FLSLAM 软件设计方案

更改记录			
版本	**更改日期**	**更改内容**	**修订人**
v0.1	2022.02.23	初始版本	谢吉东

目录

1. 综述

1.1.文档目的

导航技术是机器人领域非常关键的技术，导航的性能优劣直接影响到机器人的应用范围。本文基于对机器人导航技术调研基础上，结合对机器人功能需求的理解，描述了 SLAM 导航系统架构设计。

本设计方案介绍了导航系统整体框架及各子模块的功能实现，包含 SLAM 模块间的通信框架，协议接口，SLAM 算法集成及优化，数据传感器融合等关键技术的实现。同时描述了此软件系统于机器人的部署方式及限制条件。从而实现一套轻量级和快速的导航系统，并运用于实际的机器人产品项目。

本文档读者为本方案软件设计的设计人员、软件模块的编码实现人员，单元测试相关人员等。

1.2. 项目相关文档及参考文献

参考文档：

《移动机器人建图与定位导航性能评估规范》

《导航数据流剖析》

《Gmapping 建图原理》

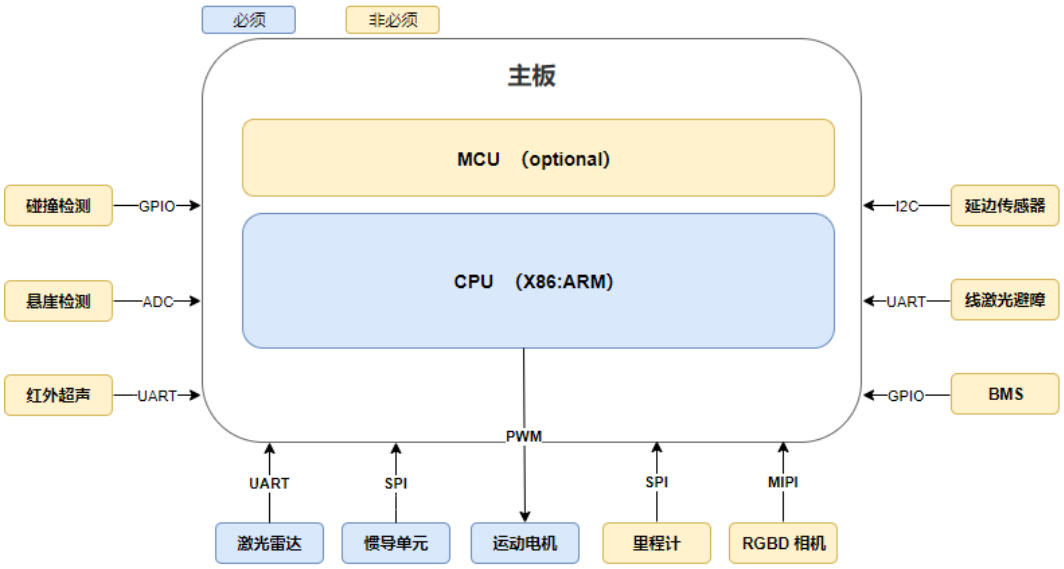
《全局及局部路径规划原理》

术语表：

序号	术语（缩略语）	说明
1	FLSLAM	Fast Light SLAM，一种快速轻量级的导航解决方案
2	IMU	惯性测量单元，测量三轴姿态角以及加速度，由加速度器及陀螺仪组成
3	PID	积分，微分控制器，广泛应用于机器人底盘运动电机控制
4	Laser	激光雷达，以发射激光束探测障碍目标的位置、速度的雷达系统
5	Odom	里程计，运动电机编码器获取的机器人的位姿数据
6	URDF	Unified Robot Description Format，是一种基于XML规范、用于描述机器人结构的格式

