**智能巡检机器人**

**软件设计文档**

SDD TEAM 305

**V2.0**分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 | 啥也不会闷头干小组 | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 18373165 | 金佳力 | 第六部分路径规划、图像识别、报警模块 |
| 18373073 | 张昊东 | 第六部分导航、自主避障模块 |
| 18373441 | 覃启浩 | 第四部分硬件体系结构、第五部分用户界面 |
| 18373023 | 吕果 | 文档审核、第六部分建图模块、第八部分需求可追踪性说明 |
| 18373089 | 肖旻远 | 文档主编、第一部分范围、第二部分需求概述、第三部分数据库设计、第四部分软件体系结构、第五部分软硬件接口、第六部分控制中心模块 |

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 | 4.20 | 肖旻远 | 吕果 | 第一版 |
| 1.1 | 4.26 | 肖旻远 |  | 完善了第二部分，修改了一些错误 |
| 1.2 | 4.27 | 肖旻远 金佳力  覃启浩 | 肖旻远 | 第四部分添加了时序图，第五部分更新了用户界面，第六部分增加了流程图 |
| 1.3 | 5.18 | 肖旻远 | 肖旻远 | 修改了第四部分总体协作图 |
| 2.0 | 6.21 | 肖旻远 | 肖旻远 | 修改了第三部分数据库设计 |

目录

[1.范围 5](#_Toc14265)

[1.1项目概述 5](#_Toc17090)

[1.2文档概述 5](#_Toc29332)

[1.3术语和缩略词 6](#_Toc29703)

[1.4 引用文档 6](#_Toc29157)

[2.需求概述 6](#_Toc4826)

[2.1业务需求 6](#_Toc15557)

[2.2数据需求 7](#_Toc17233)

[2.3功能需求 7](#_Toc25265)

[2.4 非功能需求 9](#_Toc27578)

[3.数据库设计 10](#_Toc23188)

[4.体系结构设计 11](#_Toc29143)

[4.1总体结构 11](#_Toc15241)

[4.2软件体系结构 13](#_Toc16180)

[4.2.3时序图 15](#_Toc9309)

[4.3硬件体系结构 16](#_Toc11540)

[4.4技术体系结构 17](#_Toc4172)

[5.接口设计 18](#_Toc15547)

[5.1用户界面设计 18](#_Toc2715)

[5.2硬件接口 21](#_Toc26406)

[5.3内部软件接口 22](#_Toc32755)

[6.详细设计 24](#_Toc26483)

[6.1控制中心 24](#_Toc9133)

[6.2建图模块 24](#_Toc15743)

[6.3路径规划模块 26](#_Toc20596)

[6.4自主导航模块 26](#_Toc21130)

[6.5避障模块 27](#_Toc18139)

[6.6图像识别模块 28](#_Toc14302)

[6.7报警模块 30](#_Toc485)

[7.运行与开发环境 32](#_Toc30667)

[7.1运行环境 32](#_Toc22676)

[7.2 开发环境 32](#_Toc29565)

[8.需求可追踪性说明 33](#_Toc7397)

[8.1　功能需求追踪 33](#_Toc23664)

[8.2　非功能需求追踪 34](#_Toc11257)

[8.3 需求追踪表 35](#_Toc3724)

# 1.范围

### 1.1项目概述

巡检机器人是以移动机器人为载体，摄像机等检测仪器为载荷系统，嵌入式计算机为控制系统的软硬件开发平台。主要具有障碍物检测，自动导航，火警识别与报警等功能。

当下，物业公司保安人员难招、招来后流动率极高的现象已不再是某个企业或某个城市的问题，而是当前中国物业管理行业的通病，而随着经济的发展，人工智能逐渐出现在人们的视线，智慧化园区、智能化社区的建设为人们带来了一系列的便利。巡检机器人是其中的重中之重。它能够代替传统的安保人员，在商场，社区等地方进行自动安全巡检，从而帮助企业减少招工难、管理难、运营成本高一系列的难题。

巡检机器人最基本的功能是自动巡逻。在用户预先设定完巡逻路线后，巡检机器人能够按照预设路线自主导航，并在遇到障碍物时自动避障。

巡检机器人的异常监控主要包括火灾情况和非法人员入侵两部分。机器人通过携带的摄像机采集图像并传送至服务器，软件利用图像识别技术判断是否有异常情况发生，当有异常数据出现时，软件会进行迅速的报警，帮助人员短时间采取行动处理好问题。

除上述功能外，该系统还会具有稳定可靠，巡检覆盖率高，报警准确率高等非功能性指标。

### 1.2文档概述

##### 1.2.1用途

本文档在SRS的基础上，根据需求进行具体的软件设计。包括数据库设计、体系结构设计、接口设计、详细设计这几个部分。SDD将指导我们进行后续的具体开发。

##### 1.2.2内容组织

内容组织如目录所示，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。

### 1.3术语和缩略词

##### 1.3.1 术语

嵌入式系统:一种专用的计算机系统，作为装置或设备的一部分， 用于控制、监视或者辅助操作机器和设备。 通常是一个控制程序存储在ROM中的[嵌入式处理器](https://baike.so.com/doc/485483-514107.html)控制板。

Ubuntu:一个以应用为主的Linux操作系统。

##### 1.3.2 缩略词

ROS:全称为Robot Operating System，是一个机器人软件平台，它能为异质计算机集群提供类似操作系统的功能。

### 1.4 引用文档

表1.4.1 引用文档表格

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 标题 | 版本 | 发行日期 |
| 1 | SDP-软件开发计划 | V1.3 | 2021/3/15 |
| 2 | SRS-需求规格说明书 | V1.2 | 2021/4/27 |
| 3 | 启智ROS机器人说明书 | - | - |

# 2.需求概述

### 2.1业务需求

本项目旨在设计巡检机器人，主要功能是在社区，商场等场所自动巡逻，当检测到有火警或非法人员入侵时进行报警。

在使用机器人前，需要对机器人进行初始化。可以用手推或手柄控制机器人遍历场地，机器人会自动对地图进行建模。也可以将3D地图直接载入。除此之外，还需要将机器人和用户的设备进行连接，以便当有可疑情况发生时，对用户进行报警。

初次使用机器人时，需要用户设置巡逻路线，机器人会根据保存的地图模型对路线进行微调，之后便会按照该路线巡逻。用户之后可以更改巡逻路线。

机器人在巡逻遇到障碍时，会进行自动避障。若因障碍物过大或其他因素，机器人被迫偏离预定路线，则会进行自动规划路径，回到预定路线。当机器人检测到有可疑的情况发生(如火灾，非法人员入侵)时，会对用户进行报警，并将拍摄到的可疑画面传给用户。

### 2.2数据需求

本项目共涉及用户、机器人、环境、路径、险情共五个实体：

用户实体的属性包含界面操作数据，可以完成控制机器人启动和关闭、设备连接、设置巡逻路线、主动控制机器人移动和操作设置等功能；

机器人实体的属性包括自身位置数据、自身方向、运动速度等数据。

环境实体的属性包括环境位置和环境地点；

路径实体的属性路径具体的数据。

险情实体包括险情类型、险情位置、险情信息属性。

### 2.3功能需求

##### 2.3.1地图建模

参与者:管理员，机器人

目标:使机器人能正确建立所在场景的地图

场景:

1.管理员启动机器人，并将其连接到机载平板电脑上。

2.管理员通过用手推或手柄控制等方式控制机器人遍历场景。

3.机器人通过摄像头，雷达等传感器获取地图信息。

4.机器人根据获取的地图信息进行实时建模，并在机载平板电脑上显示。

5.管理员根据显示的建模状况控制机器人运动，直至建模完成。

##### 2.3.2设置巡逻路线

参与者:管理员，机器人

目标:设置机器人巡逻的路线

场景:

1.管理员启动机器人。

2.机器人在机载平板电脑上显示地图模型。

3.管理员在模型上画出大致的巡逻路线。

4.机器人根据地图细节信息对路线进行微调。

5.管理员继续调整直至满意为止。

6.保存路线。

##### 2.3.4自动避障

参与者:机器人，机器人内部控制器

前置条件:机器人已设置巡逻路线，且能自由活动，且运动过程中机器人遇到障碍。

目标:机器人成功绕开障碍物并回到预定路线

场景:

1.机器人探测到障碍物。

2.机器人停下。

3.机器人探测障碍物周围信息。

4.机器人重新规划路线。

5.机器人按照重新规划的路线运动。

##### 2.3.5报警控制

参与者:机器人，用户(保安)

