目錄

引言

伽利略视觉导航系统是一个融合了多种传感器以视觉导航为主导的机器人定位和运动控制系统。通过加载本系统可以实现机器人的定位和导航功能。适用于自动巡检机器人工业AGV,服务机器人等多种应用场景。

建议在使用API之前先通过使用机器人了解整个工作原理和流程。机器人使用手册

整体工作流程

机器人工作主要分为两个部分，建图和导航。首先开启机器人建图状态后，遥控机器人在目标环境中移动。机器人会自动创建环境地图。创建地图后调用保存接口，地图保存在机器人上后可以供导航使用。 在创建的地图中添加机器人移动的路径和目标点数据。之后就可以启动导航了。

跨地图导航

在不同地图中添加目标点的关联信息后即可在不同的地图间导航。

跨楼层导航

跨楼层导航需要创建楼层地图和电梯地图。其基本原理和跨地图导航一样。通过添加连接点把楼层地图和电梯地图连接起来。机器人会自动根据目标点所在的楼层和当前位置找出应该采取的路线和动作。 注意电梯地图的0号点必须在电梯内部。

伽利略导航系统HTTP协议说明

程序实例

C# 版本

Python 版本

Java 版本

API的格式为API版本号加上对应的URL，以获取系统状态API为例，实际请求地址为/api/v1/system/status。API返回值都是json格式的数据。下面的文档将省略API前缀。API服务程序默认端口为3546。 机器人在启动后会向局域网发送UDP广播，端口为22002。通过此广播我们能够获取到局域网中的机器人基本信息。下面是一个具体的广播数据例子

```
{"id": "F072E1BA8162245572D2FAEEB2526C5DD916F5A0D6D0F8A14B67FA43DC501079461280B15BA5", "port": 3546, "mac": "00:e0:4c:68:6f:0f", "version": "5.0.0"}
```

广播数据包含了机器人ID，机器人Http服务端口号，机器人mac和机器人当前的http服务版本号。

对于Http调用的参数，GET和DELETE方法的参数放在URL query string里面。POST和PUT方法的参数放在body里面。

对于HTTP协议不熟悉的用户可以先参考这个文档

跨局域网调用API

通过伽利略网络代理我们可以实现远程跨局域网的机器人API调用

机器人远程代理设置网址

机器人远程调用说明

HTTP 服务的参数配置

http服务参数配置文件位于 `/home/xiaoqiang/Documents/ros/src/galileo_api/config.json`

其默认内容如下

```
{
    "username": "admin",
    "password": "admin",
    "allow_update": true,
    "no_token_check": true,
    "auto_charge": false,
}
```

参数	类型	说明
username	string	获取机器人token时的用户名。默认为admin
password	string	获取机器人token时的密码。默认为admin
allow_update	bool	是否允许自动更新。当为true时程序有更新时客户端会收到机器人更新提示。反之则不会有更新提示。
no_token_check	bool	是否开启token验证功能。默认不开启
auto_charge	bool	是否开启低电量自动充电功能，默认不开启。注意开启此功能要保证机器人导航地图中有正确的充电桩位置，同时导航地图能够正常工作

token API

为了系统的安全性，在调用机器人api的时候可以加上token验证的功能。此功能默认关闭，可以通过配置http api参数打开此功能。

URL: /token

请求方式: GET

请求参数:

参数	类型	说明

username	string	配置文件中的用户名，默认为admin
password	string	配置文件中的密码, 默认为admin

返回值

参数	类型	说明
result	bool	是否成功获取token
token	string	获取的token

获取token后在调用api时可以在url参数中加入token=xxx,xxx为你的token数据。 对于POST或PUT请求，token参数可以加在url中也可以在body的json数据里面。

下面是一个调用的例子

```
http://192.168.0.132:3546/api/v1/system/info?token=28b1c500400611ebb805493c9303c705
```

其中192.168.0.132为机器人IP，28b1c500400611ebb805493c9303c705为机器人token。

系统状态API

获取系统当前状态

URL: /system/status

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	状态String,可能的值为 Mapping, Navigating, Busy, Free

说明：系统可能处在互斥的几种状态中。通过不同的操作API系统在不同的状态间切换。只有在特定的状态下系统才能执行特定的API。

获取系统基本信息

URL: /system/info

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明

battery	int	电池电量百分比
camera_rgb	int	rgb摄像头topic发布频率，0代表没有数据
camera_depth	int	深度摄像头topic发布频率，0代表没有数据
odom	int	编码器topic发布频率，0代表没有数据
imu	int	IMU topic的发布频率，0代表没有数据
driver_port	bool	急停按钮状态，true为按下，false为未按下
charge	bool	是否正在执行充电任务
loop	bool	是否正在执行循环任务
info	object	机器人基本信息,4.5.0和5.1.0之后版本才会有此返回

说明: charge状态和galileo_status中的状态并不一样。charge状态是整个充电任务的状态而galileo status中的充电状态为局部充电任务状态。

示例返回


```
{
    "battery": 100,
    "camera_rgb": 19,
    "camera_depth": 29,
    "odom": 26,
    "imu": 49,
    "camera_processed": 30,
    "driver_port": false,
    "info": {
        "version": "6.1.5",
        "code_name": "chitu-noetic",
        "id": "AE13B83EE2846276882EE47A99391C89CD2EF6B6878D5D309F80755F8E3B7D15CB0CB9BEF55
D",
        "mac": "00:15:00:b9:90:a6",
        "port": 3546
    },
    "charge": false,
    "loop": false,
    "slam_type": "camera"
}
```


注意第一次调用时，由于需要对数据统计，可能对应的数据返回为*0*.之后调用返回数据将为正常

获取Galileo Status

URL: /system/galileo_status

请求方式: GET

请求参数: 无

返回参数：当前的galileo status数据, galileo status是对系统整体状态的一个汇报参数。具体定义如下

例子

```
{
  "header": {
    "seq": 542928,
    "stamp": {
      "secs": 1624970817,
      "nsecs": 524690511
    },
    "frame_id": "map"
  },
  "navStatus": 0, // 导航服务状态，0表示没开启closed，1表示开启opened。
  "visualStatus": -1, // 视觉系统状态，-1标系视觉系统处于关闭状态，0表示没初始化uninit，1表示正在追踪tracking,2表示丢失lost,1和2都表示视觉系统已经初始化完成。
  "mapStatus": 0, // 建图服务状态，0表示未开始建图，1表示正在建图
  "gcStatus": 0, // 内存回收标志，0表示未进行内存回收，1表示正在进行内存回收
  "gbaStatus": 0, // 闭环优化标志，0表示未进行闭环优化，1表示正在进行闭环优化
  "chargeStatus": 0, // 充电状态，0 free 未充电状态， 1 charging 充电中， 2 charged 已充满，但仍在小电流充电， 3 finding 寻找充电桩， 4 docking 停靠充电桩， 5 error 错误
  "loopStatus": 0, // 是否处于自动巡检状态，1为处于，0为不处于。
  "power": 37.05362319946289, // 电源电压v。
  "targetNumID": -1, // 当前目标点编号,默认值为-1表示无效值，当正在执行无ID的任务是值为-2，比如通过Http API 创建的导航任务。
  "targetStatus": 0, // 当前目标点状态，0表示已经到达或者取消free，1表示正在前往目标点过程中working,2表示当前目标点的移动任务被暂停paused,3表示目标点出现错误error,默认值为-1表示无效值。
  "targetDistance": -1, // 机器人距离当前目标点的距离，单位为米，-1表示无效值，该值的绝对值小于0.01时表示已经到达。
  "angleGoalStatus": 1, // 目标角度达到情况，0表示未完成，1表示完成，2表示error,默认值为-1表示无效值。
  "controlSpeedX": 0, // 导航系统计算给出的前进速度控制分量,单位为m/s。
  "controlSpeedTheta": 0, // 导航系统计算给出的角速度控制分量,单位为rad/s。
  "currentSpeedX": -0.000001009553670883178 7, // 当前机器人实际前进速度分量,单位为m/s。
  "currentSpeedTheta": 0.0002659947786014527,// 当前机器人实际角速度分量,单位为rad/s。
  "currentPosX": -1, // 当前机器人在map坐标系下的X坐标,此坐标可以直接用于设置动态插入点坐标
  "currentPosY": -1, // 当前机器人在map坐标系下的Y坐标
  "currentAngle": -1, // 当前机器人在map坐标系下的z轴转角(yaw)
  "busyStatus": 0 //当busy为true时系统将仍然后接收新指令，但是不会立即处理。当系统退出busy状态后再处理消息
}
```

获取系统日志

URL: /system/log

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
log	string	系统日志信息

获取机器人速度

URL: /system/speed

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
speed_x	float	x方向速度，单位为米每秒
speed_y	float	y方向速度，单位为米每秒
speed_angle	float	转动角速度, 单位是弧度每秒

说明:

机器人本体坐标系方向为，机器人正前方为x方向，机器人左方为y方向，机器人上方为z方向。y方向的速度是全向轮如麦克纳姆轮才有的。对于一般差速底盘y方向速度一直为0

遥控机器人

URL: /system/speed

请求方式: PUT

请求参数:

参数	类型	说明
speed_x	float	x方向速度
speed_y	float	y方向速度
speed_angle	float	转动角速度

返回参数:

参数	类型	说明
result	bool	遥控指令是否成功执行

说明: 调用此API后机器人会以指定的速度移动。移动的时间是不确定的，这个由驱动器决定，一般是几秒的时间。实际开发遥控功能时可以以5Hz左右的频率调用此API。 对于6.2.1之后版本的机器人，遥控也可以通过websocket实现，这样可以极大的提高响应速度，尤其对于使用跨网访问接口。通过websocket向/cmd_vel话题发送Twist消息即可。

关闭机器人

URL: /system/shutdown

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
result	bool	是否成功接收关机指令

校正机器人陀螺仪

URL: /system/calibrate_imu

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
result	bool	是否开始校准陀螺仪

开始校正摄像头角度位置参数

URL: /system/calibrate_camera/start

请求方式: GET

请求参数:

参数	类型	说明
camera_id	int	当没有此参数时为单摄像头标定，当为0时为前摄像头标定，当为1时为后摄像头标定

说明：执行启动校准方法后需要遥控机器人在环境中运动，使其路径形成一个闭环完成闭环优化。闭环优化完成后，继续移动一段距离，然后调用提交参数方法完成参数校准。

请求参数: 无

返回参数:

参数	类型	说明
result	string	是否开始校准陀螺仪

取消摄像头角度位置校正参数

URL: /system/calibrate_camera/cancel

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
result	string	是否取消校准陀螺仪

获取摄像头角度位置校正状态

URL: /system/calibrate_camera/status

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	标定状态，可以是CALIBRATING或CALIBRATED。分别表示标定中和标定完成
accuracy	float	在标定完成时候会包含此参数，表明标定准确度，最大值为100

提交摄像头角度位置校正参数

URL: /system/calibrate_camera/complete

请求方式: GET

说明： 只有在处于CALIBRATED状态下才能够提交标定参数

请求参数: 无

返回参数:

参数	类型	说明
status	string	标定参数是否提交成功

检查系统更新状态

URL: /system/update/check

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	检查更新程序是否成功执行
need_update	bool	是否需要更新

开始自动更新程序

URL: /system/update/start

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	系统更新程序是否成功启动

取消自动更新程序

URL: /system/update/cancel

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	取消系统更新是否成功

获取软件更新日志

URL: /system/update/log

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	是否成功获取更新日志
log	string	系统更新日志内容
is_complete	bool	系统更新是否已经完成

