**第二十五届中国机器人及人工智能大赛**

**智慧药房挑战赛项目**

**技 术 报 告**

学 校：重庆邮电大学

队伍名称：哆啦A梦的智慧车队

参赛队员：李赵正阳 周嘉靖 朱俊安

带队教师：石钧仁 张开碧

关于技术报告和研究论文使用授权的说明

本人完全了解第二十五届中国机器人及人工智能大赛有关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和赞助公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：

带队教师签名：

日 期：

**目 录**

[第1章 方案概述 1](#_Toc135212159)

[第2章 问题描述 2](#_Toc135212160)

[第3章 技术方案 2](#_Toc135212161)

[1.1 自主导航 2](#_Toc135212162)

[第4章 方案实现 3](#_Toc135212163)

[第5章 测试分析 3](#_Toc135212164)

[第6章 作品总结 3](#_Toc135212165)

[参考文献 3](#_Toc135212166)

1.药品配送小车的技术方案设计，对作品进行技术梳理，详细阐述如何实现小车的自主导航、以及等功能。技术方案的内容可包含方案总体控制思路、所需的技术及多种实现方法的对比、技术的可行性等。

2.详细的专业关键技术的实现思路。选手根据提供的相关技术资料，完成关键技术点的学习,写好代码，并编写

3.清晰描述单片机驱动方法、底盘控制模型和控制算法等。

4.详细分析计算机视觉的识别原理、具体方案以及代码实现。

5.详细论述小车的路径规划算法的方案和技术实现。

# 方案概述

## 技术路线

配送小车完成药品分拣系统和快递小哥之间的药品配送。根据规则，配送小车要识别取药小哥的所需药品的类别（A/B/C），到相应的配药窗口取药，并送至对应的取药窗口。本任务主要依靠Ubuntu系统（如图1所示）实现，需首先掌握Ubuntu基础知识。

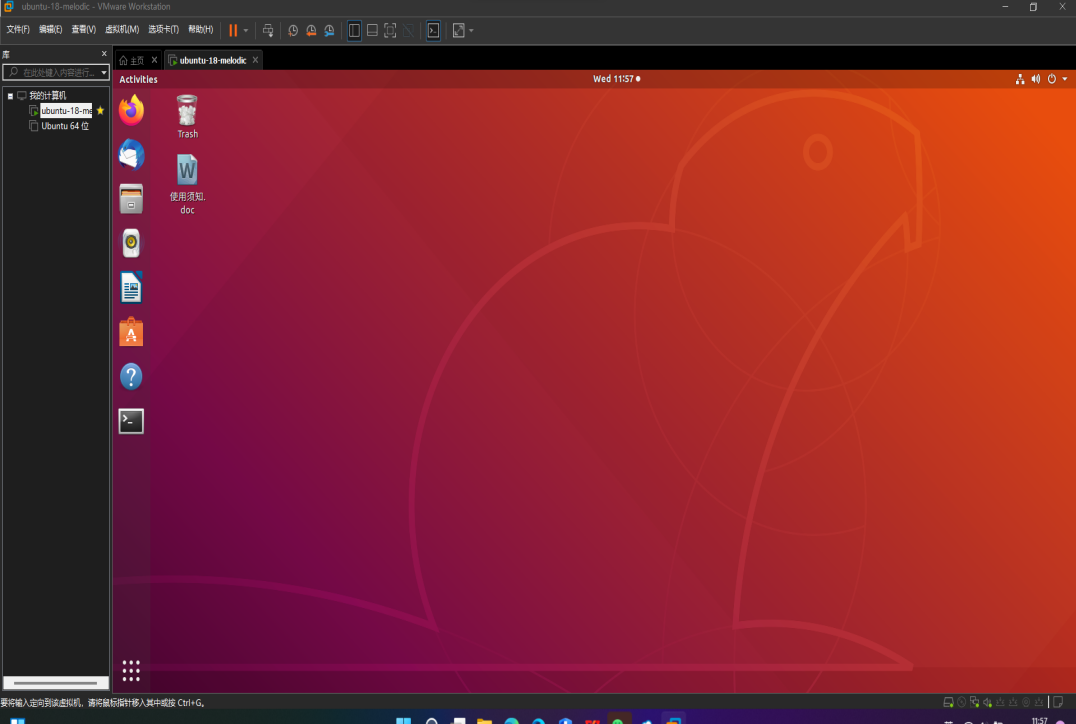


图1 Ubuntu系统

再按技术实现路线编写相关代码控制树莓派等器件，将树莓派与单片机等器件连接，进而对单片机和雷达等物件进行有效控制。要实现该任务，本组的技术实现路线分为如下几个方向：

1. 获取识别板的信息：使用EPRobot配备的深度相机，采集RGB 图像并通过图像处理或深度学习算法对快递小哥所需药品的类别以及数字手写体进行识别。可以采用目标检测或图像分类技术，将药品类别标识出来。
2. 路径规划与导航：基于ROS（Robot Operating System）框架，利用SLAM（Simultaneous Localization and Mapping）算法构建地图，并根据快递小哥所在位置和药品配送窗口的位置，规划最优路径，并通过导航算法实现小车的自主导航。
3. 动作执行与控制：通过控制小车的电机或舵机，实现小车的移动、停止、转向等动作，并根据药品的类别和目标配送窗口的位置，精确控制小车的移动和停靠。
4. 通信与协调：配送小车需要通过订阅发布话题方式获取取药窗口，配药区信息等。
5. 调度策略：根据规则，合理安排最优配送方案。

## 创新点及测试结果

1. **创新点：**

视觉识别技术的应用：通过图像处理和深度学习算法，实现对快递小哥所需药品的准确识别，提高配送系统的准确性和效率。

自主导航与路径规划：利用SLAM算法构建地图和路径规划算法，实现小车的自主导航，提高配送的灵活性和自动化程度。

1. **测试效果**

在测试阶段，本组模拟快递小哥和配送小车的实际场景，通过实地测试验证方案的可行性和稳定性。分别测试配送小车的视觉识别准确度、路径规划的精确性、动作执行的准确性以及与取药窗口的通信与协调效果。根据测试结果显示，所有功能均能如期完成。

相机或激光雷达等传感器，

使用 RGB 图像完 成图像处理的一系列功能

所需药物识别。使用视觉算法中的多模板匹配识别

配送方案规划，考虑使用贪心算法思想，最短时间得分最多的方案。

智能小车的自主定位与导航。智能小车需要使用激光雷达进行建图与定位。建图完成之后，智能小车根据雷达的定位，主函数的调度策略进行路径规划，完成识别图片的任务。在已经建图完成的前提下，使用蒙特卡洛定位辅助导航任务的进行。

# 问题描述

【填写说明：描述本方案的关键性问题，以及相关研究，分析并提出本作品所提供的算法实现或算法优化方案】

本方案关键问题主要涉及地图构建、定位、视觉识别、自主导航和路径规划、实时通信与协调、调度方案、动作执行与控制等方面，下面本文针对以上问题，分别进行叙述。

我们将任务划分为：导航、视觉、调度三个部分，并由主程序对其余程序进行调用、通信。导航采用SLAM雷达、建图、TEB方法完成地图的顶点发车与到达；视觉采用多模板库匹配算法，能从多角度、多位置对配药方案进行识别；调度采用贪心算法与多线程结构，通过ROS的订阅、发布与主程序进行通信；主程序采用状态机、标志位的方式，通过对多点的多状态分割完成状态的切换。