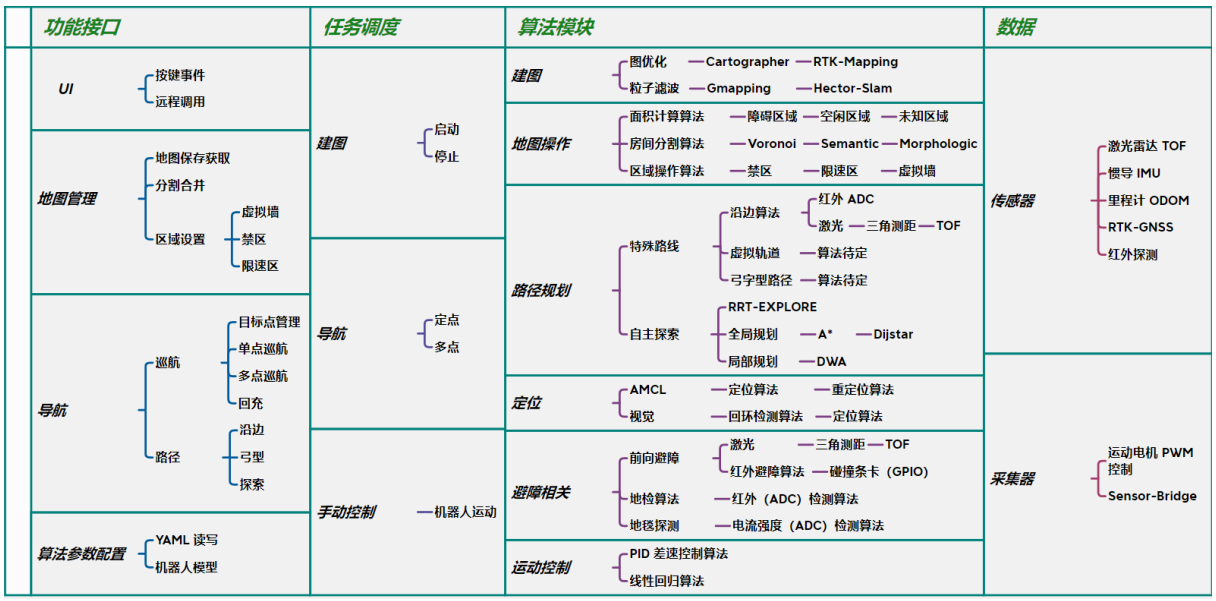
2. 硬件基本配置

导航系统软件运行于机器人系统时，对机器人硬件有一定要求，以下为机器人硬件模块要求，包含必须和非必须模块，非必须模块可以提高导航精度及效率，同时可以提升机器人导航的稳定性：



3. 软件功能概述

下图是整个 FLSLAM 系统的软件功能模块图，功能接口负责交互，通过调度模块来对个算法模块进行组合和调用，避免资源的冲突和混乱，保证算法的有序执行。主要的核心模块包括：地图构建，地图管理，导航控制，定位与重定位，机器人避障，传感数据桥接器。整个 FLSLAM 采用分布式系统，根据各功能模块来划分任务进程，同时节点间采用 DDS 发布订阅的方法解决通信问题。



4. 设计约束与技术要求

4.1. 设计约束

本系统采用 C++ 作为主要的开发语言，为提高执行效率，算法模块会包含少许汇编或者 C 代码。作为分布式系统，节点间通信采用标准 DDS IDL 作为远程调用的接口描述文件，开发人员需要在开发平台上安装 IDL C++ 接口生成器。推荐使用 Ubuntu 18.04 作为软件开发平台，其他平台未知是否有适配兼容性问题。

4.2. 技术要求

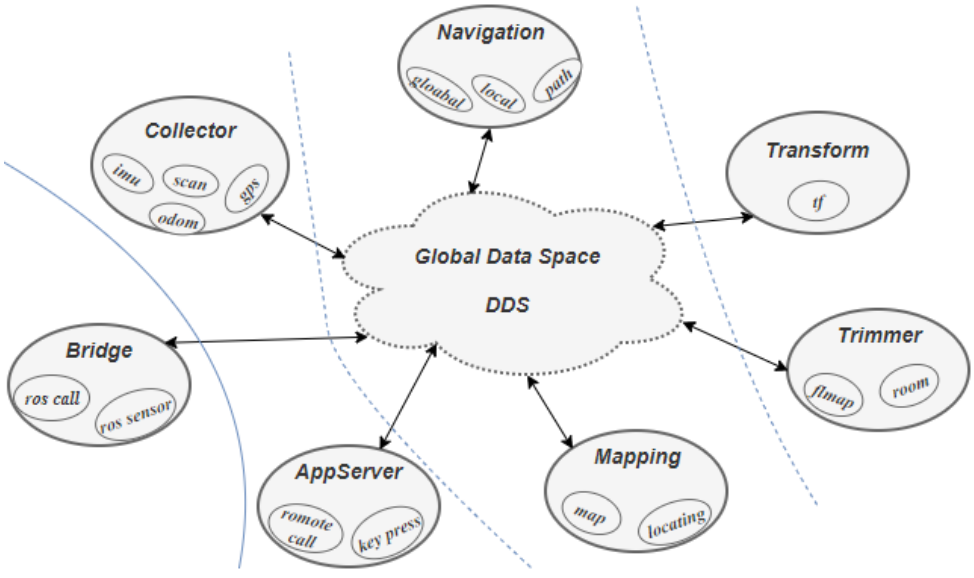
主要有以下技术指标，涵盖到建图，导航，定位，避障等关键算法功能。验证方法根据《移动机器人建图与定位导航性能评估规范》所描述步骤，结合导航应用需求得到以下主要的技术指标要求：

