**智能家庭助手**

**软件设计说明书**

**SDD08**

**V1.0**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 啊对对队 | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 18375299 | 廖纪童 | 主要编制人，控制层接口设计 |
| 18375200 | 刘裕炜 | 审核人，调度器接口设计 |
| 18375182 | 范竞元 | 项目任务概要讨论，前端接口设计 |
| 19373106 | 裴宝琦 | 进度计划讨论，后端接口设计 |
| 18374457 | 刘传 | 资源计划讨论，后端接口设计 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 0.1 | 2022.4.15 | 廖纪童 | 刘裕炜 | 完成文件基本框架，系统架构和控制层接口设计 |
| 0.2 | 2022.4.15 | 刘传 | 廖纪童 | 完成数据库和后端接口设计 |
| 0.3 | 2022.4.16 | 刘裕炜 | 裴宝琦 | 完成硬件和调度器接口设计 |
| 0.4 | 2022.4.17 | 裴宝琦 | 范竞元 | 完成状态码定义，完善后端接口设计 |
| 0.5 | 2022.4.18 | 范竞元 | 刘传 | 完成前端接口设计 |
| 0.6 | 2022.4.19 | 全部成员 | 全部成员 | 根据接口设计完善第六章系统功能的详细设计 |
| 1.0 | 2022.4.19 | 廖纪童 | 刘裕炜 | 全面审核，完成SDD文档1.0 |

# 1 范围

## 项目概述

### 系统开发背景

随着物联网、机器人技术的日渐发展，越来越多场景中的设备可以被网络赋予“智能”，从而更方便地协助人。在需求类型较为固定的领域，如银行大厅导览、餐厅送菜等场景，时常能见到工作中的智能机器人。具体到家居方面，目前较为成熟的也有扫地机器人之类功能较为简单的智能家居设备投入应用。

但是，市面上售卖的智能家居大多只局限于联网控制的空调、冰箱、电灯等电器，机器人类也仅限于扫地机器人、教育机器人等。我们希望开发一种家庭服务机器人，能够在房间内根据用户的语音指令帮助用户拿取并运送物品。该机器人可以由用户手动建图后，根据用户的语音指令，自动规划路线取物并送回给用户。该机器人可以让用户不被取物打断手头工作，方便用户的日常生活。

### 主要功能

该服务机器人的主要功能如下：

* 能够手动建立使用场景的室内布局图并保存；
* 能够接受语音指令；
* 能够自动规划路径并导航；
* 能够躲避路径上的障碍物，顺利到达目的地；
* 能够实现物品抓取功能；
* 具备基本的异常处理功能。

### 非功能性需求

该服务机器人希望达到的非功能需求如下：

* 响应速度快；
* 功耗低，可以长时间续航；
* 处理能力强；
* 高系统可用性；
* 高可移植性；
* 功能具备完整性；
* 高运行效率；
* 强健壮性。

### 应用场景

该机器人可以在用户腾不出手或无法抽身时通过语音控制，帮助用户拿取物品。可以应用于日常家居、办公场所等地。

## 文档概述

本文档定义了项目的数据库结构，整体的体系结构设计，各个层级（包括前端，后端，调度层，控制层，硬件层）的接口规范，以及功能的详细的时许图，通过此文档可以开始本项目的代码编写，在编写过程中需要尽可能的符合文档要求，并修改不符合实际场景的部分。

### 文档用途

* 确定本产品的主要功能
* 规范数据库架构，保证数据库完备性
* 定义系统的整体架构，使开发变得更加清晰
* 定义各个接口的规范，方便开发者进行编码
* 定义多个功能的详细设计，方便不同层级接口之间的对接

### 文档组织

* 数据库设计
* 体系结构设计
* 接口设计
* 功能详细设计

## 术语和缩略词

本文档所涉及的专业的业务和技术术语：

表1

|  |  |
| --- | --- |
| 术语 | 含义 |
| Ubuntu | 一种以桌面应用为主的linux操作系统的发行版 |
| C++ | 一种面向对象的计算机程序设计语言 |
| gazebo | 机器人仿真工具，模拟器，也是一个独立的开源机器人仿真平台 |

业务和技术术语表

文档所涉及的缩略词及其全称：

表2

|  |  |
| --- | --- |
| 缩略词 | 全称 |
| ROS | Robot Operating System |
| SDD | Software Design Description |
| SDP | Software Development Plan |
| SRS | Software Requirement Specification |
| STD | Software Test Description |
| STP | Software Test Procedures |
| WBS | Work Breakdown Structure |

缩略词及其全称表

## 引用文档

1. 邹欣,现代软件工程构建之法(第3版)[M].北京:人民邮电出版社, 2017
2. （英）伊恩·萨默维尔（Ian Sommerville）著；彭鑫，赵文耘译.计算机科学丛书 软件工程 原书第10版[M].北京：机械工业出版社,2018.02.
3. 北京六部工坊科技有限公司，《启智ROS版\_开发手册\_20181109》，2018-11.9
4. SDP-软件开发计划
5. SRS-软件需求说明规格书

# 需求概述

【概述系统的需求。建议给出简要的业务需求、数据需求、功能/非功能性需求描述。】

## 业务需求

作为一款家庭服务机器人，我们致力于为客户提供人性化、多功能的服务，目前我们提出了如下的业务需求：

1. 系统功能设置
2. 手动建图
3. 语音识别
4. 图像识别
5. 路径规划
6. 物品取放
7. 人机交互

## 功能需求

作为家庭服务机器人，需要满足家庭应用场景中的较多需求，因此功能需求主要包括：

1. 语音识别，通过语音和用户进行交互
2. 美观简洁的图形界面，为用户提供友好的控制方式
3. 取物/放物的功能，主要包括，导航，控制机械臂取物/放物，物品识别，自动避障
4. 一定的异常处理能力，要求可以处理部分异常，如语音识别失败等

## 非功能需求

作为家庭服务机器人，由于面向的用户群体较为广泛，因此，也需要达到如下的非功能需求：

1. 性能指标，如响应时间，功耗，处理能力等，需要满足尽可能搞笑低功耗的进行工作。
2. 质量指标，如系统可用性，可移植性，完整性，效率，健壮性等，期望构建出一个可以适配多种环境的可移植的完整的家庭服务机器人。

# 数据库设计

## 用户信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 是否允许为空 | 内容描述 |
| id | int | 0 | 否 | 数据库生成的自增ID，主键 |
| username | char(255) | NULL | 否 | 用户名，登录所用的账号 |
| password | char(255) | NULL | 否 | 用户密码，使用MD5加密后存储 |
| email | char(255) | NULL | 是 | 用户邮箱，用于接受系统信息，如验证码等 |

## 家庭地图信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 是否允许为空 | 内容描述 |
| id | int | 0 | 否 | 数据库生成的自增id，主键 |
| create\_at | char(255) | NULL | 否 | 此地图的建立时间 |
| path | char(255) | NULL | 否 | 地图信息的存储路径 |
| cnt | int | 0 | 否 | 使用次数 |
| image | string | NULL | 否 | 图片的url |

