

# RoboWare Designer 手册

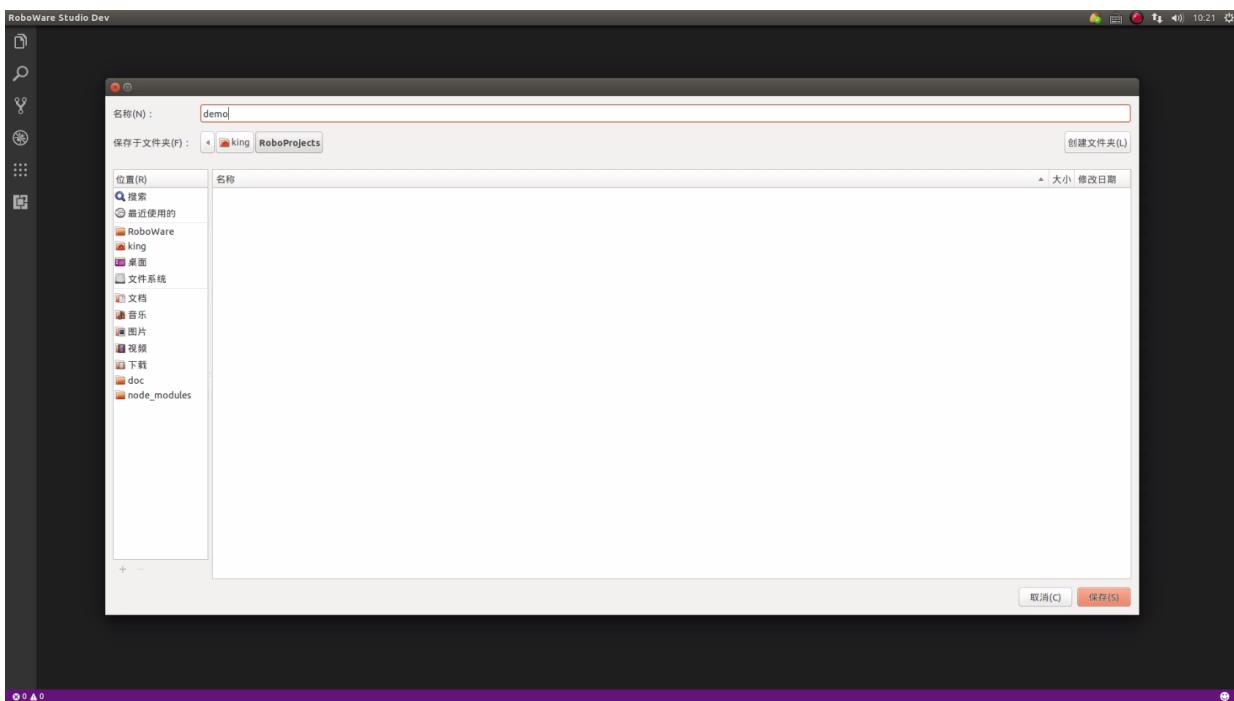
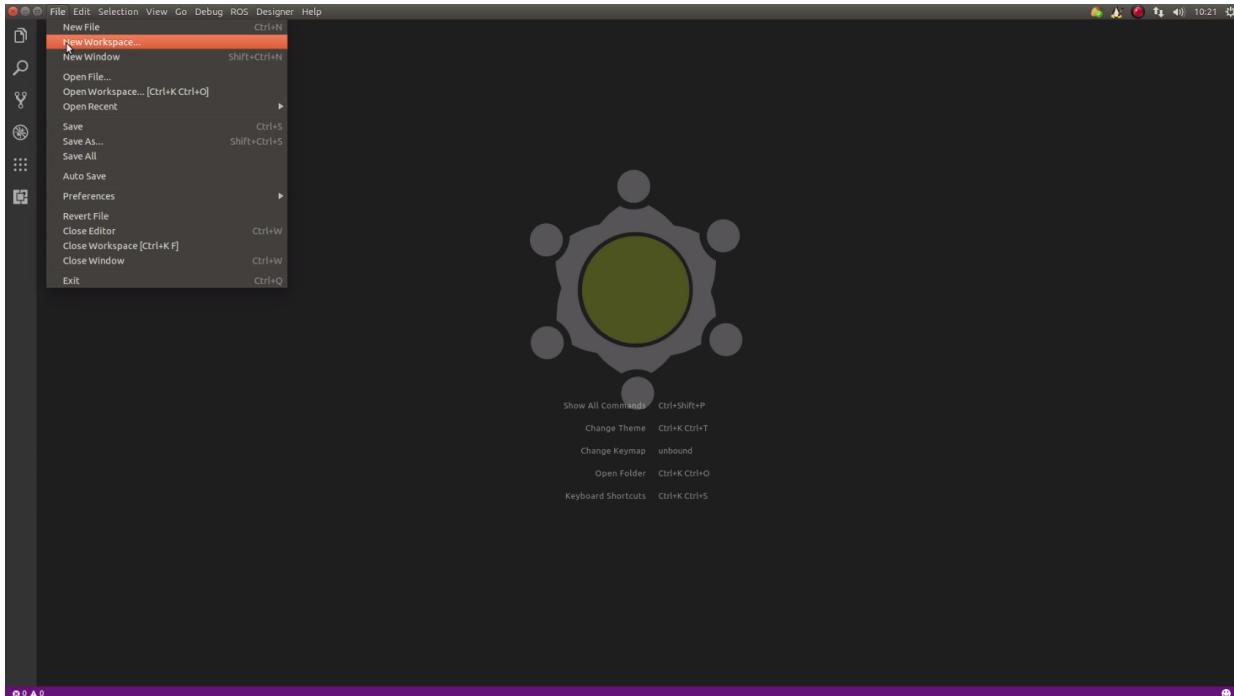
RoboWare Designer 是 RoboWare Studio 推出的 ROS 图形化开发工具。在 RoboWare Designer 中，基于 ROS 的硬件和这些硬件的软件以可视化的方式表示，并且允许你拖动它们、配置它们、编辑它们，使用图表示它们的联系。当你完成这个“游戏”过程，RoboWare Designer 会为你生成一个完全的 ROS workspace — ROS 工程文件树。

RoboWare Designer 使你在开发的初期，以可视化的角度来考虑整个 ROS 工程，并且准备你需要的各个硬件和软件资源，当你一些考虑就绪之后，RoboWare Designer 为你准备好你预定的各项 ROS 资源设施。

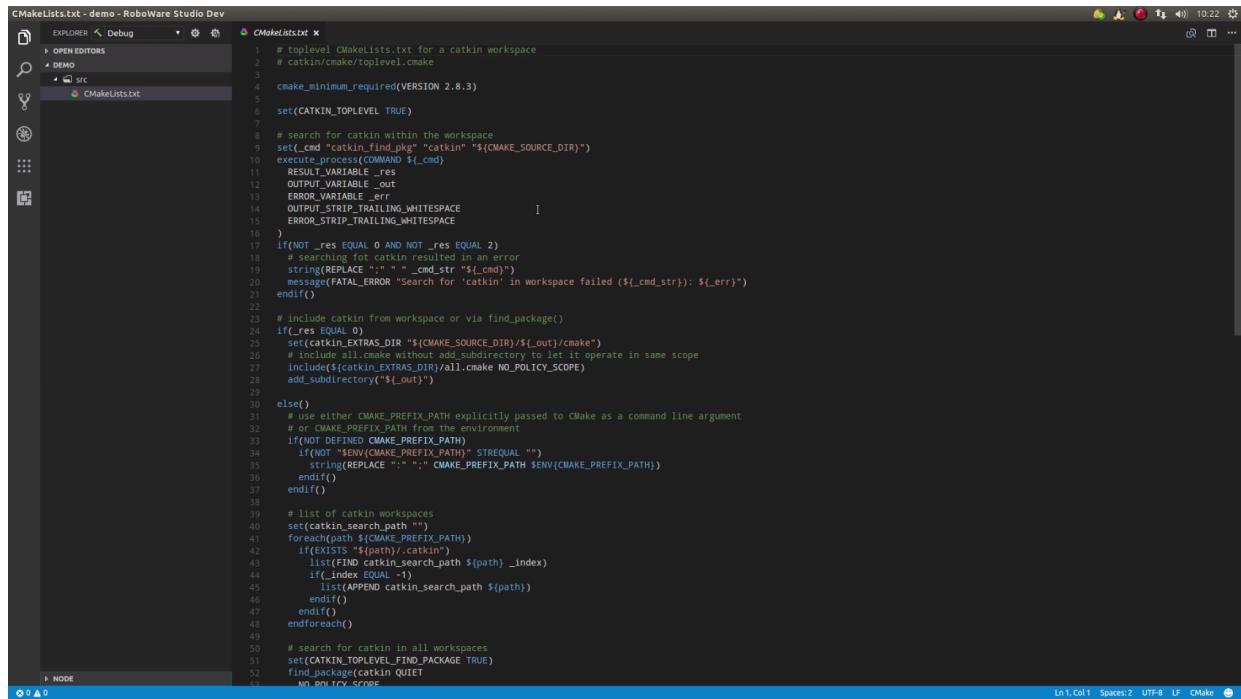
本章主要介绍 RoboWare Designer 的组成和使用指南。

# 1. 创建一个新的工作区

打开 RoboWare Studio , 点击 File 菜单 → New Workspace... , 输入工作区的名字 , 创建一个新的工作区。我们在这里给工作区命名为 demo , 你可以按照自己的需要为工作区命名 :



新的工作区只有一个 **src** 目录，里面有一个 **CMakeLists.txt** 文件。这样的工作区，我们称之为“纯净的”，能够成为 Designer 的生成目标。接下来，我们通过 Designer 为这个工作区创建需要的 ROS 工程包，以建造我们的 ROS 基础设施：

