REEMAN SLAM WEB API

Navigation version:RSN-v3.1.5

Creation date:2023-05-8

Modification date:2025-05-24

Glossary:

host:in the following`host`Indicates machine navigation system`IP`address,`IP`The third digit of the network segment cannot be`10`Network segment, such as:
`192.168.10.123`is an illegal address, please change it to another non-`10`Network segment router or change router`wifi`network segment.`127.0.0.1` express`The machine is not connected to any network, please connect to the network first. When calling the interface, the calling end needs to be connected to the same LAN as the navigation system. When using the interface, please change the`host`Replace with the corresponding`IP`address.

hostname :The name of the navigation host

Navigation mode: Navigation mode means that the machine can execute navigation instructions in this mode, which corresponds to the mapping mode. Mapping mode: Mapping mode means that the machine performs the task of scanning maps, and in this mode the map scanning can be completed.

QR: QR code used to assist positioning, some models come with it

virtual wall:Can be added on the map to limit the machine's range of activities

special area: Special areas can be added to the map to complete certain functions, such as changing the speed of the machine in the special area. The machine enters the special`The special area will be reported to the host computer through the serial port.

Basic Information

Get the current navigation version

```
-method: GET
-path:   http://(host)/reeman/current_version
-response:
  - Content-Type: application/json
  -result: {"version":"v1.1.0"}
  -mark:Returns the current version name
```

Get robot pose information

```
-method: GET
-path: http://(host)/reeman/pose
-response:
  - Content-Type: application/json
  -result: {"x":297,"y":251,"theta":0.97}
  -mark:Returns the current coordinates of the machine
```

Get the current mode

```
-method: GET
-path: http://(host)/reeman/get_mode
-response:
  - Content-Type:   application/json
  -result: {"mode":2}
  -mark:mode=1for mapping mode;mode=2for navigation mode
```

Get the machine hostname

```
-method: GET
-path: http://(host)/reeman/hostname
-response:
  - Content-Type: application/json
  -result: {"hostname":"reeman-robot"}
  -mark:Return to machine hostname
```

Obtain IMU state

```
-method: GET
-path: http://(host)/reeman/imu
-response:
  - Content-Type: application/json
  -result: {"a":0.0,"g":0.0}
  -mark:return result The result is normal, if it returns exception It is an exception.
```

Get laser data

```
-method: GET
-path: http://(host)/reeman/laser
-response:
  - Content-Type: application/json
  -result: {"coordinates": [[364,166], [365,167],[321, ...]}
  -mark:Return to laser dot matrix x,y 2D array of coordinates
```

Get power management status

```
-method: GET
-path: http://(host)/reeman/base_encode
-response:
  - Content-Type: application/json
  -result: {"battery":100,"chargeFlag":1,"emergencyButton":0}
  -mark:in,batteryis the power(0-100);chargeFlag For charging status:2-Charging at the charging station;3-Adapter charging;8-Connecting to the charging pile; other values indicate non-charging status;emergencyButton For the emergency stop switch state:0for pressing,1for bounce.
```

Get current speed

```
-method: GET
-path: http://(host)/reeman/speed
-response:
  - Content-Type: application/json
  -result: {"vx":0.3,"vth":0.5}
  -mark:vx is the linear speed, the unit ism/s;vth is the angular velocity in degrees/s
```

reset

```
-method: POST
-path: http://(host)/cmd/reloc_pose
-body:{"x":1.0,"y":3,"theta":0.14} // x,y,theta are the coordinates and direction of relocation respectively.
-response:
  - Content-Type:  application/json
  -result:{"status":"success"}
```

Absolute relocation

```
-method: POST
-path:  http://(host)/cmd/reloc_absolute
-body:{"x":1.0,"y":3,"theta":0.14} // x,y,theta are the coordinates and direction of relocation respectively.
-response:
  - Content-Type:  application/json
  -result:{"status":"success"}
```

Navigation module

Send location coordinate navigation

```
-method: POST
-path: http://(host)/cmd/nav
-body:{"x":285,"y":252,"theta":1.6}
-response:
  - Content-Type: application/json
  -result:{"status":"success"}
  -mark: theta is radians
```

