# 处理机管理项目——电梯调度

**Teacher:** 张惠娟老师　　**Author**: 1854062 许之博

# 目录



# 1 背景

## 1.1项目简介

随着国家基建事业的蓬勃发展，日常生活中接触的是越来越高的建筑物，在高层建筑中，电梯是不可或缺的上下行工具。在校园中，出入图书馆，大多都要乘坐电梯，在等待电梯时，很难不对电梯的内部运行规律和多部电梯的协同运作产生思考。结合本学期操作系统进程管理部分内容，在此对多部电梯的运行规律进行模拟。

## 1.2 项目目的

1. 学习调度算法
2. 通过控制电梯调度，实现操作系统调度过程
3. 学习特定环境下多线程编程方法
4. 实现由5部电梯，20层楼组成的电梯系统的调度模拟

## 1.3 开发平台

1. 操作系统：Windows 10
2. 开发软件：PyCharm community 2020.3.3
3. 主要调用模块：python3.8，PyQt5 5.15.4, QtDesigner5.9.7, threading

## 1.4 功能描述

1. 电梯外部控制：五部电梯是相互联结的，在电梯外部进行统一控制。每层楼的电梯口都有一对上行下行按键，当按下某一层按下某一个按键时，相当于对所有的电梯进行一起控制。根据电梯调度算法选择最合适的电梯前往执行任务。
2. 电梯内部控制：

   数字按键：每部电梯内部都有1-20，20个数字按键，对应1-20共20个楼层，根据用户所触发的数字按键，电梯前往对应楼层。

   开关门按键：每部电梯内部都有一个开门按键和一个关门按键，实现开关门操作。电梯运行时不可触发开关门操作，电梯静止时可以开关门。

   电梯停用按键：每部电梯都有一个停用按键，当点亮该按键后，该电梯停用。
3. 运行状态显示：

   电梯运行状态：可以通过数码显示管查看电梯的所在楼层，也可以通过界面右侧的空间状态图直观查看五部电梯的运行位置。通过数码管右侧的上下行状态界面查看电梯是在上行还是下行还是静止状态。

   开关门状态：实现了电梯开关门的动画显示。

## 1.5 项目结构图

```
│  ElevatorUI.py
│  main.py
│  scheduling.py
│
└─Resources
    close.png
    close_hover.png
    close_presssed.png
    doordown.png
    doordown_pressed .png
    doorup.png
    doorup_pressed.png
    down.png
    down_hover.png
    down_pressed.png
    elevator.png
    Icon.jpg
    Icon.png
    open.png
```

```
│  open_hover.png
│  open_pressed.png
│  people.png
│  state.png
│  state_down.png
│  state_up.png
│  up.png
│  up_hover.png
│  up_pressed.png
│
└─number
```

number文件夹内包括20个数字的按键。

# 2 系统设计

## 2.1 UI设计

采用QtDesigner制作出初始UI界面，而后导出python文件，代码文件中自动生成UI_MainWindow类。

```
class Ui_MainWindow(object):

    def __init__(self):
        self.schedule = MyScheduling(self)  # 与调度文件连接
    #UI界面生成
    def setupUi(self, MainWindow):

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))

        for i in range(0, len(self.label)):
            self.label[i].setText(_translate("MainWindow", "ele" + str(i)))
            self.label2[i].setText(_translate("MainWindow", str(i)))
            self.warnbtn[i].setText(_translate("MainWindow", "alarm"))
    # 连接电梯禁用监听
    def connectStopListener(self):

    # 连接开关门控制监听
    def connectDoorListener(self):

    # 连接数字按键监听
    def connectNumListener(self):

    # 连接外部方向控制监听
    def connectDirListener(self):
```

## 2.2 电梯调度算法

### 2.2.1 单部电梯调度

1. **最短寻找楼层时间优先算法（SSTF）：**

最短寻找楼层时间优先（SSTF-Shortest Seek Time First）算法，它注重电梯寻找楼层的优化。 最短寻找楼层时间优先算法选择下一个服务对象的原则是最短寻找楼层的时间。 这样请求队列中距当前能够最先到达的楼层的请求信号就是下一个服务对象。 在重载荷的情况下，最短寻找楼层时间优先算法的平均响应时间较短，但响应时间的方差较大，原因是队列中 的某 些请求可能长时间得不到响应，出现所谓的“饿死”现象。

2. **扫描算法（SCAN）：**

扫描算法（SCAN） 是一种按照楼层顺序依次服务请求，它让电梯在最底层和最顶层之间连续往返运行，在运行过程 中响应处在于电梯运行方向相同的各楼层上的请求。 它进行寻找楼层的优化，效率比较高，但它是一个非实时算法。扫描算法较好地解决了电梯移动的问题，在这个算法 中，每个电梯响应乘客请求使乘客获得服务的次序是由其发出请求的乘客的位置与当前电梯位置之间的距离来决定 的。 所有的与电梯运行方向相同的乘客的请求在一次电向上运行或向下运行的过程中完成，免去了电梯频繁的来回移动。 扫描算法的平均响应时间比最短寻找楼层时间优先算法长，但是响应时间方差比最短寻找楼层时间优先算法小，从统 计学角度来讲，扫描算法要比最短寻找楼层时间优先算法稳定。

3. **LOOK算法:**

LOOK算法是对扫描算法的一种改进算法，对LOOK算法而言，电梯同样在最底层和最顶层之间运行。 但当 LOOK 算法发现电梯所移动的方向上不再有请求时立即改变运行方向，而扫描算法则需要移动到最底层或者最 顶层时才改变运行方向。 比较多种算法，要避免“饿死”现象，且最大限度地缩短平均响应时间，提高稳定性，考虑采用LOOK算法

