**北航2019软件工程简易机器人**

**软件设计说明书**

**SDD104**

**2.0版**

|  |  |  |
| --- | --- | --- |
| 小组名称 | IMMORTAL | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 16061102 | 刘逸 | 初稿编写的主要工作，完成总体设计和目标检测抓取相关设计 |
| 16182016 | 马力 | 对初稿进行补充，完成建图、导航寻路模块设计 |
| 76066002 | 周美廷 | 对初稿进行补充，完成数据库相关设计 |
| 16061189 | 于金泽 | 完成体系结构设计和语音相关模块设计，第二稿审核和整理 |
|  | 张金源 | 完成建图、导航寻路模块设计 |

分工说明

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 | 2019.4.6 | 刘逸 | 马力，周美廷 | 初步完成范围、需求概述、运行与开发环境 |
| 1.5 | 2019.4.22 | 刘逸，马力，周美廷 | 马力，周美廷 | 完成文档初版的编写 |
| 2.0 | 2019.5.5 | 刘逸，马力，周美廷，于金泽，张金源 | 于金泽 | 对初稿中存在的问题进行改正，明确初稿中较为模糊的设计，对总体结构和详细设计进行修正并重新完成大量图片的绘制 |
|  |  |  |  |  |

目录

[1. 范围 1](#_Toc7958441)

[1.1 项目概述 1](#_Toc7958442)

[1.2 文档概述 1](#_Toc7958443)

[1.3 术语和缩略词 2](#_Toc7958444)

[1.4 引用文档 2](#_Toc7958445)

[2. 需求概述 3](#_Toc7958446)

[2.1 系统用例模型及分析 4](#_Toc7958447)

[2.1.1 启动/关闭机器人 4](#_Toc7958448)

[2.1.2 建立地图 6](#_Toc7958449)

[2.1.3 记录商品名/位置 7](#_Toc7958450)

[2.1.4 物品抓取 8](#_Toc7958451)

[2.2 数据需求 10](#_Toc7958452)

[2.3 非功能需求 10](#_Toc7958453)

[2.4 用户界面需求 12](#_Toc7958454)

[3. 体系结构设计 13](#_Toc7958455)

[3.1 总体结构 13](#_Toc7958456)

[3.1.1 软件体系结构 14](#_Toc7958457)

[3.1.2 硬件体系结构 15](#_Toc7958458)

[3.1.3 技术体系结构 16](#_Toc7958459)

[3.2 关键问题及解决方案 16](#_Toc7958460)

[3.2.1 SLAM建图 16](#_Toc7958461)

[3.2.2 导航和障碍物识别与躲避 16](#_Toc7958462)

[3.2.3 语音识别和合成 17](#_Toc7958463)

[3.2.4 物品检测识别和抓取 17](#_Toc7958464)

[4. 接口设计 18](#_Toc7958465)

[4.1 系统用户界面 18](#_Toc7958466)

[4.2 硬件接口 19](#_Toc7958467)

[4.2.1 手柄操控 19](#_Toc7958468)

[4.2.2 激光雷达 19](#_Toc7958469)

[4.2.3 SLAM建图 19](#_Toc7958470)

[4.2.4 Navigation 导航 20](#_Toc7958471)

[4.3 软件接口 20](#_Toc7958472)

[4.3.1 运动控制 20](#_Toc7958473)

[4.3.2 4.3.2导航 20](#_Toc7958474)

[4.3.3 内部接口 21](#_Toc7958475)

[5. 数据库设计 21](#_Toc7958476)

[5.1 用户信息 22](#_Toc7958477)

[5.2 商品信息 22](#_Toc7958478)

[5.3 关键词 22](#_Toc7958479)

[5.4 地图信息 22](#_Toc7958480)

[5.5 日志 22](#_Toc7958481)

[6. 详细设计 23](#_Toc7958482)

[6.1 运动控制 23](#_Toc7958483)

[6.2 SLAM建图 24](#_Toc7958484)

[6.3 Navigation导航 25](#_Toc7958485)

[6.4 避障运动 27](#_Toc7958486)

[6.5 物品抓取 29](#_Toc7958487)

[6.6 语音识别与合成 30](#_Toc7958488)

[6.6.1 硬件基础 31](#_Toc7958489)

[7. 运行与开发环境 32](#_Toc7958490)

[7.1 运行环境 32](#_Toc7958491)

[7.1.1 运行硬件： 32](#_Toc7958492)

[7.1.2 运行软件： 32](#_Toc7958493)

[7.2 开发环境 33](#_Toc7958494)

[8. 需求可追踪性说明 33](#_Toc7958495)

[8.1 功能需求 33](#_Toc7958496)

[8.1.1 到达目的地 33](#_Toc7958497)

[8.1.2 避障运动 33](#_Toc7958498)

[8.1.3 规划路径 34](#_Toc7958499)

[8.1.4 建立地图 34](#_Toc7958500)

[8.1.5 机器人操作 34](#_Toc7958501)

[8.1.6 打开/关闭机器人 34](#_Toc7958502)

[8.1.7 分析指令 34](#_Toc7958503)

[8.1.8 发出指令 35](#_Toc7958504)

[8.2 非功能需求 35](#_Toc7958505)

[8.2.1 性能 35](#_Toc7958506)

[8.2.2 安全性 35](#_Toc7958507)

[8.2.3 可扩展性 35](#_Toc7958508)

[8.2.4 可靠性 35](#_Toc7958509)

# 范围

## 项目概述

为了解放人力劳动，我们便需要制作一些机器人来帮助我们，完成一些简单但又重复的工作。我们要制造的这一款简单机器人，便是可以达到这一目的。

制作简单的机器人，是在偏向学术层的实践领域上实现现代的计算机技术与机械行业的结合。它的需求分为功能需求和非功能需求。

功能需求方面，实现控制机器人完成基本避障运动，路径规划甚至于目标检测抓取。通过简单的按键操作或者页面交互操作，可以完成机械装置的固定路径巡线，抓取目标物品，在特定的场景下，甚至可以完全机械装置的工作自动化。

非功能需求方面，需要有一定的可靠性，可以在一定时间内完成指定的路径规划、目标抓取任务；有一定的可修改性，当完成设计后方便添加更多的功能，或者在设计期间方便修改之前的设计。环境需求，嵌入式的操作系统为Linux系统，并且支持ROS开发。

这款机器人可以运用到物流行业，进行包裹分类并将包裹送到指定位置，从而节省大量人力物力，并且效率也会有所提升。它还可以应用于流水线加工行业，抓取目标物品。

## 文档概述

本文档是对简易机器人的设计说明，是简易机器人软件设计说明文档。

在第一章介绍范围；第二章进行需求概述；第三章介绍体系结构设计；第四章介绍接口设计；第五章进行数据库说明；第六章介绍详细设计；第七章介绍运行与开发环境；第八章进行需求可追踪性说明。

## 术语和缩略词

表格 1 缩略词/全称对应表

|  |  |
| --- | --- |
| 缩略词 | 全称 |
| ROS | Robot Operating System |
| SRS | System Requirements Specification |
| SDP | System Development Plan |
| ER | Entity Relationship Diagram |
| USB-HUB | Universal SerialBus-Hub |
| SLAM | simultaneous localization and mapping |
| SDD | Software Design Document |
| Kinect2 | 视觉传感器装备 |
| Ubutu | 是一个以[桌面](https://baike.baidu.com/item/%E6%A1%8C%E9%9D%A2)应用为主的开源GNU/Linux操作系统 |