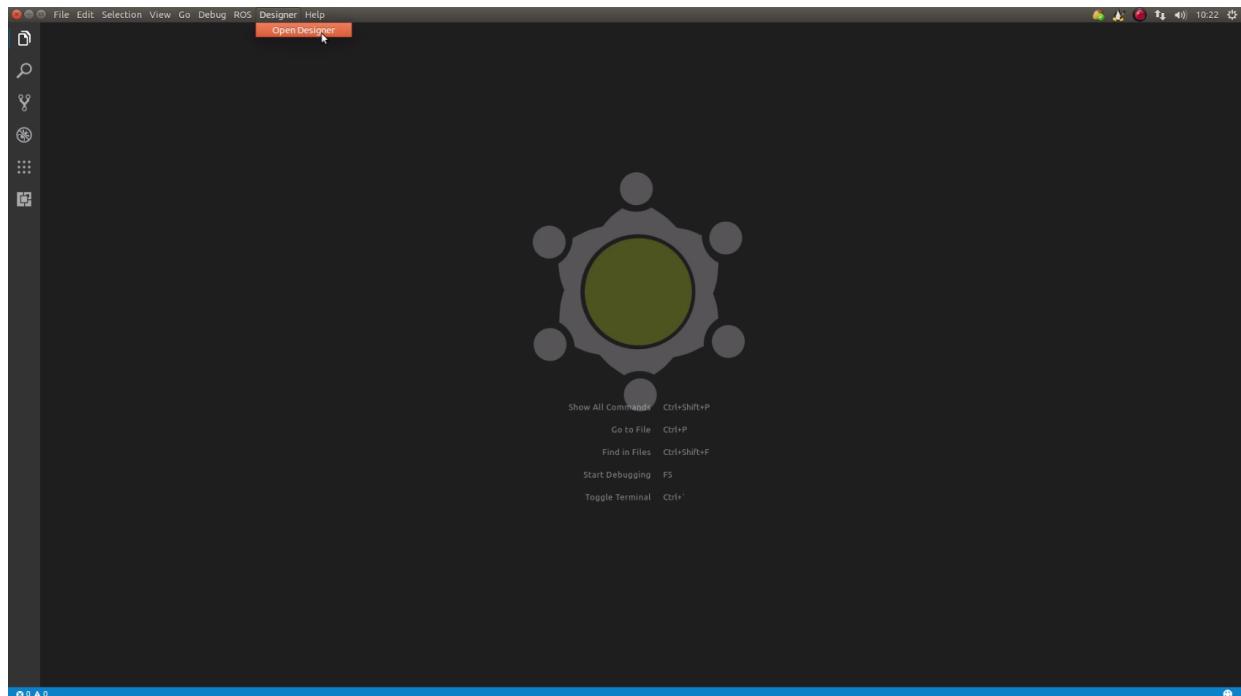


The screenshot shows the RoboWare Studio Dev interface with the CMakeLists.txt file open in the editor. The code is a standard CMake script for a catkin workspace, defining variables like CMAKE\_MINIMUM\_REQUIRED, setting CMAKE\_PREFIX\_PATH, and searching for catkin within the workspace. It also includes logic for finding catkin in the environment or workspace, and for including all .cmake files in subdirectories.

```
1 # top level CMakeLists.txt for a catkin workspace
2 # catkin/cmake/toplevel.cmake
3
4 cmake_minimum_required(VERSION 2.8.3)
5
6 set(CATKIN_TOPLEVEL TRUE)
7
8 # search for catkin within the workspace
9 set(cmake_catkin_find_pkg "catkin" "${CMAKE_SOURCE_DIR}")
10 execute_process(COMMAND ${_cmd}
11     RESULT_VARIABLE _res
12     OUTPUT_VARIABLE _out
13     ERROR_VARIABLE _err
14     OUTPUT_STRIP_TRAILING_WHITESPACE
15     ERROR_STRIP_TRAILING_WHITESPACE
16 )
17 if(NOT _res EQUAL 0 AND NOT _res EQUAL 2)
18     # searching for catkin resulted in an error
19     string(REPLACE ":" ";" _cmd_str "${_cmd}")
20     message(FATAL_ERROR "Search for 'catkin' in workspace failed (${_cmd_str}): ${_err}")
21 endif()
22
23 # include catkin from workspace or via find_package()
24 if(_res EQUAL 0)
25     set(catkin_EXTRAS_DIR "${CMAKE_SOURCE_DIR}/$/_out/cmake")
26     # include all .cmake without add_subdirectory to let it operate in same scope
27     include(${catkin_EXTRAS_DIR}/all.cmake NO_POLICY_SCOPE)
28     add_subdirectory("${_out}")
29
30 else()
31     # use either CMAKE_PREFIX_PATH explicitly passed to CMake as a command line argument
32     # or CMAKE_PREFIX_PATH from the environment
33     if(CMAKE_PREFIX_PATH)
34         if(NOT "env(CMAKE_PREFIX_PATH)" STREQUAL "")
35             string(REPLACE ":" ";" CMAKE_PREFIX_PATH ${ENV(CMAKE_PREFIX_PATH)})
36         endif()
37     endif()
38
39     # list of catkin workspaces
40     set(catkin_search_path "")
41     foreach(path ${CMAKE_PREFIX_PATH})
42         if(EXISTS "${path}/catkin")
43             list(FIND catkin_search_path ${path} _index)
44             if(_index EQUAL -1)
45                 list(APPEND catkin_search_path ${path})
46             endif()
47         endif()
48     endforeach()
49
50     # search for catkin in all workspaces
51     set(CATKIN_TOPLEVEL_FIND_PACKAGE TRUE)
52     find_package(catkin QUIET
53         NO_POLICY_SCOPE
54     )
55 
```

## 2. 打开 RoboWare Designer

点击 Designer 菜单栏 → Open Designer 打开 Designer：



Designer 由 4 部分组成：

- **工具箱**

位于左侧，里面存放着各种部件的原型

- **画布**

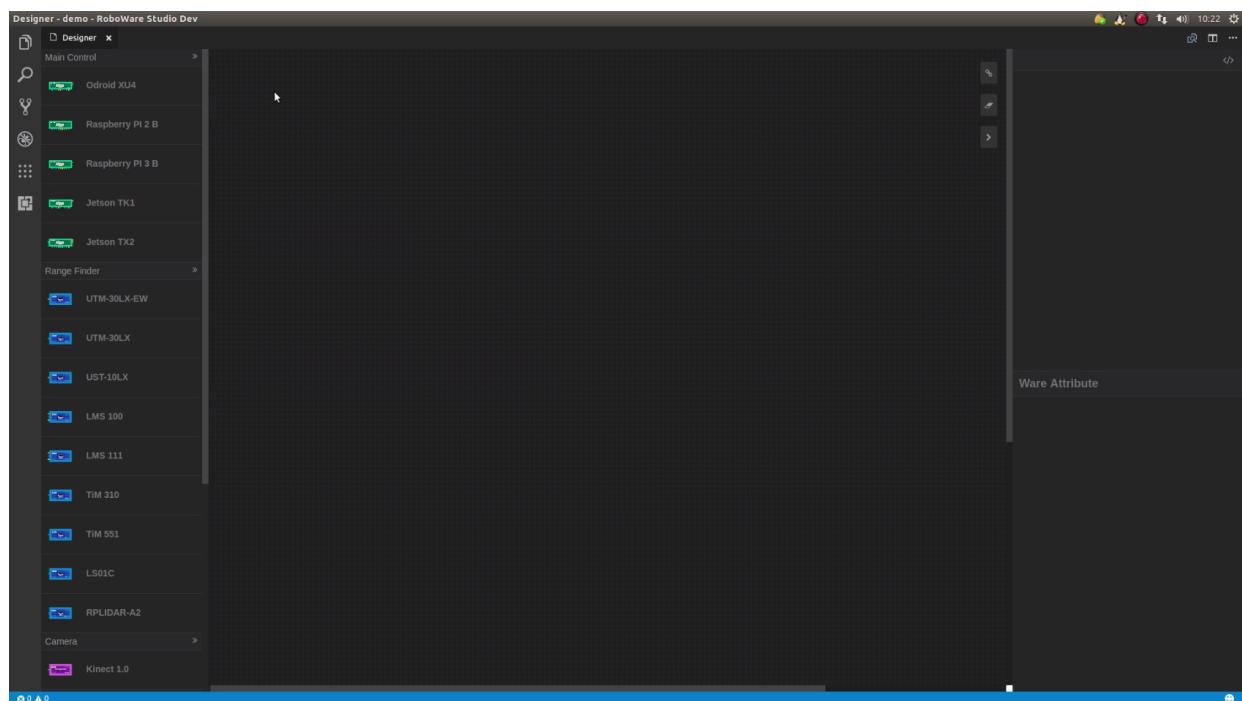
位于中间，用来放置你需要的部件，可以连接部件之间的端口、删除连线、删除部件、查看部件信息

- **注册栏**

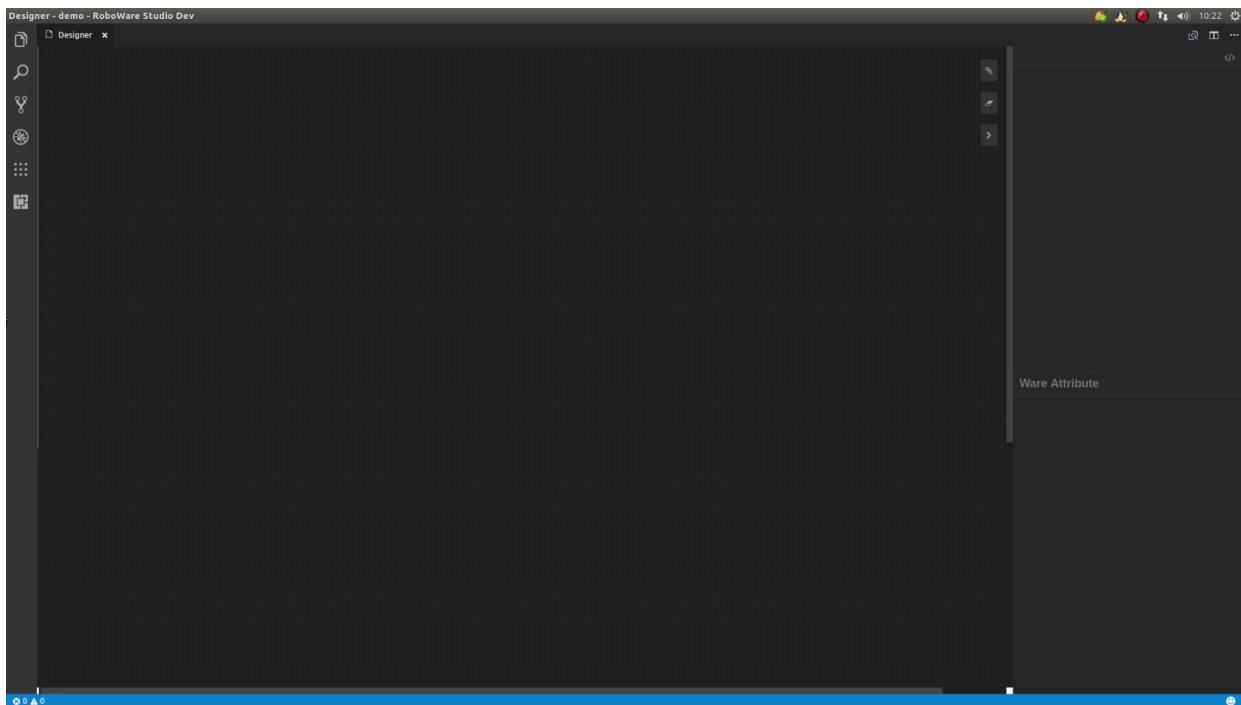
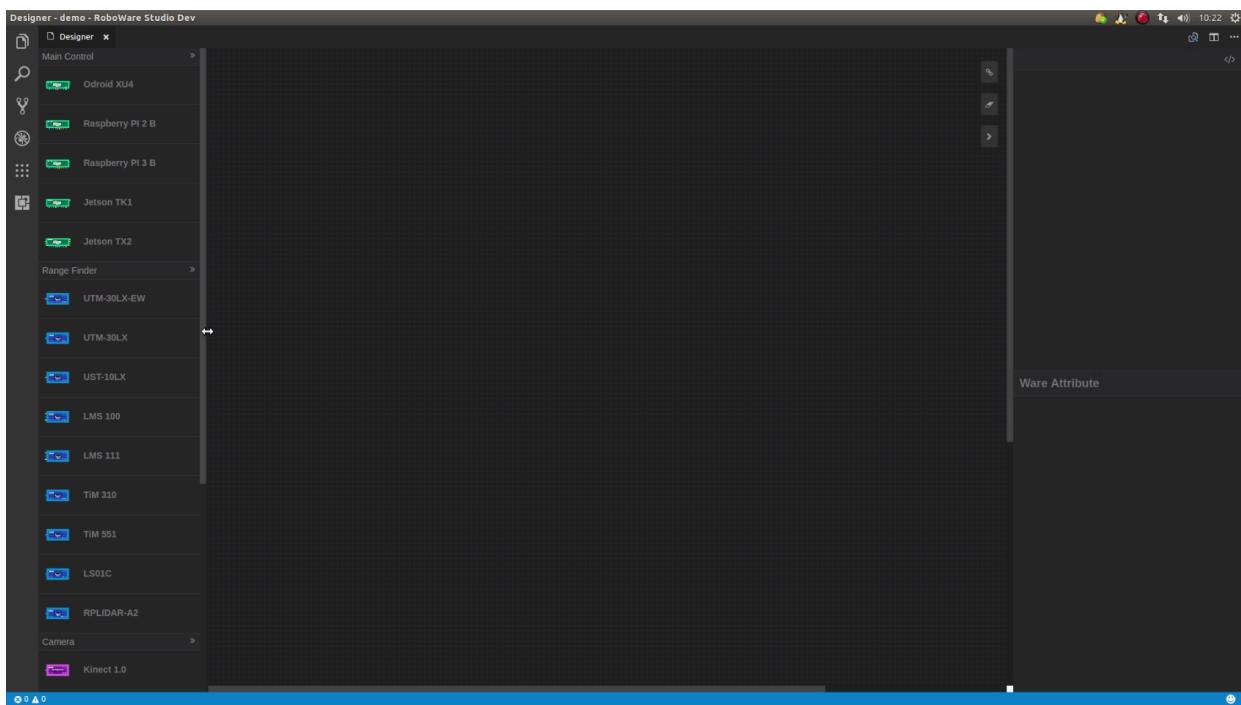
位于右上角，用来索引画布中各个部件，便于查看部件