### 2.2.2 多部电梯调度

对于每部电梯，为其设计一对上下行序列数组，其中上行序列数组正序排列，下行序列数组倒序排列，电梯根据目前 状态以及两数组中内容执行动作。 当某楼层有上行或下行请求发出时，先考虑每部电梯在完成当前序列的过程中能否响应该请求，计算出符合此条件的 电梯的响应距离，再考虑剩余电梯从其当前位置到序列终点与终点到该请求位置的响应距离之和，最后比较每部电梯 的响应距离，将该请求分配给具有最短响应距离一部电梯。

## 2.3 调度设计

设计了一个MyScheduling类，与UI建立连接后，主要管理对各种控制按钮的监听以及电梯的调度选择。

```
class MyScheduling(object):
    def __init__(self, UI):
        # 与界面文件建立连接
        self.elevators = UI

        # 500ms中更新一次电梯状态
        self.timer = QTimer()
        self.timer.timeout.connect(self.updateElevState)
        self.timer.start(500)

        self.messQueue = [[] for i in range(0,5)]# 电梯内部消息列表
        self.messQueue_reverse = [[] for i in range(0,5)]

    # 开关按键监听
    def doorListen(self, whichelev, whichcommand):
    # 电梯内部的数字按键监听
    def insideNumListen(self, whichelev, dest):
```

```
    # 外部方向键监听
    def outsideDirListen(self, whichfloor, choice):
    # 更新电梯状态
    def updateElevState(self):
    # 禁用监听
    def stopUsingListen(self, whichelev):
    #门控动画
    def doorAnim(self,whichelev, doorState):
    #人物动画
    def pepoleAnim(self,whichelev,doorState):
    #使用threading将门置于顶层
    def setDoorTop(self, whichelev):
    #使用threading将人物至于顶层
    def setFigureTop(self, whichelev):
```

# 3 具体实现

## 3.1 UI主要组件(核心代码)

### 3.1.2 按键

1. 数字按键
2. 方向按键

```
self.up_btn[i] = QtWidgets.QPushButton(MainWindow)
self.up_btn[i].setGeometry(QtCore.QRect(1120, 810 - i * 40, 35, 35))
self.up_btn[i].setStyleSheet(self.upbtn_style)
self.up_btn[i].setObjectName("upbtn"+str(i))
self.up_btn[i].clicked.connect(MainWindow.connectDirListener)
```

3. 停用按键

```
self.warnbtn.append(QtWidgets.QPushButton(self.centralwidget))
self.warnbtn[i].setGeometry(QtCore.QRect(warnbtn_pos[i] + 10, 60, 56, 31))
self.warnbtn[i].setStyleSheet("background-color: rgb(180, 0, 0);")
self.warnbtn[i].setObjectName("warnbtn" + str(i))
```

4. 开关门按键

### 3.1.2 静态显示

1. 电梯间边界
2. 楼层间边界
3. 楼层数字

### 3.1.3 动态显示

1. 数码管
2. 开关门动画

```
self.elevators.elevator_Anim[2*whichelev].setDirection(QAbstractAnimation.Forward)
self.elevators.elevator_Anim[2
*whichelev+1].setDirection(QAbstractAnimation.Forward)
```

3. 人物动画

```
self.elevators.figure_Anim[whichelev].setDirection(QAbstractAnimation.Forward)
self.elevators.figure_Anim[whichelev].start()
s = threading.Timer(1, self.setDoorTop, (whichelev,))
# 1秒之后把电梯门至于顶层
s.start()
```

