**【基于嵌入式系统的路径规划机器人】**

**软件设计说明书**

**【SDD + 105】**

**【1.0】**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 已经在写了 | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 17373033 | 沈佳钰 | 1范围 4.1 总体结构 4.2.2软件类图 4.3 硬件体系结构 6.2避障模块 6.3路径规划模块6详细设计整合、设计文档整合 |
| 17373301 | 校怡哲 | 2需求概述 4.2.2软件类图 5.3内部接口 8需求可追踪性说明 |
| 17373472 | 王宇思 | 3数据库设计4.2.2软件类图5.1用户界面设计5.2外部接口5接口设计整合 |
| 17373414 | 鲍韦彤 | 4.2软件体系结构 4体系结构设计整合 |
| 17373388 | 刘婉婷 | 4.2.2软件类图 4.4技术体系结构6.1总控模块 6.4建图模块6.5导航模块 |

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 | 4.18 | 沈佳钰 | 鲍韦彤 | 第一版，完成设计文档初稿 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 范围 1](#_Toc38051257)

[1.1 项目概述 1](#_Toc38051258)

[1.1.1 项目背景 1](#_Toc38051259)

[1.1.2 功能需求 1](#_Toc38051260)

[1.1.3 非功能需求 2](#_Toc38051261)

[1.1.4 应用场景 2](#_Toc38051262)

[1.2 文档概述 2](#_Toc38051263)

[1.2.1 用途 2](#_Toc38051264)

[1.2.2 内容组织 3](#_Toc38051265)

[1.3 术语和缩略词 3](#_Toc38051266)

[1.4 引用文档 3](#_Toc38051267)

[2. 需求概述 4](#_Toc38051268)

[2.1 系统数据需求 4](#_Toc38051269)

[2.1.1 分析类 4](#_Toc38051270)

[2.1.2 数据字典 5](#_Toc38051271)

[2.2 功能需求 5](#_Toc38051272)

[2.2.1 基本避障运动 5](#_Toc38051273)

[2.2.2 路径规划 6](#_Toc38051274)

[2.3 非功能需求 6](#_Toc38051275)

[2.3.1 性能 6](#_Toc38051276)

[2.3.2 可靠性 6](#_Toc38051277)

[2.3.3 可拓展性 6](#_Toc38051278)

[2.3.4 可用性 7](#_Toc38051279)

[2.3.5 安全性 7](#_Toc38051280)

[3. 数据库设计 7](#_Toc38051281)

[4. 体系结构设计 9](#_Toc38051282)

[4.1 总体结构 9](#_Toc38051283)

[4.2 软件体系结构 10](#_Toc38051284)

[4.2.1 软件总体结构 10](#_Toc38051285)

[4.2.2 软件整体类图 11](#_Toc38051286)

[4.2.3 时序图 12](#_Toc38051287)

[4.3 硬件体系结构 15](#_Toc38051288)

[4.4 技术体系结构 15](#_Toc38051289)

[5. 接口设计 17](#_Toc38051290)

[5.1 用户界面设计 17](#_Toc38051291)

[5.2 外部接口 17](#_Toc38051292)

[5.3 内部接口 19](#_Toc38051293)

[5.3.1 机器人运动控制接口 19](#_Toc38051294)

[5.3.2 机器人语音控制接口 19](#_Toc38051295)

[5.3.3 建图与存图控制接口 20](#_Toc38051296)

[5.3.4 导航控制接口 20](#_Toc38051297)

[5.3.5 传感器状态探测接口 20](#_Toc38051298)

[5.3.6 无线传输接口（TCP协议） 20](#_Toc38051299)

[6. 详细设计 21](#_Toc38051300)

[6.1 总控模块 21](#_Toc38051301)

[6.1.1 自动导航 21](#_Toc38051302)

[6.1.2 主动控制 22](#_Toc38051303)

[6.2 避障模块 22](#_Toc38051304)

[6.3 路径规划模块 23](#_Toc38051305)

[6.4 建图模块 23](#_Toc38051306)

[6.5 导航模块 24](#_Toc38051307)

[7. 运行与开发环境 25](#_Toc38051308)

[7.1 运行环境 25](#_Toc38051309)

[7.1.1 硬件运行环境 25](#_Toc38051310)

[7.1.2 软件运行环境 25](#_Toc38051311)

[7.2 软件环境 25](#_Toc38051312)

[7.2.1 硬件开发环境 25](#_Toc38051313)

[7.2.2 软件开发环境 26](#_Toc38051314)

[8. 需求可追踪性说明 26](#_Toc38051315)

[8.1 逆向需求追踪 26](#_Toc38051316)

[8.1.1 项目目标回顾 26](#_Toc38051317)

[8.1.2 需求文档追踪 26](#_Toc38051318)

[8.2 正向需求追踪 27](#_Toc38051319)

[8.2.1 手动操控机器人 27](#_Toc38051320)

[8.2.2 机器人自动导航 27](#_Toc38051321)

[8.2.3 机器人避障 27](#_Toc38051322)

[8.2.4 非功能需求 28](#_Toc38051323)

[8.3 需求追踪矩阵 28](#_Toc38051324)

# 范围

## 项目概述

### 项目背景

在高度智能化的今天，各种专用功能机器人的引入使用无疑会给我们的生活带来巨大的便捷。自动路径规划机器人能够在很多工作上给予我们一些帮助。本项目旨在用工程化方法开发一个基于嵌入式系统的路径规划型机器人，该机器人能够在3D空间内平稳地进行水平移动、能够识别障碍物并进行躲避、能够接收传感器信号建立实时地图、能够进行动态路径规划及导航控制等操作。

嵌入式系统如今在此类项目中得到了很好的应用，嵌入式系统由软件和硬件组成，以应用为中心，以现代计算机技术为基础，是可独立工作的小型计算机系统。其专用性强、实时性好、可裁剪性好、可靠性高、功耗低，使用嵌入式芯片，提升机器人在微型化、高智能化方面的优势，同时还能降低成本，是较佳的选择。

本项目使用具有高度灵活性的软件架构的ROS来编写机器人软件程序，它包含了大量工具软件、库代码和约定协议，是一个开放式的协作框架。使用ROS系统，可以简化跨机器人平台创建复杂、鲁棒的机器人行为这一过程的难度与复杂度。

### 功能需求

该机器人的主要功能需求有：

* 机器人的主动控制 ；
* 静态或动态障碍物避障 ；
* 机器人利用传感器实时建立环境地图 ；
* 机器人根据地图和自身的位置信息实现动态路径规划及导航控制。

### 非功能需求

* 性能需求：控制机器人的性能，使响应时间、吞吐量、资源利用率达到一定的标准，机器人具有较低的功耗与较强的续航性能；
* 安全性：实现权限控制，具有管理员权限方可对机器人发送指令，保证系统的安全性；机器人还应当能保证用户的安全，运行速度不宜过高，且行为应当具有较好的可预测性 ；
* 可维护性与可扩展性：设计针对每个基本功能编写API代码，代码具有较好的模块性、降低模块间的耦合度，提升可复用性；
* 可靠性：有一定的容错机制，系统具有较好的鲁棒性；
* 易用性：系统具有易学习性、易操作性、GUI美观。