Send target name navigation

```
-method: POST
-path: http://(host)/cmd/nav_name
-body:{"point":"A"}
-response:
  - Content-Type: application/json
  -result:{"status":"success"}
  -mark: A is the name of the target point. Normal return means the target point is found.,Otherwise return Exception
```

Send fixed route point name navigation

-method: POST
-path: http://(host)/cmd/points_path
-body:{"name":"A"}
-response:
  - Content-Type: application/json
  -result:{"status":"success"}
  -mark: A is the name of the target point. Normal return means the target point is found.,Otherwise return Exception

Get navigation status

-method: GET
-path: http://(host)/reeman/nav_status
-body:{}
-response:
  - Content-Type: application/json
  -result:{}
-mark:Return result format:
{"res":3,"reason":0,"goal":"A","dist":1.8,"mileage":2.3}nav_result{state code name dist_to_goal mileage} There are six parameters in total, separated by spaces. In state Represents the navigation phase, code represents the result of this stage,name Indicates the name of this navigation or coordinate,dist_to_goal Represents the distance from the target point,melieage Indicates the mileage traveled during this navigation.

state=6 corresponding to the stage code Respectively:

-0:success;
-1:Connecting to charging pile;
-2:The emergency stop switch is pressed;
-3:Adapter is charging;
-4:Target point not found;
-5:agvdocking failed;
-6: Positioning abnormality;
-7: The distance between fixed route points is too large;
-8: Fixed route not found;
-9: Failed to read point information;

state=1Indicates the start of navigation

state=3corresponding to the stagecodeRespectively:
-0:Navigation successful;
-1:Navigation failed;

state=4Indicates manual cancellation of navigation.

A normal navigation process: such as sending navigation toAAfter clicking the command:nav_point[A], by requesting the navigation status interface, you will obtain theGet the following return:

resp : {"res":6,"reason":0,"goal":"A","dist":0,"mileage":0} //Normal status
resp : {"res":1,"reason":0,"goal":"A","dist":0.5,"mileage":0}//Start navigation
resp : {"res":3,"reason":0,"goal":"A","dist":0,"mileage":0.6}//Navigation results

Cancel navigation

```
-method: POST
-path: http://(host)/cmd/cancel_goal
-body:{}
-response:
  - Content-Type: application/json
  -result:{"status":"success"}
-mark:Normal return indicates success, otherwise return Exception
```

Navigation charging

```
-method: POST
-path: http://(host)/cmd/charge
-body:{"type":0,"point":"Charging pile"}
    -mark:type=0 Indicates direct docking with charging piles ;type=1 Indicates canceling the docking charging pile type=2 Indicates navigating to the charging pile location and starting to connect to the charging pile."Charging pile"Bit
The setting needs to be calibrated on the web page first.
-response:
  - Content-Type: application/json
  -result:{"status":"success"}
  -mark:Normal return indicates success, otherwise return Exception
```

Set navigation maximum speed

```
-method: POST
-path: http://(host)/cmd/max_speed
-body:{"speed":0.5}
-response:
  - Content-Type:  application/json
  -result:{"status":"success"}
  -mark: The speed value range is 0.3 to 1.0m/s
```

Get navigation path

```
-method: GET
-path: http://(host)/reeman/global_plan
-body:
-response:
  - Content-Type:    application/json
  -result:{"coordinates":[[0.826,-0.35],[0.86,-0.38]]}
  -mark: Returns a two-dimensional coordinate array, representing x and y
  coordinates respectively. The actual distance of each coordinate is about 30cm.
```

Set the QR code docking direction

```
-method: POST
-path: http://(host)/cmd/agv_docking_direction
-body:{"direction":1}
-response:
  -Content-Type:application/json
  -result:{"status":"success"}
  -mark: "1" indicates the forward direction, "0" indicates the backward direction
  (navigation needs to be restarted after setting)
```

