

Table of Contents

Product Information

Introduction	1.1
1 Product Introduction	1.2

Basic Setup

2 Specification Description	2.1
-----------------------------	-----

Functions and Applications

3 Assembly and Wear Steps	3.1
4 SDK Development Methods	3.2
4.1 Development and Use Based on Python	3.2.1
API	3.2.1.1
Related Cases	3.2.1.2
4.2 Development and Use Based on ROS1	3.2.2
ROS Workspace Code and Usage	3.2.2.1
Common ROS Tool Commands	3.2.2.2
4.3 Based on serial communication usage	3.2.3
5 Successful Cases	3.3
5.1 MyArm M750 Usage Case	3.3.1
5.2 AGV + Mycobot280M5 Usage Case	3.3.2
5.3 Mercury X1 Usage Case	3.3.3

Support and Service

6 About Us	4.1
------------	-----

Acknowledgments

7 Acknowledgments	5.1
-------------------	-----

MyController-S570

Universal Intelligent Six-Degree-of-Freedom Robotic Arm

Core Documentation

This document contains comprehensive information ranging from product introduction, detailed technical parameters, to user guidelines and initial installation instructions. We will delve into the basic functions of MyController-S570, provide software development guides, and showcase successful application cases to help you understand how to effectively integrate MyController-S570 into various applications. In addition, we have provided a wealth of support and service information to ensure you receive the necessary assistance when facing any technical challenges.

gitbook-en

English version: <https://docs.elephantrobotics.com/docs/myController-S570-en>

gitbook-cn

Chinese version: <https://docs.elephantrobotics.com/docs/myController-S570-cn>

Document Description

Depending on your needs and the professional level of MyController-S570 application development, you can choose to follow the order from beginning to end or use it as a standalone reference. You can use the sidebar navigation on the left to jump to any section at any time. The entire document is divided into the following five major sections:

Product Information

The Product Information section will provide you with a basic overview of the robotic arm, including design intent, main functions, and application scenarios, helping you quickly understand the basic characteristics and usage environment of the product. In addition, this section will detail application examples and supported extended development, providing you with the necessary development guides and resources. The end of the section will provide relevant purchase links and channels for your convenience.

Product Parameters

The Product Parameters section will show you detailed technical specifications, including machine specifications, control core parameters, structural dimensions, and electrical characteristics. This information is crucial for understanding the technical standards and performance indicators of the product. In addition, it will provide Cartesian coordinate system information about the working range and accuracy of the robotic arm, serving as a reference for users who want to perform precise operations.

Functions and Applications

The Functions and Applications section provides a detailed introduction to the basic functions and software usage methods of the robotic arm, including system instructions and firmware functions. The software development guide offers guidance based on different development environments, such as Python and ROS, supporting technical

developers in application expansion. By showcasing successful application cases and providing supporting resources, it offers practical references and necessary support materials for a deeper understanding and use of the product.

Support and Service

The Support and Service section will provide you with a comprehensive troubleshooting guide and post-purchase service information, such as warranty and service terms, helping you quickly resolve issues and ensuring you understand your rights and obligations after purchase. In addition, the 'About Us' section strengthens users' understanding of the product design and manufacturers of MyController-S570, aiming to establish trust and brand loyalty.

Acknowledgments

We are very grateful for the time you have taken to read the MyController-S570 user manual. We hope this document will help you better understand and effectively use this robot, thereby stimulating your creativity. If you have any questions or need further assistance, please feel free to contact our customer support team. We look forward to seeing you complete innovative projects with MyController-S570 and welcome you to join our rapidly growing developer community.

Document Table of Contents

Summary

Product Information

- [1 Product Introduction](#)

Basic Setup

- [2 Specification Description](#)

Functions and Applications

- [3 Assembly and Wear Steps](#)
 - [4 SDK Development Methods](#)
 - [4.1 Development and Use Based on Python](#)
 - [API](#)
 - [Related Cases](#)
 - [4.2 Development and Use Based on ROS1](#)
 - [ROS Workspace Code and Usage](#)
 - [Common ROS Tool Commands](#)
 - [4.3 Based on serial communication usage](#)
 - [5 Successful Cases](#)
 - [5.1 MyArm M750 Usage Case](#)
 - [5.2 AGV + Mycobot280M5 Usage Case](#)
-

- [5.3 Mercury X1 Usage Case](#)
-

Support and Service

- [6 About Us](#)

Acknowledgments

- [7 Acknowledgments](#)

1. Product Overview

MyController S570



Portable wear, precise motion capture.

Product Introduction

The myController S570 is the first portable exoskeleton data acquisition device developed by Elephant Robotics, compatible with various types of robots. It covers scenarios such as data acquisition, remote operation, and end-to-end application development, helping researchers and developers combine humanoid, wheeled, and other types of robots in educational and training contexts, broadening the scope of teaching and research practices, and promoting the dissemination and mastery of knowledge and skills.

Applicable Fields

Research and Development: In the field of research and development, it boasts a data acquisition speed of up to 100Hz, ensuring efficient collection of robot end-to-end AI data and motion capture tasks. With data reproduction and deep processing technologies, it can fully meet the needs of robot algorithm research and development in complex scenarios, providing a solid data foundation for scientific innovation.

Education and Training: In the education and training field, its unique advantage of variable configurations allows it to be flexibly combined with robots of various structural forms such as 5-degree-of-freedom, 6-degree-of-freedom, and 7-degree-of-freedom. This enables the creation of highly intuitive robot control experiences for students and researchers, greatly helping students improve their understanding of robots and advancing the popularization and deepening of robotics knowledge.

Robot Remote Operation: For robot remote operation, it achieves seamless collaboration with various robots by providing a standard Python driver library and ROS development support, successfully enabling robot motion data collection and remote control functions. The standard data format and abundant open-source examples significantly

lower the barrier for remote robot operation, allowing even entry-level developers to easily get started and enjoy the convenience and fun of remote robot control.

Product Advantages

Ultra-light, Enjoy Long-lasting Experience

With a lightweight design, the weight of a single arm is less than 800g, approximately the weight of three smartphones, greatly reducing the burden of long-term wear and ensuring that users can easily handle the device with no significant fatigue after prolonged use.

Flexible and Precise Human Arm Simulation

With 12 degrees of freedom, it can precisely simulate human arm movements. Whether it's complex multi-angle movements or all-around posture changes, it can accurately capture and control, providing powerful support for motion data collection and analysis.

Convenient Interaction, Control at Your Fingertips

The end-control device is uniquely designed with grips on both arms, integrating two buttons and a joystick. This design makes control and interaction highly intuitive, allowing users to easily issue various commands and conveniently adjust the device status.

Versatile Data Collection to Satisfy Data Needs

With diverse data collection capabilities, its universal design excels in data acquisition. It can not only accurately record motion information but also effectively capture environmental parameters, fully meeting the diverse data needs in research and development.

Worry-free Development, Empowering Innovation

Known for its developer-friendly features, it is fully compatible with Python and ROS, two major development environments, seamlessly connecting to various developer tools. The rich programming interfaces open up vast innovation space for developers of all levels, allowing easy construction of basic applications and deep development of advanced applications.

Precise Perception, Capturing Subtle Movements

Equipped with 4096-bit high-precision magnetic encoder sensors for all joints, it captures every subtle movement change with exceptional accuracy and achieves perfect motion replication. It sets a new benchmark in motion capture, providing solid assurance for high-precision data acquisition.

High-speed Sampling, Locking in Data Essence

With a data sampling rate of up to 100Hz, it handles high-speed motion scenarios with ease. It can accurately capture fleeting motion changes, ensuring that the collected data is complete, accurate, and of high quality, providing a reliable data source for subsequent data analysis and applications.

Multi-connectivity, Adapting to Various Scenarios

It supports various connectivity options such as USB, wireless network (OTA firmware support), and Bluetooth, allowing users to flexibly choose based on different application scenarios. Whether for stable indoor connections or convenient outdoor remote control, it can be easily implemented, greatly enhancing the device’s application flexibility and adaptability to various scenarios.

Real-time Insight, Enhancing Interaction Quality

Equipped with a 2-inch display and custom software, it provides real-time feedback on device status and data information. Users can intuitively monitor device operations without relying on external devices, promptly adjusting their operational strategies, significantly improving user interaction experience and operational convenience.

Modular Design, Flexible to Meet Needs

Adopting a modular design concept, the body structure is clear and straightforward. This not only facilitates future upgrades and maintenance but also allows for customization based on the unique needs of different users. It easily adapts to a variety of complex and changing application scenarios, providing tailored solutions for users.

Design Goals

Design Goal	Functionality
Universal Multi-function Platform	The myController S570 is suitable for a wide range of applications, including education, research, and commercial demonstrations. Its six degrees of freedom and two arm spans (750mm and 650mm) support performing complex motion control tasks such as precise positioning and path planning in various working environments.
Research and Education Support	The myController S570 is ideal for machine learning and AI research, capable of performing high-precision experiments and technical demonstrations. It supports end-to-end data training and reproduction, as well as basic visual tasks, making it an ideal device for labs.
Programmability and Expandability	The myController S570's high programmability allows users to customize and program according to emerging technologies, meeting future technical needs and optimizing operations and experimental results.
Technological Innovation and Knowledge Dissemination	In commercial demonstrations, the myController S570 can serve as a platform to showcase the latest technological achievements, aiming to enhance the public’s understanding of science and technology, and promote the commercialization of technological innovations.

Product Features

Feature Description	Details
14 Degrees of Freedom Modular Design	A perfect combination of flexibility and scalability, suitable for complex operations.
Industrial-Grade Digital Servo Motors	Ensures precise control and long-term stable operation of the robotic arm.
High-Precision Encoders	Provides accurate position, speed, and acceleration information to optimize performance.
Supports Multiple Development Environments	Includes Python and ROS, meeting diverse development needs.
Supports Localized Drag Teaching	An intuitive learning and operation method, no external devices required.
High-Speed Data Interfaces	Offers multiple status interfaces, supporting complex applications.
Custom Embedded Software	Provides a user-friendly interface to simplify complex tasks.
Built-in 2-inch Display	Displays real-time robotic arm status and operation feedback.
Supports Multiple Connection Methods	Flexible application in various scenarios, seamlessly integrates with existing systems.
Central Symmetry Configuration	Ensures balanced stability and improves operation precision.

Product Value

Enhanced Experimental and Research Capabilities	myController S570 provides a high-precision platform for researchers to conduct complex data analysis and algorithm validation.
Improved Educational Quality	In educational environments, this device offers practical opportunities for students to better understand theories and develop practical skills.
Increased Commercial and Display Opportunities	myController S570 is not limited to laboratory use; its application in technical exhibitions and public demonstrations can attract viewers and potential customers, enhancing the interactivity and appeal of technology showcases.

Industry Contribution

Promoting STEM Education	By providing advanced teaching tools, it inspires students' interest in science, technology, engineering, and mathematics, cultivating future innovators.
Driving Technology Adoption	By offering an easy-to-use and highly flexible robotic platform, it lowers the technological barrier, enabling more institutions to access and utilize advanced robotics technology.
Fostering Innovation and Personal Development	Provides an open platform for developers and engineers to support interdisciplinary learning and innovation, cultivating key talent for the future of robotics and related fields.

2. Product Applications



Target Users

Higher Education Institutions and Research Laboratories	The myController S570 is designed as a teaching and research tool for high-precision experiments and technical demonstrations. It effectively supports complex data analysis, algorithm development, and validation, significantly enhancing research quality and educational effectiveness.
Technical Developers and Engineers	Supporting Python and ROS, the myController S570 is ideal for professionals who need customized programming and system integration. Its modular design and high programmability make it an ideal platform for developing and testing new control algorithms or robotic applications.
Commercial Display and Public Exhibition Organizers	myController S570 showcases its precise operation, becoming the preferred device for technological displays and product demonstrations. Dynamic demonstrations not only attract viewers but also enhance engagement, effectively promoting technological innovations and products.
Innovative Enterprises and Startups	myController S570 offers strong support for businesses seeking to integrate cutting-edge robotics technology to enhance product functionality or optimize production processes. Its exceptional performance and adaptability make it an ideal choice for exploring new technologies and solutions.

Application Scenarios

User Group	Application Scenarios	Benefits
Educators and Students in Education	- STEM Education - Robotics Projects - Interdisciplinary Research Projects - Robotics Educational Tools	- Increases student interest in technology - Enhances hands-on abilities and problem-solving skills - Fosters innovative thinking and teamwork - Serves as a tool for courses in robotics design, circuit design, and mechanics
Technical Developers and Engineers	- Prototype Development - Experimental Research - Algorithm Testing and Validation - Robotic Kinematics Verification - Complex Scenario Application Testing	- Accelerates research progress - Connects theory with practice - Promotes technological innovation - Validates robotic kinematics and complex scenario applications

User Group	Application Scenarios	Benefits
Commercial and Marketing Professionals	- Exhibitions - Technical Demonstrations - Brand Promotion	- Attracts potential customers and investors - Demonstrates company technological strength and innovative products - Enhances brand influence
Engineers and Technical Developers	- Robotic Kinematics Verification - Remote Control Scenarios Exploration and Development - Machine Learning, AI, and Vision-Based Tasks	- Explores and develops robotic remote control applications - Expands the practicality of robotic technology applications and scenarios

3. Supported Extended Development

Python	- Provides interfaces for joint angle control, servo motor control, servo motor parameter configuration, Cartesian coordinate system control, etc. - Supports advanced motion control and precise positioning, suitable for complex robotics research and application development. - The driver library can be installed via PyPI, offering comprehensive interface support, suitable for more complex educational or research projects.
ROS	- Supports basic MoveIt functions such as keyboard control and basic path planning, with support for both ROS1 and ROS2. - Strengthens real-time robotic arm display and status information collection through RVIZ simulation environment, suitable for advanced robotic path planning and operation research.
Hardware Interface	- Includes USB, etc., making it easy to connect various sensors and actuators.
Software Library	- Provides a rich set of open-source libraries and APIs to simplify the development process.
System Compatibility	- Compatible with Windows, Linux, and MacOS, adaptable to various development environments.

4. Purchase Link

If you are interested in purchasing this device, please click the following links:

Taobao: <https://shop504055678.taobao.com>

Shopify: <https://shop.elephantrobotics.com/>

AliExpress: <https://elephantrobotics.aliexpress.com/store/1101941423>

I. Robot Parameter Description -- MyController S570

In the first chapter, we explored the product's selling points and design concepts, providing you with a panoramic view for a high-level understanding of the product. Now, let's move on to Chapter 2 — Robot Parameter Description. This chapter will be key to understanding the technical details of the product. By understanding these technical parameters in detail, you will not only appreciate the advanced and practical nature of our product but also be able to effectively use these technologies to meet your specific needs.

1 Product Specifications

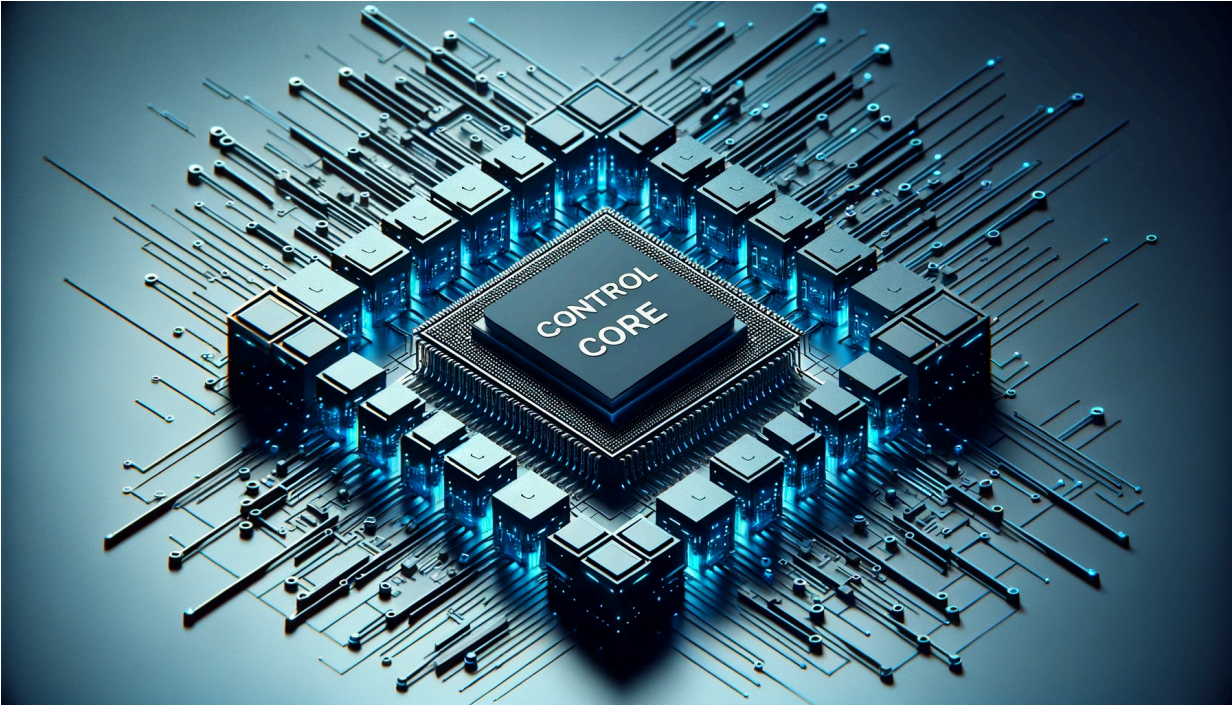
1.1 Robot Parameters

Specification	Parameter
Model	MyController S570
DOF	12 (6+6)
Horizontal Extension Range	570
Total Span	1500mm
Weight	1.5kg
Power Specification	8.4V2A
Rotation Ability	+/- 180°
End Effector	Dual-finger control + dual-button control
USB Connection	Type-C
Atom End	5*5 LED matrix
Communication Frame Rate	Up to 100Hz

1.2 Basic Software Functionality Support

Function/Development Environment	Availability
USB Control	Supported
Wireless Control	Supported
Windows	Supported
Linux	Supported
MAC	Supported
ROS	Supported
Python	Supported

2 Core Control Parameters



2.1 Main Controller Specifications

Indicator	Parameter
Main Control	M5Stack-basic
Main Control Model	ESP32
CPU	240MHz Dual-core.
	600 DMIPS, 520KB SRAM.
	Wi-Fi, dual-mode Bluetooth
Bluetooth	2.4G/5G
Wireless	2.4G 3D Antenna
Inputs	IN1, IN2, IN3, IN4, IN5, IN6
Outputs	OUT1, OUT2, OUT3, OUT4, OUT5, OUT6

2.2 Auxiliary Controller

Indicator	Parameter
Auxiliary Control	Atom
Auxiliary Control Model	ESP32*2
Auxiliary Controller Core Parameters	240MHz Dual-core.
	600 DMIPS, 520KB SRAM.
	Wi-Fi, dual-mode Bluetooth
Auxiliary Controller Flash	4MB
LCD Display	2.0"@320*240 ILI9342C IPS Panel,
	Max Brightness 853nit
Type C	*1

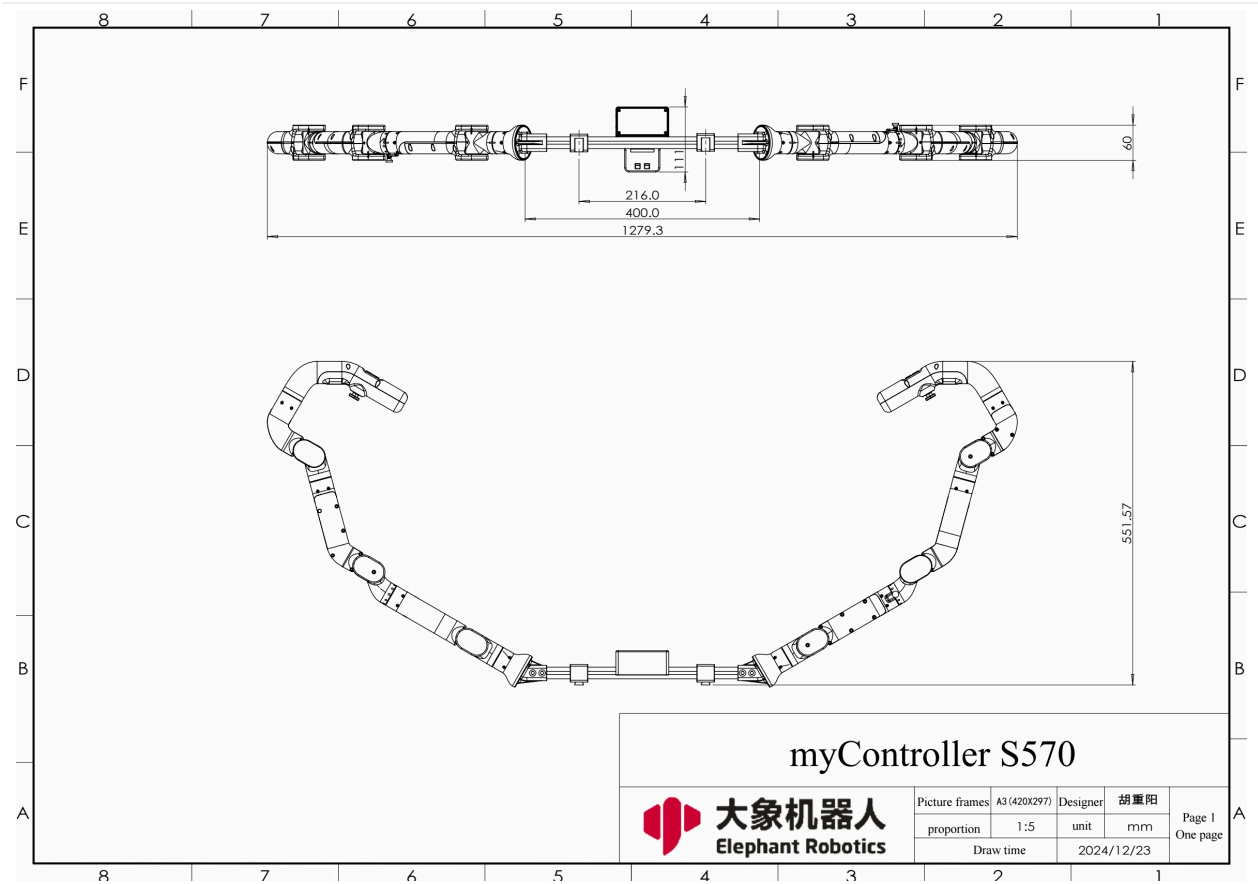
2.3 Programming Ports

Label	Function	Notes
Current Position	Read	
Current Speed	Read	
Current Acceleration	Read	
Button Status	Read	
4-Directional Sensor Status	Read	
Gyroscope Data	Read	
LED Light Color	Read/Write	

2.4 Related Documentation Support

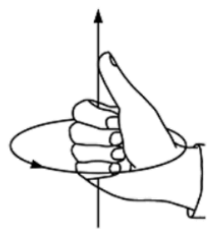
Label	Function	Notes
3D-Step	Supported	
2D Specifications	Supported	

