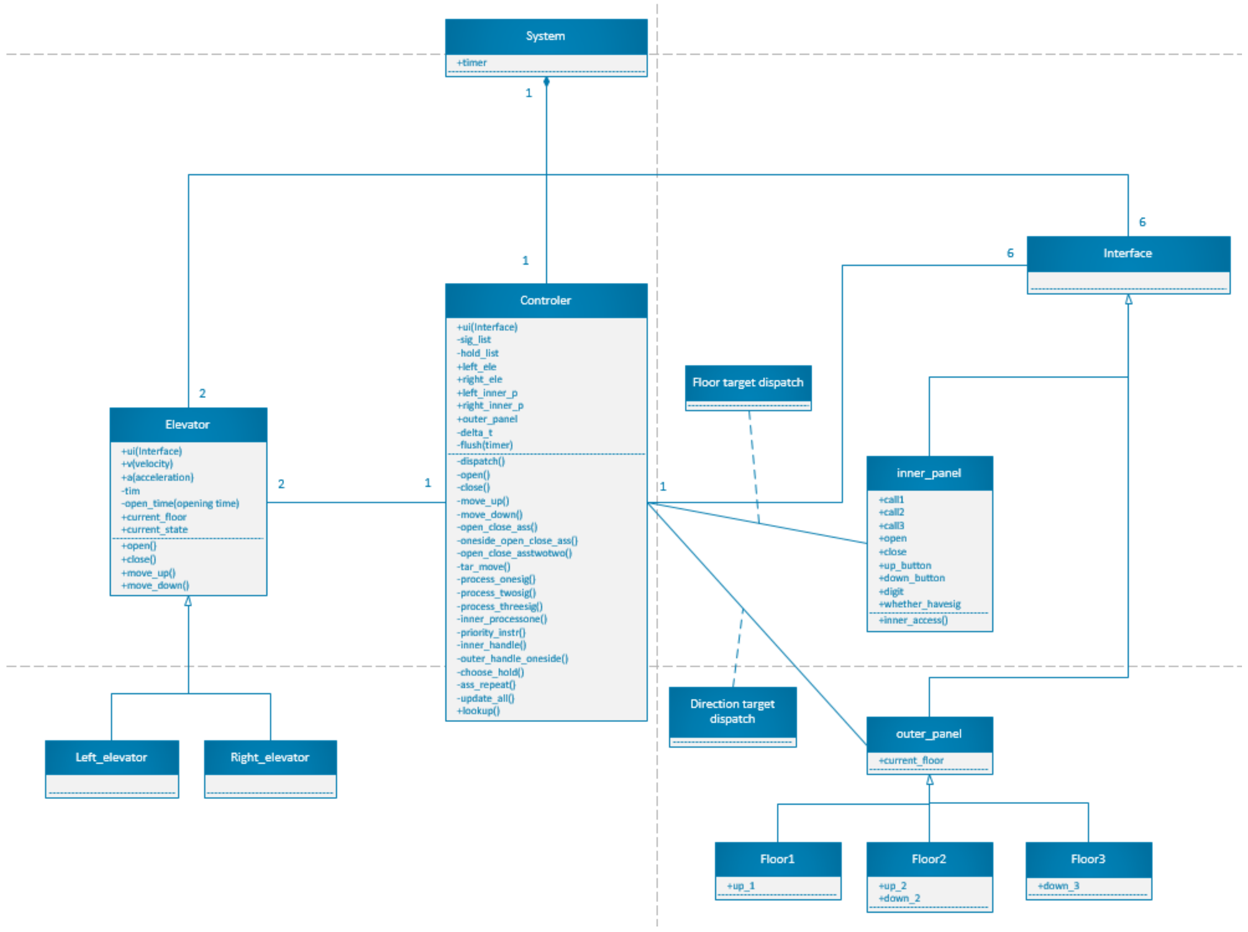
Specifications

System Architecture



Software Specification

S0 Introduction to Core Data Structure

In the controller, there are two array which is the key role in the whole system and they are used by almost all action.

They are called **sig_list** and **hold_list** respectively.

The sig_list is a 14 length array which maintain all signals in the whole system to show whether they are active. Equal to 1 means it's being active, equal to 0 means it isn't being active.(14 signals correspond to 14 buttons in the ui part)

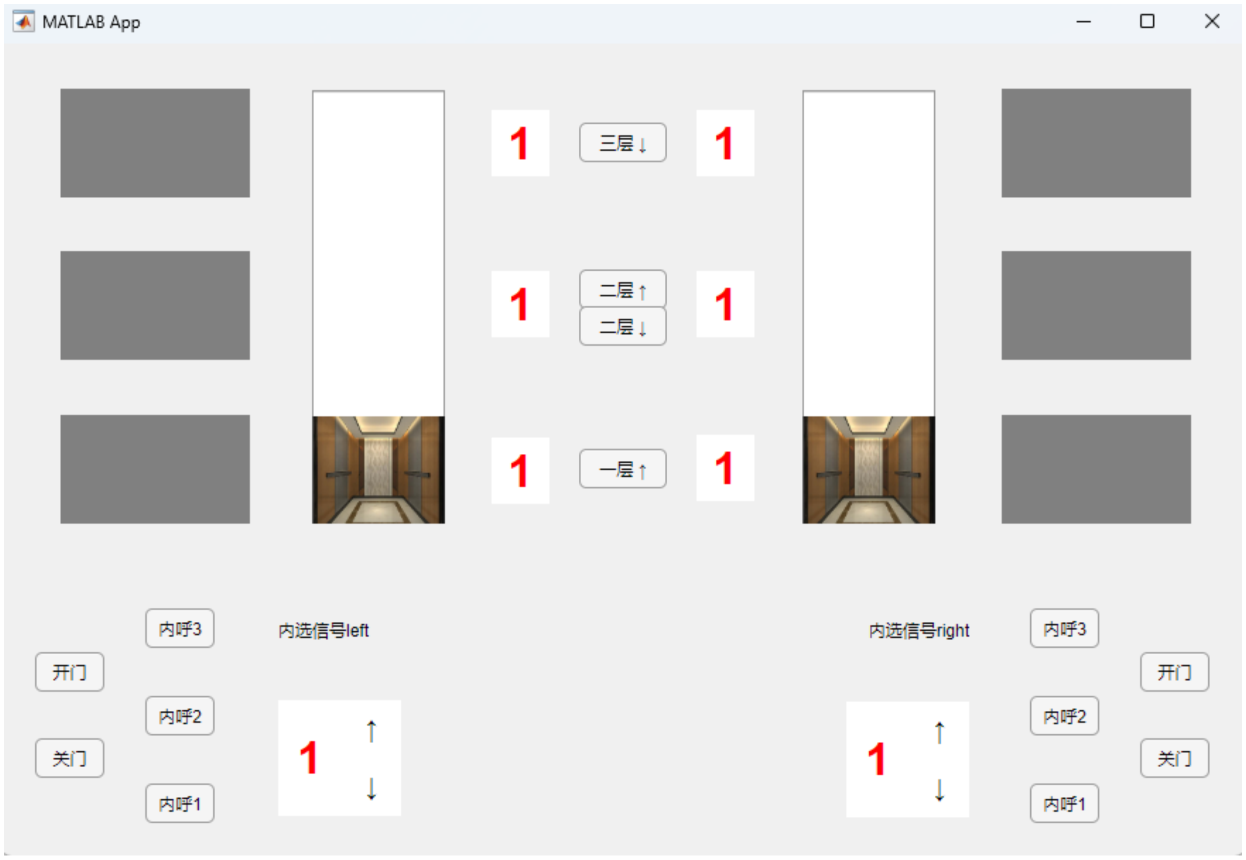
The hold_list is a 14 length array which represent whether a signal has been handled by one elevator. Equal to -1 means it's not being handled by any elevator, equal to 0 means it's being handled by left elevator, equal to 1 means it's being handled by right elevator.

Also we define the **states** of elevators as five types :

1. stop -> represent by "0" 2. uping -> represent by "1" 3. downing-> represent by "2"

4. opening -> represent by "3" 5. closing -> represent by "4" 3. open_MAX -> represent by "5"