### 应用场景

* 扫地机器人。扫地机器人需要使用传感器实时建立环境地图，并且能够规划合适的路径覆盖整个清扫区域，同时，还需要识别并躲避家具等障碍物。本项目的路径规划机器人基本上能满足这些需求；
* 巡检机器人。例如一种应用于海上石油平台的智能检测监测机器人，可以有效地提高海上开采平台的巡检 安全性、巡检质量、降低人工劳动成本；
* （餐厅）领位机器人。领位机器人要求机器人根据地图和自身的位置信息实现动态路径规划及导航控制，规划合适快捷的路径将客人引导到正确的目标位置。

## 文档概述

### 用途

本文档在需求文档的基础上，进一步细化避障机器人的设计细节，在总结需求文档的基础上，进行数据库设计、体系结构设计、接口和用户界面设计、详细设计等。以便于接下来的项目开发，为项目组组员提供参考。

### 内容组织

内容组织如下目录所示，其中包括：需求概述、数据库设计、体系结构设计、接口设计、详细设计、运行与开发环境需求、需求可追踪性说明。

具体内容组织见本文开头目录。

## 术语和缩略词

表格 1‑1

|  |  |
| --- | --- |
| 术语或缩写 | 定义 |
| 嵌入式系统 | 嵌入到对象体系中的专用计算机应用系统（Embedded system）。 |
| ROS | 机器人操作系统（Robot Operating System），编写机器人软件程序的一种软件架构。 |
| Ubuntu | 一个以桌面应用为主的Linux操作系统 |
| API | API（Application Programming Interface，应用程序接口） |
| GUI | 图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口） |

## 引用文档

启智ROS版\_开发手册\_20181109

启智ROS机器人

用户界面设计——有效的人机交互策略（第五版）\_201103\_0101版

SRS-105-软件需求规格说明书3.0

# 需求概述

本项目旨在设计一个能够自动规划路径的机器人，其可以被用户手动操纵，也可以在用户输入目的地之后自动规划路径并进行避障。

* 用户手动操控：由于机器人的所有运动都由用户手动操控，所以机器人没有办法进行自主避障，但其可以进行障碍物检测，在发现有障碍物在行进方向时，在用户界面上向操作用户发出警告，若机器人仍被操控向障碍物移动，则在离障碍物某一较近距离的时候，机器人会紧急停止。
* 自动规划路径：该机器人可以根据用户输入的目的地进行自动寻址，生成当前所处位置附近的地形信息，自动规划路径前行，并且能够自动避障。

## 系统数据需求

### 分析类

所有设计分析类见下表：

表格 2‑1 确定分析类情况表

|  |  |  |
| --- | --- | --- |
| 潜在类 | 表现形式 | 不符合条件 |
| 用户 | 角色 | 保留信息、所需服务 |
| 目的地 | 场所 | 多个属性 |
| 控制面板 | 设备 |  |
| 移动命令 | 事物 | 多个属性 |
| 动态或静态障碍物 | 事物 | 所需服务 |
| 应急处理 | 事件 |  |
| 警报 | 事件 | 多个属性 |
| 路线 | 事物 |  |
| 静态障碍物地图 | 事物 |  |
| 机器人总控系统 | 事物 |  |
| 起始点 | 场所 | 多个属性 |
| 延迟时间 | 事物 | 多个属性 |
| 当前位置 | 场所 | 所需服务、多个属性 |

### 数据字典

表格 2‑2 数据字典信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项 | 数据结构 | 数据流 | 数据存储 | 处理过程 |
| 平面x坐标 | 平面图信息 | 目的位置信息 | 建图 | 应急处理 |
| 平面y坐标 | 目的位置信息 | 图信息 | 路径规划 | 运动控制 |
| 平面点颜色 | 图信息 | 平面图信息 | 导航 |  |
| 路径方向信息 | 无障碍物区域 | 静态障碍物信息 | 运动控制 |  |
| 路径长度信息 | 未知区域 | 路径信息 |  |  |
| 机器人状态信息 | 静态障碍物 | 机器人信息 |  |  |
| 机器人方向信息 | 平面点信息 | 动态障碍物信息 |  |  |
| 指令信息 | 动态障碍物 | 导航信息 |  |  |
|  | 路径信息 |  |  |  |
|  | 初始位置信息 |  |  |  |
|  | 导航信息 |  |  |  |
|  | 机器人信息 |  |  |  |
|  | 机器人位置信息 |  |  |  |

## 功能需求

### 基本避障运动

**障碍物检测：**使用传感器检测周围环境信息，如障碍物的距离和尺寸，障碍物的具体位置等，并储存得到的信息用于路径规划。

**避障路径规划：**根据障碍物信息，运用路径规划算法规划避障路径。

**运动驱动：**根据避障路径信息，驱动机器人按照路径运动。

**故障检测：**检测运动过程中发生的故障。

### 路径规划

**追踪路径：**根据输入的路径信息，完成路径的追踪。

**运动控制：**控制机器人按输入路径运动。

**故障检测：**检测运动过程中发生的故障。

## 非功能需求

### 性能

* 在实验室环境下，给定起点与终点，路径规划应在90s之内完成。如果环境比较复杂，路径规划应在450s之内完成；
* 完成路径规划后，机器人应当在1s内从静止加速到0.1m/s，从起点到终点的平均移动速度应保证不小于0.1m/s不大于0.2m/s；
* 机器人到达终点停止运动时，机器人所在位置距离目标点的实际位置应小于0.1m；
* 机器人移动过程中在其激光雷达的最小探测范围内突然遇到障碍物，则应在1s内减速至停止；