获取当前调度服务器

URL: /system/schedule_manager

请求方式: GET

请求参数: 无

返回参数: 所有schedule manager数据，其中第一个是当前的schedule manager

返回数据示例


```
[
  {
    "ip": "192.168.0.23",
    "update_time": 1576115449217,
    "version": "1.0.0",
    "id": "9188d590-8508-47bd-a704-d34ddb803265",
    "port": 24958
  }
]
```


系统自检

URL: /system/self_test

请求方式: GET

返回参数:

参数	类型	说明
motor_driver	bool	底盘驱动状态是否正常
camera	bool	摄像头状态是否正常
charge	bool	自动充电模块是否正常
lidar	bool	雷达是否正常
bluetooth	bool	蓝牙是否工作正常
battery	bool	获取电池电压是否正常

获取当前io电平情况

注意io仍为电平输出，获取的是输出电平的状态，并不是输入信号

URL: /system/io

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	获取状态
out1	string	"-1"为未设置，"0"为低电平, "1"为高电平
out2	string	"-1"为未设置，"0"为低电平, "1"为高电平
out3	string	"-1"为未设置，"0"为低电平, "1"为高电平

控制当前io电平情况

URL: /system/io

请求方式: POST

请求参数:

参数	类型	说明
level	string	0 为低电平， 1为高电平
port	string	可以为1,2,3分别对应三个IO端口

返回参数:

参数	类型	说明
status	string	获取状态
status	string	设置电平状态

获取机器人当前配置参数

URL: /system/config

请求方式: GET

请求参数: 无

返回参数:

5.0 以前版本返回

参数	类型	说明
default_map	string	默认地图名称
maps	list	地图对应默认路径配置
navigation_speed	float	最大导航速度
max_control_speed	float	最大遥控速度
bar_distance_min	float	避障距离
k2	float	PID控制参数k2
kp	float	PID控制参数kp
ki	float	PID控制参数ki
kd	float	PID控制参数kd
look_ahead_dist	float	预估距离
theta_max	float	最大角速度

plan_width	float	车体宽度
path_change	float	避障时是否绕开
forward_width	float	预估距离
rot_width	float	车体旋转宽度
backtime	float	最大后退距离

5.0 以后版本返回

参数	类型	说明
default_map	string	默认地图名称
maps	list	地图对应默认路径配置
navigation_speed	float	最大导航速度
auto_charge	bool	是否开启低电量自动充电
slam_type	string	导航类型 "camera"为摄像头导航，"lidar"为雷达导航
lf_linear_v	float	最大导航速度
lf_theta_max	float	最大导航角速度
lf_k2	float	PID控制参数k2
lf_kp	float	PID控制参数kp
lf_ki	float	PID控制参数ki
lf_kd	float	PID控制参数kd
lf_look_ahead_dist	float	预估距离
robot_radius	float	机器人半径
lf_xy_goal_tolerance	float	位置允许误差
lf_yaw_goal_tolerance	float	角度允许误差
astar_path_changeability	bool	是否绕开
lf_bar_distance_min	float	避障距离
lf_forward_max_dist	float	避障最远检测距离
lf_move_direction	int	导航时机器人方向，0正向，1反向，2自动
go_charge_percentage	float	返回充电电量百分比，需要6.1.7及以上版本
use_smooth_switch_map	float	是否使用导航流畅切换，开启后机器人切换地图时会更加流畅，但是跨地图导航时机器人会消耗更多CPU

修改机器人配置

URL: /system/config

请求方式: POST

请求参数:

5.0 版本之前参数

参数	类型	说明
default_map	string	默认地图名称，可选参数
maps	list	地图对应默认路径配置，可选参数
navigation_speed	float	最大导航速度，可选参数
max_control_speed	float	最大遥控速度，可选参数
bar_distance_min	float	避障距离，可选参数
k2	float	PID控制参数k2，可选参数
kp	float	PID控制参数kp，可选参数
ki	float	PID控制参数ki，可选参数
kd	float	PID控制参数kd，可选参数
look_ahead_dist	float	预估距离，可选参数
theta_max	float	最大角速度，可选参数
plan_width	float	车体宽度，可选参数
path_change	float	避障时是否绕开，可选参数
forward_width	float	预估距离，可选参数
rot_width	float	车体旋转宽度，可选参数
backtime	float	最大后退距离，可选参数
deliver_wait_time	float	送餐最长等待时间，可选参数

5.0版本之后参数

参数	类型	说明
default_map	string	默认地图名称
maps	list	地图对应默认路径配置
navigation_speed	float	最大导航速度
auto_charge	bool	是否开启低电量自动充电
slam_type	string	导航类型 "camera"为摄像头导航，"lidar"为雷达导航
lf_linear_v	float	最大导航速度
lf_theta_max	float	最大导航角速度
lf_k2	float	PID控制参数k2
lf_kp	float	PID控制参数kp
lf_ki	float	PID控制参数ki

lf_kd	float	PID控制参数kd
lf_look_ahead_dist	float	预估距离
robot_radius	float	机器人半径
lf_xy_goal_tolerance	float	位置允许误差
lf_yaw_goal_tolerance	float	角度允许误差
astar_path_changeability	bool	是否绕开
lf_bar_distance_min	float	避障距离
lf_forward_max_dist	float	避障最远检测距离
lf_move_direction	int	导航时机器人方向，0正向，1反向，2自动
go_charge_percentage	float	返回充电电量百分比，需要6.1.7及以上版本
use_smooth_switch_map	float	是否使用导航流畅切换，开启后机器人切换地图时会更加流畅，但是跨地图导航时机器人会消耗更多CPU

返回参数:

修改后的机器人参数

5.0 以前版本返回

参数	类型	说明
default_map	string	默认地图名称
maps	list	地图对应默认路径配置
navigation_speed	float	最大导航速度
max_control_speed	float	最大遥控速度
bar_distance_min	float	避障距离
k2	float	PID控制参数k2
kp	float	PID控制参数kp
ki	float	PID控制参数ki
kd	float	PID控制参数kd
look_ahead_dist	float	预估距离
theta_max	float	最大角速度
plan_width	float	车体宽度
path_change	float	避障时是否绕开
forward_width	float	预估距离
rot_width	float	车体旋转宽度
backtime	float	最大后退距离

5.0 以后版本返回

参数	类型	说明
default_map	string	默认地图名称
maps	list	地图对应默认路径配置
navigation_speed	float	最大导航速度
auto_charge	bool	是否开启低电量自动充电
slam_type	string	导航类型 "camera"为摄像头导航，"lidar"为雷达导航
lf_linear_v	float	最大导航速度
lf_theta_max	float	最大导航角速度
lf_k2	float	PID控制参数k2
lf_kp	float	PID控制参数kp
lf_ki	float	PID控制参数ki
lf_kd	float	PID控制参数kd
lf_look_ahead_dist	float	预估距离
robot_radius	float	机器人半径
lf_xy_goal_tolerance	float	位置允许误差
lf_yaw_goal_tolerance	float	角度允许误差
astar_path_changeability	bool	是否绕开
lf_bar_distance_min	float	避障距离
lf_forward_max_dist	float	避障最远检测距离
lf_move_direction	int	导航时机器人方向，0正向，1反向，2自动
go_charge_percentage	float	返回充电电量百分比，需要6.1.7及以上版本
use_smooth_switch_map	float	是否使用导航流畅切换，开启后机器人切换地图时会更加流畅，但是跨地图导航时机器人会消耗更多CPU

恢复机器人默认参数

URL: /system/config

请求方式: DELETE

请求参数

参数	类型	说明
key	string	对应参数的名称,如k2,kp,ki等等

返回参数:

参数	类型	说明

status	string	恢复默认值操作状态

让机器人播放语音

URL: /system/tts

请求方式: GET

请求参数:

参数	类型	说明
text	string	需要机器人播放的语音对应的文本

返回参数:

参数	类型	说明
status	string	语音播放状态

说明: 机器人语音文件在第一次播放时需要联网下载，所以机器人未联网情况下无法播放新语音。成功播放后语音文件会自动缓存在机器人中，之后播放不再需要网络连接。

检查机器人证书

URL: /system/check_cert

请求方式: GET

请求参数: 无

返回值

```
{
  "status": "ok"
}
```

让机器人连接wifi

URL: /system/wifi

请求方式: GET

请求参数:

参数	类型	说明
ssid	string	wifi的名称
password	string	wifi密码

返回参数:

参数	类型	说明
status	string	状态说明，默认ok

注意: wifi最后是否连接成功是以机器人能否访问互联网判断的。如果你所要连接的wifi不能连接互联网则即使机器人成功连接了wifi，此时机器人仍然会返回连接错误。

是否使用预览版程序

URL: /system/use_dev

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
use_dev	bool	是否使用预览版

设置使用预览版

URL: /system/use_dev

请求方式: PUT

请求参数:

参数	类型	说明
use_dev	bool	是否使用预览版

返回参数:

参数	类型	说明
use_dev	bool	是否使用预览版

是否开启跟随功能

URL: /system/tracking_mode

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
tracking_flag	bool	是否开启跟随功能

开启关闭跟随功能

URL: /system/tracking_mode

请求方式: PUT

请求参数:

参数	类型	说明
enable	bool	是否开启跟随功能

获取系统音量设置

URL: /system/volume

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
value	int	音量百分比

设置系统音量

URL: /system/volume

请求方式: PUT

请求参数:

参数	类型	说明
value	int	音量百分比

返回参数:

参数	类型	说明
status	string	ok

创建地图API

启动创建地图

URL: /map/start

请求方式: GET

说明: 接受到启动命令后系统会处于Busy状态，线程启动成功后系统进入Mapping状态。

请求参数:

参数	类型	说明
camera_id	int	可选参数，当没有此参数时为单摄像头建图，为0时为前摄像头建图，为1时为后摄像头建图

返回参数:

参数	类型	说明
result	bool	是否成功启动导航

结束创建地图线程

URL: /map/stop

请求方式: GET

说明: 接收到结束命令进入Busy状态。后先自动保存地图，然后开始关闭建图线程，线程关闭后系统进入Free状态。

请求参数: 无

返回参数

参数	类型	说明
result	bool	是否成功结束建图

更新地图

URL: /map/update

请求方式: GET

请求参数:

参数	类型	说明
map	string	可选参数，需要更新的地图，当没有此参数时则更新上次导航使用的地图
path	string	可选参数,配合start_index使用设置机器人初始位置，对于雷达导航是必须的
start_index	int	可选参数，设置机器人的初始位置，对于雷达导航是必须的

返回参数:

参数	类型	说明
result	bool	是否成功开启更新地图

当前机器人位置

URL: /map/pose

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
x	float	机器人当前x坐标，单位为米
y	float	机器人当前y坐标，单位为米
angle	float	机器人当前朝向角度， 单位为度

获取当前正在创建的地图信息

URL: /map/current_map_image

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	是否成功获取地图信息
map_params	object	地图的meta信息
map_image	string	获取当前地图图片的URL路径

返回数据示例:


```
{
    "status": "OK",
    "map_params": {
        "origin": [
            -7.621091365814209,
            -9.969059944152832,
            0.0
        ],
        "width": 306,
        "resolution": 0.05000000074505806,
        "height": 324
    },
    "map_image": "/api/v1/map/current_map_png"
}
```

说明：机器人图片像素坐标和实际坐标的转换关系为标准的ROS地图关系。像素坐标的原点为图片左上角，x正方向向右，y正方向向下。地图坐标原点为左下角，x正方向向右，y正方向向上。转换关系如下

```
x_map = x_origin + x_pixel * res
y_map = y_origin + h_map * res - y_pixel * res
```