## 引用文档

(1).启智ROS机器人开发手册v1.1.0

(2).SDP104-机器人开发计划v1.0.3

# 需求概述

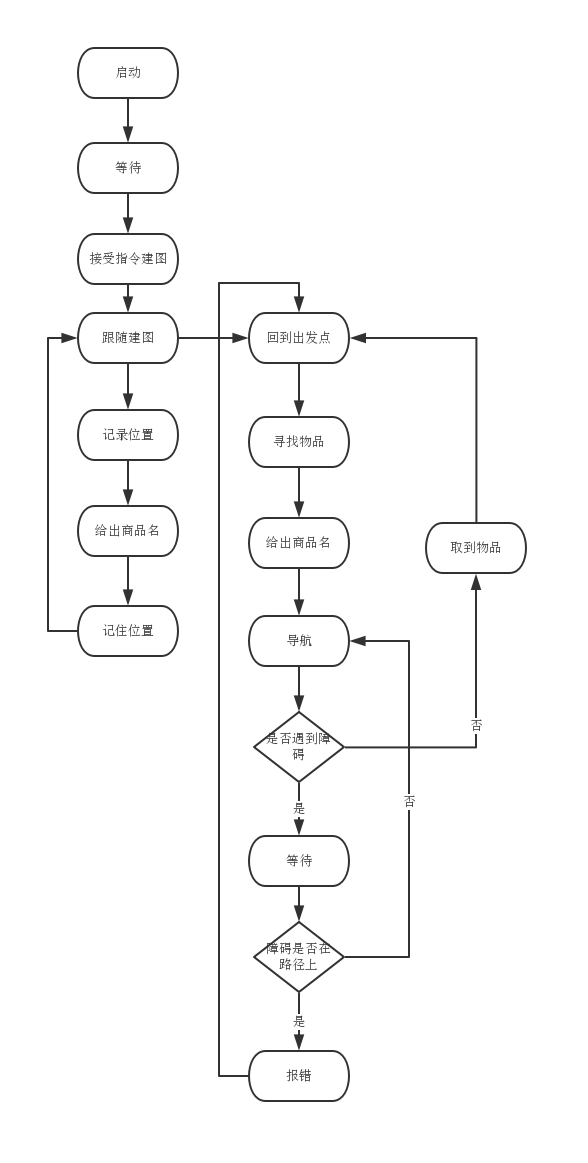


图 1 业务流程图

机器人主要可以做到取回特点地点的物品。

机器人要进行跟随建图，并在建图过程中由工作人员给出商品名，机器人记录商品名及位置。建图结束后，机器人回到出发点。之后，用户发出寻找物品的指令，机器人进行导航移动。如果在途中突然遇到障碍物，进行判断，而后根据情况进行报错或取回物品回到出发点。

超市人员可以通过机器人机载电脑或远程控制的方式安装或卸载系统、启动或关闭机器人；通过UI查看实时地图界面，掌握机器人场景建模情况。机器人将对成功获取到的语音指令及时做出相应，在到达取物地点和抓取物品成功后发出提示，并对错误和异常情况做出判断、处理和提示。

## 系统用例模型及分析

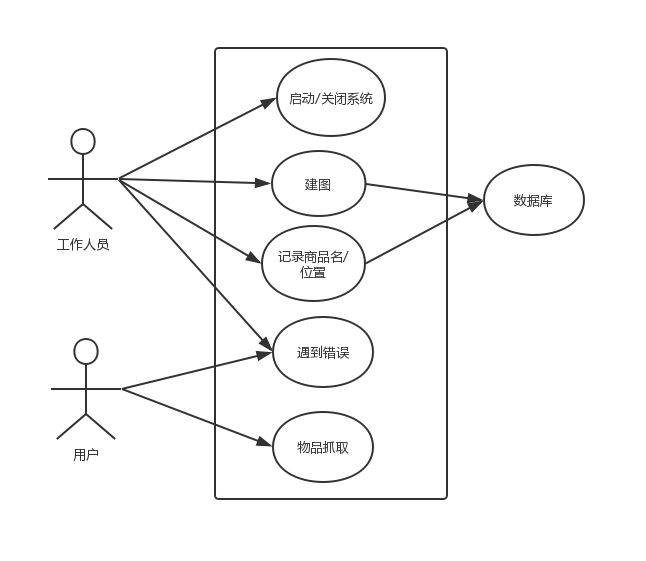


图 2 用例图

### 启动/关闭机器人

表格 2 收集数据用例

|  |  |
| --- | --- |
| **用例名** | 启动/关闭机器人 |
| **简要描述** | 无 |
| **参与者** | 工作人员 |
| **涉众** | 无 |
| **相关用例** | 无 |
| **前置条件** | 机载电脑已安装本系统；超市工作人员了解规范启动步骤；超市场地平整干燥/  机载电脑已安装本系统；超市工作人员了解规范关闭步骤；机器人处于待命状态 |
| **后置条件** | 无 |
| **基本事件流**  1. 放置机器人至出发点；  2. 通过usb接口连接机载电脑和机器人运动控件；  3. 启动机载电脑并打开机器人总开关；  4. 点击机器人系统图标，启动机器人系统；  5. 机载电脑显示系统交互界面，机器人发出语音“Start”提示；  /  1. 关闭机器人系统界面；  2. 解除机载电脑和机器人运动控件的usb连接；  3. 关闭机载电脑和机器人总开关； | |
| **备选事件流**  无 | |
| **补充约束**  机器人传感器不能被阻挡，可以正常工作 | |
| **待解决问题**  无 | |
| **相关图**  无 | |

### 建立地图

表格 3 建立地图用例

|  |  |
| --- | --- |
| **用例名** | 建立地图 |
| **简要描述** | 中枢处理建立地图 |
| **参与者** | 工作人员 |
| **涉众** | 无 |
| **相关用例** | 收集数据 |
| **前置条件** | 传感器已经成功地收集到了数据 |
| **后置条件** | 建立后的地图可以成功存储下来为了路径规划服务 |
| **基本事件流**  ①中枢通过信息建立地图  ②将地图以文件的形式存储下来 | |
| **备选事件流**  无 | |
| **补充约束**  地图可以以一定形式存储下来 | |
| **待解决问题**  无 | |
| **相关图**  图2 | |

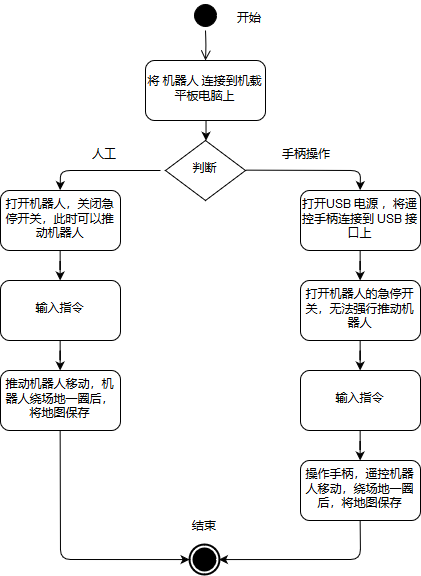


图 2 建立地图活动图

### 记录商品名/位置

表格 4 避障运动用例

|  |  |
| --- | --- |
| **用例名** | 记录商品名/位置 |
| **简要描述** | 无 |
| **参与者** | 工作人员 |
| **涉众** | 无 |
| **相关用例** | 收集数据 |
| **前置条件** | 机器人系统已正常启动；超市工作人员熟悉超市地形，发音标准；超市场地平整干燥；指定物品名称已录入系统数据库 |
| **后置条件** | 无 |
| **基本事件流**  1. 工作人员带领机器人来到指定物品所在货架前；  2 工作人员对准麦克风说出关键词“memorize+目标物”；  3. 机器人收到指令，记录物品位置并在地图上标注；  4. 机器人发出语音提示：“I have remembered the location of+目标物”；  5. 工作人员通过机载电脑界面查看物品位置记录情况；  6. 重复1-5步，直至所有物品位置记录完毕；  7. 超市工作人员带领机器人回到超市入口； | |
| **备选事件流**  无 | |
| **补充约束**  无 | |
| **待解决问题**  无 | |
| **相关图**  无 | |

### 物品抓取

表格 5 路径规划用例

|  |  |
| --- | --- |
| **用例名** | 物品抓取 |
| **简要描述** | 命令机器人到对应货架取回指定物品 |
| **参与者** | 用户 |
| **涉众** | 无 |
| **相关用例** | 记录商品名/位置 |
| **前置条件** | 已经成功记录商品名/位置 |
| **后置条件** | 无 |
| **基本事件流**  1. 顾客对准机器人麦克风说出目标物品关键词；  2. 机器人收到指令，发出语音提示：“ready to get”；  3. 机器人规划前往指定物品所在货架的路径；  4. 机器人开始移动；  5. 机器人到达指定物品所在货架；  6. 机器人发出语音提示：“ready to pick”；  7. 机器人摄像头识别物品并校准位置；  8. 机械臂启动，并开始抓取目标物品；  9. 目标物品抓取成功，机器人发出语音提示：“pick”；  10. 机器人规划前往顾客所在地的路径；  11. 机器人开始返回顾客所在地；  12. 机器人到达顾客所在地，发出语音提示：“I will pass you+目标物”；  13. 机械臂松开，将物品移交给顾客； | |
| **备选事件流**  无 | |
| **补充约束**  无 | |
| **待解决问题**  无 | |
| **相关图**  无 | |

