《C语言程序设计》

课程设计报告

16040310085牛豪豪

1. 设计题目

三层电梯状态机课程设计、

1. 课程设计要求
2. 三层电梯状态机课程设计报告(.docx文件)
   1. 状态机图及其分解描述
   2. 用流程图描述各个状态函数
   3. 状态机代码，注释清楚，结构简洁(在elevator.cpp中完成)
   4. 运行测试，描述实现的功能及测试结果
3. 完成elevator.cpp中的状态函数
4. 将以上两个文件压缩为一个文件，文件名：学号姓名.rar
5. 如果发现抄袭现象，相关同学的成绩以0分计
6. 状态机图及分解描述
   1. 状态机图

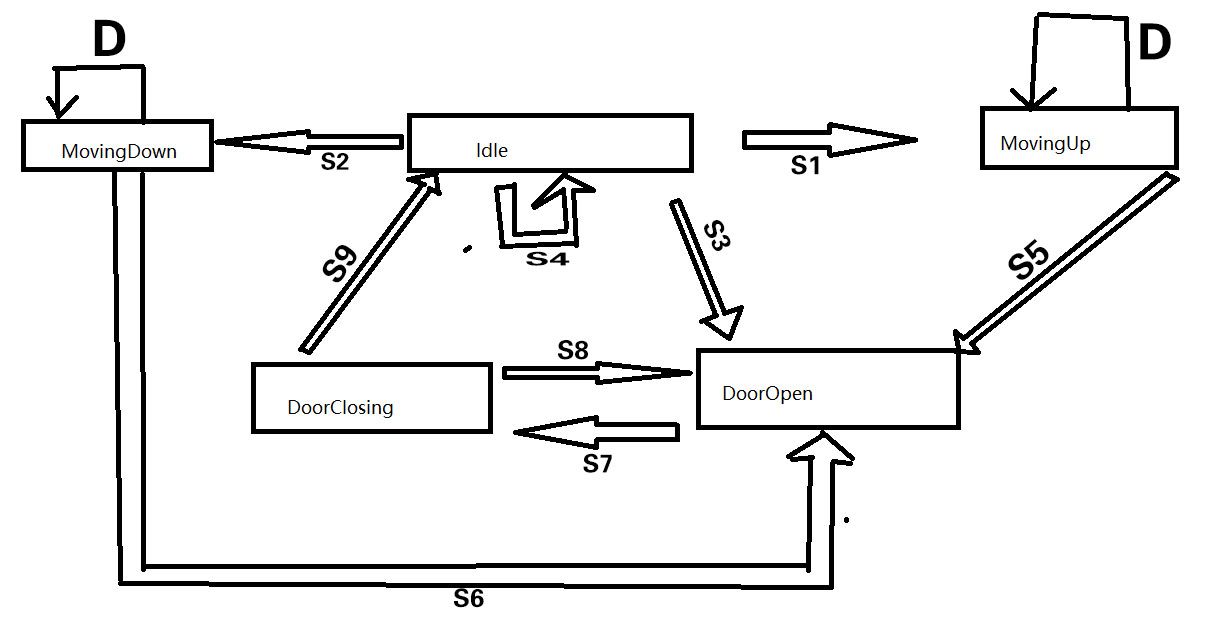


图1 三层电梯状态机图

|  |  |  |
| --- | --- | --- |
| 状态常数 | 状态函数 | 检查事件，执行动作，状态变迁 |
| Idle | StateIdle | S1,S2,S3,S4 |
| MovingUp | StateMovingUp | S5,D |
| MovingDown | StateMovingDown | S6,D |
| DoorOpen | StateDoorOpen | S7 |
| DoorClosing | StateDoorClosing | S8,S9 |

图2 三层电梯状态机分解概述表

1. 状态机分解描述
2. **Idle状态**

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。

1. **MovingUp/Down状态**

S5/S6：由向上升/向下降转向开门的过程，对应的事件有两个：其一是目的地到了，自然开门；其二是若上升/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，若有人按了开关门按键，无动作，并且及时灭掉按键。

1. **DoorOpen状态**

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。

1. **DoorClosing状态**

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。

1. 状态函数流程分析
2. **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. **Idle状态函数流程**

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),

int floor; bool up;

目标楼层floor=IdleWhatFloorToGoTo(&up); // Event

if (floor > 0 && up) {

上升SetMotorPower(1)。进入MovingUp状态。// Transition

}

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),

int floor; bool up;

目标楼层floor=IdleWhatFloorToGoTo(&up); // Event

if (floor > 0 && !up)

{上升SetMotorPower(-1)。进入MovingDown状态。// Transition}

Idle 🡪 DoorOpen

(S3) 检查E1事件GetOpenDoorLight(), 开门SetDoor(), 消费开门按钮SetOpenDoorLight(); 进入DoorOpen状态。

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。

if(GetCloseDoorLight()) { SetCloseDoorLight(false); return; }

1. **MovingUp状态函数流程**

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(); // 获取当前楼层

1. **MovingDown状态函数流程**

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(); // 获取当前楼层

1. **DoorOpen状态函数流程**

DoorOpen🡪 DoorClosing

(S7) GetNearestFloor()获取当前楼层

检查E2事件GetCloseDoorLight ()，正在开门，按了关门灯，转而关门SetDoor(); 消费关门按钮SetCloseDoorLight ()。进入DoorClosing状态。