2.5 Exoskeleton operating equipment drawings

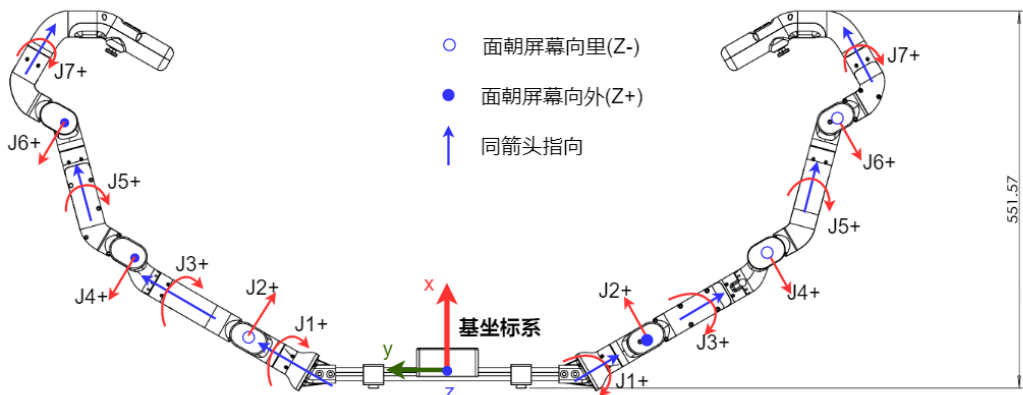


2.6 kinematic model

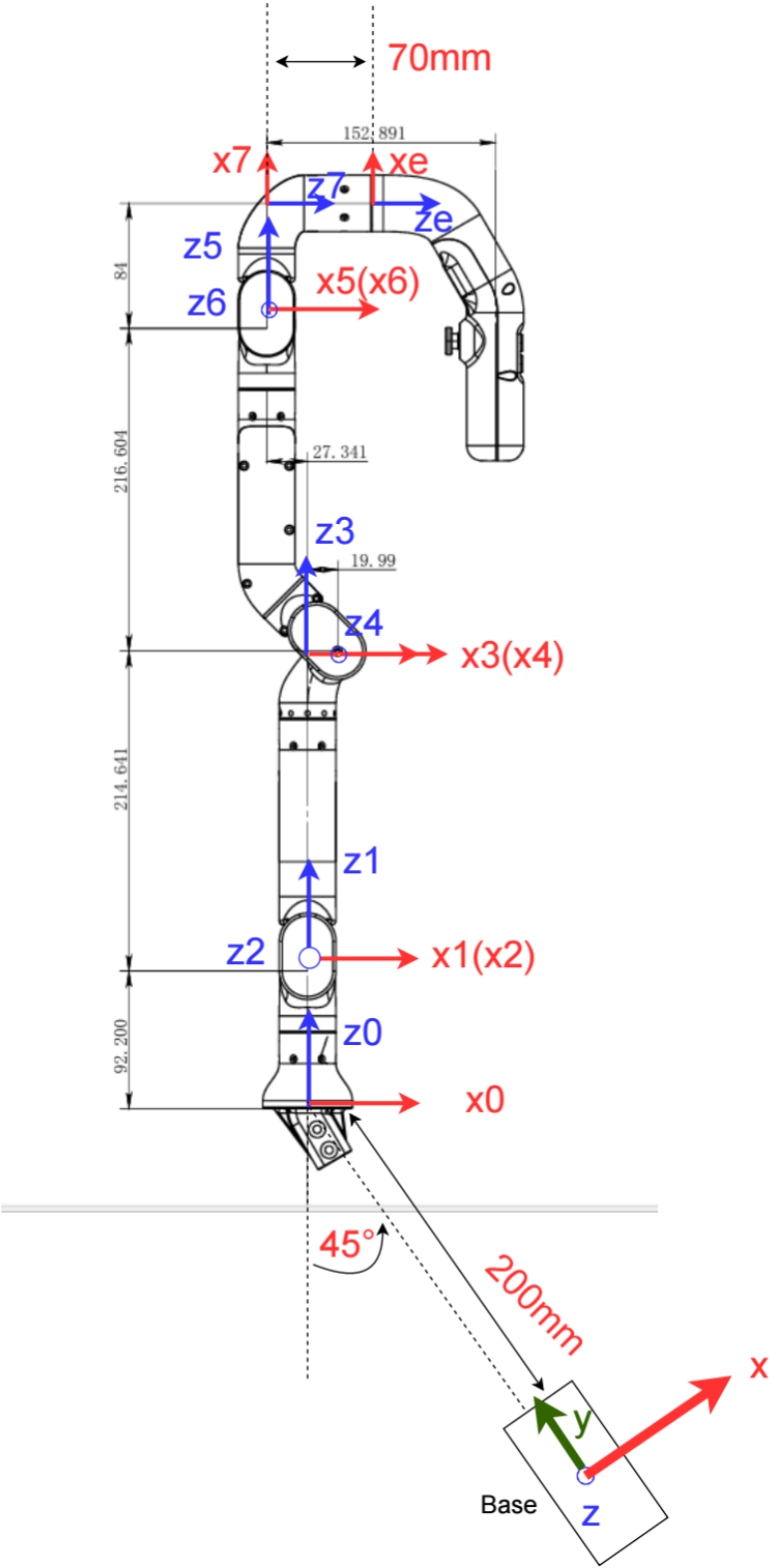
右手定则:



右手大拇指指向旋转轴方向，图示的旋转方向为关节旋转的正向。



2.7 DH Parameter



MyController S570 Wearing Method

1. USB Serial Connection

【Step 1】 First, wear the vest and fasten it properly. Then, fix the exoskeleton onto the vest.





【Step 2】 After connecting, turn on the red power button on the exoskeleton. Once the power is on, the built-in screen on the exoskeleton will start up, displaying the specific values of each joint and the button values.

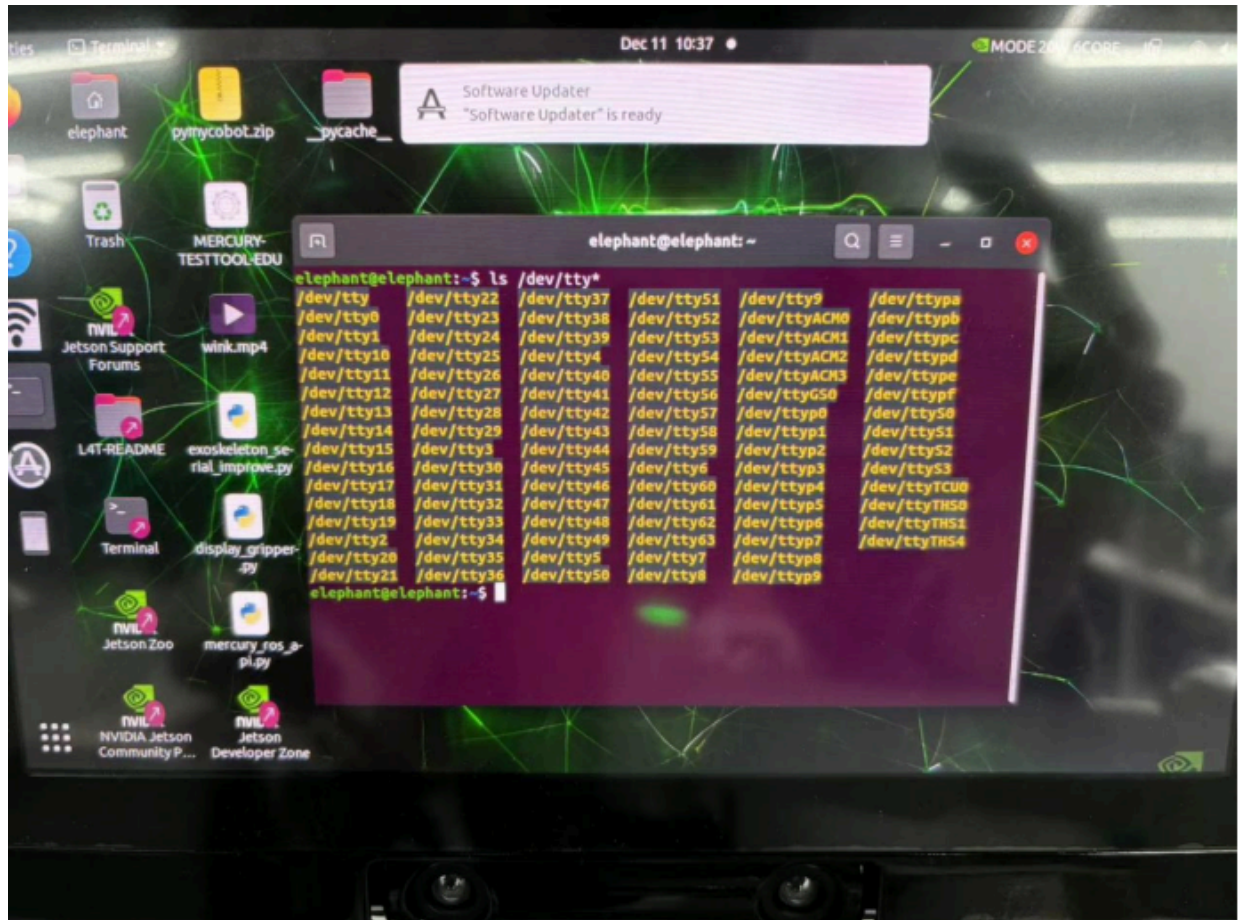


【Step 3】 Connect the exoskeleton to the robot using a USB to Type-C cable (make sure the power is on before connecting the robot).



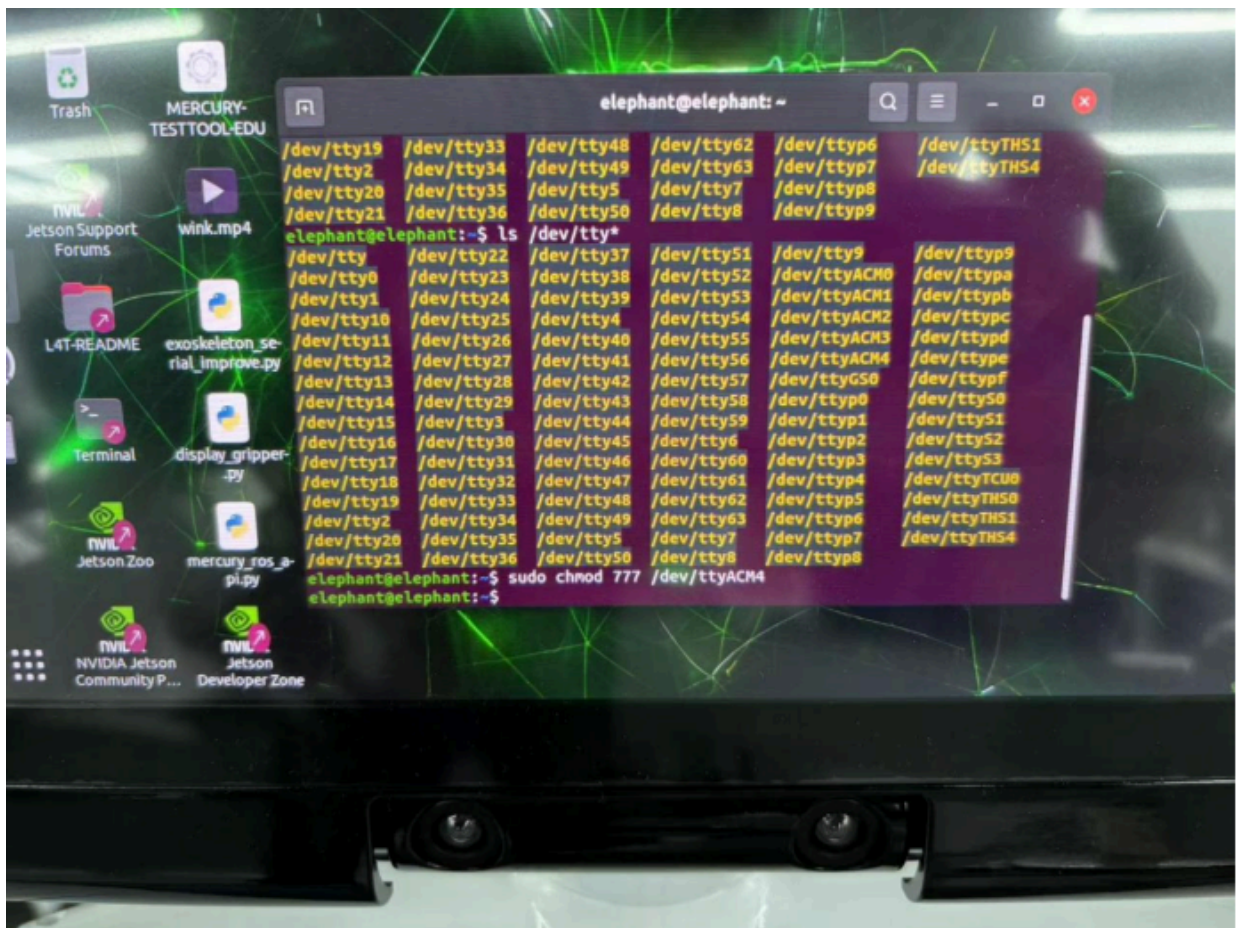
Using Mercury as an Example to Explain the Process of Selecting and Connecting the Serial Port

【Step 4】 After connecting the robotic arm, open a terminal in the system and input the following command. Then, by unplugging and plugging the serial cable connected to the exoskeleton, identify the serial port number.



The serial port number may change after unplugging and plugging in the serial cable or when the cable is replaced.

[Step 5] Grant authorization to the serial port and modify the corresponding serial port number in the source code.



【Step 6】 Modify the serial port number in the source code as required.

2.1 mercury X1(7轴)

双臂协同控制

```
import threading
from pymycobot import Mercury
from exoskeleton_api import exoskeleton

obj = exoskeleton(port="/dev/ttyACM4")
ml = Mercury("/dev/left_arm")
mr = Mercury("/dev/right_arm")

# 设置双臂为速度融合模式
ml.set_movement_type(2)
mr.set_movement_type(2)
# 设置夹爪运行模式
ml.set_gripper_mode(0)
mr.set_gripper_mode(0)

# 0 左臂, 1 右臂
def control_arm(arm):
    while True:
        if arm == 0:
            arm_data = obj.get_data(0)
            print("l: ", arm_data)
            mc = ml
        elif arm == 1:
            arm_data = obj.get_data(1)
            print("r: ", arm_data)
            mc = mr
        else:
            raise ValueError("error arm")
        # 由于外骨骼各关节转向、零点与部分机械臂构型不同, 在此设置映射关系(根据各关节实际控制转向、位置设置)
        mercury_list = [
            arm_data[0], -arm_data[1], arm_data[2], -arm_data[3], arm_data[4],
            135 + arm_data[5], arm_data[6]
        ]
        if arm_data[7] == 0:
            mc.set_gripper_state(1, 100)
        elif arm_data[8] == 0:
            mc.set_gripper_state(0, 100)
```

2. Bluetooth Connection

Power-On Situation:

Communication: Serial port is enabled by default.

UI: The top bar will display the IP and port number after powering on, followed by a green horizontal line. Below that, the angles of each joint will be displayed (7 joint angles for each robotic arm). Each robotic arm also has two buttons and one sensor. Below the joint angles, the button statuses and sensor coordinates will be shown.

Button Instructions:

-
- Press the first button to enable the WLAN service.
 - Press the second button to enable the Bluetooth service.
 - Press the third button to turn off the current service.

WLAN and Bluetooth cannot be active at the same time. When switching between these two functions, the current service must be closed first. However, serial communication can be used regardless of whether Bluetooth or WLAN is in use. In other words, as long as the exoskeleton is powered on, the serial port can be used without restrictions.

Bluetooth Service Interface Description:

- After pressing the second button, the top bar will change to: BT: Disconnect Name: BLE. This indicates that the Bluetooth service has been created but not yet connected. After connecting the exoskeleton with a mobile phone via Bluetooth, the top bar will change to: BT: Connected Name: BLE. If it is not connected, it will remain in the state of "Bluetooth service created but not connected." Once the Bluetooth service is closed, it will return to the power-on state as described in the power-on situation.

WLAN Service Interface Description:

- After pressing the first button, the top bar will show the set IP address. Regardless of whether the computer successfully connects or not, the top bar will remain unchanged. When the service is turned off, the interface will return to the power-on state as described in the power-on situation.

Serial, WLAN, and Bluetooth Service Protocol Description:

- You can use the command in the protocol document to send data. If you are using Bluetooth communication, set the MTU (Maximum Transmission Unit) of the Bluetooth Debug Assistant on the phone to 50 or higher.

This is the method for wearing the exoskeleton. If you have any questions, please contact customer service in time.

SDK Development Guide

1. Operating Environment

The MyController S570 is a high-performance robotic controller. Although it does not have a built-in system, it supports multiple operating systems, including Windows and Linux. Therefore, it is essential to integrate the **exoskeleton** with the **operating system** during use. Please prepare the operating system before using the device.

2. Development Environment

In order to meet the diverse application needs of the robot in various scenarios, we have adapted the robot to multiple programming languages. So far, we have adapted the following mainstream programming languages, and we believe you can develop using any of the following languages. Please follow the instructions strictly. Any omitted steps may result in failure to run the corresponding language. We wish you a successful experience with the robot.

If you wish to use the following programming languages, please ensure that your robot is configured with USB/Wi-Fi mode in the responder section, and confirm that the connection is correct.

- [6.1 Python](#)

Our robot supports Python, and the development of the Python API library is becoming more refined. The robot's joint angles, coordinates, gripper, and other aspects can be controlled through Python.

- [6.2 ROS1](#)

ROS (Robot Operating System) is an open-source robot operating system that provides endless possibilities for robot development and control. Our robot can be controlled through ROS's rich control features and modular approach. Whether it's joint control, path planning, or perception feedback, ROS provides corresponding tools and libraries to make the control process more flexible and efficient.

[← Previous Chapter](#) | [Next Chapter →](#)

Environment Setup

pymycobot is a Python package for serial communication with myArm, supporting Python2, Python3.5, and later versions.

Before using pymycobot to control the robotic arm, you need to set up the Python environment. The following section provides detailed instructions on how to download and install Python.

1. Python Download and Installation

Currently, Python has two versions: 2.x and 3.x. These two versions are incompatible with each other. This section uses version 3.x as an example, as it is becoming increasingly popular.

1.1 Install Python

Note: Before installation, please confirm whether your computer is 64-bit or 32-bit. Right-click `My Computer`, select `Properties`. As shown in the image below, it is a 64-bit operating system, so choose the 64-bit Python installation package.



- Official Python download site: <http://www.python.org/download/>
- Click the `Downloads` option to start downloading Python, then click `Add Python 3.10 to PATH`, and click `Install Now` to begin the installation of Python.

The screenshot shows the Python.org website with the 'Downloads' tab selected. A dropdown menu is open, showing options like 'All releases', 'Source code', 'Windows', 'macOS', 'Other Platforms', 'License', and 'Alternative Implementations'. The 'Download for Windows' section is highlighted, showing 'Python 3.10.7' and a note that Python 3.9+ cannot be used on Windows 7 or earlier. Below this, there is a table of Active Python Releases.

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619

The screenshot shows the 'Python 3.10.7 (64-bit) Setup' window. The 'Install Now' option is selected, which includes IDLE, pip, and documentation, and creates shortcuts and file associations. The 'Customize installation' option is also visible. The 'Add Python 3.10 to PATH' checkbox is checked. The window title is 'Python 3.10.7 (64-bit) Setup'.

Install Python 3.10.7 (64-bit)

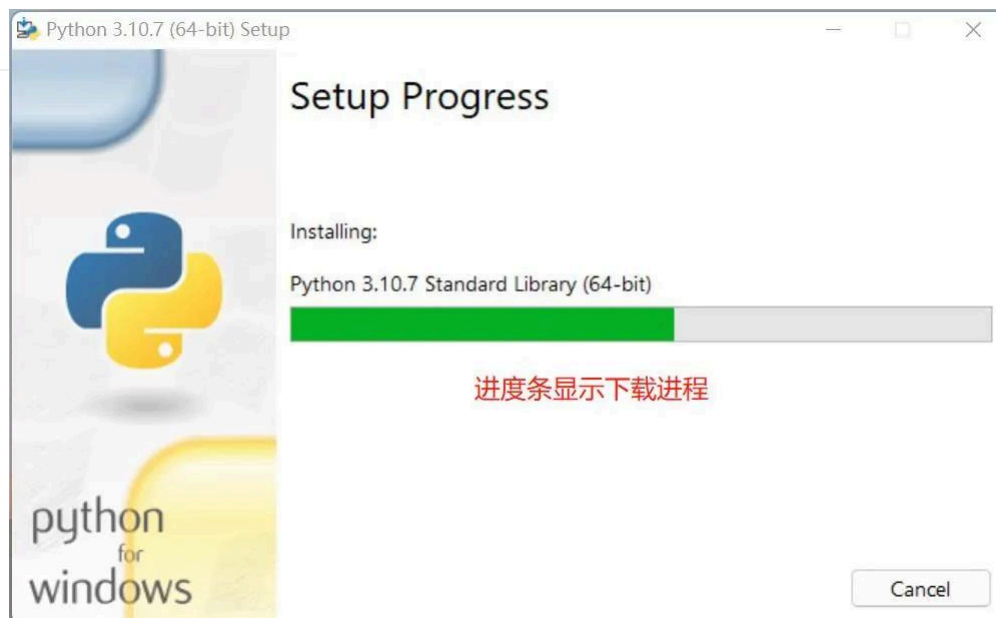
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Surface\AppData\Local\Programs\Python\Python310
Includes IDLE, pip and documentation
Creates shortcuts and file associations

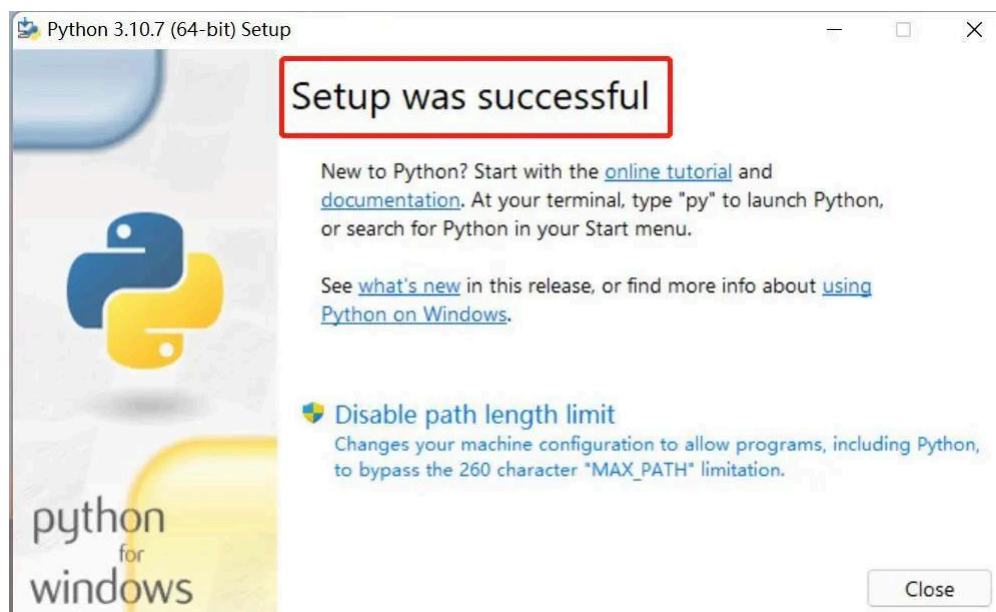
→ **Customize installation**
Choose location and features

☒ Install launcher for all users (recommended)
☒ **Add Python 3.10 to PATH**

Cancel



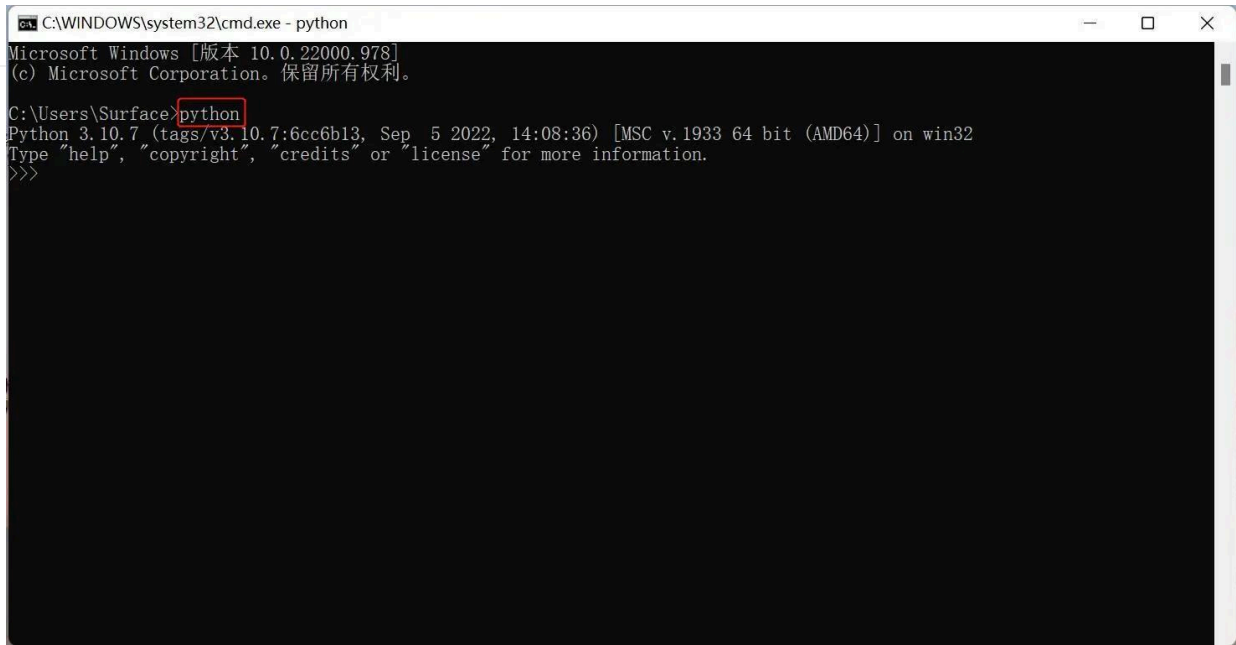
- Once the “Setup was successful” message appears, the installation is complete.



