**【项目名称】**

**软件设计说明书**

**【文档标识（唯一标识该文档的标识号，SDD+202）】**

**【1.2】**

分工说明

|  |  |  |
| --- | --- | --- |
| 小组名称 |  | |
| 学号 | 姓名 | 本文档中主要承担的工作内容 |
| 16061176 | 陈强 | 第一、二、六章编辑 |
| 16061163 | 刘华兵 | 第三、四、六、七、八章编辑 |
| 16061169 | 胡峰 | 审核 |
|  |  |  |

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 | 2019.4.19 | 陈强、刘华兵 | 陈强、刘华兵、胡峰 | 初步设计分析 |
| 1.1 | 2019.4.20 | 陈强、刘华兵 | 陈强、刘华兵、胡峰 | 详细分析 |
| 1.2 | 2019.4.21 | 陈强、刘华兵 | 陈强、刘华兵、胡峰 | 最终版本 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 范围

## 项目概述

### 系统背景

嵌入式系统出现于20世纪60年代，40多年来随着计算机技术、电子信息技术的发展，嵌入式系统的各项技术蓬勃发展，市场迅猛扩大，已深入生产和生活的各个角落。在新的互联网+的大趋势下，生活中越来越多的设备中植入了嵌入式系统，极大提高了设备的智能性，为人们的生活带来了便利。

### 主要功能

本项目将基于ROS平台，实现一个基本避障运动型机器人。此机器人在陌生的环境下自动避障，和用户进行交互，控制机器人的运动。

### 非功能性需求

在基本功能之外，我们将尽力对机器人的易用性和性能进行完善。易用性包括人机交互的友好化，考虑在设定运动方向和距离的便捷性。性能将着重提高机器人的避障能力，尽可能小的减少障碍物对既定路线的干扰，最快到达目的地。还需要软件具有良好的可扩展性，在软件需求发生变化时具有可调整性。

### 应用场景

本项目实现的避障运动功能是其他复杂功能的基础，预计在生活服务和工业上可有应用。比如生活中扫地机器人，可以避让障碍物，确保扫地进程的进行。工业上仓库中货物的运输可以利用机器人减轻人力负担。

