



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Кафедра вычислительной техники

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

Тема: «Программное обеспечение системы управления
мехатронного модуля управления защитной дверью»

Студент группы КРБО-03-17.

Орлов Д.Л.

A handwritten signature in black ink, appearing to be 'D.L. Orlov'.

Руководитель:

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 2.1. Дверь оснащена асинхронным двигателем и четырьмя датчиками положения ($s_0 - s_3$), 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 s_1s_2 , небольшую скорость на отрезках s_0s_1 и s_2s_3 (для обеспечения безопасности движения). Индикаторы должны показывать местоположение двери.

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

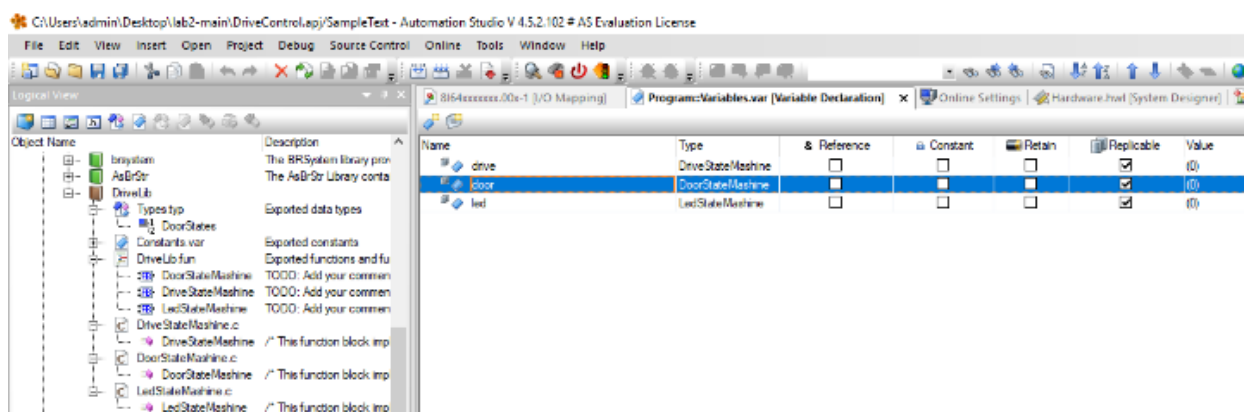
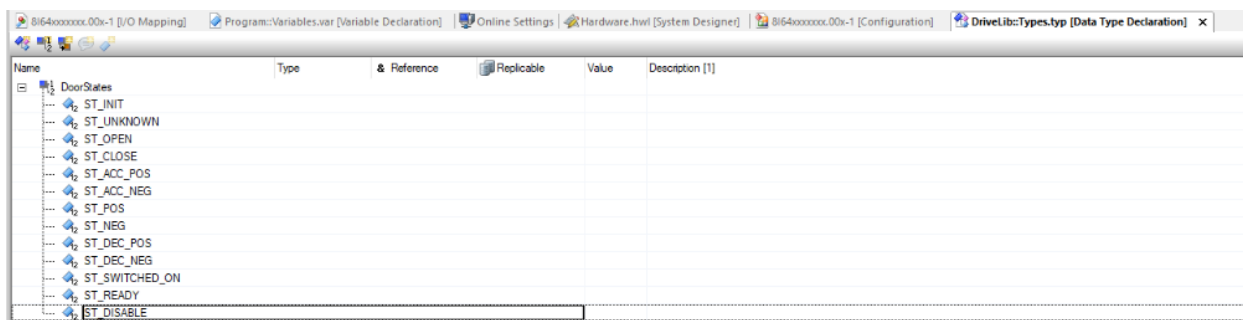


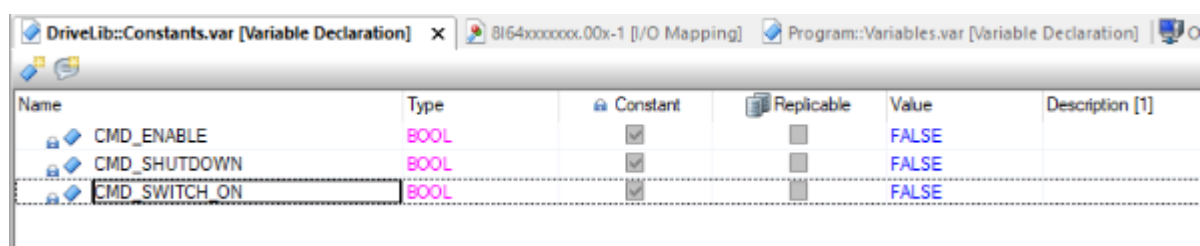
Рис. 1. Создание функциональных блоков.



