API 接口文档

基础信息

- 基础URL: `http://<server-ip>` /public/api
- 协议: HTTP/JSON

数据获取接口

1. 获取任务状态

- 端点: `GET /get_task_state`
- 描述: 获取当前机器人任务执行状态
- 示例：http://192.168.0.121/public/api/get_task_state
- 返回:

代码块

```
{
  "cancellation_reason": "无",
  "current_step_num": 4,
  "current_step_parameter": "空闲",
  "end_time": {
    "nsecs": 866343736,
    "secs": 1755748708
  },
```

```
    "error_message": "无",
    "now_nodeid": 3,
    "progress": 100,
    "start_time": {
      "nsecs": 150274515,
      "secs": 1755748680
    },
    "task_busy": 0,
    "task_id": 1,
    "task_name": "1",
    "task_status": "已完成"
  }
```


字段名	类型	说明	示例值
cancellation_reason	String	任务取消原因	"无", "用户手动取消", "遇到障碍
current_step_num	Integer	当前执行的任务步骤编号	4 (0表示无任务)
current_step_parameter	String	当前步骤参数或状态	"空闲", "导航到点(X,Y)"，表示正在执
end_time	Object	任务结束时间	**{"secs": 1755748708, "nsecs": 86634**
error_message	String	错误信息	"无", "路径规划失败"
now_nodeid	Integer	当前所在节点ID	3 (-1表示无节点)
progress	Integer	任务完成百分比	100 (0-100)
start_time	Object	任务开始时间	**{"secs": 1755748680, "nsecs": 15027**
task_busy	Integer	任务忙碌状态	0 (空闲), 1 (忙碌)
task_id	Integer	任务唯一标识ID	由任务发布者提供
task_name	String	任务名称	由任务发布者提供
task_status	String	任务当前状态	"已完成", "执行中", "已取消"

- 状态码: 200 (成功), 503 (ROS未连接)

2. 获取底盘数据

- 端点: `GET /get_chassis_data`
- 描述: 获取底盘传感器数据
- 返回: JSON格式的底盘数据

代码块


```
{
```

```
  "data": "{\"Battery\":
{\"Percent\":100,\"Voltage\":47.600002,\"Current\":0.640000,\"Temperature\":0,\
"FullVoltage\":0,\"UnderVoltage\":0,\"TotalCapacity\":0,\"Diverter\":100,\"Char
ge\":0},\"INPUTIO\":{\"INPUT\":0},\"ExitINPUTIO\":
{\"ExitINPUT\":0},\"OUTPUTIO\":{\"OUTPUT\":0},\"ExitOUTPUTIO\":
{\"ExitOUTPUT\":0},\"QR\":
{\"CodeId\":100,\"Angle\":-83.200005,\"X\":0.065500,\"Y\":-0.074900,\"ReadOk\":
0},\"ODOM\":
{\"Position_x\":1.695199,\"Position_y\":-0.834525,\"Orientation\":-2.120373,\"lin
earspeed\":0,\"angularspeed\":0},\"Gyro\":
{\"Gyro_Z\":-171.729996},\"WorkMode\":{\"Mode\":3},\"Error\":
{\"1#\":0,\"2#\":0,\"3#\":0},\"LiftMechanism\":
{\"PlatFormIsDOWN\":0,\"PlatFormIsUP\":0,\"RealPosition\":0}}"
}
```


3. 获取机器人位姿

- 端点: `GET /get_robot_pose`
- 描述: 获取机器人当前位置和姿态

代码块


```
{
  "covariance_score": 0.599570512771606,
  "current_trajectory": "",
  "last_update_duration": 0,
  "last_update_pose": {
    "orientation": {
      "w": 0,
      "x": 0,
      "y": 0,
      "z": 0
    },
    "position": {
      "x": 0,
      "y": 0,
      "z": 0
    }
  },
  "robot_pose": {
    "orientation": {
      "w": 0.0485254663046941,
      "x": -2.41944967282037e-15,
      "y": 8.93279769555815e-16,
      "z": 0.998821945653935
    },
```

```
      "position": {
        "x": -2.10999039973544,
        "y": 0.119835906061222,
        "z": 5.43115749000142e-15
      }
    }
  }
```

