**启智ROS机器人开发计划**

**软件设计说明书**

**SDD203**

**V1.0.0**

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 |
|  | 王贺正 | 软件体系结构，硬件体系结构 |
|  | 杨光 | 技术体系结构，支撑体系结构 |
|  | 高则晗 | 接口设计，数据库设计 |
|  | 阳韵非 | 详细设计，关键设计及解决方案 |
|  | 单勇 | 运行与开发环境，需求可追踪性说明，范围，需求概述 |

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0.0 | 4.21 | 王贺正，杨光，高则晗，阳韵非，单勇 | 王贺正 | 初次提交 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 范围

## 项目概述

本项目基于启智ROS机器人，开发可操纵机器人行动的嵌入式系统。主要功能有：提供可手动操作机器人行动的方法，实现机器人读取地图后的路径规划功能，实现机器人自动扫描周边环境生成地图并寻路，实现机器人使用其机械臂对物品进行夹取。同时要保证机器人在恶劣环境中的可用性，用户使用过程中的安全性，简化使用界面保证产品的易用性，以及控制机器人的电量消耗，保证机器人对于未来更多功能的可拓展性。

用户可以使用开发完成后的产品的机械臂夹取功能完成车间生产线上固定物品的取用，也可结合使用多种功能使产品成为智能家居的一部分，帮助用户完成各项家务活动。

## 文档概述

本文档用于介绍机油教驻泰伦研究队所进行的启智ROS机器人开发计划的软件设计，包括体系结构设计、接口设计、数据库设计、详细设计、运行与开发环境和需求可追踪性说明。

## 术语和缩略词

|  |  |  |
| --- | --- | --- |
| 编号 | 术语或缩略词 | 解释 |
| 1 | USB | Universal Serial Bus, 通用串行总线 |
| 2 | CPP | C++语言 |
| 3 | ROS | Robot Operating System机器人操作系统 |
| 4 | GUI | Graphic User Interface图形用户界面 |
| 5 | IDE | Integrated Development Environment集成开发环境 |
| 6 | Wi-Fi | 一个创建于IEEE 802.11标准的无线局域网技术 |
| 7 | git | 一个开源分布式版本控制系统 |

## 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 | 启智ROS机器人开发计划 | V1.1.1 | 2019.3.31 |
| 3 | 启智ROS机器人需求规格说明书 | V1.0.1 | 2019.4.1 |

# 需求概述

2.1系统用例模型及简要说明

机器人系统将采用活动图方式来展示具体操作流程。

原因：对于机器人的驾驶操作以及机械抓取操作步骤非常明确，可拆分。且在进行一项操作时着重使用机器人的某几个模块，在操作完成之前极少调用其他模块，因此不需要在用例模型图中体现出扩展性。

