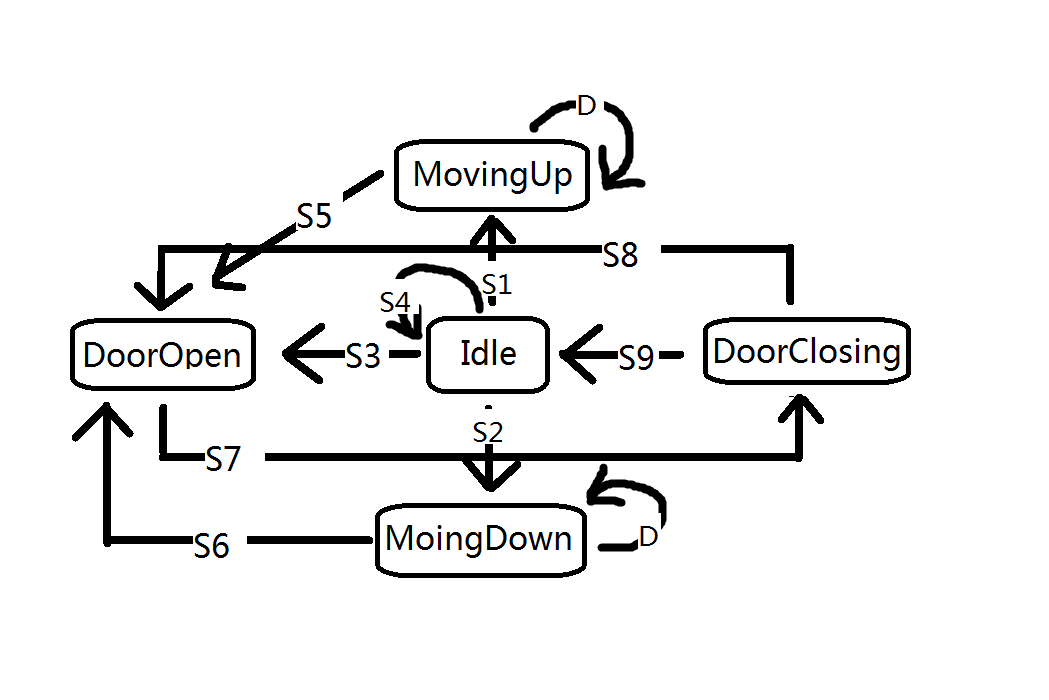
**三层电梯课程设计报告**

16040310076 陈思为

**一.状态机图及其分解描述**：



Idle状态及其事件

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

S2：由闲置转向向下降的过程，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，灭掉门内该按键。

MovingUp/Down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：其一是到了目的楼层，自然开门；其二是在上升/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，若有人按了开门或关门按键，无动作，并且及时消费掉按键。

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.各状态函数设计过程**

把主要事件记为E1~E5。

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)

**1.Idle状态（闲置状态）：**

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。

（1）Idle 🡪 MovingUp

(S1) 检查E3、E4、E5事件。

静态检测(检测将要到的目标楼层),

进入MovingUp状态。

（2）Idle 🡪 MovingDown (S2)检查E3、E4、E5事件

静态检测(检测将要到的目标楼层),

进入MovingDown状态

（3）Idle 🡪 DoorOpen

(S3)检查E1事件

开门

消费开门按钮

进入DoorOpen状态。

检查E4/E5事件

开门

消费门外up/down按钮

进入DoorOpen状态。

（4）Idle 🡪 Idle

(S4) 检查E2事件，此时门应该是关闭的,因此仅读取关门灯

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。

**2.MovingUp状态：**

（1）MovingUp 🡪 DoorOpen

(S5) 检查E3、E4、E5事件。

动态检测，目标楼层

到达目标楼层，停止

开门

进入DoorOpen状态。

消费门外up按钮

如果到了最高层

消费门外down按钮。

消费门内楼层按钮为当前楼层的按钮

（2）MovingUp状态: 安全设置，无动作。

(D) 运动状态，开关门失效，检查E1、E2事件

如果有开门或关门按钮呼叫

无动作

消费开/关门按钮

**3.MovingDown状态：**

（1）MovingDown 🡪 DoorOpen

(S6) 检查E3、E4、E5事件。

动态检测，目标楼层

到达目标楼层，停止

开门

进入DoorOpen状态。

消费门外down按钮

如果到了1层

消费门外up按钮 消费门内楼层按钮为当前楼层的按钮

（2）MovingDown状态的安全设置，无动作。

(D) 运动状态，开关门失效，检查E1、E2事件

如果有开门或关门按钮呼叫

无动作

消费开/关门按钮

**4.DoorOpen状态:**

DoorOpen🡪 DoorClosing

(S7) GetNearestFloor()获取当前楼层

检查E2事件

正在开门，按了关门灯，转而关门

消费关门按钮

进入DoorClosing状态。

开门结束后

自动进行关门

进入DoorClosing状态。

检查E1事件

正在开门，按了开门灯，无动作

消费开门按钮

**5.DoorClosing状态:**

（1）DoorClosing 🡪 DoorOpen

(S8) GetNearestFloor()获取当前楼层

检查E1事件

正在关门，按了开门灯，转而开门

消费开门按钮

进入DoorOpen状态。

检查E2事件

正在关门，按了关门灯，无动作，消费关门按钮

如果红外探测到遮挡

转而开门

进入DoorOpen状态。

（2）DoorClosing状态: DoorClosing 🡪 Idle

(S9) 关门结束后，进入Idle状态。

**三.电梯功能测试参考**

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。

14.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。

**四．程序主要内容**

#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);// 静态监测,电梯处于空闲状态, 确定下一步的运动方向和所到楼层(目标楼层)

