**操作系统——处理机管理项目1**

**电梯调度模拟**

**学号：2151298**

**姓名：杨滕超**

**班号：42036902**

**指导老师：张惠娟**

目录

**[操作系统——处理机管理项目1 1](#_Toc5737)**

**[1. 项目介绍 2](#_Toc19532)**

[1.1. 项目背景 3](#_Toc8365)

[1.2. 项目目的 3](#_Toc22300)

**[2. 需求分析 3](#_Toc6151)**

[2.1. 按键 3](#_Toc27601)

[2.2. 调度 3](#_Toc9027)

**[3. 功能描述 3](#_Toc16686)**

[3.1. 电梯调度 3](#_Toc17560)

[3.2. 随机生成任务 4](#_Toc25865)

[3.3. 其他按钮 4](#_Toc32024)

[3.4. 任意选择电梯与楼层数量 4](#_Toc7287)

**[4. 开发环境 4](#_Toc19620)**

**[5. 界面设计 4](#_Toc621)**

[5.1. 电梯楼层显示区域 5](#_Toc31378)

[5.2. 电梯门状态显示区域 5](#_Toc16470)

[5.3. 电梯选择区域 5](#_Toc13409)

[5.4. 电梯内部楼层选择区域 5](#_Toc22638)

[5.5. 电梯功能按钮区域 6](#_Toc15150)

[5.6. 外部上下行按钮 6](#_Toc30691)

[5.7. UI界面代码 7](#_Toc24019)

**[6. 系统设计 11](#_Toc14089)**

[6.1. 总体设计 11](#_Toc11461)

[6.2. 类设计 11](#_Toc11461)

**[7. 算法 13](#_Toc32311)**

[7.1. 调度算法 13](#_Toc22940)

[7.2. 电梯状态改变 14](#_Toc19471)

**[8. 程序运行 16](#_Toc30342)**

[8.1. 单部电梯选择内部楼层按钮 16](#_Toc22940)

[8.2. 楼层到达开关门操作 1](#_Toc22940)6

[8.3. 电梯故障设置 1](#_Toc22940)6

[8.4. 多部电梯同时运行，随机分配任务 1](#_Toc22940)7

[8.5. 用户输入电梯数量与楼层数量 1](#_Toc22940)7

[8.6. 对故障电梯内部按钮操作 1](#_Toc22940)8

[8.7. 所有电梯故障 1](#_Toc22940)8

1. **项目介绍**
   1. **项目背景**

某一层楼20层，有五部互联的电梯。基于线程思想，编写一个电梯调度程序。

* 1. **项目目的**

通过模拟电梯调度，学习并掌握操作系统处理机调度策略。

掌握多线程编程方法。

1. **需求分析**

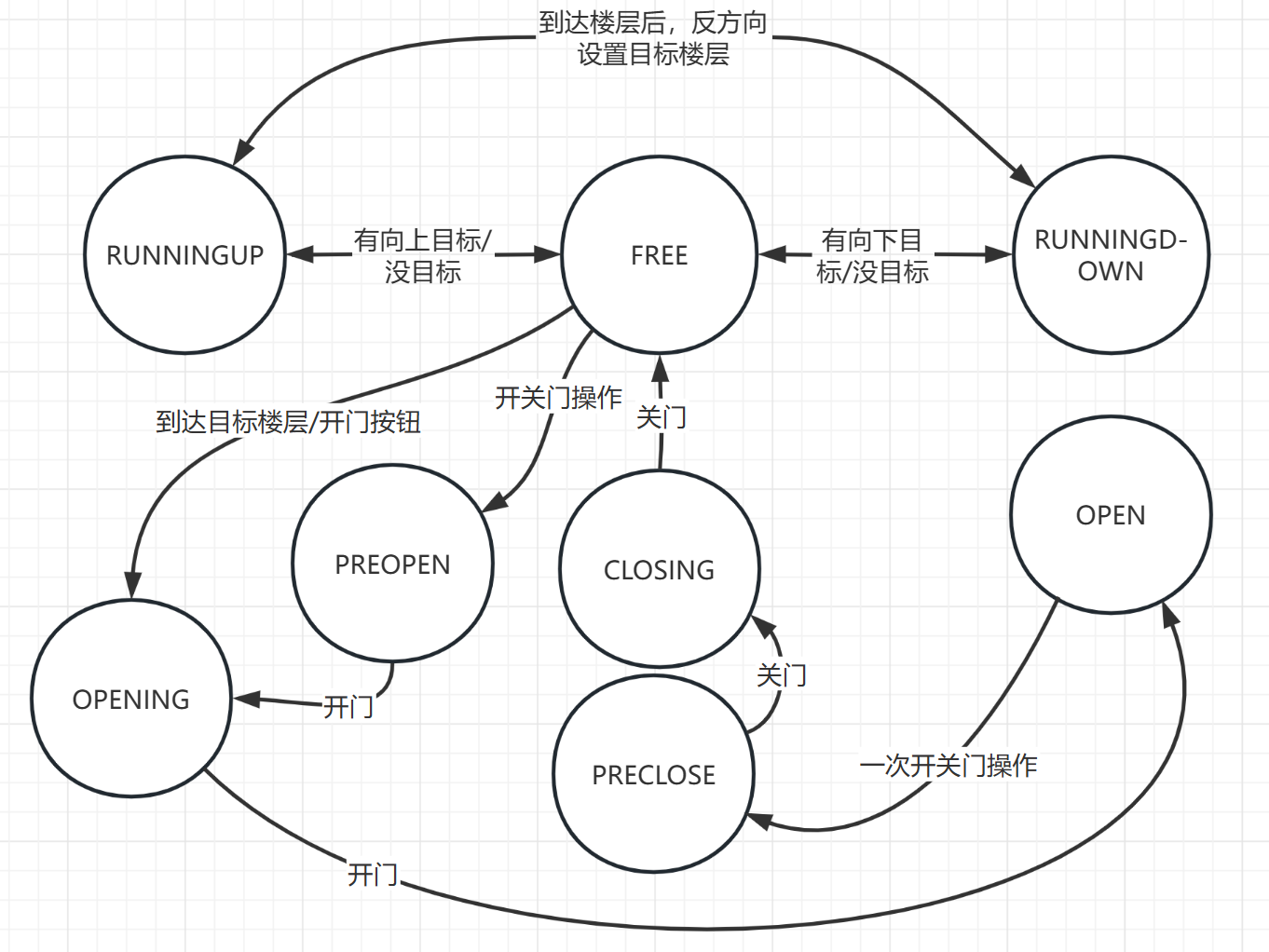
**2.1. 按键**

电梯中有一些按键，如：数字键、关门键、开门键、上行键、下行键、报警键等；有数码显示器指示当前电梯状态；每层楼、每部电梯门口，有上行、下行按钮。

**2.2. 调度**

每一个要乘坐电梯的乘客按下电梯外部楼层中的按钮相当于创建一个进程，该进程的任务即为请求获得电梯；而电梯可以类比于CPU处理器，本项目模拟调度算法将外部请求调度到电梯上。同时，也可以在电梯内部按下按钮，为该电梯创建任务。