4. 获取地图数据

- 端点: `GET /get_map_data`

- 描述: 获取当前地图数据（原始格式）

代码块

```
{"data":[-1,-1,-1,-1-1,-1],"header":{"frame_id":"map","seq":12898,"stamp":
{"nsecs":622278437,"secs":1755761775}},"info":{"height":306,"map_load_time":
{"nsecs":622278437,"secs":1755761775},"origin":{"orientation":
{"w":1.0,"x":0.0,"y":0.0,"z":0.0},"position":
{"x":-12.048816680908203,"y":-10.68303451538086,"z":0.0}},"resolution":0.050000
00074505806,"width":363}}
地图data里面这里省略了，实际是栅格地图的原始数据
```

5. 获取压缩地图数据

- 端点: `GET /get_map_data_encode`

- 描述: 获取压缩编码后的地图数据

代码块

```
{"data":"eF7tnYdC8soSgCOoqIhIFUR675DQmyAiIopUkV7f/xW8swkgXVRQ/3OTU9SUzebLZHZmdn
b37YjMD/AAHNNGA=","header":{"frame_id":"map","seq":5589,"stamp":
{"nsecs":26516596,"secs":1755754230}},"info":{"height":306,"map_load_time":
{"nsecs":26516596,"secs":1755754230},"origin":{"orientation":
{"w":1.0,"x":0.0,"y":0.0,"z":0.0},"position":
{"x":-12.048816680908203,"y":-10.683908081054689,"z":0.0}},"resolution":0.05000
000074505806,"width":363}}
压缩后的地图节省了传输速度，但是需要再次解压才能使用。
```

6. 获取激光扫描数据

- 端点: `GET /get_laser_scan`

- 描述: 获取激光雷达扫描数据

代码块

```
{
    "angle_increment": 0.00116355286445469,
    "angle_max": 3.14159250259399,
    "angle_min": -3.14159250259399,
    "header": {
      "frame_id": "laser",
      "seq": 267250,
      "stamp": {
        "nsecs": 923517239,
        "secs": 1755762269
      }
    },
    "intensities": [4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 4352, 6144, 7680, 5888, 4096, 3712, 3840, 3584, 3584, 3712, 3840, 0, 2432, 2432, 2560, 2688, 3328, 3712, 3840, 3840, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 3968, 7.4879994392395],
    "scan_time": 0,
    "time_increment": 0
}
```

7. 获取底盘参数数据

- 端点: `GET /get_chassisParameter_to_ros`
- 描述: 获取底盘参数信息

代码块

```
"{\"ChassisData\":
{\"chassisType\":0,\"WHEEL_ROUND\":180,\"REDUCTION_RATIO\":30,\"TWOWHEEL_DISTANCE\":571,\"AxleSpacing\":1000,\"ENCORDER_ROUND\":131072,\"MotorFactory\":0},\"chassisVoice\":{\"SpeekMode\":1},\"chassisManualMode\":
{\"ManualLinearSpeed\":0.300000,\"ManualAngularSpeed\":0.200000,\"AccTime\":0.800000,\"DecTime\":0.400000,\"remote_forward\":7,\"remote_backward\":8,\"remote_left_turn\":9,\"remote_right_turn\":10,\"remote_left_move\":0,\"remote_right_move\":0,\"emergency_stop\":0},\"chassisLiftMechanism\":
{\"Switch\":0,\"MotorFactory\":0,\"MotorENCORDER_ROUND\":10000,\"REDUCTION_RATIO\":32,\"DaoCheng\":5,\"UpPosition\":40,\"FindHomeSpeed\":500,\"NormalSpeed\":800}}"
```