## 文档概述

本文档为项目的软件设计文档，内容为对软件的需求进行概述，提出软件的总体体系设计，设计软件之间的接口通信，并对每个软件模块进行详细的描述，并对其进行可追踪性说明。

## 术语和缩略词

ROS(Robot Operating System)：机器人软件平台

Ubuntu：开源的Linux操作系统

SLAM(Simultaneous Localization And Mapping)：即时定位与地图构建

RoboWare Studio：基于VScode开发的ROS专用IDE

Rviz：ROS官方提供的3D可视化工具。

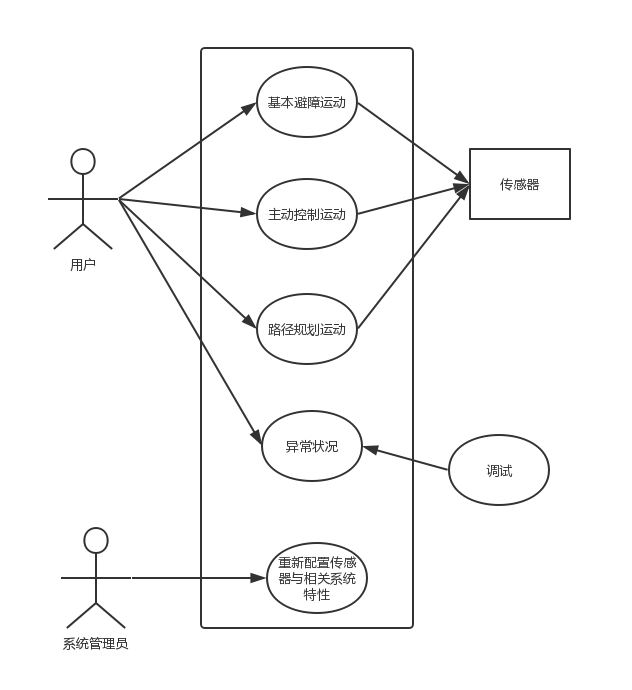
Gazebo: 机器人仿真软件，可以建立测试机器人的仿真场景。

## 引用文档

《启智ROS机器人开发手册v1.1.0》

# 需求概述

## 基本避障机器人用例图



图表 1 基本避障机器人用例图

## 基本避障机器人用例说明

### 机器人基本避障运动

**主要参与者：**用户

**目标：**机器人按照系统预设条件在环境中自动避障运动

**前置条件：进入**选择机器人模式，机器人系统完善，运动装置运行良好，摄像头等传感器运行良好

**启动：**用户选择基本避障运动模式

**场景：**

1. **系统管理员启动机器人，设置好初始参数。**
2. **用户选择避障运动模式。**
3. **检测机器人状态良好，机器人开始运动。**
4. **机器人遇到障碍，调整运动方向，避开障碍物继续行走。**
5. **用户选择停止机器人运动，机器人停止避障运动模式。**
6. **用户选择退出该模式，返回选择模式场景。**

**优先级：高**

**何时使用：第一个增量**

**使用频率：高**

**次要参与者：机器人**

### 主动控制机器人运动

**主要参与者：**用户

**目标：**用户使用键盘、遥控器等设备操控机器人在环境中运动，建立地图数据

**前置条件：进入**选择机器人模式，机器人系统完善，运动装置运行良好

**启动：**用户选择主动操控机器人模式

**场景：**

1. **系统管理员启动机器人，设置好初始参数。**
2. **用户选择主动控制机器人运动模式。**
3. **用户控制机器人运动，传感器获取信息，建立静态地图数据。**
4. **用户选择退出该模式，返回选择模式场景。**

****异常情况：****

1. **用户控制机器人操纵的方向不可行。出现在3，机器人发出报警信息。**

**优先级：高**

**何时使用：第一个增量**

**使用频率：高**

**次要参与者：机器人**

### 路径规划运动

**主要参与者：用户**

**目标：**用户给定机器人目标地点，机器人寻找路径到达目标地点

**前置条件：进入**选择机器人模式**，**机器人系统完善，运动装置运行良好

**启动：用户选择路径规划模式。**

**场景：**

1. 系统管理员启动机器人，**设置好初始参数。**
2. 用户选择路径规划运动模式。
3. 用户选定机器人运动目标地点。
4. 机器人根据传感器信息，建立地图数据。
5. 机器人按照可行的路径运动，动态刷新地图数据。
6. 机器人到达用户指定的地点。通知用户机器人到达目的地。