2.2对系统用户的描述

使用者：机器人操作员，机器人开发者，机器人附带传感器。

使用功能描述：

功能1：机器人操作员可以通过手柄直接控制机器人。

功能2：机器人开发者可以编写软件使机器人自动按照代码运行。

功能3：机器人操作员打开机器人附带传感器，使得雷达传感器扫描整个运行空间，生成三维地图，机器人可按照地图进行自动寻路方式运行。

功能4：机器人操作员可按照机器人生成的三维地图，操纵机器人附带的机械臂进行抓取操作。

2.3功能1用例概述

主要参与者：机器人操作员。

目标：通过外部设备，直接操纵机器人的行走模块使得机器人改变所处位置。

前置条件：机器人所附带计算机中已安装ROS等相应操作软件。

启动需求：机器人操作员试图移动机器人的当前所处位置。

场景流程：

1.机器人操作员将控制计算机安装到机器人上，连接USB口并连接操控手柄。

2.打开机器人底盘面板的USB电源开关。

3.启动Ubuntu终端，并输入roslaunch wpb\_home\_bringup js\_ctrl.launch指令。

4.待终端上显示出速度信息显示行时，可以拨动手柄摇杆。

5.打开机器人底盘面板上的红色急停开关，此时可以正常移动。

2.4功能2用例概述

主要参与者：机器人开发者。

目标：通过直接编写程序，使得机器人不用外部操控也能自主行走。

前置条件：同功能1。

启动需求：机器人开发者试图调试机器人运行代码。

场景流程：

1.机器人开发者打开RoboWare Studio，新建ROS包，输入名字与依赖项。

2.机器人开发者于src目录中新建CPP源文件，选择“加入到新的可执行文件”选项。

3.机器人开发者编写代码并编译。

4.启动机器人核心节点，确认打开电源，输入rosrun指令，命令机器人运动。

2.5功能3用例概述

主要参与者：机器人开发者，机器人操作员，机器人自带传感器。

目标：通过雷达传感器对周围空间的扫描，获取空间地图信息并保存，在地图的基础上进行导航移动。

前置条件：同功能1，并且完成功能1。

启动需求：机器人操作员试图让机器人能够在确定空间中自主寻路移动。

场景流程：

1.启动机器人以及激光雷达，并启动Rviz平台。

2.操作机器人在确定空间场地内移动，保存地图。

3.在可视化地图文件中，规划机器人的行走路径。

4.机器人自动在路径中行走。

2.6功能4用例概述

主要参与者：机器人操作员，机器人自带传感器。

目标：通过机器人自带传感器对于桌面物体的识别后，操纵机器人抓取物品。

前置条件：同功能1，并且对于机器人机械臂调整完成。

启动需求：机器人操作员试图让机器人识别并抓取桌面上的某个特定物件。

场景流程：

1.启动机器人头部的摄像头，检测三维空间的物体。

2.收到三维图像后，机器人操作员操控机械臂，进行抓取操作。

# 体系结构设计

## 总体结构

**3.1.1总体结构**

启智ROS机器人

命令输入

命令处理

抓取物体

报错

命令确认

命令读取

切换状态

移动

移动控制

地图建模

位置控制

机械臂控制

物体识别

激光雷达控制

自动导航

手动导航

SLAM建模

说明：

命令输入部分负责处理用户输入的命令，并将相关信息传递给命令处理部分。此部分需要根据用户输入确定是否切换当前需要执行的功能，如何切换当前需要执行的功能，确认该指令是否符合当前执行功能的需求并对不符合需求的指令进行报错。

命令处理部分负责控制机器人的行为，并将其功能分解为两大部分：移动功能与抓取物体功能。抓取物体功能需要实现物体识别功能，控制机械臂移动，并实现底盘的微移。移动功能需要根据上级传递的信息确定执行哪种功能：手动移动，内置程序移动或是自动寻路移动；同时其需要实现对地图的建模以及机器人位移的控制。

**3.1.2 程序类图**

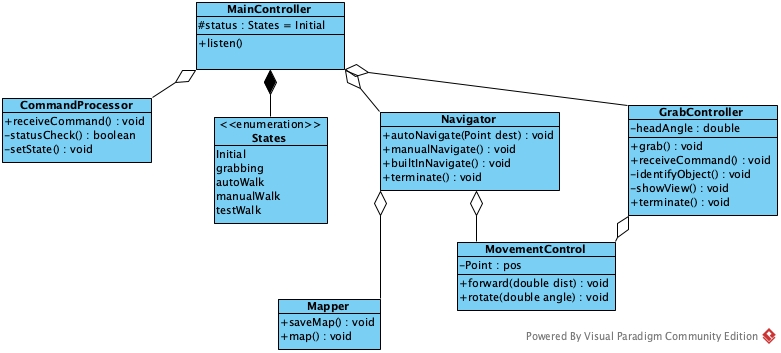


图 1 软件类图

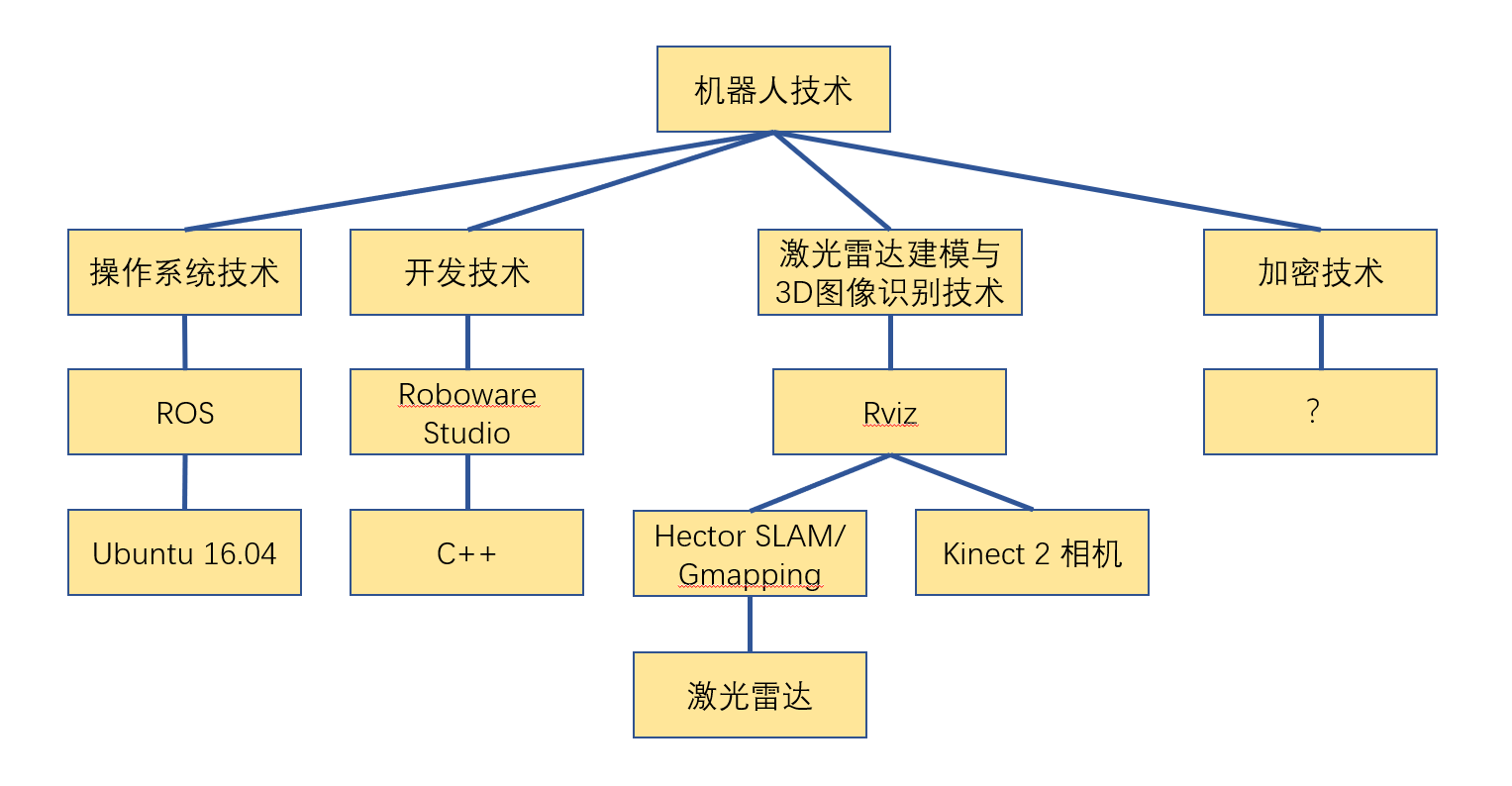
程序的类图如上图所示。最高层的主控制器类负责监听一切输入信号，根据指令处理器的反馈信号调用相关模块中的相关方法实现功能。指令处理器负责接收输入并判断系统应当执行哪种功能，或是退出当前功能，或是对当前功能进行手动操作。主控制器类将移动和抓取功能分别交给两个类。抓取控制器类负责抓取功能并控制机械臂移动，导航器类则负责三个移动功能并控制导航功能。这两个类都分别控制一个移动控制器对象，以实现机器人的移动功能。同时导航器下属一个绘图器对象以实现地图的绘制，供导航器计算路径使用。

