

《C语言程序设计》

课程设计报告

姓 名： 刘佳欣

学 号： 16040310049

院 系： 机电工程学院

专 业： 自动化

班 级： 1604031

指导教师： 段江涛

机电工程学院

2017年5月

**三层电梯状态机课程设计报告**

姓名：刘佳欣

学号：16040310049

日期：2017.5.18

1. **状态机图解及其分解描述**

DoorOpen

MovingDown

MovingUp

Idle

DoorClosing

S5 S6

S8 S7

D

D

S3

S9

S1 S2

S4

Idle状态：

电梯停止在某楼层，门是关闭的，处于静止状态。

S1：由静止转为向上升——门外calllight被呼叫，或者门内向上的楼层数被点亮。

S2：由静止转为向下降——门外calllight被呼叫，或者门内向下的楼层数被点亮。

S3：由静止转为开门——当前楼层门外呼叫或者门内开门按钮点亮。

S4：静止状态下关门，无动作，消费门内该按键。

MovingUp/Down状态：

S5/S6：由向上升/向下降转为开门的过程——1.目的地到了开门；2.若上升/下降的过程中楼层同方向被呼叫，则在中途楼层停车，开门。

D：在运动过程中不允许开关门，若有人按了开关门按键，无动作，并且消费按键。

DoorOpen状态：

S7：由开门到关门的过程——1.门内乘客手动关门；2.开门动作结束后，门自动关闭；3.在开门状态下按门内开门按钮电梯无反应，只是消费开门按钮的灯。

DoorClosing状态：

S8：由关门到开门的过程——门内开门按键被点亮，电梯开门。

S9：由关门转向闲置状态的过程——关门后，电梯无动作，处于静止状态。

1. **各状态函数**

Events：

E1: 门内开门按钮(OpenDoorLight)

E2: 门内关门按钮(CloseDoorLight)

E3: 门内楼层按钮(PanelFloorLight)

E4: 门外up呼叫按钮(Call Light)

E5: 门外down呼叫按钮(Call Light)

Idle状态：

1. 检查E3、E4、E5事件：

目标楼层floor=IdleWhatFloorToGoTo(&up) 判断

SetMotorPower(1) or SetMotorPower(-1) MovingUp or MovingDown

1. 检查E1事件：

GetOpenDoorLight() 开门SetDoor() 消费开门按钮

SetOpenDoorLight() DoorOpen

1. 检查E2事件：

GetCloseDoorLight() 消费关门灯

MovingUp状态：

1. 检查E3、E4、E5事件：

目标楼层floor=GoingUpToFloor();if(fabs(GetFloor() - floor) < Lib\_FloorTolerance)

SetMotorPower(0) SetDoor() DoorOpen

1. 检查E1、E2事件：

GetOpenDoorLight()/GetCloseDoorLight() 消费开/关门按钮SetOpenDoorLight()/SetCloseDoorLight()

MovingDown状态：同MovingUp状态

DoorOpen状态：

1. 检查E2事件：

GetCloseDoorLight () 按了关门灯，转而关门SetDoor()

消费关门按钮SetCloseDoorLight () DoorClosing

开门结束后IsDoorOpen() SetDoor() DoorClosing

1. 检查E1事件：

GetOpenDoorLight() 正在开门，无动作 消费开门按钮SetOpenDoorLight()

DoorClosing状态：

1. 检查E1事件：

GetOpenDoorLight() 关门时按了开门灯 SetDoor()

消费开门按钮SetOpenDoorLight() DoorOpen

1. 检查E2事件：

GetCloseDoorLight() 关门，按了关门灯，无动作

消费关门按钮SetCloseDoorLight()

IsBeamBroken() SetDoor() DoorOpen

关门结束后IsDoorClosed(); 进入Idle状态

**三、状态机代码**

#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "ElevatorLib.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Idle状态，电梯停止在某楼层，门是关闭的，处于静止状态，等待相关事件的发生，从而转换到下一个状态。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void StateIdle(int \*state)

{

int floor; bool up;

floor = IdleWhatFloorToGoTo(&up);

if (floor > 0 && up)

{

SetMotorPower(1);

\*state = MovingUp;

printf("Transition: from Idle to MovingUp.\n");

}

if (floor > 0 && !up)

{

SetMotorPower(-1);

\*state = MovingDown;

printf("Transition: from Idle to MovingDown.\n");

}

if (GetOpenDoorLight())

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

if (GetCallLight(GetNearestFloor(), true))

{

SetDoor(GetNearestFloor(), true);

SetCallLight(GetNearestFloor(), true, false);

SetCallLight(GetNearestFloor(), false, false);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

if (GetOpenDoorLight() || GetCallLight(GetNearestFloor(), true) || GetCallLight(GetNearestFloor(), false))

{

SetOpenDoorLight(false);

SetCallLight(GetNearestFloor(), true, false);