## 数据需求

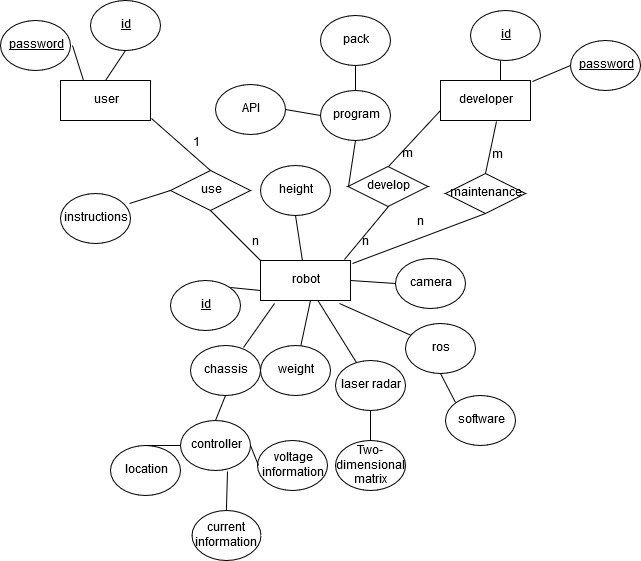


图 4 ER图

## 非功能需求

**性能**：

1. 机器人分析用户输入的指令时间在10s内完成。
2. 机器人在绕场一周之后就可以建立地图模型。
3. 在实验室环境下，确定起点与终点后，路径规划应当在100s之内完成。如果室内环境比较复杂，路径规划应当在500s之内完成。
4. 完成路径规划后，机器人应当在5s内从静止加速至0.1m/s，从起点到终点过程中的平均移动速度应当在0.1m/s到0.2m/s之间。
5. 机器人移动至目标地点后，机器人所在位置与目标地点实际位置之间的距离应当小于0.5m。
6. 如果机器人激光雷达的最小探测范围内突然出现障碍物，机器人应当在1s内减速至停止；

**安全性**：

**环境安全：**启智 ROS 机器人是室内机器人，在此环境之外运行可能会损坏机器人。工作平面需要

能够承载不小于 40kg 的重量。建议使用 商用地毯、瓷砖等材质。启智

ROS 机器人原则上在水平平面上工作， 坡道倾斜度过大可能导致倾覆。启智 ROS 机器人不具备防水功能。启智 ROS 机器人使用中务必远离明火和其他热源。

**人员安全：**启智ROS机器人整机重量约为30kg，若在运行时速度较快，与人员或室内物体相撞后可能造成人员受伤或物体损坏，应当保证任何情况下机器人的移动速率小于0.2 m/s；

**可扩展性：**

**硬件扩展性：**启智 ROS 机器人内部包含1个 USB-HUB、1个启智电池模块、1个电源控制板、1个启智控制器、3个启智伺服电机模块。位于机器人躯干部分外挂的计算机运行 ROS 操作系统， 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（用于用户自行连接设备，例如 U 盘、控制手柄）。开关面板上保留2个USB 接口供用户使用。

**软件扩展性：**对象的方法不应超过300行，新增功能不应大幅修改原有代码。

**易用性：**

1. 经过说明，用户可以在三分钟内的完成起点及目的地的设定。
2. 用户不用输入代码，直接在图形化界面操作就可以。
3. 机器人配套手柄，用户可以使用手柄控制机器人移动。

**可靠性：**

1. 在运行过程中，任何情况下都不应当与室内物体或人员发生碰撞。
2. 如果机器人路径规划的起点与终点如果没有障碍物，即可到达。
3. 机器人出现故障的概率应小于等于0.1%。

## 用户界面需求

用户看到的界面应该简洁明了，并且操作应当将可能的减少优化，使用户可以更加方便的使用机器人。用户界面应当现实机器人所在的地图以及机器人的所在，其中不论机器人移动到哪个点，用户界面上的显示也应当同步更新。另外，用户界面应当显示并实现几个基本功能如：保存地图，选择或输入指令使机器人运动，退出模拟系统。

# 体系结构设计

## 总体结构

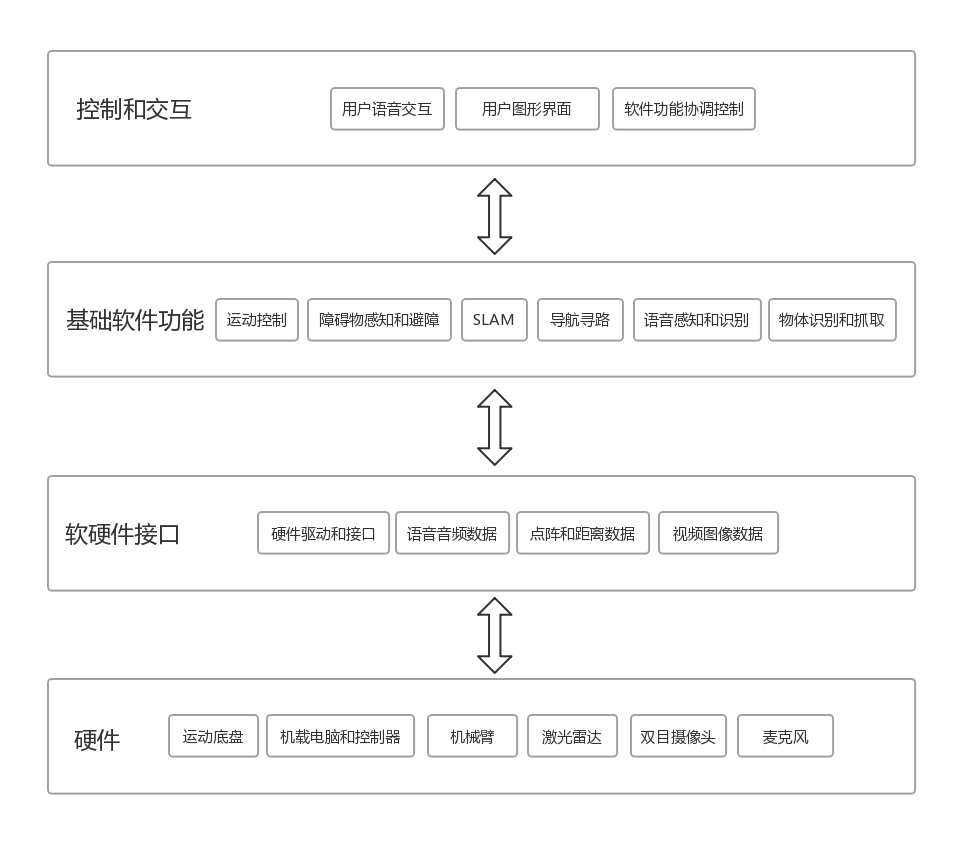


图5 系统总体结构图

总体结构分为以下几个层次：硬件、软硬件接口及交互数据、基础软件功能、用户服务和界面。

最底层的硬件包括运动底盘、机载电脑和控制器、机械臂、激光雷达、双目摄像头、麦克风等机器人提供和支持上层功能所需的硬件组成部件；在硬件层和基础软件功能之间的是软硬件接口以及相在软件和硬件之间交换的数据，包括语音音频数据、激光雷达得到的点阵和距离数据、双目视觉摄像头得到的视频图像数据等；基础软件功能包括控制机器人的运动、障碍物感知和避障、SLAM地图建立、导航寻路、语音感知和识别、物体识别和抓取等；最顶层包括用户语音交互、图形界面、软件功能协调控制。