其中 x_map, y_map为像素点对应的地图坐标

x_origin, y_origial 为左下角原点坐标

x_pixel, y_pixel 为像素坐标

h_map为地图像素高度

保存当前创建的地图

URL: /map/current_map_image

请求方式: POST

请求参数:

参数	类型	说明
name	string	创建的地图名称
type	string	地图类型，可选参数。当为floor时代表此地图为普通楼层地图，当为elevator时代表此地图为电梯地图，默认为floor
levels	int[]	地图楼层，可选参数，此地图所代表的楼层。不同楼层可以共用一个地图，默认为1

返回参数:

参数	类型	说明
result	bool	地图保存是否成功
name	string	保存后的地图名称

获取当前创建地图的png图片

URL: /map/current_map_png

请求方式: GET

请求参数: 无

返回参数: 当前正在创建的地图的png图片

获取创建地图时机器人的运动轨迹

URL: /map/track

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
trajectory	list	创建地图时机器人运动轨迹, 只有在保存地图后才能获取数据

返回数据示例:


```
{
  "trajectory": [
    {
      "y": 0.3912552,
      "x": 0.3480716,
      "z": -0.0384934
    },
    {
      "y": 0.3972594,
      "x": 0.4466029,
      "z": -0.038688
    },
    {
      "y": 0.403202,
      "x": 0.5998781,
      "z": -0.0389315
    },
    ...
  ]
}
```


导航部分

启动导航状态

URL: /navigation/start

请求方式: GET

说明： 接收到启动命令后系统会处于Busy状态，线程启动成功后系统进入Navigating状态。启动命令参数中包含导航地图文件名字。

请求参数:

参数	类型	说明
map	string	导航所使用的地图名称，当没有此参数时机器人采用上次启动导航时的地图
path	string	导航所使用的路径名称，当没有此参数时机器人采用上次启动导航的路径

start_index	int	机器人初始位置，有此参数时机器人会采用对应目标点位置作为初始位置。如start_index为1时机器人采用1号导航点位置作为初始位置。 当没有此参数时机器人将采用上次启动导航时设置的start_index。对于视觉导航此参数不是必须的。对于雷达导航此参数是必须的。当设置为-1时将采用充电桩位置作为初始位置。当摄像头导航且没有此参数时采用自动定位模式
level	int	可选参数，当前机器人所在楼层。如果没有此参数默认为1

返回参数

参数	类型	说明
result	bool	是否成功启动导航
description	string	若启动失败，其原因说明

启动调度导航

URL: /navigation/start_schedule_mode

请求方式: GET

说明: 接收到启动命令后系统会处于Busy状态，线程启动成功后系统进入Navigating状态。此导航状态不同于普通导航状态。伽利略导航功能将不能使用，机器人将由拉格朗日调度系统控制。

请求参数: 无

返回参数:

参数	类型	说明
result	bool	是否成功启动调度导航
description	string	若启动失败，其原因说明

停止导航

URL: /navigation/stop

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
result	bool	是否成功启动调度导航
description	string	若启动失败，其原因说明

停止调度导航

URL: /navigation/stop_schedule_mode

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
result	bool	是否成功启动调度导航
description	string	若启动失败，其原因说明

机器人当前位置信息

URL: /navigation/pose

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
x	float	机器人x坐标，单位为米
y	float	机器人y坐标，单位为米
angle	float	机器人朝向角度，单位为度

获取当前导航正在使用的地图和路径名称

URL: /navigation/current_path

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
map	string	当前导航使用的地图名称
path	string	当前导航使用的路径名称

获取当前位置到目标点的距离

URL: /navigation/target_distance

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
distance	float	当前位置到目标点的距离

获取已经保存的详细地图信息

URL: /navigation/saved_maps

请求参数: GET

请求参数:

参数	类型	说明
name	string	目标地图的名称，可选参数。当没有此参数时返回所有已保存的地图信息。当有此参数时，返回目标地图的信息
type	string	地图类型，可选参数。为lidar时返回雷达地图，为camera时返回摄像头地图，为all时返回所有类型地图

返回参数:

参数	类型	说明
resolution	float	地图分辨率，即每像素点对应的实际距离
origin	object	地图原点坐标
occupied_thresh	float	占用阈值，一般没什么用
free_thresh	float	未占用阈值，一般没什么用
negate	int	是否反转占用和未占用，一般没什么用
name	string	目标地图名称
md5sum	string	目标地图数据的md5sum值，用于判断地图是否是一样的
map_type	string	地图类型，值为floor时表示为楼层地图，为elevator时表示为电梯地图
levels	int[]	地图代表的楼层
slam_type	string	地图建图类型，lidar表示为雷达地图，camera表示为摄像头地图

获取保存的地图中的机器人运动轨迹

URL: /navigation/robot_tracks

请求方式: GET

请求参数:

参数	类型	说明
		目标路径的地图名称。可选参数，没有此参数时返回所有地图中的机器人

map_name	string	路径数据。当有此参数时只返回目标地图的机器人路径数据

返回参数:

当请求参数有map_name时

参数	类型	说明
trajectory	list	机器人运动轨迹点列表

当请求参数有map_name时

返回所有地图的轨迹点列表，包含地图名称和轨迹点数据

返回数据示例:

当没有map_name参数时

```
[
  {
    "map_name": "map1",
    "track": [
      {
        "y": 0.0000016,
        "x": -0.0010667,
        "z": -0.0000129
      },
      {
        "y": 0.0084065,
        "x": 0.0657083,
        "z": -0.0001012
      },
      {
        "y": -0.0036619,
        "x": 0.1893717,
        "z": -0.0003028
      },
      ...
    ]
  }
]
```

获取目标地图的PGM图片信息

URL: /navigation/map_pgm

请求方式: GET

请求参数:

参数	类型	说明
name	string	目标地图名称

返回目标地图的pgm图片数据

获取目标地图的png图片信息

URL: /navigation/map_png

请求方式: GET

参数	类型	说明
name	string	目标地图名称

返回目标地图的png图片数据

获取当前正在使用的地图

URL: /navigation/current_map

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
name	string	目标地图名称
md5sum	string	目标地图数据的md5sum值

上传当前机器人地图数据至调度服务器

URL: /navigation/upload_map

请求方式: GET

说明: 此方法是异步方法，调用后会在系统中创建一个上传地图数据的线程。上传进度可以通过task相关api获取。使用前一定要保证调度服务器和机器人可以正常通信，且机器人处于被激活状态。

请求参数:

参数	类型	说明
server_id	string	调度服务器id

返回参数:

地图上传任务数据

示例返回数据:


```
{
    "name": "upload_map",
    "loop_flag": false,
    "id": "8bc1e6dd-df1f-4a78-a17b-c60911ce5a06",
    "state": "WORKING",
```

```
    "sub_tasks": [
        {
            "server_id": "9188d590-8508-47bd-a704-d34ddb803265",
            "state": "WORKING",
            "result": "",
            "progress": 0,
            "type": "upload_map_action",
            "id": "e21c743d-c692-4ed2-b2be-1470da60ba1e"
        }
    ],
    "progress": 0.0,
    "current_task": {
        "server_id": "9188d590-8508-47bd-a704-d34ddb803265",
        "state": "WORKING",
        "result": "",
        "progress": 0,
        "type": "upload_map_action",
        "id": "e21c743d-c692-4ed2-b2be-1470da60ba1e"
    }
}
```


从调度服务器下载地图

URL: /navigation/download_map

请求方式: GET

请求参数:

参数	类型	说明
server_id	string	调度服务器id
map_id	string	需要下载的地图id

返回参数:

地图下载任务数据

返回数据示例:


```
{
    "name": "download_map",
    "loop_flag": false,
    "id": "fb7669d2-5f03-4461-a992-95422f66d49d",
    "state": "WORKING",
    "sub_tasks": [
        {
            "server_id": "9188d590-8508-47bd-a704-d34ddb803265",
            "map_id": "04f777b8-ff9d-4303-87ac-334dab2e0ffe",
            "state": "WORKING",
            "result": "",
            "progress": 0,
            "type": "download_map_action",
            "id": "2fb7eebc-8048-47d8-b62a-60365c772f87"
        }
    ],
```

```
    "progress": 0.0,
    "current_task": {
        "server_id": "9188d590-8508-47bd-a704-d34ddb803265",
        "map_id": "04f777b8-ff9d-4303-87ac-334dab2e0ffe",
        "state": "WORKING",
        "result": "",
        "progress": 0,
        "type": "download_map_action",
        "id": "2fb7eebc-8048-47d8-b62a-60365c772f87"
    }
}
```


下载机器人地图

URL: /navigation/download_map_from_robot

请求方式: GET

请求参数:

参数	类型	说明
map_name	string	需要下载的地图名称

返回数据: 地图文件压缩包

上传地图至机器人

URL: /navigation/upload_map_to_robot

请求方式: POST

请求参数:

参数	类型	说明
map_name	string	上传地图名称
map_files	文件	下载地图API所产生的地图文件压缩包

注意: 此方法的上传数据不是json格式，是MultipartFormData格式

返回

参数	类型	说明
status	string	完成状态,默认ok

开启导航任务

URL: /navigation/start_nav_task

请求方式: POST

请求参数:

参数	类型	说明
x	float	导航目标点坐标x值，单位为米
y	float	导航目标点坐标y值，单位为米
theta	float	导航目标点角度值,单位为弧度
map	string	可选参数,目标点所在的地图
path	string	可选参数,目标点所在的路径

说明:

当没有map和path参数时，机器人在当前地图中移动至对应坐标。如果包含map和path参数，机器人将自动寻找地图的连接点，并在移动过程中自动切换地图，最终到达目标地图中的目标点。此功能可以用来跨地图导航，比如，在不同的楼层间，不同的区域中进行导航。

返回参数:

导航任务数据

示例返回数据:


```
{
  "name": "navigation task",
  "loop_flag": false,
  "id": "a6858132-5e84-4d37-b968-19808dd8fd96",
  "state": "WORKING",
  "sub_tasks": [
    {
      "state": "WAITTING",
      "result": "",
      "progress": 0,
      "theta": 1.0936689376831055,
      "y": -0.1287004053592682,
      "x": -0.28466343879699707,
      "current_location": {
        "y": -1,
        "x": -1,
        "theta": -1
      },
      "type": "nav_action",
      "id": "99bfb264-8f8f-4950-940b-6f6b883e7358"
    }
  ],
  "progress": 0,
  "current_task": {
    "state": "WAITTING",
    "result": "",
    "progress": 0,
    "theta": 1.0936689376831055,
    "y": -0.1287004053592682,
    "x": -0.28466343879699707,
    "current_location": {
      "y": -1,
      "x": -1,
```

```
        "theta": -1
      },
      "type": "nav_action",
      "id": "99bfb264-8f8f-4950-940b-6f6b883e7358"
    }
  }
```


开始自动充电

URL: /navigation/go_charge

请求方式: GET

请求参数: 无

返回参数:

返回充电任务对象

停止自动充电

URL: /navigation/stop_charge

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	停止充电操作状态
task	object	充电任务对象

移动到对应目标点

URL: /navigation/move_to_index

请求方式: GET

请求参数:

参数	类型	说明
index	int	目标点index，即绘制路径时插入导航点的index
map	string	目标位置所在地图，可选参数
path	string	目标位置所在的路径
level	int	可选参数，当有此参数时会自动通过电梯移动到目标位置

说明： 当没有map和path参数时，机器人会在当前地图上移动到目标位置。如果包含map和path参数，机器人将自动寻找地图的连接点，并在移动过程中自动切换地图，最终到达目标地图中的目标点。此功能可以用来跨地图导航，比如，在不同的楼层间，不同的区域中进行导航。

