine».

```
#include <bur/plctypes.h>
#ifdef cplusplus
   extern "C"
   {
#endif
   #include "DriveLib.h"
#ifdef cplusplus
   };
#endif
void LedStateMachine(struct LedStateMachine inst)
{
   switch(inst->state){
      case ST_UNKNOWN:
         if(inst->timer==10){
            setLeds(inst,!inst->led1,!inst->led1,!inst->led1,!inst->led1);
            inst->timer=0;
         }
         break;
      case ST_CLOSE:
         setLeds(inst,1,1,1,1);
         break;
      case ST_INIT:
      case ST_OPEN:
         setLeds(inst,0,0,0,0);
         break;
      case ST_ACC_POS:
      case ST_DEC_NEG:
         setLeds(inst,1,1,1,0);
         break;
      case ST_POS:
      case ST_NEG:
         setLeds(inst,1,1,0,0);
         break;
      case ST_DEC_POS:
      case ST_ACC_NEG:
         setLeds(inst,1,0,0,0);
         break;
```

```
        }
        inst->timer++;
    }

    void setLeds(struct LedStateMachine* inst, BOOL led1, BOOL led2, BOOL
led3, BOOL led4){
        inst->led1 = led1;
        inst->led2 = led2;
        inst->led3 = led3;
        inst->led4 = led4;
    }
```