gps_base 节点

1.发布：

gps_pub = nh_local->advertise<gps::MyGPS_msg>("GPS_Base", 10); 发布 GPS 信息（自定义类型中的）

gps_fix_pub = nh_local->advertise<sensor_msgs::NavSatFix>("GPS_fix", 1);（用于发布 GPS 的定位信息，包括纬度、经度、海拔高度以及定位质量等信息）

gps_raw_pub = nh_local->advertise<std_msgs::String>("GPSRaw_Base", 10);

2.订阅：

keyboard_sub = nh->subscribe<std_msgs::UInt16>("/keyboard", 10, &TGPS_Base::KeyboardCallback, this);

根据传递的键盘消息，设定模拟的 x、y 和 yaw 的变换

joy_sub = nh->subscribe<sensor_msgs::Joy>("/joy", 10, &TGPS_Base::JoyCallback, this);

用于处理手柄数据并模拟 GPS 数据

simpose_sub = nh->subscribe<geometry_msgs::PoseStamped>("/sim_pose", 10, &TGPS_Base::SimPoseCallback, this);

处理接收到的模拟位姿（SimPose）数据

3.主函数：

调用 simdata 模拟 GPS 信号并发布，通过 UDP 或串口接收 GPS 数据，并将接收到的数据逐行传递给 DataProc 函数进行处理。

4.其他函数：

simdata 函数含义：void TGPS_Base::simdata()

模拟生成 GPS 数据并发布到 ROS 中，包括位置、速度、角度等信息，gps_pub.publish(gps_sim_msg);

DataProc 函数含义：int TGPS_Base::DataProc(const char *buf)

主要负责处理接收到的 GPS 数据并发布相关信息到 ROS 中，调用 Resolve 函数解析接收到的 GPS 数据，返回解析结果（可能包含位置、方向、速度等信息），将原始接收到的 GPS 数据以字符串形式发布到 ROS 的 gps_raw 话题中 gps_raw_pub.publish(strmsg); 发布 gps_fix_pub.publish(Nav); 和 gps_pub.publish(msg);

Resolve 函数含义：int TGPSData::Resolve(const char *buf)

该函数用于解析接收到的 GPS 数据，提取其中的位置、方向、速度等信息，并更新相关状态。返回 Ang_state

gps_pro 节点

1.发布：

path_load_pub = nh->advertise<nav_msgs::Path>("Path_Load", 10);（未发布）

path_save_pub = nh->advertise<nav_msgs::Path>("Path_Save", 10);机器人的路径信息

marker_pub = nh->advertise<visualization_msgs::Marker>("car_marker", 1);可视化工具中显示机器人的位置

2.订阅：

gps_base_sub = nh->subscribe<gps::MyGPS_msg>("/gps_base/GPS_Base", 10, &TGPS_Pro::GPSDataCallback, this);

判断是否为点云匹配模式，如果不是则接受 GPS 传过来的信息，并调用 PubPosition 函数

matching_loc_sub = nh->subscribe<nav_msgs::Odometry>("/laser_localization", 10, &TGPS_Pro::matchingLocCallback, this);

判断是否为点云匹配模式，如果是则根据信息调用 PubPosition 函数

3.主函数：（未调用）

根据逻辑用于存储路径，调用 LoadPath 函数进行存储。

4.其他函数：

PubPosition 函数：void TGPS_Pro::PubPosition(geometry_msgs::Pose pose, int quality)

