**《C语言程序设计》**

**实验设计报告**

学院:\_机电工程

专业班级: 自动化31班

学生姓名: 陈利达

学号: 16040310089

指导老师： 段江涛

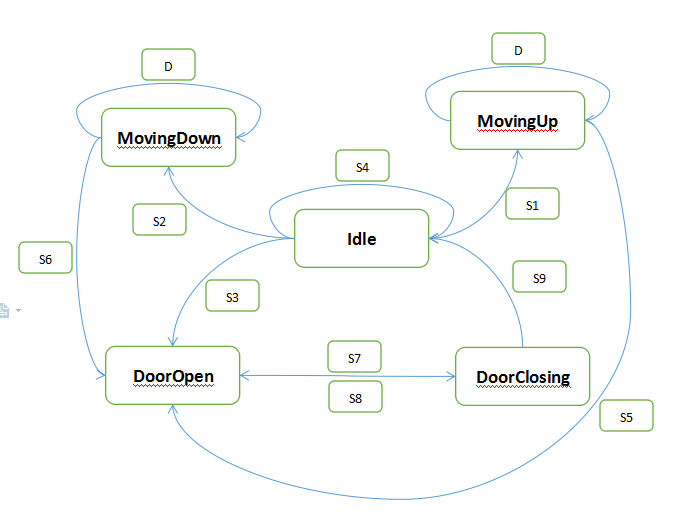
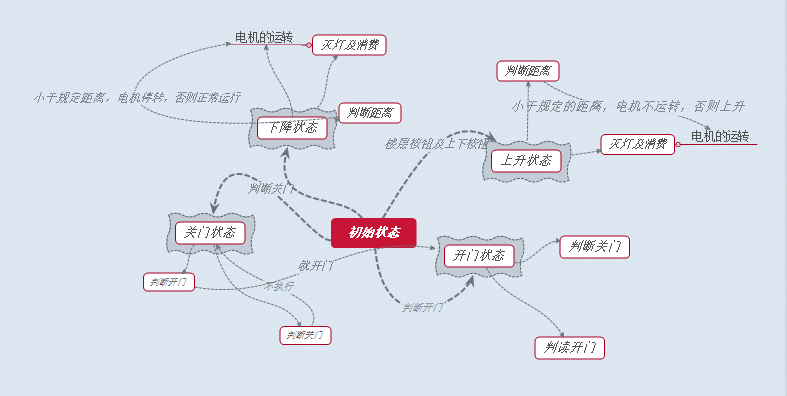
设计题目: \_\_\_三层电梯状态机编程\_\_

2017年05月25日

**三层电梯状态机课程设计报告**

1. **状态机图及其分解描述：**

将三层升降电梯的运行过程分为空闲，向上，向下，开门，关门五个状态。如下图所示。

****

|  |  |  |
| --- | --- | --- |
| **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** |

Idle状态及其事件

S1：由闲置转向上升的过程，对应的事件是高于当前楼层门外calllight被呼叫或者门内高于当前楼层数被点亮。

S2：由闲置转向下降的过程，对应的事件是低于当前楼层门外calllight被呼叫，或者门内低于当前楼层数被点亮。

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当前楼层按钮点亮。

S4：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

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，则在中途楼层停下，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。

