## 三层电梯状态机课程设计报告

张俊杰 16040310084

Idle

S1：由idle到moving up。事件：非当前楼层门外calllight被呼叫，或者门内向上的楼层数被点亮。

代码：if (floor > 0 && up)

{

SetMotorPower(1);

\*state = MovingUp;

}

S2：由idle到moving down。事件：非当前楼层门外calllight被呼叫，或者门内向下的楼层数被点亮。

代码：if (floor > 0 && !up)

{

SetMotorPower(-1);

\*state = MovingDown;

}

S3：由idle到door open。事件：当前楼层门外呼叫或者门内开门按钮点亮。

代码：if (GetOpenDoorLight())//事件

{

SetOpenDoorLight(false);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

if (GetCallLight(GetNearestFloor() ,false))

{

SetCallLight(GetNearestFloor(), true, false);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

S4：idle到idle。关门，无动作，灭掉门内该按键。

代码：if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

\*state = Idle;

}

MovingUp/Down

S5/S6：moving up/moving down到door open。事件：目的楼层到了，自然开门；若up/down的过程中途楼层同方向被呼叫，则在中途楼层停车，开门上客。

代码（Up）:if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)

{

SetMotorPower(0);//变迁

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), true, false);

SetCallLight(Lib\_FloorNum, false, false);

SetPanelFloorLight(GetNearestFloor(), false);

\*state = DoorOpen;

}

代码（Down）:if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)

{

SetMotorPower(0);

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), false, false);

SetCallLight(Lib\_FloorNum, true, false);

SetPanelFloorLight(GetNearestFloor(), false);

\*state = DoorOpen;

}

D：moving up/moving down中不允许开关门。事件：按了开关门按键，无动作，同时灭掉按键。

代码：GetOpenDoorLight();

GetCloseDoorLight();

SetOpenDoorLight(false);

SetCloseDoorLight(false);

DoorOpen

S7：door open到doorclosing。事件：门内乘客手动关门；开门动作结束后，门自动关闭；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。

代码：if (GetCloseDoorLight())

{

SetDoor(GetNearestFloor(), false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

if (IsDoorOpen(GetNearestFloor()))

{

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

}

DoorClosing

S8：door closing到door open。事件：门内开门按键被点亮，电梯开门。

代码：if (GetOpenDoorLight())

{

SetDoor(GetNearestFloor(), true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

if (IsBeamBroken())

{

SetDoor(GetNearestFloor(), true);

\*state = DoorOpen;

}

S9：door closing到idle。关门结束后，电梯没有下一步动作，处于空闲的状态。

if (IsDoorClosed(GetNearestFloor()))

{

\*state = Idle;

}

附：

void StateIdle(int \*state)

{

int floor; bool up;

floor = IdleWhatFloorToGoTo(&up);

if (floor > 0 && up)

{

SetMotorPower(1);//变迁

\*state = MovingUp;

printf("Idle到MovingUp.\n");

}

else if (floor > 0 && !up)//事件

{

SetMotorPower(-1);//变迁

\*state = MovingDown;

printf("Idle到MovingDown.\n");

}

else if (GetOpenDoorLight())//事件

{

SetOpenDoorLight(false);//消费

SetDoor(GetNearestFloor(), true);//变迁

\*state = DoorOpen;

printf("Idle到DoorOpen.\n");

}

else if (GetCallLight(GetNearestFloor() ,false))//事件

{

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

SetDoor(GetNearestFloor(), true);//变迁

\*state = DoorOpen;

printf("Idle到DoorOpen.\n");

}

if (GetCloseDoorLight())//事件

{

SetCloseDoorLight(false); //消费（不变）

\*state = Idle;

return;

}

}

void StateMovingUp(int \*state)

{

int floor;

floor = GoingDownToFloor();//事件

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

{

SetMotorPower(0);//变迁

SetDoor(GetNearestFloor(), true);//变迁

\*state = DoorOpen;

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

SetCallLight(Lib\_FloorNum, false, false);//消费

SetPanelFloorLight(GetNearestFloor(), false);//消费

\*state = DoorOpen;

}

else GetOpenDoorLight();//检测

GetCloseDoorLight();//检测

SetOpenDoorLight(false);//消费

SetCloseDoorLight(false);//消费

}

void StateMovingDown(int \*state)

{

int floor;

floor = GoingDownToFloor();

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

{

SetMotorPower(0);//变迁

SetDoor(GetNearestFloor(), true);//变迁

\*state = DoorOpen;

SetCallLight(GetNearestFloor(), false, false);//消费

SetCallLight(Lib\_FloorNum, true, false);//消费

SetPanelFloorLight(GetNearestFloor(), false);//消费

\*state = DoorOpen;

}

else GetOpenDoorLight();//检测

GetCloseDoorLight();//检测

SetOpenDoorLight(false);//消费

SetCloseDoorLight(false);//消费

}

void StateDoorOpen(int \*state)

{

GetNearestFloor();

if (GetCloseDoorLight())//事件

{

SetDoor(GetNearestFloor(), false);//变迁

SetCloseDoorLight(false);//消费

\*state = DoorClosing;

}

if (IsDoorOpen(GetNearestFloor()))//事件

{

SetDoor(GetNearestFloor(), false);//变迁

\*state = DoorClosing;

}

if (GetOpenDoorLight())//事件

{

SetOpenDoorLight(false);//消费

}

}

void StateDoorClosing(int \*state)

{

if (GetOpenDoorLight())//事件

{

SetDoor(GetNearestFloor(), true);//变迁

SetOpenDoorLight(false);//消费

\*state = DoorOpen;

}

if (GetCloseDoorLight())//事件

{

SetCloseDoorLight(false); //变迁

}

if (IsBeamBroken())//事件

{

SetDoor(GetNearestFloor(), true);//变迁

\*state = DoorOpen;

}

if (IsDoorClosed(GetNearestFloor()))

{

\*state = Idle;

}

}