|  |
| --- |
|  |
| 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 работы:** Программное обеспечение системы управления мехатронного модуля управления защитной дверью

|  |  |
| --- | --- |
| **Студент группы** КРБО-01-17 | Хмелевский В.С.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Преподаватель:** | ассистент Морозов А. А.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

Москва 2020

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

**Время на выполнение:** 4 часа.

**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 ( – ), которые реагируют на пластину, обозначенн/ую 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 , небольшую скорость на отрезках и (для обеспечения безопасности движения). Индикаторы должны показывать местоположение двери.



Система управления ворот должна быть выполнена в виде функциональных блоков, которые могут использоваться в подобных проектах.

**Ход работы**

Для управления электроприводом двери были созданы функциональные блоки DriveStateMachine, DoorStateMachine, LedStateMachine, опрделены входы, выходы и другие переменные (Рисунок 1).

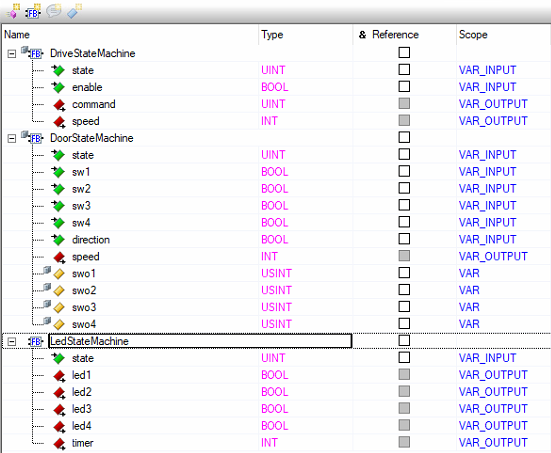


Рисунок 1. Создание функциональных блоков.

В DoorStateMashine реализована основная логика программы – задание направления вращения и скорости движения воротами в зависимости от их положения и требуемого направления вращения.

LedStateMashine – машина состояний обработки светодиодных индикаторов.

DriveStateMashine – заготовка функционального блока управления частотным преобразователем.

Также был создан список состояний двери DoorStates и список команд driveCommands, на которые реагирует система (Рисунок 2).

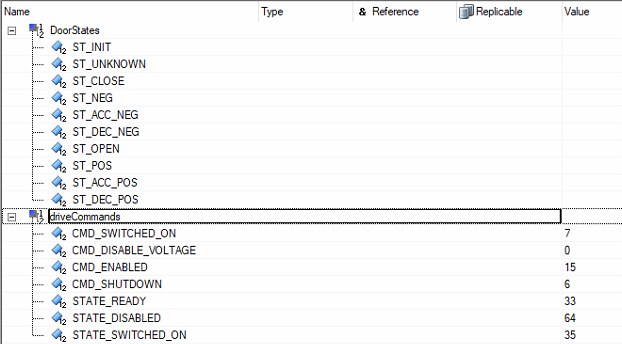


Рисунок 2. Создание машины состояний и инициализация переменных

При нажатии на кнопку в систему подается команда на изменение состояния (аналог активации датчика), что приводит к изменению скорости и/или направления. С помощью вращающегося тумблера можно переключить направление вращения двигателя.

Активация датчиков в рамке сдвижной двери отражается на состоянии светодиодов (Рисунки 3-7).

С помощью инструмента Trace проверяем состояние светодиодов (Рисунок 8), датчиков и скорость вращения двигателя (Рисунок 9).