前置条件:机器人摄像头正常，与用户的设备保持连接，且机器人探测到可疑情况发生。

目标:机器人对用户进行报警，并发送可疑图像。同时，机器人继续观察。

场景:

1.机器人探测到可疑情况发生。

2.机器人停下。

3.机器人对用户进行报警，并将发现状况的可疑图像发送给用户。

4.机器人提高摄像频率，对附近进行观察，并将图片发送给用户。

##### 2.3.6手动操控

参与者:用户，机器人

前置条件:机器人可自由运动

目标:由用户来完全操纵机器人

场景:

1.用户进入操作界面。

2.用户启用人工控制。

3.用户通过手柄或键盘，控制机器人进行平移或旋转。

### 2.4 非功能需求

##### 2.4.1性能属性

1. 响应时间：
   1. 机器人得到指令后，响应时间尽量短，后台响应时间不超过1s，用户UI得到反馈不超过1s。
   2. 机器人探测发现路径上有障碍物时，应能够在 2s 内做出响应，进行避障。
2. 功耗
   1. 系统功耗应该限定在一定的范围内，低于20W。
   2. 系统应当节约能源，设定自动待机模式，当机器人未被使用超过10分钟后，系统待机。
3. 处理能力
   1. 系统应该对CPU和存储空间有较高的利用率。
   2. 机器人可以判断、躲避障碍并重新规划路线。

##### 2.4.2质量属性

1. 系统可用性
   1. 系统应该给出操作指南，方便用户学习使用。
   2. 系统的界面应该做的简单易懂、清晰。
   3. 给出文本提示或语音提示。
2. 可移植性
   1. 机器人在ROS系统运行，应当在大部分ROS系统中兼容。
   2. 预留一定接口，用于未来可能的新增需求，保持系统的高内聚低耦合性质。
3. 健壮性
   1. 能够识别未定义指令，并给出提示。
   2. 当故障发生时，系统能够给出正确的错误处理，而不是停止运行。
4. 可靠性
   1. 系统故障率应该保持在较低水平，发生概率在1%概率以下。当故障发生时，系统不会对自身和周围环境造成破坏。

# 3.数据库设计

本项目中，数据库部分的使用主要集中在两个模块。一是登录部分，二是报警模块。在登录部分，我们需要存储用户名和密码信息。在报警模块，需要保存报警图片、报警位置和时间等信息。综上所述，我们主要需要以下几个数据表来实现。

表3.1 用户信息表

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| 用户名 | float | - |
| 密码 | float | - |

表3.2 报警信息表

|  |  |  |
| --- | --- | --- |
| 属性名 | 类型 | 说明 |
| x坐标 | float | 报警x坐标 |
| y坐标 | float | 报警y坐标 |
| 时间 | time | 报警时间 |

E-R图如图所示：

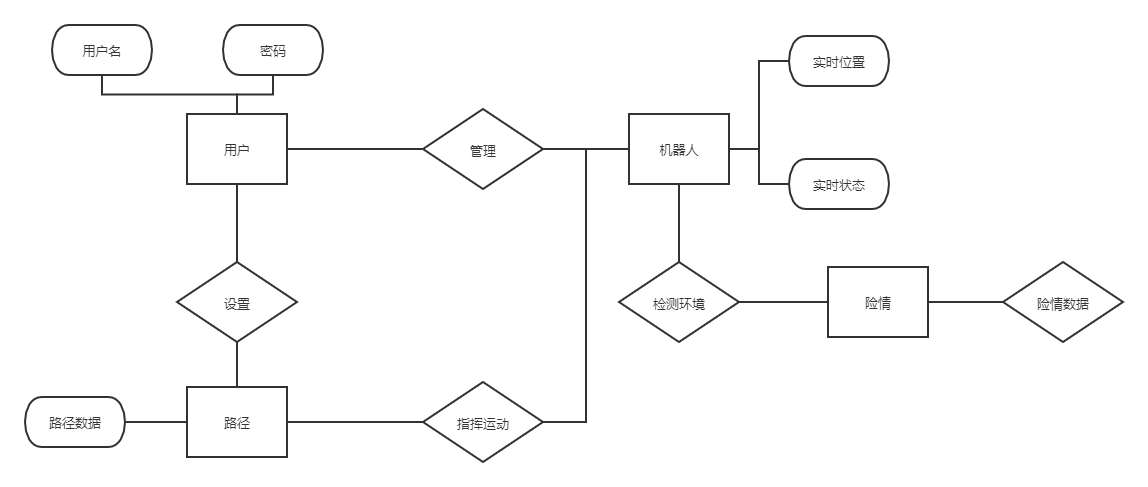


图3.1 E-R图

# 4.体系结构设计

### 4.1总体结构

系统总体层次结构如图所示，从上到下依次为用户界面、软件接口、硬件接口和硬件设备四个层次。硬件设备主要包括雷达、相机、机器人运动装置等设备。硬件接口是对这些硬件的抽象封装。软件接口部分负责将用户界面下达的指令进行处理，包括控制中心和一些具体的模块。