Set QR code exit distance

```
-method: POST
-path: http://(host)/cmd/agv_docking_dis
-body:{"distance":0.5}
-response:
  -Content-Type:application/json
  -result:{"status":"success"}
  -mark: The exit distance from the QR code location, in meters. It is recommended
  to be around 1 meter (navigation needs to be restarted after setting)
```

Get QR code navigation result

```
-method: GET
-path: http://(host)/reeman/agv_nav_status
-body:
-response:
  -Content-Type:application/json
  -result:{"status":"agv_success(tag)"}
  -mark:agv_success(tag): navigation is successful, tag is the target QR code;
  agv_fail: navigation failed
```

map module

Get current map

Get the current map data, which includes map size and other data. See the return result for details.

```
-method: GET
-path: http://(host)/reeman/map
-body:{}
-response:
  - Content-Type:  application/json
  -result: {}
```

Switch mapping mode

After switching to mapping mode, the machine will perform mapping scanning. Please push or remotely control the robot to move for scanning. After the scan is complete, click BuildEnd mapping.

```
-method: POST
-path: http://(host)/cmd/set_mode
-body: {"mode":1}
-response:
  - Content-Type:  application/json
  -result: {}
```

Switch to incremental mapping mode

Switch to incremental mapping mode. Please ensure that the positioning is correct before switching, otherwise the map may deviate from the actual environment.

```
-method: POST
-path: http://(host)/cmd/set_mode
-body: {"mode":3}
-response:
  - Content-Type:  application/json
  -result: {}
```

save map

Save the current scanned map. If you want to give up this map construction, you can switch to the navigation mode. After saving the map, you need to switch to the navigation mode.Navigate normally.

```
-method: POST
-path: http://(host)/cmd/save_map
-body:{}
-response:
  - Content-Type:  application/json
  -result: {}
```

Get the name of the current map

Returns the map name used by the current navigation system

```
-method: GET
-path: http://(host)/reeman/current_map
-response:
  - Content-Type: application/json
  -result:    {"alias":"","name":"674e32391fad7ad8a1422360357d0516"}
  -mark:inaliasalias for the map,nameUniquely identifies the map
```

Get a list of all maps

Returns a list of all maps in the current navigation system

```
-method: GET
-path: http://(host)/reeman/history_map
-response:
  - Content-Type: application/json
  -result: {map list}
```

Apply map

Switch to the specified map. After switching the map, it will be relocated to the charging pile location by default. It is recommended to switch the map at the location of the calibrated charging pile.

```
-method: POST
-path: http://(host)/cmd/apply_map
-body: {"name":"map name"}
-response:
  -Content-Type: application/json
  -result: {}
```

Export map

Download the specified map to the local area

```
-method: POST
-path: http://(host)/download/export_map
-body:{"name":"Map name"}
-config:{responseType:"blob"}
-response:
  - Content-Type: application/json
  -result:
```

Upload map

Select a map file to upload to the navigation system,

body:let data=new FormData(); data.append('file',file)

```
-method: POST
-path: http://(host)/upload/import_map
-body:{data}
-response:
  - Content-Type: application/json
  -result: {}
```

Rename map

Please be careful when renaming the map name.

```
-method: POST
-path: http://(host)/cmd/rename_map
-body:{"name" :"5d81b1a*****e6a48","alias":"test34"} // Where name is the map id name (equivalent to the map unique identifier), and alias is the alias to be modified
-response:
  - Content-Type: application/json
  -result: {}
```

Move map to trash

Move the map to the recycle bin. After 30 days, the recycle bin map will be automatically cleared. (The current map cannot be moved to the trash)

```
- method: POST
- path: http://(host)/cmd/trash_map
- body:{ "name":"map name"}
- response :
  - Content-Type: application/json
  - result: {}
```

Restore map from trash

Restore map from trash

```
- method: POST
- path: http://(host)/cmd/restore_map
- body:{ "name":"map name"}
- response :
  - Content-Type: application/json
  - result: {}
```

Delete map

Delete the map with the specified name. It cannot be restored after deletion. Please operate with caution.

```
-method: POST
-path: http://(host)/cmd/delete_map
-body:{"name":"map name"}
-response:
  - Content-Type:   application/json
  -result: {}
```