- **属性栏**

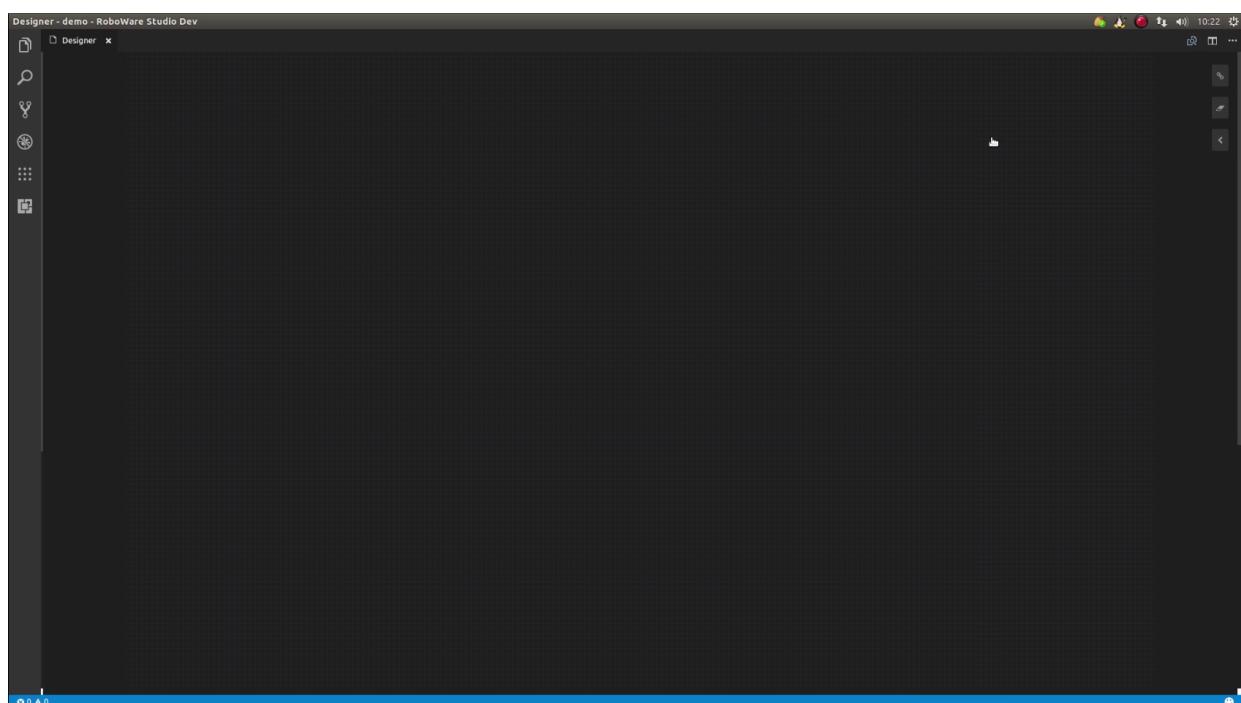
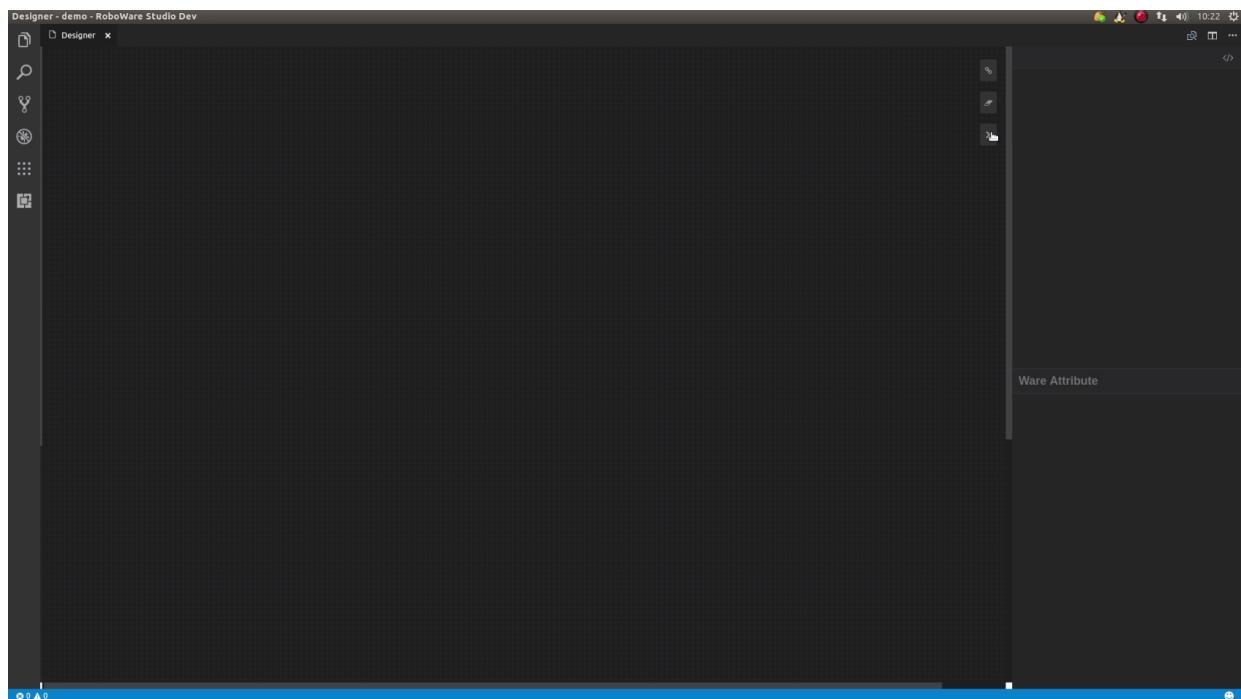
位于右下角称为，在这里可以查看画布中各个部件的属性信息



工具箱可以收缩到左边，为画布提供更大的视野：

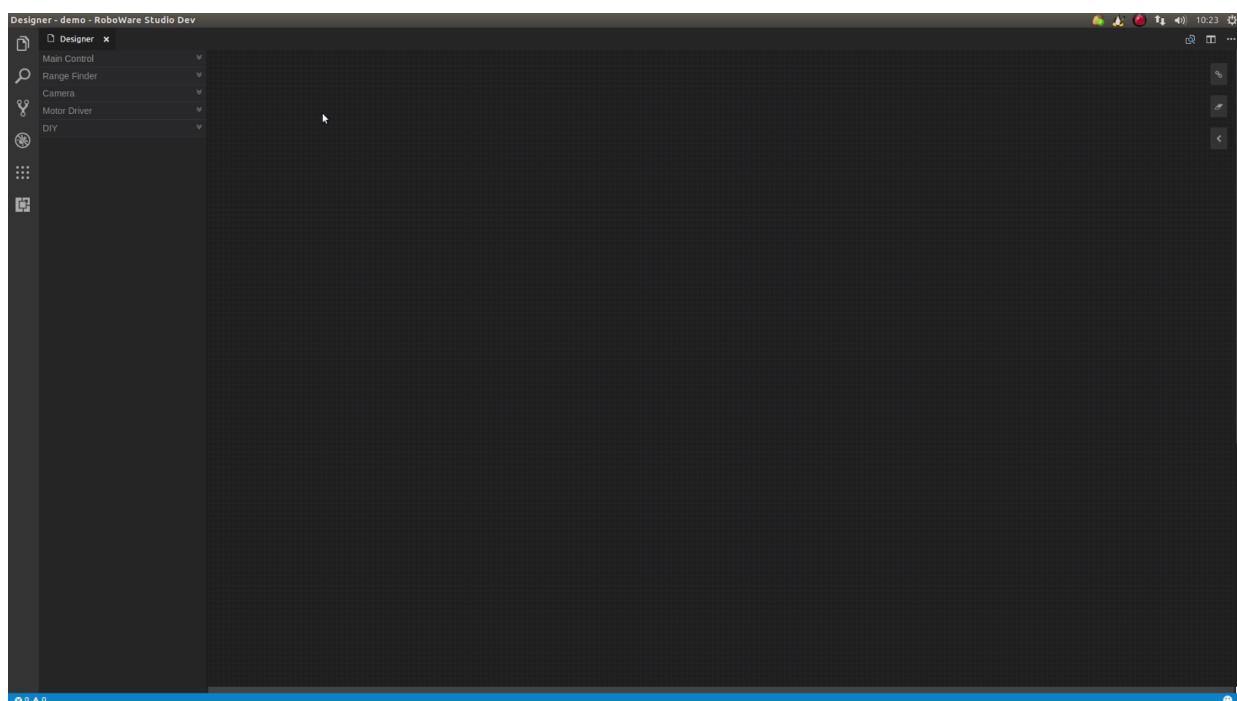
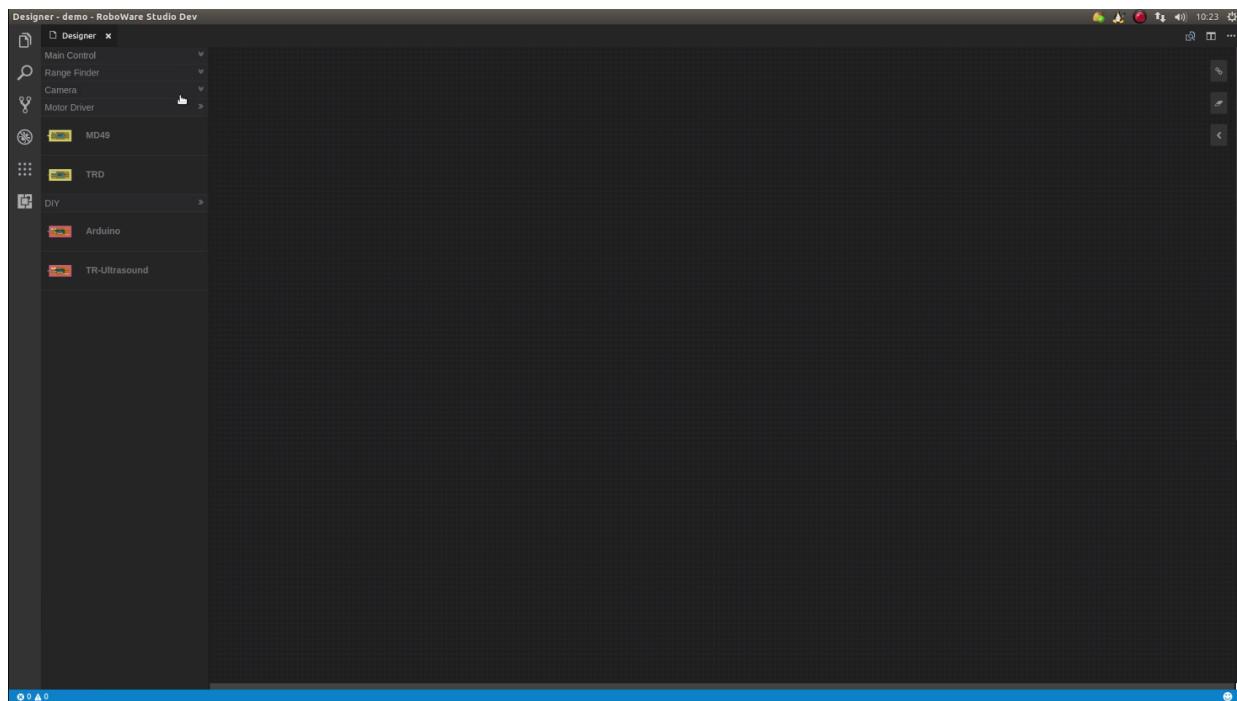


