**【简易机器人】**

**软件设计说明书**

**【*SDD108*】**

**【V2.1】**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 智慧女孩不秃头 | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 17373356 | 杨昶 | 详细设计和接口设计内容编写 |
| 17373357 | 张稚馨 | 体系结构设计内容编写 |
| 17373358 | 邹桃 | 项目概述、需求可追踪性说明内容编写 |
| 17373359 | 邢译洋 | 详细设计和接口设计内容编写 |
| 17373444 | 葛毅飞 | 需求概述、数据库设计内容编写 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| V0.4 | 2020.4.17 | 张稚馨 | 邹桃 | 增加体系结构设计部分 |
| V0.7 | 2020.4.17 | 杨昶、邢译洋 | 张稚馨 | 增加接口设计、详细设计部分 |
| V0.9 | 2020.4.18 | 邹桃 | 张稚馨 | 增加项目概述、需求可追踪性说明 |
| V1.0 | 2020.4.18 | 张稚馨 | 邹桃 | 修改体系结构设计部分 |
| V1.2 | 2020.4.18 | 杨昶、邢译洋 | 邹桃 | 修改详细设计部分 |
| V1.5 | 2020.4.18 | 邹桃 | 邢译洋 | 修改部分活动图 |
| V1.6 | 2020.4.19 | 葛毅飞 | 邹桃 | 增加需求概述部分和数据库设计部分 |
| V2.0 | 2020.4.21 | 张稚馨 | 邹桃 | 修改体系结构部分措辞 |
| V2.1 | 2020.4.21 | 邹桃 | 张稚馨 | 修改用户接口，详细设计、需求可追踪性说明部分自主避障模块 |

目录

[1 范围 1](#_Toc3494)

[1.1 项目概述 1](#_Toc26959)

[1.2 文档概述 1](#_Toc25319)

[1.3 术语和缩略词 2](#_Toc8278)

[1.4 引用文档 3](#_Toc14741)

[2. 需求概述 4](#_Toc18593)

[2.1 功能需求 4](#_Toc23763)

[2.1.1 系统用例图 4](#_Toc10027)

[2.1.2 简要说明 4](#_Toc4744)

[2.2 数据需求 5](#_Toc5040)

[2.3 系统参与者 5](#_Toc7982)

[2.4 用例模型 6](#_Toc1476)

[2.4.1 安装控件 6](#_Toc21757)

[2.4.2 启动/关闭机器人 6](#_Toc17363)

[2.4.3 选择机器人任务模式 7](#_Toc29058)

[2.4.4 控制机器人自由避障行走 7](#_Toc2774)

[2.4.5 建立地图 8](#_Toc25829)

[2.4.6 控制机器人规划路径 9](#_Toc32005)

[2.4.7 控制机器人识别、抓取特定物体 9](#_Toc14445)

[2.4.8 遇到错误条件 10](#_Toc22226)

[2.4.9 调试/重新配置系统相关特性并维护硬件 10](#_Toc13819)

[3. 数据库设计 11](#_Toc9448)

[3.1 E-R图 12](#_Toc16537)

[3.2 数据库关系模式 12](#_Toc19212)

[3.2.1 数据表及数据项介绍 12](#_Toc3667)

[3.3 关系模式规范化 13](#_Toc27402)

[3.3.1 用户使用关系 14](#_Toc13805)

[3.3.2 功能实现关系 14](#_Toc29724)

[3.3.3 控制关系 14](#_Toc22326)

[3.3.4 测距关系 14](#_Toc9831)

[3.3.5 物体识别关系 14](#_Toc6917)

[3.3.6物体抓取关系 15](#_Toc29572)

[4. 体系结构设计 16](#_Toc28717)

[4.1 总体结构 16](#_Toc11312)

[4.2 软件体系结构 17](#_Toc21728)

[4.2.1软件总体结构 17](#_Toc11412)

[4.3 硬件体系结构 18](#_Toc14428)

[4.4 技术体系结构 20](#_Toc971)

[4.5 支撑体系结构(部署和实施方案) 21](#_Toc7897)

[5. 接口设计 23](#_Toc14475)

[5.1 对外用户调用接口 23](#_Toc16913)

[5.1.1 “基本建图”功能相关用户接口 23](#_Toc7195)

[5.1.2 “定点巡航”功能相关用户接口 26](#_Toc31449)

[5.1.3 “物体抓取”功能相关用户接口 27](#_Toc26111)

[5.2 对内硬件接口 29](#_Toc25757)

[5.2.1 激光雷达 29](#_Toc26113)

[5.2.2 姿态测量系统 30](#_Toc29130)

[5.2.3 立体相机 30](#_Toc18303)

[5.2.4 机器人运动底盘 30](#_Toc8291)

[5.3 对内软件接口 30](#_Toc30755)

[5.3.1 用户交互指令部分——Instruction类 30](#_Toc14415)

[5.3.2 机器人总体调度部分——RobotControl类 31](#_Toc5494)

[5.3.3 连接网络部分——Internet类 31](#_Toc2350)

[5.3.4 机器人运行控制部分——Movement类 32](#_Toc8172)

[5.3.5 机器人自主避障部分——Obstacle类 32](#_Toc2994)

[5.3.6 传感器部分——Sensor类 32](#_Toc26450)

[5.3.7 物体抓取部分—— Grasp类 32](#_Toc10952)

[5.3.8 物体识别部分——Kinect类 33](#_Toc980)

[5.3.9 路径规划与存储部分——Path类 33](#_Toc23326)

[5.3.10 地图构建与存储部分——Map类 33](#_Toc2022)

[6. 详细设计 33](#_Toc7112)

[6.1 机器人运动模块 33](#_Toc15073)

[6.2 机器人避障模块 34](#_Toc4366)

[6.3 地图构建与存储模块 36](#_Toc26893)

[6.4 定点巡航模块 38](#_Toc12971)

[6.5 物体识别模块 40](#_Toc32109)

[6.6 物体抓取模块 42](#_Toc4357)

[7. 运行与开发环境 43](#_Toc31377)

[7.1 运行环境 43](#_Toc4390)

[7.2 开发环境 44](#_Toc21291)

[8. 需求可追踪性说明 45](#_Toc15744)

[8.1 功能需求 45](#_Toc26699)

[8.1.1 基本建图功能 45](#_Toc31163)

[8.1.2 导航功能 45](#_Toc25189)

[8.1.3 物品识别与抓取功能 45](#_Toc6696)

[8.2 非功能需求 46](#_Toc907)

[8.2.1 可移植性 46](#_Toc9099)

[8.2.2 完整性 46](#_Toc17543)

[8.2.3 可扩展性 46](#_Toc24141)

[8.2.4 健壮性 47](#_Toc23482)

[8.2.5 易用性 47](#_Toc107)

[8.2.6 安全性 47](#_Toc993)

# 1 范围

## 项目概述

面对复杂环境中的物品检测和获取、考虑人力劳动的冗余，我们需要机器人来帮助我们完成两方面的任务，其一是简单且重复的工作；其二是恶劣环境下的工作。为满足这两方面的需求，我们进行简易机器人的开发。

在主要功能方面，本项目机器人可提供导航引领、跟随服务、自动识别障碍物并进行躲避、接受交互指令获取特定物品的功能。用户可以通过语音交互、图形界面化操作对机器人进行引导和控制，完成特定任务。

在非功能性需求方面，综合客观条件，我们重点考虑合理设计GUI界面使得用户能够在短时间内熟练使用系统、同时保证系统完整性，用户不会遇到手册上无法运行的指令和功能；运行层面，考虑性能指标、可用性使得机器人运行流程系统化，面对复杂情况可以做出相应选择；在开发层面，考虑系统可移植性、可维护性和可扩展性，使得开发者能够在短时间内对系统进行改进开发；以及在机器人硬件层面健壮性和安全性方面，室内环境下机器人可以免受一些环境因素变化干扰，保证正常运行。

应用场景方面，本项目机器人可以为室内环境下的家居智能机器人，帮助用户自动完成任务中的物品获取工作；工厂服务机器人，实现货物的分拣、货物的搬运；餐厅送餐机器人，提供自动送餐服务。用户通过与机器人进行交互，使得机器人自动完成相应任务，并且能够灵活应对障碍物阻挡、环境因素干扰等一些情况，帮助用户完成任务。

## 文档概述

本文档主要用于明确本次软件工程项目开发的实现功能和软件设计结构，是这次开发的软件开发文档。根据本文档的内容，我们将进行具体的实践开发工作。

文档具体可以分为八部分：

文档总述与项目概况：主要描述整个项目的概况与本文档的概况。

需求概述：根据用户需求、业务需求和功能需求，对项目实现系统具体功能进行拆分，模块化，指导体系结构的设计。

数据库设计：通过对需求的分析总结构建系统数据结构，利用关系数据模式规范化数据库设计，得到最终的数据库设计。

体系结构设计：将系统设计展示为层次结构图、类图、协作图，并对系统工作流程进行简要说明。

接口设计：结合SRS文档用户界面设计和业务需求，分析得到系统用户界面；结合体系结构设计，编写系统的软硬件外部接口和内部接口的设计。

详细设计：对于系统的主要功能和关键模块，结合体系结构设计展示类图及活动图并说明内容，得到系统的详细设计。

运行与开发环境：在需求文档基础上，结合软件开发，进一步明确系统运行和开发的软硬件环境。

需求可追踪性说明：将系统功能性需求和非功能性需求与本文档设计结合，得到系统的需求可追踪性说明。

## 术语和缩略词

**表格1 术语和缩略词**

|  |  |
| --- | --- |
| 术语/缩略词 | 解释/全称 |
| ROS | Robot Operating System机器人操作系统 |
| Kinect2 | 一款3D体感摄影机 |
| YOLO | 滑动窗口的目标检测算法 |
| ER图 | Entity-relationship model，实体联系图。 |
| USB-HUB | Universal Serial Bus - Hub 通用串行总线集线器。 |
| 激光雷达 | 思岚（SLAMTEC） RPLIDAR A2。测距范围： 0.15 米-12 米；扫描角度： 360°；  测距分辨率：＜实际距离的 1%；角度分辨率： 0.9°；扫描频率： 10Hz。 |
| 控制面板 | 配置机器人系统的接口。 |
| 底盘控制器 | 启智控制器内部运行了启智 ROS 机器人专用固件，负责 PC 机于机器人之间的数据交互。 |
| 机械臂 | 启智 ROS 安装有一个用于抓取桌面上物品的机械臂，该机械臂提供两个控制量：上升高度和手爪的闭合宽度。 |
| 三维点云数据 | Kinect2传感器能够得到的数据，可用于物体的识别。 |