给出目前用的比较多的实现思路，然后结合情况，选出最优方案

## 定位与地图构建

ROS提供了多种定位和建图算法，如激光雷达SLAM（Simultaneous Localization and Mapping）、视觉SLAM、里程计等。激光雷达SLAM常用的算法包括GMapping、Hector SLAM、Cartographer等，它们通过激光雷达数据或视觉信息实时建立地图，并定位小车在地图中的位置。

其中激光SLAM搭载2D或3D激光雷达，若移动机器人功能要求较低，一般选择2D激光雷达即可，当环境较为复杂时，采用3D激光雷达进行扫描。激光雷达通过扫描环境获得包含距离和角度信息的分散点，分散点聚集后称为点云，每个信息点间存在差异，机器人通过对比点与点之间的信息来得到周围环境的变化情况，从而得到机器人的位姿。目前激光技术相对比较成熟，应用也较多，相对于视觉SLAM来说，计算简便，可靠性高，不易受光照影响，但也存在一些弊端，如获取信息较少，硬件成本高等。

而视觉SLAM主要是依靠相机进行定位，深度相机运用红外传感技术，同激光雷达通过反射光判断距离类似，深度相机操作简单，但其成本较高，搭载困难，应用场景有限，较适合室内定位及导航。

根据表1两种定位建图方案优缺点对比，本组选择利用激光SLAM获取激光雷达数据及视觉信息实时建立地图，并定位小车在地图中的位置。

表1 SLAM类型优缺点对比

|  |  |  |
| --- | --- | --- |
| SLAM类型 | 优点 | 缺点 |
| 激光SLAM | 技术成熟、精度高、较为可靠、受环境影响小 | 获得环境信息较少、硬件成本较高 |
| 视觉SLAM | 理论成熟、应用场景和采集信息丰富、成本低 | 计算量较大、环境要求严格 |

## 路径规划

ROS提供了多种路径规划算法，如Dijkstra算法、A\*算法、RRT（Rapidly-exploring Random Trees）算法等。这些算法可以根据地图和目标位置，生成安全、高效的路径，使小车能够避开障碍物，到达目标位置。

move\_base 节点中包含了全局规划和局部规划两个路径规划器。其将全局导航和局部导航链接在一起以完成其导航任务。

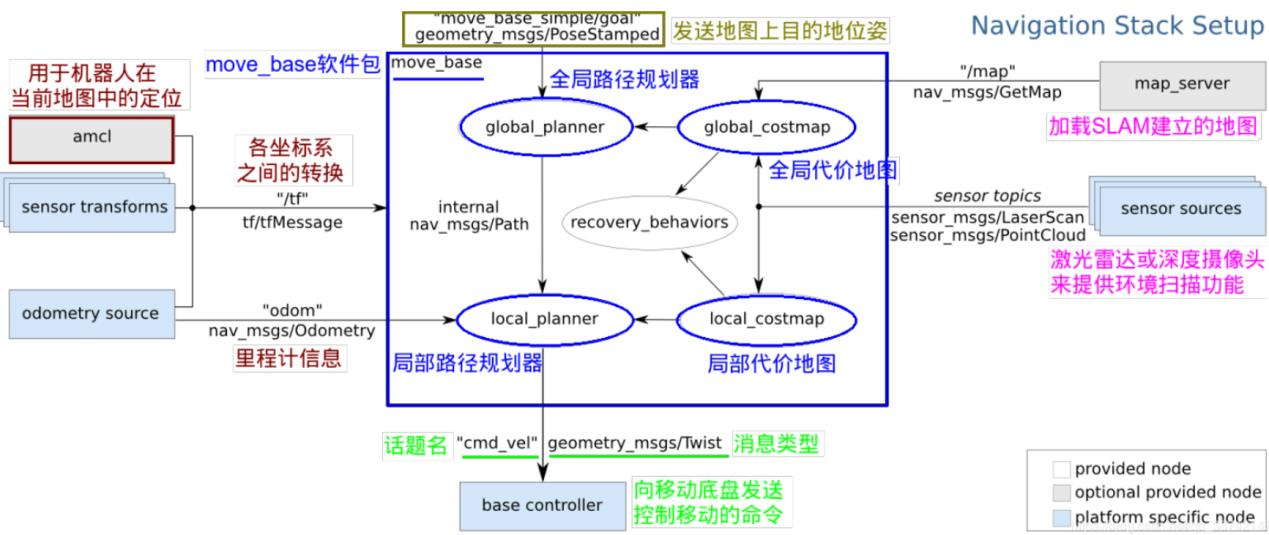


图2 movebase框架

从上图可以看到，move\_base 提供了 ROS 导航的配置、运行、交互接口，它主要包括两个部分：

1）全局路径规划（global planner）：根据给定的目标最终位置和起点进

行总体路径的规划。只关注静态障碍地图的避障；

2）本地实时规划（local planner）：根据附近的障碍物进行躲避路线规

划。例如机器人四周一个 4x4m 的方形窗口的局部路径规划。

在 ROS 的导航中，首先会通过全局路径规划，计算出机器人到目标位置的全局路线。这一功能是 navfn 这个包实现的。navfn 通过 Dijkstra 最优路径的算法，计算 costmap 上的最小花费路径，作为机器人的全局路线。

由于本次任务，小车送药途中有障碍物，若采用A\*算法，可能会使小车运输途中碰撞障碍物，所以本组采用Dijkstra 实现多点最优路径的全局规划。

## 视觉识别

当前用的较多的视觉识别方案主要有以下几种：

1. 卷积神经网络（Convolutional Neural Networks, CNN）：CNN是一种深度学习模型，广泛用于图像识别任务。通过在深度摄像头中捕获图像，并将图像输入CNN模型中进行特征提取和分类，实现对不同物体或场景的识别和分类。
2. 目标检测算法：目标检测算法能够在图像中定位和识别多个物体，并给出物体的边界框和类别。常用的目标检测算法包括基于深度学习的方法，如基于区域卷积神经网络（Region-based Convolutional Neural Networks, R-CNN）、快速区域卷积神经网络（Fast R-CNN）、单阶段检测器（Single-Shot Detector, SSD）等。
3. 特征匹配算法：特征匹配算法基于提取的图像特征，通过比对图像间的相似性来进行物体的识别。常用的特征匹配算法包括尺度不变特征变换（Scale-Invariant Feature Transform, SIFT）、加速稳健特征（Speeded-Up Robust Features, SURF）等。
4. 实例分割算法：实例分割算法能够同时实现目标检测和像素级的分割，将图像中的每个像素标注为目标的一部分。常用的实例分割算法包括掩膜区域卷积神经网络（Mask R-CNN）、全卷积网络（Fully Convolutional Network, FCN）等。

根据实际任务需求，首先，本组采用多模板匹配的算法识别取药点，通过对多点、多角度的取样，对模板库进行扩增。通过将读取出的图片进行灰度转化后，与模板库进行匹配，选取出相似度最高的模板进行匹配，通过三点定位的方法，确定所有字母的横纵坐标，并通过二值化坐标，将其相对定位确定并作为字符串进行输出，输出格式需要对各数字进行区分。其次，本次采用Pytorch训练的CNN神经网络，外加基于滑动窗口的图片分割技术，对每个数字进行分割、分别识别，最后再将数字进行组合，通过语音进行输出。

