**【墨玉-机器人系统】**

**软件设计说明书**

**【*SDD-201*】**

**【2.0】**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 墨玉（组号：201） | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 17373203 | 陆广炎 | 3数据库设计 |
| 17373261 | 陈宇轩 | 5接口设计 |
| 17373200 | 陈宇航 | 1范围2需求概述7运行环境 |
| 17373217 | 段牧知 | 5接口设计8需求可追踪性说明 排版修改 |
| 17373468 | 赵子敬 | 4体系结构设计6详细设计 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0 | 2020.4.20 | 陈宇航 | 陈宇轩 | 初稿 |
| 1.1 | 2020.4.23 | 陆广炎 | 陈宇轩 | 完善数据库设计 |
| 1.2 | 2020.4.24 | 赵子敬 | 陈宇轩 | 完善详细设计 |
| 1.3 | 2020.4.24 | 段牧知 | 陈宇轩 | 对图表信息进行整合修改，完善接口设计 |
| 1.3 | 2020.4.24 | 陈宇航 | 陈宇轩 | 修改表标，撤销界面和不必要的注释 |
| 1.4 | 2020.5.21 | 段牧知 | 陈宇轩 | 修改用户界面部分 |
| 1.5 | 2020.5.21 | 陆广炎、赵子敬 |  | 修改详细设计中后端模块实现部分以及数据库部分 |
| 2.0 | 2020.6.8 | 陈宇航 |  | 文档整体格式修改以及前端页面部分修改 |

1. 范围
   1. 项目概述

作为一种专用的计算机系统，嵌入式系统是软件和硬件的综合体，还可以涵盖机械等附属装置。它嵌入在装置或设备中实现特定的控制逻辑，如机器人控制单元等。

本项目目标为基于嵌入式系统设计一款可以灵活行进，识别目标物体并可抓取目标物体的机器人，用于酒店服务，主要功能是房间送物和引导带路。该机器人软件部分使用ROS机器人算法平台进行开发设计，硬件部分则直接采用拥有硬件里程计、激光测距雷达、立体视觉相机和语音输入输出阵列等一系列硬件设施且可完美搭载ROS算法的启智ROS机器人。

除了上述的主要功能，本项目对于设计出的机器人还有以下非功能性需求：1、机器人结构稳定，可长期使用而不会轻易损毁。2、机器人硬件软件并行设计，耦合度较低。这样即使现有硬件坏了，也可较为方便地进行软件移植，在短时间内获得可新使用的机器人。3、可扩展性强。即当用户具有新需求时，可通过对现有机器人软件系统进行扩展并重新移植，增加现有机器人的功能。4、在规定时间内(现设置时间为5s)对于正确的用户需求做出响应。5、对于错误的用户需求进行提示并正确处理；若出现错误可在有限时间内恢复；此外还拥有简单的指令格式，易于使用和学习。

* 1. 文档概述

本文档是北京航空航天大学计算机学院2020年春季学期软件工程（嵌入式方向）课程中墨玉小组的软件需求规格说明书文档。本文档适用于基于Ubuntu16.04系统的ROS机器人操作系统开发，适用的硬件平台为启智ROS机器人，此外编写设计的软件是在该硬件平台上实现一个具有目标检测和抓取功能的机器人，用于酒店服务，其主要功能是房间送物和引导带路。

此外，本文档主要用于帮助开发者小组即墨玉小组进一步明确项目的功能性需求和非功能性需求，对设计过程中需要用到的数据进行初步了解和进行开发计划的时间安排以及团队工作的分配，以便于团队成员们按计划有序高效地完成开发任务。

本文档主要由以下八部分组成：范围、需求概述、数据库设计、体系结构设计、接口设计、详细设计、运行与开发环境和需求可追踪性说明。

与本文档相关的，还有如下软件说明文档，其分别为：SDP软件开发计划文档、SRS软件需求规格说明文档、STD软件测试说明文档。本文档初次撰写于2020年4月18日，最后一次撰写于2020年6月8日。

本项目的开发计划用于总体上指导ROS机器人软件项目顺利开发进行并得到通过最终评审的项目产品。本项目开发计划面向项目组的全体成员，项目周期为3个月。

本文档各部分内容将随着小组成员的开发进展而实时更新。

* 1. 术语和缩略词

**表1.3-1 专业术语及其全称对应表**

|  |  |
| --- | --- |
| 业务及技术术语/缩略词 | 全称 |
| ROS | The Robot Operating System/机器人操作系统 |
| URDF | Unified Robot Description Format/统一机器人描述格式 |
| IMU | Inertial Measurement Unit/惯性测量单元 |
| 即时定位与地图构建（SLAM） | Simultaneous Localization and Mapping/即时定位与地图构建 |
| 黑盒测试 |  |
| 白盒测试 |  |
| Rivz可视化界面 |  |

* 1. 引用文档

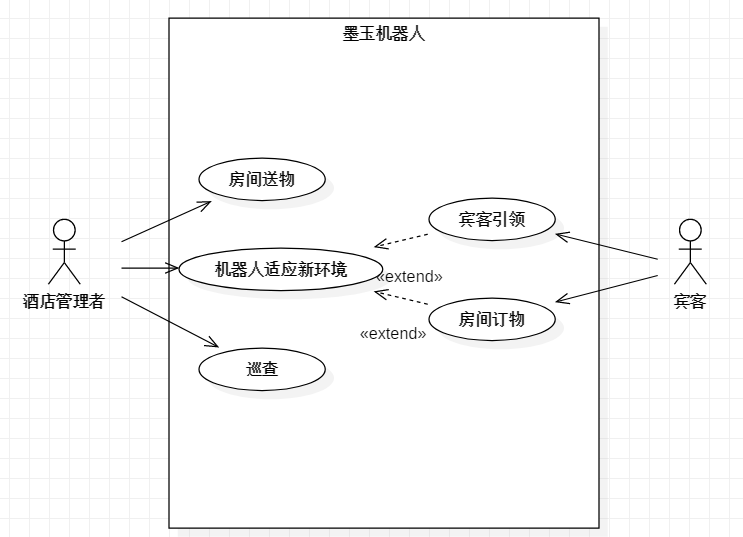
**表1.4-1 引用文档总结表**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 标题 | 版本 | 发行日期 |
| 1 | 启智ROS版开发手册 | V1.1.0 | 2018/11/09 |
| 2 | 启智ROS机器人手册 | V1.1.0 | 2020/2/19 |
| 3 | ROS系统入门 | V1.1.0 | 2020/3/05 |
| 4 | SDP-软件开发计划 | V1.1 | 2020/4/15 |
| 5 | SRS-软件需求规格说明书 | V2.0 | 2020/6/8 |

1. 需求概述
   1. 业务需求

本机器人系统主要实用功能需求是宾客引领、巡查以及房间送物三个方面。在此基础上，为了保证机器人系统长期稳定的运行服务，一方面机器人自身软硬件运行稳定性需要有保障，另一方面机器人系统应当根据酒店环境的变化（酒店公共空间布局因装修升级改造等发生变化）而做出相应的适应。

本节以下的篇幅采用用例模型说明机器人系统的业务需求：



**图2.1-1 用例图**

### 基本用例

### 宾客引领用例

|  |  |
| --- | --- |
| 用例名称 | 宾客引领 |
| 主要参与者 | 在酒店入住的客户 |
| 次要参与者 | 机器人底盘 |
| 情景目标 | 客户在机器人的引领下尽快到达房间或其他酒店设施 |
| 前置条件 | 机器人适应当前酒店环境且闲置 |
| 启动 | 顾客决定在机器人的引领下到达酒店内某一确定地点 |
| 场景 | 顾客走近机器人，进入机器人端软件用户界面  顾客在功能中选择“引领”  机器人端面板提示用户输入房间号或地点编号  顾客输入有效的房间号或地点编号  系统为机器人规划路径，机器人底盘启动运动  顾客跟随机器人运动，到达预定地点  机器人端面板显示已到达，顾客点击确认到达  机器人使用权被释放，底盘启动运动回到原处 |
| 后置条件 | 机器人完成任务后回到原处，能够被再次调用 |
| 使用时间 | 主要在办理入住较为集中的下午至晚间时段 |
| 使用频率 | 中等 |
| 优先级 | 高 |

* + 1. 巡查用例

|  |  |
| --- | --- |
| 用例名称 | 巡查 |
| 主要参与者 | 酒店管理者 |
| 次要参与者 | 机器人底盘、机器人摄像头传感器 |
| 情景目标 | 酒店管理者无需亲自走动即可查看酒店公共空间各角落情况 |
| 前置条件 | 机器人适应当前酒店环境且闲置 |
| 启动 | 酒店管理者决定进行无接触式公共空间巡查 |
| 场景 | 酒店管理者决定巡查，进入监控端软件用户界面  管理者在功能中选择“巡查”  系统显示巡查可选的线路  管理者选择希望巡查的线路  机器人启动摄像头传感器，实时图像传回监控端  系统为机器人规划路径，机器人底盘启动运动  机器人按预定线路运动，同时传回实时画面供管理者观察  机器人完成预定巡航，返回原处  管理者点击确认完成巡航，释放机器人使用权 |
| 后置条件 | 机器人完成任务后回到原处，能够被再次调用 |
| 使用时间 | 主要在需要安全巡视的夜间至凌晨时段 |
| 使用频率 | 较高 |
| 优先级 | 低 |

* + 1. 房间送物用例（酒店管理者送物）

|  |  |
| --- | --- |
| 用例名称 | 房间送物（酒店管理者送物） |
| 主要参与者 | 酒店管理者 |
| 次要参与者 | 机器人底盘、机器人置物架 |
| 情景目标 | 客户点餐或第三方外卖送达前台，无接触送至客户房间 |
| 前置条件 | 机器人适应当前酒店环境且闲置、物品到达前台管理员处 |
| 启动 | 管理员决定为用户进行无接触配送到房间服务 |
| 场景 | 管理员走近机器人，进入软件机器人端用户界面  管理者在功能中选择“送物”  机器人面板提示送物的大小重量限制，提示输入房间号  管理者将物品放在机器人置物架上，输入所送房间号  系统为机器人规划路径，机器人底盘启动运动  机器人携带物品运动到达预定地点  机器人端语音呼叫顾客取物，顾客点击确认到达  机器人使用权被释放，底盘启动运动回到原处 |
| 后置条件 | 机器人完成任务后回到原处，能够被再次调用 |
| 使用时间 | 主要在用餐高峰期的晨间、午间和晚间 |
| 使用频率 | 中等 |
| 优先级 | 高 |