8. 获取底盘命令数据

- 端点: `GET /get_chassis_to_ros`
- 描述: 获取底盘命令信息

```
"{\"Battery\":
{\"Percent\":100,\"Voltage\":47.400002,\"Current\":0.590000,\"Temperature\":0,\
"FullVoltage\":0,\"UnderVoltage\":0,\"TotalCapacity\":0,\"Diverter\":100,\"Char
ge\":0},\"INPUTIO\":{\"INPUT\":0},\"ExitINPUTIO\":
{\"ExitINPUT\":0},\"OUTPUTIO\":{\"OUTPUT\":0},\"ExitOUTPUTIO\":
{\"ExitOUTPUT\":0},\"QR\":
{\"CodeId\":100,\"Angle\":-83.200005,\"X\":0.065500,\"Y\":-0.074900,\"ReadOk\":
0},\"ODOM\":
{\"Position_x\":1.695199,\"Position_y\":-0.834525,\"Orientation\":-2.120373,\"lin
earspeed\":0,\"angularspeed\":0},\"Gyro\":
{\"Gyro_Z\":-171.910004},\"WorkMode\":{\"Mode\":3},\"Error\":
{\"1#\":0,\"2#\":0,\"3#\":0},\"LiftMechanism\":
{\"PlatFormIsDOWN\":0,\"PlatFormIsUP\":0,\"RealPosition\":0}}"
```


9. 获取激光TF数据

- 端点: `GET /get_tf_laser`
- 描述: 获取激光雷达的TF变换数据

代码块


```
{
  "child_frame_id": "/laser",
  "header": {
    "frame_id": "/base_link",
    "seq": 0,
    "stamp": {
      "nsecs": 816159767,
      "secs": 1755762413
    }
  },
  "transform": {
    "rotation": {
      "w": 1,
      "x": 0,
      "y": 0,
      "z": 0
    },
    "translation": {
      "x": 0,
      "y": 0,
      "z": 0
    }
  }
}
```

10. 获取导航状态

- 端点: `GET /get_nav_state`
- 描述: 获取完整的导航状态信息

代码块


```
{
  "battery": {
    "charge": 0,
    "current": 0.59,
    "percent": 100,
    "temperature": 0,
    "voltage": 47.5
  },
  "robot": {
    "odom_speed_x": 0,
    "odom_speed_y": 0,
    "odom_speed_z": 0,
    "pose_angle": 174.441206028229,
    "pose_x": -2.10998801647277,
    "pose_y": 0.119442398684581
  },
  "system": {
    "emergency": null,
    "mechanical_avoidance": false,
    "obstacle_slow": false,
    "obstacle_stop": false,
    "pause": false,
    "work_mode": 3
  },
  "task": {
    "cancellation_reason": "无",
    "current_step_num": 4,
    "current_step_parameter": "空闲",
    "end_time": "1755748708866343736",
    "error_message": "无",
    "now_nodeid": 3,
    "path": {
      "1": 2,
      "2": 3
    },
    "progress": 100,
    "start_time": "1755748680150274515",
    "steps": [
      {
        "action_type": "speed_high",
```

```
          "parameters": "{}",
          "step_id": 4
        },
        {
          "action_type": "move_to_node",
          "parameters": "{\"node_id\":2}",
          "step_id": 1
        },
        {
          "action_type": "speed_turtle",
          "parameters": "{}",
          "step_id": 2
        },
        {
          "action_type": "move_to_node",
          "parameters": "{\"node_id\":3}",
          "step_id": 3
        }
      ],
      "task_busy": 0,
      "task_id": 1,
      "task_name": "1",
      "task_status": "已完成"
    }
  }
```

11. 获取综合机器人数据

- 端点: `GET /get_robot_and_sensor_data`
- 描述: 一次性获取位姿、激光和TF数据

代码块

```
{
    "laser_scan": {
        "angle_increment": 0.0011635528644546866,
        "angle_max": 3.141592502593994,
        "angle_min": -3.141592502593994,
        "header": {
            "frame_id": "laser",
            "seq": 271279,
            "stamp": {
                "nsecs": 835692135,
                "secs": 1755762471
            }
        },
```

```
        "intensities": [
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            4352.0,
            90993499755859,
            7.491995334625244,
            7.491997718811035,
            7.4909987449646,
            7.489999294281006
        ],
        "scan_time": 0.0,
        "time_increment": 0.0
    },
    "laser_tf": {
        "child_frame_id": "/laser",
        "header": {
            "frame_id": "/base_link",
            "seq": 0,
            "stamp": {
                "nsecs": 937091511,
                "secs": 1755762471
            }
        },
```

```
            "transform": {
                "rotation": {
                    "w": 1.0,
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                },
                "translation": {
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                }
            }
        },
        "robot_pose": {
            "covariance_score": 0.5240917205810547,
            "current_trajectory": "",
            "last_update_duration": 0.0,
            "last_update_pose": {
                "orientation": {
                    "w": 0.0,
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                },
                "position": {
                    "x": 0.0,
                    "y": 0.0,
                    "z": 0.0
                }
            },
            "robot_pose": {
                "orientation": {
                    "w": 0.04848896573588381,
                    "x": -1.8124478712524588e-15,
                    "y": 8.284230085256766e-15,
                    "z": 0.9988237182815917
                },
                "position": {
                    "x": -2.1100186939922816,
                    "y": 0.119425396235446,
                    "z": 4.8635388114864237e-14
                }
            }
        }
    }
```

