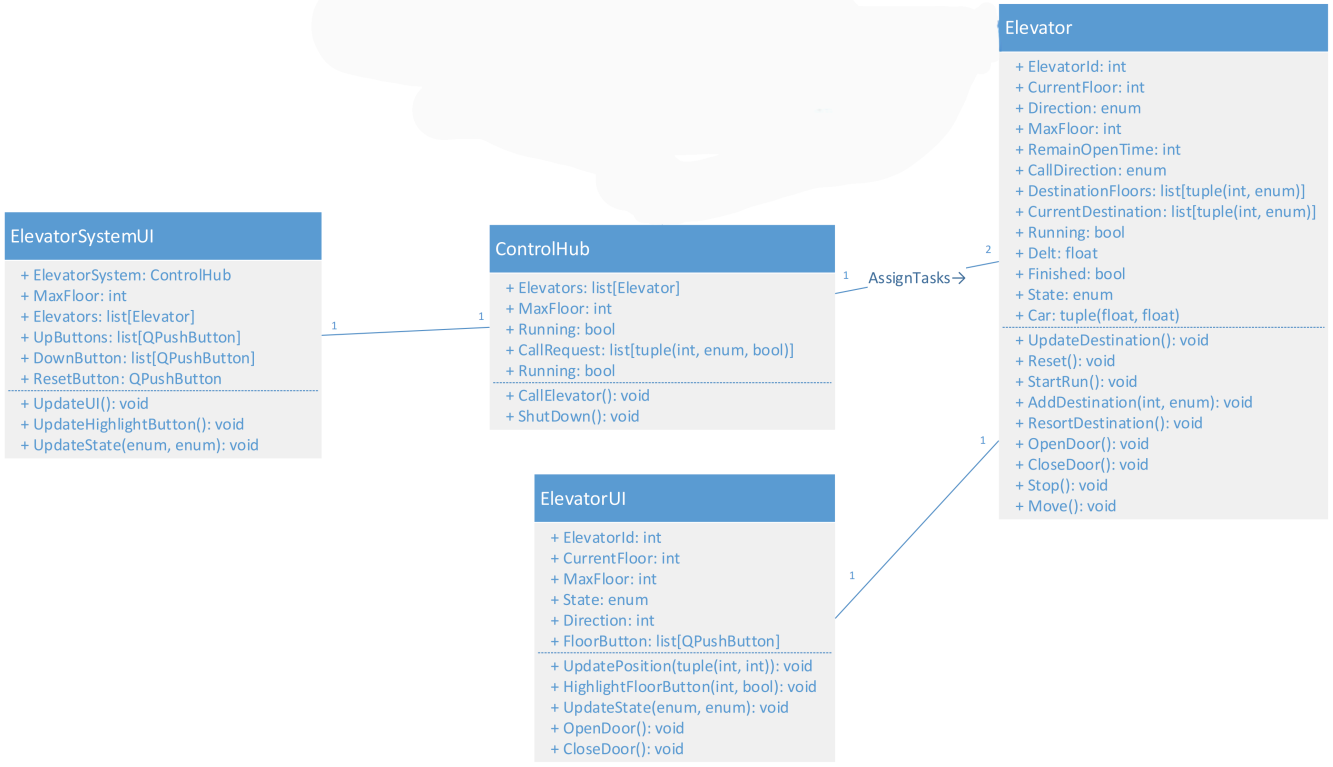
Speceficaton

Class diagram



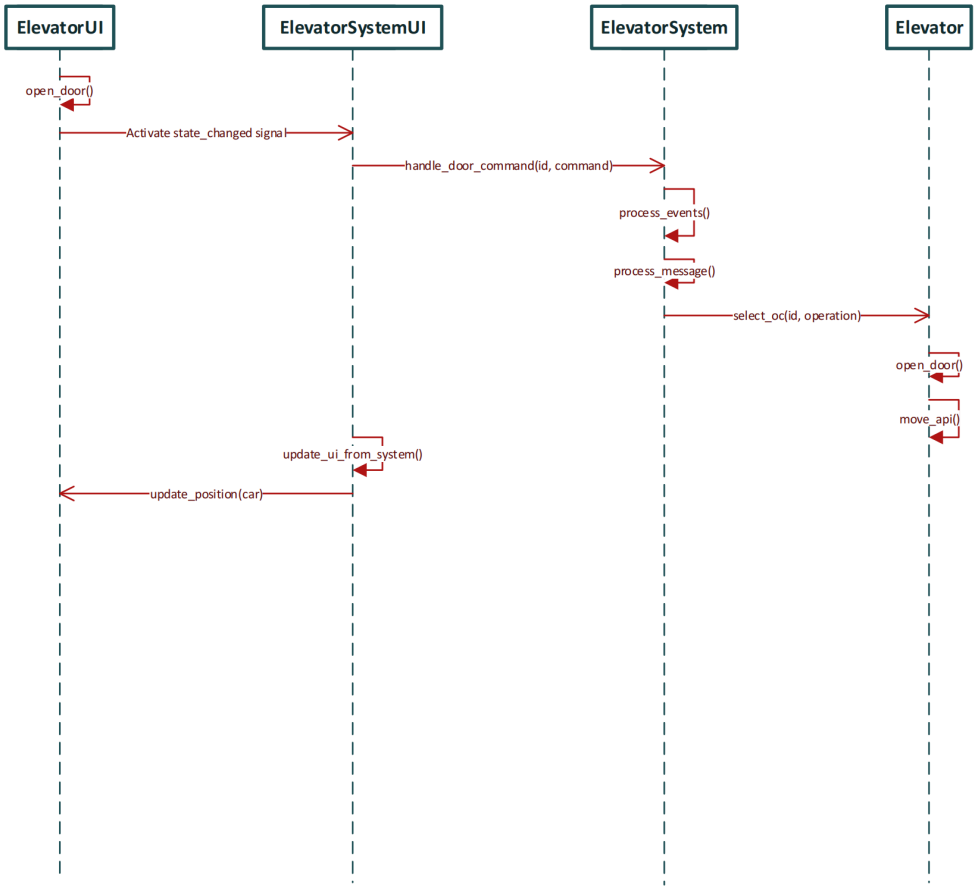
Method descriptions

Main operations Handller

S1: Operations will be handled by process_event

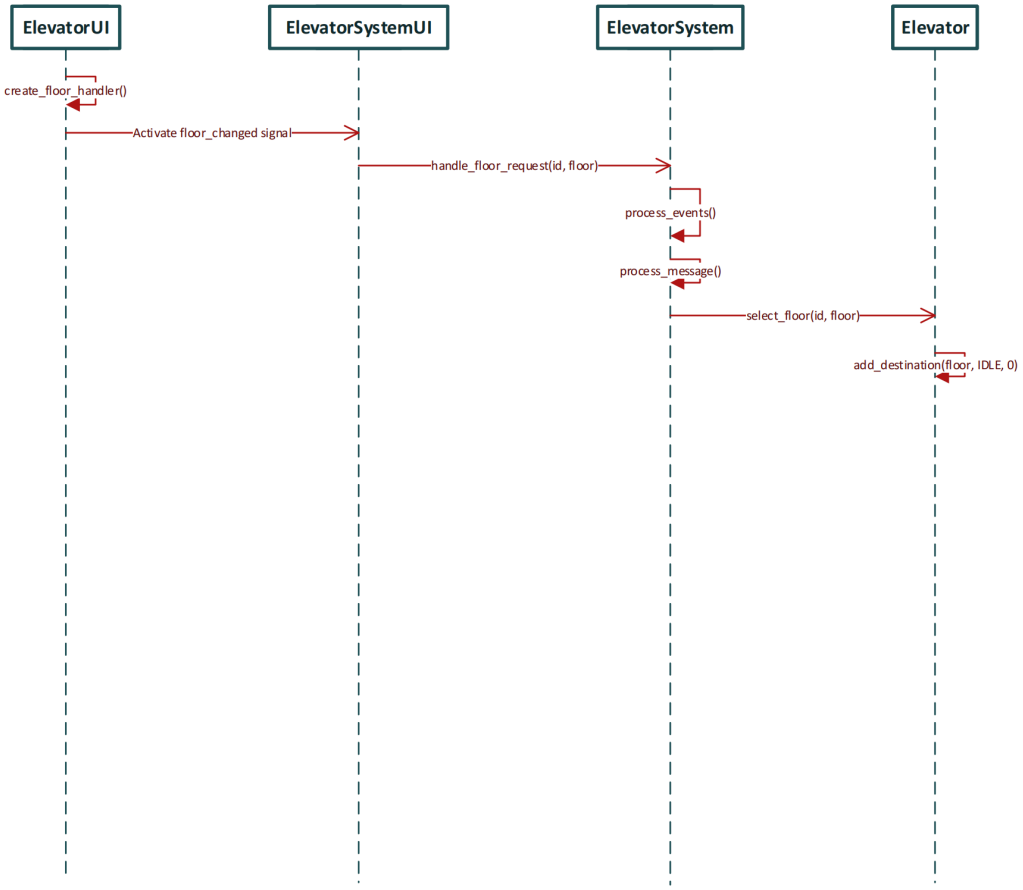
S1.0: The message processing procedure

All operations will be passed from the receivedMessage of the Zmqthread, then Process_event() will read the operation message and pass it to Process_message() to deal with the event. Different event will call different function, which will be explained below step by step.