## 节点信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 是否允许为空 | 内容描述 |
| id | int | 0 | 否 | 数据库生成的自增id，主键 |
| pos\_x | float | 0 | 否 | 节点在地图上的横坐标位置 |
| pos\_y | float | 0 | 否 | 节点在地图上的纵坐标位置 |
| map\_id | int | 0 | 否 | 节点属于的地图id，外键 |

## 事务记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 是否允许为空 | 内容描述 |
| id | int | 0 | 否 | 数据库生成的自增ID，主键 |
| type | int | 0 | 否 | 表示事务的类型，具体映射：   1. 路径规划 2. 物品识别 3. 语音识别 4. 取物 5. 放物 |
| user\_id | int | 0 | 否 | 外键，对应用户表的id |
| start\_time | char(255) | NULL | 否 | 事务开始时间 |
| finish\_time | char(255) | NULL | 否 | 事务结束时间 |
| state | int | 0 | 否 | 事务状态，具体映射：   1. 事务进行中 2. 事务以结束，且未出现异常 3. 事务异常 |

# 体系结构设计

【给出系统的总体结构，并所包含的设计单元及其之间的关系进行说明，建议采用*层次结构图，或者类图、时序图/协作图、状态图、活动图、构件图、部署图、包图*等）。可能涉及软件体系结构、硬件体系结构、技术体系结构、支撑体系（部署和实施方案）结构等各个方面，可根据实际情况每个方面分为一小节来写。】

## 体系结构概况

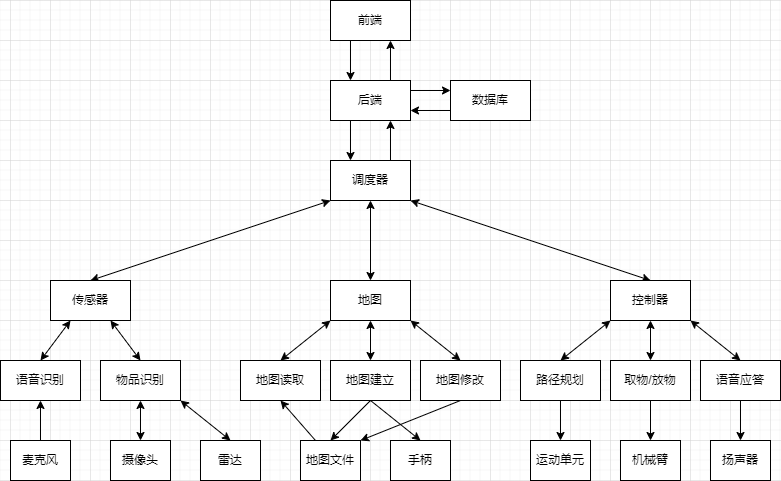
机器人系统可以分为四部分：用户层、调度层、控制层与硬件层。

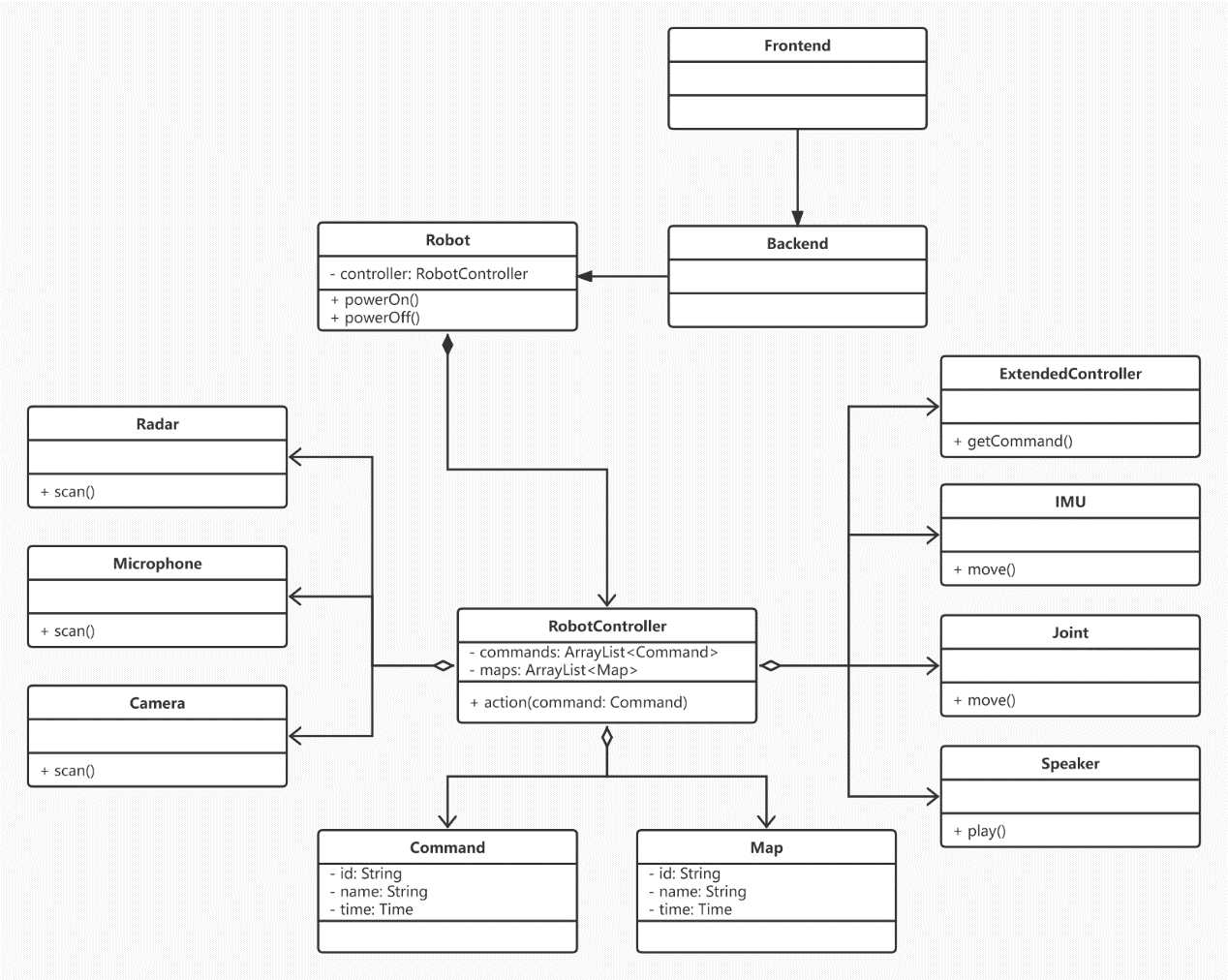
用户层负责处理用户交互与用户输入，获取相应的信息。该部分包含前端，支持用户通过GUI界面与机器人进行交互，并在交互后将相关信息传送至调度层，由调度层分配任务。

调度层负责沟通用户层与控制层，存储、处理并分发来自用户层与控制层的一切指令。调度层可以将用户存储的命令分配给对应的控制层，也可以将控制层的命令中转给下一个控制层或反馈给用户层。