### 软件体系结构

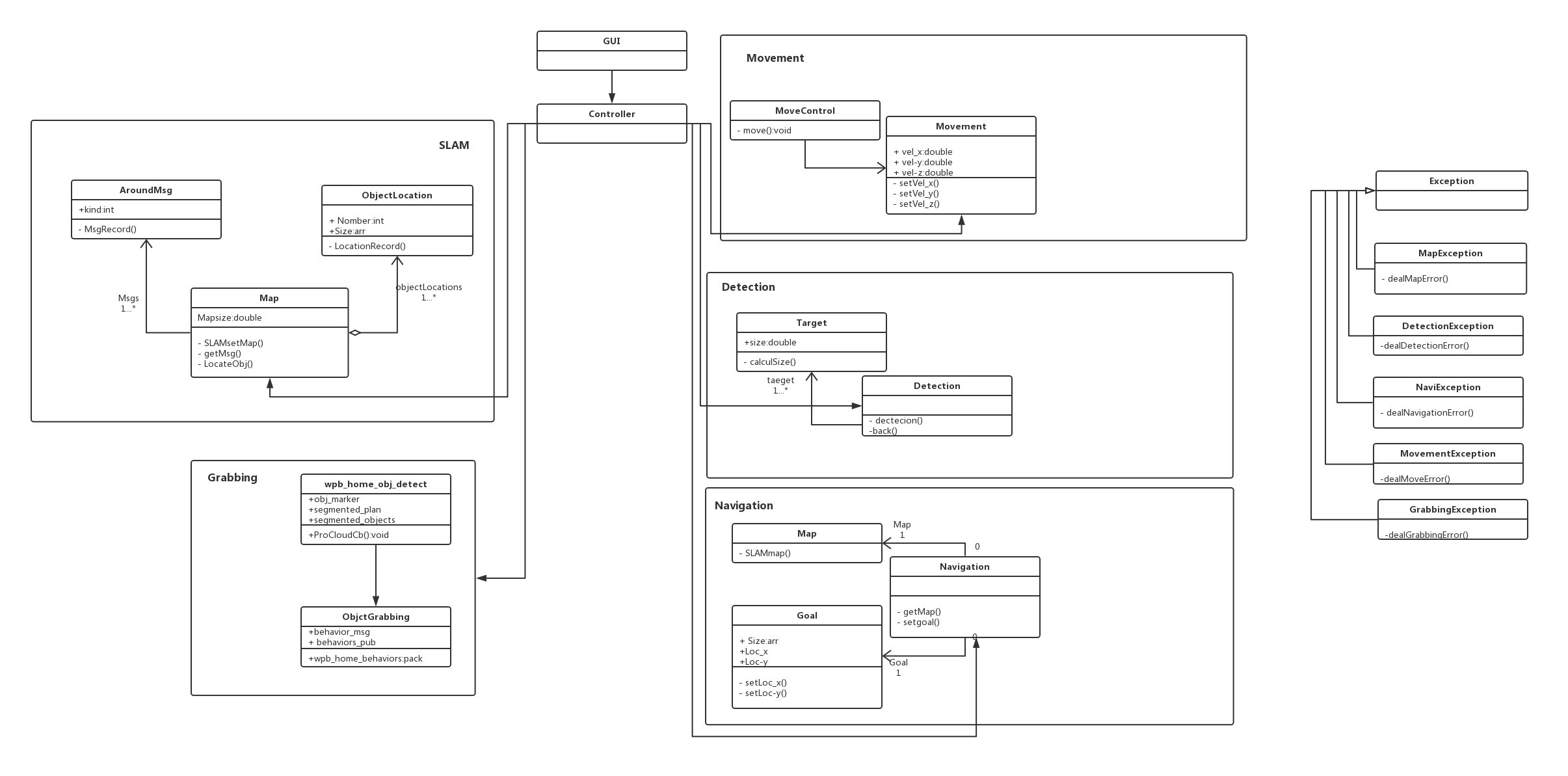


图6 软件体系结构图

上图为机器人系统的软件体系结构，其中Controller为控制类，负责控制其余模块以及进行任务调度。除Controller以外，其余类大致分为如下几个模块：Voice，SLAM，NavigationAndObjectGrabbing和ExceptionHandler。

Voice包括以下几个与语音的监听、识别、反馈相关的几个类：VoiceDetection负责监听周围环境的语音信息，进行收集，并将收集到的语音进行识别和关键字提取，并进行命令的解析，得到相关的指令结果反馈回Controller；VoiceInteraction进行语音交互，根据Controller的指令，进行语音合成并与用户进行语音交互。

SLAM包括以下几个类：Movement类负责在建图的过程中进行机器人的运动控制，这一过程主要依靠用户通过运动控制手柄进行操作，Movement类收到手柄的运动指令，并进行解析和对运动的控制；BuildMap类负责的运动的过程中对周围环境进行感知并进行地图重建和存储；ObjectRecord负责在地图构建的过程中存取关键点，在达到不同的目的点时，用户通过语音给出指令，经语音解析相关模块解析并由Controller发出指令后，ObjectRecord类被调用，存储目的点所在的位置和物品的类别。

NavigationAndObjectGrabbing包括：Navigation类负责在机器人自主运动取商品和返回出发点的过程中的运动控制和路线规划；ObstacleDetection负责在运动过程中的障碍感知、避障处理等；ObjectGrabbing类负责到达目的点之后的物品定位和抓取。

ExceptionHandler处理在机器人工作过程中出现的各种异常，进行异常的捕获和容错处理。

### 硬件体系结构

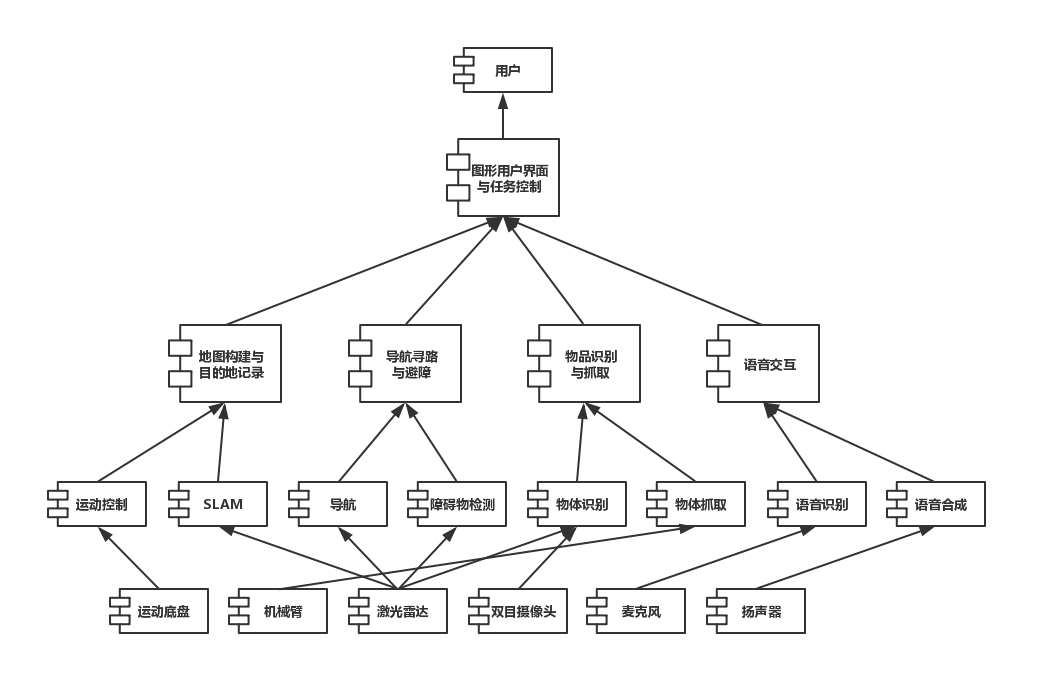


图7 硬件体系结构图

用户和管理员在使用系统时主要通过图形界面以及语音指令进行操控，机载电脑和Controller通过麦克风和Voice模块得到用户指令的语音并进行解析，将解析结果反馈给控制器，控制器发送指令给建图和物体记录模块或导航和物体抓取模块，通过运动底盘和激光雷达完成所指定的建立地图、记录物体位置任务或通过运动底盘、激光雷达、双目摄像头、机械臂完成路径规划和导航、障碍检测和躲避、物体识别和抓取任务。