## 引用文档

**表格2 引用文档**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 标题 | 版本 | 发行日期 |
| 1 | 启智ROS机器人开发手册 | V1.1.0 | -- |
| 2 | 简易机器人开发计划【SDP108】 | V1.0 | 2020 |
| 3 | 简易机器人需求规格说明书【SRS108】 | V1.2 | 2020 |

# 需求概述

## 功能需求

### 2.1.1 系统用例图



图2-1 系统用例图

### 2.1.2 简要说明

本项目简易机器人是目标抓取型机器人，它可以实现三种任务模式。实现模式之前，用户需要通过USB接口安装控制组件。其后，用户打开机器人控制开关，机载电脑进入系统功能选择界面，用户可以在此界面选择基本**避障**、**路径规划**和**目标抓取**三种模式之一。

1. **基本避障：**机器人在用户设定的速度避障自由行走，直至行走时间达到用户设定的时间。
2. **路径规划：**用户先利用手柄推动机器人建立环境地图数据。建图后，用户在交互界面选择目标地点，机器人规划最短路径并行走。
3. **目标抓取：**用户先选定目标物体，机器人识别出目标物体后行走至该位置，并抓取目标物体。

当选择的任务模式完成之后，系统重新进入功能选择见面，用户可重新选择任一任务模式，或退出简易机器人系统。当系统出现错误时，用户可联系维护人员解决问题并对系统相关特性重新配置和维护。

## 数据需求

分析类设计如下表：

**表格 3 分析类表格**

|  |  |  |
| --- | --- | --- |
| 分析类设计 | | |
| 潜在类 | 表现形式 | 不符合条件 |
| 用户 | 角色 | 保留信息、所需服务 |
| 维护人员 | 角色 |  |
| 传感器 | 设备 |  |
| 控制面板 | 设备 |  |
| 系统 | 事物 |  |
| 紧急制动 | 事件 |  |
| 地图 | 事物 |  |
| 规划路径 | 事物 |  |
| 目标地点 | 场所 | 多个属性 |
| 当前地点 | 场所 | 多个属性 |
| 延迟时间 | 事物 | 多个属性 |

## 系统参与者

1. **用户：**配置机器人运行环境，连接网络、将机器人机载电脑通过USB接口相连；使用机器人之前打开电源，不同模式下可打开急停按钮；通过机载电脑进入系统功能选择界面，选择**避障**、**路径规划**和**目标抓取**三种模式之一，控制机器人完成任务；机器人运行结束后关闭机器人。在出现错误的情况下，用户可联系维护人员对系统进行调试和维护。
2. **系统维护人员：**在机器人系统出现故障后调试软硬件，修复漏洞。
3. **启智机器人、控制机连接组件：**机载电脑控制机器人完成移动、避障、路径规划、目标抓取等任务，机器人通过雷达等传感器向机载电脑反馈数据，协同合作完成用户指令。

## 用例模型

### 2.4.1 安装控件

**主要参与者：**用户

**目标：**安装控件并配置机器人运行所需环境

**前置条件：**用户具备软硬件条件、网络环境和安装手册

**启动：**用户使用机器人

**场景：**

1. 用户将U盘、手柄等自选设备通过USB与机器人控制器连接
2. 用户需要通过USB接口安装控制组件

**优先级：**高

**何时可用：**存在于任何一个增量

**使用频率：**高

**次要参与者：**启智机器人、控制部件

### 2.4.2 启动/关闭机器人

**主要参与者：**用户

**目标：**

启动：连通电源，启动机器人

关闭：退出系统，切断电源，使机器人停止工作

**前置条件：**控件安装完毕，系统软硬件状态良好，机器人正常运行时的环境已设置完毕，电池电量充足

**启动：**用户想要开启/关闭机器人

**场景：**

1. 使用之前，用户按下开机按钮
2. 使用完毕，用户按下关机按钮

**异常情况：**

机器人电池电量不足——出现点：场景中的1，此时需要更换电量充足的电池，再执行场景中的1

**优先级：**高

**何时可用：**存在于任何一个增量

**使用频率：**高

**次要参与者：**机器人控制面板

### 2.4.3 选择机器人任务模式

**主要参与者：**用户

**目标：**用户选择需要机器人完成的任务模式

**前置条件：**安装完毕机器人控件，机器人硬件功能良好，开启机器人开关，机器人放置于合适的环境，系统显示功能选择界面

**启动：**用户打开机器人电源开关，连接好机载电脑

**场景：**

1. 系统显示三种任务模式按钮和“退出”按钮

2. 用户根据具体需求点击任一按钮

**可选方案**：

用户点击“退出”按钮——出现点：场景中的2，用户此时退出系统

**优先级：**高

**何时何用：**所有增量之前

**使用频率：**高

**次要参与者：**启智机器人、控制、连接组件

### 2.4.4 控制机器人自由避障行走

**主要参与者：**用户

**目标：**控制机器人按照设定速度在预定环境内自由行走指定时间，并可自动避障

**前置条件：**已进入功能选择界面

**启动：**用户选择避障行走模式

**场景：**

1. 系统显示初始速度和运行时间设定窗口以及“确定”按钮

2. 用户输入指定速度以及运行时间

3. 用户点击“确定”按钮

4. 系统显示“运行开始”提示语句

5. 机器人开始行走

6. 机器人遇到障碍可以避开

7. 机器人在指定时间结束后停止运动

8. 用户重新进入选择模式或者退出

**异常情况**：

用户未设置初始参数——出现点：场景中的2， 用户点击“确认”按钮后，系统使用默认的时间和速度参数

**优先级：**中

**何时何用：**第一个增量

**使用频率：** 中

**次要参与者：**启智机器人、控制、连接组件

### 2.4.5 建立地图

**主要参与者：**用户

**目标：**建立当前环境实时地图，为机器人规划路径提供数据

**前置条件：**已进入功能选择界面

**启动：**用户选择路径规划模式

**场景：**

1. 用户按下“建立地图”按钮

2. 系统弹出RVIZ可视界面

3. 用户利用手柄推机器人建立地图

4. 用户点击“保存地图”按钮

**优先级：**中

**何时何用：**第二个增量

**使用频率：** 中

**次要参与者：**启智机器人、控制、连接组件

### 2.4.6 控制机器人规划路径

**主要参与者：**用户

**目标：**建立当前环境实时地图，为机器人规划路径提供数据

**前置条件：**已进入功能选择界面

**启动：**用户选择路径规划模式

**场景：**

1. 用户点击“标定航点按钮”
2. 系统弹出RIVZ界面显示地图
3. 用户在刚刚建立好的地图上表标定航点
4. 用户点击下拉菜单选择目标航点
5. 用户点击“GO！”机器人行走至目标地点
6. 机器人到达目标地点后停止运动，并发出语音提示

**优先级：**中

**何时何用：**第二个增量

**使用频率：** 中

**次要参与者：**启智机器人、控制、连接组件

### 2.4.7 控制机器人识别、抓取特定物体

**主要参与者：**用户

**目标：**机器人识别用户放置的物体，从起始位置到物体放置位置进行抓取

**前置条件：**进入识别抓取、模式

**启动：**用户选择识别、抓取模式

**场景：**

1. 用户点击“标记航点”按钮

2. 用户带着机器人依次标记航点

3. 标记完航点用户说出”stop following”

4. 机器人停止标记航点

5. 用户启动物体识别节点

6. 机器人识别物体

7. 机器人行走至目标位置

8. 机器人调整机械臂抓取物体

9. 机器人发出语音播报“抓取完毕”

10. 用户重新进入选择模式或者退出

**优先级：**中

**何时何用：**第三个增量

**使用频率：** 中

**次要参与者：**启智机器人、机械臂、控制、连接组件

### 2.4.8 遇到错误条件

**主要参与者：**用户

**目标：**通知系统管理员系统出现问题并尽快解决

**前置条件：**机器人出现异常行为时

**启动：**用户发现机器人在任何状态下出现有悖于程序设定的行为

**场景：**

1. 用户联系系统管理员

2. 用户向系统管理员详细描述机器人出现的异常行为以及系统提示的错误信息

**优先级：**低

**何时可用：**存在于任何一个增量

**使用频率：**低

**次要参与者：**系统管理员，启智机器人、机载电脑

### 2.4.9 调试/重新配置系统相关特性并维护硬件

**主要参与者：**系统管理员

**目标：**纠正机器人出现的异常行为并对系统不完善之处进行维护

**前置条件：**机器人出现异常情况且系统管理员收集到问题

**启动：**系统管理员接到用户反馈后

**场景：**

1. 系统管理员记录并分析用户描述情况，初步推测问题成因
2. 系统管理员实地检查机器人部件是否出现损坏、接触不良、电路过载等硬件问题
3. 系统管理员用管理员权限登陆系统，检查代码是否存在漏洞
4. 系统管理员根据用户描述还原使用场景，观察用户报告的问题是否依然存在，是则返回2,否则结束调试

**优先级：**低

**何时可用：**任何一个增量

**使用频率：**低

**次要参与者：**用户、启智机器人、机载电脑

# 数据库设计

在数据库设计中，首先对逻辑结构设计，得到E-R图，之后基于关系数据模式，对物理模型进行设计，创建了数据表结构、数据项以及关系等，并为了得到理想的数据结构，将数据模型规范到第三范式。