电梯状态图如下：



1. **功能描述**

**3.1. 电梯调度**

对于电梯内部按钮按下后，按钮显示颜色变化，同时电梯会移动至目标楼层。

对于电梯外部按钮按下后，按钮显示颜色变化，系统通过算法将一部电梯调度至本层楼。

传统电梯调度算法：

* 先到先服务算法（First Come,First Served FCFS）。是一种随即服务算法，它不仅仅没有对寻找楼层进行优化，也没有实时性的特征，它是一种最简单的电梯调度算法。该算法根据乘坐电梯的先后次序调度，虽然公平、简单，但是一旦载荷较大，这种算法的性能会严重下降。
* 最短寻找楼层时间优先算法（Shortest Seek Time First SSTF）。该算法每次寻找可以最快满足的楼层（距离该电梯最近），这样可以减少平均响应时间较短，但响应时间的方差较大，原因是队列中的某些请求（例如20层的请求）可能长时间得不到响应，出现所谓的“饥饿”现象。
* 扫描算法（SCAN）：该算法是从移动臂当前位置开始沿着臂的移动方向去选择离当前移动臂最近的那个柱访问者，如果沿臂的移动方向无请求访问时，就改变臂的移动方向再选择。SCAN算法是电梯前进方向上的最短查找时间优先算法，它排除了电梯在盘面局部位置上的往复移动，SCAN算法在很大程度上消除了SSTF算法的不公平性，但仍有利于对中间磁道的请求。循环扫描算法（CSCAN）：规定电梯单向扫描，然后立即返回重新开始。
* LOOK 算法。该算法是对扫描算法的改进，当电梯发现移动方向上不再有目标时立即改变方向，无需移动到最底层或最高层

**3.2. 随机生成任务**

通过输入任务数量并点击生成按钮，系统将对非故障电梯以及外部按钮按比例随机分配任务。

**3.3. 其他按钮**

**3.3.1. 警报按钮**

通过点击警报按钮，该电梯将停止运行，并且数码显示器显示异常，并且将调度给该电梯的外部任务重新调度给其他电梯；若所有电梯均故障，系统将弹出消息窗口提示；若在已经故障的电梯中点击内部按钮，系统将同样弹出消息窗口提示。

**3.3.2. 开关门按钮**

仅当电梯处于静止状态时才可以使用开关门按钮。同时，开关门进行时不能被打断。对于电梯到达目的楼层时，同样会进行开关门操作。

**3.4. 任意选择电梯与楼层数量**

在程序执行之初，用户可以通过输入合理数字，从而选择电梯的数量与楼层的数量。

1. **开发环境**

开发环境：Windows\_NT x64 10.0.19044

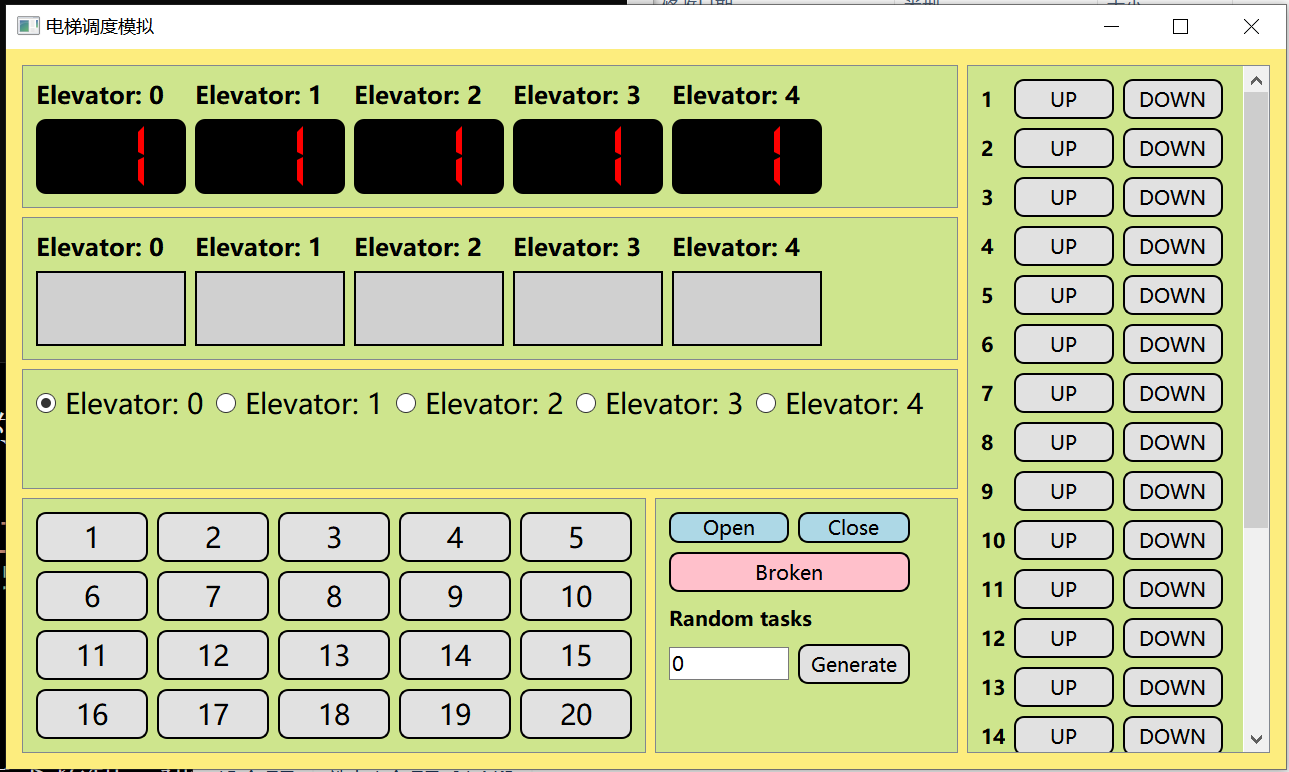
开发软件：VSCode1.76.0

开发语言：Python3

IDE：Python 3.10 64-bit

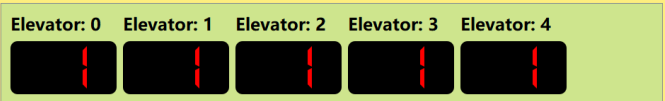
主要引用模块：PyQt5

1. **界面设计**



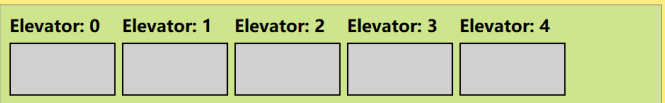
本项目设置为界面设置5个区域，分别表示不同功能区。下面将按照从上到下、从左到右的顺序介绍。

