手持机械臂用户手册 v1.1

Hanbot USER MANUAL V1.1

陈睿 主编

武汉京天电器有限公司

WUHAN JINGTIAN ELECTRICAL CO.,LIMITED

2020 年 1 月 武汉

用户手册会定期进行检查和修正，更新后的内容将出现在新版本中。本手册中的内容或信息如有变更，恕不另行通知。
对本手册中可能出现的任何错误或遗漏，或因使用本手册及其中所述产品而引起的意外或间接伤害，武汉京天电器有限公司概不负责。
安装、使用产品前，请阅读本手册。
请保管好本手册，以便可以随时阅读和参考。
本手册中所有图片仅供示意参考，请以收到的实物为准。

注意:

1、这些指令是在Ubuntu 16.04 和 ROS Kinetic测试运行的。
2、如果您想了解更多关于开源手持机械臂的信息，请参考武汉京天电器韭菜盒子用户手册。
3、手持机械臂的组装指南在本手册的最后面。
我们很高兴宣布一本新书《ROS机器人编程》，这本书是由Tuttlebot3开发人员编写的。这本书以韩文、英文、中文和日本出版。它包含以下内容：

- ROS动力学KAM:基本概念、指令和工具
- 如何在ROS上使用传感器和执行器包
- 嵌入式ROS板：OpenCR
- 用Turtlebot3进行SLAM和导航
- 如何使用ROS Java编程实现递送机器人
- *韭菜盒子（LABOX-V1）移动机械臂使用MOVEIT的仿真Gazebo

如果需要详细了解，请参阅了解更多关于ROS、SLAM和导航的信息。

目录

1 手持机械臂简介

a）手持机械臂是一个完全开源的机器人操作平台

b）手持机械臂基于 ROS 和 OpenSource。手持机械臂在硬件上完全兼容 TurtleBot3。使用者同时可以使用 MoveIt!来控制它。

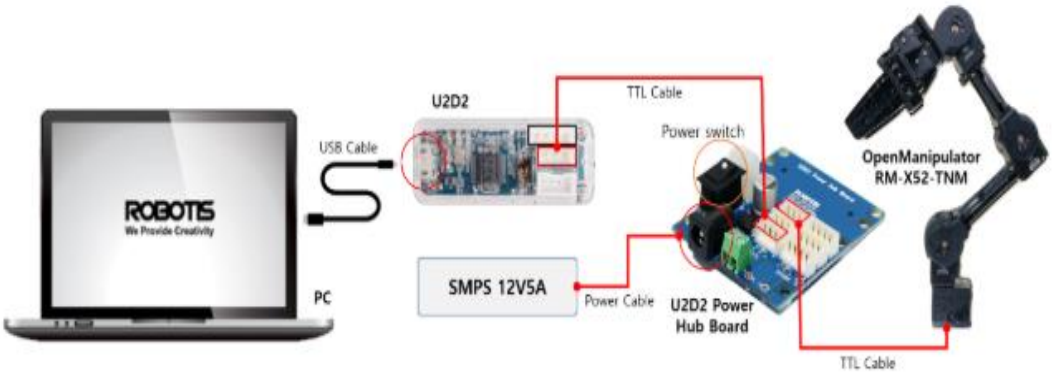
c）OpenManipulato 机械臂的大多数组件都以 STL 文件的方式上传到网络，以便使用者下载。其还支持使用者根据自己的需要去修改连接臂的长度或更改成其他结构。

d）OpenManipulato 机械臂同时支持 Gazebo 进行仿真。

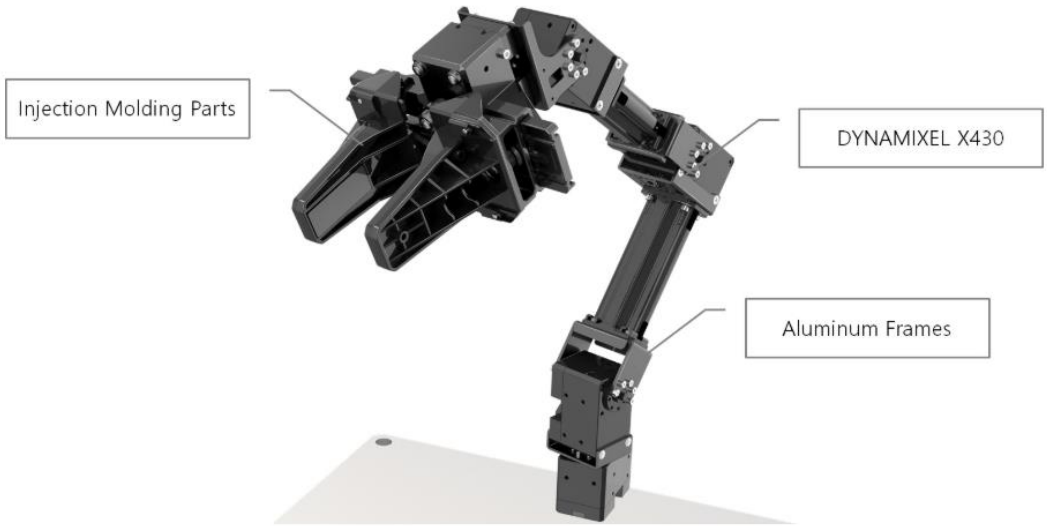
e）在 ROS 环境中，OpenManipulato 机械臂支持在 U2D2+U2D2 电源转接板。

f）OpenCR 板上使用；若在嵌入式系统上运行时，请准备 OpenCR 板。在这两种情况下，都要使用 12V 5A 电源给 DYNAMIXEL 舵机供电。

机械臂连接示意图如下所示：



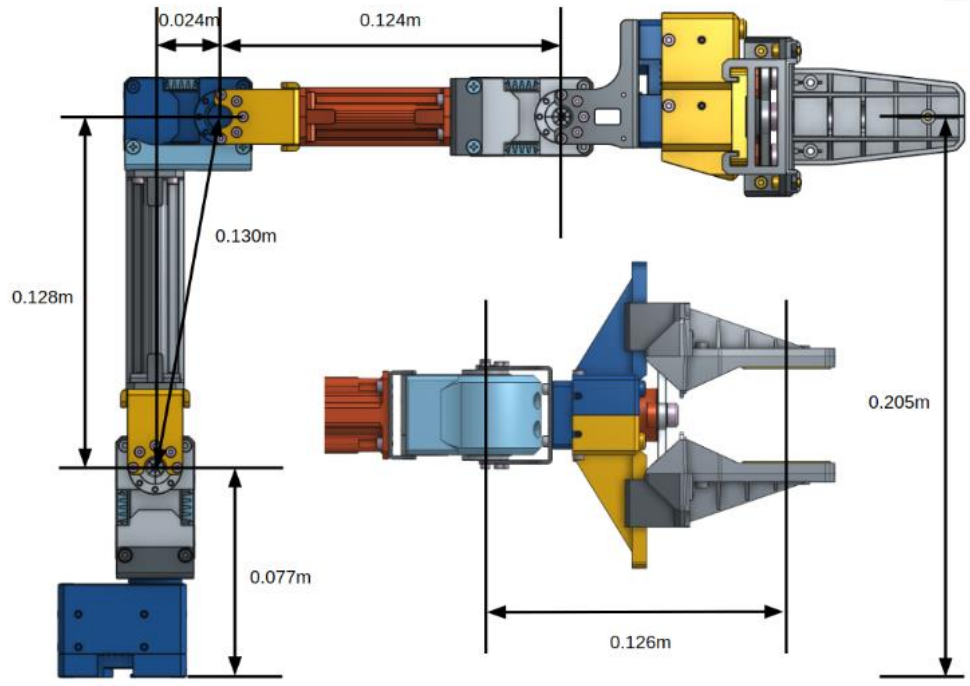
硬件介绍：



硬件参数表：

参数	Hanbot
驱动器	Dynamixel XM430-W350-T
输入电压	12V
自由度	5 （4 个转动关节 + 1 个夹爪）
最大负载	500g
重复定位精度	±0.2mm
最大关节转速	46 RPM
重量	0.7 kg
可达范围	380mm
夹爪行程	20～75mm
通讯协议	TTL 电平多点总线
软件支持	ROS、Dynamixel SDK，Arduino，Processing
主控器	PC、OpenCR

产品尺寸图：



2 手持机械臂功能总览

2.1 硬件设置

Check the Part List：手持机械臂包含配置手持主体所需的部件。为了控制和操作手持机械臂，必须使用可选部件。请检查零件清单页面，准备每个需要的零件。该 SMPS 12V5A 需要提供 12V 电源到手持机械臂的 dynamixels。该底板被用于将手持机械臂固定到工作区。您需要准备的控制器或通信板取决于您的开发环境。如果您想在嵌入式系统上进行开发，请准备 OpenCR 和 PC。要

使用 ROS，请准备好您的 PC 以及 U2D2 + U2D2 Power Hub Board 或 OpenCR。总而言之，准备的部件如下表所示。

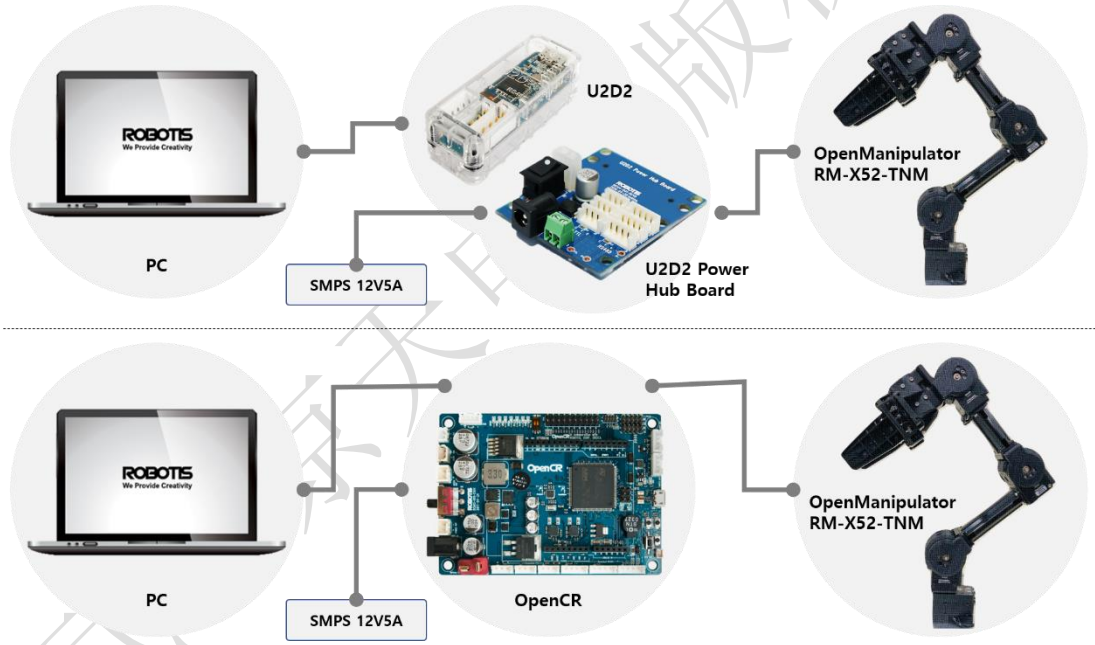
手持机械臂作为盒子的非组装部件提供。根据装配手册的说明组装手持机械臂 。

平台组成清单：

	ROS 环境	嵌入式环境
机械臂	Hanbot	Hanbot
底板	底板-02	底板-02
电源	SMPS 12V5A	SMPS 12V5A
主板	OpenCR 或 U2D2 + U2D2 电源集线器板	OpenCR

2.2 驱动连接

要在 Hanbot 上设置软件并运行它，您需要将 PC 和 Hanbot 连接到电路板。您可以单独使用 OpenCR 或 U2D2 和 U2D2 Power Hub Board 作为电路板。简单的连接图如下所示。有关详细的连接方法，请参阅[ROS]设置或[OpenCR]设置页面。



2.3 软件设置

Hanbot 支持 ROS 开发环境和嵌入式系统（OpenCR）。选择要运行 Hanbot 的开发环境，并为环境配置软件。

2.4 ROS 环境

在 ROS 开发环境中，您可以使用 ROS 提供的各种包以及我们的 ROS 包来运行 Hanbot。在这种情况下，运行 Hanbot 的所有进程都在 PC 中完成，而电路板仅用于将来自 PC 的数字信号转换为用于动态像素的通信信号。如果您想在 ROS

环境中运行 Hanbot，请根据 ROS 设置章节在 PC 上安装 Linux，ROS 和 ROS 软件包。

2.5 嵌入式系统

如果在嵌入式系统（OpenCR）中使用 Hanbot，则可以使用易于使用的系统（如 Arduino IDE）轻松应用算法来运行 Hanbot。在这种情况下，运行 Hanbot 的主要过程在嵌入式系统内部完成，并且可以使用 PC 的 GUI 程序，操纵杆控制器或简单的传感器将命令发送到嵌入式系统。如果要在嵌入式系统上运行 Hanbot，请在 PC 上安装软件，然后按照[OpenCR]设置章节设置 OpenCR。

2.6 基本操作

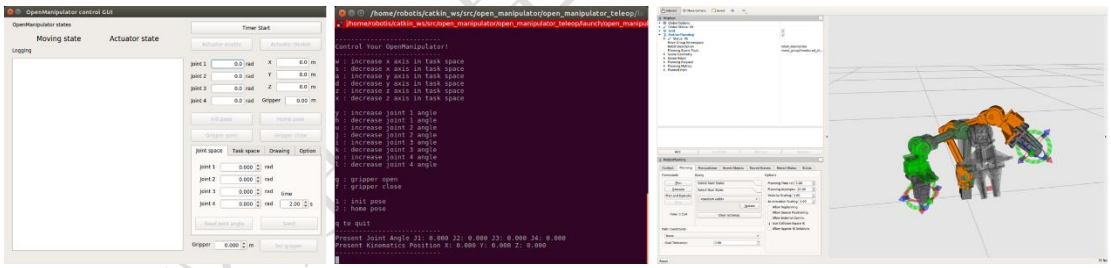
2.6.1 ROS 环境的基本操作

完成上述步骤后，通过提供的 Controller 包运行 Hanbot。您可以命令 Hanbot 通过 ROS 消息移动到特定位置。

成功运行控制器包后，可以使用我们提供的包将 ROS 消息发布到控制器包，从而运行 Hanbot。

Hanbot 也可以使用 MoveIt！进行操作。您可以运行使用 MoveIt 的控制器包！通过更改控制器包的启动文件中的变量。请参阅以下章节，并尝试使用 MoveIt！进行各种动作。

GUI 程序　　遥操作　　MoveIt!



GUI Program　　Teleoperation keyboard　　Moveit!

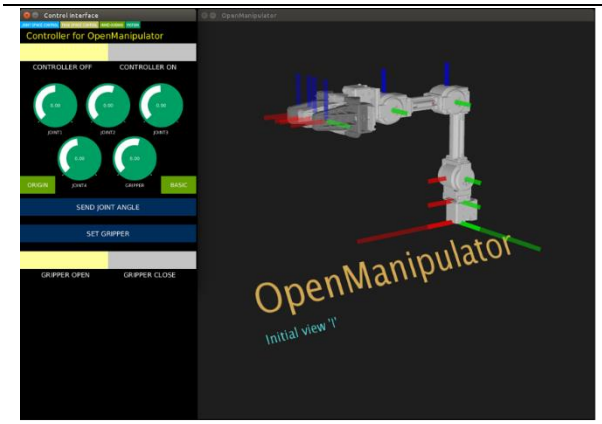
2.6.2 嵌入式系统的基本操作

Hanbot 不仅可以用 ROS 操作，还可以用简单的嵌入式系统（OpenCR）操作。在嵌入式系统（OpenCR）中，操作 Hanbot 的控制过程在 OpenCR 内部运行。请参阅以下页面将控制器上载到 OpenCR。

控制器

有多种方法可以向 OpenCR 中运行的控制器发送命令。我们提供处理源代码作为通过 USB 端口发送命令并接收操纵器状态的示例。我们还配置了控制器以使用 RC100 控制器操作机械手。

GUI 程序（Processing）　　遥控操作 RC100



GUI Program (Processing)

Teleoperation RC100

2.6.3 挑战应用

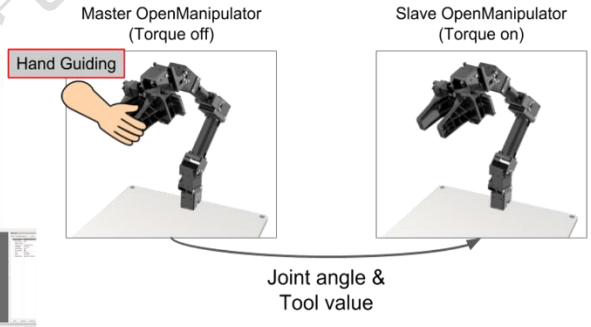
您想将 Hanbot 应用于实际工作吗？尝试将新的 ROS 包应用于 Hanbot。可以应用于各种任务。我们使用 Astra pro，Realsence D435 和 Raspberry Pi Camera V2 提供 AR 标记识别的示例。请参阅以下示例，以挑战基于摄像头的操作应用程序。

相机应用

你想在一个人类无法进入的空间中运行 Hanbot 吗？你想要简单，直观的运动规划吗？主从应用程序就是答案。我们提供了一个示例，通过链接两个 Hanbot 并使用手动引导教学操作主操纵器，可以轻松创建动作。按照以下方式尝试主从应用程序。