## E-R图



图3-1 系统E-R图

## 数据库关系模式

### 数据表及数据项介绍

**表格4 用户使用关系表**

|  |  |  |  |
| --- | --- | --- | --- |
| 用户使用关系 | | | |
| 用户ID | 用户密码 | 用户权限 | 可选功能 |
| 唯一标识用户的数据项，也是该关系的主码 | 用户自行设定 | 用户可以有不同的权限 | 1.自由行走，避障  2.路径规划  3.识别抓取物体 |

**表格5 功能实现关系表**

|  |  |
| --- | --- |
| 功能实现关系 | |
| 功能项 | 使用设备 |
| 功能项是该关系的主码，标识不同的功能 | 不同的功能对应到不同的设备，依照具体功能而定 |

**表格6 控制关系表**

|  |  |  |
| --- | --- | --- |
| 控制关系 | | |
| 控制设备 | 接收设备 | 实现功能 |
| 机器人软件 | 控制面板/机械臂/底盘传感器 | 在控制面板上交互/实现机械臂移动/对路面环境检测 |

**表格7 测距关系表**

|  |  |  |  |
| --- | --- | --- | --- |
| 测距关系 | | | |
| 当前时间 | 控制设备 | 接收设备 | 距离大小 |
| 唯一标识测距关系数据项的属性，主属性 | 激光雷达传感器 | 机器人软件 | 激光雷达传感器用于测量机器人距离障碍物的距离矢量，传输给机器人软件，用于数据的分析处理。 |

**表格8 物体识别关系表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 物体识别关系 | | | | | |
| 当前时间 | 控制设备 | 接收设备 | 俯仰角度参数 | 需要识别的物体 | 三维点云数据 |
| 唯一标识测距关系数据项的属性，主属性 | Kinect2传感器 | 机器人软件 | Kinect2传感器工作需要的一些参数数据 | 需要利用Kinect2传感器识别的物体 | Kinect2传感器得到的三维点云数据，传输到机器人软件用于识别 |

**表格9 物体抓取关系表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 物体抓取关系 | | | | |
| 当前时间 | 控制设备 | 接收设备 | 上升高度 | 手爪闭合宽度 |
| 唯一标识测距关系数据项的属性，主属性 | 机器人软件 | 机械臂 | 机械臂抓取物体需要的上升高度数据 | 机械臂抓取物体需要的手爪闭合宽度数据 |

## 关系模式规范化

为消除冗余，避免部分函数依赖和传递函数依赖，我们将关系模型规范为第三范式要求。

### 3.3.1 用户使用关系

**主码：**用户ID

**非主属性：**用户密码，用户权限和可选功能

非主属性完全依赖于主码，达到第三范式要求。

### 3.3.2 功能实现关系

**主码：**功能项

**非主属性：**使用设备

非主属性完全依赖于主码，达到第三范式要求。

### 3.3.3 控制关系

**主码：**接受设备

**非主属性：**实现功能

非主属性完全依赖于主码，达到第三范式要求。

### 3.3.4 测距关系

**主码：**当前时间

**非主属性：**距离大小

非主属性完全依赖于主码，达到第三范式要求。

### 3.3.5 物体识别关系

**主码：**当前时间

**非主属性：**俯仰角度参数、需要识别的物体、三维点云数据

非主属性完全依赖于主码，达到第三范式要求。

### 3.3.6物体抓取关系

**主码：**当前时间

**非主属性：**上升高度、手抓闭合宽度

非主属性完全依赖于主码，达到第三范式要求。

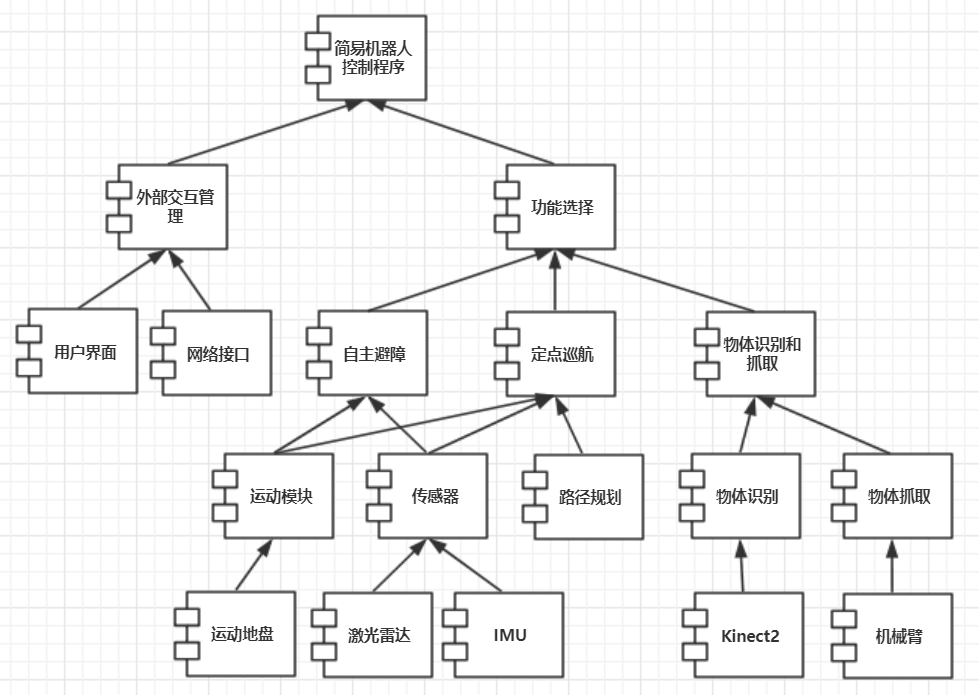
规范化之后的关系模式图：



**图3-3 物体抓取图**

# 体系结构设计

## 总体结构

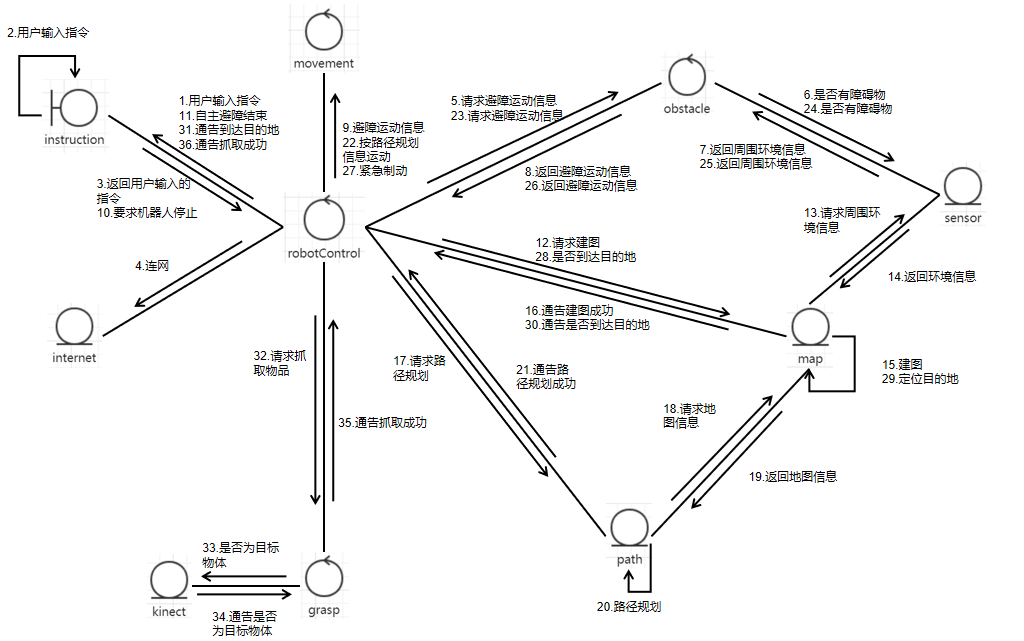


**图4-1 系统总体结构构件图**

简易机器人执行者通过网络接口实现互联网通信，通过用户界面选择机器人功能，并设置相应参数。执行者可指定机器人完成三种功能：自主避障、定点巡航，以及物体抓取。自主避障模块需要运动模块和传感器模块支持。传感器模块下设激光雷达模块和IMU模块。定点巡航需要传感器模块、运动模块以及路径规划模块支持。首先调用创建好的地图进行路径规划，然后调用运动模块控制机器人的行走，机器人运行过程中，不断监视传感器模块返回的信息，若出现障碍物，则通知行走控制模块紧急制动。同时监视机器人当前位置，若已达地图中的目标点，则停止运动。机器人还可完成目标识别与抓取，通过Kinect2视觉传感器探测出目标物体，然后机械臂完成目标物体抓取。