### 可靠性

* 在机器人可安全运行的环境中，于规定时间内软件无错运行的概率保证在95%以上，其中无错指机器人能够完整、正确的执行用户发出的指令并达到预期效果。

### 可拓展性

* 任何对象的任何方法不应超过200行；
* 新增任何功能时，不应大幅修改原有代码；
* BUG的修复不能大幅修改原有代码；

### 可用性

* 经过简单学习使用知识后，用户可以在60s内完成机器人起点、终点的设定，且错误率低于1%；

### 安全性

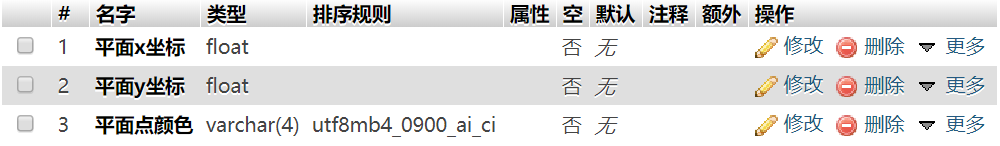
* 启智ROS机器人是室内机器人，在此环境之外运行可能会损坏机器人。
* 启智ROS机器人工作平面需要能够承载不小于40kg的重量。
* 启智ROS机器人需要运行在坚硬平面上，否则可能导致机器人卡住，运动受阻，建议使用商用地毯、瓷砖等材质。
* 启智ROS机器人应保持在水平平面上工作，坡道坡度不大于15度，坡道倾斜度过大可能导致倾覆。
* 启智ROS机器人不具备防水功能，在任何情况下，启智机器人都不应该与雨水、雾、地面积水以及其他液体接触，否则可能导致电路和机构损坏。
* 启智ROS机器人设计工作温度15℃到35℃之间，使用时务必远离明火和其他热源。
* 启智ROS机器人重量约为30kg，若在运行时速度较快，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度小于0.2m/s；

# 数据库设计

基于嵌入式系统的路径规划机器人主要完成两个功能：基本避障运动和路径规划。基本避障运动需要检测障碍物位置，并储存障碍物位置信息完成避障路径规划，这就需要数据库储存周围环境信息。路径规划功能需要按照输入路径进行追踪，所以输入的路径需要储存在数据库中。同时，机器人还有语音播报和语音控制及指令控制功能，这就需要数据库储存指令信息。机器人运动信息也需要数据库进行储存，在系统出现故障时可以根据机器人的运动信息进行调试。