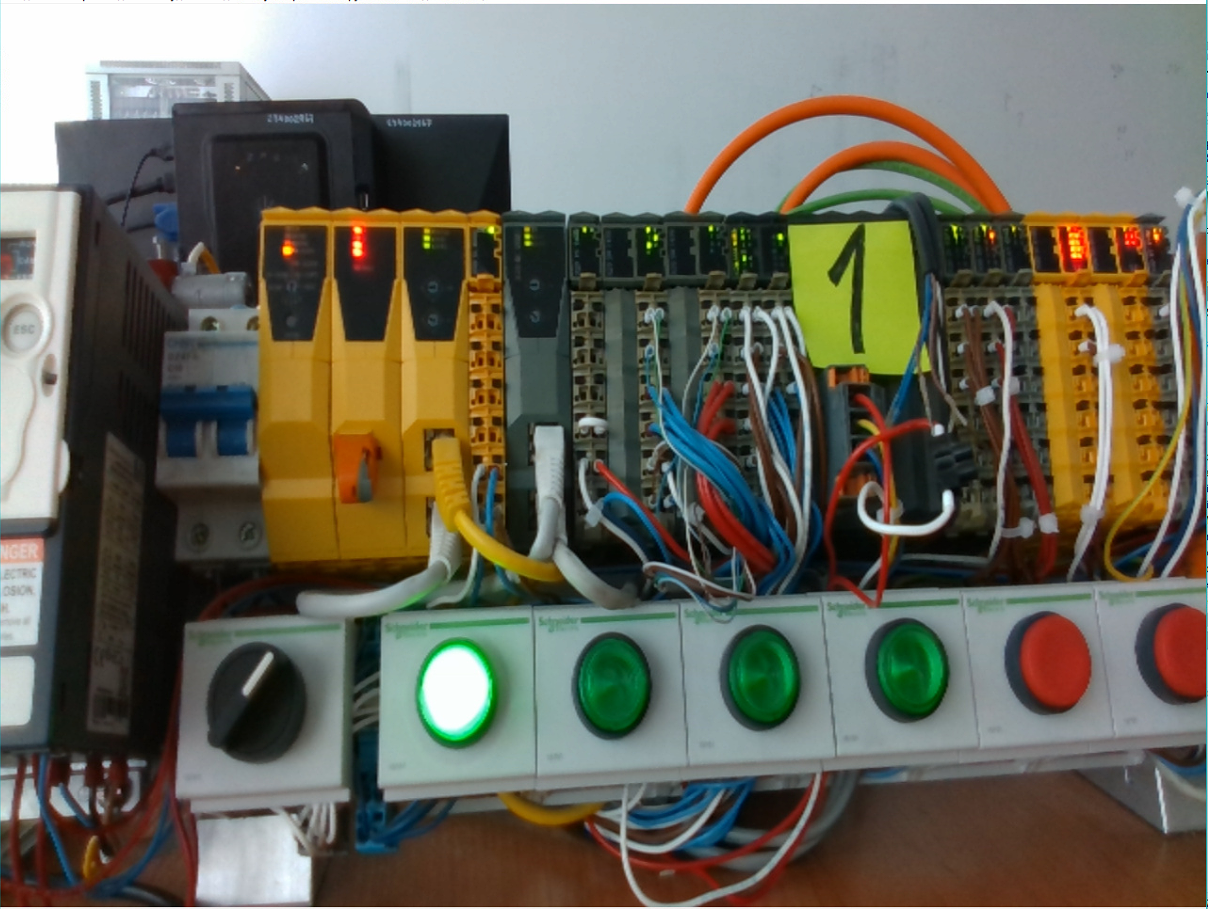


Рисунок 3. Достижение первого датчика.