## 软件体系结构

### 4.2.1软件总体结构



**图4-2 软件协作图**

共抽象出10个类：

instruction类与用户交互，拟定为边界类；

robotControl类负责机器人各模块总体调度控制，拟定为控制类；

internet类是连网组件，拟定为实体类；

movement类负责机器人运动控制，拟定为控制类；

obstacle类负责机器人自主避障模块，拟定为控制类；

sensor类是传感器，拟定为实体类；

grasp类负责控制物体抓取，拟定为控制类；

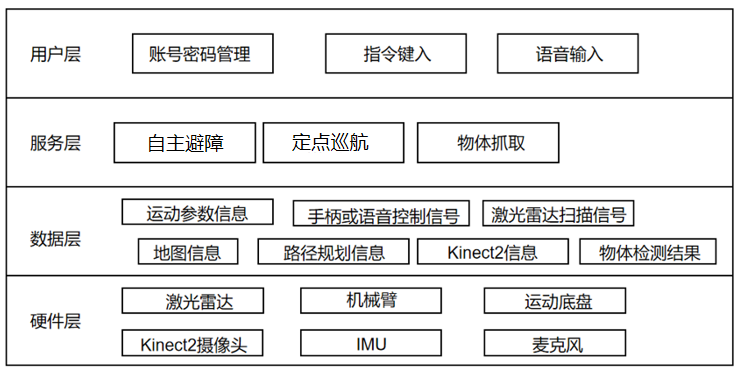
kinect类表示kinect相机，拟定为实体类；

path类存储路径规划时的路径信息，拟定为实体类；

map类存储地图信息，拟定为实体类；

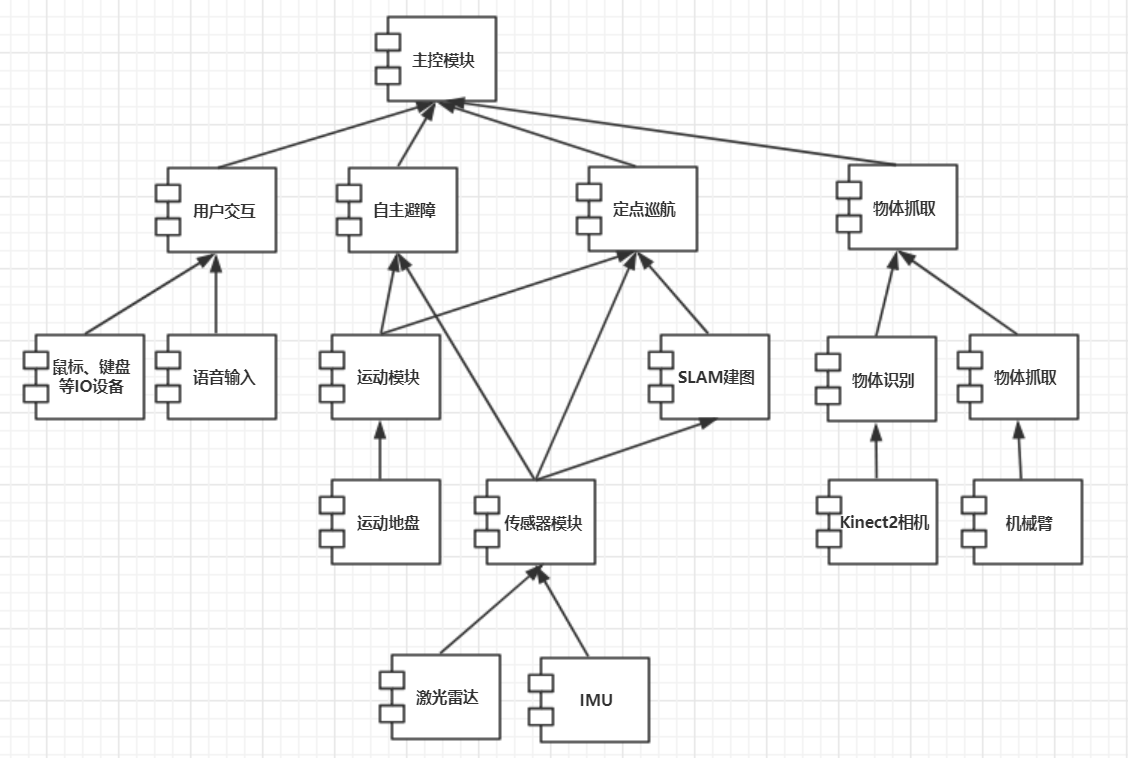
系统启动后，robotControl向instruction发出指令，请求用户输入指令；用户从输入指令后，指令传回robotControl存储；若需要连网，robotControl发出指令请求internet类连网；初始指令输入后，robotControl根据用户选择的工作模式向不同模块发出指令：若用户选择自主避障，robotControl向obstacle类请求避障运动信息；obstacle类向sensor询问周围是否有障碍物；sensor返回周围的障碍物信息；obstacle类根据sensor的反馈决定机器人下一步如何运动，然后将运动信息返回给robotControl；robotControl将运动信息反馈给movement模块；若运动时间到达上限或者用户要求机器人停止，则机器人停止运动，robotControl告诉用户自主避障结束；若用户选择定点巡航，robotControl给map类发信息，请求map类创建地图；map类收到指令后，向sensor请求周围环境信息；sensor返回周围环境信息；地图创建成功后，map通告robotControl地图创建成功；robotControl向path类发出信息请求进行路径规划；path类收到消息后，向map类请求地图信息；map类返回地图信息；path进行路径规划，成功后通告robotControl；robotControl根据路径规划出的行进路线，向movement模块发出指令，控制机器人运动；同时robot向obstacle类询问机器人周围的障碍物信息；obstacle类向sensor请求周围环境信息；sensor类返回周围环境信息；obstacle类向robotControl通告障碍物信息；若存在障碍物，robotControl模块通告movement紧急制动；robotControl询问map类机器人是否已经运动到目标地点；map类定位机器人位置信息，向robotControl通告是否已经到达目的地；若已经到达目的地，robotControl通知instruction已经到达目的地；若用户选择物体抓取模式，则robotControl通知grasp模块进行物体抓取；grasp类询问kinect类当前物体是否是目标物体；kinect返回相关信息；若找到目标物体，grasp控制机械臂进行抓取，抓取成功后，通告robotControl类；robotControl通知instruction物体抓取成功。

## 硬件体系结构



**图4-3-1 硬件层次结构图**

硬件层次结构图用户的角度向下阐述到基本的硬件。整个系统可以看作四层，最上层为用户层，主要用于管理用户账号，以及键盘io或语音等指令输入。第二层为服务层，包括自主避障、定点巡航和物体抓取，根据用户层输入的指令制定对应的服务。第三层是数据层，主要存储运动参数信息、手柄或语音控制信号、激光雷达扫描信号、地图数据、路径规划信息、Kinect2信息和物体检测结果等。最下层是基础元件，主要有激光雷达、机械臂、运动低盘、双目摄像头、IMU和麦克风。



**图4-3-2 硬件构件图**

用户和管理员在使用简易机器人时主要通过鼠标、键盘等IO设备或语音设备输入指令，指令经过解析加载到主控模块，主控模块向机器人发出指令，机器人通过传感器和运动底盘完成自主避障，或通过运动底盘、激光雷达、SLAM模块完成路径规划和移动，或通过Kinect2相机、机械臂等硬件完成目标物的识别和抓取任务。

## 技术体系结构



**图4-4 技术体系结构图**

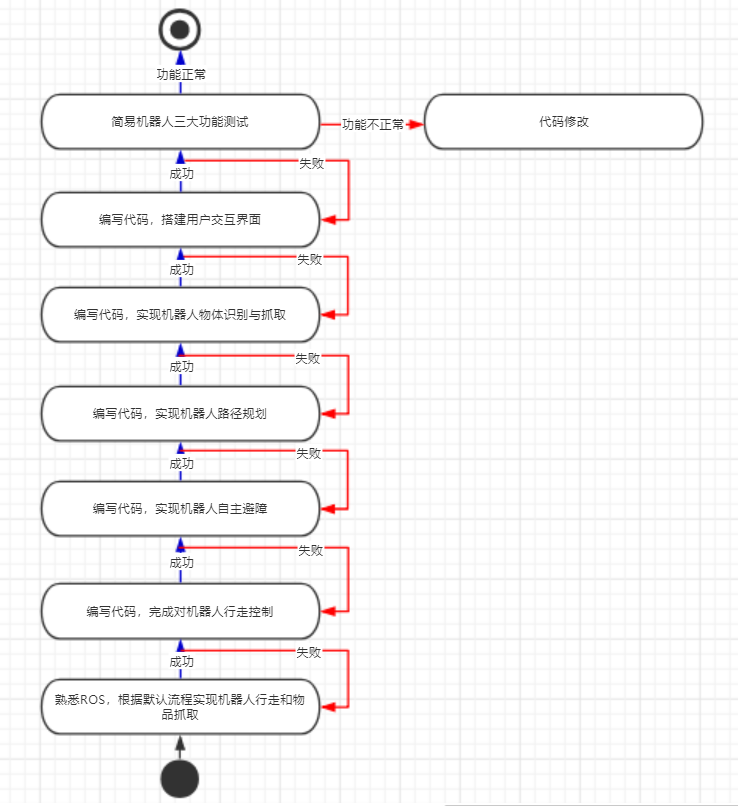
完成简易机器人自动避障、路径规划、物体识别与抓取等功能需要众多技术支持。开发技术允许简易机器人开发人员在Roboware Studio等软件上编写相关代码，实现对机器人的控制；用户交互技术允许机器人从用户处获得指令；探测与视觉技术帮助机器人完成障碍物探测、目标物体识别和地图绘制；ROS操作系统支持机器人运动、控制机械臂、进程调度等。

## **支撑体系结构(部署和实施方案)**



**图4-5-1 系统部署试图**

简易机器人系统主要由软件、PC、Wi-Fi网络和机器人硬件系统部署而成。软件部分基于Ubuntu系统，ROS系统提供机器人的运行环境，Roboware studio用于编写支持机器人完成相关功能的代码，UI界面则负责与用户交互，这三者并列于ROS系统之上。PC端通过USB-HUB与启智机器人连接，扩展PC端USB为多路接口，连接Web和启智控制器，启智控制器控制伺服电机，用以完成机械控制和PC双向电信号交互。USB-HUB同时与电池模块、传感器、电源管理板和相机连接。



**图4-5-2 实施方案流程图**

采用迭代-增量开发模式，复杂功能依赖于简单功能的实现，在简单模块的基础上进行增量开发，最终实现机器人三大功能。

在开始编写简易机器人代码前，熟悉ROS系统以及机器人的简单行走控制以及物体抓取控制。熟悉ROS机器人的基本操作后，编写简单的机器人行走控制模块，实现对机器人行走的控制，然后熟悉传感器的相关操作，编写代码实现机器人自主避障，成功后熟悉SLAM模块和障碍物探测模块，实现简易机器人路径规划，成功后熟悉Kinect2模块，完成简易机器人目标物体识别与抓取，成功后编写UI界面，实现与用户的交互。最后测试机器人三大功能，若成功，则软件开发结束，否则进行代码调试。