### 技术体系结构

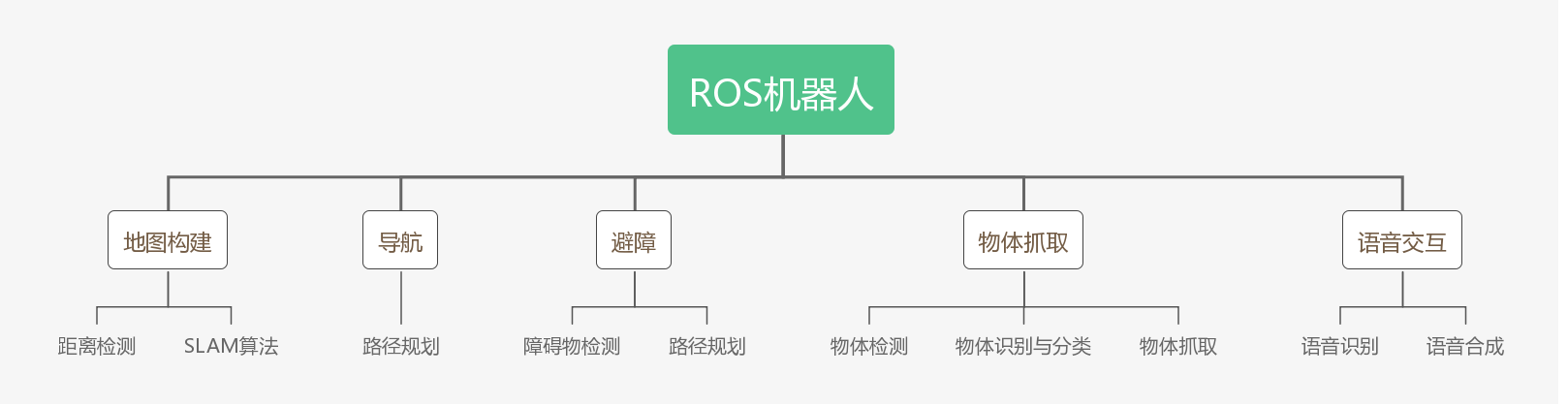


图8 技术体系结构

本设计以基于ROS的机器人为硬件基础，依赖于SLAM、导航、障碍物识别和躲避、物体检测和抓取、语音识别和合成算法等技术实现。由于ROS提供了大量相关的API，因此技术难度得以一定程度的下降。

## 关键问题及解决方案

### SLAM建图

SLAM建图是本机器人系统设计的基础之一，机器人后续的移动及避障、目的地导航、物品的识别和抓取等都需要以建图过程中得到的地图文件作为基础。

ROS 支持多种 SLAM 算法，其中主流的是 Hector SLAM和Gmapping，我们选择Hector SLAM算法。启动 Rviz后，用手推动机器人进行移动，观察 Rviz 里的颜色区域变化情况。可以发现白色的区域在不断增加，而且跟周围的墙壁桌子这些静态障碍物的轮廓越来越接近。将机器人推动绕场地一圈后，可以把地图保存下来。

将地图文件保存后，地图文件将作为后续应用的支撑基础。

### 导航和障碍物识别与躲避

在完成地图构建和目的点（商品）的记录后，通过语音指令开始前往目的商品所在的位置并进行抓取，这一过程依赖于导航和避障功能。

我们使用ROS自带的蒙特卡洛自适应定位算法与ROS机器人基础运动所需的move\_base包

在本设计中，计划通过语音进行目的物品的指定，指定为之前在建图过程中所存储的某个物品，然后从出发点向物品所在的位置进行导航。

在实际应用中，需要进行连续导航，例如从出发点导航到目的物品所在的位置，进行物品的抓取之后再返回，这样的需求是几个连续的导航任务组合，这就需要在几个航点中连续进行导航。到导航的过程中需要考虑到机器人自身定位与实际所在的位置之间的偏差，因而需要人为进行监视并实时进行误差的纠正。

### 语音识别和合成

本设计需要通过语音进行控制，在机器人的不同服务状态之间进行切换，并在相应的服务状态中处理对应的语音指令。

我们选用科大讯飞的语音识别引擎，能够得到较好的识别准确率，但是需要机器人的机载电脑连接网络进行处理。在本设计中，语音指令的识别方法是进行语音转换为文字的操作，然后从转换得到的文字中提取出相关的关键字，并得到相对应的控制指令。因此，在实际使用之前需要预先定义需要使用的语音识别关键字，并定义使用语音指令的方式，然后将定义好的关键字录入到系统中。

除去语音控制所需的语音识别以外，由于本系统需要通过语音与用户进行交互，因此也需要进行语音的合成，这里考虑使用ROS的第三方包sound\_play完成。

### 物品检测识别和抓取

除去上述的几个关键功能以外，本设计同样需要物品检测识别和抓取的相关算法。机器人移动到指定位置后，对指定位置在物体进行识别，然后操纵机械臂进行物品的抓取。

启智 ROS 机器人头部 Kinect2 可以输出三维点云，可以用于对环境里的物体进行检测。本设计中机器人需要识别目标地点存在的物品，而通常物品都是放在桌子或者货架上，使用 PCL 的平面检测算法，可以将桌面上的物品标注出来，并算出其三维坐标。这个例程主要是通过检测桌子平面，然后分割平面上方的点云团簇来检测物品的，只能得到物品存在与否以及其位置信息而不能对物品的种类等信息进行判断，因此在后续设计中我们考虑引入通过双目视觉摄像头得到的图像信息进行物品的分类。

物品抓取需要使用机器人的机械臂，操作机器人抓取的动作需要指定机械臂手爪的开和大小，这一部分需要根据被抓取的物品进行调节，由于需要抓取的物品相对固定而获取物品各个维度上的尺寸的难度相对较高，因此我们考虑在预定义物品关键词的同时进行物体抓取所需参数的指定。

# 接口设计

## 系统用户界面

依托于ros系统自带的Rviz,构建出可视化界面。通过slam建图获得地图，Rviz会将机器人模型放在建好的地图中。

可以看到此时的地图，黑色图案代表静态障碍物轮廓，它的周围出现了蓝色的色带，这个色带表示的是安全边界，色带宽度和机器人的半径大致相等。也就是说如果机器人进入这个色带，就有可能和静态障碍物（墙壁或桌椅腿）发生碰撞。

机器人的默认位置是地图的起始点，就是我们建图时机器人出发的地方。而在导航开始时，现实世界中机器人很有可能并不在之前建图时机器人出发的位置。如果现实世界中的机器人位置和 Rviz 中显示的有偏差，需要在导航前先纠正这个偏差 。

设置好机器人的初始位置后，可以开始为机器人指定导航的目标地点。点击 机器人模型。此时会再次出现绿色箭头，按住不放在屏幕上拖动画圈，设置机器人移动到终点后的朝向。

选择完目标朝向后，松开点击，全局规划器会自动规划出一条紫色的路径，这条路径从机器人当前点出发，绕开蓝色的安全边界，到移动目标点结束。

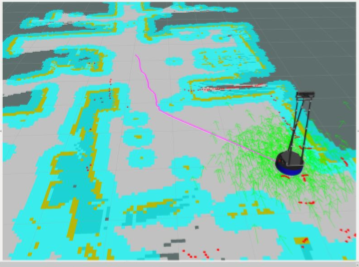


图9 系统用户界面

## 硬件接口

### 手柄操控

1. 将控制电脑安装到启智机器人上，连接相应的 USB 接口，将操控手柄连接到机器人底盘 面板的 USB 接口。
2. 打开机器人底盘面板的 USB 电源开关
3. 启动终端程序，键入指令
4. 打开机器人底盘面板上的红色急停开关，拨动手柄摇杆，控制机器人移动

### 激光雷达

1. 将启智 ROS 机器人的 USB Type-C 连接到机载平板电脑上。
2. 打开机器人底盘上的 USB 电源开关。
3. 在机载平板电脑上，启动程序，输入指令

### SLAM建图

1. 将启智 ROS 机器人的 USB Type-C 连接到机载平板电脑上。
2. 打开机器人底盘上的 USB 电源开关
3. 关闭机器人的红色急停开关
4. 在机载平板电脑上，启动程序，输入指令