1.2 Run Python

After installation is successful, open the Command Prompt window (`Win+R` , type `cmd` and press Enter), then type `python` . Two scenarios may occur.

Scenario 1:



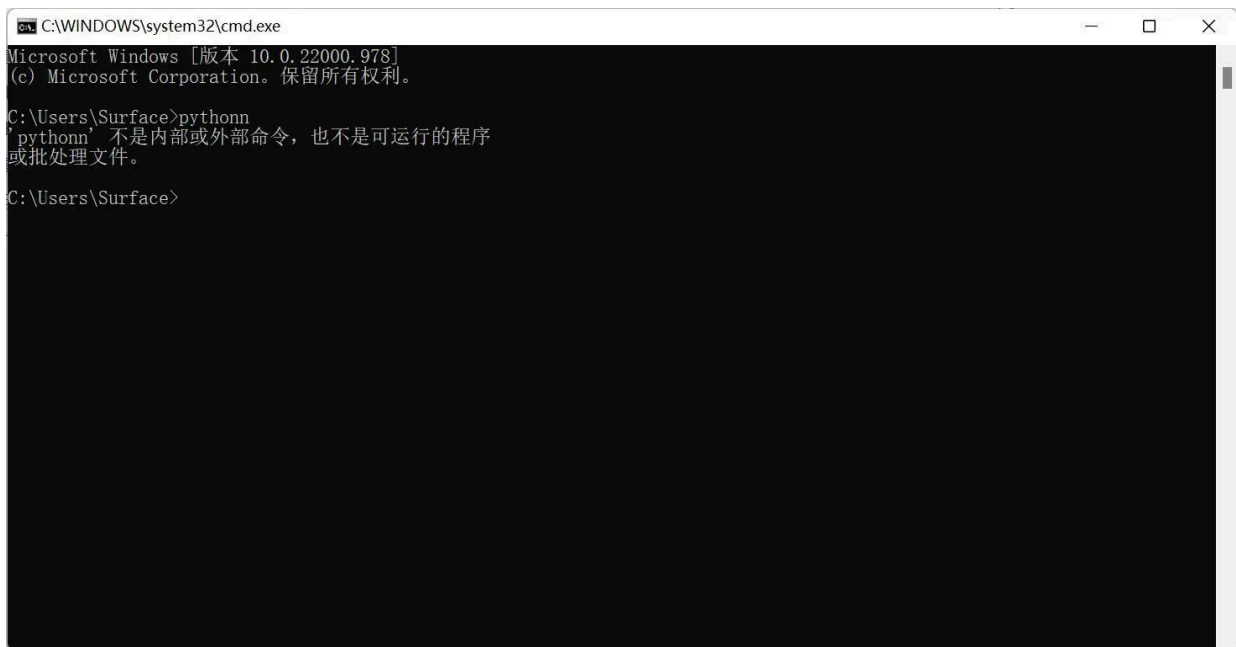
```
C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [版本 10.0.22000.978]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Surface>python
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

The message displayed on the screen indicates that Python has been successfully installed. The prompt `>>>` indicates the Python interactive environment. You can input Python code, and the execution result will appear immediately.

Scenario 2:

If you input an incorrect command (e.g., `pythonn`), you will see an error message:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.22000.978]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\Surface>pythonn
'pythonn' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\Users\Surface>
```

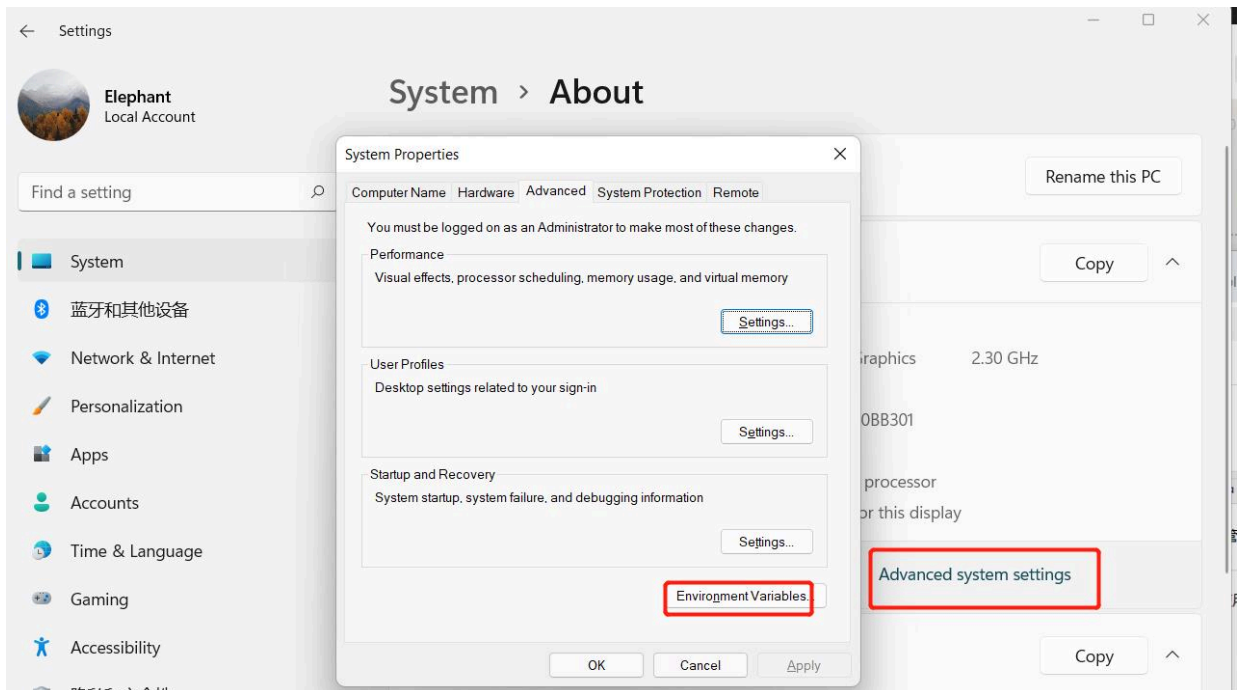
Note: The error message generally means that the environment variable is not configured properly. You can refer to section 1.3 to configure the environment variable.

1.3 Configure Environment Variables

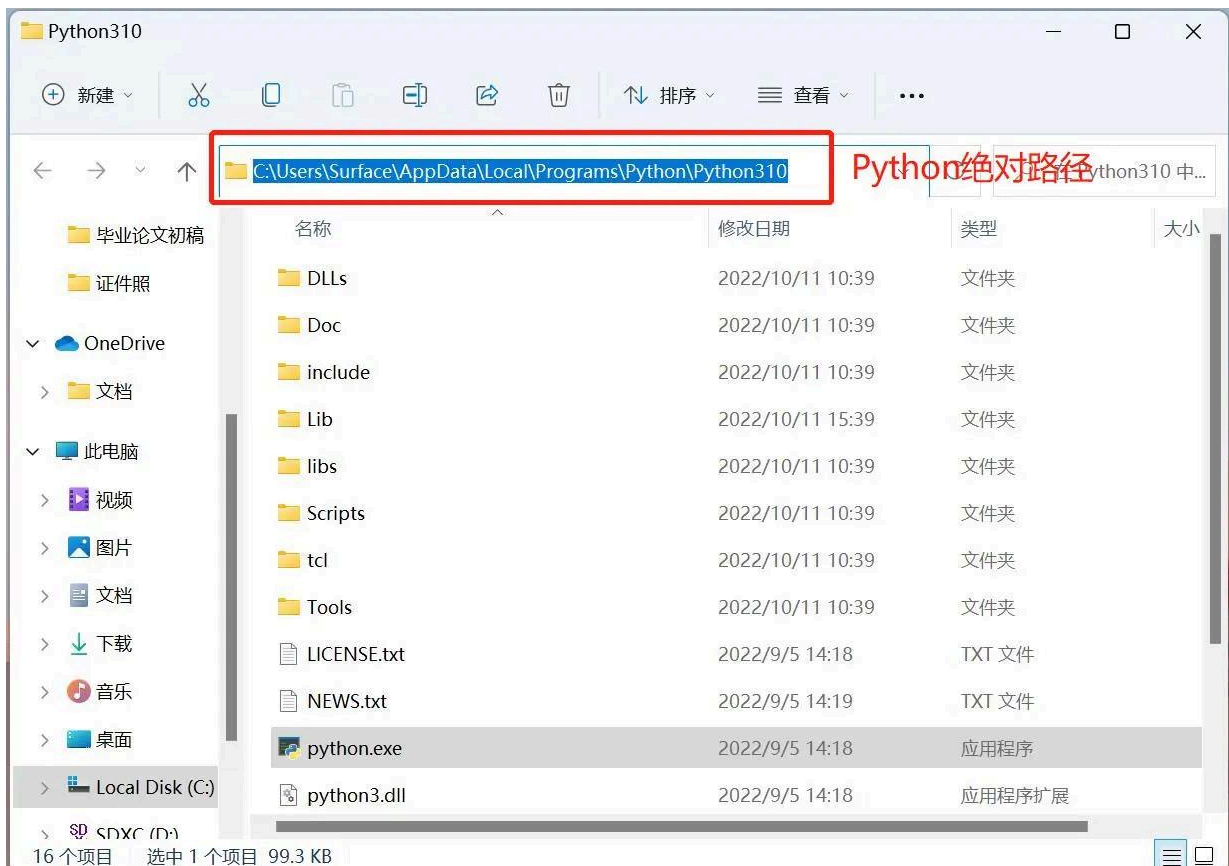
Since Windows uses the `Path` environment variable to locate `python.exe`, if it is not found, an error will occur. If you missed selecting `Add Python 3.10 to PATH` during installation, you need to manually add the path of `python.exe` to the `Path` variable, or reinstall Python and make sure to select `Add Python 3.10 to PATH`.

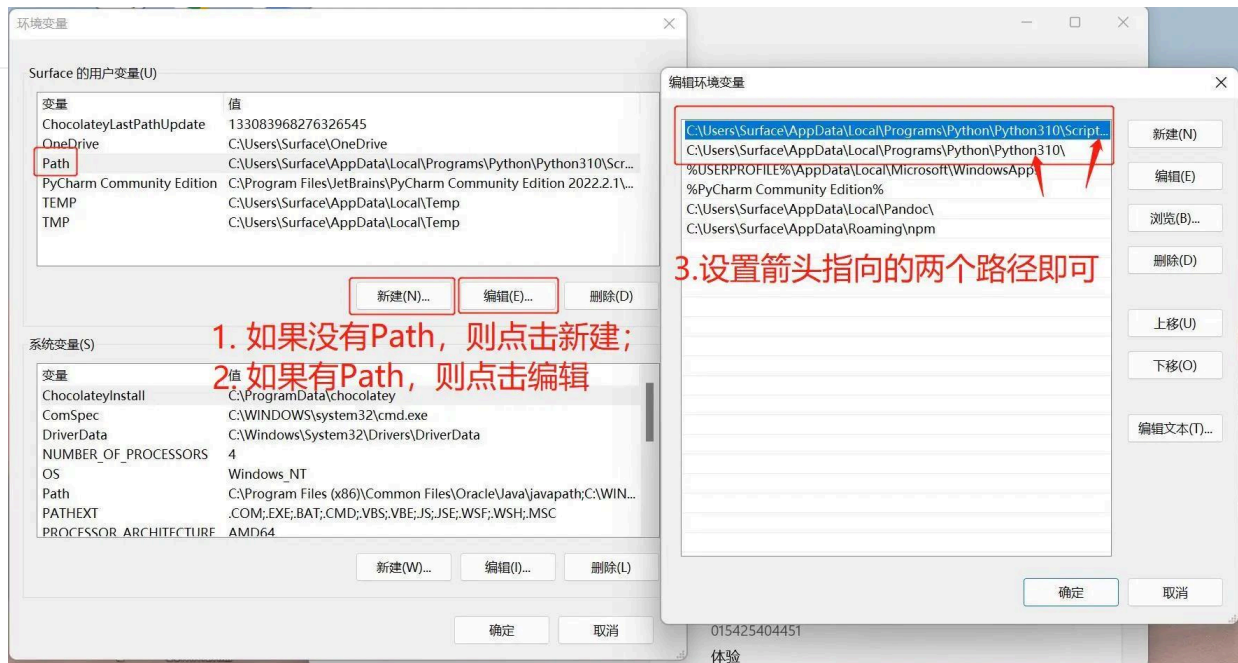
Here are the steps to manually add the path of `python.exe` :

- Right-click `My Computer` → select `Properties` → select `Advanced System Settings` → click the `Environment Variables` button at the bottom right:



- Environment variables include user variables and system variables. For user variables, you can use the downloaded program via the `cmd` command. Write the absolute path of the target program into the user variables.





- After completing the above steps, open the Command Prompt window (Win+R , type `cmd` , press Enter), type `python` , and if the following message appears, it means the setup was successful:

```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:\Users\12599>python
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

2 PyCharm Installation and Usage

PyCharm is a powerful Python editor with cross-platform support. This section introduces the installation steps of PyCharm on Windows.

Download Link: [PyCharm](#)

2.1 Download and Installation

1. Download Link: [PyCharm Download Page](#)

- PyCharm offers two versions: Professional and Community. It is recommended to install the free Community version.

2. Installation Process:

- After downloading, run the installer.
- Click `Next` to continue.
- Select the options according to your preferences, then click `Next`.
- Click `Install` to begin the installation.
- After the installation is complete, click `Finish` to end the process.

2.2 Create a Project

1. Launching PyCharm:

- Double-click the PyCharm icon on the desktop to open PyCharm.
- Click `New Project` to create a new project.

2. Setting up Python Interpreter:

- When creating the project, you need to configure the Python interpreter.
- Click `Add Interpreter`, select `New`, locate the `python.exe` in the Python installation directory, and check the `Inherit global site-package` option.

3. Setting the Project Location:

- Choose a location to store the PyCharm project in the `Location` field.

4. Create a Python File:

- In PyCharm, right-click the document icon in the project, select `New`, and then choose `Python File`. Enter the file name and create it.

5. Write Code:

- After creating the file, the code editor will open, and you can start writing your Python code.

3 Preparation Work

- **Firmware Programming:** The firmware is the system's control program for driving the robot. [MyStudio](#).
 - The `Basic` module of the exoskeleton requires the `minirobot` firmware to be flashed. Additionally, the `Atom` module located at the end of the exoskeleton should be flashed with the latest version of `atomMain`; however, as this is pre-flashed at the factory, manual flashing is not required.
- **Installing pymycobot:** Open a console terminal (shortcut `Win+R`, type `cmd` to open the terminal), and enter the following command:

```
pip install pymycobot --upgrade --user
```



- **Source code installation.** Open a console terminal (shortcut key 'Win+R', enter 'cmd' to enter the terminal) and enter the following command to install:

```
git clone https://github.com/elephantrobotics/pymycobot.git <your-path>
# Where <your-path> fill in your installation address, do not fill in the default in the current path

cd <your-path>/pymycobot
# Go to the pymycobot folder in the download package

# Run one of the following commands based on your version of python
# Install
python2 setup.py install
# or
python3 setup.py install
```

- Update pymycobot

```
pip install pymycobot --upgrade
```

4 Import pymycobot

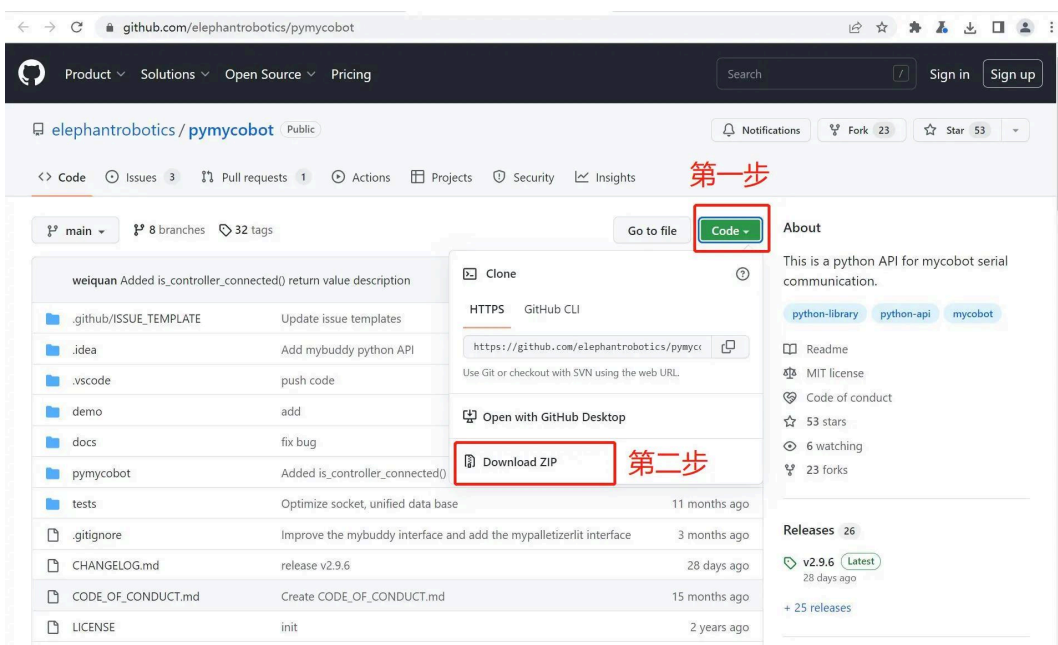
After the above preparatory work is completed, the robot arm is controlled through Python code. The myArm Master 750 version is used as an example.

- Enter the following code to import our library:

```
from pymycobot import MyArmM
```

Note:

1. If the red wavy line does not appear at the bottom of the code, the pymycobot installation was successful.
2. If there is a red wavy lines, please visit <https://github.com/elephantrobotics/pymycobot> manually download pymycobot and put it in python library.



API instructions

Application Programming Interface (API), also known as application Programming interface functions, is a predefined function. When using the following function interface, please import our API library by entering the following code at the beginning, otherwise it will not run successfully

Install the python library

`https://github.com/elephantrobotics/pymycobot.git`

Usage Example

1.Serial communication

Example Usage

1. Serial Communication

```
# Example
from pymycobot import Exoskeleton

# Connect to serial port
obj = Exoskeleton(port="COM5")

# Get data from both arms
all_data = obj.get_all_data()
print(all_data)

# Get left arm data
left_data = obj.get_arm_data(1)
print(left_data)

# Get right arm data
right_data = obj.get_arm_data(2)
print(right_data)

# Get data of a single joint on one arm
joint_data = obj.get_joint_data(1, 1)
print(joint_data)

# Set current position of right arm J7 as zero
obj.set_zero(2, 7)

# Set color of left arm atom screen
obj.set_color(1, 0, 255, 0)

# Get firmware version of right arm atom
obj.get_atom_version(2)

# Get basic firmware version
obj.get_basic_version()
```

2. Wireless Communication

```

# Example
from pymycobot import ExoskeletonSocket

# Connect to socket
obj = ExoskeletonSocket()

# Get data from both arms
all_data = obj.get_all_data()
print(all_data)

# Get left arm data
left_data = obj.get_arm_data(1)
print(left_data)

# Get right arm data
right_data = obj.get_arm_data(2)
print(right_data)

# Get data of a single joint on one arm
joint_data = obj.get_joint_data(1, 1)
print(joint_data)

# Set current position of right arm J7 as zero
obj.set_zero(2, 7)

# Set color of left arm atom screen
obj.set_color(1, 0, 255, 0)

# Get firmware version of right arm atom
obj.get_atom_version(2)

# Get basic firmware version
obj.get_basic_version()

```

get_all_data()

- **Function** : Obtain data from both arms
- **Parameters**: None
- **Return value** : Angle parameter and hand controller, coordinate floating-point list:

[[Left arm J1, J2, J3, J4, J5, J6, J7, atom button, joystick button, Button 1, Button 2, joystick x, joystick y, x, y, z, rx, ry, rz]

[Right arm J1, J2, J3, J4, J5, J6, J7, atom button, joystick button, Button 1, Button 2, joystick x, joystick y, x, y, z, rx, ry, rz]]

get_arm_data(arm)

- **Function** : Obtain single-arm data

- **Parameter**
- 'arm' : 1 left arm, 2 right arm
- **Return Value** Angle parameters and hand controller floating-point list: [J1, J2, J3, J4, J5, J6, J7, atom button, joystick button, Button 1, Button 2, joystick x, joystick y, x, y, z, rx, ry, rz]

get_joint_data(arm, arm_id)

- **Function** : Obtain data of a single arm and a single joint
- **Parameter**
- 'arm' : 1 left arm, 2 right arm
- 'arm_id' : Joint id, range int 1-7
- **Return value**: angle (int)

get_joint_data(arm, arm_id)

- **Function** : Obtain data of a single arm and a single joint
- **Parameter**
- 'arm' : 1 left arm, 2 right arm
- 'arm_id' : Joint id, range int 1-7
- **Return value**: angle (int)

set_zero(arm, arm_id)

- **Function** : Set the current position to the zero position of the joint
- **Parameter**
- 'arm' : 1 left arm, 2 right arm
- 'arm_id' : Joint id, range int 1-7
- **Return value** 1

set_color(arm, red, green, blue)

- **Function** : Set the color of the atom screen
- **Parameter**
- 'arm' : 1 left arm, 2 right arm
- 'red' : Range int 0-255
- 'green' : Range int 0-255
- 'blue' : Range int 0-255
- **Return value** 1

get_basic_version()

- **Function** : Obtain the version number of the main control basic firmware
- **Return value**: Version number

get_atom_version(arm)

- **Function** : Obtain the terminal atom firmware version number
- **Parameter**
- 'arm' : 1 left arm, 2 right arm
- **Return value**: Version number

Note : MyController S570 is extremely compatible with all Elephant Robot products, as well as all general purpose 6-7 axis robot arms. Therefore, for different robot arms, there are different API functions, please choose according to the actual model of the robot arm.

1.mycobot280series

https://docs.elephantrobotics.com/docs/mycobot_280_m5_cn/3-FunctionsAndApplications/6.developmentGuide/python/2_API.html

2.mycobot320series

https://docs.elephantrobotics.com/docs/mycobot_320_m5_cn/10-ApplicationBasePython/10.2_320_M5-ApplicationPython/2_API.html

3.MyAGVseries

https://docs.elephantrobotics.com/docs/myagv_jn23_cn/6-SDKDevelopment/6.1-ApplicationBasePython/6.1.2-API.html

4.Mercuryseries

https://docs.elephantrobotics.com/docs/Mercury_X1_cn/6-SDKDevelopment/6.1-Python/6.1.2-ApplicationBasePython.html

5.MyArm M750

https://docs.elephantrobotics.com/docs/myarm-master_750-cn/4-FunctionsAndApplications/6-SDKDevelopment/5.1-BasedOnPythonDevelopmentAndUse/2_API.html

6.MechArmseries

https://docs.elephantrobotics.com/docs/mecharm-m5-cn/7-ApplicationBasePython/7.2_API_270.html

7.MyBuddy

https://docs.elephantrobotics.com/docs/mybuddy-cn/7-ApplicationBasePython/7.2_API_mybuddy.html

[← Previous](#) | [Next page →](#)

Remote control robot arm case

The examples are based on the `exoskeleton_api.py` file

[Get address](#)

1 Serial communication