* + 1. 房间送物用例（客户自主订物）

|  |  |
| --- | --- |
| 用例名称 | 房间送物（客户自主订物） |
| 主要参与者 | 已经在房间入住的客户 |
| 次要参与者 | 机器人底盘、机器人机械臂、机器人摄像头传感器 |
| 情景目标 | 客户需要酒店提供毛巾、饮用水等物品，无接触送至房间 |
| 前置条件 | 机器人适应当前酒店环境且闲置 |
| 启动 | 客户需要酒店提供特定物品，决定使用机器人配送 |
| 场景 | 客户进入软件房间端用户界面  客户选择功能“送物”  房间端面板显示可送物品列表、剩余库存及每次可送上限  客户选择需要的物品种类和数量，点击确定  系统为机器人规划路径，机器人启动前往酒店货架  机器人启动摄像头传感器，识别定位所需物品  机器人启动机械臂装置，抓取所定位的物品  系统再次为机器人规划路径，机器人前往客户房间  机器人端语音呼叫顾客取物，顾客点击确认到达  机器人使用权被释放，底盘启动运动回到原处 |
| 后置条件 | 机器人完成任务后回到原处，能够被再次调用 |
| 使用时间 | 广泛分布在各个时段 |
| 使用频率 | 中等 |
| 优先级 | 高 |

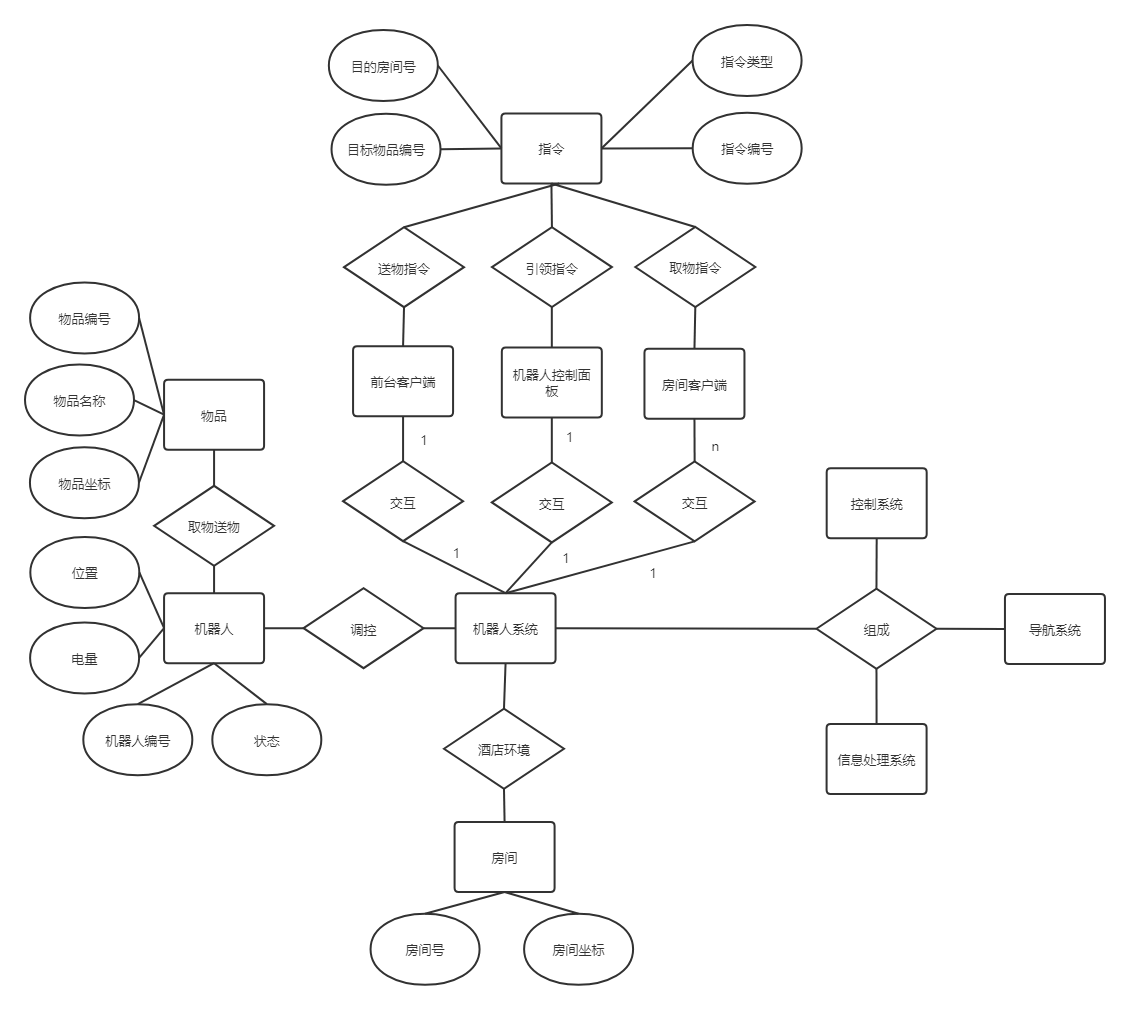
* + 1. 机器人适应新环境用例

|  |  |
| --- | --- |
| 用例名称 | 机器人适应新环境 |
| 主要参与者 | 酒店管理者 |
| 次要参与者 | 机器人底盘、机器人摄像头与雷达传感器 |
| 情景目标 | 酒店公共空间翻新或装修，格局变化，机器人应适应变化 |
| 前置条件 | 机器人闲置 |
| 启动 | 酒店最初部署或环境变化，管理者决定更新机器人系统信息 |
| 场景 | 管理者进入软件机器人端用户界面  管理者选择功能“更新环境信息”  系统为机器人规划算法，机器人底盘运动，摄像头和雷达传感器收集信息，管理者辅助  机器人学习完毕新的环境信息，保存至自身数据库  机器人回到原处，释放自身使用权 |
| 后置条件 | 机器人完成任务后回到原处，能够被再次调用 |
| 使用时间 | 初始部署机器人系统或酒店环境发生变化 |
| 使用频率 | 低 |
| 优先级 | 高 |

上述用例中，前四个是使用频率较高的主要功能用例，而最后一个是使用频率很低，但却是主要功能能够运行的前提的保障性功能用例。

* 1. 数据需求

机器人主要用于实现物品配送，人员引领等功能，这些功能依赖机器人系统调控得以实现，机器人本身拥有机器人编号、电量、状态和位置属性；对于机器人配送的物品，其本身拥有物品编号、物品名称、物品坐标属性；机器人系统出了调控机器人外，还通过前台客户端、房间客户端以及机器人本身的界面实现人机交互，在交互过程中产生指令，指令有三种——引领指令、取物指令和送物指令，综合来看指令的属性有指令编号、指令类型、目的房间号、目标物品编号；同时，实现上述功能的前提是机器人本身已经熟悉酒店环境，酒店环境主要由房间构成，房间实体的属性有房间号和房间坐标。因此系统中对象及其数据关系如下图：



**图2.3-1 系统ER图**

* 1. 功能需求

功能需求从业务需求的五大用例场景中提炼出来，其简要的功能分块有：控制、建图、引导、巡航、送物、取物六大功能，在五大业务需求用例场景的基础上再增加了控制功能，为用户提供直接控制机器人的接口。

* **控制功能**

控制功能是手动建图的基础，也是用户直接控制机器人运动的接口。

* **建图功能**

建图功能是其他四大业务需求主要功能的基础，机器人系统基于SLAM的环境感知方法，建图是导航的基础。

* **巡航功能**

巡航功能是提供给酒店管理者的，开启自动的固定线路巡航后可以无接触地查看酒店公共空间情况。

* **送物功能**

巡航功能是提供给酒店管理者的，开启自动的固定线路巡航后可以无接触地查看酒店公共空间情况。

* **取物功能**

取物功能是系统中最复杂的，用到机器人部件和节点最多的功能，需要同时用到机器人运动底盘、机械臂和传感器。

* 1. 非功能需求

性能指标（按照普通程度的室内复杂程度）：

* SLAM算法构建地图的速度不低于10分钟/100平方米。
* 机器人导航过程中的运行速度可设定等级，最高速度不超过4米/秒。
* 命令发布到响应的时间不超过1秒。
* 目的地设定到规划路径完成时间不超过1秒。

质量属性：

* 系统可移植性较强，通过安装文档可以移植到纯净的Ubuntu16.04系统。
* 系统鲁棒性较强，用户界面应对各种异常输入不会向外部抛出异常。

1. 数据库设计

3.1 数据元素表

针对上述的数据需求和功能需求，结合实际得出本系统要处理的数据元素表如下：

**表3.1-1 数据元素表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项 | 别名 | 数据类型 | 宽度 | 含义 |
| 房间号 | roomNo | varchar | 50 | 房间唯一标识 |
| 房间x坐标 | roomIdx\_x | double | 64 | 房间在地图上的x坐标 |
| 房间y坐标 | roomIdx\_y | double | 64 | 房间在地图上的y坐标 |
| 机器人编号 | robotNo | varchar | 50 | 机器人唯一标识 |
| 机器人状态 | robotState | int | 32 | 机器人状态 |
| 机器人位置x坐标 | robotIdx\_x | double | 64 | 机器人在地图上的位置x坐标 |
| 机器人位置y坐标 | robotIdx\_y | double | 64 | 机器人在地图上的位置y坐标 |
| 机器人电量 | robotPower | int | 32 | 机器人电量 |
| 物品编号 | goodsNo | varchar | 50 | 物品唯一标识 |
| 物品名称 | goodsName | varchar | 50 | 物品名字 |
| 物品x坐标 | goodsIdx\_x | double | 64 | 物品在地图上的位置x坐标 |
| 物品y坐标 | goodsIdx\_y | double | 64 | 物品在地图上的位置y坐标 |
| 指令编号 | insNo | varchar | 50 | 指令唯一标识 |
| 指令类型 | insType | int | 32 | 指令的类型——1为引领指令，2为取物指令，3为送物指令 |
| 目的房间号 | insRoomNo | varchar | 50 | 引领、取物、送物指令中机器人要到达的房间号 |
| 目标物品编号 | insGoodsNo | varchar | 50 | 取物、送物指令中目标物品的编号，引领指令时为默认值0 |

3.2 基本表

**表3.2-1 房间信息表**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 字段描述 |
| roomNo | varchar | 50 | 房间号 |
| roomIdx\_x | double | 64 | 房间在地图上的x坐标 |
| roomIdx\_y | double | 64 | 房间在地图上的y坐标 |

注：roomNo为主码

表3.2-2 机器人信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 字段描述 |
| robotNo | varchar | 50 | 机器人唯一标识 |
| robotState | int | 32 | 机器人状态 |
| robotIdx\_x | double | 64 | 机器人位置x坐标 |
| robotIdx\_y | double | 64 | 机器人位置y坐标 |
| robotPower | int | 32 | 机器人电量 |