主从控制模式

Camera Application



Master-Slave Application

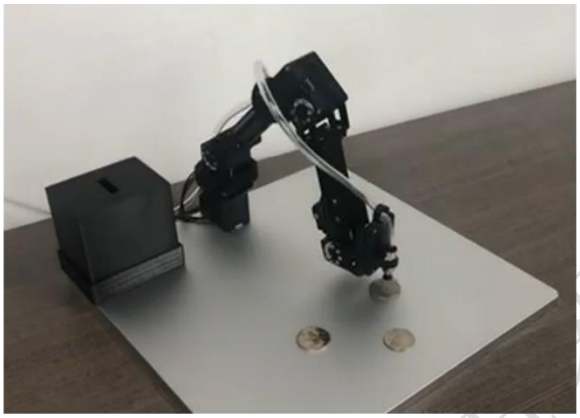
2.6.4 设计自己的操纵器

移动操作

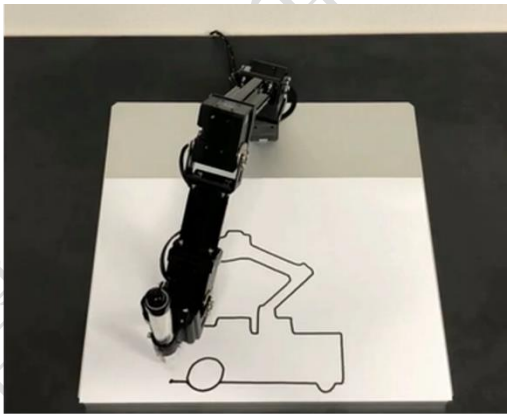
Hanbot 与 Turtlebot3 华夫派有完整的硬件组合。通过组装 TurtleBot3 华夫派和 Hanbot 来挑战您的移动操作。

末端工具修改

我们提出了一种更换和操纵机械手工具（夹具）的方法，以利用 Hanbot 实现更广泛的应用。使用笔架或真空夹具尝试新应用程序，如下例所示，并创建自己的工具以挑战更多应用程序。



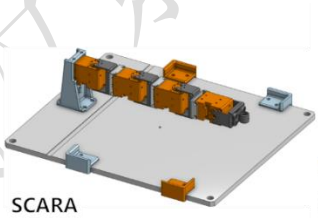
Vacuum Gripper



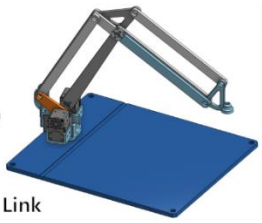
Pen Holder

2.6.5 Hanbot 的朋友们

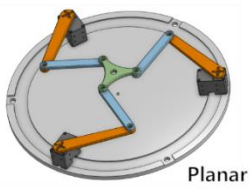
修改 Hanbot 的硬件（DOF，结构）和软件（运动学，轨迹）并操纵您自己的 Hanbot。我们提供各种 Hanbot 朋友作为硬件转换的例子。尝试用不同的结构控制机械手并享受它。并尝试针对不同结构的运动学求解算法。



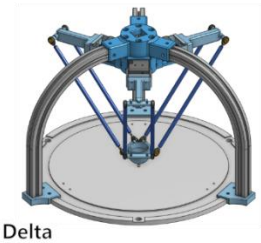
SCARA



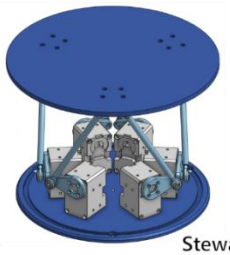
Link



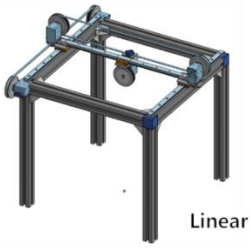
Planar



Delta



Stewart



Linear

3 手持机械臂的使用

3.1 安装 ROS 依赖包

在已经安装好 Ubuntu 和 ROS kinetic 版本操作系统后。为 Hanbot 安装 ROS 的依赖包。在终端窗口中运行以下命令。

注意：可以使用屏幕左上角的 Ubuntu 搜索图标找到终端应用程序。终端的快捷键是 Ctrl+ Alt+ t。

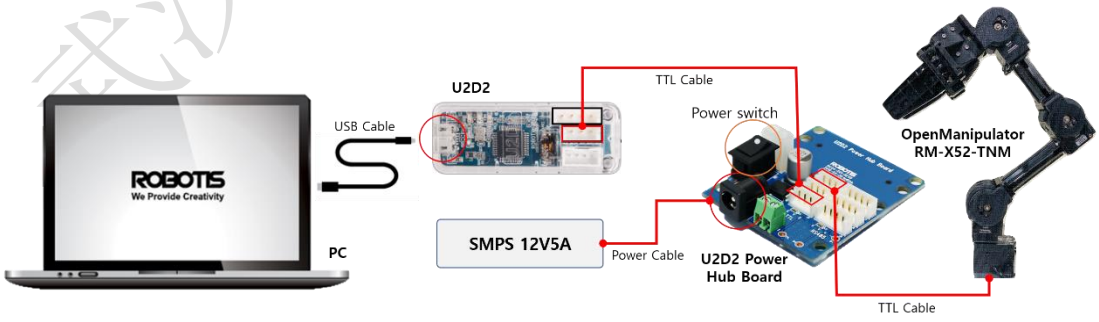
```
sudo apt-get install ros-kinetic-ros-controllers ros-kinetic-gazebo* ros-kinetic-moveit* ros-kinetic-industrial-core

cd ~/catkin_ws/src/
git clone https://github.com/ROBOTIS-GIT/DynamixelSDK.git
git clone https://github.com/ROBOTIS-GIT/dynamixel-workbench.git
git clone https://github.com/ROBOTIS-GIT/dynamixel-workbench-msgs.git
git clone https://github.com/ROBOTIS-GIT/open_manipulator.git
git clone https://github.com/ROBOTIS-GIT/open_manipulator_msgs.git
git clone https://github.com/ROBOTIS-GIT/open_manipulator_simulations.git
git clone https://github.com/ROBOTIS-GIT/robotis_manipulator.git
cd ~/catkin_ws && catkin_make
```

如果 catkin_make 命令已完成且没有任何错误，则完成使用 Hanbot 的所有准备工作。

3.2 通讯转换器

3.2.1 U2D2 连接

将微型 USB（连接到 PC），Dynamixel（Hanbot）和 12V 电源连接到 U2D2 和 U2D2 电源集线器板，如下所示。



注意：有关 U2D2 和 U2D2 电源集线器板的详细连接，请参阅 U2D2 电子手册和 U2D2 电源集线器板电子手册。

使用延迟计时器设置：

在 linux（ubuntu）环境中，USB 延迟时间默认设置为 16ms。按照以下步骤将通信延迟时间设置为通过 USB 连接的 Dynamixel 和 PC 之间的最低值（1ms）。

打开终端窗口并运行 roscore。

在运行 roscore 的情况下，打开一个新的终端窗口并输入以下命令来设置 usb 延迟时间。

```
$ rosrun open_manipulator_controller create_udev_rules
```

提示：此输入命令将 USB 延迟计时器设置为 1 ms。如果要查看该设置，请在终端中运行以下命令。

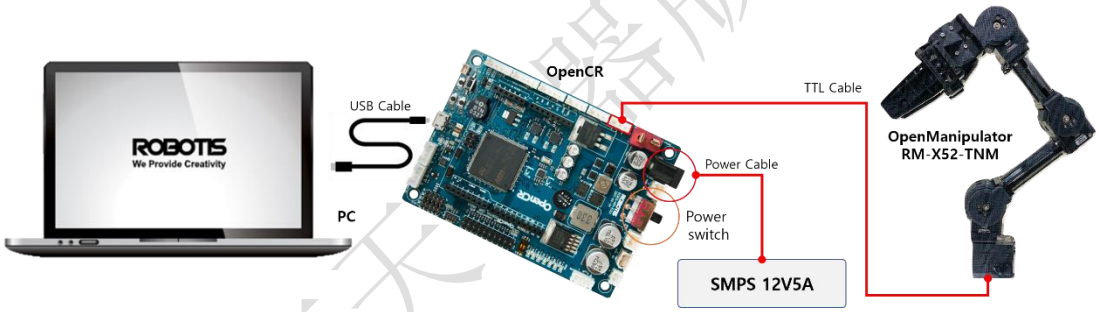
```
cat /sys/bus/usb-serial/devices/ttyUSB0/latency_timer
```

3.2.2 OpenCR 连接

本节介绍如何将 OpenCR 设置为 PC 的 ROS 包和 Hanbot 的 DYNAMIXEL 之间的通信板。

如果要在没有 ROS 的情况下在嵌入式系统（OpenCR）上运行 Hanbot，请参考[OpenCR]设置。

将 micro USB（连接到 PC），Dynamixel（Hanbot）和 12V Power 连接到 OpenCR，如下所示。



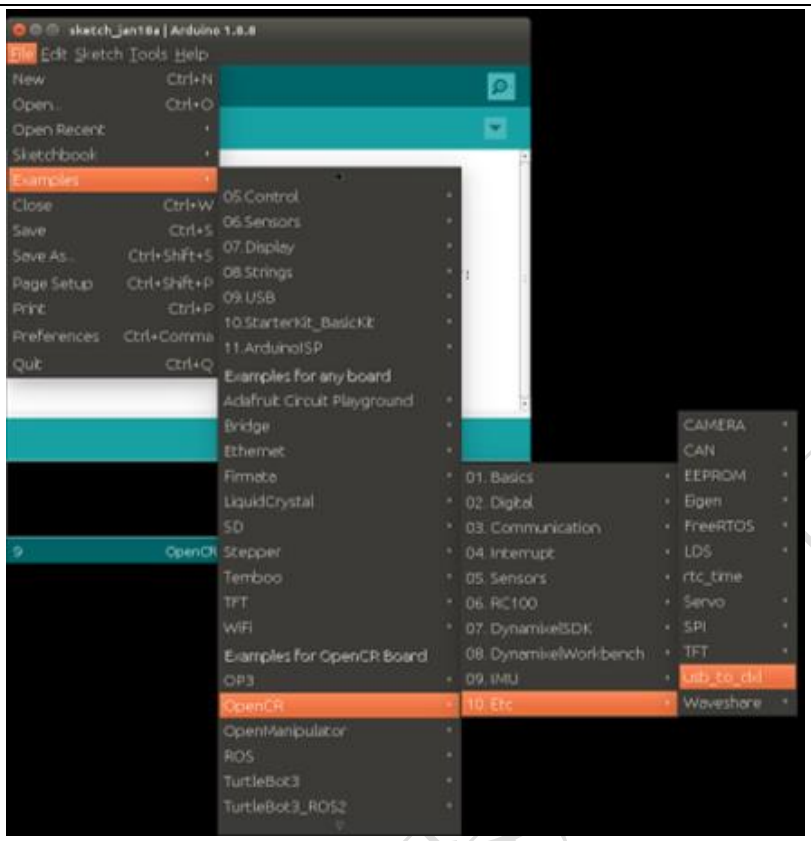
注意：请参阅 OpenCR 的详细说明。

上传源代码

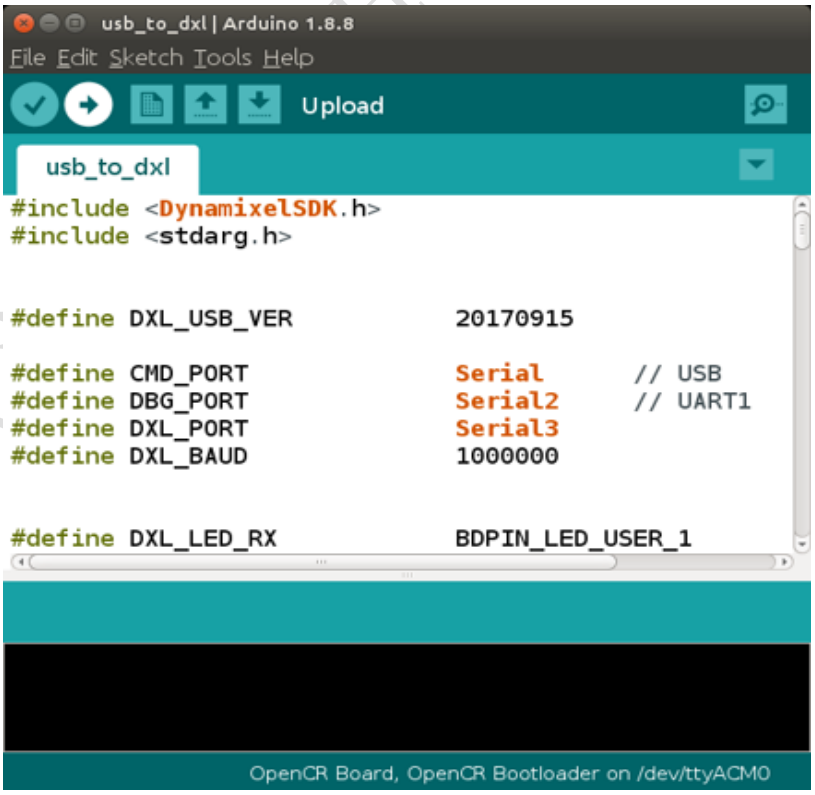
请参考以下页面安装 Arduino IDE 并在 Arduino IDE 环境中启用 OpenCR 板。

用于使用 OpenCR 的 Arduino IDE

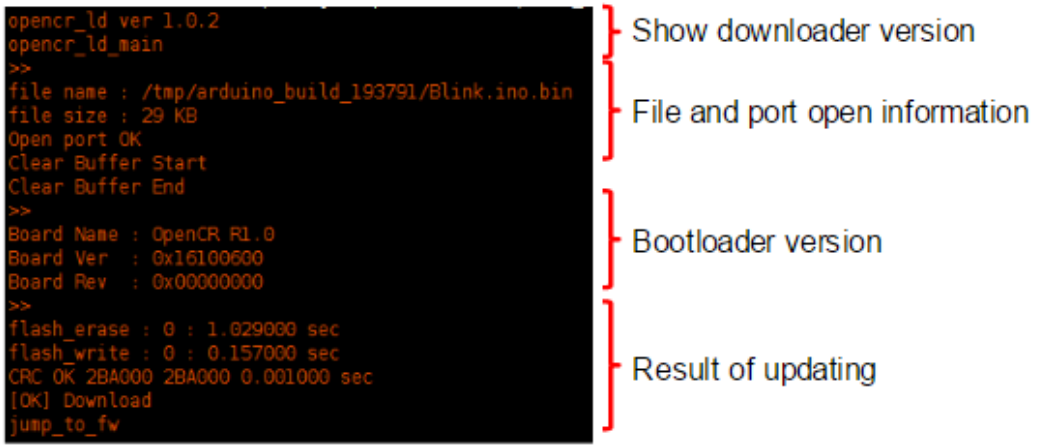
完成上述所有设置后，打开 Arduino IDE 并打开 usb to dxl 示例源代码，如下所示。进入 Examples→OpenCR→10.Etc→usb_to_dxl 上的 Arduino IDE 的 OpenCR。如下图：



将 `usb to dxl` 示例源上载到 OpenCR。



上载完成后，日志窗口中将显示以下注释。



如果显示的评论与此不同，请尝试重新上传。有关详细信息，请参阅 OpenCR。

提示：如果 cmd_read_board_name fail: 0xF020 在上传过程中出现错误，请参阅 OpenCR 电子手册进入固件恢复模式，并在进入模式后再次上传源代码。

3.3 启动机械臂

a）在启动控制器之前，让我们检查包中的 open_manipulator_controller 启动文件 open_manipulator_controller。

```
<launch>
    <arg name="use_robot_name"          default="open_manipulator"/>
    <arg name="dynamixel_usb_port"      default="/dev/ttyUSB0"/>
    <arg name="dynamixel_baud_rate"     default="1000000"/>
    <arg name="control_period"          default="0.010"/>
    <arg name="use_platform"            default="true"/>
    <arg name="use_moveit"              default="false"/>
    <arg name="planning_group_name"     default="arm"/>
    <arg name="moveit_sample_duration" default="0.050"/>
    <group if="$(arg use_moveit)">
      <include file="$(find
open_manipulator_controller)/launch/open_manipulator_moveit.launch">
        <arg name="robot_name"          value="$(arg use_robot_name)"/>
        <arg name="sample_duration" value="$(arg moveit_sample_duration)"/>
      </include>
    </group>

    <node name="$(arg use_robot_name)" pkg="open_manipulator_controller"
type="open_manipulator_controller" output="screen" args="$(arg dynamixel_usb_port)
$(arg dynamixel_baud_rate)">
        <param name="using_platform"          value="$(arg use_platform)"/>
        <param name="using_moveit"            value="$(arg use_moveit)"/>
```

```
        <param name="planning_group_name"    value="$(arg planning_group_name)"/>
        <param name="control_period"         value="$(arg control_period)"/>
        <param name="moveit_sample_duration"    value="$(arg
moveit_sample_duration)"/>
    </node>

</launch>
```