S1 Elevator UI

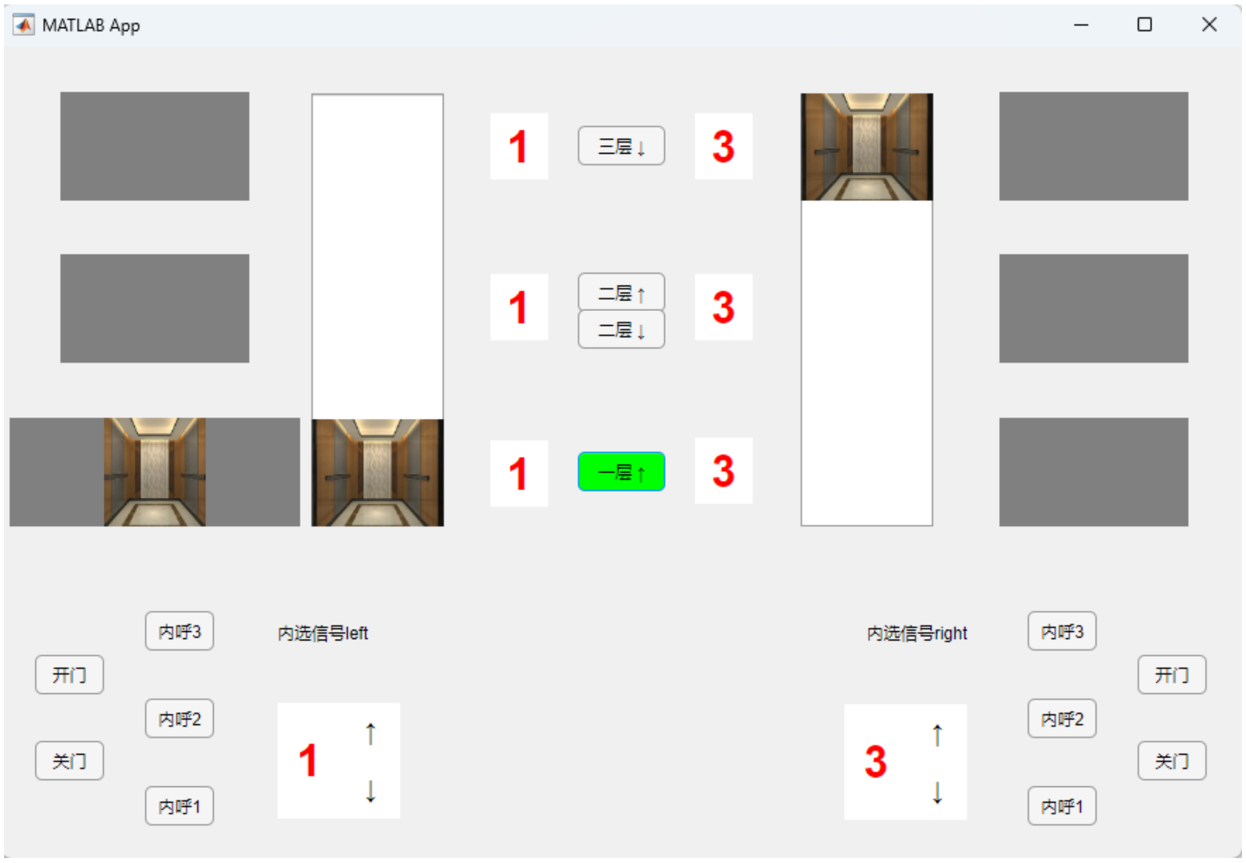
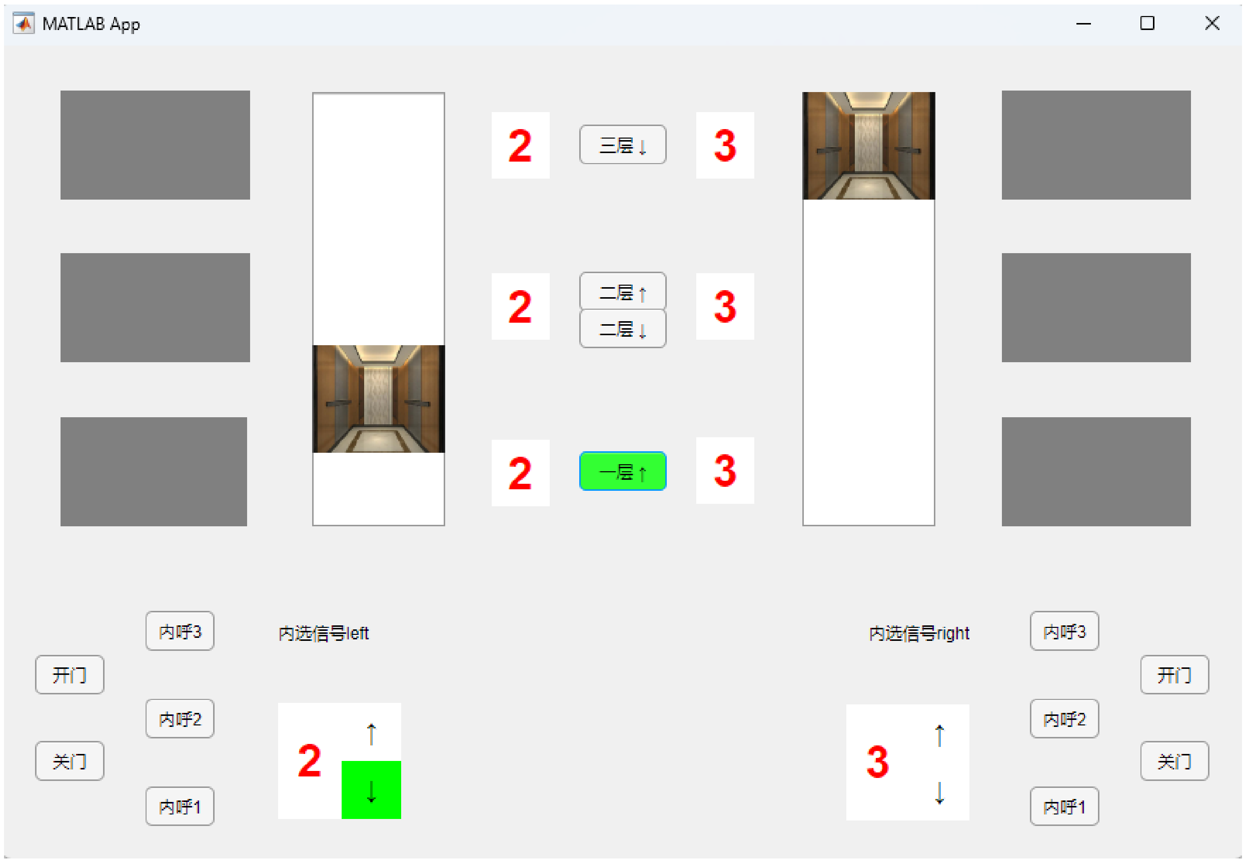


S1.1 Call elevator by pushing outer buttons.

By pressing outer panel, that is "一层↑", "二层↓", "二层↑", "三层↓" four buttons, user can call an elevator to handle them. The corresponding serial number in sig_list is 1,2,3,4.

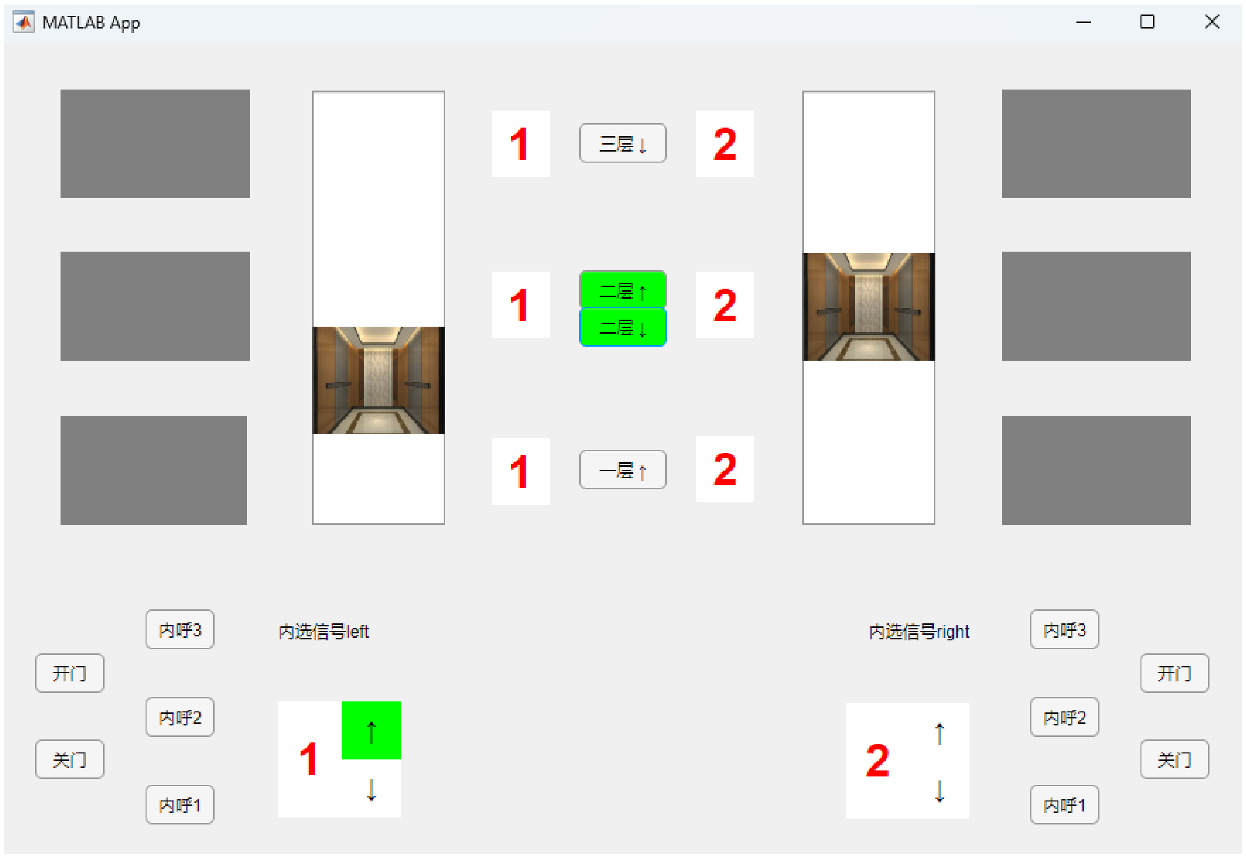
S1.1.1 Press "一层↑" button

Process will call controller.schedule(4) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list. Controller will detect it and call process_onesig() with current state of elevator system to handle it. Then open_close_ass() function will assess whether have an elevator has been in the target floor. If yes, call open() function to open correspond elevator directly. If no, call dispatch() function attempt to schedule an elevator to handle it. Then if schedule fails, just return and wait next cycle of timer. If schedule successfully, call tar_move() to move correspond elevator.