返回参数:

返回导航任务对象

取消当前导航任务

URL: /navigation/stop_nav_task

请求方式: GET

请求参数: 无

返回参数:

返回当前导航任务对象

获取导航循环任务信息

导航循环任务为沿着插入的导航点依次移动的导航任务，其效果和点击客户端上的循环选项一样。

URL: /navigation/loop_task

请求方式: GET

请求参数: 无

返回参数:

返回循环导航任务状态

开始导航循环任务

URL: /navigation/loop_task

请求方式: POST

请求参数

参数	类型	说明
wait_time	int	循环到达目标点后的等待时间

返回参数:

返回新创建的导航循环任务，如果已有导航循环任务则会返回错误

停止导航循环任务

URL: /navigation/loop_task

请求方式: DELETE

请求参数: 无

返回参数:

返回当前的导航循环任务

获取充电桩位置

URL: /navigation/charge_pose

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
x	float	充电桩位置坐标x
y	float	充电桩位置坐标y
theta	float	充电桩角度theta

保存充电桩位置

URL: /navigation/charge_pose

请求方式: POST

请求参数: 无

返回参数:

参数	类型	说明
x	float	充电桩位置坐标x
y	float	充电桩位置坐标y
theta	float	充电桩角度theta

动态切换地图

URL: /navigation/reload

method: GET

请求参数:

参	类型	说明

数	类型	说明
map	string	想要切换的地图名称
path	string	切换的路径名称，可选参数，当没有此参数时，路径使用机器人设置里面的默认路径

说明:

此方法为动态切换地图，导航程序不会重启。所以切换速度会很快。推荐在机器人需要切换场景时使用，比如上下电梯后不同楼层地图的切换。

返回参数:

```
{
  "status": "ok",
  "map": "",
  "path": ""
}
```

获取导航路径点

URL: /navigation/path

请求方式: GET

请求参数

参数	类型	说明
map_name	string	可选参数，获取的目标地图名称
path_name	string	可选参数，获取的目标路径名称

说明:

当同时有map_name和path_name参数的时候，返回数据为对应路径文件的内容。如下

```
{
  "map_name": "map1",
  "path_name": "path1",
  "path_data": "0.575 -0.356 0\n6.98246 17.95874 0\n9.969936 15.64437 0\n-2.099531 2.19303
4 0\n-2.099531 2.193034 0\n-2.099531 2.193034 0\n-2.099531 2.193034 0\n-3.247468 4.031781
0\n-2.099531 2.193034 0\n6.98246 17.95874 0\n6.98246 17.95874 0\n0.575 -0.356 0\n9.969936
15.64437 0\n-2.099531 2.193034 0\n-2.099531 2.193034 0\n-2.099531 2.193034 0\n-2.099531 2.
193034 0\n-3.247468 4.031781 0\n-2.099531 2.193034 0\n-1.278543 -2.399499 0\n-3.247468 4.0
31781 0\n-2.099531 2.193034 0\n12.3693 12.11829 0\n9.969936 15.64437 0\n-1.928585 1.842135
 0\n-1.278543 -2.399499 0\n-3.247468 4.031781 0\n-2.099531 2.193034 0\n12.3693 12.11829 0\
n9.969936 15.64437 0\n-1.928585 1.842135 0\n-1.278543 -2.399499 0\n-6.600385 -5.06042 0\n-
6.555664 -5.038059 0\n-6.510942 -5.015699 0\n-6.466221 -4.993338 0\n-6.4215 -4.970977 0\n-
6.376779 -4.948617 0\n-6.332057 -4.926256 0\n-6.287336 -4.903895 0\n-6.242614 -4.881535 0\
n-6.197893 -4.859174 0\n-6.153172 -4.836813 0\n-6.10845 -4.814453 0\n"
}
```

```
[
    "pathY01",
    "path4"
]
```


当没有map_name和path_name参数时，返回的数据为所有地图和其对应的路径名称


```
{
    "14": [
        "pathY01",
        "path\u673a\u573a\u5230\u8fbe\u53e31\u6b63",
        "path\u673a\u573a3",
        "path\u673a\u573a\u5230\u8fbe\u53e31\u53cd",
        "path\u673a\u573a2"
    ],
    "\u58f9\u667a\u4e91\u516c\u53f8": [
        "01"
    ],
    "12": [
        "pathY01",
        "path\u673a\u573a\u5230\u8fbe\u53e31\u6b63",
        "path\u673a\u573a3",
        "path\u673a\u573a\u5230\u8fbe\u53e31\u53cd",
        "path\u673a\u573a2"
    ],
    "13": [
        "pathY01",
        "path\u673a\u573a\u5230\u8fbe\u53e31\u6b63",
        "path\u673a\u573a3",
        "path\u673a\u573a\u5230\u8fbe\u53e31\u53cd",
        "path\u673a\u573a2"
    ],
    "map5": [
        "pathY01",
        "path4"
    ]
}
```


上传导航路径点

URL: /navigation/path

请求方式: POST, PUT

请求参数:

参数	类型	说明
map_name	string	地图名称
path_name	string	路径名称
path_data	string	路径文件内容

说明:

路径文件内容格式为每个路径点的坐标 x y z，z一般为0，然后换行。可以参照上面API的返回值。如果对应路径已经存在则覆盖之前的路径。

返回参数:

和请求参数一致

删除导航路径点

URL: /navigation/path

请求方式: DELETE

请求参数:

参数	类型	说明
map_name	string	地图名称
path_name	string	路径名称

说明:

注意删除路径的时候会自动删除此路径对应的导航点

返回值:

```
{
  "status": "ok"
}
```

获取导航目标点

URL: /navigation/nav_points

请求方式: GET

请求参数:

参数	类型	说明
map_name	string	目标地图名称
path_name	string	目标路径名称

说明:

返回的path_data是导航点文件内容，其格式为 x y z theta errorX errorY direction name。其中x y z为导航点坐标，一般z为0。theta为导航点角度。errorX errorY为导航点所允许的误差。direction为循环方向，name为点的名称，可以为空。和客户端中的设置一致。

当没有传入map_name和path_name参数时，系统会自动选取上次导航的地图和路径，并返回上次导航的导航点数据。

当没有传入map_name和path_name参数时，系统会自动选取上次导航的地图和路径，并返回上次导航的导航点数据。

返回参数


```
{
    "map_name": "map5",
    "path_name": "path4",
    "path_data": "0.575 -0.356 0 -1.117307 0.1 0.1 0\n6.98246 17.95874 0 -2.563412 0.1 0.1
 0\n9.969936 15.64437 0 2.045675 0.1 0.1 0\n-2.099531 2.193034 0 -1.078226 0.1 0.1 0\n-2.0
99531 2.193034 0 -1.078226 0.1 0.1 0\n-2.099531 2.193034 0 -1.078226 0.1 0.1 0\n-2.099531
2.193034 0 -1.078226 0.1 0.1 0\n-3.247468 4.031781 0 -0.9649882 0.1 0.1 0\n-2.099531 2.193
034 0 -1.078226 0.1 0.1 0\n6.98246 17.95874 0 -2.563412 0.1 0.1 0\n6.98246 17.95874 0 -2.5
63412 0.1 0.1 0\n"
}
```


上传导航目标点

URL: /navigation/nav_points

请求方式: POST, PUT

参数说明

参数	类型	说明
map_name	string	目标地图的名称
path_name	string	目标路径的名称
path_data	string	导航目标点文件内容

说明：当map_name或path_name没有传入时，将采用当前正常使用的地图和路径最为修改对象。

返回参数:

和请求参数一致

删除导航目标点

URL: /navigation/nav_points

请求方式: DELETE

参数说明

参数	类型	说明
map_name	string	目标地图名称
path_name	string	目标路径名称

说明:

删除导航点的时候并不会自动删除路径文件

```
{
  "status": "ok"
}
```

获取地图连接点

URL: /navigation/point_connections

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
points	list	当前所有地图中的所有导航点
connections	list	导航点之间的连接关系

示例数据

```
{
    "points": [
        {
            "x": 1.624922, // 导航点 x 坐标
            "y": -0.372011, // 导航点 y 坐标
            "theta": 0.0, // 导航点角度
            "name": "", // 导航点名称
            "map": "office", // 导航点所在地图名称
            "path": "path2", // 导航点所在地图路径
            "index": 0, // 导航点序号
            "type": "floor", // 导航点所属地图类型
            "levels": [ // 导航点所属楼层，可以为多层。和地图楼层一致
                1
            ]
        },
        {
            "x": 0.8710775,
            "y": -0.09214968,
            "theta": -0.2962013,
            "name": "",
            "map": "office",
            "path": "path2",
            "index": 1,
            "type": "floor",
            "levels": [
                1
            ]
        },
        {
            "x": -0.967,
            "y": -2.864,
            "theta": 1.570796,
            "name": "",
            "map": "out",
```

```
            "path": "path1",
            "index": 0,
            "type": "floor",
            "levels": [
                1
            ]
        },
        {
            "x": 5.470039,
            "y": -0.08699174,
            "theta": 0.0,
            "name": "",
            "map": "out",
            "path": "path1",
            "index": 1,
            "type": "floor",
            "levels": [
                1
            ]
        },
    ],
    "connections": [
        {
            "_id": {
                "$oid": "60ff74d3d78f798bc92b91d5"
            },
            // points为此连接所连接的目标点，表示可以通过一定方式在这两个点间切换
            "points": [
                {
                    "map": "office",
                    "path": "path2",
                    "index": 1,
                    "type": "floor",
                    "levels": [
                        1
                    ]
                },
                {
                    "map": "out",
                    "path": "path1",
                    "index": 0,
                    "type": "floor",
                    "levels": [
                        1
                    ]
                }
            ],
            "type": "direct",// 连接的方式，direct为直接连接，如不同地图中的同一个位置
            // elevator为电梯连接，用于跨楼层
            "is_available": true, // 此连接是否可用，当为false时，此连接将被排除于路径规划
            "id": "a05ee41f-86d3-4588-be99-4feb776848d6"
        }
    ]
}
```

添加导航点连接信息

URL: /navigation/point_connections

URL: /navigation/point_connections

请求方式: POST

请求参数: |参数|类型|说明| |--|--|--| |points|list|目标点信息| |type|string|连接类型，direct直接连接，表示此点为不同地图中的同一个位置。elevator电梯连接|

请求例子

```
{
    "points": [
        {
             "map": "office", // 目标点所在的地图
             "path": "path2", // 目标点所在的路径
             "index": 1 // 目标点的序号
         },
        {
             "map": "out",
             "path": "path1",
             "index": 0
         }
    ],
    "type": "direct" // 连接方式
}
```

返回参数:

和请求参数一致

返回示例

```
{
  "points": [
    {
      "map": "office",
      "path": "path2",
      "index": 1
    },
    {
      "map": "out",
      "path": "path1",
      "index": 0
    }
  ],
  "type": "direct",
  "is_available": true,
  "id": "a05ee41f-86d3-4588-be99-4feb776848d6",
  "_id": {
    "$oid": "60ff74d3d78f798bc92b91d5"
  }
}
```

删除导航点之间的连接

URL: /navigation/point_connections

请求参数

名称	类型	说明
id	string	连接的id，可以通过获取连接信息查到

返回参数:

名称	类型	说明
status	string	完成状态

切换当前地图

注意：此切换地图API只是切换当前使用的地图文件。并不是在导航过程中切换地图。导航中切换地图可以使用/navigation/reload API

URL: /navigation/switch_map

请求方式: GET

请求参数:

参数	类型	说明
map_name	string	切换的地图名称

返回参数

```
{
  "status": "ok"
}
```

执行局部运动任务

说明: 此API会根据对应参数自动创建一个包含LocalMoveAction的任务，然后自动执行

URL: /navigation/start_local_move

请求方式: GET

请求参数:

参数	类型	说明
distance	float	机器人局部运动距离，当distance为正时向前运动，当distance为负时向后运动，单位为米
angle	float	机器人局部运动转向角度，机器人先转动对应角度再直行，单位为弧度
method	int	精准对接辅助手段，0为无，1为使用雷达

返回参数:

返回此任务信息

执行导航和局部运动

说明: 机器人先导航至指定目标点然后执行局部运动，如导航至1号点然后倒退。机器人会自动创建包含一个导航动作和一个局部运动动作的任务。

URL: /navigation/start_nav_and_local_move

请求方式: GET

请求参数:

参数	类型	说明
index	int	目标点index
distance	float	机器人局部运动距离，当distance为正时向前运动，当distance为负时向后运动，单位为米
angle	float	机器人局部运动转向角度，机器人先转动对应角度再直行，单位为弧度
method	int	精准对接辅助手段，0为无，1为使用雷达

返回参数:

返回此任务信息

执行导航任务，局部任务，io任务

说明: 机器人先导航至指定目标点然后执行局部运动动作，然后再设置io电平。如导航到1号点然后倒退，再设置1号端口高电平。机器人会自动创建包含一个导航动作，一个局部运动动作和一个控制io动作的任务。

URL: /navigation/start_nav_local_move_io

请求方式: GET

请求参数:

参数	类型	说明
index	int	目标点index
distance	float	机器人局部运动距离，当distance为正时向前运动，当distance为负时向后运动，单位为米
angle	float	机器人局部运动转向角度，机器人先转动对应角度再直行，单位为弧度
method	int	精准对接辅助手段，0为无，1为使用雷达
level	string	0 为低电平， 1为高电平
port	string	可以为1,2,3分别对应三个IO端口

port	string	可以为1,2,3分别对应三个IO端口

返回参数:

返回此任务信息

任务相关API

机器人的一个动作被定义为Action。比如移动到[1,1]点。又如播放一段声音。一系列的Action组合在一起构成一个任务即Task。通过对Action和Task进行操作，我们可以轻松的控制机器人实现一系列动作。

Task 和 Action 有以下的一些状态

WAITTING 等待执行 WORKING 正在执行 PAUSED 暂停中 CANCELLED 被取消 ERROR 错误 COMPLETE 完成

获取任务信息

URL: /task

请求方式: GET

请求参数:

参数	类型	说明
id	string	目标任务id，可选参数。若无此参数则返回所有的task

返回参数:

task 数据信息

创建任务

URL: /task

请求方式: POST

请求参数:

参数	类型	说明
name	string	创建任务的名称，可以是任意字符串
sub_tasks	list	包含子任务信息的列表，子任务可以是task也可以是action
loop_flag	bool	可选参数，是否自动循环任务

例子:

```
{
    "name": "task",
```

```
        "type": "nav_action",
        "x": 0,
        "y": 0,
        "theta": 0
      },
      {
        "type": "nav_action",
        "x": 1,
        "y": 1,
        "theta": 1
      },
      {
        "type": "nav_action",
        "x": 2,
        "y": 2,
        "theta": 2
      },
    ],
    "loop_flag": false,
}
```

返回参数:

成功创建的task数据

示例返回数据

```
{
  "name": "test task",
  "loop_flag": false,
  "id": "47a97e4e-9327-4214-a780-fd5a2ae39ee3",
  "state": "WAITTING",
  "sub_tasks": [],
  "progress": 0,
  "current_task": null
}
```

修改任务

URL: /task

请求方式: PUT

请求参数:

参数	类型	说明
id	string	目标任务id
name	string	可选参数，新的任务名称
loop_flag	bool	可选参数，是否循环任务
sub_tasks	list	包含子任务信息的列表

返回参数:

修改后的任务数据

示例返回数据


```
{
  "name": "test task",
  "loop_flag": false,
  "id": "47a97e4e-9327-4214-a780-fd5a2ae39ee3",
  "state": "WAITTING",
  "sub_tasks": [],
  "progress": 0,
  "current_task": null
}
```


删除任务

URL: /task

请求方式: DELETE

请求参数:

参数	类型	说明
id	string	目标删除任务的id

返回参数:

参数	类型	说明
status	string	目标任务是否删除成功

启动任务

URL: /task/start

请求方式: GET

请求参数:

参数	类型	说明
id	string	目标任务的id

返回参数:

启动后的任务信息

暂停任务

URL: /task/pause

请求方式: GET

请求方式: GET

请求参数:

参数	类型	说明
id	string	目标任务的id

返回参数:

暂停后的任务信息

继续任务

URL: /task/resume

请求方式: GET

请求参数:

参数	类型	说明
id	string	目标任务的id

返回参数:

继续后的任务信息

取消任务

URL: /task/stop

请求方式: GET

请求参数:

参数	类型	说明
id	string	目标任务的id，可选参数。当没有id时则取消所有任务

返回参数:

取消后的任务信息

循环执行任务

URL: /task/loop

请求方式: GET

请求参数:

参数	类型	说明

loop_flag	bool	是否循环任务

获取动作Action信息

URL: /action

请求方式: GET

请求参数:

参数	类型	说明
id	string	目标action的id。可选参数，当没有此参数时返回数据库中所有保存的action

返回参数:

目标action数据信息

创建动作Action信息

URL: /action

请求方式: POST

说明:

目前Action有以下几个类别。

callback_action 回调动作

当执行此动作时机器人将会向指定的地址(url),以指定的方式(method)，发送指定的数据(data)

属性	类型	说明
url	string	回调动作地址
method	string	回调请求方法
data	object	需要回调发送到的数据

sleep_action 等待动作

当执行此动作时机器人将等待对应的时间

属性	类型	说明
wait_time	float	等待的时间，单位秒

upload_map_action 上传地图至指定调度服务器动作

执行此任务时机器人将根据调用参数将地图数据上传至指定的调度服务器

属性	类型	说明
server_id	string	调度服务器id

download_map_action 从调度服务器下载地图动作

执行此动作时机器人将根据参数从指定的调度服务器下载地图数据

属性	类型	说明
server_id	string	调度服务器id
map_id	string	下载的地图id

nav_action 导航动作

执行此动作时机器人将移动到参数指定位置

属性	类型	说明
x	float	目标位置x坐标，单位为米
y	float	目标位置y坐标，单位为米
theta	float	目标位置机器人朝向角度，单位为弧度

charge_action 充电动作

在充电桩附近执行此动作机器人将自动对接充电桩并进行充电

属性	类型	说明
x	float	充电桩位置x坐标，单位为米
y	float	充电桩位置y坐标，单位为米
theta	float	充电正面方向，单位为弧度

local_move_action 局部运动动作

局部运动动作用于机器人精准对接过程。比如控制机器人倒车进入车库等等。

属性	类型	说明
distance	float	机器人局部运动距离，当distance为正时向前运动，当distance为负时向后运动，单位为米
angle	float	机器人局部运动转向角度，机器人先转动对应角度再直行，单位为弧度
method	int	精准对接辅助手段，0为无，1为使用雷达

wait_req_action 等待请求动作

wait_req_action 等待请求动作

等待http请求动作。此动作会一直等待直到超时或者 `/action/update_wait_req` 被调用

属性	类型	说明
timeout	int	超时时间，可选参数

switch_map_action 切换地图动作

属性	类型	说明
map	string	切换的目标地图名称
path	string	切换的目标路径名称
start_index	int	切换起始位置序号，用于地图切换后的位置初始化

change_setting_action 更改系统设置

属性	类型	说明
settings	object	修改的系统参数
init_settings	object	可选参数,当有此参数时，在包含change_setting_action的task执行完毕或取消后自动恢复成init_setting

roslaunch_action 启动ros程序

属性	类型	说明
launch_cmd	string	ros启动指令，如roslaunch xiaoqiang_track xiaoqiang_track.launch

io_action 控制io动作

属性	类型	说明
level	string	0 为低电平， 1为高电平
port	string	可以为1,2,3分别对应三个IO端口

elevator_req_action 电梯请求动作

说明：机器人和电梯交互的Action。可以通过此动作控制电梯上下和开关门

属性	类型	说明
level	int	目标楼层，可选参数，当有此参数时表示机器人要通过电梯去特定楼层。执行action时要保证机器人在电梯内
direction	string	控制电梯外机按钮，可选参数。当为up时自动按下电梯外机上按钮，为down时自动按下电梯下按钮。执行action时要保证机器人在电梯外

none_stop_nav_action 不停止导航动作

说明: 到达目标点附近后不停止的导航动作。用于流畅切换目标点，连续执行none_stop_nav_action动作可以实现流畅的连续到达目标点中间不机器人不停止。

属性	类型	说明
x	float	目标点x坐标
y	float	目标点y坐标
radius	float	目标半径，当机器人到达目标点radius范围内，此任务返回complete状态

创建action

请求参数:

参数	类型	说明
type	string	action类型，必须是以上几种aciton的一个
task_id	string	创建的action所属的task,可选参数。当有此参数时将新建action加入对应task，没有此参数时自动创建一个新的临时task(task不会被保存至数据库)并将action添加进task

其他参数和对应的Action类型相关

示例请求参数:

```
{
    "type": "sleep_action",
    "wait_time": 2
}
```

返回参数:

包含新创建的action的task

示例返回参数:

```
{
  "name": "auto task",
  "loop_flag": false,
  "id": "c50d355f-9155-4a17-ade6-a61b40820b43",
  "state": "WAITTING",
  "sub_tasks": [
    {
      "wait_time": 2,
      "state": "WAITTING",
      "type": "sleep_action",
      "id": "3bbbed8f-c0dc-4a94-8d6f-05006c943818"
    }
  ],
  "progress": 0,
```

```
  "current_task": null
}
```

修改动作Action信息

URL: /action

请求方式: PUT

请求参数:

参数	类型	说明
id	string	目标action的id

其他参数和Action类型相关

返回参数:

经过修改的Action数据

删除动作Action信息

URL: /action

请求方式: DELETE

请求参数:

参数	类型	说明
id	string	目标action的id

触发等待动作

说明: 对于 `wait_req_action`，机器人会一直等待http请求触发，直到触发后才继续执行下面的任务。

URL: /action/update_wait_req

请求方式: GET

请求参数: 无

返回参数:

参数	类型	说明
status	string	成功为OK

Websocket相关API

Websocket相关API

对于需要高频率获取的数据我们提供了websocket api。比如我们可能需要很高频率的获取机器人的位置，速度等信息。

websocket默认端口3547

URL格式: ws://192.168.0.132:3547/topicName?token=28b1c500400611ebb805493c9303c705

其中192.168.0.132为机器人IP,28b1c500400611ebb805493c9303c705为机器人token。在开启token验证时此参数是必须的，反之则不需要。

topicName为数据对应的ros话题名称。理论上可以订阅机器人内部所有的ros话题。返回的数据为ros话题数据转换成的json数据

`注意在开启websocket连接后要定期发送ping消息，否则连接可能会自动断开。比如可以每秒发送一个ping消息。`

对于6.2.1及以后版本websock也可以用于向机器人发送ros话题。具体数据结构可以先打印订阅的话题参考一下。发送的数据结构和接收的数据结构一样。

下面是常用的接口

获取GalileoStatus

GalileoStatus数据定义可以参照串口api中GalileoStatus的说明。

topicName: /galileo/status

返回数据为json

获取温湿度及可燃气体数据

TopicName: /bw_env_sensors/EnvSensorData

返回数据

```
float temperature; // 温度，单位摄氏度
float rh; // 相对湿度 %RH
float smoke; // 烟雾 ppm
float pm1_0; // pm1.0 ug/m^3
float pm2_5; // pm2.5 ug/m^3
float pm10; // pm10 ug/m^3
float lel; // 可燃气体 ppm
float noise; // 噪声 db
```

