## **三层电梯课程设计报告**

## 

## 姓名 凌之恒

**教师 段江涛**

**班级 自动化1604031**

**学号 16040310007**

**日期 17年5月25日**

一，主状态图

分为五个部分分为空闲，向上，向下，开门，关门五个状态。

**DoorClosing**

**MovingDown**

**Idle**

**MovingUp**

**DoorOpen**

总体概念图

二，各小部分详解

1,Idle空闲转其他 状态及其事件

**Idle**

**MovingUp/MovingDown**

if (floor > 0 && up)//电梯从静止状态上升

{

SetMotorPower(1);

\*state = MovingUp;

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

}

else if (floor>0 && !up)//无操作，下降

{

SetMotorPower(-1);

\*state = MovingDown;

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

}

**MovingDown**

**Idle**

**MovingUp**

else if (GetOpenDoorLight())//消费开门按钮

{

SetDoor(GetNearestFloor(),true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

else if (GetCallLight(GetNearestFloor(),true))//消费门外up/down按钮

{

SetDoor(GetNearestFloor(), true);

SetCallLight(GetNearestFloor(),true,false);

\*state = DoorOpen;

}

闲置转向向上升或下降的过程。对应的事件是非当前楼层门外calllight被呼叫，或者门内向上或向下的楼层数被点亮。

**DoorOpen/DoorClosing**

c

**Idle**

else if (GetCallLight(GetNearestFloor(), false))//关闭按钮灯，防止下一周期重复处理此按钮行为。

{

SetDoor(GetNearestFloor(), true);

SetCallLight(GetNearestFloor(), false, false);

\*state = DoorOpen;

}

else if (GetCloseDoorLight())//因此仅读取关门灯，并关闭关门灯

{

SetCloseDoorLight(false);

return;

}

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。

闲置状态下关门。无动作，灭掉门内该按键。

2,MovingUp/Down电梯上升及下降 状态及其事件

**MovingUp**

**Idle**

.

.**MovingDown**

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)// 到达目标楼层,开门

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), true, false);

if (floor = Lib\_FloorNum)//判断是否为最高层

{

SetCallLight(GetNearestFloor(), false, false);

}

SetPanelFloorLight(GetNearestFloor(),false);

}

---------------------------------------------------------------------------------------------------------------------------------

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//判断是否到达目标楼层

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), false, false);

if (floor = 1)//消费门外up按钮

{

SetCallLight(GetNearestFloor(), true, false);

}

SetPanelFloorLight(GetNearestFloor(), false);

}

以上为分别电梯上下状态函数

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。

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

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

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，不进行操作。并且及时灭掉按键。

3，DoorOpen开门 状态及其事件

**DoorOpen**

。

**Idle**

。

？

**DoorClosing**

if (GetCloseDoorLight())//开门中按关门键，进行关门操作

{

SetDoor(GetNearestFloor(),false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

if (IsDoorOpen(floor))//开门结束后，自动关门

{

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

}

if (GetOpenDoorLight())//开门中，按开门后无动作

{

SetOpenDoorLight(false);

}

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。

4，DoorClosing关门 状态及其事件

**DoorClosing**

**Idle**

。

**DoorOpen**

。

？

if (GetOpenDoorLight())//关门中按开门键，进行开门操作

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

由关门到开门的过程，对应门内开门按键点亮，电梯开门。

if (GetCloseDoorLight())//关门，按了关门灯

{

SetCloseDoorLight(false);