MATLAB App
1
三层↓
3
1
二层↑
二层↓
3
1
一层↑
3
内呼3
开门
内呼2
关门
内呼1
内选信号left
1
↑
↓
内选信号right
3
↑
↓
内呼3
开门
内呼2
关门
内呼1

MATLAB App
2
三层↓
3
2
二层↑
二层↓
3
2
一层↑
3
内呼3
开门
内呼2
关门
内呼1
内选信号left
2
↑
↓
内选信号right
3
↑
↓
内呼3
开门
内呼2
关门
内呼1

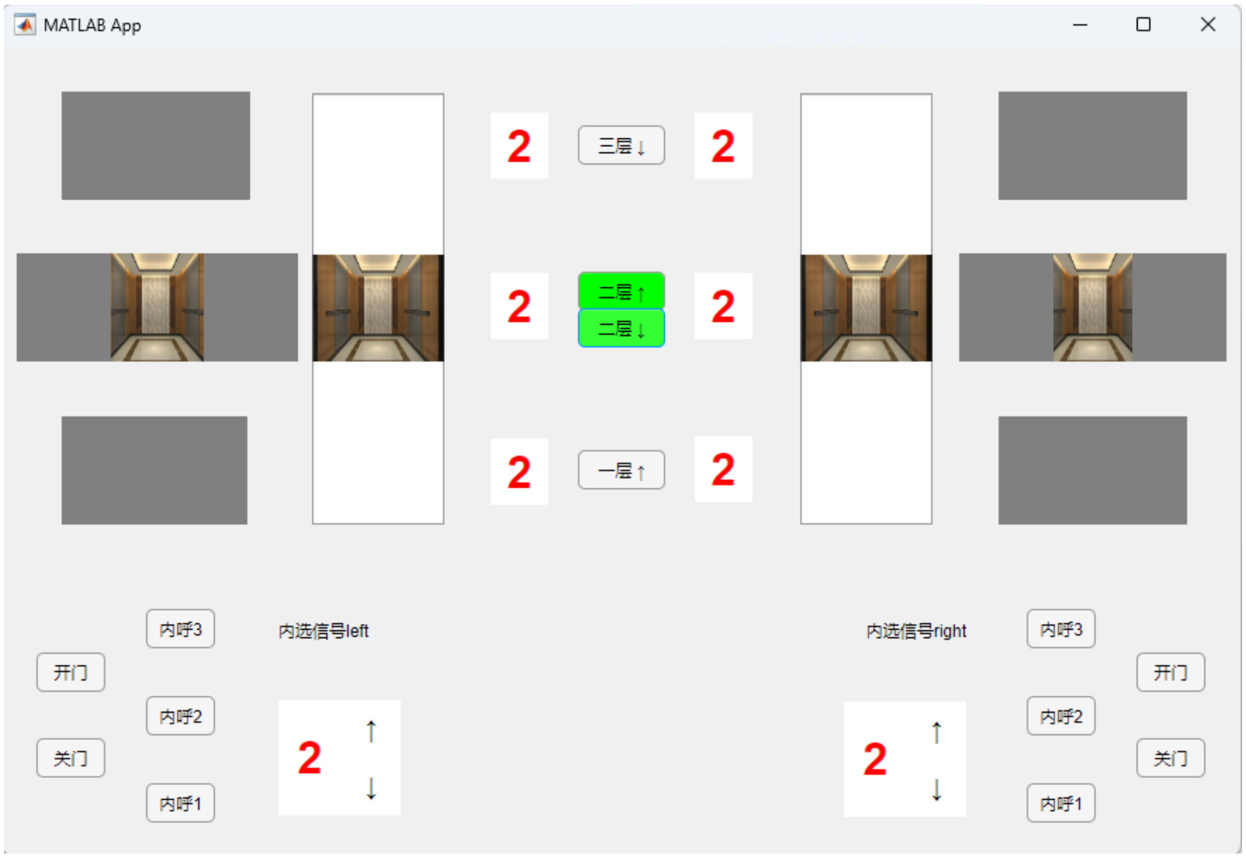
S1.1.2 Press "二层↑" button

Process will call controller.schedule(2) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list. Controller will detect it and call process_onesig() with current state of elevator system to handle it. Then open_close_ass() function will assess whether have an elevator has been in the target floor. If yes, call open() function to open correspond elevator directly. If no, call dispatch() function attempt to schedule an elevator to handle it. Then if schedule fails, just return and wait next cycle of timer. If schedule successfully, call tar_move() to move correspond elevator.



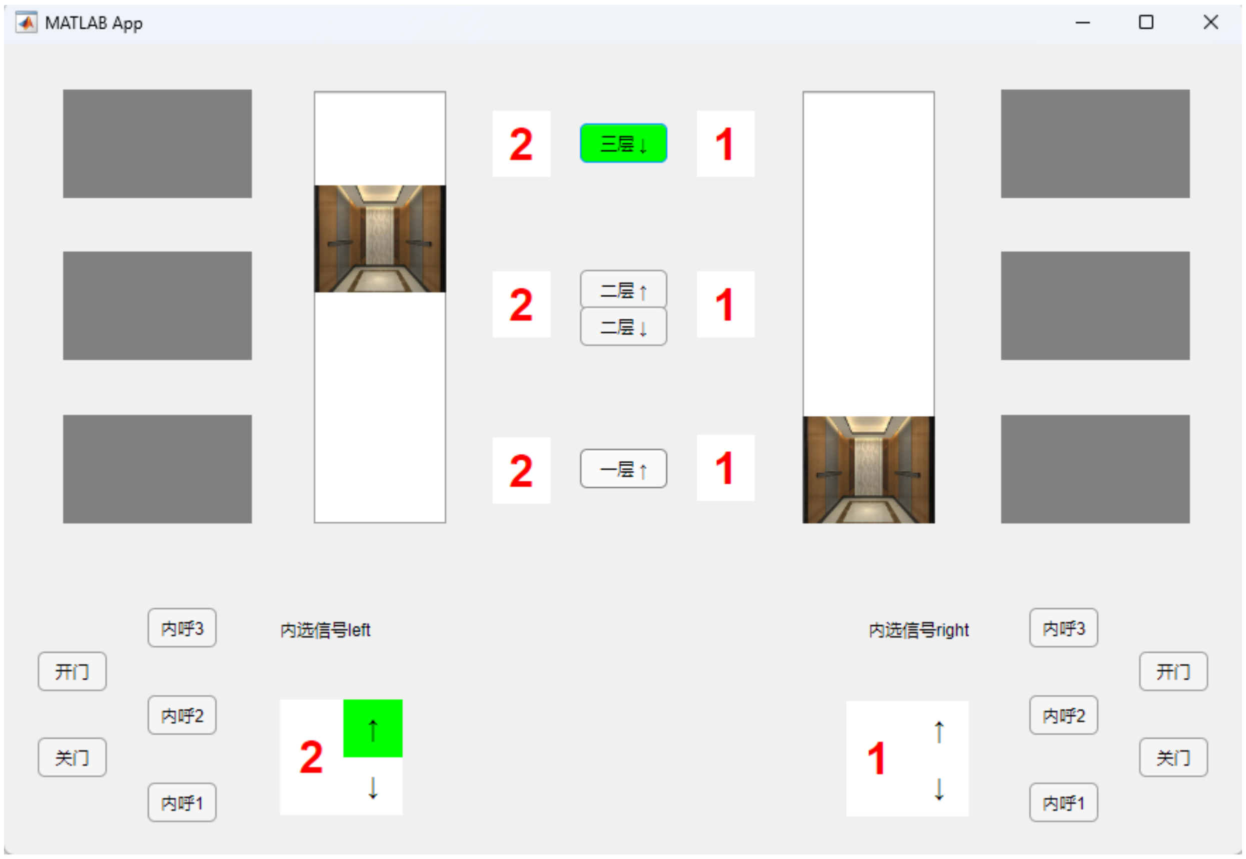
S1.1.3 Press "二层↓" button

Process will call controller.schedule(3) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list. Controller will detect it and call process_onesig() with current state of elevator system to handle it. Then open_close_ass() function will assess whether have an elevator has been in the target floor. If yes, call open() function to open correspond elevator directly. If no, call dispatch() function attempt to schedule an elevator to handle it. Then if schedule fails, just return and wait next cycle of timer. If schedule successfully, call tar_move() to move correspond elevator.



S1.1.4 Press "三层↓" button

Process will call controller.schedule(1) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list. Controller will detect it and call process_onesig() with current state of elevator system to handle it. Then open_close_ass() function will assess whether have an elevator has been in the target floor. If yes, call open() function to open correspond elevator directly. If no, call dispatch() function attempt to schedule an elevator to handle it. Then if schedule fails, just return and wait next cycle of timer. If schedule successfully, call tar_move() to move correspond elevator.



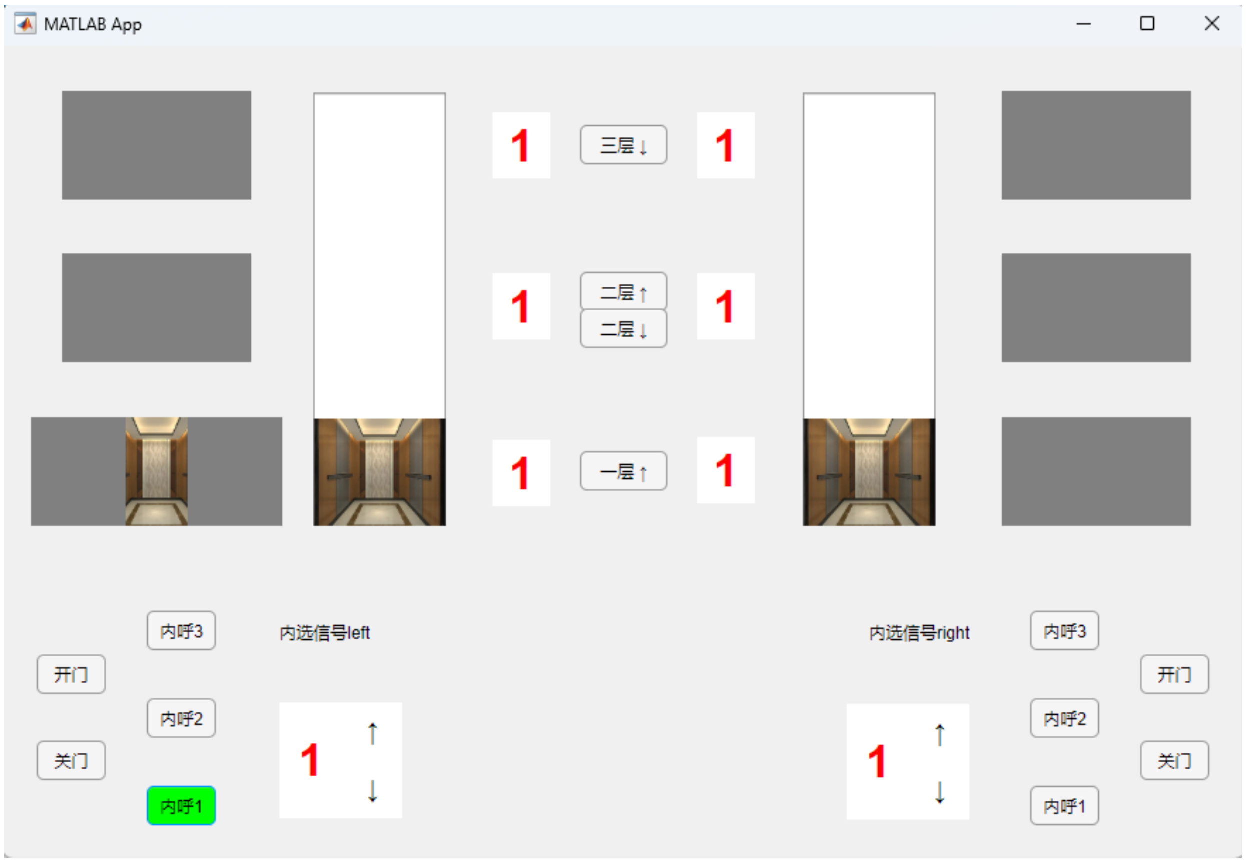
S1.2 Call elevator by pushing inner buttons.