if (floor > 0 && up)// 门外up呼叫按钮

{

SetMotorPower(1); // 电机运转，上升

\*state = MovingUp;//状态变为上升

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

}

if (floor > 0 && !up) // 门外down呼叫按钮

{

SetMotorPower(-1);// 电机运转，下降

\*state = MovingDown;//状态变为下降

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

}

if (GetOpenDoorLight())//门内开门按钮呼叫

{

SetDoor(GetNearestFloor(), true); // 获取最近楼层或所在楼层，开门

SetOpenDoorLight(false);//消费开门按钮灯

\*state = DoorOpen;//状态变为开门

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

}

if (GetCallLight(GetNearestFloor(), true))//门外上升按钮呼叫

{

SetCallLight(GetNearestFloor(), true, false);//消费所在楼层上升按钮灯

SetDoor(GetNearestFloor(), true); // 获取最近楼层或所在楼层，开门

\*state = DoorOpen;//状态转变为，开门

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

}

if (GetCallLight(GetNearestFloor(), false))//门外下降按钮呼叫

{

SetCallLight(GetNearestFloor(), false, false); //消费所在楼层上升按钮灯

SetDoor(GetNearestFloor(), true); //获取最近楼层或所在楼层，开门

\*state = DoorOpen;//状态转变为，开门

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

}

if (GetCloseDoorLight())//门内关门按钮呼叫

{

SetCloseDoorLight(false);//消费关门按钮灯

return;

}

}

void StateMovingUp(int \*state)

{

int floor;

floor = GoingUpToFloor();//获取上升目的楼层

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)// 到达目标楼层

{

SetMotorPower(0);//电动机停止

SetDoor(GetNearestFloor(), true); // 开门

SetPanelFloorLight(GetNearestFloor(), false);//消费门内楼层为当前楼层的按钮

SetCallLight(GetNearestFloor(), true, false);//消费门外上升按钮

SetCallLight(GetNearestFloor(), false, false);//消费门外下降按钮

\*state = DoorOpen;//状态转变为开门

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

}

if (GetOpenDoorLight() )// 获取门内开门按钮

{

SetOpenDoorLight(false);//消费开门按钮

}

if (GetCloseDoorLight())// 获取门内开门按钮

{

SetCloseDoorLight(false);//消费关门按钮

}

}

void StateMovingDown(int \*state)

{

int floor;

floor = GoingDownToFloor();//获取下降目的楼层

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)// 到达目的楼层

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true); // 开门

SetPanelFloorLight(GetNearestFloor(), false);//消费门内楼层为当前楼层的按钮

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), true, false);//消费门外上升按钮

SetCallLight(GetNearestFloor(), false, false);//消费门外下降按钮

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

}

if (GetOpenDoorLight())// 门内开门按钮呼叫

{

SetOpenDoorLight(false);//消费开门按钮

}

if (GetCloseDoorLight())// 门内关门按钮呼叫

{

SetCloseDoorLight(false);//消费门内关门按钮

}

}

void StateDoorOpen(int \*state)

{

if (GetCloseDoorLight())// 门内关门按钮呼叫

{

SetDoor(GetNearestFloor(), false); // 关门

SetCloseDoorLight(false);//消费关门按钮

\*state = DoorClosing;//状态变为正在门关闭

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

}

if (IsDoorOpen(GetNearestFloor()))// 开门结束后

{

SetDoor(GetNearestFloor(), false); // 关门

\*state = DoorClosing;//状态变为正在关门

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

}

if (GetOpenDoorLight())// 门内开门按钮呼叫

SetOpenDoorLight(false);//消费开门按钮

}

void StateDoorClosing(int \*state)

{

if (GetOpenDoorLight())//门内开门按钮呼叫

{

SetDoor(GetNearestFloor(), true); // 开门

\*state = DoorOpen;//状态变为开门

printf("Transition: from DoorClose toDoorOpen.\n");

}

if (GetCloseDoorLight())// 门内关门按钮呼叫

{

SetCloseDoorLight(false);//消费关门按钮

}

if (IsBeamBroken())// 红外感觉到门中有遮挡物

{

SetDoor(GetNearestFloor(), true); // 开门

\*state = DoorOpen;//状态变为开门

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

}

if (IsDoorClosed(GetNearestFloor()))// 关门结束

\*state = Idle;//状态变为闲置状态

}

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

\* 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");

}

}

}