**5.1. 电梯楼层显示区域**



本区域提供电梯编号以及对应楼层的显示，实时显示电梯当前到达楼层。

**5.2. 电梯门状态显示区域**



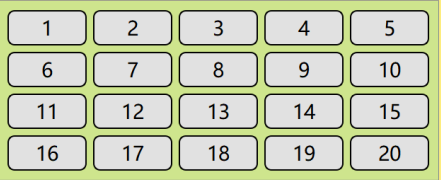
本区域提供电梯编号以及对应电梯门状态显示，通过动画形式表现当前电梯门处于何种状，其中包括开启、关闭、开启中以及关闭中。

**5.3. 电梯选择区域**



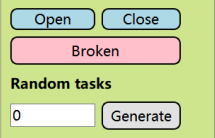
为了保证对界面合理的规划以及更多电梯的复用，这里使用单选按钮表示当前电梯的选择。选择某一电梯，相应的电梯内部按钮也会有相应显示与点击效果，从而达到了对同一套电梯内部按钮的复用。

**5.4. 电梯内部楼层选择区域**



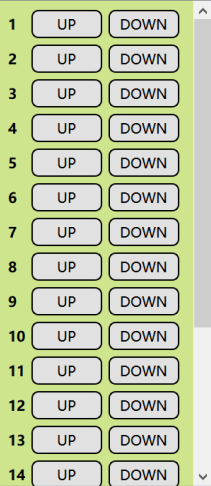
本区域提供某一部电梯内部楼层选择按钮的显示，选择后出现高亮效果，根据单选按钮而呈现出不同电梯的内部情况。

**5.5. 电梯功能按钮区域**



本区域提供开门、关门、故障按钮以及随机任务生成器。其中前三者同样根据单选按钮而呈现出不同电梯的内部情况，从而达到控制不同电梯的效果。注意到，电梯上下移动以及开关门的同时，开关门按钮将无法生效，只有当电梯停止在某层楼开关门按钮才有效。对应故障按钮，随时有效，按下后该电梯所有按钮都将无效。随机任务生成器通过输入合理数值将随机生成任务。

**5.6. 外部上下行按钮**



本区域提供每层楼的上下行按钮，本人同样考虑过仅仅对两个上下行按钮以及楼层选择的复用效果，但发现操作过于繁琐以及显示效果并不尽如人意，于是将每层楼上下行按钮都显示出来，并且通过可以通过滚动条在固定大小区域查看更多的楼层按钮。同样地，选中后会有高亮效果。

**5.7. UI界面代码**

# 样式表

        self.setStyleSheet(

            'QPushButton{border-radius: 10px; border: 2px solid black}'

            'QPushButton:pressed{padding-left:5px; padding-top:5px}'

            'QLCDNumber{background-color: black; color: red; border-radius: 10px}'

            'QLabel{font-weight: bold;}'

            '#mainWindow{background-color: #fded7e}'

            'QScrollArea{background-color: #cee58d}'

        )

        # 标题

        self.setWindowTitle('电梯调度模拟')

        # 窗口名字

        self.setObjectName('mainWindow')

        # 窗口大小

        self.resize(WIDTH, HEIGHT)

        # 网格整体布局

        gridLayout = QGridLayout()

        # 滚动区域中放LCD

        scrollArea\_lcd = QScrollArea()

        grid = QGridLayout()

        for i in range(NUM\_OF\_ELEVATOR):

            # 文本标记哪个电梯

            label = QLabel('Elevator: %d' % i)

            label.setFont(QFont("Microsoft YaHei", 12))

            # 电梯数字显示

            lcd = QLCDNumber(INIT\_FLOOR + 1)

            lcd.setDigitCount(len(str(NUM\_OF\_FLOOR)))

            lcd.setSegmentStyle(QLCDNumber.Flat)

            lcd.setObjectName('elev%d' % i)

            lcd.setMinimumSize(150, 75)

            grid.addWidget(label, i // 5 \* 2, i % 5)

            grid.addWidget(lcd, i // 5 \* 2 + 1, i % 5)

        scroll\_w = QWidget()

        scroll\_w.setStyleSheet('background-color: #cee58d')

        scroll\_w.setLayout(grid)

        scrollArea\_lcd.setWidget(scroll\_w)

        # 电梯门

        scrollArea\_door = QScrollArea()

        grid = QGridLayout()

        for i in range(NUM\_OF\_ELEVATOR):

            # 文本标记哪个电梯

            label = QLabel('Elevator: %d' % i)

            label.setFont(QFont("Microsoft YaHei", 12))

            # 电梯door\_back显示

            door\_back = QLabel()

            door\_back.setMinimumSize(150, 75)

            door\_back.setStyleSheet('background-color: #6C6C6C;  border: 2px solid black')

            door = QLabel()

            door.setObjectName('%d' % i)

            door.setStyleSheet('background-color: #d0d0d0;  border: 2px solid black;')

            grid.addWidget(label, i // 5 \* 2, i % 5)

            grid.addWidget(door\_back, i // 5 \* 2 + 1, i % 5)

            grid.addWidget(door, i // 5 \* 2 + 1, i % 5)

        scroll\_w = QWidget()

        scroll\_w.setStyleSheet('background-color: #cee58d')

        scroll\_w.setLayout(grid)

        scrollArea\_door.setWidget(scroll\_w)

        # 电梯选择按钮

        scrollArea\_radioButton = QScrollArea()

        grid = QGridLayout()

        for i in range(NUM\_OF\_ELEVATOR):

            radio = QRadioButton('Elevator: %d' % i)

            radio.setFont(QFont("Microsoft YaHei", 14))

            radio.setObjectName('elev%d' % i)

            grid.addWidget(radio, i // 5, i % 5)