注：robotNo为主码

表3.2-3 物品信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 字段描述 |
| goodsNo | varchar | 50 | 物品唯一标识 |
| goodsName | varchar | 50 | 物品名字 |
| goodsIdx\_x | double | 64 | 物品在地图上的位置x坐标 |
| goodsIdx\_y | double | 64 | 物品在地图上的位置y坐标 |

注：goodsNo为主码

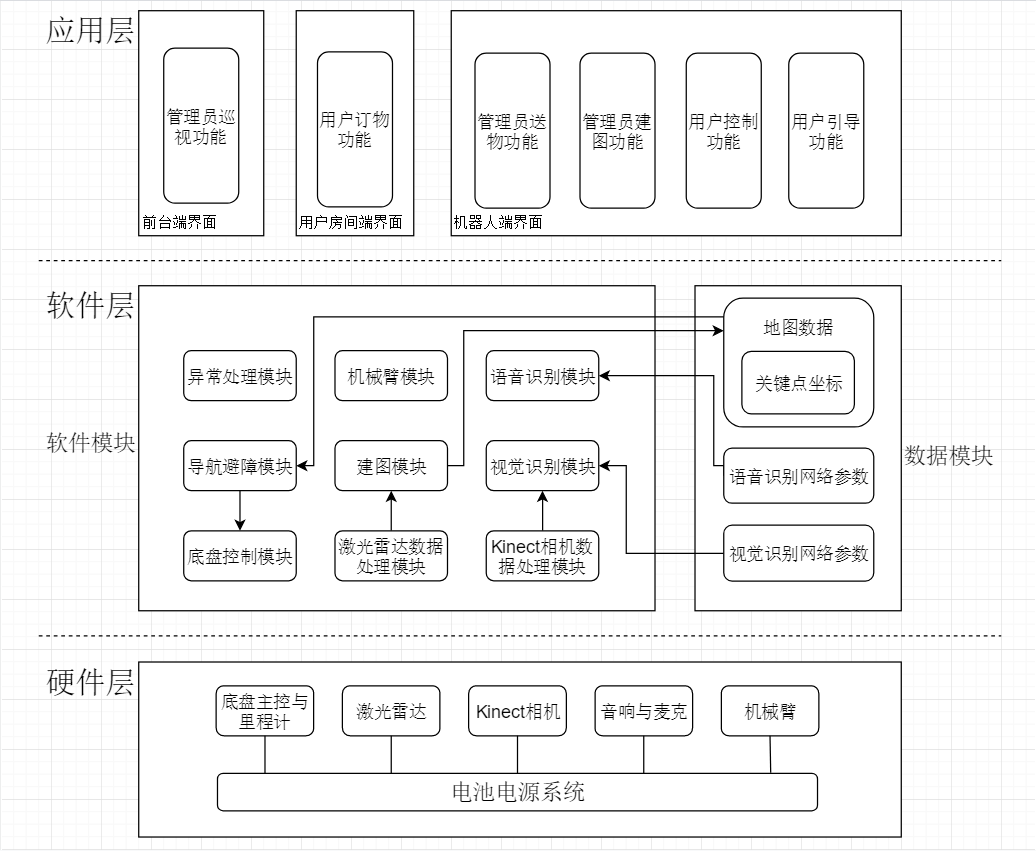
表3.2-4 指令信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 字段描述 |
| insNo | varchar | 50 | 指令唯一标识 |
| insType | int | 32 | 指令的类型——1为引领指令，2为取物指令，3为送物指令 |
| insRoomNo | varchar | 50 | 引领、取物、送物指令中机器人要到达的房间号 |
| insGoodsNo | varchar | 50 | 取物、送物指令中目标物品的编号，引领指令时为默认值0 |

注：insNo为主码，insRoomNo为对应房间信息表的外码，insGoodsNo为对应物品信息表的外码

1. 体系结构设计

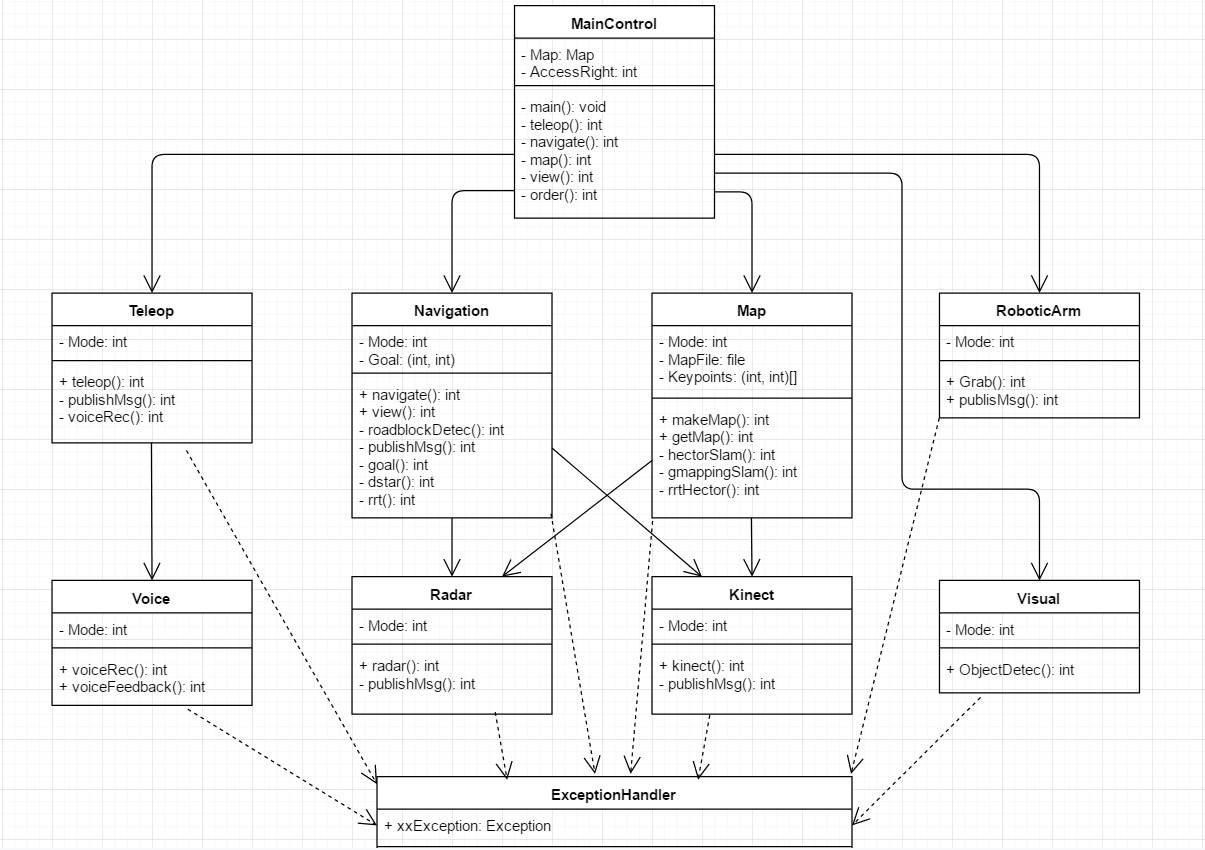
4.1 总体体系结构设计



**图4.1-1 总体设计层次结构图**

总体体系结构根据嵌入式系统开发的特点，自下而上划分为硬件层、软件层和应用层。其中硬件层是机器人的硬件系统，包括了底盘主控与里程计、激光雷达、Kinect相机、音响与麦克风、机械臂以及为所有设备供电的电池电源系统。软件曾在上图中被划分为软件模块与数据模块，软件模块被划分为九个模块单元，分别负责不同的功能部分，数据模块是系统运行时需要存储的数据（不包含在ROS系统直接传递的数据），软件模块与数据模块交互完成任务。最上层是应用层，是用户直接与系统交互的接口。根据需求，应用层设计为前台端、用户房间端和机器人端三种用户界面，拥有不同的功能权限，以实现五项用例需求。

4.2 软件体系结构设计



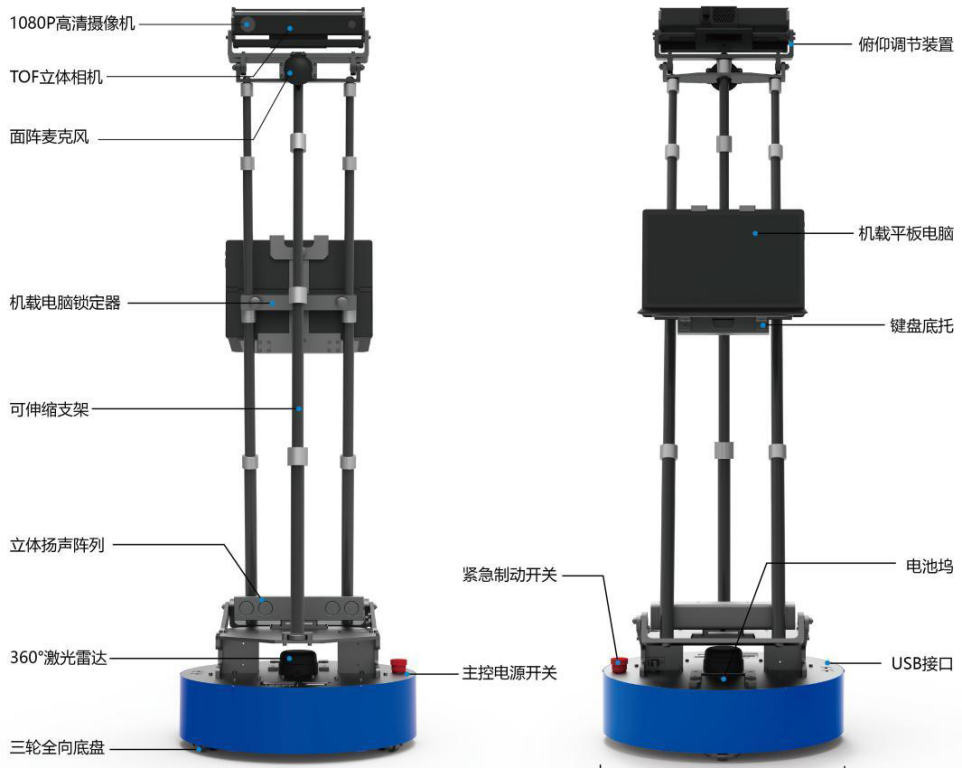
**图4.2-1 软件体系结构类图**

软件开发是本项目开发的重点，如上图是我们初步设计的软件体系结构。主控类之下共有八个主要功能类实现不同功能，而异常处理类负责处理运行中的异常。在设计初期，并不能够完善地预知设计和运行过程中需要处理哪些方面的异常，因此暂时假设每个主要功能类都可能抛出异常。具体的类名和功能如下：