模块	序号	指标	预设条件	意义	目标	备注
建图精度	1	灰度率	turtlebot3 -《规范》	检测地图的灰度率	< 2%	很少未知区域
	2	重影率	turtlebot3 -《规范》	统计重影区域范围	< 0.5%	有少部分重影
	3	边界模糊率	turtlebot3 -《规范》	统计模糊边界区域	< 10%	边界空隙点为模糊区域
	4	分辨率	turtlebot3 -《规范》	算法可建图分辨率	0.05 ~ 0.5	0.1 作为梯度
	5	建图范围	turtlebot3 -《规范》	算法可建图区域	> 160 M2	地图可作为导航为标准
建图效率	6	建图速度	turtlebot3 -《规范》	衡量建图效率	5Min/100M2	100 平米空间建图时间
	7	成功率	turtlebot3 -《规范》	建图成功率	> 94%	
路径规划	10	窄道路线规划	turtlebot3 -《规范》	窄道通过率	> 90%	狭窄通道以 1.5 机器人宽度为标准
	11	目标位姿调整	turtlebot3 -《规范》	导航到目标时姿态	> 90%	
避障	12	静态避障	turtlebot3 -《规范》	静态物避障成功率	> 80%	100 次避障
	13	动态避障	turtlebot3 -《规范》	动态物避障成功率	> 90%	100 次避障
脱困	14	凹形区域脱困	turtlebot3 -《规范》	凹形脱困成功率	> 80%	100 次避障
定位	15	定位及时性	turtlebot3 -《规范》	衡量定位效率	< 5s	取平均值
	16	成功率	turtlebot3 -《规范》	衡量定位成功率	> 90%	

5. 方案设计

5.1. 软件总体架构

FLSLAM 采用分布式架构，由以下七大基础节点组成，后续根据用户需求动态调整节点，节点间通过 DDS 话题方式通信：



系统架构说明：

•Bridge：桥接节点，在缺乏机器人硬件和需要使用 ROS 生态下的仿真工具时，可以在 ROS 环境上运行此节点，通过此桥接节点与导航系统通信。比如发送仿真传感器数据到 FLSLAM，从 FLSLAM 接收地图，路径信息可视化呈现在 ROS 的仿真工具上。

• AppServer：用户接口节点，从云端或者手机接收控制指令，控制导航系统行为。

• Collector：数据采集节点，从传感器采集导航所需要的数据，比如惯导，激光雷达数据，里程计，并通过 DDS 话题发布到系统环境。

• Navigation：导航节点，计算出全局路径，局部路径，并控制运动电机朝目标点运行。

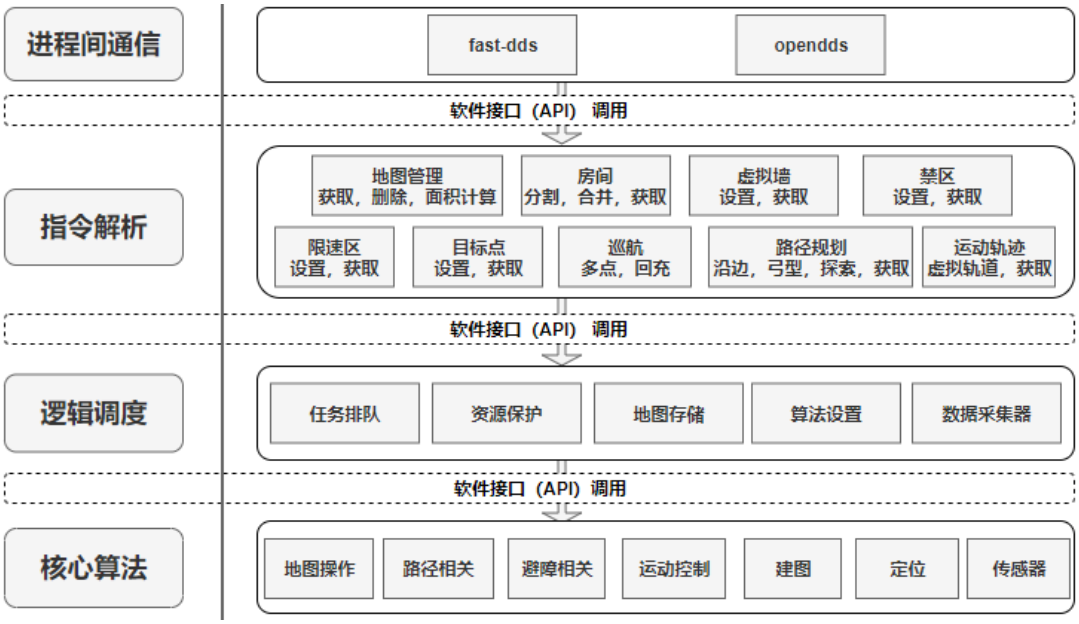
• Mapping：地图构建节点，订阅传感器数据，通过算法运算后，得到地图数据，并发布到系统环境。

• Transform：坐标系转换器，使用树型数据结构存储，根据时间戳缓冲并维护多个参考系之间的坐标变换关系，可以帮助用户在任意时间，将点、向量等数据的坐标，在两个参考系中完成坐标变换。