The screenshot shows a software window with multiple tabs. The active tab is '8i64xxxxxx.00x-1 [I/O Mapping]'. Below the tabs is a table with the following columns: Name, Type, & Reference, Replicable, Value, and Description [1]. The table contains a list of variables under the 'DoorStates' category.

| Name | Type | & Reference | Replicable | Value | Description [1] |
|----------------|------|-------------|------------|-------|-----------------|
| DoorStates | | | | | |
| ST_INIT | | | | | |
| ST_UNKNOWN | | | | | |
| ST_OPEN | | | | | |
| ST_CLOSE | | | | | |
| ST_ACC_POS | | | | | |
| ST_ACC_NEG | | | | | |
| ST_POS | | | | | |
| ST_NEG | | | | | |
| ST_DEC_POS | | | | | |
| ST_DEC_NEG | | | | | |
| ST_SWITCHED_ON | | | | | |
| ST_READY | | | | | |
| ST_DISABLE | | | | | |

Рис. 2. Инициализация переменных.



The screenshot shows a software window with multiple tabs. The active tab is 'DriveLib::Constants.var [Variable Declaration]'. Below the tabs is a table with the following columns: Name, Type, Constant, Replicable, Value, and Description [1]. The table contains three command variables.

| Name | Type | Constant | Replicable | Value | Description [1] |
|---------------|------|-------------------------------------|--------------------------|-------|-----------------|
| CMD_ENABLE | BOOL | <input checked="" type="checkbox"/> | <input type="checkbox"/> | FALSE | |
| CMD_SHUTDOWN | BOOL | <input checked="" type="checkbox"/> | <input type="checkbox"/> | FALSE | |
| CMD_SWITCH_ON | BOOL | <input checked="" type="checkbox"/> | <input type="checkbox"/> | FALSE | |

Рис. 3. Инициализация команд.

Активация датчиков в рамке сдвижной двери отражается на состоянии светодиодов.