参数解释：

use_robot_name 是一个设置操纵器名称的参数（ROS 消息的名称空间）。

dynamixel_usb_port 是一个参数，用于将使用端口设置为与 Hanbot 的 Dynamixel 连接。如果使用 U2D2，则应设置/ dev / ttyUSB @。如果使用 OpenCR，则应设置/ dev / ttyACM @（@表示连接到 Dynamixel 的端口号）。

dynamixel_baud_rate 是设置动态像素波特率的参数。Hanbot 中使用的动态像素的默认波特率是 1000000.是

control_period 设置动态像素和 PC 之间的通信周期的参数（控制循环时间）。

use_platform 是一个参数，用于设置是使用实际的 Hanbot 还是 Hanbot 模拟。请参考 ROS Simulation 章节。

use_moveit，planning_group_name 和 moveit_sample_duration 是应该设置加载 move_group 包的参数。请参考 MoveIt! 章节。

b）设置参数后，启动 Hanbot 控制器以启动[ROS]操作。

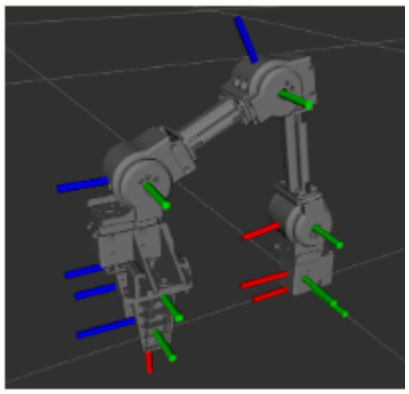
请打开终端窗口，输入 roscore 作为输入以下命令。

```
roscore
```

运行 roscore 后，打开另一个终端窗口并在终端中输入以下命令。

```
roslaunch open_manipulator_controller
open_manipulator_controller.launch
```

建议将 Hanbot 置于以下姿势并启动控制器，以使 Hanbot 的每个组件不冲突。



如果 Hanbot 控制器已成功启动，终端将显示以下消息

```
    ========

    PARAMETERS
     * /open_manipulator/control_period: 0.01
     * /open_manipulator/moveit_sample_duration: 0.05
     * /open_manipulator/planning_group_name: arm
     * /open_manipulator/using_moveit: False
     * /open_manipulator/using_platform: True
     * /rosdistro: kinetic
     * /rosversion: 1.12.14

    NODES
      /
        open_manipulator
(open_manipulator_controller/open_manipulator_controller)

    ROS_MASTER_URI=http://localhost:11311

    process[open_manipulator-1]: started with pid [23452]
    Joint Dynamixel ID : 11, Model Name : XM430-W350
    Joint Dynamixel ID : 12, Model Name : XM430-W350
    Joint Dynamixel ID : 13, Model Name : XM430-W350
    Joint Dynamixel ID : 14, Model Name : XM430-W350
    Gripper Dynamixel ID : 15, Model Name :XM430-W350
    [ INFO] [1544509070.096942788]: Succeeded to init /open_manipulator
```

提示：

- 如果无法加载 Dynamixels，请使用 Dynamixel-Workbench 软件包中的以下命令检查 Dynamixels 设置。
- `rosrun dynamixel_workbench_controllers find_dynamixel /dev/ttyUSB0`
- 即使您找不到任何 Dynamixels，请检查固件以使用 ROBOTIS 软件（R + Manager 2.0 或 R + Manager 1.0）
- 如果您想更改 Dynamixel ID，请检查 Hanbot.cpp open_manipulator_lib 文件夹。关节的默认 ID 为 11,12,13,14，夹具的默认 ID 为 15

3.3.1 示教 GUI 程序

您可以使用 GUI 程序来操作 Hanbot。启动 open_manipulator_control_gui 节点。该程序显示状态并允许用户控制 Hanbot。

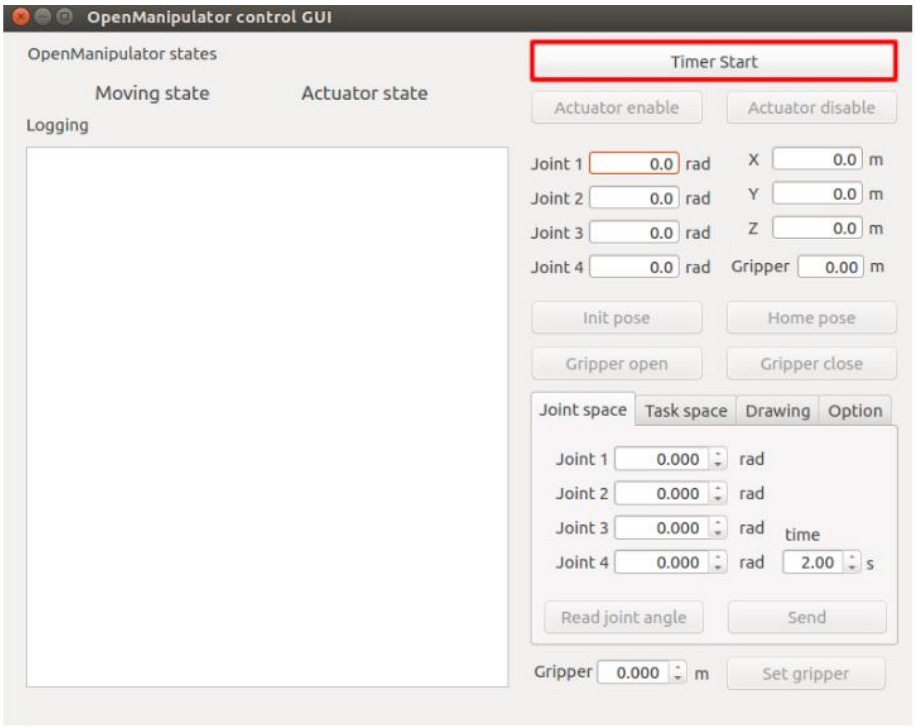
启动机械臂

```
roslaunch open_manipulator_controller open_manipulator_controller.launch
```

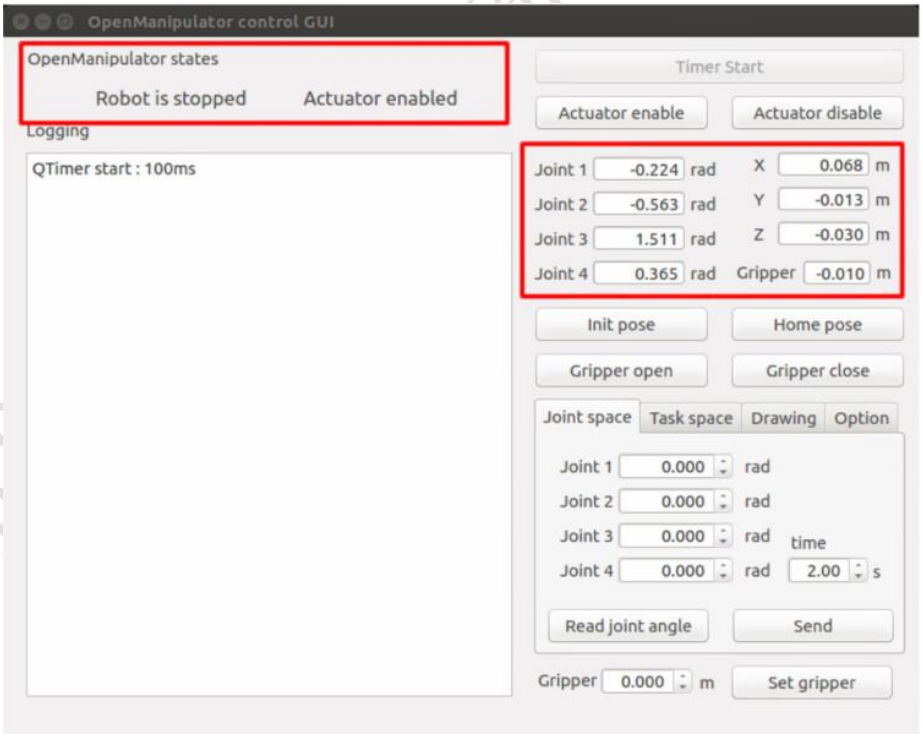
启动 GUI 界面

```
roslaunch open_manipulator_control_gui open_manipulator_control_gui.launch
```

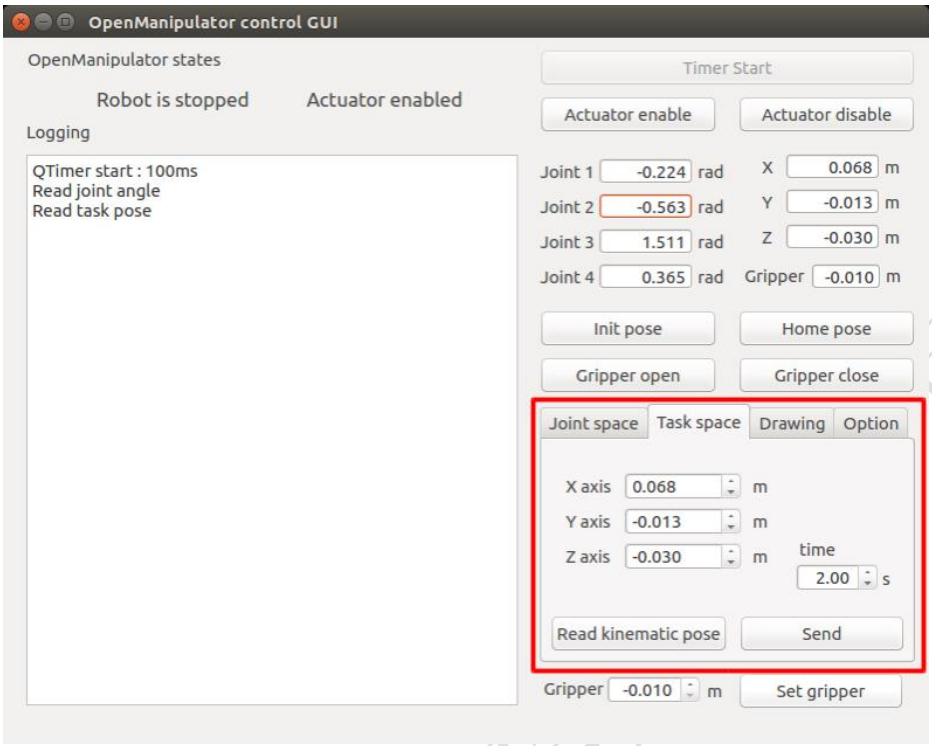
GUI 界面如下所示，首先点击单击 `Timer Start` 按钮，与机械臂建立连接。



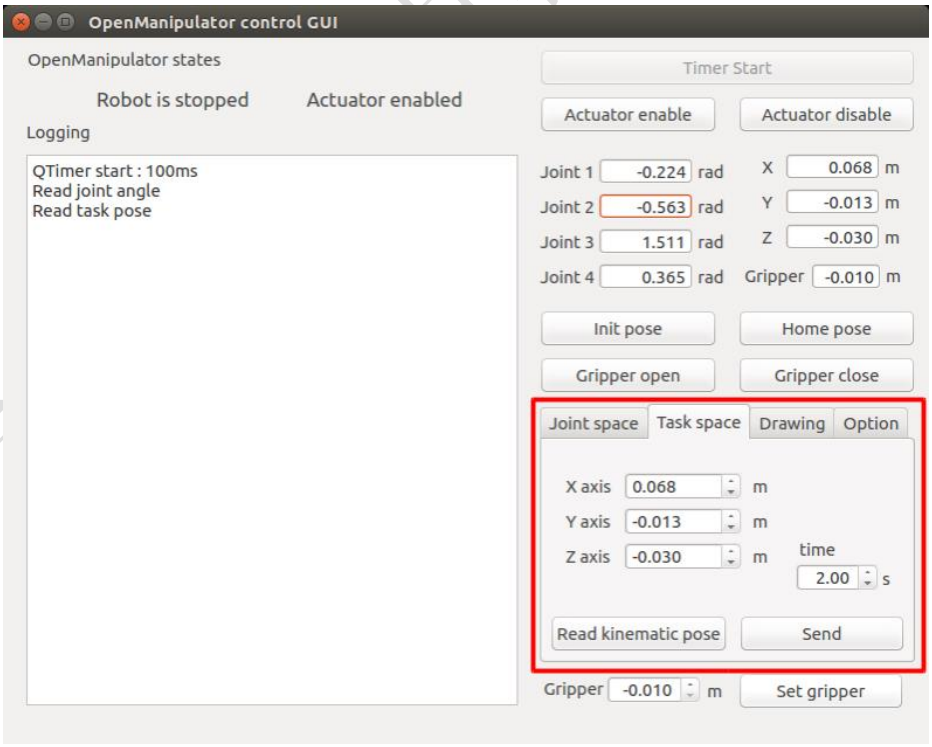
检查 Hanbot 的状态（关节状态，运动学姿态）。



在关节空间中操作 Hanbot 。输入轨迹的关节角度和总时间。然后单击 send 按钮。

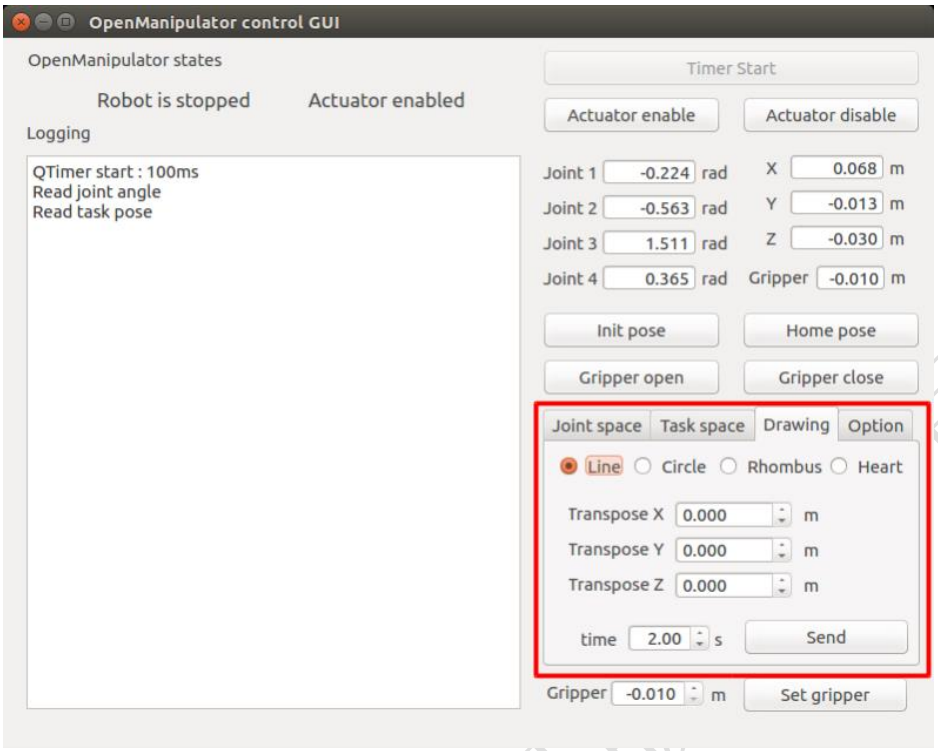


在任务空间中操作 Hanbot 。在任务空间中输入 Hanbot 末端执行器（工具）的运动学姿势以及轨迹的总时间。然后单击 send 按钮。



使用 Hanbot 创建绘图轨迹。首先，选择绘图轨迹类型（直线，圆，菱形，心形）。并根据绘图轨迹类型和绘图轨迹的总时间输入参数。然后单击

send 按钮。



3.3.2 遥操作

1）键盘操作

启动 open_manipulator_teleop_keyboard 节点，使用键盘进行简单的遥操作测试。

```
roslaunch open_manipulator_teleop open_manipulator_teleop_keyboard.launch
```

如果节点成功启动，则以下指令将出现在终端窗口中。向我们解释了如何通过电脑的键盘来控制机械臂运动。

```
---------------------------
Control Your Hanbot!
---------------------------
w : increase x axis in task space
s : decrease x axis in task space
a : increase y axis in task space
d : decrease y axis in task space
z : increase z axis in task space
x : decrease z axis in task space

y : increase joint 1 angle
h : decrease joint 1 angle
u : increase joint 2 angle
j : decrease joint 2 angle
i : increase joint 3 angle
```

```
k : decrease joint 3 angle
o : increase joint 4 angle
l : decrease joint 4 angle

g : gripper open
f : gripper close

1 : init pose
2 : home pose

q to quit
--------------------------
Present Joint Angle J1: 0.000 J2: 0.000 J3: 0.000 J4: 0.000
Present Kinematics Position X: 0.000 Y: 0.000 Z: 0.000
 ----------------------------
```

2） PS4 操纵杆

使用 PS4 操纵杆安装远程操作包。

```
sudo apt-get install ros-kinetic-joy ros-kinetic-joystick-drivers
ros-kinetic-teleop-twist-joy
sudo pip install ds4drv
```

使用以下命令通过蓝牙将 PS4 操纵杆连接到 PC

```
sudo ds4drv
```

按住 Playstation 按钮+共享按钮 10 秒，进入 PS4 配对模式。如果 PS4 上的指示灯变为蓝色，请在终端中输入以下命令并控制 Hanbot。

```
export ROS_NAMESPACE=/open_manipulator
roslaunch teleop_twist_joy teleop.launch
roslaunch open_manipulator_teleop
open_manipulator_teleop_joystick.launch
```

3）XBOX 360 操纵杆

使用 XBOX 360 操纵杆安装远程操作包。

```
sudo apt-get install xboxdrv ros-kinetic-joy ros-kinetic-joystick-
drivers ros-kinetic-teleop-twist-joy
```