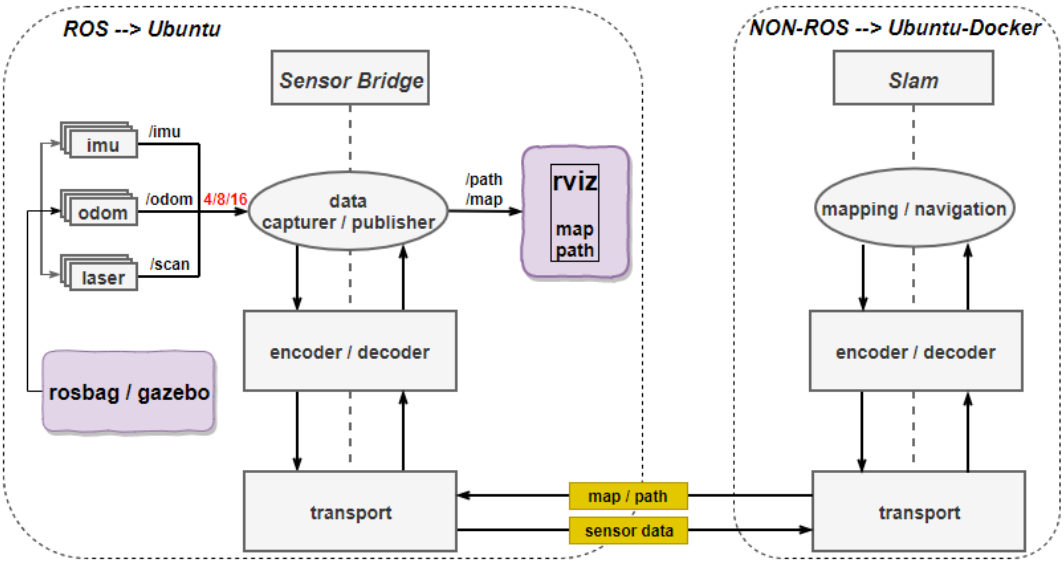
• Trimmer：地图修剪器，对 Mapping 节点生成的地图优化修剪，并完成房间分割，虚拟墙，禁区设置等任务。

软件体系架构：

服务节点按以下软件架构实现节点，从软件层上可以分为通信层，指令解析层，调度层，核心算法层。各层通过接口调用访问，以 C++ 类接口方式解耦。



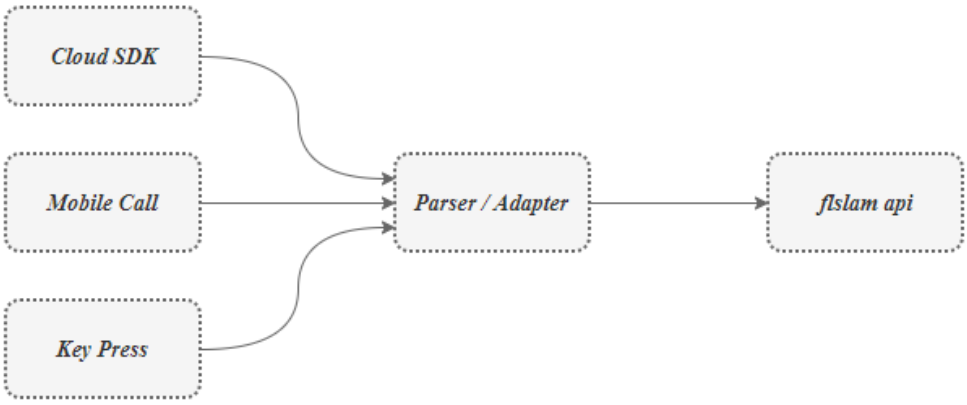
5.1.1. Bridge 桥接器



左侧部分即为桥接器所完成的功能，设计这个节点的目的是为了更多的使用到 ROS 生态软件包，仿真模拟工具等。这个节点运行于调试电脑，目前规划为运行于 ROS@Ubuntu 调试机上，在机器人运行过程中，通过可视化工具查检测导航状态，故障反馈，提高了调试的便利性。主要有三个功能：

- 采集 ROS 空间下的传感器数据，发送到 FLSLAM，进行建图，路径规划
- 订阅导航系统空间的地图，路径数据，并发布到 ROS 空间，实现可视化调试
- 发送私有控制指令到 FLSLAM，改变导航行为，查看机器人运行状态