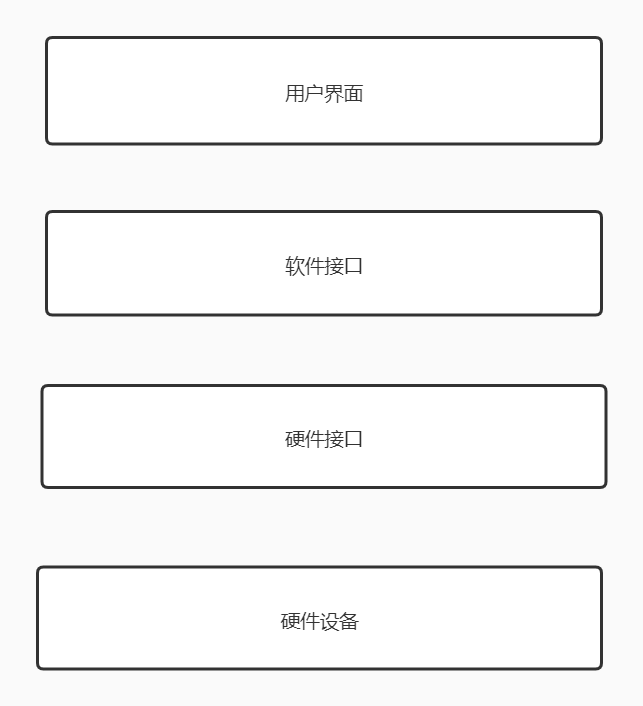


图 4.1.1 总体层次结构

整体结构构建图如下：

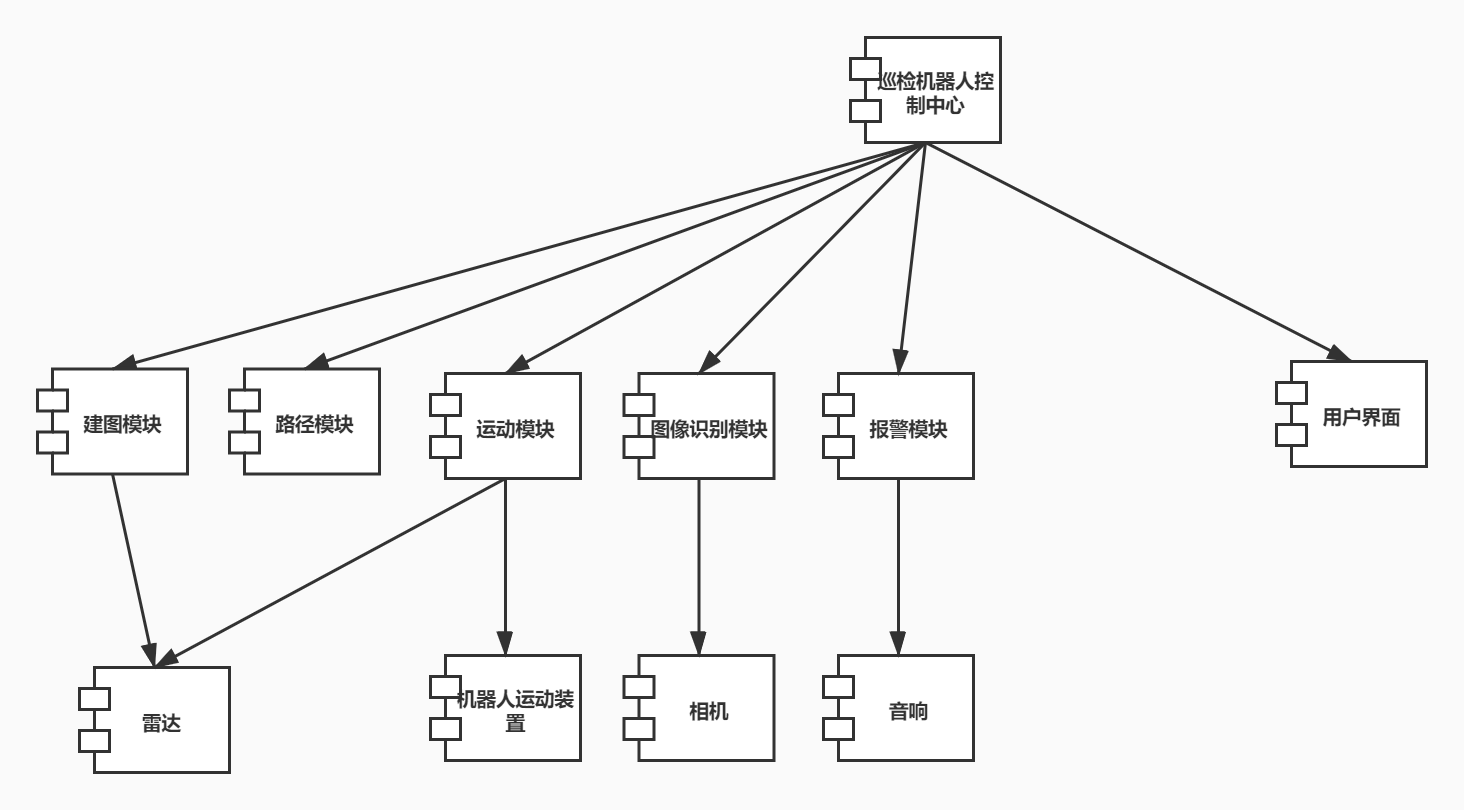


图 4.1.2 整体构件图

### 4.2软件体系结构

##### 4.2.1 软件协作图

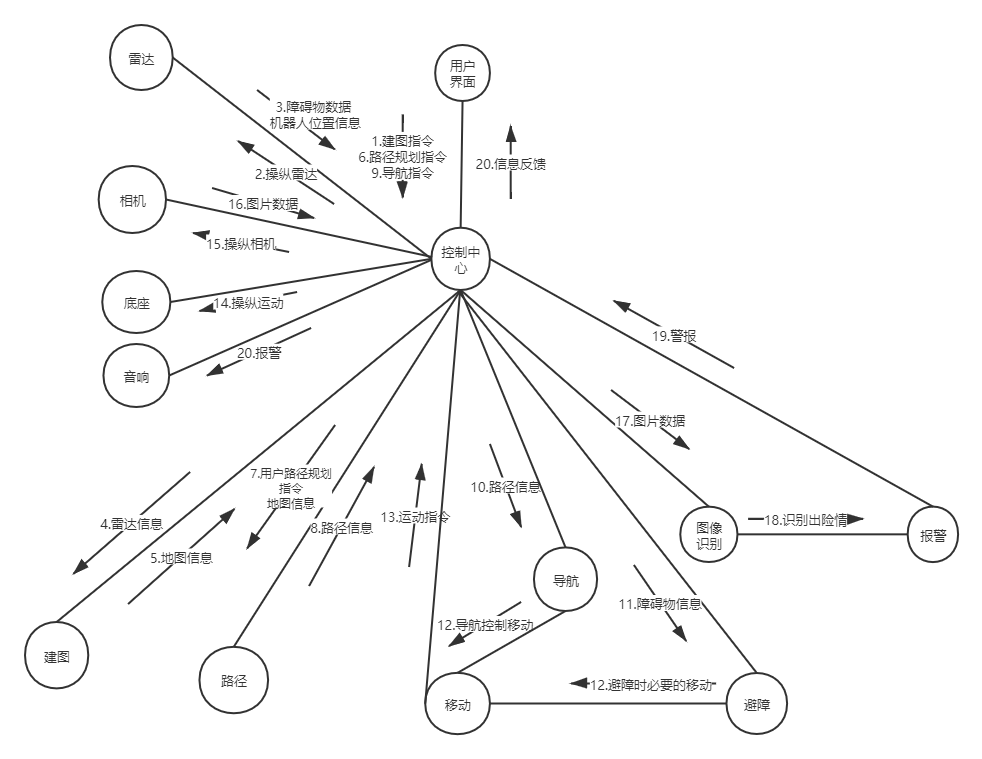


图4.2.1 软件协作图

如图所示，整个软件结构的核心是控制中心部分。这部分一方面负责用户和机器人的交互，另一方面负责机器人软硬件之间的交互。随着用户指令的下达，控制中心将其分配给相应的模块，模块进行处理后向控制中心提出申请，控制机器人硬件进行运动、拍摄等相关操作。