### Navigation 导航

1. 将启智 ROS 机器人的 USB Type-C 连接到机载平板电脑上。
2. 打开机器人底盘上的 USB 电源开关
3. 打开机器人的红色急停开关
4. 打开文件管理器。将“map.pgm”和“map.yaml”拷贝到工作目录 “catkin\_sw/src/wpb\_home/wpb\_home\_tutorials/maps”里。
5. 启动终端程序，输入指令

## 软件接口

### 运动控制

从 Ubuntu 桌面左侧的启动栏里点击“RoboWare Studio”的图标，启动 RoboWare Studio。完成运动控制源文件。从 Ubuntu 桌面左侧的启动栏里点击“Terminal”终端图标，启动终端程序，在终端里编译，直到显示编译进度“[100%]”字样，表示编译成功。

启动启智机器人的核心节点。启动“Terminal”，输入指令。按下回车键后，会显示节点初始化信息。

启动 vel\_ctrl 节点。首先确认机器人底盘的电源都已经打开。启动“Terminal”，输入指令。按下回车键后，可以看到机器人以 0.1 米/秒的速度缓慢向前移动。

### 4.3.2导航

从 Ubuntu 桌面左侧的启动栏里点击“RoboWare Studio”的图标，启动 RoboWare Studio。编写导航源代码。从 Ubuntu 桌面左侧的启动栏里点击“Terminal”终端图标，启动终端程序，在终端里编译，直到显示编译进度“[100%]”字样，表示编译成功。

启动启智机器人的核心节点。启动“Terminal”，输入指令。按下回车键后，会启动 Rviz，初始化地图并在后台启动导航服务。

启动 simple\_goal 节点。首先确认机器人底盘的电源都已经打开，底盘轮子上电抱死。启动启智机器人的核心节点。启动“Terminal”，输入指令。按下回车键后，可以看到 simple\_goal 节点发出导航服务请求的提示“Sending goal”。 此时再查看 Rviz 的显示界面，可以看到一条紫红色的线条从机器人脚下伸向机器人正前方的位置

机器会开始向前移动，机器人运动到目标地点后，便会停止移动。

### 内部接口

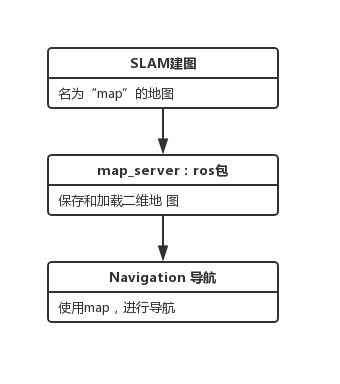


图10 内部接口类图

内部接口有很多，但在其中最为典型的便是map的使用。通过SLAM建图获得地图；使用名为map＿server的ROS包保存和加载地图；最后Navigation导航使用地图进行导航。

# 数据库设计

在我们的机器人设计中会简单使用数据库。以下是大概的会存上去的数据：

## 用户信息

用户信息会包括：

* 用户密码
* 用户名
* SSID
* 登录信息
* 用户权

等信息。这些信息是为了能识别用户身份是否可以使用机器人系统。

## 商品信息

商品信息包括：

* 商品名称
* 商品位置
* 商品图
* 商品是否已被取走（bool）

## 关键词

关键词主要是用于语音识别中的功能，此部分包括：

* 关键词
* 每个关键词功能

## 地图信息

地图信息是以yml格式存储，使用过程中直接使用fetch获取地图信息

## 日志

日志部分是简单地记录数据库中的修改日志，登录活动信息为了方便日后若有异常去查看。

# 详细设计

## 运动控制

**输入/输出：**

输入水平运动中的x，y方向的速度，和z关于旋转的角速度。输出的结果为机器人的运动指令。

**实现功能：**

实现机器人的基本运动功能，包括基本的水平运动功能和转向运动功能。其他模块通过调用运动模块的MoveControl，输入需要的x，y，z速度，得到响应，实现机器人的运动。

**类及流程图：**

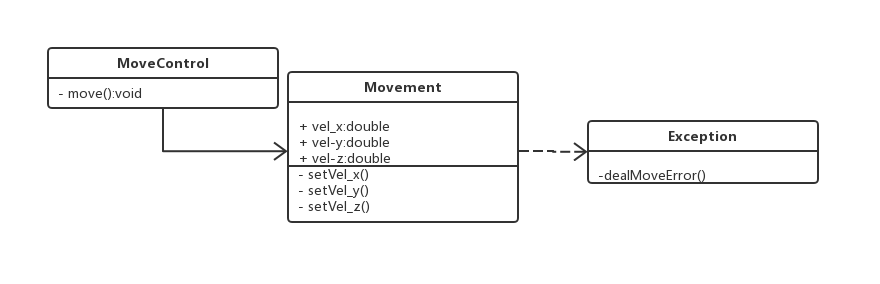


图11 运动控制类图

运动模块涉及到三个类，类Movement是调用机器人控制系统的方法来操作机器人的运动，模块接口是MoveControl类，导航模块可以通过MoveControl来输入相应信息实现机器人的运动。在设置速度中如果遇到了异常会使用异常处理类来处理。

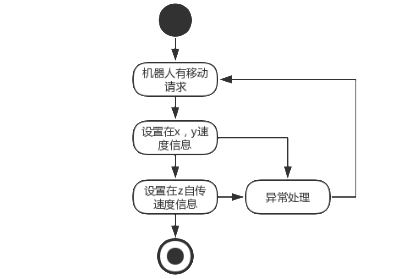


图12 运动流程图

## SLAM建图

**输入/输出：**

输入360度雷达扫描得到的周围信息。输出的结果为SLAM建成的地图。

**实现功能：**

实现通过雷达传感器扫描周围物体建成地图，记录下商品位置，并将地图保存下来。其他模块通过已经保存下来的地图，可以规划导航路径，判断是否有突发情况出现障碍物。

**类及流程图：**

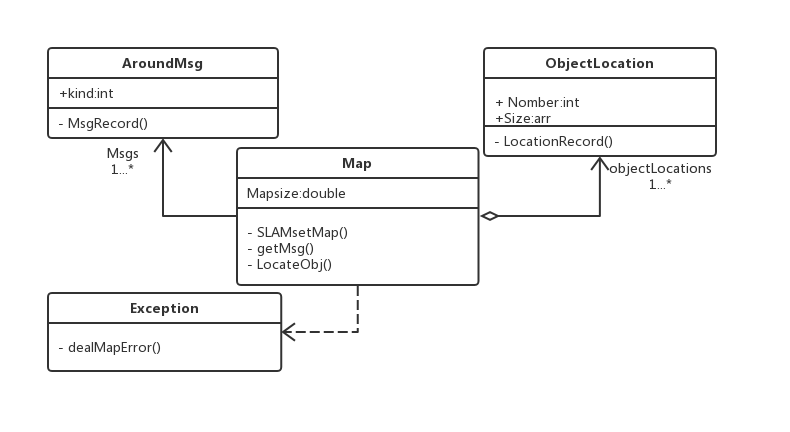


图13 建图类图

通过AroundMsg类中方法得到传感器收集到的周围的环境信息，通过ObjectLocation类中方法得到商品的位置信息，Map作为主类，调用两个类得到环境信息和物体位置信息，生成地图，通过save方法保存下来。其中Exception作为异常处理类处理在地图生成过程中的异常。