**异常情况：**

1. 用户给定地点不可达。出现在3，对用户返回目的地不可达信息。
2. 机器人找不到可到达的路径，即地点不可达。出现在6，对用户返回目的地路径堵塞信息。

**优先级：**中

**何时使用：**第二个增量

**使用频率：**中等

**次要参与者：机器人**

### 异常状况

**主要参与者**:用户

**目标**：如果机器人发生异常状况，通知系统管理员进行维护。

**前置条件**：机器人发生故障，使用中出现异常。

**启动**：机器人出现异常行为。

**场景**：

1：机器人发生异常行为，通知系统管理员。

**优先级**：低

**何时使用**：第三个增量

**使用频率**：低

**次要参与者**：机器人、系统管理员

### 重新配置传感器与相关系统特性

**主要参与者：**系统管理员

**目标：**如果机器人出现异常行为和信息，用户使用过程中出现问题，对机器人进行调整，维护。

**前置条件**：机器人出现异常信息。

**启动：**系统管理员维护机器人。

**场景：**

1. 系统管理员收集分析异常信息，判断错误所在地。
2. 复现错误情景，收集错误信息。
3. 修复系统错误，测试是否修复。

**优先级：**低

**何时使用：**第三个增量

**使用频率：**低

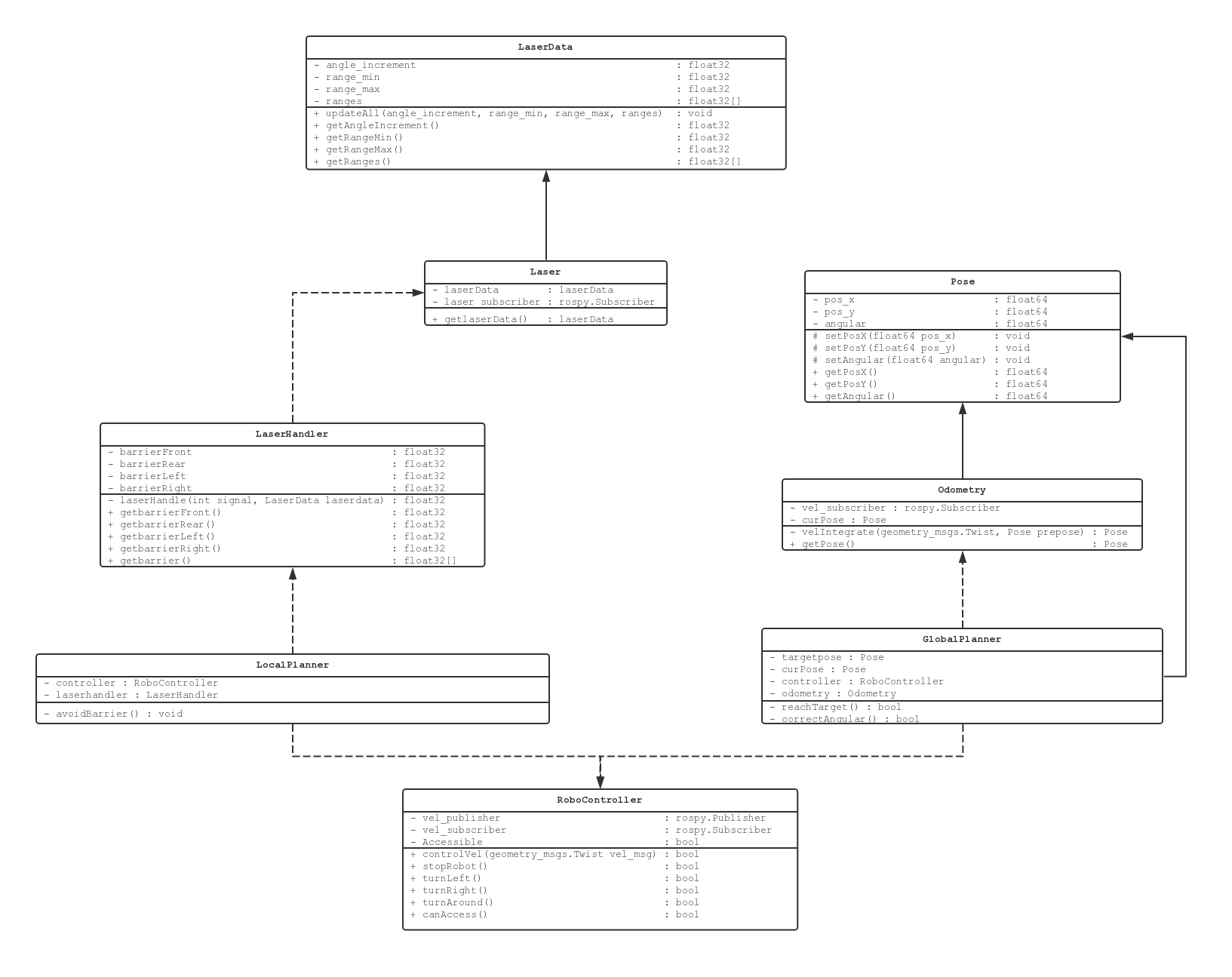
**次要参与者：**机器人、用户。

# 体系结构设计

## 总体结构

### 软件体系结构

基本避障机器人的实现基于ROS机器人操作系统，通过接收与处理激光雷达的数据感知周围障碍物信息，遇到障碍则作出反应调整运动速度与运动方向避让障碍。与此同时，后台的里程计不断接收速度信息进行积分运算，得到机器人的位置和方向（相对于运动起点）。软件体系结构基于面向对象的设计方法，功能实现所需类的关系如图表 1 基本避障机器人UML类图所示。