##### 4.2.2 软件整体类图

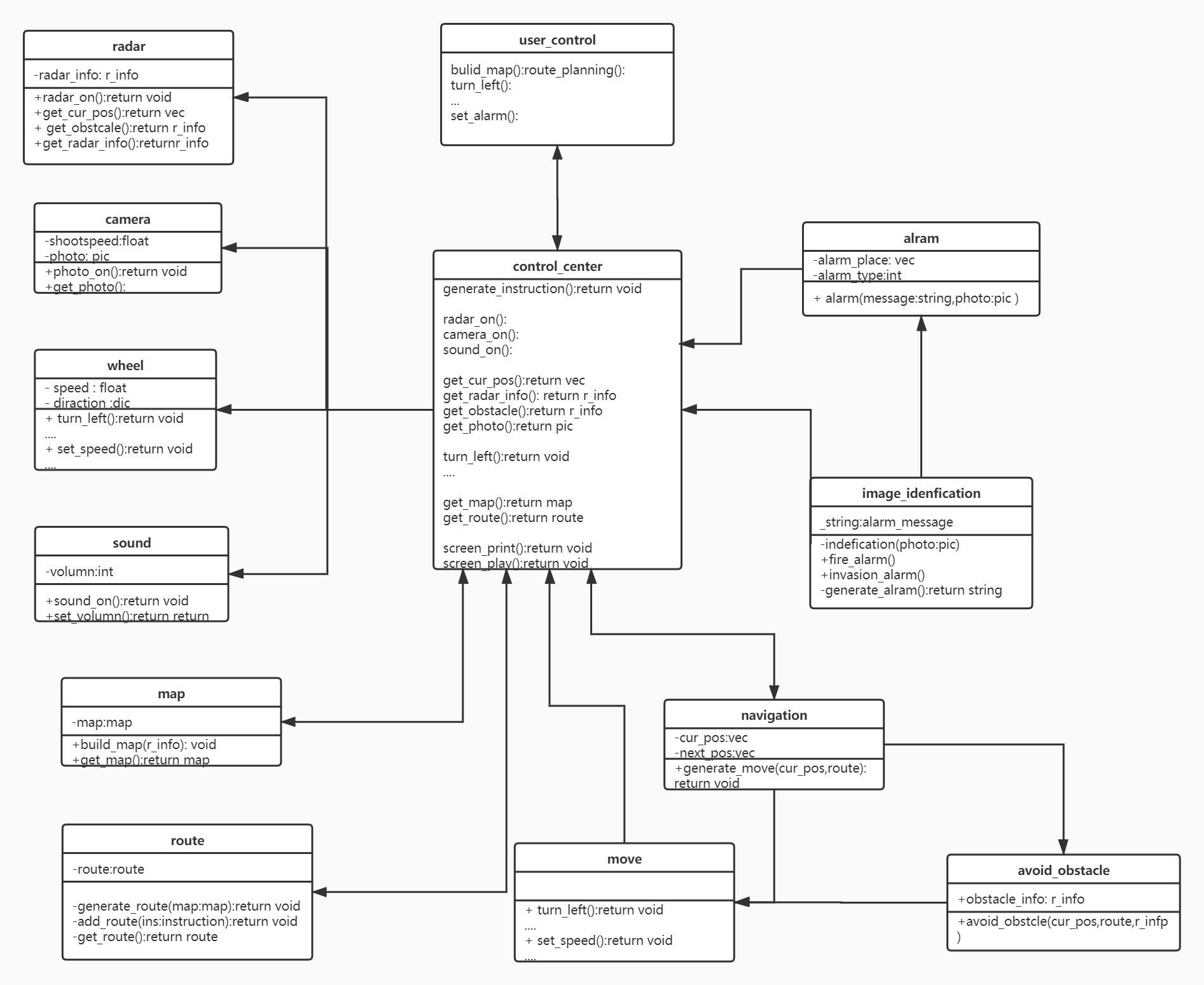


图4.2.2 软件整体类图

根据软件整体结构，画出整体类图。总共抽象出12个类。在第五部分中对每个类会有详细介绍。

### 4.2.3时序图

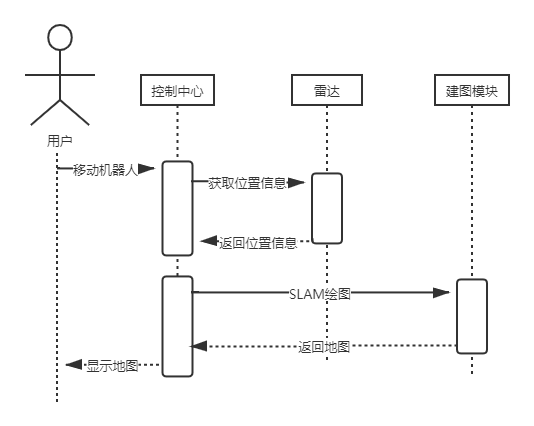


图 4.2.3 时序图（建图部分）

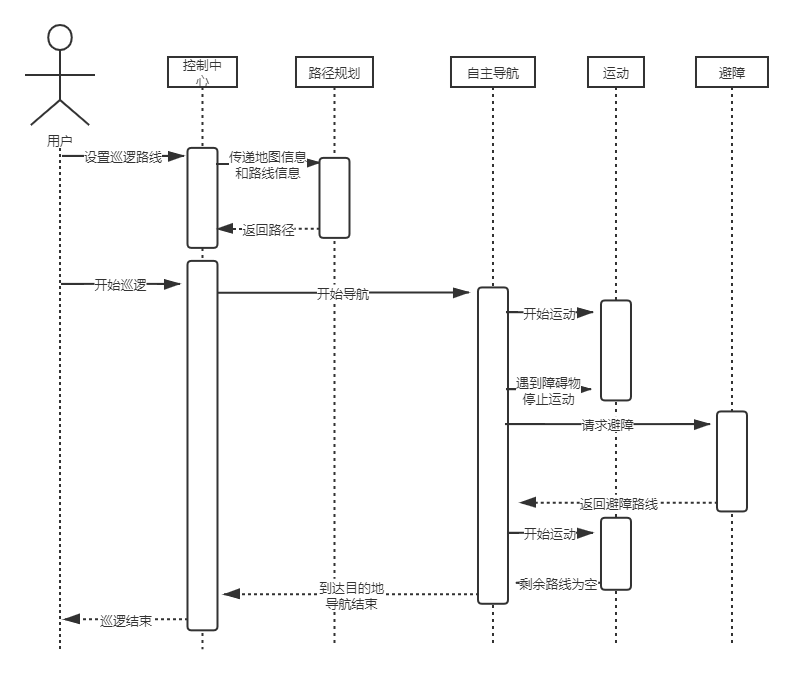


图 4.2.4时序图（运动部分）

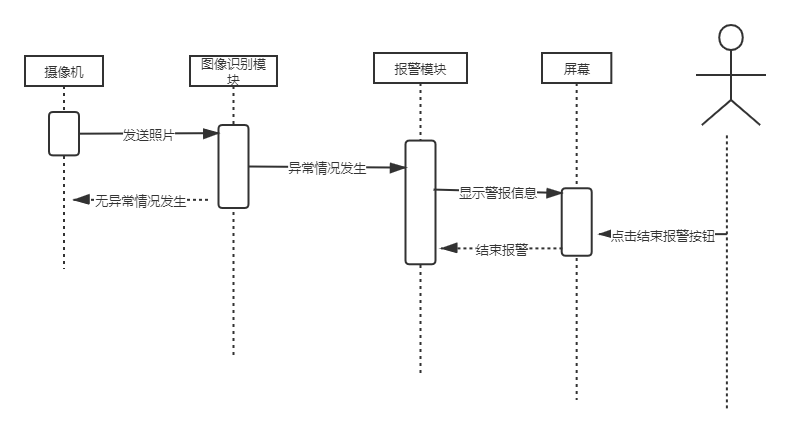


图 4.2.5 时序图（报警部分）

### 4.3硬件体系结构

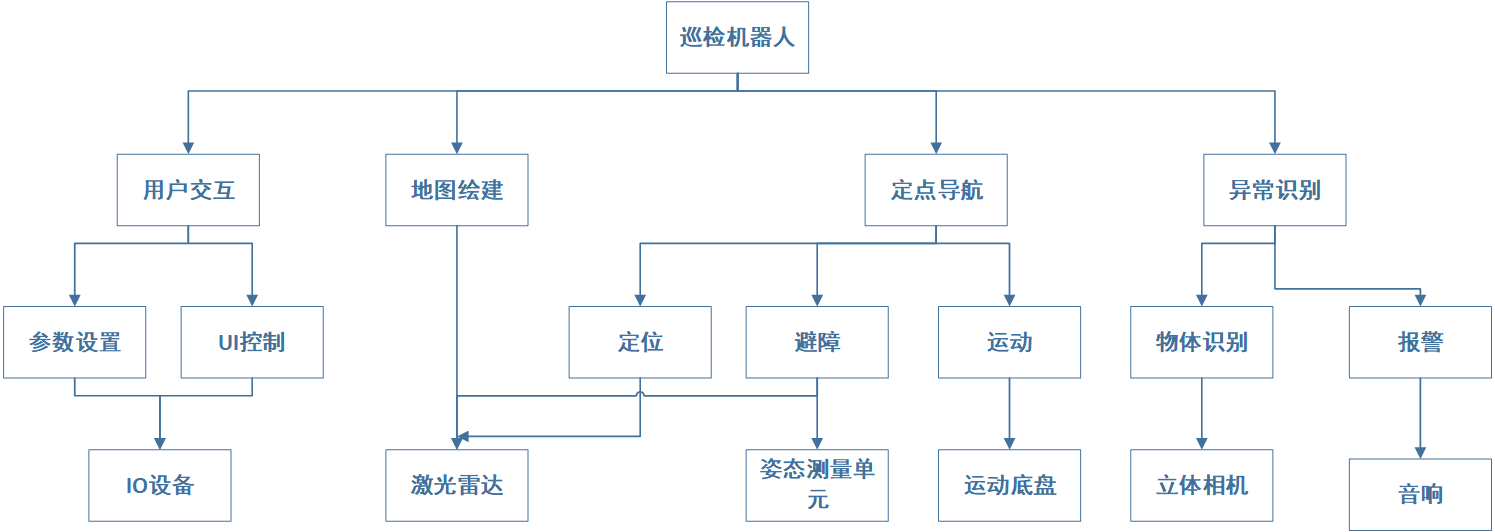


图4.3.1 硬件体系结构

巡检机器人涉及的硬件有IO设备，激光雷达，姿态测量单元IMU，运动底盘、立体相机和扬声器。

表4.3.1 各硬件作用

|  |  |
| --- | --- |
| 硬件 | 作用 |
| IO设备 | 在用户交互时设置参数，在UI界面为机器人下达命令并接收反馈。 |
| 激光雷达和姿态测量单元IMU | 激光雷达结合SLAM技术可以绘建地图，在机器人移动阶段定位，配合姿态测量单元IMU进行避障，使得机器人按照规划的路线自主导航到达目的地。 |
| 运动底盘 | 驱使机器人移动 |
| 扬声器 | 对于报警和报错情况给出语音反馈 |

各硬件设备按上图依赖关系共同协作以支持机器人的运行。

### 4.4技术体系结构

本项目的巡检机器人主要用到以下几个技术：IMU姿态传感、SLAM环境建图、Navigation导航模块、物品识别。

表4.4.1技术简介

|  |  |
| --- | --- |
| 技术名称 | 简要介绍 |
| IMU姿态传感技术 | 通过ROS机器人的IMU单元实时获取机器人的滚转、倾斜和朝向信息。 |
| SLAM环境建图 | 通过激光雷达，实时扫描机器人周围的障碍物分布情况，创建环境地图。 |
| Navigation导航模块 | 根据里程计等传感器的信息流和机器人的全局位置，通过导航算法计算得出机器人速度控制指令。 |
| 物品识别 | 通过相机获得物品的外形、尺寸和三维空间坐标。 |