S1.1 Open door & S1.2 Close door

- Since close door is actually the same process and only modify the two open_door() funciton, I do not show two repeated images.
- First, push open button. The button connects with function open_door().
- open_door() will emit a signal called state_changed.
- Signal state_changed is connected with function handle_door_command() in ElevatorSystemUI.
- Function handle_door_command() assign a value to ElevatorSystem.zmqThread.receivedMessage, which is assigned to ElevatorSystem.serverMessage.
- Function process_events() will run every 1ms, and it called function process_message().
- Function process_message() will reveice serverMessage "open_door# /close_door# " and call select_oc(), which finally called function open_door() in elevator and change its ElevatorState.
- Elevator has an always running funciton called move_api(), which send message to the zmqThread.
- Function update_ui_from_system() will run every 100ms, and it called update_position() in ElevatorUI to display the door open.



S1.3 Select floor

We can easily find that the processes of UI sending messages are almost the same.

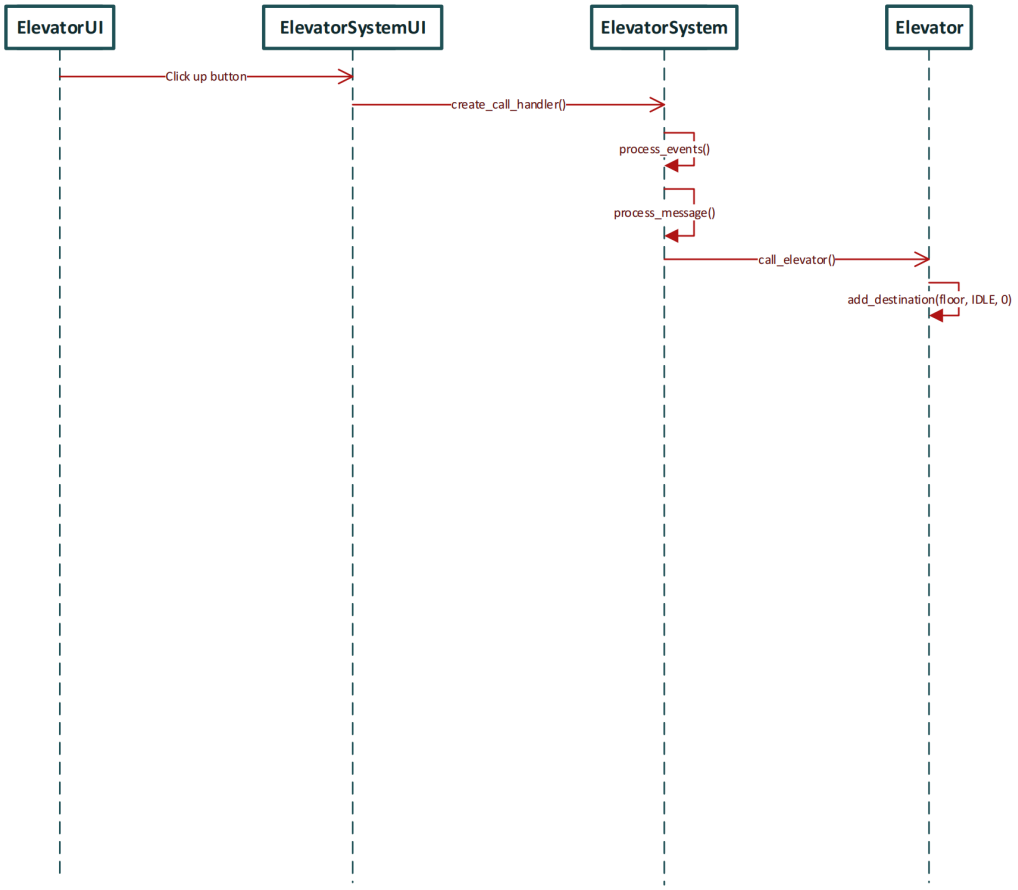
1. First select the floor number in elevator and that floor button is connected to funtion create_floor_handler(). create_floor_handler() will emit a signal floor_changed.