            # 初始为第0个电梯被选中

            if i == 0:

                radio.setChecked(True)

        scroll\_w = QWidget()

        scroll\_w.setStyleSheet('background-color: #cee58d')

        scroll\_w.setLayout(grid)

        scrollArea\_radioButton.setWidget(scroll\_w)

        # 电梯内按钮

        scrollArea\_btn = QScrollArea()

        grid = QGridLayout()

        # 楼层按钮

        for i in range(NUM\_OF\_FLOOR):

            btn = QPushButton('%d' % (i + 1))

            btn.setObjectName('%d' % i)

            btn.setFont(QFont("Microsoft YaHei", 14))

            btn.setStyleSheet('background-color: #E1E1E1;')

            btn.setMinimumSize(20, 50)

            btn.clicked.connect(partial(self.floor\_btn\_clicked, i))

            grid.addWidget(btn, i // 5, i % 5)

        scroll\_w = QWidget()

        scroll\_w.setStyleSheet('background-color: #cee58d')

        scroll\_w.setLayout(grid)

        scrollArea\_btn.setWidget(scroll\_w)

        # 开关门按钮 故障按钮 自动产生任务

        scrollArea\_btn\_feature = QScrollArea()

        grid = QGridLayout()

        # 控件

        btn\_open = QPushButton('Open')

        btn\_close = QPushButton('Close')

        btn\_broken = QPushButton('Broken')

        task\_text = QLabel('Random tasks')

        task\_num\_input = QLineEdit('0')

        btn\_task\_generate = QPushButton('Generate')

        # 名字

        btn\_open.setObjectName('open')

        btn\_close.setObjectName('close')

        btn\_broken.setObjectName('broken')

        task\_num\_input.setObjectName('task\_num\_input')

        btn\_task\_generate.setObjectName('generate')

        # 大小

        btn\_broken.setMinimumSize(120, 40)

        btn\_task\_generate.setMinimumSize(100, 40)

        # 初始颜色

        btn\_open.setStyleSheet('QPushButton{background-color: lightblue}'

            'QPushButton:hover{background-color: lightsteelblue}'

            )

        btn\_close.setStyleSheet('QPushButton{background-color: lightblue}'

            'QPushButton:hover{background-color: lightsteelblue}'

            )

        btn\_broken.setStyleSheet('QPushButton{background-color: pink}'

            'QPushButton:hover{background-color: hotpink}')

        btn\_task\_generate.setStyleSheet('QPushButton:hover{background-color: rgb(150, 150, 150)}')

        # 设置字体

        font\_size = 10

        btn\_open.setFont(QFont("Microsoft YaHei", font\_size))

        btn\_close.setFont(QFont("Microsoft YaHei", font\_size))

        btn\_broken.setFont(QFont("Microsoft YaHei", font\_size))

        task\_text.setFont(QFont("Microsoft YaHei", font\_size))

        task\_num\_input.setFont(QFont("Microsoft YaHei", font\_size))

        btn\_task\_generate.setFont(QFont("Microsoft YaHei", font\_size))

        # 输入限制

        task\_num\_input.setValidator(QIntValidator(bottom = 0, top = (NUM\_OF\_ELEVATOR \* NUM\_OF\_FLOOR + 2 \* (NUM\_OF\_FLOOR - 1)) // Q))

        # 绑定槽函数

        btn\_open.clicked.connect(self.open\_animate)

        btn\_close.clicked.connect(self.close\_animate)

        btn\_broken.clicked.connect(self.broken\_btn\_clicked)

        btn\_task\_generate.clicked.connect(self.generate\_task)

        # 控件大小

        task\_num\_input.setMaximumWidth(120)

        # 将控件加入网格布局

        grid.addWidget(btn\_open, 0, 0)

        grid.addWidget(btn\_close, 0, 1)

        grid.addWidget(btn\_broken, 1, 0, 1, 2)

        grid.addWidget(task\_text, 2, 0, 1, 2)

        grid.addWidget(task\_num\_input, 3, 0, 1, 1)

        grid.addWidget(btn\_task\_generate, 3, 1, 1, 1)

        scroll\_w = QWidget()

        scroll\_w.setStyleSheet('background-color: #cee58d')

        scroll\_w.setLayout(grid)

        scrollArea\_btn\_feature.setWidget(scroll\_w)

        # 电梯外按钮

        scrollArea\_outer\_btn = QScrollArea()

        grid = QGridLayout()

        for i in range(NUM\_OF\_FLOOR):

            label = QLabel('%d' % (i + 1))

            btn\_up = QPushButton('UP')

            btn\_down = QPushButton('DOWN')

            btn\_up.setFixedSize(100, 40)

            btn\_down.setFixedSize(100, 40)

            btn\_up.setObjectName('up%d' % i)

            btn\_down.setObjectName('down%d' % i)

            btn\_up.setStyleSheet('background-color: #E1E1E1;')

            btn\_down.setStyleSheet('background-color: #E1E1E1;')

            btn\_up.clicked.connect(partial(self.outer\_btn\_request, i, RUNNINGUP))

            btn\_down.clicked.connect(partial(self.outer\_btn\_request, i, RUNNINGDOWN))

            label.setFont(QFont("Microsoft YaHei", 10))

            btn\_up.setFont(QFont("Microsoft YaHei", 10))

            btn\_down.setFont(QFont("Microsoft YaHei", 10))

            grid.addWidget(label, i, 0)

            grid.addWidget(btn\_up, i, 1)

            grid.addWidget(btn\_down, i, 2)

        scroll\_w = QWidget()

        scroll\_w.setStyleSheet('background-color: #cee58d')

        scroll\_w.setLayout(grid)

        scrollArea\_outer\_btn.setWidget(scroll\_w)

        # 网格布局

        gridLayout.addWidget(scrollArea\_lcd, 0, 0, 1, 2)

        gridLayout.addWidget(scrollArea\_door, 1, 0, 1, 2)

        gridLayout.addWidget(scrollArea\_radioButton, 2, 0, 1, 2)

        gridLayout.addWidget(scrollArea\_btn, 3, 0, 1, 1)

        gridLayout.addWidget(scrollArea\_btn\_feature, 3, 1, 1, 1)

        gridLayout.addWidget(scrollArea\_outer\_btn, 0, 3, 4, 1)

        self.setLayout(gridLayout)