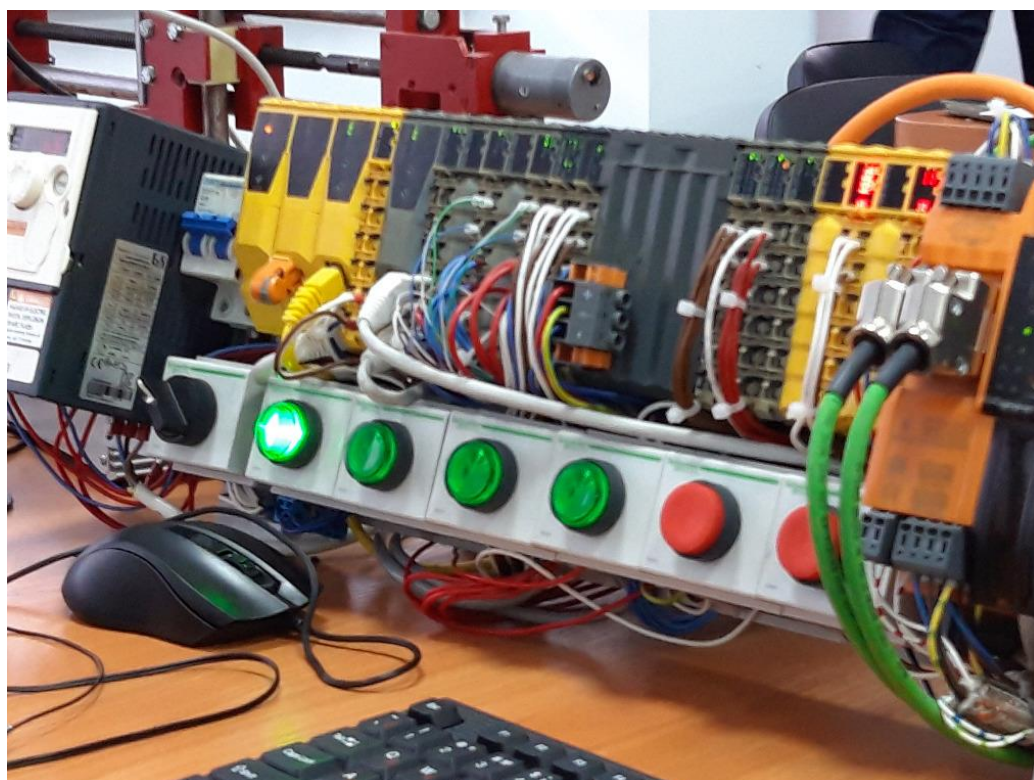


Рисунок 4. Достижение первого датчика.

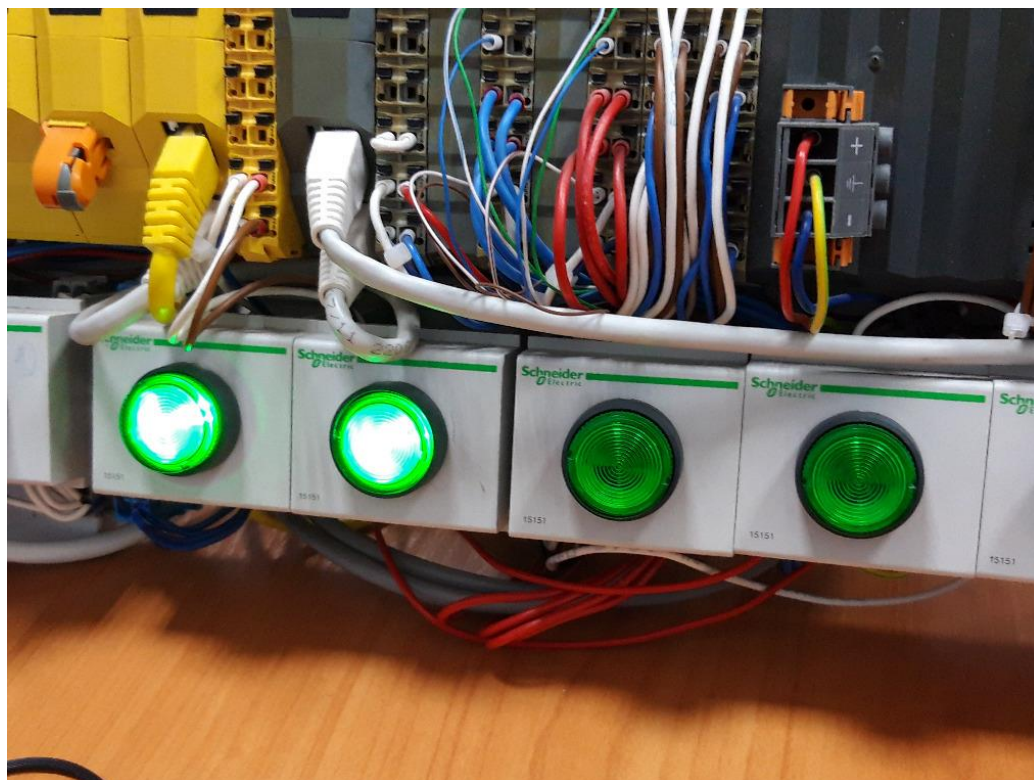


Рис. 5. Достижение второго датчика.