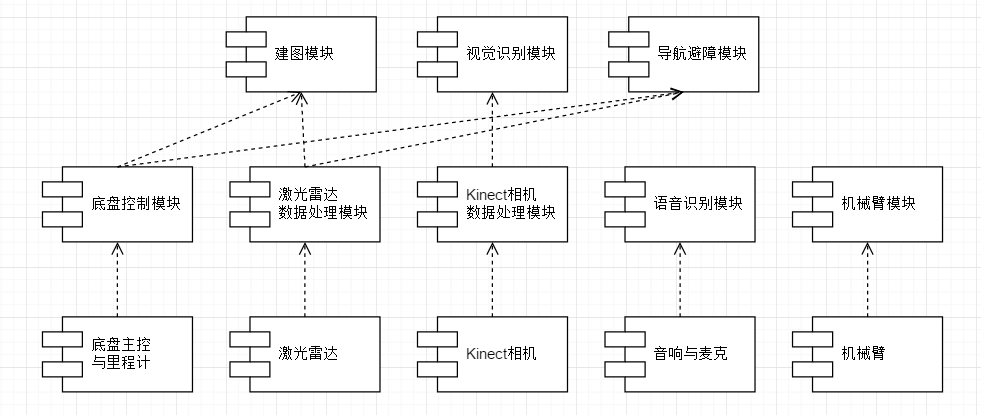
**表 4.2-2 类概况表**

|  |  |
| --- | --- |
| 类名 | 功能概述 |
| MainControl | 调用各具体功能的主要控制类 |
| Teleop | 直接控制机器人底盘的运动，包括了键盘控制和语音控制 |
| Navigation | 实现定点导航，以及调用摄像头画面巡航功能 |
| Map | 实现建图、地图存储管理、提供地图信息功能 |
| RoboticArm | 控制机械臂各个关节的运动，封装成抓取动作 |
| Radar | 从雷达硬件获取扫描到的信息，向上提供信息 |
| Kinect | 从Kinect相机获取RGB及深度信息，向上提供信息 |
| Visual | 专用于物品识别，接收图像信息，给出识别结果 |
| Voice | 接收麦克风的语音信息，提供识别结果，根据指令发出反馈语音 |

4.3 硬件体系结构



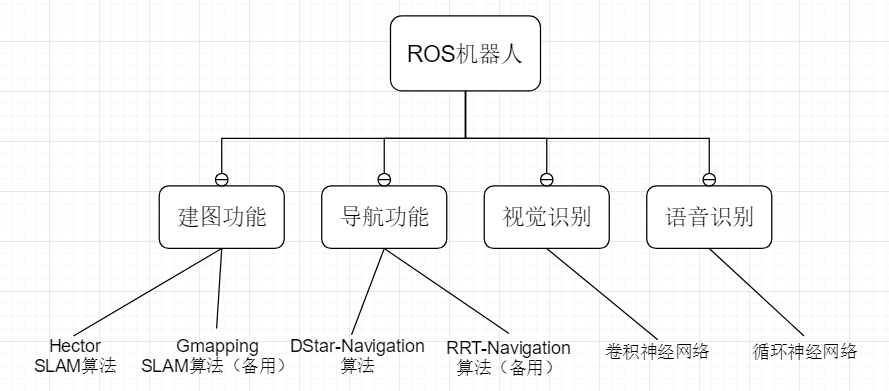
**图4.3-1 启智ROS机器人硬件结构组成**



**图4.3-2 硬件体系结构构件图**

本项目使用的主要硬件设备是启智ROS机器人，其结构组成图中展示了主要的硬件设备组成，而硬件体系结构构件图展示了本项目为硬件设备分组并应用硬件的情况。在构建图中，我们为每一组应用到的硬件单独编写相应的软件处理模块，以确保系统运行的稳定性。处理模块为上层软件功能或应用功能提供服务。

4.4 技术体系结构



**图4.4-1技术体系结构图**

本项目实现预定功能主要的技术如上图所示（地盘控制、坐标变换和里程计的应用等底层控制技术未列出），其中建图和导航是系统正常运行的核心。本项目拟采用基于二维激光雷达信息的Hector-SLAM算法作为建图的主要运行算法，其优点在于建图速度快且完善。基于图像和深度信息的Gmapping-SLAM算法作为备用算法，在雷达失效的情况下使用。导航单元拟使用DStar算法作为主要的路径规划和导航算法，在算法无法高效工作时采用备用的RRT算法。而视觉识别和语音识别模块主要采用主流的卷积神经网络和循环神经网络实现，具体的架构和算法有待开发过程中试验和筛选。

1. 接口设计

5.1 系统用户界面

用户界面是系统与用户交互的直接媒介。用户界面设计应考虑到该系统面向的客户类型为外行型/初学型，应设置为菜单界面为正文和图标混合类型。为了减少用户的输入量，输入将采取选择方式，并以设置好默认值。用户可通过鼠标和语音框来完成对应任务的操作。

