

1. 状态机图及其分解描述：

将三层电梯的运行过程分解为五个状态，分别为：空闲等待、上升、下降、开门和关门。如上图所示。

Idle状态及其事件：

S1:由空闲等待转向上升，只能出现以下的情况1.上面楼层的门外呼叫按钮被人点亮2.电梯门内的上面楼层数的按钮被点亮。

S2:由空闲等待转向下降，只能出现以下的情况 1.下面楼层的门外呼叫按钮被人点亮2.电梯门内的下面楼层数的按钮被点亮。

S3:由空闲转向开门，只能出现以下的情况 1.当前楼层的门外呼叫按钮被人点亮2.电梯门内的开门按钮被人点亮。

S4:由空闲状态下转向关门，空闲状态下转向关门，没有任何的响应，只是将门内的关门按钮消费掉。

MovingUp&MovingDown状态及其所发生的事件：

S5或S6:由上升或下降的状态转向开门的过程，只能发生以下两种情况：1.我们所要到达的目标楼层到了，电梯门会被打开。2.在电梯上升或者下降的时候经过某一楼层时，该楼层的同样是向下或者向上的方向按钮被呼叫，因此电梯会开门让该楼层的乘客进入电梯。

D:电梯在运行的过程中，如果门内的开门关门按钮被点亮，那么将不会有任何的反应，只是将门内被点亮的开门或关门按钮灭掉。

Dooropen状态及其所发生的事件：

S7:有开门转向关门的过程中，只能发生下面三种情况：1.开门状态下电梯门内的换门按钮被人点亮。2.在开门后经历一定的时间之后，电梯门自动关闭。3.开门的状态下，按动门内的开门按钮，电梯没有反应，并且灭掉开门按钮的灯。

DoorClosing状态及其所发生的事件：

S8:由关门到开门状态的转变，所对应的事件应该是在关门的过程中突然按下门内的开门键，电梯的门立刻被打开。

S9:由关门转向空闲等待的过程。关门的动作结束后，电梯没有下一步的动作，此时电梯处于空闲等待的状态。

1. 各个状态函数的设计流程：

把主要的事件记为E1.E2.E3.E4.E5.它们分别为：

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(CallLight)

E5:门外down呼叫按钮(CallLight)

Idle状态：

S1:检查E3.E4.E5事件，即检查是否有门外的呼叫按钮被点亮以及门内的楼层数按钮被点亮，确定电梯下一步所应该到达的楼层floor=IdleWhatFloorTogoTo(&up,)启动电机，使得电梯向上运动 SetMotorPower(1)。并且进入MovingUp状态。

S2：同S1一致，只是区别在于此时是下方楼层的楼层数按钮被点亮以及下方楼层的门外按钮呼叫灯被点亮确定电梯所要到达的楼层，启动电机，使电梯向下运动，并且进入MovingDown状态。

S3：检查E1事件即检查是否有门内的开门按钮被按动GetOpenDoorLight()，如果门内的开门按钮被点亮，那么将门打开SetDoor(),之后消费开门按钮即将开门按钮熄灭SetOpenDoorLight(),并且进入DoorOpen状态。

检查E4.E5事件也就是检查电梯门外是否有呼叫灯被点亮，GetCallLight(),检查门外是否有up或者down按钮被呼叫，开门SetDoor(),消费门外的up或者down按钮SetCallLight()电梯进入开门状态。

S4：由空闲状态转换至空闲状态，首先检查E2事件，也就是是否在空闲状态下电梯门内的关门键被点亮。GetCloseDoorLight()此时的电梯处于空闲状态，因此电梯门应该是关闭的，所以只读取关门灯，并消费关门灯，电梯本身并不会有任何的动作。

If(GetCloseDoorLight()){SetCloseDoorLight();return}

MovingUp状态：

S5：由上升状态转变为开门状态。检查E3.E4.E5事件，目标楼层floor=GoingUpToFloor();

如果检测到已经到达目标楼层if(fabs(GetFloor()-floor)<Lib\_FloorTolerance)，那么bool up = GetCallLight(GetNearestFloor(), 1)检测门外向上的灯是否被点亮以及bool down = GetCallLight(GetNearestFloor(), 0)检测门外向下的灯是否被点亮。之后将电梯停止，并将电梯门打开，最终将向上或向下的按钮消费掉。

D：检查E1E2事件，检查门内的开门关门按钮是否被点亮，由于在运动状态下，因此不会有任何的反应。

MovingDown状态：

S6：由下降状态转变至开门状态与S5一致，这里便不再赘述。。

D：同上所述。

DoorOpen状态：

S7：由开门状态转为关门状态。检查E2事件即检查是否在开门状态下门内的关门按钮被点亮，由开门状态转至关门状态，GetCloseDoorLight(),SetDoor();之后消费关门按钮。开门结束后，过一段时间并没有任何操作，将自动进入关门状态IsDoorOpen();SetDoor(); 检查E1事件，正在开门，但是按了开门灯，无动作，消费开门按钮。