2. Signal floor_changed is connected with function handle_floor_request() in ElevatorSystemUI.

3. Function handle_floor_request() assign a value to ElevatorSystem.zmqThread.receivedMessage, which is assigned to ElevatorSystem.serverMessage.

4. Function process_events() will run every 1ms, and it called function process_message(), which will process the message from serverMessage.

5. Function process_message() called with call select_floor() after receiving the select_floor@ # message, which finally called function add_destination() in elevator and add your destination to the list.



S1.4 Call up and S1.5 Call down

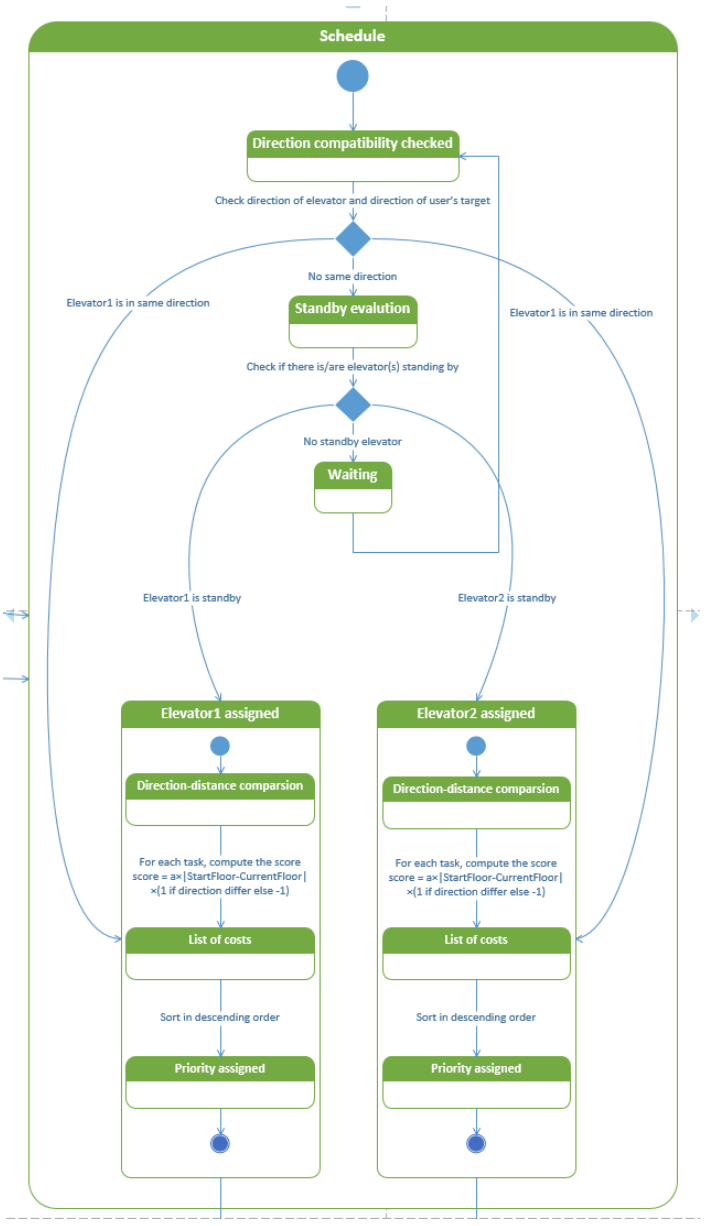
Similarly, we only represent call up since call down is the same with call up.

1. Click the up button will trigger ElevatorSystemUI to run the funciton create_call_handler().
2. This funciton assign a value to ElevatorSystem.zmqThread.receivedMessage, which is assigned to ElevatorSystem.serverMessage.
3. Function process_events() will run every 1ms, and it called function process_message().
4. Function process_message() will append the call_request list after receving the message "call_up@2" or other call requests, funciton call_elevator() will iteration the call_request list and assign the request to elevators:

- call_elevator() will call add_destination(), and the externel schedule is finished.

Schedule

S2.1 Schedule for externel call



This part is implemented as function call_elevator() in ElevatorSystem. One thread will always run during the system running. Once the list call_requests is not empty, it will start to schedule.

1. The function will first traverse the self.call_requests. If there is call_request that is not valid(explained later), remove it.