控制层负责抽象的模块功能，通过接收调度层的信息协调与控制功能所需的各个硬件。控制层包括机器人运动控制模块、语音控制模块、地图控制模块等。

硬件层负责硬件的原子功能，该层的所有功能在其余层都是具有单一功能原子操作，受到控制层调用。硬件层包括激光雷达模块、运动模块、机械臂控制模块等。





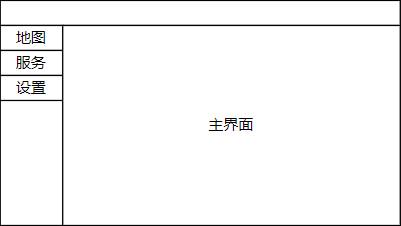
# 接口设计

## 用户界面接口

用户界面的主要作用为给用户显示控制系统和展示系统的界面，本系统用户界面主要提供建图、标注、服务和设置等功能。

### 启动界面

启动界面主要显示三项功能为查看所有地图、服务和设置。都放在侧边栏中，用户可以点击任意一个进行主界面的切换。如果未登录会提示用户进行登录。



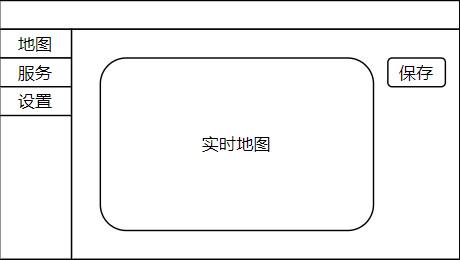
### 查看所有地图界面

此界面主要向用户展示已经建立好的所有地图的粗略信息，并且有一个加号按钮，用户可以点击此按钮进入建图界面。



### 建图界面

用户进入建图界面点击开始建图按钮后机器人会自动开始建图，界面会实时显示机器人建立的地图，建图完毕后用户点击保存按钮并输入相关信息后，地图即可自动保存。



### 标注界面

用户进入标注界面可以在地图上对航点进行标注。



接口定义如下：

表格 1 启动标注接口定义

|  |  |
| --- | --- |
| 启动标注 | |
| URL | /backend/start\_boot |
| http请求方法 | post |
| Request  Response | 用户名username |
| 密码keyword |
| 备注message |
| 错误码code |
| 内容描述 | 用户输入信息，进入标注程序 |

表格 2 选择地图接口定义

|  |  |
| --- | --- |
| 选择地图 | |
| URL | /backend/start\_mark |
| http请求方法 | post |
| Request  Response | 地图名称name |
| 地图id |
| 备注message |
| 错误码code |
| 内容描述 | 在数据库查询地图所对应的文件路径并加载地图，启动航点脚本，启动RVIZ。 |

表格 3 保存标注接口定义

|  |  |
| --- | --- |
| 保存标注 | |
| URL | /backend/save\_mark |
| http请求方法 | post |
| Response | 备注messge |
| 内容描述 | 调用保存航点脚本，保存航点信息。 |

表格 4 显示航点列表接口定义

|  |  |
| --- | --- |
| 显示航点列表 | |
| URL | /backend/list\_mark |
| http请求方法 | post |
| Response | 航点列表list |
| 内容描述 | 后端默认航点文件，解析出标注的航点名称列表反馈给前端。 |

表格 5 重命名航点接口定义

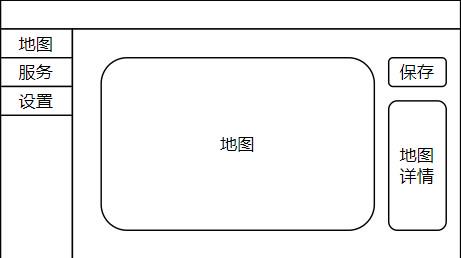
|  |  |
| --- | --- |
| 重命名航点 | |
| URL | /control/rename\_mark |
| http请求方法 | post |
| Request  Response | 地图名称name  航点路径index\_location  重命名内容rename  备注message  错误码code |
| 内容描述 | 后端修改航点文件，并将地图的航点文件保存到对应路径。 |

表格 6 结束标注接口定义

|  |  |
| --- | --- |
| 重命名航点 | |
| URL | /backend/end\_mark |
| http请求方法 | post |
| Response | 备注message |
| 内容描述 | 退出标注功能，回到主界面。 |

### 地图详情界面

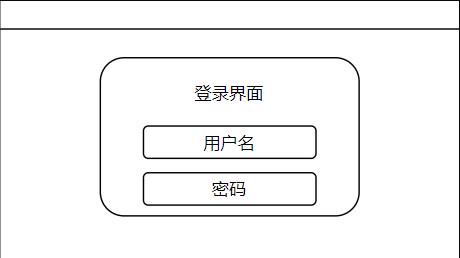
用户进入地图详情界面可以查看地图的详细细腻些，并且可以修改地图信息。



/TODO

### 登录界面

用户可以输入用户名和密码登录至主界面。



表格 1 登录接口定义

|  |  |
| --- | --- |
| 登录 | |
| URL | /backend/register |
| http请求方法 | post |
| Request  Response | 用户名username  密码keyword  备注message  错误码code |
| 内容描述 | 用户输入用户名和密码，登录系统。 |

表格 2 结束服务接口定义

|  |  |
| --- | --- |
| 结束服务 | |
| URL | /backend/end\_service |
| http请求方法 | Post |
| Response | 备注message |
| 内容描述 | 停止服务脚本，回到主界面 |

### 服务界面

用户进入此界面可以看见实时显示的机器人，并且可以语音操控机器人。



表格 1 开始服务接口定义

|  |  |
| --- | --- |
| 开始服务 | |
| URL | /backend/start\_service |
| http请求方法 | post |
| Request  Response | 地图名map\_name  备注message |
| 内容描述 | 访问数据库，加载地图，然后启动RVIZ，用户标注起始点后机器人开始服务。 |

表格 2 结束服务接口定义

|  |  |
| --- | --- |
| 结束服务 | |
| URL | /backend/end\_service |
| http请求方法 | Post |
| Response | 备注message |
| 内容描述 | 停止服务脚本，回到主界面 |

## 后端接口

后端使用GO语言的Gin框架编写，前端调用后端接口，发送给后端request，后端进行相应的处理，并做出相应的应答，返回值一般为json格式。

所有与用户相关的接口request中均不包含密码一项，因为本项目使用了token技术，详见用户接口关于token的介绍。

接口格式：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 |  | | |
| http请求方法 |  | | |
| URL |  | | |
| Request | 字段名 | 类型 | 备注 |
|  |  |  |  |
|  |  |  |  |
| Response | 字段名 | 类型 | 备注 |
|  |  |  |  |
|  |  |  |  |
| 内容描述 |  | | |

### 系统服务接口

**表1 恢复初始设置**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 恢复初始设置 | | |
| http请求方法 | GET | | |
| URL | system/reset | | |
| Request | 字段名 | 类型 | 备注 |
|  | UserId | int | 用户id |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 调用后，恢复到初始设置，清空此ID对应的记录 | | |

**表2 系统更新**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 系统更新 | | |
| http请求方法 | GET | | |
| URL | system/update | | |
| Request | 字段名 | 类型 | 备注 |
|  | UserId | int | 用户id |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 调用后，自动更新所有部分到最新版本 | | |