```
# example
from exoskeleton_api import Exoskeleton

# Connect the serial port
obj = Exoskeleton(port="COM5")

# Get dual-arm data
all_data = obj.get_all_data()
print(all_data)

# Get left arm data
left_data = obj.get_arm_data(1)
print(left_data)

# Get right arm data
right_data = obj.get_arm_data(2)
print(right_data)

# Obtain single arm and single joint data
joint_data = obj.get_joint_data(1, 1)
print(joint_data)

# Set the current position to zero on right arm J7
obj.set_zero(2, 7)

# Set the left arm atom screen color
obj.set_color(1, 0, 255, 0)
```

2 socket communication

```
# example
from exoskeleton_api import ExoskeletonSocket

# Connection server
obj = ExoskeletonSocket()

# Get dual-arm data
all_data = obj.get_all_data()
print(all_data)

# Get left arm data
left_data = obj.get_arm_data(1)
print(left_data)

# Get right arm data
right_data = obj.get_arm_data(2)
print(right_data)

# Obtain single arm and single joint data
joint_data = obj.get_joint_data(1, 1)
print(joint_data)

# Set the current position to zero on right arm J7
obj.set_zero(2, 7)

# Set the left arm atom screen color
obj.set_color(1, 0, 255, 0)
```

3 mercury X1(7 axes)

pymycobot version 3.5.0b66 (and above)

Dual-arm cooperative control

```

import threading
from pymycobot import Mercury
from exoskeleton_api import exoskeleton

obj = exoskeleton(port="/dev/ttyACM4")
ml = Mercury("/dev/left_arm")
mr = Mercury("/dev/right_arm")

# Set both arms to speed fusion mode
ml.set_movement_type(2)
mr.set_movement_type(2)
# Set the jaw operation mode
ml.set_gripper_mode(0)
mr.set_gripper_mode(0)

#0 left arm, 1 right arm
def control_arm(arm):
    while True:
        if arm == 0:
            arm_data = obj.get_data(0)
            print("l: ", arm_data)
            mc = ml
        elif arm == 1:
            arm_data = obj.get_data(1)
            print("r: ", arm_data)
            mc = mr
        else:
            raise ValueError("error arm")

    # Due to the different configurations of each joint steering, zero point and part of the robot arm of the exoskeleton
    mercury_list = [
        arm_data[0], -arm_data[1], arm_data[2], -arm_data[3], arm_data[4],
        135 + arm_data[5], arm_data[6]
    ]

    # Press the button to control the opening and closing of the claw
    if arm_data[7] == 0:
        mc.set_gripper_state(1, 100)
    elif arm_data[8] == 0:
        mc.set_gripper_state(0, 100)
    mc.send_angles(mercury_list, 6)

# Left arm
threading.Thread(target=control_arm, args=(0, )).start()
# Right arm
threading.Thread(target=control_arm, args=(1, )).start()

```


MyController S570

Installing Different Versions of Ubuntu System on Linux

1 Virtual Machine Installation

Go to the [official website](#) to download the VirtualBox virtual machine or visit the [official website](#) to download the VM Ware virtual machine.

Of course, if you already have a virtual machine, you can skip this step.

We choose to download VirtualBox because it is free.

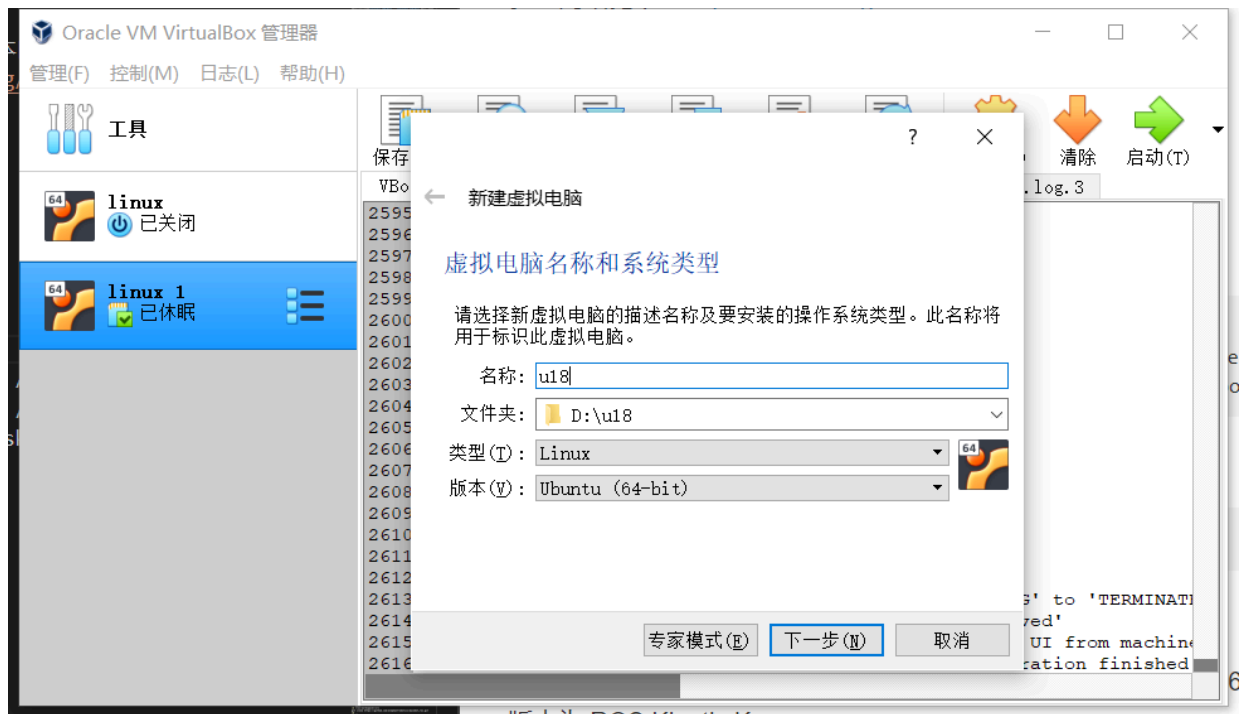


2 Create a Virtual Machine

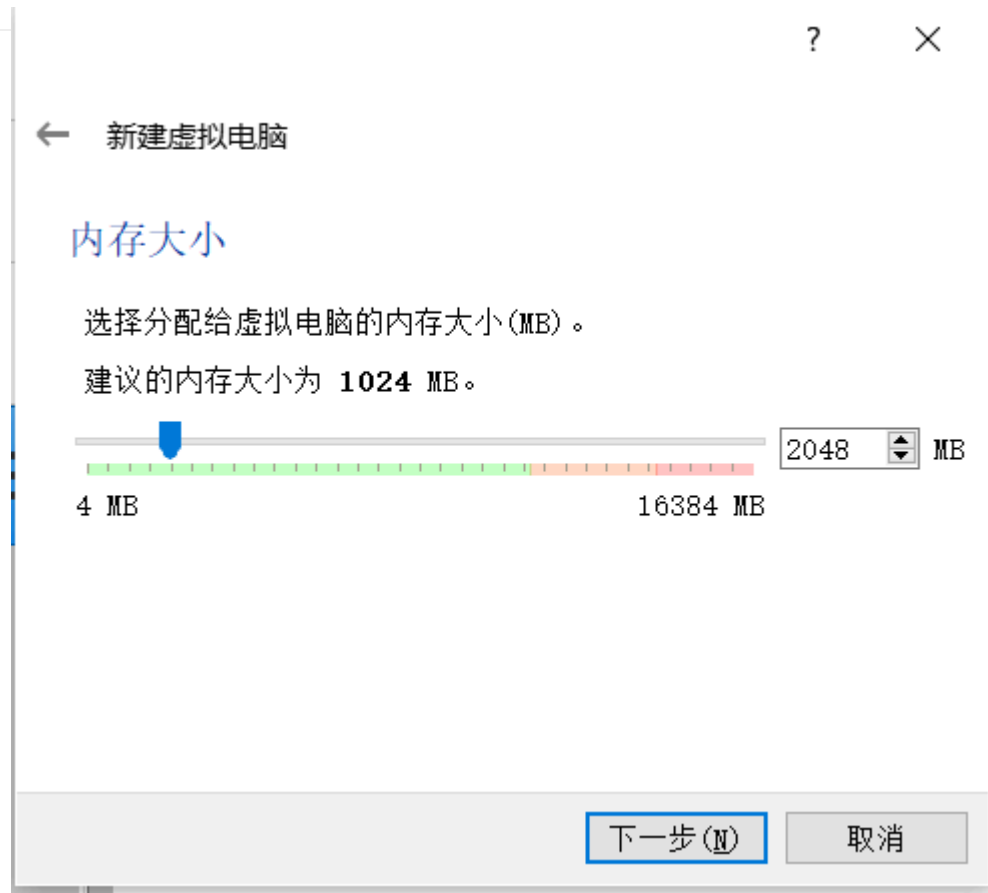
2.1 Create a Virtual Machine

Select "New" in the control panel

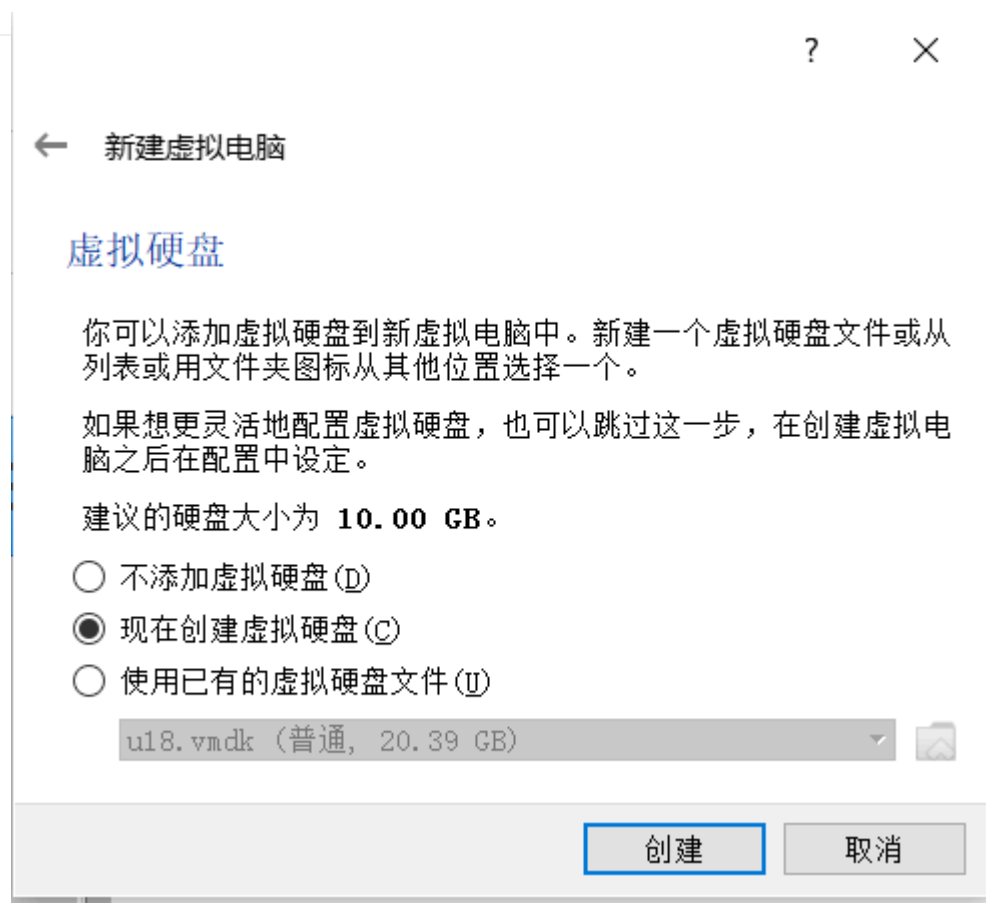
Enter the name of the virtual machine and the location to store it, select the virtual machine type as Linux, and choose the 64-bit Ubuntu version, then proceed to the next step.



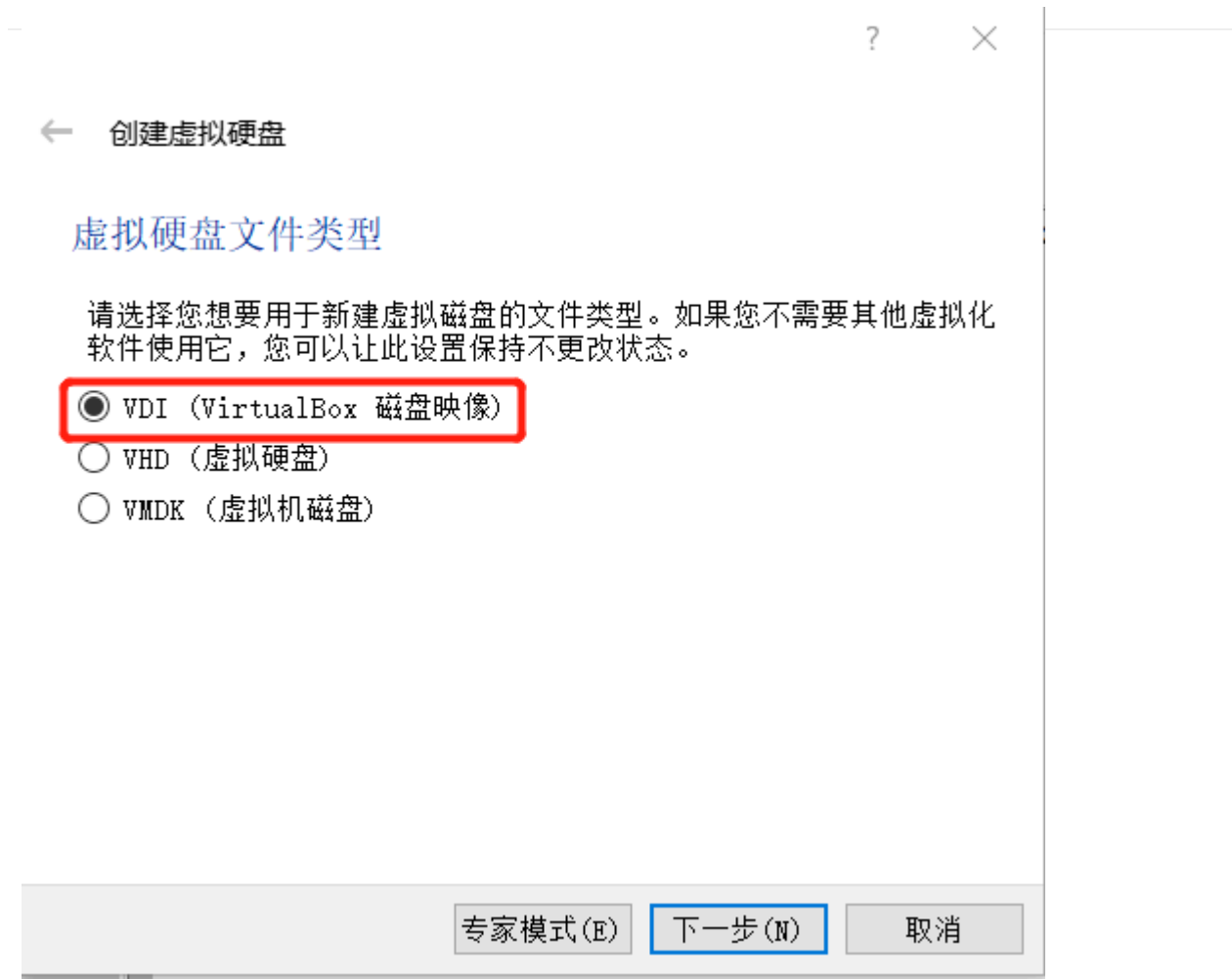
Configure the memory size according to your needs and proceed to the next step.



Select **Create a Virtual Hard Disk Now**, and proceed to create.



Choose the **VDI** type for the virtual hard disk, then proceed to the next step.



Allocate the virtual hard disk size. Since you need to install the Ubuntu system and will be performing operations within this system, it is recommended to set the size to no less than 20GB.

← 创建虚拟硬盘

存储在物理硬盘上

请选择新建虚拟硬盘文件是应该为其使用而分配(动态分配)，还是应该创建完全分配(固定分配)。

动态分配的虚拟磁盘只是逐渐占用物理硬盘的空间（直至达到 **分配的大小**），不过当其内部空间不用时不会自动缩减占用的物理硬盘空间。

固定大小的虚拟磁盘文件可能在某些系统中要花很长时间来创建，但它往往使用起来较快。

☒ 动态分配 (D)

☐ 固定大小 (F)

下一步 (N)

取消

选择虚拟硬盘的大小。此大小为虚拟硬盘文件在实际硬盘中能用的极限大小。



2.2 Import Ubuntu System

2.2.1 Download Ubuntu System.

Choose the Ubuntu version according to your needs for installation.

Note: ROS2 requires the 20.04 version.

[16.04 Version](#)

[18.04 Version](#)

[20.04 Version](#)

The installation method and process for the three versions are the same. Here, we will use the 18.04 version as an example.

If you need help burning these images to disk, see the [Image Burning Guide](#).

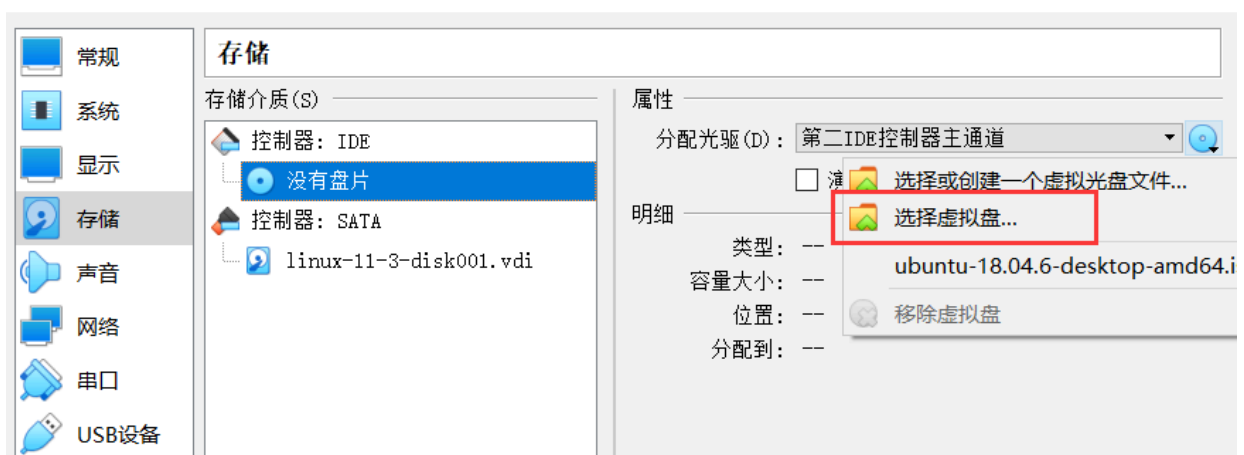
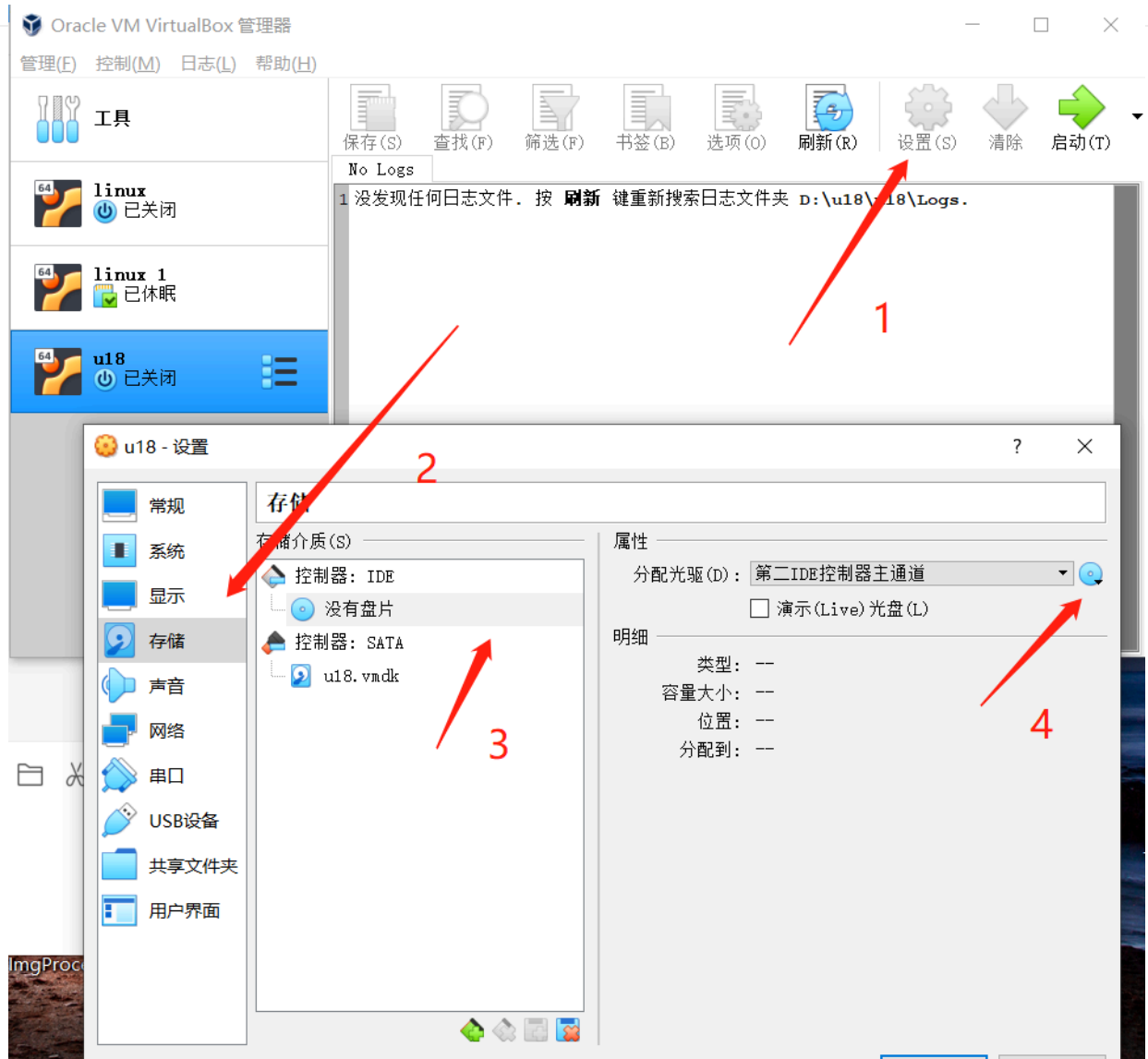
Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (standard download)
ubuntu-18.04.6-desktop-amd64.iso.torrent	2021-09-16 21:46	188K	Desktop image for 64-bit PC (AMD64) computers (BitTorrent download)
ubuntu-18.04.6-desktop-amd64.iso.zsync	2021-09-16 21:46	4.7M	Desktop image for 64-bit PC (AMD64) computers (zsync metafile)
ubuntu-18.04.6-desktop-amd64.list	2021-09-15 20:42	7.8K	Desktop image for 64-bit PC (AMD64) computers (file listing)
ubuntu-18.04.6-desktop-amd64.manifest	2021-09-15 20:36	59K	Desktop image for 64-bit PC (AMD64) computers (contents of live filesystem)
ubuntu-18.04.6-live-server-amd64.iso	2021-09-15 20:42	969M	Server Install Image for 64-bit PC (AMD64) computers (standard download)

After downloading, you will have the following file:

名称	修改日期	类型	大小
今天 (2)			
ubuntu-20.04.3-desktop-amd64.iso	2022/1/12 15:08	ISO 文件	2,999,93
ubuntu-18.04.6-desktop-amd64.iso	2022/1/12 14:45	ISO 文件	2,455,20

2.2.2 Import Ubuntu into the Virtual Machine

Find the virtual machine you previously installed in VirtualBox, go to **Settings**, and assign a disk controller under **Storage**:



Then open the virtual machine to start the Ubuntu installation and click Start.

2.2.3 Ubuntu Installation

Wait for the system to boot, and when the **Welcome** screen appears, select "Chinese (Simplified)" and click the "Install Ubuntu" button;



Click the "Continue" button;



Select "Erase disk and install Ubuntu", and click the "Install Now" button;



Click the "Continue" button in the dialog box that pops up;



Set the geographical location and click the "Continue" button;



Set the username and password, and click the "Continue" button;

Wed 17:24

安装

您是谁？

您的姓名：

u18

✓

您的计算机名：

u18-VirtualBox

✓

与其他计算机联络时使用的名称。

选择一个用户名：

u18

✓

选择一个密码：

●●●

密码强度：过短

确认您的密码：

●●●|

✓

☐ 自动登录

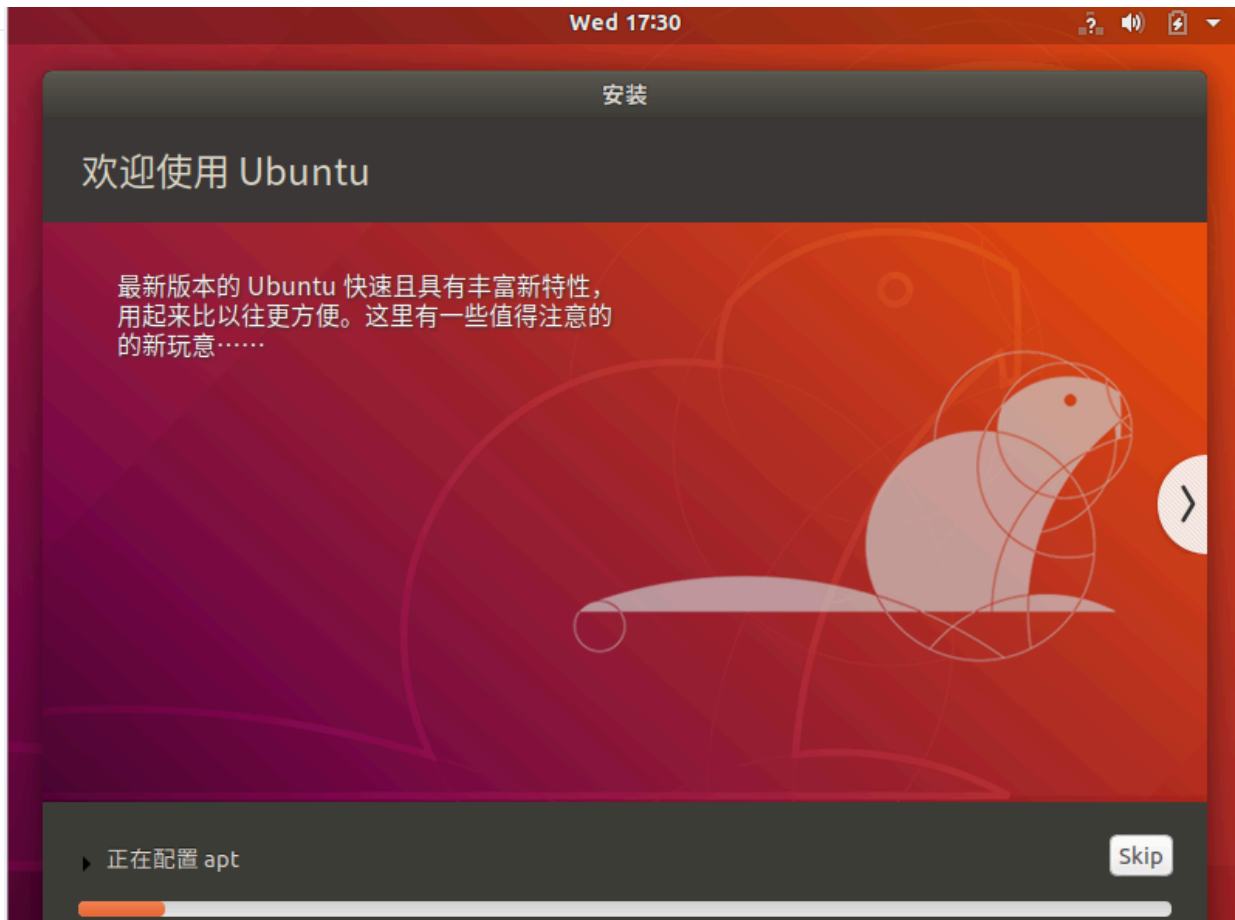
☒ 登录时需要密码

后退(B)

继续

●●●●●●●●

Wait for the system to install; please be patient.



Once installation is complete, click the "Restart Now" button in the dialog box to finish the installation.



3 ROS environment construction

3.1 ROS installation

The basic development environment needs to install the robot operating system ROS, MoveIt and git version manager. The following describes the installation methods and processes respectively.

3.1.1 version selection

ROS have one-to-one relationship with ubuntu, different versions of ubuntu correspond to different versions of the ROS, reference website below: <http://wiki.ros.org/Distributions>

Here are the ROS versions that Ubuntu supports: Ubuntu 16.04 / ROS Kinetic Ubuntu 18.04 / ROS Melodic Ubuntu 20.04 / ROS Noetic

Please install the ROS version according to the Ubuntu version you have installed

If the version is different, the download will fail. Here we choose the system as Ubuntu 18.04, corresponding to ROS version ROS Melodic

NOTE: currently, we do not provide any references to Windows installation ROS, if necessary, please refer to [<https://www.ros.org/install/>] (<https://www.ros.org/install/>)

3.1.2 Start installation

1 Add source

There is no ROS software source in Ubuntu's own software source list, so you need to configure the ROS software source to the software list repository before you can download ROS. Open a console terminal (Ctrl+Alt+T) and enter the following command:

- Official Source:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```

- If the download speed is slow, you are advised to select a nearby mirror source to replace the preceding command. For example, Tsinghua University is:

```
sudo sh -c ' /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ $(lsb_release -cs)
main" > /etc/apt/sources.list.d/ros-latest.list'
```

You will be asked to enter the user password. Enter the user password you set when installing Ubuntu.

2 Set the secret key

Configure the public network key, this step is to let the system confirm that our path is safe, so that there is no problem to download the file, otherwise it will be deleted immediately after downloading:

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

The command output is as follows:

```

u18@u18-VirtualBox:~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:
80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
Executing: /tmp/apt-key-gpghome.kD1wi001jJ/gpg.1.sh --keyserver hkp://keyserver
.ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: 密钥 F42ED6FBAB17C654: 公钥“Open Robotics <info@osrfoundation.org>”已导入
gpg: 合计被处理的数量: 1
gpg: 已导入: 1
u18@u18-VirtualBox:~$

```

3 Install

After adding a new software source, you need to **update the software source list**, open a console terminal (Ctrl+Alt+T), and enter the following command:

```
sudo apt-get update
```

To install ROS, open a console terminal (Ctrl+Alt+T) and enter the following command according to your Ubuntu version:

```
# Ubuntu 16.04
```

```
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
```

```
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
```

```
sudo apt install ros-noetic-desktop-full
```

A full ROS installation is recommended to prevent the loss of libraries and dependencies.

The installation process takes a long time and requires patience

- If the following error message is displayed on the console terminal during installation, change the software source list in `/etc/apt/sources.list`.

```
无法修复缺失的软件包。
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/flann/libflann1.9_1.9.1+dfsg-2_amd64.deb 连接失败 [IP: 91.189.91.39 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/flann/libflann-dev_1.9.1+dfsg-2_amd64.deb 连接失败 [IP: 91.189.91.39 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fttk1.3/libfttk-cairo1.3_1.3.4-6_amd64.deb 连接失败 [IP: 91.189.91.39 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fttk1.3/libfttk-gl1.3_1.3.4-6_amd64.deb 连接失败 [IP: 91.189.91.39 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fttk1.3/libfttk1.3-dev_1.3.4-6_amd64.deb 连接失败 [IP: 91.189.91.38 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fyba/libfyba-dev_4.1.1-3_amd64.deb 连接失败 [IP: 91.189.91.38 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/fonts-dejavu/ttf-dejavu-core_2.37-1_all.deb 连接失败 [IP: 91.189.91.39 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/libc/libccd/libccd-dev_2.0-1_amd64.deb 连接失败 [IP: 91.189.91.39 80]
E: 无法下载 http://cn.archive.ubuntu.com/ubuntu/pool/universe/f/freexl/libfreexl-dev_1.0.5-1_amd64.deb 连接失败 [IP: 91.189.91.38 80]
E: 中止安装。
u182@u182-VirtualBox:~$
```

- Open a console terminal (Ctrl+Alt+T) and enter the following command:

```
sudo gedit /etc/apt/sources.list
```

- Replace all official software sources in `sources.list` with the following Alibaba Cloud software sources:

Ubuntu 16.04 version:

```
deb http://mirrors.aliyun.com/ubuntu/ xenial main
```

```
deb-src http://mirrors.aliyun.com/ubuntu/ xenial main
```

```
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates main
```

```
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates main
```

```
deb http://mirrors.aliyun.com/ubuntu/ xenial universe deb-src http://mirrors.aliyun.com/ubuntu/ xenial universe
```

```
deb http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
```

```
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-updates universe
```

```
deb http://mirrors.aliyun.com/ubuntu/ xenial-security main
```

```
deb-src http://mirrors.aliyun.com/ubuntu/ xenial-security main
```

deb <http://mirrors.aliyun.com/ubuntu/> xenial-security universe

deb-src <http://mirrors.aliyun.com/ubuntu/> xenial-security universe

Ubuntu 18.04 version:

deb <http://mirrors.aliyun.com/ubuntu/> bionic main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> bionic main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> bionic-security main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> bionic-security main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> bionic-updates main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> bionic-updates main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> bionic-proposed main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> bionic-proposed main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> bionic-backports main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> bionic-backports main restricted universe multiverse

Ubuntu 20.04 version:

deb <http://mirrors.aliyun.com/ubuntu/> focal main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> focal main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> focal-security main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> focal-security main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> focal-updates main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> focal-updates main restricted universe multiverse

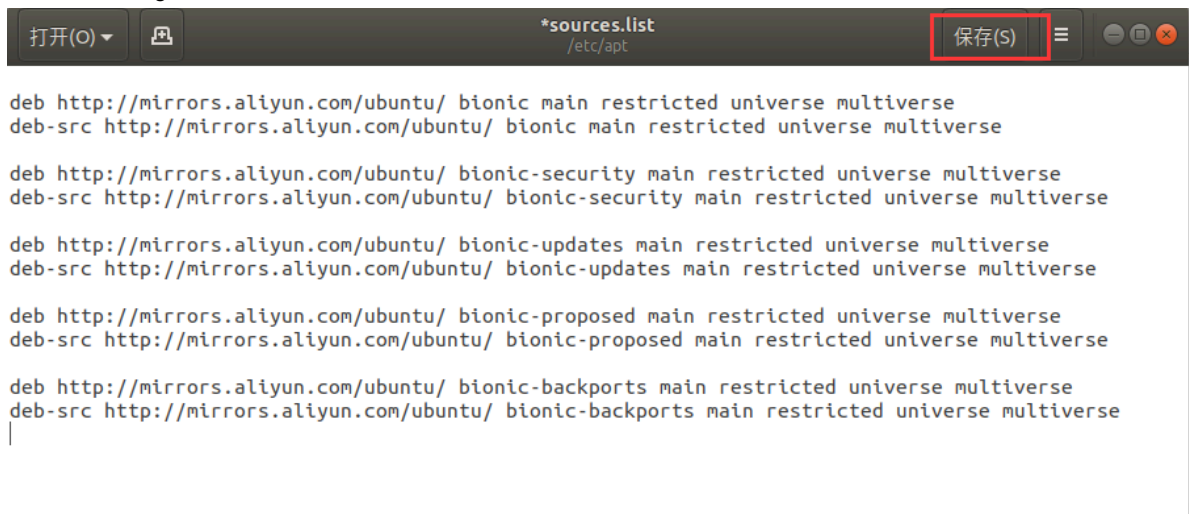
deb <http://mirrors.aliyun.com/ubuntu/> focal-proposed main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> focal-proposed main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> focal-backports main restricted universe multiverse

deb-src <http://mirrors.aliyun.com/ubuntu/> focal-backports main restricted universe multiverse

- After the configuration, the contents of the sources.list file are as follows. Click Save and exit.



- To update the software source list, enter in the console terminal:

```
sudo apt-get update
```

- Enter the instructions to install ROS in the console terminal:

```
# Ubuntu 16.04
```

```
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
```

```
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
```

```
sudo apt install ros-noetic-desktop-full
```

The installation process takes a long time and requires patience

4 Configure the ROS environment to the system

rosdep allows you to easily install the source code that you want to compile, or that is required by some of the ROS core components of the system, by executing the following commands from the terminal to open a console terminal (Ctrl+Alt+T).

If rosdep is not installed on your system, use the command `sudo apt install python-rosdep` to install it.

If your Ubuntu system is version 20.04, run the `sudo apt install python3-rosdep` command to install it, and then execute the rosdep initialization command.

```
u18@u18-VirtualBox:~$ sudo rosdep init
sudo: rosdep: 找不到命令
u18@u18-VirtualBox:~$ sudo apt install python-rosdep
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
下列【新】软件包将被安装：
  python-rosdep
升级了 0 个软件包，新安装了 1 个软件包，要卸载 0 个软件包，有 1 个软件包未被升级。
需要下载 3,236 B 的归档。
解压缩后会消耗 21.5 kB 的额外空间。
获取:1 http://packages.ros.org/ros/ubuntu bionic/main amd64 python-rosdep all 0.21.0-1 [3,236 B]
已下载 3,236 B，耗时 1秒 (4,120 B/s)
正在选中未选择的软件包 python-rosdep。
(正在读取数据库 ... 系统当前共安装有 247871 个文件和目录。)
正准备解包 .../python-rosdep_0.21.0-1_all.deb ...
正在解包 python-rosdep (0.21.0-1) ...
正在设置 python-rosdep (0.21.0-1) ...
u18@u18-VirtualBox:~$
```

Initialize rosdep:

```
sudo rosdep init
```

If an error message like the one below appears:

```
u182@u182-VirtualBox:~$ sudo rosdep init
ERROR: cannot download default sources list from:
https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/sources.list.d/20-
default.list
Website may be down.
```

Workaround: Modify the hosts file and enter the following command in the console terminal:

```
sudo gedit /etc/hosts
```

At the end of the file content, add the IP addresses of the following two urls to achieve access:

```
199.232.28.133 raw.githubusercontent.com
```

```
151.101.228.133 raw.githubusercontent.com
```



After the modification is complete, execute in the console terminal:

```
sudo rosdep init
```

```
rosdep update
```

After the initialization, in order to avoid the need to reactivate the ROS function path every time the terminal window is closed, we can configure the path to the environment variable **, so that the ROS function path can automatically take effect every time a new terminal is opened. Open a console terminal (Ctrl+Alt+T) :

3.1.3 Setting up the ros environment

Bash

Run the following command:

Ubuntu 16.04

```
# Adds the ros environment to the environment variable of the current console echo "source
/opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
```

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
```

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Install ROS extra dependencies

Enter the following command **Install ROS Extra dependencies** in the terminal to open a console terminal (Ctrl+Alt+T) :

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

If your Unbutu system is version 20.04, run the following command to install it:

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

```
# Ubuntu 16.04
```

```
sudo apt install ros-kinetic-joint-state-publisher-gui
```

```
# Ubuntu 18.04
```

```
sudo apt install ros-melodic-joint-state-publisher-gui
```

```
# Ubuntu 20.04
```

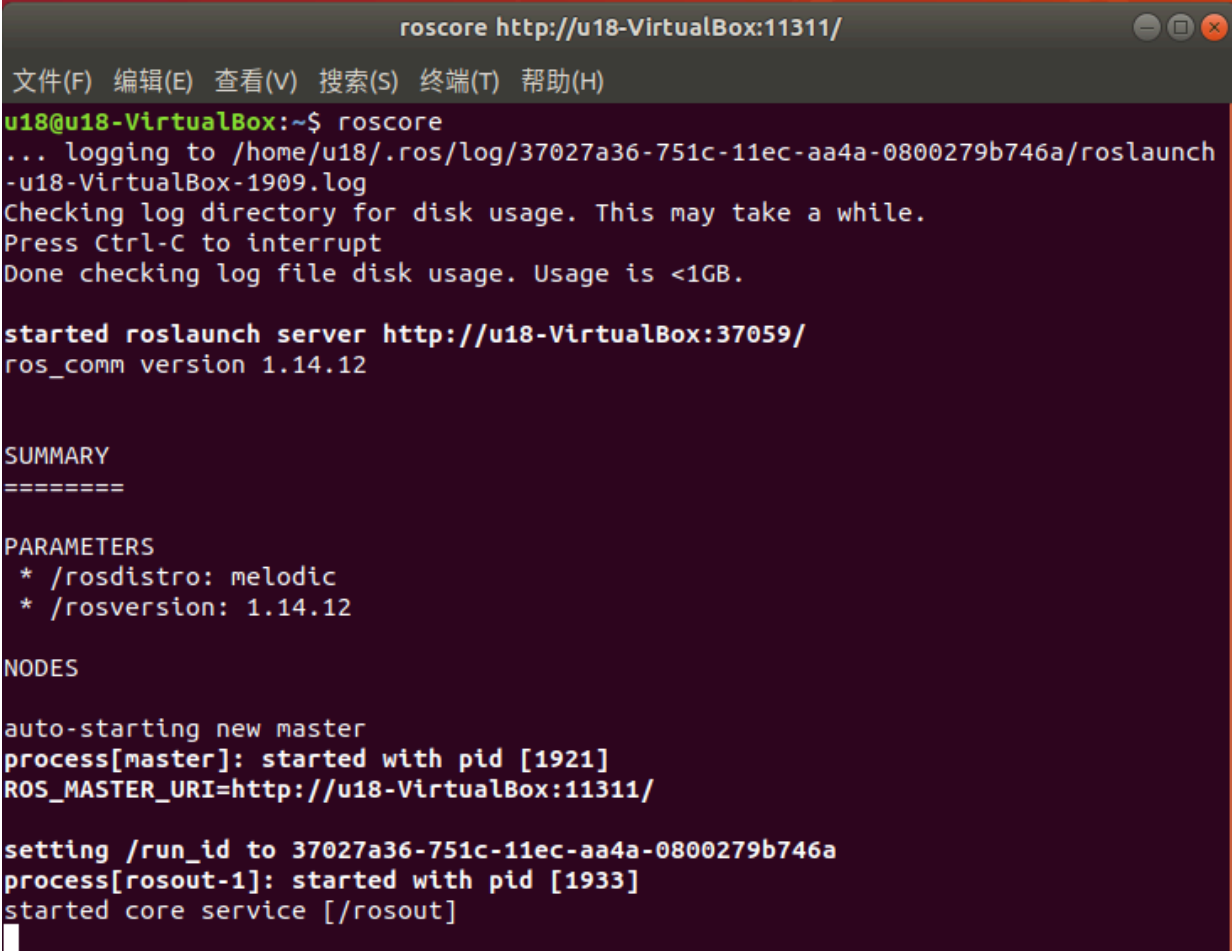
```
sudo apt install ros-noetic-joint-state-publisher-gui
```

3.1.4 Verifying installation

The startup of ROS system requires a ROS Master, that is, a node manager, and we can start ROS Master by entering the roscore command in the terminal.

To verify that ROS is installed successfully, open a console terminal (Ctrl+Alt+T) and execute the following command on the terminal:

If the following information is displayed, the ROS installation is successful

A terminal window titled 'roscore http://u18-VirtualBox:11311/' with standard window controls. The terminal shows the execution of the 'roscore' command. It logs the user 'u18' and the process ID '1909'. It checks the log directory for disk usage, reporting '<1GB'. It then announces the start of the 'roslaunch server' on 'http://u18-VirtualBox:37059/' using 'ros_comm version 1.14.12'. A 'SUMMARY' section follows, then 'PARAMETERS' listing '/rostdistro: melodic' and '/rosversion: 1.14.12'. The 'NODES' section shows 'auto-starting new master', 'process[master]: started with pid [1921]', and 'ROS_MASTER_URI=http://u18-VirtualBox:11311/'. Finally, it shows 'setting /run_id to 37027a36-751c-11ec-aa4a-0800279b746a', 'process[rosout-1]: started with pid [1933]', and 'started core service [/rosout]'.

```
roscore http://u18-VirtualBox:11311/
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
u18@u18-VirtualBox:~$ roscore
... logging to /home/u18/.ros/log/37027a36-751c-11ec-aa4a-0800279b746a/roslaunch
-u18-VirtualBox-1909.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://u18-VirtualBox:37059/
ros_comm version 1.14.12

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.12

NODES

auto-starting new master
process[master]: started with pid [1921]
ROS_MASTER_URI=http://u18-VirtualBox:11311/

setting /run_id to 37027a36-751c-11ec-aa4a-0800279b746a
process[rosout-1]: started with pid [1933]
started core service [/rosout]
```

The roscore command starts a node manager, which is used for node management. In a ros system, there is and only one, which is the prerequisite for ros nodes to run. Therefore, before starting ros nodes, the first step needs to execute roscore.

More detailed installation instructions, you can refer to the official installation guide, website:

<http://wiki.ros.org/ROS/Installation>

2. MoveIt installation

MoveIt is the component of a series of mobile operation function packages in ros, mainly including motion planning, collision detection, kinematics, 3D perception, operation control and other functions.