技术架构图如下：

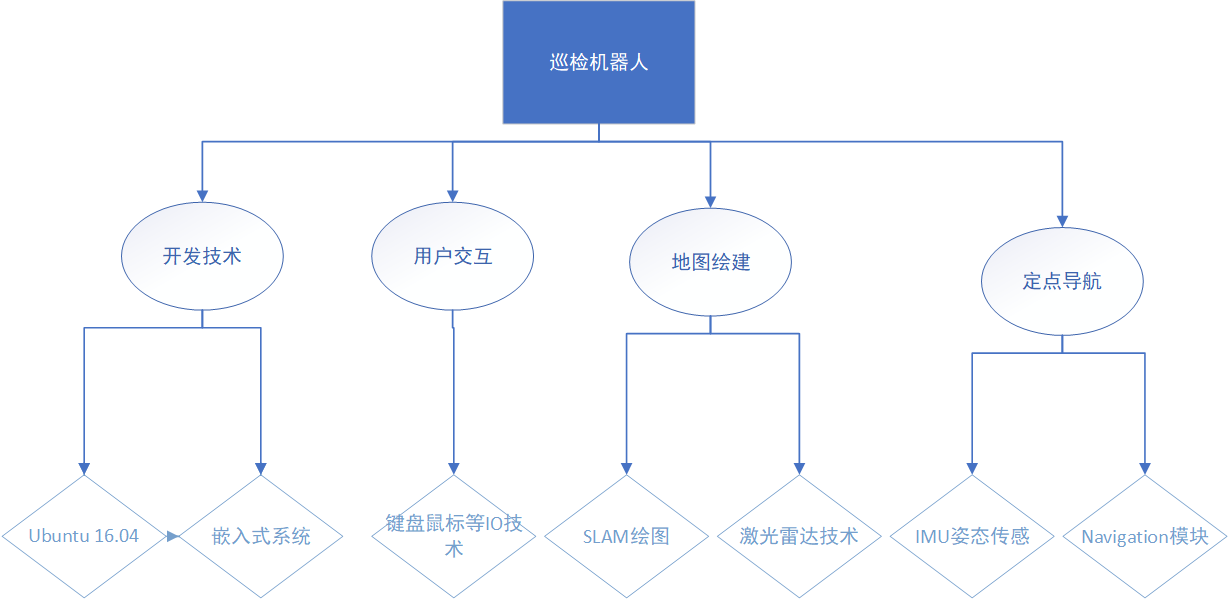


图4.4.1 技术架构图

巡检机器人的技术结构分为四个方面：开发技术、用户交互、地图绘建和顶点导航。

开发技术采用Ubuntu16.04操作系统和嵌入式系统的软硬件技术。

用户交互主要通过鼠标和键盘灯IO设备来实现。

地图绘建功能依托SLAM算法和激光雷达技术。

定点导航功能需要ROS的IMU姿态传感器和ROS的Navigation模块。

# 5.接口设计

### 5.1用户界面设计

##### 5.1.1主界面

在主界面，用户可以看到摄像头的实时图像、报警新鞋和地图，能控制设备开关，还可以通过按钮进入地图建模界面、设备连接界面、查看巡逻路线、查看报警情况、开始自动巡检模式和进入手动控制模式。



图5.1.1 主界面

##### 5.1.2地图建模界面

地图建模界面显示了当前的地图模型和摄像头实时拍摄到的画面，用户可以通过按钮手动控制机器人移动，并根据图像信息进行地图建模。

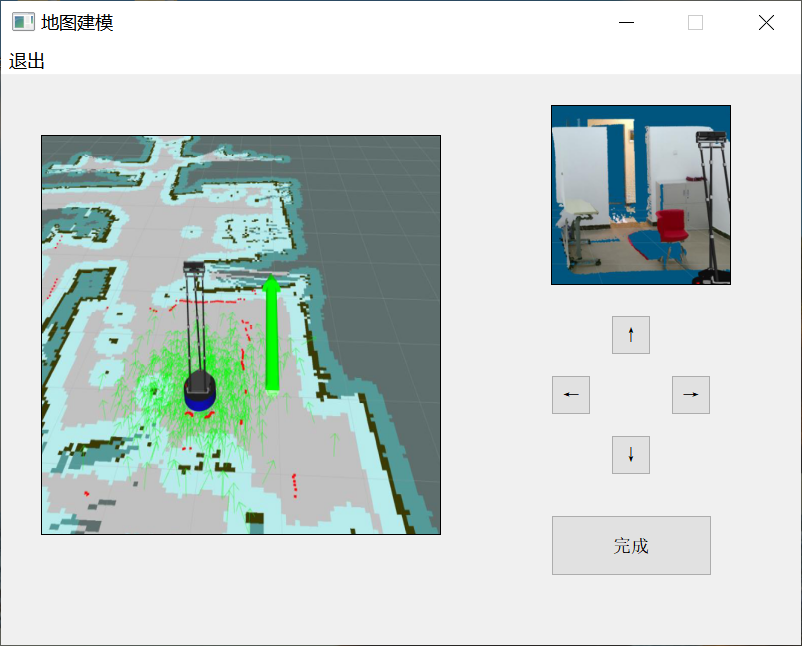


图5.1.2 地图建模界面

##### 5.1.3巡逻路线设置界面

在巡逻路线设置界面中，用户可在已建好的地图模型上通过交互界面的按钮，设置自动巡逻起点、巡逻终点和巡逻路线。

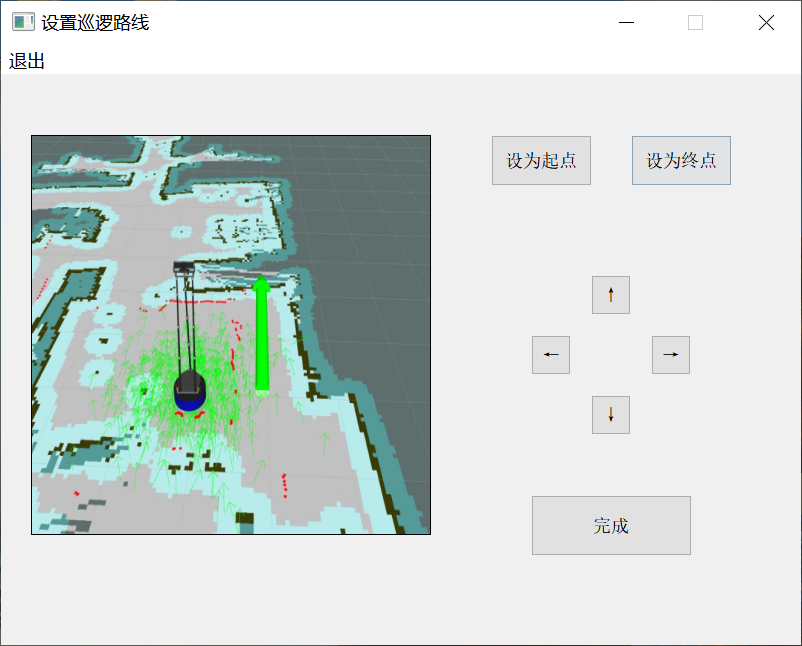


图5.1.3 巡逻路线设置界面

##### 5.1.5手动控制界面

手动控制界面与自动巡检模式界面基本一致，用户可以手动操作机器人自由移动，机器人在遇到报警情况时同样会显示报警信息。

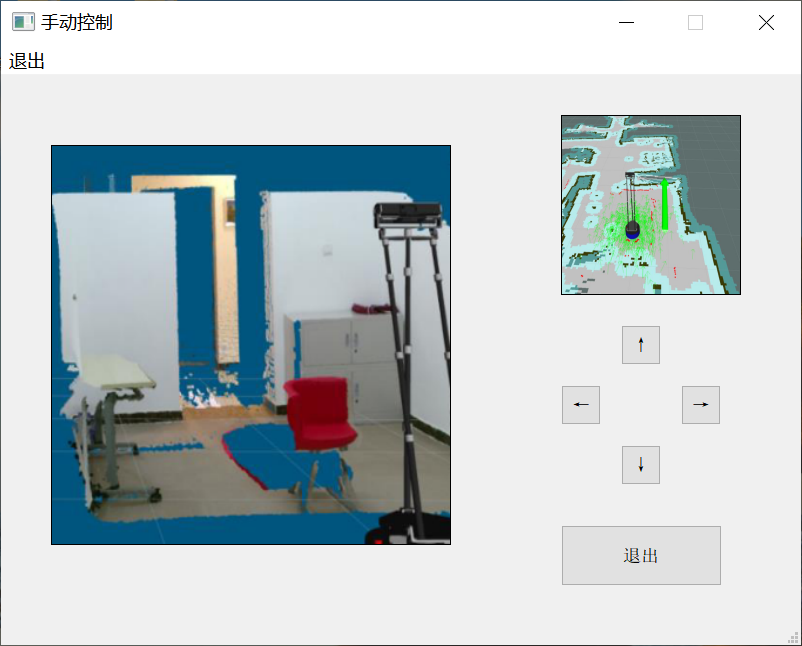


图5.1.5 手动控制界面

### 5.2硬件接口

启智ROS机器人拥有多种传感器。在本项目中，我们用到激光雷达、姿态测量系统、立体相机三种传感器。下列内容参考自启智机器人官方介绍。

##### 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度，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。

##### 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最新一代的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。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，实现火情识别、非法人员入侵识别的功能。