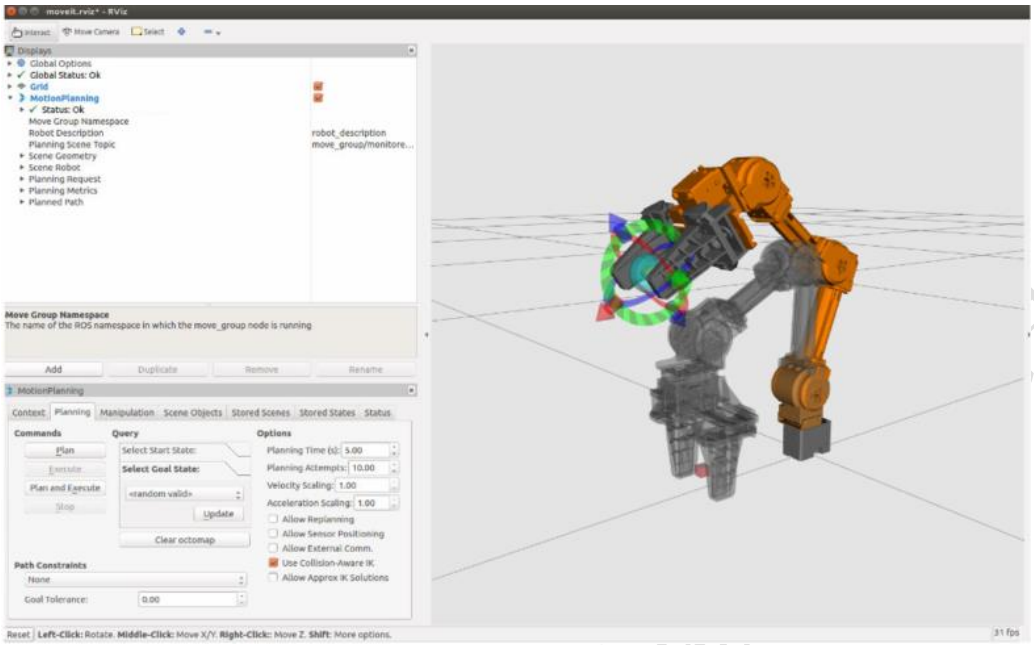
使用无线适配器或 USB 电缆将 XBOX 360 操纵杆连接到 PC，并为 XBOX 360 操纵杆启动遥控操作包。

```
sudo xboxdrv --silent
export ROS_NAMESPACE=/open_manipulator
roslaunch teleop_twist_joy teleop.launch
roslaunch open_manipulator_teleop
open_manipulator_teleop_joystick.launch
```

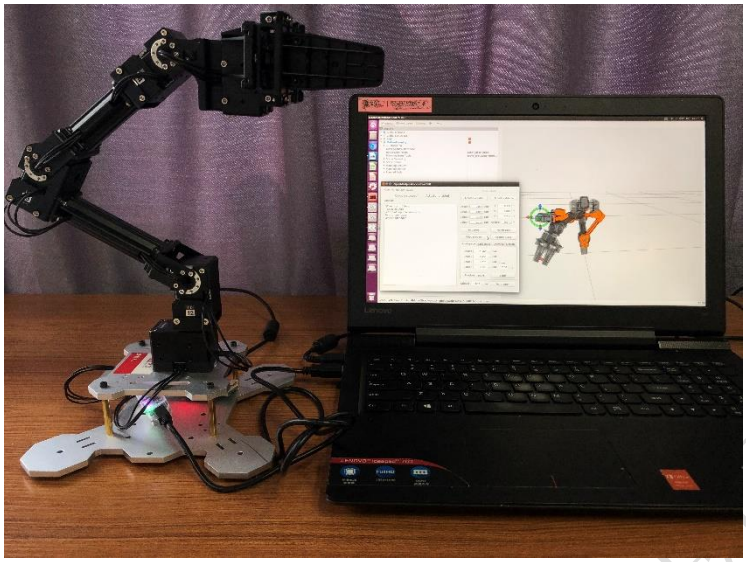
3.3.3 Moveit 使用

启动 open_manipulator_controller。

roslaunch open_manipulator_controller
open_manipulator_controller.launch use_moveit:=true
启动控制器以使用 MoveIt！时，必须关闭 Hanbot 启动文件。



服务服务器列表：open_manipulator_controller 具有的与 MoveIt！相关的服务服务器的列表。
/open_manipulator/moveit/get_joint_position（open_manipulator_msgs / GetJointPosition）
用户可以使用此服务接收由 move_group 计算的联合位置。
/open_manipulator/moveit/get_kinematics_pose（open_manipulator_msgs / GetKinematicsPose）
用户可以使用此服务接收由 move_group 计算的运动姿势。
/open_manipulator/moveit/set_joint_position（open_manipulator_msgs / SetJointPosition）
用户可以使用此服务通过 move_group 在关节空间中创建轨迹。用户输入目标关节的角度和轨迹的总时间。
/open_manipulator/moveit/set_kinematics_pose（open_manipulator_msgs / SetKinematicsPose）
用户可以使用此服务通过 move_group 在任务空间中创建轨迹。用户在任务空间中输入 Hanbot 末端执行器（工具）的运动学姿势（仅方向）和轨迹的总时间。
提示：如果您想使用反向运动学 position_only，请检查 open_manipulator_moveit- > config- > kinematics.yaml 并将 position_only_ik 参数设置为 True。
实际操作 MoveIt 如下图所示：



拖动 RVIZ 软件里面机械臂前方的小球到任意位置，注意如果机械臂本体变为红色则该位置不可达。在页面左下方 motion planning 操作面板中选择 planning 模块如下图，（为保证机械臂稳定，请用手按住底座）点击 plan and excute，机械臂开始运动到目标位置。可根据用户需要在规划栏中选取不同的路径规划算法，通过比较发现不同的算法运动轨迹，规划成功率各不相同。

3.4 二维码识别抓取应用

3.4.1 相机概述与驱动安装

在英特尔 RealSense™深度相机 D435 是一个 USB 供电的深度相机和由一对深度传感器，RGB 传感器和红外投影仪。它是制造商和开发人员为其原型开发添加深度感知功能的理想选择。

产品规格

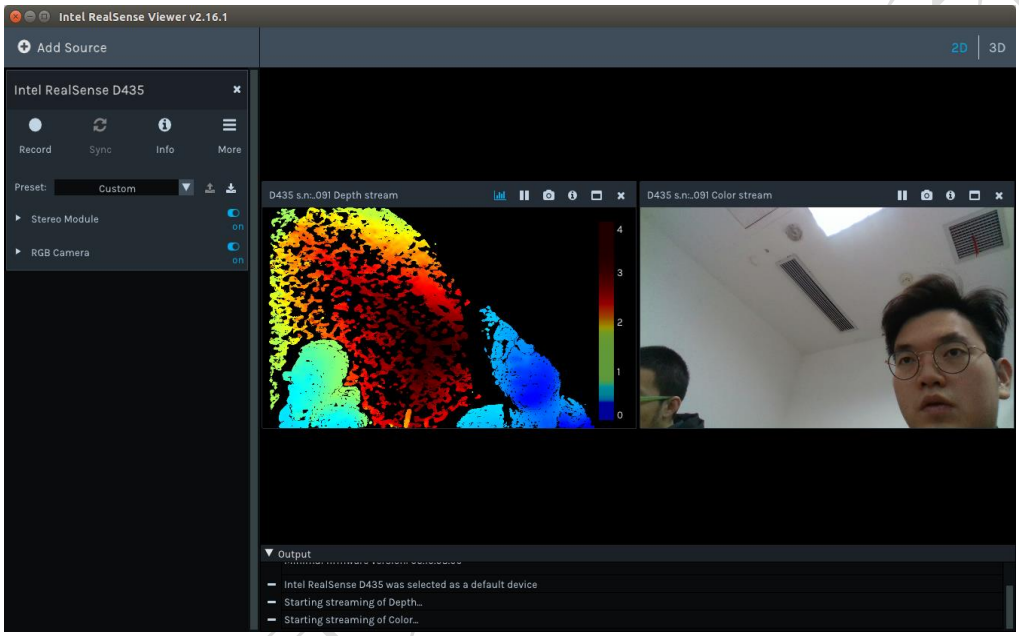
项目	产品规格
使用环境	室内室外
RGB 传感器分辨率和帧速率	1920 x 1080 30 fps
RGB 传感器视场	69.4°（高）x 42.5°（V）x 77°（深）（+/- 3°）
深度流输出分辨率	最高 1280 x 720
深度流输出帧率	高达 90 fps
深度视野（FOV）	85.2°（高）x 58°（长）x 94°（深）（+/- 3°）
最小深度距离（Min-Z）	0.2 米
最大范围	大约 10 米
尺寸	90 毫米 x 25 毫米 x 25 毫米
连接器	USB 3.0 Type - C.

驱动安装 realsense D435 安装步骤

https://github.com/IntelRealSense/librealsense/releases/tag/v2.16.1
在官网上下载源码，解压到根目录下。

参考官网
https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md
一般内核不会太老(检查命令：uname -r)，如果内核版本低于 4.15 请自行将内核升级到 4.15 或者更高的版本。
我的是 4.15.0-38-generic
许多步骤可以省略。于是按下述步骤安装：
1.拔掉摄像头和电脑的连接
2.sudo apt-get update
3.sudo apt-get install git libssl-dev libusb-1.0-0-dev pkg-config libgtk-3-dev
4.sudo apt-get install libglfw3-dev
5.cd librealsense(tab)

6. sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules && udevadm trigger
7. ./scripts/patch-ubuntu-kernel-4.16.sh (或者./scripts/patch-realsense-ubuntu-lts.sh)出现黄色提示可以忽略
8. mkdir build && cd build
9. cmake ../
10. cmake ../ -DBUILD_EXAMPLES=true （可以跳过）
11. sudo make uninstall && make clean && make && sudo make install
在任意目录下敲
（连上摄像头）realsense-viewer
显示界面如下：打开对应需要显示的画面启动相机



在 ROS 下运行以下命令：

```
$ roslaunch realsense2_camera rs_camera.launch
```

您可以使用 rviz 或 image_view 来验证驱动程序。您可以从应用程序顶部的下拉菜单中选择与英特尔®实感™深度相机 D435 相关的数据主题名称。

```
$ rqt_image_view
```

3.4.2 安装 AR Marker Package

注意：
这些说明在 Ubuntu 16.04 和上进行了测试 ROS Kinetic Kame。
该 open_manipulator_perceptions 软件包需要 ar_track_alvar 包装。
在运行以下说明之前，请确保运行 Hanbot 环境，机械臂上电状态。

安装

```
  $ sudo apt-get install ros-kinetic-ar-track-alvar ros-kinetic-ar-track-alvar-msgs ros-kinetic-image-proc
  $ cd ~/catkin_ws/src
  $ git clone https://github.com/ROBOTIS-GIT/open_manipulator_perceptions.git
  $ cd ~/catkin_ws && catkin_make
```

注意：

必须安装 Realsense D435 ROS 包。

```
  $ roslaunch open_manipulator_ar_markers ar_pose.launch camera_model:=realsense_d435
```

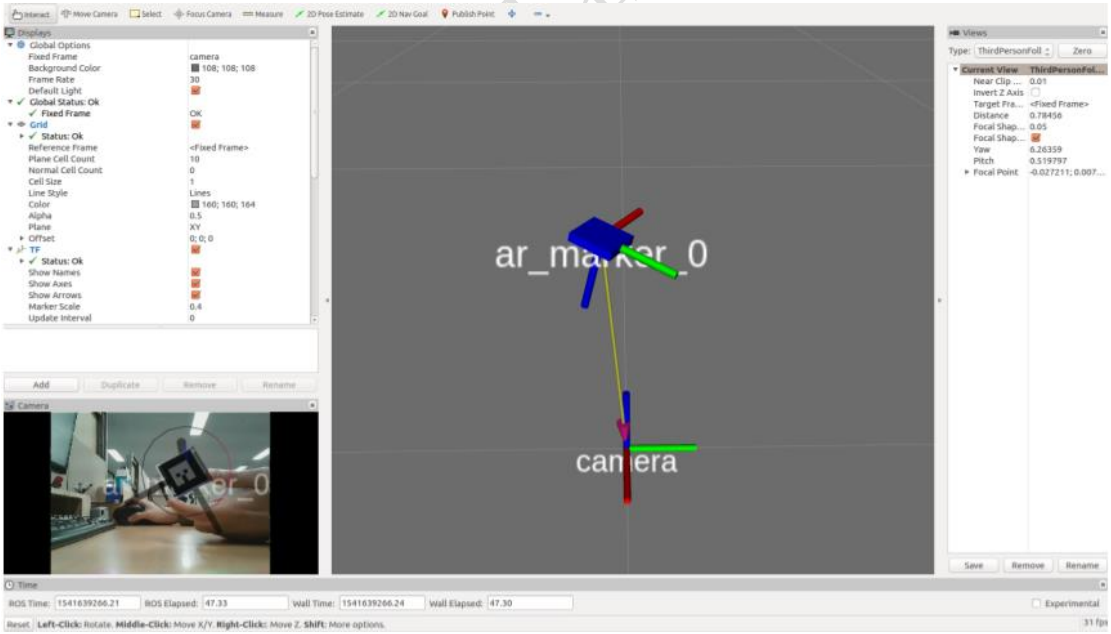
[远程电脑]

```
  $ roslaunch open_manipulator_ar_markers ar_pose.launch camera_model:=raspicam
```

RViz

当相机识别 AR 标记时，AR 标记的姿势显示在 RViz

上。



3.4.3 拾取和放置示例

在此示例中，Hanbot 使用 Raspberry Pi Camera V2 来拾取和放置块。使用 3D 打印机将相机框架打印到 Hanbot 以安装相机。将 3cm x 3cm 的 ar 标记贴在立方体块上。运行示例包时，按顺序选择 ar 标记的 id 1,2 和 3，并将它们堆叠在一侧。

1）安装 ROS 包

注意：

- 这些说明在 `Ubuntu 16.04` 和上进行了测试 `ROS Kinetic Kame`。
- 在运行以下说明之前，请确保运行 Hanbot 控制器。
- 安装相机包并安装 AR 标记包。

注意：

- 要使用 **Raspberry Pi Camera V2**，请将其安装在**远程 PC 上**

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/open_manipulator_applications.git
$ cd ~/catkin_ws && catkin_make
```

如果 catkin_make 命令已完成且没有任何错误，则完成所有准备工作。

2）执行示例

请打开终端窗口，输入 roscore 作为输入以下命令。

```
$ roscore
```

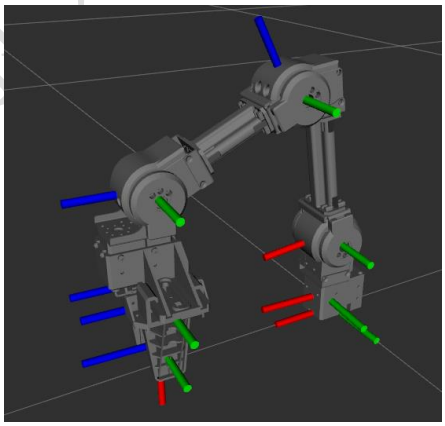
运行 roscore 后，运行 Hanbot 的控制器。打开另一个终端窗口，在终端中输入以下命令。

```
$roslaunch open_manipulator_controller open_manipulator_controller.launch
```

警告！

请在运行 Hanbot 之前检查每个关节位置。由于联合位置超出范围，它可能会停止运行。

下面的图片向您展示了 Hanbot 的理想姿势。当未启用 DYNAMIXEL 扭矩时，请调整每个关节以及下图。



如果 Hanbot 控制器已成功启动，终端将显示以下消息。

```
SUMMARY
========

PARAMETERS
 * /open_manipulator/control_period: 0.01
```

```
 * /open_manipulator/moveit_sample_duration: 0.05
 * /open_manipulator/planning_group_name: arm
 * /open_manipulator/using_moveit: False
 * /open_manipulator/using_platform: True
 * /rosdistro: kinetic
 * /rosversion: 1.12.14

NODES
  /
    open_manipulator (open_manipulator_controller/open_manipulator_controller)

ROS_MASTER_URI=http://localhost:11311

process[open_manipulator-1]: started with pid [23452]
Joint Dynamixel ID : 11, Model Name : XM430-W350
Joint Dynamixel ID : 12, Model Name : XM430-W350
Joint Dynamixel ID : 13, Model Name : XM430-W350
Joint Dynamixel ID : 14, Model Name : XM430-W350
Gripper Dynamixel ID : 15, Model Name :XM430-W350
[ INFO] [1544509070.096942788]: Succeeded to init /open_manipulator
```

然后打开另一个终端窗口并在终端中输入以下命令。此命令用于执行识别 ar 标记的包。输入您使用的相机类型和 ar 标记的大小。在这个例子中，我们使用 Raspberry Pi Camera V2 和 3cm ar 标记。

```
$ roslaunch open_manipulator_ar_markers ar_pose.launch camera_model:=raspicam user_marker_size:=3.0
```

然后打开另一个终端窗口并在终端中输入以下命令。

```
$ roslaunch open_manipulator_pick_and_place open_manipulator_pick_and_place.launch
```

因此，您可以在终端窗口中看到以下消息。您可以检查机器人状态。