**表3 获取更新列表**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 获取更新列表 | | |
| http请求方法 | GET | | |
| URL | system/get-update-list | | |
| Request | 字段名 | 类型 | 备注 |
|  | UserId | int | 用户id |
| Response | 字段名 | 类型 | 备注 |
|  | NewestVersion | json | 返回此当前版本信息 |
| NowVersion | json | 返回最新版本信息 |
| 内容描述 | 调用后，返回当前和最新的版本信息 | | |

**表4 更新系统的某一部分**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 更新某一部分 | | |
| http请求方法 | POST | | |
| URL | system/update-part | | |
| Request | 字段名 | 类型 | 备注 |
|  | UserId | int | 用户id |
| Type | int | 表示需要更新的部分 |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 调用后，更新某部分到最新版本 | | |

### 用户接口

本项目使用了中间件和token，首先项目设定了一些中间件，

简单来说，Gin中间件的作用有两个：

1.Web请求到到达我们定义的HTTP请求处理方法之前，拦截请求并进行相应处理(比如：权限验证，数据过滤等)，这个可以类比为前置拦截器或前置过滤器，

2.在我们处理完成请求并响应客户端时，拦截响应并进行相应的处理(比如：添加统一响应部头或数据格式等)，这可以类型为后置拦截器或后置过滤器。

简单来说，在保证对程序员的可读性的基础上，通过这些中间件，可以保证系统权限的正确性，如只有登录之后，才能发起一个新的事务，随后为每个用户在登陆后生成一个token，这保证了短时间内的网页刷新不会需要重复登陆，同时使用redis存储token，以达到相应的优化效果。

**表1 登录**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 登录 | | |
| http请求方法 | POST | | |
| URL | user/login | | |
| Request | 字段名 | 类型 | 备注 |
|  | UserName | string | 用户名 |
|  | Password | string | 密码 |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 调用此接口，进行登录操作，为相应的用户生成token，并获取用户ID，并存入redis | | |

**表2 注销**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 注销 | | |
| http请求方法 | GET | | |
| URL | user/logout | | |
| Request | 字段名 | 类型 | 备注 |
|  | 不设置请求结构体，只需要获取redis中存储的信息，即可定位到相应的用户 | | |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 调用此接口，进行注销操作 | | |

**表3 忘记密码**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 忘记密码 | | |
| http请求方法 | POST | | |
| URL | user/forget-password | | |
| Request | 字段名 | 类型 | 备注 |
|  | UserName | string | 忘记密码的用户名 |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 通过调用接口，为相应的用户的邮件发送重置密码的验证码，并在redis中添加相应信息，如果出现错误，如用户未设定邮箱等情况，返回相应报错 | | |

**表4 重置密码**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 充值密码 | | |
| http请求方法 | POST | | |
| URL | user/reset-password | | |
| Request | 字段名 | 类型 | 备注 |
|  | Token | string | 通过邮箱收到的验证码 |
| NewPassword | string | 新密码 |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 通过得到的验证码来重置密码，根据redis存储的信息可以得到验证码对应的用户，修改密码后，删除redis中的相应信息 | | |

**表5 查询用户状态**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 查询用户状态 | | |
| http请求方法 | GET | | |
| URL | user/get-user-state | | |
| Request | 字段名 | 类型 | 备注 |
|  | 不设置请求结构体，只需要获取redis中存储的信息，即可定位到相应的用户 | | |
| Response | 字段名 | 类型 | 备注 |
|  | UserId | int | 用户id |
|  | State | int | 表示用户状态   1. 正常登录 2. 正在调用某一事务 3. 用户异常 |
| 内容描述 | 获取用户当前的状态，主要判断用户当前是否在调用某一事务 | | |

### 地图接口

**表1 建立地图**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 建立地图 | | |
| http请求方法 | GET | | |
| URL | map/create-map | | |
| Request | 字段名 | 类型 | 备注 |
|  | 不设置请求结构体，只需要获取redis中存储的信息，即可定位到相应的用户 | | |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 前端发送请求，后端调用机器人相应的API，建立地图，同时在数据库中添加地图信息，并反馈给前端，出现错误的时候，以文本形式把错误反馈给前端 | | |

**表2 查询地图状态**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 查询地图信息 | | |
| http请求方法 | GET | | |
| URL | map/get-map-data | | |
| Request | 字段名 | 类型 | 备注 |
|  | MapId | int | 需要查询的地图id |
| Response | 字段名 | 类型 | 备注 |
|  | CreateAt | string | 地图创建时间 |
| UpdateAt | string | 上次修改时间 |
| Path | string | 地图信息存储路径 |
| Cnt | int | 地图使用次数 |
| 内容描述 | 查询地图的相应状态，若MapId不属于当前的UserId | | |

**表3 地图展示**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 地图展示 | | |
| http请求方法 | POST | | |
| URL | map/open-map | | |
| Request | 字段名 | 类型 | 备注 |
|  | MapId | int | 要打开的地图ID |
| Response | 字段名 | 类型 | 备注 |
|  | Image | base64数据流 | 地图的图片信息 |
| NodeList | 数组 | 此地图对应的节点信息 |
| 内容描述 | 通过MapId来打开相应的map，后端通过数据库存储的map的图片的url来返回给前端图片 | | |

**表4 修改地图**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 修改地图 | | |
| http请求方法 | POST | | |
| URL | map/modify-map | | |
| Request | 字段名 | 类型 | 备注 |
|  | MapId | int | 需要修改的地图ID |
|  | NewPath | string | 新地图的路径 |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 调用接口，来修改地图，并保存新的地图信息到数据库 | | |

### 功能接口

**表1 取物接口**

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 取物 | | |
| http请求方法 | POST | | |
| URL | func/get-goods | | |
| Request | 字段名 | 类型 | 备注 |
|  | GoodName | string | 物品名称 |
| Response | 字段名 | 类型 | 备注 |
|  | 不设置返回值，通过Gin中http包内已经定义好的如http.StatusBadRequest和https.StatusOK等端口来返回相应的文本信息，来判断接口的调用结果 | | |
| 内容描述 | 根据标注信息，获取到物品所在的位置，调用机器人的接口，规划路径，并进行物品抓取，运送到用户处 | | |

### 状态码定义

通过定义后端的状态码，给前端准确的信息反馈。详细的状态反馈在对应的message里也会体现。

表格 15 后端状态码定义表

|  |  |
| --- | --- |
| 200 | 成功 |
| 404 | 其他异常 |
| 1 | 保存地图异常 |
| 2 | 删除地图异常 |
| 3 | 地图不存在异常 |
| 4 | 保存航点异常 |
| 5 | 删除航点异常 |
| 6 | 修改航点异常 |
| 7 | 航点不存在异常 |

## 调度层接口

调度层接口是系统后端与机器人的调度器之间的接口。后端决定执行某个任务，于是向调度器发出指令要求其调度相关硬件资源予以执行。这里规定了调度器开放给后端的接口，允许后端向调度器发送的指令。