Set the calibration position

Set the calibration position. The corresponding coordinates should be obtained before setting.

```
-method: POST
-path: http://(host)/cmd/position
-body:{"name":"test","type":"delivery","pose":
{"x":-3.05,"y":0.41,"theta":0.56}} // name is the location name, type is the point type, and pose is the coordinates.
Note: type must be one of the following types: delivery-distribution point, normal-route point, production-production point, charge charging pile
- response :
- Content-Type: application/json
  - result: {}
```

Delete the calibration position

```
-method: DELETE
-path: http://(host)/cmd/position
-body:{"test"} // The name of the location to be deleted
-response:
  - Content-Type:  application/json
  -result: {}
```

Get calibration position

Get the calibrated coordinate position list

```
-method: GET
-path: http://(host)/reeman/position
-body:{}
-response:
  - Content-Type:  application/json
  -result:  {"waypoints":[{"name":"1","type":"delivery","pose":
{"x":-
2.06,"y":0.77,"theta":0.19}},{"name":"3","type":"avoid","pose":{"x":-
0.91,"y":0.48,"theta":-0.01}}]}
  -mark:Returns the list of calibration location locations, wherenameIndicates the name of the calibrated location,typerefers to the location type,poseis the position coordinate
```

Get route

Get the route marked in the mark position interface and return the route array.

```
-method: GET
-path: http://(host)/reeman/navi_routes
-body: {}
-response :
  -Content-Type: application/json
  -result:  {"route_name":[[x1,y1],[x2,y2]]"}
  -mark: Returns a route collection, where route_name is the route name, and the two-dimensional array is the point coordinates of the route. x1 and y1 are the route point coordinates.
```

Get fixed route

Get the fixed route data marked in the navigation route interface

```
-method: GET
-path:  http://(host)/reeman/path_model
-body: {}
-response :
  -Content-Type:  application/json
  -result:
{"model":"","name":"","version":"","point":[{"name":"1","type":"waypoint","xPos
ition":"2.2","yPosition":"1.1","vehicleOrientationAngle":"3.14"},{...}],"path":
[],"cruise":[],"location":[],"property":""}
-mark: Returns the calibrated fixed route data, where the point array object
is the calibrated various types of point data. The main processing data is this
data. Other data can be ignored. The point types are: waypoint = route point;
delivery = delivery point;  avoidance = avoidance point;  temporary = temporary
stop point; production = production point; recycle = recycling point; charge =
charging pile, xPosition, yPosition, vehicleOrientationAngle represent the left
x-axis, y-axis and angle respectively.  To convert the  navigation  map
coordinates, the coordinates need to be divided by 1000
```

Get special area

Get special area

```
-method: GET
-path:   http://(host)/reeman/special_polygon
-body:{}
-response:
  - Content-Type: application/json
  -result:
  {"polygons":[{"name":"","polygon":[[],[]],"speed":0.6,"type":0},{}...]}
```

virtual wall

Get a virtual wall

```
-method: GET
-path: http://(host)/reeman/restrict_layer
-body:{}
-response:
  - Content-Type: application/json
  -result:Returns a two-dimensional array of virtual wall points
```

Update virtual wall

```
-method: POST
-path: http://(host)/cmd/restrict_layer
-body:{ "waypoints":[{"pose":{"point1":{"x":1.1,"y":2.2},"point2":
{"x":3.3,"y":4.4}}}]} //Where waypoints is an array of virtual wall segments,
pose represents a virtual  wall,  and point1 and  point2  represent the
coordinates of the two endpoints of the virtual wall.
-response:
  - Content-Type:   application/json
  -result: {}
```