2. After getting the call_requests that are valid, check each call_request in it, if the destination can be found in elevators' destinations, ignore it and continue to check the next call_request. If no valid call_requests, noting will be done.

3. For each call_request that is valid in the list, our system will check two elevator's states, and if it can be scheduled, add it to our choice list. Our schedule method are as follows:

    - a request can be scheduled if and only if there exist at least one elevator that is idle or the elevator can enable it to take a ride-sharing.
    - elevator that can enable the request take a ride-sharing will have higher priority than idle elevator.
    - to give better experience, we add a distance to the priority, less distance + high state score will eventually decide the score.

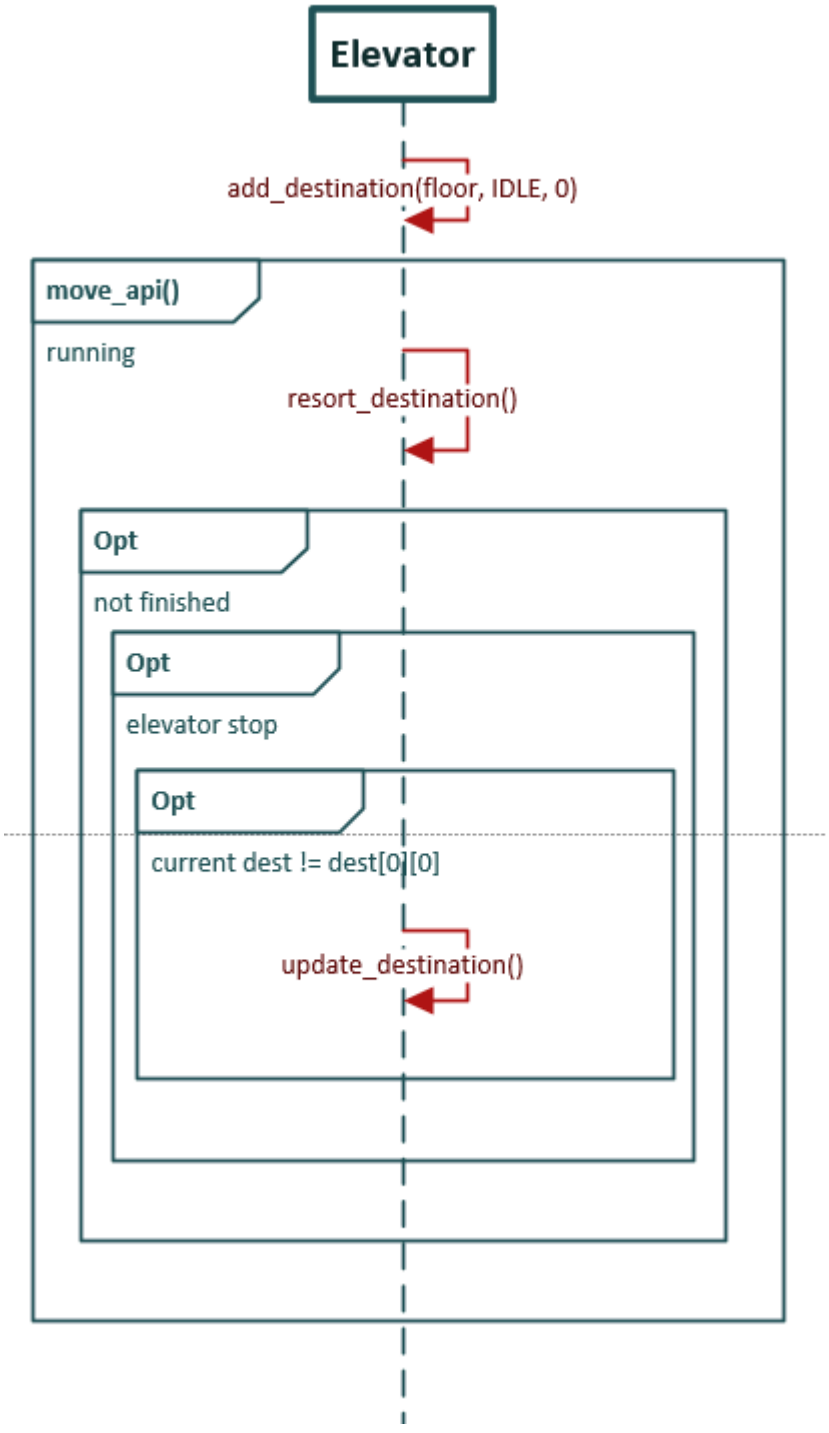
4. ride-sharing means that:

    - the elevator is moving to the request's floor.
    - The elevator is moving with the same direction with the request and their distance is larger than 1.
    - the elevator is not moving but is doing door operations and has the same direction with the call.