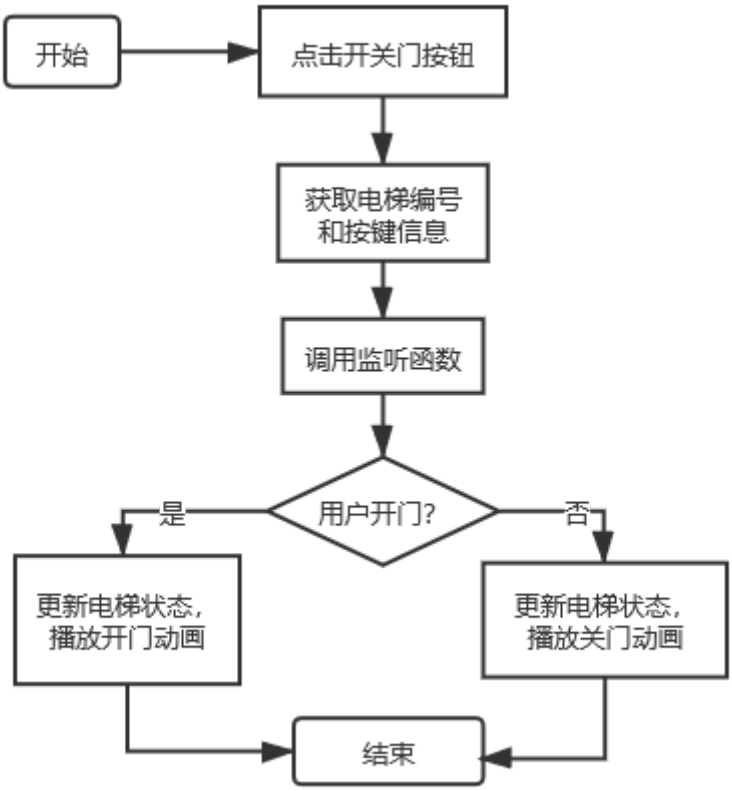
4. 电梯上下行动画

## 3.2 调度类主要组件

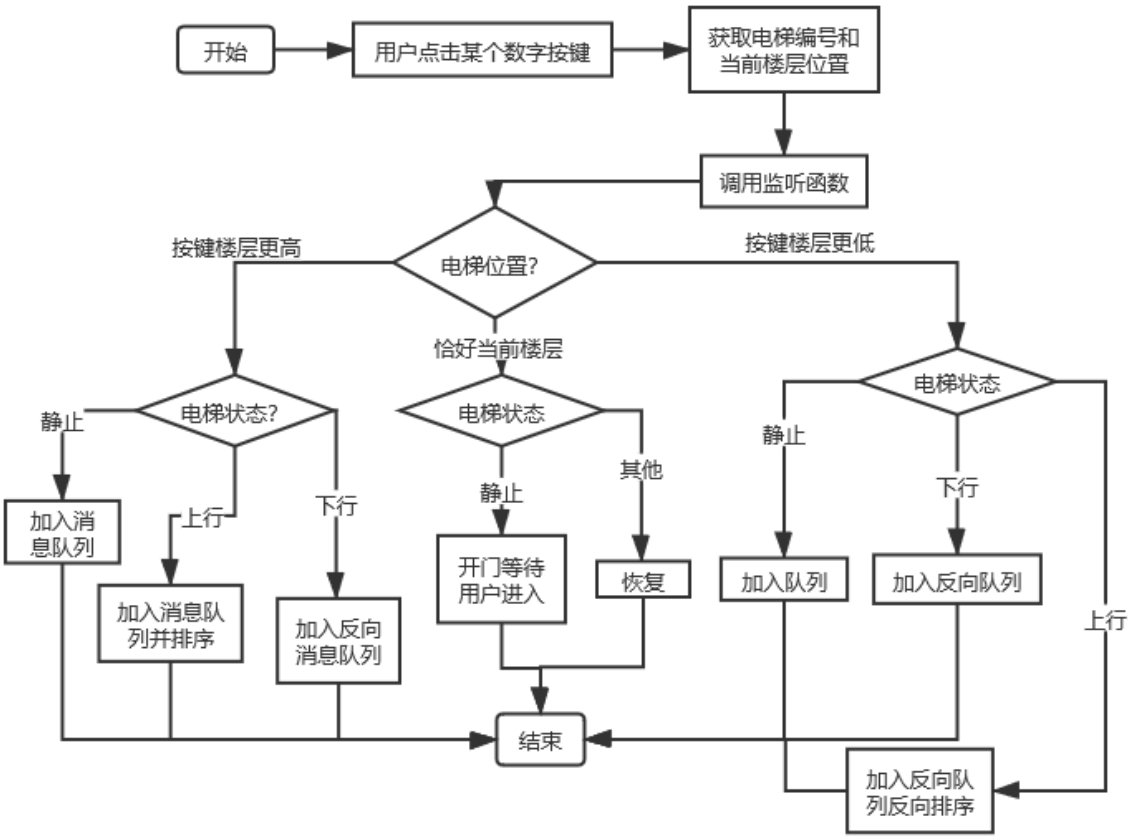
### 3.2.1 开关门按键监听

在电梯运行时，用户不可点击开关门按键。

电梯静止时，用户可以点击开关门按键，而后系统调用开关门按键监听，更新电梯状态，播放开关门动画。