画布右上角有三个按钮，点击第3个按钮可以把注册栏和属性栏收起来，为画布提供更大的视野：



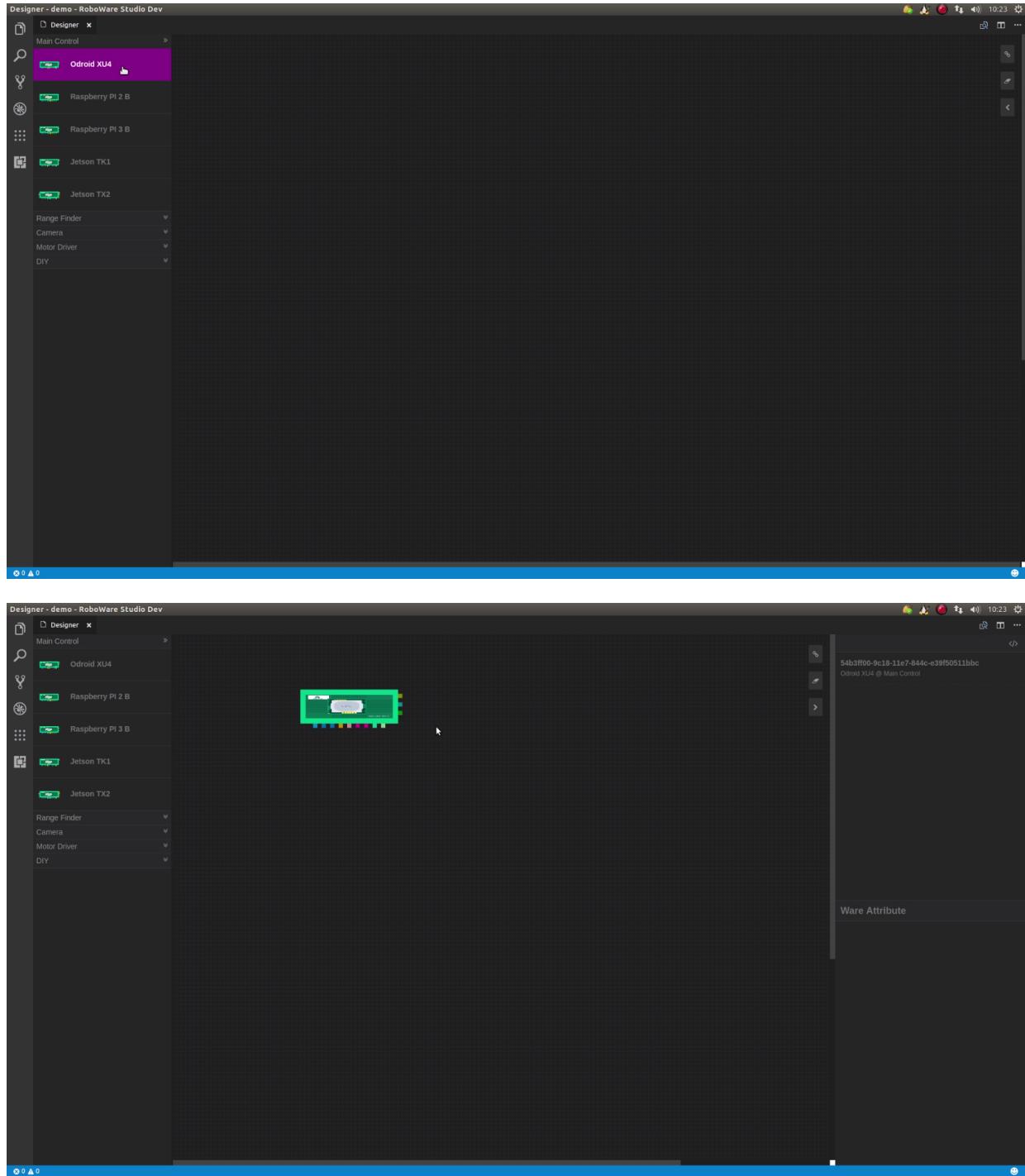
工具箱有 4 种部件：分别是 Main Control (主控板卡)、Range Finder (激光测控仪)、Camera (照相机)、DIY。

这些不同分类的部件可以折叠，以方便的选择部件：

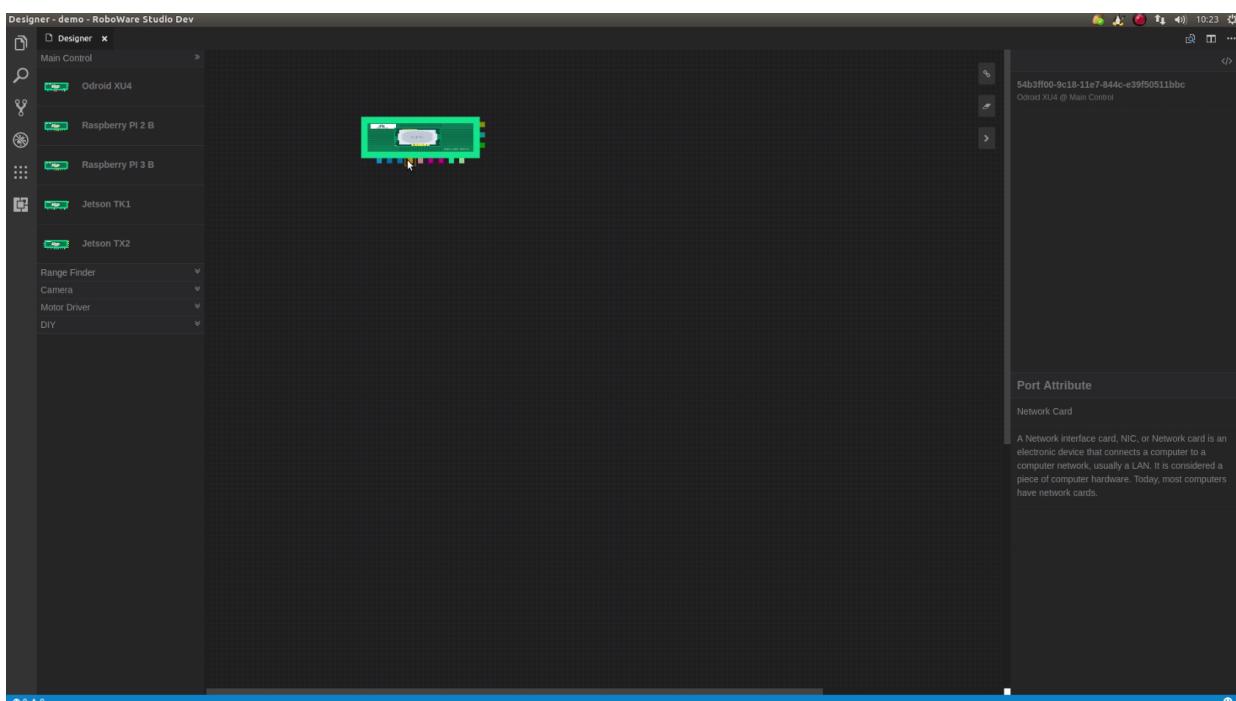
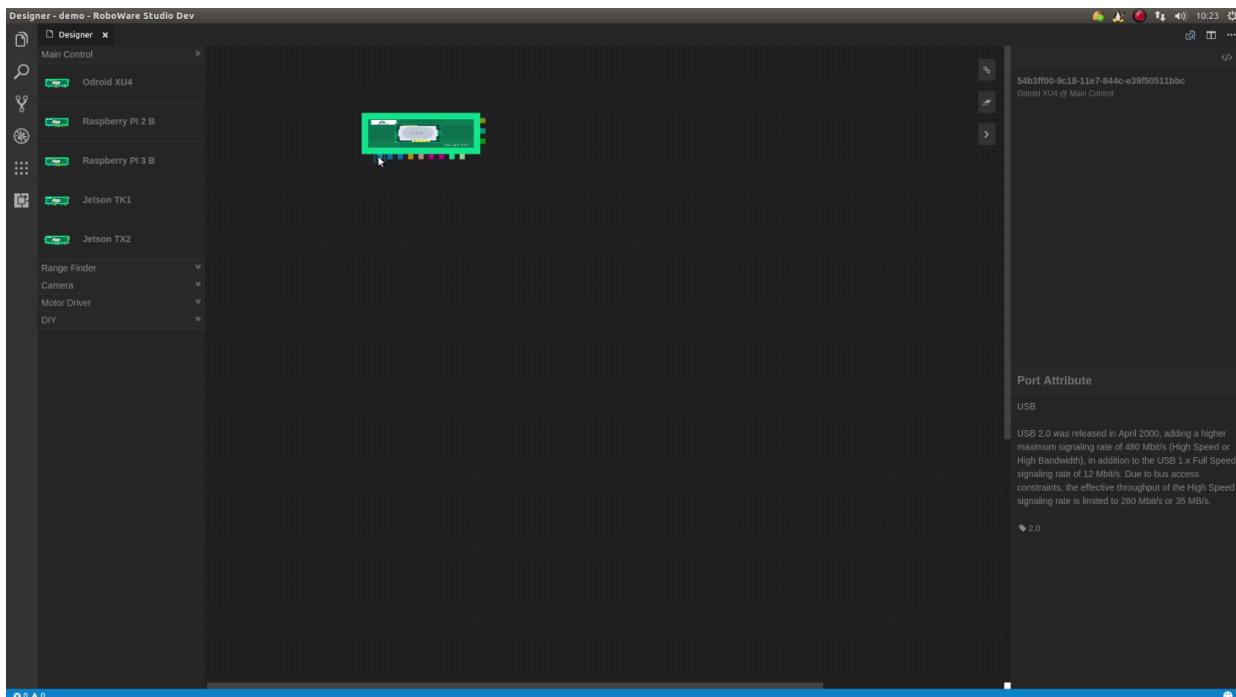