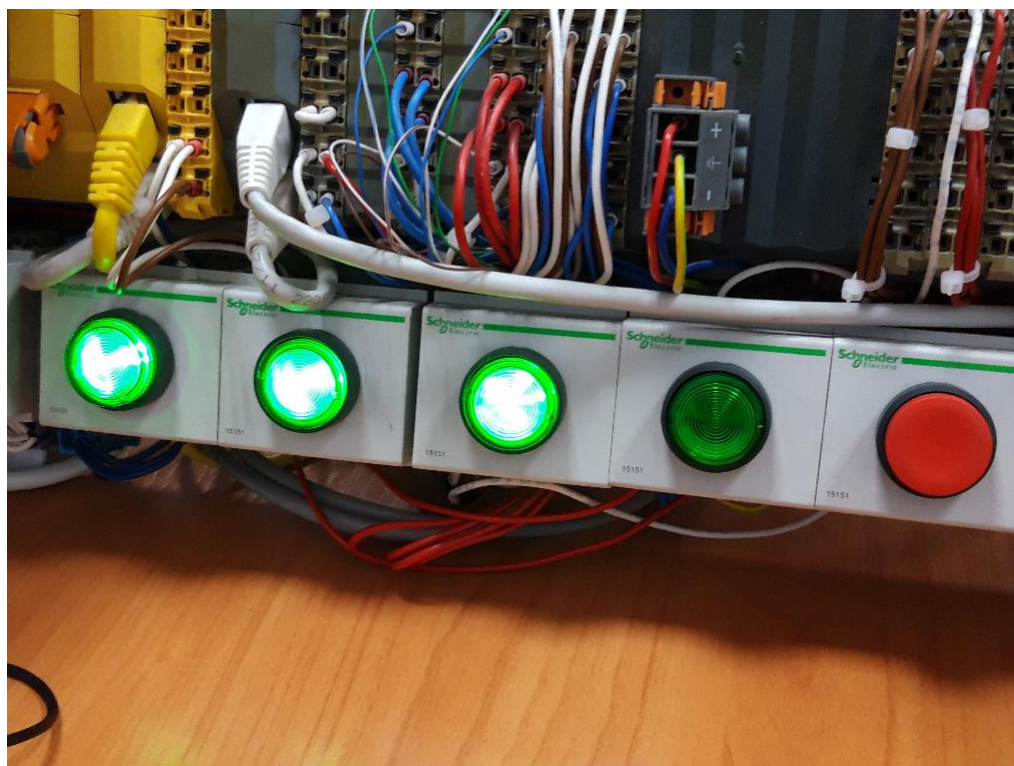


Рис. 6. Достижение третьего датчика.

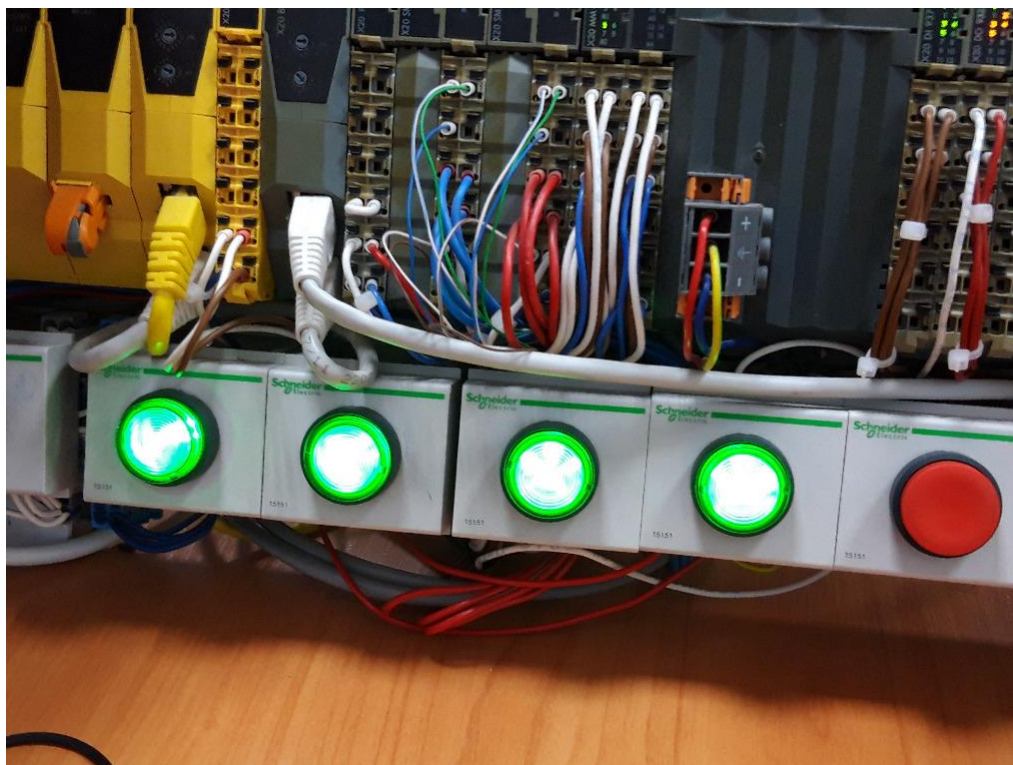


Рис. 7. Достижение четвёртого датчика.