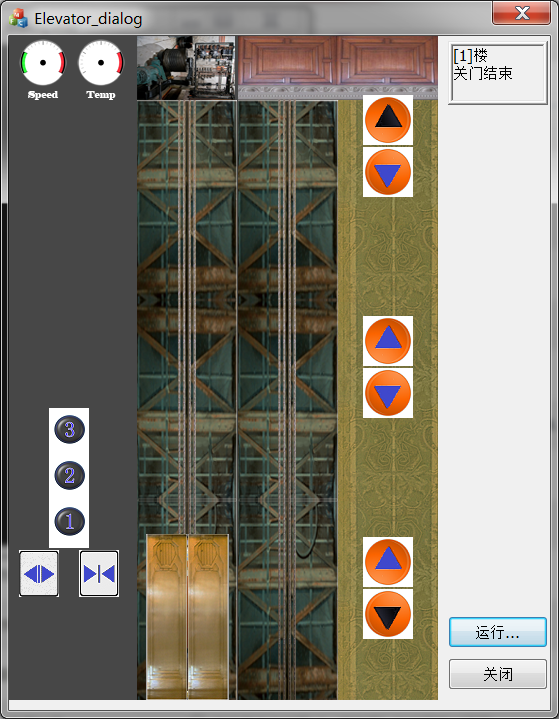
DoorOpen状态及其事件

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。

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**

**门外呼叫按钮：**

**Up/Down Call Light**

失效

**门内开关门按钮：**

**Open/Close Door Light**

Call Light

**门内楼层数字按钮：**

**Panel Floor Light**

Close Door Light

失效

Panel Floor Light

Open Door Light

**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)** |

**各状态函数流程：**

**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。**

**Idle变为MovingUp**

**(S1)检查E3、E4、E5事件，已经封装在下列函数中。（根据布尔变量up的值，以及 IdleWhatFloorToGoTo（）返回值来确定电梯下一步应该是怎样的状态。）**

**静态检测(检测将要到的目标楼层),**

**关闭本层门外up按钮，SetCallLight()—— 即消费门外up按钮，防止下一周期重复处理此按钮行为。**

**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变为MovingDown**

**(S2)检查E3、E4、E5事件，已经封装在下列函数中。消费门外down按钮。**

**1层以上，一定时间无动作，自动下降到1楼。AutoTo1Floor();[其它状态，取消此功能，CancelTo1Floor()]**

**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变为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。**

**消费按钮：关闭按钮灯，防止下一周期重复处理此按钮行为。**

**GetNearestFloor(); // 获取当前楼层**

**Idle变为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; }**

**MovingUp状态：**

**MovingUp变为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**

**MovingUp状态: 安全设置，无动作。**

**(D)运动状态，开关门失效，检查E1、E2事**

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

**SetOpenDoorLight()/SetCloseDoorLight()。**

**MovingDown状态：**

**MovingDown变为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**

**MovingDown状态: 安全设置，无动作。**

**(D)运动状态，开关门失效，检查E1、E2事件**

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

**SetOpenDoorLight()/SetCloseDoorLight()。**

**DoorOpen状态:**

**DoorOpen变为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()。**

**DoorClosing状态:**

**DoorClosing变为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状态:**

**DoorClosing变为Idle**

**(S9)关门结束后IsDoorClosed(); 进入Idle状态。**

1. **状态机代码**

#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);

if (floor > 0 && up) {

SetMotorPower(1);

\*state = MovingUp;

}

else if (floor = floor > 0 && !up){

SetMotorPower(-1);

\*state = MovingDown;

}

else if (GetOpenDoorLight()){

SetOpenDoorLight(false);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

else if (GetCallLight(floor, up))

{

SetDoor(floor, true);

SetCallLight(floor, up, false);

\*state = DoorOpen;

}

else if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

return;

}

//else { printf("你进入了一个程序猿小哥也不知道的次元"); }

}

void StateMovingUp(int \*state)

{

int floor; bool up;

floor = GoingDownToFloor();

up = GetCallLight(floor, true);

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

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), true, false);

if (GetNearestFloor() == Lib\_FloorNum) { SetCallLight(GetNearestFloor(), false, false); }

SetPanelFloorLight(GetNearestFloor(), false);

}

else if (GetOpenDoorLight() == true) { SetOpenDoorLight(false); }

else if (GetCloseDoorLight() == true) { SetCloseDoorLight(false); }

}

void StateMovingDown(int \*state)

{

int floor; bool up;

floor = GoingUpToFloor();

up = GetCallLight(floor, true);

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

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), false, false);

if (GetNearestFloor() == 1){ SetCallLight(GetNearestFloor(), true, false); }

SetPanelFloorLight(GetNearestFloor(), false);