1. **系统设计**

**6.1. 总体设计**

设计类PCB作为外部楼层按钮点击的请求，类Elevator作为电梯内部处理的类，同时设计myWin类处理界面显示以及相应用户点击等操作，同时该类将上述两个类联系起来，总体调控电梯以及任务状态。

全局变量如下：

|  |  |  |
| --- | --- | --- |
| **变量名** | **作用** | **初始值** |
| GOON | 程序是否继续 | True |
| NUM\_OF\_ELEVATOR | 电梯数量 | 5 |
| NUM\_OF\_FLOOR | 楼层数量 | 20 |
| Q | 随机生成任务数量阈值 | 1.25 |
| DOOR\_OPERATION\_TIME | 开门、关门操作时间 | 4000(ms) |
| WIDTH, HEIGHT | 主窗口大小 | 1280, 720 |
| INIT\_FLOOR | 电梯初始楼层 | 0 |
| FREE | 电梯状态：门关闭、空闲 | 0 |
| RUNNINGUP | 电梯状态：上行 | 1 |
| RUNNINGDOWN | 电梯状态：下行 | 2 |
| OPENING | 电梯状态：开门中 | 3 |
| CLOSING | 电梯状态：关门中 | 4 |
| BROKEN | 电梯状态：故障 | 5 |
| OPEN | 电梯状态：门开启 | 6 |
| PREOPEN | 电梯状态：将要开门 | 7 |
| PRECLOSE | 电梯状态：将要关门 | 8 |
| WAITING | 任务状态：等待 | 9 |
| RUNNING | 任务状态：运行中 | 10 |
| FINISHED | 任务状态：完成 | 11 |
| outer\_task | 外部任务队列 | [] |
| cur\_floor | 每部电梯当前楼层 | [INIT\_FLOOR] \* NUM\_OF\_ELEVATOR |
| elevs | 电梯队列 | [] |

**6.2. 类设计**

**6.2.1. PCB类**

该类表示外部任务请求，其中self.start\_floor: 任务发出楼层，self.status: 任务状态，默认为WAITING，self.direction: 任务请求方向。

每当外部楼层按钮被点击时，就会创建一个PCB的实例，加入外部任务队列。

**6.2.2. Elevator类**

该类继承于QThread类，表示电梯以及内部逻辑处理。数据成员包括：self.id: 电梯编号，self.status: 电梯状态，self.des: 该电梯目标，self.inner\_target: 该电梯内部按钮产生的目标。

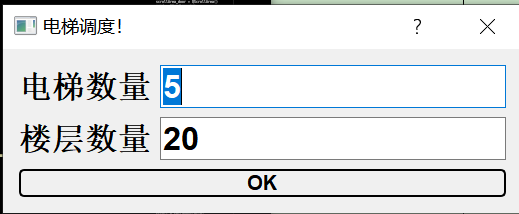
包含内部方法如下：

|  |  |
| --- | --- |
| **方法名** | **作用** |
| move() | 电梯移动控制 |
| elev\_err() | 电梯故障处理 |
| set\_err() | 将电梯设置为故障状态 |
| status\_check() | 电梯状态检查改变 |
| outer\_task\_check() | 外部任务完成检查 |
| run() | 线程运行函数 |

**6.2.3. myDialog类**

该类继承于QDialog类，在程序运行一开始显示，并接受合理数字输入，由此决定电梯与楼层的数量。初始值分别为5和20。若直接关闭该窗口程序会直接结束；若点击OK按钮或回车，系统会检查输入是否合理，若合理则直接进入主程序，否则继续等待正确输入。

运行如下图所示：



**6.2.4. myWin类**

该类继承于QWidget类，主要用于主界面的显示以及外部电梯任务的调度处理，由此将6.2.1. PCB类以及6.2.2. Elevator类联系起来。其中数据成员包括self.timer: 用于定时刷新界面以及处理外部任务调度。

包含内部方法如下：

|  |  |
| --- | --- |
| **方法名** | **作用** |
| setup\_UI() | 界面绘制 |
| floor\_btn\_clicked(target\_floor) | 电梯内部楼层按钮处理槽函数 |
| broken\_btn\_clicked() | 电梯故障按钮处理槽函数 |
| outer\_btn\_request(start\_floor, direction) | 外部楼层按钮处理槽函数 |
| generate\_task() | 随机产生任务按钮槽函数 |
| update\_lcd() | 更新电梯楼层显示 |
| outer\_task\_deal() | 外部任务队列处理函数 |
| check\_all\_broken() | 检查所有电梯是否都故障函数 |
| update\_btn() | 更新按钮状态显示 |
| update() | 刷新界面以及任务处理 |
| set\_elev\_free(elev\_id) | 设置指定电梯为FREE状态 |
| set\_elev\_open(elev\_id) | 设置指定电梯为OPEN状态 |
| check\_open\_close\_door() | 检查是否需要开关门动画 |
| elev\_open\_close\_opening(elev\_id) | 开门动画 |
| elev\_open\_close\_closing(elev\_id) | 开门后的关门动画 |
| open\_animate() | 开门按钮槽函数 |
| close\_animate() | 关门按钮槽函数 |

1. **算法**

**7.1. 调度算法**

    # 外部任务调度

    def outer\_task\_deal(self):

        outer\_task\_mutex.lock()

        global outer\_task

        # 分配任务

        for task in outer\_task:

            # 只关注在等待的任务

            if task.status == FINISHED or task.status == RUNNING:

                continue

            # 初始距离为楼层数

            dis = [NUM\_OF\_FLOOR] \* NUM\_OF\_ELEVATOR

            for i, elev in enumerate(elevs):

                if elev.status == BROKEN:

                    continue

                # 下行顺路

                if elev.status == task.direction and elev.status == RUNNINGDOWN and cur\_floor[i] > task.start\_floor:

                    dis[i] = cur\_floor[i] - task.start\_floor

                # 上行顺路

                elif elev.status == task.direction and elev.status == RUNNINGUP and cur\_floor[i] < task.start\_floor:

                    dis[i] = task.start\_floor - cur\_floor[i]

                # 静止

                elif elev.status == FREE:

                    dis[i] = abs(task.start\_floor - cur\_floor[i])

            # 选择距离最近的

            idx = dis.index(min(dis))

            elevs[idx].des.add(task.start\_floor)

            print('将 %d 请求分配给电梯%d' % (task.start\_floor, idx))

            task.status = RUNNING

        # 过滤掉已经完成的

        outer\_task = [t for t in outer\_task if t.status != FINISHED]

        outer\_task\_mutex.unlock()

