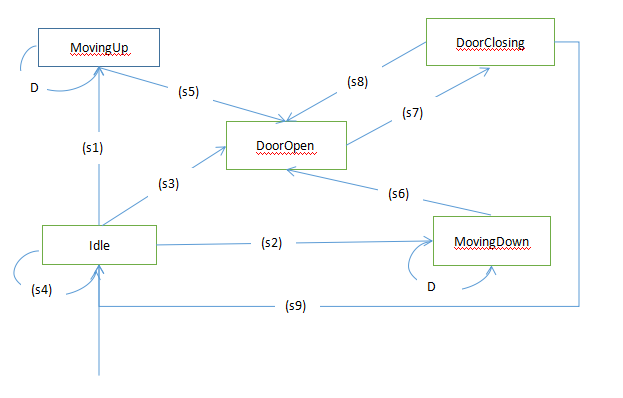
**三层电梯状态机课程设计**

**姓名：王兴德 学号：15069130003**

**一、状态机图及其分解描述**



分为五个状态：静置状态（Idle），上升状态(MovingUp)，下降状态(MovingDown)，关门状态(DoorClosing)，开门状态(DoorOpen)。

包含事件（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)

（1）静置状态（Idle）: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。

(S1) 检查E3、E4、E5事件。

静态检测，bool up; 目标楼层=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; }

（2）上升/下降状态（MovingUp/ MovingDown）：

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up按钮; 到了最高层Lib\_FloorNum, 消费门外down按钮。消费门内楼层按钮。

(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。

（3）开门状态（DoorOpen）:

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。

（4）关门状态(DoorClosing):

正在关门 🡪（ DoorOpen）：

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

检查E2事件，无动作，消费关门按钮。

DoorClosing状态: 正在关门 🡪 Idle

(S9) 关门结束后，进入Idle状态。IsDoorClosed();

**三、实现的功能及测试结果**

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。
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。

**四、Elevator.cpp**

#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("上升\n");

}

if (floor > 0 && !up){

SetMotorPower(-1);

\*state = MovingDown;

printf("下降\n");

}

if (floor = -1)

{

if (GetOpenDoorLight())

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCallLight(GetNearestFloor(), up))

{

SetCallLight(GetNearestFloor(),up,false);

\*state = DoorOpen;

}

if (GetCallLight(GetNearestFloor(), !up))

{

SetCallLight(GetNearestFloor(), !up,false);

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

\*state = DoorClosing;

return;

}

}

}

void StateMovingUp(int \*state)

{

int floor = GoingUpToFloor();

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

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

SetPanelFloorLight(GetNearestFloor(), false);

\*state = DoorOpen;

if (GetCallLight(GetNearestFloor(), true))

SetCallLight(GetNearestFloor(), true, false);

}

if (GetCloseDoorLight())

SetCloseDoorLight(false);

if (GetOpenDoorLight())

SetOpenDoorLight(false);

}

void StateMovingDown(int \*state)

{

int floor = GoingDownToFloor();

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

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

SetPanelFloorLight(GetNearestFloor(), false);

\*state = DoorOpen;

if (GetCallLight(GetNearestFloor(), false))

SetCallLight(GetNearestFloor(), false, false);

}

if (GetCloseDoorLight())

SetCloseDoorLight(false);

if (GetOpenDoorLight())

SetOpenDoorLight(false);

}

void StateDoorOpen(int \*state)

{

bool Open;

if (GetCloseDoorLight())

{

SetDoor(GetNearestFloor(), false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

if (IsDoorOpen(GetNearestFloor()))

{

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(true);

\*state = DoorOpen;

}

}

void StateDoorClosing(int \*state)

{

bool Open;

if (GetOpenDoorLight())

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

if (IsBeamBroken())

{

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

if (IsDoorClosed(GetNearestFloor()))

{

\*state = Idle;

}

}

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

\* 主控循环： 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:

if (GetNearestFloor() != 1)

{

AutoTo1Floor();// Idle状态，一定时间无动作，自动到一楼

}

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");

}

}

}