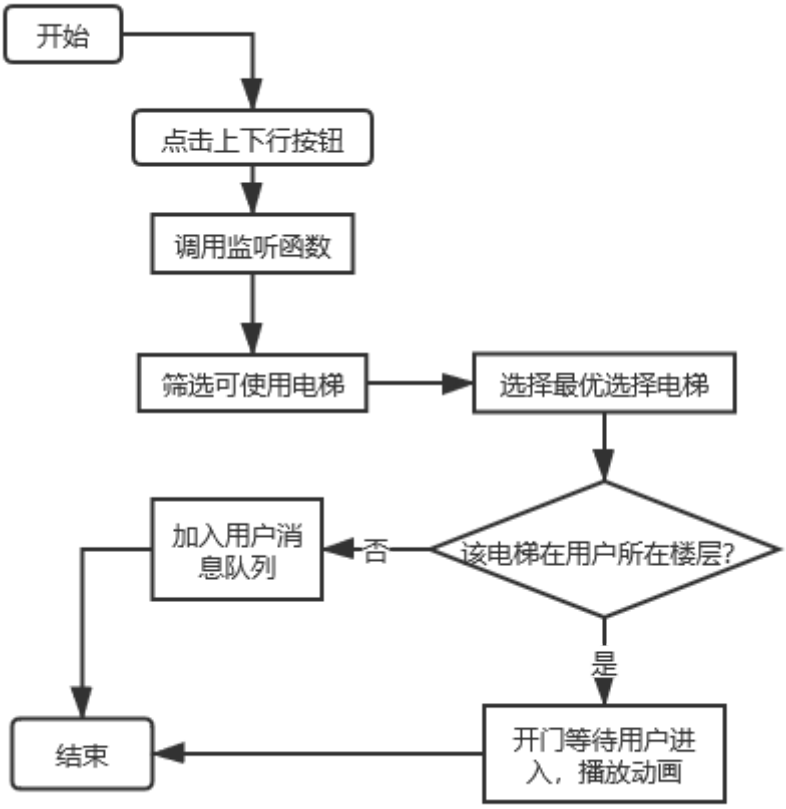
### 3.2.2 数字按键监听




### 3.2.3 外部方向键监听

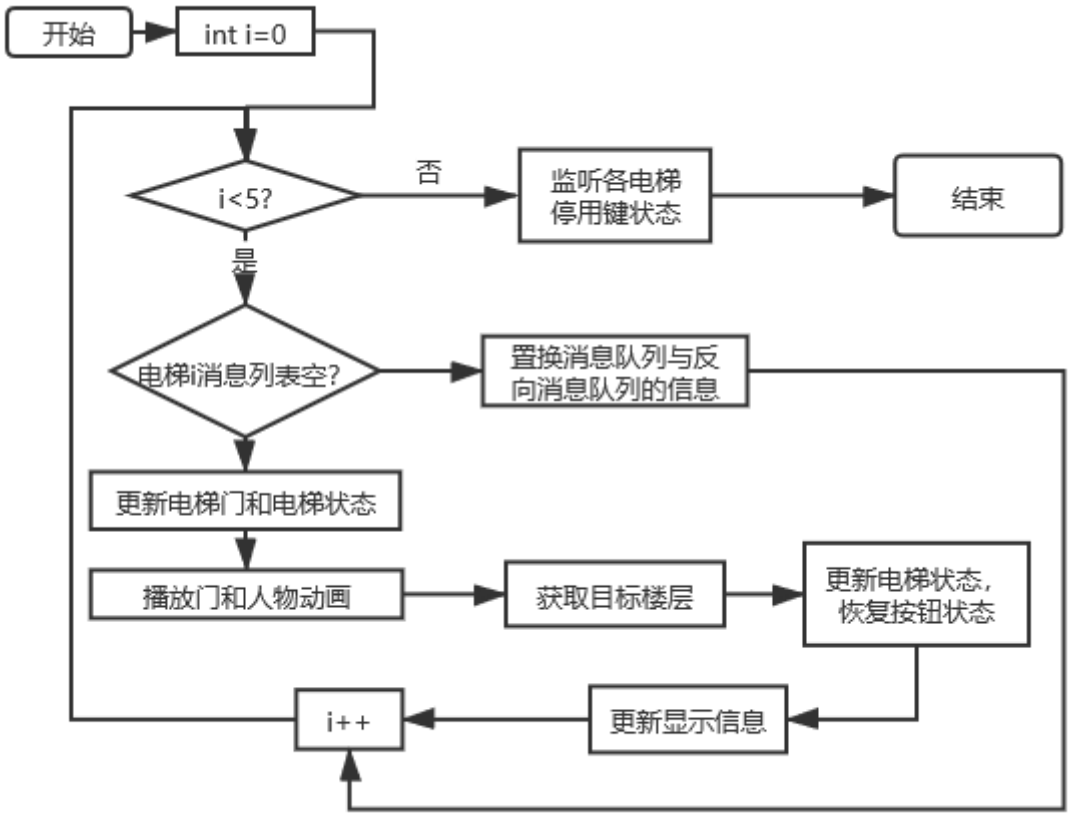
选取最佳电梯的调度依据：

1. 请求目标楼层正处于电梯的运动方向上，离该目标越近的电梯的优先级越高
2. 电梯静止，电梯离请求目标楼层越近，优先级越高