对于跨地图导航的说明

在日常使用中，我们经常会遇到需要跨地图导航的情况。比如在不同楼层间进行导航，在不同的仓库区域中进行导航。Galileo API针对这种场景也进行了专门优化。具体的使用方法如下。

我们以跨楼层导航为例，首先我们要创建不同楼层的地图。然后在每个楼层中分别绘制导航点和导航路径。保证机器人可以在各个楼层正常使用。假设一层的地图为map1，二层的地图为map2。一层二层之间以电梯相连接。

为了使机器人能够通过电梯跨楼层，我们需要在地图中添加电梯信息。在一层和二层的电梯门口分别添加添加一个导航点。比如，一层中的电梯导航点为1号点，二层中的电梯导航点为0号点。然后我们需要给这两个点添加连接信息，告诉机器人我们可以通过一定方式从map1中的1号点到达map2中的0号点。

下面是一个具体的连接数据例子

```
{
    "points": [
        {
            "x": 1.624922, // 导航点 x 坐标
            "y": -0.372011, // 导航点 y 坐标
            "theta": 0.0, // 导航点角度
            "name": "", // 导航点名称
            "map": "map1", // 导航点所在地图名称
            "path": "path1", // 导航点所在地图路径
            "index": 0 // 导航点序号
        },
        {
            "x": 0.8710775,
            "y": -0.09214968,
            "theta": -0.2962013,
            "name": "",
            "map": "map1",
            "path": "path1",
            "index": 1
        },
        {
            "x": -0.967,
            "y": -2.864,
            "theta": 1.570796,
            "name": "",
            "map": "map2",
            "path": "path1",
            "index": 0
        },
        {
            "x": 5.470039,
            "y": -0.08699174,
            "theta": 0.0,
            "name": "",
            "map": "map2",
            "path": "path1",
            "index": 1
        },
    ],
    "connections": [
        {
            "_id": {
                "$oid": "60ff74d3d78f798bc92b91d5"
            },
            // points为此连接所连接的目标点，表示可以通过一定方式在这两个点间切换
            "points": [
```

```
                {
                    "map": "map1",
                    "path": "path1",
                    "index": 1
                },
                {
                    "map": "map2",
                    "path": "path1",
                    "index": 0
                }
            ],
            "type": "direct",// 连接的方式，direct为直接连接，如不同地图中的同一个位置
            // elevator为电梯连接，用于跨楼层
            "is_available": true, // 此连接是否可用，当为false时，此连接将被排除于路径规划
            "id": "a05ee41f-86d3-4588-be99-4feb776848d6"
        }
    ]
}
```

机器人在知道这些信息之后，我们通过api向机器人发布目标点，机器人就会自动根据连接信息切换地图并移动至目标点。

比如机器人现在在map1的0号点，我们需要机器人移动至map2中的1号点。机器人就会自动先移动到map1中的1号点。然后通过电梯移动至二层区域，移动至map2中的0号点，并切换至map2地图。然后移动至map2中的1号点。

机器人跨地图自动规划程序会考虑连接的可用性，总的移动距离，切换地图的次数，经过的目标点数等等细信息，最终为用户提供一个最优的路线。 地图的自动切换也并不仅限于两个，实际上只要地图之间有合适的连接信息，程序会自动的搜索最优的切换方式。比如map1和map2之间有连接，map2和map3之间有连接。那么机器人就会在运动过程中自动的由map1切换至map2然后切换至map3。

实际调用API时需要两个接口。 `/navigation/point_connections` 用于添加点之间的连接信息。 `/navigation/start_nav_task` 用于发布导航任务，注意在调用时添加map和path参数。详细说明请查看对应API文档。目前支持的连接方式只有direct连接，电梯支持尚在开发之中。

此程序会调用获取机器人当前电量，机器人当前位置，控制机器人向前移动0.5m的API。注意把代码里面的机器人IP换成自己的机器IP

伽利略视觉导航系统ROS通信协议

除了使用windows客户端或者串口来控制、使用伽利略视觉导航系统外，还可以通过ros的topic机制来实现。

在伽利略视觉导航系统串口通信协议里面，导航串口可以发送和接收串口数据，这是通过galileo_serial_server包实现的，而这个包的实现原理就是将导航串口接收的数据转换成ros话题后以 `/galileo/cmds` 话题发布给伽利略系统，同时通过订阅伽利略系统状态话题 `/galileo/status` 然后封装后下发给导航串口。

因此在ros系统里面，我们可以绕开串口直接发布 `/galileo/cmds` 话题就可以控制伽 利略系统，直接订阅 `/galileo/status` 就可以获取伽利略系统状态。

1.命令发布话题 /galileo/cmds

伽利略视觉导航系统会实时订阅这个话题，给这个话题发送数据就可以对应控制伽利略系统的状态。

这个话题所属类型为 `galileo_msg::GalileoNativeCmds` ，具体定义在 `/home/xiaoqiang/Documents/ros/src/galileo_msg/msg/GalileoNativeCmds.msg` 文件。

话题内容有三个成员：header、length、data。这个话题是为了封装串口数据设计的，在《伽利略视觉导航系统串口通信协议》里面，串口指令是由“包头+数据长度+数据内容”构成的。

因此根据串口通信协议，把包头用标准的std_msgs/Header替换成header，数据长度赋 值给length，数据内容赋值给data，就可以构造出有效的/galileo/cmds话题。

例如，下面cpp代码将构造一个开启导航系统的话题数据，数据内容请参考串口通信协议。

```cpp
#include "galileo_msg/GalileoNativeCmds.h"
galileo_msg::GalileoNativeCmds currentCmds;
currentCmds.header.stamp = ros::Time::now();
currentCmds.header.frame_id = "galileo_msg";
currentCmds.length = 0x02;
currentCmds.data.resize(0x02);
currentCmds.data[0] = 0x6d;
currentCmds.data[1] = 0x00;
```

2.伽利略视觉导航系统状态话题 /galileo/status

伽利略视觉导航系统会持续自动发布 `/galileo/status` 话题，订阅这个话题查询系统状态。

这个话题所属类型为 `galileo_msg::GalileoStatus` ，具体定义在 `/home/xiaoqiang/Documents/ros/src/galileo_msg/msg/GalileoStatus.msg` 文件。

```
std_msgs/Header header
int32 navStatus
int32 visualStatus
int32 chargeStatus
int32 loopStatus
float32 power
int32 targetNumID
int32 targetStatus
float32 targetDistance
int32 angleGoalStatus
float32 controlSpeedX
float32 controlSpeedTheta
float32 currentSpeedX
float32 currentSpeedTheta
```

本话题里面数据成员的意义请参考《伽利略视觉导航系统串口通信协议》中第一部分“导航串口下发的数据包”中的内容定义。

使用 `rostopic echo /galileo/status` 可以直接打印输出话题内容。

ROS中控制导航系统的例子

开启导航系统

首先查阅串口协议文档可知，开启导航对应的指令为 `m 0` ,这是一个两个字节的指令。则可以在终端输入以下指令开启导航

```
rostopic pub /galileo/cmds galileo_msg/GalileoNativeCmds "header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
length: 2
data: [0x6d, 0]" -1
```

其中 length为指令长度，data为指令数据。0x6d对应字符'm'。 有一个比较简单的输入方法先输入 `rostopic pub /galileo/cmds` 之后双击tab键自动补全

```
rostopic pub /galileo/cmds galileo_msg/GalileoNativeCmds "header:
  seq: 0
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
length: 2
data: ''"
```

会出现上面的内容。对data进行修改就可以发布对应的指令了。

伽利略视觉导航系统串口通信协议

导航系统主机带一个usb转串口模块(可以选ttl电平或者rs232电平，默认ttl电平)，通过这个串口用户可以使用导航系统的功能和获取导航系统的状态，下文将这个串口命名为导航串口。

导航串口波特率为115200，8个数据位，1个停止位，无奇偶校验。

1. 导航串口下发的数据包

发布频率固定为30hz，即导航系统每秒自动通过导航串口下发30个数据包。

数据包格式：包头+长度+数据内容+结束符0x00

包头：占3个字节，0xcd 0xeb 0xd7

长度：占1个字节，长度不包括包头和长度本身字符，这个长度还包括字符串结束符0x00，当前值固定为21*4+1=85=0x55。

内容：由12个4字节小端模式二进制表示的数字串联在一起构成。

包头	长度	nav_status	visual_status	map_status	…	数据n	…	busy_
0xcd0xeb0xd7	0x55	4个字节	4个字节	4个字节	…	4个字节	…	4个

完整数据包内容构成一个c语言结构体，结构体具体构成如下所示：

```
typedef struct{
```

```
  int nav_status;// 导航服务状态，0表示没开启closed，1表示开启opened。
  int visual_status;// 视觉系统状态，-1标系视觉系统处于关闭状态，0表示没初始化uninit，1表示正在追踪tracking,2表示丢失lost,1和2都表示视觉系统已经初始化完成。
  int map_status; // 建图服务状态，0表示未开始建图，1表示正在建图
  int gc_status; // 内存回收标志，0表示未进行内存回收，1表示正在进行内存回收
  int gba_status; // 闭环优化标志，0表示未进行闭环优化，1表示正在进行闭环优化
  int charge_status; // 充电状态，0 free 未充电状态，1 charging 充电中，2 charged 已充满，但仍在小电流充电，3 finding
                     // 寻找充电桩，4 docking 停靠充电桩，5 error 错误
  int loop_status; // 是否处于自动巡检状态，1为处于，0为不处于。
  float power;// 电源电压【946】v。
  int target_numID;// 当前目标点编号,默认值为-1表示无效值，当正在执行无ID的任务是值为-2，比如通过Http API 创建的导航任务。
  int target_status;// 当前目标点状态，0表示已经到达或者取消free，1表示正在前往目标点过程中working,2表示当前目标点的移动任务被暂停paused,3表示目标点出现错误error,默认值为-1表示无效值。
  float target_distance;// 机器人距离当前目标点的距离，单位为米，-1表示无效值，该值的绝对值小于0.01时表示已经到达。
  int angle_goal_status;// 目标角度达到情况，0表示未完成，1表示完成，2表示error,默认值为-1表示无效值。

  float control_speed_x;// 导航系统计算给出的前进速度控制分量,单位为m/s。
  float control_speed_theta;// 导航系统计算给出的角速度控制分量,单位为rad/s。
  float current_speed_x;// 当前机器人实际前进速度分量,单位为m/s。
  float current_speed_theta;// 当前机器人实际角速度分量,单位为rad/s。
  unsigned int time_stamp;// 时间戳,单位为1/30毫秒，用于统计丢包率。对于ROS API时间戳在状态的header里面
  float current_pose_x; // 当前机器人在map坐标系下的X坐标,此坐标可以直接用于设置动态插入点坐标
  float current_pose_y; // 当前机器人在map坐标系下的Y坐标
  float current_angle; // 当前机器人在map坐标系下的z轴转角(yaw)
  int busy_status; //当busy为true时系统将仍然后接收新指令，但是不会立即处理。当系统退出busy状态后再处理消息
}Galileo_Status;
```