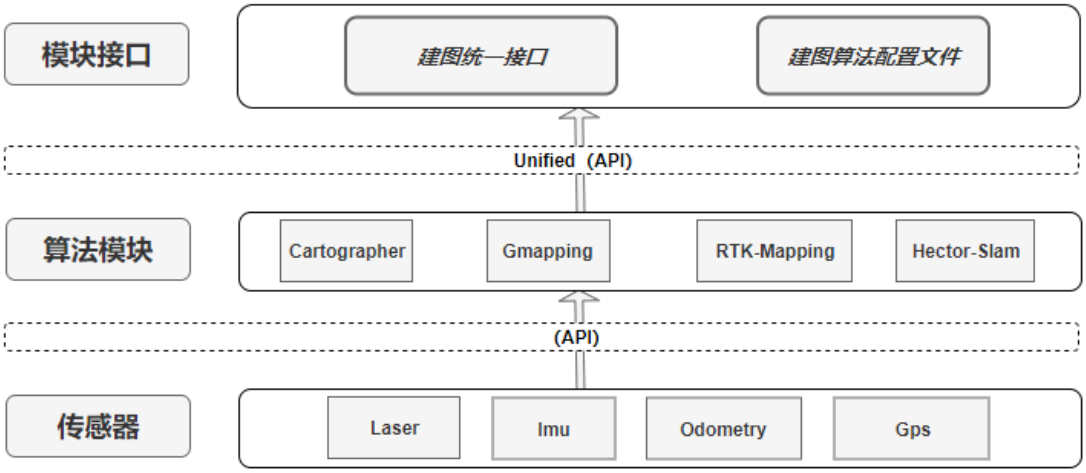
5.1.2. AppServer



用户交互入口，可以接收来自于云端，移动设备端，机器人实体按键的事件。通过解析转换后，封装成统一的控制指令到导航系统。同时这个节点需要有机器人状态机管理机制，统筹控制机器人任务执行时序。

Cloud SDK 对接不同的云服务，实现远程控制机器人，此场景适合扫地机等产品。

5.1.3. Mapping



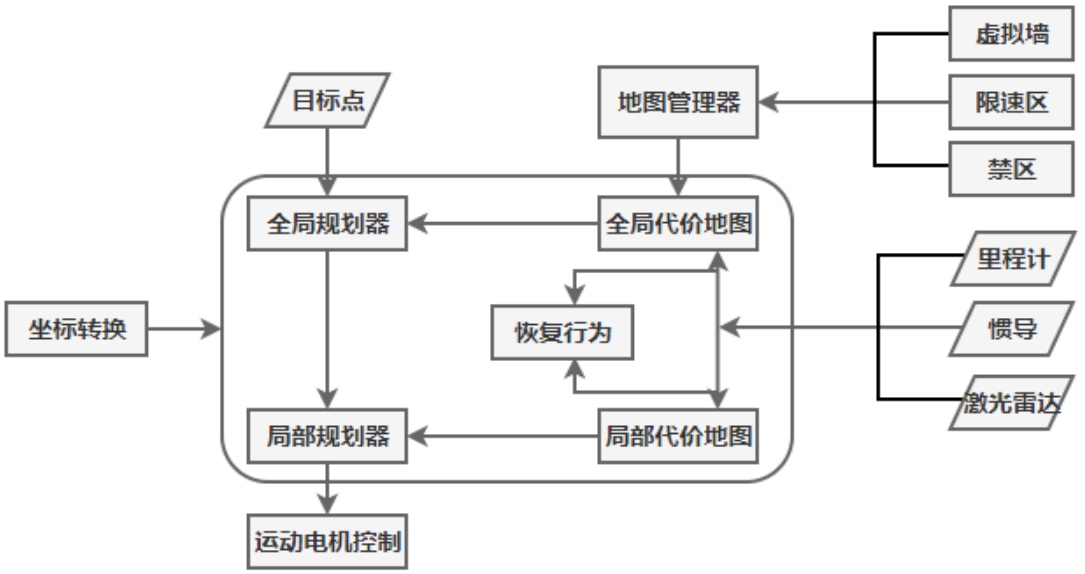
此节点订阅激光雷达，惯导，里程计等传感器数据，构建出地图后通过 DDS 发布到导航系统。