### 3. 创建部件和查看部件属性

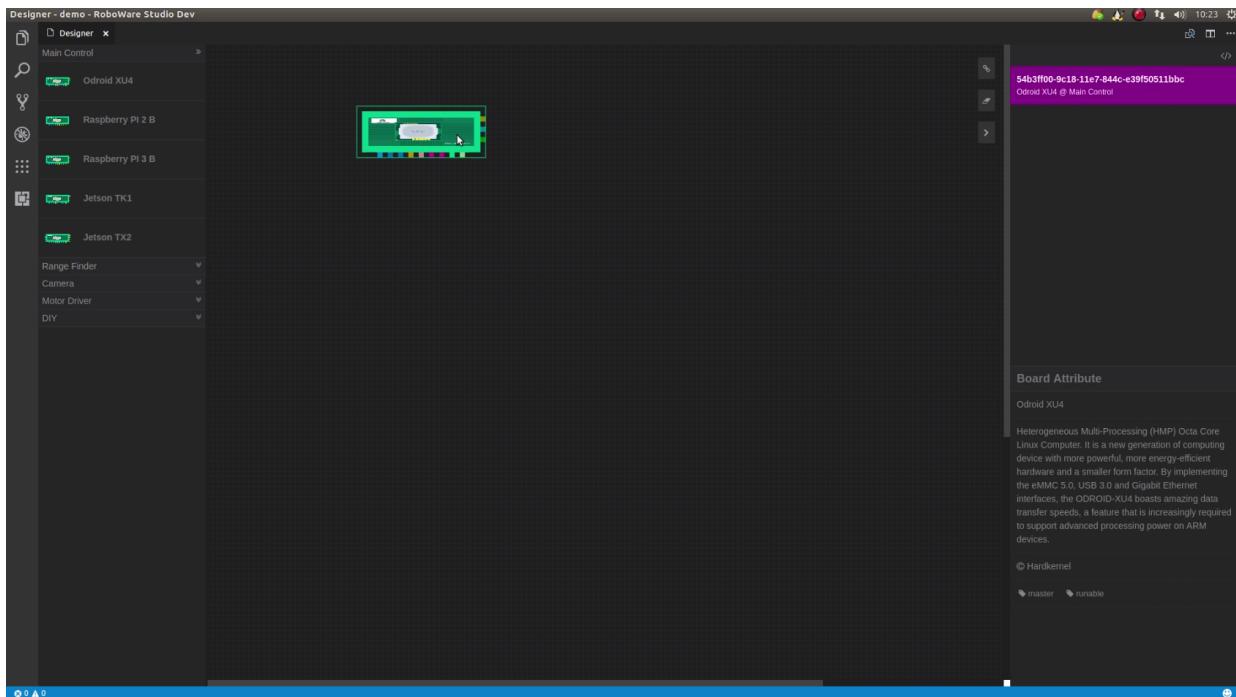
点击工具箱上的部件，然后在画布上点击，一个新的部件就会添加到画布上，同时注册栏会登记这个部件：



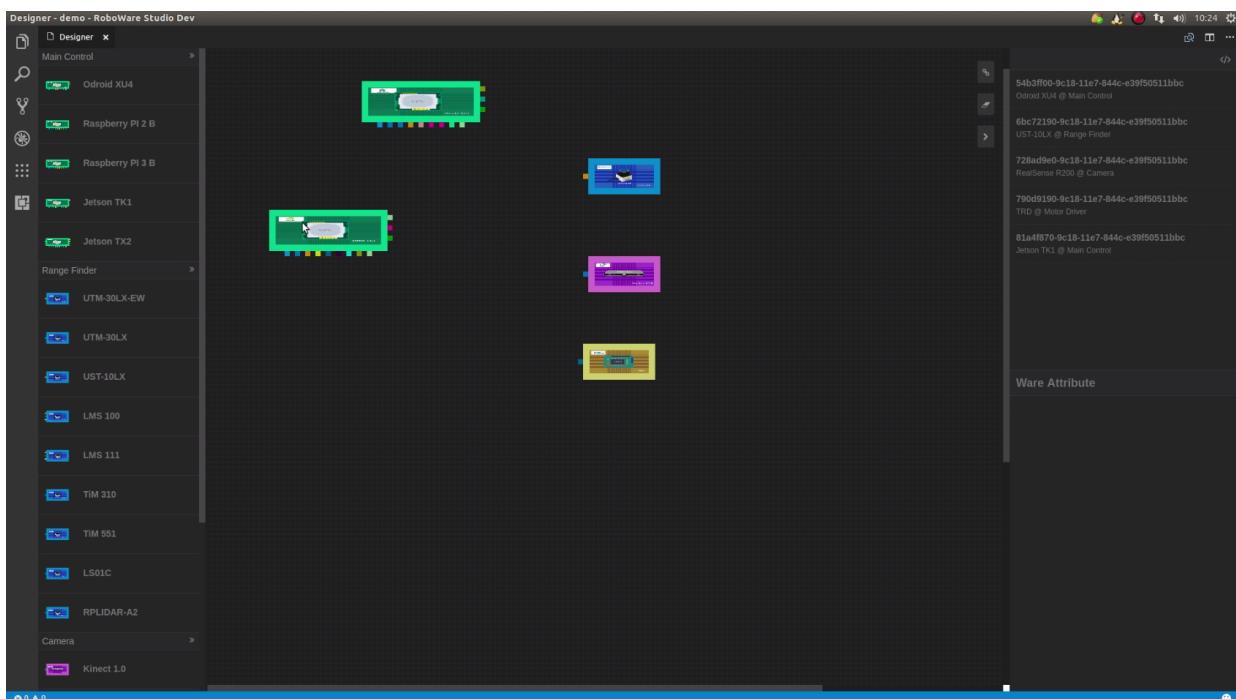
点击部件本身和部件上的端口，可以在属性栏看到它们的属性：



部件目前有 5 个属性：名字、描述信息、开发厂商、ROS 驱动包、标签：

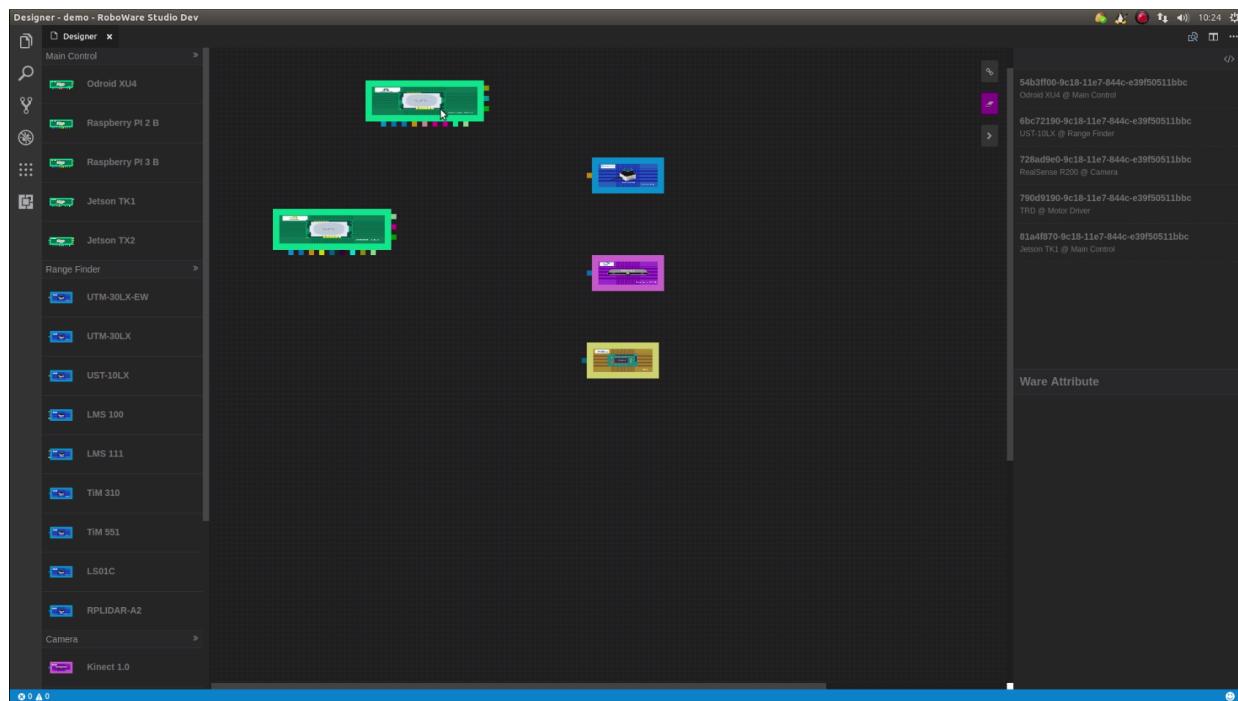
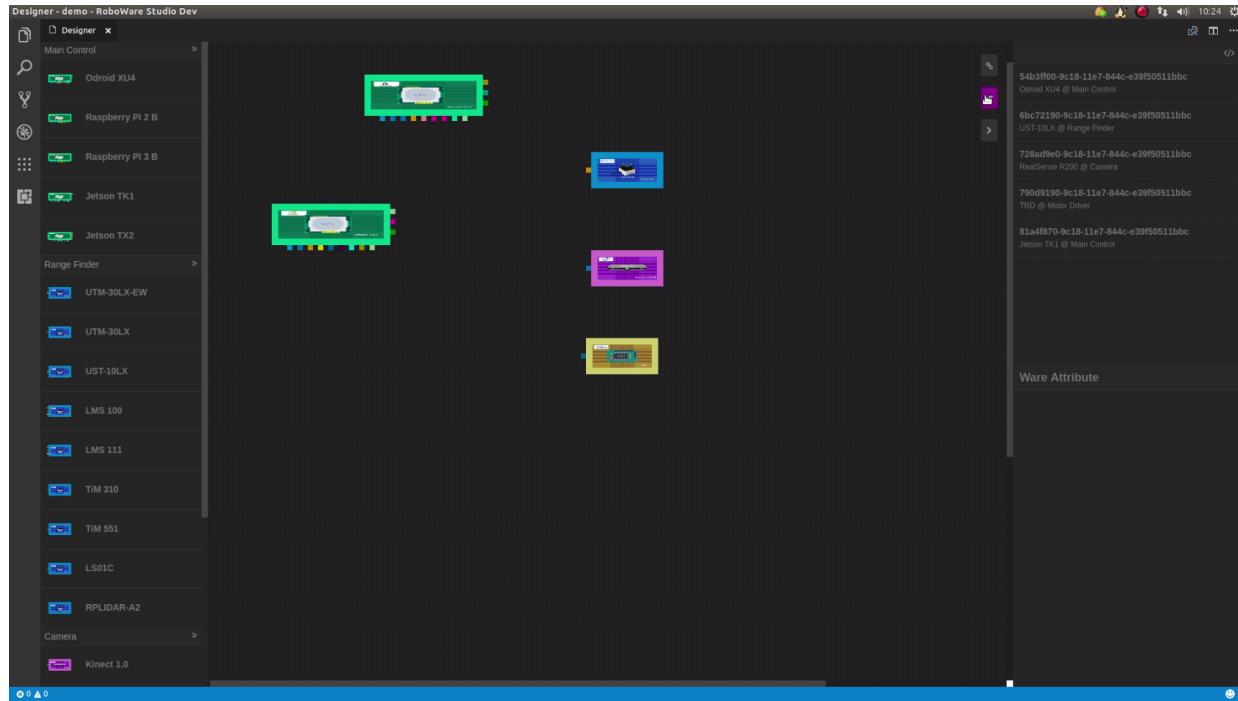