**3.1.3 硬件体系结构**

|  |  |
| --- | --- |
| 服务层 | 手动移动，内置程序移动，自动寻路移动，抓取物体 |
| 数据层 | 手柄控制信号，激光雷达扫描信号，地图信息，摄像头视频信息，物体检测结果 |
| 硬件层 | 激光雷达，机械臂，底盘，Kinect2摄像头 |

本工程的硬件系统分为三层。最高层服务层为需求所提出的四项功能，供用户选择并实现对应功能，包括手动移动，内置程序移动，自动寻路移动，抓取物体四项功能。第二层为数据层，负责向上一层提供实现功能所必须的数据。其中激光雷达扫描信号和地图信息为自动寻路功能的基础，手柄控制信息为手动移动功能的基础，摄像头视频信息、手柄控制信号和物体检测结果为抓取物体的基础。最底层为机器人的各个部件，负责提供数据层所需的信息。

**3.1.3技术体系结构**



与机器人直接操控相关的技术可以分为操作系统技术，开发技术，激光雷达建模与3D图像识别技术，加密技术。

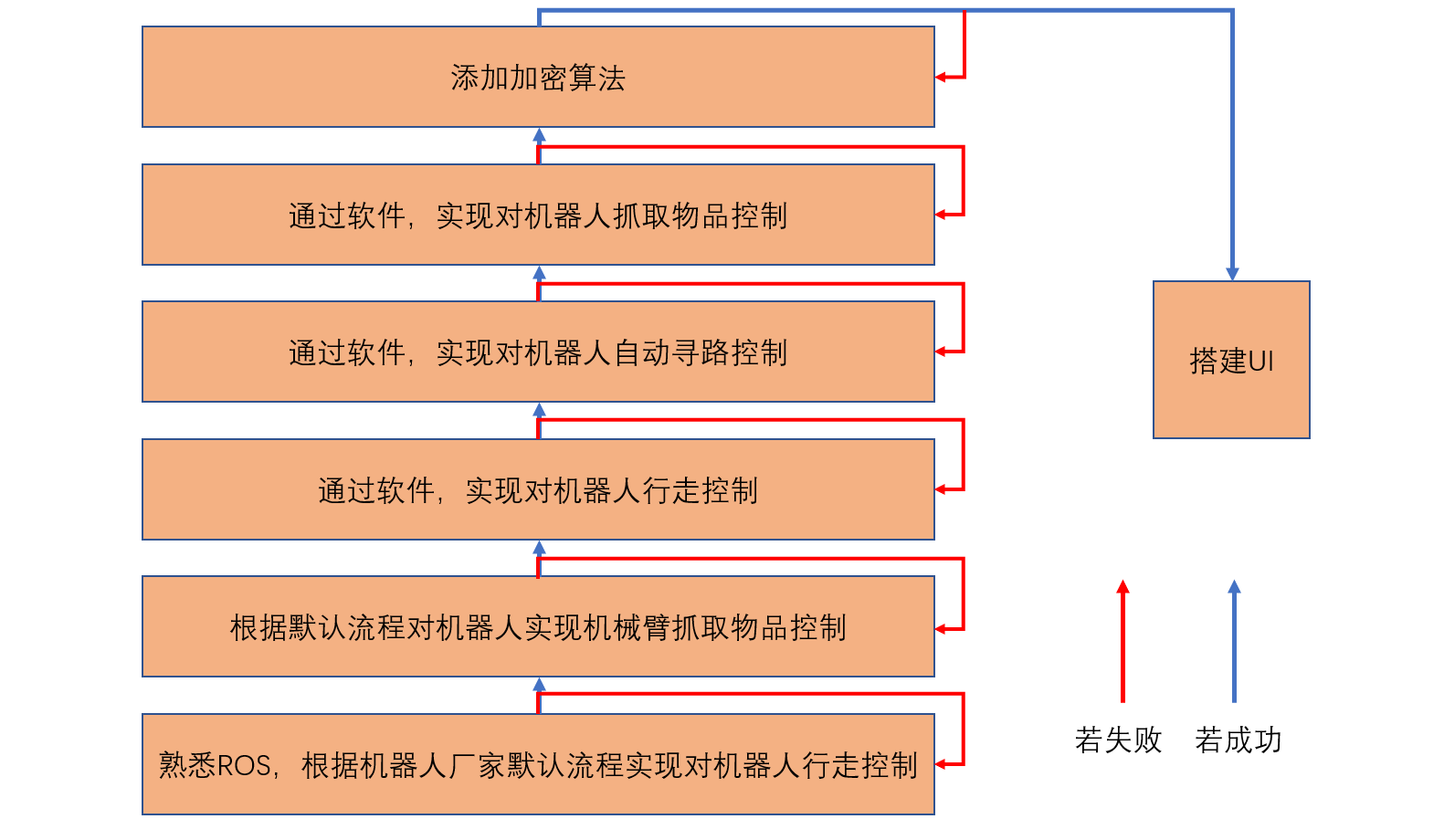
操作系统技术的最底层是Ubuntu 16.04 Linux系统，在这之上是相应Kinetic版本的ROS。ROS以Ubuntu系统作为基础而开发。