}

if (IsBeamBroken())//判断是否红外线阻挡

{

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

if (IsDoorClosed(floor))//关门结束后，电机静止，等待下一步指令

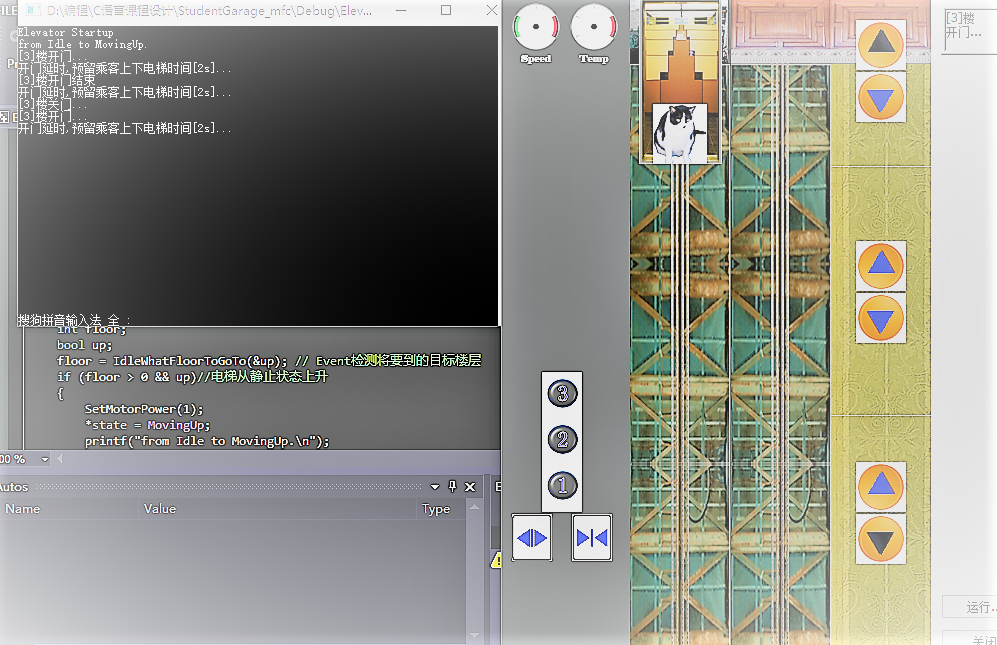
{

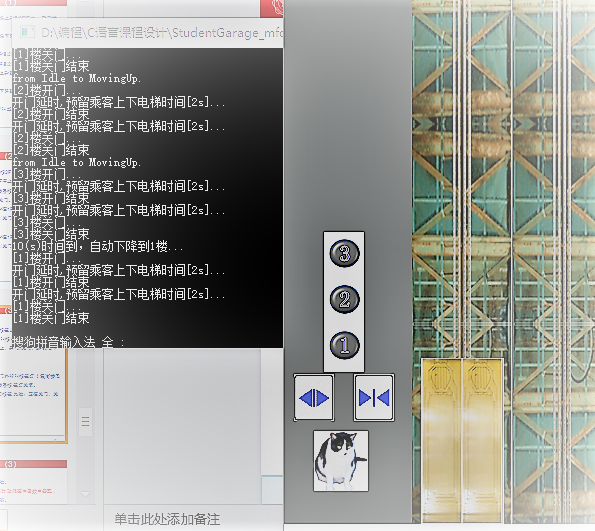
\*state = Idle;