图表 2 基本避障机器人UML类图

1. LaserData类

该类封装了来自激光雷达的数据：

* 1. angle\_increment激光雷达每次扫描增加的角度
  2. range\_min距离障碍物最小距离
  3. range\_max距离障碍物最大距离
  4. ranges激光雷达原始数据，数组类型，长度为360，表示机器人周围360方向上距离障碍物的距离，最大为inf表示激光雷达扫描范围内无障碍物

可以通过update()方法进行数据的传入与更新，通过get()方法获取封装的数据。

1. Laser类

该类通过订阅Subscribe激光雷达发布的topic实时地获取激光雷达扫描到的数据并封装成LaserData。可以通过getlaserData()方法获取激光雷达数据。

1. LaserHandler类

该类对激光雷达的数据进行一般的处理，提取出基本避障机器人周围四个方向上的障碍物信息：前进方向、后方、左侧、右侧。可以通过get()方法获取处理结果。

1. LocalPlanner类

该类实现了基本避障机器人的避障功能。通过调用LaserHandler类获取机器人四个方向上的障碍物信息后，通过avoidBarrier()方法计算出躲避障碍物所需要运动速度与运动方向，并将这些信息传给RoboController控制机器人的运动。

1. RoboController类

该类实现了基本避障机器人的运动控制功能。通过向/cml\_vel这一topic发布速度更新消息控制机器人的运动速度与运动方向。该类提供了controlVel()方法用于直接控制机器人的运动状态，也封装了stopRobot()方法控制机器人停止、turnLeft()和turnRight()方法控制机器人左转和右转、turnAround()方法控制机器人掉头。同时为了实现LocalPlanner与GlobalPlanner的优先级控制设置了accessible控制变量，当LocalPlanner进行避障时将accessible设置为false防止GlobalPlanner的干预。

1. Pose类

该类封装了机器人的位置和方向信息。机器人运动过程中为了确定自身位置设置的坐标系定义为“以机器人初始速度方向为Y轴正方向，初始运动方向的右侧垂直方向为X轴正方向”，(pos\_X, pos\_Y)即为机器人在水平面上的坐标。angular为机器人运动方向的方向角，以逆时针方向为正方向，规定X轴正方向方向角为0。(pos\_X, pos\_Y, angular)就能确定机器人在水平面的位置和方向。可以通过set()和get()方法设置和获取机器人的位置和方向。

1. Odemetry类

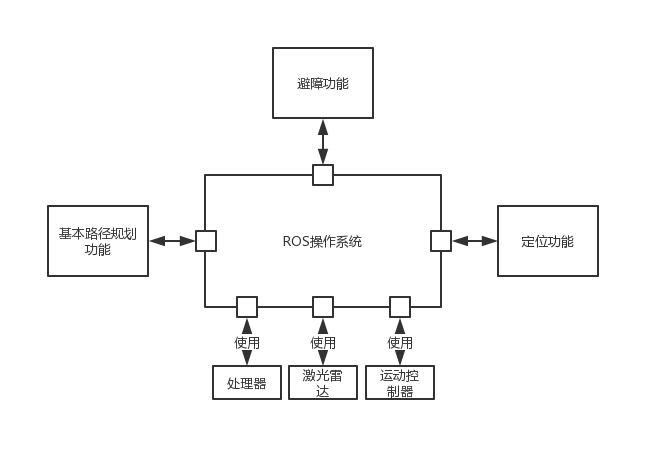
该类实现了里程计的功能。通过每个时间间隔t通过订阅/cmd\_vel这一topic获取机器人的运动速度和角速度与t积分计算出当前时间间隔t内运动过的距离和角度，并与旧值累加就能计算出机器人当前的位置和方向。velIntegrate()方法实现了这一功能。也提供了getPose()方法获取机器人当前位置和方向。

1. GlobalPlanner类

该类实现了简单的全局路径规划功能。由于机器人为了规避障碍会调整运动方向，correctAngular()方法通过检测初始运动方向上有无障碍物决定是否将机器人运动方向调整到初始运动方向，但需要同时满足无障碍物和RoboController未受LocalPlanner控制两个条件才能进行调整，这是为了防止避障运动过程被打断。如果机器人运动到了目标点平行位置(Y坐标相同)，则会调用reachTarget()方法实现向终点的运动。