算法思路

涉及到对outer\_task内容的更改，这里使用锁QMutex的实例化对象outer\_task\_mutex保证一次只能一个线程访问其中的内容。

遍历outer\_task中的任务，这里仅对处于等待状态，即task.status == WAITING的任务进行分配，因此处于FINISHED和RUNNING状态的task直接跳过

设置处理该任务对于每部电梯的代价列表dis，初始值设置为NUM\_OF\_FLOOR最大值

遍历每部电梯，若电梯故障直接跳过；若电梯上行且顺路，即任务楼层比该电梯当前楼层高，则更新dis列表中的值；下行顺路同理；若电梯静止状态，即FREE，则取绝对值更新dis列表。

选择代价最小的电梯作为调度目标，并将任务状态设置为RUNNING。

最后过滤掉已经完成的任务。

**7.2. 电梯状态改变**

# 状态改变

    def status\_check(self):

        # 故障了就不管了

        if self.status == BROKEN:

            return

        # 开关门状态

        if self.status in (OPEN, OPENING, CLOSING, PREOPEN, PRECLOSE):

            return

        # 到达楼层

        if len(self.des) != 0:

            for floor in self.des:

                if floor == cur\_floor[self.id]:

                    # 开关门

                    self.status = PREOPEN

                    self.des.discard(floor)

                    self.inner\_target.discard(floor)

                    # break

                    # 交给开关门操作变化状态最后为free

                    return

        # 没有任务

        if len(self.des) == 0:

            self.status == FREE

        # 有任务

        else:

            if min(self.des) > cur\_floor[self.id]:

                self.status = RUNNINGUP

            elif max(self.des) < cur\_floor[self.id]:

                self.status = RUNNINGDOWN

            # 要上要下去近的

            else:

                if abs(max(self.des) - cur\_floor[self.id]) < abs(min(self.des) - cur\_floor[self.id]):

                    self.status = RUNNINGUP

                else:

                    self.status = RUNNINGDOWN

首先需要检查该部电梯的状态，若是处于故障、开关门中以及开门状态，此时电梯状态不能改变，则直接退出。

接着检查该部电梯的目标，若现在的楼层处于电梯目标楼层中，进行一次开关操作，并将目标设置为完成状态。

若电梯任务列表为空，则需要将电梯设置为FREE状态，表示电梯关门静止状态。

若电梯有目标，此时需要判断：

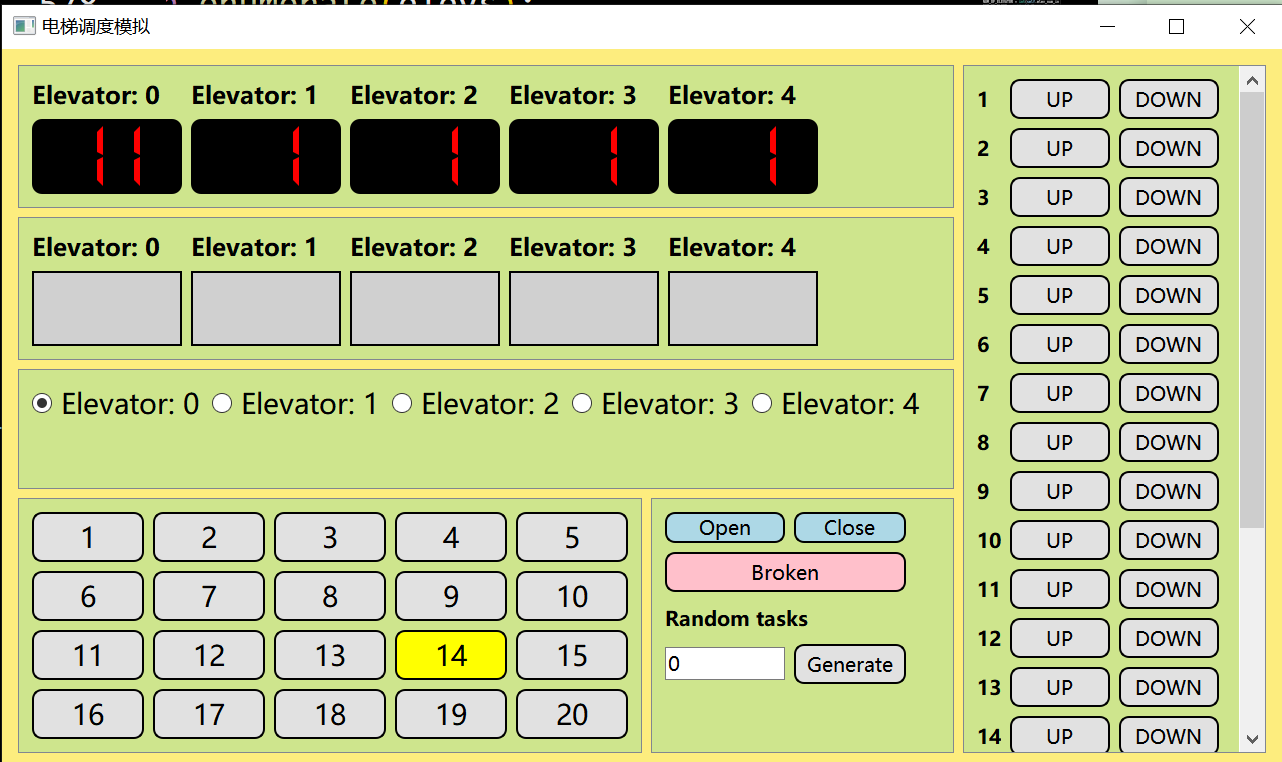
若最低的楼层目标比当前楼层高，则将电梯设置为RUNNINGUP状态，表示向上移动。