导航串口可以接收的指令

最大支持100hz上传频率，每条命令由包头+数据长度+数据内容构成。指令可以 串联一起发送但是不推荐这样使用，因为每条指令是否有效还要取决于导航系统状态。

开启、关闭、重载导航系统

0xcd	0xeb	0xd7	0x02	0x6d	0xXX
包头	包头	包头	数据长度	“m”	值为4表示关闭，0表示为开启

通过本命令可以开启、关闭导航系统，导航系统开启后需要对视觉系统进行初始化，视觉系统初始化过程机器人可能会原地旋转（当前视角无法初始化时需要调整视角），视觉初始化后才能使用导航系统的功能。

因为视觉系统初始化时间较长，所以不建议频繁地进行关闭、开启导航系统操作。导航系统的状态可以由nav_status获得、视觉系统的状态可以由visual_status获得。

现在导航系统支持地图的动态切换，不再需要重启导航系统便可以载入不同的导航地图。同时地图也支持自动切换，导航系统可根据当前环境自动切换至对应的地图。在地图切换后需要重新载入对应地图的导航路径，此时即可用到重载功能。

0xcd	0xeb	0xd7	0x02	0x6d	0xXX
包头	包头	包头	数据长度	“m”	9

重载时不会对导航地图进行处理，只重新载入当前的路径。所以调用重载之前需要设置好地图和路径。

发布下一个目标点

0xcd	0xeb	0xd7	0x02	0x67	0xXX
包头	包头	包头	数据长度	“g”	值为0到255，表示目标点编号

发布前需要检查视觉系统有没有完成初始化，如果没有完成初始化，系统会忽略这条指令。

视觉系统如果已经初始化了，主机接收这条指令后，先根据当前目标点状态判断是否要先执行取消当前移动任务的操作。当前目标点状态为free或者error时不执行任何动作直接继续下一步，当前目标点状态为working或paused时会自动取消当前目标点然后继续下一步。

接着系统会将当前目标点编号更新成设定值，再验证目标点编号是否有效。

如果目标点有效，立即控制机器人往目标点移动，同时目标点的状态更新成working

如果目标点无效，反馈的目标点状态更新成error，同时机器人保持不动

目标到达后，目标点状态会自动更新成free，机器人退出移动任务同时保持不动

暂停、继续、取消当前目标点

0xcd	0xeb	0xd7	0x02	0x69	0xXX
包头	包头	包头	数据长度	“i”	值为0表示暂停，值为1表示继续，值为2表示取消

发布暂停指令后，系统会检查目标点状态，状态为working时本命令才有效，为其它状态则忽略。指令判断有效后机器人会停止运动、保存移动任务同时目标点状态会更新成paused。

发布继续指令后，系统会检查目标点状态，状态为pause时本命令才有效，为其它状态则忽略。指令判断有效后机器人继续往目标点运动同时反馈的目标点状态变成working，

发布取消指令后，系统会检查目标点状态，状态不时free时本命令才有效，为其它状态则忽略。指令判断有效后机器人停止运动，退出移动任务,目标状态更新成free。

移动到动态目标点

添加目标点

0xcd	0xeb	0xd7	0x0a	0x67	0x69	0xXX	0xXX	0xXX	0xXX	0xX
包头	包头	包头	数据长度	“g”	"i"	x，y，theta的float32坐标，共4 * 3Byte				

机器人移动到x,y坐标，角度为theta

手动遥控速度指令

0xcd	0xeb	0xd7	0x02	0x66	0xXX
包头	包头	包头	命令长度	前进指令	速度大小，为最大速度百分比，数值范围为0到100

0xcd	0xeb	0xd7	0x02	0x62	0xXX
包头	包头	包头	命令长度	后退指令	速度大小，数值范围为0到100

0xcd	0xeb	0xd7	0x02	0x63	0xXX
包头	包头	包头	命令长度	左转指令	速度大小，数值范围为0到100

0xcd	0xeb	0xd7	0x02	0x64	0xXX
包头	包头	包头	命令长度	右转指令	速度大小，数值范围为0到100

0xcd	0xeb	0xd7	0x02	0x73	0xXX
包头	包头	包头	长度	停止指令	制动量大小，数值范围为0到100

可以控制机器人的移动，这些指令独立于视觉导航系统，导航系统工不工作都会执行这些指令，因此可以用来实现机器人的遥控功能。

这些指令生效后不会影响目标点状态，即遥控的速度指令和导航系统的速度指令是并行的会相互覆盖。如果导航运动过程中机器人发生错误需要切换到遥控，用户应该先发布目标点取消指令，否则可能会出现机器人还是不受自己控制的情况。

主机关机指令

0xcd	0xeb	0xd7	0x02	0xaa	0x44
包头	包头	包头	数据长度	数据1	数据2

主机已经配置成通电后自动开机，这样可以避免按键的不方便。使用本条指令可以实现串口关机。

自动巡检状态控制

自动巡检功能即为机器人按照当前所有的目标点循环移动。在到达目标点时停靠特定时间，此时间可以通过api进行设置。

开启自动巡检功能

注意只有在开启导航服务之后才可以启动自动巡检功能。当此功能成功开启之后galileo_status中的loopFlag会设置成True。

0xcd	0xeb	0xd7	0x02	0x6d	0x05
包头	包头	包头	数据长度	“m”	5

关闭自动巡检功能

0xcd	0xeb	0xd7	0x02	0x6d	0x04
包头	包头	包头	数据长度	“m”	6

设置自动巡检目标点停靠时间

0xcd	0xeb	0xd7	0x03	0x6d	0x05	0xXX
包头	包头	包头	数据长度	“m”	5	目标点停靠时间，单位为秒

调度系统导航控制

调度导航状态是配合拉格朗日调度系统使用的导航状态，在此状态下机器人的导航任务由调度系统发布。

开启调度导航

0xcd	0xeb	0xd7	0x02	0x6d	0x07
包头	包头	包头	数据长度	“m”	7

关闭调度系统导航

0xcd	0xeb	0xd7	0x02	0x6d	0x08
包头	包头	包头	数据长度	“m”	8

重载调度系统程序，不关闭导航地图，用于动态切换地图。在调用前需要设置好当前的地图和路径。
|0xcd|0xeb|0xd7|0x02|0x6d|0x0a| |:--:|:--:|:--:|:--:|:--:|:--:|:--:| |包头|包头|包头|数据长度|“m”|10|

开始建立地图，结束建立地图,保存和更新地图

0xcd	0xeb	0xd7	0x02	0x56	0xXX

包头	包头	包头	数据长度	“V”	0表示为开启，值为1表示关闭

自动充电相关指令

0xcd	0xeb	0xd7	0x02	0x6a	0xXX
包头	包头	包头	数据长度	“j”	0表示为开始充电，值为1表示停止充电，2表示保存充电桩位置

迎宾模式

0xcd	0xeb	0xd7	0x02	0x48	0xXX
包头	包头	包头	数据长度	"H"	0表示为开启迎宾模式，1未关闭迎宾模式

版本历史

版本	时间	更新说明
V1.0	2018-3-15	初稿，覆盖基本功能
V1.2	2018-3-22	修改开启导航系统指令
V1.3	2018-7-11	添加动态目标点功能
V1.4	2018-11-8	增加自动巡检功能
V1.5	2018-11-23	增加地图创建相关状态，增加创建地图控制指令
V1.6	2019-1-25	添加自动充电指令
V1.7	2019-10-11	添加调度系统导航控制指令
V1.8	2020-04-02	增加动态导航载入控制
V1.9	2020-08-07	增加无ID目标点任务状态
V2.0	2021-09-21	更新接口

伽利略导航系统SDK使用说明

`目前SDK基于ros通信，在网络不稳定情况下稳定性较差。同时SDK功能相对较少，推荐使用HTTP API进行开发。SDK有两个版本，机器人在5.0.0以下版本用v1，以上采用v2`

为了降低用户开发和使用伽利略导航系统的难度，我们提供了伽利略导航系统SDK。目前对以下语言提供了支持

C++ pipeline failed

Python pipeline passed

C# pipeline passed

Java & Android pipeline passed

所有语言的SDK都是基于 `C++ SDK` 进行开发。下面对 `C++ SDK` 进行说明。其他语言可以进行参考，SDK的使用方法都是一致的。对于不同语言SDK的详细信息可以参考每个SDK的README文件。

快速开始

获取当前局域网内所有机器人

```
#include "GalileoSDK.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化SDK,注意只能实例化一个SDK对象
    while (true)
    {
        auto servers = sdk.GetServersOnline(); // 获取当前局域网内所有机器人
        if(servers.size() == 0){
            std::cout << "No server found" << std::endl;
        }
        for (auto it = servers.begin(); it < servers.end(); it++)
        {
```

```
            std::cout << it->getID() << std::endl; // 输出机器人ID
        }
        Sleep(1000);
    }
}
```

连接机器人

下面这种连接方式是阻塞的，sdk会等待10000ms连接局域网内任意一个机器人。如果局域网内有多个机器人，程序会返回MULTI_SERVER_FOUND错误。当局域网内有且仅有唯一的一个机器人时，sdk自动连接。 如果10000ms没有成功连接机器人，Connect方法会返回超时TIMEOUT错误。

```
#include "GalileoSDK.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化SDK
    auto res = sdk.Connect("", true, 10000, NULL, NULL); // 连接局域网内任意一个机器人
    if(res == GalileoSDK::GALILEO_RETURN_CODE::OK)
        std::cout << "Connect Success" << std::endl;
    std::cout << "Connect Failed" << std::endl;
}
```

连接机器人，异步方式

下面的连接方式是异步的，可以通过设置回调函数处理机器人连接状态。这个方法是非阻塞的，Connect方法会立即返回。当连接成功或超时时OnConnect回调会被调用。具体的状态信息可以通过回调函数的第一个参数获取

```
#include "GalileoSDK.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化SDK
    sdk.Connect("", true, 10000,
        // OnConnect回调函数
        [](GalileoSDK::GALILEO_RETURN_CODE res, std::string id) -> void {
            std::cout << "OnConnect Callback: result " << res << std::endl;
            std::cout << "OnConnect Callback: connected to " << id << std::endl;
        },
        // OnDisconnect回调函数
        [](GalileoSDK::GALILEO_RETURN_CODE res, std::string id) -> void {
            std::cout << "OnDisconnect Callback: result " << res << std::endl;
            std::cout << "OnDisconnect Callback: server " << id << std::endl;
    });
    while(true){
        Sleep(10000);
    }
}
```

通过物联网跨局域网连接机器人

只有v1版本支持此功能

```
#include "GalileoSDK.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化sdk
    // 通过物联网连接机器人，可以跨局域网连接。
    // 参数为目标机器人id，超时时间timeout，目标机器人密码，回调函数
    if (sdk.Connect("71329A5B0F2D68364BB7B44F3F125531E4C7F5BC3BCE2694DFE39B505FF9C730A614F
F2790C1", 10000, "xiaoqiang", NULL, NULL) != GalileoSDK::GALILEO_RETURN_CODE::OK)
    {
        std::cout << "Connect to server failed" << std::endl;
    }
    while (true)
    {
        if (sdk.PublishTest() == GalileoSDK::GALILEO_RETURN_CODE::OK) {
            std::cout << "pub succeed" << std::endl;
        }
        else {
            std::cout << "pub failed" << std::endl;
        }
        Sleep(1000);
    }
}
```