### 硬件体系结构

基本避障机器人的实现依赖于ROS机器人操作系统提供的机器人实时状态数据，并通过ROS向机器人发送控制指令，而ROS操作系统实现了与硬件的通信与硬件控制。基本避障机器人系统的硬件体系结构如所示。



图表 3 基本避障机器人硬件体系架构

### 技术体系结构

基本避障机器人的实现基于ROS的Publish-Subscribe通讯架构，可以很方便的实现并发与多线程功能。开发实现过程中使用到的Topic有/scan、/cmd\_vel、/tf分别对应于激光雷达信息获取与处理、机器人运动控制和机器人定位功能。

## 关键问题及解决方案

### 机器人运动控制

ROS提供了完整的机器人运动控制功能。只需要向/cmd\_vel这一Topic发布运动控制信息。运动控制信息数据类型为geometry\_msgs.Twist，成员变量及其控制效果为：

1. geometry\_msgs.Twist.linear.x 控制机器人前后运动
2. geometry\_msgs.Twist.linear.y 控制机器人左右运动
3. geometry\_msgs.Twist.angular.z 控制机器人旋转

### 激光雷达数据的获取

使用ROS操作系统的Publish-Subscribe机制订阅/scan这一Topic的数据，就可以接收激光雷达的数据。ROS封装好的激光雷达数据类型为LaserScan，格式为：

std\_msgs/Header header

float32 angle\_min

float32 angle\_max

float32 angle\_increment

float32 time\_increment

float32 scan\_time

float32 range\_min

float32 range\_max

float32[] ranges

float32[] intensities

为了实现基本避障功能，需要用到angle\_increment, angle\_min, angle\_max, ranges[]这四个数据，封装成了LaserData类。

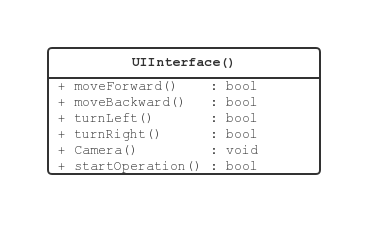
### 定位

为了实现完成避障运动到达终点的目标，需要实时确定机器人的位置。机器人使用虚拟里程计来确定运动过的距离从而确定相对于起点的位置。里程计作为独立于避障功能的后台进程，需要始终运行，在合适的时间T内通过对线速度与角速度进行积分确定运动过的距离。积分间隔T应该选取合适的值，若过大，则会导致精度不足，过小，又回导致频繁占用系统资源导致其他功能受到影响。

# 接口设计

## 用户界面接口

系统用户界面提供了机器人控制功能和显示摄像头采集到的视频数据功能，便于用户对机器人的运动控制，也需要提供开始按钮，启动机器人自主避障运动程序，接口如图表 4 基本避障机器人用户界面接口所示。



图表 4 基本避障机器人用户界面接口

# 数据库设计

基本避障机器人的功能部署与实现不需要数据库。

# 详细设计

## 障碍信息感知

障碍信息感知需要利用激光雷达获取的周围环境数据，并对数据进行处理，解析出机器人前行方向、左侧、右侧、后方与障碍物的距离。涉及到LaserData类、Laser类以及LaserHandler类。类图如图表 1 基本避障机器人UML类图所示，顺序图如图表 2 障碍物信息感知顺序图所示。

图片包含 屏幕截图

描述已自动生成

图表 5 障碍物信息感知顺序图

## 定位与确定方向

机器人的定位与确定方向是基于积分的策略实现。通过对恒定时间段T内的速度与角速度进行积分并累加即可获得当前的位置和方向。涉及到的类是Pose类、Odemetry类。类图如图表 1 基本避障机器人UML类图所示，顺序图图表 3 获取位置信息顺序图如所示。