为了实现这些功能，数据库中建立四个数据表：环境信息数据表，路径信息数据表，指令信息数据表和运动状态数据表。

环境信息数据表储存环境位置坐标和坐标对应位置代表意义，比如障碍点，已探明不存在静态障碍物点和还没有探索到的未知点。这些不同类别的点通过平面点颜色区分。



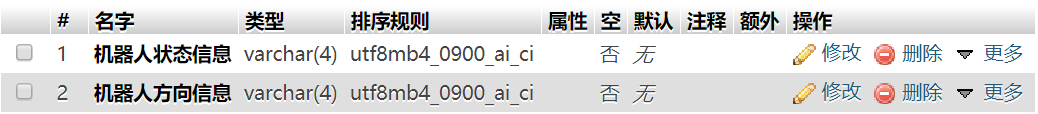
路径信息数据表储存输入路径的方向信息和长度信息。



指令信息数据表储存语音模板和其他指令信息。

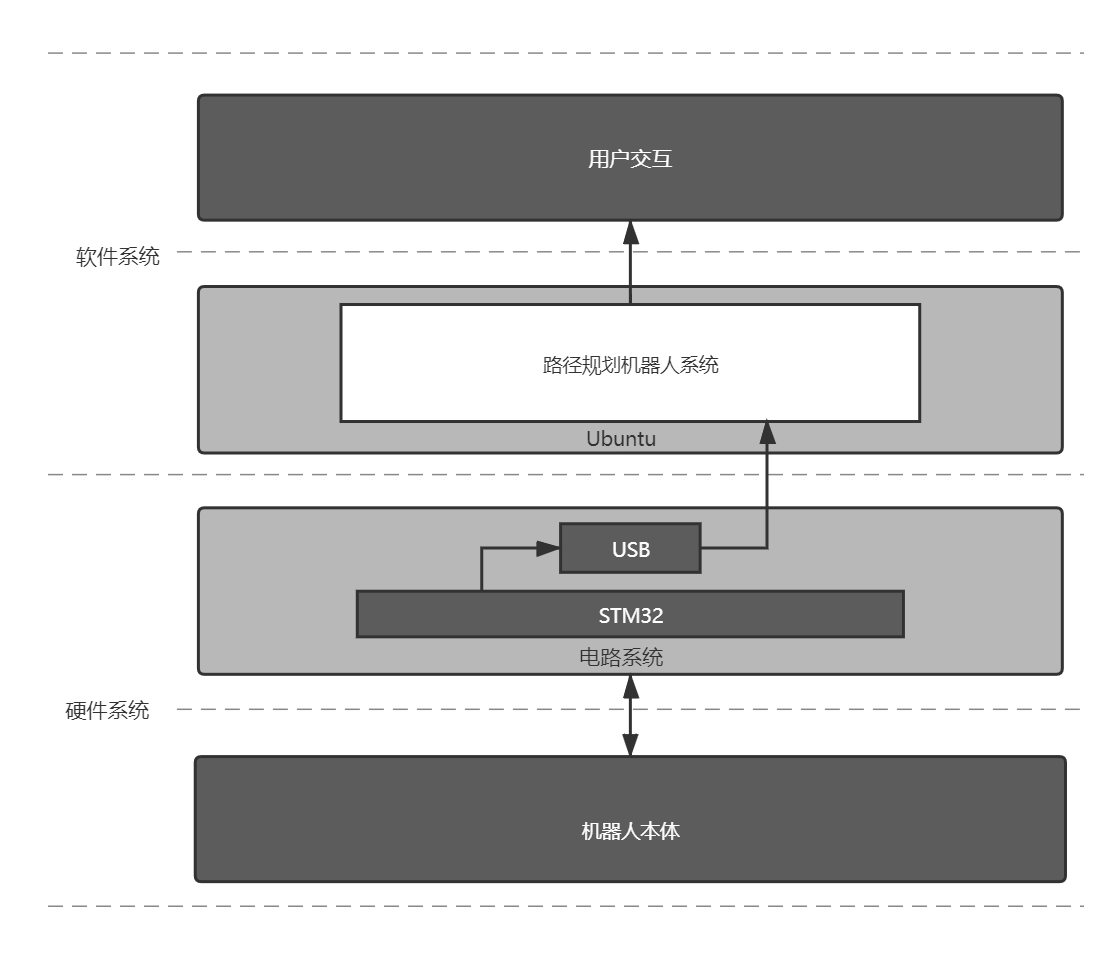


运动状态数据表储存机器人运动状态和方向信息。



# 体系结构设计

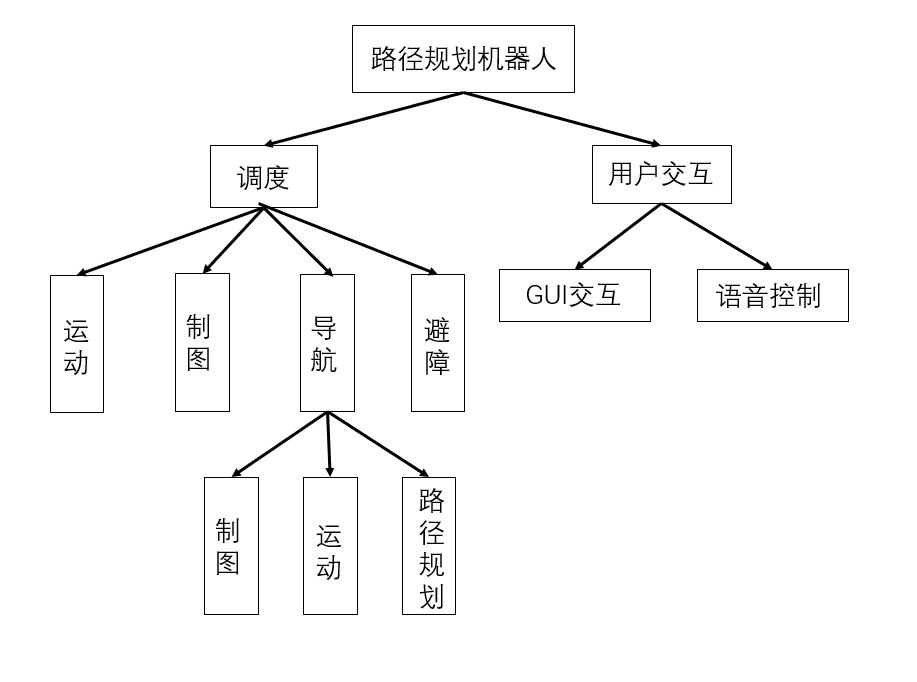
## 总体结构



**图4-1 系统总体结构**

## 软件体系结构

### 软件总体结构



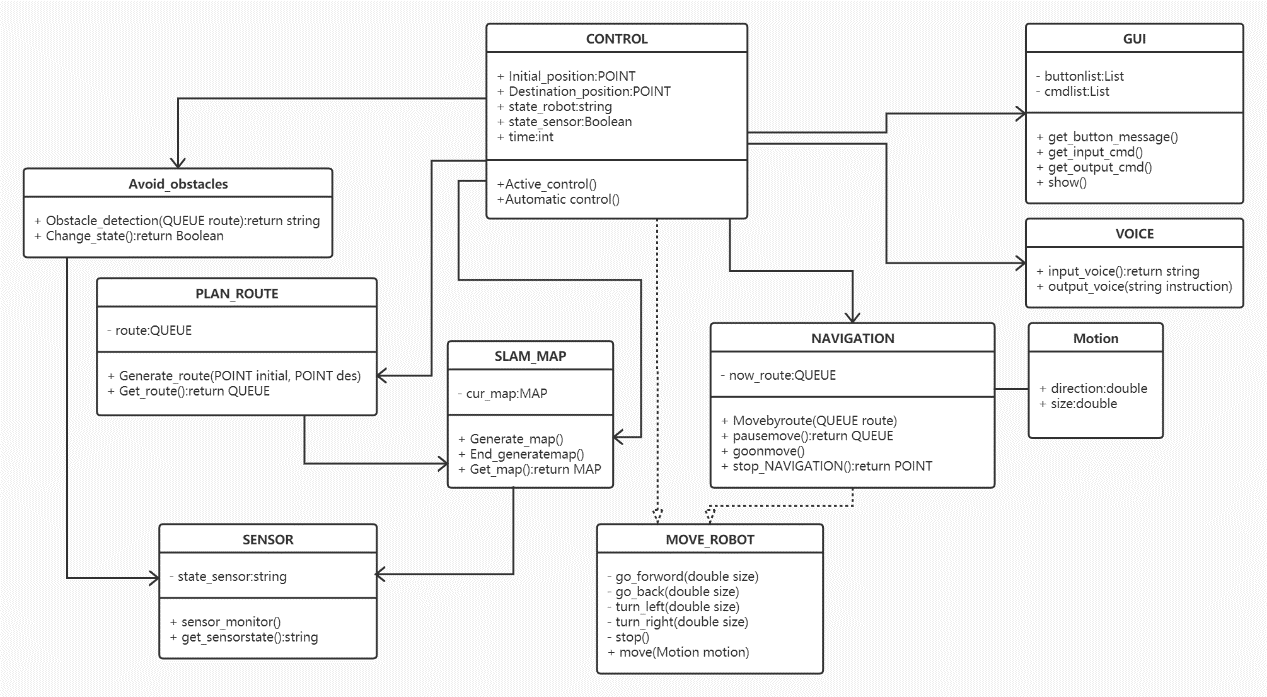
**图4-2 软件总体结构图**

为了实现路径规划机器人，将功能分为两类：调动控制和用户交互。

用户交互控制主要提供调度控制所需要的参数信息，包括GUI交互与语音控制两部分。

调度控制主要调度运动模块、建图模块、导航模块和避障模块。其中， 导航模块中还会调用制图模块、运动模块、路径规划模块，来实现导航过程。

### 软件整体类图

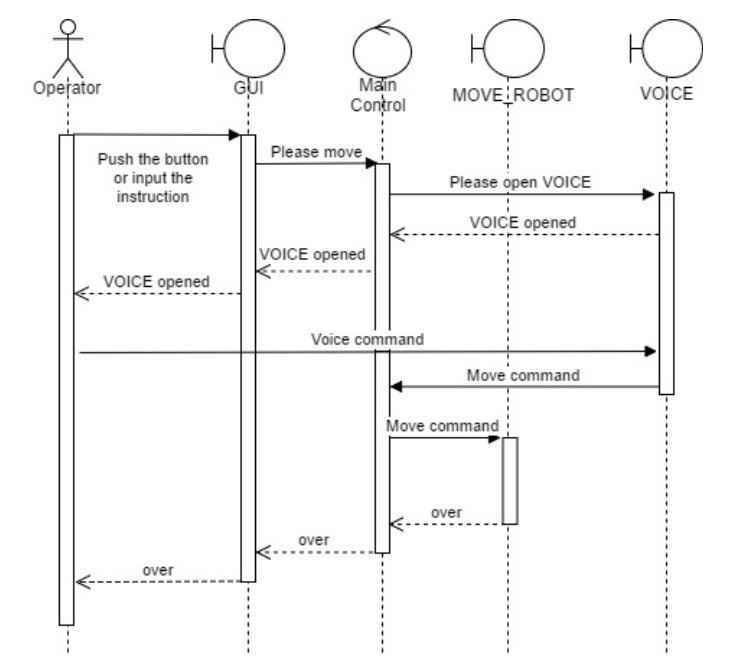


**图4-3 软件整体类图**

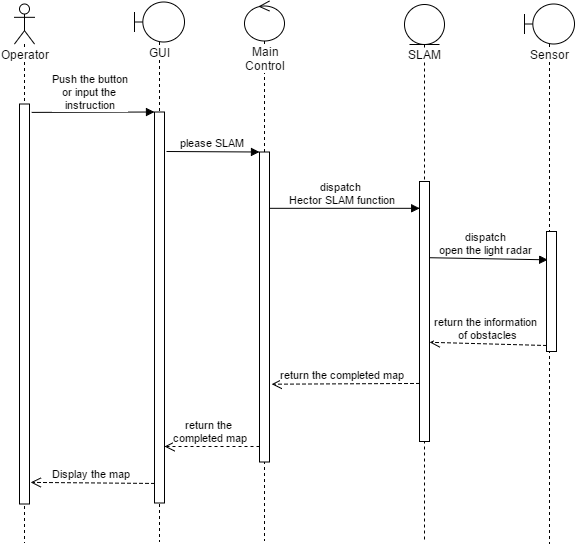
在软件总体结构的基础上给出总体类结构设计，将软件总体结构中的关键模块细化为相应的类。