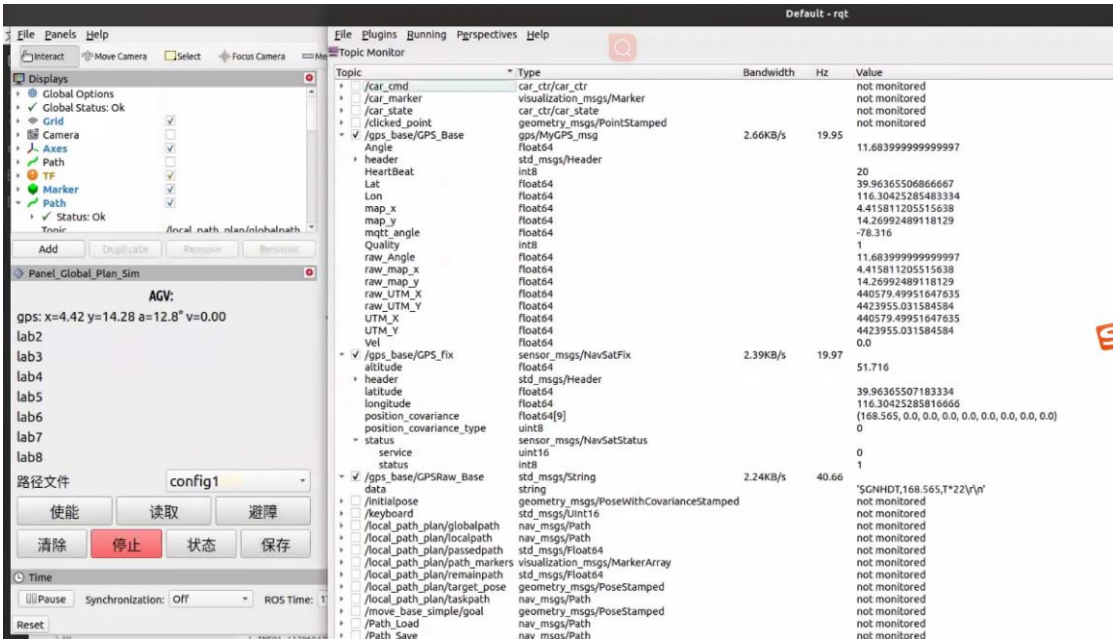
发布位置点 marker_pub.publish(displayCarPosition(pose_stamped, quality))可视化工具中显示机器人的位置，

用于保存发布机器人的路径信息到 ROS 话题 path_save_pub.publish(path_msg)（saveflag 开关直接获取 nh_local->getParam("saveflag", flag);）；

LoadPath 函数：void TGPS_Pro::LoadPath(char *filename)