}

else if (GetOpenDoorLight() == true) { SetOpenDoorLight(false); }

else if (GetCloseDoorLight() == true) { SetCloseDoorLight(false); }

}

void StateDoorOpen(int \*state)

{

if (GetCloseDoorLight()){

SetDoor(GetNearestFloor(), false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

else if (IsDoorOpen(GetNearestFloor())) {

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

}

else if (GetOpenDoorLight()) {

SetOpenDoorLight(false);

}

}

void StateDoorClosing(int \*state)

{

if (GetOpenDoorLight()){

SetDoor(GetNearestFloor(), false);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

else if (GetCloseDoorLight()){

SetCloseDoorLight(false);

}

else if (IsBeamBroken()){

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

else if (IsDoorClosed(GetNearestFloor())){

\*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:

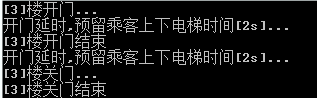
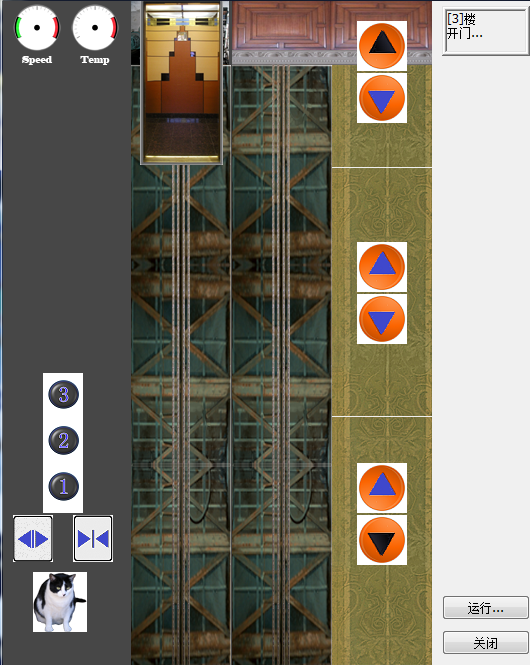
printf("没有这种状态!!!\n");

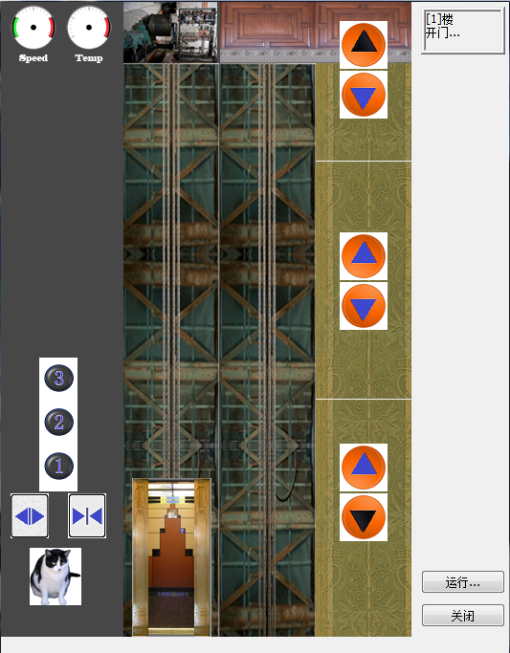
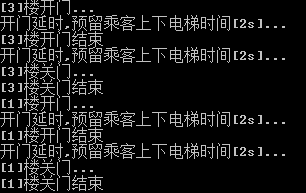
}