### 时序图

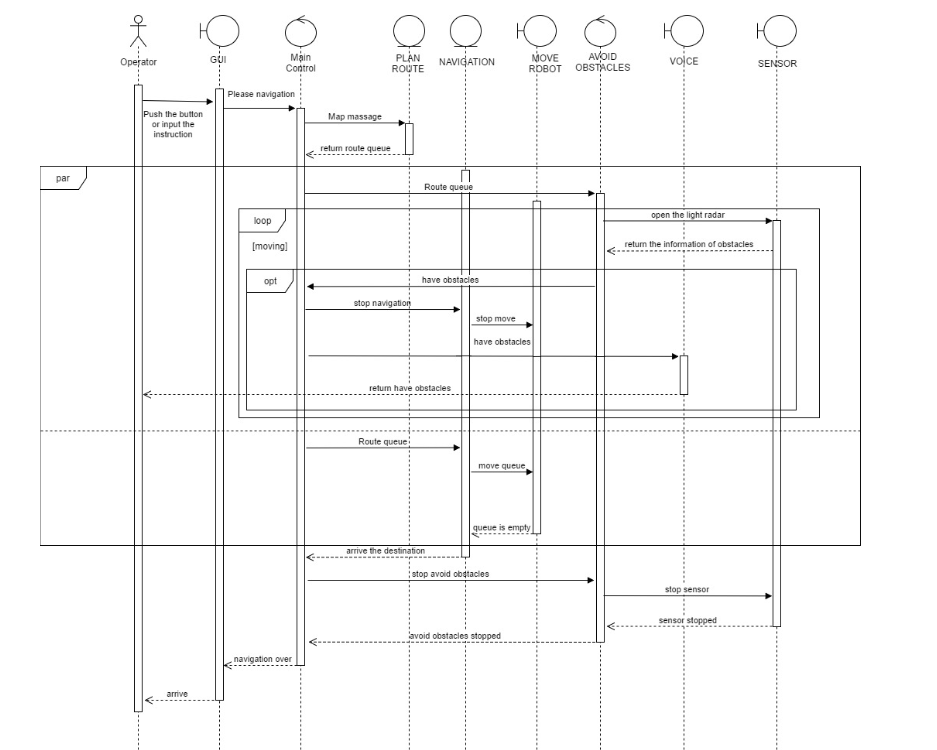
根据整体设计给出三个用例（主动控制、建图、导航）对应的时序图如下：



**图4-4 时序图（主动控制）**

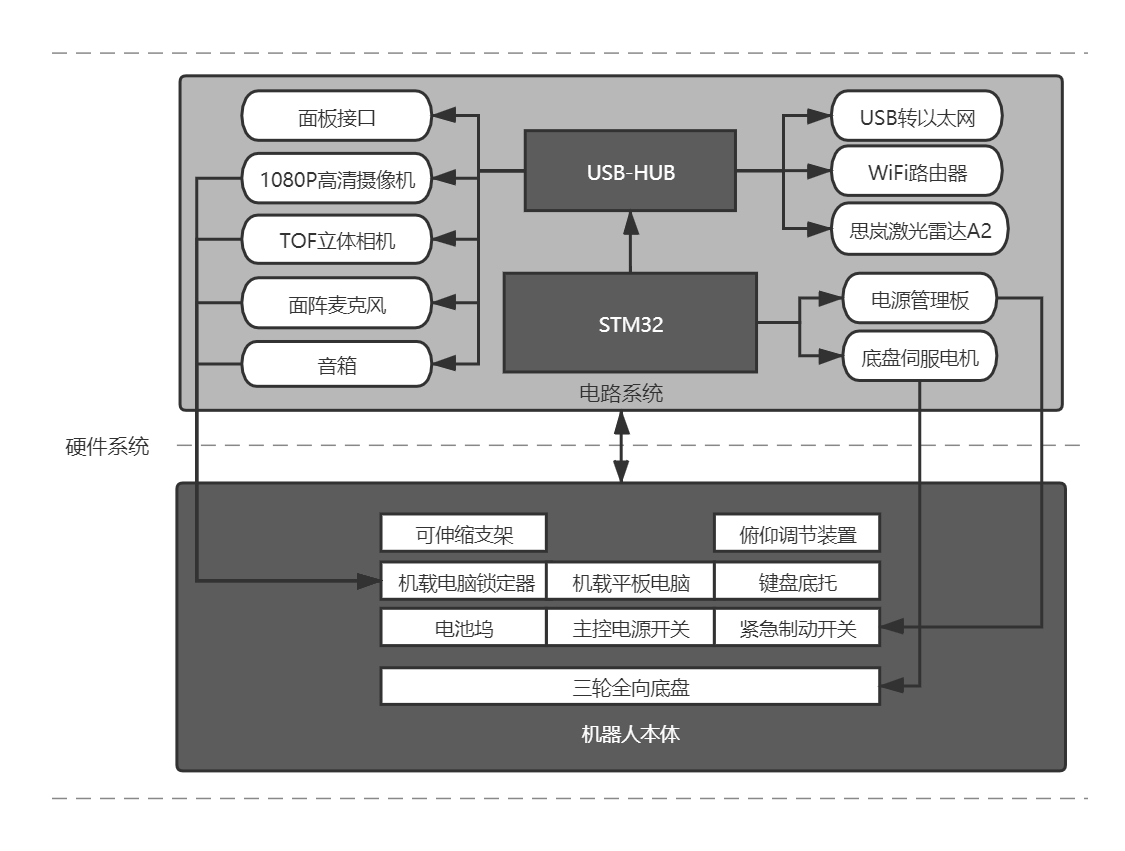


**图4-5 时序图（建图）**



**图4-6 时序图（导航）**

## 硬件体系结构



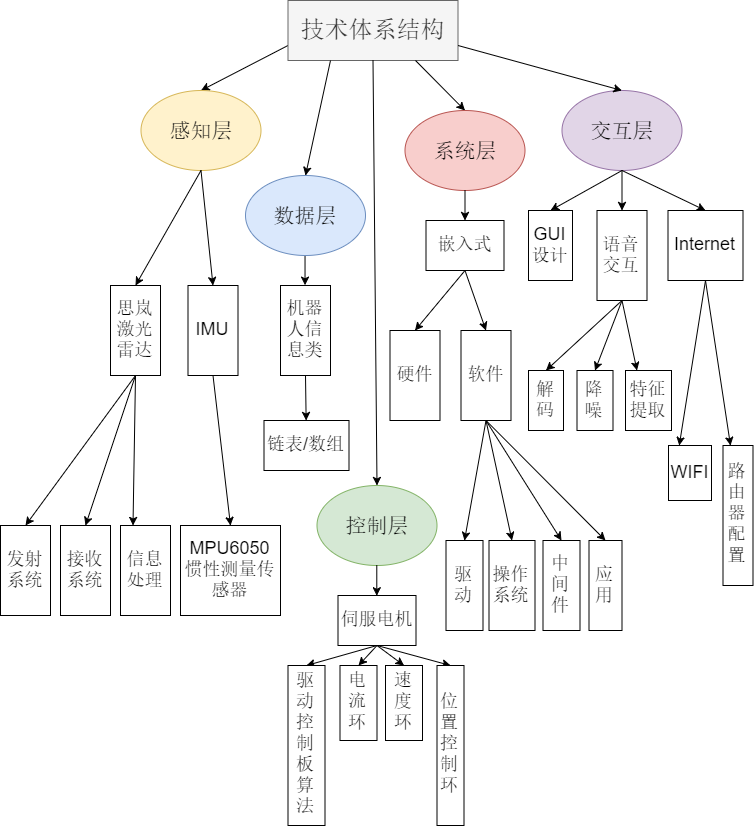
**图4-7 硬件体系结构**

## 技术体系结构

技术体系结构定义了整个信息系统中的技术环境和基础结构。大致分为如下5层：

1. 感知层：有各类传感器感知外界信息，并把从外界获取的信息进行转换，形成有意义的可用的数据。即传感器、信号处理、模式识别等。
2. 数据层：存储感知层获取的某些非即时使用的信息，如当前环境的地图信息等。必要时可用数据库技术。
3. 控制层：规划机器人的动作，包括了机械、电子（拖动）这些看得见的部分，然后还包括了于控制理论为核心的运动控制，规划理论为核心的运动规划。
4. 系统层：包含了机器人所需的嵌入式系统的软硬件技术。如驱动、操作系统、CPU、各种功能所需的软件代码库及接口设计等。
5. 交互层：进行人机交互。包含GUI设计、语音交互等。语音交互需要连接到互联网，其中包含了网络通信技术。

在上述层次的基础上，将为实现基于嵌入式系统的路径规划机器人业务而采用的技术和产品画出层次结构(如图4-6所示)。