## 通信

话题通讯（异步通信）可理解为信件，寄信人无法第一时间知道收信人是否收到话题 ，(Topic) 节点间用来传输数据的总线，数据由发布者传输到订阅者，消息(Message) 即话题的数据，具有一定的类型和数据结构，有标准类型和用户自定义类型，用.msg文件定义，本组的调度方案是用话题的发布与订阅来实现调度策略控制小车。

ROS与底盘控制器之间的通信以及底盘驱动板和 ROS 的通信是基于串口通信的，STM32将机器人的底层里程计、电池状态、IMU 姿态信息等数据提交到 ROS 系统当中。如下图所示：

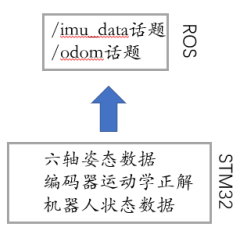


图3 通信示意图

## 调度策略

一般配送有两种调度策略，一是基于顺序原则，根据取药区顺序进行配送，二是基于贪心算法思想，得出较优的调度策略。

根据规则，由于配送药品类型、取药区堆积数量、取药小哥等待时间都对最后的得分结果有影响。因此本组采用贪心算法进行调度。【特点】

通过主程序与调度程序的通信，完成方案下发、调度方案返回、3分钟定时同步及方案清除等，通过ROS的通信，对小车进行调度。

# 技术方案

【填写说明：从原理层面，详细介绍系统所采用的技术方案，先总体介绍，给出算法实现或算法优化路线思路，然后分模块详细介绍。着重介绍算法实现思路，以及所涉及的模型、算法等；原创工作详细描述，非原创工作简略描述，并尽可能标注引用文献】

建议优化思路：

1、编码器、激光雷达以及惯导数据融合的实现方案及创新点

2、小车自主构图的实现方案及创新点

3、小车自主导航的实现方案及创新点

4、自主切换构图、导航的实现方案及创新点

5、其它

## 硬件设计

由于本次硬件是统一的小车，所以本文对此不进行多余赘述。

## 软件设计

### SLAM 建图算法原理

SLAM主要是用来解决机器人在未知环境中行进时的自身定位、运动轨迹确定和地图构建等问题。SLAM技术正是为了解决这个问题所需要用到的诸多技术的结合体，SLAM系统框架如下图所示。

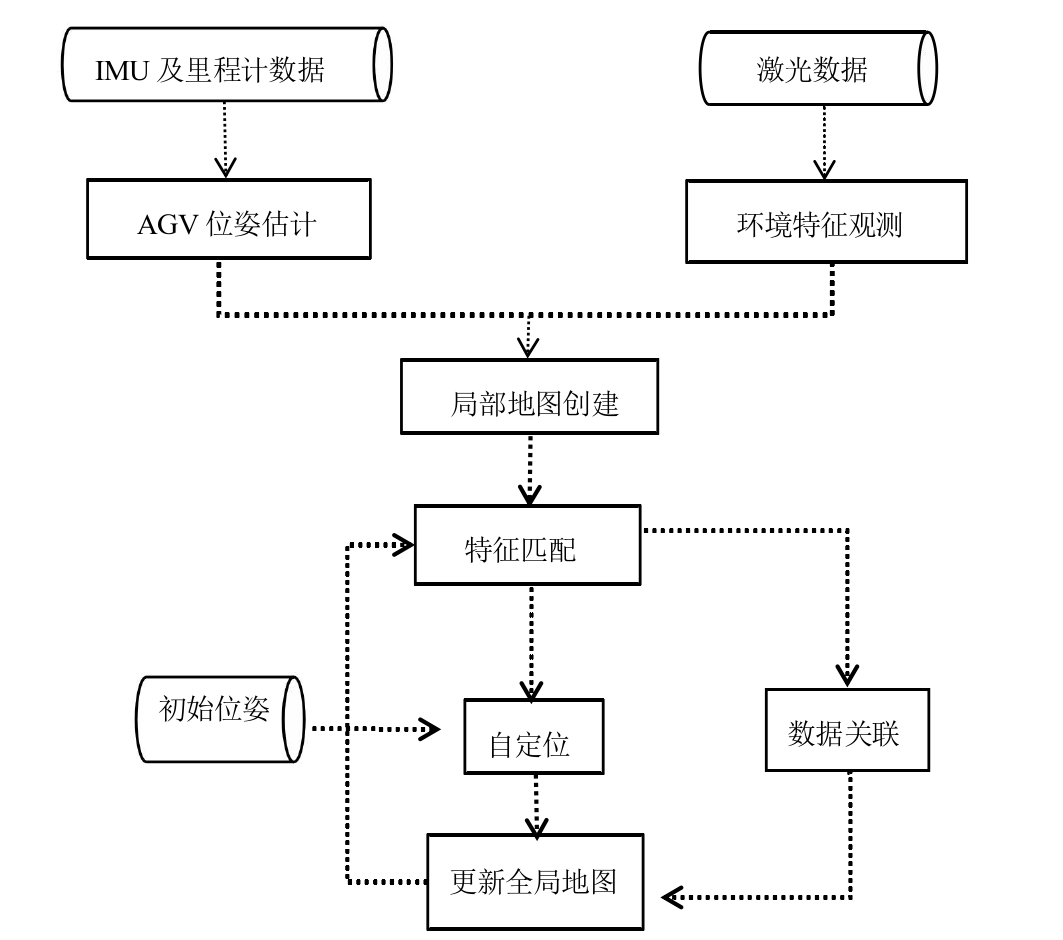


图4 SLAM系统框架

起始时刻，小车从某个起点出发，此时得到起始位姿并开始行走，惯性测量单元（Inertial Measurement Unit，IMU）和里程计开始工作，其数据可反映小车的运行速度和航偏角等信息，可进行小车的位姿估计，与此同时，激光测距仪不断获取小车所处位置周围的环境特征，二者相结合构建局部地图，再通过特征匹配结合初始位姿来实现小车的自定位。在行走的过程中，通过激光测距仪获得数据进行环境信息的实时更新，特征匹配后进行数据关联处理，若局部地图匹配正确，则全局地图纳入当前匹配好的局部地图，从而实现全局地图的不断更新。