DoorClosing状态：

S8：检查E1事件，开门。GetOpenDoorLight() SetDoor();消费开门按钮，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。

如果红外探测到遮挡，转而开门，进入开门状态。

S9关门结束后，进入等待状态。

3.具体程序如下

#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); //检查是否有门内的上面楼层数的按钮被点亮 或者是否有上面楼层的门外按钮被呼叫

if (floor > 0 && up)

{

SetMotorPower(1); //上升 打开电机 电梯向上运动 进入向上的状态

\*state = MovingUp;

return;

}

else if (floor>0&&!up) //如果检查的结果与上面的相反 都是向下楼层的门内以及门外的按钮被点亮

{

SetMotorPower(-1); //下降 打开电机 电梯向下运动 进入向下的状态

\*state = MovingDown;

return;

}

else if (GetOpenDoorLight()) //检查是否有门内的开门按钮被点亮

{

SetOpenDoorLight(0);

SetDoor(GetNearestFloor(), 1);

\*state = DoorOpen;

return;

}

else if (GetCallLight(floor, up)) //检查是否有门外的按钮被点亮 如果有 打开门 消费掉门外的灯

{

SetDoor(floor, 1);

SetCallLight(floor, up, 0);

\*state = DoorOpen;

return;

}

else if (GetCallLight(1, 1))

{

SetDoor(1, 1);

SetCallLight(1, up, 0);

\*state = DoorOpen;

return;

}

else if (GetCloseDoorLight()) //检查门内的关门按钮是否被按 如果被按动 等待空闲状态下无响应 直接将其消费

{

SetCloseDoorLight(0);

return;

}

}

void StateMovingUp(int \*state)

{

int floor = GoingUpToFloor();

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance) //判断电梯是否到达指定楼层

{

bool up = GetCallLight(GetNearestFloor(), 1);//判断向上的门外按钮是否被点亮

bool down = GetCallLight(GetNearestFloor(), 0);//判断向下的门外按钮是否被点亮

SetMotorPower(0); //关掉电机使电梯停止

if (up == 1) //如果向上的灯亮 将其消费

{

SetCallLight(GetNearestFloor(), 1, 0);

}

if (down == 1) //如果向下的灯亮 将其消费

{

SetCallLight(GetNearestFloor(), 0, 0);

}

SetPanelFloorLight(floor, 0);

SetDoor(floor,1);

\*state = DoorOpen;

}

if(GetOpenDoorLight() || GetCloseDoorLight()) //按开门或者关门键是无效的在电梯上升或下降的过程中

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

void StateMovingDown(int \*state)

{

int floor = GoingDownToFloor();

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

{

bool up = GetCallLight(GetNearestFloor(), 1);//门外的向上的灯是否被点亮

bool down = GetCallLight(GetNearestFloor(), 0);//门外向下的灯是否被点亮

SetMotorPower(0);

SetDoor(GetNearestFloor(), 1);

if (up == 1) //如果向上的灯被点亮 就将灯消费掉

{

SetCallLight(GetNearestFloor(), 1, 0);

}

if (down == 1) //如果向下的灯被点亮 就将灯消费掉

{

SetCallLight(GetNearestFloor(), 0, 0);

}

SetPanelFloorLight(GetNearestFloor(), 0);

\*state = DoorOpen;

}

else if (GetOpenDoorLight() || GetCloseDoorLight())//电梯在运行的过程中按门外的灯是无效的

{

SetOpenDoorLight(0);

SetCloseDoorLight(0);

}

}

void StateDoorOpen(int \*state)

{

if (GetCloseDoorLight()) //检测门内的关门按钮是否被点亮

{

SetDoor(GetNearestFloor(), 0); //转而关门

SetCloseDoorLight(0); //消费门内的关门灯

\*state = DoorClosing;

}

else if (IsDoorOpen(GetNearestFloor())) //开门结束后自动进行关门

{

SetDoor(GetNearestFloor(),0);

\*state = DoorClosing;

}

else if (GetOpenDoorLight()) //检测是否有开门灯被点亮 若点亮 将其消费

SetOpenDoorLight(0);

}

void StateDoorClosing(int \*state)

{

if (GetOpenDoorLight()) //检查是否门内的开门按钮被点亮

{

SetDoor(GetNearestFloor(), 0); //转而开门

SetOpenDoorLight(0); //消费按钮

\*state = DoorOpen;

}

else if (GetCloseDoorLight()) //检查门内的关门按钮是否被点亮 如果被点亮直接将其消费

{

SetCloseDoorLight(0);

}

else if (IsBeamBroken()) //检查是否有遮蔽物 如果红外检测到了遮蔽物

{

SetDoor(GetNearestFloor(), 1); //将门打开

\*state = DoorOpen;

}

else if (IsDoorClosed(GetNearestFloor())) //关门结束后 进入待机状态

{

//SetOpenDoorLight(0);

\*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");

}

}

}