C++系统支撑Roboware Studio开发IDE，是机器人开发的核心编程语言。

激光雷达和Kinect2相机是雷达建模和3D图像识别的底层硬件传感器，在这之上是机器人的自动寻路功能和物体识别抓取功能。Hector SLAM或者Gmapping算法管理激光雷达，使得雷达的数据能在Rviz平台上建模。同时，Rviz平台还能对Kinect2相机的拍摄数据进行处理，标识物体。

至于加密技术，考虑到机器人需要实时的控制流数据，而现阶段还没有对何种具体加密技术达成一致，所以暂时不确定。

**3.1.4支撑体系（部署体系）**



整个软件的部署体系从底层逐步往上，又因为机器人的当前功能实现重度依赖更下层功能的正确实现，所以部署体系采用层次化结构，若当前部署步骤未能达成，则只能重新进行该步骤，直到成功为止，然后才能进行下一步。

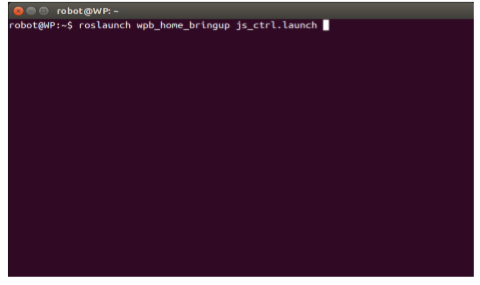
在未进行实际软件编写时，需要对机器人的行走功能以及抓取功能进行熟悉与测试，使用厂家建议的操纵模式和步骤。熟悉硬件结束后，由最简单的机器人控制开始，搭建软件框架，填写代码，测试代码运行正确性。软件控制机器人行走成功完成后，需要将机械臂抓取功能整合到所开发的软件中去。最后，在机器人的控制信号传输过程中添加加密算法，具体添加位置可能在软件客户端机器与机器人端平板电脑。若不需要加密算法，则这一步可以省略。在软件整体完成后，添加用户界面。若不需要用户界面，则这一步可以省略。

## 关键问题及解决方案

### 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. 从 Ubuntu 桌面左侧的启动栏里点击“Terminal”终端图标，启动终端程序（也可以通过 同时按下键盘组合键“Ctrl + Alt + T”来启动）。在终端程序中键入如下指令：

roslaunch wpb\_home\_bringup js\_ctrl.launch



* 1. 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。

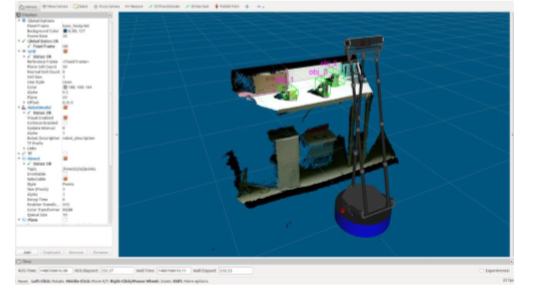
（1）手柄的左摇杆，控制机器人运动的平移向量，上下分别控制前后移动，左右分别控制侧向平移；