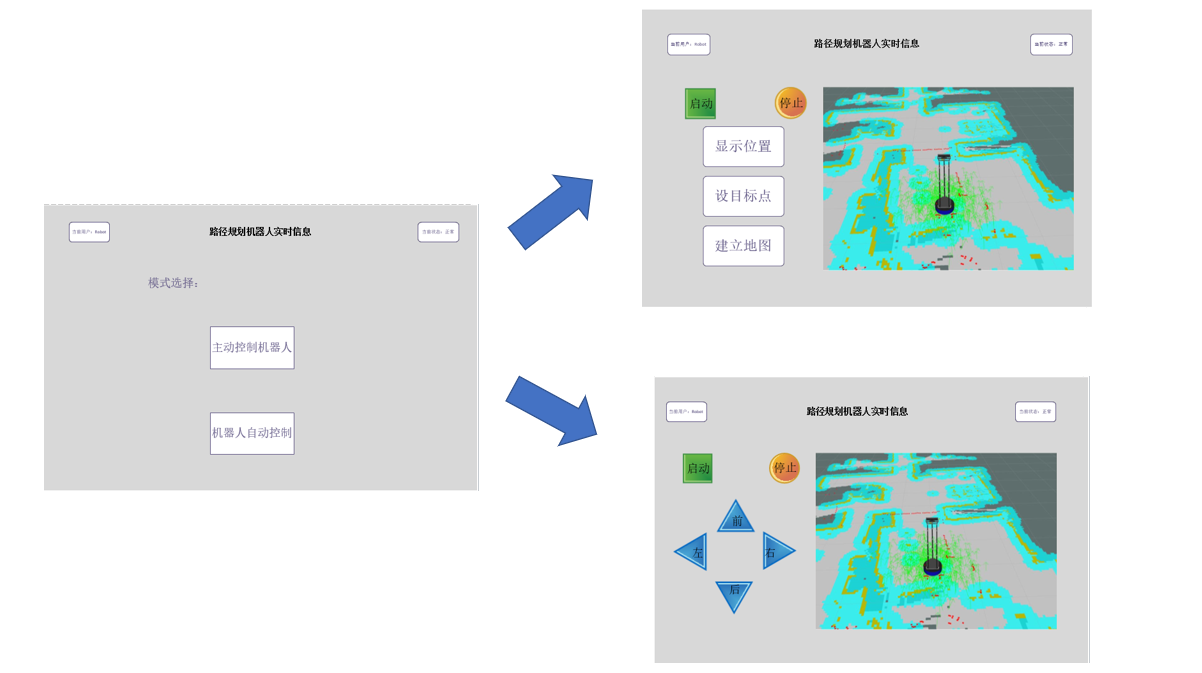
**图4-8 技术体系结构**

# 接口设计

## 用户界面设计

用户通过机载平板电脑控制机器人。控制机器人需要有选择模式的界面、用户控制机器人进行移动的界面和机器人自动控制追踪路径界面。

选择模式界面提供用户主动控制机器人模式和机器人自动控制模式两个选项。用户主动控制机器人模式用户可以控制机器人启动停止和移动方向。机器人自动控制模式用户可以控制机器人启动停止，并提供显示位置，设目标点和建立地图选项按钮。

****

**图5-1 用户界面示意图**

## 外部接口

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）。

启智控制器内置MPU6050惯性测量传感器。基本参数如表1所示。

|  |  |
| --- | --- |
| 封装 | SMD |
| 类别 | 传感器 |
| 类型 | 三轴加速计，陀螺仪 |
| 输出类型 |  |
| 工作温度 | -40°C-85°C |

**表格5-1 惯性测量传感器参数**

激光雷达是地面移动机器人常用的一种传感器，其工作原理是用一个高速旋转的激光测 距探头将周围360°的障碍物分布状况测量出来，形成障碍物轮廓的俯视二维点阵输入到 ROS 系统里。ROS系统使用激光雷达参数如表2所示。

|  |  |
| --- | --- |
| 型号 | 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 |