```
Pick and Place demonstration!
---------------------------
1 : Home pose
2 : Pick and Place demo. start
3 : Pick and Place demo. Stop
---------------------------
---------------------------
Present Joint Angle J1: 0.000 J2: 0.000 J3: 0.000 J4: 0.000
Present Tool Position: 0.000
Present Kinematics Position X: 0.000 Y: 0.000 Z: 0.000
------------------------------
```

有三个命令。请在终端输入该号码。

初始位姿：移动到初始位姿。

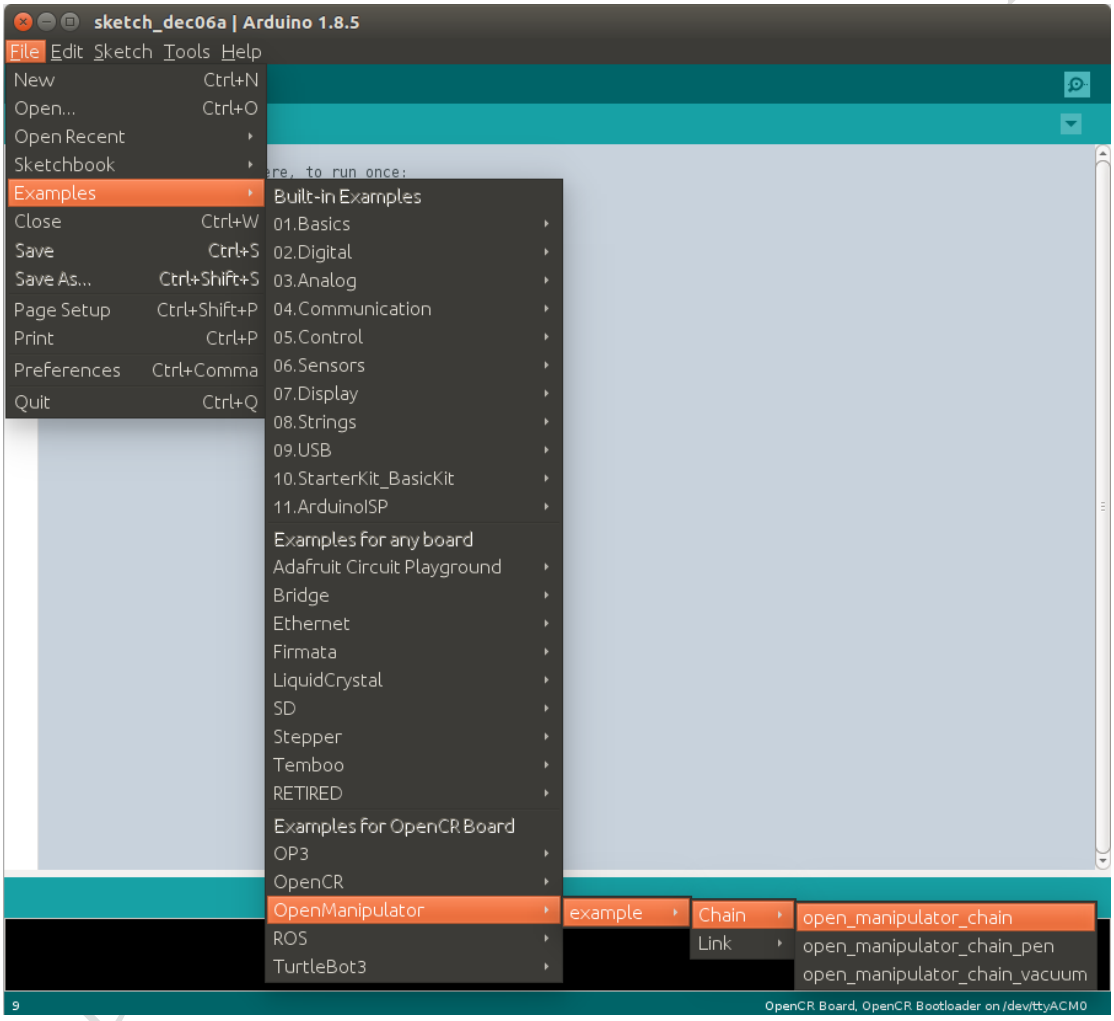
拾取和放置演示。开始：开始拾取和放置演示。

拾取和放置演示。停止：停止拾取和放置演示。

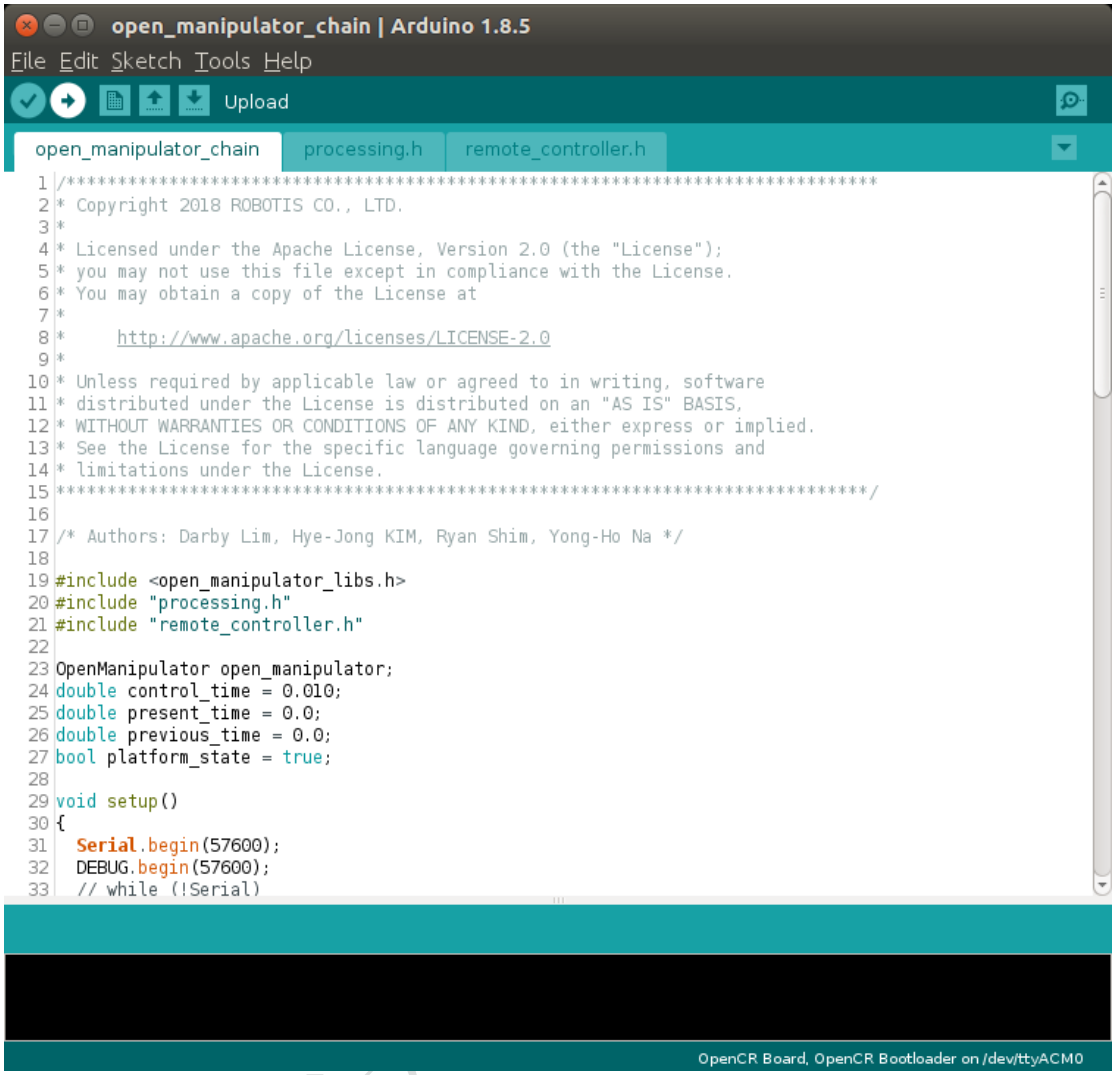
3.5 OpenCR Processing GUI 示教操作

3.5.1 上传控制器程序

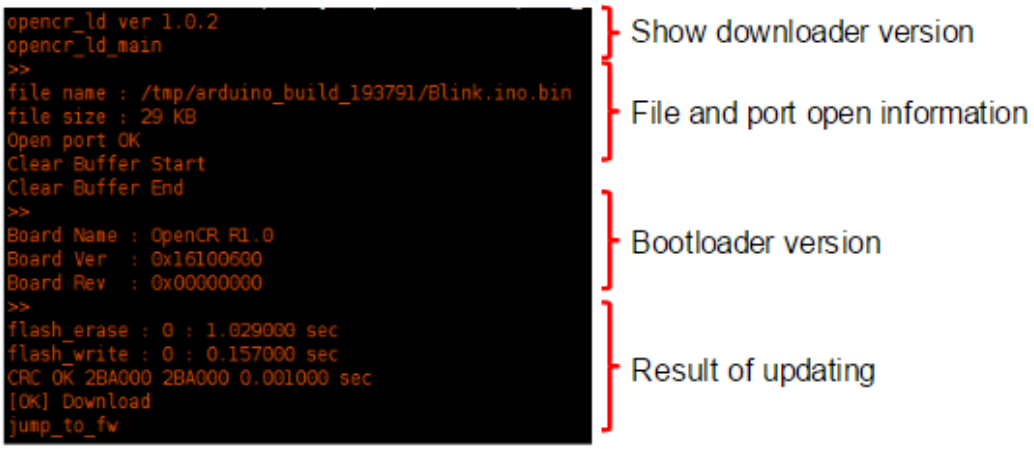
查找示例源代码。
转到 Examples> OpenManipulator> example> Chain>
open_manipulator_chain 上的 Arduino IDE 的 OpenCR。



将示例源上传到 OpenCR。



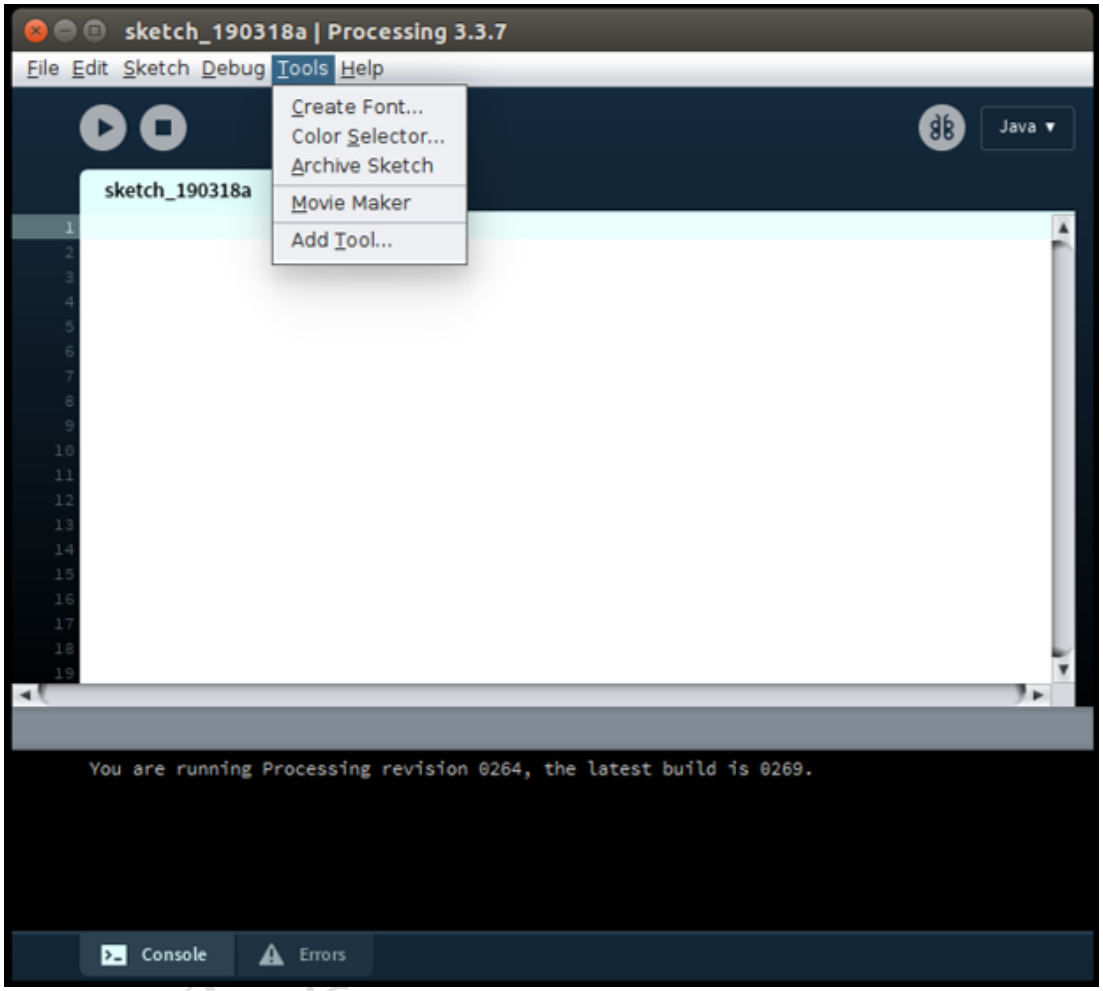
上传完成后，以下注释将显示在日志窗口中。



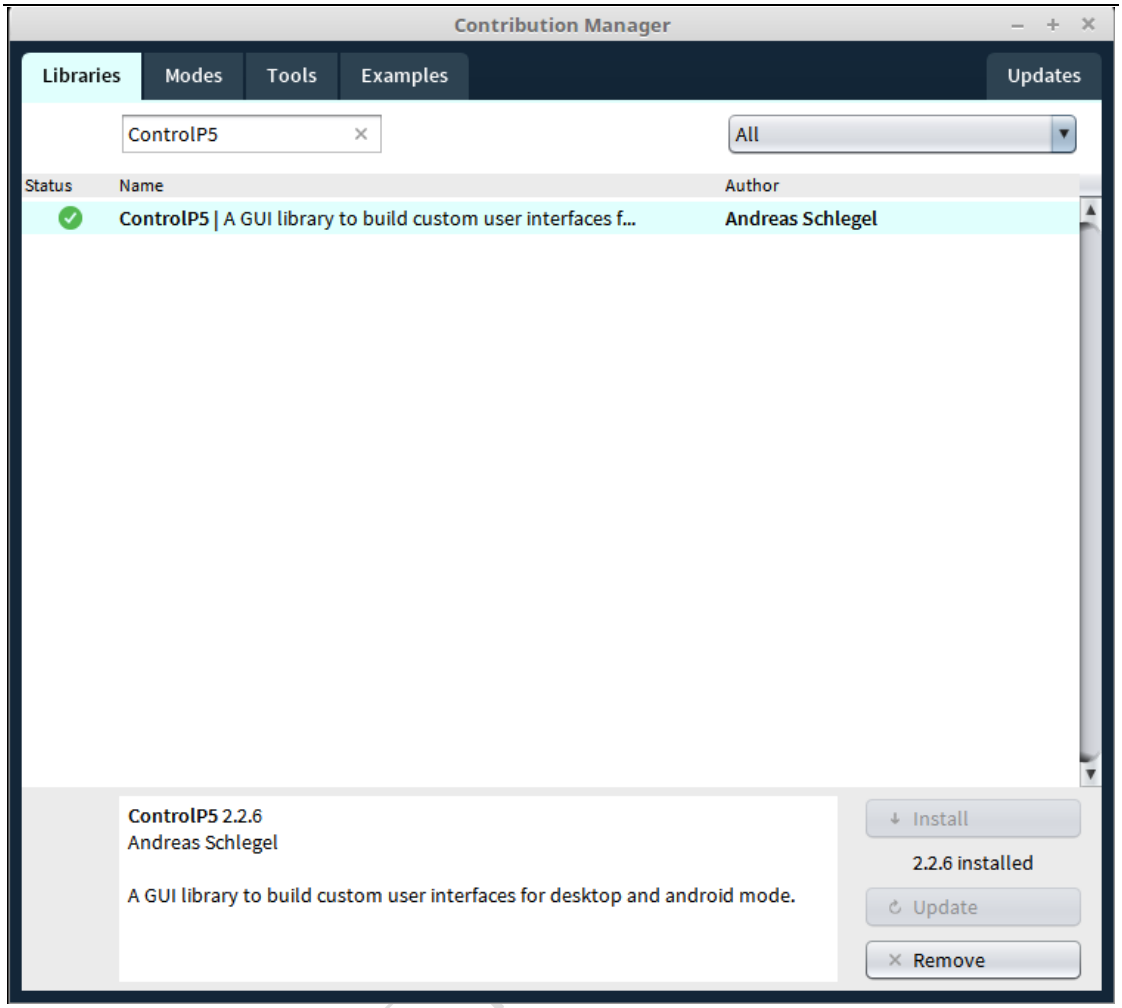
提示：如果 `cmd_read_board_name fail: 0xF020` 在上传过程中经常发生错误，请参考 OpenCR 电子手册进入固件恢复模式，并在进入模式后再次上传源代码。

3.5.2 设置 Processing（GUI）

1. 从链接下载 processing 并安装。
下载 processing：https：//processing.org/download/
2. 启动 processing 并转到 Tools> Add Tool..



3. 搜索 ControlP5 并安装。



4. 下载 OpenMANIPULATOR 的处理源代码。

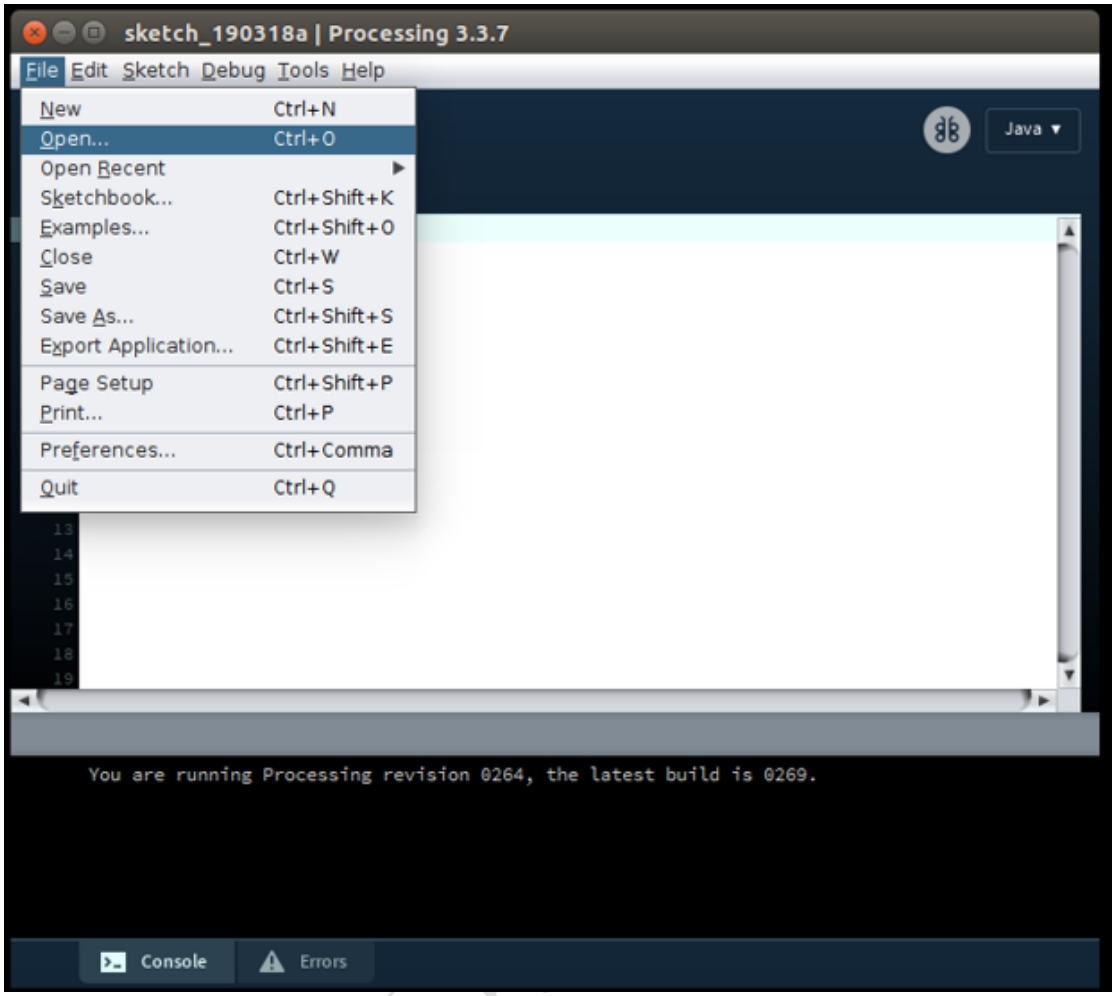
```
$git clone https://github.com/JTDQ/hanbot_processing.git
```

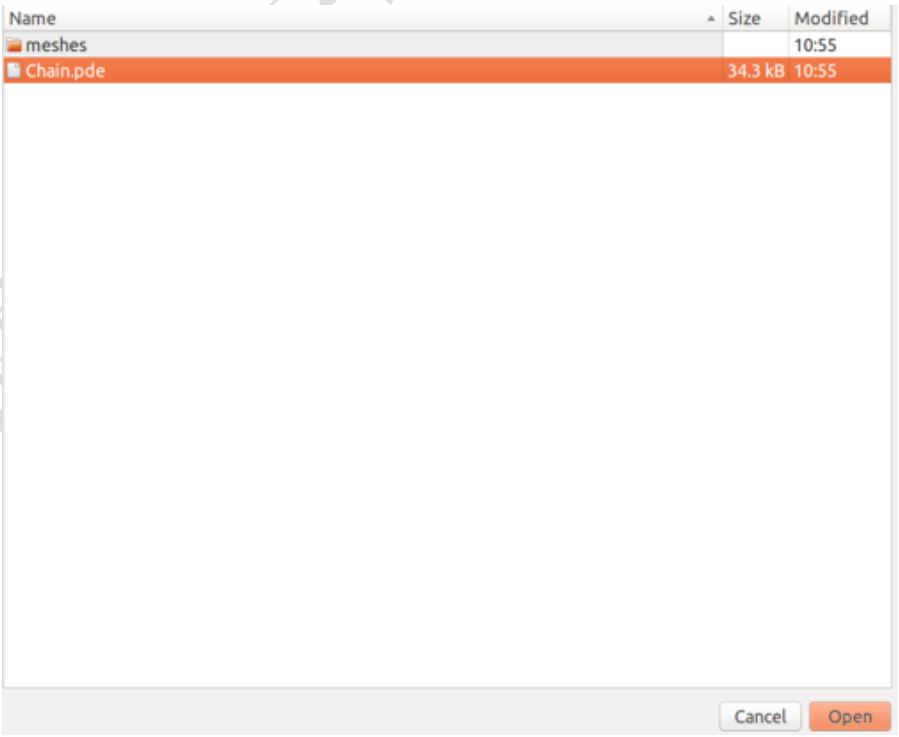
3.5.3 启动 Processing

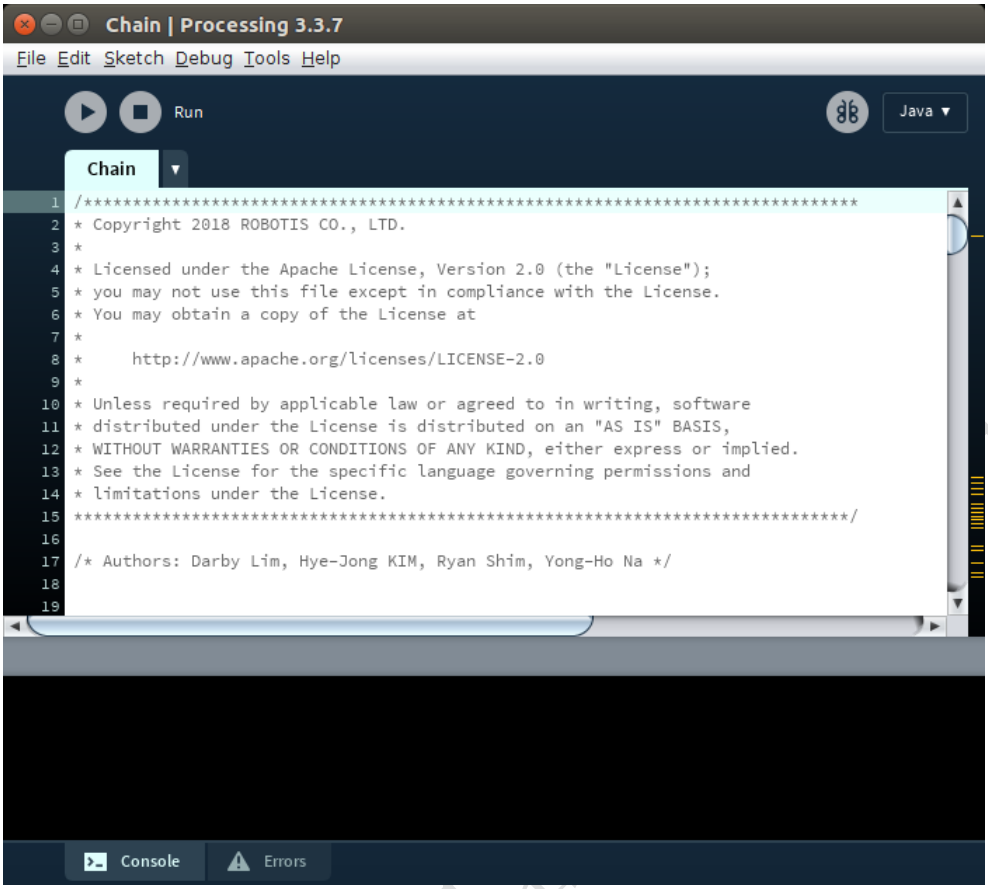
警告：在启动处理之前，必须将 OpenCR1.0 连接到 OpenMANIPULATOR-X。按下 Reset ButtonOpenCR1.0 以启用 OpenMANIPULATOR-X 的扭矩。

1. 将 USB 电缆和电源连接到 OpenCR1.0。
2. 打开 OpenCR1.0 开关。
3. 按下 Reset ButtonOpenCR1.0 并检查 OpenMANIPULATOR-X 的 DYNAMIXEL 是否启用了扭矩。
4. 搜索从下载的文件处理的文件夹，你下载的源代码 > open_manipulator_processing> Chain> Chain.pde 并打开它在处理 IDE。

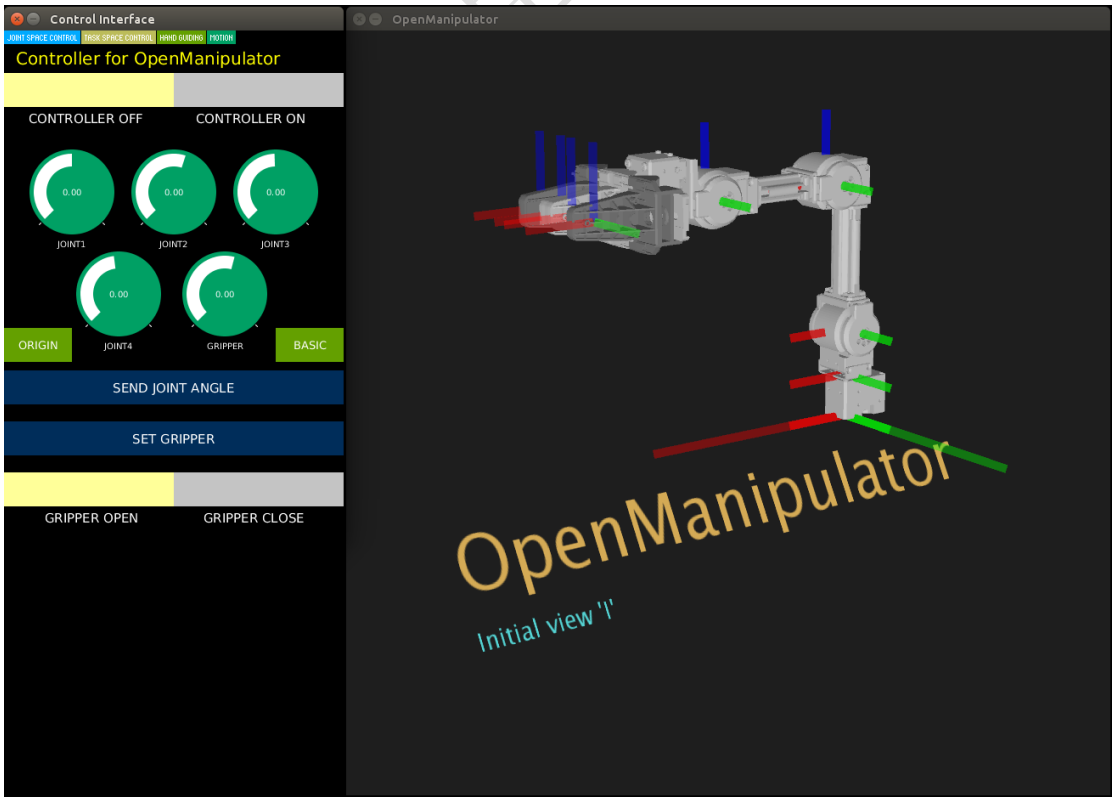
注意：在运行**处理示例**之前，将 **OpenCR OpenMANIPULATOR 示例**上载到 **OpenCR**。


sketch_190318a | Processing 3.3.7
File Edit Sketch Debug Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook... Ctrl+Shift+K
Examples... Ctrl+Shift+O
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Export Application... Ctrl+Shift+E
Page Setup Ctrl+Shift+P
Print... Ctrl+P
Preferences... Ctrl+Comma
Quit Ctrl+Q
Java
You are running Processing revision 0264, the latest build is 0269.
Console
Errors


Name
Size
Modified
meshes
10:55
Chain.pde
34.3 kB
10:55
Cancel