（2）手柄的右侧摇杆控制机器人运动的转动向量，拨向左是左转，拨向右是右转。

（3）机器人的运动平移向量和转动向量是可以叠加的，即可以一边平移一边旋转，这是全向底盘的优势。

### 桌面物体检测

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。



服务机器人常常遇到需要寻找某个物体的需求，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。

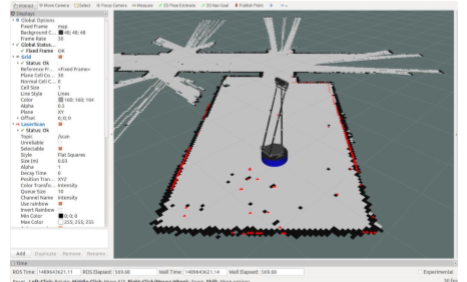
### 物品抓取

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。外部调用例程源码文件的位置为：

~/catkin\_ws/src/wpb\_home/wpb\_home\_tutorials/src/wpb\_home\_grab\_client.cpp

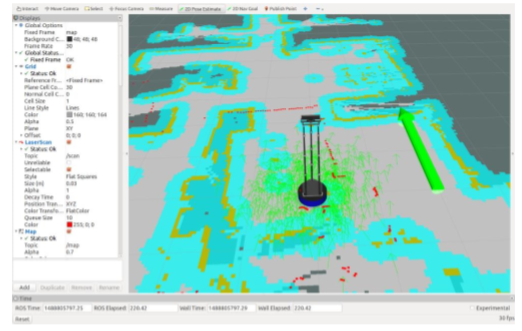
### 地图建立

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 支持多种 SLAM 算法，其中主流的是 Hector SLAM 和 Gmapping，通过激光雷达所采集到的数据进行地图的建立。

### 导航系统



amcl，英文全称“Adaptive Monte Carlo Localization”（蒙特卡洛自适应定位算法）。这是一种使用概率理论在已知地图中对机器人自定位置进行估计的方法。

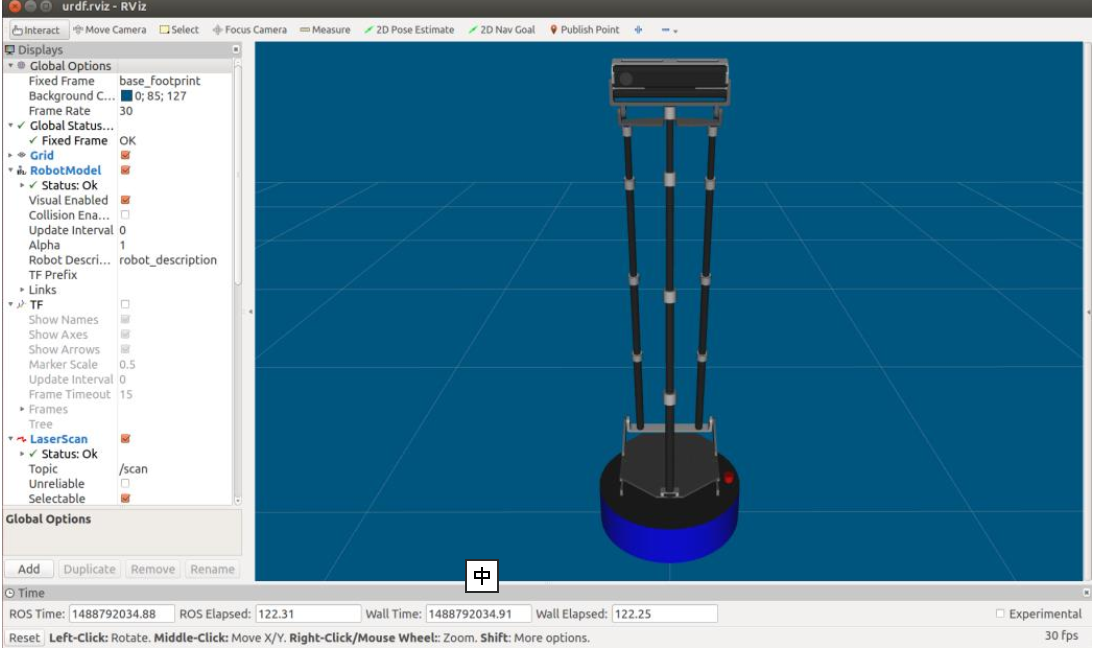
move\_base，这个包将机器人导航需要用到的地图、坐标、路径和行为规划器连接到了一起，同时还提供了导航参数的设置接口，是 ROS Navigation 的核心中枢。

# 接口设计

## 系统用户界面设计

本系统的用户界面分为在小型计算机屏幕上显示的供开发调试用的用户界面和供目标用户使用的遥控器、按键和指示界面。

系统用户界面是由ROS开发套件提供的Rviz，它可以图形化地显示ROS中各模块算法的运行信息，得到一个三维的模拟效果图。它还可以对地图进行查看、保存、加载等操作。



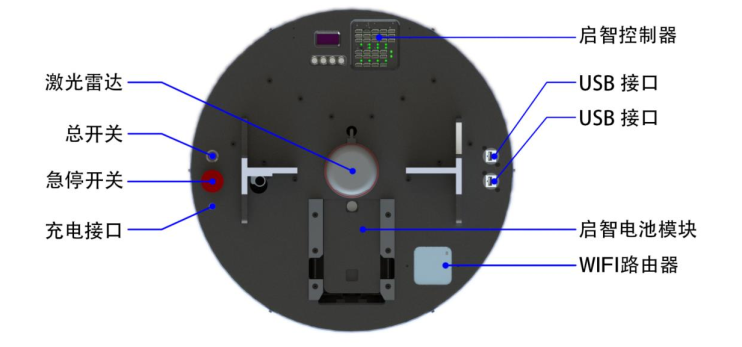
用户使用的系统界面主要是无线手柄和底盘控制器上的开关、LED、12864液晶屏等。