}

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。

三，运行实例





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。

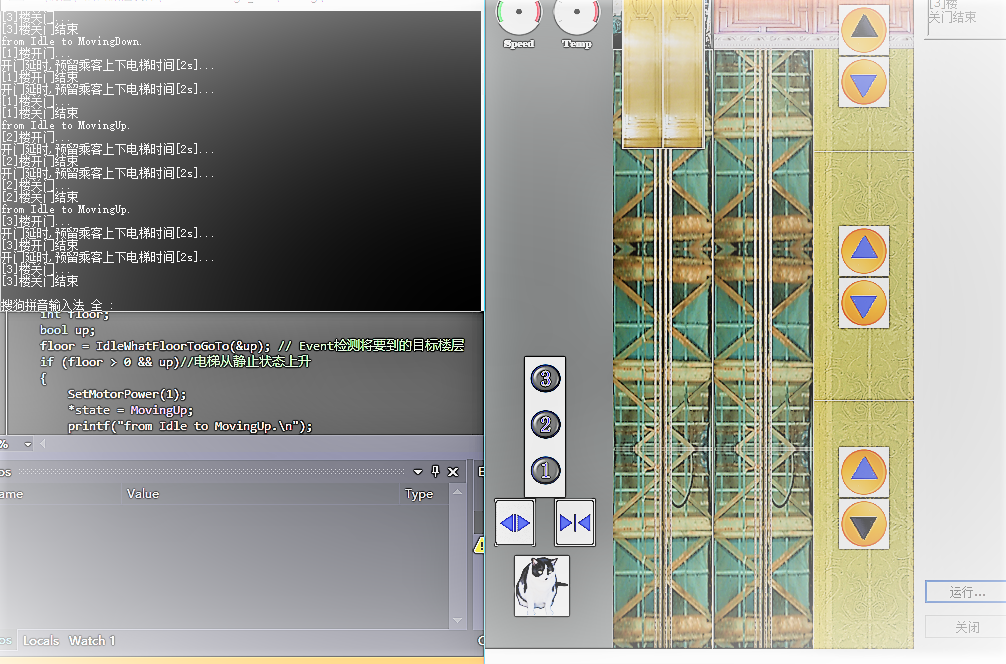
电梯停于2F或3F时，按1F向上呼叫按钮；电梯下降到1F停止，开门/关门。

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。

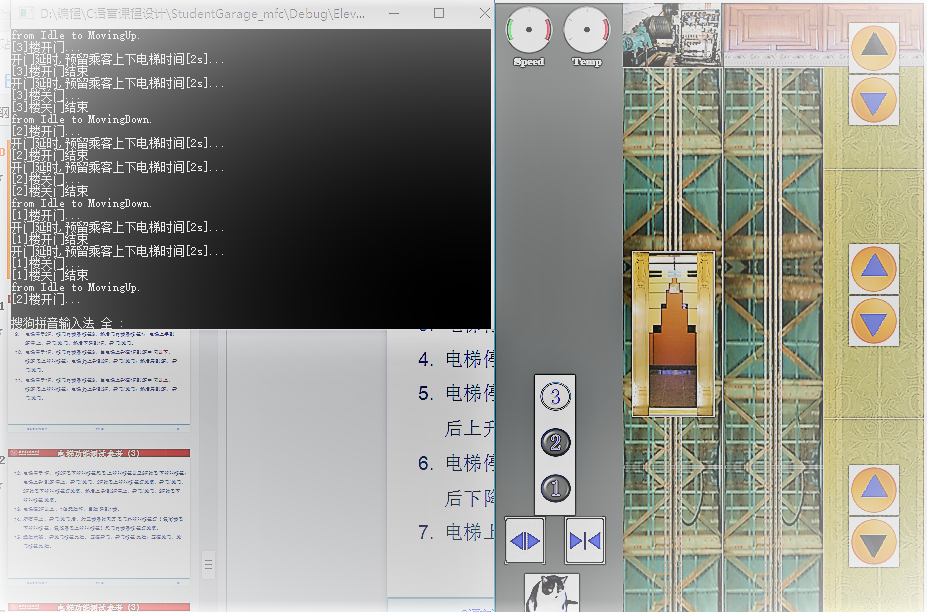
电梯停于3F时，按2F向下呼叫按钮；电梯下降到2F停止，开门/关门。

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。

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。



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。



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。

四，所编程序

#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。

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

void StateIdle(int \*state)//空闲状态函数

{

int floor;

bool up;

floor = IdleWhatFloorToGoTo(&up); // Event检测将要到的目标楼层

if (floor > 0 && up)//电梯从静止状态上升

{

SetMotorPower(1);

\*state = MovingUp;

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

}

else if (floor>0 && !up)//无操作，下降

{

SetMotorPower(-1);

\*state = MovingDown;

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

}

else if (GetOpenDoorLight())//消费开门按钮

{

SetDoor(GetNearestFloor(),true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

else if (GetCallLight(GetNearestFloor(),true))//消费门外up/down按钮

{

SetDoor(GetNearestFloor(), true);

SetCallLight(GetNearestFloor(),true,false);

\*state = DoorOpen;

}

else if (GetCallLight(GetNearestFloor(), false))//关闭按钮灯，防止下一周期重复处理此按钮行为。

{

SetDoor(GetNearestFloor(), true);

SetCallLight(GetNearestFloor(), false, false);

\*state = DoorOpen;

}

else if (GetCloseDoorLight())//因此仅读取关门灯，并关闭关门灯

{

SetCloseDoorLight(false);

return;

}

}

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

{

int floor = GoingUpToFloor();

bool up;

up = GetCallLight(floor, true);

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)// 到达目标楼层,开门

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), true, false);

if (floor = Lib\_FloorNum)//判断是否为最高层

{

SetCallLight(GetNearestFloor(), false, false);

}

SetPanelFloorLight(GetNearestFloor(),false);

}

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

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

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

{

int floor = GoingDownToFloor();

bool up;

up = GetCallLight(floor, true);

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//判断是否到达目标楼层

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), false, false);

if (floor = 1)//消费门外up按钮

{

SetCallLight(GetNearestFloor(), true, false);

}

SetPanelFloorLight(GetNearestFloor(), false);

}

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

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

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

{

int floor = GetNearestFloor();

bool up;

if (GetCloseDoorLight())//开门中按关门键，进行关门操作

{

SetDoor(GetNearestFloor(),false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

if (IsDoorOpen(floor))//开门结束后，自动关门

{

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

}

if (GetOpenDoorLight())//开门中，按开门后无动作

{

SetOpenDoorLight(false);

}

}

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

{

int floor = GetNearestFloor();

bool up;

if (GetOpenDoorLight())//关门中按开门键，进行开门操作

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCloseDoorLight())//关门，按了关门灯

{

SetCloseDoorLight(false);

}

if (IsBeamBroken())//判断是否红外线阻挡

{

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

if (IsDoorClosed(floor))//关门结束后，电机静止，等待下一步指令

{

\*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");

}

}

}