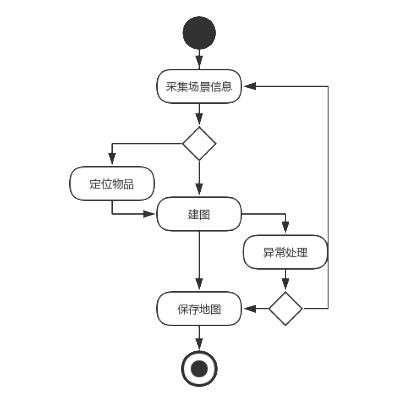


图14 建图l流程图

## Navigation导航

**输入/输出：**

输入要到达目的地的相关坐标位置。输出的结果为规划的路径。

**实现功能：**

实现通过输入目的地，生成导航的路径信息。这其中调用了生成的Map信息。

**类及流程图：**

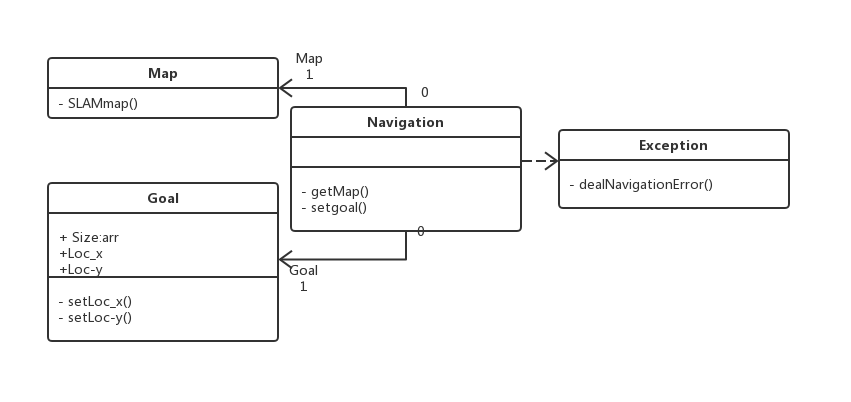


图15 导航类图

主类Navigation调用Map类中的方法使用保存下来的地图信息，并通过Goal类中的方法得到目标商品的位置、大小等信息，在主类中生成导航路径。在导航中出现异常，如碰到障碍物，或者目标商品不存在，会调用异常类中的方法。

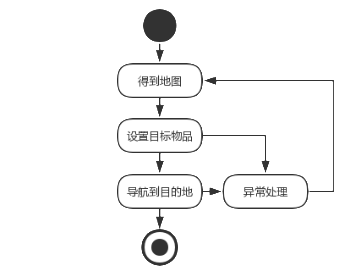


图16 导航流程图

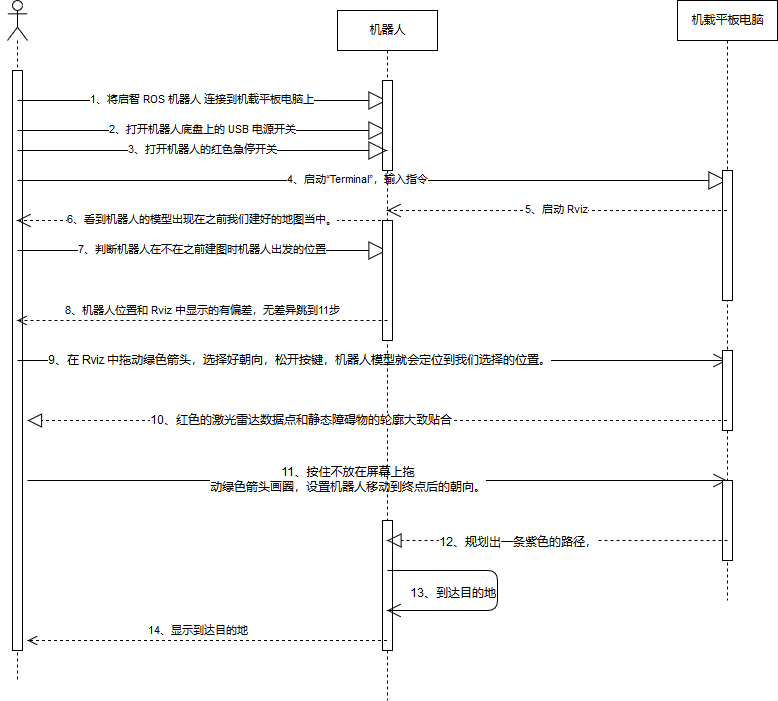


图17 导航时序图

Navigation导航时序图如图所示。用户发布命令并接受反馈；机器人接受命令并做出行动；机载平板电脑分析指令，并将指令翻译给机器人令其执行。

## 避障运动

**输入/输出：**

输入检测到的障碍物的信息。输出的结果为避障运动，返回出发点发出警报要求重新建图。

**实现功能：**

实现在导航过程中碰到突发情况出现障碍物实现避障行为。最为稳定有效的方法是回到出发点并请求重新建图。

**类及流程图：**

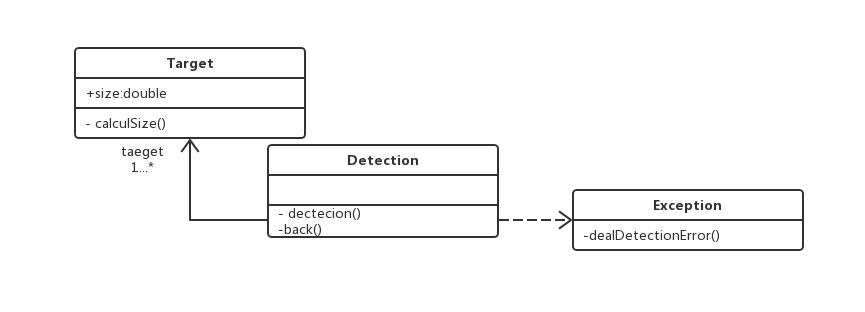


图18 避障运动类图

主类Detection中记录避障算法，检测障碍物的Target类中有判断物体为移动或者静止的方法。在避障处理中有异常会调用异常类中方法。

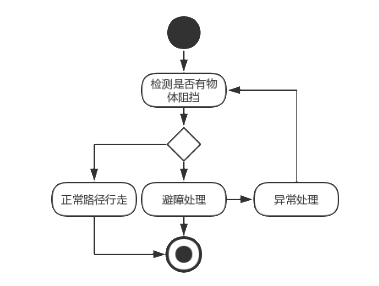


图19 避障运动流程图

检测模块实现的是物品检测功能，我们机器人用到Kinect2视觉传感器和激光雷达，首先Kinect2和激光雷达检测到物体阻挡如果没有机器人按照正常路径行走，行走时一旦检测到物体阻挡就进行避障处理，机器人先判断物体是静态或动态，若静态机器人直接绕开，若动态机器人等物体离开之后才按照正常路径行走。

## 物品抓取

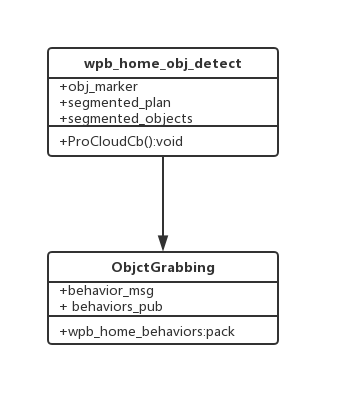


图20 物品抓取类图

wpb\_home\_obj\_detect创建一个 tf\_listener 用来进行点云三维坐标转换；订阅 Kinect2 的三维点云数据用于做物体检测；向外发布一个名为“obj\_marker”的主题，用于在 Rviz 中标注物品的空间位置；向外发布一个名为“segmented\_plane”的主题，用于在 Rviz 中显示检测出的平面 的点云集合；向外发布一个名为“segmented\_objects”的主题，用于在 Rviz 中显示检测出的物品 的点云集合。以上工作完成后，调用 ros::spin()挂起主线程，之后的工作交给各个数据流的回调函数完成；Kinect2 的三维点云数据流的回调函数为void ProcCloudCB ()，Kinect2 每生成一帧点云就会调用一次这个函数。

启智 ROS 安装机械臂后，可以完成桌面上物品的抓取功能。启智 ROS 抓取行为以服务的形式安排在 wpb\_home\_behaviors 包中，因为其代码比较长，直接修改调用比较容易出错，我们可以通过外部调用的形式启动抓取功能。

ObjectGrabbing声明名为 behaviors\_pub 的消息发布对象，后面会通过这个对象向启智 ROS 的行为主题发送命令消息；声明一个字符串消息，在后面的操作中，会将控制指令放入这个字符消息并发布出去。