## 5.3内部软件接口

##### 5.3.1 用户交互部分

user\_control类：此部分实现了用户对机器人的所有操作。 用户通过GUI界面输入的所有指令都将从此类转移到内部软件系统。类中调用实现了turn\_left()、speed\_up()、stop()、route\_planning()等方法，实现用户对机器人的手动运动控制、路径规划等操作，这些指令会传送到center\_control类中进行下一步处理。

##### 5.3.2 控制中心

center\_control类：此部分是机器人的指令汇总中心。机器人的所有指令都将通过该类设置或访问机器人相关数据，实现软硬件之间的交互。center\_control主要完成两类工作。第一类是硬件控制工作，几个模块类可以通过控制中心调用几个硬件类（camera类、radar类等）来控制所有硬件的开关和调整相关参数，还能从硬件中获取相关数据，例如从camera类中调用get\_photo()获取拍摄的图像，利用radar类的get\_position()获取机器人的位置等。第二类是用户交互工作，例如图像处理模块将险情从这里向用户UI进行转发，运动部分将机器人实时位置向用户UI进行转发。

* get\_cur\_pos():获取当前机器人位置
* get\_radar\_info(): 获取雷达数据进行建图
* get\_photo():获取相机拍摄结果
* move\_forward():机器人前进
* move\_backward():机器人后退
* turn\_left():机器人左转
* turn\_right():机器人右转
* get\_map():获得地图
* get\_route():获得路径
* screen\_print():用户界面输出文字
* screen\_play()用户界面输出图片

##### 5.3.3 硬件部分

硬件部分对应本项目中使用到的机器人硬件，主要包括radar类、camera类等。每个类都提供开关方法。

camera类中有get\_photo()方法来获取拍摄图像。

radar类中有get\_position()方法获取机器人的位置，get\_obstacle()方法来获取障碍物信息，get\_radar\_info()获取雷达数据实现建图。

##### 5.3.4 地图部分

map类：地图部分的核心是map类。map类中实现了build\_map()、save\_map()、read\_map()三个方法实现建图、存图、读图的操作。

##### 5.3.5 路径部分

route类：路径部分的核心是route类。route类中实现了generate\_route()、save\_route()、get\_route()三个方法实现生成路径、存储路径和读取路径。

##### 5.3.6 机器人运动部分

navigation类：自主导航类是机器人运动的核心控制类。通过控制中心中的get\_route()方法，获取路径。之后根据获取到的路径，navigation类可以直接调用move类完成机器人运动。此外，navigation类还需要调用obstacle\_avoid类实现自主避障。

move类：move类是直接负责机器人运动的类。实现了turn\_left()、turn\_right()、speed\_up()、speed\_down()等方法，navigation调用这些方法实现对机器人运动的控制。

obstacle\_avoid类：避障类。通过指令中心调用radar类的has\_obstacle()方法，来进行选择是否需要避障。如果监测到障碍物，就调用move类进行避障运动和避障后重新回到路径的运动。

##### 5.3.7 图像识别部分

image\_identification类：image\_identification类是图像处理部分的核心。该类通过调用控制中心get\_photo()获取相机的拍摄图像，再对图像进行处理。如果有险情直接调用generate\_alarm()方法完成报警。

alarm类：alarm()类中的generate\_alarm()方法生成警报，被调用后会向控制中心报警，控制中心会向用户界面生成报警信息。

# 6.详细设计

### 6.1控制中心

控制中心是本项目中最重要的部分。向上负责和用户界面进行信息交互，负责将用户指令分发到相应模块中，并将位置信息、报警信息反馈给用户。向下负责软硬件之间的交互，每个模块对指令进行处理后，会向控制中心请求访问、操纵硬件。例如建图指令下达给建图模块后，该模块会向控制中心索要雷达的数据，这时控制中心会访问雷达相关数据，再分配给建图模块。

### 6.2建图模块

建图模块会首先检测激光雷达、姿态测量单元等硬件设备是否能正常使用，如果正常则继续执行，否则显示错误信息；确认硬件正常后，管理员可以操作机器人在场地各个角落移动，机器人获取雷达、测量仪的信息并根据SLAM算法实时进行地图建模，循环操作直到建图结束，最后保存地图信息即可。

流程图：

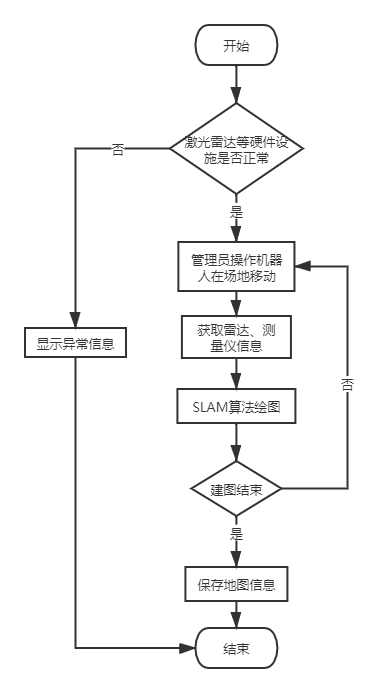


图6.2.1 建图模块流程图

类图：

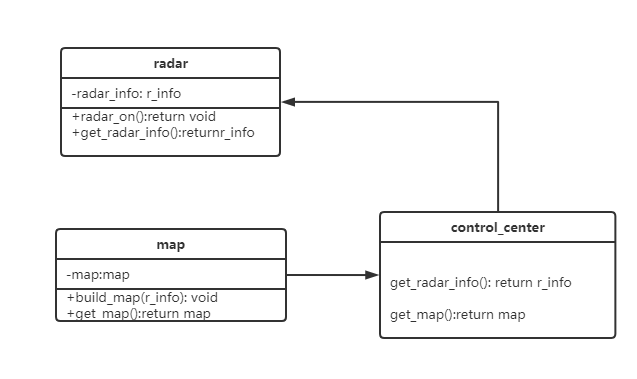


图6.2.2 建图模块类图

伪代码：

1. void build\_map()
2. {
3. if equipment\_abnormal() {
4. control\_center.SCREEN\_print(“设备异常！”);
5. } else {
6. while(control\_center.slam()){ //当建图模式启动
7. generate\_map(control\_center.get\_r\_info()); //使用雷达信息建图
8. }
9. }
11. }

### 6.3路径规划模块

路径规划模块是利用用户在UI给出的路径指令，进行分析处理生成最终路径。需要从control\_center获得用户指令数据。

伪代码：

1. **void** Route\_Plan ()
2. {
3. route=CONTROL\_CENTER.get\_user\_input\_message();//ROS系统从GUI界面获取用户设计的巡逻路线
4. While(check(route)==false){//如果用户规划的路线不合法
5. CONTROL\_CENTER.show(“无效路线输入”);//提醒用户输入的路线不符合已扫描地图情况
6. route=CONTROL\_CENTER.get\_user\_input\_message();//ROS系统从GUI界面获取用户设计的巡逻路线
7. }
8. While(user\_confirm()==false){//用户对系统设计的路线不满意
9. route=Modify\_route(route);//Ros系统根据实际地图情况修改合适的路线
10. }
11. }

### 6.4自主导航模块

自主导航模块实现的是机器人的导航功能，用户自己在平板电脑上输入大致的巡逻路线，系统会根据已经保存的地图信息，结合用户所设计的路线，对机器人的运动路线进行详细精确的设计，根据设计过的路线进行运动，完成自主导航。

机器人在根据规划好的路径进行运动的同时，系统的避障模块也会同时参与到机器人的运动过程中，雷达等传感器会感知机器人即将运动的路线上是否有障碍物，如果没有障碍物，沿着原定的导航路线进行运动；如果遇到障碍物，机器人会停止运动，避障模块计算出新的运动路线以及运动速和方向，指导机器人躲避障碍后沿着原有的路线完成巡逻活动。