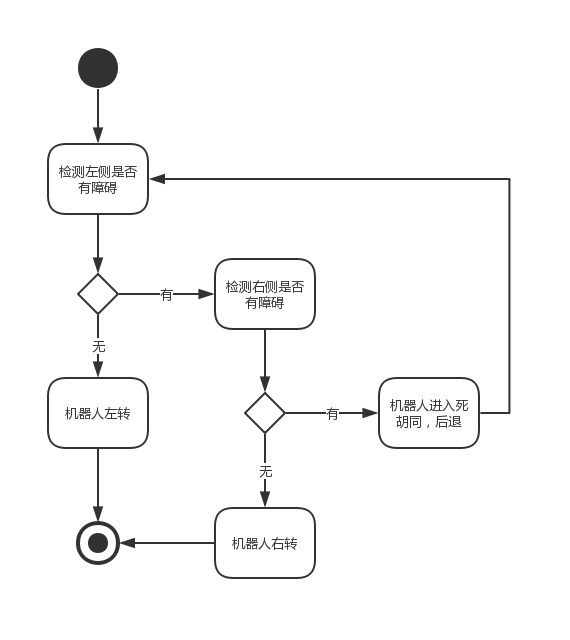
图片包含 屏幕截图

描述已自动生成

图表 6 获取位置信息顺序图

## 规避障碍

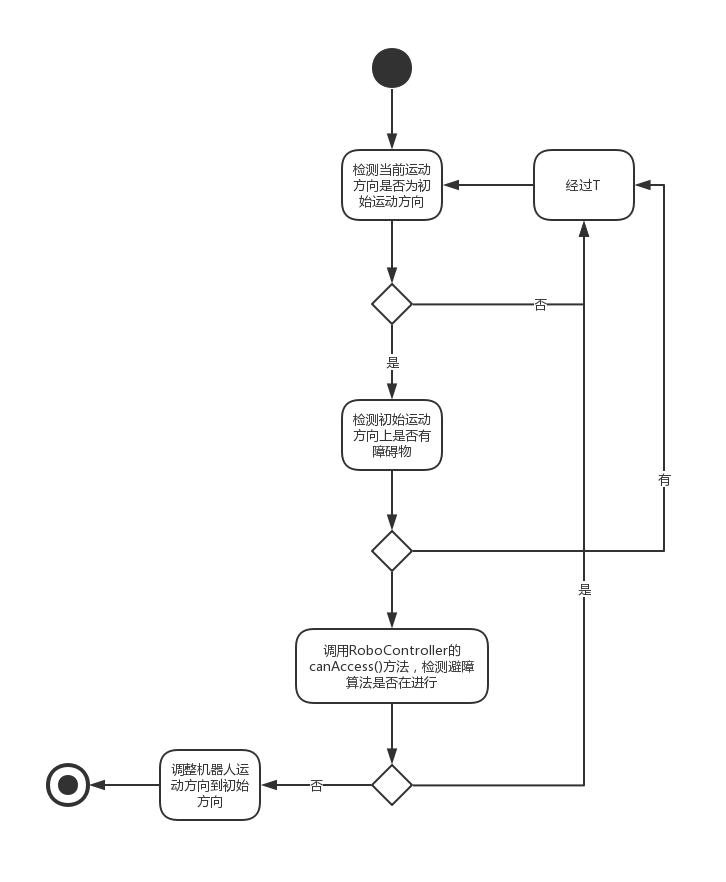
如果检测到机器人运动方向上有障碍物，则需要调用规避障碍的程序，为此可以设置一个阈值MinDist，如果与障碍物的距离小于这个阈值，则需要进行规避。规避障碍的实现涉及LocalPlanner类中的avoidBarrier()方法，算法活动图如图表 4 机器人避障算法活动图所示。