获取机器人当前电压

在连接机器人后，机器人的状态信息可以通过 `GetCurrentStatus` 方法获取。具体有哪些状态信息可以参照 `galileo_msg/GalileoStatus.h` 文件定义。

```
#include "GalileoSDK.h"
#include "galileo_msg/GalileoStatus.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化SDK
    sdk.Connect("", true, 10000, NULL, NULL); // 连接局域网内任意一个机器人
    galileo_msg::GalileoStatus status; // 创建伽利略状态对象
    while (true)
    {
        if (sdk.GetCurrentStatus(&status) == GalileoSDK::OK) // 获取当前系统状态
        {
            std::cout << status.power << std::endl; // 输出系统电压
        }
        else
        {
            std::cout << "Get status failed" << std::endl;
        }
        Sleep(1000);
    }
}
```

控制机器人前进

```
#include "GalileoSDK.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化SDK
```

```
    sdk.Connect("", true, 10000, NULL, NULL); // 连接局域网内任意一个机器人
    sdk.SetSpeed(0.1, 0); // 设置机器人速度，以0.1m/s速度前进
}
```

控制机器人移动到特定坐标

对于V1版本

假设机器人已经建立好地图且绘制好相关路径，我们可以通过MoveTo方法控制机器人移动到特定坐标位置。下面代码中的posx和posy即为目标点坐标。你可以替换成自己的实际目标点坐标。

对于V2版本可以直接设置目标点角度

```
#include "GalileoSDK.h"
#include "galileo_msg/GalileoStatus.h"

int main(){
    GalileoSDK::GalileoSDK sdk; // 实例化SDK
    sdk.Connect("", true, 10000, NULL, NULL); // 连接局域网内任意一个机器人
    sdk.StartNav();
    galileo_msg::GalileoStatus status;
    sdk.GetCurrentStatus(&status);
    // 等待正常追踪,正常追踪时visualStatus和navStatus都为1
    while (status.visualStatus != 1 || status.navStatus != 1)
    {
        std::cout << "Wait for navigation initialization" << std::endl;
        sdk.GetCurrentStatus(&status);
        Sleep(1000);
    }
    // 以下为V1例子
    uint8_t goalNum;
    sdk.MoveTo( posx, posy, &goalNum); // 移动到特定目标点v1
    // 以下为V2例子
    sdk.MoveTo( posx, posy, theta); // 移动到特定目标点v2
    // 等待移动开始
    sdk.GetCurrentStatus(&status);
    while (status.targetStatus != 1)
    {
        std::cout << "Wait for goal start" << std::endl;
        sdk.GetCurrentStatus(&status);
        Sleep(1000);
    }
    std::cout << "Goal started" << std::endl;
    // 等待移动完成
    while (status.targetStatus != 0)
    {
        std::cout << "Wait for goal complete" << std::endl;
        sdk.GetCurrentStatus(&status);
        Sleep(1000);
    }
}
```

SDK 说明

```
GALILEO_RETURN_CODE
Connect(std::string targetID, bool auto_connect, int timeout,
        void (*OnConnect)(GALILEO_RETURN_CODE, std::string),
        void (*OnDisconnect)(GALILEO_RETURN_CODE, std::string));
```

连接机器人

输入

`std::string targetID` 目标机器人ID

`bool auto_connect` 是否开启自动连接功能。当targetID为空字符串，且auto_connect为true时。机器人会自动连接局域网内的机器人。当局域网内有多台机器人时，此方法会返回MULTI_SERVER_FOUND错误。

`int timeout` 超时时间，单位毫秒。当在此时间内没有成功连接机器人则此方法返回超时错误。

`void (*OnConnect)(GALILEO_RETURN_CODE, std::string)` 连接回调函数。当此值为NULL时，Connect会以同步阻塞方式执行。Connect会等待连接成功直至超时。当此值不是NULL时，Connect会以非阻塞方式执行。当机器人成功连接或连接超时时，SDK会调用此回调函数。

`void (*OnDisconnect)(GALILEO_RETURN_CODE, std::string)` 连接断开回调函数。当此值非NULL,机器人连接断开时SDK会调用此回调函数。

返回

GALILEO_RETURN_CODE

```
std::vector<ServerInfo> GetServersOnline()
```

获取当前局域网内所有机器人信息

输入 无

输出

局域网内所有机器人列表

```
ServerInfo *GetCurrentServer();
```

获取当前连接的机器人信息

输入 无

输出 当前连接的机器人信息

在SDK未连接机器人时，此方法返回NULL。在SDK成功连接机器人后此方法返回机器人信息

```
GALILEO_RETURN_CODE SendCMD(uint8_t[] cmd, int length);
```

发送伽利略指令，具体信息参考伽利略串口说明文档

输入

`uint8_t[] cmd` 伽利略导航指令

`int length` 指令长度

输出

`GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StartNav();
```

开启导航服务。此方法只会发送开始导航指令，并不会等待导航开启完成。只能在机器人未处于导航状态且未处于建图状态下才可以执行此方法。

输入 空 输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StopNav();
```

关闭导航服务。此方法只会发送关闭导航指令，并不会等待导航关闭完成。只能在机器人处于导航状态下才可以执行此方法。

输入 空 输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SetGoal(int goalIndex);
```

设置导航目标点。此方法只会发送设置导航点指令，并不会等待目标点完成。只能在机器人处于导航状态下执行此方法。

输入

`int goalIndex` 目标点标号

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE PauseGoal();
```

暂停当前导航任务

输入 无

输出 GALILEO_RETURN_CODE

```
GALILEO_RETURN_CODE ResumeGoal();
```

继续当前导航任务

输出 无

输出 GALILEO_RETURN_CODE

```
GALILEO_RETURN_CODE CancelGoal();
```

取消当前导航任务

输入 无

输出 GALILEO_RETURN_CODE

```
GALILEO_RETURN_CODE InsertGoal(float x, float y);
```

只有v1版本支持此功能

插入导航点

输入

`float x` 插入点x坐标，此坐标为机器人map坐标系下坐标 `float y` 插入点y坐标，此坐标为机器人map坐标系下坐标

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE ResetGoal();
```

只有v1版本支持此功能

重置目标点。所有通过InsertGoal插入的目标点都会重置。只保留原始目标点。

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SetSpeed(float vLinear, float vAngle);
```

设置机器人速度

输入

`float vLinear` 直线运动速度，单位m/s `float vAngle` 转动速度， 单位 rad/s

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE Shutdown();
```

关闭机器人主机

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SetAngle(uint8_t sign, uint8_t angle);
```

只有v1版本支持此功能

设置机器人角度

输入

`uint8_t sign` 转向0表示正向旋转，1表示反向旋转 `uint8_t angle` 转动角度

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StartLoop();
```

开始循环导航。开启后机器人会按照导航点的顺序依次执行。

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StopLoop();
```

停止导航循环

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SetLoopWaitTime(uint8_t time);
```

设置循环等待时间。机器人在循环过程中到达目标点的等待时间

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StartMapping();
```

开始建图服务。机器人只能在不处于导航状态或建图状态下才能执行此方法。此方法只是发送开启建图指令，并不会等待建图服务开始运行。

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StopMapping();
```

停止建图服务。机器人只能在建图状态下才能执行此方法。此方法只是发送停止建图指令，并不会等待建图服务停止。

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SaveMap();
```

只有v1版本支持此功能

保存当前地图。当前地图会被保存在机器人的 `slamdb` 文件夹，但是在Windows客户端并不会显示。想要在Windows客户端显示需要把当前地图复制到 `saved-slamdb` 文件夹

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE UpdateMap();
```

只有v1版本支持此功能

更新地图

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StartChargeLocal();
```

开始局部充电功能。局部充电采用惯导定位，利用充电桩传感器对准充电桩。局部充电只能在机器人在充电桩附近使用。此方法只是发送开启局部充电指令，并不会等待充电动作完成

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StopChargeLocal();
```

停止局部充电功能

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SaveChargeBasePosition();
```

保存充电桩位置

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StartCharge(float x, float y); // V1
GALILEO_RETURN_CODE StartCharge(); // V2
```

开始充电。此方法采用融合导航定位算法，可以在保证机器人处于导航状态下，在地图任意位置执行此指令。此指令首先会控制机器人移动到充电桩附近,然后再启动局部充电指令。此指令是一个阻塞指令，程序会等待机器人运动到充电桩后再退出。V1版本需要自己传入充电桩坐标位置，V2版本不需要传入充电桩位置，程序可以自动获取。

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE StopCharge();
```

取消充电指令

输入 无

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE MoveTo(float x, float y, uint8_t *goalNum); // V1
GALILEO_RETURN_CODE MoveTo(float x, float y, float theta); // V2
```

控制机器人移动到指定坐标

输入

`float x` 目标点x坐标 `float y` 目标点y坐标 `goalNum` 新插入的目标点id，此参数相当于一个返回值。在成功插入点后此值会被设置为新插入点的ID `theta` 目标点角度

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE GetGoalNum(uint8_t *goalNum);
```

`只有v1版本支持此功能`

获取当前目标点总数

输入

`uint8_t *goalNum` 相当于返回值，成功调用后此值会被设置为当前目标点总数

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE GetCurrentStatus(galileo_msg::GalileoStatus *);
```

获取当前机器人伽利略状态

输入

`galileo_msg::GalileoStatus * status` 相当于返回值，成功调用后此值会被设置为当前伽利略系统状态

输出 `GALILEO_RETURN_CODE`

```
void SetCurrentStatusCallback(void (*callback)(
        GALILEO_RETURN_CODE, galileo_msg::GalileoStatus));
```

设置状态更新回调函数，当伽利略系统状态更新时会执行设置的回调函数并将最新的系统状态传递给此回调函数。

```
void SetGoalReachedCallback(
        void (*callback)(int goalID, galileo_msg::GalileoStatus));
```

设置到达目标点函数。当机器人到达任意目标点时会执行此处设置的回调函数。并且将目标点ID和当前系统状态传递给此回调函数

```
GALILEO_RETURN_CODE WaitForGoal(int goalID);
```

等待到达目标点。此方法会阻塞直至到达目标点或目标点被取消。

输入

`int goalID` 目标点ID

输出 `GALILEO_RETURN_CODE`

```
bool CheckServerOnline(std::string targetid);
```

检查目标机器人是否在线

输入

`std::string targetid` 目标机器人ID

输出

`bool` 目标机器人是否在线

```
GALILEO_RETURN_CODE SendAudio(char audio[]);
```

向机器人发布语音

输入

`char audio[]` 语音uft8字符串

输出 `GALILEO_RETURN_CODE`

```
void Dispose();
```

释放SDK占用的资源

```
GALILEO_RETURN_CODE EnableGreeting(bool flag);
```

控制迎宾模式。在迎宾模式下当有人在机器人前走过时机器人会播放欢迎光临语音。

输入

`bool flag` 是否开启迎宾模式

输出 `GALILEO_RETURN_CODE`

```
GALILEO_RETURN_CODE SendGalileoBridgeRequest(std::string method, std::string url, std::str
ing body, HttpBridgeResponse& response, int timeout)
```

只有v1版本支持此功能

调用Galileo Http API。Galileo Http API说明文档

输入

method http请求方式，如get，post等 url http请求url body http请求body response http请求响应，其数据结构为 std::string uuid，请求的id。 uint16_t status_code，响应的http状态码。std::string body， http响应body。 timeout 发送请求的超时值。

输出 `GALILEO_RETURN_CODE`

例子

```
GalileoSDK::HttpBridgeResponse res;
status = sdk.SendGalileoBridgeRequest("get", "/api/v1/system/status", "", res, 10 * 1000);
if (status != GalileoSDK::GALILEO_RETURN_CODE::OK)
{
    std::cout << "send http get req to server failed" << std::endl;
    std::cout << "status: " << status << std::endl;
    return;
}
```

注意事项

1. 在机器人刚启动时可能SDK会连接失败。机器人需要在启动后等待初始化完成才能成功连接（可能要十秒左右）。连接失败时可以重试连接。