类图：

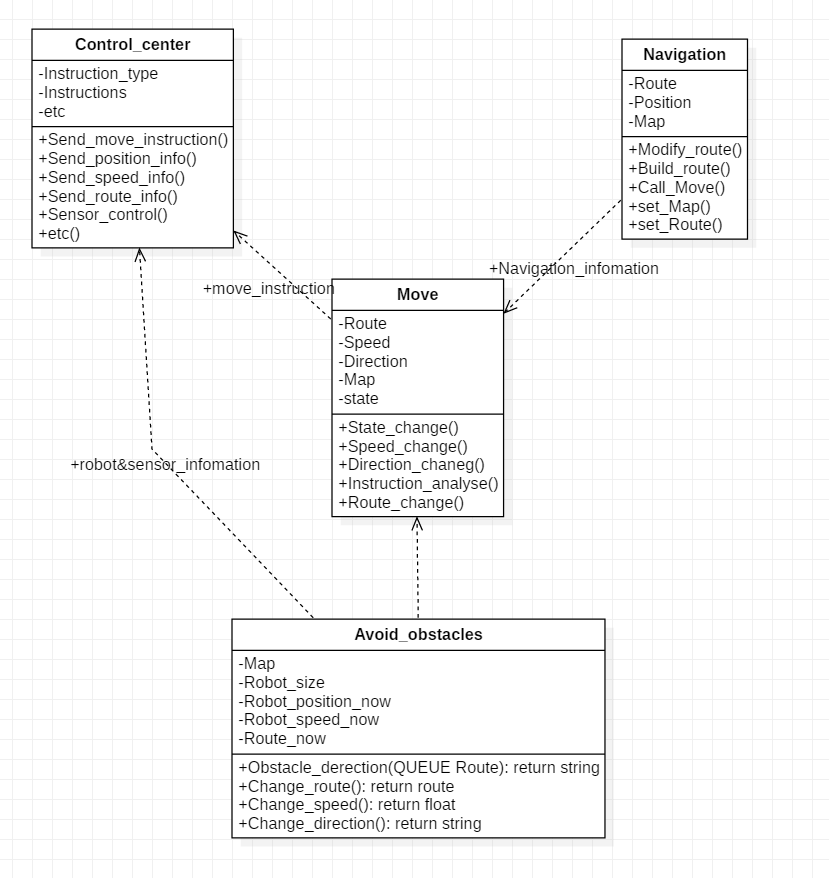


图6.4.1 自主导航模块类图

伪代码：

1. **void** Automatic control ()
2. {
3. Route=Control.get\_route();//从控制中心获得路径
4. Position=Control.get\_position;//从控制中心获得位置信息
5. Move(Route，Position);//通过控制模块开始自主导航的移动
6. Move.obstacles\_avoid();//开启自动避障模块
7. }

### 6.5避障模块

避障模块实现机器人的自主避障功能，能够在运动的过程中遇到障碍物的时候自主设计运动路线，避开障碍物，回到已规划路线。在遇到障碍物的时候，机器人会根据已经保存的地图信息、扫描得到的障碍物信息、以及自身体积形状等方面的数据，计算出一条可以躲避障碍物，并且回归到已规划巡逻路线上的避障路线，同时计算出为了躲避障碍物所需要进行移动的速度和方向，从而达到避障的目的。

类图：

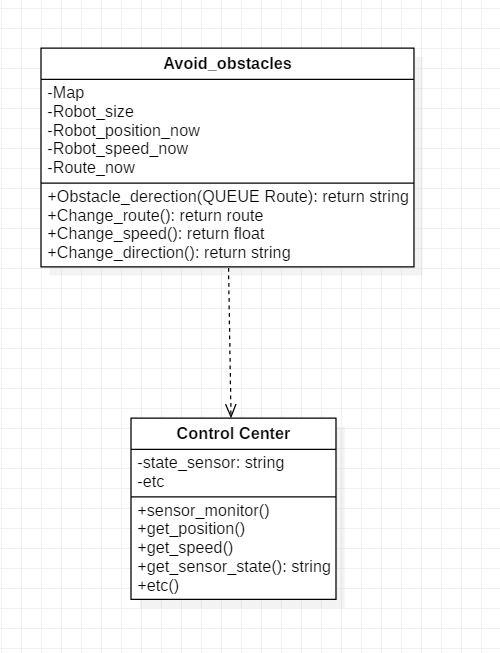


图6.5.1 避障模块类图

伪代码：

1. String avoid\_obstacle(QUEUE route)
2. {
3. Control.sensor\_monitor();//开启障碍物检测
4. **while** Control.get\_sensor\_state() == **true** //检测前方路径有障碍物
5. Change\_route();
6. Change\_speed();
7. Change\_direction();
8. **return** **true**;
9. end **while**
10. }

### 6.6图像识别模块

图像识别模块在接收到传感器传来的图像后，首先对图像进行识别，判断是否有异常情况发生(火灾，不明人员闯入等)。若有异常情况发生，则向报警模块发出指令，使机器人进行报警及应急响应。否则继续等待下一张输入的图片。

流程图：

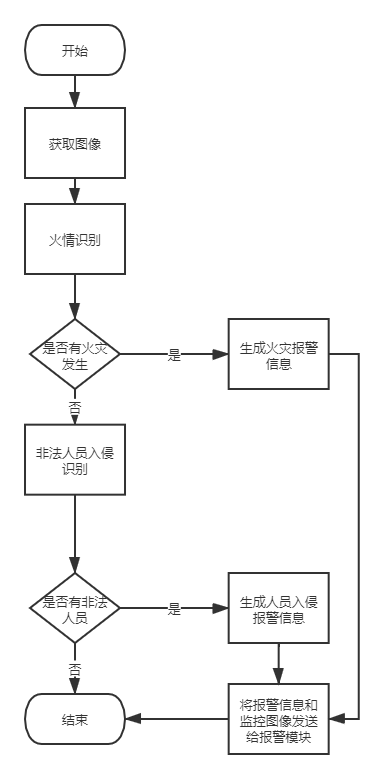


图 6.6.1 图像识别模块流程图

类图：

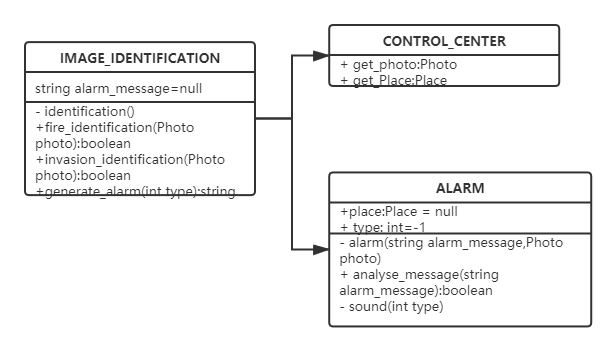


图6.6.1 图像识别模块类图

伪代码：

1. **void** Identification()
2. {
3. Photo photo=CONTROL\_CENTER.get\_photo();//获取图像
4. if fire\_identification(photo) == true //检测有火情发生
5. alarm\_message=generate\_alarm(1); //生成火灾警报信息
6. else if invasion\_identification(photo) == true //检测有非法人员入侵
7. alarm\_message=generate\_alarm(2); //生成人员入侵警报信息
8. if alarm\_message != null
9. ALARM.alarm(alarm\_message, photo);
10. }
12. **string** generate\_alarm(int type)
13. {
14. Place place=CONTROL\_CENTER.get\_place();
15. string message=place + ' ' + type;
16. return message;
17. }

### 6.7报警模块

报警模块在接收到报警信息后，首先判断报警信息是否有效(可识别且该报警情况已被勾选)，若无效则发出短促的报警声并在屏幕上显示”无效的报警信息”语句及完整报警信息。若有效则判断异常情况类型，并进行相应的报警。在开始报警一段时间后，屏幕上会显示“结束报警”按钮，当用户点击该按钮后结束报警。