📤 数据任务发布接口

1. 发布任务

- 端点: `POST /pub_task`
- 描述: 向机器人发送前往任务站点
- 请求数据格式(json):

代码块

```
{
    "task_id": 2,
    "task_name": "前往站点-2",
    "direction": 0,
    "steps": [
        {
            "step_id": 2,
            "action_type": "move_to_node",
            "parameters": "{\"node_id\":2}"
        }
    ]
}
```

参数名	类型	必填	说明
task_id	int	是	任务 ID，数字类型，无固定规则
task_name	string	是	任务名称
direction	int	是	方向（暂未启用，默认 `0`）
steps	array	是	任务步骤列表
└ step_id	int	是	步骤 ID，多个步骤时需保持唯一
└ action_type	string	是	步骤类型，例如 `move_to_node`（前往站点）
└ parameters	string	是	站点参数，需为字符串格式，如 `"{\"node_id\":2}"`

- 返回

代码块

```
{
    "status": "success",
    "steps_count": 1,
    "task_id": 2,
    "task_name": "前往站点-2"
}
```

参数名	类型	说明
status	string	执行结果
steps_count	int	本次任务包含的步骤数
task_id	int	任务 ID
task_name	string	任务名称

- 其他功能对照表（请求方式同上，只需修改其中的请求参数即可）
- 示例1（前往站点）：

代码块

```
{
    "task_id": 2,
    "task_name": "前往站点-2",
    "direction": 0,
    "steps": [
        {
            "step_id": 2,
            "action_type": "move_to_node",
            "parameters": "{\"node_id\":2}"
        }
    ]
}
```

- 示例2（语音播报）：

代码块

```
{
    "task_id": 2,
    "task_name": "语音播报",
    "direction": 0,
    "steps": [
```

```
        {
            "step_id": 2,
            "action_type": "voice",
            "parameters": "{\"text\":\"出发了\"}"
        }
    ]
}
```

- 示例3（不带(parameters)参数请求-打开避障）：

代码块

```
{
    "task_id": 1,
    "task_name": "1",
    "direction": 0,
    "steps": [
        {
            "step_id": 1,
            "action_type": "open_obstacle"
        }
    ]
}
```

功能名称	步骤类型	(parameters)参数
语音播报	voice	"{\"text\":\"出发了\"}"
延迟间隔播报语音	delay	"{\"time\":1,\"voice\":\"123123\",\"voiceInterval\":3}"
打开避障	open_obstacle	无
关闭避障	close_obstacle	无
上升到顶部	go_up	无
下降到底部	go_down	无
升级到多少厘米	go_height	"{\"height\":50}"
输入有	input_have	"{\"index\":1,\"voice\":\"请放货\",\"voiceInterval\":5}"

扩展输入有	expandinput_have	"{\"index\":1}"
输入无	input_nothave	"{\"index\":1,\"voice\":\"请确认空位\",\"voiceInterval\":5}"
扩展输入无	expandinput_nothave	"{\"index\":1}"
输出打开	output_open	"{\"index\":1}"
扩展输出打开	extendoutput_open	"{\"index\":1}"
输出关闭	output_close	"{\"index\":1}"
扩展输出关闭	extendoutput_close	"{\"index\":1}"
改变行进方向	heading_direction	"{\"value\":90}"
改变后方方向	rear_direction	"{\"value\":90}"
高速模式	speed_high	无
中速模式	speed_medium	无
低速模式	speed_low	无
龟速模式	speed_turtle	无
直行模式	straight_mode	无
横移模式	lateral_mode	无
转到目标角度	roate_to_angle	"{\"angle\":90}"
转到节点角度	roate_to_node_angle	"{\"nodeid\":2}"