**表格5-2 激光雷达参数**

机器人顶部配备了一个Kinect2相机，相机具体参数如表3所示。

|  |  |  |
| --- | --- | --- |
| 颜色 | 分辨率 | 1920x1080(32bit) |
| 帧率 | 30/15fps |
| 深度 | 分辨率 | 512x424(16bit) |
| 帧率 | 30fps |
| 红外相机 | 分辨率 | 512x484 |
| 频率 | 30Hz |
| 可侦测范围 | | 0.5-4.5m |
| 深度误差 | | 小于检测范围的0.5% |
| 角度 | 水平 | 70° |
| 垂直 | 60° |

**表格5-3 Kinect2相机参数**

软件外部接口主要分为软件与用户交互接口和软件与硬件交互接口。软件与用户交互接口主要为机载平板电脑，界面设计如上一节所示。软件与硬件交互接口主要由操作系统完成。

## 内部接口

### 机器人运动控制接口

* void go\_forward(double size)：向前运动。
* void go\_back(double size)：向后运动。
* void turn\_left(double size)：向左转。
* void turn\_right(double size)：向右转。
* void stop()：停止。
* void move(Motion motion)：运动。

这些方法通过软件信息控制机器人硬件系统，完成机器人运动过程。

### 机器人语音控制接口

* string get\_keyword()

读取PocketSphinx发布的语音识别的结果

* void send\_alarm\_signal(int s)

发送警报信号，语音播报系统会根据之前添加的警报关键字播报

* void send\_start\_signal(int s)

发送导航开始的播报信号，语音播报系统会根据之前添加的启动导航关键字播报

### 建图与存图控制接口

* void send\_build\_map\_signal(int s)

发送建图信号，雷达系统会启动传感器并更新周围环境数据

* void send\_save\_map\_signal(int s)

发送存图信号，雷达系统收到此信号认为建图已经结束，关闭传感器并将已经探测到的所有周围环境数据进行存储

### 导航控制接口

* void send\_nav\_point(point p)

为导航系统设置导航目标点

* bool get\_nav\_state()

得到当前导航状态，若为真则导航成功，若为假则导航异常或失败

### 传感器状态探测接口

* bool get\_sensor\_state()

得到传感器当前的探测状态，若当前有距离前进方向过近的障碍物，则返回false

### 无线传输接口（TCP协议）

* SOCKET socket( int af, int type, int protocol )

创建一个socket

* int bind( SOCKET s, struct sockaddr \*name, int namelen)

将本地地址与socket绑定

* int listen( SOCKET s, int backlog )

在套接字上监听连接请求

* int connect( SOCKET s, struct sockaddr \*name, int namelen )

与name指定的地址建立连接

* SOCKET accept( SOCKET s, struct sockaddr \*addr, int \*addrlen )

接受与本socket的连接请求

* int send( SOCKET s, char \*buf, int len, int flags )

在连接的socket上发送数据

* int recv( SOCKET s, char\* buf, int len, int flags )

在连接的或绑定的socket上接收数据

# 详细设计

## 总控模块

总控模块由两部分组成，机器人自动导航和机器人主动控制。

### 自动导航

这部分主要实现机器人的导航功能，即用户输入目的位置，系统会根据生成的地图信息已经机器人的初始位置和目的位置信息进行路径规划，根据规划好的路径完成导航。

在机器人根据规划好的路径运动的同时，系统的避障模块也会同时运行，传感器监测机器人即将行进的路径上是否有障碍物，如果检测到障碍物，立即停止机器人的运动。在一段设定的时间内，障碍物没有消失，则重新生成地图，并重新开始导航；如果障碍物消失，机器人继续运动。

伪代码如下：

1. **void** Automatic control ()
2. {
3. Destination\_posution = GUI.get\_button\_message(); //从GUI界面获取目的位置
4. Initial\_position = SENSOR.getPosition(); //从传感器获取当前位置
5. SLAM\_MAP.Generate\_map(); //调度建图模块建图
6. QUEUE route = PLAN\_ROUTE.Generate\_route(Destination\_position, Initial\_position);
7. //路径规划类根据起始点和终点规划路径
8. NAVIGATION.Movebyroute(route); //导航模块根据路径导航
9. }

### 主动控制

这部分主要实现用户手动控制机器人的功能，用户输入需要机器人进行的运动，比如前进几米、左转多少度、后退几米、停止等，总控模块调用运动接口的运动函数，机器人做出用户需要的动作。

伪代码如下：

1. **void** Active\_control()
2. {
3. **switch** (GUI.get\_input\_cmd()) { // 从GUI获取主动控制的命令
4. **case** 0:
5. MOVE\_ROBOT.goforward(route.get(i).size);
6. **break**;
7. **case** 1:
8. MOVE\_ROBOT.goback(route.get(i).size);
9. **break**;
10. ……
11. }
12. }

## 避障模块

避障模块主要实现机器人在运动过程中的障碍物检测和报警功能。障碍物检测在机器人运动的同时一直运行，如果在机器人即将行进的路径上检测到障碍物，则机器人状态设置为停止，传感器状态设置为检测到障碍物，并开始报警功能。报警功能使得目前运动的机器人知道前方有障碍物，并停止运动，系统同时播报前方出现障碍物，机器人停止的信息。

伪代码如下：

1. String Obstacle\_detection(QUEUE route)
2. {
3. SENSOR.sensor\_monitor();//开启障碍物检测
4. **while** SENSOR.get\_sensorstate() == **true** //检测前方路径有障碍物
5. Change\_state();
6. **return** **true**;
7. end **while**
8. }

## 路径规划模块

路径规划模块主要根据生成的地图信息以及机器人初始位置和目的位置信息规划机器人自动导航的合理路径。

伪代码如下：

1. **void** Generate\_route(POINT initial, POINT des)
2. {
3. route = wpbh\_local\_planner.route\_plan(initial,des);
4. //调用wpbh\_local\_planner包里的路径规划函数（会进行优化）
5. }
7. QUEUE Get\_route()
8. {
9. **return** route;
10. }

## 建图模块

启智机器人采用激光雷达收集到的数据进行SLAM（Simultaneous Localization And Mapping，即时定位与地图构建）。ROS 支持多种 SLAM 算法，其中主流的是 Hector SLAM 和 Gmapping。我们拟定用Hector SLAM算法进行地图的构建。