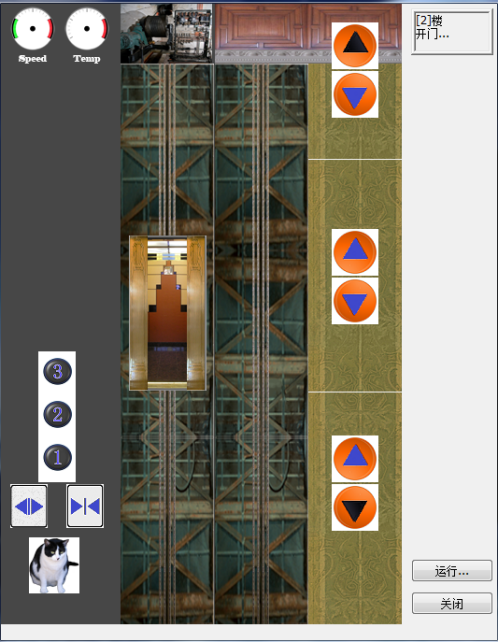
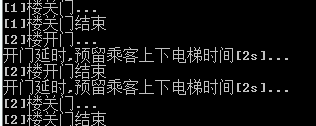
}

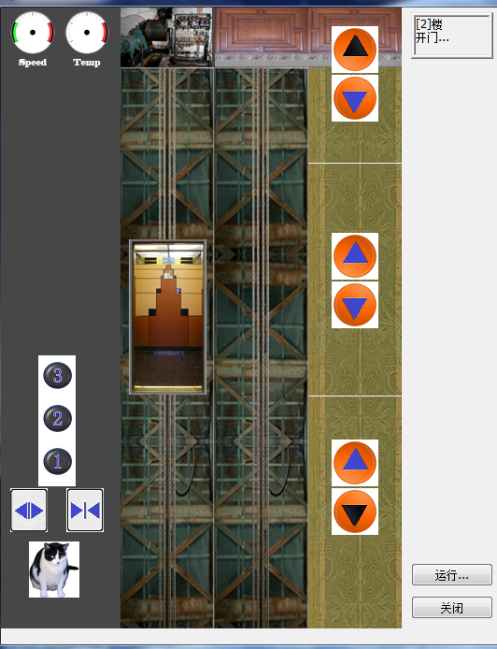
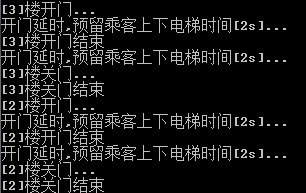
}

1. **电梯功能测试**

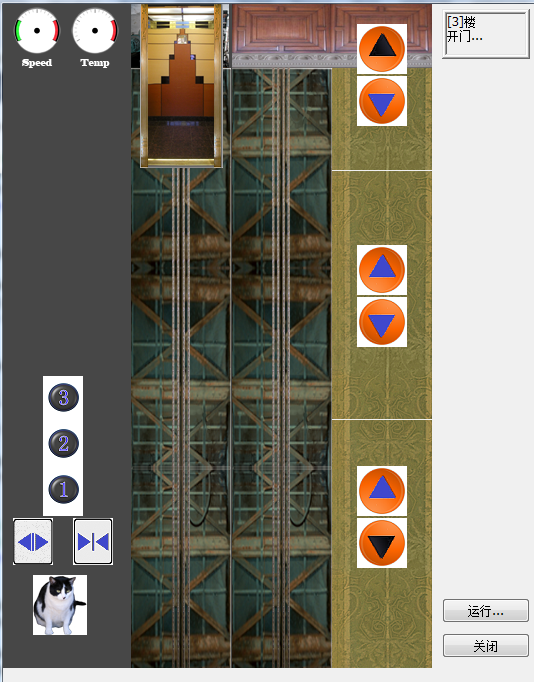
**1.电梯停于1F或2F时，按3F向下呼叫按钮；电梯上升到3F停止，开门/关门。**

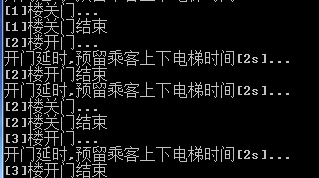
**2.电梯停于2F或3F时，按1F向上呼叫按钮；电梯下降到1F停止，开门/关门。**