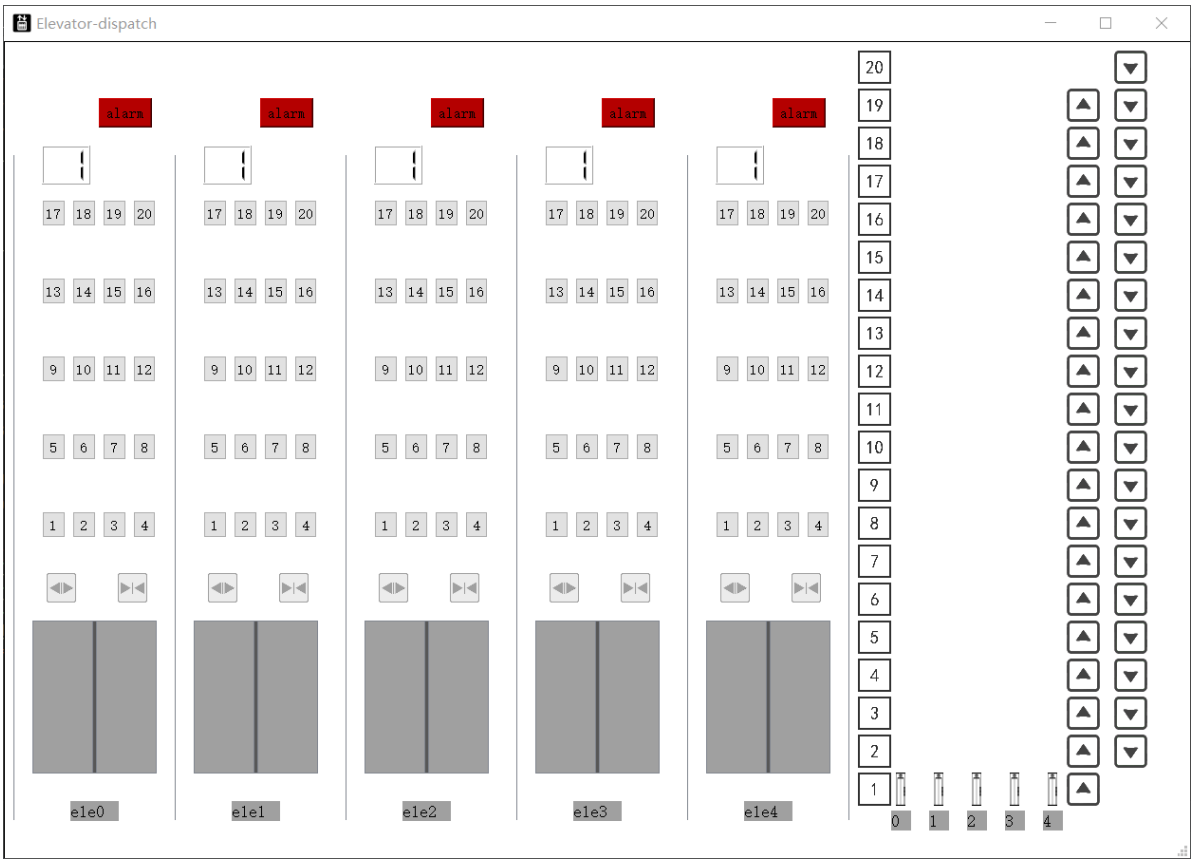

### 3.2.4 更新电梯状态

设定定时器QTimer, 每500ms更新一次




# 4 功能展示

## 4.1 操作说明

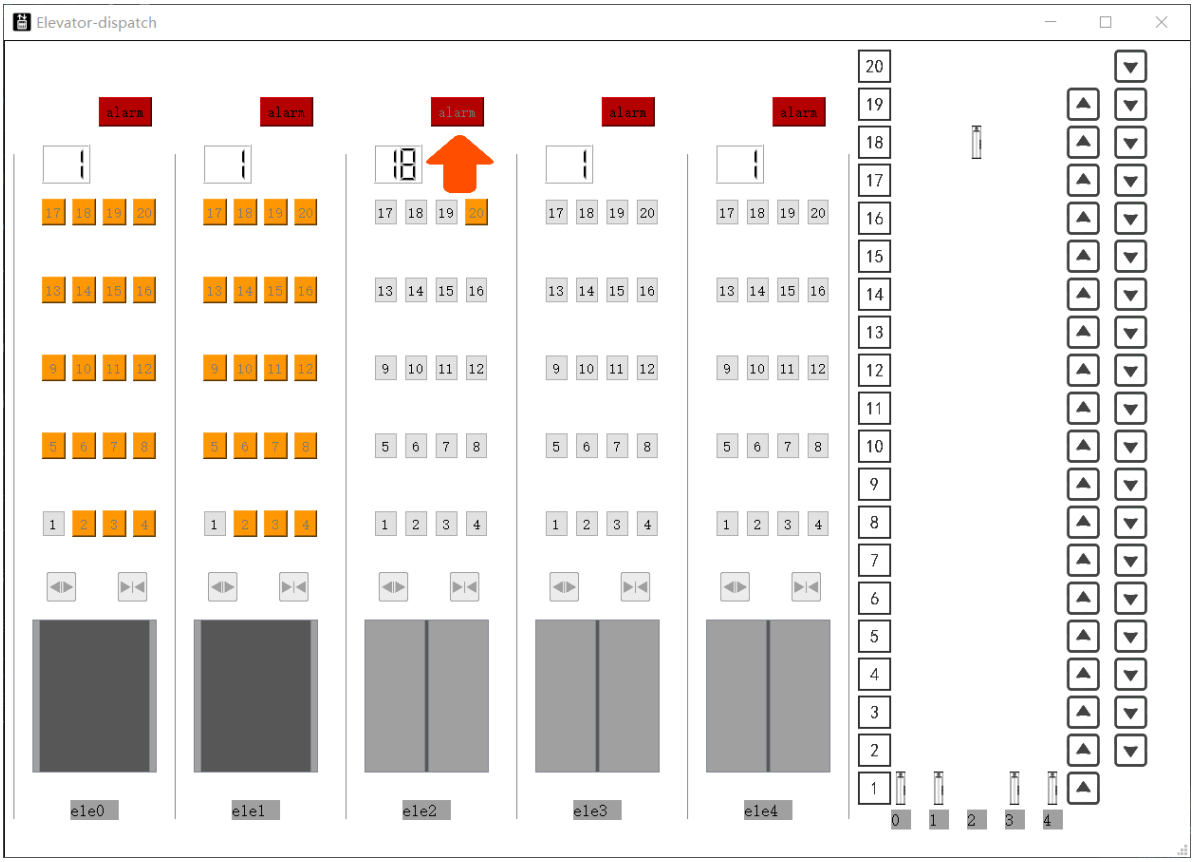


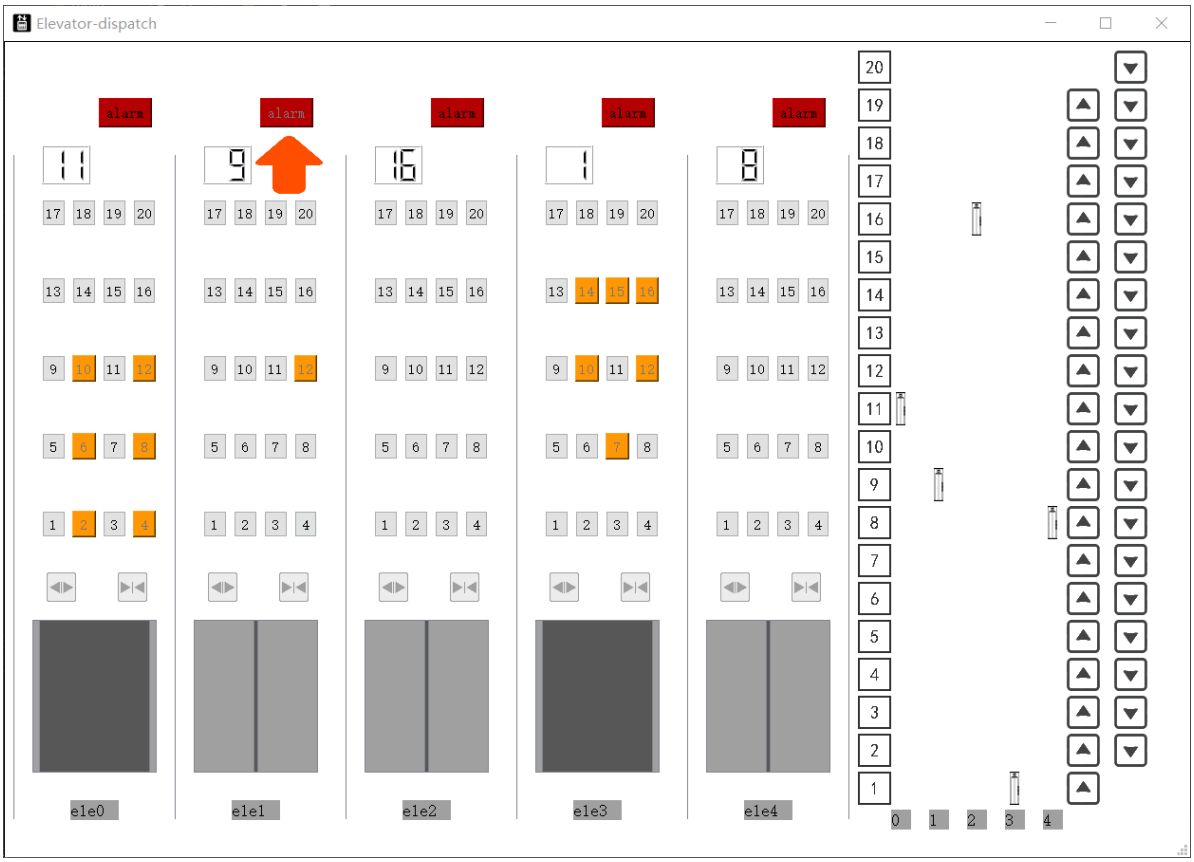

1. 最上方的红色方块为电梯停用按键，点击后电梯禁用但无法恢复。
2. 数字按键1-20位于电梯内部，点击后电梯向目标位置运动。
3. 数字按键上方和停用按键之间为状态显示，数码管显示电梯所处楼层，数码管右侧在电梯运行时会显示上下行状态，静止时为空白。
4. 数字键下方为电梯内部开关门按键，只有在电梯静止时才能使用。
5. 界面右侧为20层楼以及每层楼对应的上下行按键，点击按键，产生请求，对五部电梯统一调度。