2. 发布速度控制

- 端点: `POST /pub_vel`
- 描述: 发布机器人速度控制命令
- 请求数据格式(json):

代码块

```
{
    "linear": {
        "x": -0.3,
        "y": 0,
        "z": 0
    },
```

```
    "angular": {
        "x": 0,
        "y": 0,
        "z": 0
    }
}
```

参数名	类型	必填	说明
linear.x	Number	是	范围(0至-1),横向速度（x 方向）
linear.y	Number	是	范围(0至-1),横向速度（y 方向）
linear.z	Number	是	范围(0至-1),大于0左转，小于0右转
angular.x	Number	是	范围(0至-1),大于0前进，小于0后退
angular.y	Number	是	范围(0至-1),绕 y 轴角速度，通常为 0
angular.z	Number	是	范围(0至-1),绕 z 轴角速度（旋转速度）

- 停止：发布的参数都为0即可停止
- 返回

代码块

```
{
    "command": {
        "angular": {
            "x": 0,
            "y": 0,
            "z": 0
        },
        "linear": {
            "x": 0,
            "y": 0.3,
            "z": 0
        }
    },
```

```
    "status": "success"
}
```

- command:发布的数据
- status: 执行结果

3. 取消任务

- 端点: `POST /pub_cancel_task`
- 描述: 取消当前执行的任务
- 无请求数据格式
- 返回

代码块

```
{
    "message": "任务已取消",
    "status": "success"
}
```

- message:响应内容
- status: 执行结果

4. 发布底盘参数

- 端点: `POST /pub_ros_to_chassisParameter`
- 描述: 向机器人底盘发布参数配置
- 请求数据格式(json):

代码块

```
{
    "ChassisData": {
        "chassisType": 0,
        "WHEEL_ROUND": 180,
        "REDUCTION_RATIO": 30,
        "TWOWHEEL_DISTANCE": 571,
        "AxleSpacing": 1000,
        "ENCORDER_ROUND": 131072,
        "MotorFactory": 0
    },
    "chassisVoice": {
```

```
            "SpeekMode": 1
        },
        "chassisManualMode": {
            "ManualLinearSpeed": 0.3,
            "ManualAngularSpeed": 0.2,
            "AccTime": 0.8,
            "DecTime": 0.4,
            "remote_forward": 7,
            "remote_backward": 8,
            "remote_left_turn": 9,
            "remote_right_turn": 10,
            "remote_left_move": 0,
            "remote_right_move": 0,
            "emergency_stop": 0
        },
        "chassisLiftMechanism": {
            "Switch": 0,
            "MotorFactory": 0,
            "MotorENCORDER_ROUND": 10000,
            "REDUCTION_RATIO": 32,
            "DaoCheng": 5,
            "UpPosition": 40,
            "FindHomeSpeed": 500,
            "NormalSpeed": 800
        },
        "Battery": {
            "Percent": 100,
            "FullVoltage": 0,
            "UnderVoltage": 0,
            "TotalCapacity": 0,
            "Diverter": 100
        },
        "WorkMode": {
            "Mode": 3
        }
    }
```

参数名	类型	必填	说明
chassisType	int	是	底盘类型（0：双轮 / 1：四轮 / 2：舵轮 / 3：麦克纳姆轮X / 4：麦克纳姆轮O）
WHEEL_ROUND	int	是	车轮周长（单位 mm）
REDUCTION_RATIO	int	是	电机减速比

