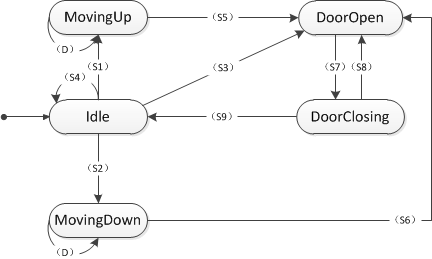
三层电梯状态机课程设计报告

谢文文

16040310038



１.状态之间转换的本质：

利用主控程序“状态机”实现状态到状态转换（即变迁），本质上状态机在我看来是一个比较有条理的“循环体”，无限循环。由状态机的状态数目决定循环体的子程序的个数。

２．状态机的运行：

状态机的运行需要外部条件的触发，即需要对外部的指令做出相应的反应。等待条件的输入，由状态机的内部对该条件做出相应的判断，运用到一些类似物理传感器相关的知识。不同的状态之间有的可以相互转换，而有的只能进行单向传递。

3 .状态到状态；条件的判断

ａ:Idle MovingUp/Down;状态机内部需要获取的条件（floor=IdleWhatFloorToGoTo()）,由此来检测楼层的高度，来运行MotorPower;上升还是下降。

b :Idle DoorOpen;此状态的变换可以由两个条件来执行。一种是门内开门按钮OpenDoorNight(),另一种是门外按钮CallNight(),由此进入DoorOpen状态。

c :Idle Idle;此时状态机读入的条件是关门灯按钮CloseDoorNight()，并关闭关门灯。

d :MovingUp to DoorOpen;floor=GoingUpToFloor()，条件检测if(fabs(GetFloor()-floor)<Lib\_FloorTolerance);若符合条件，电机停止工作，进入DoorOpen状态。应当注意的是当电机仍然处于工作状态时，此时开关门无效。GetOpenDoorLight()/GetCloseDoorLight()，无动作，消费开关门按钮，SetOpenDoorLight()/SetCloseDoorLight()。

e :MovingDown to DoorOpen;floor=GoingDownToFloor()，条件检测if(fabs(GetFloor()-floor)<Lib\_FloorTolerance);若符合条件，电机停止工作，进入DoorOpen状态。应当注意的是当电机仍然处于工作状态时，此时开关门无效。GetOpenDoorLight()/GetCloseDoorLight()，无动作，消费开关门按钮，SetOpenDoorLight()/SetCloseDoorLight()。两个状态都应该记得消费按钮。

g :DoorOpen to DoorClosing;第一种情况是GetCloseDoorNight(),进入正在关门状态；第二种是开门结束后IsDoorOpen(),进入关门SetDoor()。

h :DoorClosing to DoorOpen;一是GetOpenDoorNight(),SetDoor();消费开门按钮SetOpenDoorNight();进入DoorOpen状态，二是红外探测到遮挡IsBeamBroken(),转而开门SetDoor(),进入DoorOpen状态。

i :DoorClosing to Idle;关门结束后判断条件语句IsDoorClosed(),进而进入Idle状态。

状态机的难点和注意点

1. 状态机的内部工作原理的理解，学习状态机编程的过程中，和学习51单片机的编程有些类似，源文件的调用和主控程序的理解，找到一条程序主线，在这里就是五个循环之间的转换，形成一个循环体。
2. 状态机程序编写中条件语句的编写，如何对相应事件的输入运用条件实现变迁。

流程图

DoorOpen

按钮开关

Idle

Moving

Down

MovingUp

无指令

上升

下降

等待条件的输入

IDLE

功能的实现

电梯内部程序运行时可以自动控制升降在无指令时处于调停状态。

电梯可以根据指令的输入做出相应的反应根据条件实现变迁；开始循环，直至循环结束。

电梯内部的传感器检测系统和发射系统会在极短周期内检测外部条件的变化，从而实现半自动化控制。

m#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); //检测将要到达的楼层。

if (floor > 0 && up) //楼层高于当前楼层，上升。

{

SetMotorPower(1);

\*state = MovingUp; // 状态机由Idle状态变为MovingUp。

}

if (floor>0 && !up)

{

SetMotorPower(-1);

\*state = MovingDown; // 状态机由Idle状态变为MovingDown。

}

if (GetOpenDoorLight()) //等待开门按钮的指令。

{

SetOpenDoorLight(false);

SetDoor;

\*state = DoorOpen;

}

if (GetCallLight(floor = GetNearestFloor(), true) || GetCallLight(floor = GetNearestFloor(), false))

{

SetCallLight(floor = GetNearestFloor(), true, false);

SetCallLight(floor = GetNearestFloor(), false, false);

SetDoor;

\*state = DoorOpen; //电梯门外按钮作用下进入DoorOpen状态

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

\*state = Idle; //门内关门按钮指令进入Idle状态

}

}

void StateMovingUp(int \*state)

{

int floor;

GetFloor();

floor = GoingUpToFloor();

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

{

SetMotorPower(0); SetDoor(floor, true);

\*state = DoorOpen;

SetPanelFloorLight(floor, false);

}

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(false);

SetCloseDoorLight(false); //消费按钮。防止下一周期继续该按钮

\*state = DoorOpen;

}

}

void StateMovingDown(int \*state)

{

int floor; bool up;

GetFloor();

floor = GoingUpToFloor();

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

{

SetMotorPower(0); SetDoor();

\*state = DoorOpen;

SetPanelFloorLight(floor,false); //消费当前门内楼层按钮

}

if (GetOpenDoorLight() || GetCloseDoorLight())

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

\*state = DoorOpen;

}

}

}

void StateDoorOpen(int \*state)

{

if (GetCloseDoorLight())

{

SetDoor(floor,false);

SetCloseDoorLight(false);

\*state = DoorOpen; //由不受外部干扰的开门状态

}

if (IsDoorOpen())

{

SetDoor(floor,true);

\*state = DoorOpen;

}

if (GetOpenDoorLight())

{

SetOpenDoorLight(false);

\*state = DoorOpen;

}

}

}

void StateDoorClosing(int \*state)

{

int Floor; Floor = GetNearestFloor();

if (GetOpenDoorLight() || IsBeamBroken()) //检测电梯门外是否有障碍物

{

SetDoor(floor,true);

SetOpenDoorLight(false);

\*state = DoorOpen;

}

if (GetCloseDoorLight())

{

SetCloseDoorLight(false);

\*state = DoorOpen;

}

if (IsDoorClosed())

{

\*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");

}

}

}