• 算法部分目前采用 cartographer 开源算法，对算法，数据流处理部分优化改造，最终构建出全地图和子图，仅发布全地图到导航系统

• RTK 采用绕圈打点的方式来制作地图外围边缘，对于内部环境，在机器人运行过程中，探测障碍物后，在地图上标记的方式来更新地图

• 以上几种算法采用相同地图数据格式，保证对外接口的统一性

5.1.4. Navigation



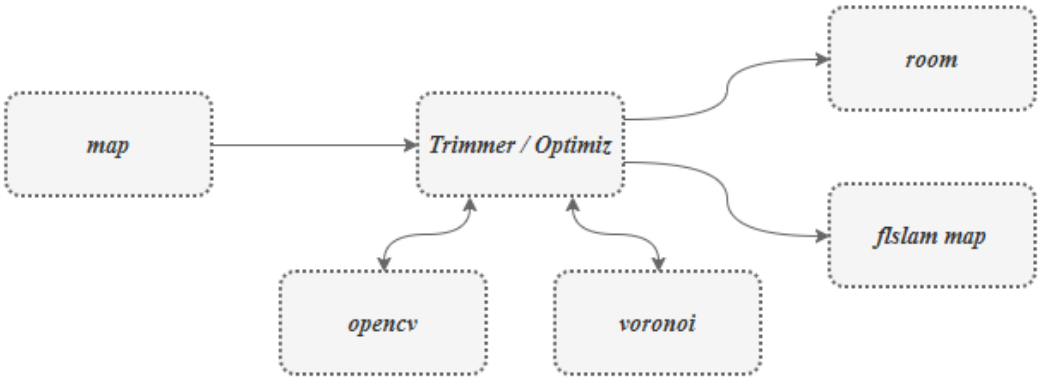
导航节点软件架构与地图节点类似，订阅传感器数据，导航目标点。

• 通过规划算法计算出全局路径，发布路径信息

• 根据路径信息，自动输出底盘控制的运动电机两轮速度，控制机器人到达目标位置

• 手动控制机器人运行时，根据用户指令，控制运动电机到达目标位置

5.1.5. Trimmer



地图修剪器订阅地图节点发布的地图数据，把地图转换成 PGM 图片，进行优化分割。

• 调用 Opencv 的接口用图优化的方式优化，可以封闭地图上的噪点，完成后发布优化后的地图数据

• 对 PGM 图片调用 Voronoi 的接口，把图片分割为多个房间，同时发布房间信息

5.2. 模块交互逻辑

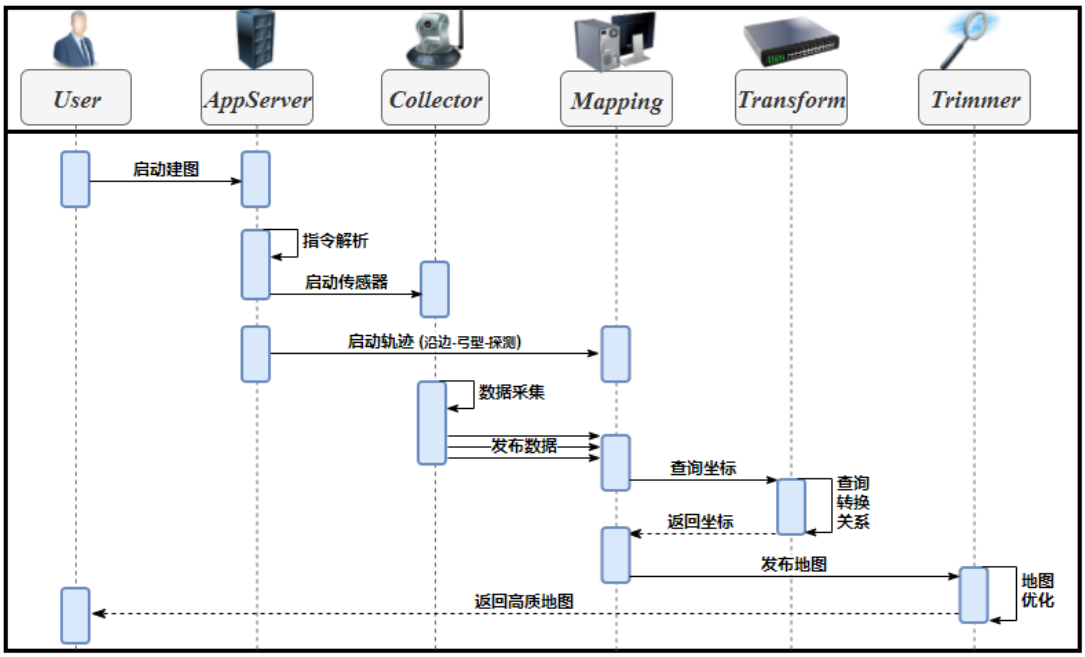