调度器开放如下接口：

表 <编号> 调度器接口定义表（TODO：编号）

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名 | 订阅 | 发布 | 说明 |
| mapEstablish | 无 | 无 | 启动手动建图工作的指令 |
| navigate | 无 | 无 | 导航指令 |
| grab | 无 | 无 | 抓取物品指令 |
| voiceService | 无 | 无 | 开启语音服务指令，包括语音输入和语音反馈 |

## 控制层接口

### 运动控制

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /get\_object\_location | 需要订阅command、摄像头、相关主题 | 无 | 在收到抓取物品要求时，获取物品所在位置 |
| /fetch\_object | 需要订阅command、机械臂模块、IMU模块 | 给机械臂模块发送指令 | 抓取指定位置的物品 |
| /move | 需要订阅调度器模块 | /cmd\_move | 将机器人向指定目标点移动 |

### 语音控制

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /establish\_voice\_command | 需要订阅麦克风相关主题 | 需要对调度层发布信息 | 向调度层发布指令信息，保存在 Command 类里 |

### 地图控制

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /build\_map\_auto | 需要订阅传感器相关主题 | 无 | 自动构建地图 |
| /build\_map\_manual | 需要订阅传感器相关主题 | 无 | 手动构建地图 |
| /store\_current\_map | 无 | 无 | 保存当前地图状态 |

## 硬件层接口

硬件层接口是操作硬件模块的层次。它给控制层提供操作硬件模块的接口，而在其内部向硬件模块发送对应指令，以操作硬件完成工作。

硬件层的接口主要由ROS和机器人功能包提供，主要内容如下：

### 激光雷达模块

激光雷达向机器人四周360°扫描，得到的数据可以通过/scan获取。接口如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /scan | 需要使用雷达的模块 | /rplidarNode | 其从/scan获取数据，存储于一个LaserScan类型内供上级模块使用。 |

### 运动模块

速度控制包由被发布到主题cmd\_vel中，启智ROS机器人的核心节点wpb\_home\_core会从这个主题获取速度消息包，对机器人底盘硬件进行速度控制。

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /cmd\_vel | /wpb\_home\_core | 无 | 该主题向/wpb\_home\_core发布速度消息包，控制机器人运动 |

### 机械臂控制模块

在主题/wpb\_home/mani\_ctrl里广播sensor\_msgs::JointState 类型的数据。机器人的核心节点会从这个主题获取机械臂控制量，然后发送到硬件去执行。

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /wpb\_home/mani\_ctrl | /wpb\_home\_core | 无 | 该主题向/wpb\_home\_core发布机械臂控制消息包，控制机器人机械臂运动 |

### Kinect2 三维点云模块

Kinect2的ROS节点在/kinect2/sd/points发布三维点云，订阅这个主题即可收到三维点云数据。

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /kinect2/sd/points | 需要使用点云的节点 | 无 | 该主题发布三维点云消息包 |

### 摄像头模块

摄像头节点在/camera\_image发布实时收到的图片，订阅这个主题即可收到摄像头捕获的图片。

|  |  |  |  |
| --- | --- | --- | --- |
| 接口名称 | 订阅 | 发布 | 说明 |
| /camera\_image | 无 | 无 | 该主题发布摄像头图片消息包 |

# 详细设计

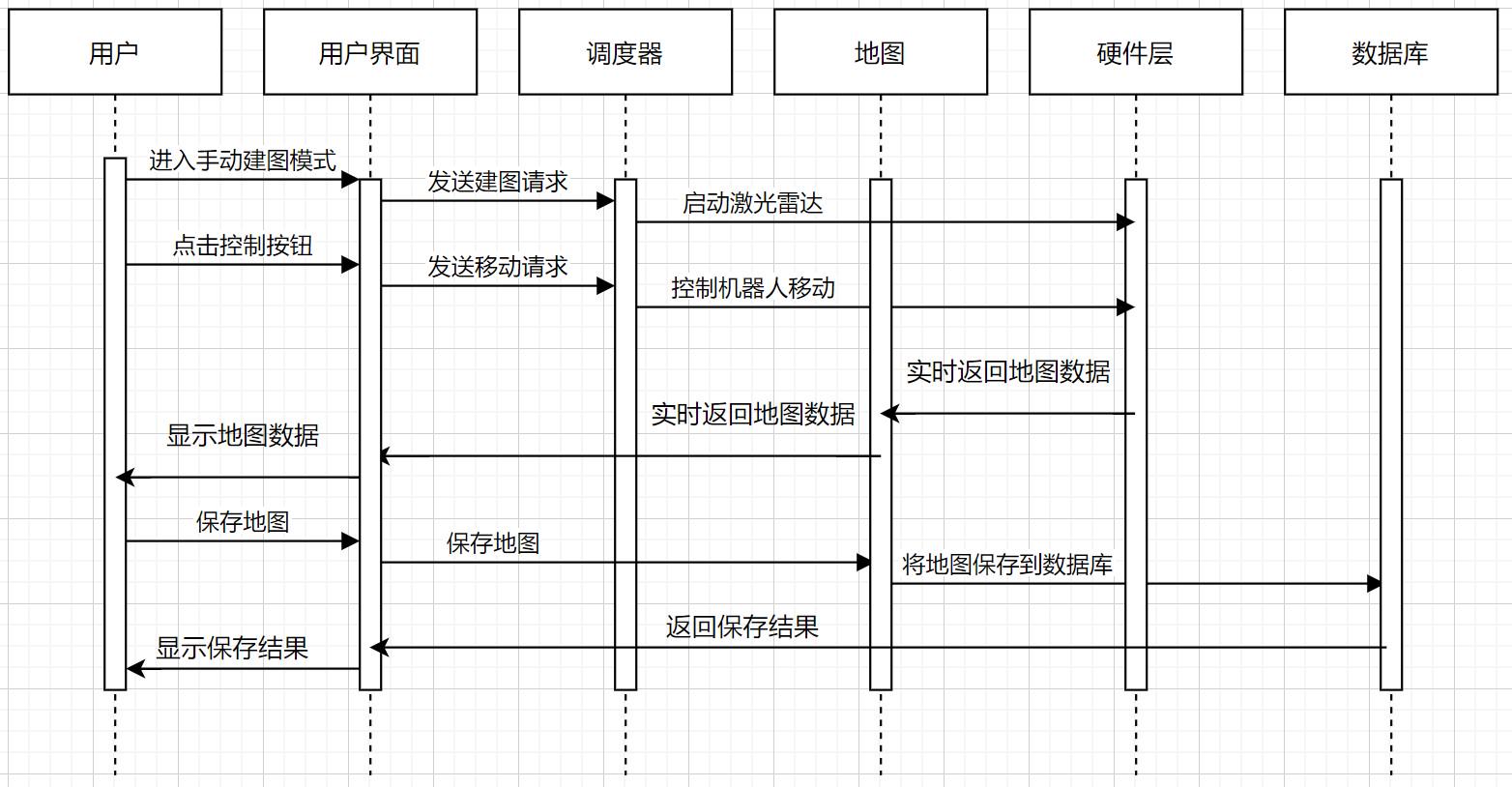
【对于系统的每个关键模块（构件），用小节逐个给出一个模块（构件）的详细设计方案。建议采用详细的*类图、状态图、活动图、流程图*等。】