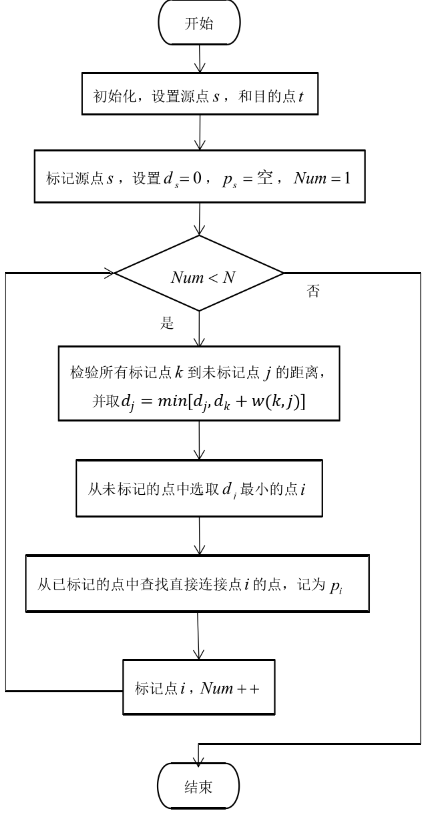
目前ROS中集成了Gmapping、Hector以及Google公司的Cartographer等SLAM建图方案，其中Gmapping是基于粒子滤波的算法，Hector是基于优化的算法，Cartographer是基于图优化的算法。Cartographer需要复杂的矩阵运算，因此计算量较大；Hector对雷达的要求较高，测量噪声小，没有回环检测，所以在机器人速度比较低的情况下，建图效果才会比较理想，其可适用于空中或者地面不平坦的情况；Gmapping是目前2D范围内应用最广的SLAM方法，其构建小场景地图所需的计算较小且精度较高，随着场景增大所需的粒子增加，每个粒子都携带一张地图，因此在构建大地图时所需内存和计算量都会增加。考虑到本次实验在小场景下完成，故选取Gmapping作为本次SLAM建图方法。

### **自主导航**

本智能车系统以树莓派为控制核心，采用惯性运动单元（IMU）传感器实现车位置、姿态判断，通过激光雷达对周围环境的实时扫描数据结合IMU数据，经树莓派处理后建立二维栅格地图。在该地图的基础上，通过机器人操作系统（ROS）中Move\_base功能包和AMCL功能包为系统提供路径规划和小车定位，结合激光雷达对前进过程中的新增障碍物进行躲避，最终实现自动导航功能。

### 路径规划

全局路径规划算法本组采用的Dijkstra算法，主要是求解一条从起始点到其他所有节点的最短路径长度，其主要原理是每次迭代时选择的下一节点是距离当前节点最近的子节点，即每次迭代都是最短路径，为了保证最终路径最短，该算法在每一次迭代时，都要更新起始点到所有遍历节点之间的最短路径，由此可以看出，Dijkstra算法虽然原理简单、易于实现，但其搜索过程中遍历节点数目过多，会存储大量临时路径，空间时间复杂度增加，因而难以适用于复杂的环境。Dijkstra算法流程图如图5所示。



**图5 Dijkstra算法流程图**

# 方案实现

【填写说明：从工程实现的角度，详细阐述第3章提出的技术方案的具体实现过程，包括但不限于数据的采集、加工、管理、分析工具的使用，以及其中所遇到的困难，解决的方法等】

## 取药环节

调度方面：通过发布取药主题、命令主题，对小车的导航程序下发指令，其订阅话题为：

* 1. 话题名为“/cmd\_send”，用于控制小车去相应的区域：
     1. 去往配药区
     2. 去往答题区
     3. 去往取药区
     4. 去往起点
  2. 话题名为“/pick\_up\_num”，用于设置去取药区的窗口：
     1. 4号窗口
     2. 3号窗口
     3. 2号窗口
     4. 1号窗口

导航方面：

## 配药环节

调度方面：通过发布配药主题、命令主题，对小车的导航程序下发指令，其订阅话题为：

* 1. 话题名为“/dispense\_window”，用于设置去配药区的窗口：
     1. C窗口
     2. A窗口
     3. B窗口
  2. 话题名为“/pick\_up\_num”，用于设置去取药区的窗口：
     1. 4号窗口
     2. 3号窗口
     3. 2号窗口
     4. 1号窗口

导航方面：

## 答题环节

识别视觉方面：采用python2调用python3的线程，python3线程内采用Pytorch2.0及CUEA11.7版本，调用根据模板库训练出的模型对目标进行识别、由于数字为3-5位数字，因此采用自适应的滑动窗口的方式，通过设置三种滑动窗口的起点、大小、间隔等，分别将图片进行三次识别，选取出识别效果最好的模式作为该识别的结果，并通过从左到右的方式返回字符串。

导航方面：到达答题区时，调用识别数字的函数，返回得到识别后的数字字符串后，将其通过ROS的主题给语音的主题进行输出，等待成功或失败的反馈。若识别成功，则立即向调度线程发送主题，表示识别成功；若失败，则立即向主程序发送识别失败字样，主程序立即向导航发送取药指令。

## 图像识别

小车在起点、起点周围进行多角度的图片抓拍，并对抓拍到的图片进行截取和分类，分别将A/B/C的图片放入A/B/C对应的文件夹内，最终的模板库为每类图片有22张，成功将取药窗口的多类情况进行识别、读取、返回。我们的方案将识别板1的取药方案定为3分钟识别一次，因此对于取药的视觉程序，并不需要作为一个单独的节点存在，因此将其封装为类，在main.py中进行调用。

小车在答题区，采用多模式的滑动窗口匹配策略，当匹配到适合的模式后，对图片中的数字进行分割识别，之后将结果从左到右进行排列，转化成字符串后返回识别结果。

## 扩展功能

### 语音播报

语音播报方面：通过发布订阅主题，进行语音的发布和接收。在中接收所需要发送的语音，通过对所需发送的语音进行分类。此外，ROS的语音播报采用安装的语音包的方式进行语音播报。当接收到的字符串为数字，则采用sound\_play自带的语音功能进行输出；当接收到的字符串为”dispense”时，则调用本地语音的”到达配药点”音频；当接收到的字符串为”pickup”时，调用本地的”到达取药点”语音。

音频处理方面：数字播报采用sound\_play中自带的默认语音festival，进行播报，文字播报采用.wav文件进行播放。由于音频文件的编码或采样率在不同操作系统上的不兼容性，导致Windows正常播放的音频在Linux上出现了混乱的情况，因此采用ffmpeg进行音频转换，将采样频率更换为44100Hz的新文件。

### 调度策略

本组实现调度策略主要基于贪心算法思想，首先通过第一次识别成功板2（如下图所示）缩短药品配送时间，然后根据每次识别板1的结果进行配送方案的输出，并利用订阅发布主题实现对小车调度的控制。