**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。**

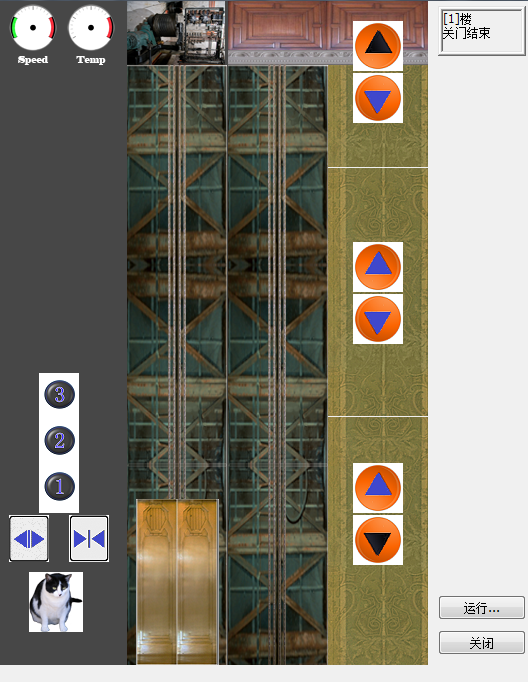
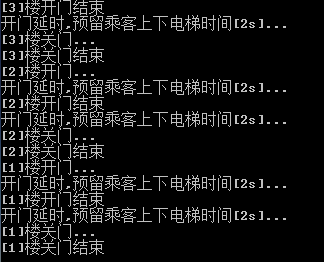
**4.电梯停于3F时，按2F向下呼叫按钮；电梯下降到2F停止，开门/关门。**

1. **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。**

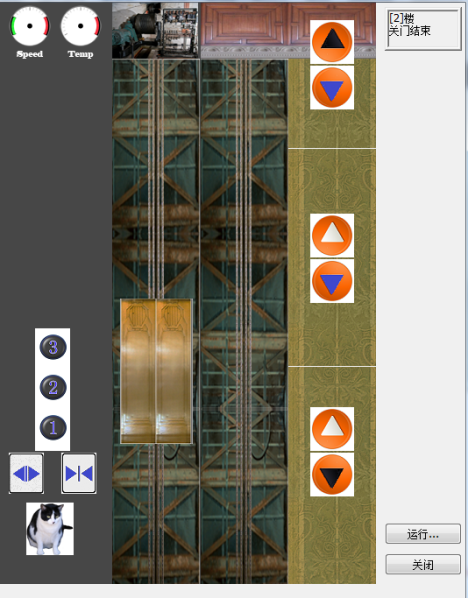


****

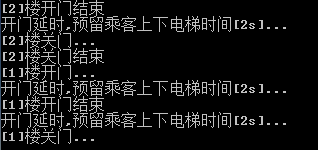
1. **电梯停于3F，2F和1F均有按钮呼叫；电梯先下降到2F，开门/关门，然后下降到1F停止，开门/关门。**