# 接口设计

## 对外用户调用接口

这一部分接口主要面向用户提供，由用户调用使机器人在用户指令下执行正常功能。由于我们的大部分用户不具有充足的计算机与机器人设计相关知识，因此我们考虑采用图像化界面包装此部分接口，使用户可能通过方便直观的“点击按钮”“输入指令”的方式控制机器人的运行和其他功能，而不需要向机器人管理员等底层开发者一样在程序、在软硬件交互层面对接口进行调用。具体而言，我们根据用户会用到的“自主避障”“定点巡航”“物体抓取”3个功能，分别对对应的接口进行“包装”似调用，并构造基于电子计算机的图形化界面；用户点击图像化界面的按钮，就可以方便快捷地调用3个主要功能下属小功能；通过在对应输入框输入数据或指令，就向机器人发送指令，与机器人进行互动。

亲爱的用户朋友您好！欢迎使用由“智慧少女不头秃”小组设计制造的智慧导航与抓取机器人！

下面请选择您期望使用的功能：

物体抓取

定点巡航

自主避障

**图5-1-1 交互界面**

### “自主避障”功能相关用户接口

用户启动基于计算机的图像化界面，进入主界面，点击“自主避障”按钮，可以进入“自主避障”模式（这是用户使用机器人的3大基本功能之一），进入新的图像化界面如下，输入运动速度和运动时间，机器人可以在室内环境下自由行走，遇到障碍物则自动调整方向，继续前进。下面是输入参数界面：

正在进行自主避障！当前机器人位置为：X: 0 Y: 0

请输入如下参数：

速度参数 X:\_\_\_m/s Y:\_\_\_\_m/s (注意：机器人各速度不能大于2m/s)

运动时间：\_\_\_\_\_\_\_\_分钟

**图5-1-2 自主避障输入界面**

输入参数完成后，机器人开始运动，运动界面如下：

机器人正在根据指令运行，请稍候！

当前机器人位置为：X: 0 Y: 0



**图5-1-3 自主避障运行界面**

**图5-1-3 自主避障运行界面**

当机器人运动时间达到用户设定的运动时间上限时，则结束自主避障功能并在5s后自动返回选择3大主要功能的界面。

已经完成自主避障！

当前机器人位置为：X: 0 Y: 0

即将自动跳转进入三大主要功能选择界面！

跳转倒计时：5 s……



**图5-1-4 自主避障结束界面**

### 5.1.2 “定点巡航”功能相关用户接口

用户启动基于计算机的图像化界面，进入主界面，点击“定点巡航”按钮，可以进入“定点巡航”模式（这是用户使用机器人的3大基本功能之一）。初次进入定点巡航功能，需要进行基本建图操作。新的图像化界面如下：

正在进行基本建图！当前机器人位置为：X: 0 Y: 0

请选择以下两种模式之一指挥机器人运行，在运行中收集地图信息：

模式一：请输入目标导航点坐标

模式二：请输入导航方向与运动距离：



选择模式一

选择模式二

**图5-1-5 导航选择界面**

用户可以根据自身喜好和实际需求任意选择“模式一”或“模式二”，以不同的方式对机器人的以完成建图为目的的运行进行指挥；在点击“模式一”或“模式二”后，用户会进入不同的新界面。

如果选择点击“模式一”，需要在新界面中输入期望机器人运行到的地点的XY左边（同时给出机器人此时的XY坐标作为参考）。输入完毕后，点击“输入完毕，确认发送按钮”，发送指令到机器人。

正在进行基本建图！当前机器人位置为：X: 0 Y: 0

你已选择模式一！

请输入目标导航点坐标：X: Y:



输入完毕，确认发送

**图5-1-6 输入模式一**

如果选择点击“模式二”，需要在新界面中输入期望机器人进行运行的方向和运动距；方向必须是N、S、W、E中的一个，同时给出机器人此时的XY坐标作为参考。输入完毕后，点击“输入完毕，确认发送按钮”，发送指令到机器人。

正在进行基本建图！当前机器人位置为：X: 0 Y: 0

你已选择模式二！

请输入导航方向： 请输入运动距离：



输入完毕，确认发送

**图5-1-7 输入模式二**

无论选择模式一还是模式二，在点击“输入完毕，确认发送”后，都会进入以下界面并实时返回机器人坐标，直至机器人完成之前输入的运动指令。

机器人正在根据指令运行，请稍候！

当前机器人位置为：X: 0 Y: 0



**图5-1-8 开始界面**

当机器人完成当前运动指令，程序自动判断当前地图是否完整，如果不完整，回到选择基本建图“模式一”“模式二”的界面；如果地图已经完整，就显示如下界面，并在5s后自动返回“定点巡航”的界面。

已经完成基本建图！

当前机器人位置为：X: 0 Y: 0

即将自动跳转进入“定点巡航”界面！

跳转倒计时： 5 s……



**图5-1-9 建图完成界面**

回到“定点巡航”模式后，进入新的图像化界面如下：

正在进行定点巡航！当前机器人位置为：X: 0 Y: 0

请输入目标导航点坐标：X: Y:



输入完毕，确认发送

**图5-1-10 巡航输入界面**

在新界面中输入期望机器人运行到的地点的XY坐标（同时给出机器人此时的XY坐标作为参考）。输入完毕后，点击“输入完毕，确认发送按钮”，发送指令到机器人。机器人根据期望巡航点坐标自动进行路径规划和运动，并显示如下界面：

机器人正在根据指令运行，请稍候！

当前机器人位置为：X: 0 Y: 0



**图5-1-11 巡航运行界面**

当机器人完成当前运动指令，显示如下图像化界面，并在5s后自动返回选择3大主要功能的界面。

已经完成定点巡航！

当前机器人位置为：X: 0 Y: 0

即将自动跳转进入三大主要功能选择界面！

跳转倒计时： 5 s……



**图5-1-12 巡航结束界面**

### 5.1.3 “物体抓取”功能相关用户接口

用户启动基于计算机的图像化界面，进入主界面，点击“物体抓取”按钮，可以进入“物体抓取”模式（这是用户使用机器人的3大基本功能之一），进入新的图像化界面如下：

正在进行物体抓取！当前机器人位置为：X: 0 Y: 0

请输入目标抓取点坐标： X: Y:

请读入待抓取物体识别用图：

浏览图片

输入完毕，确认发送

**图5-1-13 抓取选择界面**

在新界面中输入期望机器人运行到的地点的XY坐标（同时给出机器人此时的XY坐标作为参考）和希望抓取物体的识别用图（点击“浏览图片”按钮读入）。输入完毕后，点击“输入完毕，确认发送按钮”，发送指令到机器人。机器人根据期望抓取地点坐标自动进行路径规划和运动，并显示如下界面：

机器人正在根据指令运行，请稍候！

当前机器人位置为：X: 0 Y: 0



**图5-1-14 巡航运行界面**

由于“物体抓取”要求较高的运动精度，所以在完成指令后，显示如下图像化界面的同时机器人根据当前自定位得到的XY坐标和之前用户输入的期望XY坐标进行比较分析，判断是否存在位置误差，如果存在，则以较低的运动速度运动以调整机器人当前位置，直至自定位XY坐标和期望XY坐标重合。

已经完成巡航运动部分！

正在对当前位置进行检测和矫正，请稍候！



**图5-1-15 巡航完成界面**

当自定位XY坐标和期望XY坐标重合，显示以下界面，机器人通过相机拍摄周围物体，于之前浏览的“待抓取物体识别用图”比对，识别出需要抓取的物体，由此确定其具体位置。

已经到达预定抓取目标位置！

正在搜寻待抓取物体，请稍候！



**图5-1-16 搜寻物体界面**

当完成了对需要抓取的物体的识别及其具体位置的确认。对进行第一次抓取物品的尝试。每完成一次抓取动作，就判断是否抓取到了物体，如果没有抓取到，就自动进行调整并重复进行抓取，直至抓取成功。

已经搜索到待抓取物体！

正在进行抓取，请稍候！



**图5-1-17 抓取运行界面**

当机器人抓取物体成功后，向用户显示如下图像化界面，并在5s后自动返回选择3大主要功能的界面。

已经完成目标抓取！

当前机器人位置为：X: 0 Y: 0

即将自动跳转进入三大主要功能选择界面！

跳转倒计时： 5 s……



**图5-1-18 抓取完成界面**

## 对内硬件接口

我们使用的是启智ROS机器人，搭载的硬件设备主要是多种用于接受、识别外界信号的传感器和用于机器人自身运动的底盘。其中，主要使用以下三种传感器：激光雷达、姿态测量系统、立体相机。

### 5.2.1 激光雷达

启智 ROS 机器人的底盘上安装了一枚红外激光雷达，该雷达的扫描角度为 360°，能够很高效的检测出周围的障碍物分布，并可以通过 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。

启智 ROS机器人的底座位置还安装了一枚激光测距雷达，它可以提供0.15米-2米的测距范围、360°的扫描角度、小于实际距离的1%的测距分辨率和0.9度的角度分辨率，可以根据获取的深度信息分析出周围障碍物的分布，从而获得较高精度的深度信息，帮助我们的机器人获取场地信息构建地图。

### 5.2.2 姿态测量系统

启智 ROS 机器人的底盘内置了一个三轴姿态测量系统，可以实时检测机体的朝向、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风险，及时采取紧急措施。

### 5.2.3 立体相机

启智 ROS 机器人的头部安装了一台 RGB-D 立体相机，我们以物理机械的方式调整了这个相机的安装位置和视角，以便它对准需要进行视觉识别的目标区域。该立体相机可以输出 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识别和定位，从而用于识别和抓取目标物体。机器人检测到目标物体后，根据其他传感器获取的信息（例如深度、高度、角度等），对机械臂的行为进行决策，以便抓取目标物体。