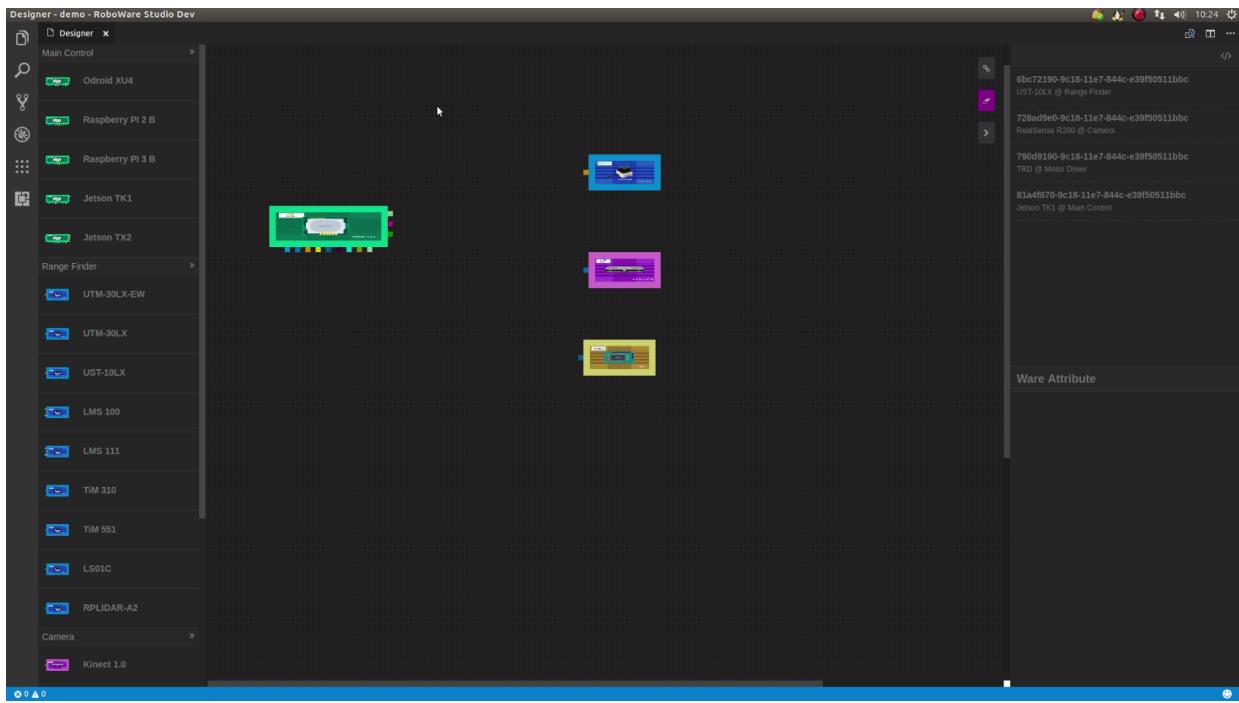
你可以不停地从工具箱拖动部件到画布上，每个部件都附带着其依赖的 ROS 软件包（ROS Node），Designer 会为你自动下载并且配置这些软件包：



# 4. 删除部件和端口连接线

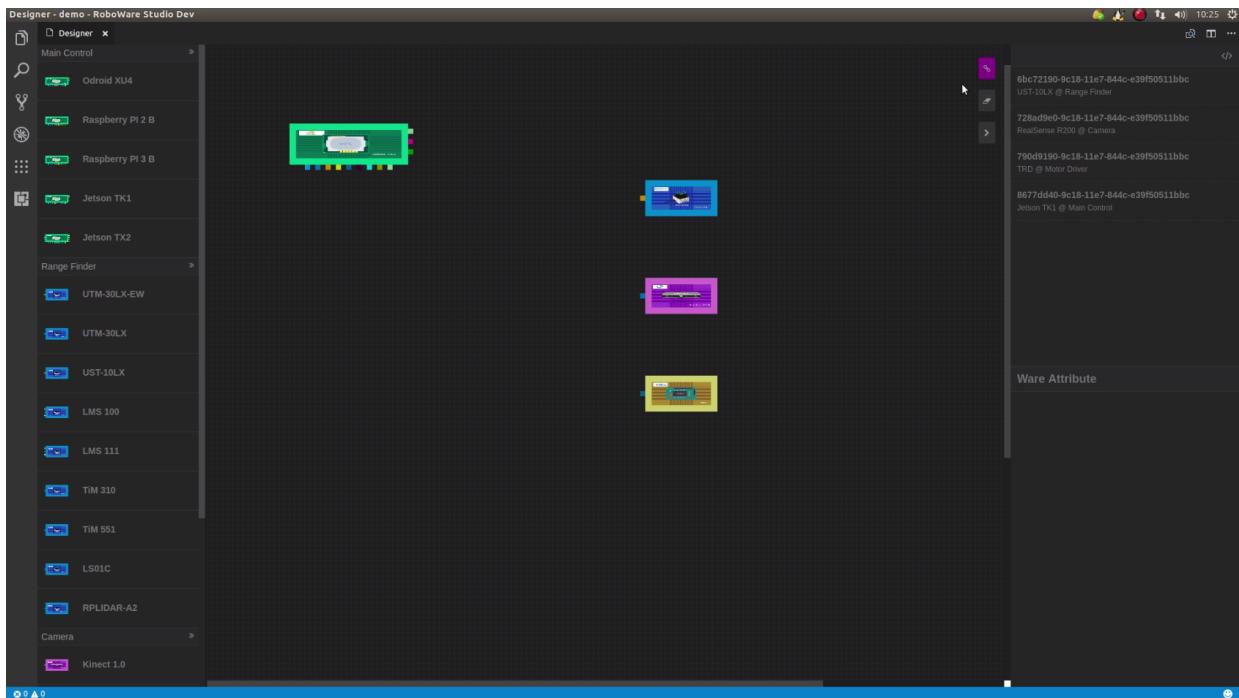
点击画布右上角的第二个菜单按钮，这时候就打开了删除的功能。当鼠标捕捉到部件的时候，点击鼠标就会删除这个部件，部件上端口的连线也会全部删除。当鼠标捕捉到部件的端口时，只删除该端口上的连接线：

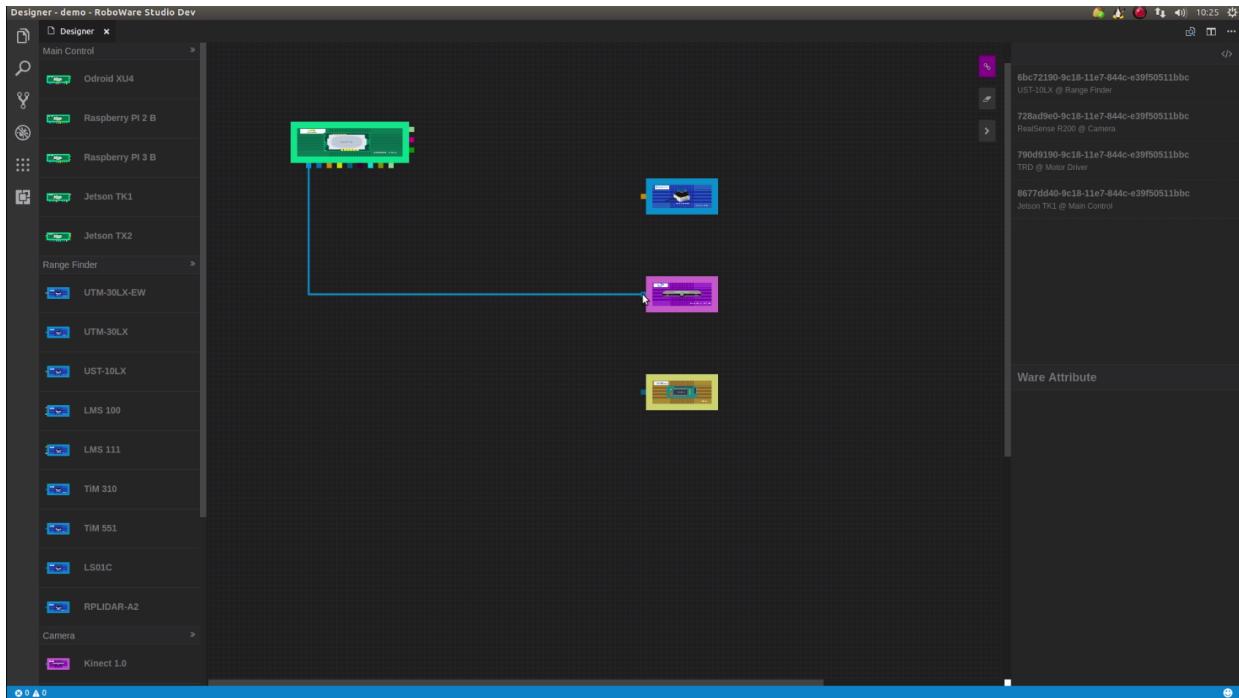
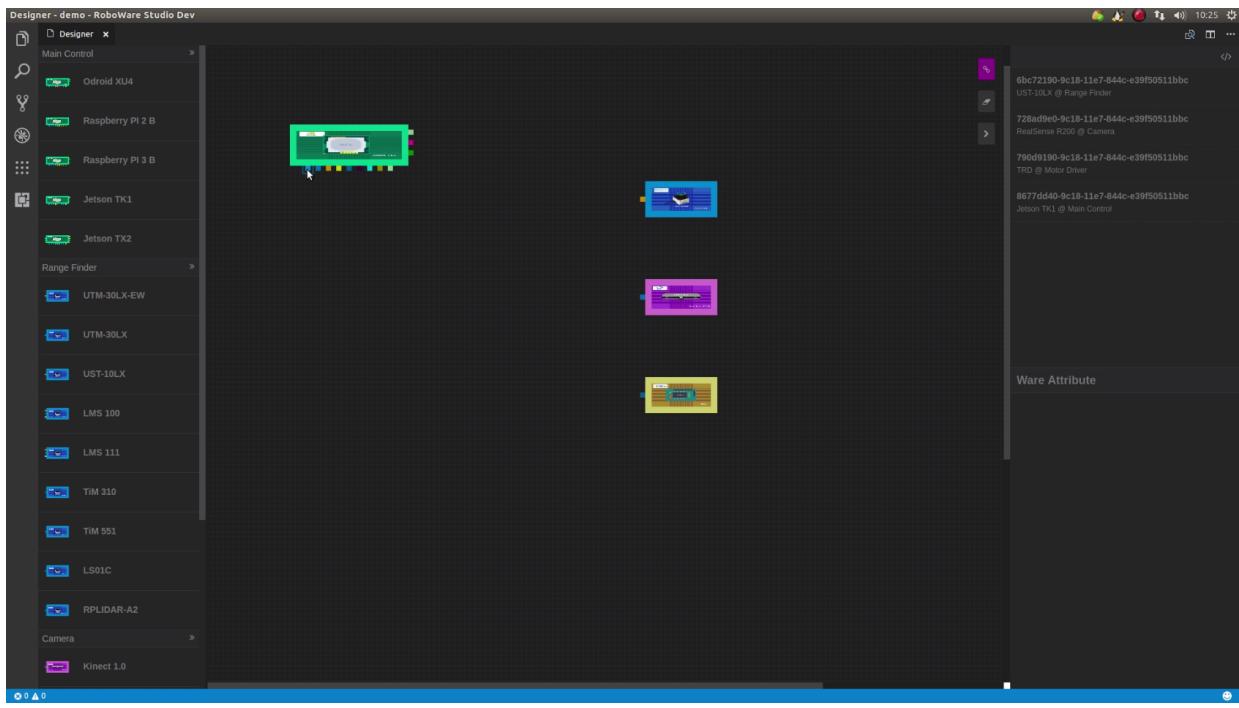


