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。

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_serial_server::GalileoNativeCmds` ，具体定义在 `/home/xiaoqiang/Documents/ros/src/galileo_serial_server/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。

```
#include "galileo_serial_server/GalileoNativeCmds.h"
galileo_serial_server::GalileoNativeCmds currentCmds;
currentCmds.header.stamp = ros::Time::now();
currentCmds.header.frame_id = "galileo_serial_server";
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_serial_server::GalileoStatus` ，具体定义在 `/home/xiaoqiang/Documents/ros/src/galileo_serial_server/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_serial_server/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_serial_server/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12*4+1=49=0x31。

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	Power	…	数据n	…	Time_sta
0xcd0xeb0xd7	0x31	4个字节	4个字节	4个字节	…	4个字节	…	4个字节

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tr
```

```
acking,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。
  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;
```

2. 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。

2.a. 开启、关闭导航系统

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。

2.b. 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。

2.b.1 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。

2.b.2 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。

2.b.3 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。

2.c 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。

2.d 动态添加目标点

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和y的float32坐标，共八位				

接收到此命令后，系统会在当前目标点的最后添加一个目标点。可以利用动态添加目标点功能实现移动到某个坐标位置。此坐标为map坐标系下坐标。

重置目标点

0xcd	0xeb	0xd7	0x03	0x67	0x72	0x00
包头	包头	包头	数据长度	“g”	"r"	0

接收到此命令后，系统将清空所有动态添加的目标点，只保留最初的目标点。此功能可以用于动态点的初始化。保证使用正确的目标点id。

2.e.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。

2.f 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。

2.g 角度设定指令

0xcd	0xeb	0xd7	0x03	0x61	0xXX	0xXX
包头	包头	包头	数据长度	“a”	角度值符号，0表示正，1表示负	角度值，范围为0到180

系统收到指令后，会控制机器人原地旋转到设定角度值。

参考角度需要视觉定位系统提供，因此这条指令需要视觉系统已经初始化完成，即视觉系统状态不为uninit，否则指令无效。

发布这条指令前还需要检查目标点状态，如果目标点状态不为free或者error,本指令也无效，这样可以防止和导航系统的运动控制冲突。

本指令结果由angle_goal_status状态反映。

2.g 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

2.h 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，2表示保存,3表示更新地图

2.i 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

3. 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

伽利略导航系统SDK使用说明

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++

Python

C#

Java & Android

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READMMD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);
    }
}
```

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_serial_server/GalileoStatus.h` 文件定义。

```
#include "GalileoSDK.h"
#include "galileo_serial_server/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_serial_server::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

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。

```
#include "GalileoSDK.h"
#include "galileo_serial_server/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_serial_server::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);
    }
    uint8_t goalNum;
    sdk.MoveTo( posx, posy, &goalNum); // 移动到特定目标点
    // 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 || status.targetNumID != goalNum)
    {
        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);
```

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();
```

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);
```

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();
```

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();
```

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);
```

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。

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);
```

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

输出 `GALILEO_RETURN_CODE`

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

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_serial_server::GalileoStatus *);
```

获取当前机器人伽利略状态

输入

`galileo_serial_server::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_serial_server::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_serial_server::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

```
void Dispose();
```

释放SDK占用的资源

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。