### 5.2.4 机器人运动底盘

启智 ROS 机器人采用了三轮全向式移动底盘，相比传统的双轮差动底盘，拥有更多的自由度。全向底盘可以在不改变朝向的情况下往水平面上的任何方向移动，这在进行目标跟踪和运动避障时，可以减少机体位置调整的步骤，减少调节时间，提高执行效率。

## 对内软件接口

在设计软件时，我们使用了面向对象的设计思路，将整个软件按照各个不同的功能分为多个类，或者说，多个模块（详细信息请见第三部分的“类图”部分）；不同的类/模块通过通信或者相互调用的方式进行工作，因此我们需要构建相应的接口，以便不同类的协作。下面将介绍主要的软件内部接口。

### 5.3.1 用户交互指令部分——Instruction类

Instruction类主要负责与用户进行交互，实现机器人对用户指令的读取，它可以在“自主避障”“定点巡航”“物体抓取”中被机器人控制类RobotControl类调用，读取用户输入的机器人控制指令（主要是预定到达地点的XY坐标，或者期望运动方向+期望运动距离），并将结果返回到机器人控制类RobotControl类。

### 5.3.2 机器人总体调度部分——RobotControl类

RobotControl类主要实现对机器人各模块的总体调度与总体功能的实现，它具有Model1、Model2、Model3、Model4四个模块，可以在组合后完成面向用户的“自主避障”“定点巡航”“物体抓取”三个功能。Control类可以调用以上所有类以完成对应的功能，也提供stop()、rerun()等接口与其他类配合完成功能。下面就具体的Model1、Model2、Model3、Model4三个模块进行叙述：

Model1是避障行走模块，将会调用goAhead()、turn()来控制机器人的行走并且周期性调用hasFrontObstacle功能检测前方障碍物情况；

Model2是地图构建模块，将会调用goAhead、turn来控制机器人的行走并且周期性调用buildMap()功能提取当前位置场地信息加入已有地图中进行构建；

Model3是导航模块，会调用buildMap()、getMap()、newPoint()、savePoint()来构建当前地图，并且根据用户的选择使用initWay()并且调用goAhead()、turn()来控制机器人生成导航路径、根据导航路径行走。

Model4是抓取模块，会调用自己的setPoint()来标定预设点位，并且循环调用detect\_object直到检测到目标物体；检测到物体后将会调用goAhead()、turn()来将机器人运动到预设点位，调用operation来操作机器人抓取目标物体。

总结一下，“自主避障”功能对应Model1，“定点巡航”功能对应Model1、Model2和Model3，“物体抓取”功能对应Model1、Model3和Model4。

### 5.3.3 连接网络部分——Internet类

Internet类主要实现机器人连网，使得机器人可以借助网络完成内部信息的上传和远程下载数据。它可以被机器人总体调度类RobotControl类调用，在机器人需要连接网络时实现机器人与网络的连接。

### 5.3.4 机器人运行控制部分——Movement类

Movement类主要实现对机器人运行的控制，它实现了goAhead(float,float)和turn(float)两个接口，分别实现了机器人的行走和转向功能。该类可以在“自主避障”“定点巡航”“物体抓取”中被机器人控制RobotControl类调用，以实现对机器人运动的控制。

### 5.3.5 机器人自主避障部分——Obstacle类

Obstacle类主要实现对机器人运动方向上障碍物的识别和自主避障，当识别出障碍物后，它会调用机器人控制类的stop()接口，使机器人停止运行，并不断调用自身对前方障碍物进行检测，直至检测不到前方障碍物，再调用机器人控制类的rerun()接口，使机器人继续运行。在机器人运行过程中，RobotControl类不断向Obstacle类询问机器人周围的障碍物信息；Obstacle类向Sensor类请求周围环境信息；Sensor类返回周围环境信息；Obstacle类向RobotControl类通告障碍物信息；若存在障碍物，RobotControl类通告Movement紧急制动；

### 5.3.6 传感器部分——Sensor类

Sensor类主要实现对传感器的调用，通过传感器收集信息，并读取出这些数据。Sensor类在构建地图时被Map类调用，返回传感器收集到的周围环境信息；在机器人运动过程中被Obstacle类，返回传感器收集到的周围环境信息，用以判断机器人运动方向上是否存在障碍物。

### 5.3.7 物体抓取部分—— Grasp类

Grasp类主要实现物体抓取功能，它下有Object\_detector和Grabber类，分别有对外接口detect\_object、operation、getPos等接口，实现探测、操作等功能。上述接口主要被“物体抓取”功能的类调用以实现物品识别、抓取的任务。在完成“物体抓取”时，RobotControl类通知Grasp类进行物体抓取；Grasp类询问Kinect类当前物体是否是目标物体；Kinect类返回相关信息；若找到目标物体，Grasp控制机械臂进行抓取，抓取成功后，通告RobotControl类。

### 5.3.8 物体识别部分——Kinect类

Kinect类主要实现对相机的调用，通过相机拍摄图像，并读取拍摄到的图像。在完成“物体抓取”时，Grasp类询问Kinect类当前物体是否是目标物体，Kinect类通过拍摄的图像结果与目标物品比较，返回相关信息。

### 5.3.9 路径规划与存储部分——Path类

Path类主要实现对机器人运动路径的规划与路径的存储。它提供了planPath(float)和getPath(float)两个接口，分别实现了机器人运动路径的规划（并存储）和读取（读取之前规划后存储的路径）。该类可以在“定点巡航”“物体抓取”中被机器人控制类调用，以实现对机器人运动的路径规划。比如，在进行“定点巡航”时，Path类收到RobotControl类发出的请求信息，进行路径规划；在进行路径规划时，Path类向Map类请求地图信息，根据Map类返回的地图信息进行路径规划，在成功后通告RobotControl类。

### 5.3.10 地图构建与存储部分——Map类

Map类主要实现对当前场地地图的构建和存储，它实现了buildMap()和getMap()两个主要对外接口，分别用于实现构建和获取地图。该类可以被诸多类调用，比如在用户选择进行基本建图时被RobotControl类调用，进行地图的常见与存储；在进行建图时，Map类调用Sensor类获取周围环境信息；地图构建成功后，Map类通告RobotControl类地图创建成功。

# 详细设计

## 机器人运动模块

**输入：**期望的机器人运动线速度和角速度

**输出：**机器人运动控制消息（以一定比特率输出转化的速度信息，用来控制伺服电机）

**功能：**控制机器人，使机器人按照符合用户指令的方式进行运动

**类图：**

****

**图6-1 运动模块类图**

这个模块中有6个控制运动的运动的消息：linear.x, linear.y, linear.z , angular.x, angular.y，angular.z。

linear.x, linear.y, linear.z控制机器人运动的线速度，分别代表三维坐标轴的x、y、z三个方向，其中linear.x表示机器人前进方向的线速度；

angular.x, angular.y, angular.z控制机器人运动的角速度，也就是控制机器人旋转的速度。该模块有一订阅者，订阅”/cmd\_vel”话题，如果接收到速度消息，则调用callback函数向底层运动硬件发送转化过的速度消息，机器人可以运动。

需要注意的是，用户指令并不直接给出期望的机器人运动线速度和角速度，而给出的是目标巡航地点、或者运动方向和运动距离。机器人控制模块在接收到用户指令后，会在进行路径规划的同时，将用户指令给出的运动信息结合实际情况（包括当前场地实际情况，机器人硬件条件，机器人当前坐标和朝向等），转换为合适的期望线速度、角速度，并通过机器人运动模块使机器人进行对应的运动。

## 机器人避障模块

**输入**：雷达探测机器人前进方向是否存在障碍物的探测结果

**输出：**机器人是否停止运行，是否重新开始运行及重新开始运行后的角速度、线速度

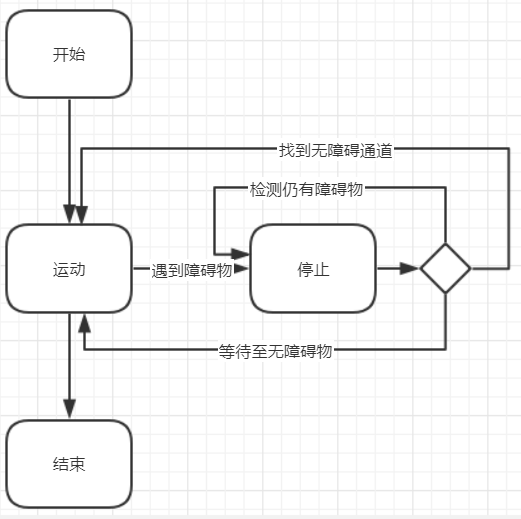
**功能：**实现机器人自主避障功能，并且实现在运动过程中的避障，在遇到障碍物后寻找无障碍通道，或者及时停止运行并在障碍物挪走后恢复原本的运行方式

**类图：**

****

**图6-2-1 避障模块类图**

**活动图：**



**图6-2-2 避障模块活动图**

避障模块在机器人运行过程中被时时调用。它接收雷达的距离数组range信息，触发autonomous\_behave()回调函数对雷达信息进行处理：首先，调用check\_obstacle()函数判断当前通道内是否有障碍物：如果存在障碍物，则调用find\_path()函数尝试寻找无障碍物通道，如果找到无障碍通道，则及时发布角速度信息和线速度信息，由Movement模块接收，使机器人按无障碍通道运行从而避开障碍物；如果没有找到无障碍通道，则保存当前线速度、角速度信息，再发布角速度、线速度均为0的信息，使机器人停止运行，并在此后持续不断地接收雷达的距离数组range信息，触发autonomous\_behave()回调函数对雷达信息进行处理，调用check\_obstacle()函数判断当前通道内是否有障碍物，当判断障碍物不存在时，及时恢复之前保存的线速度、角速度信息，使机器人按照停止前的方式继续运动。