Open



5. 运行 processing 源代码，将显示以下图形 GUI。

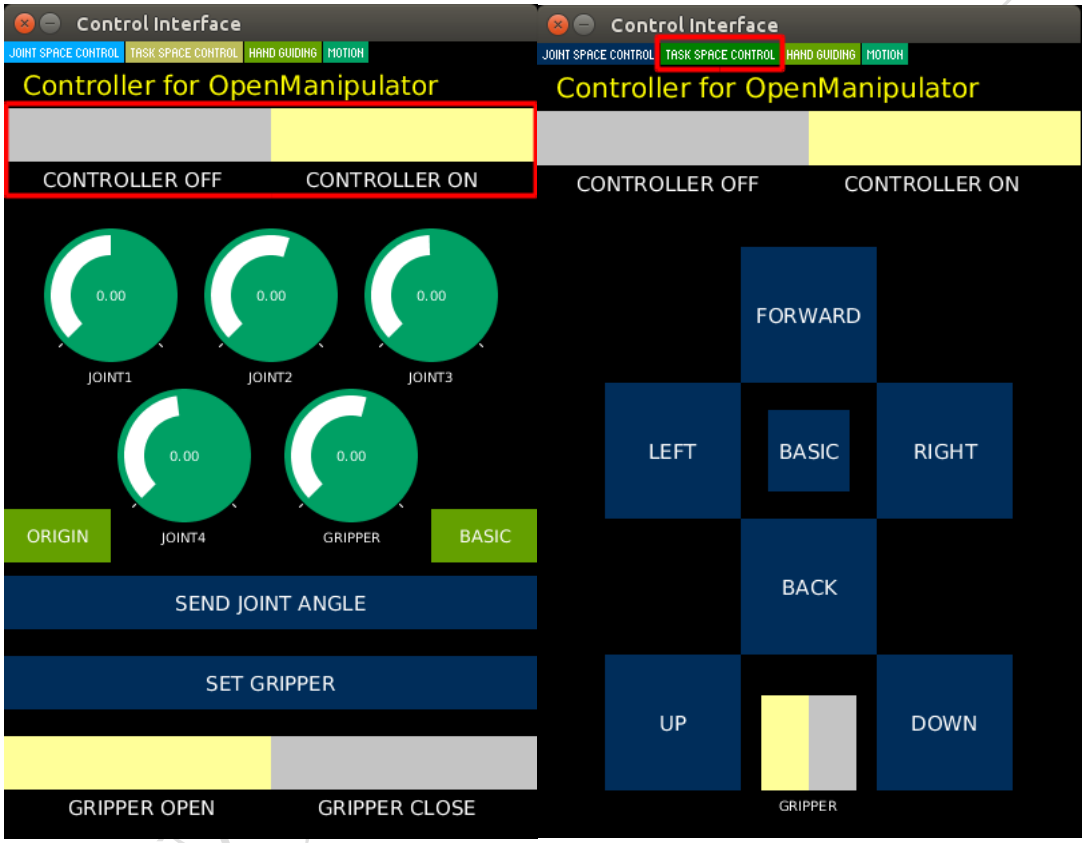


提示：如果处理控制台显示错误消息且 GUI 无法正常运行，请检查以下各项：

- 检查 ControlP5 是否正确安装。
- 检查图形驱动程序是否正确安装。
- 检查 Java 库是否已正确安装。

3.5.4 控制界面

注意：在运行 processing 源代码之前，将 OpenCR 示例源代码上传到 OpenCR 。
用户可以使用 processing 程序来操纵 OpenMANIPULATOR。
要操作 OpenMANIPULATOR，请单击切换按钮到 CONTROLLER ON。

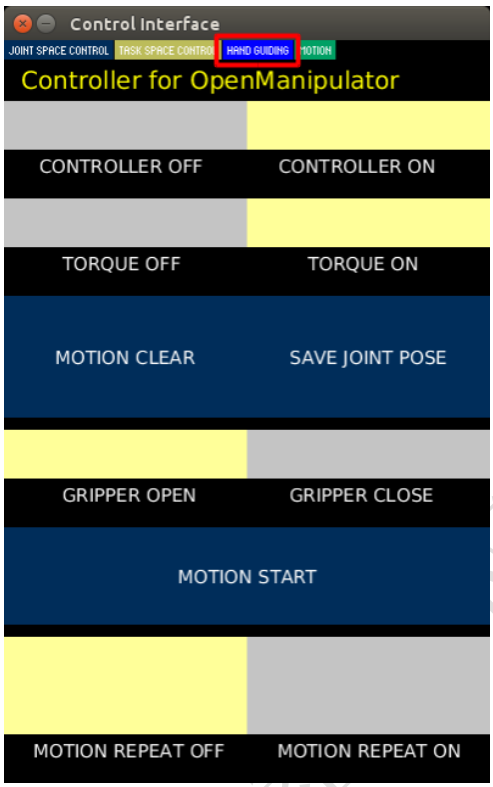


用户可以在关节空间内操作 OpenMANIPULATOR-X 。设置关节角度。然后点击 SEND JOINT ANGLE 按钮。并设置抓爪参数。然后点击 SET GRIPPER 按钮。
用户可以在任务空间中操作 OpenMANIPULATOR-X 。单击 TASK SPACE CONTROL 按钮更改选项卡。单击所需的方向按钮以操纵 OpenMANIPULATOR。

3.5.5 拖动示教再现

1. 单击 HAND GUIDING 按钮更改选项卡。
2. 用手抓住 OpenMANIPULATOR-X，然后将切换按钮单击到 TORQUE OFF。
3. 用手将 OpenMANIPULATOR-X 移至所需姿势。
4. 单击 SAVE JOINT POSE 以保存当前姿势。
5. 重复步骤 3 和 4 以创建演示。
6. 单击切换按钮以 TORQUE ON
7. 单击 MOTION START 按钮开始保存的姿势。

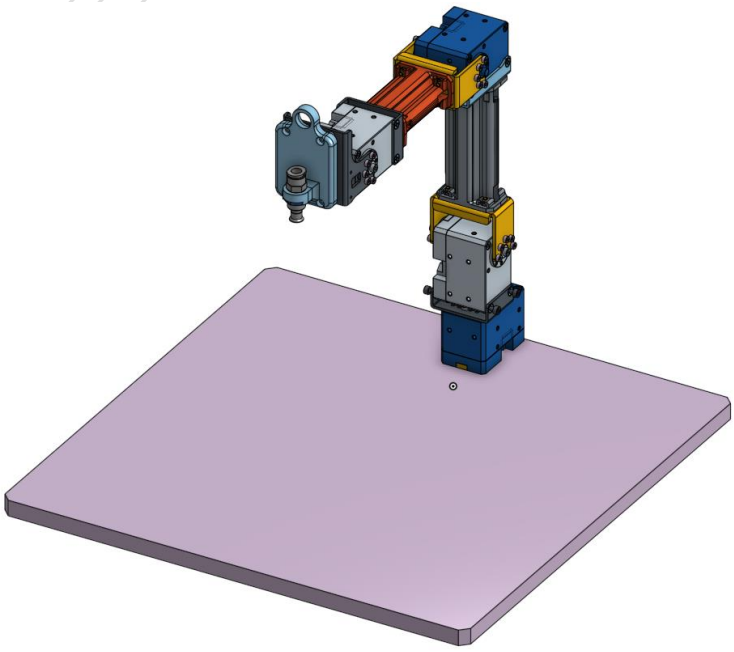
MOTION REPEAT ON 如果要重复演示，请单击切换按钮。保存的姿势之间的轨迹总时间与 2 秒相同。



3.6 末端工具更换

3.6.1 真空夹具

取下 Hanbot 的普通夹具并安装真空夹具。您可以从下表中的链接下载真空夹具的 STL 文件，然后将其通过 3D 打印机打印出来。



零件清单

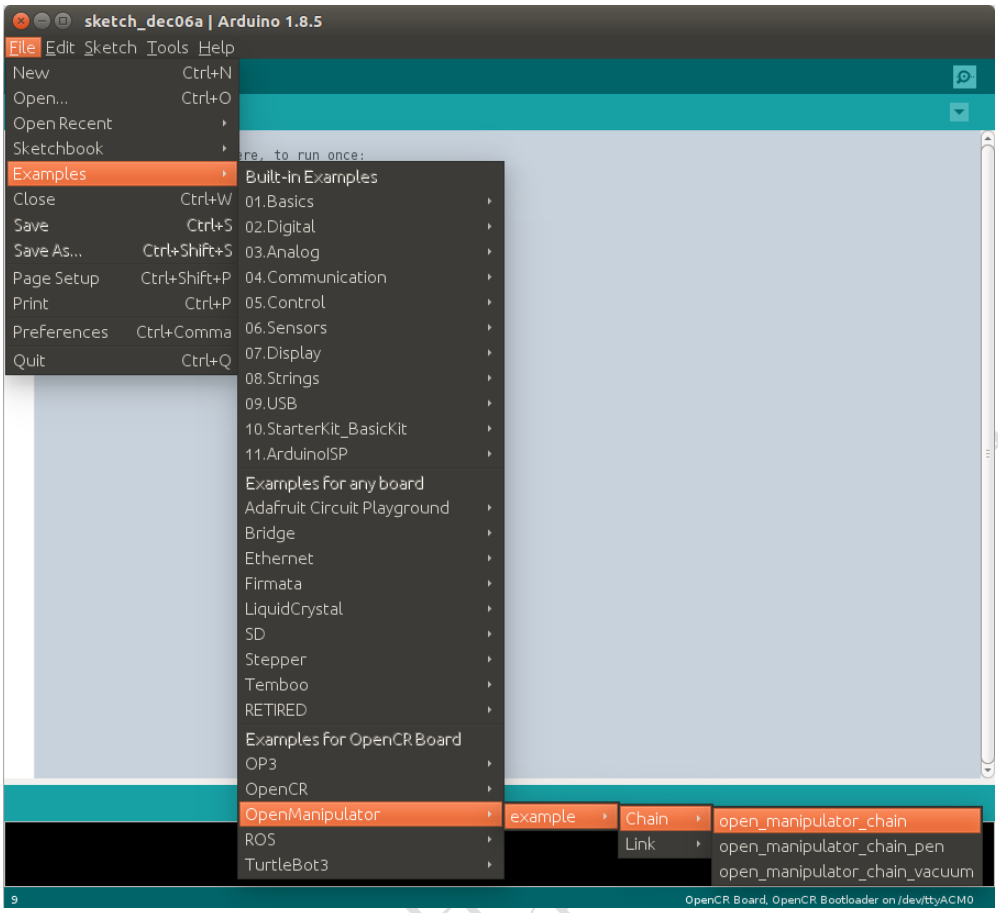
零件名称	数量
真空支架（3D 打印）	1
ARDUINO 4 继电器屏蔽	1
12V 空气泵电机	1
UD0640-20-C（空气管 60）	1
UD0860-20-C（空气管 80）	1
MSCNL6-1（耦合 60）	1
MSCNL8-1（耦合 80）	2
MVPKE8（吸盘）	1
MHE3-M1H-3 / 2G-1/8（控制阀）	1
NEBV-Z4WA2L-PE-2.5-N-LE2-S1（阀门电缆）	1

软件设置

查找示例源代码。

进入 `Examples` → `Hanbot` → `example` → `Chain` →

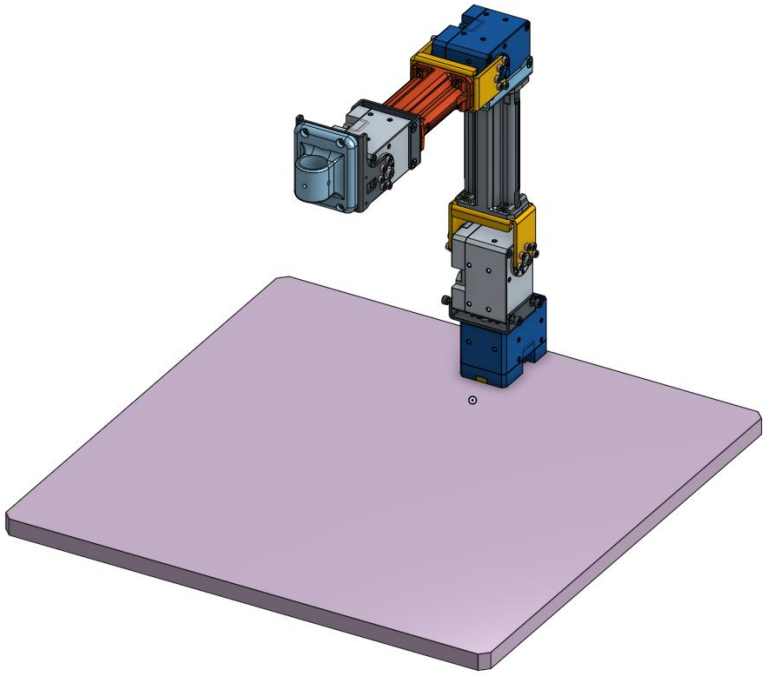
`open_manipulator_chain_vacuum` 上的 Arduino IDE 的 OpenCR。



吸硬币实际操作图：

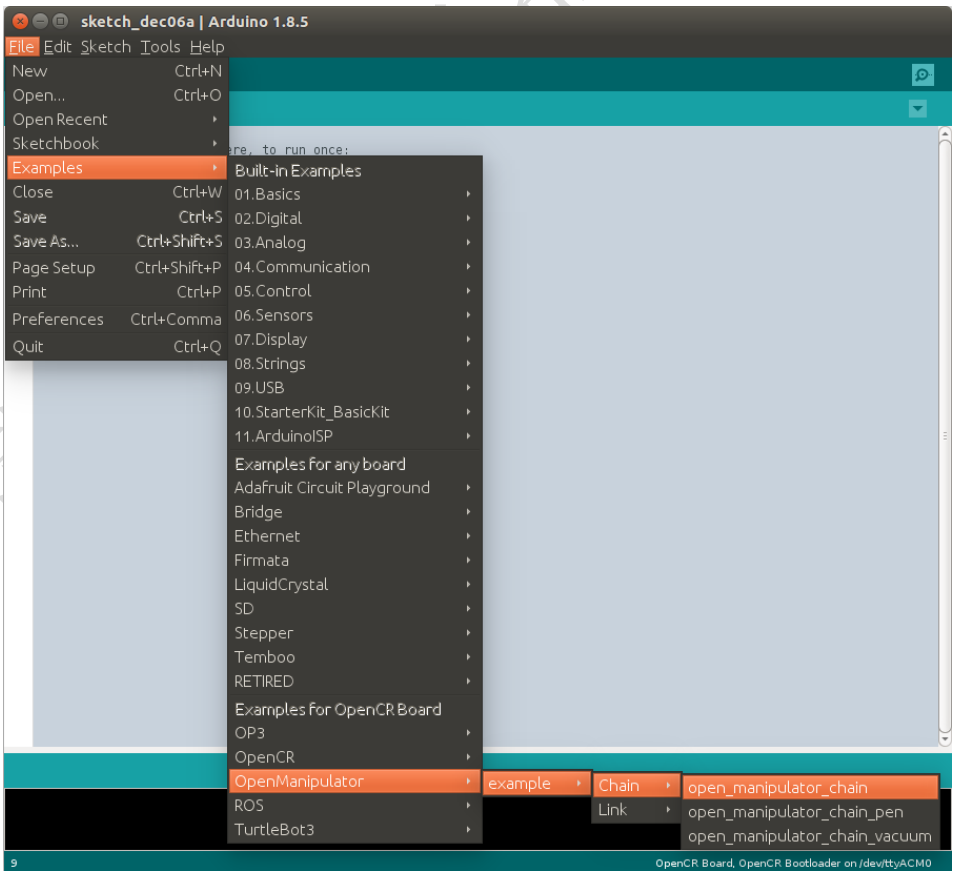
3.6.2 笔筒

取下 Hanbot 的普通夹具并安装笔架。您可以从下表中的链接下载笔架的 STL 文件，然后将其打印到 3D 打印机。

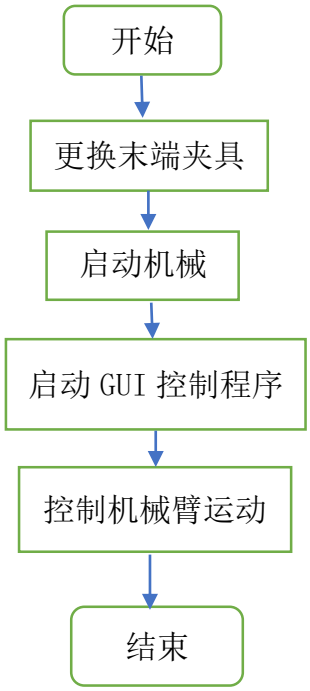


查找示例源代码：

进入 Examples→Hanbot→example→Chain→open_manipulator_chain_pen 上的 Arduino IDE 的 OpenCR。



操作流程：

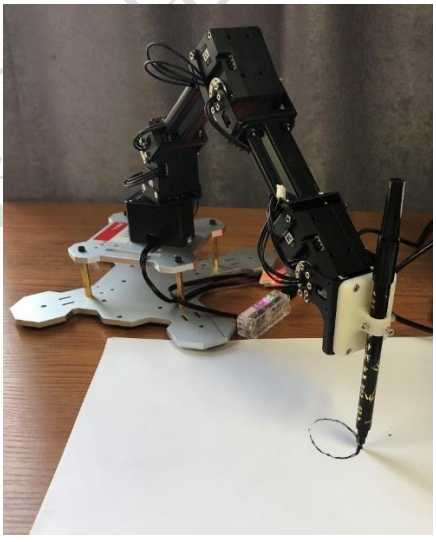


安装步骤如下：

1）拆下 Hanbot 的普通夹具

2）将笔筒套件安装在机械臂末端，笔筒的直径为 10mm。

3）把笔固定在笔筒中，拧紧螺丝，开始画画测试。

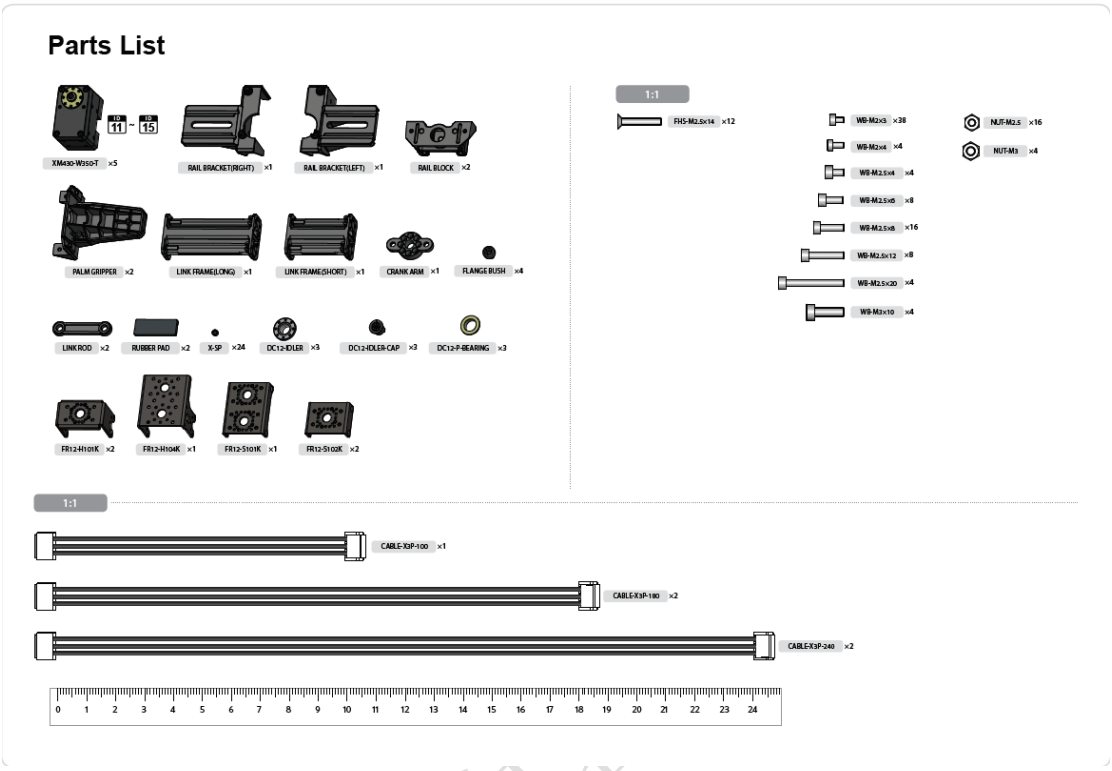


小结

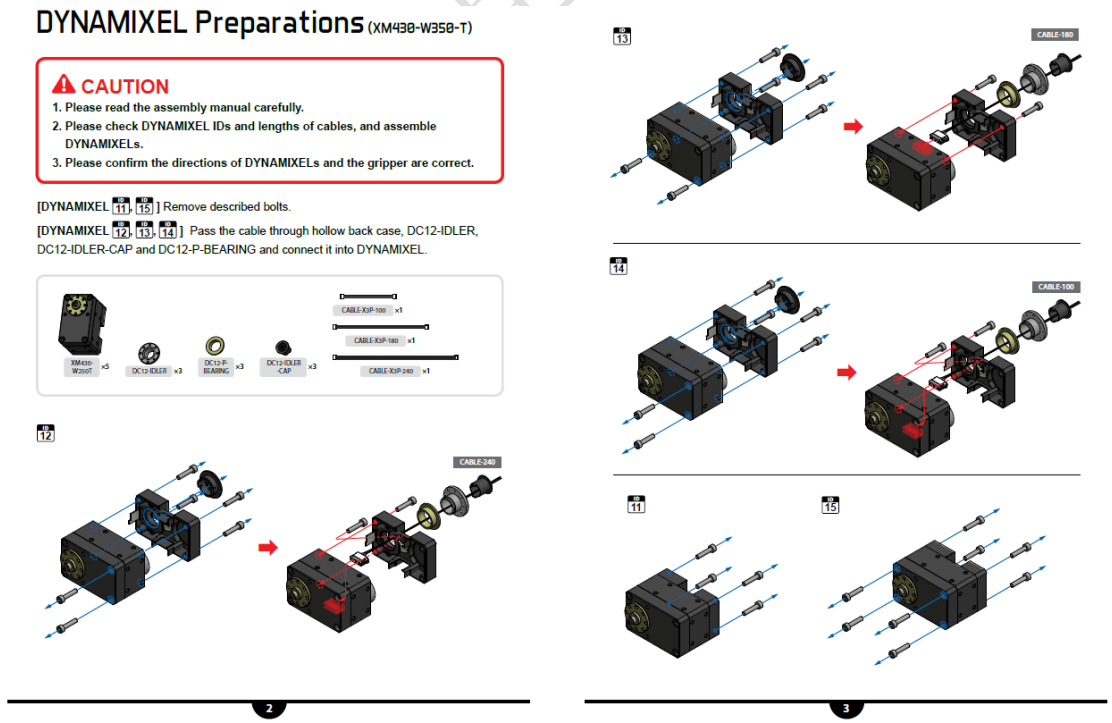
以 Hanbot 手持机械臂为例对机械臂基本使用进行了介绍。阐述了手持机械臂的功能和软件安装过程，通过例程的演示展现了机械臂的示教、遥操作已经在 Moveit 下的使用。最后给机械臂末端提供了拓展，将真空夹具和笔筒替换了机械臂的末端的夹爪，对机械臂的应用范围进行了拓展。

附录 手持机械臂组装指南

零件清单：

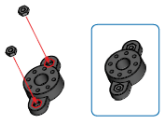


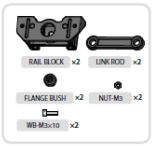
安装步骤：



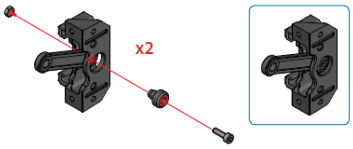
JING TIAN

1 Insert *NUT-M3* from underneath the *CRANK ARM*