TWOWHEEL_DISTANCE	int	是	前后轴距（mm）
ENCORDER_ROUND	int	是	编码器一圈脉冲数
MotorFactory	int	是	底盘类型为0，1，3，4时电机厂商（0：雷赛 / 1：步科 / 2：杰美康 / 3：ZLAC8015D / 4：SH60A4A04030 / 5：DS20240C/ 6：ISVL60） 底盘类型为2时电机厂商（0：步科 / 1：同毅 / 2：和利时V1.0 / 3：和利时V2.0/ 4：雷赛/ 5：DS20240C/ 6：ISVL60
SpeekMode	int	是	播报模式（0：关闭，1：开启）
ManualLinearSpeed	float	是	手动控制线速度（m/s）
ManualAngularSpeed	float	是	手动控制角速度（rad/s）
AccTime	float	是	加速时间（s）
DecTime	float	是	减速时间（s）
remote_forward	int	是	遥控器前进按键编号
remote_backward	int	是	遥控器后退按键编号
remote_left_turn	int	是	遥控器左转按键编号
remote_right_turn	int	是	遥控器右转按键编号
remote_left_move	int	是	遥控器左移按键编号（麦克纳姆轮用）
remote_right_move	int	是	遥控器右移按键编号（麦克纳姆轮用）
emergency_stop	int	是	急停按键编号
Switch	int	是	是否启用升降机构（0：关闭，1：开启）
MotorFactory	int	是	电机厂商编号
MotorENCORDER_ROUND	int	是	升降电机编码器一圈脉冲数
REDUCTION_RATIO	int	是	升降电机减速比
DaoCheng	int	是	升降机构导程（mm/转）
UpPosition	int	是	升降最高位置（mm）
FindHomeSpeed	int	是	回零速度（脉冲/s）
NormalSpeed	int	是	正常运行速度（脉冲/s）

Percent	int	是	电池电量百分比（0-100）
FullVoltage	int	是	满电电压（V）
UnderVoltage	int	是	欠压阈值（V）
TotalCapacity	int	是	电池总容量（Ah）
Diverter	int	是	分流器倍率（一般 100 表示 1:100）
Mode	int	是	工作模式（0：：手动 / 3：自动）

- 返回

代码块

```
{
    "message": "底盘参数已发布",
    "published_data": {
        "Battery": {
            "Diverter": 100,
            "FullVoltage": 0,
            "Percent": 100,
            "TotalCapacity": 0,
            "UnderVoltage": 0
        },
        "ChassisData": {
            "AxleSpacing": 1000,
            "ENCORDER_ROUND": 131072,
            "MotorFactory": 0,
            "REDUCTION_RATIO": 30,
            "TWOWHEEL_DISTANCE": 571,
            "WHEEL_ROUND": 180,
            "chassisType": 0
        },
        "WorkMode": {
            "Mode": 3
        },
        "chassisLiftMechanism": {
            "DaoCheng": 5,
            "FindHomeSpeed": 500,
            "MotorENCORDER_ROUND": 10000,
            "MotorFactory": 0,
            "NormalSpeed": 800,
            "REDUCTION_RATIO": 32,
            "Switch": 0,
            "UpPosition": 40
        },
```

```
        "chassisManualMode": {
            "AccTime": 0.8,
            "DecTime": 0.4,
            "ManualAngularSpeed": 0.2,
            "ManualLinearSpeed": 0.3,
            "emergency_stop": 0,
            "remote_backward": 8,
            "remote_forward": 7,
            "remote_left_move": 0,
            "remote_left_turn": 9,
            "remote_right_move": 0,
            "remote_right_turn": 10
        },
        "chassisVoice": {
            "SpeekMode": 1
        }
    },
    "status1": "success"
}
```


- message:响应内容
- published_data:请求的内容
- status: 执行结果

5. 发布底盘命令

- 端点: `POST /pub_ros_to_chassis`
- 描述: 向底盘发布控制命令
- 请求数据格式(json):

代码块


```
{
    "output": {
        "index": 3,
        "state": 1
    }
}
```


参数名	类型	说明
output	Object	输出开关控制

index	int	输出通道号
state	int	1：开启，0：关闭

- 请求数据格式(json):

```
代码块
1  {
2    "exoutput": {
3      "index": 1,
4      "state": 1
5    }
6  }
```

参数名	类型	说明
exoutput	Object	扩展输出开关控制
index	int	输出通道号
state	int	1：开启，0：关闭

- 返回

```
代码块
1  {
2      "message": "底盘命令已发布",
3      "status": "success"
4  }
```

- message:响应内容
- status: 执行结果

6. 发布初始位姿

- 端点: `POST /pub_initial_pose`
- 描述: 发布机器人初始位姿估计
- 请求数据格式(json):

```
代码块
    "x": -0.9288165152072896,
    "y": 0.8421190071851008,
    "angle": 51.241914347415054
}
```

参数名	类型	说明
x	Number	ROS坐标X
y	Number	ROS坐标Y
angle	Number	ROS机器人角度

- 返回

```
代码块
{
    "message": "初始位姿已发布",
    "pose": {
        "angle": 51.241914347415054,
        "x": -0.9288165152072896,
        "y": 0.8421190071851008
    },
    "status": "success"
}
```