图片包含 箭头

描述已自动生成

图 识别板2示意图

由于要实现双向通信，于是本组建立了两个发布订阅者，构建的配送协议如下图所示：

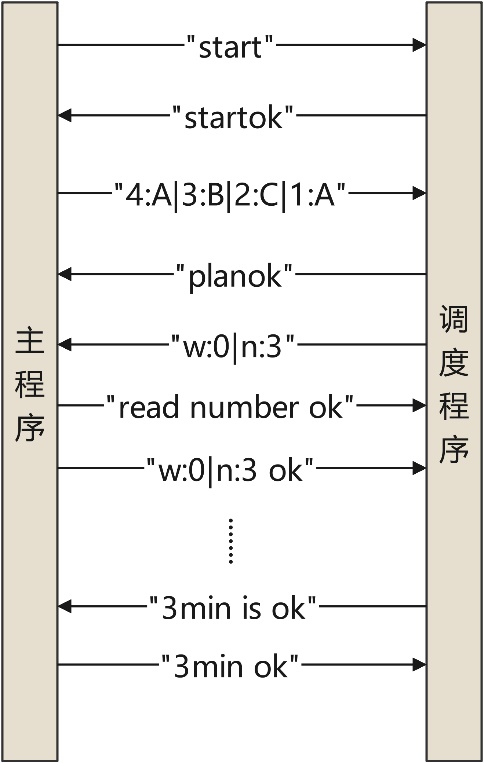


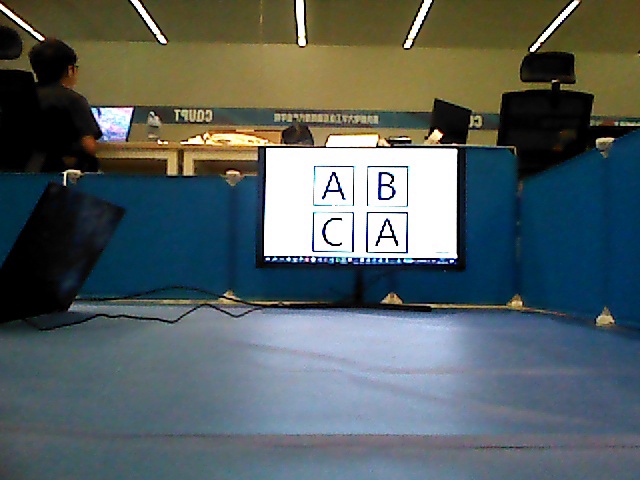
图 配送协议示意图

如上图所示，当主函数开始运行，发送"start"开启调度程序的时钟，用于同步裁判程序当前取药区的药品数量，开始成功后向主函数发送"startok"，代表其时钟开启成功。接着小车开始识别配送任务，并将其识别结果发送给调度函数，调度函数依据贪心策略进行配送方案规划，当方案制定完成后，发送"planok"告知小车，并依次发送小车每一轮需要去往的取药区及配药区。当小车完成该轮任务后，再进行下一轮任务的发送。此外，为保证3min后能够重新回到起点进行新一轮的药品任务配送识别，我们还开了一个线程进行3min指令的传递。

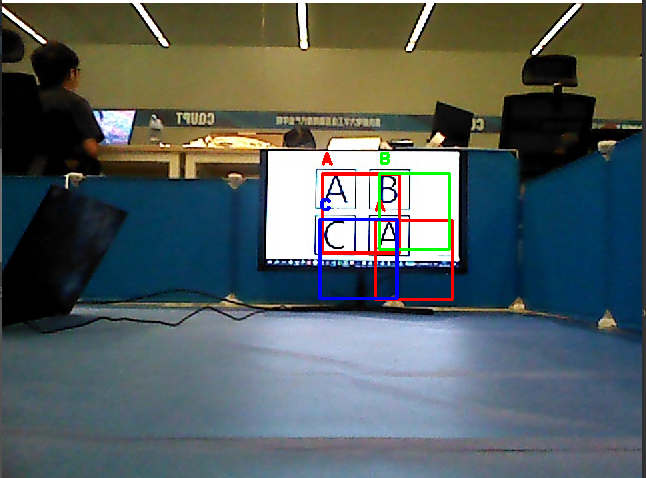
# 测试分析

【填写说明：通过测试与对比，论证系统的有效性，包括数据来源、数据规模、环境配置、测试过程、分析与结论等等。各参赛队务必重视数据测试，所有对自己作品准确性、有效性、稳定性，甚至作品受欢迎的程度的宣称，都应该得到数据结果或对比实验的支持，否则评审人有理由怀疑其真实性】

识别版1的识别与测试：

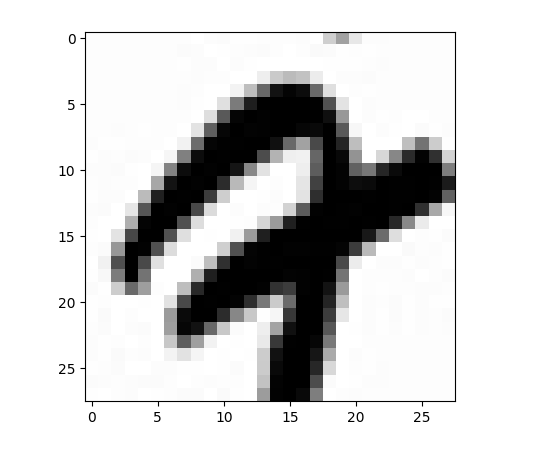


小车抓拍到的照片



识别结果

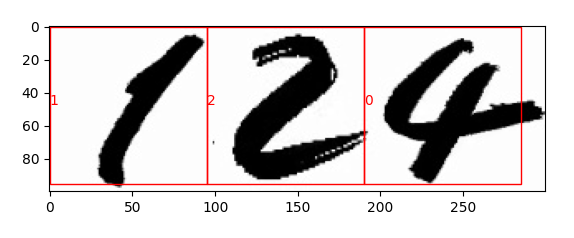
识别版2的识别：



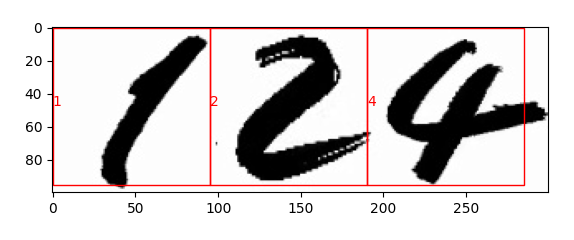
训练数字4



识别内容



部分错误的识别



修正后的识别

# 作品总结

本组作品基本实现上述所有功能，优点在于比较了许多算法，最终结合小车以及道路实际情况选用了较为合适的算法，比如调度策略使用贪心算法、路径规划中的全局规划使用Dijkstra算法，未来还可从以下几个方面进行提升，例如：提升车端感知性能、多车协同、改进算法提高视觉识别精度和速度等。在以后的日子里我们会不断学习，争取将更高级的东西融入产品当中。这次比赛有些匆匆忙忙，导致我们的作品需要完善的地方很多，这些不足我们团队会在接下来的学习中不断改进，更加严格要求细节，希望在未来的日子里能做的更好。

【填写说明：从创意、技术路线、工作量、数据和测试效果等方面对作品进行自我评价和总结，并对作品的进一步提升和应用拓展提出展望】

# 参考文献

【请按照标准参考文件格式填写】

[1]杨桂华,卫嘉乐.基于改进A\*与DWA算法的物流机器人路径规划[J].科学技术与工程,2022,22(34):15213-15220.

[2]张学峰,王仕仪,刘傲,任彬,郁洋.基于ROS的室内导航配送机器人设计[J].现代信息科技,2022,6(12):159-164.

[3]李宏辉,韩桂明,崔松涛,贺文翔,张诗森.基于ROS的无人派件精灵系统设计[J].电子质量,2022(05):73-75.

[4]曹文慧. 基于ROS的智能物流小车多目标点自主导航技术研究[D].沈阳工业大学,2021.