图表 7 机器人避障算法活动图

## 运动方向纠正与接近终点

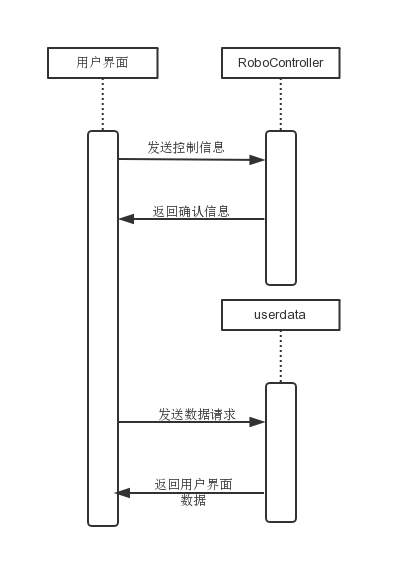
机器人在避障运动开始前，管理员可以设置终点的位置(0, tar\_Y)。为了使机器人接近终点，采取了贪心的策略，即机器人为了避障转弯后如果检测到初始运动方向上没有障碍物，则转弯到初始运动方向上。当机器人到达y=tar\_Y时（由于不可能准确到达y=tar\_Y，可以设置一个接近阈值TarBt），启动接近终点算法控制机器人到达终点位置。GlobalPlanner类的correctAngular()和reachTarget()方法实现了以上两个功能。correctAngular()算法活动图如图表 5 返回初始运动方向算法活动图所示。

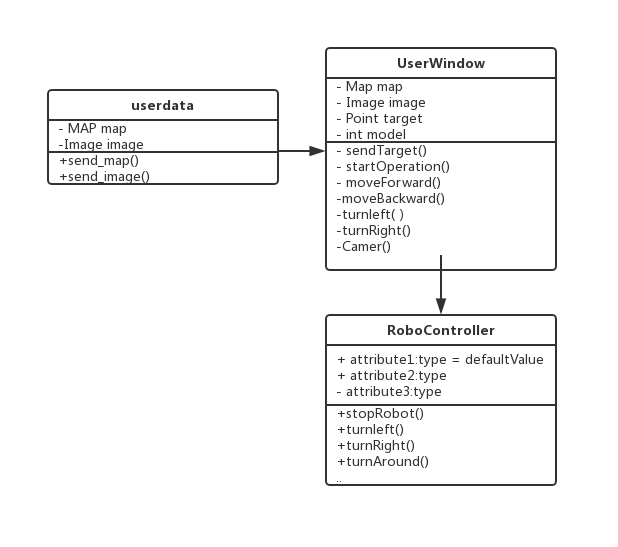


图表 8 返回初始运动方向算法活动图

## 用户输入与控制

主要类为用户窗口类，用户通过窗口发送控制命令，改变机器人的运动模式，机器人控制类根据用户的控制信息改变自身运动行为。用户数据类根据用户窗口所需的信息发送用户数据给窗口。





# 运行与开发环境

## 运行环境

### 硬件环境

ROS机器人，配置有360度激光雷达，独立运动控制，支持多线程的CPU。

### 软件环境

1. Kinetic ROS操作系统
2. Ubuntu 16.04 Desktop 操作系统
3. Python 2.7

## 软件环境

1. RoboWare Studio 机器人开发环境
2. Gazebo 机器人仿真工具
3. Rviz 可视化工具
4. Rqt 数据显示工具

# 需求可追踪性说明

1. 机器人直线行驶、机器人转弯、机器人减速停止

需求分析中关于机器人基本运动功能的功能性需求通过3.2.1机器人运动控制阐述了实现所需要的数据结构，通过图表 2 基本避障机器人UML类图阐述了在总体设计中的位置，通过RoboController类阐述了具体实现。

1. 检测周围障碍

需求分析中关于机器人障碍检测的功能性需求通过3.2.2激光雷达数据的获取参数了实现需求所需要的具体数据结构，通过图表 2 基本避障机器人UML类图参数了所需要的类以及在总体设计中的位置，通过LaserData类、Laser类、LaserHandler类参数了具体的实现。

1. 转弯规避周围障碍

需求分析中关于机器人障碍规避的功能性需求通过LocalPlanner类阐述了具体的实现，通过6.3规避障碍阐述了规避算法的流程。

1. 返回初始方向、返回终点

需求分析中关于机器人基本路径规划的功能性需求通过GlobalPlanner类阐述了具体实现，通过6.4运动方向纠正与接近终点阐述了基本规避算法的流程。

1. 确定方向距离

需求分析中关于机器人定位功能的功能性需求通过Odemetry类阐述了具体实现，通过6.2定位与确定方向阐述了功能的实现顺序流程。