2.1 Update the software source list

Open a console terminal (Ctrl+Alt+T) and enter the following command in the terminal window to update the software source list:

```
sudo apt-get update
```

2.2 Install MoveIt

Open a console terminal (Ctrl+Alt+T) and execute the MoveIt installation by entering the following command in the terminal window:

```
sudo apt-get install ros-noetic-moveit
```

3. Git Installation

3.1 Add Software Source

To install Git, add the software source for Git to Ubuntu's software repository. Open a terminal (shortcut Ctrl+Alt+T) and enter the following command in the terminal window:

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

3.2 Update Software Source List

Open a terminal (shortcut Ctrl+Alt+T), and enter the following command to update the software source list:

```
sudo apt-get update
```

3.3 Install Git

Open a terminal (shortcut Ctrl+Alt+T), and enter the following command to install Git:

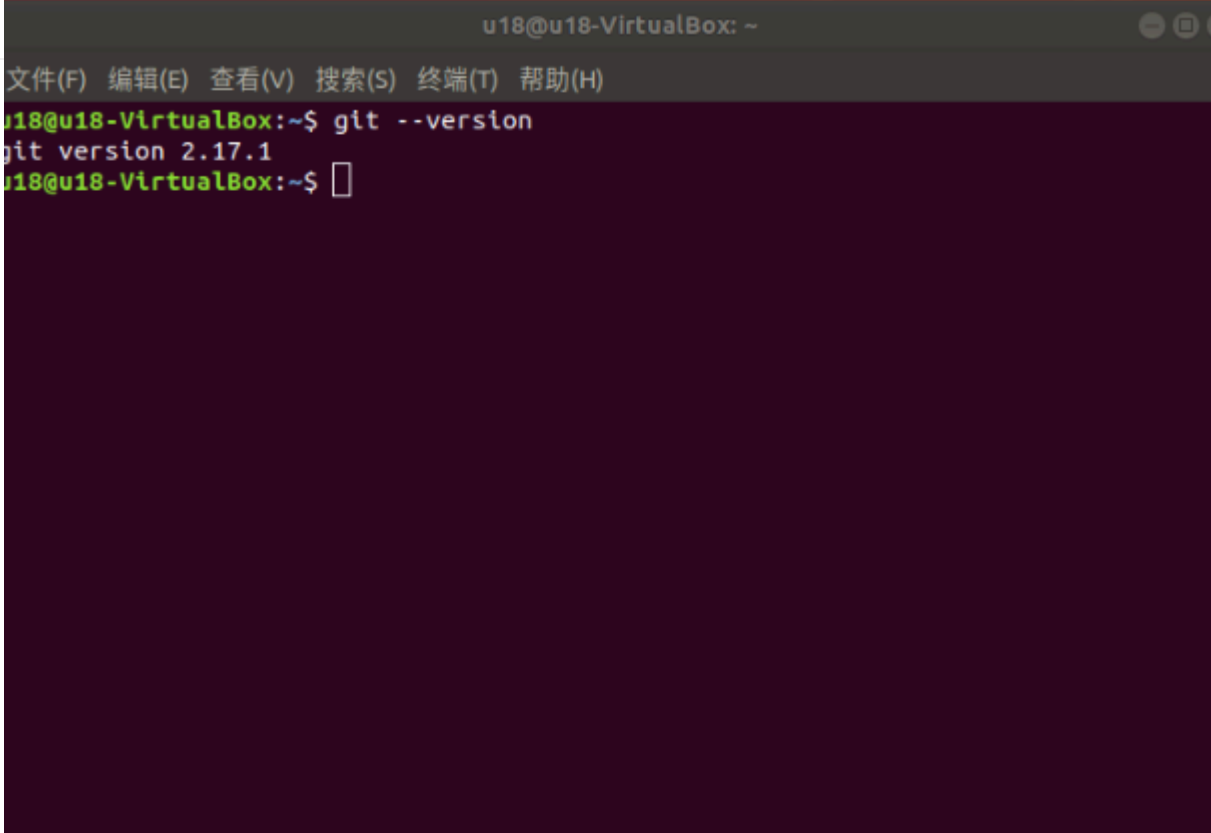
```
sudo apt-get install git
```

3.4 Verify Installation

Check the Git version. Open a terminal (shortcut Ctrl+Alt+T) and enter the following command:

```
git --version
```

If the terminal shows the Git version number as below, the installation is successful:

A terminal window titled 'u18@u18-VirtualBox: ~' with a menu bar in Chinese (文件(F), 编辑(E), 查看(V), 搜索(S), 终端(T), 帮助(H)). The terminal shows the command 'git --version' being executed, resulting in the output 'git version 2.17.1'. The prompt 'u18@u18-VirtualBox:~\$' is visible at the end of the line.

```
u18@u18-VirtualBox: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
u18@u18-VirtualBox:~$ git --version  
git version 2.17.1  
u18@u18-VirtualBox:~$
```

3.5 Usage

Git will be required for downloading ROS packages later. You can refer to the following links for Git usage:

- [Git Documentation](#)
- [Git Tutorial](#)

mycobot_ros Installation

`mycobot_ros` is a ROS package developed by ElephantRobotics, designed for their desktop six-axis robotic arm series.

Project repository: https://github.com/elephantrobotics/mycobot_ros/tree/mycontroller_s570

Prerequisites

Before installing the package, make sure you have a ROS workspace. Here is an example command to **create a workspace**. The default workspace is named `catkin_ws`. Open a terminal (shortcut Ctrl+Alt+T) and enter the following commands:

```
mkdir -p ~/catkin_ws/src # Create the folder
```

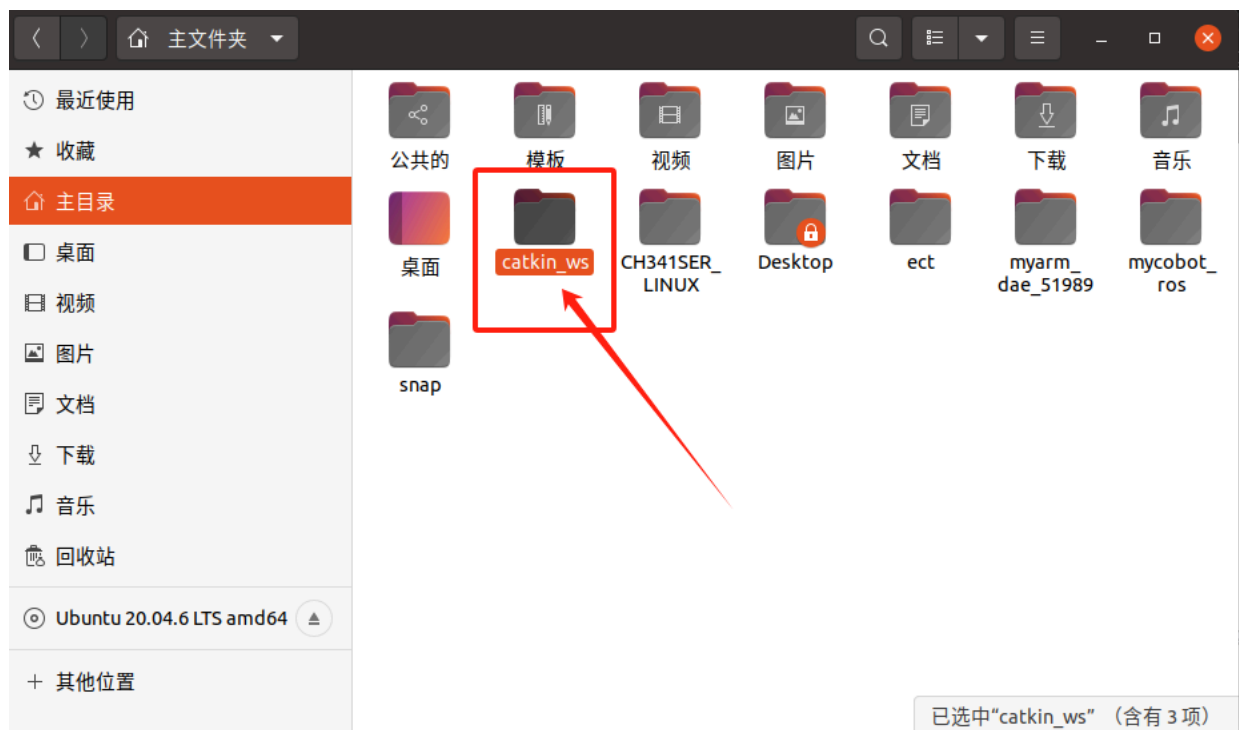
```
cd ~/catkin_ws/src # Enter the folder
```

```
catkin_init_workspace # Initialize the current directory as a ROS workspace
```

```
cd .. # Go back to the parent directory
```

```
catkin_make # Build the workspace code
```

After creation, a folder will be generated:



Add the workspace environment

Bash

The default ROS1 workspace is `catkin_ws` .

Ubuntu 16.04

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Ubuntu 18.04

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Ubuntu 20.04

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

Then run:

```
source ~/.bashrc
```

2. Installation

NOTE:

This package depends on ROS and MoveIt. Make sure that ROS and MoveIt are installed successfully before proceeding.

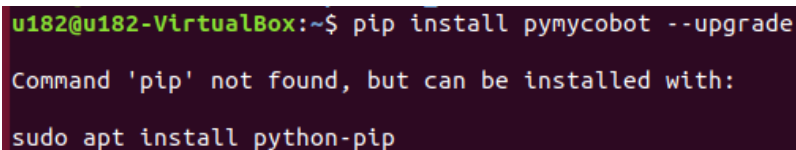
This package interacts with the actual robotic arm through the Python API - `pymycobot`.

API project repository: <https://github.com/elephantrobotics/pymycobot>

Quick installation:

```
pip install pymycobot --upgrade
```

If you encounter the following error when running the command `pip install pymycobot --upgrade`:



```
u182@u182-VirtualBox:~$ pip install pymycobot --upgrade
Command 'pip' not found, but can be installed with:
sudo apt install python-pip
```

Follow the instructions to install `pip`:

```
sudo apt install python-pip
```

If you are using Ubuntu 20.04, run:

```
sudo apt install python3-pip to install pip. Once pip is installed, run:
```

```
pip install pymycobot --upgrade
```

Note: The installation relies on Git, so make sure Git is installed on your computer.

The default ROS1 workspace is `catkin_ws`.

Open the terminal:

```
cd ~/catkin_ws/src # Enter the src folder in the workspace
```

```
git clone https://github.com/elephantrobotics/mycobot\_ros.git # Clone the repository from GitHub
```

```
cd .. # Go back to the workspace
```

```
catkin_make # Build the workspace code
```

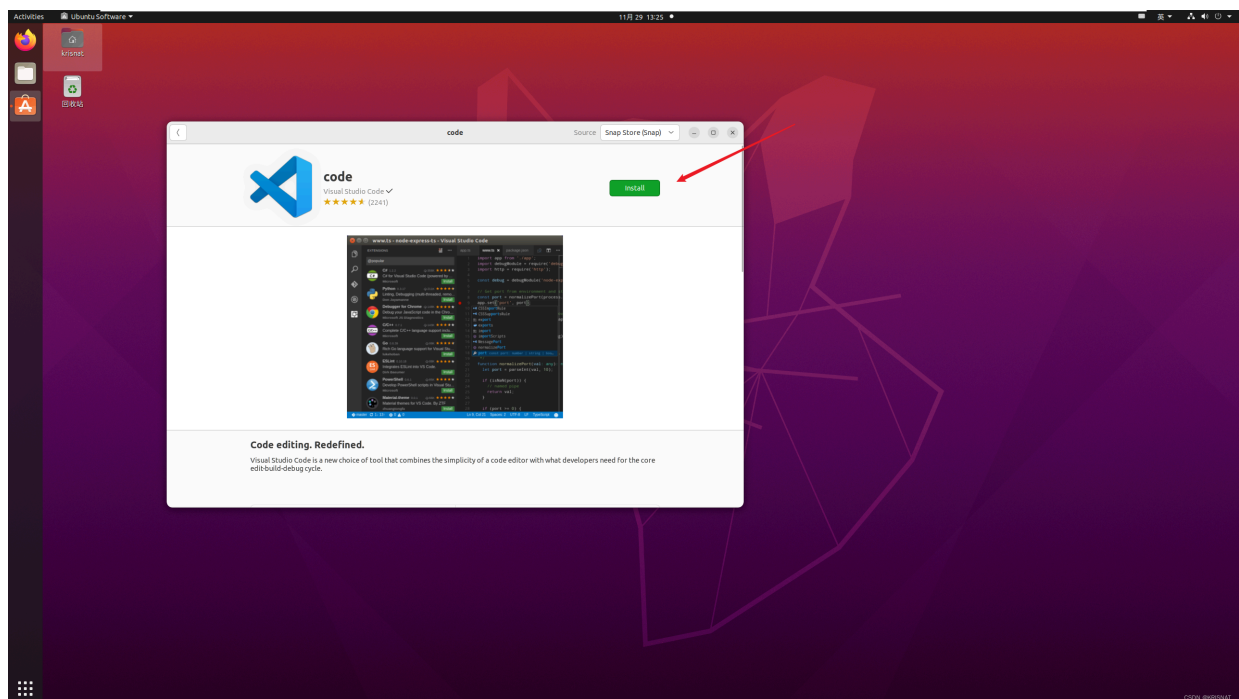
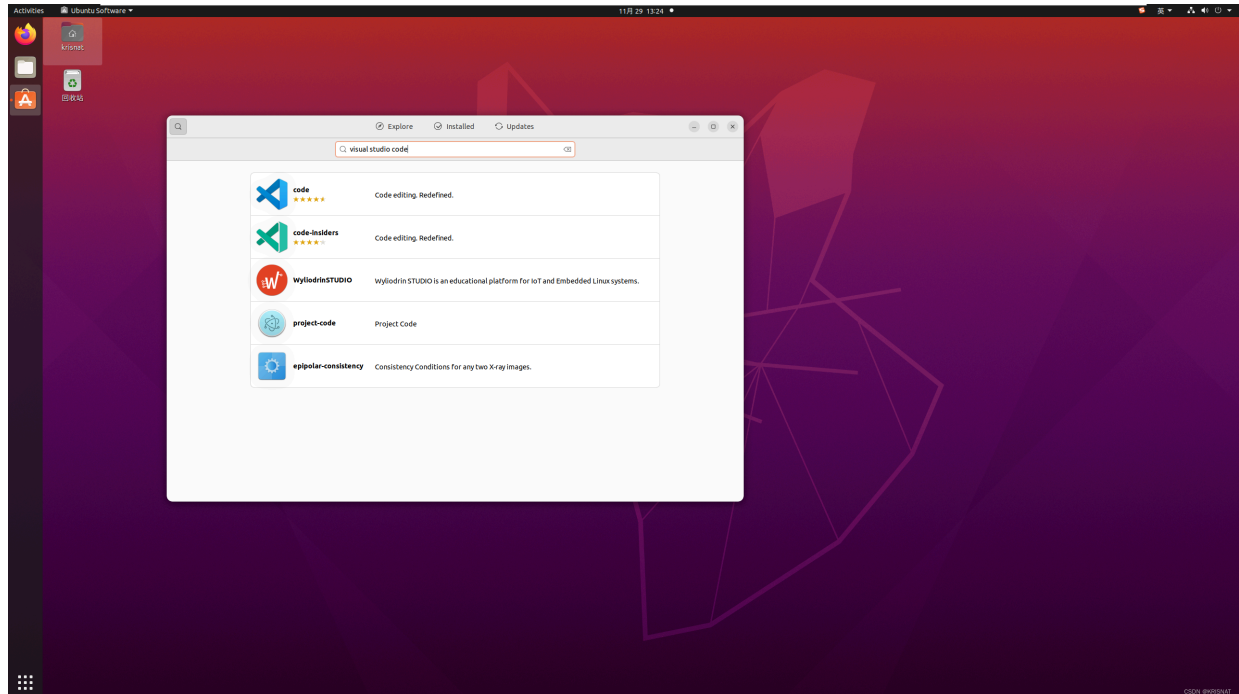
```
cd ..
```

```
source devel/setup.bash # Add environment variables
```

At this point, the environment setup is complete.

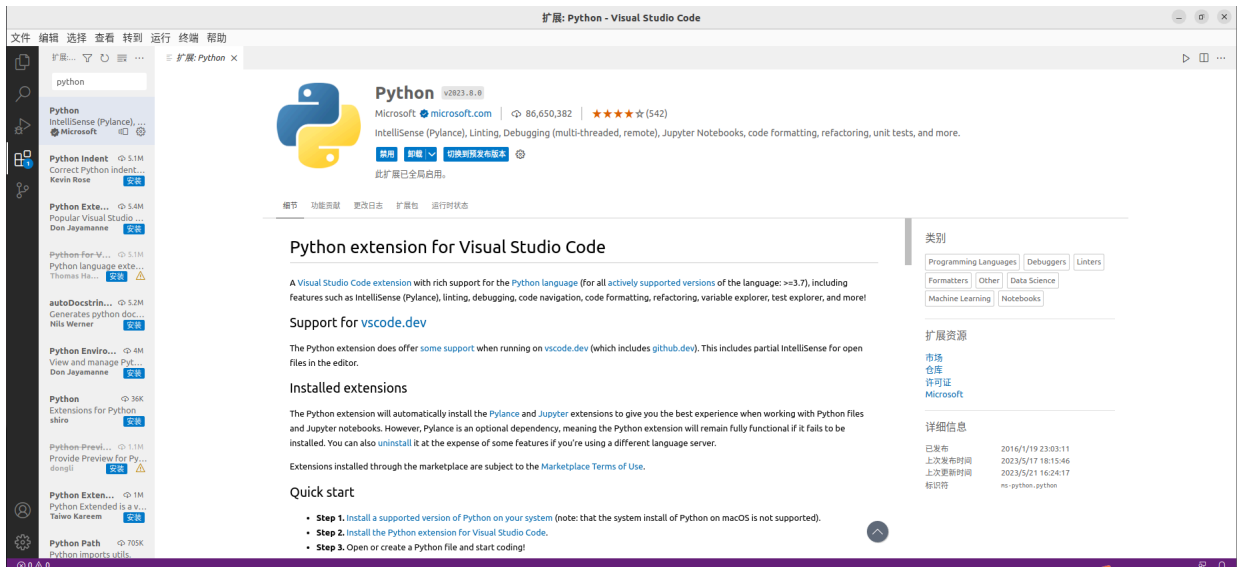
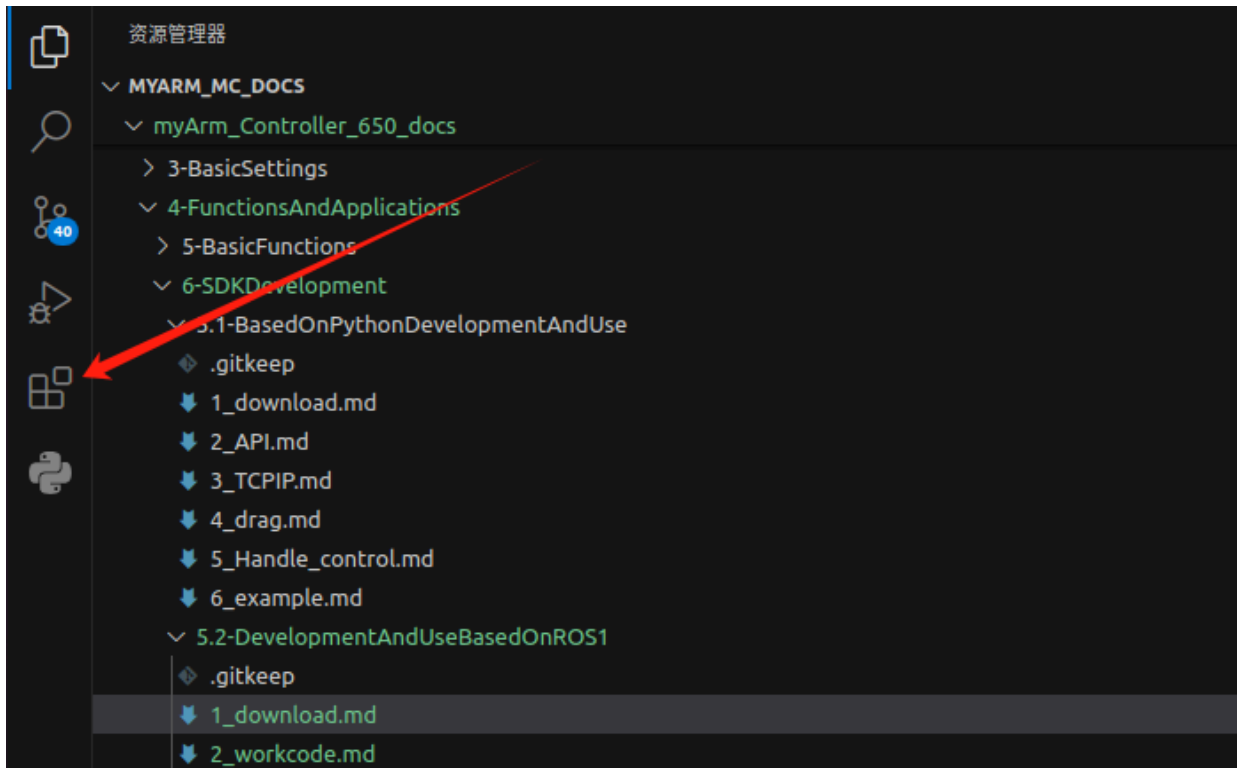
For future programming, we also need to download Visual Studio Code (VS Code).

Here is the simplest method: Install it from the Ubuntu Software Center [Simple and Convenient]



After installation, we also need to set up the Python environment.

To run Python files is very simple, click on the left extension panel (Ctrl+Shift+X), search for "Python", and install the extension.



Once installed, you can create a new Python file by clicking "File" on the top left, selecting the file type as Python, write your Python code, and then run it from the menu bar.

Introduction and Usage of RViz

RViz is a 3D visualization platform in ROS that allows for graphical display of external information and also enables sending control information to objects, thereby facilitating robot monitoring and control.

1. RViz Installation and Interface Overview

If you installed ROS with the full installation option, RViz has already been installed, and you can try running it directly. If not, you can install RViz separately:

- **Ubuntu 16.04**

```
sudo apt-get install ros-kinetic-rviz
```

- **Ubuntu 18.04**

```
sudo apt-get install ros-melodic-rviz
```

- **Ubuntu 20.04**

```
sudo apt-get install ros-noetic-rviz
```

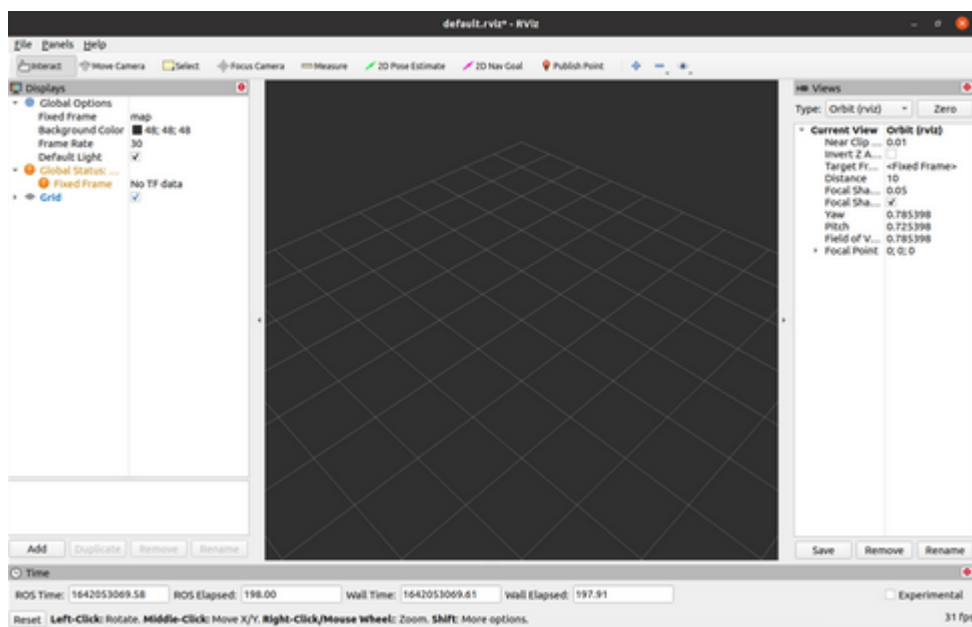
After installation, open a new terminal (shortcut Ctrl+Alt+T) and enter the following command:

```
roscore
```

Then, open another new terminal (Ctrl+Alt+T) and enter the command to open RViz:

```
roslaunch rviz rviz or rviz
```

After opening RViz, the following interface will be displayed:



2. Description of Each Area

- The left panel displays the list of displays, which are used to render certain content in the 3D world, and may have some available options in the display list.
- The top part is the toolbar, which allows users to select various functional tools.
- The central area is the 3D view: it is the main screen where various data can be viewed in 3D. The background color, fixed frame, and grid can be set in the Global Options and Grid sections in the left panel.
- The bottom area shows the time, including system time and ROS time.
- The right area is for setting the viewpoint, allowing different observation angles.