# 附录A 主要程序代码

|  |
| --- |
| 附录1 |
| mian.py |
| 1. *#!/usr/bin/python* 2. *# -\*- coding:utf8 -\*-* 3. import rospy 4. from std\_msgs.msg import Int32 5. import identifyPic 6. import roadPlan 7. from std\_msgs.msg import String 8. class Main(): 9. def \_\_init\_\_(self): 10. rospy.init\_node('lzzy\_main', anonymous=True) 11. self.dispense\_pub = rospy.Publisher('/dispense\_window', Int32, queue\_size=10) 12. self.pick\_up\_pub = rospy.Publisher('/pick\_up\_num', Int32, queue\_size=10) 13. self.cmd\_pub = rospy.Publisher('/cmd\_send', Int32, queue\_size=10) 14. self.arrive\_sub=rospy.Subscriber('/arrive\_flag', Int32, self.arriveSeccess,queue\_size=10) 15. self.roadPlan\_pub = rospy.Publisher('/roadPlan\_sub', String, queue\_size=10) 16. self.roadPlan\_sub=rospy.Subscriber('/roadPlan\_pub', String, self.receiveRoadPlan,queue\_size=10) 17. self.descNumFlag=False 18. self.arriveOKFlag=True 19. self.nowPlan="" 20. self.threeMinOKFlag=False 21. self.roadWindow=-1 22. self.roadNum=-1 23. self.status = -1 *# 0 起点 1 图片识别成功，配药 2方案完毕 3答题区 4取药区* 24. self.myIdentifyPic=identifyPic.IdentifyPic() 25. self.roadPlan\_pub.publish("start") 26. rospy.on\_shutdown(self.my\_shutdown) 28. def run(self): 29. loopCount=5 30. while not rospy.is\_shutdown() and loopCount>0: 31. loopFlag=True 32. picResult="" 33. while loopFlag is True and not rospy.is\_shutdown() and self.status==0: 34. picResult=self.myIdentifyPic.getPic() 35. if(len(picResult) == 15): 36. loopFlag=False 37. self.status=1 38. rospy.sleep(0.5) 39. self.roadPlan\_pub.publish(picResult) 40. if(self.status==2 and self.roadWindow!=-1 and self.roadNum!=-1 and self.arriveOKFlag==True): 41. self.dispense\_pub.publish(self.roadWindow) 42. rospy.sleep(0.1) 43. self.cmd\_pub.publish(1) 44. self.roadWindow=-1 45. self.arriveOKFlag=False 46. while not rospy.is\_shutdown() and self.status != 0 and self.status!=2: 47. rospy.sleep(0.1) 48. rospy.signal\_shutdown("Task over!") 49. rospy.spin() 50. *# self.myRoadPlan.toggle\_thread()s* 51. def receiveRoadPlan(self,msg): 52. rospy.loginfo("roadPlan"+msg.data) 53. if(msg.data=="startok" and self.status==-1): 54. self.status=0 55. if(msg.data[0]=="w"): 56. self.nowPlan=msg.data 57. self.roadWindow=int(msg.data[2]) 58. self.roadNum=int(msg.data[-1]) 59. self.status=2 61. if(msg.data=="3min is ok"): 62. self.threeMinOKFlag=True 63. if(self.arriveOKFlag==True): 64. self.status=0 65. self.threeMinOKFlag=False 67. def arriveSeccess(self,msg): 68. *# 1表示到达配药点，3表示到达答题区，5表示到达取药区，7表示到达起点* 69. if(msg.data==1): 70. print("接收到 到达 配药区") 71. rospy.sleep(1) 72. if(self.descNumFlag==False): 73. self.cmd\_pub.publish(3) 74. else : 75. self.pick\_up\_pub.publish(self.roadNum) 76. self.roadNum=-1 77. self.cmd\_pub.publish(5) 78. if(msg.data==3): 79. *# 打开摄像头识别数字* 80. identifyNumFlag=False 81. self.descNumFlag=True 82. if(identifyNumFlag==True): 83. self.roadPlan\_pub.publish("read number ok") 84. rospy.sleep(1) 85. self.pick\_up\_pub.publish(self.roadNum) 86. self.roadNum=-1 87. self.cmd\_pub.publish(5) 88. if(msg.data==5): 89. print("接收到 到达 取药区") 90. rospy.sleep(1) 91. self.roadPlan\_pub.publish(self.nowPlan+" ok") 92. self.status=4 93. rospy.sleep(1) 94. self.cmd\_pub.publish(7)      *# 发布去配药区命令* 95. if(msg.data==7): 96. print("接收到 到达 起点") 97. self.arriveOKFlag=True 98. if(self.threeMinOKFlag==True): 99. self.status=0 100. self.threeMinOKFlag=False 102. def my\_shutdown(self): 103. rospy.loginfo("Task over!") 104. if \_\_name\_\_ == "\_\_main\_\_": 105. obj=Main() 106. try: 107. obj.run() 108. except rospy.ROSInterruptException: 109. pass |