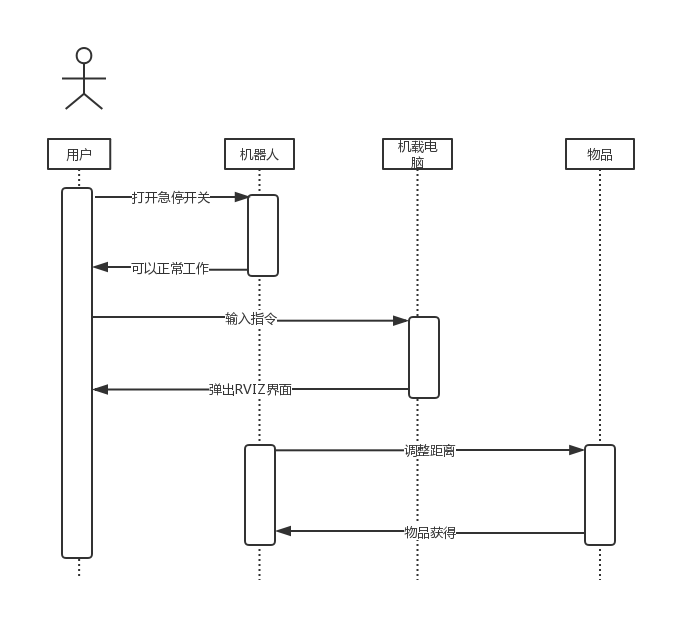


图21 物品抓取时序图

物品抓取的步骤：用户打开机器人、用户输入指令、机器人调整与物品的距离并抓取物品。

## 语音识别与合成

机器人主要通过语音与使用者（包括管理员和用户）进行交互，机器人系统首先预定义需要识别指令的关键字以及需要记录的物品的物品名，在实际使用的过程中，将麦克风收集得到的语音音频文件提交到服务器进行语音的解析，将得到的文字转换的结果返回，然后通过识别转换结果中的关键字判断发出的指令。机器人通过语音与用户进行交互，因此需要语音合成与播报相关的功能。

### 硬件基础

ROS机器人提供的阵列麦克风，以及用于网络连接的相关硬件

#### 语音识别与合成时序图

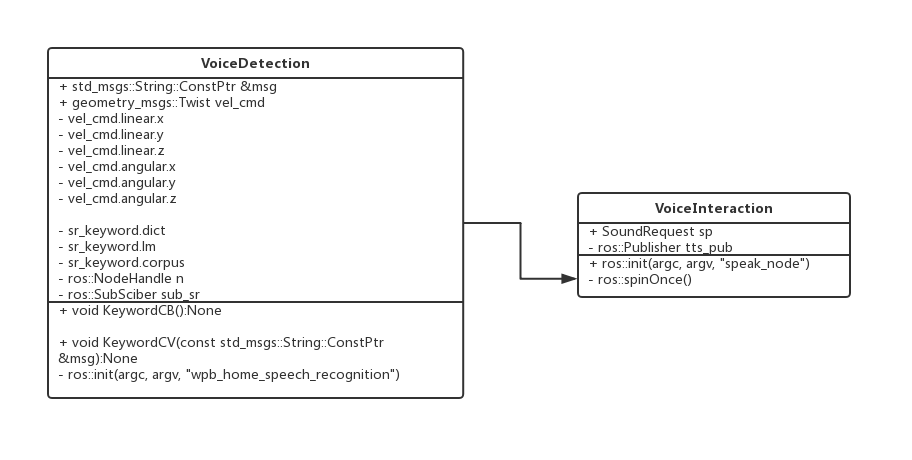


图22 语音识别与合成类图

#### 模块组成和类图

**语音识别类VoiceDetection：**

语音识别类负责对麦克风收集到的语音音频文件进行解析，得到转换后的文本文件，并中文本文件中得到关键字并映射为关键字对应的指令，将指令及指令所需的参数传回给Controller。

**语音交互与合成类VoiceInteraction：**

语音交互与合成负责将从Controller类得到的运行信息、状态反馈等文字信息转换为语音音频文件并进行播放，从而实现与用户的语音交互。

# 运行与开发环境

## 运行环境

### 运行硬件：

启智ROS机器人是室内机器人，在此环境之外运行可能会损坏机器人。工作平面需要能够承载不小于40kg的重量。如果表面太软，则机器人可能卡住，运动受阻。建议使用商用地毯、瓷砖等材质。启智ROS机器人原则上在水平平面上工作，坡道坡度不大于15度，坡道倾斜度过大可能导致倾覆

机器人不具备防水功能，在任何情况下，启智ROS机器人都不应该与雨水，雾，地面积水以及任何其他液体接触，否则可能导致电路和机构损坏

启智ROS机器人设计工作温度为15°C到35°C之间，使用中务必远离明火和其他热源

### 运行软件：

ROS 是一个构建在 Ubuntu 上的机器人开发系统，由一系列的软件包和通讯架构组成。启智 ROS 机器人默认使用的 ROS 是 indigo 版本，基于 Ubuntu 14.04，是 ROS 第一个长期维护版本，也是目前世界范围内应用最广泛的 ROS 版本。

启智机器人出厂标配的电脑里已经安装好 ROS、IDE 和启智机器人的源码包，可以直接使用。启智机器人除了基础功能的软件包以外，还提供了一个扩展软件包，该软件包里包含了大量复合任务的实现例程。因为该软件包需要依赖 xfyun\_waterplus 和 waterplus\_map\_tools 等第三方软件包，所以和基础软件包相互独立，以方便持续维护更新。

## 开发环境

**开发硬件：**

* 嵌入式开发板各类传感器
* 运动装置
* 开发面板

**开发软件：**

* 嵌入式操作系统：LINUX系统
* 应用软件开发语言： C/C++/Python
* 开发环境使用的是 RoboWare Studio，这是济南汤尼机器人科技有限公司基于 Visual Studio Code 开发的 ROS 专用 IDE。

# 需求可追踪性说明

## 功能需求

### 到达目的地

首先最大也是最终我们想要的功能就是能让机器人到达目的地。要到达目的地有些任务必须先做：获得地图（8.1.4），规划路径（8.1.3），避开障碍（8.1.2）。获得地图后，机器人大脑相当于已经知道它周围的环境。知道了之后，现在如果它想要到达某个地点那它必须规划路径。路径规划中要考虑的是起点与终点之间存在的障碍（要避开）。这些满足之后，机器人可根据规划移动到目的地。

### 避障运动

避障运动的首要关键是要让机器人激光雷达扫描出存在的障碍并更新地图（收集数据）（8.1.4）。第二次在同一个地点就可以直接按照已存在的地图规划路径（8.1.3），以便避开障碍。

### 规划路径

要使机器人能规划路径之前有两种情况需要考虑，第一种是在地图还没完全建立起来的时候，此时要先用激光雷达扫描机器人范围内的环境建立地图（8.1.4）。地图建立起来之后便可判断哪里存在障碍，需要怎么走才能到达目的地，从那条路会比较快。

第二种是地图已经完全建立起来，此时不用再扫描（除非地图有变化），直接按照已存在的数据及地图使用Navigation导航规划路径。

### 建立地图

建立地图是机器人使用激光雷达扫描它范围内周围的环境并同时同步更新地图数据来获得的。地图建立使用SLAM建图，然后把地图存下来。地图建立起来之后就可以规划路径了。

### 机器人操作

机器人操作是由用户输入的指令来判断的。机器人能真正运动之前在用户和机器人之间必须有指令的输入，然后让机器人大脑分析这个指令，让机器人可以根据分析出的命令进行操作（比如移动到某个地点等等）。

### 打开/关闭机器人

打开和关闭机器人都可以从机器人本身直接操作的，所以只需保证开关面板正常工作即可。要使机器人准备就绪，需要在打开机器人后也开启电脑上的系统，真正让机器人与电脑连接。

### 分析指令

分析指令之前，显然要先让用户输入指令。输入后由ROS系统来分析指令，转化成机器人可以理解的指令，使它运动起来。

### 发出指令

机载平板电脑及ROS系统、本设计中的GUI和语音识别模块可以正常使用。

## 非功能需求

### 性能

－优化Slam建图算法，确保绕场一周即可建立地图模型。

－优化导航算法，使机器人可以在短时间内规划出一条路径。

－对传感器的数据进行分析，可以在短时间内应付突发情况。

### 安全性

为了确保安全性，首先要控制机器人的速度小于0.2 m/s。然后要设定机器人与障碍之间的判断距离。意思是说若机器人扫描出前面在这个距离范围内有障碍，必须减速后停止。距离设定需要确保机器人有足够时间先减速后停止。

### 可扩展性

－对象的方法不应超过300行，新增功能不应大幅修改原有代码。

### 可靠性

－对传感器的数据进行分析，可以在短时间内应付突发情况。

－优化导航算法，如果机器人路径规划的起点与终点如果没有障碍物，即可到达。