机器人综合实验报告

实　验　题　目	基于 ROS 的智能感知与自主导航的多功能救援车
指　导　教　师	黄英亮
小　组　组　员	
提　交　时　间	2025.6.8

目录

（一）实验目的

本实验旨在全面了解基于 ROS 的智能感知与自主导航的多功能救援车的功能。通过实验，深入认识救援车的机械结构、硬件配置和软件系统，验证其在复杂环境下的自主操作能力。具体目标包括：

1. **机械结构分析**：了解救援车的机械臂和底盘的设计功能及其特点。
2. **硬件配置评估**：评估救援车的传感器、执行器和控制器等硬件的性能。
3. **软件系统测试**：测试基于 ROS 的控制软件在自主导航、目标识别和抓取等方面的表现。
4. **综合功能验证**：验证救援车在复杂环境下的综合操作能力。

（二）实验设备

1. **履带式机器人底盘**：具备自主导航、避障、爬坡能力。
2. **六自由度机械臂**：基于 ROS 系统，具备自主运动规划能力，搭载图像视觉和人机交互模块。
3. **传感器设备**：包括深度摄像头（Intel RealSense D435）、激光传感器、姿态传感器等。
4. **控制平台**：基于 Ubuntu 的机器人操作系统（ROS）和 Moveit！框架。

（三）实验步骤

1. 系统初始化

启动 ROS 系统，加载机械臂和底盘的集成 launch 文件，确保所有传感器正常工作。此步骤的目的是确保实验设备的各个部分能够协同工作。

2. 机械结构分析

机械结构：

- 全地形底盘：全地形底盘采用履带式设计，并配有四个可以摆动的副履带辅助越障。每个副履带的摆臂各自独立，通过高减速比的减速箱提供高扭矩，增

加救援机器人的全地形越障能力。

- 二自由度自稳云台与激光雷达：SLAM 建图的基础，始终保持雷达水平，提高建图质量。
- 可扩展性：可以在原有的底盘基础上在上层搭载更多模块，可以提供更为多样的功能。

先对地震救援机器人的总体机械结构进行大致设计，主要为底盘与搭载结构。对结构设计中进行创新点的发掘，将机器人的雏形在 Solidworks 中进行简单的设计。然后对于小细节进行完善与改进。对照国标件对零件进行合理选择，并根据其重量以及所欲要的速度与扭矩选择电机与减速箱，选择合适的减速比。运用 SolidWorks Simulation 有限元分析对材料进行受力分析。最后，将设计好的零件图发往厂家去加工制作，在材料到齐后对其进行组装，并且在调试后对于不合理的地方进行改装与改造。下图 1 为机械部分使用 Solidworks 软件建模的地震救援机器人结构图：

图 2.1 机器人整车结构图

3. 电子配置评估

（一）底盘电子配置

地震救援机器人电子控制分为主履带驱动单元、雷达云台自稳单元和上位机通讯单元、传感器单元、其他模块驱动单元五个部分组成，各单元均采用基于 Cortex-M4 内核的 STM32F405RGT6 微控制器作为主控核心，各单元之间相互独立、分散式分布，但同时各单元间能够相互传输数据，并通过上位机通讯单元与上位机通讯，共同构成底层通讯架构，并与蓄电池动力源共同构成底层电源架构。

1. 底层电源架构

由于地震环境恶劣，为使各控制结构能够相对独立运作，以期在某一部分发生故障时，不影响主体部分的运转，故采取双电源供电的结构。双供电结构分为两大部分：底层供电单元和上位机供电单元。由于上位机单元各个模块均为标准化的工业级产品，需要电流较小，极少出现过热、短路、断路等电路故障，因此采用集中式供电方案，各个单元相对独立、互不影响，便于统一控制和管理供电；使用一块大疆电池单独供电。

底层电路各单元及电机驱动所需电流较大，为在保证各个模块正常工作的情况下降低电源电流，供电单元采取分布式供电。

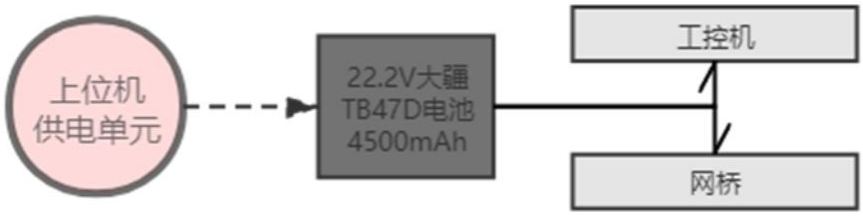


图　3.1 上位机供电框架图

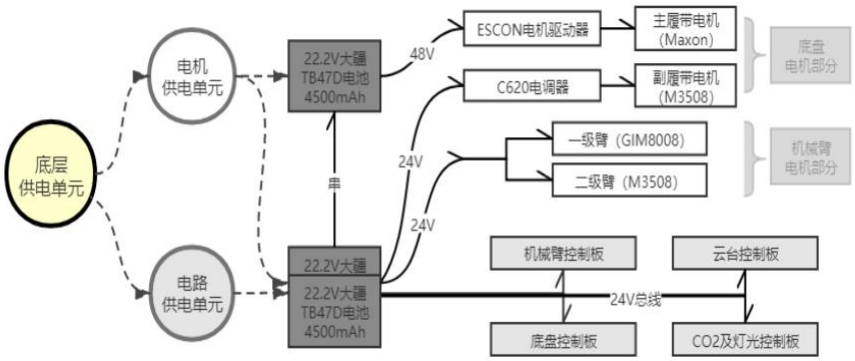


图 3.2 底层供电框架

2.主履带及副履带驱动单元

主履带采用了 ESCON-Module-50-8 驱动板搭配 MAXON 电机，通过 PWM 输出信号，驱动板将此信号结算为速度，并传输给电机，实现主履带电机的速度闭环控制。两者配套，精度较高。副履带电机的控制采用增量式 PID 算法，

```
PID->iIncpid
= PID->Proportion *(( PID->Error_1 -PID->Error_2 )+
  ( PID->Integral * PID->Error_1 * Time ) +
  ( PID->Derivative * (PID->Error_1 - 2* PID->Error_2 + PID->Error_3) * Timed )
```

图 3.2.1 PID 计算公式

根据公式，需要进行参数整定。即确定调节器的比例系数、积分时间、微分时间和采样周期的具体数值。以此改善系统的动态和静态指标，取得最佳的控制效果。

3.自稳云台单元

地震救援时路面环境坎坷崎岖，十分容易发生摇晃，造成姿态的改变，从而对激光雷达的性能和建图的准确性产生重要影响。因此，此机器人将激光置于自主设计的二维云台上，利用集成好的 JY61 进行姿态解算，经过卡尔曼滤波和互补滤波，求解出欧拉角。利用反向补偿的思想，控制可二维运动的两个 AX-12 数字舵机平台实时动态响应，保持激光的平衡和稳定。并使用包含三轴加速度计、三轴陀螺仪和三轴磁强计陀螺仪 MPU9250 作为惯性单元，STM32F10RCT6 单片机作为独立运算单元，通过四元数法进行姿态解算相比，将会更加稳定，进一步提高了建图的准确性。

4.通讯网络架构

在地震救援机器人进行救援的过程中，对底层各个模块之间的通信速度和对于来自上位机和操作者的指令响应速度具有很高的要求。因此，在底层通讯架构的搭建中对通信方式的选择至关重要，上位计算机和下位各关节控制器间的通信既要满足硬件连接简单，扩充方便，又要满足通信的高可靠性和实时性。我们的底层通讯网络采用 CAN 总线作为通信标准，采用上、下位机二级分布式结构， 我们通过 USB 串口通信与 CAN 通信高速转接模块在上位工控机与底层电控层之间进行网络通讯。

此机器人通讯网络采用 ISO 11898 标准的 CAN 总线作为通信标准，在电控层的多个模块的 MCU 节点之间传递信息。CAN 总线是一种有效支持分布式控制和实时控制的串行通讯网络，与其他的通信网络相比具有可靠性高、实时性强和灵活性好的优点，非常适合作为机器人控制系统中电控层的通讯方式。

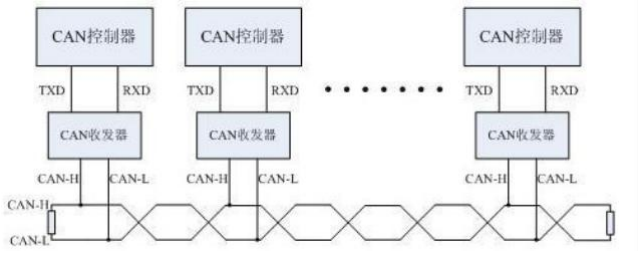


图 4.1 CAN 总线示意图

此机器人采用标准长度 ID，并使用 32 位筛选器的标识符掩码模式，以过滤不符合的 ID 信息至 FIFO 中，一帧信息共 8 字节，其中前后 4 字节分开，每 4 个字节传递一个整型或单精度浮点数。

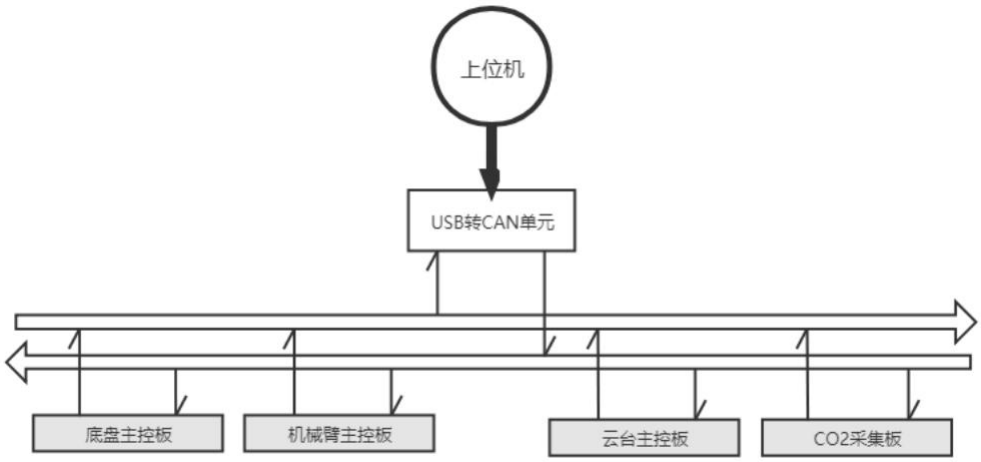


图 4.2 通讯框架

（二）机械臂电子配置

机械臂电子控制拟采用基于 Cortex-M4 内核的 STM32F405RGT6 微控制器作为主控核心，经过 USB 转 CAN 板进行数据转换与上位机进行通讯。

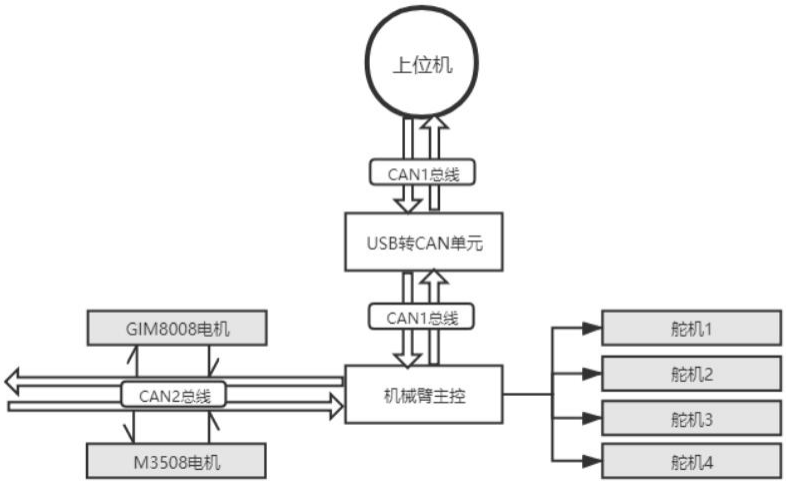


图 5.1.1 整体电控框图

机械臂一级臂拟使用 GIM8008 机器人关节电机控制，二级臂拟使用 M3508 无刷电机控制，顶端爪子拟使用 4 个数字舵机（HV 大扭力舵机）控制，底座的旋转拟使用 60KG 数字舵机控制。通过机械臂控制板上的信号输出，可以使其协同运作，实现目标物体的抓取。对于舵机的选择，我们之所以选择大扭矩舵机，一方面是因为机械臂重量的问题，机械臂运动时，如果上层关节不能很好地依靠自身扭力自稳，就会造成整体中心不稳，被抓取的目标物也会晃动；另一方面，可以使携带的传感器不受干扰地进行工作。

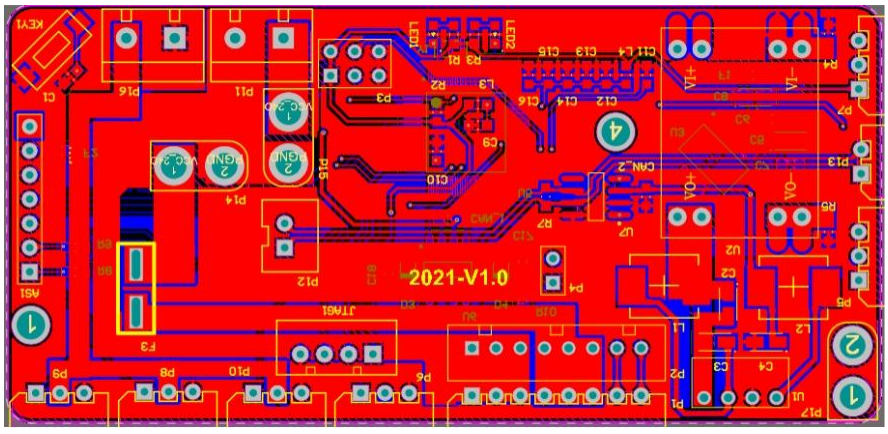


图 5.1.2 机械臂控制板 PCB 图

• 机械臂电机控制

机械臂的一级臂与二级臂使用了 PID 闭环控制，其中一级臂的 HT-03 云台电机使用配套的 MIT 驱动板，实现了位置-速度双闭环操作，本机械臂通过 CAN 总线发送信号给驱动板，以实现到达指定位置的运动；其中二级臂使用 M3508 无刷电机

与指示目前位置的电位器搭配，并在自制操控板上，通过 MCU 进行位置-速度双闭环操作，其中位置闭环通过采集电位器电压判断目前位置，并与目标位置进行位置闭环操作，并计算运动速度，经过速度闭环，并传递给电机

对于机械臂的控制，给的是位置命令，需要精确的位置环，因此整定位置-速度-电流三环控制系统。从内到外依次是电流环、速度环和位置环。

电流环：电流环的输入是速度环 PID 调节后的那个输出，电流环的输入给定和“电流环的反馈”值进行比较后的差值在电流环内做 PID 调节输出给电机，电流环的输出就是电机的电流。

速度环：速度环的输入就是位置环 PID 调节后的输出以及位置设定的前馈值，我们称为“速度设定”，这个“速度设定”和“速度环反馈”值进行比较后的差值在速度环做 PID 调节（主要是比例增益和积分处理）后输出就是上面讲到的“电流环的给定”。速度环的反馈来自于编码器的反馈后的值经过“速度运算器”得到的。

位置环：位置环的输入就是外部的脉冲（通常情况下，直接写数据到驱动器地址的伺服例外），外部的脉冲经过平滑滤波处理和电子齿轮计算后作为“位置环的设定”，设定和来自编码器反馈的脉冲信号经过偏差计数器的计算后的数值在经过位置环的 PID 调节（比例增益调节，无积分微分环节）后输出和位置给定的前馈信号的合值就构成了上面讲的速度环的给定。位置环的反馈也来自于编码器。

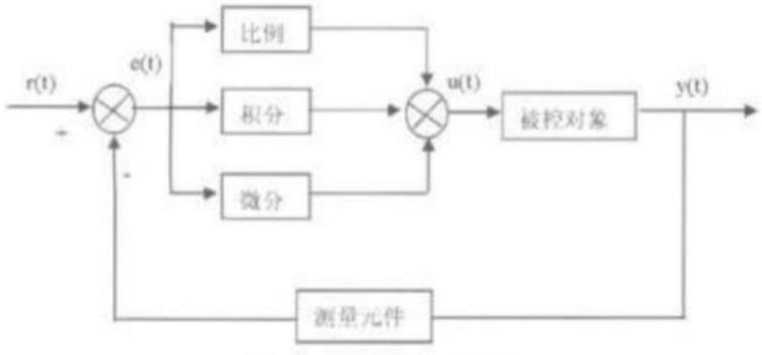


图 5.2.1 PID 闭环控制图

根据对机械臂的实际应用，经过多次研究，分析，实验选取下列的 PID 计算公式：输出，实现对电机的闭环控制，故电机采用了 CAN 总线指令控制。

$$
\begin{aligned}
&PID->iIncpid \\
&= PID->Proportion *((PID->Error\_1 - PID->Error\_2)+ \\
&\quad (PID->Integral * PID->Error\_1 * Tine) + \\
&\quad (PID->Derivative * (PID->Error\_1 - 2 * PID->Error\_2 + PID->Error\_3) * Tined)
\end{aligned}
$$

其中：

PID->Proportion：比例系数

PID->Integral：积分系数

PID->Derivative：微分系数

Tined：微分时间

根据公式，需要进行参数整定。即确定调节器的比例系数、积分时间、微分时间和采样周期的具体数值。整定的实质是通过改变调节器的参数，使其特性和过程特性相匹配，以改善系统的动态和静态指标，取得最佳的控制效果。

通过“Serial Chart 串口波形绘制”软件，观察响应曲线，调节 PID 参数，使得电机响应速度相对稳准快。

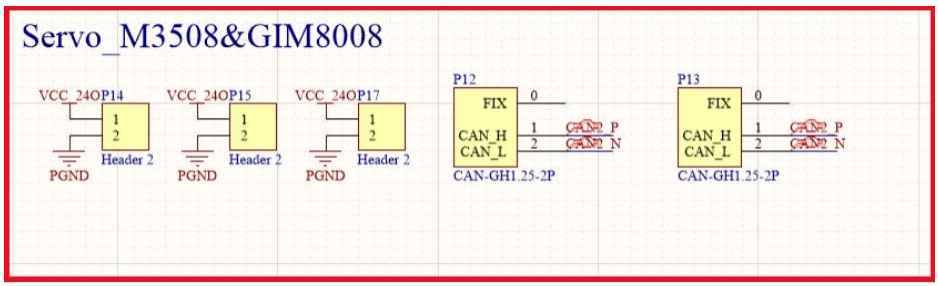


图 5.2.2 电机控制电路原理图

• 机械臂舵机控制

对于舵机的控制，我们拟在原先 PWM 控制角度的基础上，加入了步进的思路，即让舵机从当前角度平滑地变化至目标角度。这样可以使得机械臂的运动更加稳定，不会出现急剧的变化。

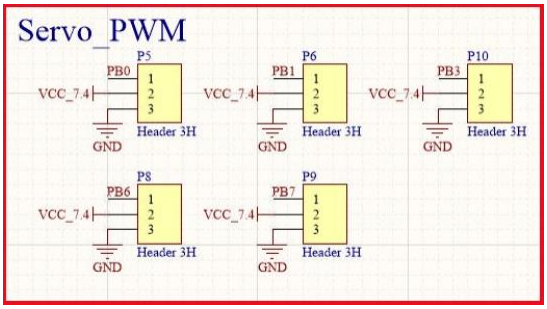


图 5.3.1 舵机控制电路原理图

• 通信网络架构

机器人的分布式控制系统中，对通信方式的选择至关重要，上位计算机和下位各关节控制器间的通信既要满足硬件连接简单，扩充方便，又要满足通信的高可靠性和实时性。我们拟采用 CAN 总线作为通信标准，采用上、下位机二级分布式结构，上位机负责整个系统管理及运动学计算、轨迹规划等，下位机由多个 CPU 组成，每个 CPU 控制一个模块。CAN 总线是一种有效支持分布式控制和实时控制的串行通讯网络，与一般的通信网络相比具有可靠性高、实时性和灵活性好的优点，非常适合作为机器人控制系统中的通讯方式。在我们的机械臂系统中，上位机通过自主设计的 USB 转 CAN 单元，连接到 CAN 网络，与底层各个模块通信。

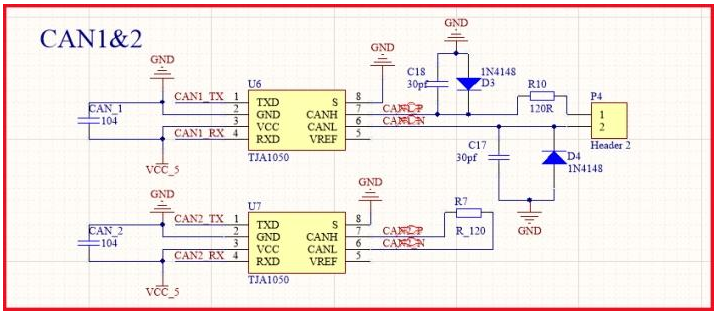


图 5.3.2 通信电路原理图

• 电源管理及隔离设计

外部输入主控板的电压为 24V 直流电压（电机工作电压），通过可调降压模块降至 6.8V（舵机工作电压），再用 DMO2 芯片降至 5V（树莓派和一些传感器工作电压），最终通过 AMS1117 芯片降至 3.3V 供 STM32F4 芯片使用。

我们拟采用模拟地和数字地隔离的方法使电源供电更加安全。

同时在电路设计中拟使用电源隔离、通讯隔离等隔离设计，使电路可以稳定运作，抗压抗干扰能力强。

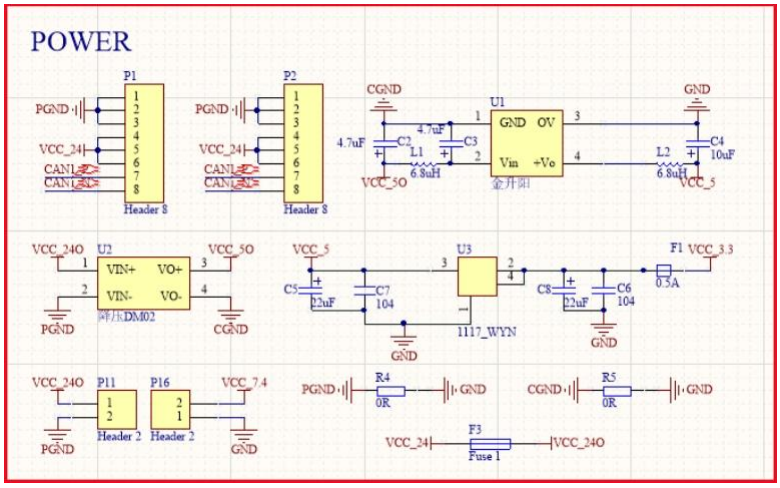


图 5.3.3 电源电路原理图

4. 软件系统测试

我们拟采用的系统的整体框架的工作流程为：

机载传感器（如激光传感器、姿态传感器、摄像头等）将相关数据与机械臂实时状态通过 5.8Ghz 网络发送给性能较强的终端机、终端机处理后向工控机发送运动指令，经工控机处理后发送给底盘使机械臂产生实际运动，执行相关动作。使用 Intel RealSense D435 深度摄像头等外部环境输入设备实时获取外部环境信息并生成点云，在机械臂的自主规划工作空间的路径中加入外部环境，通过 RRT 算法随机点生成树找到一条在三维空间内可以到达的路径，通过 ROS 与 Moveit！框架完成路径规划与避障，从而实现机械臂的智能自主控制。

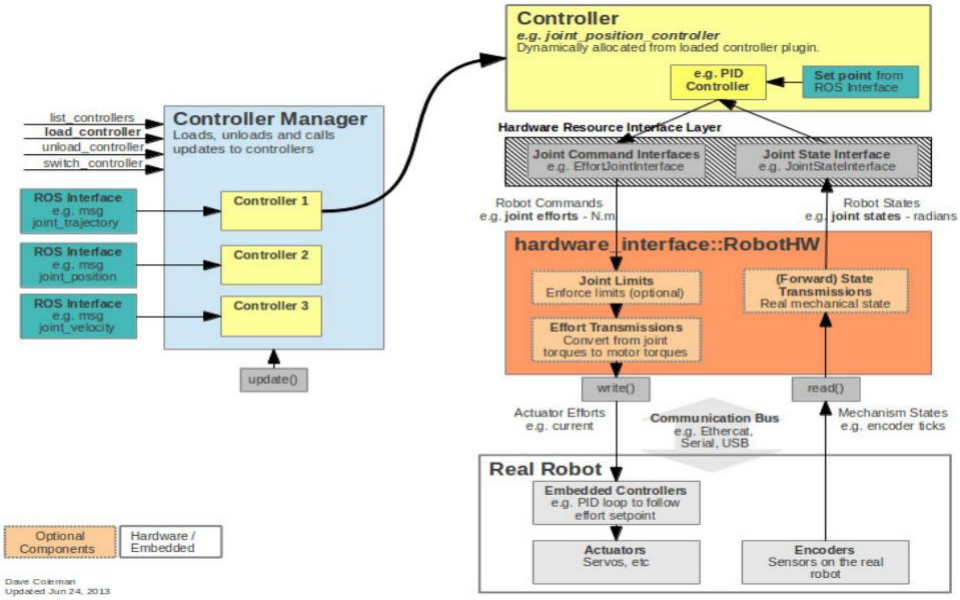


图 6.1　ROS 软件系统控制流程图

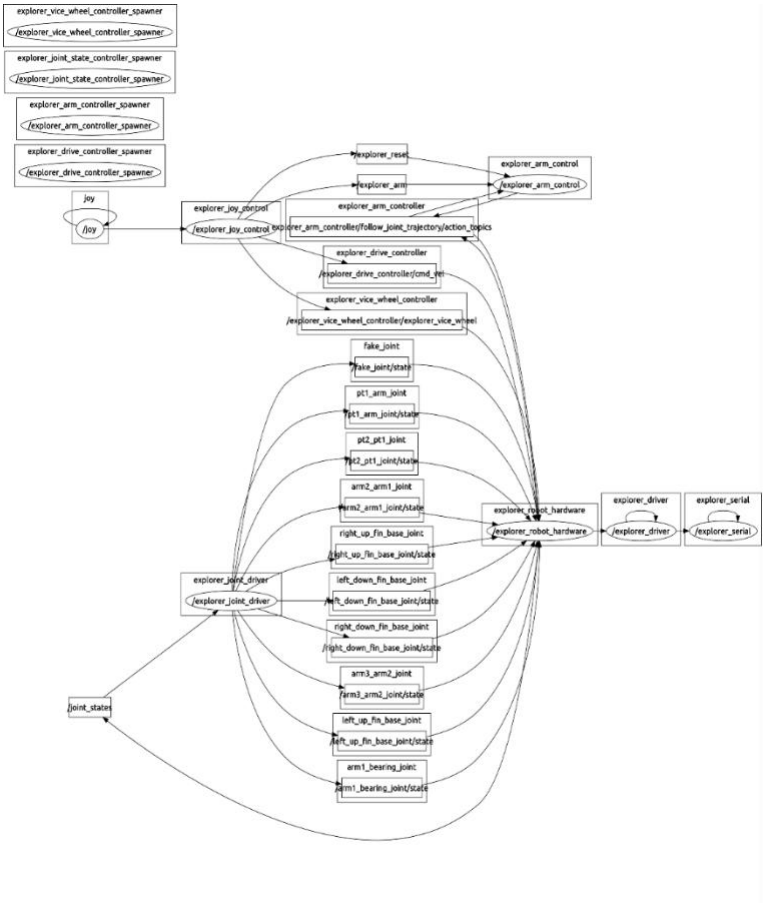


图 6.2　ROS 软件系统机械臂部分节点图

为了实现上述软件整体框架，我们拟使用基于 X86 架构的 Intel Core i5 平台的工控机作为上位机，该平台具有高性能、可拓展性强、易于维护等优点，并结合使用依托于 Ubuntu 的机器人操作系统（ROS，即 Robot Operation System）对机械臂进行控制和底层通信。ROS 系统作为目前一个成熟的机器人控制及应用平台，具有通用性强、跨语言、稳定性高以及扩展性强等优点。

机器人与终端机之间的无线通信拟使用成熟的 5.8GHz 网桥间通信方案，该方案具有信号稳定、延迟低、传输距离远等特点。机械臂机载摄像头拟采用深度摄像头和普通摄像头结合的方式。其中深度摄像头采用 Intel RealSense D435，它能够在提供 RGB 彩色视频图像的同时，提供深度信息，也就是视距内的点的距离信息。将使用摄像头采集到的 RGB 彩色视频流经过远程图传传递到控制电脑端，使用基于深度学习的目标识别技术对杂物进行识别并定位。ROS 中的可视化工具 RViz 可以对机械臂进行调试和仿真，在每一次输入末端执行器的位姿信息之后在 Rviz 之中显示机械臂的规划的目标状态，并且在重新规划之后显示新的目标状态。并通过向其中发送信息来实时更正机械臂的状态使得机械臂的状态可以与实物状态实时联系起来。将深度摄像头生成的点云显示出来，并结合多传感器，对机械臂的运动写入碰撞保护。这个过程通过改良的 RRT 算法计算出可行的机械臂运动路径。

- 视觉定位系统与控制规划

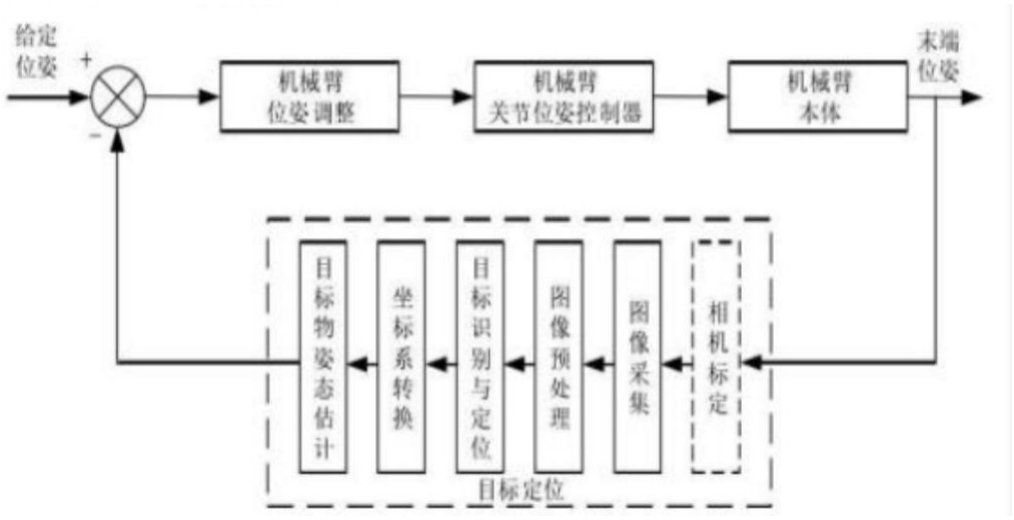


图 6.3.1　视觉定位系统与控制规划

为实现目标的快速定位与抓取，考虑使用位姿式的控制方式，控制系统结构如图所示。摄像机对图像信息进行采集，上位机对采集的目标图像进行信息分析，并通过坐标转换得出目标物相对于机械臂的位置，在根据机械臂当前的位姿信息对机械臂的运动轨迹进行规划，最后奖位置点信息传给运动控制系统，控制机械臂运动。

·图像深度信息获取

机械臂机载摄像头采用深度摄像头和普通摄像头结合的方式。其中深度摄像头采用 Intel RealSense D435，它能够在提供 RGB 彩色视频图像的同时，提供深度信息，也就是视距内的点的距离信息。RealSense D435 采用结构光来获取图片的深度信息。，其基本原理是由结构光投射器向被测物体表面投射可控制的光点、光条或光面结构，并由图像传感器获得图像，通过系统几何关系，利用三角原理计算得到物体的三维坐标。

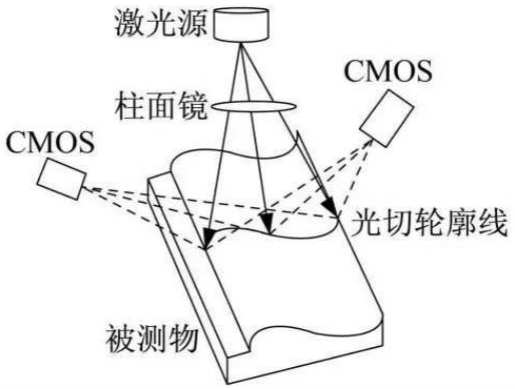


图 6.3.2　结构光法测量原理图

·图像识别

1）深度学习框架：YOLOv3

YOLO（You Only Look Once）是一种基于深度神经网络的对象识别和定位算法，YOLO 将目标区域预测和目标类别预测整合于单个神经网络模型中，实现在准确率较高的情况下实时快速目标检测与识别，其增强版本 GPU 中能跑 45fps，简化版本 155fps。YOLOv3 在 YOLOv2 的基础进行了一些改进，这些更改使其效果

变得更好。 在 320×320 的图像上，YOLOv3 运行速度达到了 22.2 毫秒，mAP 为 28.2。其与 SSD 一样准确，但速度快了三倍。

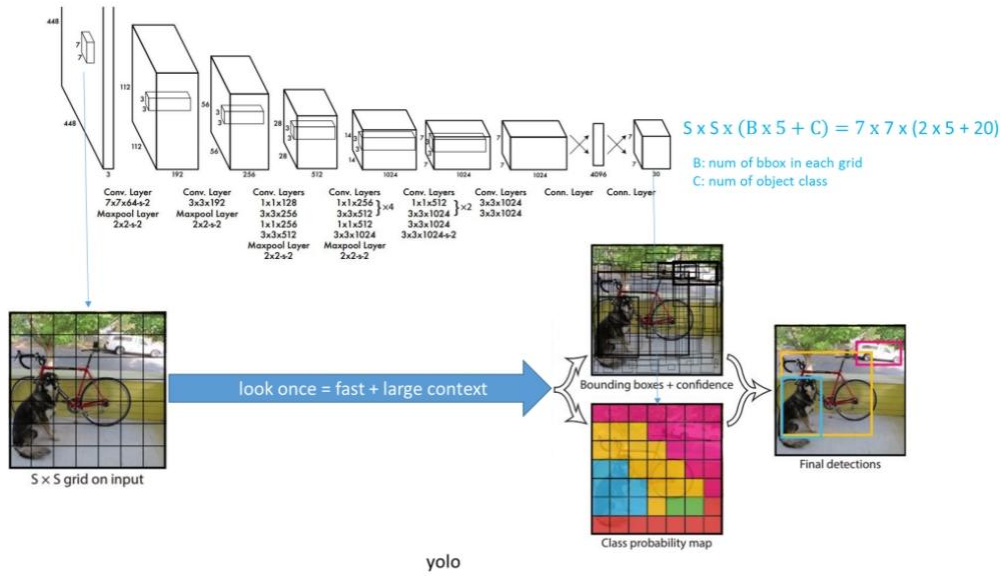


图 6.3.3 YOLO 网络结构图

YOLO 将对象检测重新定义为一个回归问题。它将单个卷积神经网络(CNN)应用于整个图像，将图像分成网格，并预测每个网格的类概率和边界框。例如，以一个 100x100 的图像为例。我们把它分成网格，比如 7x7。

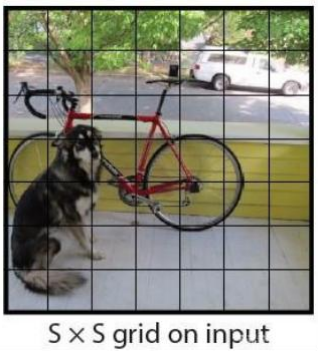


图 6.3.4 网格划分示意图

然后，对于每个网格，网络都会预测一个边界框和与每个类别（汽车，行人，交通信号灯等）相对应的概率。

图 6.3.5　Yolo 网络的预测概率

每个边界框可以使用四个描述符进行描述：边界框的中心、高度、宽度、值映射到对象所属的类。

此外，该算法还可以预测边界框中存在对象的概率。如果一个对象的中心落在一个网格单元中，则该网格单元负责检测该对象。每个网格中将有多个边界框。在训练时，我们希望每个对象只有一个边界框。因此，我们根据哪个 Box 与 ground truth box 的重叠度最高，从而分配一个 Box 来负责预测对象。

最后，我们对每个类的对象应用一个称为“非最大抑制（Non Max Suppression）”的方法来过滤出“置信度”小于阈值的边界框。这为我们提供了图像预测。

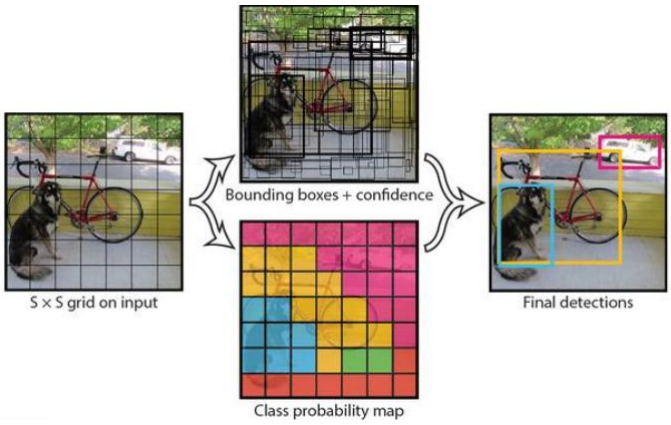


图 6.3.6 最终的预测效果

2）深度学习在机器人上的应用

我们拟使用的 YOLOv3 深度学习框架是基于 C++的实现，将其移植到 ROS 上用于机器人的目标检测。

使用预先训练好的 caffemodel 模型文件在本地电脑上进行目标识别的正确率较为理想，如下图所示：

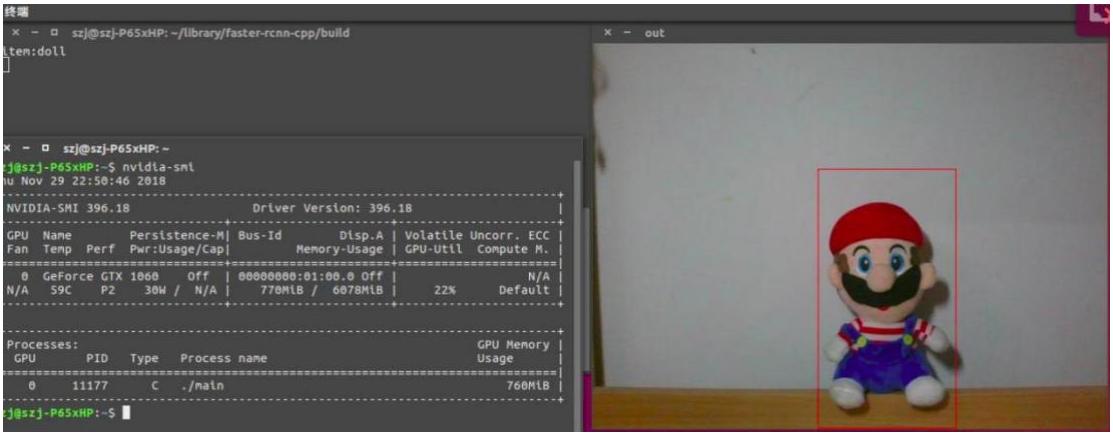


图 6.3.7 深度学习在本地电脑上的测试结果

我们希望将深度学习移植到机器人操作系统中，所以将其移植到 ROS 平台上后进行测试和在本地电脑上相比同样得到了非常高的效果。如下图所示。

图 6.3.8 深度学习在机器人上的实际应用

我们希望将深度学习移植到机械臂运行体系中，所以将其移植到 ROS 平台上后进行模拟，测试在杂物实际检测中的效果。如下图所示。

图 6.3.9　深度学习在杂物检测的实际应用

我们将常见的杂物逐个摆放货架上进行测试，使用机械臂的摄像头对视野中的目标进行检测并识别出常见杂物种类。

图中红色方框框选出的区域就是使用深度学习算法对机械臂上的摄像头得到的单帧视频信息识别后的结果。可以看到，预先训练好的 caffemodel 模型文件进行实物目标测试，准确率较高，能够框选出常见的杂物种类，并且将其一一框选出来，同时标注物品的种类。

所以只要预先训练好的样本集的规模符合要求，使用深度学习算法进行目标识别即使对于难度较大的目标和比较复杂的环境中进行识别也能够轻松胜任。

• 运动规划

Rapidly-exploring Random Tree 是一种通过随机构建空间填充树来有效搜索非凸、高维空间的算法。树是从搜索空间随机抽样的样本逐步构建的，并且本质上倾向于探索大部分的未知区域。因此其广泛应用于路径规划等。而其改造版 RRT 算法则可以从更高维来快速的探索未知的区域。因此可将改良的 RRT 算法用于机械臂的全方位自主移动协调规划之中，通过随机点生成树来找到一条在三维空间内可以到达的路径，从而实现机械臂的智能控制。本项目考虑使用 RRT 算法来实现机械臂自主运动的规划。

RRT 算法流程：

以 qstar 为起始点，qgoal 为目标点，建立随机扩展树，其初始化和扩展步

骤如下：

1)、选择 qstar 为根节点；

2)、在空间中随机选择一个采样点 qrandom；

3)、在已产生的随机树上，搜索一个距离采样点 qrandom 最近的 qnear；

4)、在点 qrandom 与点 qnear 的连线上，以一定的步长 r，并从 qrandom 出发创造一个新的节点 qnew。如果从 qrandom 到 qnew 之间没有碰撞发生，则将这个新的节点 qnew 加入到搜索树上并同时从 qrandom 到 qnew 的边插入到搜索树中；反之，则选点失败无拓展发生。

5)、不断重复以上步骤，直到 qnew 与 qgoal 的距离小于给定的阈值则可以认为拓展成功。

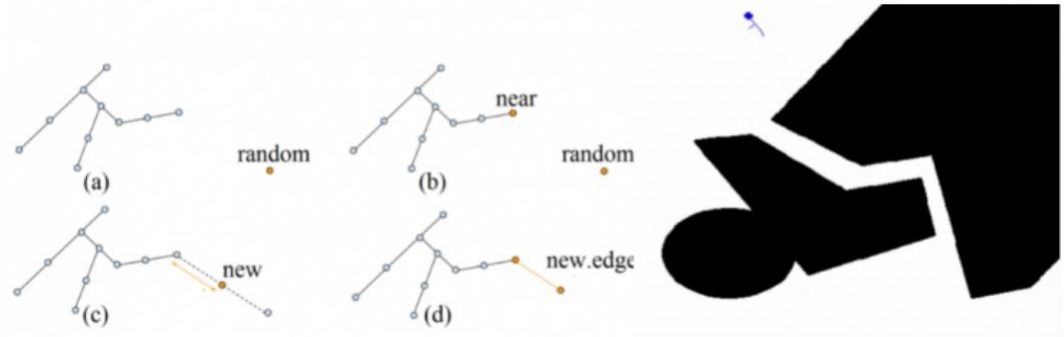
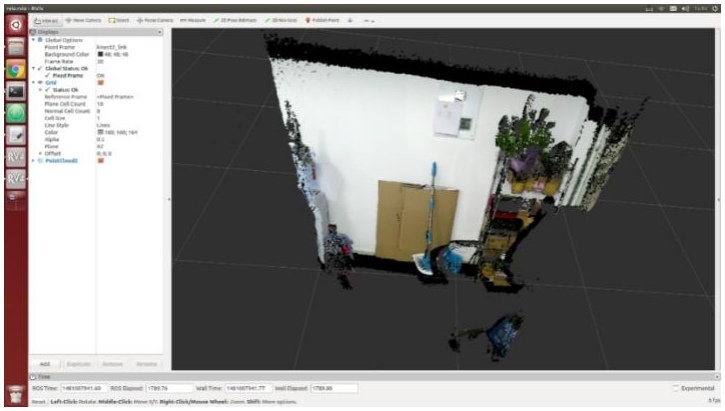


图 6.4.1 运动规划树

• 碰撞检测

在机械臂运动中实时检测环境中的障碍物。确保机械臂运动过程中不碰到相关的障碍物。通过 Rviz 的可视化中加入点云等实时采集的外部环境信息，让机械臂的规划和移动可以在 Rviz 中实时显示出来。如图所示：



（四）ROS 基础学习

ROS（机器人操作系统）

关于 ROS 的简单介绍：

ROS 是一个为机器人软件开发者设计的程序库和工具集合。提供硬件抽象、设备驱动、函数库、可视化工具、消息传递和软件包管理等功能。

1.安装和配置 ROS 环境

安装 ROS

按照 ROS 安装指南完成 ROS 的安装。

注意：使用软件包管理器（如“apt”）安装的 ROS 软件包不具备写入权限，用户也不应修改它们。

管理环境

安装 ROS 期间，需要 source 一个 setup.*sh 文件以配置环境。

确保 ROS_ROOT 和 ROS_PACKAGE_PATH 环境变量正确设置。

使用 printenv | grep ROS 检查环境变量是否设置正确。

环境变量设置文件可能来自：

软件包管理器安装的 ROS 软件包。

rosbuild workspaces 中使用的工具（如 rosws）。

编译或安装 catkin 软件包时自动生成。

2.创建 ROS 工作空间

选择 catkin。

创建和构建一个 catkin 工作空间的步骤：

bash

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

第一次运行 catkin_make 时，它会在 src 目录下创建一个 CMakeLists.txt 的链接。

Python 3 兼容性

对于 ROS Melodic 和早期版本，如果需要 Python 3 兼容性，构建 ROS 时设置：

```
bash
catkin_make -DPYTHON_EXECUTABLE=/usr/bin/python3
```

这将配置 catkin_make 使用 Python 3。

配置环境

在 devel 文件夹中，source 新生成的 setup.*sh 文件以设置环境：

```
bash
source devel/setup.bash
```

确保 ROS_PACKAGE_PATH 环境变量包含当前工作空间目录。

验证安装

使用 echo $ROS_PACKAGE_PATH 验证工作空间是否被正确添加到环境变量中。

3.ROS 文件系统导航

预备工作

确保安装了 ros-tutorials 程序包，以便使用教程中提到的工具。

文件系统概念简介

软件包（Packages）：ROS 代码的软件组织单元，可能包含程序库、可执行文件、脚本等。

Manifests (package.xml)：描述软件包的元信息，定义包之间的依赖关系。

文件系统工具

ROS 提供了专门的命令工具来简化在多个 ROS 包中的查找和导航。

使用 rospack

rospack 命令用于获取软件包的信息,特别是 find 参数用于返回软件包的路径。

用法示例：rospack find [package_name]，如 rospack find roscpp。

使用 roscd

roscd 是 rosbash 命令集的一部分，允许直接切换到软件包或软件包集的目录中。

用法示例：roscd [location-name[/subdir]]，如 roscd roscpp 将切换到 roscpp 包的目录。

roscd log

roscd log 命令用于切换到存储 ROS 日志文件的目录。

使用 rosls

rosls 命令允许直接按软件包的名称执行 ls 命令，无需输入绝对路径。

用法示例：rosls [location-name[/subdir]]，如 rosls roscpp_tutorials。

Tab 补全

ROS 工具支持 Tab 补全功能，简化命令输入过程。

4.在 ROS 中创建软件包

简介

ROS 软件包是 ROS 系统中组织代码的基本单位，可以包含节点、库、配置文件等。

创建 ROS 软件包的步骤

初始化工作空间：确保有一个 ROS 工作空间

```
bash
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

创建软件包：使用 catkin_create_pkg 命令创建新的软件包。

```
bash
cd ~/catkin_ws/src
catkin_create_pkg --build-type catkin -d <package_name> <dependencies>
```

其中<package_name>是你要创建的包的名称，<dependencies>是包的依赖项。

编辑 package.xml

package.xml 文件包含了软件包的元数据，如名称、版本、维护者、许可证等。

编写 C++节点

在 src 目录下创建你的节点源代码，例如<node_name>.cpp。

编写消息和服务

在 msg 和 srv 目录下定义自定义消息和服务。

构建软件包

返回工作空间根目录，运行 catkin_make 来构建你的软件包。

bash

cd ~/catkin_ws

catkin_make

测试软件包

使用 rosrun 运行节点，使用 rostopic 查看话题，使用 rosservice 调用服务。

使用 Python 编写 ROS 发布者和订阅者节点

编写发布者节点

预备工作：确保已安装 ros-tutorials 包。

创建 scripts 目录：在 beginner_tutorials 包中创建 scripts 目录。

下载示例脚本：获取 talker.py 示例脚本并赋予执行权限。

编辑 CMakeLists.txt：添加 catkin_install_python 调用以构建 Python 脚本。

发布者节点代码分析

导入 rospy 和 std_msgs.msg.String。

定义 talker 函数，创建发布者对象，初始化节点，设置循环频率，发布消息。

使用 rospy.loginfo 打印日志信息，pub.publish 发布消息。

编写订阅者节点

下载示例脚本：获取 listener.py 示例脚本并赋予执行权限。

编辑 CMakeLists.txt：确保 Python 脚本被包含在构建过程中。

订阅者节点代码分析

导入 rospy 和 std_msgs.msg.String。

定义 callback 函数，用于处理接收到的消息。

定义 listener 函数，初始化节点，创建订阅者对象，调用 rospy.spin 保持节点运行。

构建节点

在 catkin 工作空间中运行 catkin_make 命令构建节点。