导航系统主要有三大用户操作，涉及到多个模块间交互：

• 地图构建，包含延边建图，自主探测建图，弓字形探测建图

• 地图编辑，包含房间设置，虚拟墙设置，禁区设置，需要更新地图，代价地图和可视化地图

• 导航规划，单点导航，多点导航等

5.2.1. 地图构建时序图



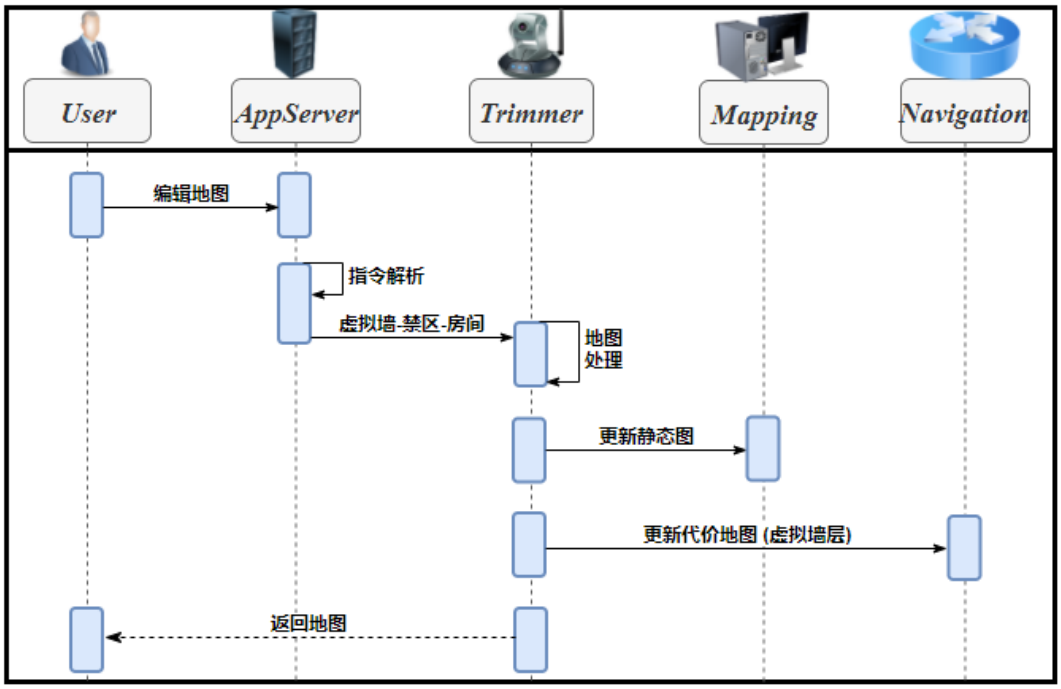
地图构建说明：

• AppServer 接收到建图的指令，信息包含需要传感器类型，建图方式，首次建图或者二次建

图。然后启动 Collector 从硬件驱动获取传感器数据，启动 Mapping 处理传感器数据构建栅格地图。

•Mapping 启动后，订阅 Collector 发布的传感器数据，处理数据时要结合机器人的位置和姿态，不断查询当前位姿及坐标，来验证数据的合法完整性。

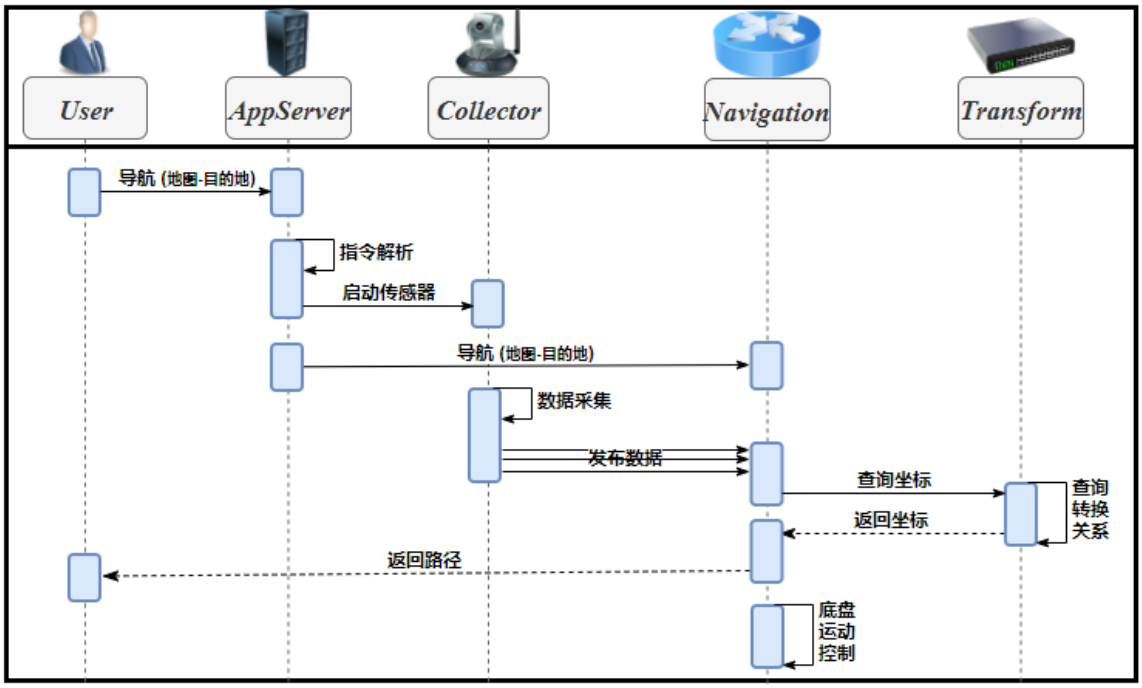
5.2.2. 地图编辑时序图



地图编辑说明：

•Trimmer 对地图编辑后，需要把更新的地图同步到 Mapping，用于二次轨迹建图。同时制作新的图层更新到 Navigation 所使用代价地图，代价地图增加一个或者多个图层。