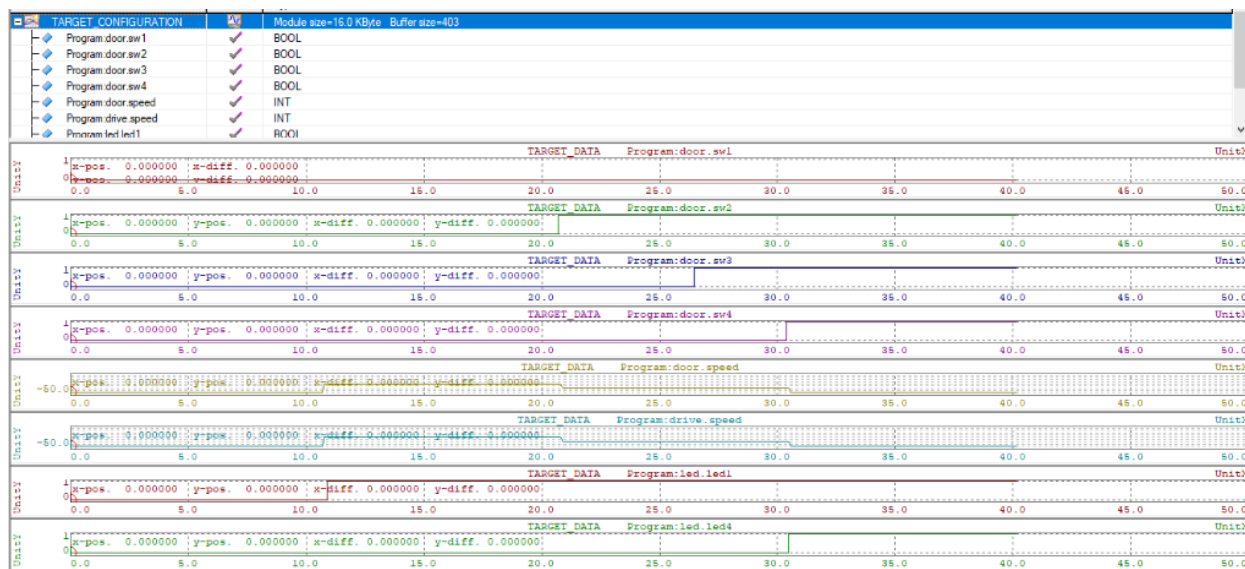


Рис. 8. Графики работы датчиков, скорости привода двери и состояния светодиодов (1, 4). При нажатии датчика 1 скорость двери скачкообразно увеличивается, а при достижении меток 2, 3 и 4 она шагобразно снижается.

По своей структуре конечный автомат использованного алгоритма является недетерминированным автоматом Мили.

