**三层电梯状态机**

**课程设计报告**

**16040310033**

**赵伟**

**三层电梯状态机图**

**Moving up S5 Door closing**

**D S2 S1**

**Idle S7 S8**

**D S3 S4**

**Moving down Door open**

**S6**

**1.状态机图及其分解描述**：

三层电梯的运行过程分为空闲，向上，向下，开门，关门五个状态，如图所示。

Idle 状态及其事件

S1（闲置转向关门）：安全设置，无动作。

S2（由空闲转向上升的过程）：非当前楼层门外calllight被呼叫，或者门内向上的楼层数被点亮；

S3（由空闲转向下降的过程）：非当前楼层门外calllight被呼叫，或者门内向下的楼层数被点亮；

S4（由空闲转向开门）：当前楼层门外呼叫或者门内开门按钮点亮；

MovingUp/Down 状态及其事件

S5/S6（由向上升/向下降转向开门的过程）：目的楼层到了，或者中途同方向楼层被呼叫；

D在运动过程中出于安全考虑不允许开关门。

DoorOpen 状态及其事件

S7（由开关到关门的过程）：门内乘客手动关们; 开门动作结束后，门自动关闭；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。

**2.各状态函数设计过程：**

**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)**

Idle状态：

S1：检查E2事件, 此时门应该是关闭的，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。

S2: 检查E3、E4、E5事件，静态检测(检测将要到的目标楼层), 输入目标楼层floor=IdleWhatFloorToGoTo(&up); 消费门外up按钮SetCallLight()。

S3: 同S2，消费门外down按钮。一层以上，一定时间无动作，自动下降到1楼。AutoTo1Floor()。

S4检查E1事件, 开门, 消费开门按钮; 检查E4/E5事件, 开门SetDoor(),消费门外up/down按钮。

MovingUp/Down DoorOpen

S5：检查E3、E4、E5事件，动态检测，检测目标楼层 ,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。消费门内楼层按钮为当前楼层的按钮SetPanelFloorLight()。

(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，检测目标楼层。 到达目标楼层，停止，开门;消费门外down按钮， 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SetPanelFloorLight()

DoorOpen DoorClosing

S7：检查E2事件，正在开门，按了关门灯，转而关门; 消费关门按钮。开门结束后IsDoorOpen(); 自动进行关门，检查E1事件GetOpenDoorLight(), 正在开门，按了开门灯，无动作,消费开门按钮

DoorClosing

S8：检查E1事件; 正在关门，按了开门灯，转而开门SetDoor(); 消费开门按钮，检查E2事件GetCloseDoorLight()，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 IsBeamBroken()，转而开门SetDoor()，进入DoorOpen状态。S9：关门结束后IsDoorClosed(); 进入Idle状态。