|  |
| --- |
| 附录2 |
|  |
|  |

|  |
| --- |
| 附录3 |
|  |
|  |

|  |
| --- |
| 附录4 |
|  |
|  |

|  |
| --- |
| 附录5 |
| RoadPlan调度策略 |
| 1. *#!/usr/bin/python* 2. *# -\*- coding:utf8 -\*-* 3. import rospy 4. from std\_msgs.msg import Int32 5. from std\_msgs.msg import String 6. import time 7. from threading import Thread,Lock 8. class Main(): 9. def \_\_init\_\_(self): 10. rospy.init\_node('roadPlan', anonymous=True) 11. self.roadPlan\_pub = rospy.Publisher('/roadPlan\_pub', String, queue\_size=10) 12. self.roadPlan\_sub = rospy.Subscriber('/roadPlan\_sub', String, self.receiveMessage, queue\_size=10) 13. rospy.on\_shutdown(self.my\_shutdown) 14. *# 初始值均为 1* 15. self.A = 1 16. self.B = 1 17. self.C = 1 18. self.flag = 0  *# 取药成功为1，否则为0* 19. self.lock = Lock()  *# 互斥锁* 20. self.time\_flag=0*#药品时间间隔不变* 21. *# 记录取药的顺序* 22. self.window = [] 23. *# 记录送药到哪个区域的顺序* 24. self.num = [] 25. self.peisong\_dict = {} 26. self.min3\_flag=0 *#未到3min* 27. *# 放这合不合适* 28. *# self.new\_dict = {}* 29. self.running = False 30. self.t = 0 31. self.quyao\_to\_num = {'A': 1, 'C': 0, 'B': 2, '1': 6, '2': 5, '3': 4, '4': 3} 32. self.list=['C','A','B'] 33. *# 记录上次更新时间的时间戳* 34. self.last\_update\_A = time.time() 35. self.last\_update\_B = time.time() 36. self.last\_update\_C = time.time() 37. self.min\_jishi = time.time()  *# 获取当前的时间戳* 38. *#第一次开始计数* 39. self.one=1 40. self.timeshengxiao\_a=0*#0-代表第一次变化世家，1-代表后面的计数* 41. self.timeshengxiao\_b=0 42. self.timeshengxiao\_c=0 43. self.current\_time=0.0 44. self.count=0 45. def init(self): 46. self.t = Thread(target=self.update\_values) 47. self.t.start() 48. print('开始计数') 49. *# return self.last\_update\_A,self.last\_update\_B,self.last\_update\_C* 50. def update\_values(self): 51. if self.one==1: 52. *# 记录上次更新时间的时间戳* 53. self.last\_update\_A = time.time() 54. self.last\_update\_B = time.time() 55. self.last\_update\_C = time.time() 56. self.min\_jishi = time.time()  *# 获取当前的时间戳* 57. self.one=0 58. min\_jishi = self.min\_jishi 59. self.a = 0.0 60. self.b = 0.0 61. self.c = 0.0 62. self.running = True 63. *# self.a = (120 - (current\_time - last\_update\_A)) / 2.0* 64. *# self.b = (120 - (current\_time - last\_update\_B)) / 2.0* 65. *# self.c = (120 - (current\_time - last\_update\_C)) / 2.0* 66. while self.running and not rospy.is\_shutdown(): 67. *# 获取当前时间戳* 68. if self.count == 6: 69. print('比赛完成') 70. self.toggle\_thread() 71. self.current\_time = time.time() 72. current\_time = self.current\_time 73. if current\_time - min\_jishi >= 3\*60  :*# 60* 74. print('3min时间到') 75. self.count+=1 76. *# self.min3\_flag = 1* 77. *#在这里发送信号* 78. print('3min is ok') 79. self.roadPlan\_pub.publish('3min is ok') 80. min\_jishi=current\_time 81. if self.time\_flag == 0: 82. *# 如果距离上次更新 A 的时间已经超过 2 分钟，则将 A 的值加 1* 83. if current\_time - self.last\_update\_A >= 2 \* 60: 84. self.A += 1 85. self.last\_update\_A = current\_time 86. *# 如果距离上次更新 B 的时间已经超过 1 分钟，则将 B 的值加 1* 87. if current\_time - self.last\_update\_B >= 1 \* 60: 88. self.B += 1 89. self.last\_update\_B = current\_time 90. *# 如果距离上次更新 C 的时间已经超过 40 秒，则将 C 的值加 1* 91. if current\_time - self.last\_update\_C >= 40: 92. self.C += 1 93. self.last\_update\_C = current\_time 94. if self.time\_flag == 1:*#时间变了* 95. if self.timeshengxiao\_a == 1 and current\_time - self.last\_update\_A >= 1 \* 60: 96. print('进入A的1min计数') 97. self.A += 1 98. self.last\_update\_A = current\_time 99. elif self.timeshengxiao\_a == 0 and current\_time - self.last\_update\_A >= self.a:*#第一次生效* 100. print('进入A的减半计数') 101. self.A += 1 102. self.timeshengxiao\_a = 1 103. self.last\_update\_A = current\_time 104. *# 如果距离上次更新 B 的时间已经超过 1 分钟，则将 B 的值加 1* 105. if self.timeshengxiao\_b == 1 and current\_time - self.last\_update\_B >= 0.5 \* 60: 106. print('进入B的30s计数') 107. self.B += 1 108. self.last\_update\_B = current\_time 109. elif self.timeshengxiao\_b == 0 and current\_time - self.last\_update\_B >= self.b:*#第一次生效* 110. print('进入B的减半计数') 111. self.B += 1 112. self.timeshengxiao\_b = 1 113. self.last\_update\_B = current\_time 114. *# 如果距离上次更新 C 的时间已经超过 40 秒，则将 C 的值加 1* 115. if self.timeshengxiao\_c == 1 and current\_time - self.last\_update\_C >= 20: 116. print('进入C的20s计数') 117. self.C += 1 118. self.last\_update\_C = current\_time 119. elif self.timeshengxiao\_c == 0 and current\_time - self.last\_update\_C >= self.c:*#第一次生效* 120. print('进入C的减半计数') 121. self.C += 1 122. self.timeshengxiao\_c = 1 123. self.last\_update\_C = current\_time 124. time.sleep(4) 125. *# 输出当前 A、B、C 的值以及时间戳* 126. print("A: {}, B: {}, C: {}, Time: {}".format(self.A,self.B,self.C,current\_time)) 127. def parse\_string(self, input\_str): 128. A = self.A 129. B = self.B 130. C = self.C 131. quyao\_to\_num=self.quyao\_to\_num 132. *# 将字符串以 | 分割成列表* 133. str\_list = input\_str.split('|') 134. *# 将列表中的每个元素以 : 分割成键值对，存入字典中* 135. result\_dict = {} 136. for item in str\_list: 137. key, value = item.split(':') 138. result\_dict[key] = value 139. keys1 = [key for key, value in result\_dict.items() if value == 'A']  *# 得到A要送的几个地方# 使用列表推导式筛选出值为 A 的键* 140. *# print(keys1)* 141. keys2 = [key for key, value in result\_dict.items() if value == 'C'] 142. keys3 = [key for key, value in result\_dict.items() if value == 'B'] 143. if len(keys1) != 0 and A > 0: 144. print('有{}个A要送'.format(len(keys1))) 145. *# for i in range(len(keys1)):* 146. *# print('这是第 %d 次循环' % (i + 1))* 147. if '2' in keys1: 148. self.window.append(quyao\_to\_num['A']) 149. self.num.append(quyao\_to\_num['2']) 150. keys1.remove('2') 151. elif '1' in keys1: 152. self.window.append(quyao\_to\_num['A']) 153. self.num.append(quyao\_to\_num['1']) 154. keys1.remove('1') 155. elif '4' in keys1: 156. self.window.append(quyao\_to\_num['A']) 157. self.num.append(quyao\_to\_num['4']) 158. keys1.remove('4') 159. elif '3' in keys1: 160. self.window.append(quyao\_to\_num['A']) 161. self.num.append(quyao\_to\_num['3']) 162. keys1.remove('3') 163. if len(keys2) != 0 and C > 0: 164. if keys2: 165. print('有{}个C要送'.format(len(keys2))) 166. for i in range(len(keys2)): 167. *# print('这是第 %d 次循环' % (i + 1))* 168. if '2' in keys2: 169. self.window.append(quyao\_to\_num['C']) 170. self.num.append(quyao\_to\_num['2']) 171. keys2.remove('2') 172. elif '1' in keys2: 173. self.window.append(quyao\_to\_num['C']) 174. self.num.append(quyao\_to\_num['1']) 175. keys2.remove('1') 176. elif '4' in keys2: 177. self.window.append(quyao\_to\_num['C']) 178. self.num.append(quyao\_to\_num['4']) 179. keys2.remove('4') 180. elif '3' in keys2: 181. self.window.append(quyao\_to\_num['C']) 182. self.num.append(quyao\_to\_num['3']) 183. keys2.remove('3') 184. if len(keys3) != 0 and B > 0: 185. *# while B != 0:* 186. if keys3: 187. print('有{}个B要送'.format(len(keys3))) 188. for n in range(len(keys3)): 189. *# print('这是第 %d 次循环' % (n + 1))* 190. if '2' in keys3: 191. self.window.append(quyao\_to\_num['B']) 192. self.num.append(quyao\_to\_num['2']) 193. keys3.remove('2') 194. elif '1' in keys3: 195. self.window.append(quyao\_to\_num['B']) 196. self.num.append(quyao\_to\_num['1']) 197. keys3.remove('1') 198. elif '4' in keys3: 199. self.window.append(quyao\_to\_num['B']) 200. self.num.append(quyao\_to\_num['4']) 201. keys3.remove('4') 202. elif '3' in keys3: 203. self.window.append(quyao\_to\_num['B']) 204. self.num.append(quyao\_to\_num['3']) 205. keys3.remove('3') 206. if len(keys1) != 0 and A > 0: 207. *# print('有{}个A要送'.format(len(keys1)))* 208. *# for i in range(len(keys1)):* 209. *# print('这是第 %d 次循环' % (i + 1))* 210. if '2' in keys1: 211. self.window.append(quyao\_to\_num['A']) 212. self.num.append(quyao\_to\_num['2']) 213. keys1.remove('2') 214. elif '1' in keys1: 215. self.window.append(quyao\_to\_num['A']) 216. self.num.append(quyao\_to\_num['1']) 217. keys1.remove('1') 218. elif '4' in keys1: 219. self.window.append(quyao\_to\_num['A']) 220. self.num.append(quyao\_to\_num['4']) 221. keys1.remove('4') 222. elif '3' in keys1: 223. self.window.append(quyao\_to\_num['A']) 224. self.num.append(quyao\_to\_num['3']) 225. keys1.remove('3') 226. print(self.window) 227. print(self.num) 228. *# 用于实时更新取药之后，药物余量* 229. def judge(self): 230. *# new\_dict = dict(list(zip(self.window, self.num)))* 231. *# print(new\_dict)* 232. *# 判断是否前往取药区，若是，则相应药品减1，并将其从送货的列表中移除* 233. if self.flag == 1: 234. *# yao = next(key for key, value in new\_dict.items() if value == self.num[0])* 235. yao = self.window[0] 236. if yao == 1: 237. self.lock.acquire() 238. self.A -= 1 239. self.lock.release() 240. print('A以送达') 241. elif yao == 2: 242. self.lock.acquire() 243. self.B -= 1 244. self.lock.release() 245. print('B以送达') 246. elif yao == 0: 247. self.lock.acquire() 248. self.C -= 1 249. self.lock.release() 250. print('C以送达') 251. del self.window[0] 252. del self.num[0] 253. print('window剩余要送药品', self.window) 254. print('num剩余配药区', self.num) 255. self.flag = 0 256. def toggle\_thread(self): 257. self.running = False 258. *# if not self.running:* 259. *#     self.t.join()* 260. *#随时接收消息* 261. def receiveMessage(self,msg): 262. data = msg.data 263. print(data) 264. window=self.window 265. num=self.num 266. quyao\_to\_num = self.quyao\_to\_num 267. window=self.window 268. num=self.num 269. list=self.list 270. *# 1表示到达配药点，3表示到达答题区，5表示到达取药区，7表示到达起点* 271. if (len(data) == 15): 272. *# self.roadPlan\_pub.publish("planok")* 273. self.window=[] 274. self.num = [] 275. print(self.window) 276. rospy.sleep(1) 277. self.roadPlan\_pub.publish("planok") 278. self.parse\_string(data) 279. self.roadPlan\_pub.publish('w:{}|n:{}'.format(self.window[0],self.num[0])) 280. elif (data == "start"): 281. *# 开时钟,调用* 282. self.init() 283. print("开时钟") 284. rospy.sleep(1) 285. self.roadPlan\_pub.publish("startok") 286. if (len(data) == 10): 287. print("配送完成需要下一次数据") 288. self.flag=1 289. self.judge() 290. rospy.sleep(1) 291. if(len(window)>0): 292. print(self.list[window[0]])*#c* 293. self.roadPlan\_pub.publish(self.list[window[0]]) 294. self.w=list[window[0]] 295. if self.list[window[0]]=='C': 296. rospy.sleep(1) 297. while self.C <= 0: print("等待") 298. self.roadPlan\_pub.publish('w:{}|n:{}'.format(window[0], num[0])) 299. elif self.list[window[0]]=='A': 300. rospy.sleep(1) 301. while self.A <= 0: print("等待") 302. self.roadPlan\_pub.publish('w:{}|n:{}'.format(window[0], num[0])) 303. elif self.list[window[0]]=='B' : 304. rospy.sleep(1) 305. while self.B <= 0: print("等待") 306. self.roadPlan\_pub.publish('w:{}|n:{}'.format(window[0], num[0])) 307. if (data == 'read number ok'): 308. print("#时间变少") 309. self.time\_flag=1 310. self.a = (120 - (self.current\_time - self.last\_update\_A)) / 2.0 311. self.b = (60 - (self.current\_time - self.last\_update\_B)) / 2.0 312. self.c = (40 - (self.current\_time - self.last\_update\_C)) / 2.0 313. *# print(self.a)* 314. *# print(self.b)* 315. *# print(self.c)* 316. self.last\_update\_A = time.time() 317. self.last\_update\_B = time.time() 318. self.last\_update\_C = time.time() 319. if(data == '3min ok'): 320. print('收到3min ok') 321. self.roadPlan\_pub.publish('收到3min ok') 322. self.min3\_flag = 0 323. def run(self): 324. while not rospy.is\_shutdown(): 325. rospy.sleep(0.1) 326. def my\_shutdown(self): 327. rospy.loginfo("Task over!") 328. if \_\_name\_\_ == "\_\_main\_\_": 329. obj = Main() 330. obj.run() |