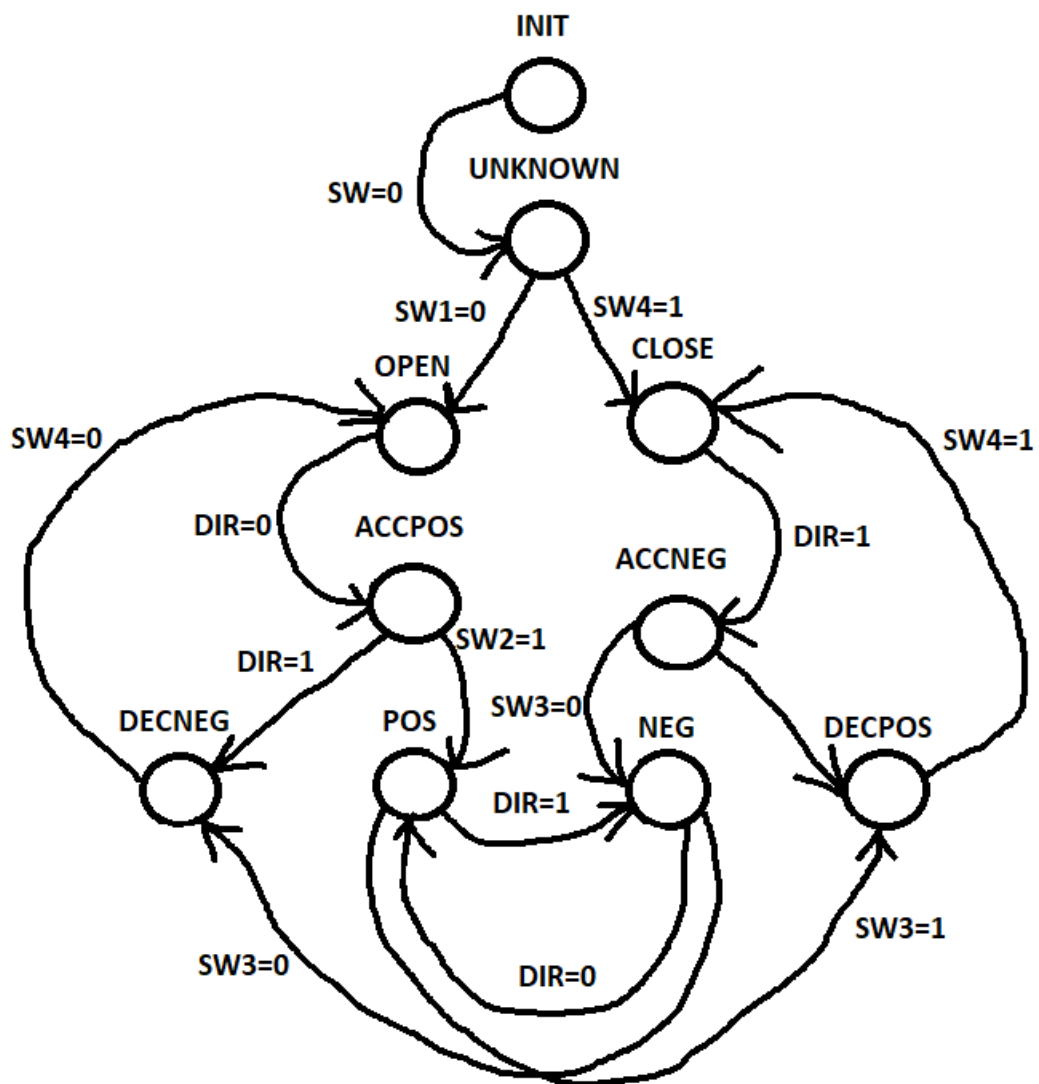


Рис. 9. Структурная схема конечного автомата двери.

Приложение 1. Листинг кода программы.

Main.c

```

#include <bur/plctypes.h>
#ifdef _DEFAULT_INCLUDES
#include <AsDefault.h>
#endif
void _INIT ProgramInit(void)
{

```

```

    drive.enable = 1;
}
void _CYCLIC ProgramCyclic(void)
{
    DoorStateMashine(&door);

    drive.speed = door.speed;
    DriveStateMashine(&drive);

    led.state = door.state;
    LedStateMashine(&led);
    led.timer++;
}
void _EXIT ProgramExit(void)
{
}

```

DriveStateMachine

```

#include <bur/plctypes.h>
#ifdef __cplusplus
    extern "C"
    {
#endif
#include "DriveLib.h"
#ifdef __cplusplus
    };
#endif
/* This function block implements state switching for motor control */
void DriveStateMashine(struct DriveStateMashine* inst)
{
    if(inst->enable == 0)
    {
        inst->command = CMD_SHUTDOWN;
    }
    else
    {
        switch(inst->state & 0x6F)
        {
            case ST_DISABLE:
                inst->command = CMD_SHUTDOWN;
                break;
            case ST_SWITCHED_ON:
                inst->command = CMD_SWITCH_ON;
                break;
            case ST_READY:
                inst->command = CMD_ENABLE;
                break;
        }
    }
}