伪代码如下：

1. **void** Generate\_map ()
2. {
3. SENSOR.sensor\_monitor(); //打开传感器
4. Terminal.input(“roslaunch wpb\_home\_tutorials hector\_mapping.launch\n”);
5. //在终端里输入建图命令
6. cur\_map.add(SENSOR.get\_barrier\_flow());//当前地图增添障碍物信息
7. **return**;
8. }
9. **void** End\_generatemap (Boolean flag)
10. {
11. **if** (flag) {
12. Terminal.input(“rosrun map\_server map\_saver -f map \n”);
13. //保存当前地图
14. }
15. }
17. MAP Get\_map ()
18. {
19. **return** cur\_map；
20. }

## 导航模块

用户在GUI设置好目标点之后，路径规划进行规划。从路径规划类获取规划好的一串路径指令信息，导航类调用MOVE\_ROBOT类，按照路径信息移动。

伪代码如下：

1. Boolean stop\_state = **false**; //导航是否暂停
2. POINT Movebyroute（QUEUE route） {
3. **while** (! stop\_state)  { //如果暂停状态位为false
4. **for** (**int** i = 0; i < route.length; i++) {
5. //按照路径信息调用move类的方法移动
6. **switch** (route.get(i).type) {
7. **case** 0:
8. MOVE\_ROBOT.goforward(route.get(i).size);
9. **break**;
10. **case** 1:
11. MOVE\_ROBOT.goback(route.get(i).size);
12. **break**;
13. ……
14. }
15. }
16. }
17. }
18. QUEUE pausemove() //暂停导航
19. {
20. stopstate = **true**;
21. **return** now\_route;  //返回当前路径
22. }
23. **void** goonmove() //继续导航
24. {
25. stopstate = **false**;
26. }
27. **void** stopNAVIGATION() //停止导航，返回机器人当前位置
28. {
29. MOVE\_ROBOT.stop();
30. **return** now\_position;
31. }

# 运行与开发环境

## 运行环境

### 硬件运行环境

本系统运行的计算机配置：

CPU：建议 1.2GHz Pentium III 处理器及以上。最低700MHz。

内存：建议 RAM 要求512MB。

硬盘：建议10 GB 以上的可用空间。

显示器分辨率：最低Super VGA (800x600) 或更高分辨率的显示器（颜色设

置为32位真彩色）。

其他：总开关、急停开关、充电接口、键盘、鼠标、2个以上USB接口。

### 软件运行环境

本系统运行的软件环境为Ubuntu 16.04的支持kinetic版本的ROS系统。

用户账号名：Robot 密码：6

## 软件环境

### 硬件开发环境

本系统开发的硬件平台为启智ROS机器人的硬件平台，具体主要包括USB-HUB、启智电池模块、电源控制板、启智控制器、启智伺服电机模块等。

其中核心芯片为STM32系列，专为要求高性能、低成本、低功耗的嵌入式应用设计的ARM。

### 软件开发环境

软件开发环境是 RoboWare Studio，基于 Visual Studio Code 开发的 ROS 专用 IDE，软件开发语言为C++。

软件开发基于启智机器人的源码包，包括基础软件包和扩展软件包。

操作系统Win7以上以及装有Linux的双系统或虚拟机、其中装有IDEA、Eclipse、PyCharm、VSstudio等IDE以及各IDE运行所需要的环境。

# 需求可追踪性说明

## 逆向需求追踪

### 项目目标回顾

本项目旨在用工程化方法开发一个基于嵌入式系统的路径规划型机器人，该机器人能够在三维空间内平稳地进行水平移动、能够识别障碍物并进行躲避、能够接收传感器信号建立实时地图、能够进行动态路径规划及导航控制等操作。

### 需求文档追踪

机器人能够在三维空间内平稳地进行水平移动：机器人运动（手动控制）的功能需求。

能够识别障碍物并进行躲避：机器人自动避障的功能需求。

能够接收传感器信号建立实时地图、能够进行动态路径规划及导航控制：机器人自动导航的功能需求。

## 正向需求追踪

### 手动操控机器人

用户可以通过语音、触摸屏对机器人进行手动操控。机器人可以根据用户发出的指令进行相应的移动操作。

需求追踪：在设计阶段，设计运动类MOVE\_ROBOT负责操控机器人的运动，设计语音类VOICE负责管理语音输入的部分。在编码阶段，实现驱动硬件的运动控制接口set\_x\_speed、set\_y\_speed、set\_ angular\_speed等等，并实现驱动语音硬件的接口get\_keyword。用户通过GUI界面调度总控系统，总控系统完成消息传输，使得机器人能够根据用户的指令运动。

### 机器人自动导航

机器人可以接受用户输入的地点，并根据已经建立的周围障碍物地图进行自动路径规划，根据规划的路径自动导航至目的地。

需求追踪：在设计阶段，设计模块SLAM\_MAP，负责建立并存储周围障碍物的地图，模块PLAN\_ROUTE内相连路径规划的算法，负责规划路径，模块NAVIGATION负责接受规划的路径，并根据该路径调用机器人运动模块，引导机器人根据路径移动。在编码阶段，实现驱动传感器识别并存储的接口函数send\_build\_map\_signal、send\_save\_map\_signal，实现send\_nav\_point、get\_nav\_state等规划路径以及导航的接口。

### 机器人避障

机器人在运动前，可以根据静态障碍物的位置规划出路径，并根据该规划的路径进行行进，此静态避障需求在机器人自动导航的部分完成。机器人运动过程中，可以对前行中不在地图中的动态障碍物进行识别，并进行避障。

需求追踪：在设计阶段，模块SENSOR用于与传感器实时交互。在编码阶段，实现get\_sensor\_state接口，可以在运动过程中得到前方是否有障碍物的信息，若有障碍物，机器人停止并重新规划路径。实现驱动语音硬件进行障碍物警示播报的send\_alarm\_signal接口，在遇到障碍物时diaoyong 声音模块，向用户发出有障碍物的提示。

### 非功能需求

在非功能需求中，性能、可靠性、可拓展性、可用性、安全性等需求。

需求追踪：非功能需求的满足应当在编码与设计的每个部分进行考虑，例如路径规划时间的限制可以通过优化算法来进行保证、机器人紧急制动时间的限制可以通过制动加速度的调整来进行保证。

机器人的可靠性、可用性、可拓展性需求的满足将侧重在测试阶段与修复阶段，设置测试样例并对机器人的性能进行评测，对于不满足条件的部分进行修复。

机器人的安全性应该由硬件部分进行保证，在整个迭代过程中检查机器人硬件的安全问题， 如有问题需即使对硬件进行维修或更换。

## 需求追踪矩阵

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 模块  需求 | 运动模块 | 语音模块 | 传感器模块 | 建图模块/存图模块 | 路径规划模块 | 导航模块 |
| 手动控制 | √ | √ |  |  |  |  |
| 自动导航 | √ |  | √ | √ | √ | √ |
| 避障 |  | √ | √ |  |  |  |
| 非功能需求（部分） | √ |  |  |  | √ |  |

表格 8‑1 数据字典信息表