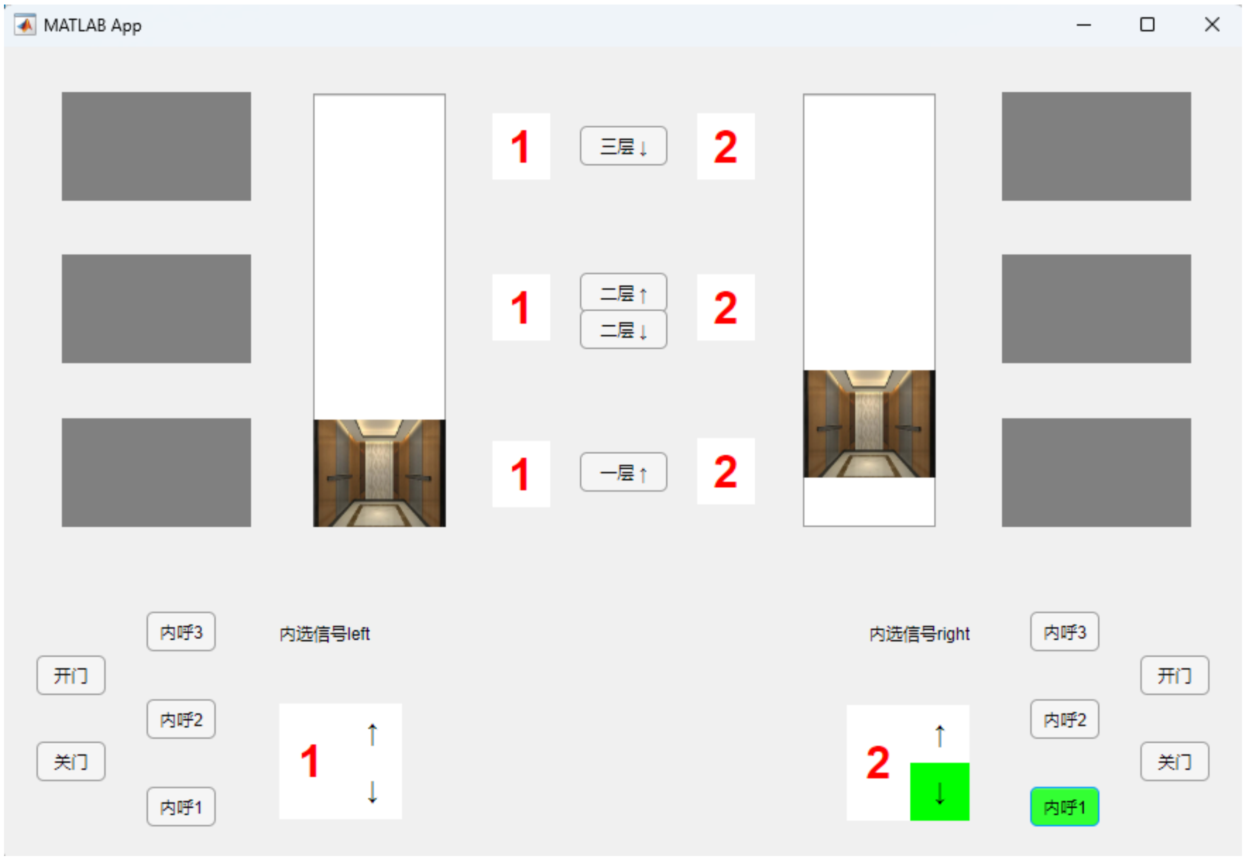
By pressing inner panel, that is "内呼1", "内呼2", "内呼3" three buttons, user can control the elevator to go to target floor. The corresponding serial number in sig_list is 7,8,9,10,11,12.

S1.2.1 Press "内呼1" for both left and right elevator

Process will call controller.schedule(9 or 12) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list.

Controller will detect it and call inner_handle() with current state of elevator system to handle it. Then assess the elevator state. If in the target floor, call open() function to open the door directly. If not, call priority_instr() function with current state of elevator system and schedule an inner instruction to run. Then call inner_processone() to handle it and update hold_list, move elevator.

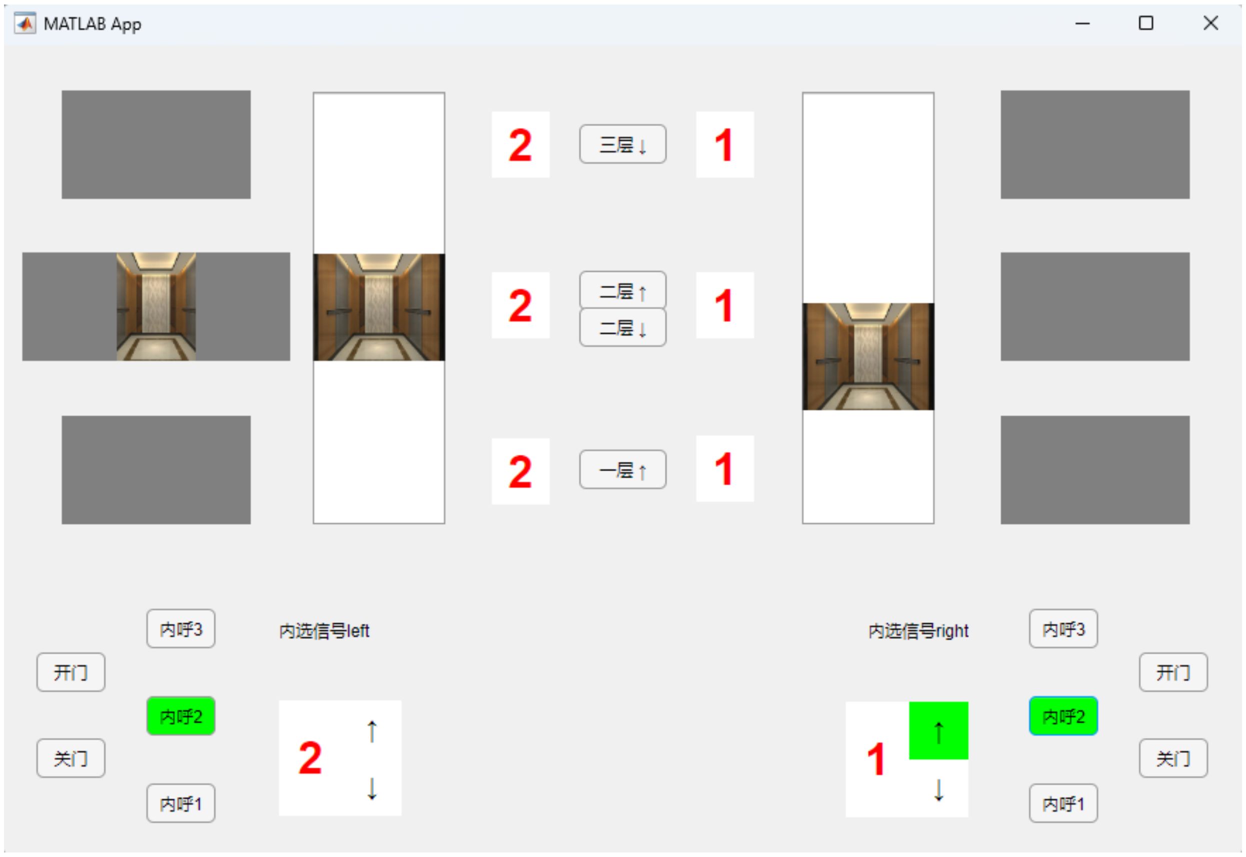

MATLAB App
1
三层↓
1
1
二层↑
二层↓
1
1
一层↑
1
内呼3
内选信号left
内选信号right
内呼3
开门
开门
内呼2
内呼2
关门
关门
内呼1
内呼1
1
↑
↓
1
↑
↓


MATLAB App
1
三层↓
2
1
二层↑
二层↓
2
1
一层↑
2
内呼3
内选信号left
内选信号right
内呼3
开门
开门
内呼2
内呼2
关门
关门
内呼1
内呼1
1
↑
↓
2
↑
↓

S1.2.2 Press "内呼2" for both left and right elevator

Process will call controller.schedule(8 or 11) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list.