总开关控制整个系统的电源，急停开关会立即切断电机的电源。其他按键可以选择机器人的工作模式。

LED会指示当前机器人是否正在工作。

无线手柄会在机器人设置为人工操作时用于控制机器人运动。

液晶屏会输出系统的运行模式和该模式的运行状态。同时还会显示电池电压等基础状态。



## 外部接口设计

### 硬件接口

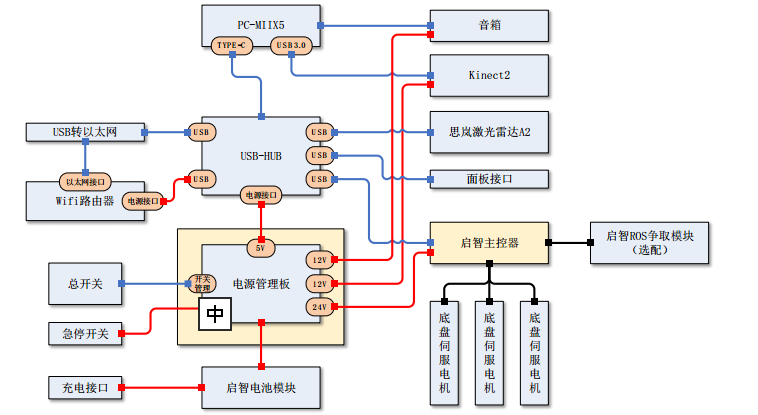
本系统的外部硬件接口主要是充电接口以及USB接口上的各外设。Wi-Fi链接可以通过局域网对机器人进行控制，USB接口上可以连接U盘对机器人提供软件更新和数据保存加载等功能。

### 软件接口

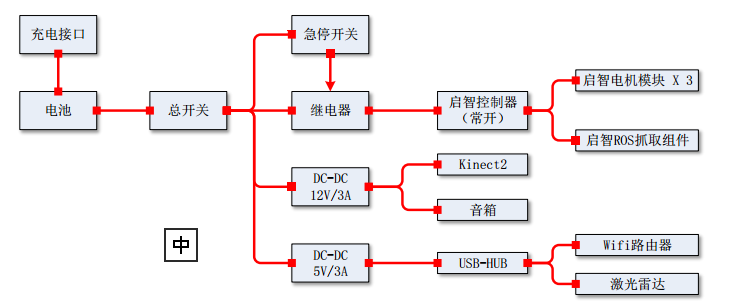
本系统的外部软件接口主要为通过Wi-Fi的控制接口。在主控计算机上，运行有一个SSH服务端，可以让开发者登录进行各种高权限操作，还可以在其中启动VNC服务端实现图形化地对系统进行调试。对于用户，考虑到安全因素不会开放外部的软件接口。

## 内部接口设计

### 硬件接口



系统内部的硬件主要分为三个大的部分：主控计算机、机器人主体、传感器和音箱等外设，各部分内部均由其制造商设计制造。主控计算机接入USBHUB并通过USBHUB和所有低速设备相连。主机和机器人主体之间的数据通过USB转UART的转换模块通信。主机和各传感器之间根据速度不同，与激光雷达、网卡、面板上的用户外设使用USB2.0连接，与Kinect2 RGB-D相机使用USB3.0相连，与音箱为模拟信号连接。



启智 ROS 机器人的电源由电池模块供给，该电源模块内部由 7 枚 3500mA/h 容量的锂离子电池串联组成，内置电池保护板。该模块输出电压与当前剩余电量有关， 剩余电量越少电压越低，正常工作输出电压范围 23.1V 至 29.4V。

底盘内部的电源控制板控制开关机、执行机构的急停，并为各控制器、传感器提供电源。

### 软件接口

本系统的软件全部基于元操作系统ROS，而ROS是相对独立地运行于Ubuntu操作系统之上。

ROS中主要的层次结构为节点、话题、消息和服务。节点类似于用户的线程。消息是实现线程间通信的机制，而话题是消息的一个属性，来实现消息的局部广播。服务则是固定的线程之间重载通信方式。各个用户线程之间，一般使用消息进行通信，与驱动程序之间使用服务方式。