我们根据接口的规范进行了更详细的设计规划，对于更多设计细节我们会在后续的实现后进行迭代补充。

## 基础模块设计

### 手动建图

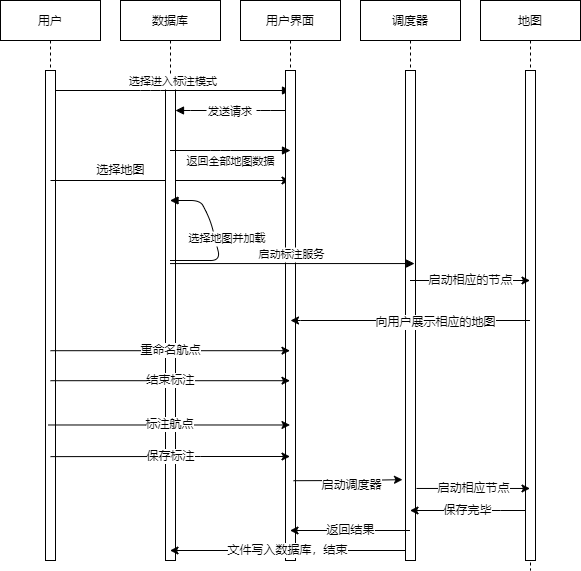
用户首先需要打开手动建图界面，点击开始按钮发起手动建图，将命令传给调度器，调度器给硬件层传递MapEstablish()命令，开始建图，在此期间，用户可以点击控制按钮使得前端给调度器发送移动请求，最后调度器控制硬件层的移动。用户建图期间，前端可以实时显示地图。建图完毕后，用户可以选择是否保存已经建立好的地图，后端最后返回给前端是否保存成功并显示给用户。



### 地图标注

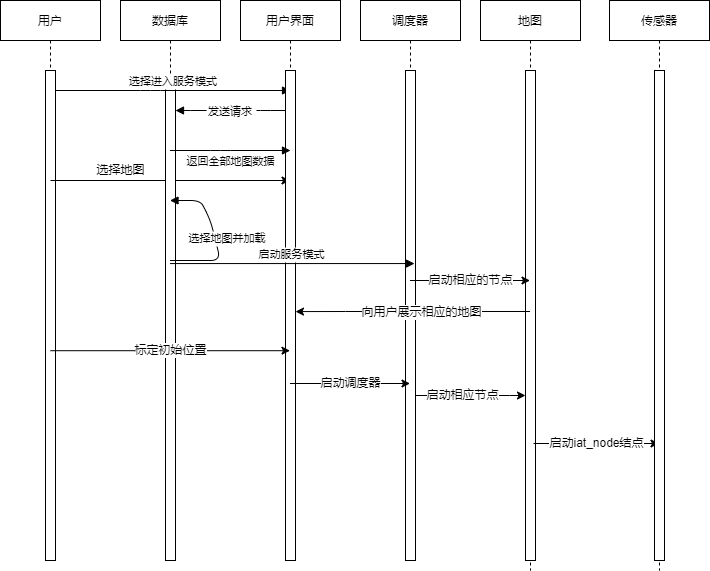
地图标注接口主要由MapTools工具实现。MapTools工具能够在Rviz下打开，标记导航点并保存导航点的数据。

用户在用户界面选择标注模式，进入到地图选择界面，后端会从数据库中找到对应的地图，将地图拷贝到map文件夹下，然后通知总调度器启动标注服务。接着调度器会选择的地图将会被直接呈现在用户界面上，用户标注导航点后，在界面上点击保存，将通知总调度器调用地图的save\_mark()函数，save\_mark()函数保存导航点数据，同时用户界面将导航点文件拷贝到指定位置，并将文件信息写入数据库，然后通知用户标注完成，结束任务。



### 启动服务

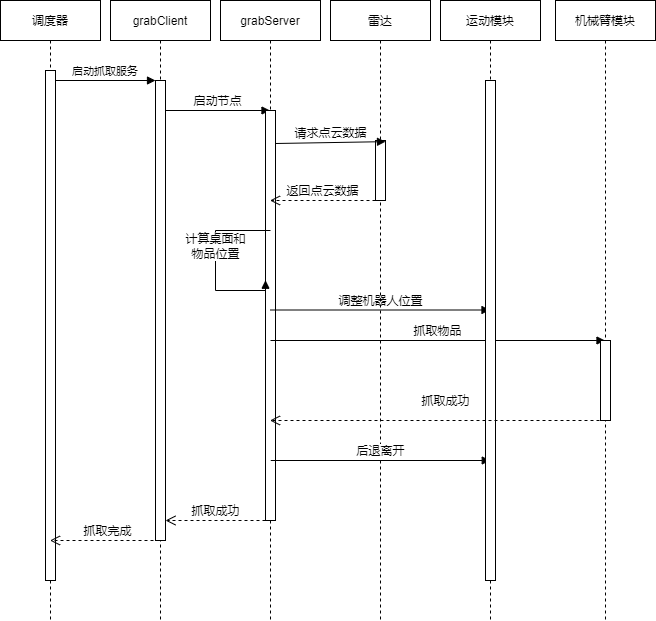
用户首先需要启动服务。在主界面选择服务模式后，进入地图选择界面，选好地图后，用户界面将根据选定地图的文件路径将文件载入指定文件夹，然后启动调度器，运行start\_service()函数，启动任务管理系统，并调用相应函数向用户展示地图。接下来调用传感器的函数启动相应节点，做好随时接受语音指令的准备。在此期间，用户完成对机器人初始位置进行标定，协助机器人确定位置。