流程图：

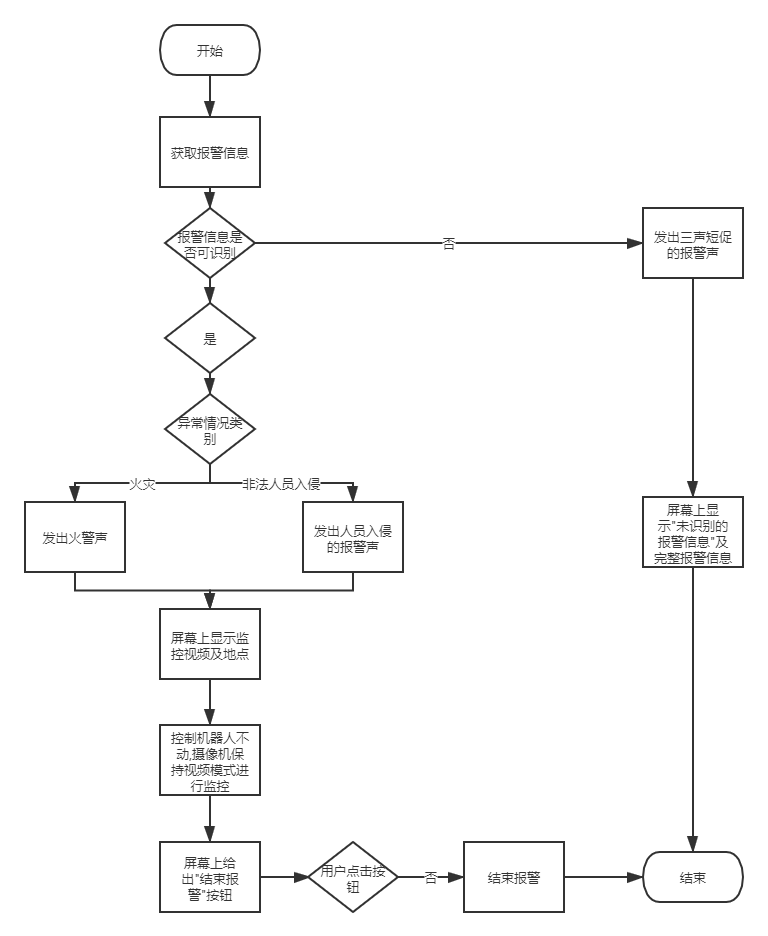


图 6.7.1 报警模块流程图

类图：

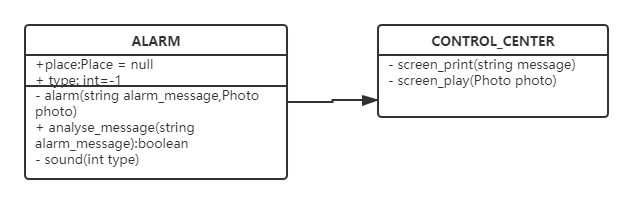


图6.7.2 报警模块类图

伪代码：

1. **void** alarm(string message,Photo photo)
2. {
3. if analyse\_message()==true {//警报信息有效
4. switch(type){
5. case 1:
6. CONTROL\_CENTER.screen\_print("!!!火灾发生!!!");
7. CONTROL\_CENTER.screen\_print("地点:"+place);
8. CONTROL\_CENTER.screen\_play(photo);
9. //屏幕上显示火灾情况
10. sound(1); //发出火警声
11. break;
12. case 2:
13. CONTROL\_CENTER.screen\_print("!!!异常人员侵入!!!");
14. CONTROL\_CENTER.screen\_print("地点:"+place);
15. CONTROL\_CENTER.screen\_play(photo);
16. //屏幕上显示情况
17. sound(2); //发出异常人员侵入的警声
18. break;
19. ...
20. }

# 7.运行与开发环境

### 7.1运行环境

硬件环境：启智ROS机器人，具体参数如下：

表7.1.1 硬件环境参数

|  |  |  |
| --- | --- | --- |
| 名称 | 数量 | 参数 |
| 主控器 | 1 | Intel I3 处理器、4G 内存、128GSSD、触摸屏、键盘 |
| 激光雷达 | 1 | 360°无死角、最大距离 8 米 |
| 视觉传感器 | 1 | Kinect 2 |
| 伺服电机模块 | 3 | 20W 伺服电机、内置驱动 |
| 轮子 | 3 | 3个全向轮 |
| 电池 | 1 | 24V3.5AH 锂离子动力电池 |

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

### 7.2 开发环境

硬件开发环境：一台高性能笔记本电脑。具体参数如下：

表7.2.1 硬件开发环境

|  |  |
| --- | --- |
| 处理器 | i7-8750H |
| 内存 | 8G |
| 操作系统 | 64位 |
| GPU | NVIDIA GTX1060 |

软件开发环境：Ubuntu16.04系统，ROS Kinetic软件包，IDE使用微软Visual Studio Code，使用语言为C++。

# 8.需求可追踪性说明

### 8.1　功能需求追踪

##### 8.1.1　地图建模

本系统要求机器人能对地图进行建模并保存。用户可以在地图建模界面操作机器人移动，并调用建图模块，机器人会实时获取雷达、测量仪的测量数据，并通过SLAM地图建模算法完成建模工作，最终将地图数据进行保存，实现地图建模需求。

##### 8.1.2　设置巡逻路线

本系统要求机器人能在已有的地图上设置巡逻路线并保存。用户在巡逻路线设置界面可以通过操作按钮，调用路径部分的generate\_route()函数，在地图上绘制巡逻路线，绘制成功的路线可以保存到数据库中，需要时通过get\_route()函数进行读取，最终实现设置巡逻路线需求。

##### 8.1.3　报警情况设置

本系统要求机器人能对需要报警/不需要报警的情况进行设置。在报警情况设置界面，用户可以对已有的几种报警情况（火情报警等）进行设置，勾选表示需要报警，否则不报警，设置完的数据会保存到报警模块，在机器人发现对应情况并调用报警模块时，会根据报警情况设置选择是否报警，从而实现报警情况设置需求。

##### 8.1.4　自动避障

本系统要求机器人能在遇到障碍时自动进行路径规划，避开障碍。机器人遇到障碍时，会自动调用避障模块，根据保存的地图信息、扫描得到的障碍物信息和自身的体积形状信息，通过Navigation导航技术计算出一条避开障碍物并回到巡逻路线的路径，操作机器人回到正常路线，实现自动避障。

##### 8.1.5　报警控制

本系统要求机器人能发现紧急情况并报警。启智ROS机器人安装了立体相机，可以对目标物进行准确的定位，并通过物品识别技术判断其是否符合报警情况；如果是，机器人会调用报警模块，生成报警信息并显示在用户界面上，从而实现报警控制。

##### 8.1.6　手动操控

本系统要求机器人能通过交互界面让用户手动操作机器人移动。用户可以通过鼠标、键盘、手柄等UI设备和机器人交互，这些指令会通过用户交互部分和控制中心传递给机器人，机器人通过运动底盘进行移动，实现手动操纵需求。

### 8.2　非功能需求追踪

##### 8.2.1　性能属性

1. 响应时间

机器人收到各种指令后，会通过控制中心传达给相应模块；对于一些相对复杂的模块（如自动避障），我们会通过优化相应算法的方式提高响应速度，满足性能需求。

1. 功耗

本系统各模块分工明确，结构良好，控制中心在长时间未操作时也会操控机器人进入待机状态，功耗需求可以满足。

1. 处理能力

机器人对判断、躲避障碍并重新规划路线的功能已经在各模块中实现。

##### 8.2.2　质量属性

1. 系统可用性

系统的各个操作界面清晰简洁，在地图建模、报警等情况下也会在界面显示出文本信息，系统可用性良好。

1. 可移植性

本系统用到的硬件接口和内部软件接口在大部分ROS系统中都能实现，可移植性良好。

1. 健壮性

在对系统输入指令时，控制中心会进行识别，若识别为未定义指令，会在界面显示提示信息；各模块在设计时都考虑了错误处理程序，在识别到故障发生时，会停止当前的操作并在用户界面显示对应错误信息。由此可见系统健壮性良好。

1. 可靠性

系统整体设计并无大碍，故障发生概率小；系统发生故障时，相应模块会停止操作并报错，并不会对自身和周围环境造成破坏。由此可见系统可靠性良好。

### 8.3 需求追踪表

表8.1 需求追踪表

|  |  |
| --- | --- |
| SRS文档需求 | SDD文档对应部分 |
| 地图建模 | 5.1.2地图建模界面5.3.4地图部分6.2建图模块 |
| 巡逻路线设置 | 5.1.3巡逻路线设置界面5.3.5路径部分 |
| 报警控制 | 5.1.4报警情况设置界面6.7报警模块 |
| 自动避障、自主导航 | 6.4自主导航模块6.5避障模块 |
| 路径规划 | 5.3.5路径部分6.3路径规划模块 |
| 手动操作机器人移动 | 5.1.5手动控制界面5.3.1用户交互部分5.3.6机器人运动部分 |
| 传感器环境监测信息、图像处理数据 | 3数据库设计5.2硬件接口5.3.7图像识别部分 |
| 系统功能的完整性，系统对相应指令的处理能力，系统可移植性，系统可扩展性 | 4 体系结构设计 |