主要界面有login界面，任务选择界面，具体任务对应的参数设置界面。

具体的用户界面应当随业务需求和功能需求划分前台端（监控端）、房间端和机器人端三个不同端口，根据各自的需求分别设计。但不同端的用户界面风格应保持一致。

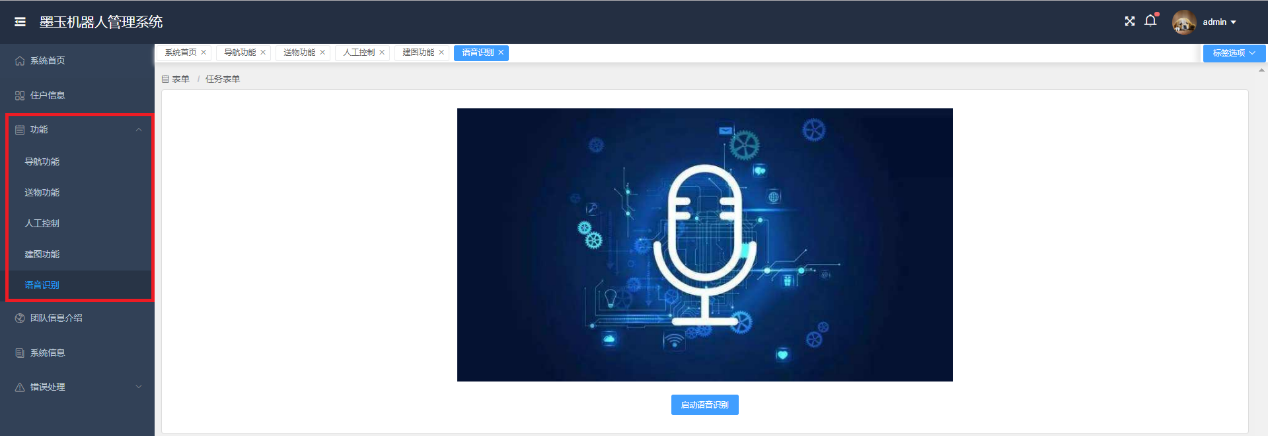
**前台端**



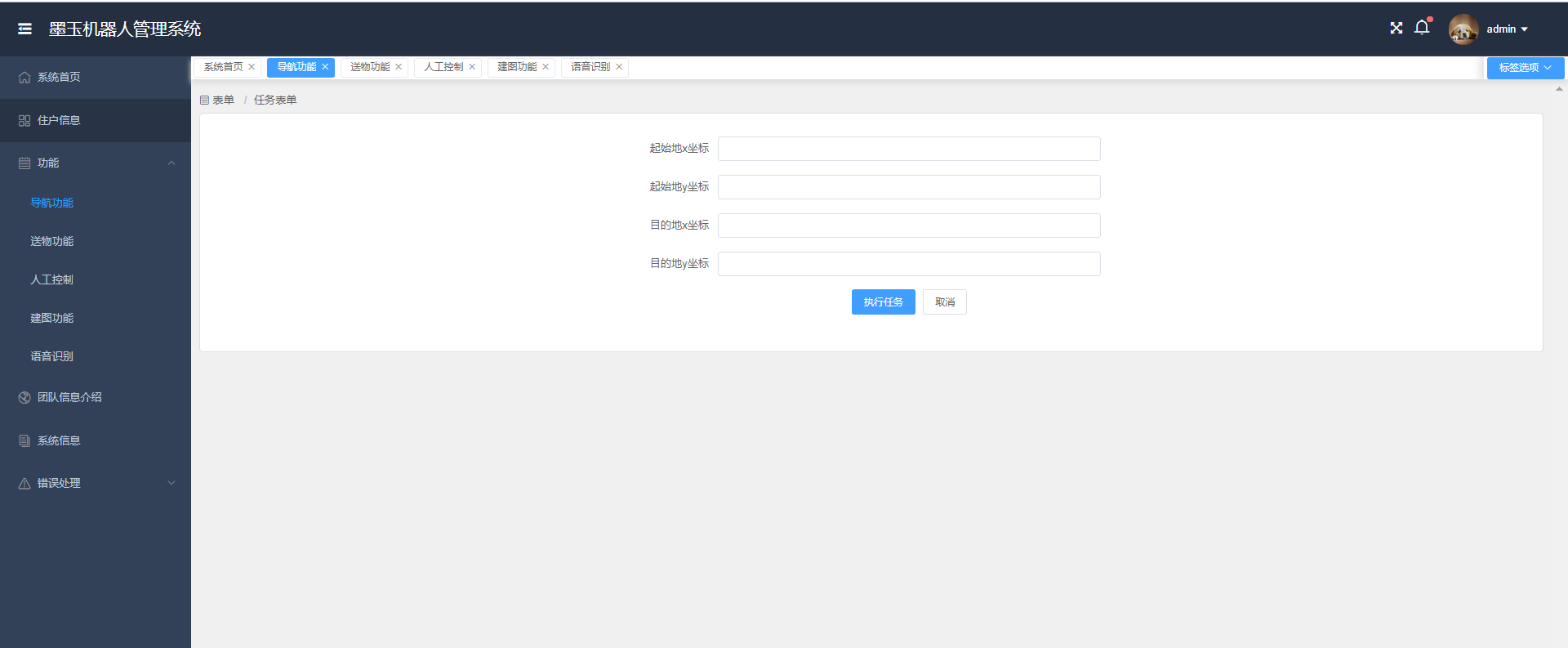
**图5.1-1 前台登录界面**



**图5.1-2 任务记录界面**

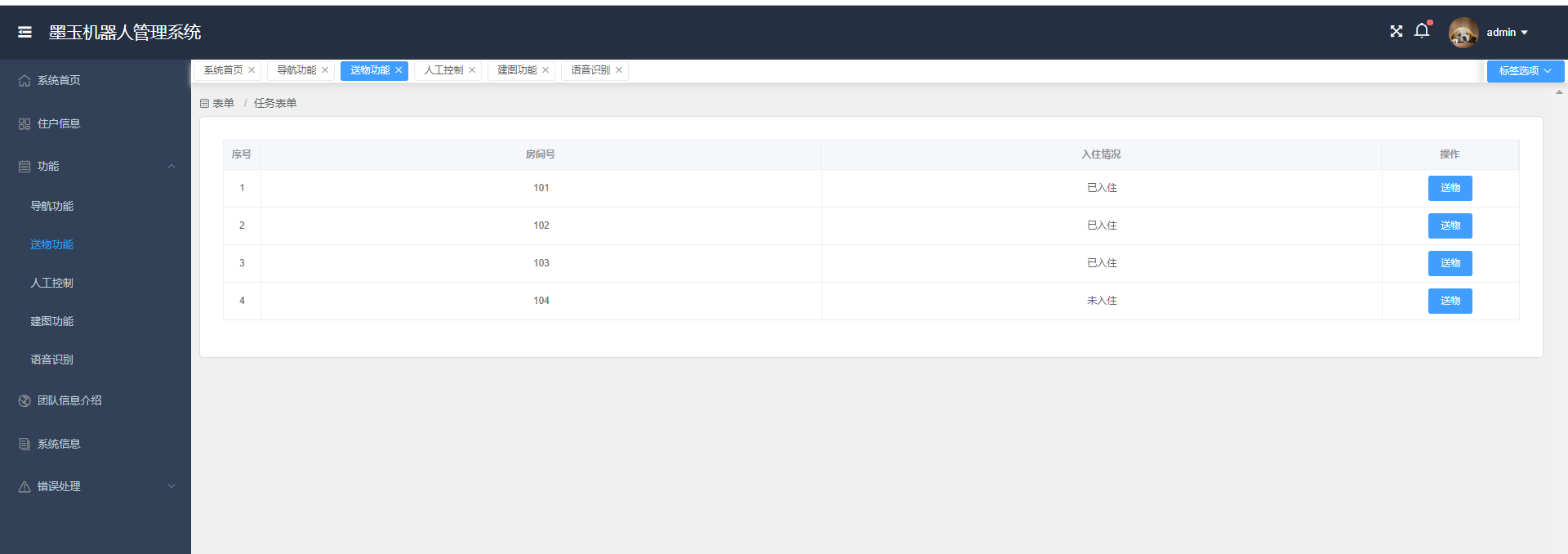


**图5.1-3 前台功能选择界面**



**图5.1-4 前台巡航功能界面**

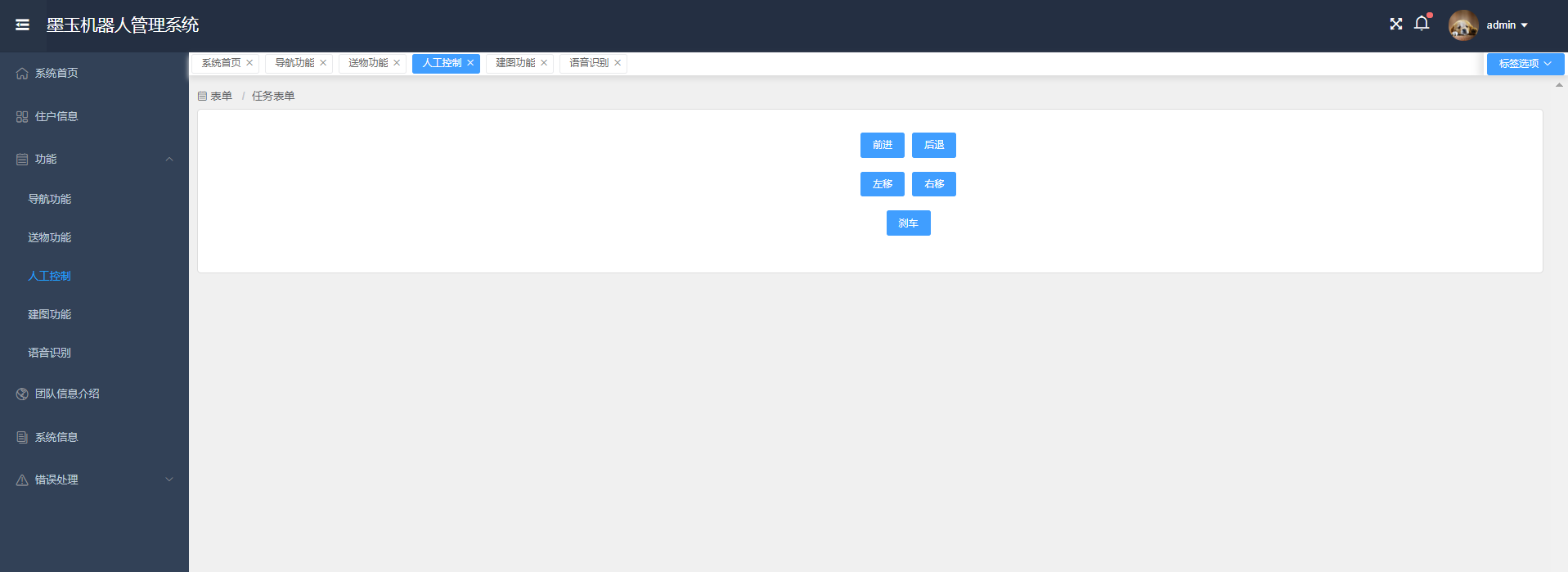
前台端（监控端）对应的主要功能仅有巡航功能，而机器人状态查询的视图仅为查询资料，无需赘述，巡航监控的用户界面如上图所示。此外监控端需要管理员权限，因此需要多加管理员登陆界面。

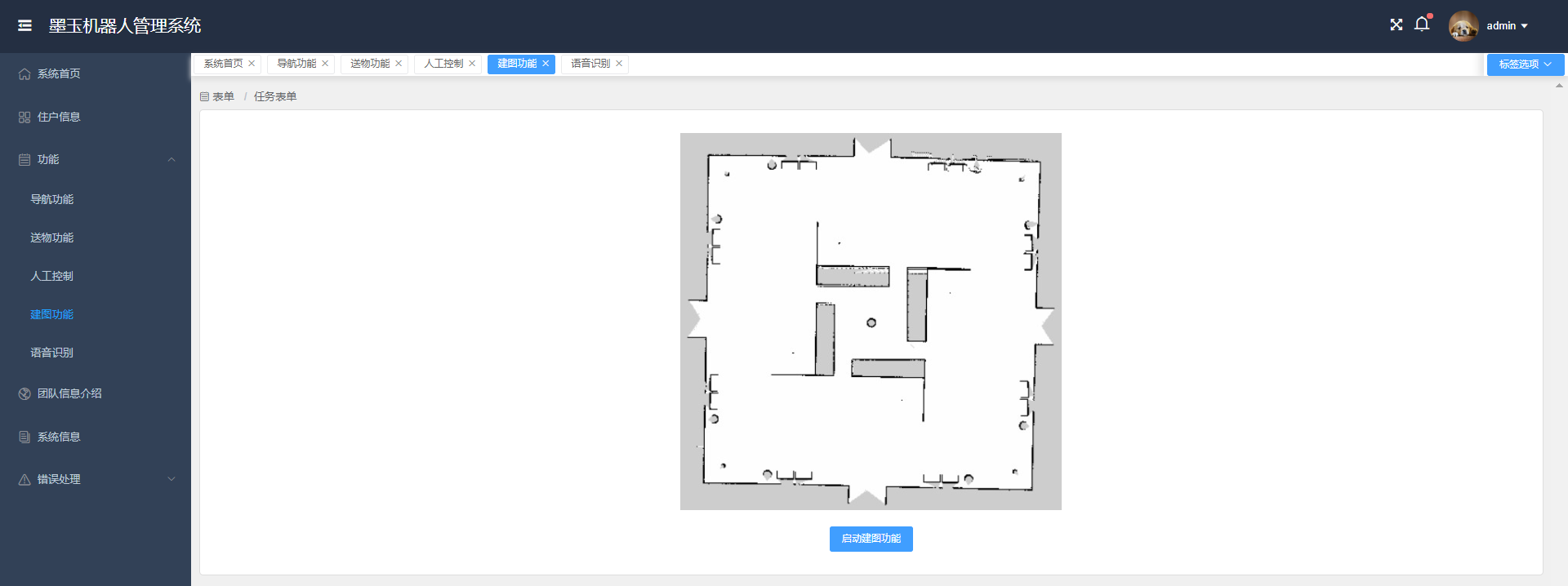
**房间端** 

**图5.1-5 房间端送货功能**

房间端对应的主要功能仅有取物，其功能选择页面主要供客户使用，因此无需管理员权限去登陆，在此机器人取物过程中的用户界面和点击确认送达的界面不再赘述。

**机器人端**

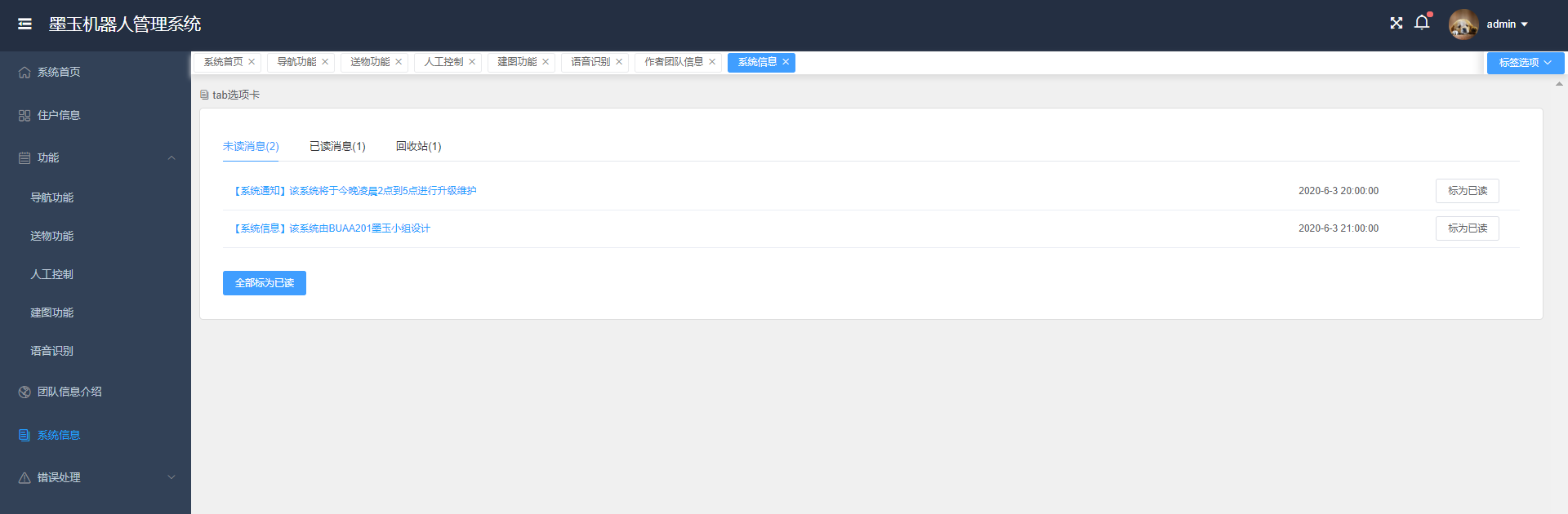




**图5.1-6 机器人端界面**

机器人端是功能最多的端，也是用户界面开发主要的端口，其进入机器人端首先区分管理者和客户，管理者需登录，其功能划分也有所不同，如上图的草图所示，但其功能较为直观，进一步细化的用户界面草图也无需赘述。

