《C语言程序设计》

设计报告

学院:机电工程学院

专业班级：1604031班

学生姓名:熊儒可 学号:16040310079

指导老师：段江涛

设计题目:三层电梯状态机仿真程序

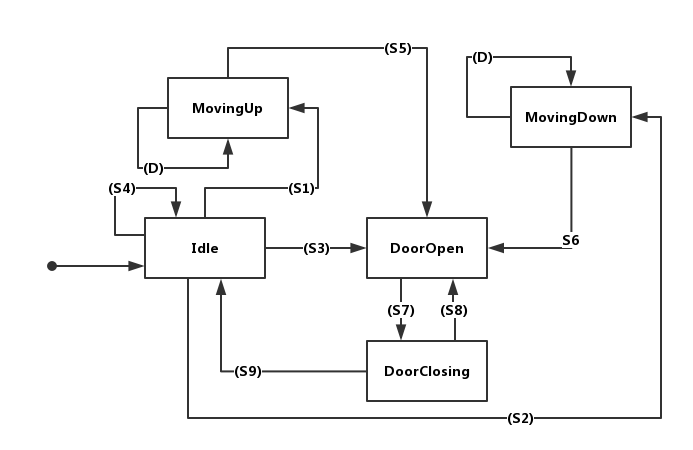
2017年05月18日

一．设计题目：三层状态机课程设计

二．设计方案：穷举法（考虑每个楼层各种情况）

对所有可能发生情况就行列举，通过日常电梯运动规律，一共分为5种情况：静止(Idle),上升（MovingUp）,下降（MovingDown）,门开（DoorOpen）,门正在关（DoorClosing）。分别进行讨论。

三．状态机图及其分解描述



|  |  |  |
| --- | --- | --- |
| 状态常数 | 状态函数 | 检查事件，执行动作（出弧标识），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 |

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状态时：

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。

\*State

检测到其他楼层灯亮？

检测到该楼层按钮灯亮？

否 否

是

\*state不变，函数继续运行

是

上升按钮亮？

是

\*state=DoorOpen

\*state=MovingUp

V 否

\*state=MovingDown

返回\*state

(S1) 其余楼层灯亮，检测将要到的目标楼层

目标楼层floor=IdleWhatFloorToGoTo(&up);

如果函数返回值为1，且上方灯亮

上升 SetMotorPower(1)。进入MovingUp状态。

(S2) 如果函数返回值为1，且下方灯亮

下降 SetMotorPower(-1)。进入MovingDown状态。

(S3) 本楼层灯亮 Floor=GetNearestFloor();使楼外按钮对应。

如果门内灯亮 GetOpenDoorLight(), 开门SetDoor(), 消费开门按钮

SetOpenDoorLight(); 进入DoorOpen状态。

如果门外灯亮GetCallLight(), 开门SetDoor(),消费门外up/down按钮

SetCallLight(); 进入DoorOpen状态。

(S4) 门内关门灯亮 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，并返回主函数。

1. MovingUP状态：

\*State

检测到最近楼层灯亮？

是 否

\*state不变，函数继续运行

\*state=DoorOpen

返回\*state

(S5) 到达目标楼层，目标楼层floor=GoingUpToFloor();

停止 SetMotorPower(0)，开门SetDoor();进入DoorOpen状态。

消费门外按钮，通过两个if 语句完成

if (GetCallLight(floor, true))

{SetCallLight(floor, true, false);

\*state = DoorOpen;return;}

if (GetCallLight(floor, false))

SetCallLight(floor, false, false);

SetPanelFloorLight(floor, false);}

(D) 电梯运动中，开关门失效，按下开关门按钮

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

1. MovingDown状态：

\*State

\*state不变，函数继续运行

检测到最近楼层灯亮？

是 否

\*state=DoorOpen

返回\*state

(S6) 到达目标楼层，floor=GoingUpToFloor();

停止 SetMtorPower(0)，开门SetDoor();进入DoorOpen状态。

同MovingUp状态，消费门外按钮，通过两个if 语句完成。

(D) 电梯运动中，开关门失效

GetOpenDoorLight()/GetCloseDoorLight()，无动作，消费开/关门按钮

SetOpenDoorLight()/SetCloseDoorLight()。

4.DoorOpen状态：

\*State

是否按下关门灯？

开门是否完毕？

否

是 是 否

\*state=DoorClosing

\*state不变，函数继续运行

返回\*state

(S7) 门内关门灯亮 GetCloseDoorLight ()，关门SetDoor(); 消费关门按钮SetCloseDoorLight ()。进入DoorClosing状态。