This section is a brief overview. For more detailed information, you can refer to the [User Guide](#).

[← Previous](#) | [Next page →](#)

I. Simulation

We provide some **static simulations** and **dynamic simulations** to interact with the MyController S570.

1.1 Static Simulation

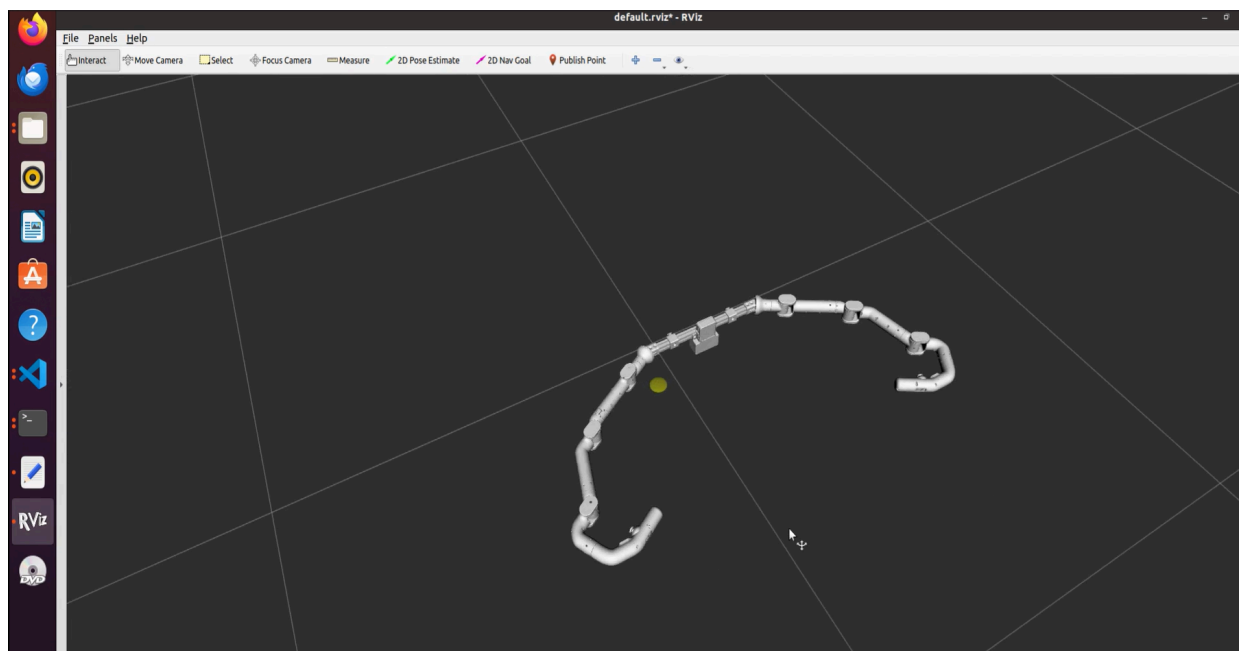
Static simulation here refers to interacting with the MyController S570 in the simulation using the sliding module in rviz.

To do this, open a terminal in the workspace and enter the following commands:

```
source devel/setup.bash # Add environment variables
```

```
roslaunch myarm_m test02.launch
```

This will also open rviz and generate the simulation model of MyController S570.



1.2 Dynamic Simulation

① First turn on the switch on MyController S570, and check whether the screen displays normally (that is, the Angle of each joint and the information of the button appear).

② Connect the MyController S570 to the computer through the serial port

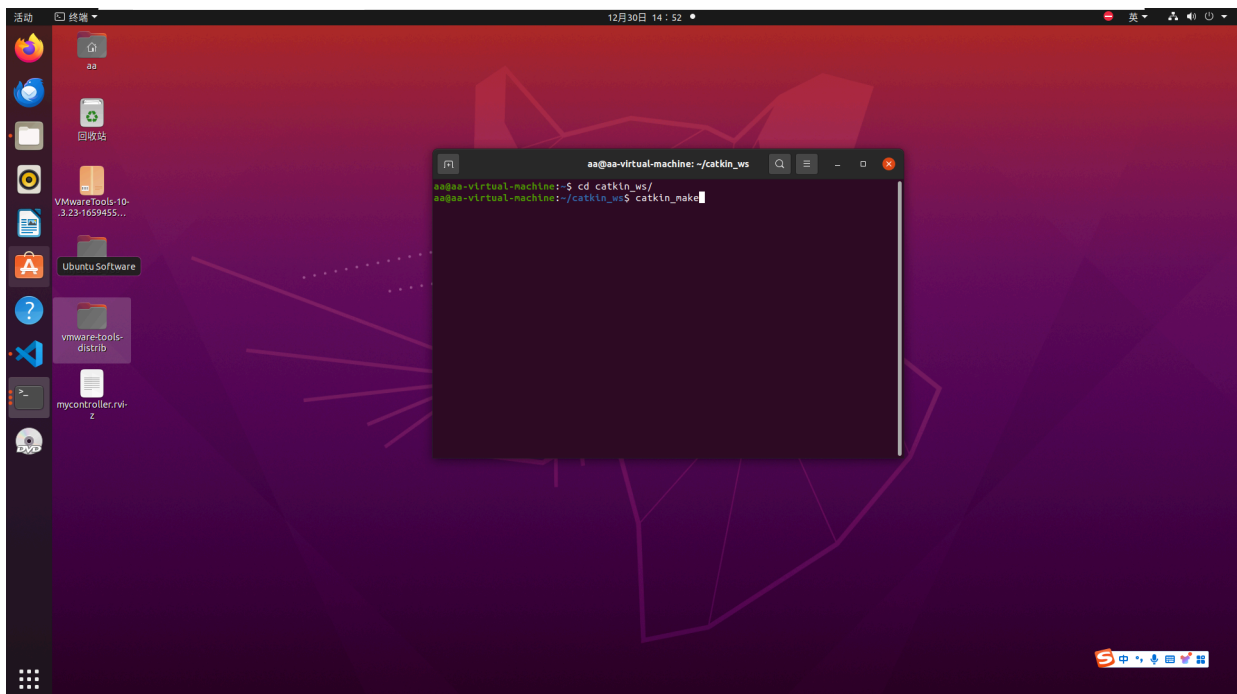
③ In the system, open the terminal command line and enter:

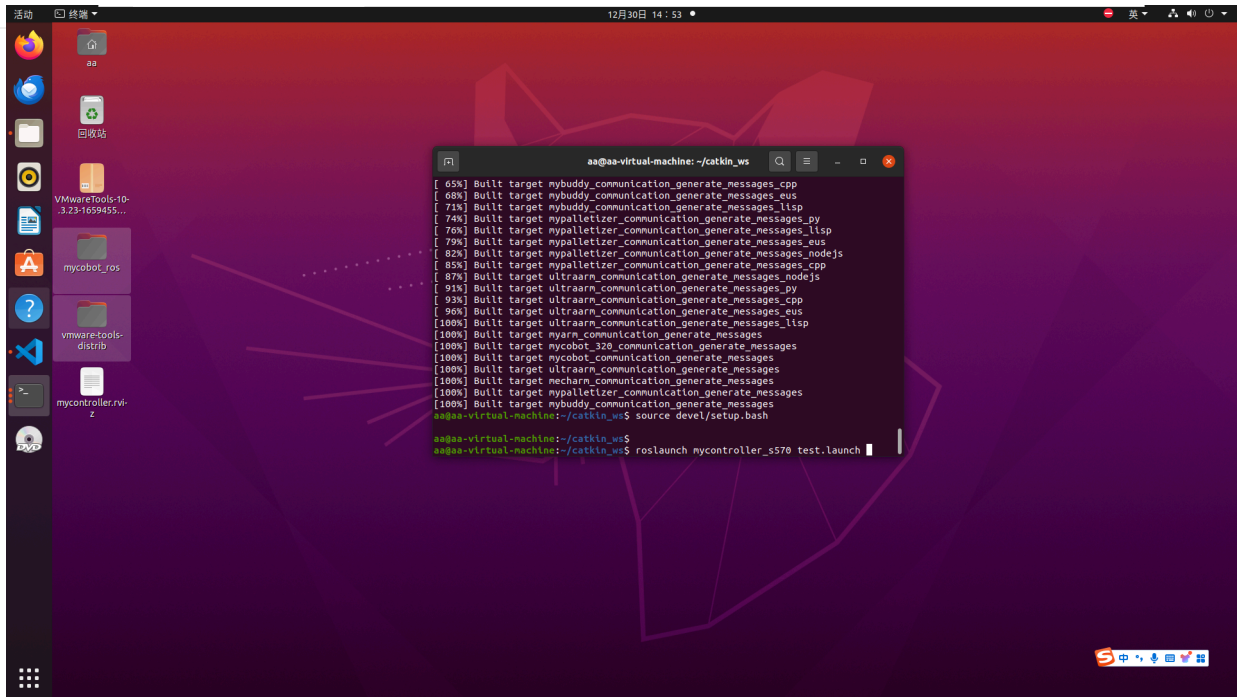
```
cd catkin_ws
```

```
catkin_make
```

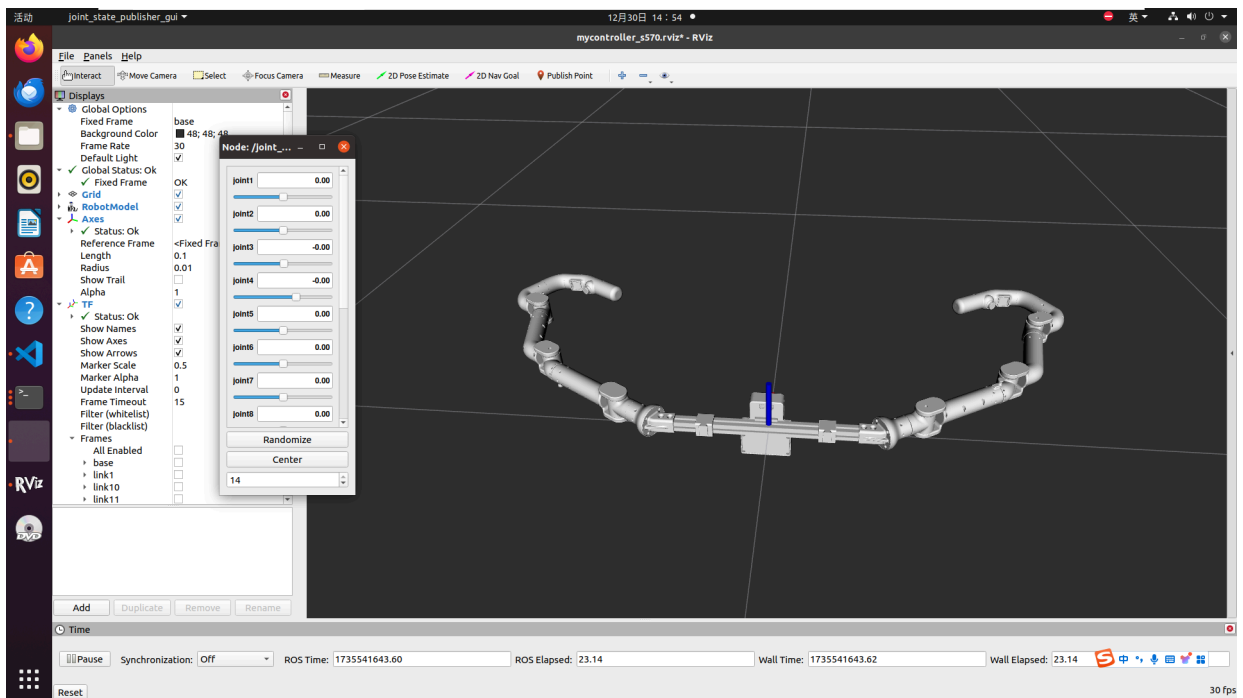
```
source devel/setup.bash
```

```
roslaunch mycontroller_s570 test.launch
```



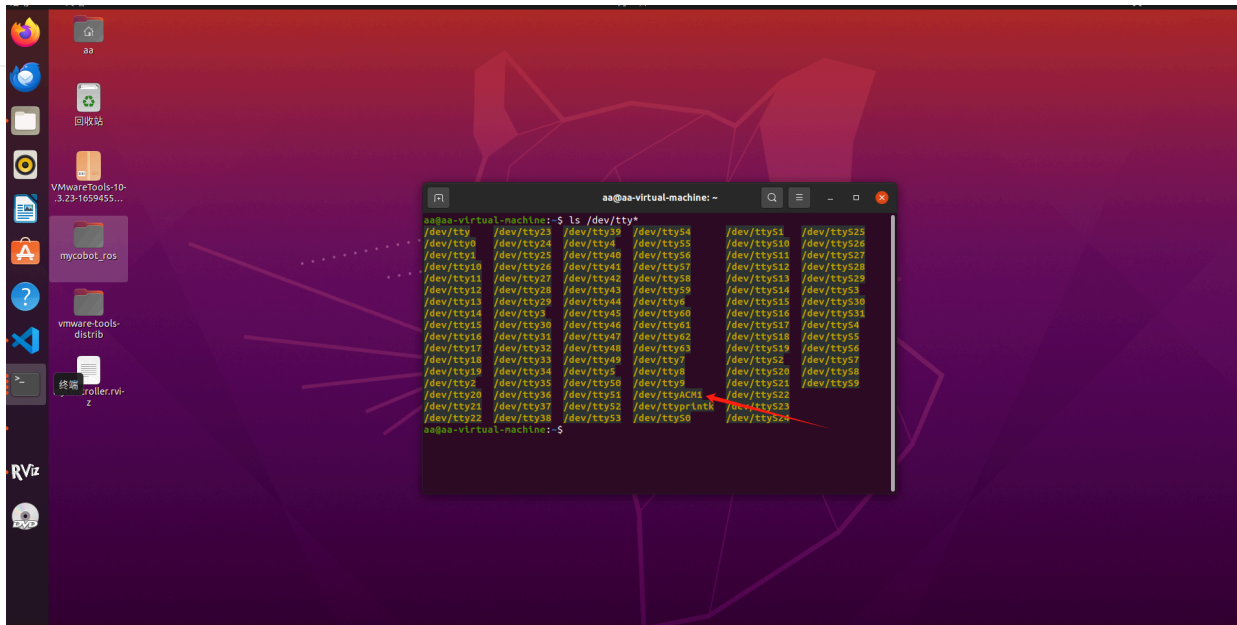


At the same time, rviz will be opened to generate a simulation model of MyController S570

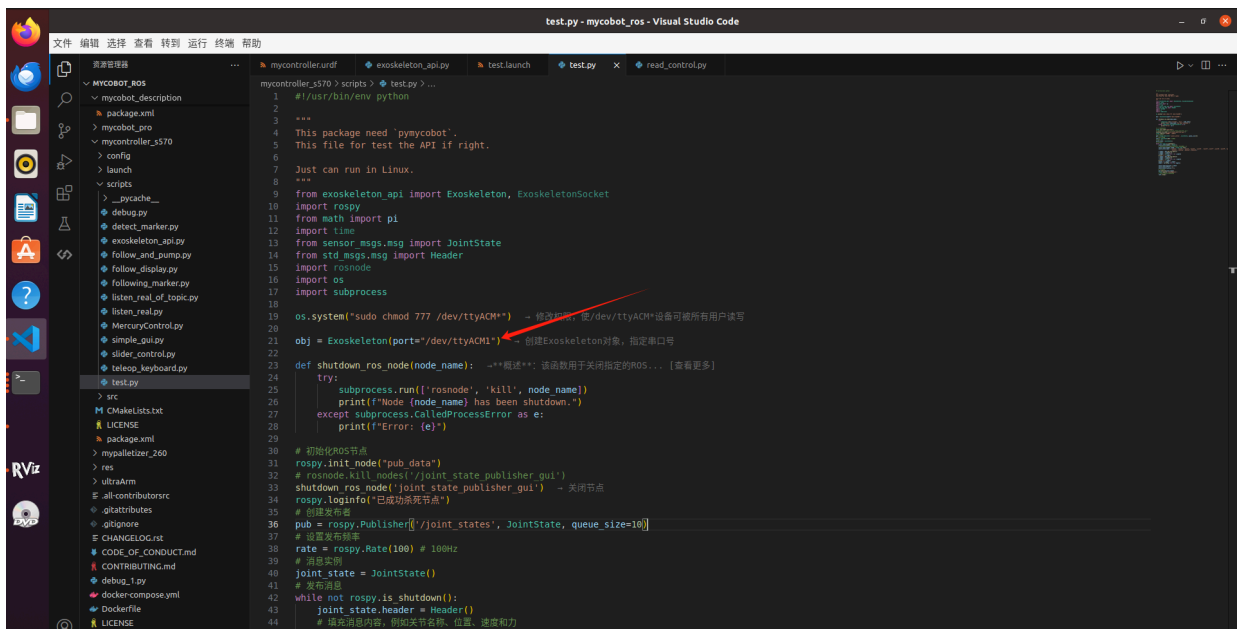


④ Create a terminal and enter: (You can view the serial port number of the S570 connecting to the system)

```
ls /dev/tty*
```



⑤ Open the script file, modify the serial port number, and save:

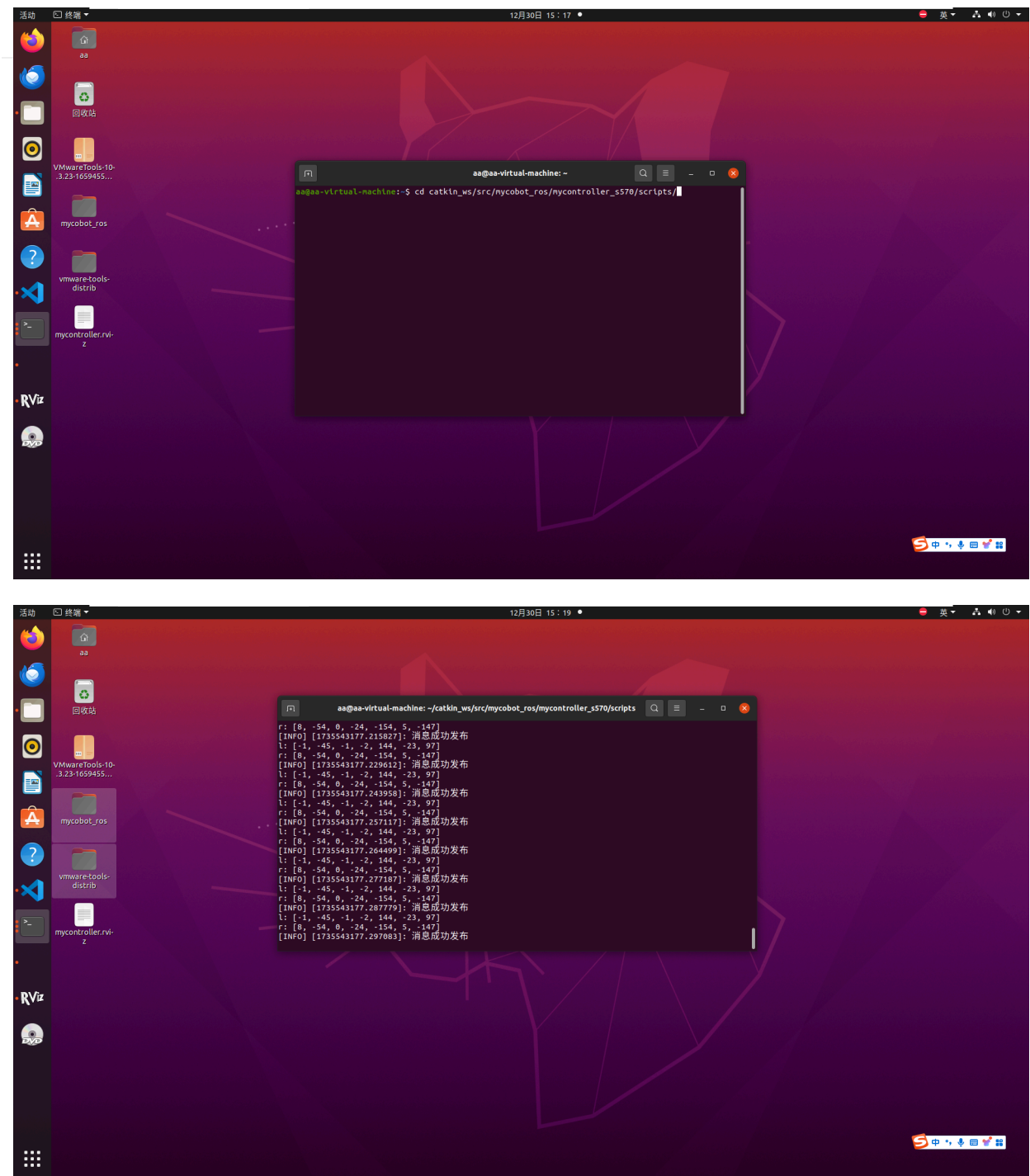


⑥ Create a new terminal and enter:

```

cd catkin_ws/src/mycobot_ros/mycontroller_s570/scripts
python3 test.py

```



At this point, the exoskeleton successfully achieved model following motion

Effects Video:



[← Previous Page](#) | [Next Page →](#)

1. Introduction

ROS provides some useful command-line tools to retrieve various information about different nodes. The commonly used commands are as follows:

- `roscall` : Manage nodes
- `rostopic` : Manage topics
- `rosservice` : Manage services
- `rosmmsg` : Manage msg messages
- `rossrv` : Manage srv messages
- `rosparam` : Manage parameters

Official reference: <http://wiki.ros.org/ROS/CommandLineTools>

2. roscall

`roscall` is used to retrieve node information.

```
1 | roscall ping      测试到节点的连接状态
2 | roscall list      列出活动节点
3 | roscall info      打印节点信息
4 | roscall machine   列出指定设备上节点
5 | roscall kill      杀死某个节点
6 | roscall cleanup   清除不可连接的节点
```

3. rostopic

`rostopic` includes a command-line tool for displaying debugging information related to ROS topics, such as publishers, subscribers, publish rate, and ROS messages. It also includes an experimental Python library to dynamically retrieve and interact with topic information.

```
1 | rostopic bw      显示主题使用的带宽
2 | rostopic delay    显示带有 header 的主题延迟
3 | rostopic echo     打印消息到屏幕
4 | rostopic find     根据类型查找主题
5 | rostopic hz       显示主题的发布频率
6 | rostopic info     显示主题相关信息
7 | rostopic list     显示所有活动状态下的主题
8 | rostopic pub      将数据发布到主题
9 | rostopic type     打印主题类型
```

3.1 rostopic list

Invoke the command directly, and the console will print the list of topic names in the current running state.

`rostopic list -v` : Get detailed information about the topics (e.g., the number of publishers and subscribers).

3.2 rostopic pub

You can invoke the command to publish messages to the subscribers.