move

Control the movement of the robot

Send commands to control the movement of the robot. If the robot needs to move continuously, commands need to be sent continuously at a frequency of about 300 milliseconds.

```
-method: POST
-path: http://locahost/cmd/speed
-body: {"vx": greater than 0 means forward, less than 0 means backward, the default speed is 0.3 m/s, "vth": greater than 0 means left turn, less than 0 means right turn, the default angular velocity is 0.5 radians/s}
-response:
    - Content-Type:   application/json
    -result: {}
-mark: 1. Forward: forward straight parameter is positive, angle parameter is 0
    For example, forward: body = {"vx":0.4,"vth":0}

    2. Backward: backward straight parameter is negative, angle parameter is 0
    For example, backward: body: {"vx":-0.4,"vth":0}

    3. Turn left: straight parameter is 0, angle parameter is positive
    For example, turn left: body: {"vx":0,"vth":0.5}

    4. Turn right: straight parameter is 0, angle parameter is negative
    For example, turn right: body: {"vx":0,"vth":-0.5}

    5. Stop: straight parameter, rotation parameter are both 0
```

The robot moves a specified distance

```
-method: POST

-path: http://locahost/cmd/move

-body: {"distance": moving distance (cm), "direction": moving direction (0 means backward, 1 means forward, "speed": moving speed (m/s)}

-response :

   -Content-Type: application/json

   -result: {}

-mark :

   1. Forward: the diantance parameter is the distance of this movement, in cm; direction is the direction of this movement, forward is 1; speed is the moving speed, in meters/second.

      For example, 100cm forward at 0.8 meters per second: body = {"distance":100,"direction":1,"speed":0.8}

   2. Backward: the diantance parameter is the distance of this movement, in cm; direction is the direction of this movement, backward is 0; speed is the moving speed, in meters/second.

      For example, 100cm backward at 0.8 meters per second: body = {"distance":100,"direction":0,"speed":0.8}
    5.Stop: body = {"distance":0,"direction":1,"speed":0}
```

The robot turns to a specified angle

```
  - method: POST
  - path: http://locahost/cmd/turn
  - body: {"direction": Turn direction 1 is left turn, 0 is right turn; "angle": Turn angle, "speed": Turn angular speed, radian degrees/second}
-     response :
    - Content-Type:  application/json
    - result: {}
- mark :
    1. Turn left: The angle parameter is the angle of this movement, in degrees; direction is the direction of this movement, 1 for left turn; speed is the angular speed of movement, in radians/second.
       For example, turn left 90 degrees at 0.6 radians per second: body = {"direction":1,"angle":90,"speed":0.6}
    2. Turn right: The angle parameter is the angle of this movement, in degrees; direction is the direction of this movement, 0 for left turn; speed is the angular speed of movement, in radians/second.
       For example, turn right 90 degrees at 0.6 radians per second: body = {"direction":0,"angle":90,"speed":0.6}
    5. Stop: body = {"angle":0,"direction":1,"speed":0}
```

upgrade

Upgrade navigation system

Select the upgrade file to upload to the navigation system for upgrade. It will be automatically upgraded after the upload is successful.

body:let data=new FormData(); data.append('file',file)

```
-method: POST
-path: http://(host)/upload/upgrade
-body:{data}
-response:
  - Content-Type:  application/json
  -result: {}
```

external control

Power board external power supply interface

Controlling the external power supply switch, specifically referring to the control of the power board's external power supply interface (24V or 36V power supply).

Control 24V and 36V external power supply

```
-method: POST
-path: http://locahost/cmd/external_power_supply
-body: {"type:":"24v","operation":"0"}
       mark:type: 24v indicates 24V power supply, 36v indicates 36V power supply, 24v36 indicates simultaneous control of 24V and 36V power supply; operation indicates operation: "0" indicates off, "1" indicates on.
-response:
    -Content-Type:  application/json
    -result: {}
```

Turn off the robot

```
-method: POST
-path: http://locahost/cmd/shutdown
-body: {}
-response:
    -Content-Type: application/json
    -result: {}
```

Jacking module control

Control the lifting motor to rise and fall

jack up

```
-method: POST
-path: http://locahost/cmd/hydraulic_up
-body: {}
-response:
    - Content-Type: application/json
    -result: {}
```

decline

```
-method: POST
-path: http://locahost/cmd/hydraulic_down
-body: {}
-response:
    - Content-Type: application/json
    -result: {}
```