5.2.3. 导航规划时序图



导航规划说明：

•Navigation 得到原始地图和目的地坐标后，首先计算出全局路径，机器人总体方向是按全局路径行进

• 在行进过程中，导航模块订阅传感器数据，实时探测周边障碍物，对路径微量调整，避免机器碰撞障碍物

5.3. 接口描述

5.3.1. 地图消息结构（nav_msgs/OccupancyGrid）

```
{
    std_msgs/Header header                  "消息包头：序列，时间戳，坐标系"
        uint32 seq
        time stamp
        string frame_id
    nav_msgs/MapMetaData info               "地图属性：长度，宽度，分辨率，位置，姿态"
        time map_load_time
        float32 resolution
        uint32 width
        uint32 height
        geometry_msgs/Pose origin
            geometry_msgs/Point position    "位置：x，y，z"
                float64 x
                float64 y
                float64 z
```

```
        geometry_msgs/Quaternion orientation      “位姿：四元数”
            float64 x
            float64 y
            float64 z
            float64 w
    int8[] data                                   “地图数据：占据，空闲，未知”
}
```

5.3.2. 目的地坐标消息结构（geometry_msgs/PoseStamped）

```
{
    std_msgs/Header header                        “消息包头：序列，时间戳，坐标系”
        uint32 seq
        time stamp
        string frame_id
    geometry_msgs/Pose pose
        geometry_msgs/Point position              “位置：x，y，z”
            float64 x
            float64 y
            float64 z
        geometry_msgs/Quaternion orientation      “位姿：四元数”
            float64 x
            float64 y
            float64 z
            float64 w
}
```

5.3.3. 激光雷达消息结构（sensor_msgs/LaserScan）

```
{
    std_msgs/Header header                        “消息包头：序列，时间戳，坐标系”
        uint32 seq
        time stamp
        string frame_id
        float32 angle_min
        float32 angle_max
        float32 angle_increment
        float32 time_increment
        float32 scan_time
        float32 range_min
        float32 range_max
        float32[] ranges
        float32[] intensities
}
```

5.3.4. 惯导消息结构（sensor_msgs/Imu）

```
{
```

```
    std_msgs/Header header                    "消息包头：序列，时间戳，坐标系"
        uint32 seq
        time stamp
        string frame_id
    geometry_msgs/Quaternion orientation
        float64 x
        float64 y
        float64 z
        float64 w
    float64[9] orientation_covariance
    geometry_msgs/Vector3 angular_velocity
        float64 x
        float64 y
        float64 z
    float64[9] angular_velocity_covariance
    geometry_msgs/Vector3 linear_acceleration
        float64 x
        float64 y
        float64 z
    float64[9] linear_acceleration_covariance
}
```

5.4. 数据存储设计

5.4.1. 机器人模型（XML 格式定义）

```
{
<robot name="flslam_demo">
  <link name="scan_link">
    <visual>
      <origin xyz="0 0 0" />
      <geometry>
        <cylinder length="0.05" radius="0.03" />
      </geometry>
      <material name="gray" />
    </visual>
  </link>
  <link name="base_link" />
  <joint name="scan_link_joint" type="fixed">
    <parent link="base_link" />
    <child link="scan_link" />
    <origin xyz="0.1007 0 0.0558" />
  </joint>
</robot>
```

```
}
```

5.4.2. 导航启动参数（Lua 格式定义）

```
{
options = {
  map_builder = MAP_BUILDER,
  trajectory_builder = TRAJECTORY_BUILDER,
  map_frame = "map",
  tracking_frame = "base_footprint",
  published_frame = "odom",
  odom_frame = "odom",
  lookup_transform_timeout_sec = 0.2,
  submap_publish_period_sec = 0.3,
  pose_publish_period_sec = 5e-3,
  trajectory_publish_period_sec = 30e-3,
  rangefinder_sampling_ratio = 1.,
  odometry_sampling_ratio = 1.,
  fixed_frame_pose_sampling_ratio = 1.,
  imu_sampling_ratio = 1.,
  landmarks_sampling_ratio = 1.,
      }
}
```

5.5. 第三方软件

Ceres，Fast-DDS，TinyXML，Protobuf，Boost

6. 其他软件相关

6.1. 硬件环境

导航运行硬件环境要求如下：

硬件模块	参数
CPU	X86 或者 ARM， 主频双核 1.2g 或以上
RAM	DDR 512M 或以上
Laser	帧率 5hz 或以上
IMU	帧率 500hz 或者以上

6.2. 软件环境

导航运行软件环境要求如下：

操作系统	参数
Debian	Debian 9 “stretch” 或以后版本
Ubuntu	Ubuntu 18.04 或以后版本
ARM-Linux	Linux 4.8 或以后版本

6.3. 编程软件要求

以 C++ 作为主要开发语言，使用 Ubuntu 18.04 作为开发平台