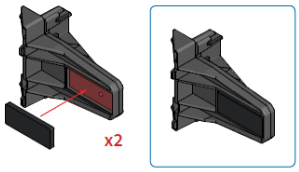




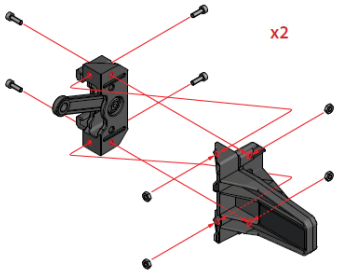
2 Assemble *LINK ROD* in the middle of the *RAIL BLOCK* using *FLANGE BUSH* and *WB-M3x10*

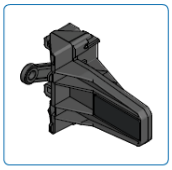


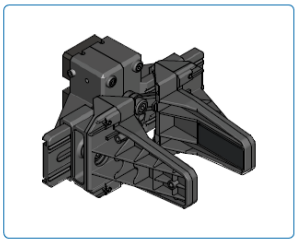
3 Attach *RUBBER PAD* to the inner side of the *PALM GRIPPER*

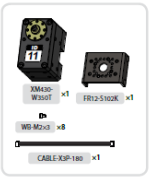


4 Assemble *LINK ROD* + *RAIL BLOCK* and *PALM GRIPPER* using *WB-M2.5x8* and *NUT-M2.5*



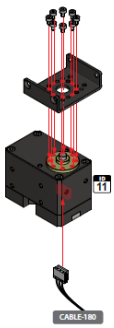


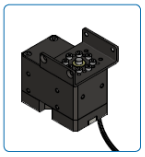


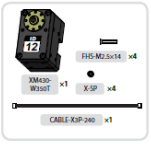


8 Assemble *FR12-S102K* and *DYNAMIXEL(ID 11)* using *WB-M2x3*

Connect *CABLE-X3P-180* to *DYNAMIXEL(ID 11)*



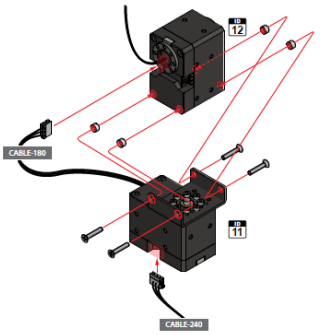


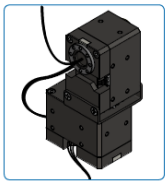


9 Insert *X-SP* into the bolt holes of *DYNAMIXEL(ID 12)* and assemble to *FR12-S102K* using *FHS-M2.5x14*

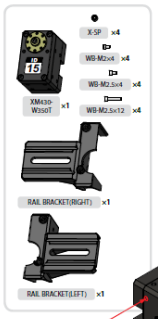
Connect *DYNAMIXEL(ID 12)* and *DYNAMIXEL(ID 11)* using *CABLE-X3P-180*

Connect *CABLE-X3P-240* to *DYNAMIXEL(ID 11)*





JING TIAN

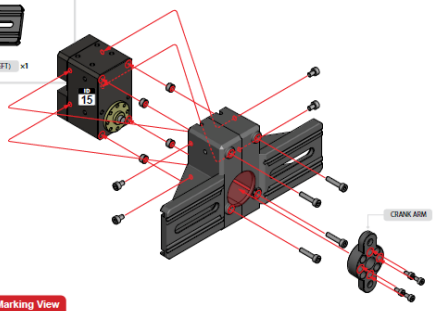


5

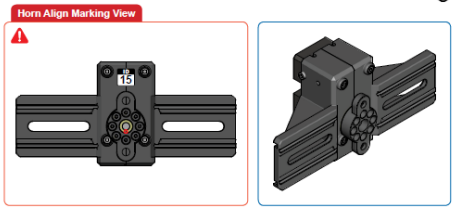
Assemble *RAIL BRACKET(RIGHT)* to *RAIL BRACKET(LEFT)*

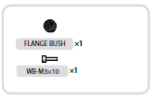
Assemble *DYNAMIXEL(ID 15)* to *RAIL BRACKET(RIGHT)* + *RAIL BRACKET(LEFT)* using *WB-M2.5x12* and *WB-M2.5x4*

Assemble *CRANK ARM* to the horn of *DYNAMIXEL(ID 15)* using *WB-M2x4*



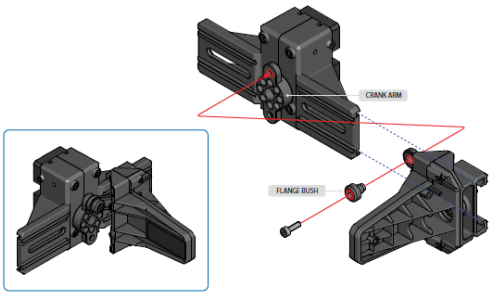
Horn Align Marking View



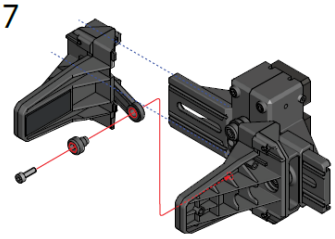


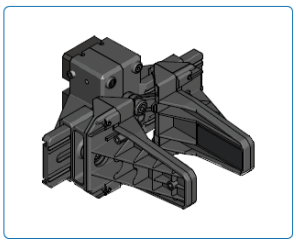
6

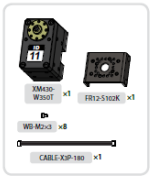
Slide *LINK ROD* + *RAIL BLOCK* + *PALM GRIPPER* into *RAIL BRACKET* Assemble *FLANGE BUSH* to *CRANK ARM* and *LINK ROD* using *WB-M3x10*



7





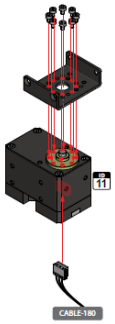


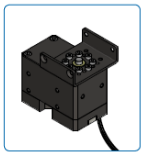
8

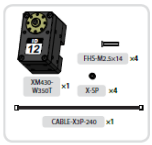
Assemble *FR12-S102K* and *DYNAMIXEL(ID 11)* using *WB-M2x3*

Connect *CABLE-X3P-180* to *DYNAMIXEL(ID 11)*

Horn Align Marking View





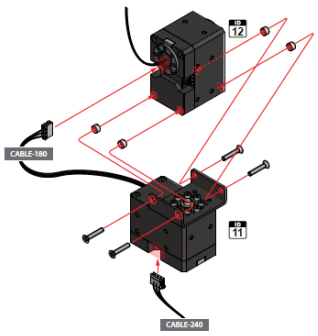


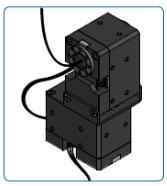
9

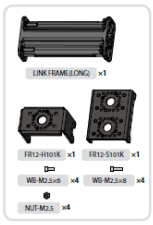
Insert *X-SP* into the bolt holes of *DYNAMIXEL(ID 12)* and assemble to *FR12-S102K* using *FHS-M2.5x14*