开门结束后IsDoorOpen(); 自动进行关门SetDoor()，进入DoorClosing状态。

检查E1事件GetOpenDoorLight(), 正在开门，按了开门灯，无动作,消费开门按钮SetOpenDoorLight()。

1. **DoorClosing状态函数流程**

DoorClosing 🡪 DoorOpen

(S8) GetNearestFloor()获取当前楼层

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。

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()。

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

DoorClosing 🡪 Idle

(S9) 关门结束后IsDoorClosed(); 进入Idle状态。

**程序**

**#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。**

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

**bool up;**

**void StateIdle(int \*state) //空置状态函数**

**{**

**int floor;**

**bool up;**

**floor = IdleWhatFloorToGoTo(&up); // 判断目标楼层以及接下来的操作步骤**

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

**{**

**SetMotorPower(-1);**

**\*state = MovingDown; // 下降状态**

**}**

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

**{**

**SetMotorPower(1);**

**\*state = MovingUp; // 上升状态**

**}**

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

**if (GetOpenDoorLight()) //空置状态开门**

**{**

**SetDoor(GetNearestFloor(), 1);**

**SetOpenDoorLight(0);**

**\*state = DoorOpen; //开门状态**

**}**

**if (GetCallLight(GetNearestFloor(), 1)) //门外**

**{**

**SetDoor(GetNearestFloor(), 1); //开门**

**SetCallLight(GetNearestFloor(), 1, 0); //关灯**

**\*state = DoorOpen;**

**}**

**if (GetCallLight(GetNearestFloor(), 0)) //门外灯亮起**

**{**

**SetDoor(GetNearestFloor(), 1); //开门**

**SetCallLight(GetNearestFloor(), 0, 0); //门外灯熄灭**

**\*state = DoorOpen;**

**}**

**if (GetCloseDoorLight())**

**{**

**SetCloseDoorLight(0); //防止死循环，消费该功能按键**

**return;**

**}**

**}**

**void StateMovingUp(int \*state) //电梯上升状态函数**

**{**

**int floor;**

**floor = GoingUpToFloor();**

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

**{**

**SetMotorPower(0); //电机停止运转，等待指令**

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

**\*state = DoorOpen; //进入开门状态**

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

**SetCallLight(GetNearestFloor(), true, false); //消费门外上楼键**

**else (SetCallLight(GetNearestFloor(), false, false)); //消费顶层下楼键**

**}**

**SetPanelFloorLight(GetNearestFloor(), false); //消费门内制定楼层按键**

**if (GetOpenDoorLight())**

**SetOpenDoorLight(false);**

**if (GetCloseDoorLight())**

**SetCloseDoorLight(false);**

**}**

**void StateMovingDown(int \*state) //电梯下降状态函数**

**{**

**int floor;**

**floor = GoingDownToFloor();**

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

**{**

**SetMotorPower(0); //电机停止运转，电梯停下等候下一步指令**

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

**\*state = DoorOpen; //进入开门状态**

**if (GetNearestFloor() != 1)**

**SetCallLight(GetNearestFloor(), false, false);**

**else SetCallLight(1, true, false); //消费1层up键**

**}**

**SetPanelFloorLight(GetNearestFloor(), false); //消费门内该楼层按键**

**if (GetOpenDoorLight())**

**SetOpenDoorLight(false); //门内灯灭**

**if (GetCloseDoorLight())**

**SetCloseDoorLight(false);**

**}**

**void StateDoorOpen(int \*state) //开门状态函数**

**{**

**int floor;**

**floor = GetNearestFloor(); //主控程序获取当前楼层**

**if (GetCloseDoorLight())**

**{**

**SetDoor(floor, false); //关门**

**SetCloseDoorLight(false);**

**\*state = DoorClosing; //进入关门状态**

**}**

**if (IsDoorOpen(floor))**

**{**

**SetDoor(floor, false); //关门**

**\*state = DoorClosing; //进入关门状态**

**}**

**if (GetOpenDoorLight())**

**SetOpenDoorLight(false); //消费门内开门灯**

**}**

**void StateDoorClosing(int \*state) //关门状态函数**

**{**

**int floor;**

**floor = GetNearestFloor(); //获取当前楼层**

**if (GetOpenDoorLight())//电机停止运转，电梯停下等候下一步指令**

**{**

**SetDoor(floor, true); //开门**

**SetOpenDoorLight(false); //消费门内开门灯**

**\*state = DoorOpen; //进入开门状态**

**}**

**if (GetCloseDoorLight())**

**SetCloseDoorLight(false); //消费门内关门灯**

**if (IsBeamBroken()) //红外探测函数**

**{**

**SetDoor(floor, true);**

**\*state = DoorOpen; //进入开门状态**

**}**

**if (IsDoorClosed(floor))**

**\*state = Idle; //进入Idle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");**

**}**

**}**

**}**