5. If the choice list is not empty, choose the elevator with the highest score, then call add_destination of the chosen elevator to add the request to its destination list. And then make current request's valid mark to false, then it will be invalid. In other words, request that has been scheduled is invalid.

6. If the choice list is empty, continue and do not invalidate the request.

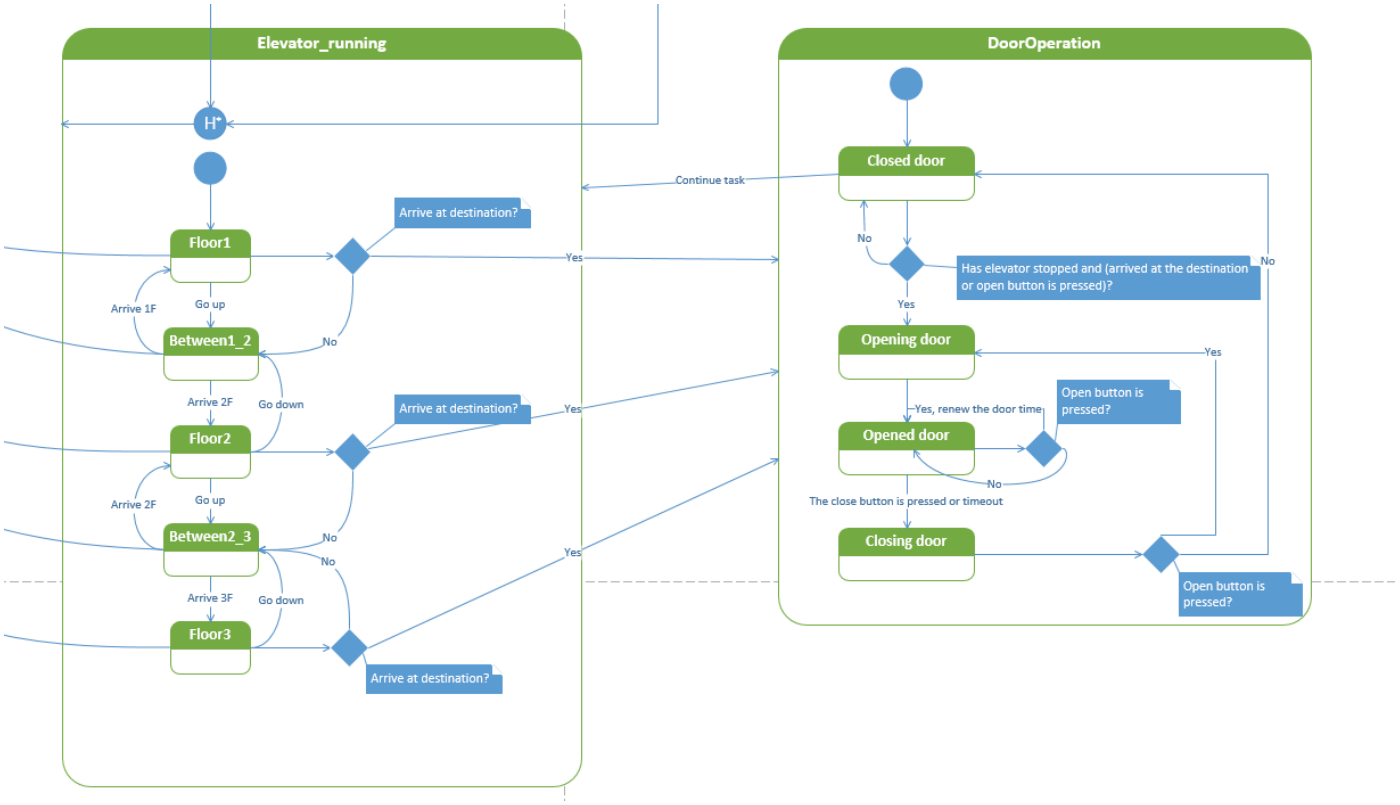
S2.2 Schedule for internel call



- As shown in the image, after function add_destination() is called, destination_floors list is not empty, then the internel schedule is working.
- Funtion move_api() won't stop if the program is running. Inside its loop, it will first call function resort_destination(which use the same scheduling method with call_elevator()) to decide which destination to choose and then move.
- When the elevator is moving, destination won't be updated. If elevator is stopped from moving, it will open the door and call update_destination() to

remove the request that has the same floor number and same direction will its current direction(if the destination comes from selecting floor, the direction will not be considered). from its destinations floors.

