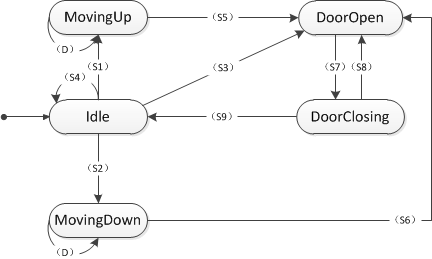
三层电梯状态机仿真程序



|  |  |  |
| --- | --- | --- |
| Idle | **StateIdle** | **S1,S2,S3,S4** |
| MovingUp | StateMovingUp | S5,D |
| MovingDown | StateMovingDown | S6,D |
| DoorOpen | StateDoorOpen | S7 |
| DoorClosing | StateDoorClosing | S8,S9 |

**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。**

**Idle 🡪 MovingUp**

**(S1) 检查E3、E4、E5事件，已经封装在下列函数中。**

**静态检测(检测将要到的目标楼层**

**Idle 🡪 MovingDown**

**(S2)检查E3、E4、E5事件，已经封装在下列函数中。**

**静态检测(检测将要到的目标楼层)**

**Idle 🡪 DoorOpen**

**(S3)**

1. **检查E1事件GetOpenDoorLight(), 开门SetDoor(), 消费开门按钮SetOpenDoorLight(); 进入DoorOpen状态。**
2. **检查E4/E5事件GetCallLight(), 开门SetDoor(),消费门外up/down按钮SetCallLight(); 进入DoorOpen状态。**

**Idle** 🡪 **Idle**

**(S4) 检查E2事件GetCloseDoorLight(),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。**

**MovingUp状态：MovingUp 🡪 DoorOpen**

**(S5) 检查E3、E4、E5事件，已封装在下列函数中。**

**动态检测，目标楼层floor=GoingUpToFloor();**

**if(fabs(GetFloor() - floor) < Lib\_FloorTolerance) 到达目标楼层，停止SetMotorPower(0)，开门SetDoor();进入DoorOpen状态。**

**消费门外up按钮SetCallLight(); 到了最高层Lib\_FloorNum, 消费门外down按钮。**

**消费门内楼层按钮为当前楼层的按钮SetPanelFloorLight()。**

**GetNearestFloor()**

**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GetOpenDoorLight()/GetCloseDoorLight()，无动作，消费开/关门按钮SetOpenDoorLight()/SetCloseDoorLight()。**

**MovingDown状态：MovingDown 🡪 DoorOpen**

**(S6) 检查E3、E4、E5事件，已封装在下列函数中。**

**动态检测，目标楼层floor=GoingDownToFloor();**

**if(fabs(GetFloor() - floor) < Lib\_FloorTolerance) 到达目标楼层，停止SetMotorPower(0)，开门SetDoor();进入DoorOpen状态。**

**消费门外down按钮SetCallLight(); 到了1层, 消费门外up按钮SetUpLight()。**

**消费门内楼层按钮为当前楼层的按钮SetPanelFloorLight()。**

**GetNearestFloor()**

**MovingDown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GetOpenDoorLight()/GetCloseDoorLight()，无动作，消费开/关门按钮SetOpenDoorLight()/SetCloseDoorLight()。**

**DoorClosing状态: DoorClosing 🡪 DoorOpen**

**(S8) GetNearestFloor()获取当前楼层**

1. **检查E1事件GetOpenDoorLight(); 正在关门，按了开门灯，转而开门SetDoor(); 消费开门按钮SetOpenDoorLight()。进入DoorOpen状态。**
2. **检查E2事件GetCloseDoorLight()，正在关门，按了关门灯，无动作，消费关门按钮SetCloseDoorLight()。**
3. **如果红外探测到遮挡 IsBeamBroken()，转而开门SetDoor()，进入DoorOpen状态。**

**DoorClosing状态: DoorClosing** 🡪 **Idle**

**(S9) 关门结束后IsDoorClosed(); 进入Idle状态。**

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); //静态检测

if (floor > 0 && up)

{

SetMotorPower(1); //以最大功率上升

\*state = MovingUp; //进入MovingUp状态

}

floor = IdleWhatFloorToGoTo(&up);

if (floor > 0 && !up)

{

SetMotorPower(-1);

\*state = MovingDown;

}

floor=GetNearestFloor(); //获取最近楼层

if (GetOpenDoorLight())

{

SetDoor(floor, true);

SetOpenDoorLight(false); //消费开门按钮

\*state = DoorOpen;

}

if (GetCallLight(floor, true)||GetCallLight(floor, false))//门外up或down呼叫按钮工作

{

SetDoor(floor, true);

SetCallLight(floor, true, false);

SetCallLight(floor, false, false);//开门，并消费门外up和down按钮，防止下一周期重复进行此按钮行为

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

}

void StateMovingUp(int \*state)

{

int floor; bool up;

floor = GoingUpToFloor();

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//电梯运行到目标楼层

{

SetMotorPower(0);

SetDoor(floor, true);

\*state = DoorOpen;

SetCallLight(floor, true, false);

}

if (floor == Lib\_FloorNum)//到了最高层，消费门外down按钮

{

SetCallLight(floor, false, false);

}

SetPanelFloorLight(floor, false);

GetNearestFloor();

if (GetOpenDoorLight() || GetCloseDoorLight())//运动状态，开关门失效，无动作

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

void StateMovingDown(int \*state)

{

int floor; bool up;

floor = GoingDownToFloor();

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

{

SetMotorPower(0);

SetDoor(floor, true);

\*state = DoorOpen;

SetCallLight(floor, false, false);

if (floor == 1) //到了1层，消费门外up按钮

{

SetCallLight(floor, true, false);

}

}

SetPanelFloorLight(floor, false);

GetNearestFloor();

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);//无动作，消费开关门按钮

}

}

void StateDoorOpen(int \*state)

{

int floor; bool up;

floor = GetNearestFloor();

if (GetCloseDoorLight())

{

SetDoor(floor, false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

if (IsDoorOpen(floor))//开门结束后，自动进行关门，进入DoorClosing状态

{

SetDoor(floor, false);

\*state = DoorClosing;

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

}

}

void StateDoorClosing(int \*state)

{

int floor; bool up;

floor = GetNearestFloor();

if (GetOpenDoorLight())

{

SetDoor(floor, true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

if (IsBeamBroken())//如果红外探测到遮挡，转而开门，进入DoorOpen状态

{

SetDoor(floor, true);

\*state = DoorOpen;

}

if(IsDoorClosed(floor))

\*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");

}

}

}