SetCallLight(GetNearestFloor(), false, false);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

}

void StateMovingUp(int \*state)

{

int floor;

floor = GoingUpToFloor();

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

printf("Transition: from MovingUp to DoorOpen.\n");

SetCallLight(GetNearestFloor(), true, false);

SetCallLight(GetNearestFloor(), false, false);

SetPanelFloorLight(GetNearestFloor(), false);

}

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

void StateMovingDown(int \*state)

{

int floor;

floor = GoingDownToFloor();

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

printf("Transition: from MovingDown to DoorOpen.\n");

SetCallLight(GetNearestFloor(), true, false);

SetCallLight(GetNearestFloor(), false, false);

SetPanelFloorLight(GetNearestFloor(), false);

}

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

void StateDoorOpen(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetCloseDoorLight())

{

SetDoor(GetNearestFloor(), false);

SetCloseDoorLight(false);

\*state = DoorClosing;

printf("Transition: from DoorOpen to DoorClosing.\n");

}

if (IsDoorOpen(GetNearestFloor()))

{

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

printf("Transition: from DoorOpen to DoorClosing.\n");

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

}

}

void StateDoorClosing(int \*state)

{

int floor; bool open;

floor = GetNearestFloor();

if (GetOpenDoorLight())

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

printf("Transition: from DoorClosing to DoorOpen.\n");

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

if (IsBeamBroken())

{

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

printf("Transition: from DoorClosing to DoorOpen.\n");

}

if (IsDoorClosed(GetNearestFloor()))

{

\*state = Idle;

printf("Transition: from DoorClosing to Idle.\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 状态机，每隔一定时间(如，100ms)被调用一次，采集系统的运行状态

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main\_control(int \*state)

{

if (IsElevatorRunning()) // 仿真正在运行

{

switch (\*state)

{

case Idle:

// Idle状态，一定时间无动作，自动到一楼

if (GetNearestFloor() != 1) {

AutoTo1Floor();

}

StateIdle(state);

break;

case MovingUp:

CancelTo1Floor(); // 其它状态，取消自动到一楼

StateMovingUp(state);

break;

case MovingDown:

CancelTo1Floor();

StateMovingDown(state);

break;

case DoorOpen:

CancelTo1Floor();

StateDoorOpen(state);

break;

case DoorClosing:

CancelTo1Floor();

StateDoorClosing(state);

break;

default:

printf("没有这种状态!!!\n");

}

}

}

**四、电梯功能测试**

1.电电梯停于1F或2F时，按3F向下呼叫按钮；电梯上升到3F停止，开门/关门。

2.梯停于2F或3F时，按1F向上呼叫按钮；电梯下降到1F停止，开门/关门。

3.电梯停于1F时，按2F向上呼叫按钮；电梯上升到2F停止，开门/关门。

4.电梯停于3F时，按2F向下呼叫按钮；电梯下降到2F停止，开门/关门。

5.电梯停于1F，2F和3F均有按钮呼叫；电梯先上升到2F，开门/关门，然后上升到3F停止，开门/关门。

6.电梯停于3F，2F和1F均有按钮呼叫；电梯先下降到2F，开门/关门，然后下降到1F停止，开门/关门。

7.电梯上升途中或下降途中，任何反方向按钮呼叫均无效。

8.电梯停于1F，按3F向下呼叫按钮，然后立即按2F向下呼叫按钮；电梯上升到3F停止，开门/关门，然后下降到2F停止，开门/关门。

9.电梯停于2F，按门内楼层按钮3，然后门内楼层按钮1；电梯上手到3F停止，开门/关门，然后下降到1F，开门/关门。

10.电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以下，按2F向上呼叫按钮；电梯先上升到2F，开门/关门；然后再到3F，开门/关门。

11.电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以上，按2F向上呼叫按钮；电梯先上升到3F，开门/关门；然后再到2F，开门/关门。

12.电梯停于1F，按2F向下呼叫按钮和向上呼叫按钮以及3F的向下呼叫按钮；电梯上升到2F停止，开门/关门，2F的向上呼叫按钮灯关闭，开门/关门， 2F的向下呼叫按钮灯关闭，然后上升到3F停止，开门/关门，3F的向下呼叫按钮关闭。

13.电梯在2F以上，10s无动作，自动降到1楼。

14.所有停止，开门/关门后，对应楼层的同方向门外呼叫按钮灯（最高楼向下呼叫按钮，最底层向上呼叫按钮）和门内楼层按钮灯关闭。

15.运动状态，开关门按钮失效。正在开门，开门按钮失效；正在关门，关门按钮失效。

16.空闲状态，门是关闭的，因此按关门按钮失效。

17.开关门结束，延时2秒用于乘客上下电梯【延时功能在库函数中实现，不用在状态函数中实现】，然后进入关门状态。

18.开门结束前，按关门按钮，转而进入关门状态；关门结束前，按开门按钮，转而进入开门状态。