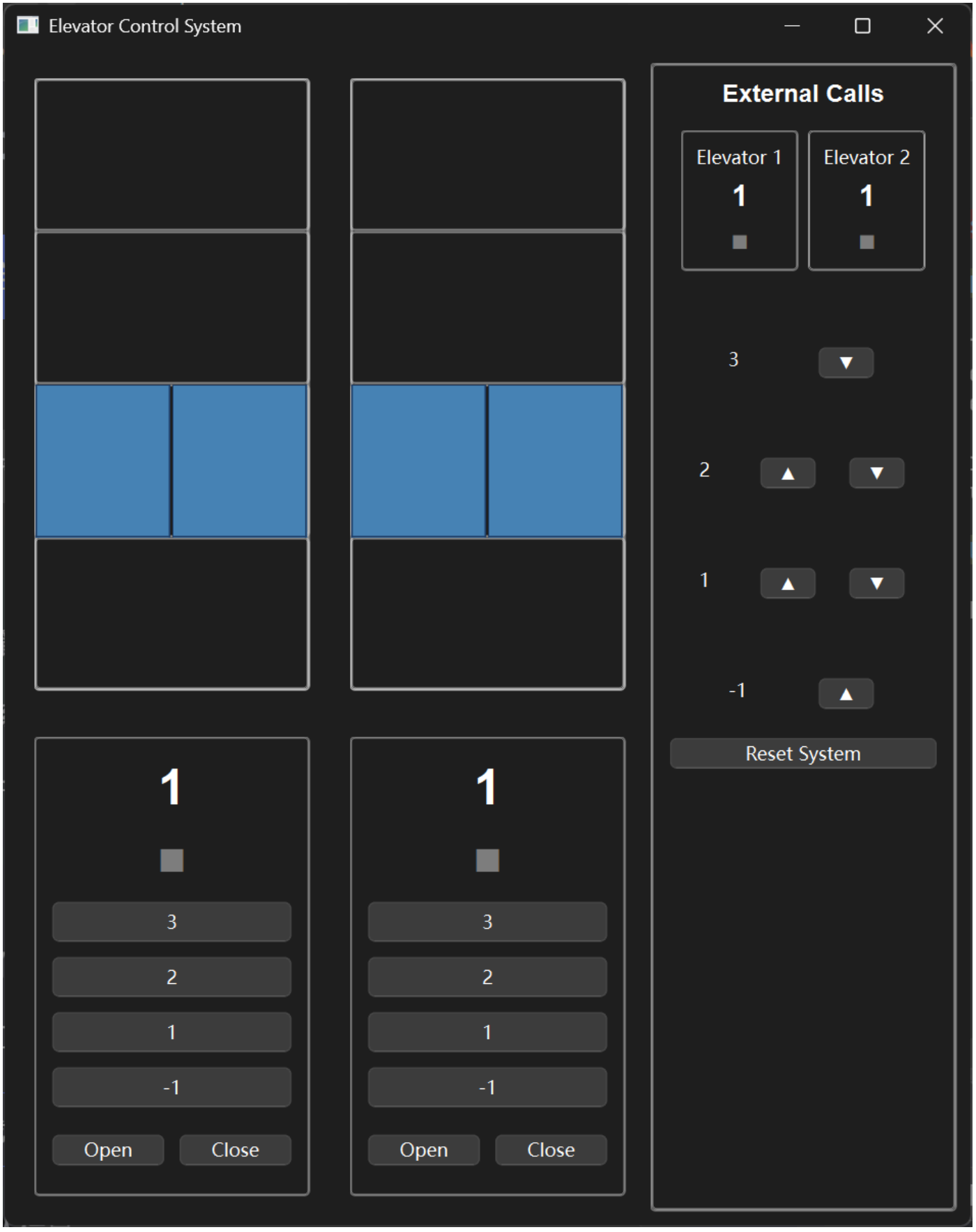
Safety



S3.1 Safety

- As shown in the image, elevator can arrive and open door only when they are in legal states, which is four integer floors.
- Also, elevator should stop when door open. If elevator is moving, the door won't open.
- During the door is open, if passengers press the open button, it will renew the door open time. Pressing close button or wait for a certain time, door will automatically close.

