C语言程序设计课程报告

**班级：1604031**

**学号：16040310077**

**姓名：周磊磊**

**时间：5月20日**

## 三层电梯状态机课程设计报告

**一．状态机图总体概括及各状态描述**：

将三层升降电梯的运行过程分为Idle，MovingUp/MovingDown，DoorOpen，DoorClosing五个状态（其中MovingUp/MovingDown可合并为一个状态）。 如下图所示：

**（S4）**

（S9）

（S1/S2） （S3） （S8）（S7）

（S5/S6）

Idle

DoorClosing

MovingUp/MovingDown

DoorOpen

**MovingUp/MovingDow**n

**（D）**

Idle状态事件分析：

S1/S2：由闲置转向向上升/下降的过程，对应的事件是非当前楼层门外CallLight被呼叫，或者门内向上/下的楼层数被点亮。

S3：由闲置转向开门，对应的事件是当前楼层门外呼叫或者门内开门按钮点亮。

S4：闲置状态下关门，无动作，消费门内该按键。

MovingUp/MovingDown状态及其事件：

S5/S6：由向上升/向下降转向开门的过程，对应的事件有两个：其一是目的地到了，自然开门；其二是若上升/下降的过程中途（在来得及刹车的情况下）楼层同方向被呼叫，则在中途楼层停车，开门上客。

D：在运动过程中出于安全考虑不允许开关门，若有人按了开关门按键，无动作，并且及时灭掉按键。

DoorOpen状态及其事件

S7：由开门到关门的过程。对应三个事件，其一是门内乘客手动关门；其二是开门动作结束后，门自动关闭；其三是在开门状态下按门内开门按钮电梯无反应，只是灭掉开门按钮的灯。

DoorClosing状态及其事件

S8：由关门到开门的过程，对应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。对应关门结束后，电梯没有下一步动作，处于空闲的状态。

**二.各状态函数详解**

把主要事件记为E1~E5。具体规定如下：

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：**

(S1) 检查E3、E4、E5事件。（根据布尔变量up的值，以及 IdleWhatFloorToGoTo（）返回值来确定电梯下一步应该是怎样的状态。）

属于静态检测。bool up; floor=IdleWhatFloorToGoTo(&up);

关闭本层门外up按钮，SetCallLight()—— 即消费门外up按钮，防止下一周期重复处理此按钮行为。

(S2) 同(S1)，消费门外down按钮。

1层以上，一定时间无动作，自动下降到1楼。AutoTo1Floor();[其它状态，取消此功能，CancelTo1Floor()]

(S3) 检查E1事件, 开门, 消费开门按钮;

上升 (up && E4事件), 开门，消费门外up按钮

下降 (!up && E5事件), 开门，消费门外down按钮

(S4) 检查E2事件,此时门应该是关闭的,因此仅读取关门灯，并关闭关门灯，即消费按键行为，防止下一周期重复处理该按钮的行为。

if(GetCloseDoorLight()) { SetCloseDoorLight(false); return; }

**运动中状态：**MovingUp/MovingDown 🡪 DoorOpen

(S5) 检查E3、E4、E5事件。

动态检测，目标楼层floor=GoingUpToFloor();

if(fabs(GetFloor() - floor) < Lib\_FloorTolerance) 到达目标楼层，停止，开门，消费门外按钮。

(D) 检查E1、E2事件，无动作，消费开/关门按钮。

(S6) 检查E3、E4、E5事件。动态检测，目标楼层=GoingDownToFloor();其它与(S5)类似。

**开门状态:** 电梯门打开 🡪 DoorClosing

(S7) 检查E2事件，转而关门，GetCloseDoorLight(),SetDoor(); 消费关门按钮。

开门结束后，自动进入关门状态。IsDoorOpen();SetDoor();

检查E1事件, 无动作，消费开门按钮。

**关门状态:**

(S8) 检查E1事件，转而开门。GetOpenDoorLight();SetDoor(); 消费关门按钮。进入DoorOpen状态

检查E2事件，无动作，消费关门按钮。

(S9) 关门结束后，进入Idle状态。IsDoorClosed();

**三．程序的主要内容：**

#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);//静态检测

if (floor > 0 && up)//s1

{

SetMotorPower(1);

\*state = MovingUp;

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

}

if (floor > 0 && !up)//事件s2

{

SetMotorPower(-1);

\*state = MovingDown;

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

}

if (GetOpenDoorLight())//事件s3中的E1

{

SetDoor(GetNearestFloor(), 1);

SetOpenDoorLight(0);

\*state = DoorOpen;

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

}

if (up&&GetCallLight(GetNearestFloor(), 1))//事件s3中的E4

{

SetDoor(GetNearestFloor(), 1);

SetCallLight(GetNearestFloor(), 1, 0);

\*state = DoorOpen;

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

}

if (!up&&GetCallLight(GetNearestFloor(), 1))//事件s3中的E5

{

SetDoor(GetNearestFloor(), 1);

\*state = DoorOpen;

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

}

if (GetCloseDoorLight())//事件s4

{

SetCloseDoorLight(0);

return;

}

}

void StateMovingUp(int \*state)

{

int floor;

bool up;

floor = GoingUpToFloor();//动态检测

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//s5

{

SetMotorPower(0);//

SetDoor(floor, 1);

SetPanelFloorLight(GetNearestFloor(), 0);

SetCallLight(floor, 1, 0);// 消费

SetCallLight(floor, 0, 0);

\*state = DoorOpen;

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

}

if (GetNearestFloor() == 3)

SetCallLight(floor, 1, 0);

if (GetOpenDoorLight()) //事件D 1

SetOpenDoorLight(0);

if (GetCloseDoorLight())//事件D 2

SetOpenDoorLight(0);

}

void StateMovingDown(int \*state)

{

int floor;

bool up;

floor = GoingDownToFloor();//动态检测

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//s6

{

SetMotorPower(0);

SetDoor(floor, 1);

SetPanelFloorLight(GetNearestFloor(), 0);//消费门内楼层按钮

SetCallLight(GetNearestFloor(),1|| 0, 0);//消费门外up/down按钮

\*state = DoorOpen;

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

}

if (GetNearestFloor() == 1)

SetCallLight(floor, 0, 0);

if (GetOpenDoorLight())//事件D

SetCloseDoorLight(0);

if (GetCloseDoorLight())//事件D

SetOpenDoorLight(0);

}

void StateDoorOpen(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetCloseDoorLight())//事件s7 1

{

SetDoor(GetNearestFloor(), 0);

SetCloseDoorLight(0);//消费

\*state = DoorClosing;

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

}

if (IsDoorOpen(floor))//事件s7 2

{

SetDoor(floor, 0);

\*state = DoorClosing;

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

}

if (GetOpenDoorLight())//事件s7 3

{

SetOpenDoorLight(0);//消费

}

}

void StateDoorClosing(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetOpenDoorLight())//事件s8 1

{

SetDoor(floor, 1);

SetOpenDoorLight(0);

\*state = DoorOpen;

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

}

if (GetCloseDoorLight())//事件s8 2

{

SetCloseDoorLight(0);

return;

}

if (IsBeamBroken())//事件s8 3

{

SetDoor(floor, 1);

\*state = DoorOpen;

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

}

if (IsDoorClosed(floor))

{

\*state = Idle;

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

}

}

}

**四．三层电梯运行结果描述：**

1.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.电梯停于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。

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。

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。

16040310077 周磊磊

2017年5月20日