若最高的楼层目标比当前楼层低，则将电梯设置为RUNNINGDOWN状态，表示向下移动。

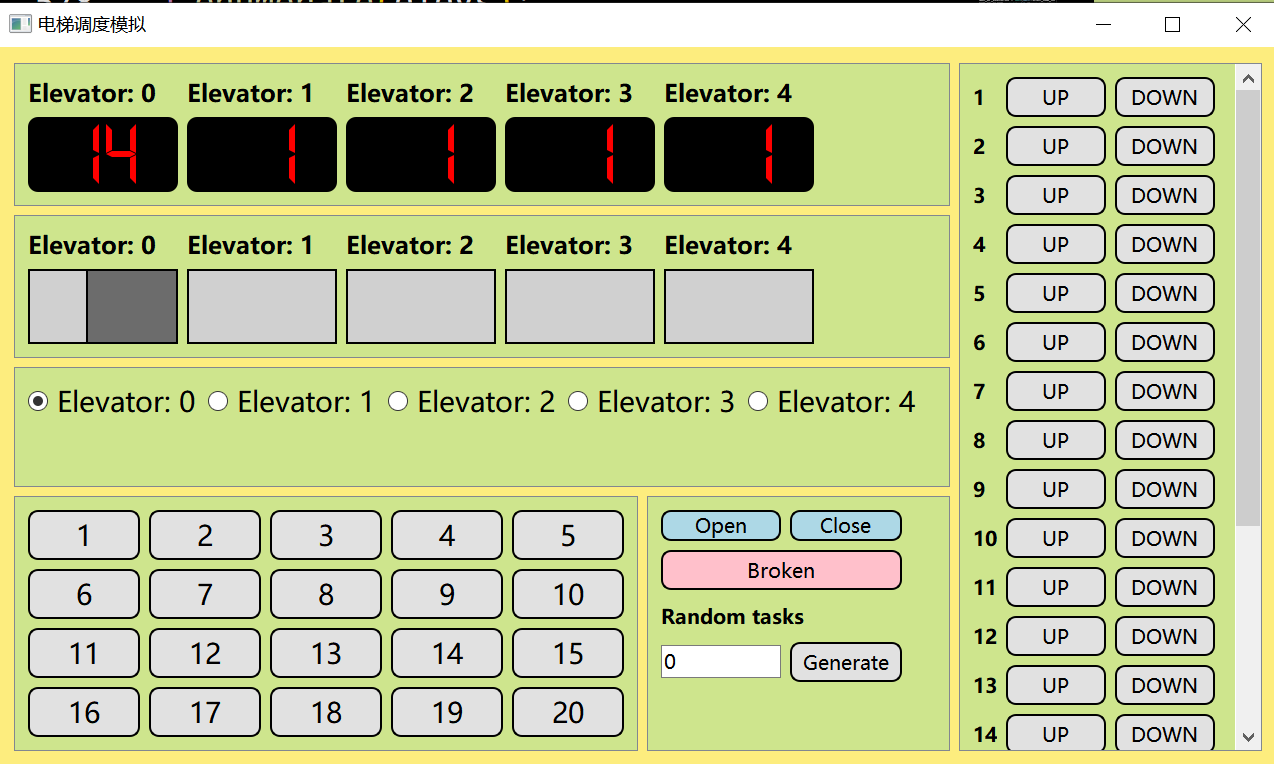
否则，比较最高的目标楼层以及最低的目标楼层，选择距离当前楼层近的移动。

**8. 程序运行**

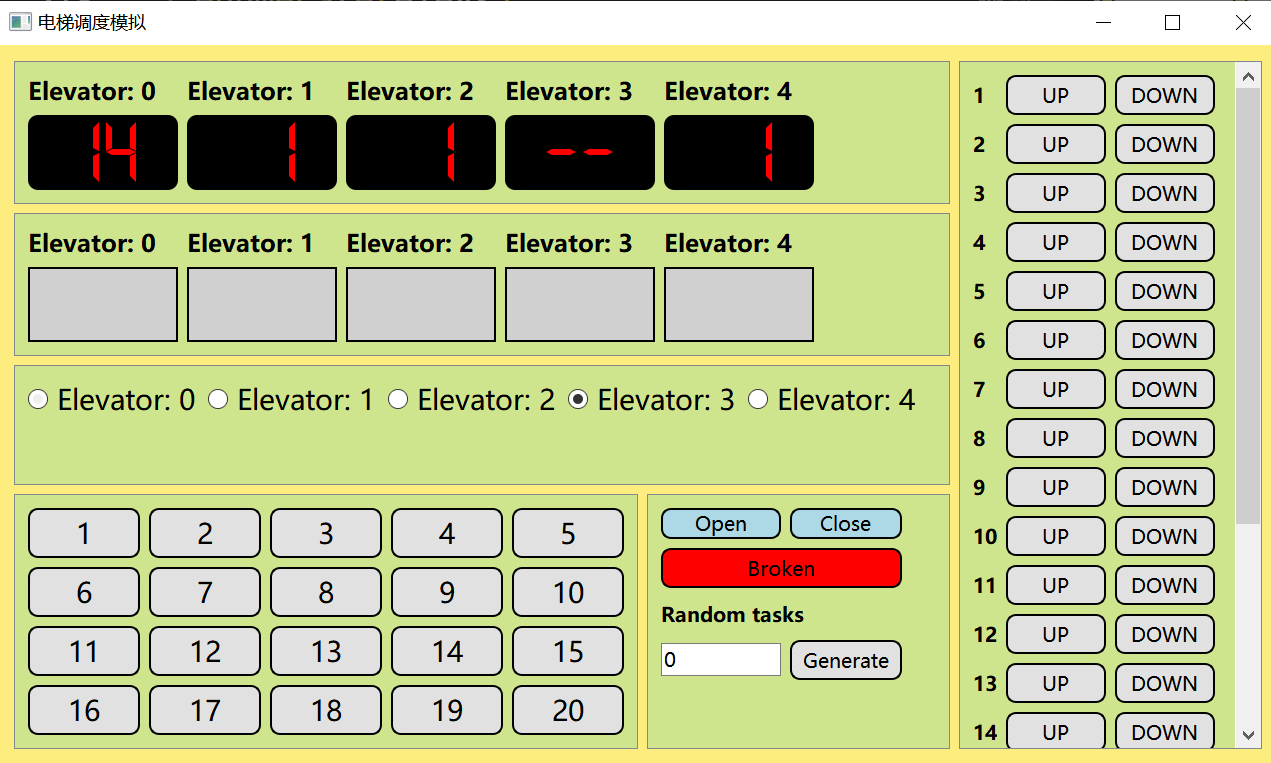
**8.1. 单部电梯选择内部楼层按钮**



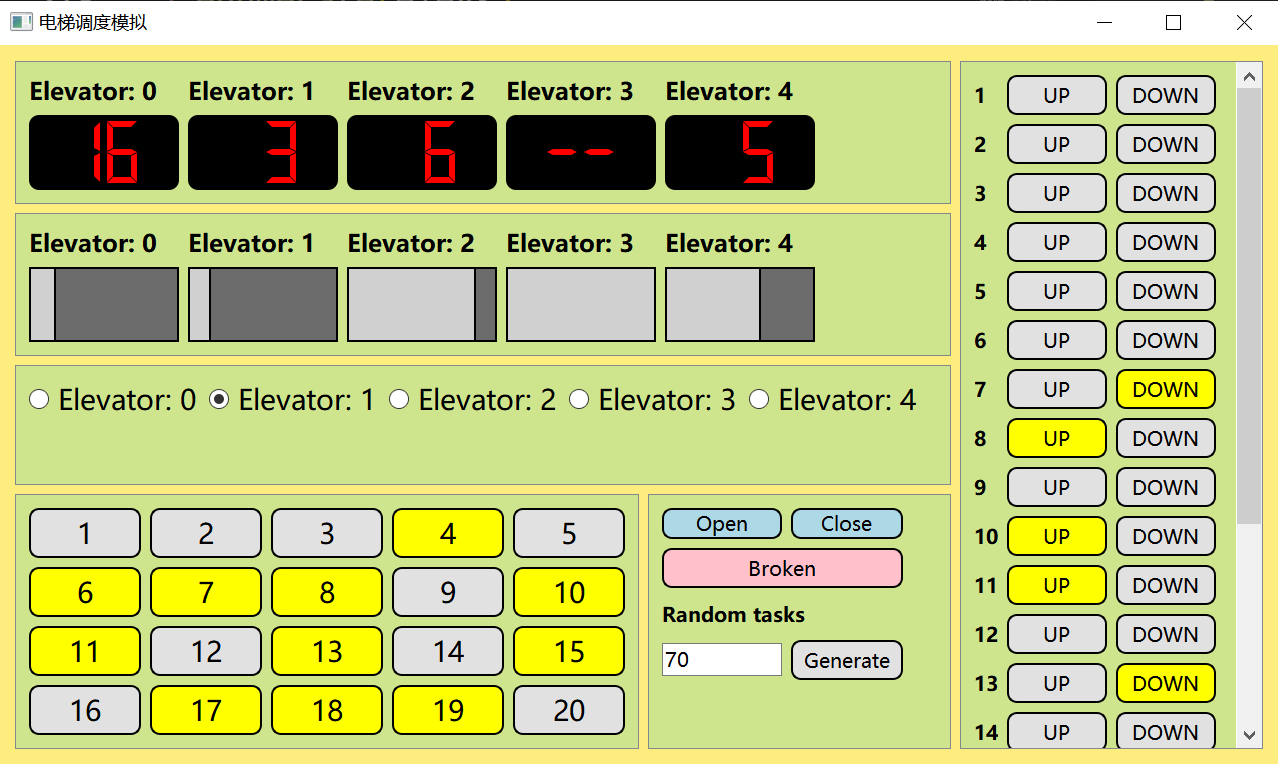