UI



S4.1 UI show

As shown in the image, all states of elevator system will be displsyed on this UI. Since the update logic is mentioned in main operations, this part is to show the visualization of the system.

The whole UI is composed of 3 parts, car, internel ui and externel ui, which is the left-up one, left-down one and the right one. In car, elevator is shown as 2 blue blocks, which indicates the state of door. If the door is open, two blocks will move to the right side and left side separately.

In the internel ui, we have floor buttons and open/close buttons. Pressing it will trigger functions dealing with each request, and they are mentioned in S1.1 to S1.3. Above the buttons shows the location and moving direction of elevator, which will be the same with motions in car and those in ecternel ui.

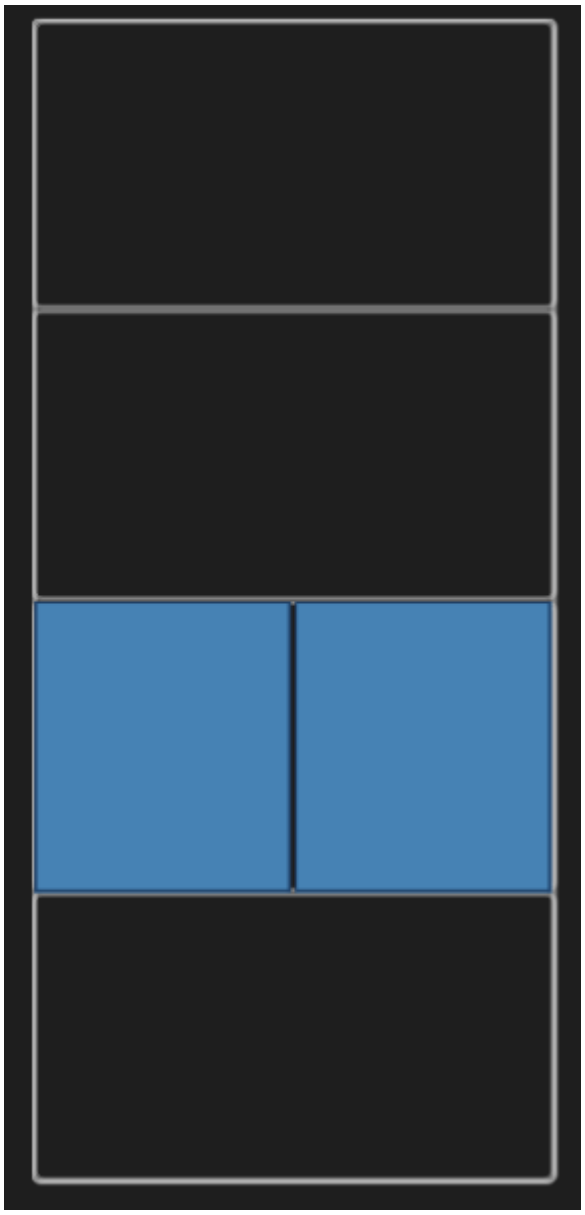
In externel ui, the location and moving direction of elevator is also shown in the top. Below that is the up and down buttons in floors, and pressing them also triggers their functions, which is mentioned in S1.4 and S1.5.

S4.2 ElevatorUI

S4.2.1 The class ElevatorUI will draw the initial graph of the internal UI:

- the buttons of the internal floor are stored in self.floor_buttons, which is a dictionary for 0, 1, 2, 3. And the item of the floor is a QPushButton, Toggled by create_floor_handler()
- the buttons of door operations are stored in self.open_btn and self.close_btn, which are QPushButton and toggled by cread_door_handler()

S4.2.2 and the graph of the shaft of current elevator:



- The car consists of 2 doors: self.left_door and self.right_door, which are QLabel.

S4.3 ElevatorSystemUI

S4.3.1 floor state display

draw floor and direction display in self.elevator_i_floor3 and self.elevator_i_direction3 for i = 1,2 for two elevators.

S4.3.2 call buttons

there are call buttons under the display:

- self.up_buttons and self.down_buttons: dictory from floor(0,1,2,3) two the button
- up_buttons[f] will have items only if f=0,1,2
- down_buttons[f] will have items only if f=1,2,3

S4.3.3: the update of elevator system

- a QTimer() self.update_timer will call update_ui_from_system every 20ms
- hihglight button calls
- update the elevator door location and opening status.
- handle click event and pass message to receivedMessage