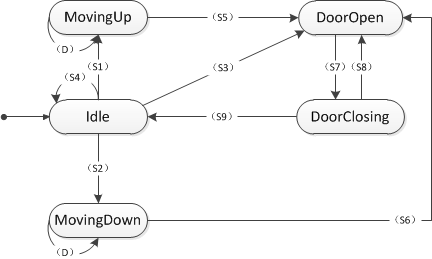
课程设计 16040310059 王子渊

1. 状态介绍
2. Idle状态：电梯停在某楼层，门关闭并处于静止状态，等待相关事件发生，从而转换到下一个状态。Idle转换到MovingUp或MovingDown状态，可能发生了E3,E4,E5三个事件中的一个。通过静态检测检测将要到的目标楼层,如果目标楼层大于0且为up，则电梯全速上升，如果目标楼层大于0且不为up，则电梯全速下降。检查门内开门按钮，从Idle到DoorOpen状态，消费开门按钮。检查门外呼叫事件，到达所要去的楼层后消费up/down按钮。如果检测到门内关门按钮，关闭门内按钮灯，消费按键。
3. MovingUp状态：进行动态检测，如果到达目标楼层，速度为0，即停止。然后开门，进入DoorOpen状态，利用SetCallLight()消费门外up按钮，并消费门内楼层按钮。运动状态时开关门失效，门内无论按开门关门按钮灯都灭。
4. MovingDown状态：同MovingUp状态进行动态检测。进入开门状态后消费门外down 按钮。运动状态时开关门失效。
5. DoorOpen状态：检查门内关门按钮，开门转而关门，消费关门按钮，进入DoorClosing状态。返回门是否是开的，如果返回true,自动进行关门。如果正在开门，这时再按开门灯，不会有动作，消费开门按钮。
6. DoorClosing状态：检查E1事件，关门状态转为开门，消费开门按钮，进入DoorOpen状态。检查E2事件，正在关门，再按关门灯，无动作并消除关门按钮。如果红外探测到遮挡，就转为开门，进入开门状态。返回门是否为关闭，如果返回true,就进入Idle状态。

二、状态图解

 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)

#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("from idle to moving up/n");

}

else if (floor>0&&!up)

{

SetMotorPower(-1);

\*state = MovingDown;

printf("from idle to moving down");

}

else if (GetCallLight(1,1))

{

SetDoor(1, 1);

SetCallLight(1, 1, 0);

\*state = DoorOpen;

}

else if (GetOpenDoorLight())

{

SetOpenDoorLight(0);

SetDoor(GetNearestFloor(), 1);

\*state = DoorOpen;

}

else if (GetCallLight(floor, up))

{

SetCallLight(floor, up, 0);

SetDoor(floor, 1);

\*state = DoorOpen;

printf("from idle to DoorOpen");

}

else if (GetCloseDoorLight())

{

SetCloseDoorLight(0);

printf("from idle to idle");

return;

}

}

void StateMovingUp(int \*state)

{

int floor = GoingUpToFloor();

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

{

SetMotorPower(0);

SetDoor(floor, 1);

SetCallLight(floor, 1&&0, 0);

SetPanelFloorLight(floor, 0);

\*state = DoorOpen;

return;

}

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(0);

SetCloseDoorLight(0);

}

}

void StateMovingDown(int \*state)

{

int floor = GoingDownToFloor();

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

{

SetMotorPower(0);

SetDoor(floor, 1);

SetCallLight(floor, 1 && 0, 0);

SetPanelFloorLight(floor, 0);

\*state = DoorOpen;

return;

}

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(0);

SetCloseDoorLight(0);

}

}

void StateDoorOpen(int \*state)

{

if (GetCloseDoorLight())

{

SetDoor(GetNearestFloor(), 0);

SetCloseDoorLight(0);

\*state = DoorClosing;

}

else if (IsDoorOpen(GetNearestFloor()))

{

SetDoor(GetNearestFloor(),0);

\*state = DoorClosing;

}

else if (GetOpenDoorLight())

SetOpenDoorLight(0);

}

void StateDoorClosing(int \*state)

{

if (GetOpenDoorLight())

{

SetDoor(GetNearestFloor(), 0);

SetOpenDoorLight(0);

\*state = DoorOpen;

}

else if (GetCloseDoorLight())

{

SetCloseDoorLight(0);

}

else if (IsBeamBroken())

{

SetDoor(GetNearestFloor(), 1);

\*state = DoorOpen;

}

else if (IsDoorClosed(GetNearestFloor()))

\*state = Idle;

}

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

\* 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");

}

}

}