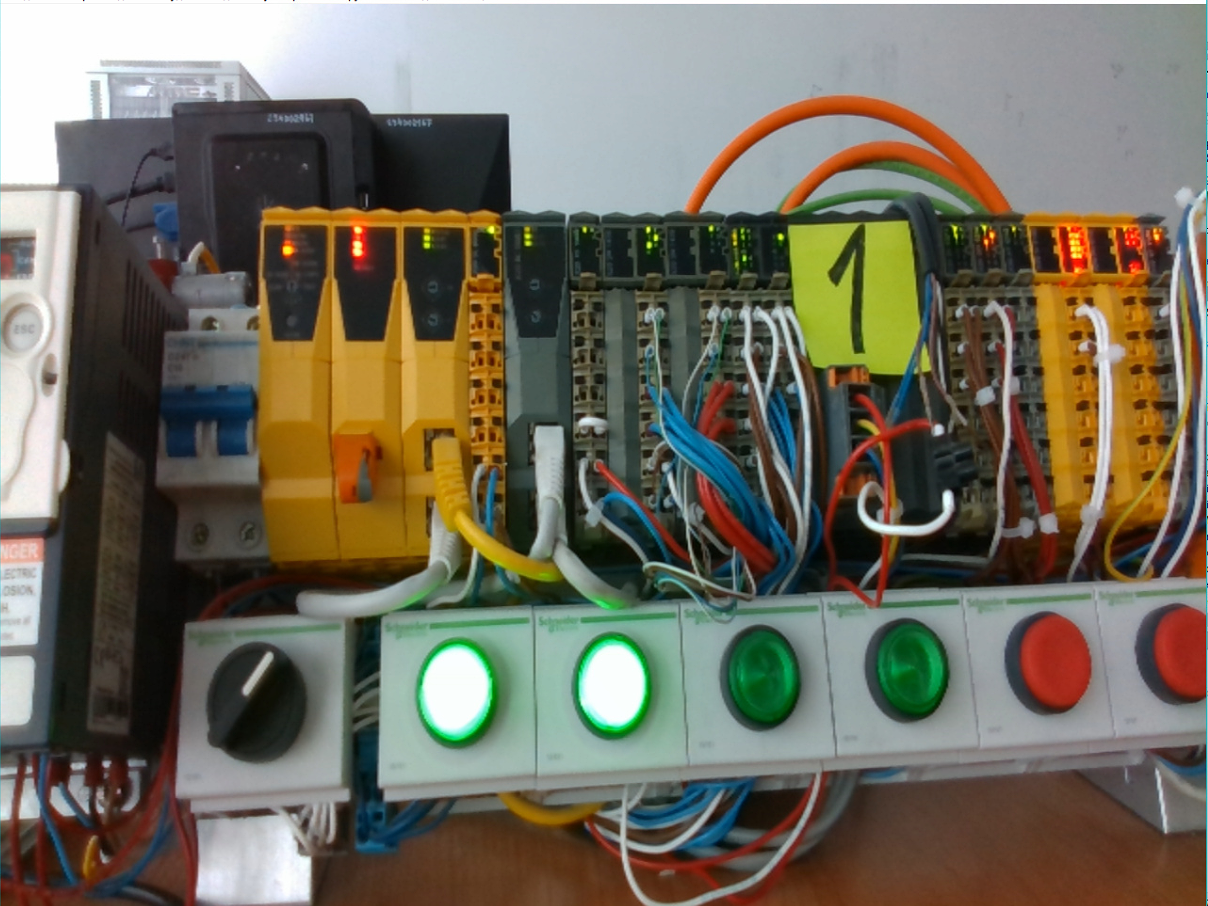


Рисунок 4. Достижение второго датчика.