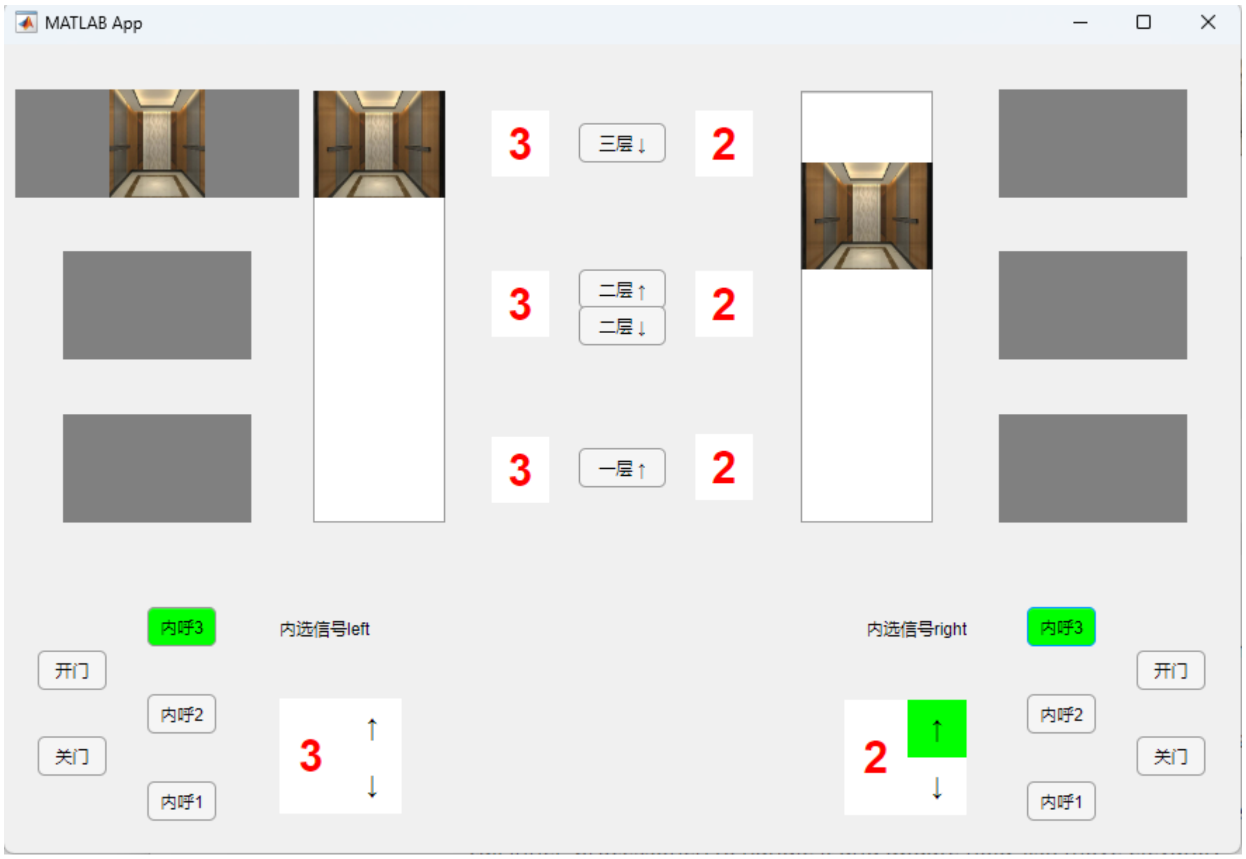
Controller will detect it and call inner_handle() with current state of elevator system to handle it. Then assess the elevator state. If in the target floor, call open() function to open the door directly. If not, call priority_instr() function with current state of elevator system and schedule an inner instruction to run. Then call inner_processone() to handle it and update hold_list, move elevator.



S1.2.3 Press "内呼3" for both left and right elevator

Process will call controller.schedule(7 or 10) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list.

Controller will detect it and call inner_handle() with current state of elevator system to handle it. Then assess the elevator state. If in the target floor, call open() function to open the door directly. If not, call priority_instr() function with current state of elevator system and schedule an inner instruction to run. Then call inner_processone() to handle it and update hold_list, move elevator.

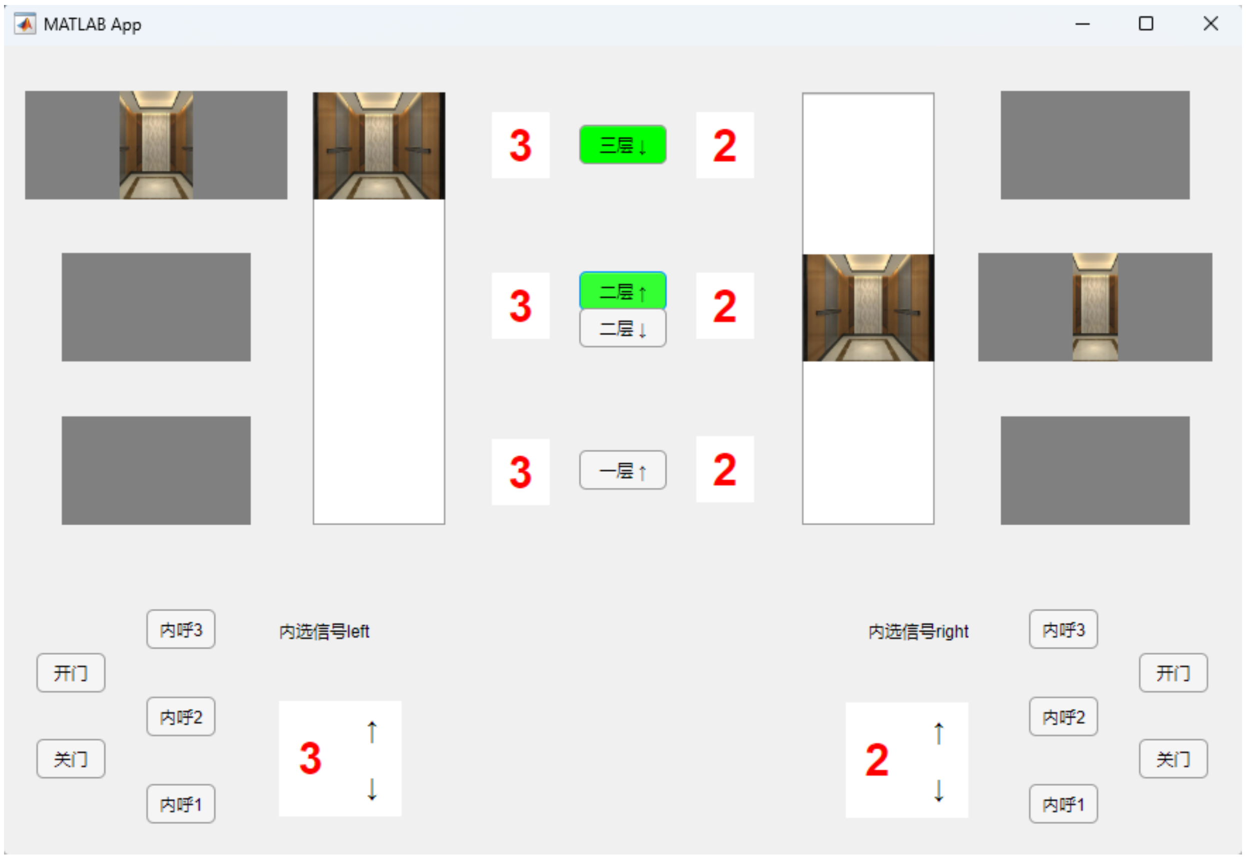


S1.3 Control elevator door

S1.3.1 Press outer button open the door

If an elevator can respond to instruction at target floor, press floor button will open the door directly.

Process will call controller.schedule(1,2,3,4) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list. Controller will detect it and call process_***sig() with current state of elevator system to handle it. Then call open() function by open_close_ass() to open the door.



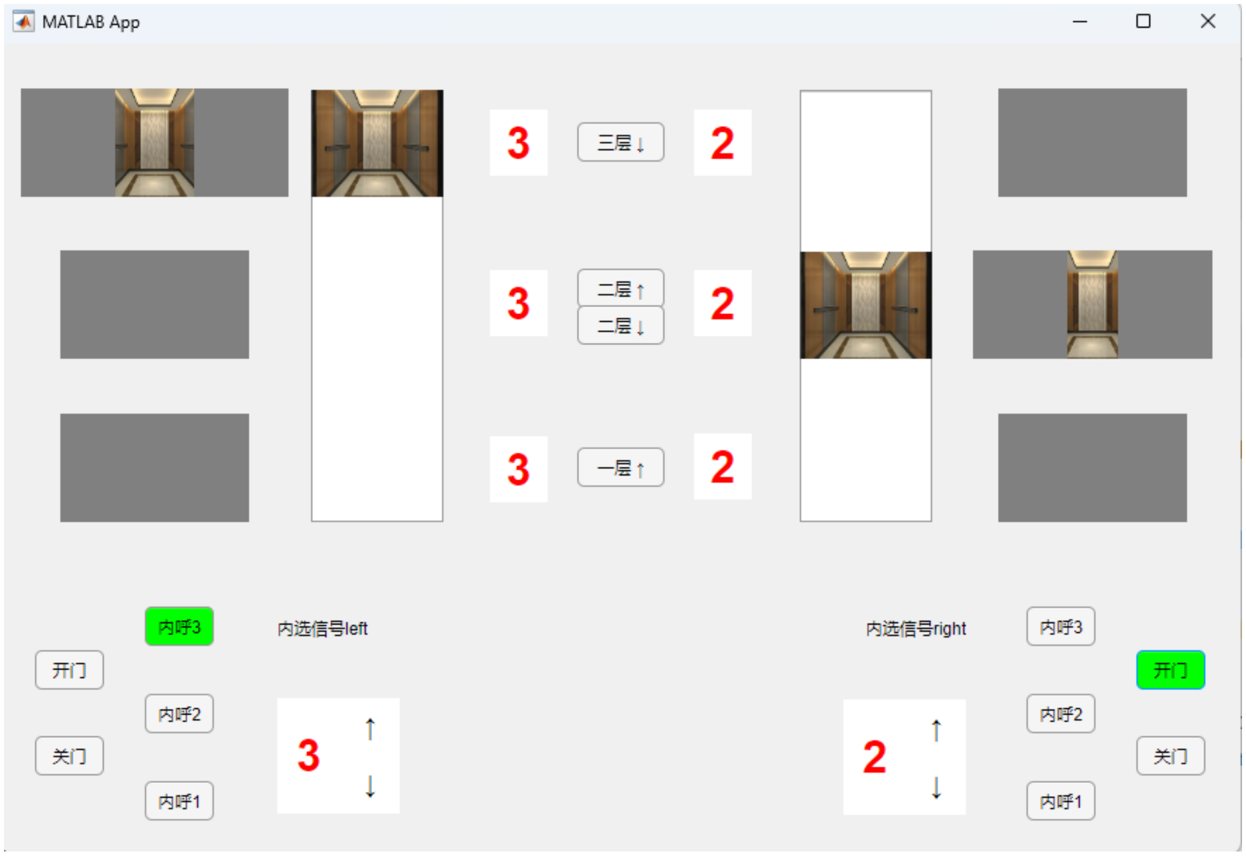
S1.3.2 Press inner button open the door

The elevator will open the door when arrive the target floor designated by "内呼*" and also press "开门" button can open the door too.