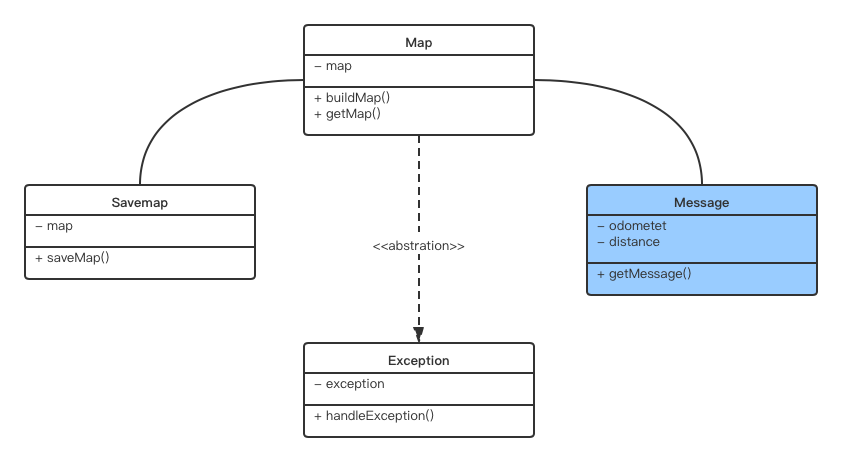
## 地图构建与存储模块

**输入：**机器人自定位信息、传感器（红外激光雷达）采集的当前位置场地信息

**输出：**构建完成的完整场景地图

**功能：**完成“定点巡航”中的基本建图功能，在运动指令下探测运行中当前位置的场景信息，以此构建场地地图，保存完整地图并提供读出接口，为“定点巡航”后续部分和“物体抓取”功能做准备

**类图：**

****

**图6-3-1 地图构建存储模块类图**

这个模块包括Map、Savemap、Message和Exception四个部分。

Map部分主要包括buildMap()和getMap()两个函数。BuildMap()函数主要用于完成实时地图的构建，具体是通过使用gampping包根据激光数据（或者深度数据模拟的激光数据）构建地图；getMap()函数可以被其他部分调用，作用是返回构建好的地图。

Savemap部分主要包括savemap()函数，用于将构建好的实时地图以map.pgm和map.yaml的格式保存，以备后续使用。

Message部分主要包括getMessage()函数，主要用于提供机器人的测位信息和传感器信息来对机器人进行定位，以实现SLAM建图。

Exception部分主要包括handleException()函数，主要用于在异常情况下的进行分析、处理和远程向机器人管理员汇报。

**活动图：**

****

**图6-3-2 地图构建存储模块活动图**

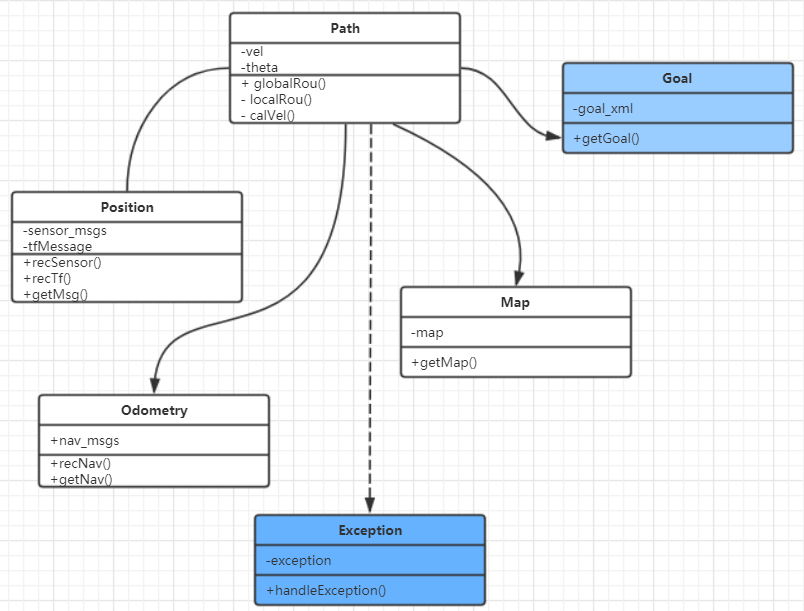
## 路径规划模块

**输入：**测位信息（速度+角度）、坐标变换关系、传感器信息、地图、目标坐标

**输出：**机器人速度（V）、角度（θ）

**功能**：根据用户输入的运动指令对机器人路径进行规划（包括全局路径和局部路径），并控制机器人以特定速度和方向到达目的位置

**类图**：



**图6-4-1 定点巡航类图**

这个模块包括Path、Position、Map、Odometry和Exception六个部分。

Path部分主要包括globalRou()、locaRou()和calVel()三个函数，这个类主要实现了根据输入的信息进行路径规划，vel和theta是路径规划之后需要计算出的机器人下一步速度与角度。globalRou()函数主要用于规划全局路径，即根据costmap利用A\*等算法建立出大致路径；locaRou()函数主要用于规划局部路径，机器人根据全局路径截取出局部路径并确定目标地点，之后根据姿态信息、定位信息算出多条局部路径，根据一定的评优准则选取最优的一条作为机器人下一步走的轨迹；calVel()函数主要确定路径规划之后机器人下一步运行对应的速度与角度。

Position部分主要包括recSensor()、recTf()和getMsg()三个函数，主要实现了在规划路径的过程中调用amcl包和激光雷达等传感器对机器人的位置做出估计（即sensor\_msgs），然后利用tf树得到机器人的坐标转换信息tfMessage，将全局坐标转换为以机器人为中心的坐标。

Odometry部分主要包括recNac()和getNav()两个函数，主要实现了规划路径的过程中还需要根据测位信息nav\_msgs对机器人的速度和角度采样，并根据这些信息对多条路径进行预测。

Goal部分主要包括getGoal()函数，其功能是提供目标地点的xml数据。

Map部分主要包括getMap()函数，其功能是提供建立好的场景地图。

Exception部分主要包括handleException()函数，主要用于在异常情况下的进行分析、处理和远程向机器人管理员汇报。

**活动图：**

****

**图6-4-2 路径规划活动图**

## 物体识别模块

**输入：**摄像头拍摄的真实世界物体的图像

**输出：**进行物体抓取时机器人应当处在的位置

**功能：**识别输入图像中的物体，与数据库中的资料进行比对，得出图像中物体对应的坐标，计算出机器人应当运动到哪里进行物体抓取

**类图：**



**图6-5-1 物体识别模块类图**

**活动图：**



**图6-5-2 物体识别模块活动图**

首先，在真正应用识别程序对物体进行识别前，需要预先对摄像头进行标定。这是由于通过摄像头拍摄的图片进行物体识别对摄像头拍摄图片的精密性和准确率要求较高，而摄像头这种光学仪器本身由于自身设计缺陷与外界光学污染等原因存在图像的畸变情况，因此，为了防止生成的待识别图像出现扭曲或变形，我们选择在拍摄图像前先对摄像头进行标定，以减小在拍摄时出现误差的可能性和误差的大小，尽量避免在物体识别模块由于数据源造成的误差和错误。

我们实现的物体识别模块的使用启智ROS机器人携带的立体相机拍摄真实世界的待识别物体的图像作为模块的输入；Kinect类通过相机拍摄得到图片信息（/image\_color），并将其转化为物体的位置和种类信息，再交给calculator模块；calculator模块通过比较物体之间的相对位置，计算出要抓取物体对应的位置，作为整个物体识别模块的计算结果输出。

## 物体抓取模块

**输入：**物体识别模块计算出的应当进行物体抓取的位置

**输出：**对机器人底盘的运动指令，使机器人到达应当进行物体抓取的位置；对机械臂进行操纵的指令，使机械臂做出动作，抓取到物体

**功能：**使机器人按照物体识别模块的结果，到达应当进行物体抓取的位置；对机械臂进行操纵，使机械臂做出动作，抓取到物体

**类图：**

****

**图6-6-1 物体抓取模块类图**

**活动图：**



**图6-6-2 物体抓取模块活动图**

物体抓取模块主要包括move2waypoint()和grab()两个函数：move2waypoint()根据输入的物体识别模块计算出的应当进行物体抓取的位置，与机器人控制模块进行交互，使机器人运动到相应的位置；grab()根据预设的参数调整机械臂的姿态，控制机器人机械臂进行抓取，并实时检测抓取动作是否成功，以决定是否重复进行抓取，直至完成抓取任务。

# 运行与开发环境

## 运行环境

系统运行时，应有达到此配置的计算机三台以上：

**表格10 运行环境**

|  |  |
| --- | --- |
| 硬件 | 性能 |
| 内存 | 大于8GB |
| 硬盘 | 大于100GB |
| 启智ROS机器人 | 一台 |

启智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。所处环境地面应该坚硬、平滑，工作平面能够承载不小于 40kg 的重量。如果表面太软，则机器人可能卡住，运动受阻。我们建议在商用地毯、瓷砖等材质的环境下运行机器人。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。同时，启智 ROS 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。在工作温度上，启智ROS机器人设计的工作温度为15℃到35℃之间，使用中务必远离明火和其他热源。

## 开发环境

启智ROS机器人执行指令依赖的软件环境

**表格11 软件环境**

|  |
| --- |
| 软件环境 |
| ubuntu16.04 |
| kinetic版ROS |
| libfreenect2 |
| IAI\_Kinect2 |
| RPLIDAR ROS |
| 讯飞语音识别工具 |
| 导航地图工具 |
| RoboWare Studio 等编写启智ROS机器人代码的IDE |
| windows7以上支持linux环境的虚拟机或双系统 |
| CMake编译环境 |
| 3D可视化工具Rviz |
| ROS源代码 |

# 需求可追踪性说明

## 功能需求

### 自主避障功能

1. 功能模块：Movement类、GUI类、Obstacle类、robotControl类、Instruction类和Exception类。
2. 主要流程：用户通过界面操作选择自主避障功能，相应GUI类传递信息给交互接口Instruction类，Instruction类将信息传递给robotControl类。robotControl类实时调用Obstacle类判断前方是否有障碍物，将返回的结果传递给Movement类指导机器人在室内环境下自行移动，直至达到用户指定的运动时间。