考虑到系统需要定期维护和更新，添加了系统通知界面，让用户和工作人员可以做好提前准备。

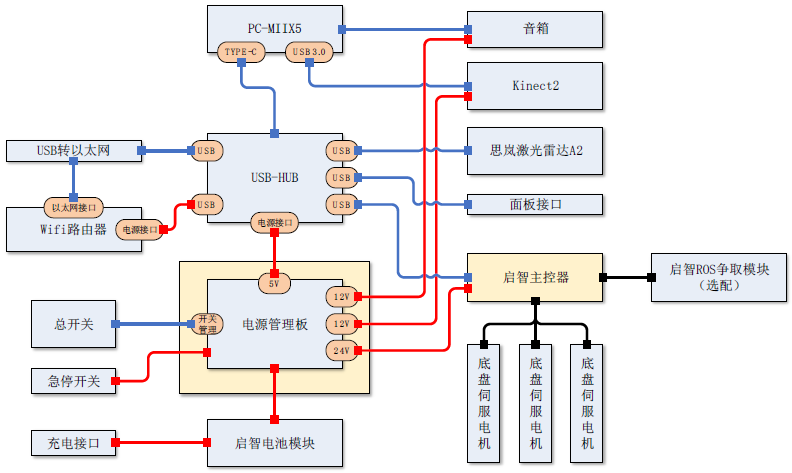


**图5.1-7 系统通知界面**



**图5.1-8 作品信息页面**

5.2 硬件接口



**图5.2-1 启智ROS电气接口结构图**

* 机载平板电脑：位于机器人躯干部分外挂的计算机，电脑上装载ROS，是机器人运行的基本硬件环境。通过USB接口与机器人底盘内的USB-HUB 连接。USB-HUB将计算机的USB接口扩展为多路。扩展后的USB接口分别连接到启智控制器（以异步串口方式访问）、USB转以太网接口、激光雷达、面板接口。
* 传感器模块：包括激光雷达、立体相机和姿态测量系统
  + 激光雷达：一个高速旋转的激光测距探头，将机器人周围360°的障碍物分布情况探测出来，形成障碍物轮廓的俯视二维点阵输入到ROS系统里，之后通过SLAM技术进行机器人的自身定位，为机器人的移动导航提供数据基础。
  + 姿态测量系统：机器人的底盘内置了一个三轴姿态测量系统，可以实时监测机体的朝向、翻滚及俯仰角度。为机器人的导航及行进提供重要数据，并让机器人在运动过程中发现倾倒的风险，即时采取紧急措施。
  + 立体相机：RGB-D立体相机，可通过机械安装调节其视角，以便对准需要进行视觉识别的目标区域。相机可以输出RGB彩色视频流和Depth深度数据三维点云，借助OpenCV和PCL等开源图像库，可以对目标物进行准确的识别和定位，以便进行后续的任务。
* 底盘控制器：控制器内部运行了启智ROS机器人专用固件，负责PC机与机器人之间的数据交互。
* 伺服电机：在启智伺服电机模块内的驱动控制板算法中，将输入的速度信息进行了单位换算，输入速度为减速器输出轴的转速。模块反馈的位置信息为编码器反馈的位置信息。
* 电池模块：启智ROS机器人的电源由电池模块供给，该电源模块内部由7枚3500mA/h容量的锂离子电池串联组成，内置电池保护板。该模块输出电压与当前剩余电量有关，剩余电量越少电压越低，正常工作输出电压范围23.1V至29.4V。

5.3 内部接口

本项目使用了层次模型作为软件体系结构，因此主要设计目标即为设计层次间的交互的接口即可。每个用户需求都被拆分为更具体的子需求，这些子需求或者在本层执行完毕，要不会被进一步抽象，将指令传输到下一层。下面将对内部接口进行具体介绍。

5.3.1表现层/业务层接口

表现层/业务层接口的主要工作在于将机器人需要满足的功能和用户有关执行命令的数据进行转换，以规范清晰的形式展示在前端。

本部分涉及的接口有：

（1）语音识别接口

（2）数据格式映射接口

5.3.2 业务层/持久层接口

业务层/持久层接口的主要工作在于将业务层的数据需求交给持久层，将业务层所需要用到的具体数据从持久层传回到业务层进行处理。另外，命令执行过程可能会因为系统故障或其他软硬件问题出现异常。这时候持久层需要将异常信息传递回业务层，由业务层进行异常中断处理。

本部分涉及到的接口有：

（1）SLAM地图绘制接口

（2）物体三维扫描接口

（3）底盘与机械臂运动控制接口

（4）导航路径点读取系统

（5）中断异常处理接口

5.3.3 持久层/数据库层接口

持久层/数据库层接口的主要工作在于将最底层数据库存储的信息进行更删改查等基本操作。

本部分涉及到的接口有：

（1）用户数据读/取接口

（2）地图表读/写接口

5.4 软件接口

系统软件的内部模块接口如下表所示：

**表5.4-1 系统内部软件接口一览表**

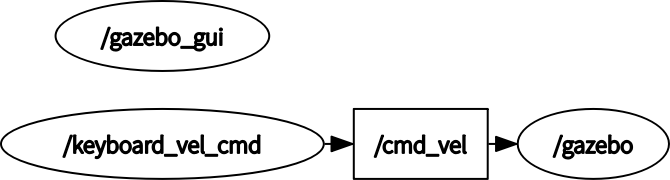
|  |  |  |  |
| --- | --- | --- | --- |
| 所属功能模块 | 返回值类型 | 接口名 | 功能概述 |
| 控制模块  Teleop | int | teleop | 进入控制模式，处理控制指令 |
| int | voiceRec | 将识别出的语音信息处理为控制信息 |
| 导航避障模块  Navigation | int | navigate | 接收目的地信息，发布前往目的坐标指令 |
| int | view | 调用相机图像信息，发布以供用户巡查 |
| int | roadblockDetec | 检测前方障碍物，若有障碍物则重新规划 |
| int | dstar | DStar算法接口（主要导航算法） |
| int | rrt | RRT算法接口（备用导航算法） |
| 建图模块  Map | int | MakeMap | 外部调用的建图接口，发布起止指令 |
| int[] | getMap | 向外部提供当前地图信息 |
| int | hectorSlam | Hector-SLAM算法（主要建图算法） |
| int | gmappingSlam | Gmapping-SLAM算法（备用建图算法） |
| int | rrtHector | 自动导航+探索建图 |
| 语音识别模块  Voice | int | voiceRec | 检测接收到的语音信息，提供检测结果 |
| int | voiceFeedback | 根据指令控制音响发出语音反馈 |
| 机械臂模块  RoboticArm | int | grab | 机械臂的抓取动作 |
| 雷达数据模块  Radar | int | radar | 接收雷达传感器信息，封装并向上提供 |
| Kinect数据模块  Radar | int | kinect | 接Kinect相机RGB与深度信息，封装并向上提供 |
| 视觉识别模块  Visual | int | objectDetec | 检测接收到的图像信息，提供检测结果 |

1. 详细设计

6.1 底盘控制模块

底盘控制模块根据接收命令形式的不同设计有两种工作模式——键盘控制模式和语音控制模式。两种工作模式的本质都是将用户直接输入的指令转化为向机器人底层硬件发布速度指令。依托ROS通信架构，键盘控制模式直接从键盘接收用户输入的方向及速度控制指令，将其转化为底层硬件需要的速度控制矢量，作为Message发布到相应的Topic上；而语音控制模式依赖语音识别模块，将语音识别模块分析好的控制指令转化为硬件需要的速度控制矢量发布Message。