```

```
}  
}
```

DoorStateMachine

```
#include <bur/plctypes.h>  
#ifdef __cplusplus  
    extern "C"  
    {  
#endif  
#include "DriveLib.h"  
#ifdef __cplusplus  
    };  
#endif  
/* This function block implements state switching for door control */  
void DoorStateMashine(struct DoorStateMashine* inst)  
{  
    switch(inst->state)  
    {  
        case ST_INIT:  
        {  
            if(inst->sw1 == 0 && inst->sw2 == 0 && inst->sw3 == 0 && inst->sw4 == 0) inst->state = ST_UNKNOWN;  
            break;  
        }  
        case ST_UNKNOWN:  
        {  
            if(inst->direction == 0 && inst->sw1 == 0) inst->state = ST_OPEN;  
            if(inst->direction == 1 && inst->sw4 == 1) inst->state = ST_CLOSE;  
            break;  
        }  
        case ST_OPEN:  
        {  
            inst->speed = 0;  
            if(inst->direction == 0) inst->state = ST_ACC_POS;  
            break;  
        }  
        case ST_CLOSE:  
        {  
            inst->speed = 0;  
            if(inst->direction == 1) inst->state = ST_ACC_NEG;  
            break;  
        }  
        case ST_ACC_POS:  
        {  
            inst->speed = 200;  
            if(inst->sw2 == 1) inst->state = ST_POS;  
            if(inst->direction == 1) inst->state = ST_DEC_NEG;  
            break;  
        }  
    }  
}
```



```

case ST_ACC_NEG:
{
    inst->speed = -200;
    if(inst->sw3 == 0) inst->state = ST_NEG;
    if(inst->direction == 0) inst->state = ST_DEC_POS;
    break;
}
case ST_POS:
{
    inst->speed = 100;
    if(inst->sw3 == 1) inst->state = ST_DEC_POS;
    if(inst->direction == 1) inst->state = ST_NEG;
    break;
}
case ST_NEG:
{
    inst->speed = -100;
    if(inst->sw3 == 0) inst->state = ST_DEC_NEG;
    if(inst->direction == 0) inst->state = ST_POS;
    break;
}
case ST_DEC_POS:
{
    inst->speed = 100;
    if(inst->sw4 == 1) inst->state = ST_CLOSE;
    break;
}
case ST_DEC_NEG:
{
    inst->speed = -100;
    if(inst->sw4 == 0) inst->state = ST_OPEN;
    break;
}
}

```

LedStateMachine

```

#include <bur/plctypes.h>
#ifdef __cplusplus
extern "C"
{
#endif
#include "DriveLib.h"
#ifdef __cplusplus
};
#endif
/* This function block implements state switching to control indicators */
void LedStateMashine(struct LedStateMashine* inst)
{
    switch(inst->state)
    {
        case ST_INIT:
        {

```

```

        break;
    }
case ST_UNKNOWN:
    {
        if(inst->timer%10 == 9)
        {
            inst->led1 = !inst->led1;
            inst->led2 = !inst->led2;
            inst->led3 = !inst->led3;
            inst->led4 = !inst->led4;
        }
        break;
    }
case ST_OPEN:
    {
        inst->led1 = 0;
        inst->led2 = 0;
        inst->led3 = 0;
        inst->led4 = 0;
        break;
    }
case ST_CLOSE:
    {
        inst->led1 = 1;
        inst->led2 = 1;
        inst->led3 = 1;
        inst->led4 = 1;
        break;
    }
case ST_ACC_POS:
    {
        if(inst->timer%10 == 4) inst->led1 = 1;
        inst->led2 = 0;
        inst->led3 = 0;
        inst->led4 = 0;
        break;
    }
case ST_ACC_NEG:
    {
        inst->led1 = 1;
        inst->led2 = 1;
        inst->led3 = 1;
        if(inst->timer%10 == 4) inst->led4 = 0;
        break;
    }
case ST_POS:
    {
        inst->led1 = 1;
        if(inst->timer%10 == 9) inst->led2 = 1;
        inst->led3 = 0;
        inst->led4 = 0;
        break;
    }

```

```

    }
case ST_NEG:
{
    inst->led1 = 1;
    inst->led2 = 1;
    if(inst->timer% 10 == 9) inst->led3 = 0;
    inst->led4 = 0;
    break;
}
case ST_DEC_POS:
{
    inst->led1 = 1;
    inst->led2 = 1;
    if(inst->timer% 10 == 4) inst->led3 = 1;
    inst->led4 = 0;
    break;
}
case ST_DEC_NEG:
{
    inst->led1 = 1;
    if(inst->timer% 10 == 4) inst->led2 = 0;
    inst->led3 = 0;
    inst->led4 = 0;
    break;
}
}
}

```

Вывод. В ходе лабораторной работы был осуществлён алгоритм управления сдвижной дверью в реальном времени при помощи логики конечных автоматов.