Process will call controller.schedule(5~14) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list.

If instruction is "开门", controller will detect it and call inner_handle() with current state of elevator system to handle it. Then do some state assessment and call open() to open the door.

If instruction is "内呼*" and elevator is in the target floor, controller will detect it and call inner_handle() with current state of elevator system to handle it. Then assess the elevator state and call open() function to open the door directly.

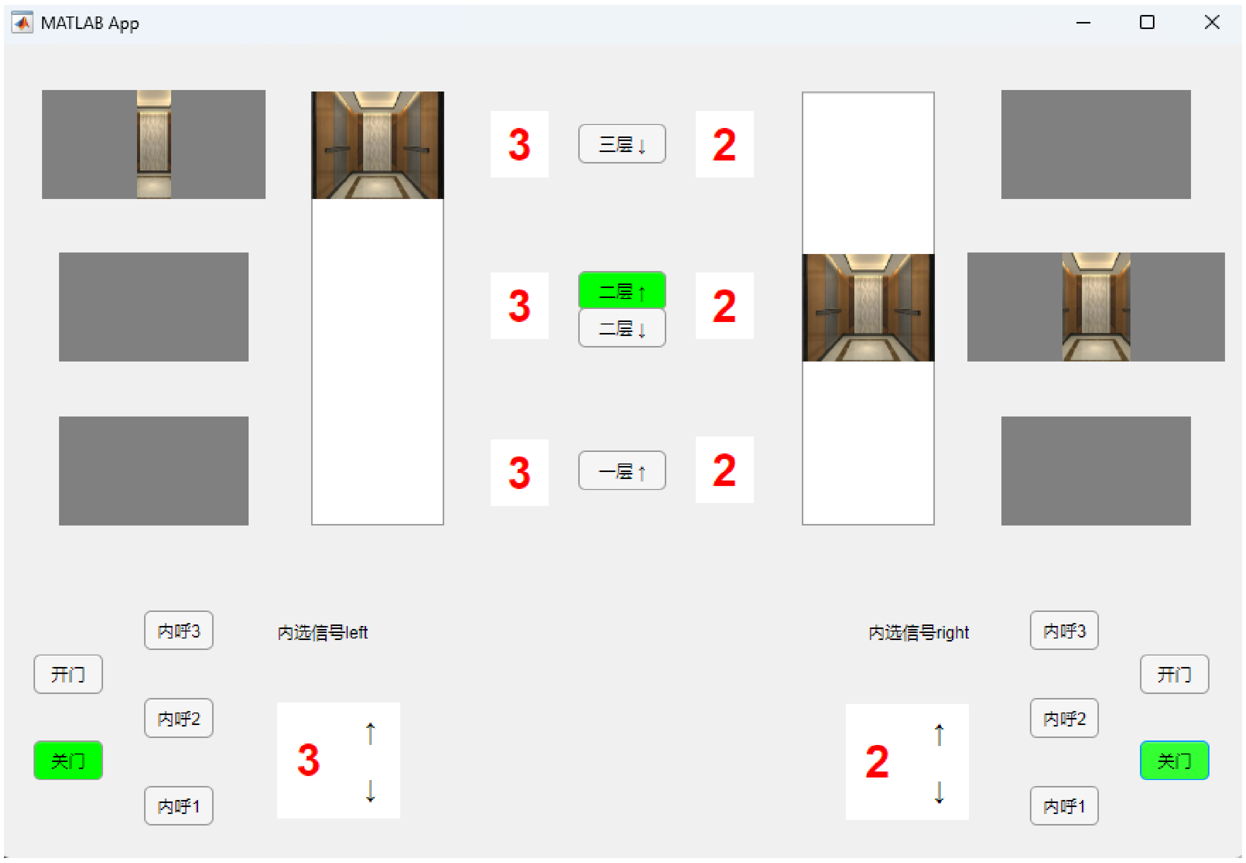


S1.3.3 Press "关门" button for both left and right elevator to close door

When elevator reach maximum open state, it will wait 5 seconds by default. Passenger can press "关门" button to close door immediately.

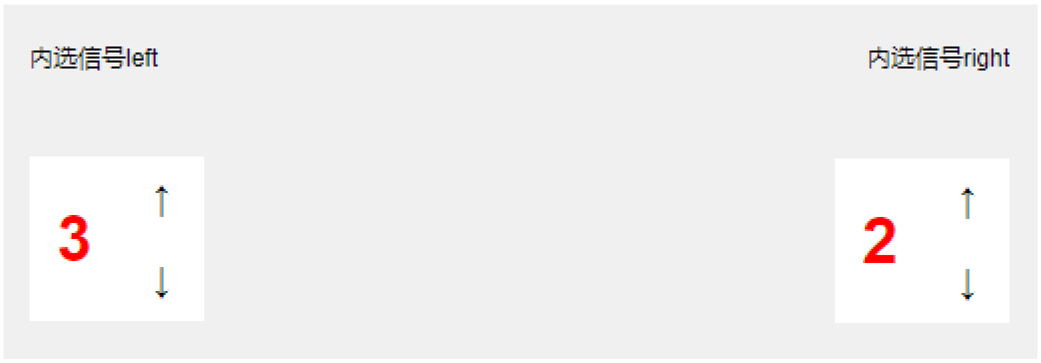
Process will call controller.schedule(6 or 14) and change color in front-end. Back-end will get the signal and maintain a list which called sig_list.

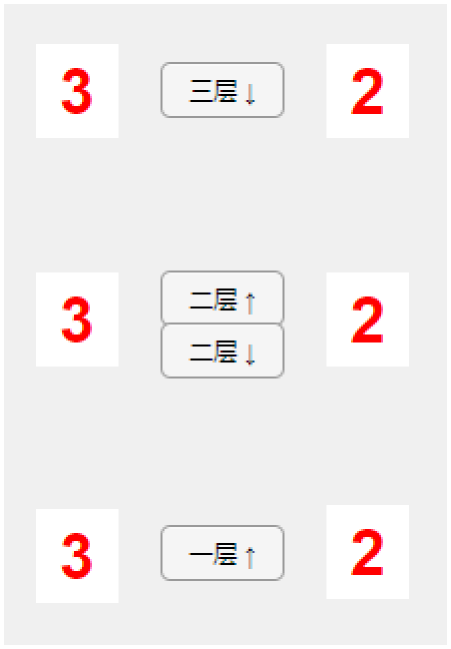
Controller will detect it and call inner_handle() with current state of elevator system to handle it. Then do some state assessment, if elevator state is 5(Maximum open state) and call close() to open the door.



S1.4 Floor number display

In each timer cycle, controller will call update_all() function and retrieve whole system state include elevator state, elevator's current floor. And display correspond number and state for both inner and outer by change Label and Background color.





S1.5 Trace display

In each timer cycle, controller will assess whether elevator will move. If an elevator will move, controller will call move_up() or move_down() function.\

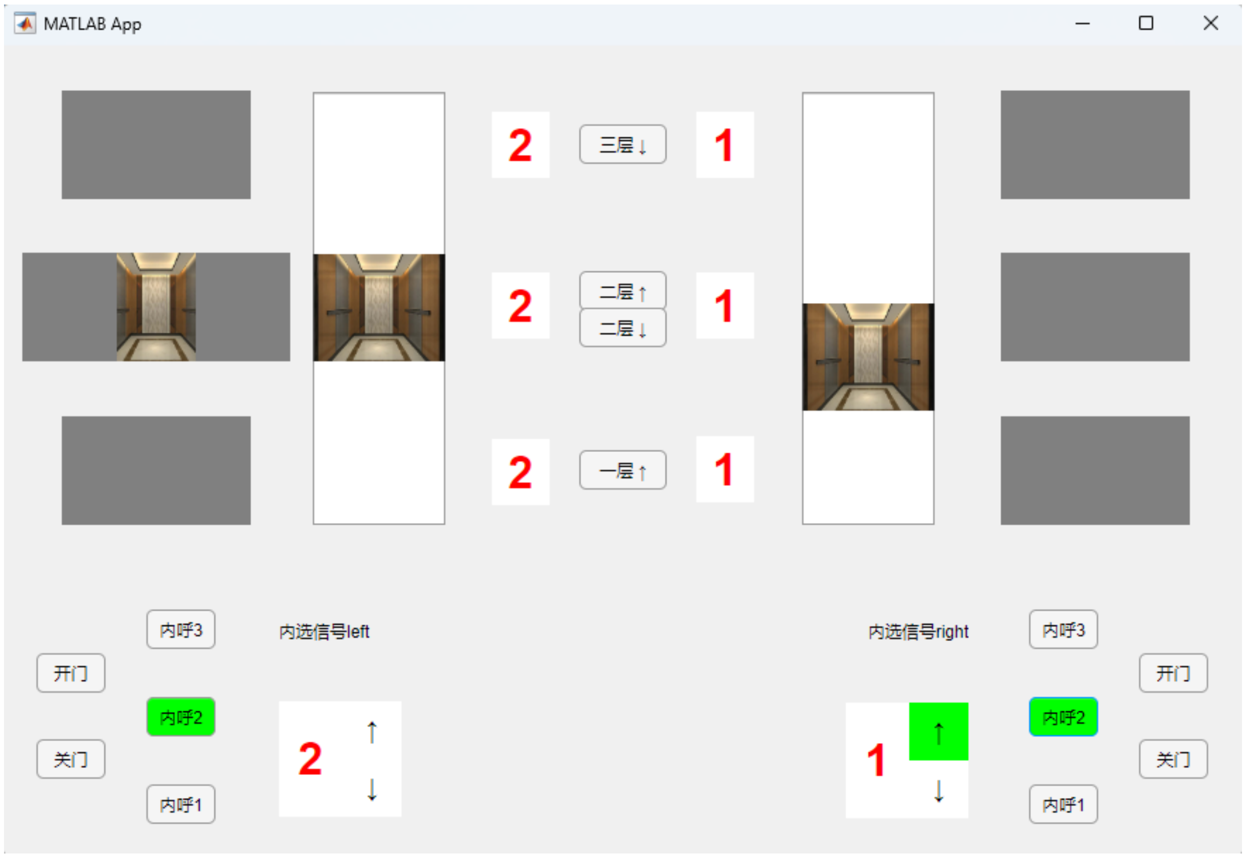
In these two functions, controller will setup acceleration and update velocity by