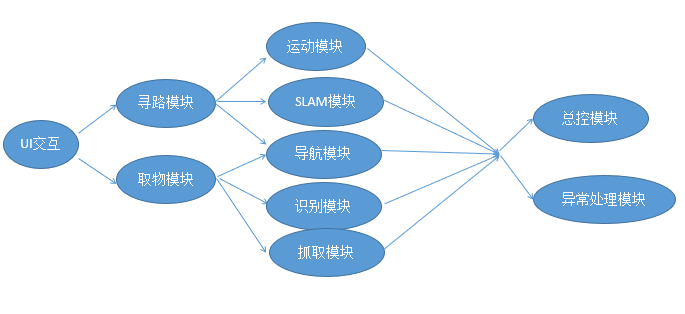
# 数据库设计

本系统的数据库采用简单的文件保存方式。当系统执行某些节点时，会创建一个以时间戳命名的日志文件，将所有程序输出顺序打印在文件中。这些日志只有开发者才有权限查看。

# 详细设计

我们的智能机器人主要有两个基本功能：寻路功能和抓取物品。要实现这两个主要功能，我们需要设计两个大的顶层模块：寻路模块和取物模块，从而实现对上述两个功能的集成。

这两个大的顶层模块由多个功能模块所构成， 系统的总体模块设计图如下图所示：



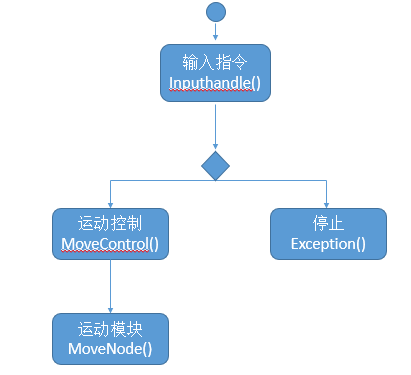
* 1. **运动模块**

**功能**：运动模块主要负责控制机器人实现基本的运动，如前进后退以及左右

转向。运动模块可以接收到激光雷达传来的数据，从而可以实时进行障碍物检测，一旦检测到障碍，便立即停止机器人的运动，并发出提示。除此之外，该模块还可以控制转向和移动的速度变化。

**输入：**从键盘读取输入(用于调试)，可以使用手柄操控输入指令。

**输出：**转化为控制底盘运动的指令，控制底盘



MoveMode类，该类可以设置机器人的移动速度以及转向速度属性。它包含控制机器人移动的move和stop方法，并可以根据MoveControl类的返回结果，实时监测障碍物，实现避障功能。InputHandle类的设置是为了处理机器人的简单运动，即用户自身通过手柄等输入运动指令。MoveController类是一个运动控制类，它则用来自主控制机器人的移动。

同时，我们还设计了一个Exception类，它包含了机器人在运行过程中可能会遇到的各项异常以及异常处理的方法。

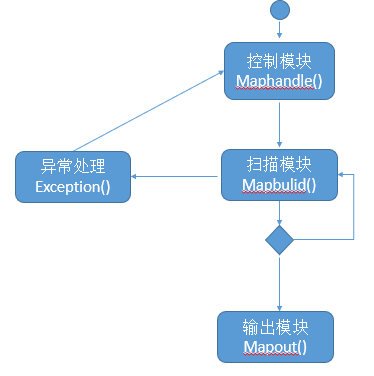
* 1. **建图模块**

**功能**：建图模块主要根据激光雷达的扫描结果，可以实时扫描机器人周围的

障碍物分布状况，借助HectorSLAM算法，创建环境地图。在每次将机器人应用到新场景中时，我们需要调用建图模块，然后推动机器人进行移动。当推动机器人绕场地一圈后，就可以将地图保存下来，便于之后自主导航和其它功能进行使用。

**输入**：激光雷达传递的数据

**输出**：一个构建好的场景地图



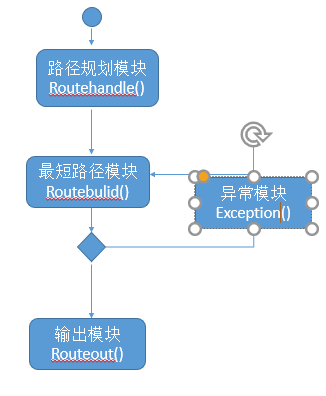
该模块的主类是Maphandle类，它保存了场景地图的大小以及详细信息等属性。它接收Mapbuild类的返回结果，使用merge方法进行地图的合成。Mapbuild类是与外部传感器的交互类，它接收激光雷达的扫描结果，并整合成地图信息。Mapout则用来控制整个建图过程，主要是实现HectorSLAM算法。同样在建图过程中如果出现异常则会调用异常处理类。

* 1. **路径规划模块**

**功能**： 我们整体的路径规划模块用来找出一条从起始点到目标点的最短距离。

**输入**：起始位置和目标位置

**输出**：一条路径(位置坐标序列)，指导机器人进行运动



该模块的主类是Routehandle类，用来实现路径规划。Routebuild类根据场景地图以及机器人此刻所处的位置，规划出遍历路径来，选择一条最优的遍历路径。Routeout类的将选择的路径输出。

