C语言程序设计

课程报告

姓名：周鹏辉

班级：1604031

学号：16040310041

指导老师：段江涛

2017年5月

状态图及表格

E

MovingDown

C

Doorclosing

B

DoorOpen

A

Idle

D

MovingUp

程序流程图

**main**

**StateMovingDown**

**StateDoorClosing**

S**tateMovingUp**

**StateIdle**

**StateDoorOpen**

StateIdle

StateIdle

**GetCallLight**

**DoorOpen**

**CloseLight**

**MovingUp**

**MovingDown**

**State**

**MovingUp**

**StateMovingDown**

**Getfloor-floor**

**Getfloor-floor**

**移动状态不开门**

**移动状态不开门**

**State**

Door

Open

GetCloseLight

IsDoorOpen

GetDoorOpen

**01**

**02**

**04**

**03**

GetOpenLight

IsDoorClosed

GetCloseLight

IsBeamBroken

**Idle状态及其事件**

Idle-Moving:由闲置转向向上升的过程，对应的事件GetCallLight，或者门内向上的楼层数被点亮。

Idle-MovingDown：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。

Idle-DoorOpen：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。

Idle-Idle：闲置状态下关门，无动作，灭掉门内该按键。

**MovingUp/Down状态及其事件**

MovingUp/MovingDown-Idle：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。

运行过程不开门：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。

**DoorOpen状态及其事件**

DoorOpen-DoorClosing：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。

**DoorClosing状态及其事件**

DoorClosing-DoorOpen：由关门到开门的过程，对应门内开门按键被点亮，电梯开门。

DoorClosing-Idle：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。

**各函数设计思路:**

为方便描述，用E(event)代替进行简易描述。

设置如下：

E1: OpenDoorLight

E2: CloseDoorLight

E3: PanelFloorLight

E4: Up Call Light

E5: Down Call Light

**StateIdle**

情况一：检查E3、E4、E5事件。

该检查属于静态检测。

部分代码如下：

bool up;

floor=IdleWhatFloorToGoTo(&up);

函数判断将要去往的楼层。

消费本层门外up按钮。

函数名如下：

SetCallLight()

函数用于消费门外up按钮，防止下一周期重复处理此按钮行为。

情况二：一段时间不进行操作且电梯位于1层以上时，电梯自动下降到1楼。

所用函数名如下：

AutoTo1Floor();

用于自动返回一楼。

情况三：检查E1事件, 开门, 消费开门按钮;

函数如下：

if (floor > 0 && up)

上升，到达后消费门外up按钮

if (floor > 0 && !up)

下降，到达后消费门外down按钮

情况四：检查E2事件,此时门应该是关闭的,因此仅读取关门灯，并关闭关门灯

部分代码如下

if(GetCloseDoorLight())

{ SetCloseDoorLight(false);

return; }

该函数用以判断门关闭直接消费。

**StateMovingUp/MovingDown 🡪 StateDoorOpen**

情况五：检查E3、E4、E5事件。

该检查为动态检测

函数如下

floor=GoingUpToFloor();

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

该函数用以判断到达目标楼层，停止，开门，消费门外按钮。

情况六：检查E3、E4、E5事件。

该检查为动态检测

函数如下

floor=GoingDownToFloor();

细节部分与情况五类似

**DoorClosing**

情况七：检查E2事件，转而关门

函数如下

GetCloseDoorLight(),SetDoor();

关门状态函数如下

IsDoorOpen();SetDoor();

**StateDoorClosing**

情况八：检查E1事件，转而开门。

函数如下

GetOpenDoorLight();SetDoor();

情况九：关门结束后，进入Idle状态。

函数如下

IsDoorClosed();

程序如下：

#include "stdafx.h"

StateDoorCloseing:

#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); //确定前往楼层

if (floor > 0 && up){ //判断状态

SetMotorPower(1); //打开电机

\*state = MovingUp; //进入下一状态

}

if (floor > 0 && !up){ //状态同上

SetMotorPower(-1);

\*state = MovingDown;

}

if (GetOpenDoorLight()){ //判断开门灯

SetDoor(floor, true); //开门

SetOpenDoorLight(false); //消费开门灯

\*state = DoorOpen; //进入下一状态

}

if (GetCloseDoorLight()){ //无状态，无变化

SetCloseDoorLight(false);

return;

}

int getfloor;

getfloor = GetNearestFloor();

if(GetCallLight(GetNearestFloor(),true)||GetCallLight(GetNearestFloor(), false)){ //判断电梯与呼唤灯是否同一层

SetDoor(getfloor, true); //开门

SetCallLight(getfloor, true, false); //消费

SetCallLight(getfloor, false, false);

\*state = DoorOpen; //进入下一状态

}

}

void StateMovingUp(int \*state)

{

int floor;

floor = GoingUpToFloor();

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance){ //判断是否到达

SetMotorPower(0); //关闭电机

SetDoor(floor, true); //开门

SetCallLight(floor, true, false); //消费外灯

SetCallLight(floor, false, false);

SetPanelFloorLight(floor, false); /\*\*消费内部楼层灯 \*\*/

\*state = DoorOpen; /\*\*下一状态 \*\*/

}

if (GetOpenDoorLight() || GetCloseDoorLight()){ /\*\*移动状态无法开关门 \*\*/

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

void StateMovingDown(int \*state) //状态与上升相同

{

int floor;

floor = GoingDownToFloor();

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

SetMotorPower(0);

SetDoor(floor, true);

SetCallLight(floor, true, false);

SetCallLight(floor, false, false);

SetPanelFloorLight(floor, false);

\*state = DoorOpen;

}

if (GetOpenDoorLight() || GetCloseDoorLight()){

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

void StateDoorOpen(int \*state)

{

int floor = GetNearestFloor();

if (GetCloseDoorLight()){ //判断关门灯

SetDoor(floor,false); //关门

SetCloseDoorLight(false); //消费

\*state = DoorClosing;

}

if (IsDoorOpen(floor)){ //开门中进行关门

SetDoor(floor, false);

\*state = DoorClosing;

}

if (GetOpenDoorLight()){ //开门灯无状态

SetOpenDoorLight(false);

}

}

void StateDoorClosing(int \*state)

{

int floor = GetNearestFloor();

if (GetOpenDoorLight()){ //状态与开门时关门类似

SetDoor(floor, true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCloseDoorLight()){

SetCloseDoorLight(false);

}

if (IsBeamBroken()){ //判断是否有物体在电梯门口

SetDoor(floor ,true); //开门

\*state = DoorOpen; //进入下一状态

}

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");

}

}

}