****

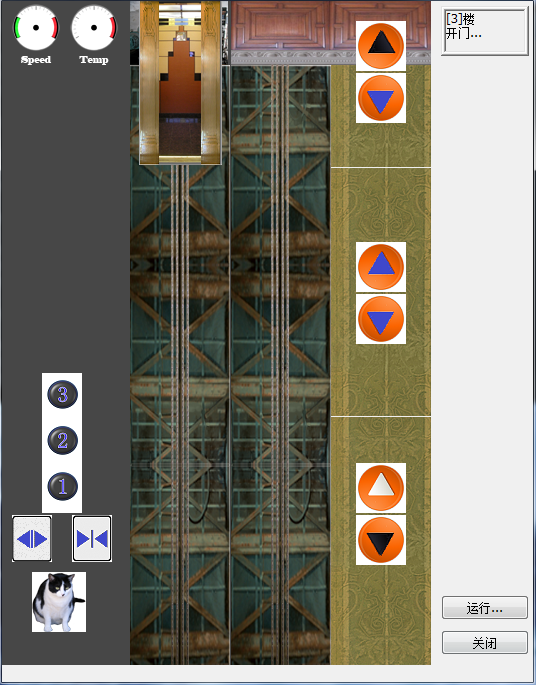
1. **电梯上升途中或下降途中，任何反方向按钮呼叫均无效。**



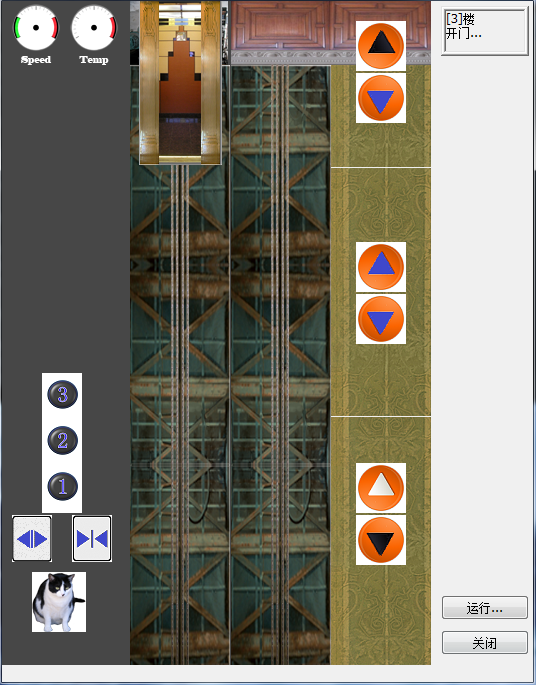
1. **电梯停于1F，按3F向下呼叫按钮，然后立即按2F向下呼叫按钮；电梯上升到3F停止，开门/关门，然后下降到2F停止，开门/关门。**



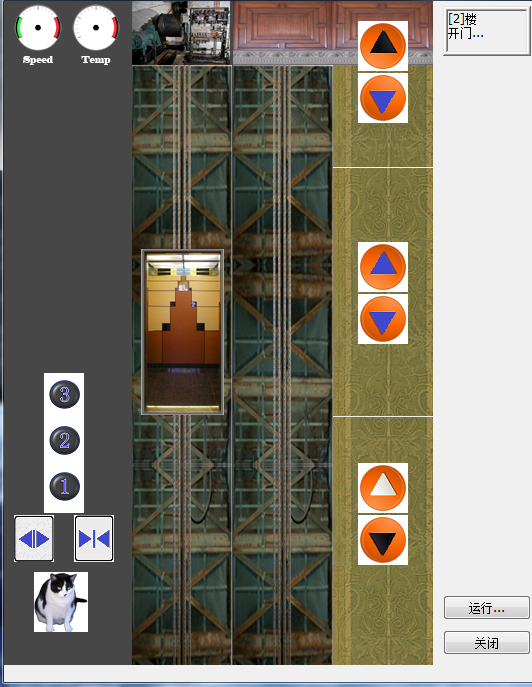
1. **电梯停于2F，按门内楼层按钮3，然后门内楼层按钮1；电梯上手到3F停止，开门/关门，然后下降到1F，开门/关门。**



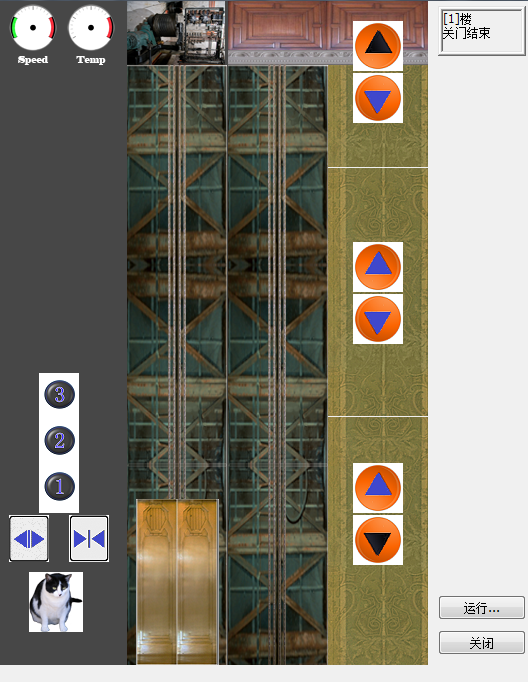
1. **电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以下，按2F向上呼叫按钮；电梯先上升到2F，开门/关门；然后再到3F，开门/关门。**