**8.2 楼层到达开关门操作**



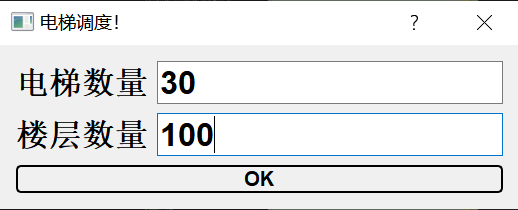
**8.3. 电梯故障设置**



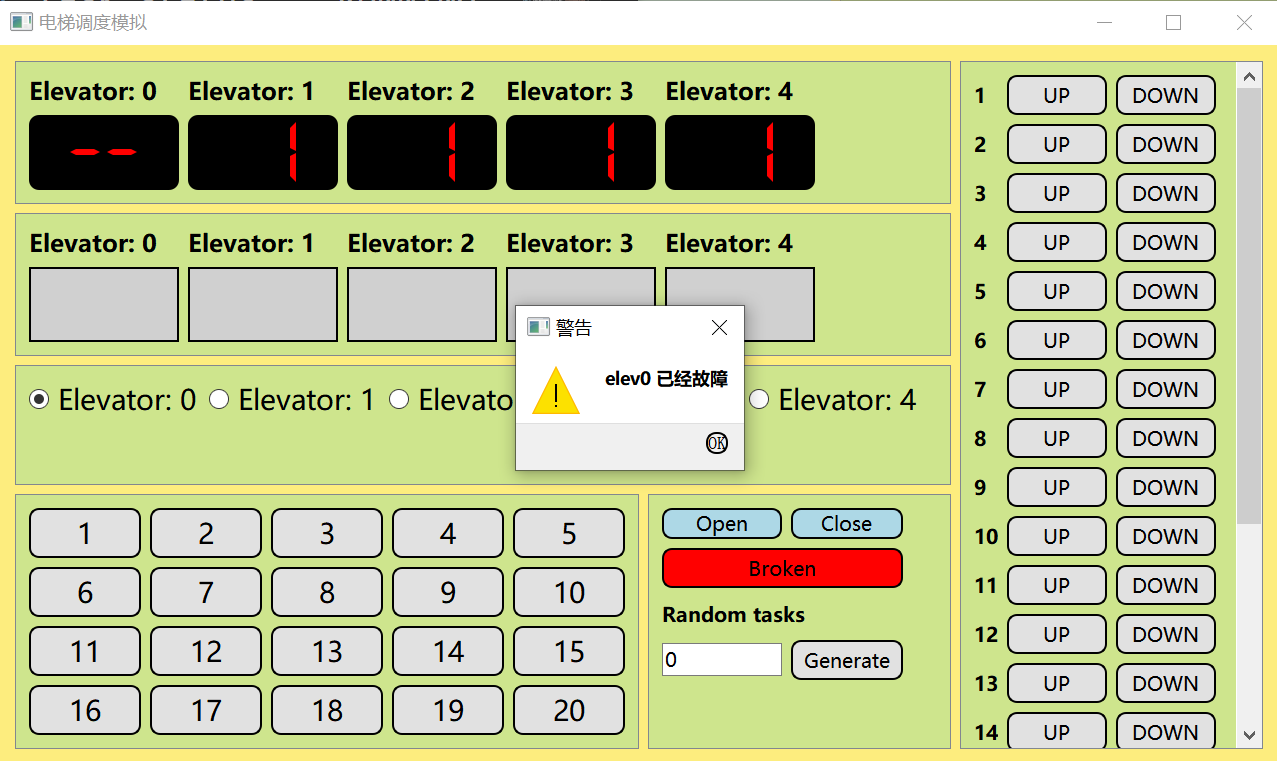
**8.4. 多部电梯同时运行，随机分配任务**



**8.5. 用户输入电梯数量与楼层数量**



**8.6. 对故障电梯内部按钮操作**



**8.7. 所有电梯故障**