Connect *DYNAMIXEL(ID 12)* and *DYNAMIXEL(ID 11)* using *CABLE-X3P-180*

Connect *CABLE-X3P-240* to *DYNAMIXEL(ID 11)*



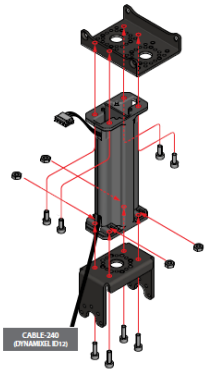


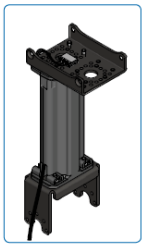


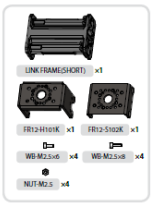
10 Pass ***CABLE-X3P-240*** from ***DYNAMIXEL(ID 12)*** through the hole of ***LINK FRAME(LONG)***

Assemble ***FR12-H101K*** to ***LINK FRAME(LONG)*** using ***WB-M2.5x8*** and ***NUT-M2.5***

Assemble ***LINK FRAME(LONG)*** to ***FR12-S101K*** using ***WB-M2.5x6***



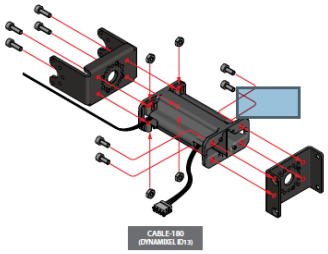


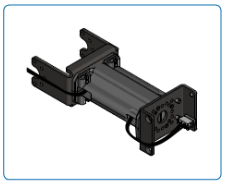


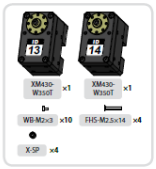
11 Pass ***CABLE-X3P-180*** from ***DYNAMIXEL(ID 13)*** through the hole of ***LINK FRAME(SHORT)***

Assemble ***FR12-H101K*** to ***LINK FRAME(SHORT)*** using ***WB-M2.5x8*** and ***NUT-M2.5***

Assemble ***LINK FRAME(SHORT)*** to ***FR12-S102K*** to using ***WB-M2.5x6***



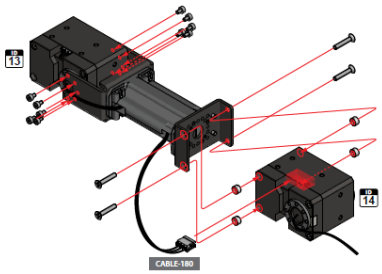


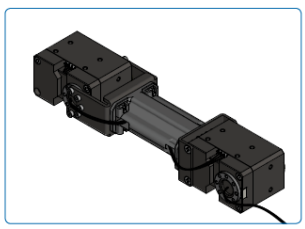


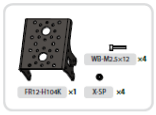
12 Assemble ***DYNAMIXEL(ID 13)*** to ***FR12-H101K*** using ***WB-M2x3***

Insert ***X-SP*** into the bolt holes of ***DYNAMIXEL(ID 14)*** and assemble to ***FR12-S102K*** using ***FHS-M2.5x14***

Connect ***DYNAMIXEL(ID 13)*** and ***DYNAMIXEL(ID 14)*** using ***CABLE-X3P-180***

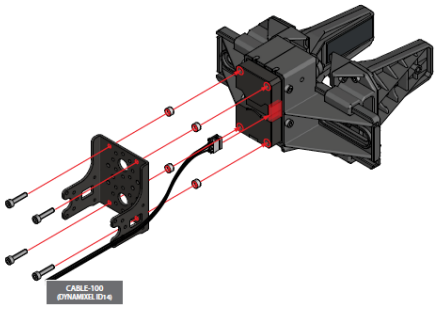


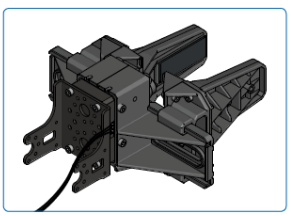


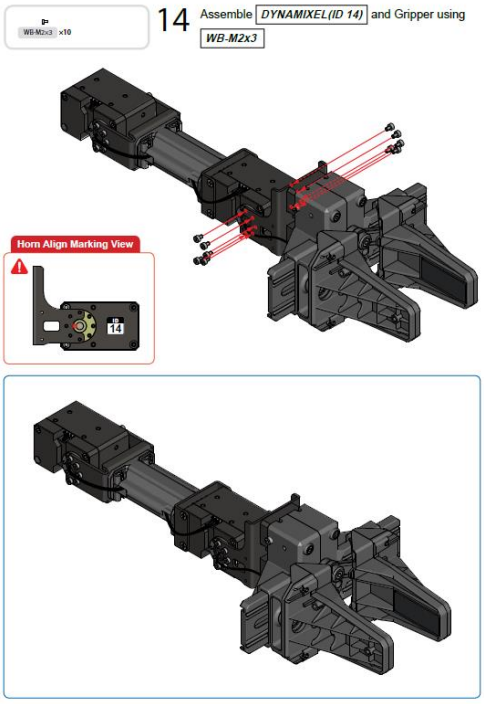


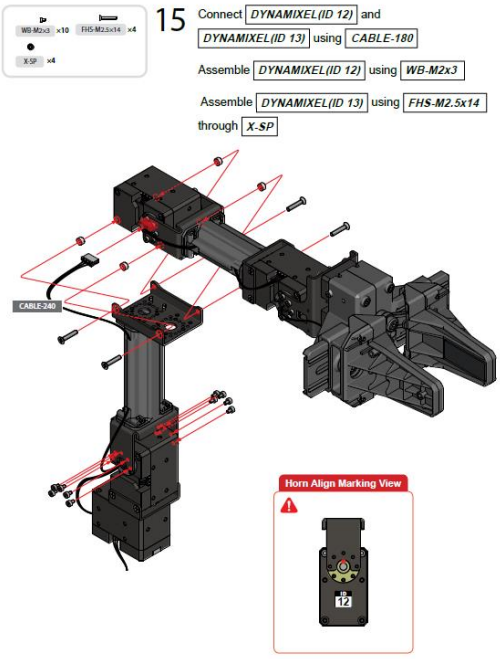
13 Connect ***DYNAMIXEL(ID 14)*** and ***DYNAMIXEL(ID 15)*** using ***CABLE-X3P-100***

Insert ***X-SP*** into the bolt holes of ***DYNAMIXEL(ID 15)*** and assemble ***FR12-H104K*** to Gripper using ***WB-M2.5x12***

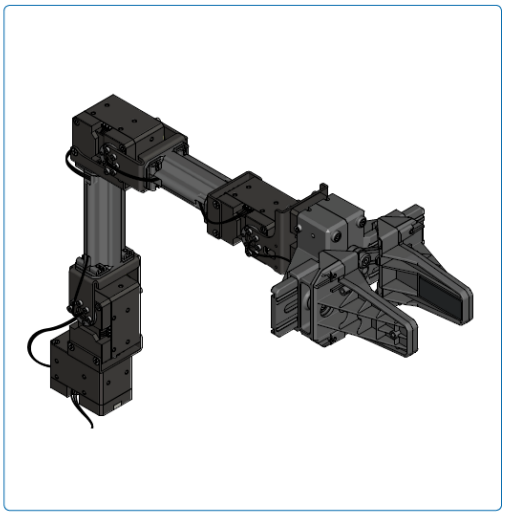


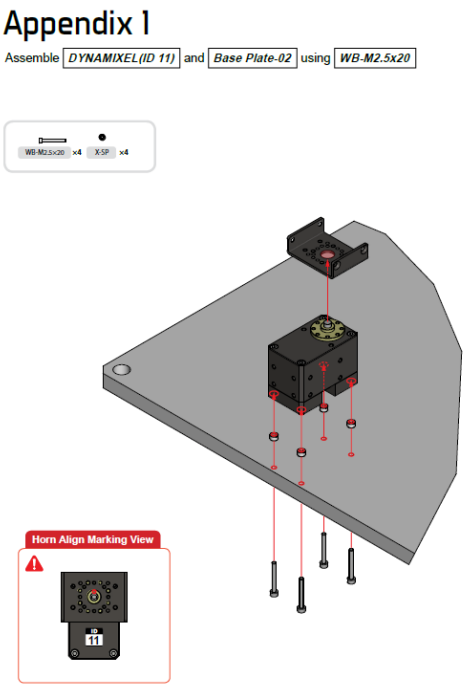



14
Assemble DYNAMIXEL(ID 14) and Gripper using WB-M2x3
Horn Align Marking View


15
Connect DYNAMIXEL(ID 12) and DYNAMIXEL(ID 13) using CABLE-180
Assemble DYNAMIXEL(ID 12) using WB-M2x3
Assemble DYNAMIXEL(ID 13) using FHS-M2.5x14 through X-SP
CABLE-240
Horn Align Marking View

底座安装：




Appendix 1
Assemble DYNAMIXEL(ID 11) and Base Plate-02 using WB-M2.5x20
Horn Align Marking View