同样的，如果在规划过程中出现异常，那么就会调用异常处理模块进行处理。

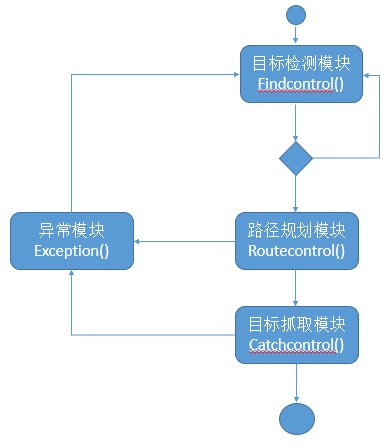
## 识别物体

**功能**： 该模块的主要功能就是识别场景地图中的特定物体，并给出其所在

位置。该模块可以调用已经训练好的分类器，然后识别双目相机所传回的画面，从而定位出目标物在场景中所处的坐标位置。

**输入**：要识别的物体(特定的物体)

**输出**：目标物所在的位置



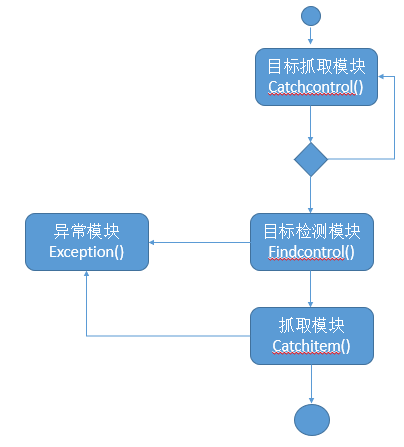
如果检测到目标物的话，就返回目标物所处的位置坐标，然后又进入路径规划模块进行路径规划后，到达制定地点完成抓取。在识别过程中如果出现异常，就会调用异常处理类，在抓取过程和规划过程出现异常则会重新进行目标检测。

* 1. **抓取模块**

**功能**： 该模块的主要功能是控制机械臂，对目标物进行抓取。

**输入**：物体的位置信息

**输出**：对机械臂的操控信息，进行抓取



首先需要调用目标检测模块对要进行抓取的目标物进行识别。如果没有检测到目标物的话，则会控制机器人进行移动，进而切换别的场景进行检测。如果还是没有目标物的话，则会进行错误处理，并且提示用户。检测到目标物后，就会调用抓取模块完成抓取动作。

* 1. **总控模块**

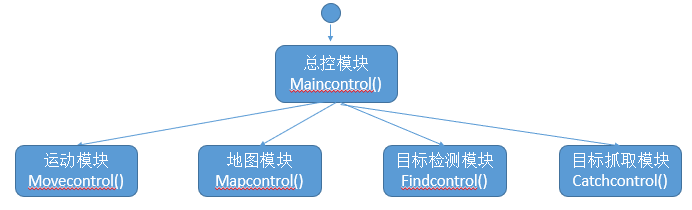
**功能**： 各模块的功能就是实现对各个子模块的统一控制，从而完成机器人

的各项功能。

**输入**：用户所发布的指令

**输出**：对各个子模块的控制信息

该模块的类结构图如下所示：



该模块主要有四个子类，分别实现了对于运动模块、建图模块、目标检测模块以及抓取模块的控制，从而完成与用户指派的各项功能。

# 运行与开发环境

## 运行环境

系统运行的硬件环境为有、单片机运动控制器外设、Kinect2视觉传感器、激光雷达、机械支架、控制臂等的树莓派。树莓派3B+是一台卡片式小型计算机，它有一个处理器系统芯片（SoC）BCM2837B0，内嵌四核运行于1.4GHz主频的Cortex-A53处理器内核，支持64位ARMv8指令集。它还有1GB低功耗低电压的LPDDR2 DRAM，并且支持以太网，Wi-Fi，USB，串口等对外通信。

系统运行的软件环境为Ubuntu 16.04 Linux系统以及其上的Kinetic版本ROS元操作系统。Ubuntu是常用的Linux发行版，它为该系统提供了一套稳定易开发的运行环境。Kinetic版本ROS系统提供一系列程序库和工具以帮助软件开发者创建机器人应用软件。它提供了硬件抽象、设备驱动、函数库、可视化工具Rviz、消息传递和软件包管理等诸多功能。ROS遵循BSD开源许可协议。

## 软件环境

系统开发所需的软件环境主要为（IDE）“RoboWare Studio”。

RoboWare Studio是ROS的一款IDE，基于微软开源的VSCode开发，它支持自动配置ROS编译选项，部署代码到下位机，git版本控制等功能，使得开发更加快速、简单，是目前最好用的ROS开发工具了。

核心编程语言使用C++。

# 需求可追踪性说明

功能需求：

功能1：机器人操作员可以通过手柄直接控制机器人。

启智ROS机器人

命令处理

移动控制

手动导航

功能2：机器人开发者可以编写软件使机器人自动按照代码运行。

启智ROS机器人

命令处理

移动控制

自动导航

功能3：机器人操作员打开机器人附带传感器，使得雷达传感器扫描整个运行空间，生成三维地图，机器人可按照地图进行自动寻路方式运行。

启智ROS机器人

命令处理

激光雷达控制

自动导航

生成路径

地图建模

功能4：机器人操作员可按照机器人生成的三维地图，操纵机器人附带的机械臂进行抓取操作。

机械臂控制

启智ROS机器人

物体识别

命令处理

移动

位置控制

抓取物体

非功能性需求：

可靠性：在机器人运行过程中，若出现异常行为，会在软件里记录异常行为发生时的代码运行位置以及次数，通过测试阶段不断多测的测试，使这个频率发生的概率将为最低。

安全性：机器人将拒绝除制定操作设备以外的任意设备给出的指令，采取的办法为设置登录账号，登录密码。

资源特性：机器人工作时的电压将会显示在控制面板上，可以随时查看工作电压，防止电压过高以及过低。

易分析性：程序会纪录机器人的每一次行为轨迹，若发生异常，可以及时读取行为轨迹，确定机器人在何时何处发生异常。

易改变性：开发人员可根据异常轨迹对相应代码进行修改。

易操作性：GUI界面上可进行机器人操作模式的选择，录入地图，机械臂启等操作均可通过按键完成。