

《C语言程序设计》

课程设计报告

姓 名： 刘 思 凯\_\_\_\_\_\_\_

学 号： \_\_\_\_ 16040310064\_\_\_

院 系： 机电工程学院

专 业： 自动化

班 级： \_\_ \_\_1604031 \_\_\_\_\_\_\_

指导教师： \_\_\_ \_段江涛

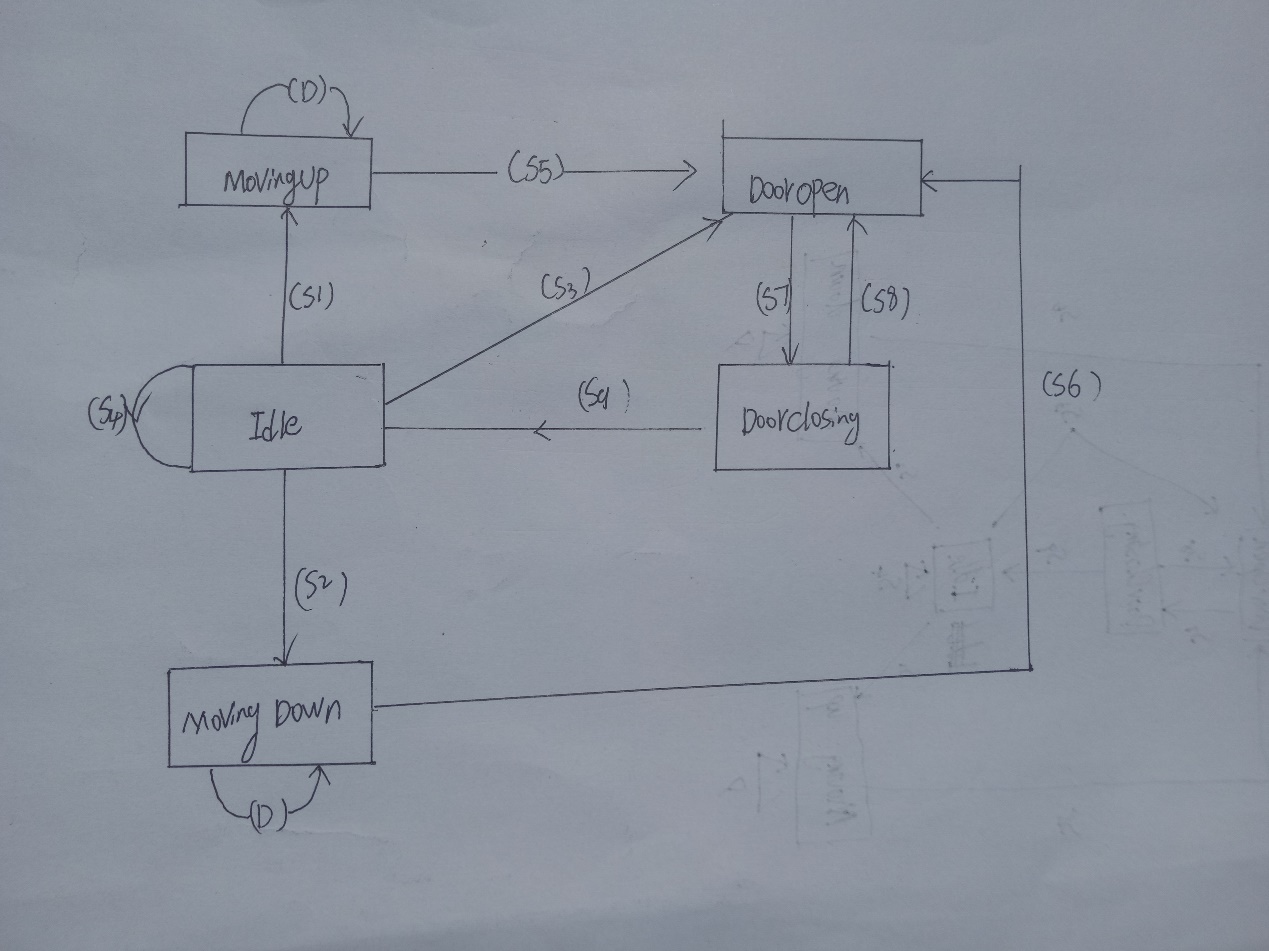
机电工程学院

2017年5月

## 三层电梯课程设计报告

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。



**Idle状态及其事件**

S1：由闲置转向向上升的过程，对应的事件是非当前楼层门外calllight被呼叫，或者门内向上的楼层数被点亮。

S2：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。

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开门按钮点亮。

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，则在中途楼层停车，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**