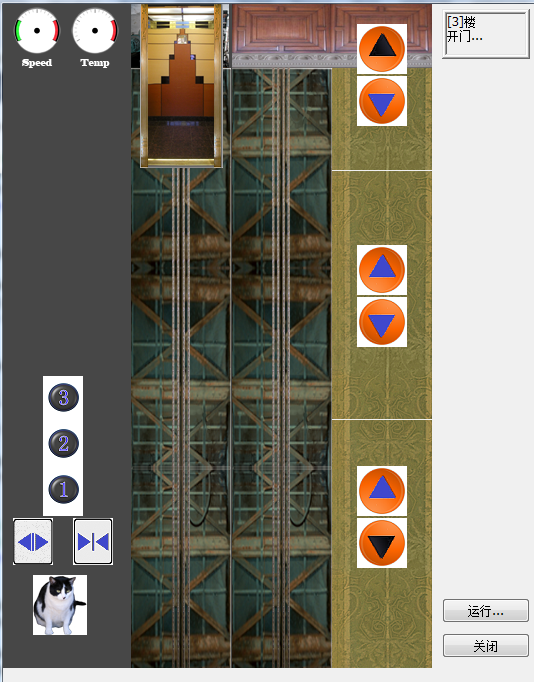
1. **电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以上，按2F向上呼叫按钮；电梯先上升到3F，开门/关门；然后再到2F，开门/关门。**



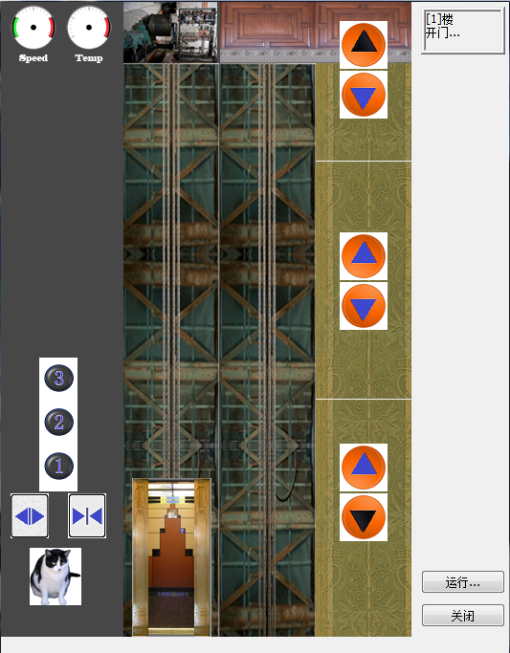
1. **电梯停于1F，按2F向下呼叫按钮和向上呼叫按钮以及3F的向下呼叫按钮；电梯上升到2F停止，开门/关门，2F的向上呼叫按钮灯关闭，开门/关门， 2F的向下呼叫按钮灯关闭，然后上升到3F停止，开门/关门，3F的向下呼叫按钮关闭。**