### 定点巡航功能

1. 功能模块：GUI类、Movement类、robotControl类、Instruction类、Path类、Map类、Obstacle类和Exception类。
2. 主要流程：用户通过界面操作选择导航功能，相应GUI类传递信息给交互接口Instruction类，Instruction类将信息传递给robotControl类。robotControl类将导航请求发送给Path类，调用Map类进行基本建图功能，用户通过界面操作控制机器人移动，此时Switch类将移动指令信息发送给Movement类实现对机器人移动，建图完成后返回信息给GUI类，用户在界面得到建图完成的通知。建图过程中，如遇突发情况，会进入Exception类，机器人停止运动。接着，用户输入的运行速度、导航终点位置数据传递给Path类，Path类调用Map类中存储的地图信息，进行路径规划，如果规划失败则进入Exception类，否则调用Movement类进行运动。运动过程中，调用Obstacle类实时监测周围环境，若遇到障碍物则进入异常状态，进行避障操作和重新开始运动。

### 物品识别与抓取功能

1. 功能模块：GUI类、Instruction类、Kinect类、Grasp类、Movement类和Exception类。
2. 主要流程：用户通过界面操作选择抓取物品的功能，相应GUI类传递信息给交互接口Instruction类，将抓取任务发送给Grasp类，用户输入物品所在位置，或选择物品类型返回所选择的物品数据，Grasp类调用Kinect类进行物品检测和识别，如果未能识别到所选物品，则会进入Exception类，否则调用Grasp类进行物品抓取，物品抓取会调用Movement类，调整机器人位置和机械臂状态来抓取物品，若在抓取过程中物品掉落或无法抓取，则进入Exception类，重新抓取物品或放弃抓取。

## 非功能需求

### 性能指标

1. 室内环境简单，周围环境较为空旷的情况下，确定机器人的始发点和目标点后，路径规划应在60s内完成；若周围环境复杂，路径规划应在300s内完成；**设计考虑**：Path类中调用localRou函数时进行计时，若无法在规定的300s时间内完成路径规划，则进入异常状态。
2. 机器人路径规划应择优进行，同时兼顾路径的短程性、前行过程中机身行进的流畅性，以及到达目的地所用时间较小；

**设计考虑**：Path类中存在全局路径规划globalRou函数和局部路径规划localRou函数，全局路径规划完成后，对指定初始和终止地点在局部规划的多条路径中考虑为路径距离、前进障碍物数量、所用时间因素乘以不同权值来得到期望的路径规划结果。

1. 路径规划完成后，机器人速度应在1s内由静止提升至0.1m/s，机器人移动全程平均速度介于0.1m/s到0.2m/s之间；

**设计考虑**：Path类进入calVel函数后，根据计算出来的速度和角度进行移动，保证计算出来的速度在0.1m/s-0.2m/s之间。Movement类中的和速度也需要介于0.1m/s到0.2m/s之间，否则进入异常状态进行处理。

1. 保证机器人移动的精确性：机器人到达目的地后，其实际位置与目标位置的距离不超过0.1m；

**设计考虑**：到达目的地时需要调用Goal类，检测是否到达指定地点，如果所处位置与期望目的地超过0.1m则返回异常信息。

1. 控制机器人目标识别响应时间：机器人到达指定位置后，应在60s内识别出目标物体；

**设计考虑**：Kinect类调用识别模块时开启计时，如果在60s内没有返回识别成功信息，则返回异常信息。

1. 控制机器人物体抓取响应时间：机器人识别出目标物体后，在60s内完成目标物的抓取；

**设计考虑**：Grasp类调用抓取函数grab开启计时，如果在60s内没有返回抓取成功的信息则返回异常信息。

1. 机器人应拥有处理突发事件的能力，避免对自己或外界造成伤害：若机器人的激光雷达最小探测范围（0.15m）内突然出现障碍物，机器人应在1s内由原来状态转变为静止状态；

**设计考虑**：在Obstacle类中，实时反馈周围0.2m内障碍物情况，如果在以0.15m为半径的园内出现障碍物则反馈信息给Movement类，将各项运动信息置0。

1. 若10min内用户没有向机器人发出新指令，并且机器人不处于当前任务未完成状态，应自动进入休眠状态（即待在原地，机械手爪将之最低并且收起），以保证较低的功耗和较小的占用空间；

**设计考虑：**具体功能执行完后会给Instruction返回执行完毕的信息，从此时开始计时，若在10min内没有收到用户输入的指令，则调用Movement类对机械臂和运动状态进行调整，进入休眠状态。

1. 若机器人处于休眠状态，用户发出指令后，机器人应在5s内被唤醒，然后对用户发出的指令进行响应。

**设计考虑：**用户发出指令后，Instruction类将根据不同功能调用具体类别，在5s内得到具体功能的反馈信息，若在5s内没有收到反馈信息则进入异常状态。

### 系统可用性

1. 周围环境不是特别复杂（即理论上机器人一定能找到从始发点到目的地的一条安全且对于机器人来说足够宽敞的环境）的情况下，确定始发点和目标点后，机器人应能够规划出至少一条路径，并在移动前对路径全程进行检查，保证机器人能安全通过；

**设计考虑：**costmap函数实时进行局部路径规划，如果成功选择一条安全路径，则在行进过程中实时采样机器人速度、角度，对行驶路线进行更新。

1. 环境原因导致机器人无法规划出一条安全宽敞的路径，应不予移动，并且向操控者发出报警提示；

**设计考虑：**规划失败时将进入Exception类，反馈相关信息给Instruction类，Instruction类反馈信息给GUI类，向用户界面发出报警信息；同时将Movement类中运动参数全部置零。

1. 当用户没有向机器人发出新指令，并且机器人不处于当前任务未完成的状态时，应待在原地待命，并且不进行任何动作；规定10min内没有接到新指令，即进入休眠状态；
2. 机器人运行过程中，始终避免与禁止的物体发生碰撞；若机器人已经处于静止状态，但是有物体持续向机器人方向移动并且最终导致碰撞，碰撞责任不由机器人以及机器人开发人员承担；

**设计考虑：**机器人在静止状态未进入休眠状态时，激光雷达仍会工作，若在0.15m内有障碍物，则将发出报警信息。

1. 抓取物体的过程中，机械手爪抓取物体的力度应当适中，力度不应过大而损坏物体，也不应过小导致物体掉落。

**设计考虑：**选择物品前需要用户确认物品不含以下信息，物品过于柔软、光滑和沉重，否则机械臂在抓取物品时将损坏物品或自身损坏。

### 可移植性

系统应当具有可移植性：保证了新环境中安装有程序正常运行所依赖的一切环境和包之后，机器人路径规划、移动、目标检测以及物体抓取等相关代码应可移植到新环境中，并且机器人程序可正常运行，并且原有环境下可以进行的一切指令的运行效果与新环境中运行无异。

**设计考虑：**目前所依赖的环境和包采取启智源代码以及Kinect系列的源代码，新环境需要满足是Ubuntu16.04的操作系统。

### 完整性

系统应实现规格说明书所规定的一切功能，用户按照手册使用机器人，应保证所有指令都能正常执行。

**设计考虑：**使用机器人手册将在后续第一次代码开发完成后进行编写，用户可执行的功能为本次软件开发的具体功能。

### 可扩展性

1. 任何对象的任何方法代码长度均不应超过200行；
2. 每个对象通过消息交互，对象之间应该尽量保证透明性：即提供了对象之间消息传递的方式后，一个对象代码的修改应努力保证不影响另一个对象的代码正常运行；
3. 90%bug的修改时间不应该超过3日；
4. 新增加功能不应导致大幅修改原有代码；
5. 代码设计时应参照设计模式。

### 健壮性

1. 合理的温度、湿度、光照的变化不应导致机器人失效；
2. 地理位置以及环境的嘈杂程度不应对机器人的工作效果产生影响。

### 易用性

机器人操作保证对用户友好：经过简单介绍后，不熟悉机器人的用户应能够在3min内完成机器人始发点、目标点以及目标物体的设定。

**设计考虑：**界面操作详细设计见接口设计中“对外用户调用接口”。

### 安全性

1. 机器人应在室内环境下运行，保证机器人所处环境地面坚硬、平滑、无较大裂缝、干燥、温度适宜，坡道坡度不大于15°，能够承载不小于40kg的重量，并且能够提供足够抓地力，例如瓷砖、木地板、大理石、毯砖等；

**设计考虑：**在运行机器人前，在用户界面中展现机器人运行环境条件，以免因为环境因素损坏机器人硬软件。

1. 机器人没有防水功能，为保证用户以及机器人的安全，应保证机器人所处环境内没有可能导致泼洒的液体，例如盛有饮料的水杯，更不能让机器人抓取可能导致液体泼洒的物体；

**设计考虑：**在运行机器人前，在用户界面中展现机器人运行环境条件，抓取任务时，物品选取需满足一定条件：非敞口含有液体的容器、非陶瓷玻璃制品、非超过10kg的物品。

1. 为保证用户以及机器人的安全，应尽量保证机器人所属环境内没有可能因为移动或者掉落导致破碎的玻璃或陶瓷制品制品，如水杯、花瓶等，更不能让机器人抓取此类容易因掉落而破碎的危险物品；
2. 启智ROS机器人的承载能力为10kg，任何情况下都不能允许机器人抓取重量超过10kg的物体；
3. 机器人的工作环境温度应在15℃~35℃，应避免过冷或过热对机器人电池造成影响，引发安全隐患。同时，机器人工作环境中应避免明火、肉眼可见的放电等不安全因素；
4. 机器人整机重量约为30kg，组成部件均为坚硬物体，且不可避免的有棱角，因此机器人移动速率不应超过0.2m/s，以防止对用户以及机器人造成损伤。

**设计考虑：**实时移动速率应在用户界面显示，若用户设置速率高于0.2m/s，则按照0.2m/s的速率前行，且在用户界面告知用户限制速率大小。