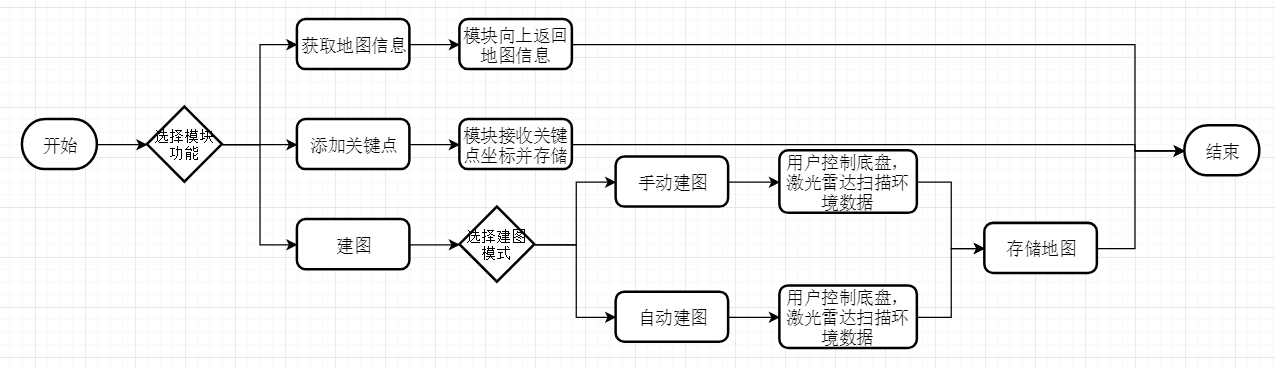
（语音控制模块现已调试完成）

**图6.1-1 仿真下的底盘控制模块交互图**

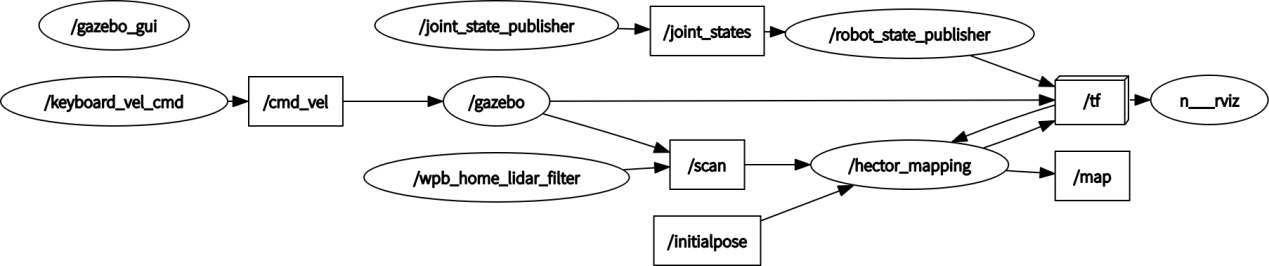
6.2 建图模块

建图模块负责建立环境地图并管理地图，向其他模块提供地图信息。同时，环境中的关键位置坐标也在建图模块一并存储和管理，因此建图模块有三方面功能：建图、获取地图、添加关键点。建图功能是建图模块的核心，其分为手动建图和自动建图两种模式，手动建图需要同时启用底盘控制模块，用户控制底盘移动的同时激光雷达扫描四周环境建立地图，自动建图只需提前指定若干探索点，机器人依据导航算法自主探索环境，同时激光雷达扫描四周环境建立地图。激光雷达使用Hector-SLAM算法，当激光雷达出现故障时，转而使用Kinect相机并启用备用的Gmapping-SLAM算法。

（硬件异常处理现已完成）

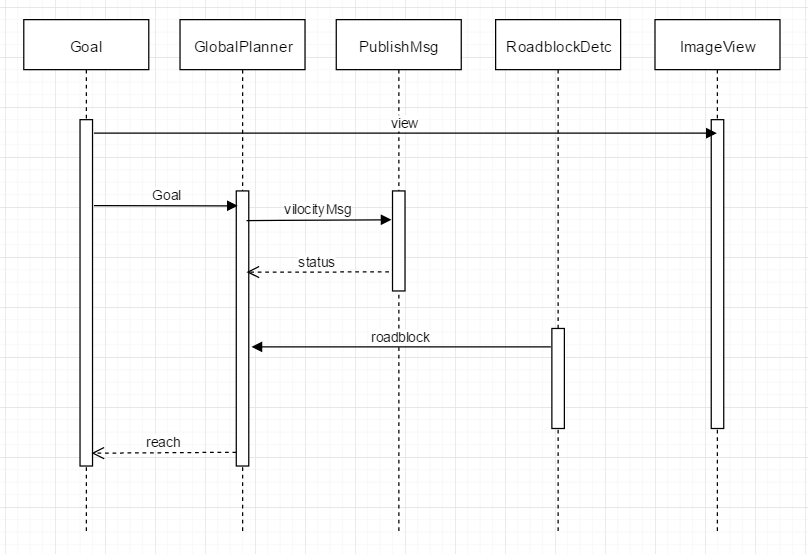


**图6.2-1 建图模块运行流程图**

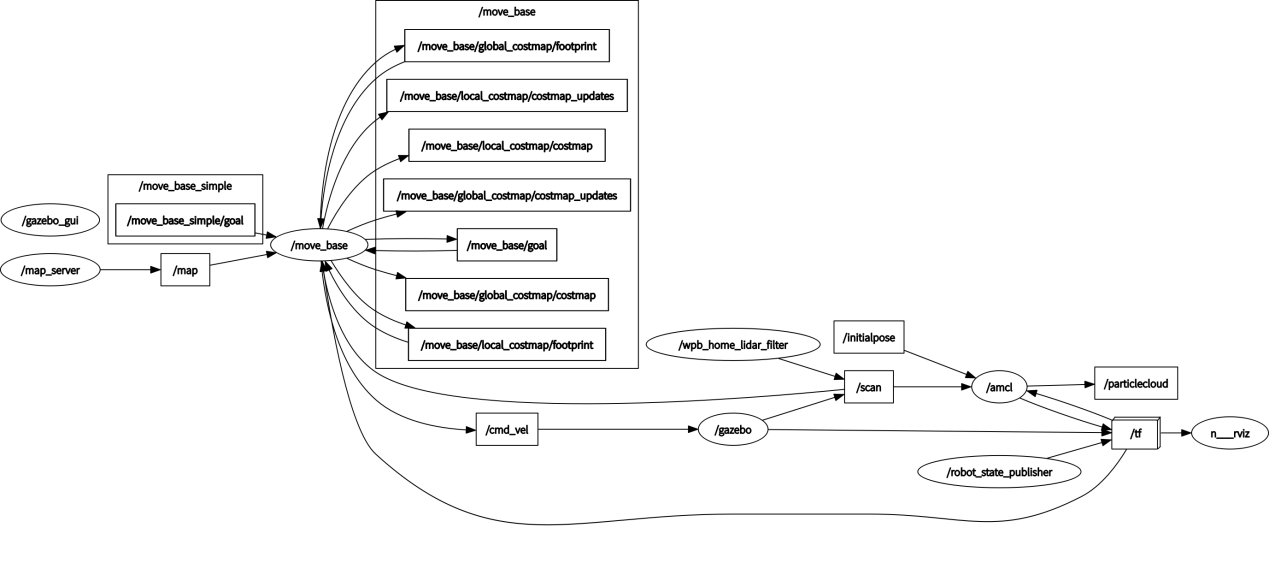
**图6.2-2 仿真下的建图模块交互图**

6.3 导航与避障模块

导航模块是实现多项需求用例的核心，它与控制模块相通之处在于都会通过ROS消息发布机制向地盘硬件对应的Topic发布速度控制信息Message，但导航模块发布消息不直接来源于用户控制，因此其运行的稳定性和避障能力更为重要。导航模块需要同时提供目的地路径规划、发布速度控制消息、调用Kinect相机数据（以供巡查）和基于激光雷达数据避障四方面的功能，四方面协作的活动图如下：

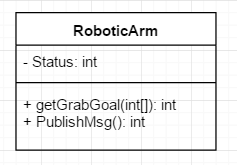


**图6.3-1 导航模块活动图**

**图6.3-2 仿真下的导航模块交互图**

6.4 机械臂模块

机械臂模块主要将机械臂硬件关节的运动封装起来，模块接收抓取指令（待抓取位置的三维中心坐标以及物品抓取宽度），将指令转化为对机械臂关节运动的控制信息Message，通过ROS消息发布机制发布到对应的Topic。

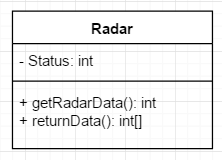


**图6.4-1 机械臂模块类图**

6.5 激光雷达数据模块

激光雷达数据模块用于向其他模块提供处理好的激光雷达数据信息，将激光雷达原始数据处理后返回给调用者。

（仿真环境中无需激光雷达数据模块，由仿真器直接发布消息到Topic）

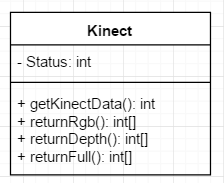


**图6.5-1 激光雷达数据模块类图**

6.6 Kinect相机数据模块

Kinect相机数据模块用于向其他模块提供处理好的Kinect相机数据信息。Kinect相机是带有深度信息的相机，主要提供RGB色彩信息和Depth深度信息，其基本数据相互组合可以产生多种不同图像效果，模块根据调用者的需求返回不同种类的图像格式。

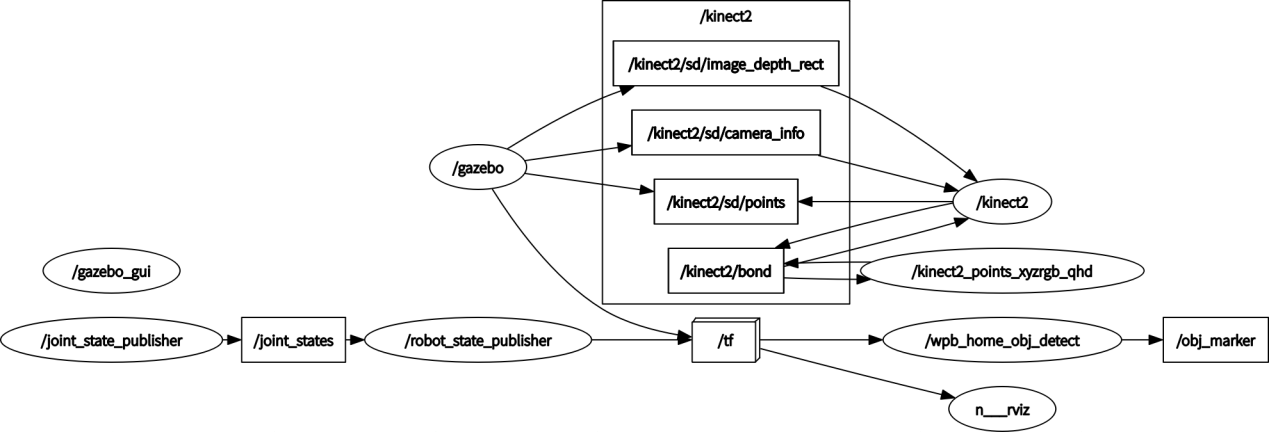
（仿真环境中无需Kinect相机数据模块，由仿真器直接发布消息到Topic）



**图6.6-1 Kinect相机数据模块类图**

6.7 视觉识别模块

视觉识别模块主要用于用户订物用例，机器人自动前往货架，检测并抓取物品送至客户房间，其中检测物品的工作由本模块完成。视觉识别模块管理着预先训练好的网络参数，在应用时接收相机发布的图像数据，通过神经网络以鉴别画面中是否有物品，检测物品的位置和大小。

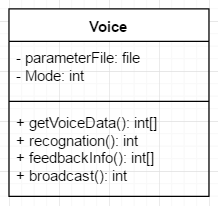


**图6.7-1 仿真下的视觉识别模块交互图**

6.8 语音识别模块

语音识别模块主要用于用户语音控制。同样通过将事先训练好的网络参数内置于模块中，模块接收麦克风收集的音频信息，识别出具体指令返回给调用者。此外，语音识别模块还兼有发布语音的功能，上层指令调出向用户反馈的语音，由语音识别模块向音响发布消息播放语音。