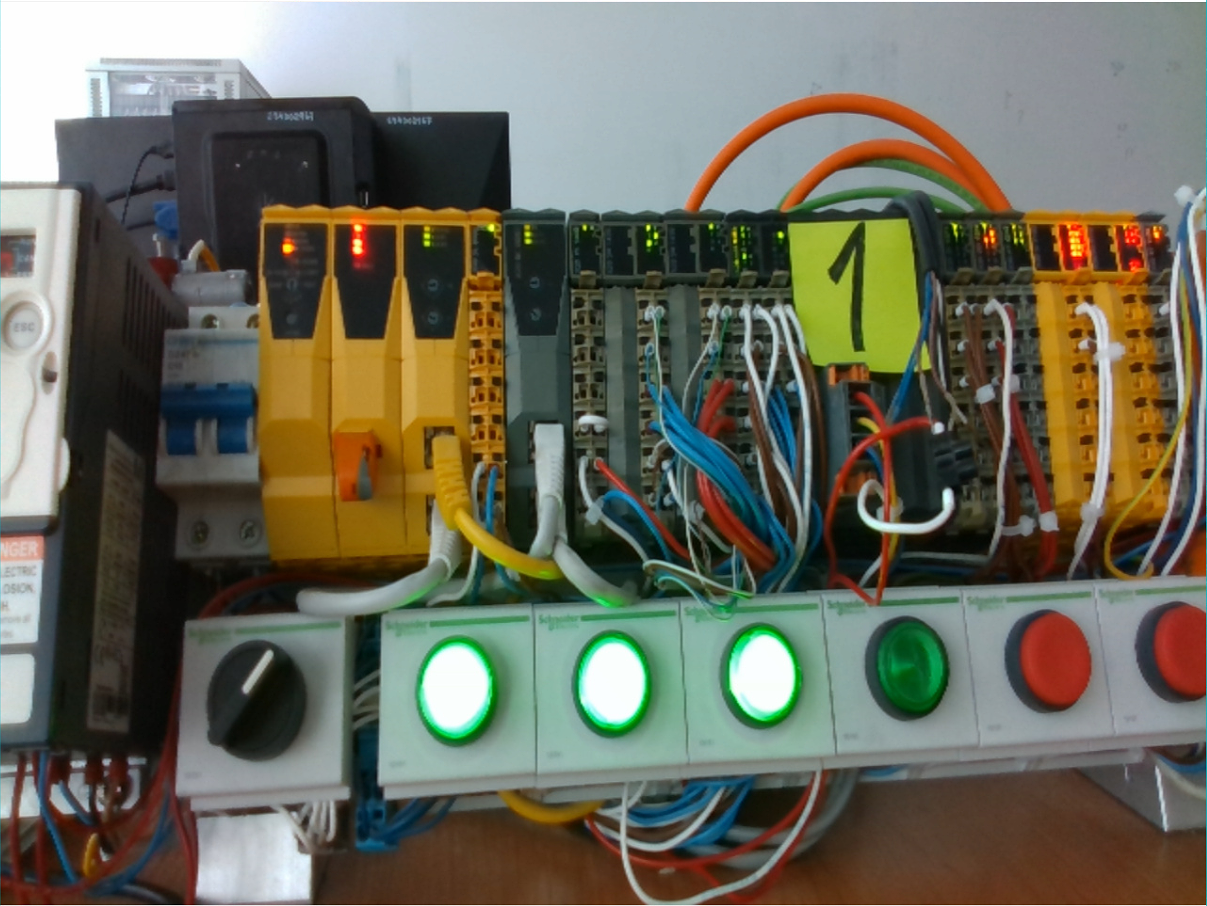


Рисунок 6. Достижение третьего датчика.