**3.各状态函数设计流程图：**

空闲状态

**F**

进入开门状态

判断是否是门内按钮

判断输入电梯层数是否大于0并且上升

**T F**

判断电梯是否下降

电梯进入上升状态

**T F**

保持空闲状态

进入电梯下降状态

（下降）上升状态

判断是否在电梯上升时有门外按钮指令

检测目标楼层

到达目标楼层

保持（下降）上升状态，保证安全

进入开门状态，一段时间后自动进入关门状态

重新进入（下降）上升状态，到达下一目标楼层。若没有下一目标楼层，则一段时间后进入下降状态，到达一层。

关门状态

判断开门灯是否亮起

进入空闲状态

进入开门状态

检测目标楼层

进入上升或下降状态

到达目标楼层

开门状态

检测目标楼层

进入关门状态

到达目标楼层

**4.本程序的主要内容**

**#include "stdafx.h"9**

**#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;**

**bool s;**

**floor = IdleWhatFloorToGoTo(&up);**

**if (floor > 0 && up)**

**{**

**SetMotorPower(1); //检查门内楼层按钮，上升**

**\*state = MovingUp;**

**printf("Transition: from Idle to Moving up.\n");**

**}**

**else if (floor > 0 && !up)**

**{**

**SetMotorPower(-1); //同上，下降**

**\*state = MovingDown;**

**printf("Transition: from Idle to Moving down.\n");**

**}**

**else if (GetOpenDoorLight())**

**{**

**SetDoor(floor, true);**

**SetOpenDoorLight(false); //检查门内开门按钮**

**\*state = DoorOpen;**

**printf("Transition: from Idle to Door open.\n");**

**}**

**else if (GetCallLight(1, true))**

**{**

**SetDoor(1, true);**

**SetCallLight(1, true, false); //检查门外呼叫按钮**

**\*state = DoorOpen;**

**printf("Transition: from Idle to Door open.\n");**

**}**

**else if (GetCallLight(GetNearestFloor(), true))**

**{**

**SetDoor(floor, true);**

**SetCallLight(floor, true, false);**

**\*state = DoorOpen;**

**printf("Transition: from Idle to Door open.\n");**

**}**

**else if (GetCallLight(GetNearestFloor(), false))**

**{**

**SetDoor(floor, true);**

**SetCallLight(floor, false, false);**

**\*state = DoorOpen;**

**printf("Transition: from Idle to Door open.\n");**

**}**

**else if (GetCloseDoorLight())**

**{**

**SetCloseDoorLight(false);**

**return;**

**}**

**}**

**void StateMovingUp(int \*state)**

**{**

**int floor;**

**floor = GoingUpToFloor();//动态监测**

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

**{**

**SetMotorPower(0);**

**SetCallLight(floor, false, false);//消费按钮**

**SetPanelFloorLight(floor,false);//消费按钮**

**SetDoor(floor, true);**

**if (GetNearestFloor() == Lib\_FloorNum)**

**{**

**SetCallLight(floor, false, false);//改为消费向下按钮**

**}**

**\*state = DoorOpen;**

**printf("Transition: from MovingUp to Door open.\n");**

**}**

**if (GetOpenDoorLight() || GetCloseDoorLight())**

**{**

**SetOpenDoorLight(false);**

**SetCloseDoorLight(false);**

**return;**

**}**

**}**

**void StateMovingDown(int \*state)**

**{**

**int floor;**

**floor = GoingDownToFloor();**

**int GetNearestFloor();**

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

**{**

**SetMotorPower(0);**

**SetDoor(floor, true);**

**\*state = DoorOpen;**

**printf("Transition: from MovingDown to Door open.\n");**

**if (floor > 1)**

**{**

**SetCallLight(floor, false, false);**

**}**

**else**

**{**

**SetCallLight(floor, true, false);//改为消费向上按钮**

**}**

**SetPanelFloorLight(floor, false);**

**}**

**else if (GetOpenDoorLight() || GetCloseDoorLight())**

**{**

**SetOpenDoorLight(false);**

**SetCloseDoorLight(false);**

**return;**

**}**

**}**

**void StateDoorOpen(int \*state)**

**{**

**int floor;**

**floor = GoingUpToFloor();**

**if (GetCloseDoorLight())**

**{**

**SetDoor(GetNearestFloor(), false);**

**SetCloseDoorLight(false);**

**\*state = DoorClosing;**

**printf("Transition: from Door open to Door closing.\n");**

**}**

**else if (IsDoorOpen(GetNearestFloor()))**

**{**

**SetDoor(GetNearestFloor(), false);**

**\*state = DoorClosing;**

**printf("Transition: from Door open to Door closing.\n");**

**}**

**else if (GetOpenDoorLight())**

**{**

**SetOpenDoorLight(false);**

**\*state = DoorClosing;**

**printf("Transition: from Door open to Door closing.\n");**

**}**

**}**

**void StateDoorClosing(int \*state)**

**{**

**int floor;**

**if (GetOpenDoorLight())**

**{**

**SetDoor(floor, true);**

**SetOpenDoorLight(false);**

**\*state = DoorOpen;**

**printf("Transition: from Door closing to Door open.\n");**

**}**

**else if (GetCloseDoorLight())**

**{**

**SetCloseDoorLight(false);**

**return;**

**}**

**else if (IsBeamBroken())**

**{**

**SetDoor(floor, true);**

**\*state = DoorOpen;**

**printf("Transition: from Door closing to Door open.\n");**

**}**

**else if (IsDoorClosed(GetNearestFloor()))**

**\*state = Idle;**

**printf("Transition: from Door closing to Idle.\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");**

**}**

**}**

5.调试运行结果