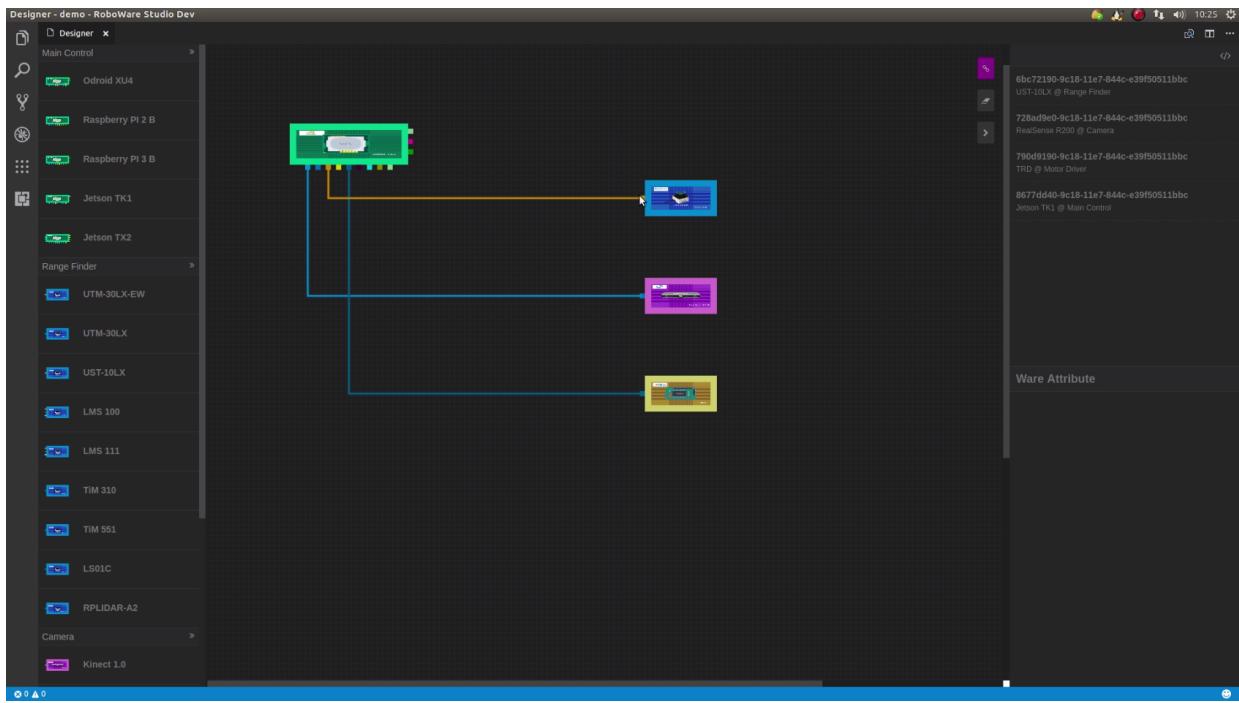


## 5. 连接端口

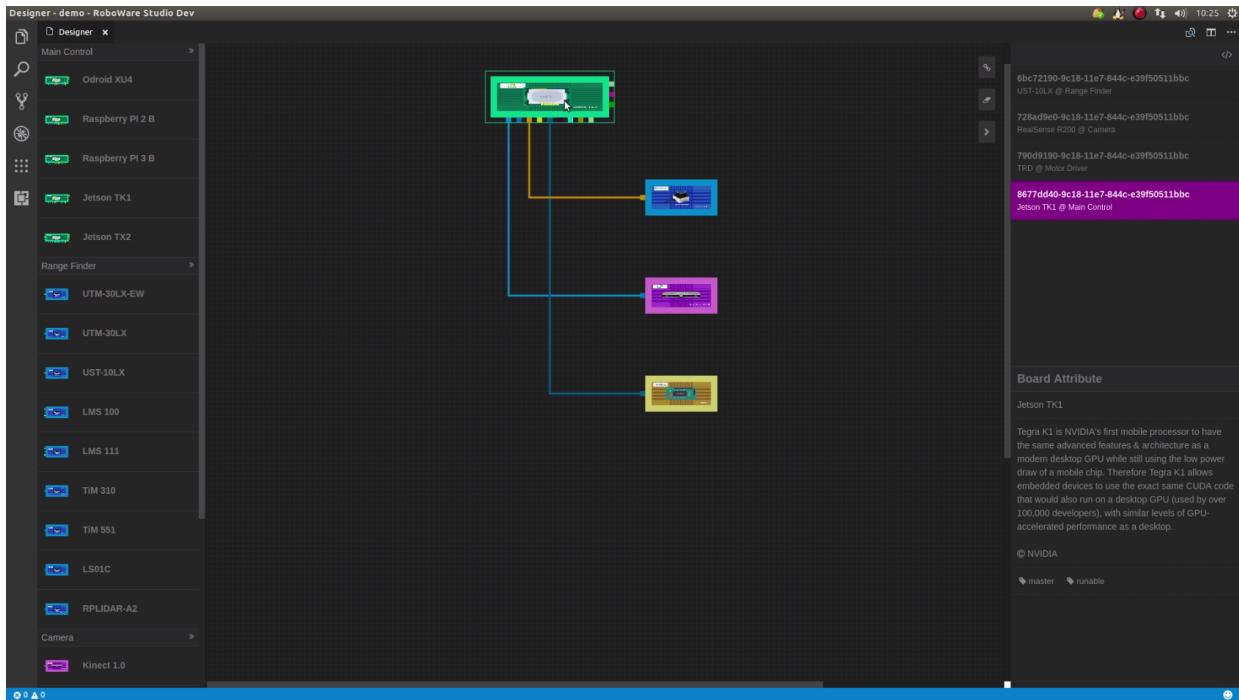
点击画布右上角的第二个菜单按钮，这时候就打开了连接的功能。点击一个部件的端口，再点击另一个部件相同的端口，这两个端口就可以连接在一起：