- message:响应说明
- pose:请求数据
- status: 执行结果

7. 发布底盘任务

- 端点: `POST /pub_chassis_task`
- 描述: 执行指定次数任务
- 请求数据格式(json):

```
代码块
```

```
{
    "TaskNumber": 1,
    "CyclesNumber": 3
}
```

参数名	类型	说明
TaskNumber	int	任务编号
CyclesNumber	int	执行次数

- status: 执行结果

8. 发布机器人控制命令

- 端点: `POST /pub_robot_control`
- 描述: 发布机器人控制指令-紧急停止
- 请求数据格式(json):

代码块

```
{
    "command": "pause",
    "value": "true"
}
```

参数名	类型	说明
command	int	命令类型 (start:开始, pause:停止)
value	int	参数值 (true/false)

- 返回

代码块

```
{
  "command": "pause",
  "status": "success",
  "value": "true"
}
```

- command:请求类型
- value:参数值 (true/false)
- status: 执行结果

9. 二维码标定

- 端点: `GET/POST /qrcode_command`
- 描述: 调用二维码相关服务功能
- 请求数据格式(json):

```
代码块
1  {
2      "command": 1
3  }
```

参数名	类型	说明
command	int	命令类型 (1:开始标定, 2:停止标定, 3:清空所有, 4:删除指定，5：更新二维码坐标到导航系统)
id	int	二维码ID (命令4时必需)

- 返回

```
代码块
1  {
2      "command": 1,
3      "id": 0,
4      "status": "success"
5  }
```

- command:请求类型
- id:标记二维码ID
- status: 执行结果

10. 启动导航

- 端点: `POST /robot/start_navigation`
- 描述: 开启导航系统
- 请求数据格式(无)
- 返回

```
代码块
1  {
2      "description": "导航任务正在运行",
3      "result": "导航任务已在运行，操作取消",
4      "status": 1
5  }
```

- description:检测结果
- result:执行结果
- status: 执行结果(导航状态码1，为启动导航)

11. 结束导航

- 端点: `POST /robot/stop_navigation`
- 描述: 关闭导航系统
- 请求数据格式(无)
- 返回

```
代码块
1  {
2      "description": "当前没有任务在运行",
3      "result": "导航任务已停止",
4      "status": -1
5  }
```

- description:检测结果
- result:执行结果
- status: 执行结果(导航状态码-1，为结束导航)

12. 开始建图

- 端点: `POST /robot/start_mapping`

- 描述: 启动建图任务
- 请求数据格式(无)
- 返回

```
代码块
1  {
2      "description": "建图任务正在运行",
3      "result": "建图任务已启动",
4      "status": 9
5  }
```

- description:检测结果
- result:执行结果
- status: 执行结果(导航状态码9，启动建图任务)

13. 结束建图

- 端点: `POST /robot/stop_mapping`
- 描述: 结束建图任务
- 请求数据格式(无)
- 返回

```
代码块
1  {
2      "description": "当前没有任务在运行",
3      "result": "建图任务已停止",
4      "saveMap": "",
5      "status": -1
6  }
```

- description:检测结果
- result:执行结果
- saveMap:保存地图
- status: 执行结果(导航状态码-1，结束建图任务)

14. 保存建图

- 端点: `POST /robot/save_map`
- 描述: 保存建图需要在建图状态下才能保存地图
- 请求数据格式(无)
- 返回

```
代码块
1  {
2      "description": "建图任务正在运行",
3      "result": "地图 'map' 已保存成功",
4      "status": 9
5  }
```

- description:检测结果
- result:执行结果
- status: 执行结果(导航状态码9，保存建图任务)

15. 急停/恢复

- 端点: `POST /pub_robot_control`
- 描述: 紧急停止于恢复任务
- 请求数据格式(json):

```
代码块
1  {
2      "command": "pause",
3      "value": "true"
4  }
```

参数名	类型	说明
command	string	命令参数： pause
value	string	true:开启暂停　false:取消暂停恢复任务

- 返回

代码块

```
{
    "command": "pause",
    "status": "success",
    "value": "false"
}
```

- command:执行的命令
- success:执行结果
- value: 发送的值