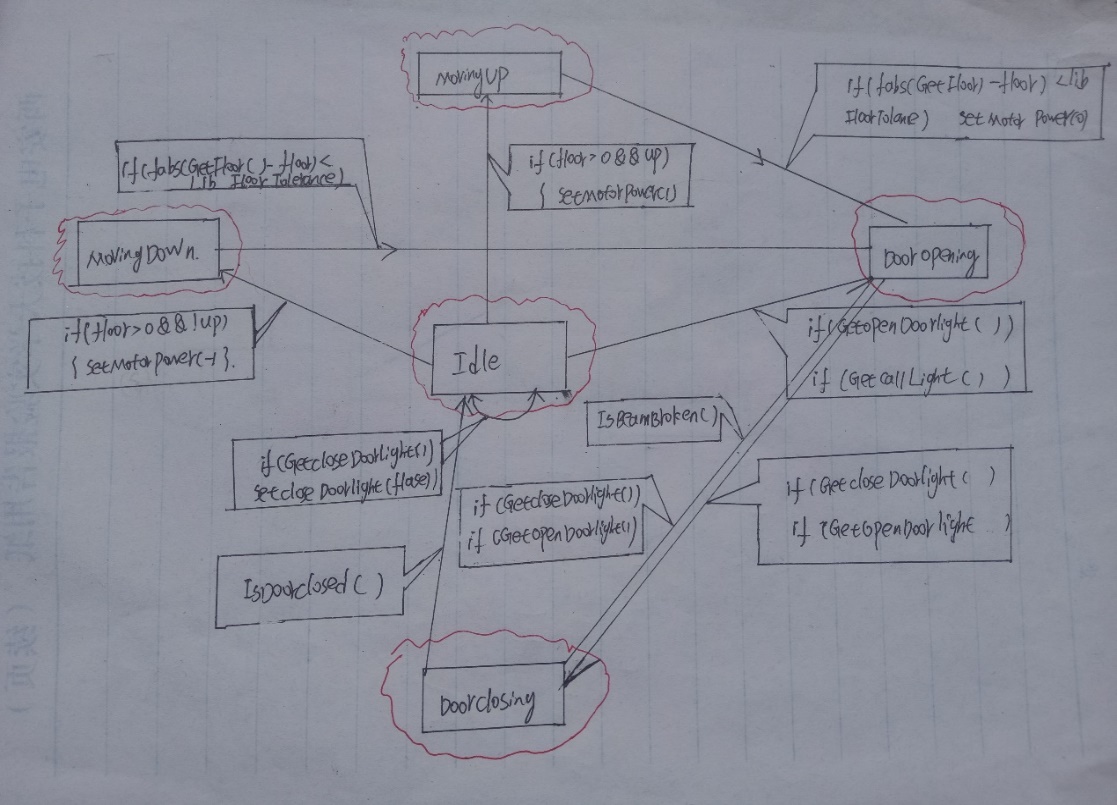
S7：由开门到关门的过程。对应三个事件，其一是门内乘客手动关门；其二是开门动作结束后，门自动关闭；其三是在开门状态下按门内开门按钮电梯无反应，只是灭掉开门按钮的灯。

**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。

1. **用流程图描述各个状态函数**

****

1. **各状态函数设计过程**

把主要事件记为E1~E5。具体规定如下：

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)

**Idle状态：**

(S1) 检查E3、E4、E5事件。（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。）

属于静态检测。bool up; floor=IdleWhatFloorToGoTo(&up);

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。

(S2) 同(S1)，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()]

(S3) 检查E1事件, 开门, 消费开门按钮;

上升 (up && E4事件), 开门，消费门外up按钮

下降 (!up && E5事件), 开门，消费门外down按钮

(S4) 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，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; }

**运动中状态：**

MovingUp/MovingDown 🡪 DoorOpen

(S5) 检查E3、E4、E5事件。

动态检测，目标楼层floor=GoingUpToFloor();

if(fabs(GetFloor() - floor) < Lib\_FloorTolerance) 到达目标楼层，停止，开门，消费门外按钮。

(D) 检查E1、E2事件，无动作，消费开/关门按钮。

(S6) 检查E3、E4、E5事件。动态检测，目标楼层=GoingDownToFloor();其它与(S5)类似。

**开门状态:** 电梯门打开 🡪 DoorClosing

(S7) 检查E2事件，转而关门，GetCloseDoorLight(),SetDoor(); 消费关门按钮。

开门结束后，自动进入关门状态。IsDoorOpen();SetDoor();

检查E1事件, 无动作，消费开门按钮。

**关门状态:**

(S8) 检查E1事件，转而开门。GetOpenDoorLight();SetDoor(); 消费关门按钮。进入DoorOpen状态

检查E2事件，无动作，消费关门按钮。

(S9) 关门结束后，进入Idle状态。IsDoorClosed();

**附录：**

源程序代码

#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; //进入movingup状态

}

if (floor > 0 && !up)

{

SetMotorPower(-1);

\*state = MovingDown; //进入movingdown状态

}

if (GetOpenDoorLight())

{

SetDoor(floor, true);

SetOpenDoorLight(false);

\*state = DoorOpen; //进入dooropen状态

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

\*state = Idle;

return;

}

}

void StateMovingUp(int \*state)

{

int floor;

floor= GoingUpToFloor();

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())

{

SetOpenDoorLight(false);

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

}

void StateMovingDown(int \*state)

{

int floor;

floor = GoingDownToFloor();

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

{

SetMotorPower(0);

SetDoor(floor, true);

SetCallLight(floor, true, false);

SetCallLight(floor, false, false);

GetPanelFloorLight(GetNearestFloor());

\*state = DoorOpen;

}

if (GetOpenDoorLight());

{

SetOpenDoorLight(false);

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

}

}

void StateDoorOpen(int \*state)

{

int floor = GetFloor();

if (GetCloseDoorLight())

{

SetDoor(floor, false);

SetCloseDoorLight(false);

\*state = DoorClosing;

}

else if (IsDoorOpen(GetNearestFloor()))

{

SetDoor(GetNearestFloor(), false);

\*state = DoorClosing;

}

if (GetOpenDoorLight());

{

SetOpenDoorLight(false);

}

}

void StateDoorClosing(int \*state)

{

int floor = GetFloor();

if (GetOpenDoorLight())

{

SetDoor(floor, 1);

SetOpenDoorLight(0);

\*state = DoorOpen;

}

if (GetCloseDoorLight())//关门状态下按关门无反应

{

SetCloseDoorLight(0);

}

if (IsBeamBroken())

{

SetDoor(floor,true);

\*state = DoorOpen;

}

if (IsDoorClosed(floor))//关门结束，进入空闲状态

\*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");

}

}

}