$$V = V0 + at$$

and update position by

$$X = V0t + 1/2at^2$$

And front-end will update correspond shaft position so that we can simulate elevator movement.



S2 Elevator handler

S2.1 Schedule algorithm

S2.1.1 Proximity principle

S2.1.1.1 Near to respond

Each time when two elevators both can be scheduled, we will calculate the gap between current position and target position of two elevators in function dispatch(). So that we can always let the near one to respond the elevator instructions.

S2.1.2 Current instruction priority principle

S2.1.2.1 Open/close deny

When elevator is moving, their state absolutely not equal to "0"(Stop state). So each time program call the callback_function, it will call the ass_repeat() at first. And in this function, we deny the open and close signal by reset it's sig_list corresponding value to 0 if the elevator is moving so that we can make sure our elevator will not open/close when it's running.

S2.1.2.2 Opposite instruction deny

When elevator is called by one instruction, it has it's own state of "upping" or "downing". If this time some opposite direction instructions are active, the dispatch() function will not dispatch these instructions to the elevator by checking the current position and state of it.

S2.1.3 Shortest running distance and energy-saving principle

Our implementation is based on the principle of the shortest running path of the system, as follows:

Assume two elevator are in the first floor

1. Call second_Button_up and third_Button will only call one elevator hold it.
2. Call second_Button_down and third_Button will only call one elevator hold it.
3. Call second_Button_up and first_Button will only call one elevator hold it.
4. Call left_call_2 and third_Button will only call left elevator hold it.
5. Call left_call_3 and third_Button will only call left elevator hold it.
6. Call right_call_2 and third_Button will only call right elevator hold it.
7. Call right_call_3 and third_Button will only call right elevator hold it.

Also if two elevator are in the third floor

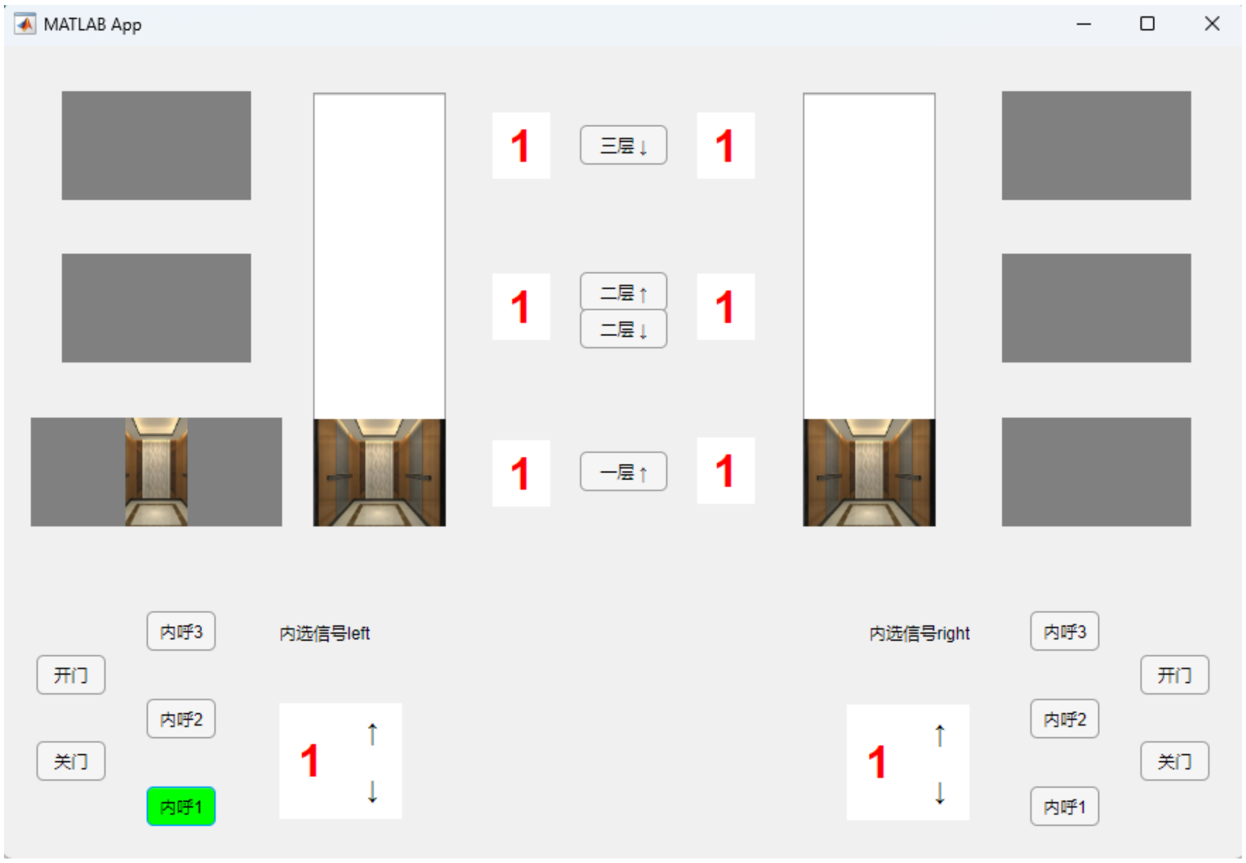
1. Call second_Button_down and first_Button will only call one elevator hold it.
2. Call second_Button_up and first_Button will only call one elevator hold it.
3. Call second_Button_down and third_Button will only call one elevator hold it.
4. Call left_call_2 and first_Button will only call left elevator hold it.
5. Call left_call_1 and first_Button will only call left elevator hold it.
6. Call right_call_2 and first_Button will only call right elevator hold it.
7. Call right_call_1 and first_Button will only call right elevator hold it.

Our algorithm is in the dispatch() function, we specify each conditions with corresponding variables setup . And return tar_direction with real value to execute normally or [-1,-1] to yield it make sure our running system is correct.

S2.2 Physical constraints

S2.2.1 Automatically open

When a dispatch action end and an elevator arrive at target floor, controller will set the elevator state to 3(Opening state) at the end of move_up() or move_down() function. At next timer cycle, controller will detect it's new state and call open() function to open the door automatically.