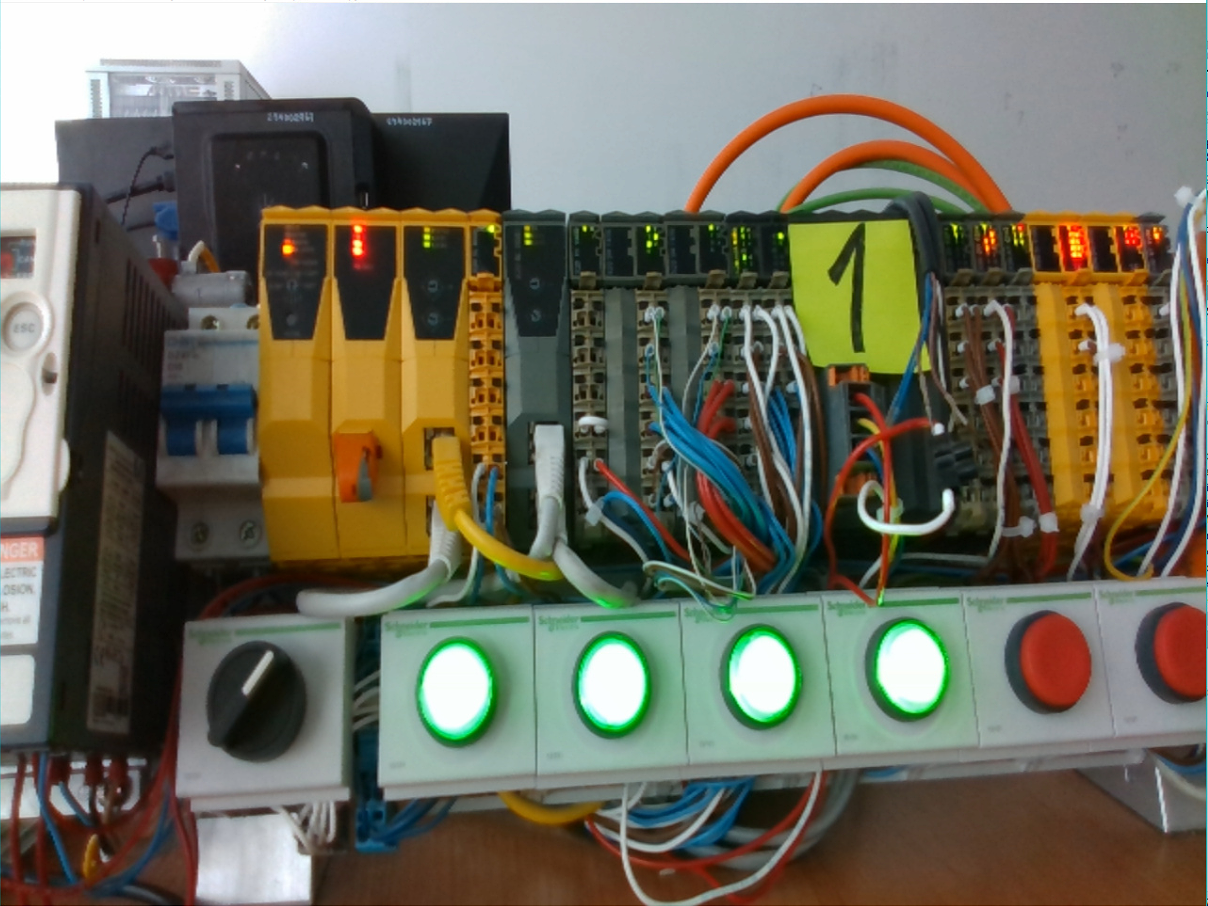


Рисунок 7. Достижение четвёртого датчика.



Рисунок 8. Сигналы датчиков и скорость двигателя



Рисунок 9. Состояние светодиодов.

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

**Приложение А.**

**Листинг кода программы.**

**Main.c**

#include <bur/plctypes.h>  
  
#ifdef \_DEFAULT\_INCLUDES  
 #include <AsDefault.h>  
#endif  
  
void \_INIT ProgramInit(void)  
{  
 driveSM.enable=1;  
}  
void \_CYCLIC ProgramCyclic(void)  
{  
 DoorStateMachine(&doorSM);  
 driveSM.speed = doorSM.speed;  
 DriveStateMachine(&driveSM);  
 ledSM.state = doorSM.state;  
 LedStateMachine(&ledSM);  
}

**Приложение Б.**

**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)  
**{**  
 UINT mask = inst->state & 0x6f;  
 if(!inst->enable)  
 {  
 inst->command = CMD\_SHUTDOWN;  
 }  
 else  
 {  
 switch (mask)  
 {  
 case STATE\_DISABLED:   
 inst->command = CMD\_SHUTDOWN;  
 break;  
 case STATE\_READY:  
 inst->command = CMD\_ENABLED;  
 break;  
 case STATE\_SWITCHED\_ON:  
 inst->command = CMD\_SWITCHED\_ON;  
 break;  
 }  
 }   
  
**}**

**Приложение В.**

**Листинг программного кода функционального блока DoorStateMachine**

#include <bur/plctypes.h>  
#ifdef \_\_cplusplus  
 extern "C"  
 {  
#endif  
 #include "DriveLib.h"  
#ifdef \_\_cplusplus  
 };  
#endif  
/\* TODO: Add your comment here \*/  
void DoorStateMachine(struct DoorStateMachine\* inst)  
{  
 /\*TODO: Add your code here\*/  
 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!=inst->swo1)  
 inst->state=ST\_CLOSE;  
 if(inst->direction==1 && inst->sw1!=inst->swo1)  
 inst->state=ST\_NEG;  
 if(inst->direction==1 && inst->sw2!=inst->swo2)  
 inst->state=ST\_ACC\_NEG;  
 if(inst->direction==1 && inst->sw3!=inst->swo3)  
 inst->state=ST\_DEC\_NEG;  
   
 if(inst->direction==1 && inst->sw4!=inst->swo4)  
 inst->state=ST\_OPEN;  
 if(inst->direction==0 && inst->sw4!=inst->swo4)  
 inst->state=ST\_POS;  
 if(inst->direction==0 && inst->sw3!=inst->swo3)  
 inst->state=ST\_ACC\_POS;  
 if(inst->direction==0 && inst->sw2!=inst->swo2)  
 inst->state=ST\_DEC\_POS;  
 break;  
 }  
   
   
   
 case ST\_CLOSE:  
 {  
 inst->speed = 0;  
 if(inst->direction==1)  
 inst->state=ST\_NEG;  
 break;  
 }  
 case ST\_NEG:  
 {  
 inst->speed = -200;  
 if(inst->direction==1 && inst->sw2!=inst->swo2)  
 inst->state=ST\_ACC\_NEG;  
 if(inst->direction==0)  
 inst->state=ST\_DEC\_POS;  
 break;  
   