## 功能模块设计

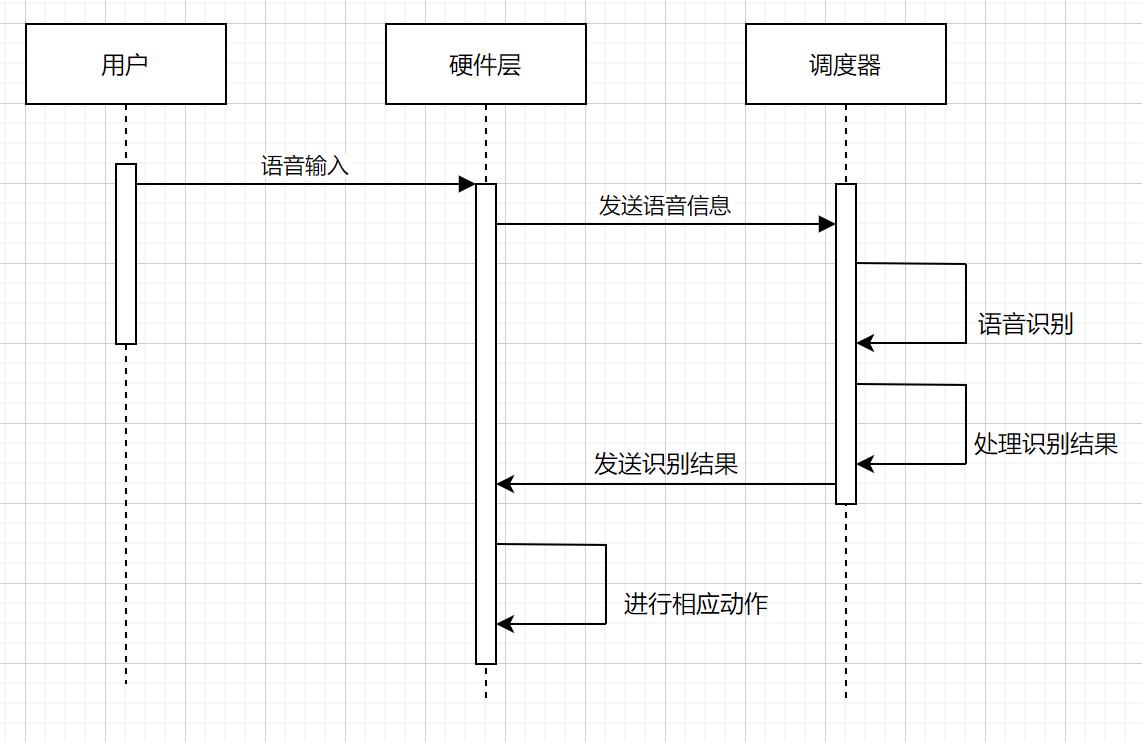
### 物品抓取

调度器要求执行物品抓取。首先，调度器启动grabClient节点，作为抓取服务结果的接收者。然后启动grabServer节点，作为抓取任务的执行者。grabServer向摄像头请求点云数据，得到摄像头返回的数据后，grabServer计算得到桌面和物品的位置，然后向运动模块发送指令以调整机器人到合适的位置。之后，grabServer向机械臂控制模块发送事件控制机械臂抓取物品。机械臂向grabServer反馈抓取完成后，grabServer向控制模块发送事件让机器人后退离开桌面，然后向grabClient节点发布抓取成功的消息。grabClient收到抓取成功消息后，返回通知调度器抓取结束。



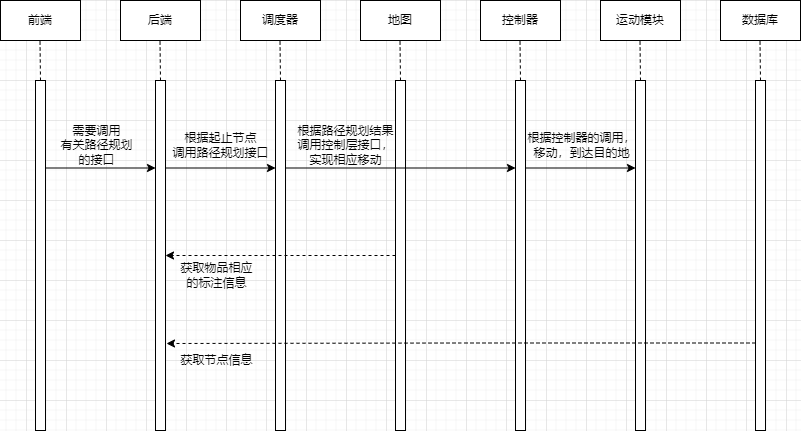
### 语音识别

用户通过给麦克风发送语音运行语音识别，硬件层会向调度器发送语音信息，调度器会处理语音信息并进行识别，识别成功后会调用硬件层相应的功能



### 路径规划

当用户发起物品取放的事务的时候，会调用相应的后端API，根据需要取放的物品，从数据库中获取到相应航点的位置，确定起点和终点后，调用通过调度器调用机器人的global\_planner包进行路径规划，通过控制器调用硬件来实现移动