门内开门灯未亮，开门结束后IsDoorOpen(); 自动进行关门SetDoor()，进入DoorClosing状态。

如正在开门，按了开门灯，无动作,消费开门按钮SetOpenDoorLight()。

1. DoorClosing状态：

\*State

是否按下开门灯？

是

\*state=DoorOpen

否

是

是否有障碍物阻挡？

返回\*state

否

关门是否完毕？

是

\*state=Idle

否

\*state不变，函数继续运行

(S8) 如果按下开门灯 GetOpenDoorLight(); 开门SetDoor(); 消费开门按钮

SetOpenDoorLight()。进入DoorOpen状态。

如果正在关门，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。

(S9) 关门结束后IsDoorClosed(); 进入Idle状态。

四．运行测试。

经过测试，该三层电梯状态机能实现以下功能：

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。

五．源代码

#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);//获取将要到达楼层

if (floor > 0 && up)

{

SetMotorPower(1);

\*state = MovingUp;

printf("Transition: from Idle to MovingUp.\n");

}

if (floor > 0 && !up)

{

SetMotorPower(-1);

\*state = MovingDown;

printf("Transition: from Idle to MovingDown.\n");

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

floor = GetNearestFloor();//使参数floor对应该楼层

if (GetCallLight(floor, true))//如果该层楼外向上灯亮，开门，消费向上灯

{

SetCallLight(floor, true, false);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

if (GetCallLight(floor, false))//如果该层楼外向下灯亮，开门，消费向上下灯

{

SetCallLight(floor, false, false);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

}

void StateMovingUp(int \*state)

{

int floor;

floor = GoingUpToFloor();//获取上升时即将到达的楼层

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

{

if (GetCallLight(floor, true))//第一个if控制消费目标楼层向上灯按钮及门内按钮

{

SetCallLight(floor, true, false);

SetPanelFloorLight(floor, false);

SetMotorPower(0);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from MovingUp to DoorOpen.\n");

return;

}

if (GetCallLight(floor, false))//第二个if控制消费目标楼层向下灯按钮及门内按钮

SetCallLight(floor, false, false);

SetPanelFloorLight(floor, false);

SetMotorPower(0);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from MovingUp to DoorOpen.\n");

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

}

void StateMovingDown(int \*state)

{

int floor;

floor = GoingDownToFloor();//获取下降时即将到达的楼层

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

{

if (GetCallLight(floor, true))//第一个if控制消费目标楼层向上灯按钮及门内按钮

{

SetCallLight(floor, true, false);

SetPanelFloorLight(floor, false);

SetMotorPower(0);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from MovingDown to DoorOpen.\n");

return;

}

if (GetCallLight(floor, false))//第二个if控制消费目标楼层向下灯按钮及门内按钮

SetCallLight(floor, false, false);

SetPanelFloorLight(floor, false);

SetMotorPower(0);

SetDoor(floor, true);

\*state = DoorOpen;

printf("Transition: from MovingDown to DoorOpen.\n");

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

}

void StateDoorOpen(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetCloseDoorLight())

{

SetDoor(floor, false);

SetCloseDoorLight(false);

\*state = DoorClosing;

printf("Transition: from DoorOpen to DoorClosing.\n");

}

if (IsDoorOpen(floor))//检测门是否完全打开

{

SetDoor(floor, false);

\*state = DoorClosing;

printf("Transition: from DoorOpen to DoorClosing.\n");

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

return;

}

}

void StateDoorClosing(int \*state)

{

int floor;

floor = GetNearestFloor();

if (GetOpenDoorLight())

{

SetDoor(floor, true);

SetOpenDoorLight(false);

\*state = DoorOpen;

printf("Transition: from DoorClosing to DoorOpen.\n");

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

if (IsBeamBroken())//检测是否有障碍物

{

SetDoor(floor, true);

SetOpenDoorLight(false);

\*state = DoorOpen;

printf("Transition: from DoorClosing to DoorOpen.\n");

}

if (IsDoorClosed(floor))//检测门是否完关闭

{

\*state = Idle;

printf("Transition: from DoorClosing 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");

}

}

}