（语音识别和发布模块现已调试完成）



**图6.8-1 语音识别模块类图**

6.9 数据库实现

本项目采用python自带的sqlite3轻量级数据库实现了第3部分介绍的数据库设计。主要进行了创建数据库、创建数据表、定义数据对象、定义数据操作的工作。

6.9.1 创建数据库

connect = sqlite3.connect(Utils.dbName)

可以使用python的sqlite3库连接指定名字的数据库，如果该数据库不存在，则创建。

6.9.2 创建数据表

c.execute('''

        CREATE TABLE roomMessage(

            roomNo varchar(50) PRIMARY KEY,

            roomIdx\_x DOUBLE,

            roomIdx\_y DOUBLE

        )

    ''')

连接数据库后，使用cursor执行创建数据表的sql指令，即可完成数据表创建。

6.9.3 定义数据对象

class Room :

    def \_\_init\_\_(self, rNo, rIdx\_x, rIdx\_y) :

        self.roomNo = rNo

        self.roomIdx\_x = rIdx\_x

        self.roomIdx\_y = rIdx\_y

class Robot :

    def \_\_init\_\_(self, rNo, rState, rIdx\_x, rIdx\_y, rPower) :

        self.robotNo = rNo

        self.robotState = rState

        self.robotIdx\_x = rIdx\_x

        self.robotIdx\_y = rIdx\_y

        self.robotPower = rPower

class Goods :

    def \_\_init\_\_(self, gNo, gName, gIdx\_x, gIdx\_y) :

        self.goodsNo = gNo

        self.goodsName = gName

        self.goodsIdx\_x = gIdx\_x

        self.goodsIdx\_y = gIdx\_y

class Instruction :

    def \_\_init\_\_(self, iNo, iType, iRoomNo, iGoodsNo) :

        self.insNo = iNo

        self.insType = iType

        self.insRoomNo = iRoomNo

        self.insGoodsNo = iGoodsNo

根据第3部分的基本数据表，定义数据对象来实现数据库与应用程序的交互。

6.9.3 定义数据操作

def insertRoom(room) :

    connect = sqlite3.connect(Utils.dbName)

    cursor = connect.cursor()

    table = 'roomMessage'

    attrs = 'roomNo, roomIdx\_x, roomIdx\_y'

    values = (Utils.toStr(str(room.roomNo)) + ',' + str(room.roomIdx\_x) + ',' + str(room.roomIdx\_y))

    cursor.execute(Utils.insertSql.replace('\_TABLE',table).replace('\_ATTRS',attrs).replace('\_VALUES',values))

    connect.commit()

    connect.close()

def queryRoom(roomNo) :

    connect = sqlite3.connect(Utils.dbName)

    cursor = connect.cursor()

    table = 'roomMessage'

    condition = ('roomNo = ' + Utils.toStr(roomNo))

    cursor.execute(Utils.querySql.replace('\_TABLE',table).replace('\_CONDITION',condition))

    room = cursor.fetchone()

    connect.close()

    return Room(room[0], room[1], room[2])

def updateRoom(room) :

    connect = sqlite3.connect(Utils.dbName)

    cursor = connect.cursor()

    table = 'roomMessage'

    # primary key cannot update

    updateStm = Utils.toUdeStm('roomIdx\_x', room.roomIdx\_x) + ',' + Utils.toUdeStm('roomIdx\_y' , room.roomIdx\_y)

    condition = ('roomNo = ' + Utils.toStr(room.roomNo))

    cursor.execute(Utils.updateSql.replace('\_TABLE',table).replace('\_UPDATESTM', updateStm).replace('\_CONDITION',condition))

    connect.commit()

    connect.close()

def deleteRoom(roomNo) :

    connect = sqlite3.connect(Utils.dbName)

    cursor = connect.cursor()

    table = 'roomMessage'

    condition = ('roomNo = ' + Utils.toStr(roomNo))

    cursor.execute(Utils.deleteSql.replace('\_TABLE',table).replace('\_CONDITION',condition))

    connect.commit()

    connect.close()

利用6.9.3定义的数据对象作为中间媒介，为每一个基本表提供增删改查的基本数据操作，来达到应用程序和数据的交互目的。

1. 运行与开发环境
   1. 运行环境

硬件运行环境：

**表7.1-1 硬件运行环境**

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 参数 |
| 主控器 | 1 | 机载平板电脑 |
| 视听传感器 | 1 | 1080P高清摄像机、TOF立体相机、面阵麦克风 |
| 360°激光雷达 | 1 | 360°无死角、最大距离8米 |
| 视觉传感器 | 1 | Kinect 2 |
| 电气模块 | 1 | 主控电源开关、紧急制动开关、USB接口、20W伺服电机、内置驱动、电源控制板、电池模块、控制器、伺服电机模块 |
| 移动装置 | 1 | 三轮全向底盘、3个全向轮 |
| 外挂 | 1 | 机载电脑锁定器、3个可伸缩支架、俯仰调节装置、键盘底托、电池坞、升级机械手爪 |
| 电池 | 1 | 24V3.5AH锂电池组 |

软件运行环境：

1. 基本运行环境：

**表7.1-2 基本运行环境**

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 说明 |
| Linux Ubuntu 16.04 | 1 | ROS系统的运行基础环境 |
| ROS Kinetic | 1 | ROS系统，提供机器人应用软件的系统运行环境 |

1. 应用运行环境：

**表7.1-3 应用运行环境**

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 说明 |
| URDF模型 | 1 | ROS系统直接加载 |
| 电机码盘里程计 | 1 | 启智ROS装备了带编码器的直流伺服电机，直接接收计数从而推算机器人移动里程 |
| IMU姿态传感 | 1 | 内置IMU单元实时获取机器人滚转、倾斜和朝向信息 |
| 三维立体视觉 | 1 | 利用TOF立体相机提供的数据进行三维模型重建 |
| SLAM环境建图 | 1 | 利用激光雷达提供的数据创建环境地图 |
| 自主定位导航 | 1 | 激光雷达与里程计数数据结合进行自主导航 |
| 动态目标跟随 | 1 | 锁定目标尾随移动 |
| 物品检测 | 1 | 通过立体相机检测三维空间物品 |
| 人脸检测 | 1 | 借助高分辨率相机进行人脸检测 |
| 传感器融合 | 1 | 将立体三维点云和二维地图进行融合 |
| 语音识别 | 1 | 借助语音引擎进行识别并提供语音合成定制 |
| 物品抓取 | 1 | 驱动抓取装置抓取物品 |

* 1. 软件环境

硬件开发环境：

**表7.2-1 硬件开发环境**

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 参数 |
| 处理器CPU | 1 | 双核、四核CPU，能够流畅运行Linux Ubuntu即可 |
| 内存 | 1 | 8GByte以上 |
| 硬盘 | 1 | 100GByte以上，足以容纳Linux系统、ROS系统、IDE及相关开发库文件。 |

软件开发环境：

**表7.2-2 软件开发环境**

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 说明 |
| Linux Ubuntu 16.04 | 1 | ROS系统和相关运行库的基础环境，在开发和测试时均需使用 |
| ROS系统 | 1 | 开发机器人应用依赖的基本库。 |
| VS Code | 1 | 软件开发时使用的Editor/IDE。 |

1. 需求可追踪性说明

8.1 功能需求说明

SDD文档中功能/非功能需求都可由机器人不同的功能组合进行实现。而这些模块每一个都对应着一个或多个SRS文档在接口部分用提及的代码接口。SRS通过对于接口和体系进行科学的设计，很好地实现了对于机器人应满足的功能需求。

SRS功能需求与SDD模块的对应关系如下表所示：

**表8.1-1 SRS需求用例与功能需求对应关系**

|  |  |  |
| --- | --- | --- |
| SRS需求用例 | 用例描述 | SDD对应模块 |
| 2.1宾客引领 | 宾客底达酒店完成入住手续后由机器人引领宾客到达房间 | 巡航功能 |
| 2.2巡查 | 酒店管理者无需亲自走动即可查看酒店公共空间各角落情况 | 巡航功能 |
| 2.3房间送物（酒店管理者送物） | 客户点餐或第三方外卖送达前台，酒店管理者接受物品并使用机器人无接触送至客户房间 | 巡航功能 |
| 2.4房间送物（客户自主订物） | 客户需要酒店提供毛巾、饮用水等物品，机器人自动运行至储物间，无接触送至房间 | 巡航功能、  取物/送物功能 |
| 2.5机器人适应新环境 | 酒店公共空间翻新或装修，格局变化，机器人应适应变化 | 控制功能、  建图功能 |

**表8.1-2 SRS功能需求与SDD模块对应表**

|  |  |  |
| --- | --- | --- |
| SRS功能需求 | 功能描述 | SDD对应模块 |
| 控制功能 | 用户可以通过键盘和语音两种方式对机器人下达命令。使用键盘操控时，系统会自动启动键盘控制节点，之后用户可以通过键盘控制机器人的移动。使用语音操控 | 6.1 底盘控制模块  6.8 语音识别模块 |
| 建图功能 | 机器人既可由用户操控手动建图，也可由本身实现自动建图。启动建图后机器人会启动可视化节点，依据SLAM算法建图并储存行程信息。 | 6.2 建图模块  6.5 激光雷达数据模块  6.6 Kinect相机数据模块 |
| 巡航功能 | 机器人可按照固定路线进行巡航，同时在路上可启动可视化节点以实现避障的功能。 | 6.3 导航与避障模块  6.6 Kinect相机数据模块 |
| 送物/取物功能 | 机器人可通过移动机械臂的方式取走或传递其开启可视化节点后观察到的物体。 | 6.3 导航与避障模块  6.4 机械臂模块  6.6 Kinect相机数据模块  6.7 视觉识别模块 |

8.2 非功能需求说明

8.2.1 运行性能

· SLAM算法构建地图的速度不低于10分钟/100平方米。

· 机器人导航过程中的运行速度可设定等级，最高速度不超过4米/秒。

· 命令发布到响应的时间不超过1秒

· 目的地设定到规划路径完成时间不超过1秒

8.2.2 可扩展性

· 用户需求可能会发生变化，这时候机器人软件开发系统应满足易于通过添加新接口或类实现新功能的需求

8.2.3 稳定性

· 系统鲁棒性较强，用户界面应对各种异常输入不会向外部抛出异常

8.2.4 安全性

· 区分用户权限，分为普通用户和root管理者两种操作者。二者对于数据的访问权限和对机器人的操作权限不同，避免出现系统被恶意攻击破坏的情况出现。