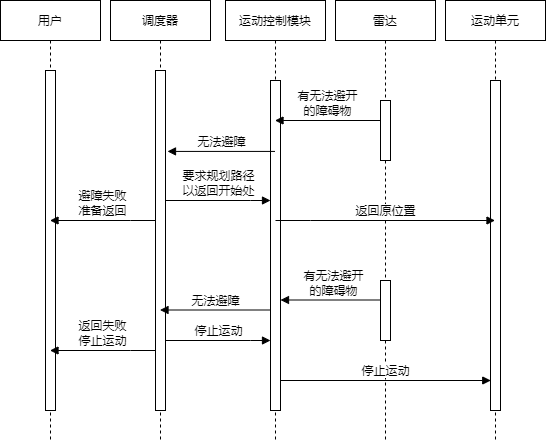


## 错误处理设计

本部分对机器人运行中可能出现的错误及其解决方案进行了设计，并在下文给出详细叙述与对应的时序图。

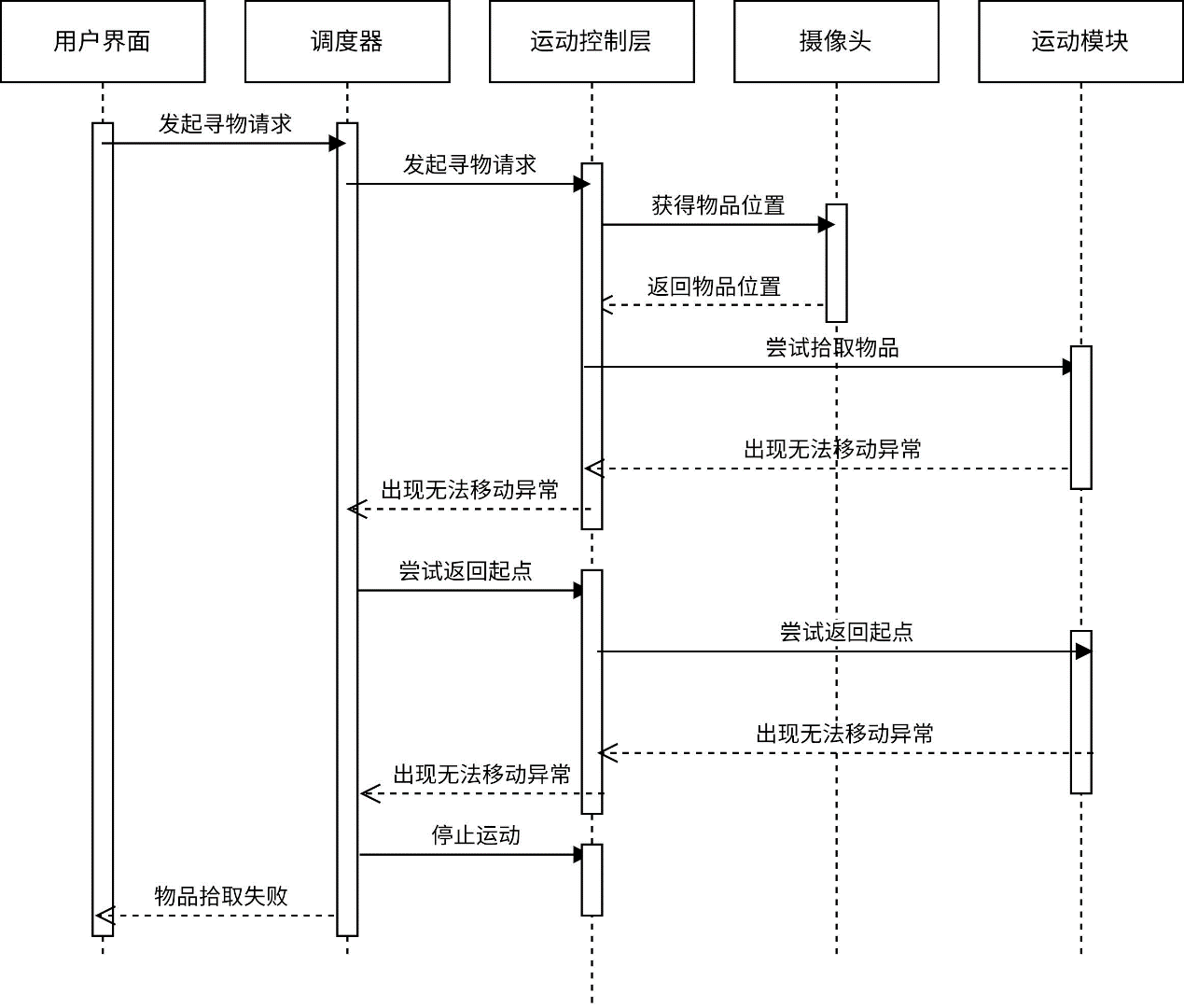
### 路径规划失败

机器人在路径移动时，若检测到障碍物过多或者因其他因素导致的路径规划算法失效，使机器人无法到达指定位置时，运动控制模块将会向调度器返回终止命令。然后调度器会要求运动控制模块尝试回到指令开始时所在的位置。调度器向用户界面返回相应错误提示，表示机器人目前遇到障碍。若在回到原地时同样无法到达指定位置，则调度器通知运动模块停止运动，并向用户界面返回提示表示机器人无法回到原地。



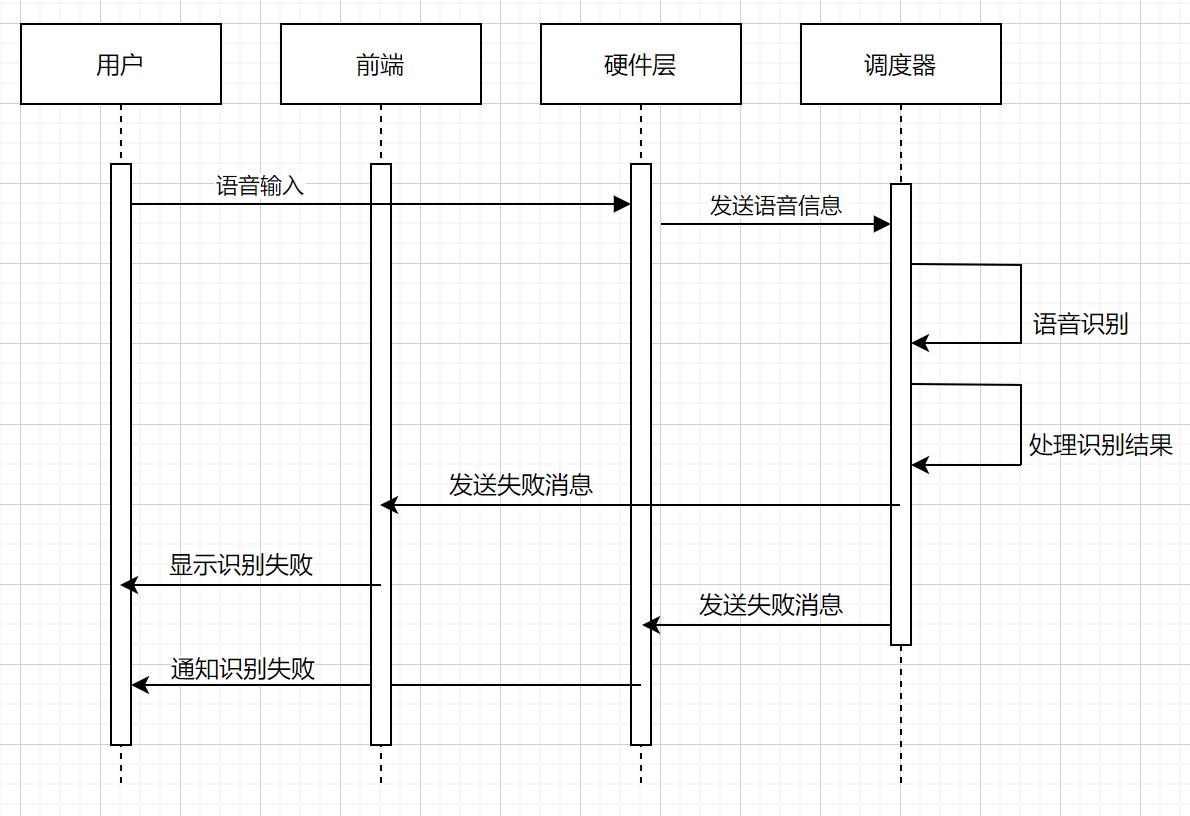
### 物品识别失败

在机器人抓取物品时，如果图像识别算法无法识别物品，或没有识别到相应的物品时，运动控制模块将会向运动调度器返回终止命令，同时尝试回到接受指令时所在的位置；调度器向用户界面返回相应错误提示，表示机器人目前未识别到物品。若在回到原地时同样无法到达指定位置，则调度器通知运动模块停止运动，并向用户界面返回提示表示机器人无法回到原地。



### 语音识别失败

当用户通过语音唤醒机器人，并给机器人下达指令的时候，机器人会做出相应指令。如果语音控制模块无法识别语音对应的指令，机器人将不会进行任何指令操作。同时，语音控制模块将会向调度器反馈识别失败信息，调度器向用户的UI界面提示识别失败，向语音控制模块发送识别失败消息并控制麦克风模块播放失败提示。



# 运行与开发环境

## 运行环境

本系统运行的硬件环境为:

* 启智ROS机器人；
* 启智ROS机器人配套机械臂；
* 互联网连接。

本系统运行的软件环境为：

* Ubuntu 18.04 LTS；
* ROS Melodic Morenia；
* 启智ROS机器人相关基础包和驱动；
* Gazebo仿真环境。

在开发阶段，系统主要在仿真环境下运行；在完成仿真开发迭代后，会将系统在真实的机器人环境下进行运行和测试。

## 开发环境

本系统开发所需的硬件环境为：

* 开发用计算机；
* 启智ROS机器人；
* 启智ROS机器人配套机械臂；
* 互联网连接。

本系统开发所需的软件环境为：

* Ubuntu 18.04 LTS；
* ROS Melodic Morenia；
* 启智ROS机器人相关基础包和驱动；
* Rviz可视化平台与gazebo仿真环境；
* RoboWare Studio集成开发环境。

# 需求可追踪性说明

【说明系统设计方案如何满足系统需求规格说明书要求。建议给出SRS文档中功能/非功能需求——SDD文档体系结构设计这两部分内容的对应关系。】

|  |  |  |
| --- | --- | --- |
| 标号 | SRS文档内容 | SDS文档对应内容 |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |