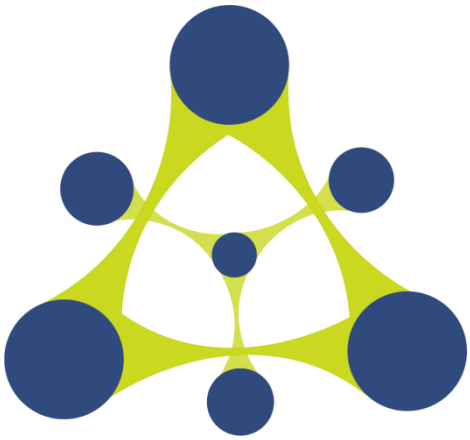


RoboWare Studio 软件使用手册

版本：1.1.0
时间：2017-12-28

目 录

1 介绍

1.1 编写目的

编写 RoboWare Studio 使用说明的目的是充分叙述 RoboWare Studio 所能实现的功能及其运行环境，以便 RoboWare Studio 用户了解本软件的使用范围和使用方法，并为 RoboWare Studio 软件的维护和更新提供必要的信息。

1.2 软件特性

RoboWare Studio 是一个 ROS 集成开发环境。它使 ROS 开发更加直观、简单，并且易于操作。可进行 ROS 工作区及包的管理，代码编辑、构建及调试。

RoboWare Studio 的主要特性有：

（1）易于安装及配置

下载后双击即可安装，RoboWare Studio 可自动检测并加载 ROS 环境，无需额外配置。这种“开箱即用”的特性能够帮助开发者迅速上手。

（2）辅助 ROS 开发，兼容 indigo/jade/kinetic 版本

RoboWare Studio 专为 ROS (indigo/jade/kinetic)设计，以图形化的方式进行 ROS 工作区及包的创建、源码添加、message/service/action 文件创建、显示包及节点列表。可实现 CMakelists.txt 文件和 package.xml 文件的自动更新。

（3）友好的编码体验

提供现代 IDE 的重要特性，包括语法高亮、代码补全、定义跳转、查看定义、错误诊断与显示等。支持集成终端功能，可在 IDE 界面同时打开多个终端窗口。支持 Vim 编辑模式。

（4）C++和 Python 代码调试

提供 Release、Debug 及 Isolated 编译选项。以界面交互的方式调试 C++和 Python 代码，可设置断点、显示调用堆栈、单步运行，并支持交互式终端。可在用户界面展示 ROS 包和节点列表。

（5）远程部署及调试

可将本地代码部署到远程机器上，远程机器可以是 X86 架构或 ARM 架构。可在本地机器实现远程代码的部署、构建和实时调试。

（6）内置 Git 功能

Git 使用更加简单。可在编辑器界面进行差异比对、文件暂存、修改提交等操作。可对任意 Git 服务仓库进行推送、拉取操作。

（7）遵循 ROS 规范

从代码创建、消息定义，到文件存储路径的创建及选择等，RoboWare Studio 会引导开发者进行符合 ROS 规范的操作，协助开发者编写高质量、符合规范的 ROS 包。

1.3 软件更新说明

当前版本：1.1.0。

（1）增加 catkin_tools 构建选项；

（2）为保证 catkin_tools 功能正常使用，对于之前版本的 ROS 工作区，需先删除工作区根目录.vscode 文件夹下的 tasks.json 文件，然后重新打开工作区；

（3）修复已知 Bug。

历史版本：1.0.2。

（1）为保证编译时错误定位功能正常使用，对于之前版本的 ROS 工作区，需先删除工作区根目录.vscode 文件夹下的 tasks.json 文件，然后重新打开工作区；

（2）优化远程同步速度；

（3）增加 CMakeLists.txt 错误定位功能；

（4）修复 package.xml 包依赖自动修改问题。

历史版本：1.0.1。

（1）创建包时可在包名后添加依赖；

（2）修复编辑 catkin 包的问题。

历史版本：1.0.0。

（1）集成 RoboWare Designer；

（2）增加 clang-format 代码格式化功能；

（3）增加 catkin 依赖包编辑功能。

历史版本：0.7.2。

（1）增加 rosbag 循环播放功能；

（2）修复远程调试不正常退出的问题；

（3）修复 Kinetic 版本下编译报错的问题。

历史版本：0.7.1。

（1）初次启动默认为非 Vim 编辑模式；

（2）增加远程编辑.bashrc 文件功能；

（3）增加启动 rqt 的快捷菜单；

（4）更新 Python 语法支持插件；

（5）更新调试插件；

（6）根据构建选项自动切换调试模式。

历史版本：0.7.0。

（1）采用全新的 C++语法补全方案；

（2）增加远程启动 roscore 命令的快捷菜单；

（3）修复已知 Bug。

历史版本：0.6.0。

（1）rosbag：record 和 play 功能；

（2）采用 vscode 1.12.2 内核；

（3）改进 C++语法提示功能；

（4）修复已知 Bug。

历史版本：0.5.1。

（1）增加 Vim 编辑模式；

（2）向导创建 Node 和 C++类；

（3）远程和本地 Build 目录结构同步；

（4）远程清理远程端编译结果；
（5）修复已知 Bug。
历史版本：0.5.0。
（1）优化框架，安装包体积缩小；
（2）采用 VS Code 1.10.1 内核；
（3）ROS 包管理工具移至左侧选项卡，实现 ROS 包卸载功能；
（4）可在文件列表中标记非活动 ROS 包功能；
（5）增加 C++和 Python 代码片段功能；
（6）修复已知 Bug。
历史版本：0.4.2。
（1）修改、删除源文件时实现自动修改 CMakeLists.txt；
（2）指定一个或多个包，进行构建；
（3）指定一个或多个包，进行远程部署、远程构建；
（4）实现远程运行 launch 文件；
（5）增加.bashrc 菜单，一键打开编辑.bashrc 文件；
（6）批量清理生成的 ROS 节点；
（7）修复已知 Bug。
历史版本：0.4.1。
（1）增加 RoboWare Studio 32 位版；
（2）在 ROS 包管理器中增加 ROS 包安装功能；
（3）增加基于 clang 的代码补全机制；
（4）采用 VS Code 1.8.1 内核；
（5）修复已知 Bug。
历史版本：0.4.0。
（1）增加 ROS 命令图形化列表功能；
（2）增加 ROS 包管理器；
（3）在文件-首选项菜单中增加语言设置选项；
（4）修复已知 Bug。
历史版本：0.3.1。
（1）采用 VS Code 1.7.2 内核；
（2）增加资源管理器视图上的构建选项列表；
（3）增加 isolated 构建方式；
（4）实现远程部署、构建及调试功能；
（5）增加 CMakeLists.txt 的语法支持；
（6）实现在集成终端中运行 ROS 命令；
（7）增加 rqt_reconfigure 和 rqt_graph 命令的快捷菜单；
（8）实现图形化用户配置界面；
（9）修复已知 Bug。

2 软件安装

2.1 准备

安装前，请查看系统环境并确认：

（1）操作系统为 Ubuntu。

（2）已完成 ROS 的安装配置。ROS 安装步骤可参照官方网站：

http://wiki.ros.org/indigo/Installation/Ubuntu

（3）可使用 catkin_make 构建 ROS 包。（若无法构建，您可能需要运行

$ sudo apt-get install build-essential

来安装基本构建工具。）

（4）为支持 Python 相关功能，需要安装 pylint。

$ sudo apt-get install python-pip

$ sudo python -m pip install pylint

（5）为支持 clang-format 相关功能，需要安装 clang-format-3.8 或更高版本。

$ sudo apt-get install clang-format-3.8

2.2 安装

下载 RoboWare Studio 最新版，双击下载的.deb 文件即可完成安装。或在终端执行以下命令进行安装：

$ cd /path/to/deb/file/

$ sudo dpkg -i roboware-studio_<version>_<architecture>.deb

其中，<version>表示软件版本号，<architecture>表示机器的处理器架构（amd64 为 64 位版本，i386 为 32 位版本）。将<version>和<architecture>替换为当前文件信息即可（小技巧：可在输入“sudo dpkg -i ”后按 Tab 键自动补全文件名）。安装后，RoboWare Studio 会自动检测并加载 ROS 环境，无需额外配置。

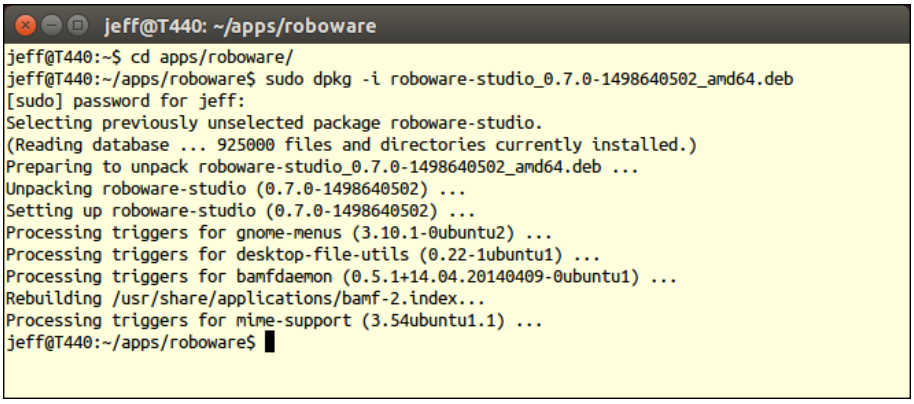


图 2-1 RoboWare Studio 终端方式安装

2.3 升级

下载最新的 RoboWare Studio 安装包 deb 文件，参照安装步骤直接安装即可，旧版本会被自动覆盖。

2.4 卸载

打开任一终端，执行以下指令卸载 RoboWare Studio：

$ sudo apt-get remove roboware-studio

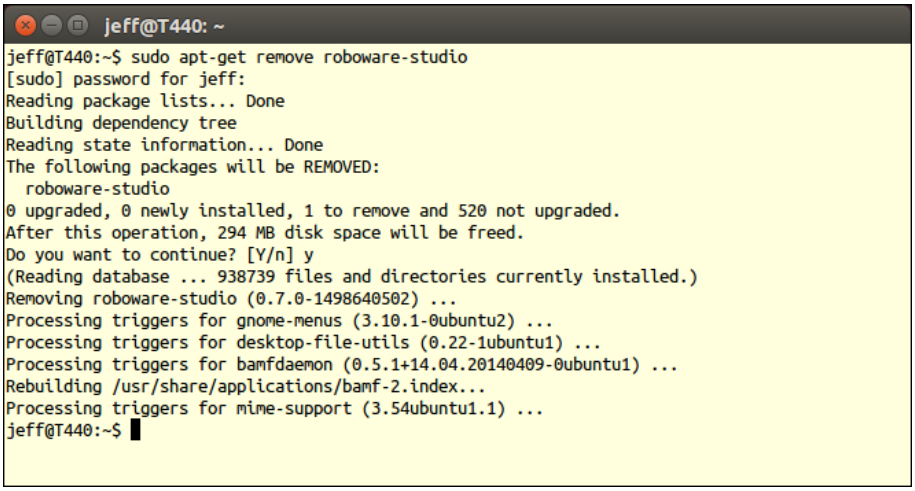


图 2-2 RoboWare Studio 卸载

2.5 启动

方式一（推荐）：点击屏幕左上角的 Ubuntu 图标，打开 Dash，搜索“roboware-studio”，单击启动。

方式二：通过终端启动，打开任一终端，执行：

$ roboware-studio

3 软件使用教程

3.1 本地开发模式

3.1.1 创建工作区

在欢迎界面，点击“新建工作区”按钮（或选择“文件 - 新建工作区”），选择路径并填写工作区名称，如“catkin_ws”，则会创建一个名为“catkin_ws”工作区，并显示在资源管理器窗口。

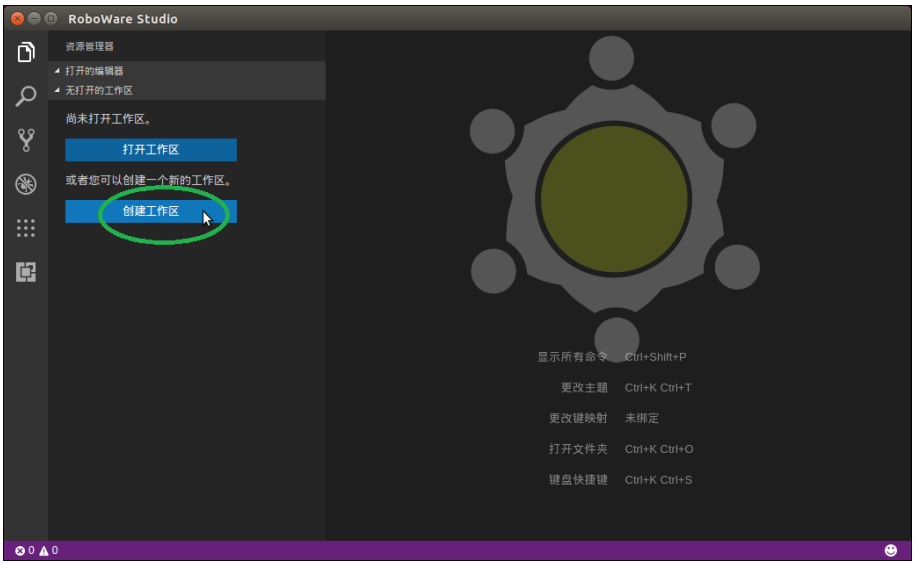


图 3-1 RoboWare Studio 欢迎界面

图 3-2 创建完成 catkin_ws 工作区

3.1.2 打开/关闭工作区

在欢迎界面，点击“打开工作区”按钮（或选择“文件 - 打开工作区”），选择需要打开的 ROS 工作区，打开后即可显示在资源管理器窗口。

选择“文件 - 关闭工作区”，RoboWare Studio 会关闭当前的工作区并返回欢迎界面。

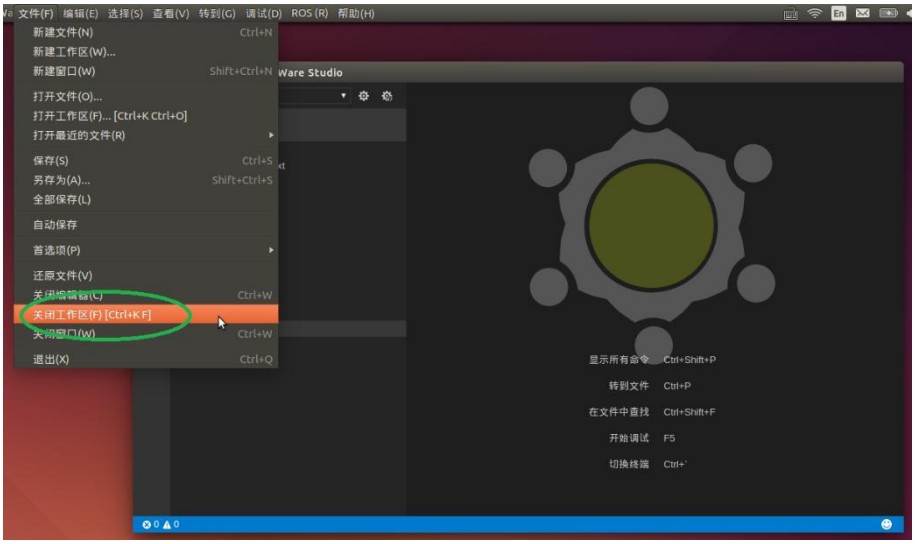


图 3-3 关闭 ROS 工作区

3.1.3 创建 ROS 包

右键点击 ROS 工作区下的“src”，选择“新建 ROS 包”，输入包名称及其依赖包的名称，如：

“my_package roscpp std_msgs”

回车后，会创建名为“my_package”、以“roscpp”和“std_msgs”为依赖的 ROS 包。

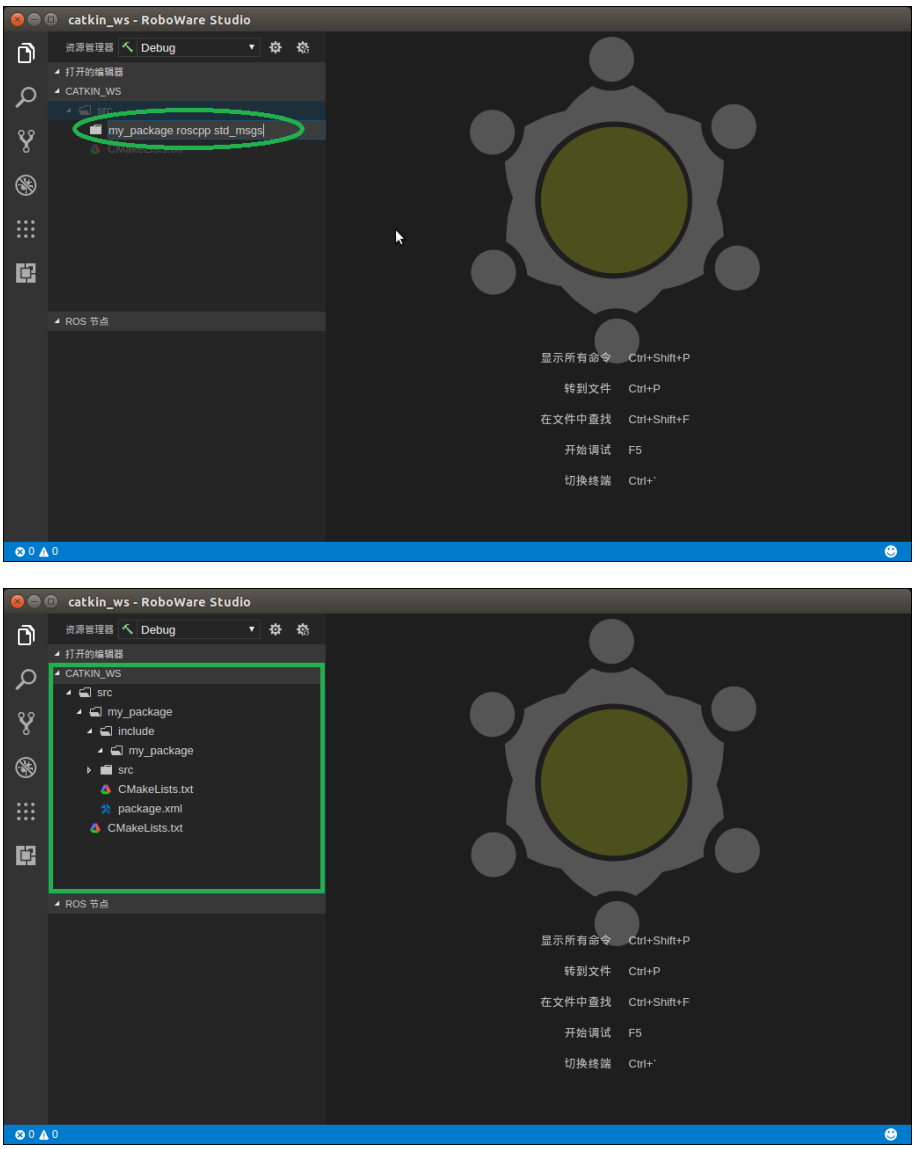


图 3-4 创建名为“my_package”，依赖为“roscpp”、“std_msgs”的 ROS 包

3.1.4 添加新的动态链接库或可执行文件（ROS 节点）

右键点击包名文件夹（如“my_package”），选择“新建 Src 文件夹”，会自动创建 ROS 标准的 src 源码目录，其它必要的目录也可通过此右键菜单来创建。

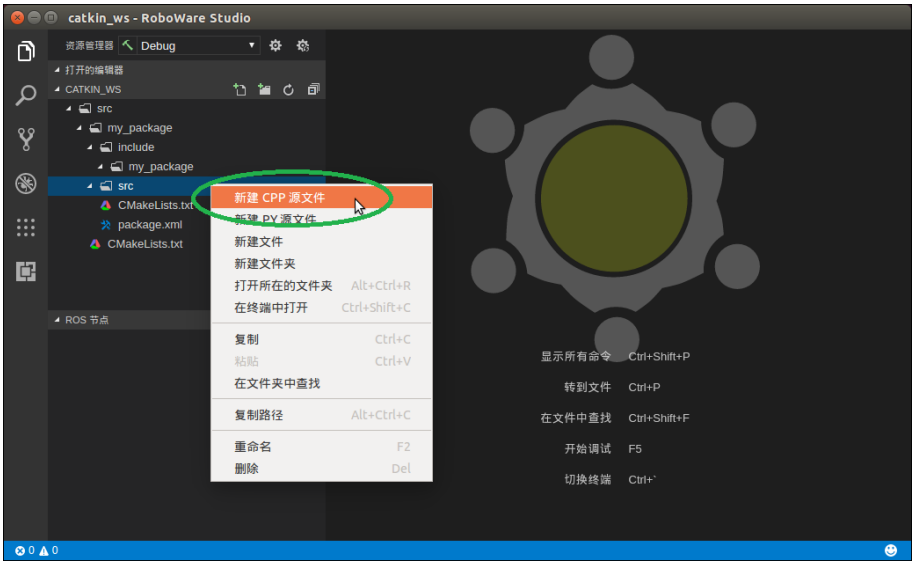
右键点击 ROS 包目录下的“src”，选择“新建 CPP 源文件”，输入文件名后，点击回车键，会弹出以下列表：

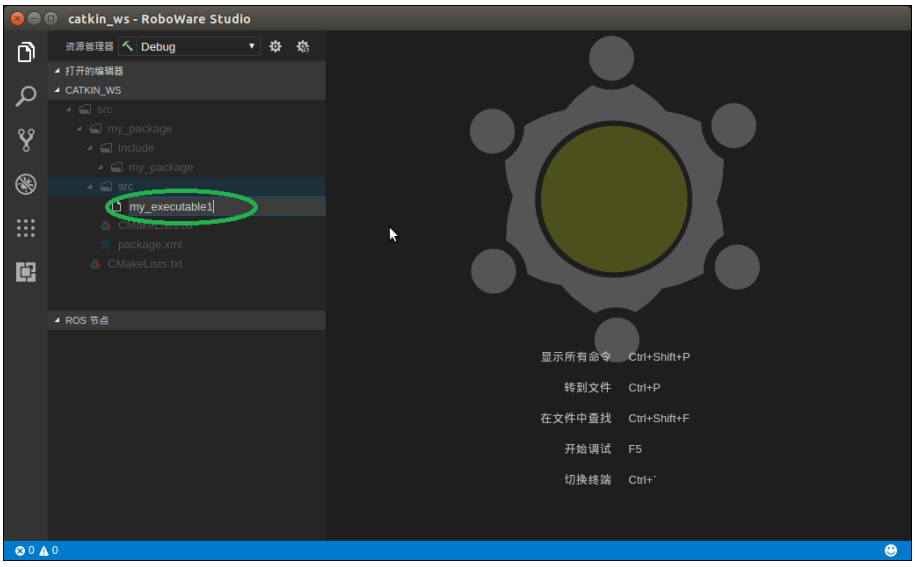
-加入到新的库文件中
-加入到新的可执行文件中

在列表中选择类型，则会创建一个与 CPP 文件同名的动态链接库或可执行文件（ROS 节点），此时 CMakeLists.txt 文件会自动更新。

同理，右键点击 ROS 包目录下的“include/包名”，选择“新建头文件”，也可通过同样方式进行添加。


catkin_ws - RoboWare Studio
Debug
设置为活动状态
激活所有 ROS 包
新建 Include 文件夹
新建 Src 文件夹
新建 Msg 文件夹
新建 Srv 文件夹
新建 Action 文件夹
新建 Launch 文件夹
新建 Cfg 文件夹
新建 C++ ROS 节点
新建 Python ROS 节点
新建 C++ 类
编辑依赖的 ROS 包列表
新建文件
新建文件夹
打开所在的文件夹 Alt+Ctrl+R
在终端中打开 Ctrl+Shift+C
复制 Ctrl+C
粘贴 Ctrl+V
在文件夹中查找
显示所有命令 Ctrl+Shift+P
转到文件 Ctrl+P
在文件中查找 Ctrl+Shift+F
开始调试 F5
切换终端 Ctrl+`


catkin_ws - RoboWare Studio
Debug
CATKIN_WS
src
my_package
include
my_package
src
CMakeLists.txt
package.xml
CMakeLists.txt
新建 CPP 源文件
新建 PY 源文件
新建文件
新建文件夹
打开所在的文件夹 Alt+Ctrl+R
在终端中打开 Ctrl+Shift+C
复制 Ctrl+C
粘贴 Ctrl+V
在文件夹中查找
复制路径 Alt+Ctrl+C
重命名 F2
删除 Del
显示所有命令 Ctrl+Shift+P
转到文件 Ctrl+P
在文件中查找 Ctrl+Shift+F
开始调试 F5
切换终端 Ctrl+`


catkin_ws - RoboWare Studio
Debug
CATKIN_WS
my_executable1
package.xml
CMakeLists.txt
显示所有命令 Ctrl+Shift+P
转到文件 Ctrl+P
在文件中查找 Ctrl+Shift+F
开始调试 F5
切换终端 Ctrl+`

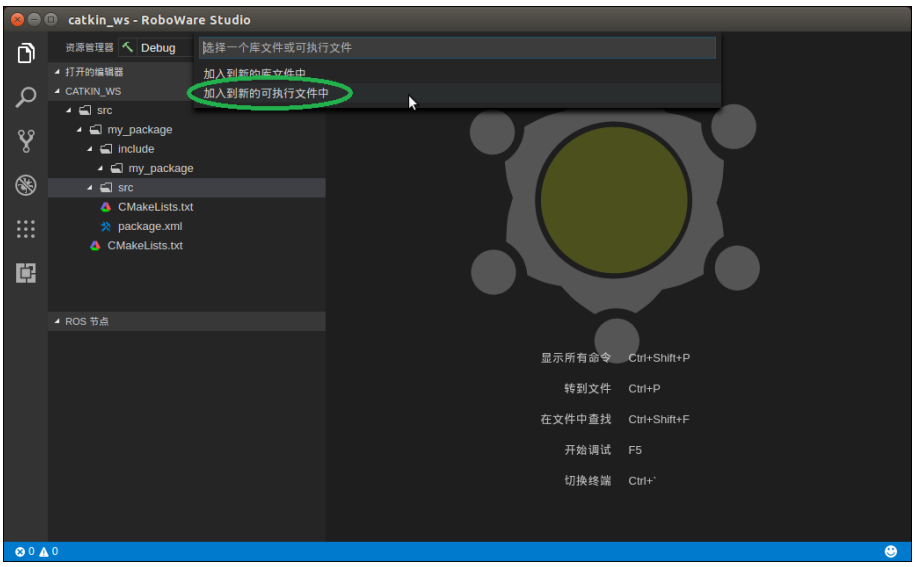


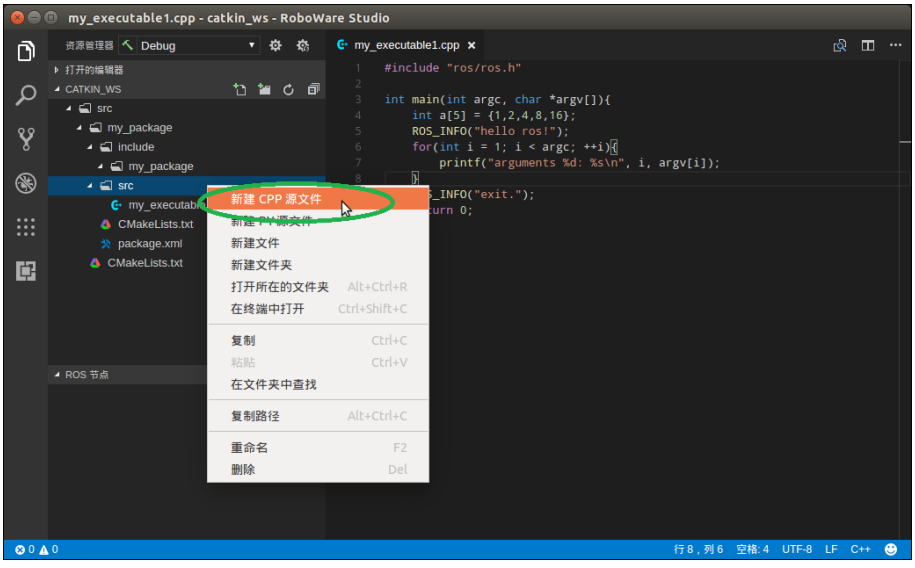
图 3-5 新建 CPP 源文件，并添加为可执行文件（ROS 节点），可执行文件名默认与 CPP 文件同名

3.1.5 添加 C++源代码到动态链接库或可执行文件（ROS 节点）

右键点击 ROS 包目录下的“src”，选择“新建 CPP 源文件”，输入文件名（如“animal”）后，点击回车键，会弹出以下列表：

-my_library1
-my_library2
-my_executable1
-加入到新的库文件中
-加入到新的可执行文件中

其中 my_library1、my_library2、my_executable1 为已建立的库和可执行文件（节点），以列表的形式列出。选择对应的条目（如 my_executable1），CPP 文件会添加到 my_executable1 可执行文件中。此时 CMakeLists.txt 文件会自动更新。



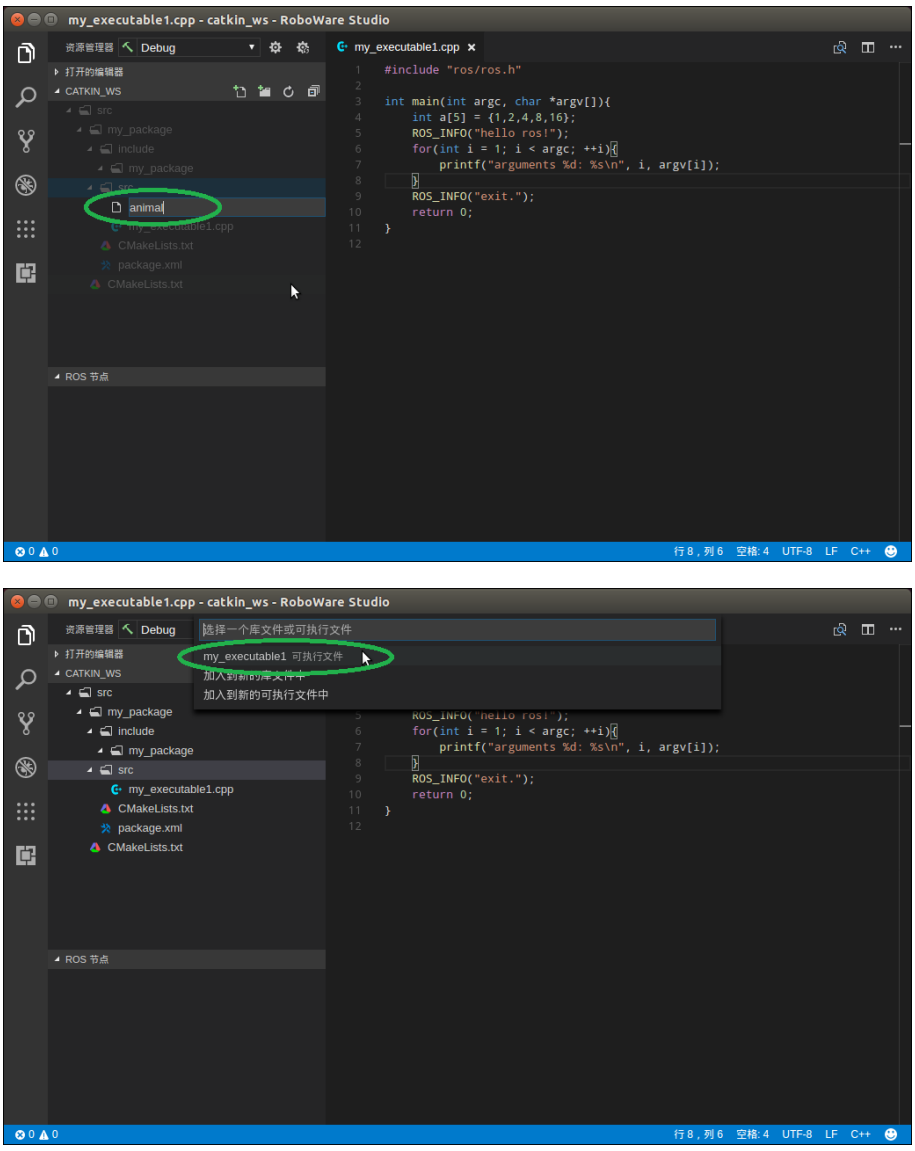


图 3-6 新建名为“animal”的 CPP 文件，并添加到 “my_executable1” 可执行文件中

添加完成后，包的 CMakeLists.txt 文件内容会自动更新，如下所示。

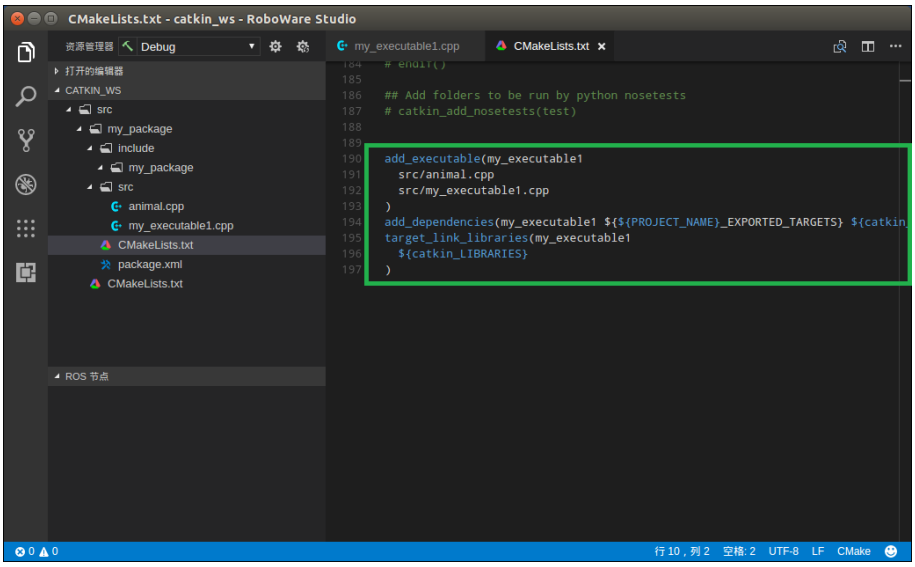


图 3-7 CMakeLists.txt 文件自动更新

3.1.6 编辑 catkin ROS 依赖包

右键点击包名文件夹（如“my_package”），选择“编辑依赖的 ROS 包列表”，加入新增的依赖包名称，如：

“std_msgs”

回车后，会自动修改 CMakeLists.txt 的依赖包列表，如依赖多个 ROS 包的时候需要用空格把每个依赖包隔开。

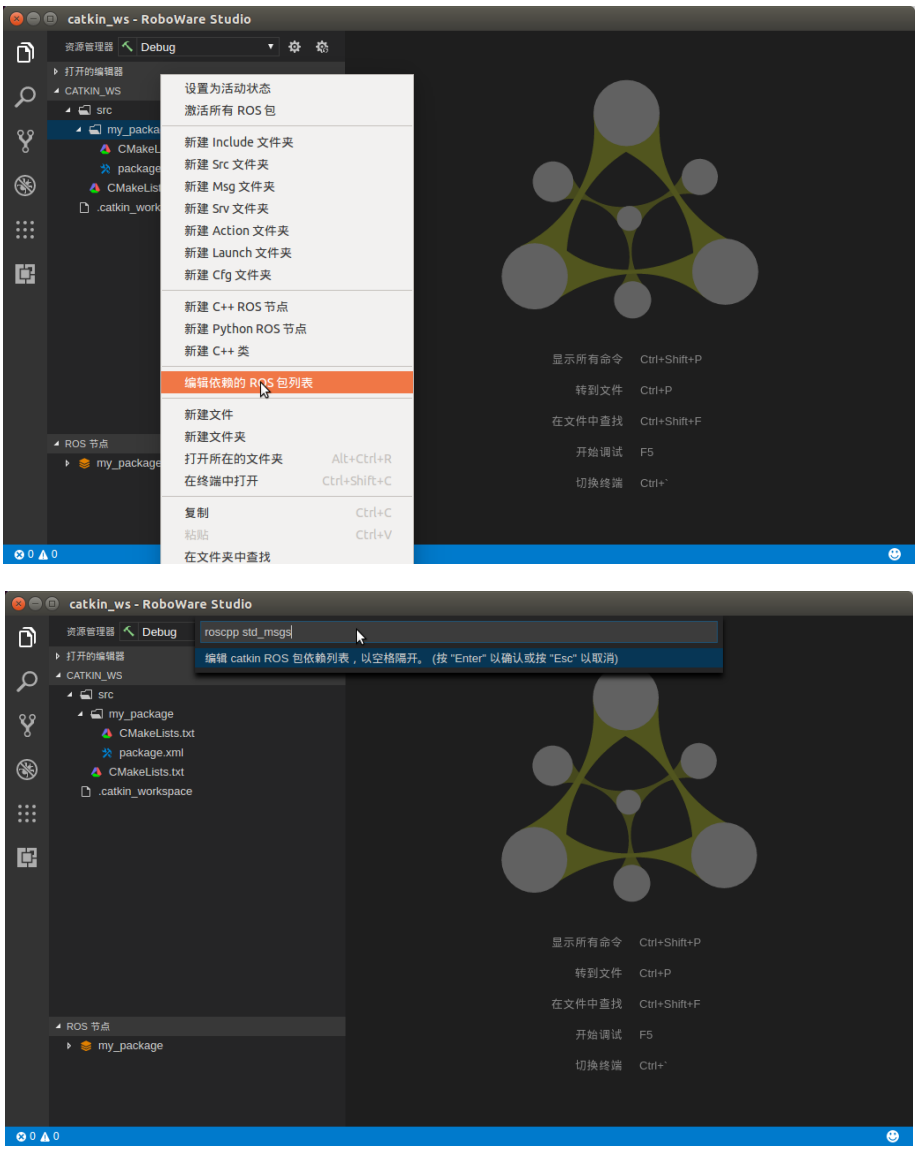


图 3-8 编辑 catkin ROS 依赖包

3.1.7 添加 message/service/action

右键点击包名文件夹（如“my_package”），选择“新建 Msg 文件夹”、“新建 Srv 文件夹”、“新建 Action 文件夹”可分别创建 message、service、action 文件夹。右键点击相应文件夹即可添加 message、service、action 文件。此时 CMakeLists.txt 文件会自动更新。

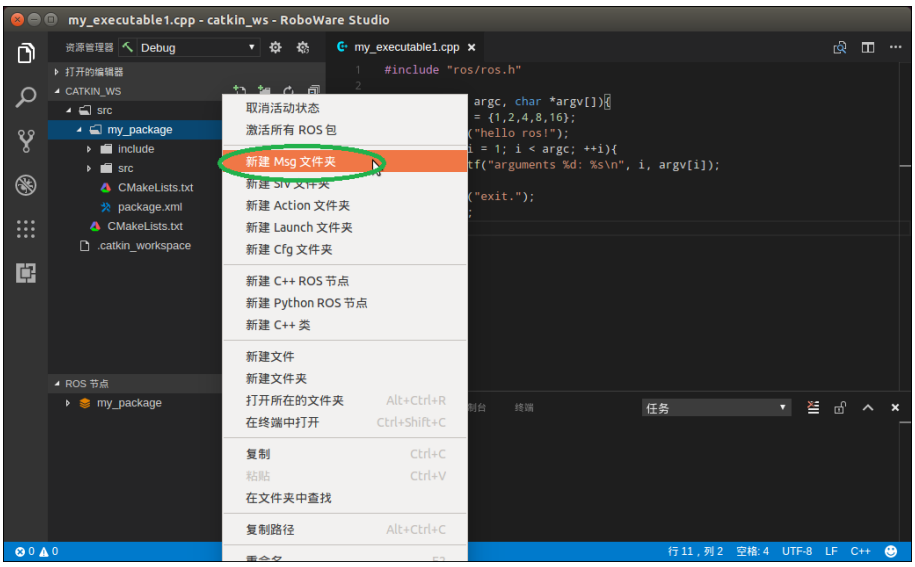


图 3-9 创建“msg”文件夹

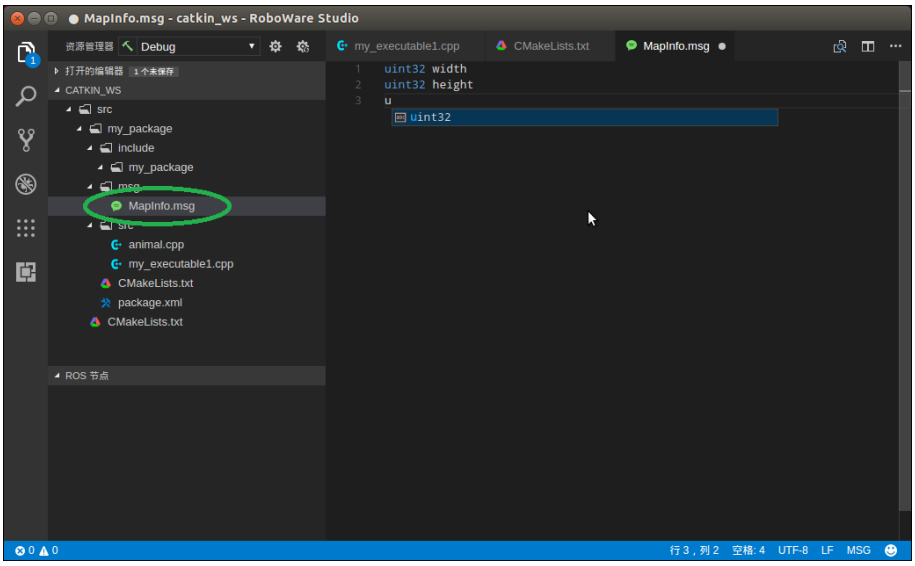
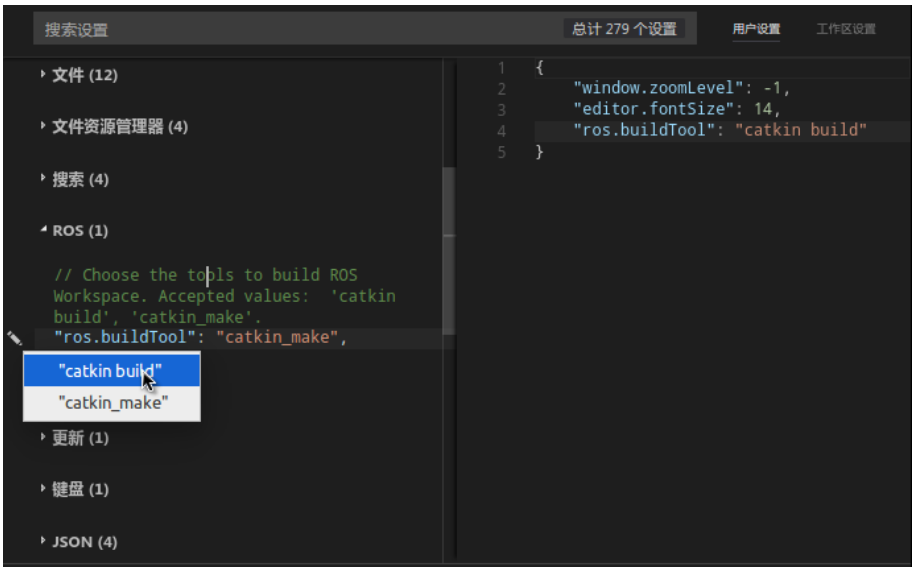


图 3-10 创建名为“MapInfo”的 message 文件

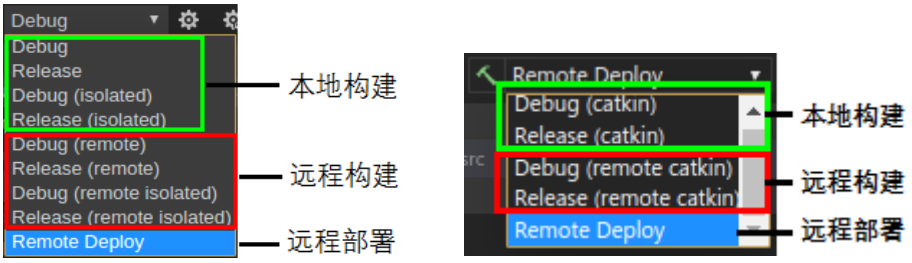
3.1.8 构建工作区

RoboWare Studio 支持 catkin_make 构建工具和 catkin_tools 构建工具。

选择菜单“文件 – 首选项 – 设置”可打开设置界面，点击“ROS - ros.buildTool”标签左侧的编辑标志，即可选择构建工具。



catkin make 与 catkin_tools 构建工具选择



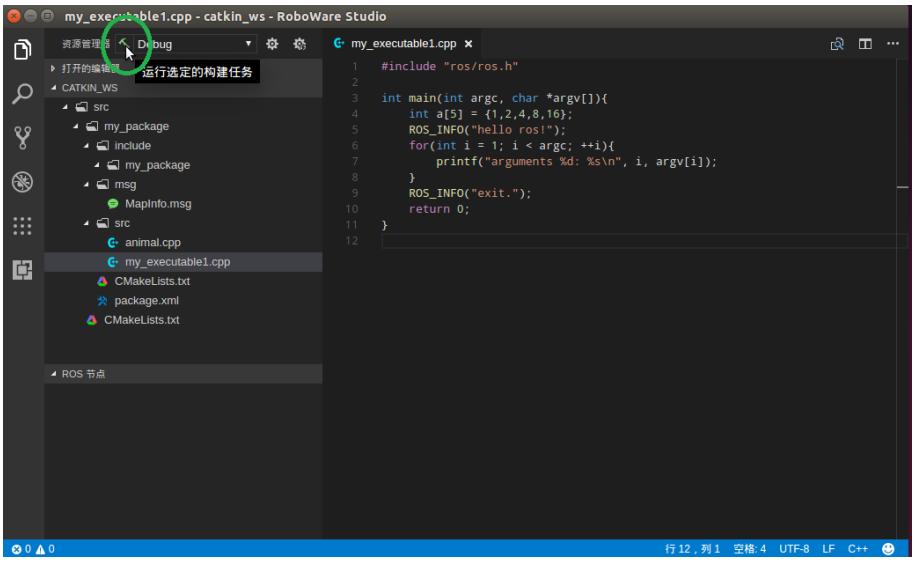
catkin make　　catkin_tools

图 3-11 构建选项

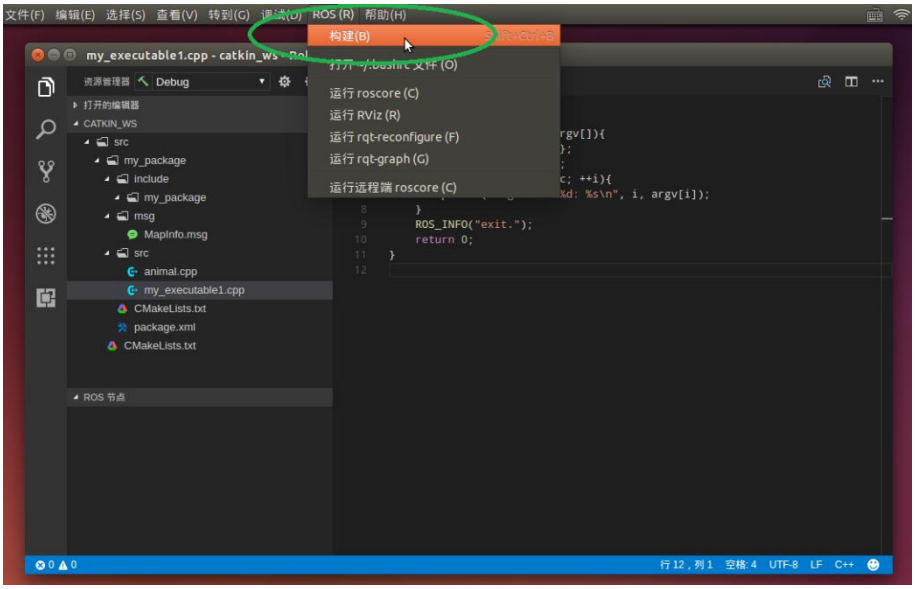
其中，Debug 和 Release 选项分别表示构建调试版和发布版，默认构建方式为本地构建。catkin make 方式下，带有“isolated”的选项表示利用“catkin_make_isolated”命令进行构建。带有“remote”的选项表示进行远程构建。“Remote Deploy”选项表示部署本地代码到远程计算机。关于远程开发的具体步骤会在下一节“远程开发模式”介绍，在此以本地构建为例进行说明。

完成构建选项选择后，点击配置列表左侧的构建按钮，或选择“ROS”-“构建”即可构建对应版本的 ROS 包。构建完成后，资源管理器窗口下方的“ROS 节点”子窗口会显示当前工作区下所有的 ROS 包及节点列表。

选择“查看 - 输出”可打开“输出”窗口，显示构建输出结果。若构建过程中出现错误，按住“CTRL”键并点击错误提示，即可跳转到源代码对应位置。



点击构建按钮构建



菜单选择 “ROS - 构建”

图 3-12 构建 ROS 包

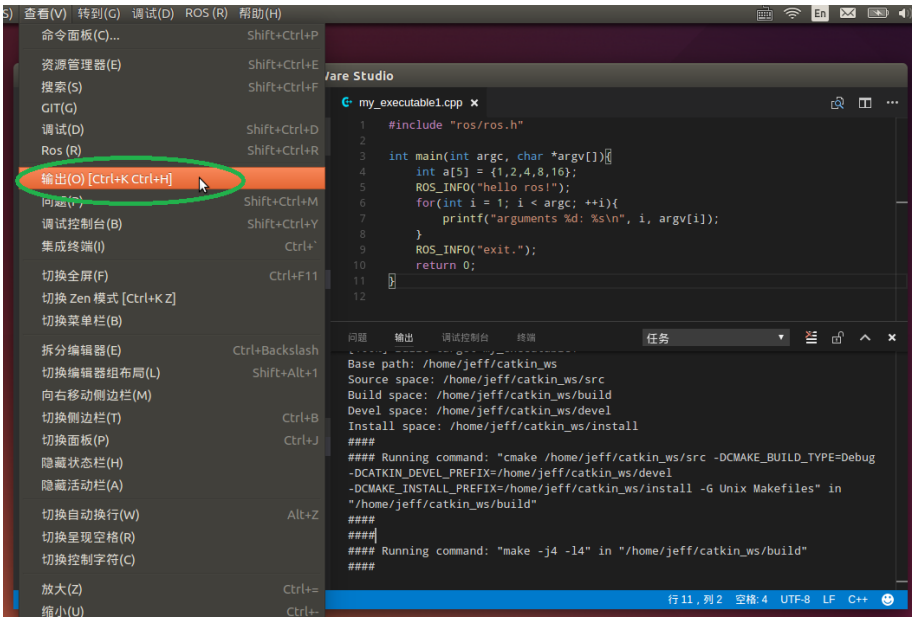
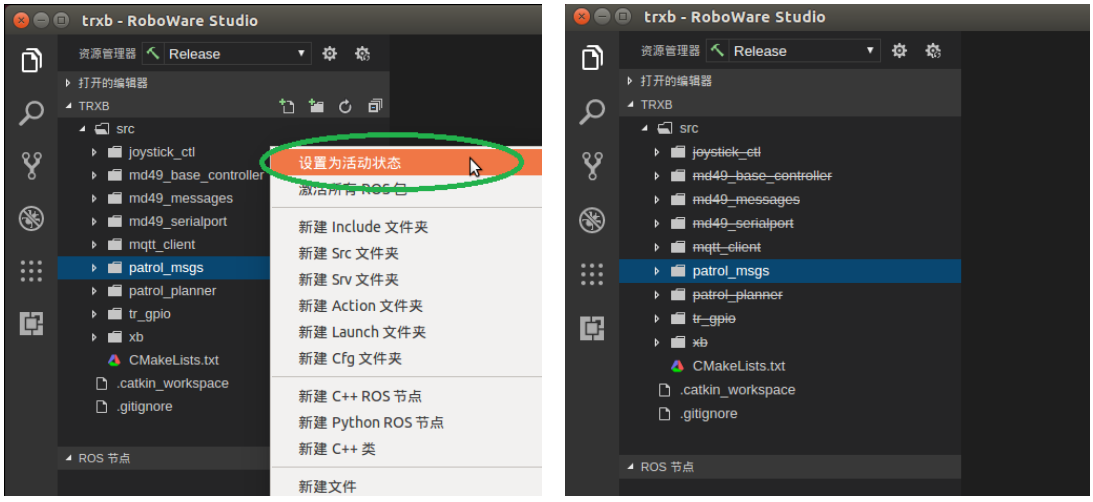


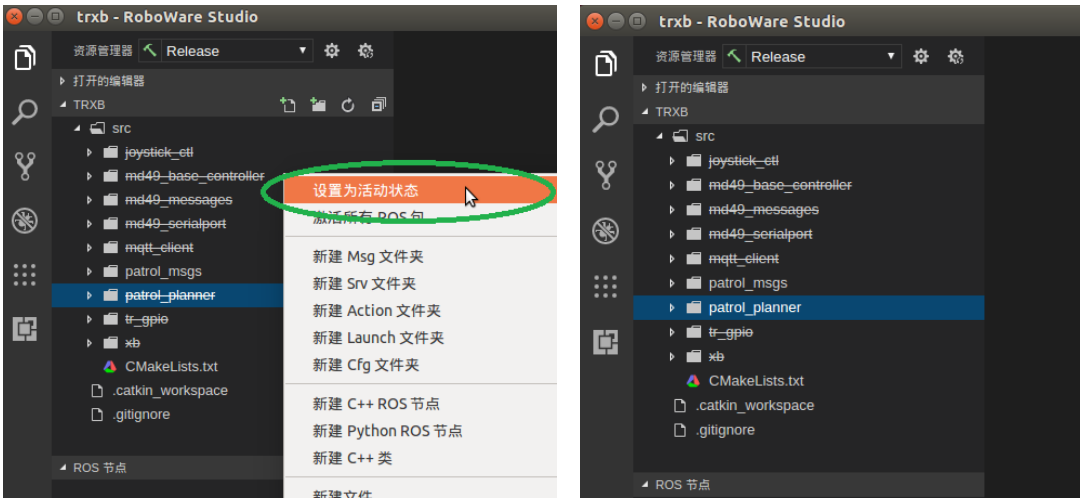
图 3-13 查看构建输出结果

3.1.9 构建工作区下的一个或多个包

默认情况下，点击“构建”按钮会构建当前工作区下的所有包。如果只想构建其中的一个或多个包，可右键点击包名，将其设置为活动状态。可同时将一个或多个包设置为活动状态。此时，不被编译的包即称为“非活动包”，在目录列表中将会以删除线标记出来。点击“构建”按钮，RoboWare Studio 只会对处于活动状态的包进行构建。



将 patrol_msgs 包设置为活动状态



将 patrol_planner 包设置为活动状态

图 3-14 ROS 包的活动状态设置

将 patrol_msgs 包和 patrol_planner 包设置为活动状态后，再点击“构建”按钮，则只会对 patrol_msgs、patrol_planner 两个包进行构建。

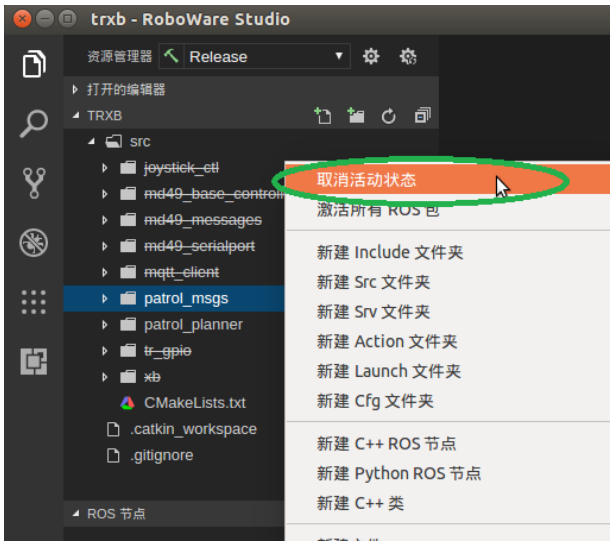


图 3-15 取消 patrol_msgs 包的活动状态

取消 patrol_msgs 包的活动状态后，此时再点击“构建”按钮，则只会对处于活动状态的 patrol_planner 包进行构建。

右键点击任一包名，选择“激活所有 ROS 包”则会使当前工作区的所有包处于活动状态，点击构建时则会对所有包进行构建。

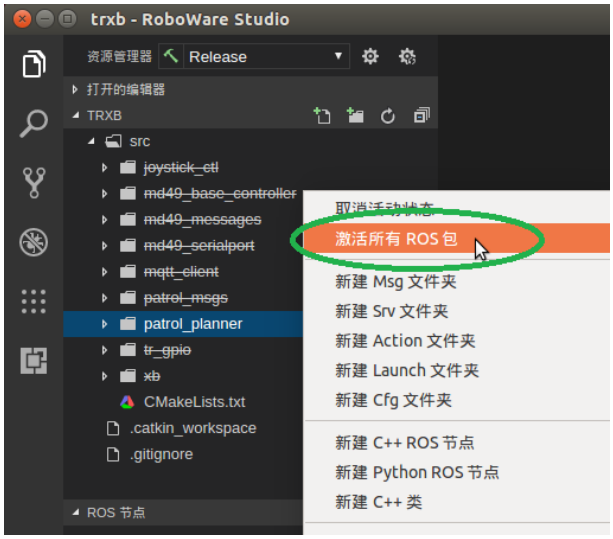


图 3-16 激活当前工作区所有包

3.1.10 清理构建结果

构建完成后，资源管理器窗口下方的“ROS 节点”子窗口会显示当前工作区下所有的 ROS 包及节点列表。

点击“ROS 节点”子窗口上的“清理”按钮，则会对构建结果进行清理。

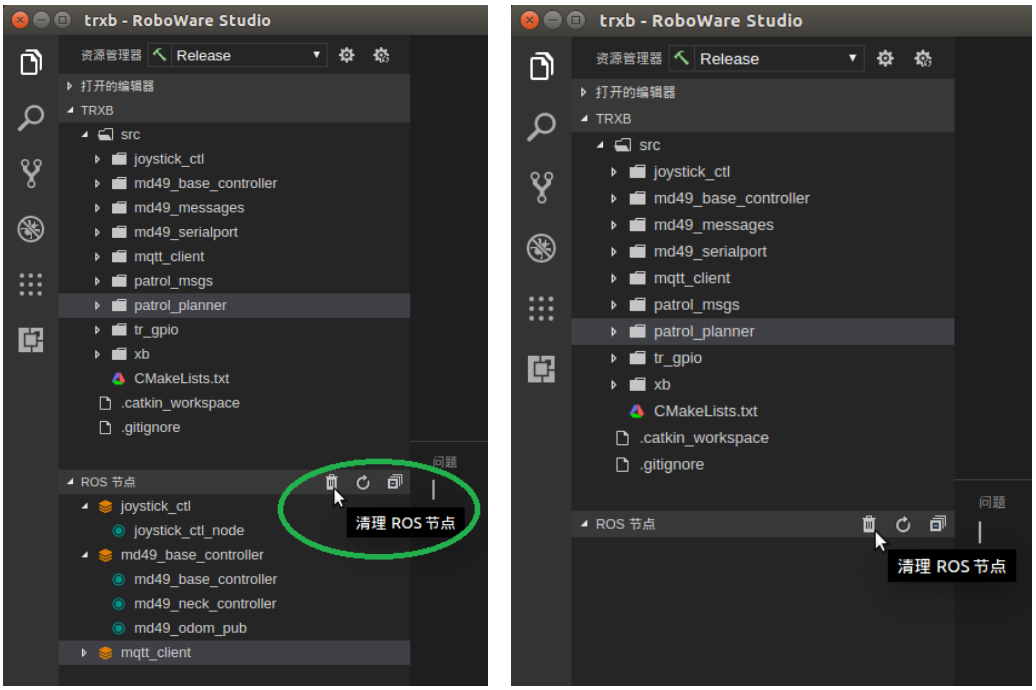


图 3-17 清理构建结果

3.1.11 集成终端使用

选择“查看 - 集成终端”选项，可在编辑窗口下方打开集成终端窗口。集成终端默认打开路径为当前 ROS 工作区根目录。可在集成终端中执行任意命令行指令。

可以点击集成终端窗口右上方的“+”按钮打开新的集成终端，并在下拉列表中进入对应的集成终端。

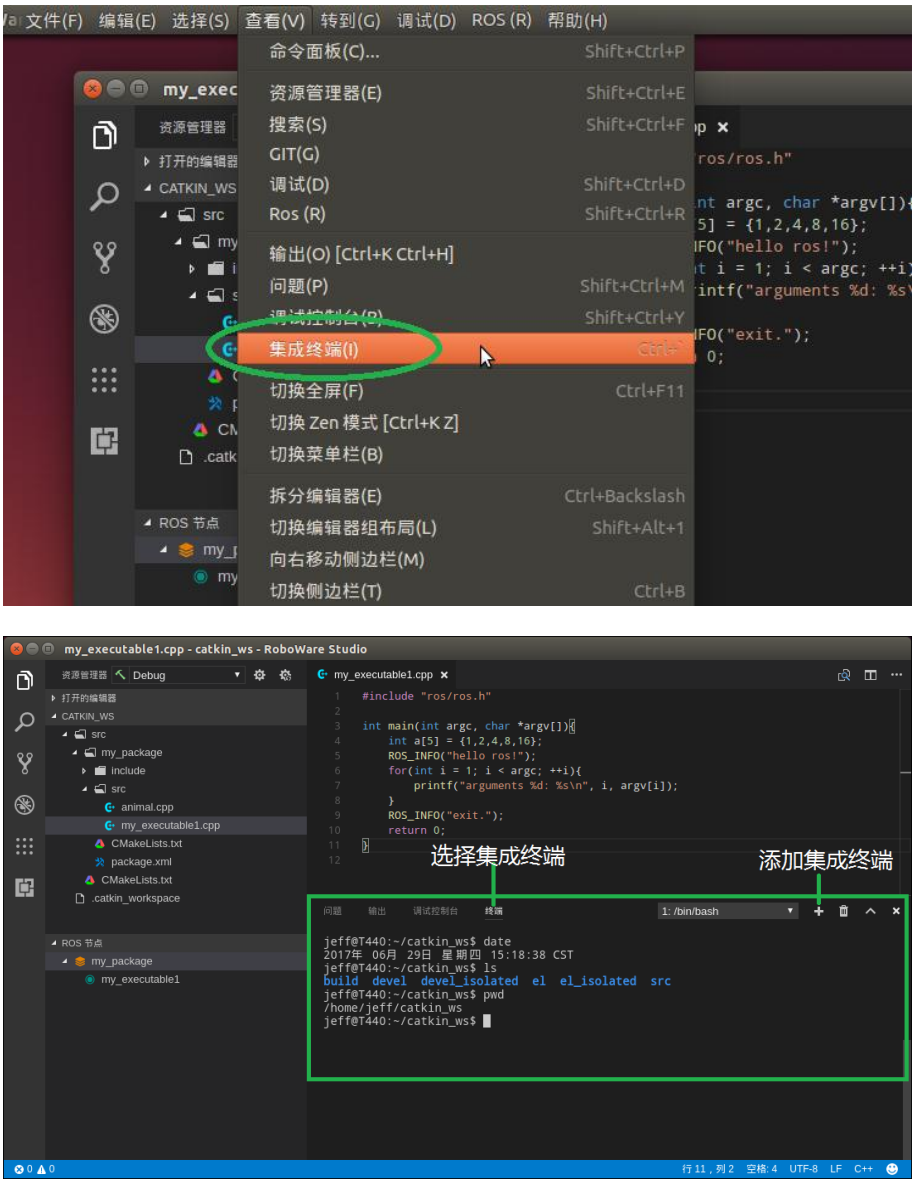


图 3-18 打开“集成终端”窗口，并在集成终端中运行命令行指令

3.1.12 调试 C++源代码

在资源管理器视图下，将构建选项选择为“Debug”模式，点击构建按钮进行构建（或选择“ROS”–“构建”）。

这时，点击左侧侧边栏的“调试”图标进入调试视图，可以看到调试器选项已经自动设置为“C++”。

点击左侧侧边栏的“资源管理器”图标进入资源管理器视图，在左下方的“ROS 节点”子窗口中，选择 ROS 包下的某个节点，右侧会显示调试界面。点击“启动调试”按钮即可进行调试。

点击“配置参数”按钮可设置运行输入参数。点击“选择参数”按钮可在历史列表中选择运行输入参数。

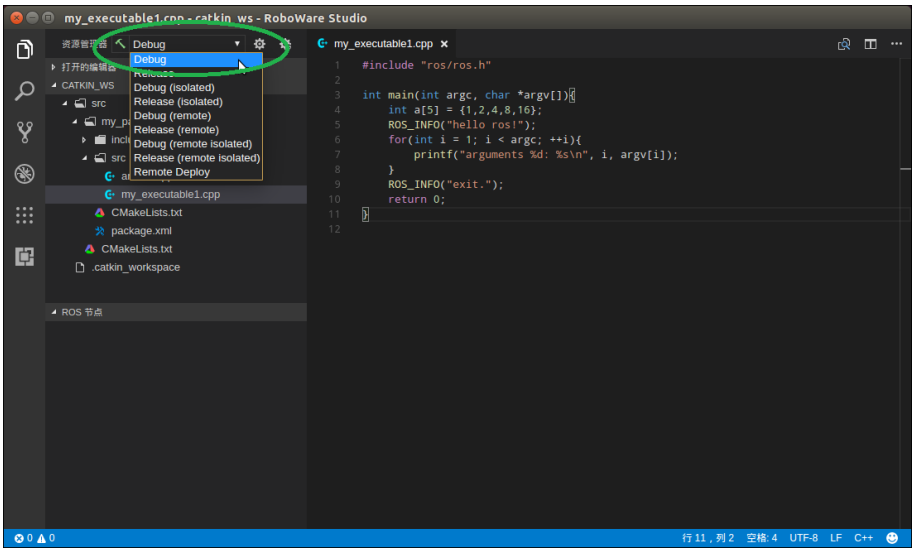


图 3-19 在资源管理器视图下，将构建选项选择为 “Debug” 并构建

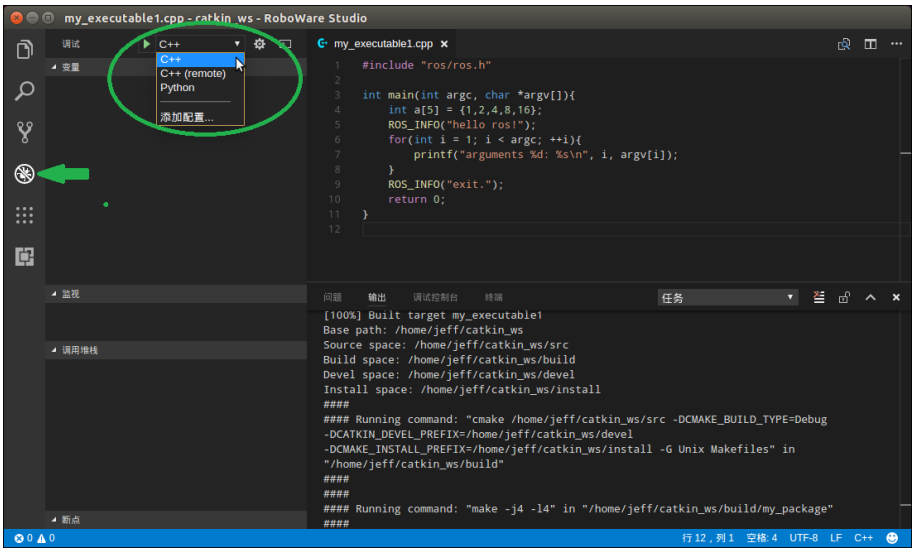


图 3-20 在调试视图下，调试器自动设置为 “C++”

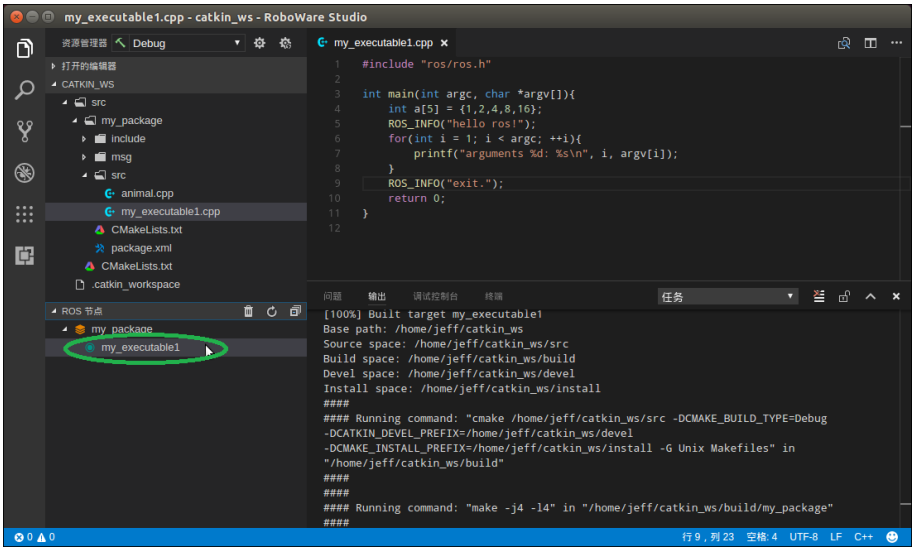


图 3-21 在“ROS 节点”子窗口中，选择 ROS 包下的某个节点

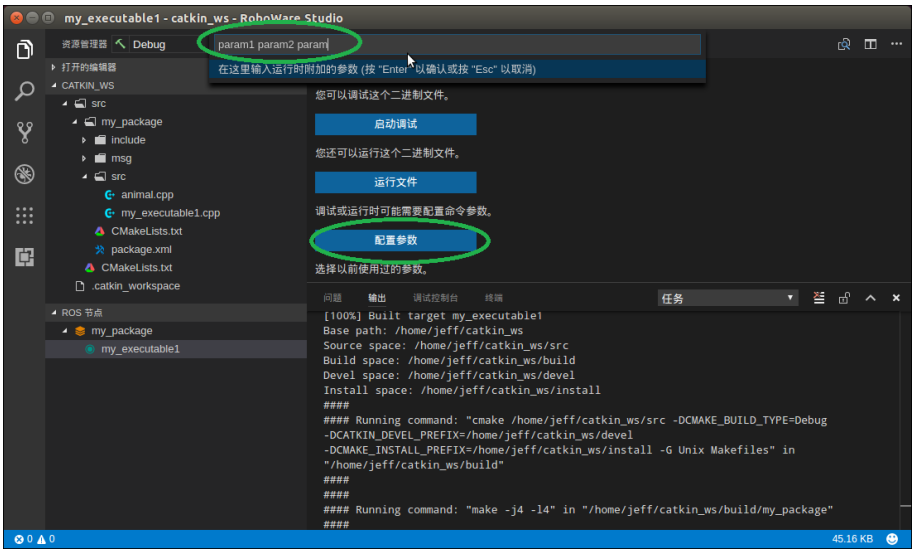
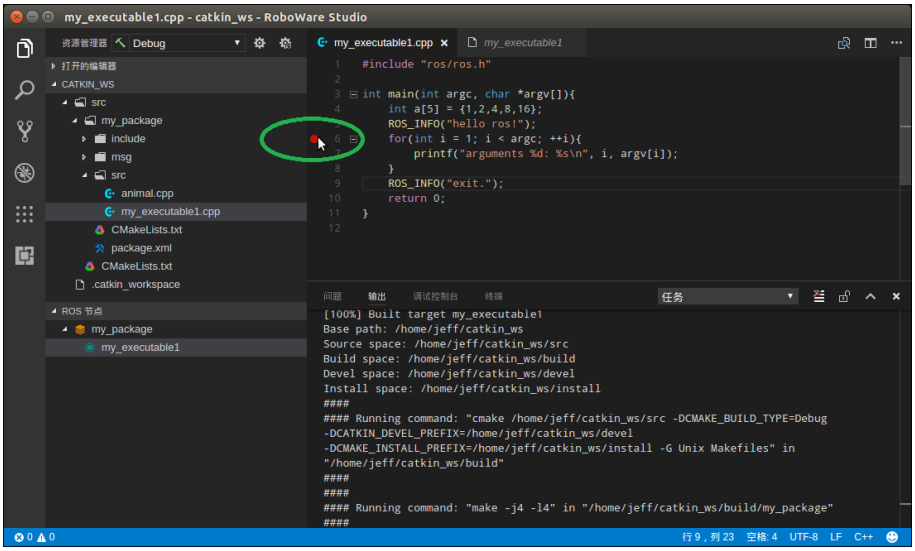
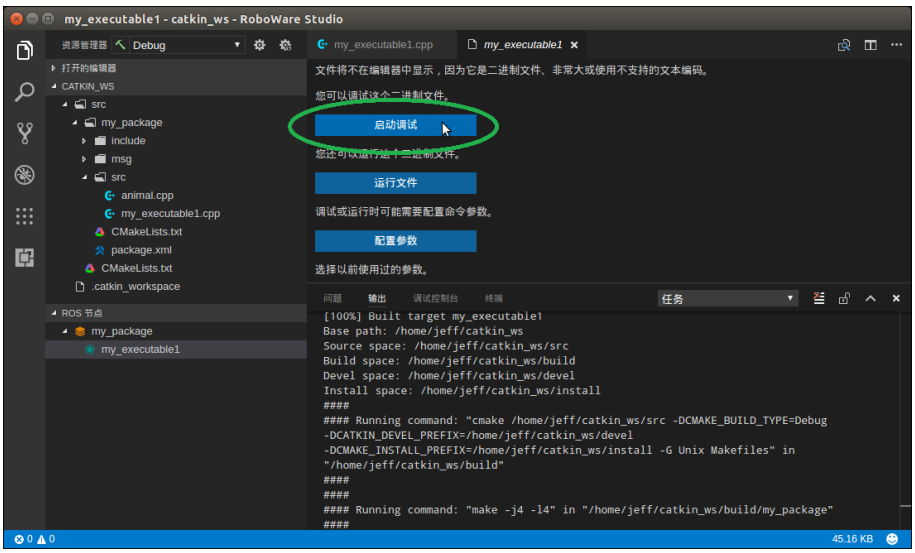


图 3-22 选择“配置参数”按钮，输入运行参数





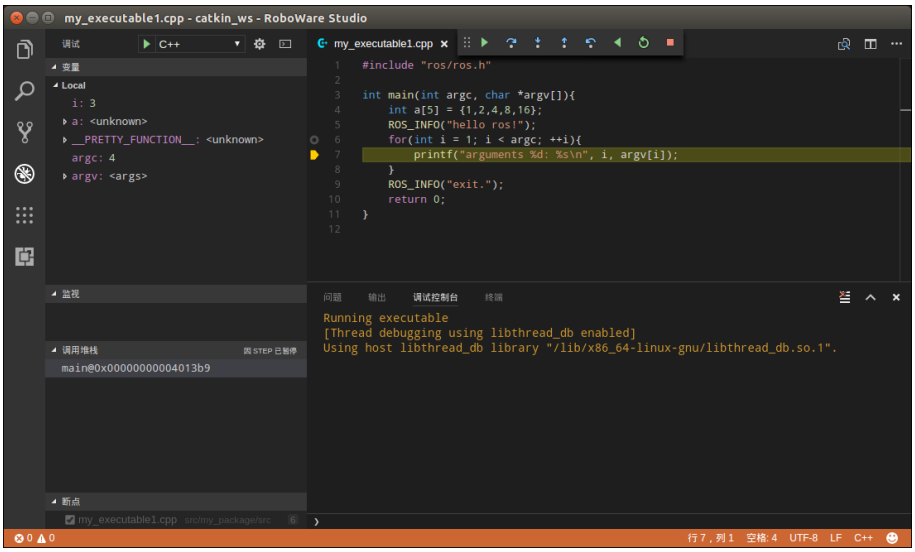


图 3-23 设置断点后，选择 “启动调试” 按钮开始调试

3.1.13 调试 Python 源代码

点击左侧侧边栏的 “调试” 图标进入调试视图，将调试器设置为 “Python” 。

点击左侧侧边栏的 “资源管理器” 图标进入资源管理器视图，打开 Python 源代码文件，设置断点，点击调试视图下的 “调试” 按钮或按 F5 键进行调试。

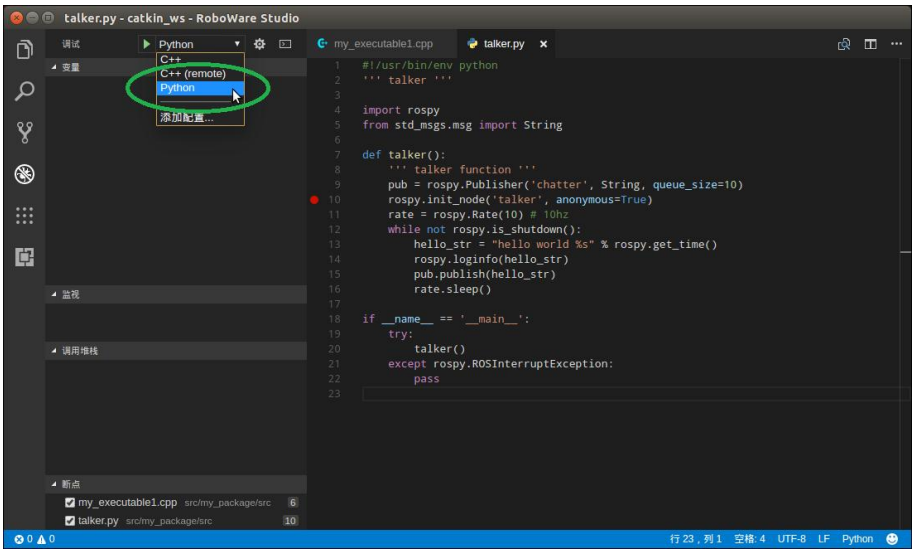


图 3-24 在调试视图中，将调试器设置为 “Python”

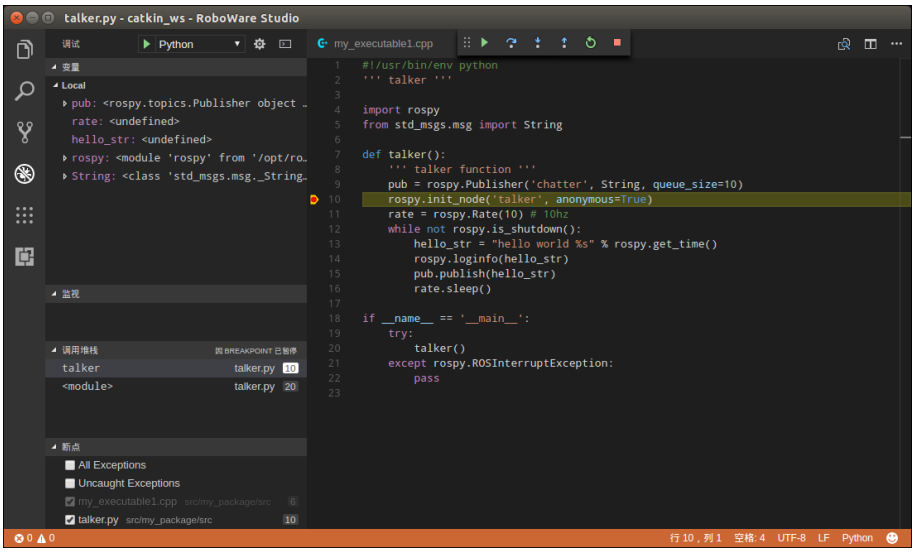


图 3-25 设置断点，点击调试视图下的“调试”按钮或按 F5 键进行调试

3.1.14 添加并启动 launch 文件

首先，右键点击包名文件夹（如“my_package”），选择“新建 Launch 文件夹”可创建 launch 文件夹。然后，右键点击 launch 文件夹，输入文件名添加 launch 文件。

编辑完后，右键 launch 文件，选择“运行 Launch 文件”即可，RoboWare Studio 会自动打开集成终端并运行 launch 文件。

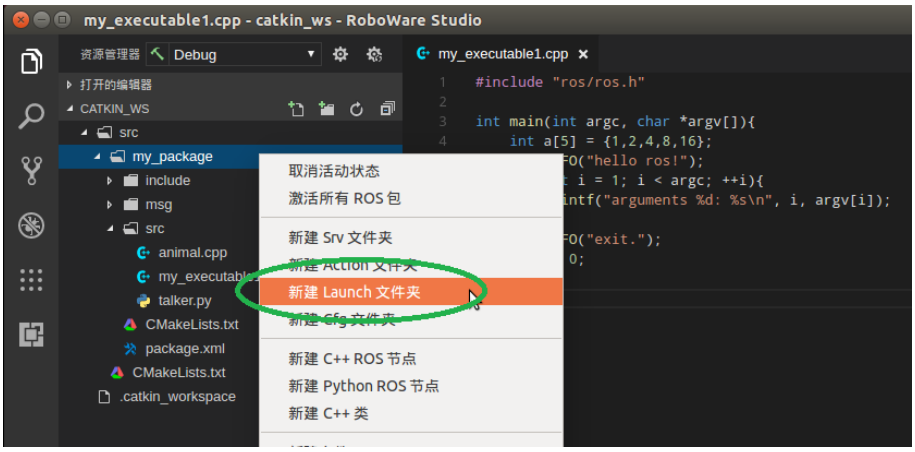


图 3-26 新建 launch 文件夹

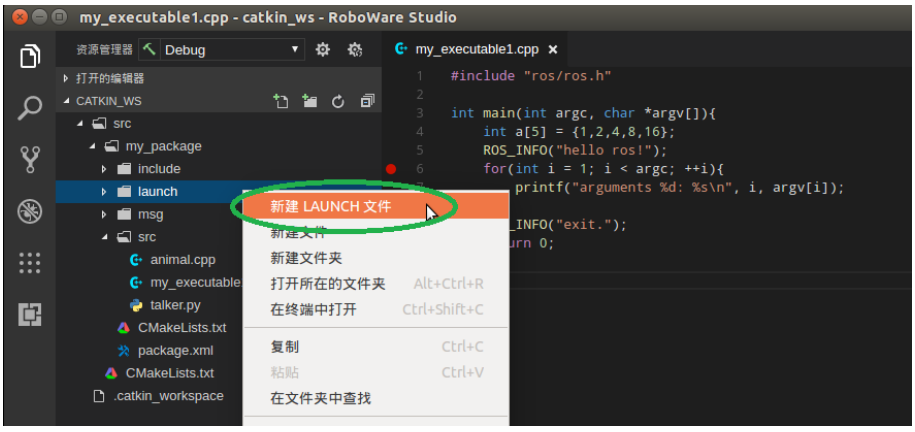


图 3-27 新建 launch 文件

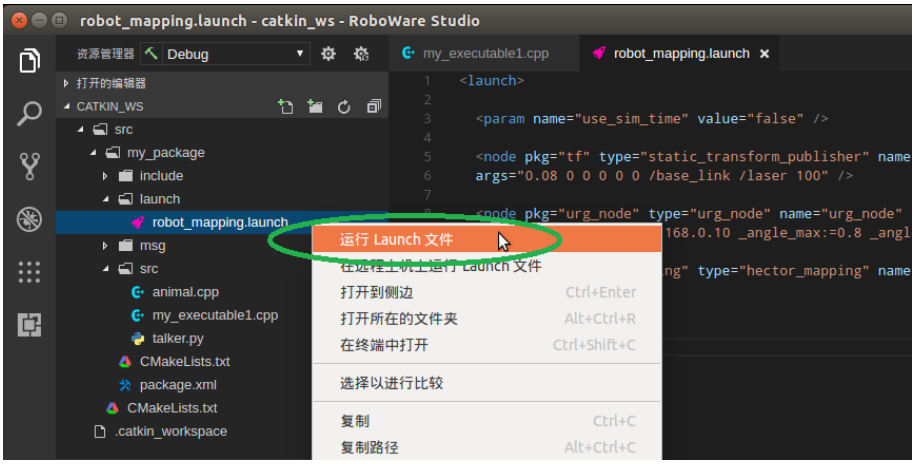


图 3-28 运行 launch 文件

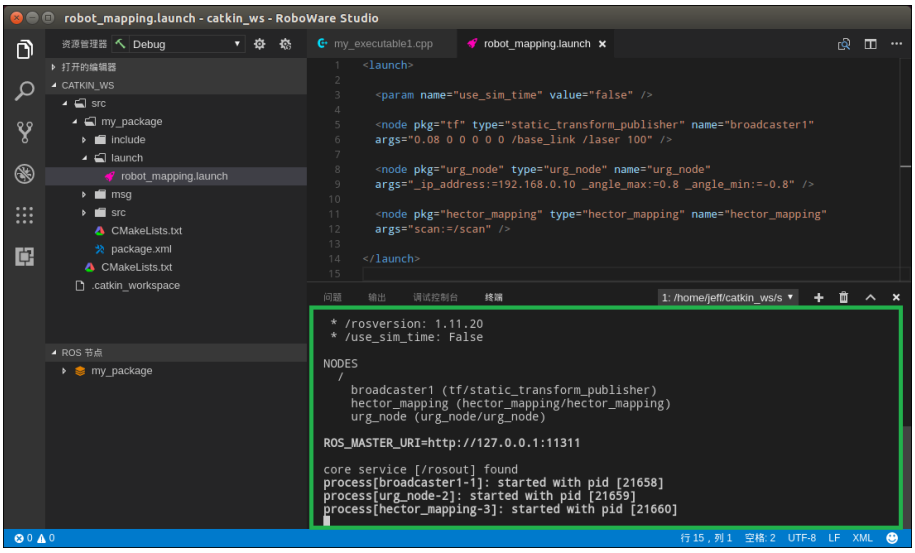


图 3-29 launch 文件在集成终端中运行

3.1.15 编辑~/.bashrc 文件

选择菜单“ROS – 打开~/.bashrc 文件”即可打开并编辑.bashrc 文件。

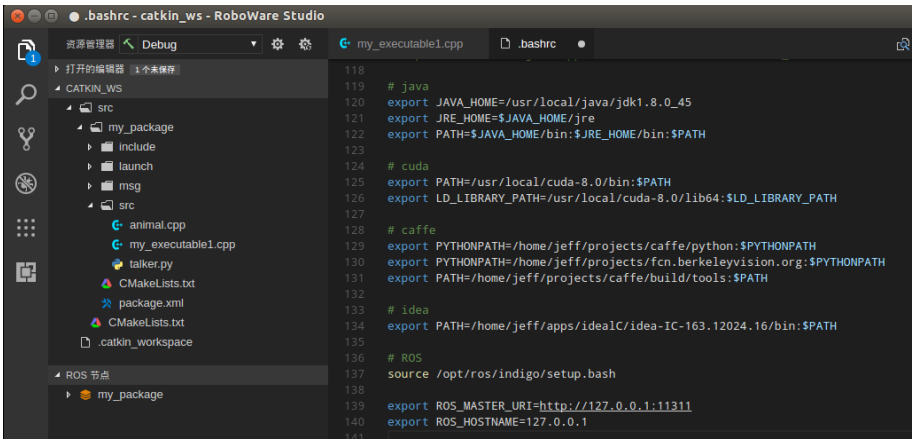


图 3-30 打开并编辑~/.bashrc 文件

另外，在配置好远程参数后，也可选择 “ROS-打开远程端.bashrc 文件”，来打开并编辑远程机器上的.bashrc 文件，修改完成后按“Ctrl+S”或选择“文件-保存”即可完成远程保存。

3.2 远程开发模式

3.2.1 配置 SSH 公钥无密登录

首先，在本地计算机生成公钥和私钥。打开终端，执行命令：

$ ssh-keygen

一直按回车键选择默认选项，会在~/.ssh 目录下生成 id_rsa 和 id_rsa.pub 两个文件。然后将 id_rsa.pub 文件复制到远程计算机：

$ scp ~/.ssh/id_rsa.pub username@ip_address:/home/username

其中 username 为远程计算机用户名，ip_address 为远程计算机的 IP 地址，示例如下所示。

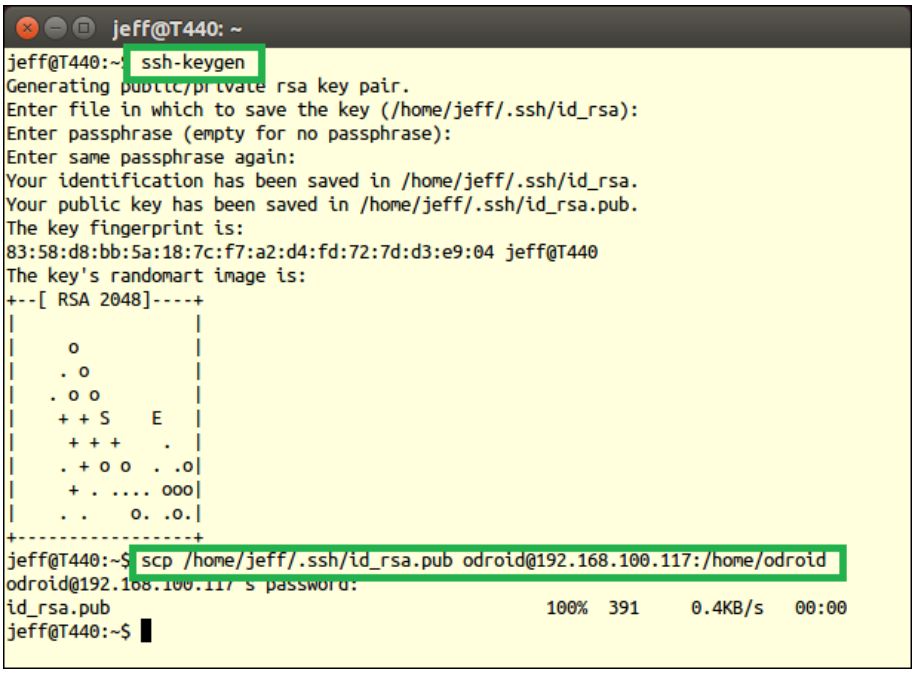


图 3-31 在本地计算机生成密钥，并将公钥文件拷贝到远程计算机

将公钥文件 id_rsa.pub 拷贝到远程计算机后，SSH 登录到远程计算机：

$ ssh username@ip_address

其中 username 为远程计算机用户名，ip_address 为远程计算机的 IP 地址。

登录后，将 id_rsa.pub 的文件内容追加写入到远程计算机的~/.ssh/authorized_keys 文件

中，并修改 authorized_keys 文件的权限：

$ cat id_rsa.pub >> ~/.ssh/authorized_keys

$ chmod 600 ~/.ssh/authorized_keys

图 3-32 登录到远程计算机，将公钥文件写入 authorized_keys 并修改文件权限

配置完成后，再登录远程计算机就无需输入密码。接下来，即可配置 RoboWare Studio 的远程调试参数进行远程调试。

3.2.2 修改远程计算机/etc/profile

首先，登录远程计算机：

$ ssh username@ip_address

其中 username 为远程计算机用户名，ip_address 为远程计算机的 IP 地址。

登录后，切换到 root 用户权限，将 ROS 环境变量信息写入到/etc/profile 文件中：

$ sudo su

$ echo "source /opt/ros/indigo/setup.bash" >> /etc/profile

在此需要注意，示例中的 ROS 版本为“indigo”，对于其它版本替换为对应名称即可。示例如下所示。

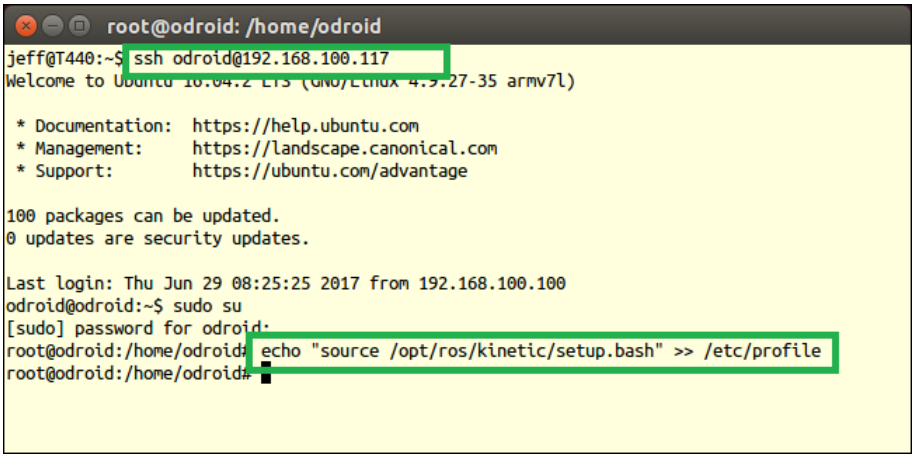
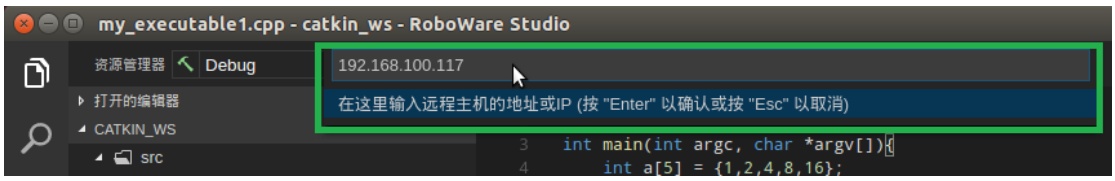


图 3-33 登录远程计算机并配置环境变量

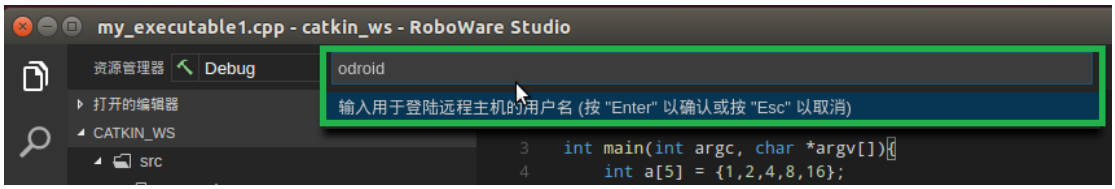
3.2.3 远程参数配置

启动 RoboWare Studio 后，点击“远程参数配置”按钮，依次配置远程计算机 IP 地址、远程计算机用户名、本地计算机密钥文件、远程计算机部署路径参数，如下所示。

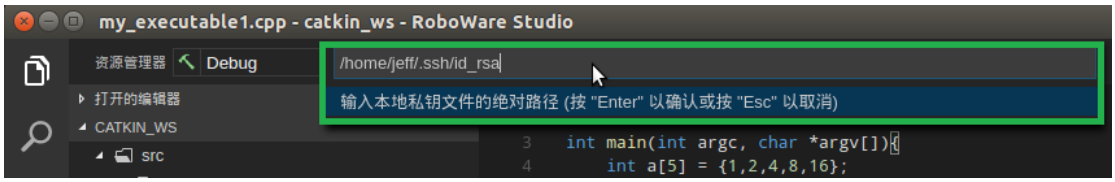
点击“远程参数配置”图标



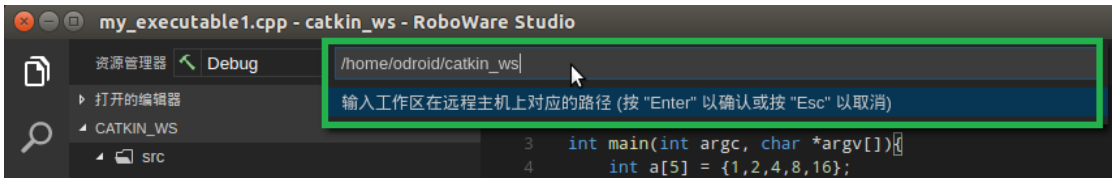
输入远程计算机 IP 地址



输入远程计算机用户名



输入本地计算机密钥文件



输入工程在远程计算机的部署路径

图 3-34 远程参数配置

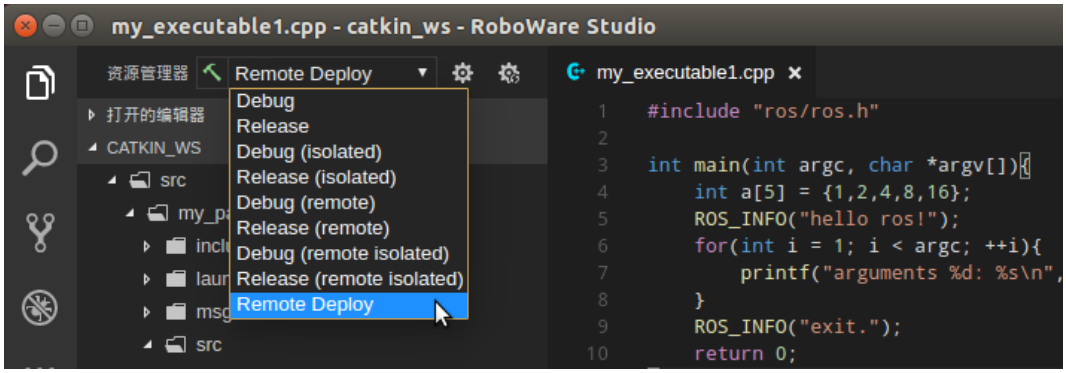
输入完成后，即完成了远程参数的配置。

3.2.4 远程部署

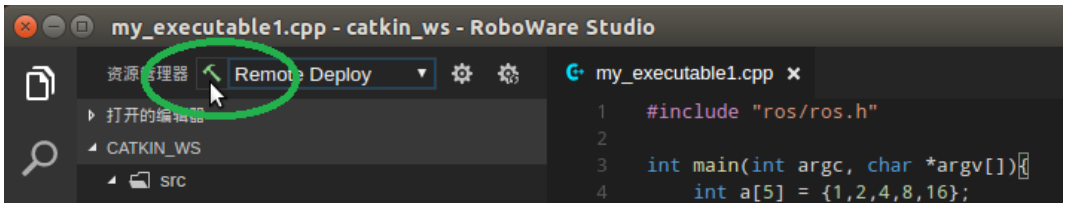
完成远程参数配置后，再进行远程部署。

首先，在资源管理器视图下，选择“Remote Deploy”远程部署选项，点击列表左侧的按钮进行远程部署。RoboWare Studio 会将当前整个工作区的源代码部署到远程计算机的指定路径下（请参照上一节的“远程参数配置”进行远程部署路径的设置）。

在远程部署过程中，左下角状态栏图标会跳动。部署完成后，会在“输出”窗口显示部署成功的信息（Deploy Finished!）。

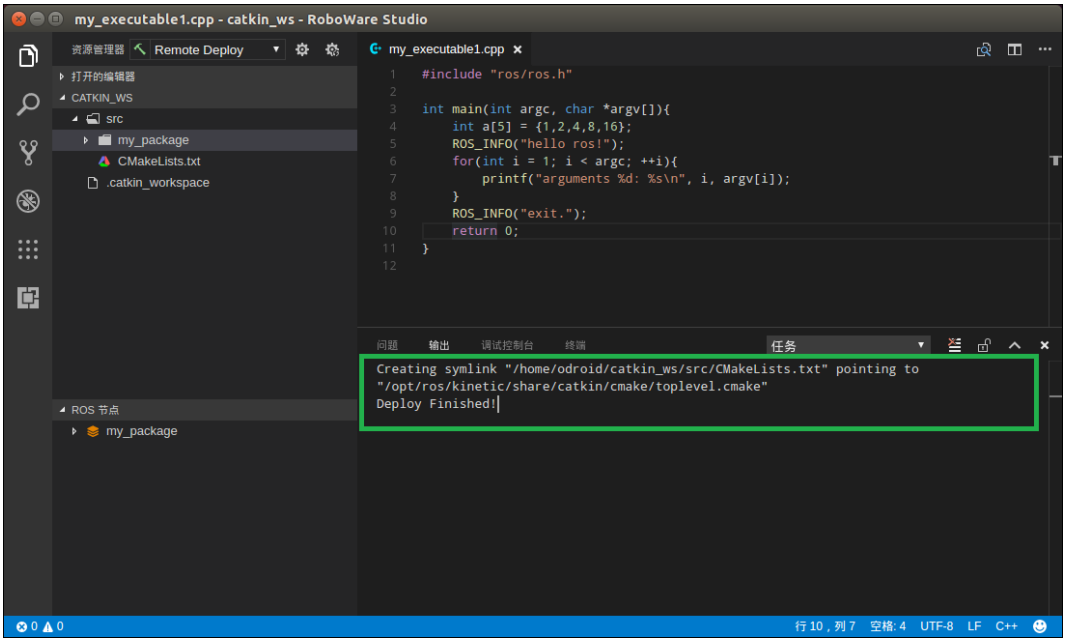


选择远程部署选项“Remote Deploy”



点击按钮开始进行远程部署

左下角状态栏提示正在远程部署



远程部署完成提示

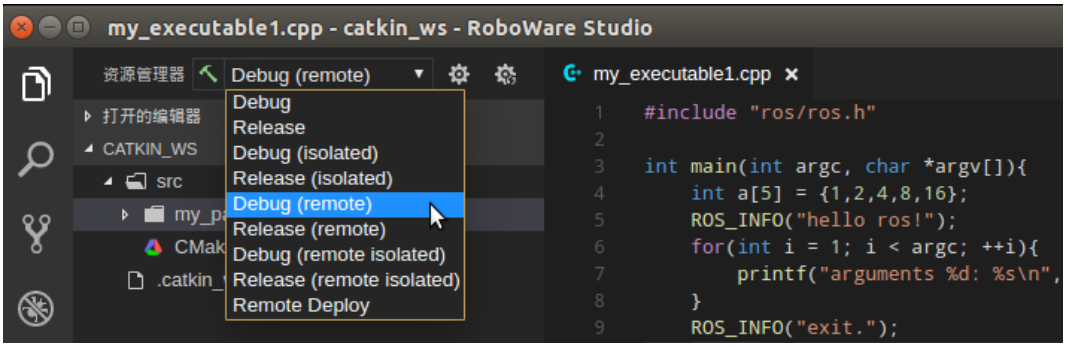
图 3-35 远程部署

3.2.5 远程构建

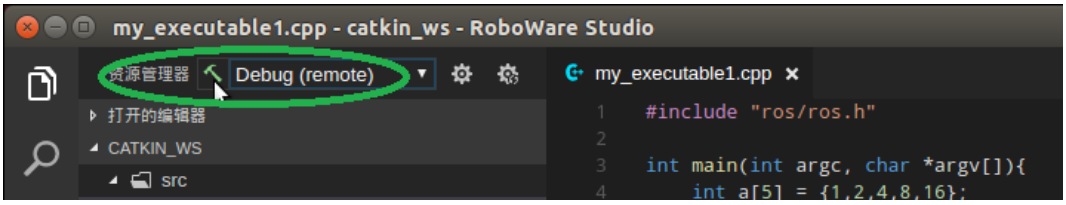
完成远程部署后，即可进行远程构建。远程构建与本地构建一样，可以选择 catkin_make 和 catkin_tools 两种构建方式，若选择 catkin_tools 构建方式，需要在远程机器上安装 catkin_tools 工具。在此仅以 catkin_make 构建方式进行说明。

首先，在资源管理器视图下，选择“Debug (remote)”构建选项，点击列表左侧的按钮进行

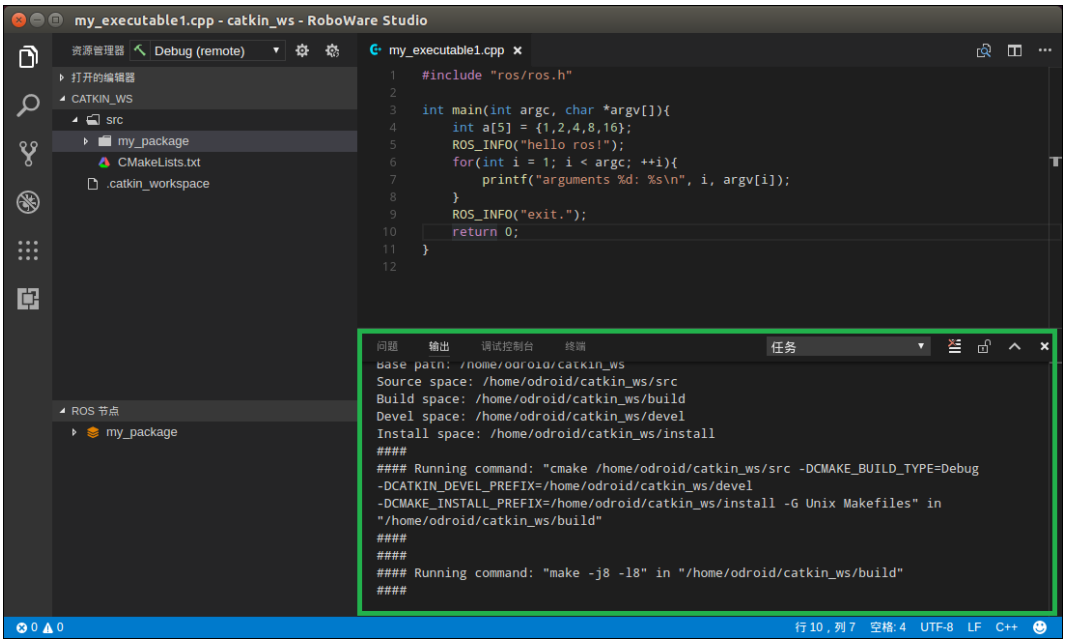
远程构建。RoboWare Studio 会将构建指令发送到远程计算机，并在“输出”窗口显示构建信息。



选择远程构建选项 “Debug(remote)”



点击按钮进行远程构建



远程构建信息显示

图 3-36 远程构建

3.2.6 远程清理

远程构建完成后，资源管理器窗口下方的 “ROS 节点” 子窗口会显示远程工作区下所有的 ROS 包及节点列表。

点击 “ROS 节点” 子窗口上的“清理”按钮，则会对远程构建结果进行清理。

在此需要注意，点击“清理”按钮，左上角的配置构建选项中如果为“remote”选项，则会清除远程构建结果，否则，则清除本地构建结果。

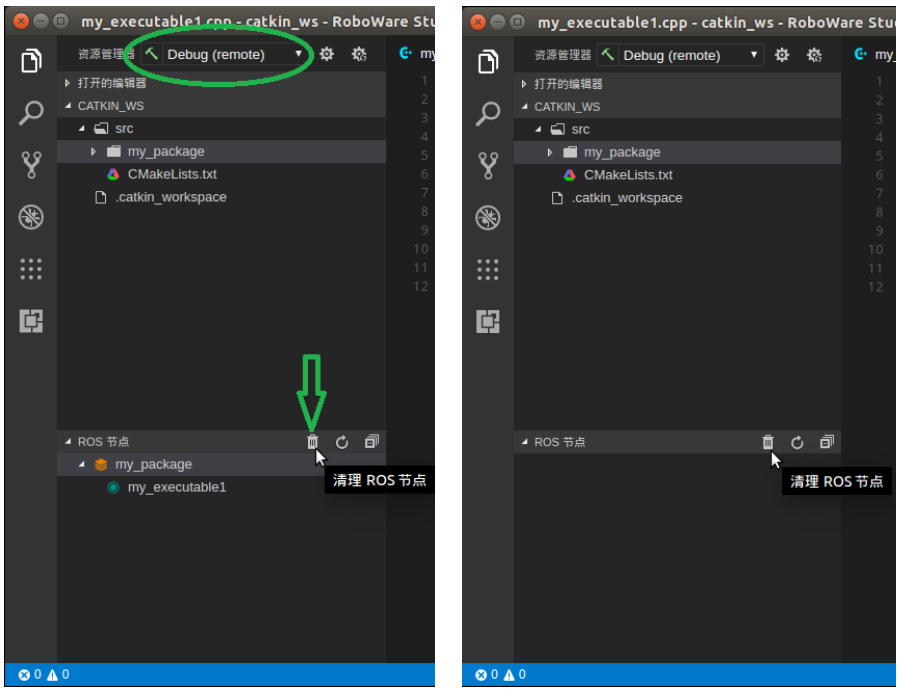


图 3-37 清理远程构建结果

3.2.7 远程调试

完成远程构建后，即可进行远程调试。为了确保已经完成 Debug 版的远程构建，需要选择“Debug (remote)”构建选项，左下方的“ROS 节点”窗口中此时应该能看到包和节点。如下所示：

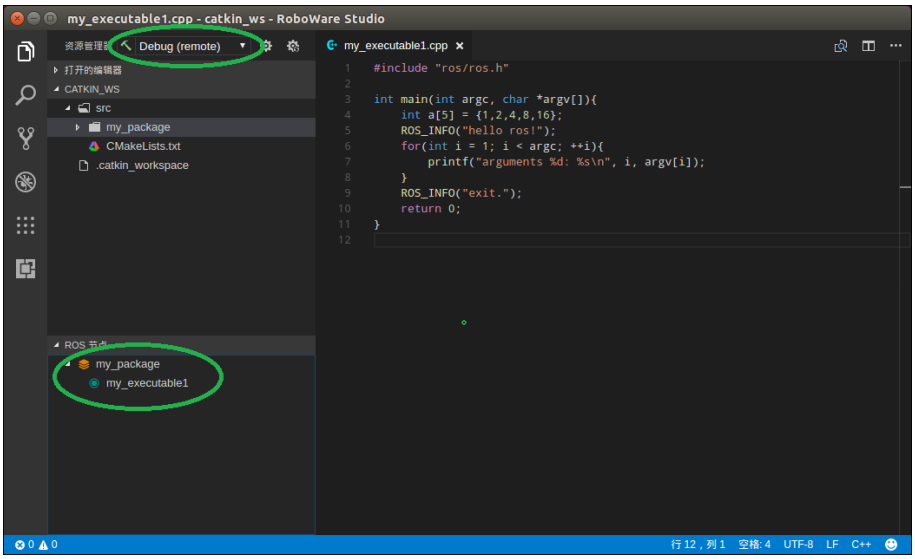


图 3-38 “Debug (remote)”构建选项下的包和节点列表

这时，点击左侧侧边栏的“调试”图标进入调试视图，可以看到调试器选项已经自动设置为“C++ (remote)”。

然后选择“ROS 节点”窗口中的节点，进行远程调试即可，调试步骤与本地调试步骤相同。

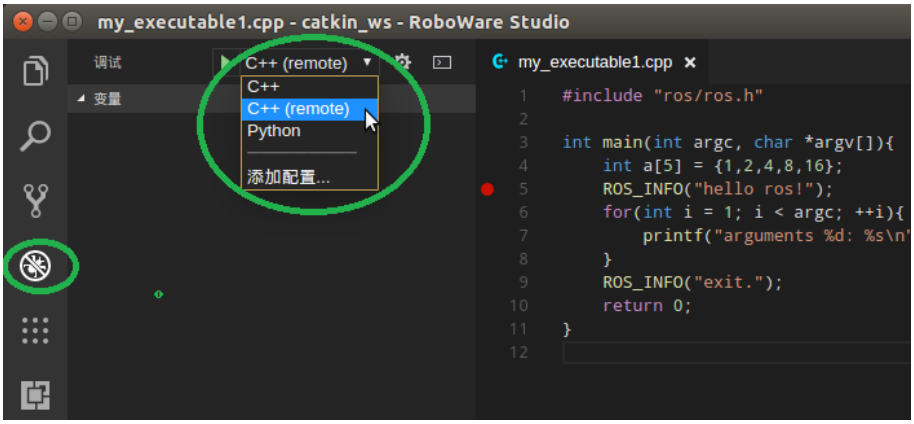


图 3-39 在调试视图下，调试器自动设置为“C++ (remote)”

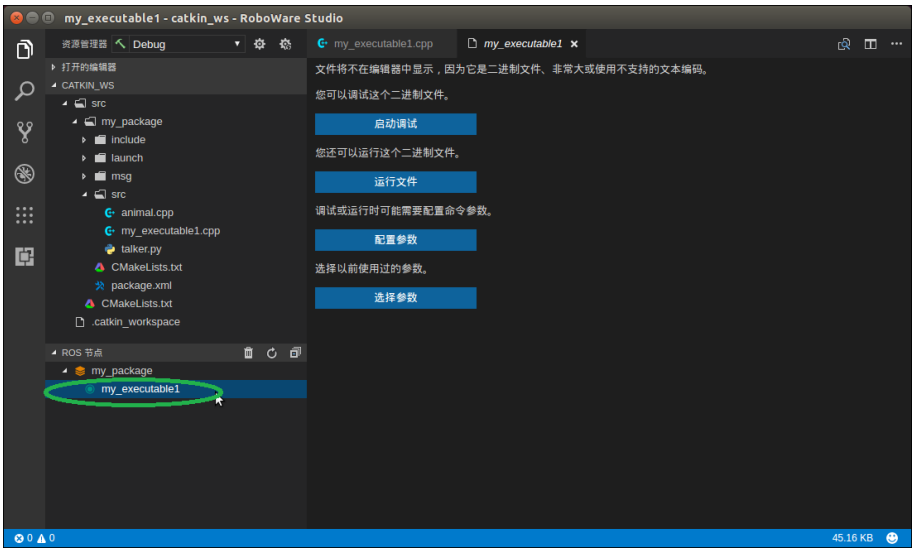


图 3-40 选择生成的节点，即可进行调试（与本地调试方法相同）

3.2.8 远程部署/构建一个或多个包

默认情况下，部署时会将当前工作区下的所有包部署到远程主机。如果只想部署其中的一个或多个包，可右键点击包名，将其设置为活动状态。可同时将一个或多个包设置为活动状态。此时，进行部署时，RoboWare Studio 只会将处于活动状态的包部署到远程主机。

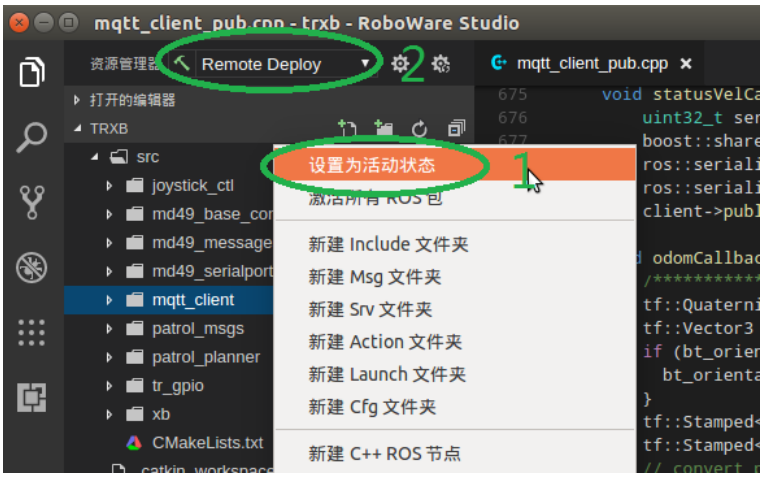


图 3-41 远程部署一个或多个包

与本地构建方式相同，当设置一个或多个活动包时，点击“构建”按钮，只会对处于活动状态的包进行构建。当清除所有包的活动状态后，则会回到初始状态（即所有包都处于非活动状态）。当所有包都处于非活动状态时，点击“构建”按钮，则会对当前工作区内的所有包进行构建。

3.2.9 远程启动 launch 文件

右键 launch 文件，选择“在远程主机上运行 Launch 文件”，RoboWare Studio 会在集成终端中启动远程主机的 launch 文件。在集成终端中使用“Ctrl + c”快捷键可终止运行。

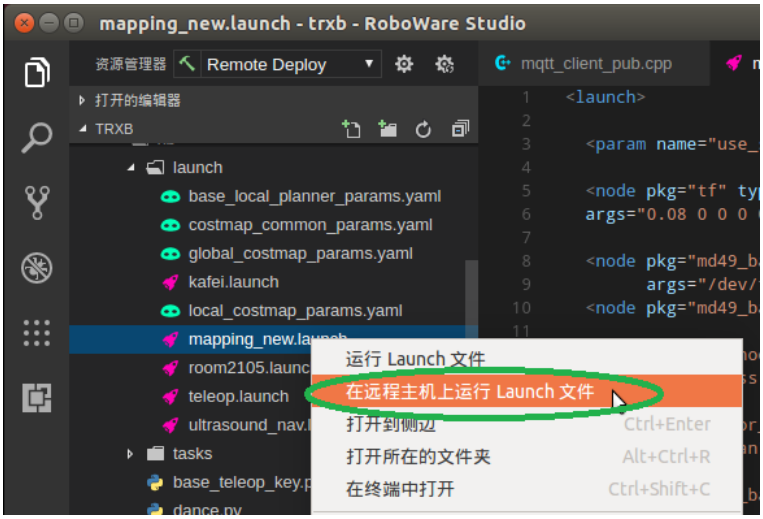


图 3-42 启动 launch 文件

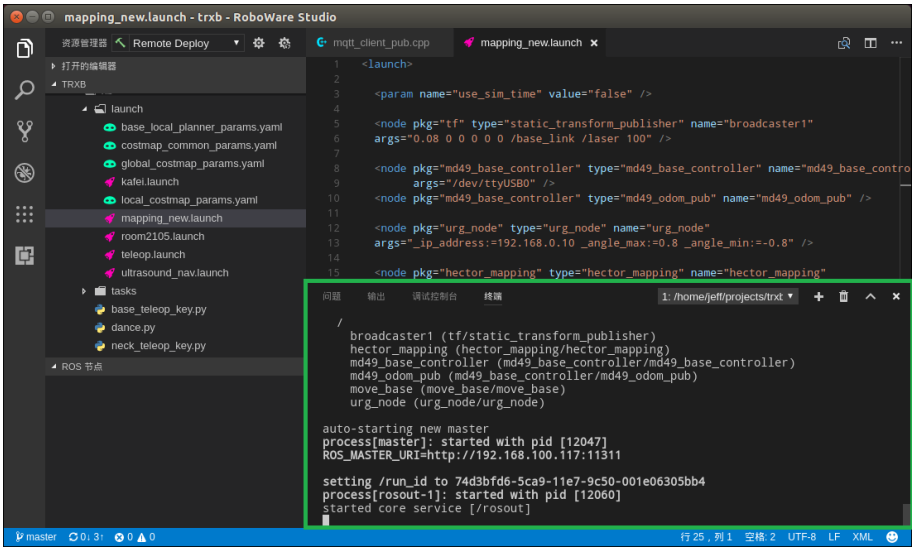


图 3-43 Launch 文件在集成终端启动

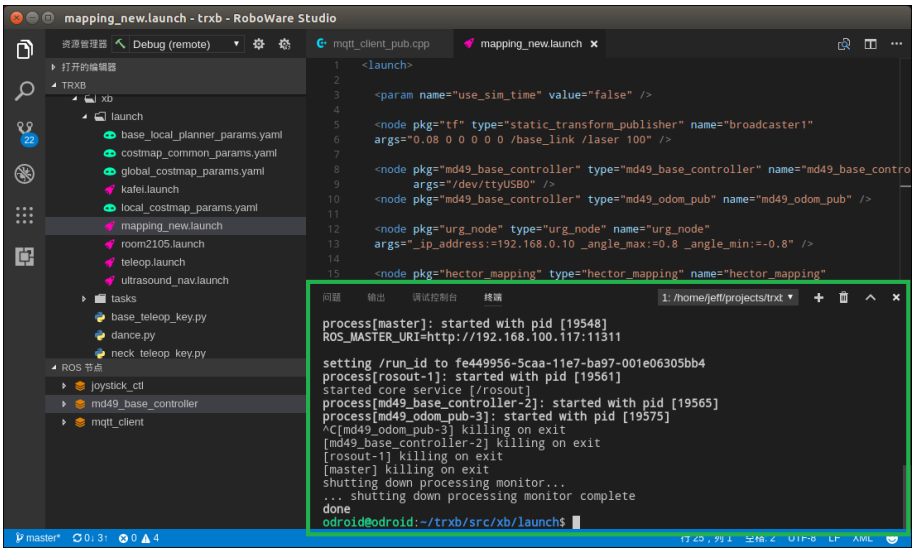


图 3-44 在集成终端“Ctrl + c”终止运行

3.3 创建 C++/Python 节点、类向导

RoboWare Studio 提供创建 ROS 节点（C++和 Python）向导和创建类向导的功能。首先，点击右键包名，选择“Add C++ ROS Node”、“Add Python ROS Node”可分别创建 C++节点和 Python 节点文件。选择“Add C++ Class”选项可创建一个 C++类文件。

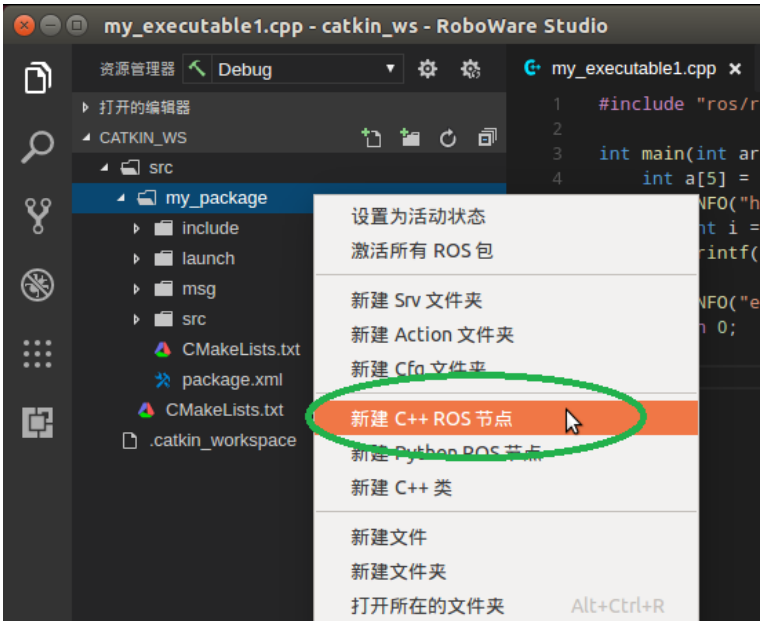


图 3-45 节点、类创建向导

通常，创建节点文件时，会同时创建两个节点文件，包括一个订阅者（sub）和一个发布者（pub）文件。例如，若需要创建一个名为“chatter”的 C++订阅者节点，右键包名选择“Add C++ ROS Node”后，在输入框中输入“chatter”，回车后，即可在该包的 src 目录下创建一个 chatter_sub.cpp 文件和一个 chatter_pub.cpp 文件，文件会自动添加节点初始化、订阅和发布代码。在此，若想仅保留订阅者节点，只需右键点击“chatter_pub.cpp”删除即可，CMakeLists.txt 内容会自动更新。

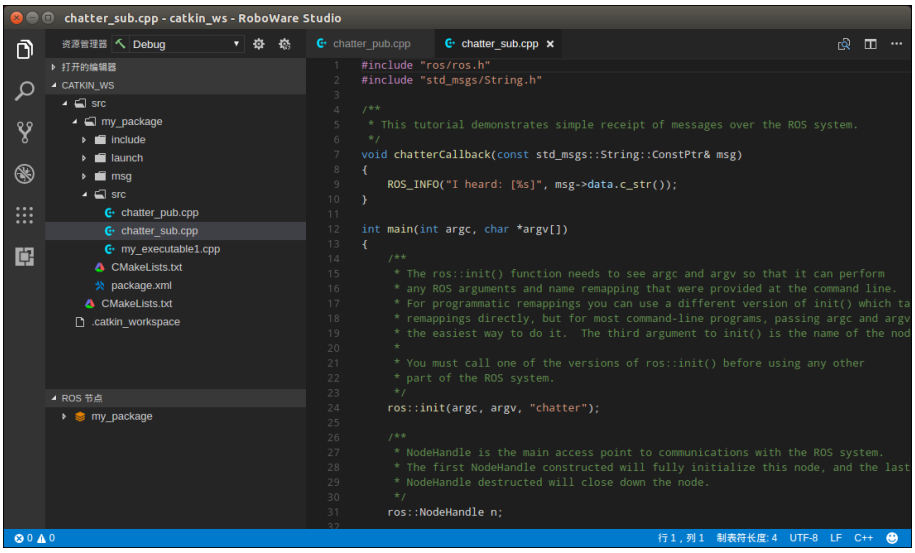
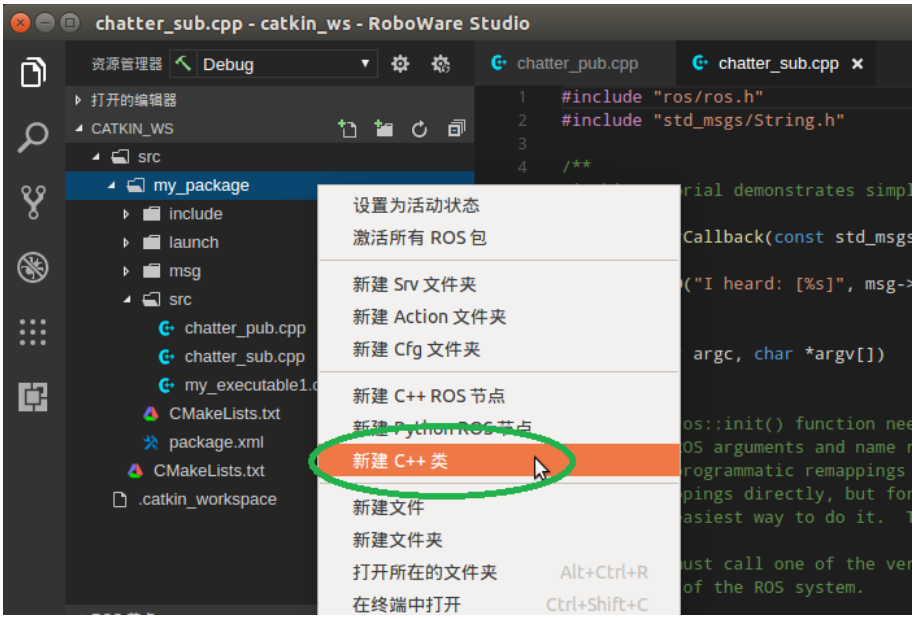


图 3-46 创建 chatter 节点

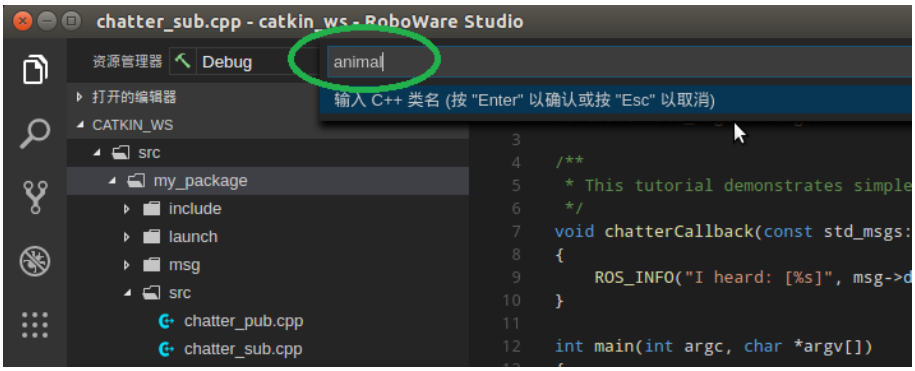
同理，创建 Python 节点时，也会产生一个“*_sub.py”节点文件和一个“*_pub.py”节点文件，并自动生成节点代码。生成后，将不需要的节点删除即可。

创建 C++类步骤与创建节点步骤相同，在此，以创建名为“animal”的类为例。

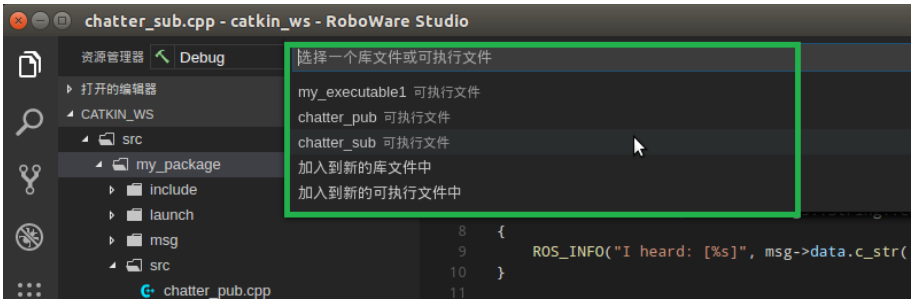
首先，右键点击包名，选择“Add C++ Class”；然后输入类名 animal；接下来，选择引用该类的节点或库；最后回车完成，会分别在当前包的 include/<package_name>路径和 src 路径下生产 animal.h 头文件和 animal.cpp 源文件，并自动修改 CMakeLists.txt。如下图所示。



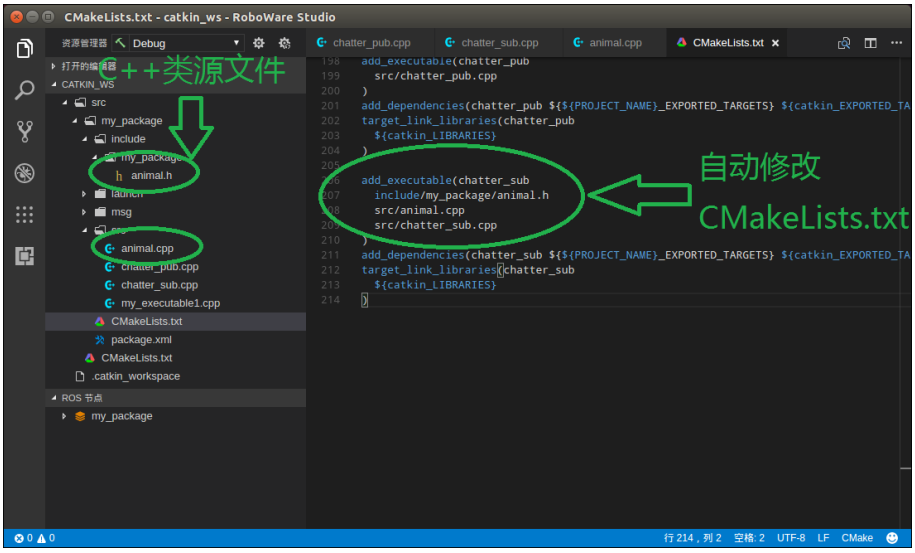
选择添加 C++类向导



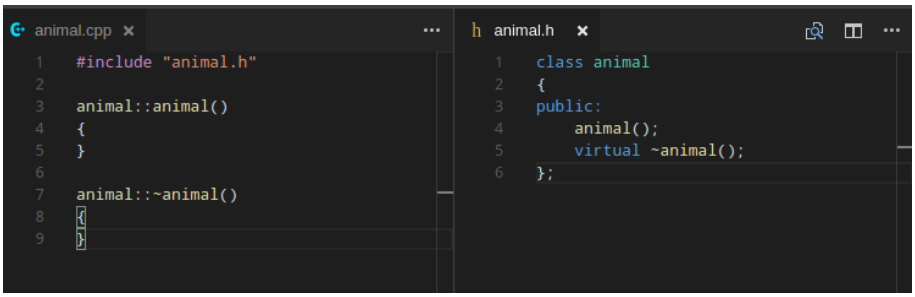
输入类名“animal”



选择引用该类的节点或库



完成 animal 类文件创建



animal 类文件内容

图 3-47 创建类

3.4 软件首选项配置

RoboWare Studio 软件可方便地进行的首选项配置，以点选的方式实现用户配置、工作区配置以及主题配置等。选择“文件 – 首选项 – 设置”即可打开配置界面。

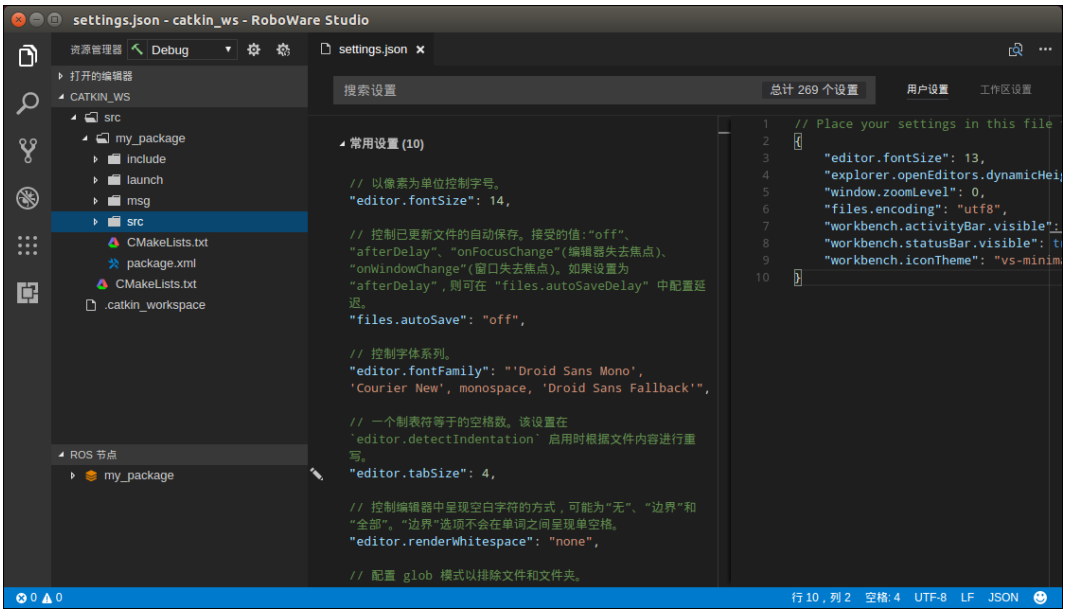


图 3-48 RoboWare Studio 软件首选项图界面

3.5 ROS 命令图形化列表

3.5.1 roscore、RViz、rqt、rqt-reconfigure、rqt-graph 启动

在菜单栏中，选择“ROS-运行 roscore”、“ROS-运行 RViz”、“ROS-运行 rqt”、“ROS-运行 rqt-reconfigure”、“ROS-运行 rqt-graph”可分别启动 roscore、RViz、rqt、rqt-reconfigure、rqt-graph。

另外，在配置好远程参数后，也可选择“ROS-运行远程端 roscore”来启动远程机器上的 roscore。

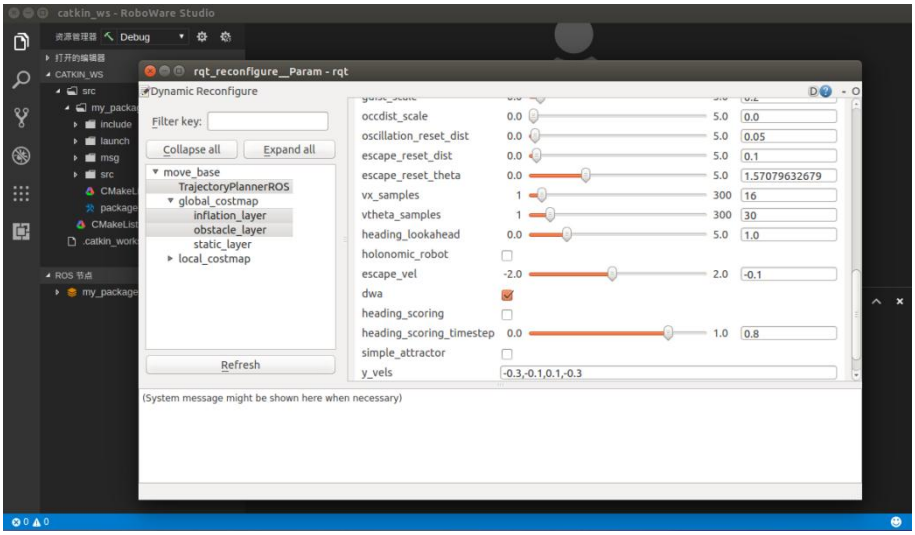


图 3-49 启动 rqt-reconfigure

3.5.2 显示活动话题/节点/服务，以及已安装的包/消息/服务

点击左侧侧边栏的“ROS”图标进入 ROS 视图，则会显示活动话题/节点/服务，以及已安装的包/消息/服务。点击相应的标签名称可进行折叠、展开。请注意，显示活动话题/节点/服务需要有正在运行的节点。

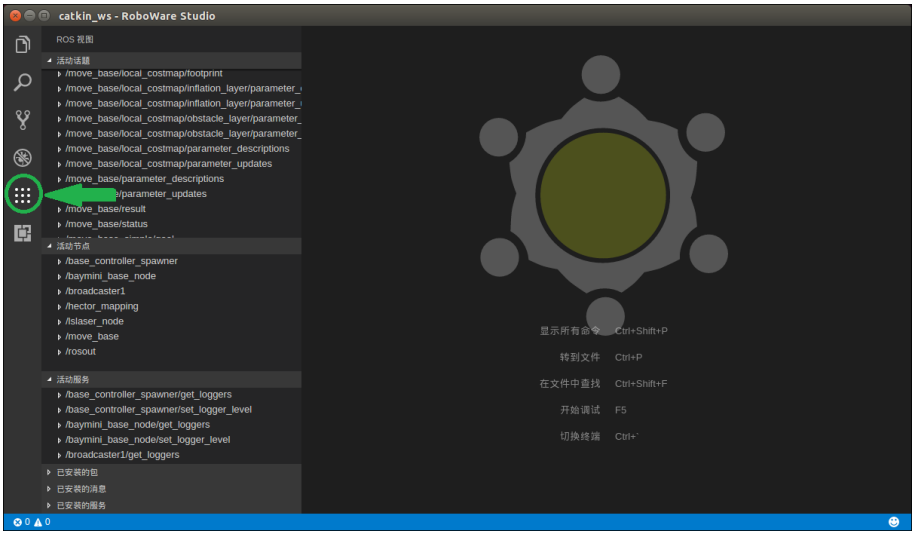


图 3-50 进入 ROS 视图

单击活动话题/节点/服务名称，或已安装的消息/服务名称，即可查看相应的信息，双击活动话题名称可查看话题的实时数据流。

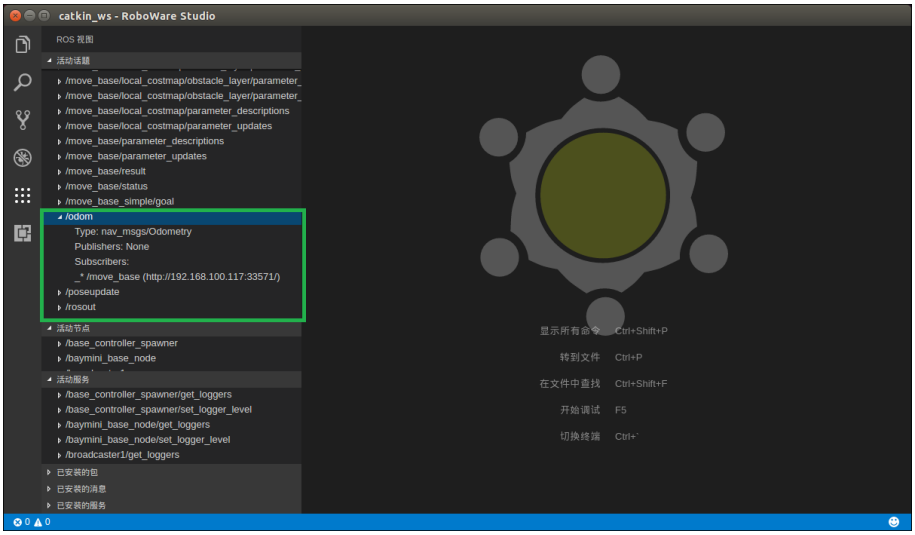


图 3-51 单击查看话题的类型信息

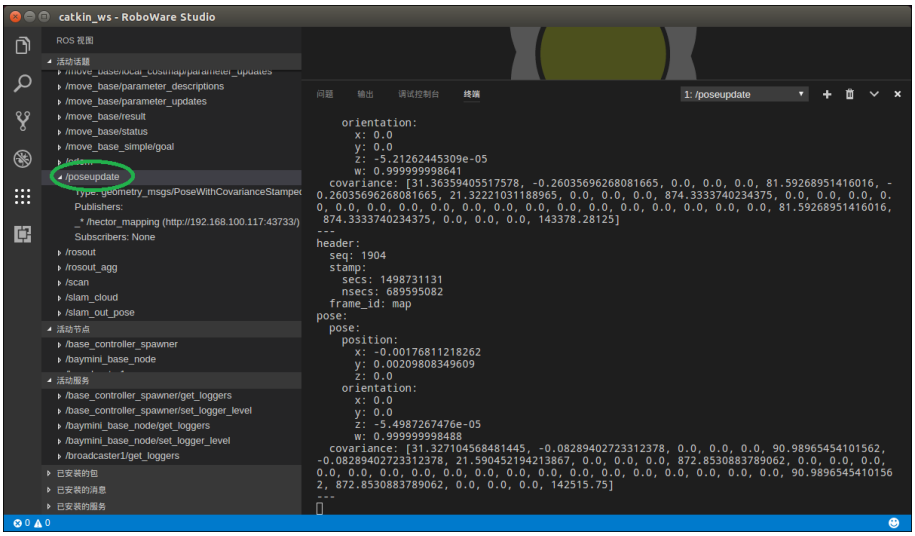


图 3-52 双击查看话题的实时数据流

3.5.3 rosbag 记录与播放

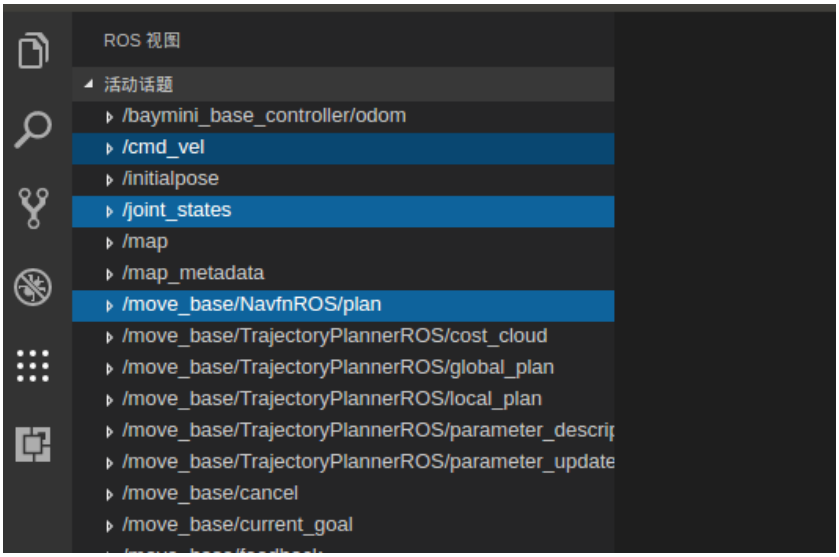
I．记录

如上一节所述，点击左侧侧边栏的 “ROS” 图标，进入 ROS 视图后，展开活动话题（Active Topics）标签，则会显示当前正在发布的话题。此时，可以使用 rosbag 工具对当前发布的话题进行记录。

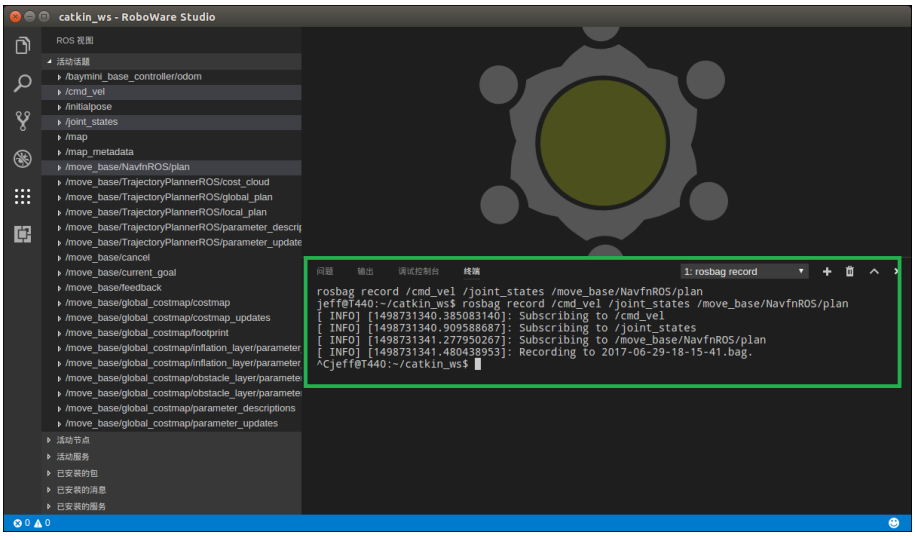
如下图所示，直接点击 “记录 ROS 话题” 按钮可进行记录。默认情况下，会对当前所有活动话题进行记录。如果想要记录其中的一个或多个话题，则可以按住 Ctrl 键的同时，点击鼠标左键进行选择，完成选择后再点击 “记录 ROS 话题” 按钮，即可对选中的话题进行记录。

记录的 bag 文件存储路径为当前工作区的根目录，文件名称为当前时间（精确到秒）：“yyyy-MM-dd-HH-mm-ss.bag” 。记录任务在 RoboWare Studio 的集成终端进行，因此，可以在集成终端按“Ctrl+c”快捷键停止记录。

图 3-53 执行 rosbag 记录当前活动话题



选择多个需要记录的 Topic



在集成终端按“Ctrl+c”停止记录

图 3-54 记录一个或多个话题（按住 Ctrl 键后点击选择）

II．播放

点击左侧标签回到资源管理器视图，在工作区根目录下选择 bag 文件，右键点击文件名，选择“播放 BAG 文件”即可播放选中的 bag 文件，选择“循环播放 BAG 文件”即可循环播放选中的 bag 文件。播放过程是在集成终端进行，因此也可以通过点击“终止终端”按钮或“Ctrl+c”快捷键停止播放。

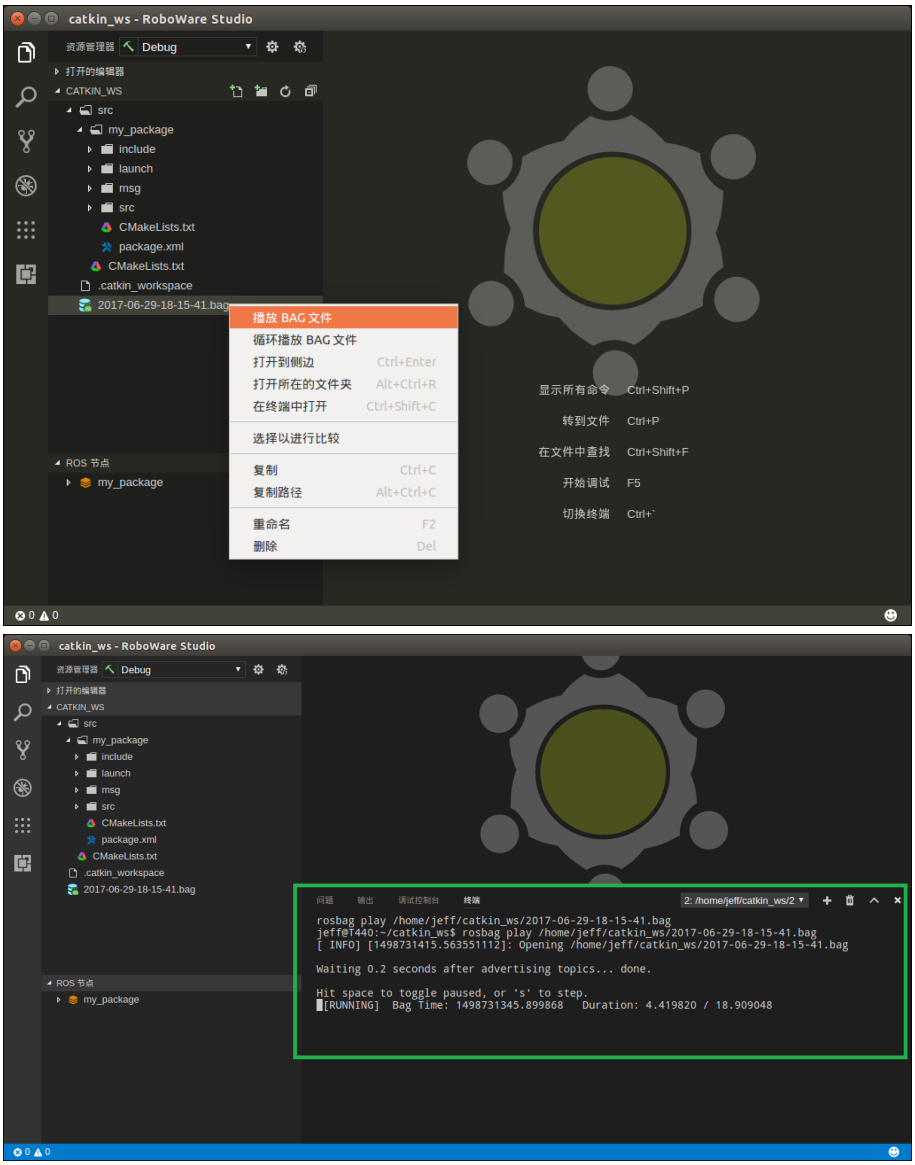
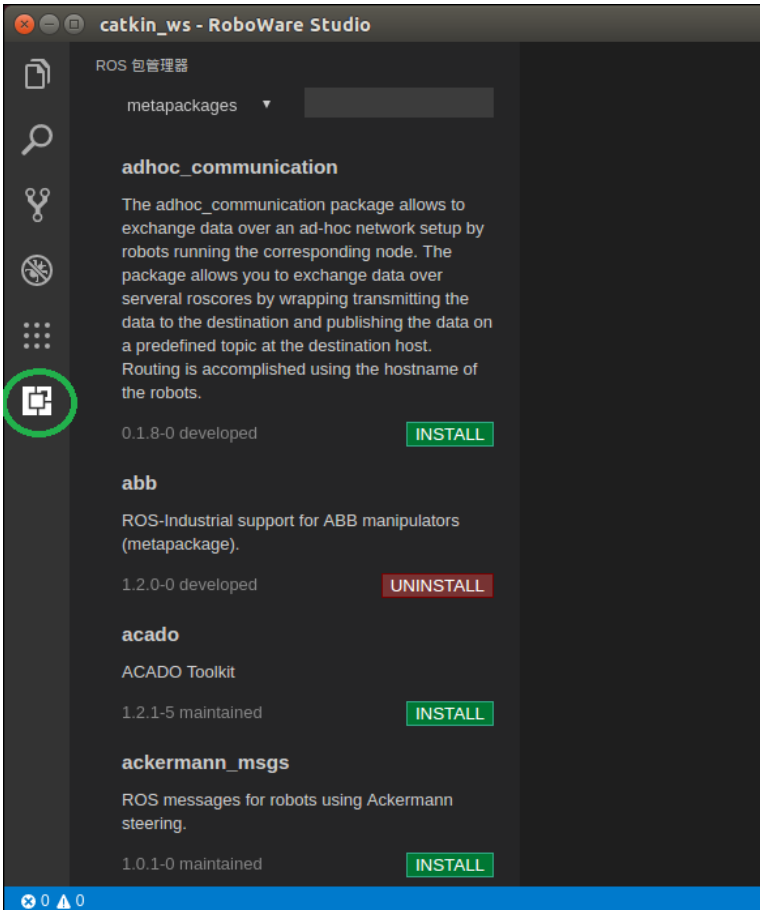


图 3-55 播放 bag 文件

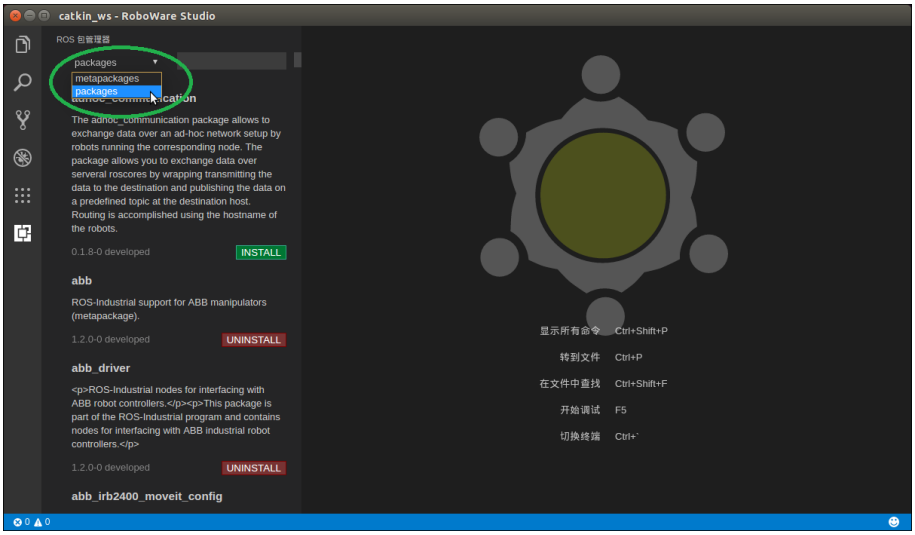
3.6 ROS 包管理器

在 0.5.0 版本中，ROS 包管理器移到了左侧工具栏中，可在工具栏的右键菜单中选择“ROS 包管理器”调出包管理器按钮。RoboWare Studio 会自动检测系统的 ROS 版本，并列出此版本对应的所有 ROS 包（包含已安装和未安装包）。其中，可切换 ROS 包（packages）和元包（meta packages）选项，显示不同内容列表。在右侧文本框可输入包名进行搜索。

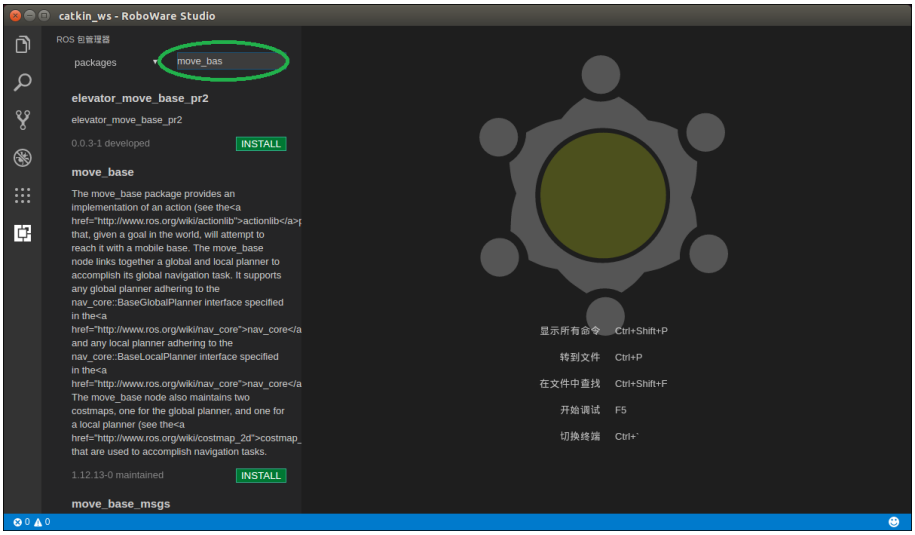
可以查看包的描述信息，点击包介绍可打开包的维基页面。点击安装按钮可安装包，当出现命令提示时请在集成终端中输入超级用户密码，系统会自动调用 apt-get 安装选中的 ROS 包。安装以后的包可点击卸载将其卸载掉，请谨慎卸载 ROS 包，卸载不当可造成 ROS 不能正常运行。



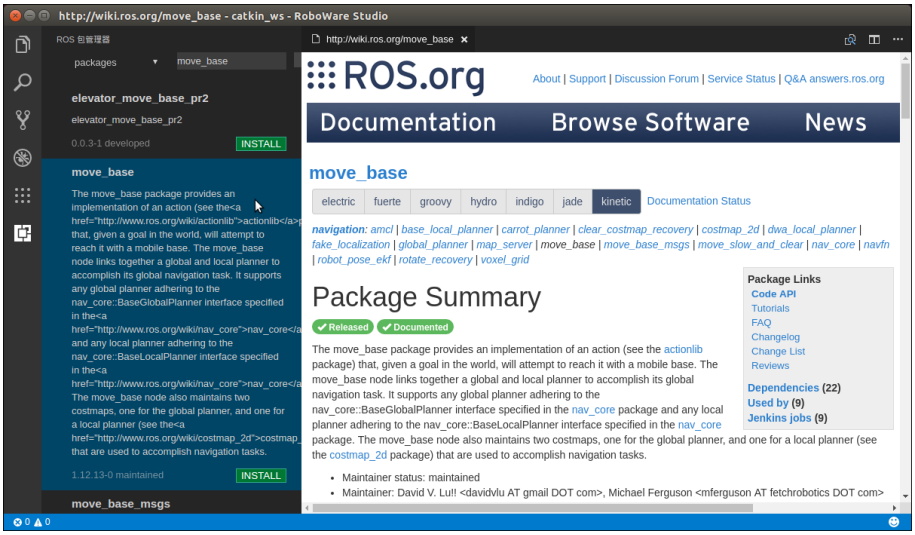
打开 ROS 包管理器



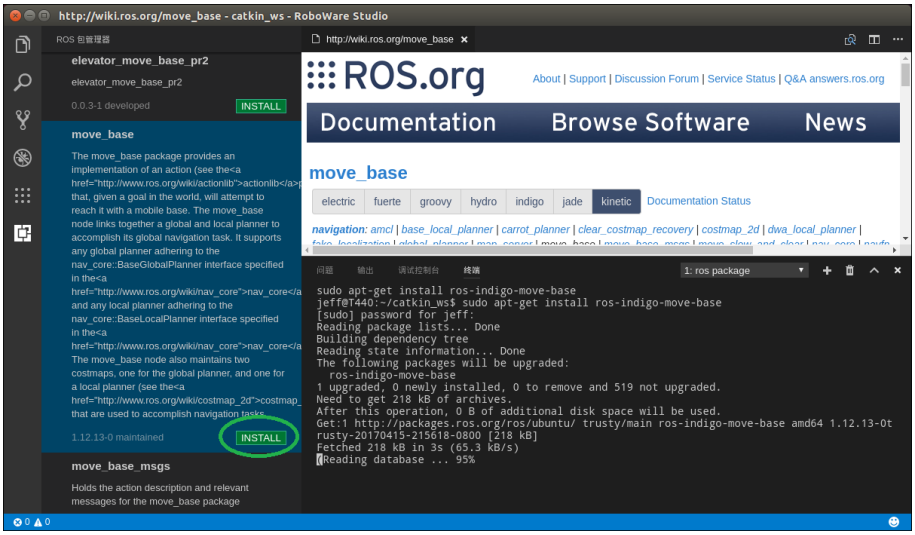
选择 ROS 包（packages）或元包（meta packages）选项



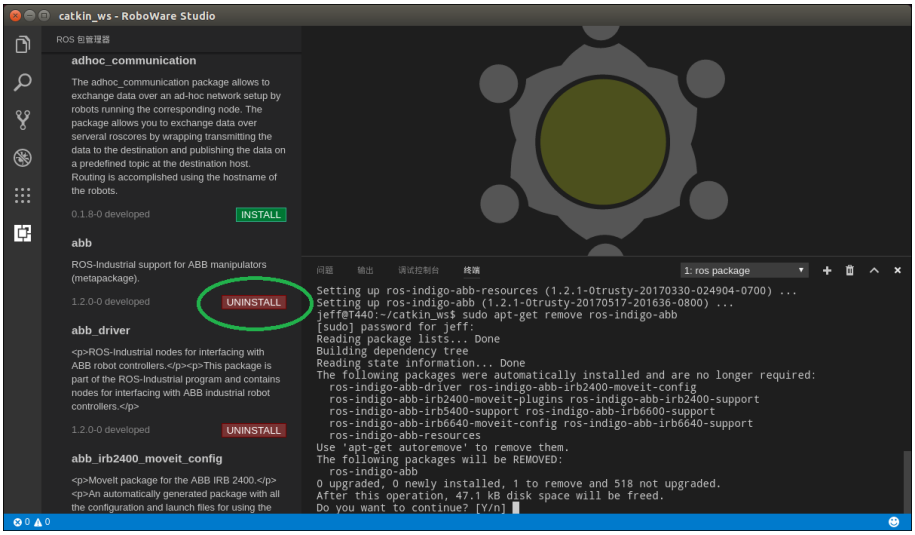
输入包名称进行搜索



点击包名称后，进入包的维基页面



包的一键安装

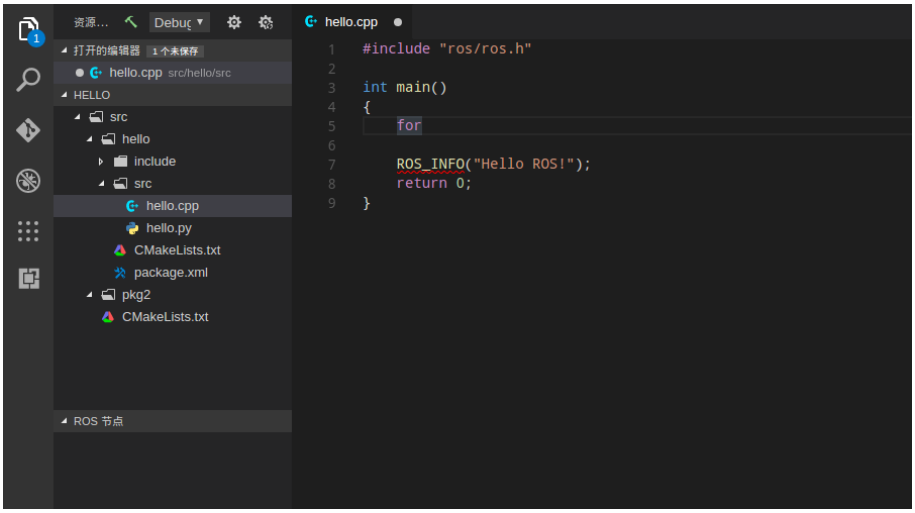


包的一键卸载

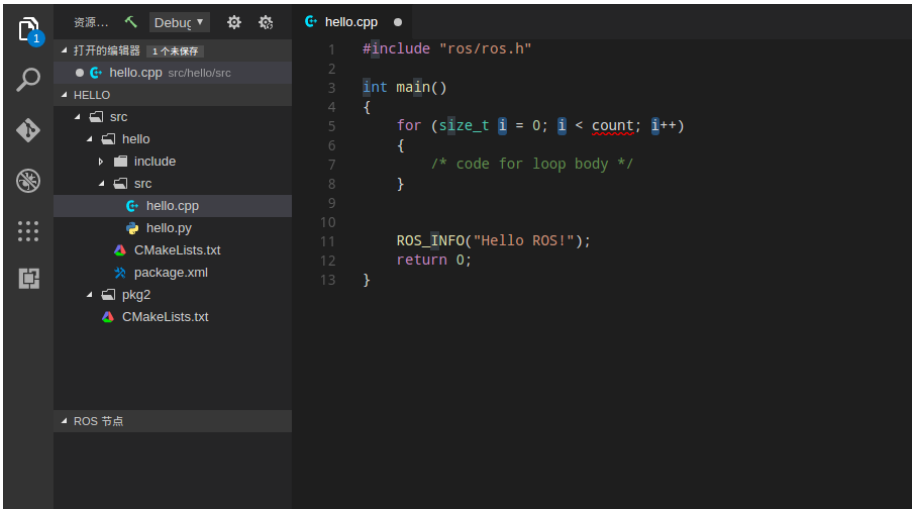
图 3-56 ROS 包管理器

3.7 代码片段

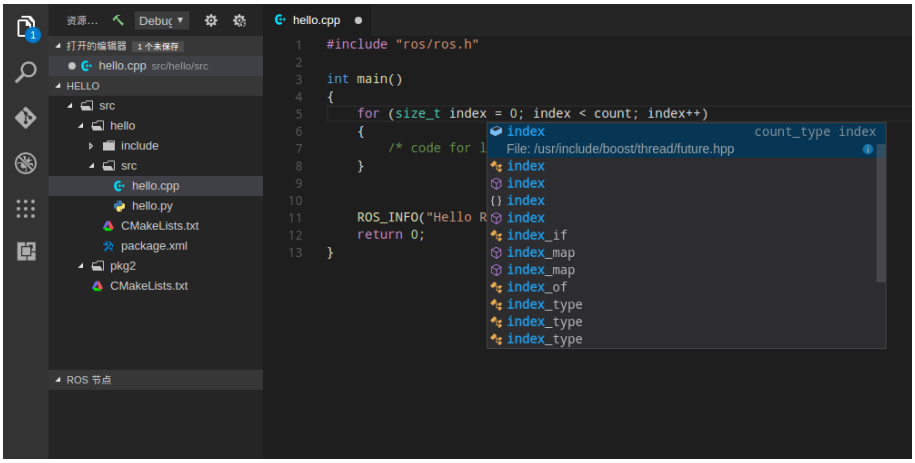
代码片段是 RoboWare Studio 中预定义的一些代码模板，可在编辑代码时键入对应的前缀，按下 Tab 键，编辑器会自动补全对应的完整代码，在编写代码时使用片段可以极大地提高编码速度。与一般编辑器不同的是，RoboWare Studio 中的代码片段还支持参数修改。以 C/C++的 for 循环为例：



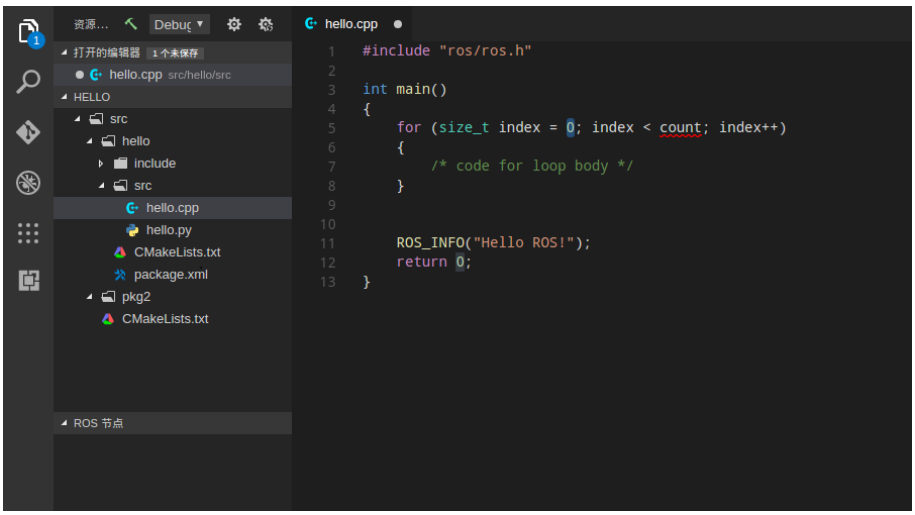
在 cpp 文件中键入 for 前缀



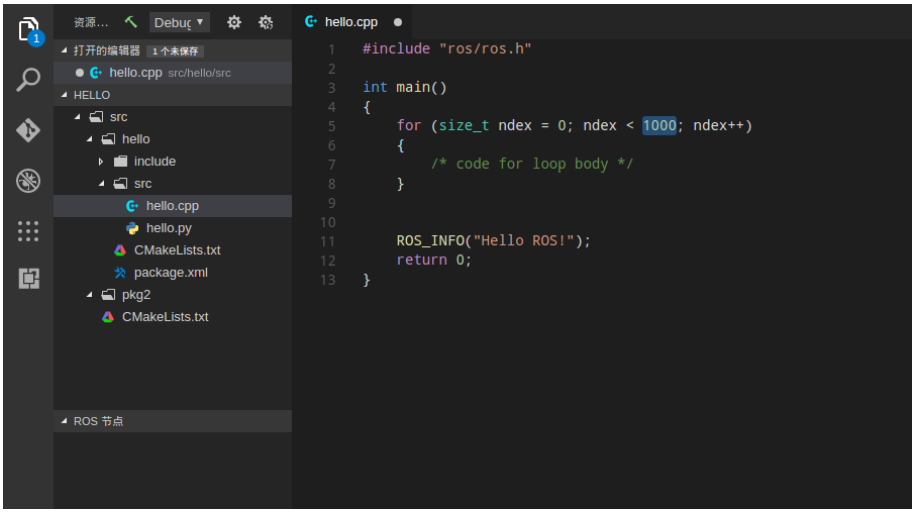
按下 Tab 键即可将其补足为完整的 for 循环语句



此时光标定位在循环变量 i 上，直接键入 index 即可将其所有引用全部改名为 index



再次按 Tab 键，光标定位到终止条件，最终光标会定位到循环体



再次按 Tab 键，光标自动定位到下一个预编辑位置，设置循环变量初始值

图 3-57 代码片段

以下列出 C++和 Python 语言所有可用的代码片段：

C++语言结构	对应前缀
#if	#i
#ifdef	#ifd
#ifndef	#ifn
#else	#e
#define	#def
#undef	#und
inc	inc
Inc	Inc
typedef	type
enum	enum
union	uni
struct	stru
class	class
namespace	name
main	main
if	if
else	el
else if	ei
for	for
do	do
while	while
switch	swi
case	case
try	try
map	map
string	str

vector	vect
template	temp
ros::init	ros::init
ros::Publisher	ros::pub
ros::Subscriber	ros::sub
ros while	roswhile

Python 语言结构	对应前缀
py	py
py3	py3
ase	ase
asne	asne
asr	asr
as	as
fail	fail
prop	prop
ifmain	ifmain
.	.
if	if
if/else	if/else
elif	elif
else	else
while	while
while/else	while/else
for	for
for/else	for/else
try/except	try/except
try/finally	try/finally
try/except/else	try/except/else
try/except/finally	try/except/finally
try/except/else/finally	try/except/else/finally
with	with
def	def
def(class method)	def(class method)
def(static class method)	def(static class method)
def(abstract class method)	def(abstract class method)
class	class
lambda	lambda
if(main)	if(main)
async/def	async/def
async/for	async/for
async/for/else	async/for/else
async/with	async/with

pdb	pdb
rospy.init_node	ros.init
rospy.Publisher	ros.pub
rospy.Subscriber	ros.sub
ros while	roswhile

3.8 Vim 编辑模式

选择“编辑 – Toggle Vim Mode”，即可在普通编辑模式和 Vim 编辑模式下进行切换。

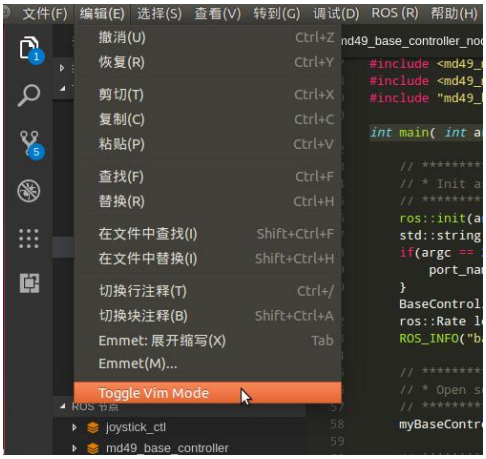


图 3-58 切换 Vim 编辑模式

Vim 插件具有以下特性。

（1）支持模式：普通模式（normal）、插入模式（insert）、命令行模式（command-line）、可视模式（visual）。

（2）支持命令组合，如：c3w、daw、2dd……

（3）支持指令按键映射功能。

（4）支持/ 和 ?增量查找功能。

（5）内置了 easymotion、surround、commentary 等常用插件。

（6）提供与与.vimrc 类似的 Vim 配置功能。

选择“文件 – 首选项 – 设置”，打开设置窗口，找到“Vim Configuration”模块，即可进行 Vim 参数设置。

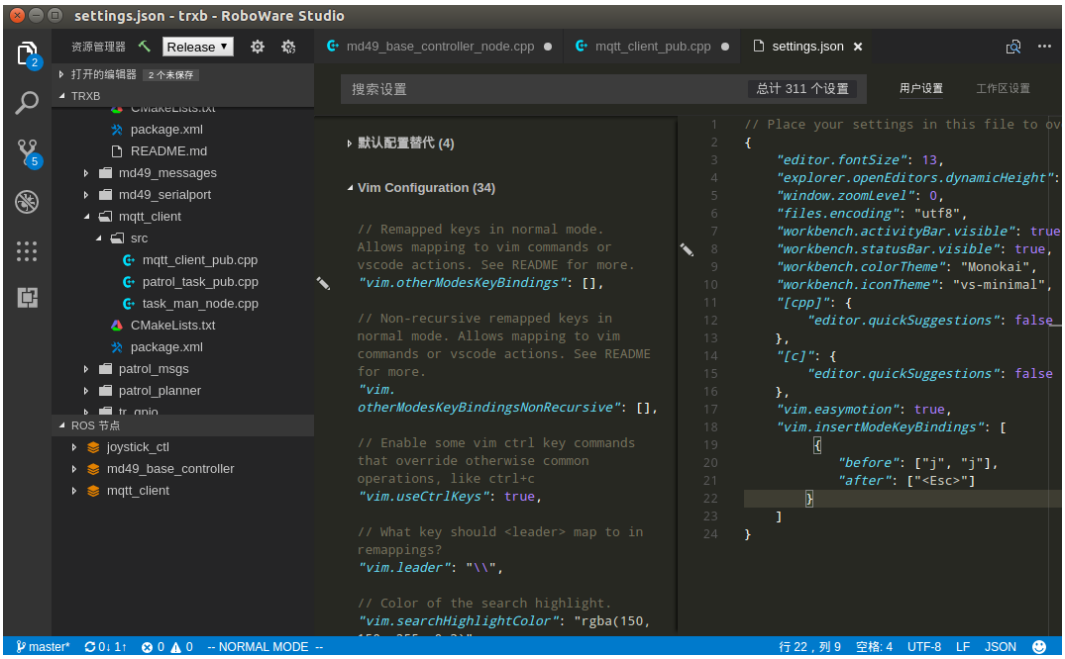


图 3-59 Vim 参数设置窗口

例如，将其中的“vim.easymotion”选项设置为“true”后，即可使用 easymotion 功能（其中，<leader>键的默认值为 “\” ）。

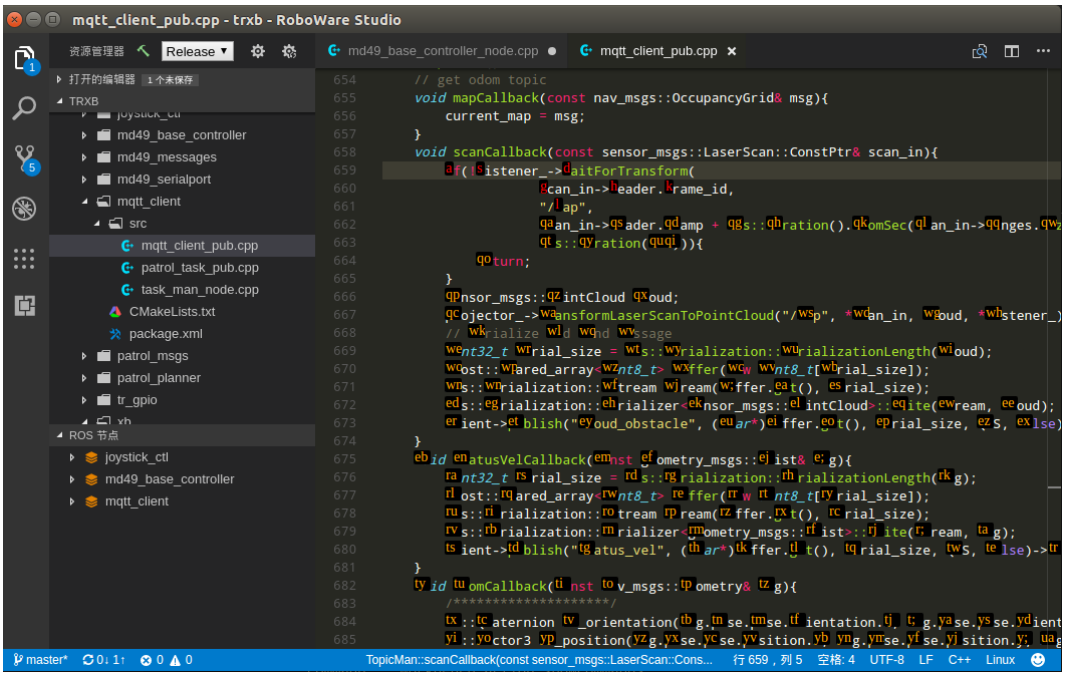


图 3-60 Easymotion 插件功能示例

4 FAQ

4.1 如何导入已有的 ROS 工作区？

分两种情况：

（1）对于普通的 ROS 工作区，直接在欢迎界面点击“打开工作区”按钮（或在菜单中选择“文件 - 打开工作区”），选择工作区路径打开即可。

（2）对于旧版本 RoboWare Studio 打开过的 ROS 工作区，需要将工作区根目录下的“.vscode”文件夹删除，再打开工作区即可。

4.2 如何进行软件升级？

下载最新的 RoboWare Studio 安装包 deb 文件，参照安装步骤直接安装即可，旧版本会被自动覆盖。

4.3 如何修改界面语言？

在菜单栏中，选择“文件-首选项-语言设置”，打开配置文件。

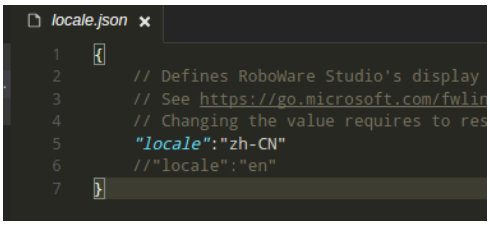


图 4-1 语言设置配置文件

"locale":"zh-CN"表示设置为中文界面，

"locale":"en"表示设置为英文界面，

可用“//”进行注释。

修改完成后，重启 RoboWare Studio 即可生效。

4.4 新建工作区时提示“路径不是 ROS 工作区”。

可能原因为 ROS 环境变量未导出。安装完 ROS 后，需要在当前用户家目录下的.bashrc 文件中添加以下内容：

source /opt/ros/indigo/setup.bash （indigo 版）

或

source /opt/ros/kinetic/setup.bash （kinetic 版）

ROS 的安装及配置说明可参照官网教程：

http://wiki.ros.org/indigo/Installation/Ubuntu （indigo 版）

http://wiki.ros.org/kinetic/Installation/Ubuntu （kinetic 版）

4.5 提示错误“Linter pylint is not installed”。

需要安装 pylint，请参照本手册“软件安装”-“准备”中的步骤。打开命令行终端，执行以下命令：

$ sudo apt-get install python-pip

在集成终端中安装 python 插件时会提示输入 root 密码，请按照提示输入。

4.6 提示系统 git 版本低的问题。

需要升级 git。打开命令行终端，执行以下命令：

$ sudo apt-add-repository ppa:git-core/ppa

$ sudo apt-get update

$ sudo apt-get install git

4.7 名为 “test” 的节点无法生成问题。

不要将 ROS 节点名称命名为 test，否则会无法生成节点。

4.8 构建过程中卡住问题。

对于内存太小的机器，如果当前工作区内的 ROS 包数量太多，在构建整个工作区时，会因内存不足导致构建卡住。此时，可采用单独构建的方式依次对每个包构建。

4.9 新建、删除文件时资源管理器不能自动刷新。

RoboWare Studio 依靠 ubuntu 的文件系统监视功能实现资源管理器的自动刷新，但 ubuntu 的文件监视数量限制可能设置的太小，导致资源管理器没有自动刷新。为解决这个问题需要设置 ubuntu 的文件监视数量限制。方法如下：

使用 root 权限打开文件/etc/sysctl.conf 进行编辑。

$ sudo vi /etc/sysctl.conf，也可用 gedit 或任意文本编辑器

找到 fs.inotify.max_user_watches 选项，适当增加等号后面的数值，比如改为 100000。重启 ubuntu 即可。

4.10 编辑器无法输入、选择、复制问题。

这是因为 RoboWare Studio 处于 Vim 编辑模式，切换到普通模式即可，在菜单栏选择“编辑-切换 VIM 编辑模式”可切换到普通编辑模式。

4.11 编辑时如何进行前进、后退？能否自定义其快捷键？

默认的前进快捷键为 “Ctrl+Shift+-" ，后退快捷键为“Ctrl+Alt+-”。可以进行快捷键自定义，点击菜单栏的“文件 - 首选项 – 键盘快捷方式”，找到“前进”、“后退”选项进行修改即可，也可对其它快捷键进行自定义。

4.12 CMakelists.txt 错误无法定位问题。

先删除工作区根目录.vscode 文件夹下的 tasks.json 文件，然后重新打开工作区。

4.13 Meta Package 无法编辑依赖、无法新建节点问题。

目前尚不支持 meta package 编辑依赖和新建节点，将 meta package 下的所有包拷贝到 src 目录下即可。