## 4.2 功能截屏展示

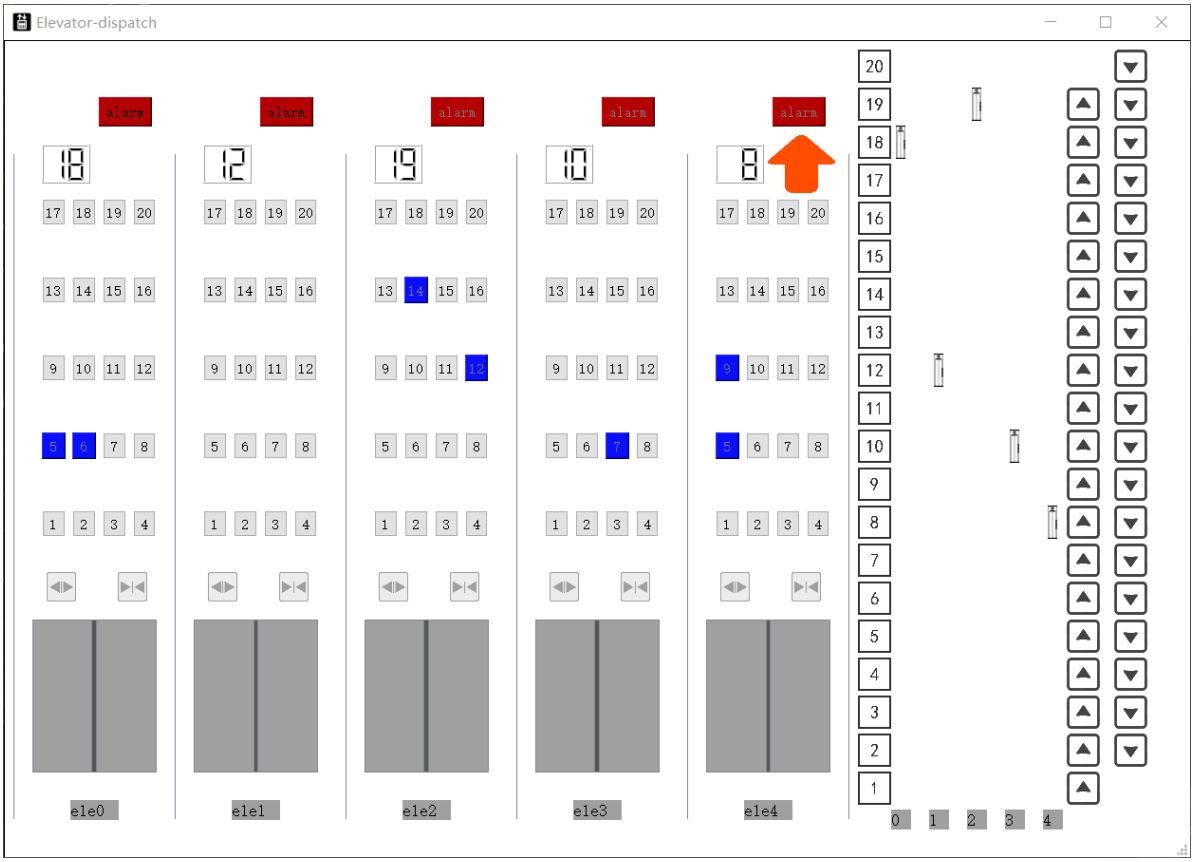
### 4.2.1 正常运行

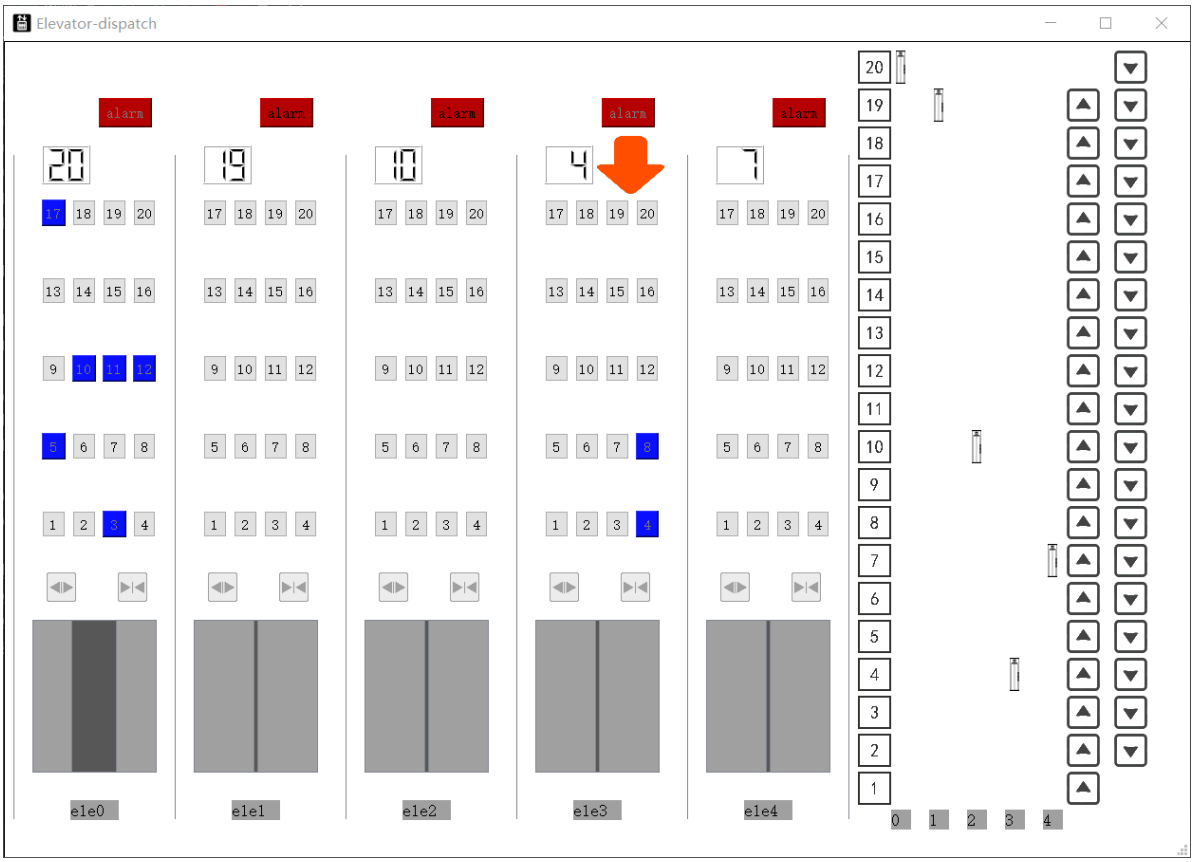
电梯内部数字键控制


Elevator-dispatch
alarm
ele0
ele1
ele2
ele3
ele4

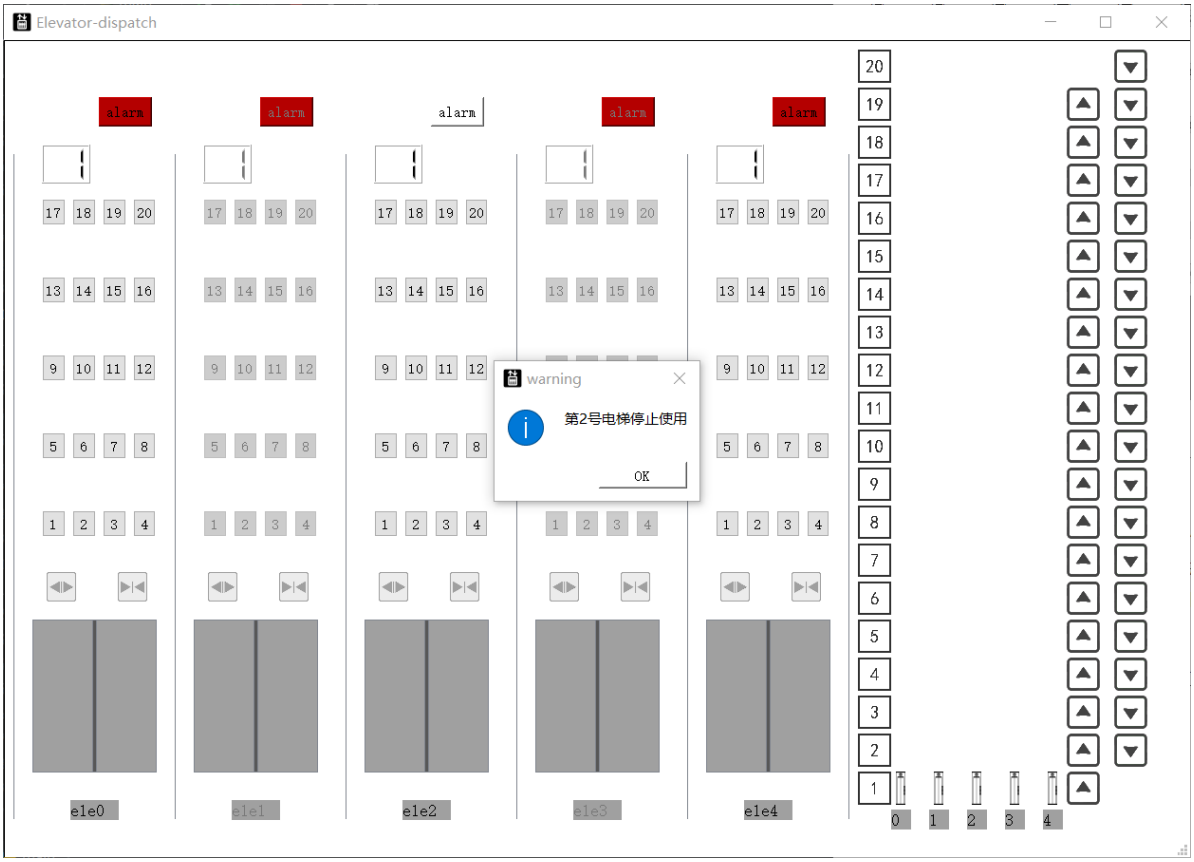

Elevator-dispatch
alarm
ele0
ele1
ele2
ele3
ele4

电梯外部方向键控制

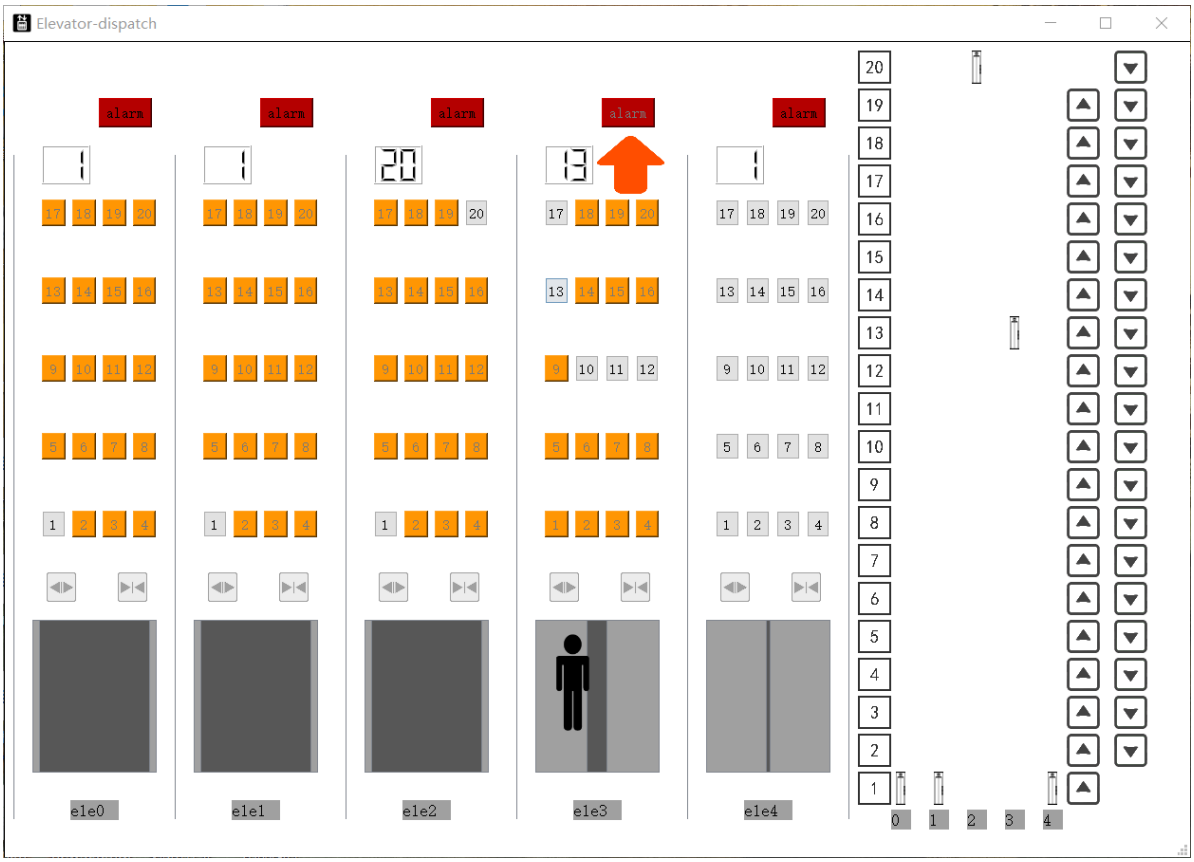

Elevator-dispatch
alarm
alarm
alarm
alarm
alarm
18
12
19
10
8
e1e0
e1e1
e1e2
e1e3
e1e4


Elevator-dispatch
alarm
alarm
alarm
alarm
alarm
20
19
10
4
7
e1e0
e1e1
e1e2
e1e3
e1e4

### 4.2.2 电梯停用


Elevator-dispatch
alarm
warning
第2号电梯停止使用
OK
ele0
ele1
ele2
ele3
ele4

### 4.2.3 开关门


Elevator-dispatch
alarm
ele0
ele1
ele2
ele3
ele4