**二、技术报告格式要求：**

为规范报告，规定以下报告格式，该格式仅供参赛队参考，不作为评估报告质量的依据

1、技术报告（论文）文本结构规范：

1) 封面

封面内容应包括有报告名称、研究论文题目（可选）、参赛学校、队伍名称、参赛队员、带队老师姓名等内容。内容格式参照附录1：技术报告封面样本。

其中，如果技术报告中包含了研究论文附录部分，需要将研究论文题目列写在封面技术报告题目之下。篇目数量不限，论文题目力求简短，概括精髓，一般控制在25个字以内。

2) 关于技术报告以及研究论文使用授权的说明：单设一页，排在封面后。具体格式参见附录2：技术报告以及研究论文使用授权书。

3) 目录：应是技术报告的提纲，即报告组成部分的章节标题。

4) 主要符号表：如果论文中使用了大量的物理量符号、标志、缩略词、专门计量单位、自定义名词和术语等，应编写成注释说明汇集表。若上述符号和缩略词使用数量不多，可以不设专门的汇集表，但必须在论文中出现时加以说明。

5) 技术报告内容应该包括有引言(第一章)、正文(若干章)、结论(最后一章)等部分。

在引言部分，对于智能汽车制作情况进行概述， 对于相关联的文献进行综述。阐明后面报告内容框架安排。

正文部分，是技术报告的主体。应该分章节对于本细则第一部分所涵盖的内容进行说明。其中应该突出制作过程中创造、发明与分析研究等方面的内容。

如果技术报告后面附带有研究论文，其中研究论文的内容可以在正文中进行简要介绍。

结论部分：应该概括说明模型汽车制作过程整体情况和技术指标。指出其中存在的问题和可以进一步改进方向，对工作中遇到的重要问题要着重指出，并提出自己的见解。

6) 参考文献：列出作者直接阅读过、在正文中被引用过、正式或非正式发表的刊物、文献及资料。参考文献的写法应遵循国际上通用的习惯以及我国有关国家标准规定，且应全文统一，不能混用。参考文献一律放在论文结论后，不得放在各章之后。在论文正文中引用了参考文献的部位，须用上标标注[参考文献序号]。

7) 附录应包括如下内容：

(1) 有些不宜放在正文中，但有参考价值的内容，如公式的推演、编写的算法语言程序设计、图纸、数据表格等。（可选）

(2) 程序源代码；(必须包括)

(3) 研究论文。 (可选，数量不限)

如上部分，附录编号按照 附录A,B,C,D… 依次进行。

8) 编排说明：以上各项首页均从奇数页开始。