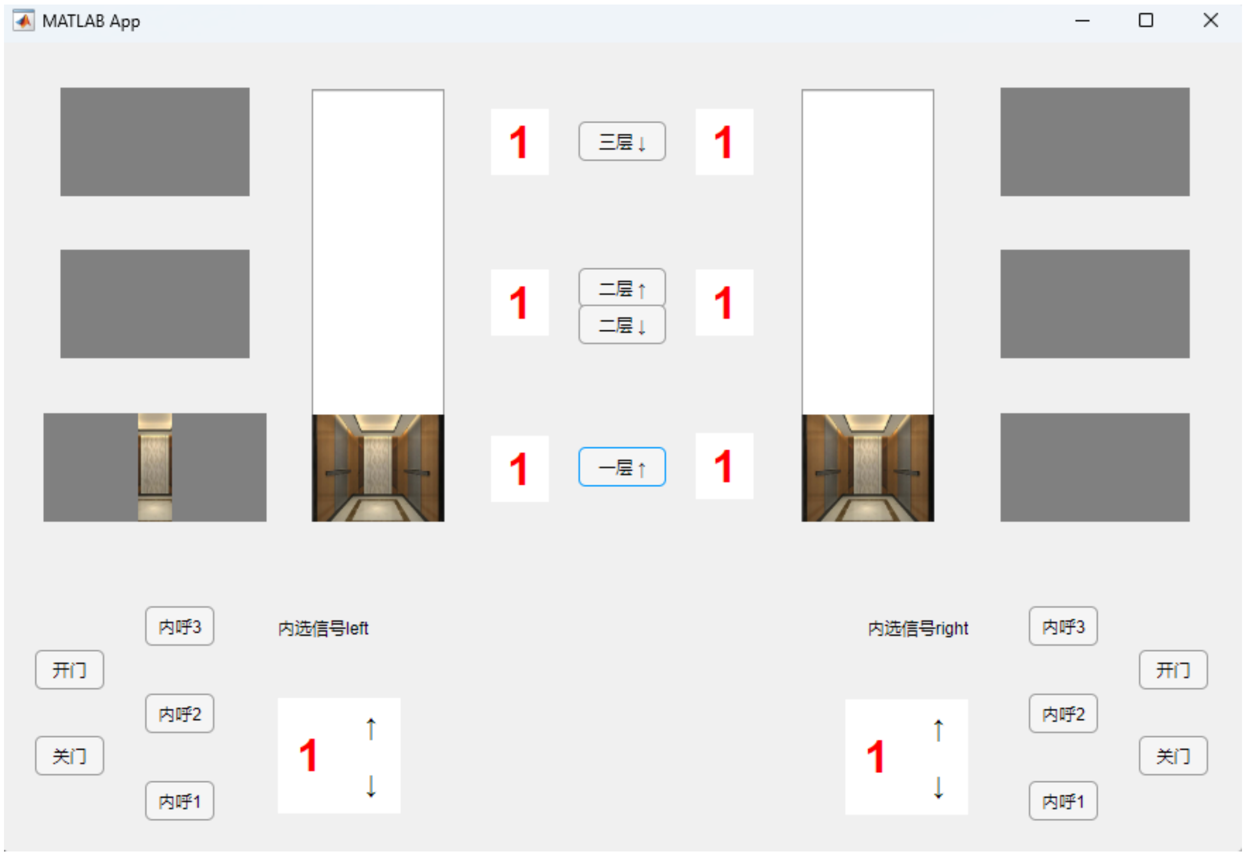


S2.2.2 Open_max wait

When a door open to Maximum state, controller will detect it and accumulate elevator's property : **tim** until it equal another property : **open_tim** at the beginning of open() function. Accumulation process will not change elevator state and position so that we can achieve "wait" purpose.

S2.2.3 Automatically close

When wait process end, controller will change the elevator state to 4(closing state). In the next cycle, it can control the elevator begin to close by calling close() function until fully closed.



S2.2.4 Only start after the door is fully closed

All move or dispatch function are limited by elevator state. Only state equal to 0(stop state) will elevator move.

S2.2.5 Height limitation

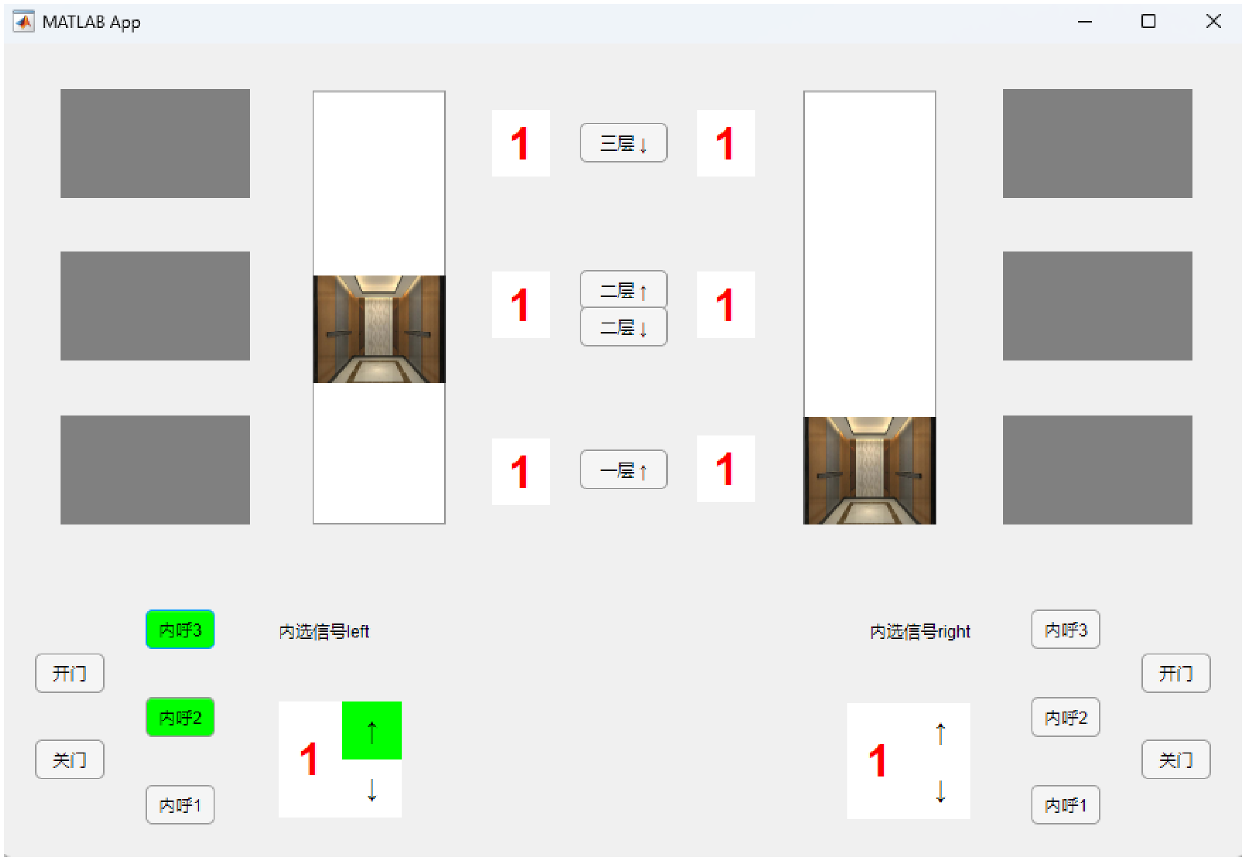
In move_up() and move_down() function, we set the boarder of height corresponding to first and third floor. If elevator will exceed the boarder, we just set it equal to boarder number and set the elevator state to 0(stop) or 3(opening) so that it will not exceed boarder.

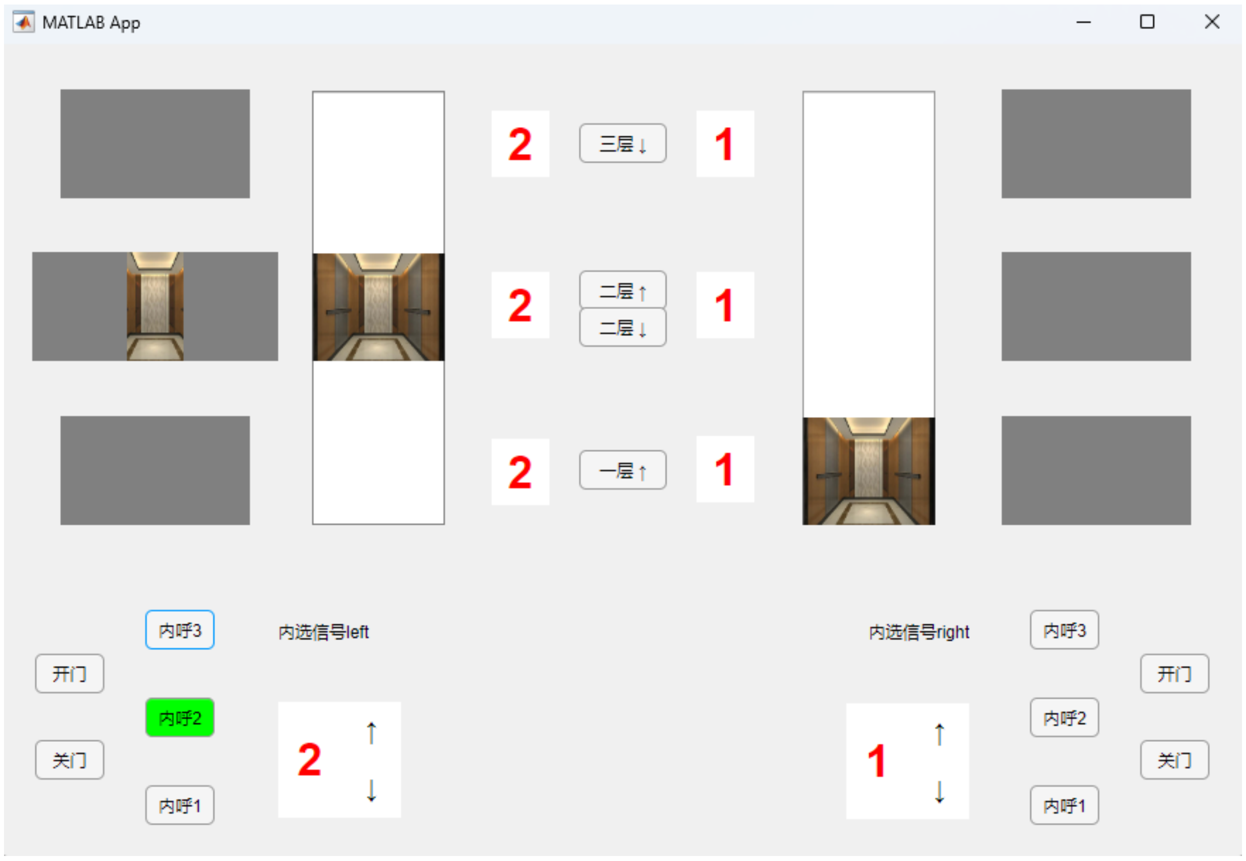
S2.2.6 Double-click cancel

S2.2.6.1 Inner double-click cancel

When passenger want to cancel a command which is not executing now, he can double press the button.

If this happen, interface part will detect it by checking sig_list and record first-press time in ui part's **double_sig** array. If second-press happen, check interval between first-press time and second-press time. Only interval less than 0.5 seconds and the corresponding value in **hold_list** doesn't equal elevator's number will command be canceled(Reset to 0 in sig_list).


MATLAB App
1
三层↓
1
1
二层↑
二层↓
1
1
一层↑
1
内呼3
开门
内呼2
关门
内呼1
内选信号left
1
↑
↓
内选信号right
1
↑
↓
内呼3
开门
内呼2
关门
内呼1


MATLAB App
2
三层↓
1
2
二层↑
二层↓
1
2
一层↑
1
内呼3
开门
内呼2
关门
内呼1
内选信号left
2
↑
↓
内选信号right
1
↑
↓
内呼3
开门
内呼2
关门
内呼1