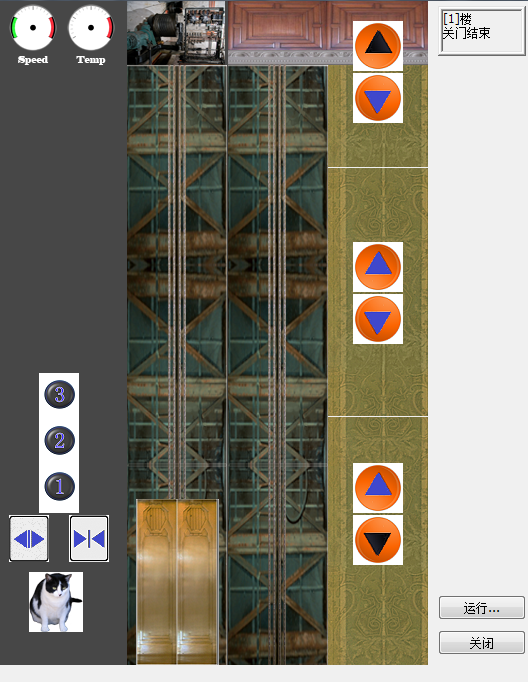
1. **电梯在2F以上，10s无动作，自动降到1楼。**



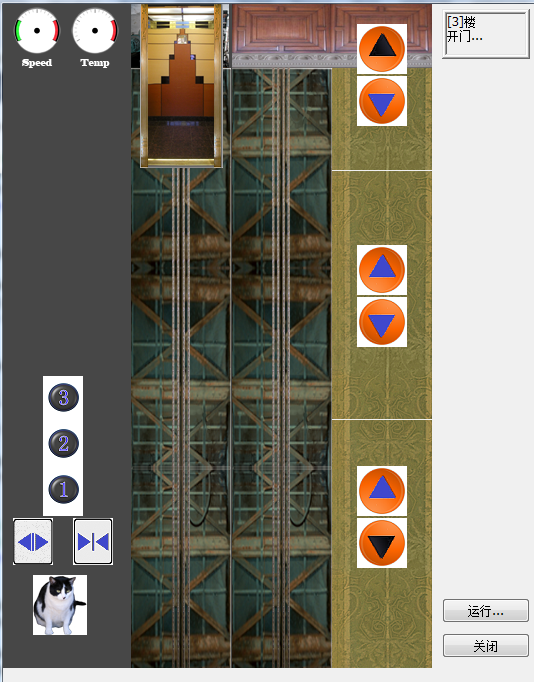
1. **所有停止，开门/关门后，对应楼层的同方向门外呼叫按钮灯（最高楼向下呼叫按钮，最底层向上呼叫按钮）和门内楼层按钮灯关闭。**

****

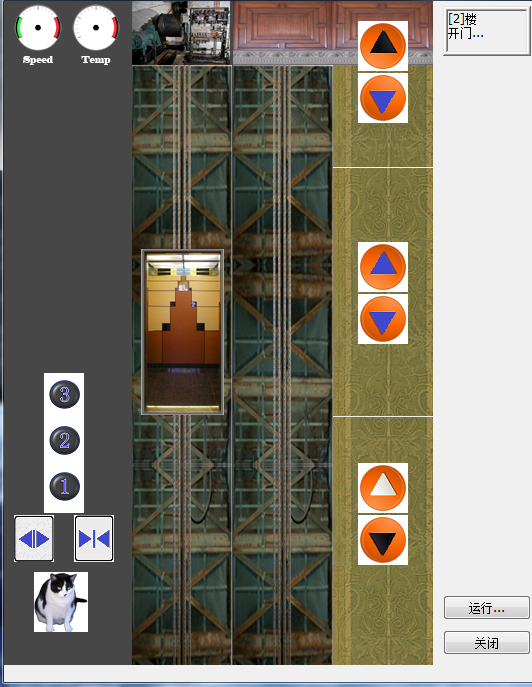
1. **运动状态，开关门按钮失效。正在开门，开门按钮失效；正在关门，关门按钮失效。**



1. **空闲状态，门是关闭的，因此按关门按钮失效。**



1. **开关门结束，延时2秒用于乘客上下电梯【延时功能在库函数中实现，不用在状态函数中实现】，然后进入关门状态。**



1. **开门结束前，按关门按钮，转而进入关门状态；关门结束前，按开门按钮，转而进入开门状态。**