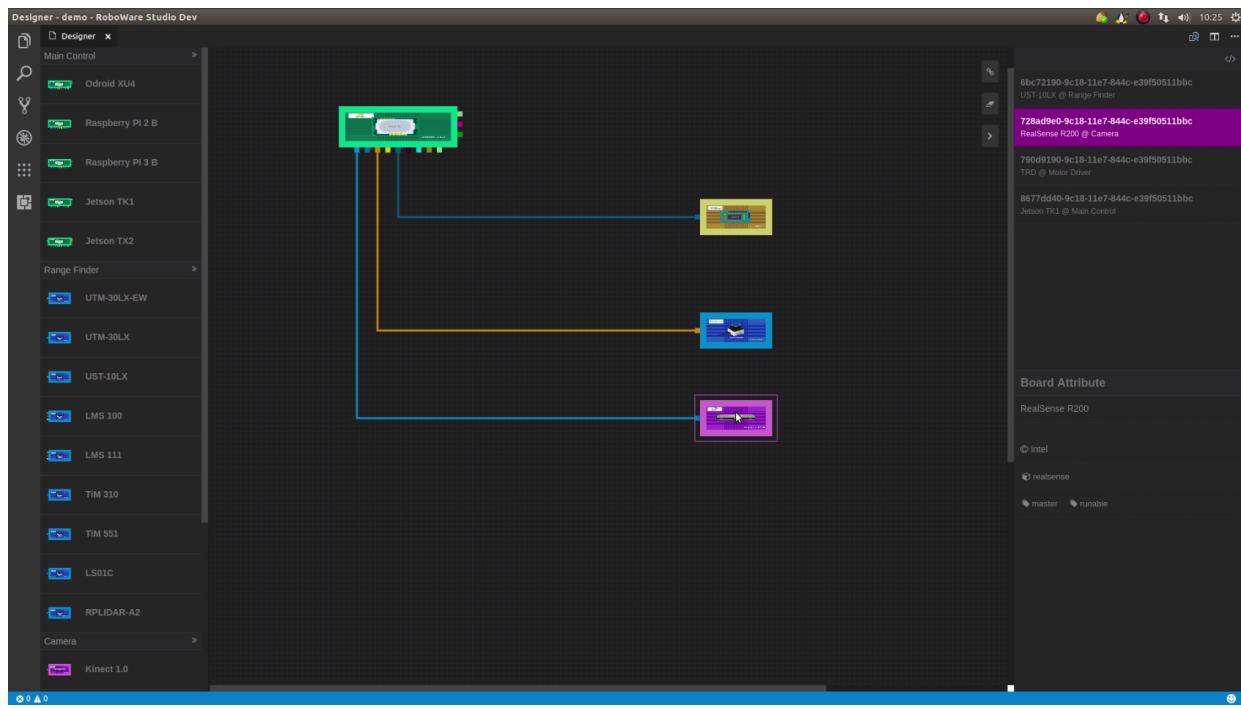






部件之间的连接线是动态的，当你移动部件的时候，这些连接线可以聪明的自动调整：



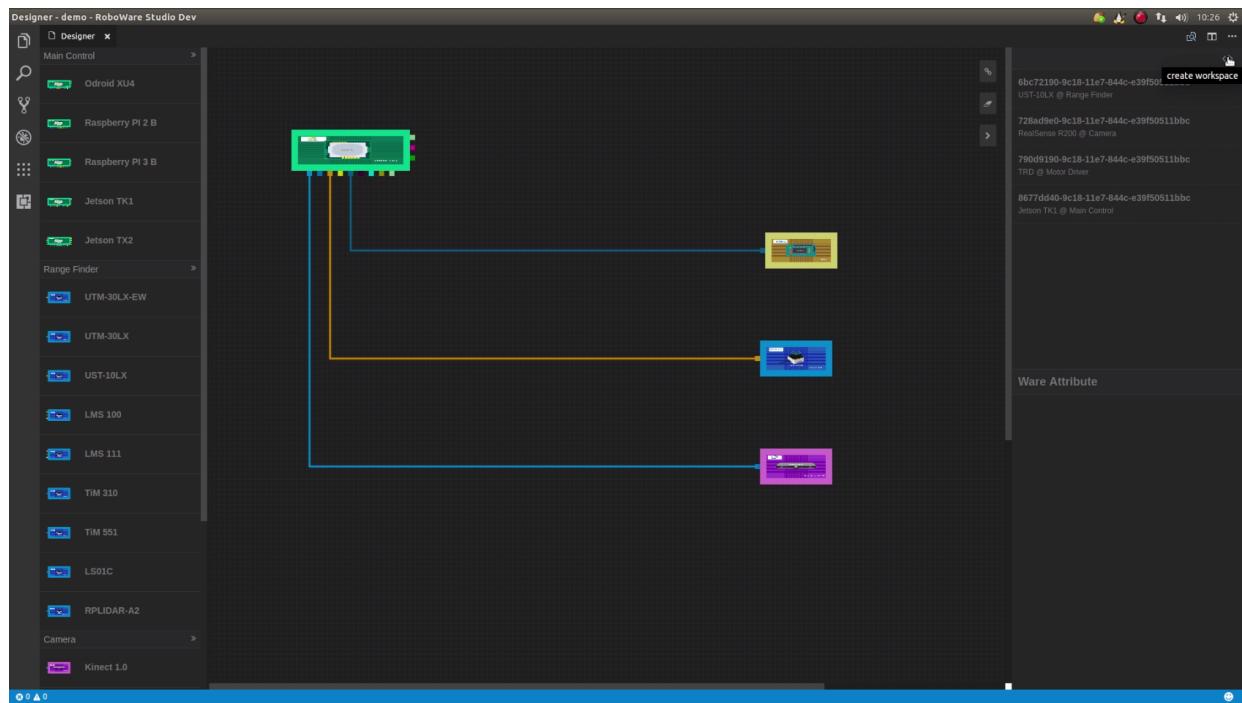


## 6. 为工作区创建 ROS 软件包

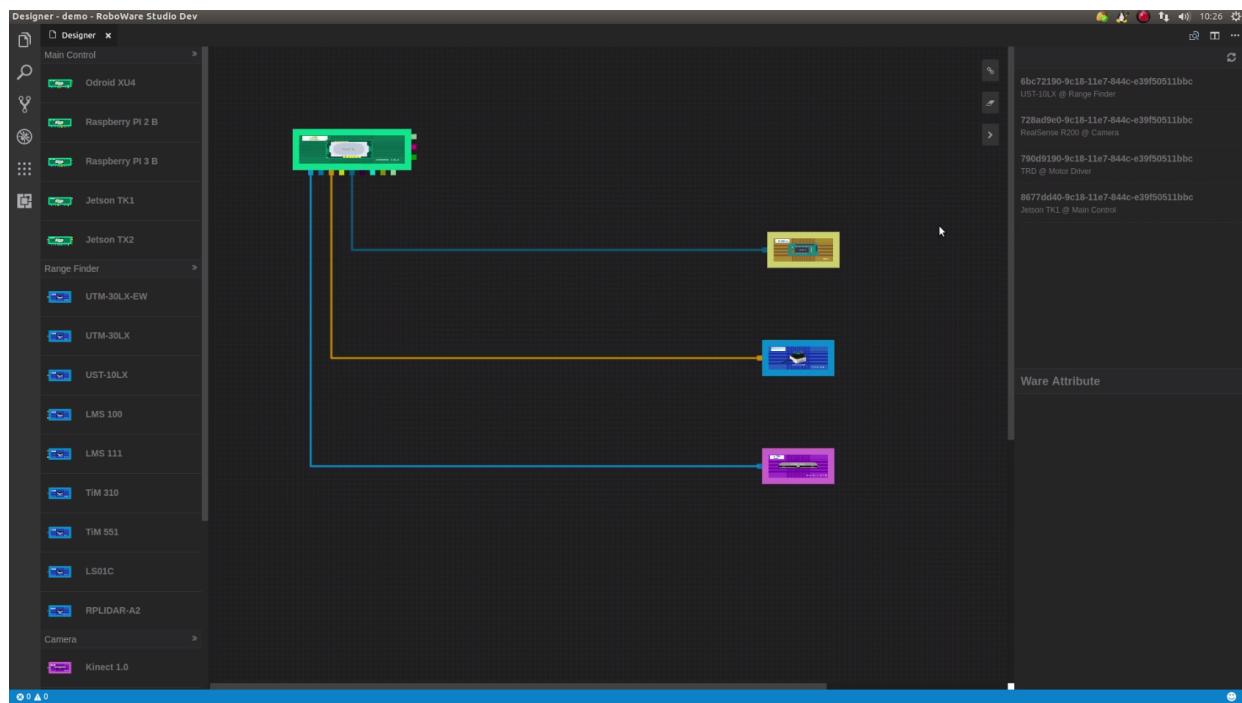
当你在画布放置完成自己的部件后，就可以开始为工作区创建 ROS 软件包了。我们在这里拖动了 4 个部件，分别是：

分类	名字	开发厂商	依赖 ROS 软件包
主控板卡	Jetson TK1	NVIDIA	
相机	RealSense R200	Intel	realsense
激光扫描仪	UST-10LX	Hokuyo	urg_node
电机驱动	TRD	Tony Robotics	

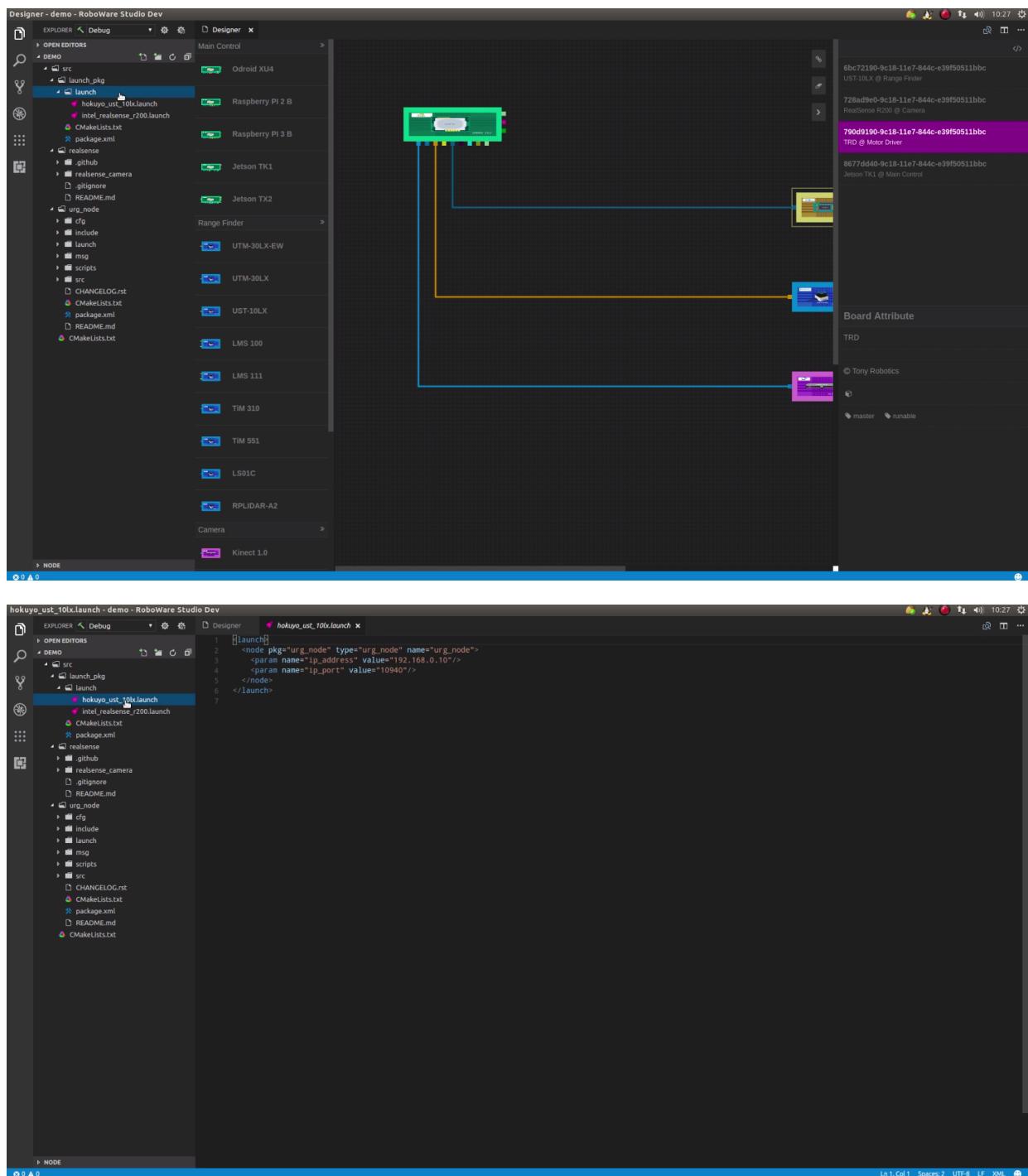
点击注册栏上面的**create workspace**按钮:



Designer 会提取画布上所有部件的 ROS 依赖软件的信息，执行软件包的自动下载和 launch 配置任务：



当 Designer 完成任务后，你能在工作区看到下载的 ROS 软件包和配置完成的 launch 文件:



```

intel_realsense_r200.launch - demo - RoboWare Studio Dev
EXPLORER Debug DESIGNER intel_realsense_r200.launch
OPEN EDITORS DEMO
src
  launch_pkgs
    launch
      hokuyo_ust_10lx.launch
      intel_realsense_r200.launch
      CMakeLists.txt
      package.xml
    realsense
      .github
      realsense_camera
        gitignore
        README.md
      urp_node
        rfo
        include
        launch
        msg
        scripts
        src
          CHANGELOG.rst
          CMakeLists.txt
          package.xml
          README.md
          CMakeLists.txt

```

The screenshot shows the RoboWare Studio Dev interface. The left sidebar displays the project structure under the 'src' directory, including 'launch\_pkgs' and 'realsense' packages. The 'intel\_realsense\_r200.launch' file is selected in the 'OPEN EDITORS' tab. The right side shows the 'DESIGNER' tab with the XML code for the launch file:

```

<?xml version="1.0"?>
<package>
  <!-- launch_pkgs/name -->
  <version>0.0.0</version>
  <!-- Example: -->
  <!-- maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <!-- maintainer email="King@todo.todo">King</maintainer>
  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Common used license strings: -->
  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <!-- license>TODO</license>
  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- url type="website">http://wiki.ros.org/launch_pkgs</url> -->
  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- author email="jane.doe@example.com">Jane Doe</author> -->
  <!-- The *_depend tags are used to specify dependencies -->
  <!-- Examples: -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!-- <build_depend>catkin</build_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!-- <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use run_depend for packages you need at runtime: -->
  <!-- <run_depend>message_runtime</run_depend> -->
  <!-- Use test_depend for packages you need only for testing: -->
  <!-- <test_depend>rostest</test_depend> -->
  <!-- <buildtool_depend>catkin</buildtool_depend> -->
  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>
</package>

```

```

package.xml - demo - RoboWare Studio Dev
EXPLORER Debug DESIGNER package.xml
OPEN EDITORS DEMO
src
  launch_pkgs
    launch
      hokuyo_ust_10lx.launch
      intel_realsense_r200.launch
      CMakeLists.txt
    package.xml
    realsense
      .github
      realsense_camera
        gitignore
        README.md
      urp_node
        rfo
        include
        launch
        msg
        scripts
        src
          CHANGELOG.rst
          CMakeLists.txt
          package.xml
          README.md
          CMakeLists.txt

```

The screenshot shows the RoboWare Studio Dev interface. The left sidebar displays the project structure under the 'src' directory, including 'launch\_pkgs' and 'realsense' packages. The 'package.xml' file is selected in the 'OPEN EDITORS' tab. The right side shows the 'DESIGNER' tab with the XML code for the package file:

```

<?xml version="1.0"?>
<package>
  <!-- launch_pkgs/name -->
  <version>0.0.0</version>
  <!-- Example: -->
  <!-- maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <!-- maintainer email="King@todo.todo">King</maintainer>
  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Common used license strings: -->
  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <!-- license>TODO</license>
  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- url type="website">http://wiki.ros.org/launch_pkgs</url> -->
  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- author email="jane.doe@example.com">Jane Doe</author> -->
  <!-- The *_depend tags are used to specify dependencies -->
  <!-- Examples: -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!-- <build_depend>catkin</build_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!-- <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use run_depend for packages you need at runtime: -->
  <!-- <run_depend>message_runtime</run_depend> -->
  <!-- Use test_depend for packages you need only for testing: -->
  <!-- <test_depend>rostest</test_depend> -->
  <!-- <buildtool_depend>catkin</buildtool_depend> -->
  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>
</package>

```

## 7. 开始工作

至此，Designer 完成了 ROS 设备的布局和依赖软件包的安装和配置，基础设施构建完毕。接下来，就是真正开始运行和开发 ROS 工作的时候了。