从文件中加载轨迹信息，并根据零点坐标进行调整，最后存储在

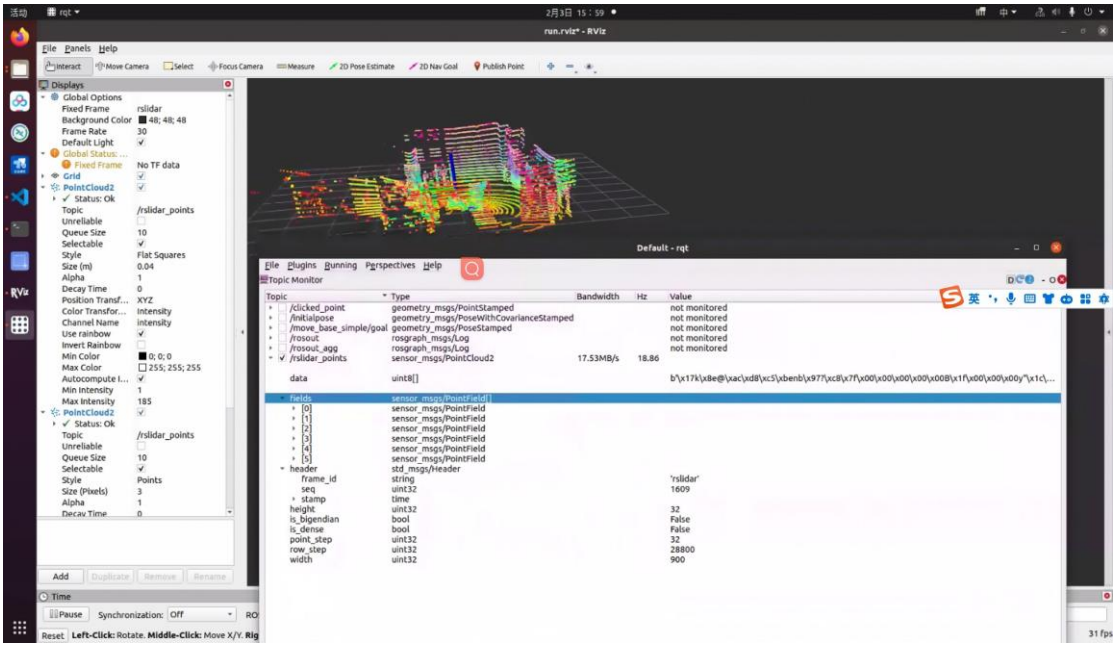
PathLoadBuf 中。在加载时，还会输出加载的轨迹点数量。



rslidar_sdke 节点

重点：

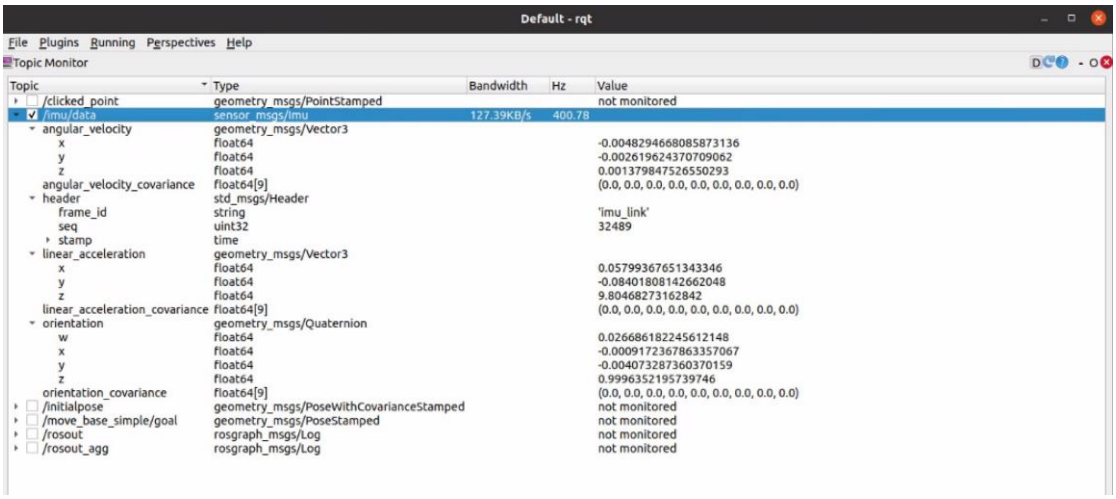
包为厂家给的，只需要在config文件中进行修改雷达相关参数，并在launch文件中启动 rslidar_sdk_node 节点，发布原始点云信息为/rslidar_points（消息类型：sensor_msgs/PointCloud2）



节点：xsens_ros_mti_driver

重点：

包为厂家给的，直接调用/imu/data（消息类型 sensor_mags/Imu）



Topic	Type	Bandwidth	Hz	Value
/clicked_point	geometry_msgs/PointStamped			not monitored
/imu/data	sensor_msgs/Imu	127.39KB/s	400.78	
angular_velocity	geometry_msgs/Vector3			
x	float64			-0.0048294668085873136
y	float64			-0.002619624370709062
z	float64			0.001379847526550293
angular_velocity_covariance	float64[9]			(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
header	std_msgs/Header			
frame_id	string			'imu_link'
seq	uint32			32489
stamp	time			
linear_acceleration	geometry_msgs/Vector3			
x	float64			0.05799367651343346
y	float64			-0.08401808142662048
z	float64			9.80468273162842
linear_acceleration_covariance	float64[9]			(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
orientation	geometry_msgs/Quaternion			
w	float64			0.026686182245612148
x	float64			-0.0009172367863357067
y	float64			-0.004073287360370159
z	float64			0.9996352195739746
orientation_covariance	float64[9]			(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
/initialpose	geometry_msgs/PoseWithCovarianceStamped			not monitored
/move_base_simple/goal	geometry_msgs/PoseStamped			not monitored
/rosout	rosgraph_msgs/Log			not monitored
/rosout_agg	rosgraph_msgs/Log			not monitored