 }  
 case ST\_ACC\_NEG:  
 {  
 inst->speed = -600;  
 if(inst->direction==1 && inst->sw3!=inst->swo3)  
 inst->state=ST\_DEC\_NEG;  
 if(inst->direction==0)  
 inst->state=ST\_ACC\_POS;  
 break;  
 }  
 case ST\_DEC\_NEG:  
 {  
 inst->speed = -200;  
 if(inst->direction==1 && inst->sw4!=inst->swo4)  
 inst->state=ST\_OPEN;  
 if(inst->direction==0)  
 inst->state=ST\_POS;  
 break;  
 }  
   
   
   
 case ST\_OPEN:  
 {  
 inst->speed = 0;  
 if(inst->direction==0)  
 inst->state=ST\_POS;  
 break;  
 }  
 case ST\_POS:  
 {  
 inst->speed = 200;  
 if(inst->direction==0 && inst->sw3!=inst->swo3)  
 inst->state=ST\_ACC\_POS;  
 if(inst->direction==1)  
 inst->state=ST\_DEC\_NEG;  
 break;  
 }  
 case ST\_ACC\_POS:  
 {  
 inst->speed = 600;  
 if(inst->direction==0 && inst->sw2!=inst->swo2)  
 inst->state=ST\_DEC\_POS;  
 if(inst->direction==1)  
 inst->state=ST\_ACC\_NEG;  
 break;  
 }  
 case ST\_DEC\_POS:  
 {  
 inst->speed = 200;  
 if(inst->direction==0 && inst->sw1!=inst->swo1)  
 inst->state=ST\_CLOSE;  
 if(inst->direction==1)  
 inst->state=ST\_NEG;  
  
 break;  
 }  
 }  
   
 inst->swo1=inst->sw1;  
 inst->swo2=inst->sw2;  
 inst->swo3=inst->sw3;  
 inst->swo4=inst->sw4;  
}

**Приложение Г.**

**Листинг программного кода функционального блока LedStateMachine**

#include <bur/plctypes.h>  
#ifdef \_\_cplusplus  
 extern "C"  
 {  
#endif  
 #include "DriveLib.h"  
#ifdef \_\_cplusplus  
 };  
#endif  
/\* TODO: Add your comment here \*/  
void LedStateMachine(struct LedStateMachine\* inst)  
{  
 switch(inst->state)  
 {  
 case ST\_INIT:  
 {  
   
 break;  
 }  
   
 case ST\_UNKNOWN:  
 {  
 if(inst->timer%5==4)  
 {  
 inst->led1=!inst->led1;  
 inst->led2=!inst->led2;  
 inst->led3=!inst->led3;  
 inst->led4=!inst->led4;  
 }  
 break;  
 }  
   
   
 case ST\_CLOSE:  
 {  
 inst->led1=1;  
 inst->led2=1;  
 inst->led3=1;  
 inst->led4=1;  
 break;  
 }  
 case ST\_NEG:  
 {  
 inst->led1=0;  
 if(inst->timer%10==9)  
 inst->led2=!inst->led2;  
 inst->led3=1;  
 inst->led4=1;  
 break;  
 }  
 case ST\_ACC\_NEG:  
 {  
 inst->led1=0;  
 inst->led2=0;  
 if(inst->timer%5==4)  
 inst->led3=!inst->led3;  
 inst->led4=1;  
 break;  
 }  
 case ST\_DEC\_NEG:  
 {  
 inst->led1=0;  
 inst->led2=0;  
 inst->led3=0;  
 if(inst->timer%10==9)  
 inst->led4=!inst->led4;  
 break;  
 }  
   
   
 case ST\_OPEN:  
 {  
 inst->led1=0;  
 inst->led2=0;  
 inst->led3=0;  
 inst->led4=0;  
 break;  
 }  
 case ST\_POS:  
 {  
 inst->led1=0;  
 inst->led2=0;  
 if(inst->timer%10==9)  
 inst->led3=!inst->led3;  
 inst->led4=1;  
 break;  
 }  
 case ST\_ACC\_POS:  
 {  
 inst->led1=0;  
 if(inst->timer%5==4)  
 inst->led2=!inst->led2;  
 inst->led3=1;  
 inst->led4=1;  
 break;  
 }  
 case ST\_DEC\_POS:  
 {  
 if(inst->timer%10==9)  
 inst->led1=!inst->led1;  
 inst->led2=1;  
 inst->led3=1;  
 inst->led4=1;  
 break;  
 }  
 }  
 inst->timer++;  
}

https://github.com/NarcoParco/Lab2\_B-R\_PO-MIRTS