For example, publishing a string message to a subscriber in the RoboWare auto-generated Publish/Subscribe model case.

```
1 | rostopic pub /主题名称 消息类型 消息内容
2 | rostopic pub /liaotian std_msgs huluwa
```

Example of publishing a movement message for the turtle:

```
1 | rostopic pub /turtle1/cmd_vel geometry_msgs/Twist
2 |   "linear:
3 |     x: 1.0
4 |     y: 0.0
5 |     z: 0.0
6 |   angular:
7 |     x: 0.0
8 |     y: 0.0
9 |     z: 2.0"
10 | //只发布一次运动信息
11 |
12 | rostopic pub -r 10 /turtle1/cmd_vel geometry_msgs/Twist
13 |   "linear:
14 |     x: 1.0
15 |     y: 0.0
16 |     z: 0.0
17 |   angular:
18 |     x: 0.0
19 |     y: 0.0
20 |     z: 2.0"
21 | // 以 10HZ 的频率循环发送运动信息
```

3.3 rostopic echo

Get the current message published by a specified topic.

3.4 rostopic info

Get detailed information about the current topic:

1. Message type
2. Publisher information
3. Subscriber information

4. rosservice

`rosservice` contains command-line tools for listing and querying ROS services.

If calling certain services, and the workspace is not configured with the path, you may need to enter the workspace and invoke `source devel/setup.bash .`

```
1 | rosservice args 打印服务参数
2 | rosservice call  使用提供的参数调用服务
3 | rosservice find  按照服务类型查找服务
4 | rosservice info  打印有关服务的信息
5 | rosservice list  列出所有活动的服务
6 | rosservice type  打印服务类型
7 | rosservice uri   打印服务的 ROSRPC uri
```

4.1 rosservice args

Print service parameters

```
1 | rosservice args /spawn
2 | x y theta name
```

4.2 rosservice call

Call a service to generate a **new turtle**.


```

1 | rosservice call /spawn "x: 1.0
2 | y: 2.0
3 | theta: 0.0
4 | name: 'xxx'"
5 | name: "xxx"
6 |
7 | //生成一只叫 xxx 的乌龟
8 |

```

5. rosmmsg

`rosmmsg` is a command-line tool used to display information about ROS message types.

```

1 | rosmmsg show    显示消息描述
2 | rosmmsg info    显示消息信息
3 | rosmmsg list    列出所有消息
4 | rosmmsg md5     显示 md5 加密后的消息
5 | rosmmsg package 显示某个功能包下的所有消息
6 | rosmmsg packages 列出包含消息的功能包

```

5.1 rosmmsg package

List all msg files in a package.

```

1 | rosmmsg package 包名

```

5.2 rosmmsg show

Display the message description.

```

1 | //rosmmsg show 消息名称
2 | rosmmsg show turtlesim/Pose
3 | 结果:
4 | float32 x
5 | float32 y
6 | float32 theta
7 | float32 linear_velocity
8 | float32 angular_velocity

```


6. rossrv

`rossrv` is a command-line tool used to display information about ROS service types, similar to `rosmmsg`.

```
1 | rossrv show      显示服务消息详情
2 | rossrv info      显示服务消息相关信息
3 | rossrv list      列出所有服务信息
4 | rossrv md5       显示 md5 加密后的服务消息
5 | rossrv package   显示某个包下所有服务消息
6 | rossrv packages  显示包含服务消息的所有包
```

6.1 rossrv package

```
1 | rossrv package 包名
```

6.2 rossrv show

Display the service description.

```
1 | //rossrv show 消息名称
2 | rossrv show turtlesim/Spawn
3 | 结果:
4 | float32 x
5 | float32 y
6 | float32 theta
7 | string name
8 | ---
9 | string name
```

7. rosparam

`rosparam` contains a command-line tool for retrieving and setting ROS parameters using YAML-encoded files on the parameter server.

```
1 | rosparam set      设置参数
2 | rosparam get      获取参数
3 | rosparam load     从外部文件加载参数
4 | rosparam dump     将参数写出到外部文件
5 | rosparam delete   删除参数
6 | rosparam list     列出所有参数
```

7.1 rosparam set

Set a parameter.

```
rosparam set name xxx
```

7.2 rosparam get

Get a parameter.

```
rosparam get name
```

7.3 rosparam delete

Delete a parameter.

```
rosparam delete name
```

7.4 rosparam load (Prepare the YAML file)

Load parameters from an external file.

```
rosparam load xxx.yaml
```

7.5 rosparam dump

Dump parameters to an external file.

```
rosparam dump yyy.yaml
```

8. Introduction to URDF

- Unified Robot Description Format (URDF) is an XML format used to describe a robot model. The URDF package in ROS contains a C++ parser, and URDF files are used to describe robot models.
- URDF cannot be used alone. It needs to be rendered into a graphical model in Rviz or Gazebo. URDF is just a file that must be visualized in Rviz or Gazebo.

8.1 URDF File Description

Code example (only part of the code is shown here):

```

<link name="base">
  <visual>
    <geometry>
      <mesh filename="package://mycobot_description/urdf/myarm_c650/myarm_basic_base.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0" rpy = "0 0 0"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://mycobot_description/urdf/myarm_c650/myarm_basic_base.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0" rpy = "0 0 0"/>
  </collision>
</link>

<link name="link1">
  <visual>
    <geometry>
      <mesh filename="package://mycobot_description/urdf/myarm_c650/joint1.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0.0065" rpy = "3.14159 0 0"/>
  </visual>
  <collision>
    <geometry>
      <mesh filename="package://mycobot_description/urdf/myarm_c650/joint1.dae"/>
    </geometry>
    <origin xyz = "0.0 0 0.0065" rpy = "3.14159 0 0"/>
  </collision>

```

```

<joint name="joint1" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort = "1000.0" lower = "-2.7925" upper = "2.7925" velocity = "0"/>
  <parent link="base"/>
  <child link="link1"/>
  <origin xyz= "0 0 0.070" rpy = "3.14159 0 0"/>
</joint>

<joint name="joint2" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort = "1000.0" lower = "-1.3962" upper = "1.3962" velocity = "0"/>
  <parent link="link1"/>
  <child link="link2"/>
  <!-- <origin xyz= "0 0 0" rpy = "1.5708 3.14159 0"/> -->
  <origin xyz= "0 0 -0.080" rpy = "1.5708 0 0"/>
</joint>

<joint name="joint3" type="revolute">
  <axis xyz="0 0 1"/>
  <limit effort = "1000.0" lower = "-1.3962" upper = "1.3962" velocity = "0"/>
  <parent link="link2"/>
  <child link="link3"/>
  <origin xyz= "0.0565 -0.296 0" rpy = "0 0 0"/>
</joint>

```

As can be seen, the URDF file is not complex. It is mainly composed of **link** and **joint** elements that repeat.

8.2 link Section

The `link` element describes rigid bodies with inertia, visual features, and collision properties.

8.2.1 Attributes

- `name` : Describes the name of the link itself.

8.2.2 Elements

- **visual**
 - The visualization properties of the link. It specifies the shape of the link (e.g., box, cylinder). Multiple visual elements can be used to form the shape of the link. More complex shapes can be imported from STL files generated in software like SolidWorks, while simpler shapes (e.g., end effectors) can be written directly. You can also adjust the geometry of the link to match theoretical models or actual models.
 - **name** (optional): The name of the link's geometry shape.
 - **origin** (optional, defaults to identity if not specified):
 - Specifies the coordinate system for the geometry relative to the link's coordinate system.
 - `xyz` (optional, defaults to zero vector): Represents offsets in the x, y, and z directions (in meters).
 - `rpy` (optional, defaults to identity if not specified): Represents rotation along the RPY axes (in radians).
- **geometry** (required):
 - Describes the shape of the visual object, which can be one of the following:
 - **box**: Rectangular, with length, width, and height.
 - **cylinder**: Cylindrical, with radius and length.
 - **sphere**: Spherical, with radius.
 - **mesh**: A mesh, defined by a file, with an optional scale. Recommended to use Collada (.dae) files, though STL files are also supported, but they must be local files.
- **collision** (optional):
 - Collision properties of the link. They are used for simplifying collision calculations and differ from visualization properties. A link can have multiple collision properties.
 - **name** (optional): Specifies the name of the link's geometry.
 - **origin** (optional, defaults to identity if not specified): Specifies the reference coordinate system for the collision component relative to the link's coordinate system.
 - **geometry**: Same as the geometry element described above.

For detailed elements and their usage, refer to the [official documentation](#).

8.3 joint Section

The `joint` section describes the kinematics and dynamics of a joint and specifies its relationship with other links. The main attributes and elements are as follows:

8.3.1 Joint Attributes:

- **name**:

Specifies the unique name of the joint.

- **type:**
-

Specifies the type of joint, which can be one of the following:

- **revolute:** A revolute joint that rotates along an axis, with the range specified by upper and lower limits.
- **continuous:** A continuous revolute joint that rotates around an axis without upper or lower limits.
- **prismatic:** A prismatic joint that slides along an axis, with the range specified by upper and lower limits.
- **fixed:** This is not a real joint as it cannot move. All degrees of freedom are locked. A fixed joint does not require an axis, calibration, dynamics, limits, or `safety_controller`.
- **floating:** This joint allows motion in all 6 degrees of freedom.
- **planar:** This joint allows motion in a plane perpendicular to the axis.

8.3.2 Joint Elements

- **origin** (optional, defaults to identity if not specified):

The transformation from the parent link to the child link. The joint is located at the origin of the child link. Modifying this parameter can adjust the position of the links. It can be useful for adjusting discrepancies between the actual model and the theoretical model, but large modifications are not recommended as this can affect the link's STL file position and may interfere with collision detection.

- **xyz** (optional, defaults to zero vector): Specifies the offset in the x, y, and z directions in meters.
- **rpy** (optional, defaults to zero vector): Specifies the rotation angles around the fixed axes: roll (around the x-axis), pitch (around the y-axis), and yaw (around the z-axis), in radians.
- **parent** (required):

The name of the parent link. This is a mandatory attribute.

- The name of the parent link refers to the link's name in the robot's structure tree.
- **child** (required):

The name of the child link. This is a mandatory attribute.

- The name of the child link refers to the link's name in the robot's structure tree.
- **axis** (optional, defaults to (1,0,0)):

The axis of the joint in the joint's coordinate system. This is the axis of rotation (for revolute joints), the axis of motion (for prismatic joints), or the standard plane for planar joints. This axis is specified in the joint's coordinate system. Modifying this parameter can adjust the axis around which the joint rotates, and it's commonly used to adjust the rotation direction. If the model rotates in the opposite direction to the actual model, simply multiply by -1. Fixed and floating joints do not require this element.

-
- **xyz** (required): Specifies the components of the axis vector in x, y, and z directions as a normalized vector.

- **calibration** (optional):
-

Specifies the reference point for the joint, used to correct the absolute position of the joint.

- **rising** (optional): When the joint moves in the positive direction, a rising edge is triggered at the reference point.
- **falling** (optional): When the joint moves in the positive direction, a falling edge is triggered at the reference point.
- **dynamics** (optional):

This element specifies the physical properties of the joint. The values are used to describe the joint's behavior, especially in simulations.

- **limit** (required for revolute or prismatic joints):
 - This element defines the kinematic constraints of the joint.
 - **lower** (optional, defaults to 0): Specifies the lower limit of the joint's motion (in radians for revolute joints, in meters for prismatic joints). Continuous joints ignore this attribute.
 - **upper** (optional, defaults to 0): Specifies the upper limit of the joint's motion (in radians for revolute joints, in meters for prismatic joints). Continuous joints ignore this attribute.
 - **effort** (required): Specifies the maximum force that the joint can apply during its operation.
 - **velocity** (required): Specifies the maximum speed at which the joint can move.

For more detailed information about the elements and their usage, refer to the [documentation](#).

[← Previous page](#) | [Next page →](#)

Communication and Message Commands

Notice: Before communicating directly through the communication protocol, it is necessary to burn the latest firmware of the exoskeleton on BASIC and burn the latest ATOM firmware

Please read the interface protocol document in detail and proceed with your program development after testing.

1 Settings of USB Communication

Make sure the following settings are prepared:

- Interface of mainline: USB Type-C connection PC and Basic
- Port ratio: 1000000
- Data bit: 8
- Parity bit: none
- Stop bit: 1

2 Introduction to Command Frame & Sole Instruction

The PC transmits data through BASIC, and the robotic arm decodes the data into commands with return values and uses serial communication protocol to send the results back to the PC through BASIC within 5 milliseconds. The communication frequency can reach 200Hz.

3 Formats of Message Commands' Sending and Receiving

Both sending and receiving should be represented in hexadecimal. Each command should contain 5 parts as shown below. Part 3 and 4 can be left a blank.

- 1 Pin of command: 0xFE 0xFE
 - Invariable
 - Indispensable
- 2 Effective lengthen:
 - Aggregated length including pin, serial number, functional codes and end
 - Indispensable
- 3 Serial number: 00 ~ 8F
 - Corresponding number of developed commands
 - You may leave it blank.
- 4 Functional codes:
 - Purpose-oriented
 - You may leave it blank.
- 5 End: 0XFA
 - Invariable
 - Indispensable

4 Explanation for Commands

Type	Data	Length	Function
Command Frame	start bit: 0	1	Head frame identification, 0XFE
	start bit: 1	1	Head frame identification, 0XFE
	bit of data length	1	Different commands correspond to different data length
	command bit	1	depending on different commands
Command Frame	data	0-16	commands with data, depending on different commands
End Frame	end bit	1	stop bit, 0XFA

5 Explanation for Sole-Instruction Commands

Read all data of both arms.

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X01
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 01 FA

Return value: data structure

Data Domain	Description	Data
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Return length	0X3C
Data[3]	Return value: Command frame	0X01
Data[4]	Return value: Left arm joint 1 high	Left_Servo1_high
Data[5]	Return value: Left arm joint 1 low position	Left_Servo1_Low
Data[6]	Return value: Left arm joint 2 high	Left_Servo2_high
Data[7]	Return value: Left arm joint 2 low	Left_Servo2_Low
Data[8]	Return value: Left arm joint 3 high	Left_Servo3_high
Data[9]	Return value: Left arm joint 3 low position	Left_Servo3_Low
Data[10]	Return value: Left arm joint 4 high	Left_Servo4_high
Data[11]	Return value: Left arm joint 4 low position	Left_Servo4_Low
Data[12]	Return value: Left arm joint 5 high	Left_Servo5_high
Data[13]	Return value: Left arm joint 5 low position	Left_Servo5_Low
Data[14]	Return value: Left arm joint 6 high	Left_Servo6_high
Data[15]	Return value: Left arm joint 6 low position	Left_Servo6_Low
Data[16]	Return value: Left arm joint 7 high	Left_Servo7_High
Data[17]	Return value: Left arm joint 7 low position	Left_Servo7_Low
Data[18]	Return value: Left Arm remote sensing button status	Left_button_status (bit1: Remote sensing button bit2: Button 1 bit3: Button 2)
Data[19]	Return value: Left Arm remote sensing x value	Left_remote_sensing_x-value
Data[20]	Return value: Left Arm remote sensing y value	Left_remote_sensing_y-value
Data[21]	Return value: Right arm Joint 1 high	Right_Servo1_high

Data Domain	Description	Data
Data[22]	Return value: Right arm joint 1 low position	Right_Servo1_Low
Data[23]	Return value: Right arm joint 2 high	Right_Servo2_high
Data[24]	Return value: Right arm joint 2 low position	Right_Servo2_Low
Data[25]	Return value: Right arm joint 3 high	Right_Servo3_high
Data[26]	Return value: Right arm joint 3 low position	Right_Servo3_Low
Data[27]	Return value: Right arm joint 4 high	Right_Servo4_high
Data[28]	Return value: Right arm joint 4 low position	Right_Servo4_Low
Data[29]	Return value: Right arm joint 5 high	Right_Servo5_high
Data[30]	Return value: Right arm joint 5 low position	Right_Servo5_Low
Data[31]	Return value: Right arm joint 6 high	Right_Servo6_high
Data[32]	Return value: Right arm joint 6 low position	Right_Servo6_Low
Data[33]	Return value: Right arm joint 7 high	Right_Servo7_High
Data[34]	Return value: Right arm joint 7 low position	Right_Servo7_Low
Data[35]	Return value: Right Arm remote sensing button status	Right_button_status (bit1: Remote sensing button bit2: Button 1 bit3: Button 2)
Data[36]	Return value: Right Arm remote sensing x value	Right_remote_sensing_x-value
Data[37]	Return value: Right Arm remote sensing Y-value	Right_remote_sensing_y-value
Data[38]	Return value: Left arm X-axis high	Right_Servo2_high
Data[39]	Return value: Left arm X-axis low	Right_Servo2_Low

Data Domain	Description	Data
Data[40]	Return value: High Y-axis of the left arm	Right_Servo3_high
Data[41]	Return value: Low Y-axis of the left arm	Right_Servo3_Low
Data[42]	Return value: Left arm Z-axis high	Right_Servo4_high
Data[43]	Return value: Low Z-axis of the left arm	Right_Servo4_Low
Data[44]	Return value: Left arm rx rotation high	Right_Servo5_high
Data[45]	Return value: Left arm rx rotation low position	Right_Servo5_Low
Data[46]	Return value: Left arm ry rotation high	Right_Servo6_high
Data[47]	Return value: Left arm ry rotation low position	Right_Servo6_Low
Data[48]	Return value: Left arm rz rotation high	Right_Servo7_High
Data[49]	Return value: Left arm rz rotation low position	Right_Servo7_Low
Data[50]	Return value: Right arm X-axis high	Right_Servo2_high

Example:

Assuming Atom has successfully connected:

```
Return value of port: FE FE 24 01 00 00 00 00 00 00 00 00 00 00 00 00 14 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 14 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FA
```

Read all data from a single arm

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X02
Data[4]	Specify a single arm	0x01(Left)/0x02(Right)
Data[5]	end frame	0XFA

Example:

Port transmission: FE FE 03 02 01 FA

Return value: data structure

Data Domain	Description	Data
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Return length	0X13
Data[3]	Return value: Command frame	0X02
Data[4]	Return value: Left arm joint 1 high	Left_Servo1_high
Data[5]	Return value: Left arm joint 1 low position	Left_Servo1_Low
Data[6]	Return value: Left arm joint 2 high	Left_Servo2_high
Data[7]	Return value: Left arm joint 2 low	Left_Servo2_Low
Data[8]	Return value: Left arm joint 3 high	Left_Servo3_high
Data[9]	Return value: Left arm joint 3 low position	Left_Servo3_Low
Data[10]	Return value: Left arm joint 4 high	Left_Servo4_high
Data[11]	Return value: Left arm joint 4 low position	Left_Servo4_Low
Data[12]	Return value: Left arm joint 5 high	Left_Servo5_high
Data[13]	Return value: Left arm joint 5 low position	Left_Servo5_Low
Data[14]	Return value: Left arm joint 6 high	Left_Servo6_high
Data[15]	Return value: Left arm joint 6 low position	Left_Servo6_Low
Data[16]	Return value: Left arm joint 7 high	Left_Servo7_High
Data[17]	Return value: Left arm joint 7 low position	Left_Servo7_Low
Data[18]	Return value: Left Arm remote sensing button status	Left_button_status (bit1: Remote sensing button bit2: Button 1 bit3: Button 2)
Data[19]	Return value: Left Arm remote sensing x value	Left_remote_sensing_x-value
Data[20]	Return value: Left Arm remote sensing y value	Left_remote_sensing_y-value
Data[21]	Return value: Left arm X-axis high	Right_Servo2_high
Data[22]	Return value: Left arm X-axis low	Right_Servo2_Low

Data Domain	Description	Data
Data[23]	Return value: High Y-axis of the left arm	Right_Servo3_high
Data[24]	Return value: Low Y-axis of the left arm	Right_Servo3_Low
Data[25]	Return value: Left arm Z-axis high	Right_Servo4_high
Data[26]	Return value: Low Z-axis of the left arm	Right_Servo4_Low
Data[27]	Return value: Left arm rx rotation high	Right_Servo5_high
Data[28]	Return value: Left arm rx rotation low position	Right_Servo5_Low
Data[29]	Return value: Left arm ry rotation high	Right_Servo6_high
Data[30]	Return value: Left arm ry rotation low position	Right_Servo6_Low
Data[31]	Return value: Left arm rz rotation high	Right_Servo7_High
Data[32]	Return value: Left arm rz rotation low position	Right_Servo7_Low
Data[33]	End frame	0XFA

Example:

Assuming Atom has successfully connected:

```
Return value of port: FE FE 13 02 00 00 00 00 00 00 00 00 00 00 00 00 14 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 FA
```

Read data from a single encoder of a single arm

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X03
Data[4]	Specify a single arm	0x01(Left)/0x02(Right)
Data[5]	Specify the servo motor	Joint_id (1-7)
Data[6]	end frame	0XFA

Example:

Port transmission: FE FE 04 03 01 01 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0X03
Data[4]	high angle of steering gear	Servo_Encoder_High
Data[5]	low angle of No.1 steering gear	Servo_Encoder_Low
Data[6]	end frame	0XFA

Example:

Assuming Atom has successfully connected:

Return value of port: FE FE 04 03 08 00 FA

Set zero point

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X04
Data[4]	Specify a single arm	0x01(Left)/0x02(Right)
Data[5]	Specify the servo motor	Joint_id (1-7)
Data[6]	end frame	0XFA

Example:

Port transmission: FE FE 04 04 01 01 FA

Return value: Data structure

Data Domain	Description	Data
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Return length	0X03
Data[3]	Return value: Command frame	0X04
Data[4]	Return value: 1	1
Data[5]	End character	0XFA

Example:

Port return: FE FE 03 04 01 FA

Set the RGB values of the LED lights

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X05
Data[4]	Specify a single arm	0x01(Left)/0x02(Right)
Data[5]	Red proportion value	R_Value
Data[6]	Green proportion value	G_Value
Data[7]	Blue proportion value	B_Value
Data[8]	end frame	0XFA

Example:

Port transmission: FE FE 06 05 01 00 FF 00 FA

Return value: Data structure

Data Domain	Description	Data
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Return length	0X03
Data[3]	Return value: Command frame	0X05
Data[4]	Return value: 1	1
Data[5]	End character	0XFA

Example:

Port return: FE FE 03 05 01 FA

Read the version number of the basic firmware

Data Domain	Description	Data
Data[0]	Identification frame	0XFE
Data[1]	Identification Frame	0XFE
Data[2]	Data Length frame	0X02
Data[3]	Instruction Frame	0X06
Data[4]	End frame	0XFA

Example:

Serial port transmission: FE FE 02 06 FA

Return value: Data structure

Data Domain	Description	Data
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Return length	0X03
Data[3]	Return value: Command frame	0X06
Data[4]	Return value: Firmware version number *10	0x0C
Data[5]	End character	0XFA

Example:

Port returns: FE FE 03 06 0C FA

Read the atom firmware version number

Data Domain	Description	Data
Data[0]	Identification frame	0XFE
Data[1]	Identification Frame	0XFE
Data[2]	Data Length frame	0X02
Data[3]	Instruction Frame	0X07
Data[4]	Instruction Frame	0x01(left)/0x02(right)
Data[5]	End frame	0XFA

Example:

Serial port transmission: FE FE 02 07 01 FA

Return value: Data structure

Data Domain	Description	Data
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Return length	0X03
Data[3]	Return value: Command frame	0X06
Data[4]	Instruction Frame	0x01(left)/0x02(right)
Data[5]	Return value: Firmware version number *10	0x0C
Data[6]	End character	0XFA

Example:

Port returns: FE FE 04 07 01 0C FA

MyController S570 Control M750 Program Example

Connect the exoskeleton and two MyArm M750 robots to the PC via USB, and run the following script.

Note: Ensure each serial port corresponds to the correct device; The three joints of the exoskeleton are rotated 90 degrees and fixed



Note 2: Before using MyArmM750, you can determine whether the rotation direction of each joint of MyArmM750 matches the rotation direction of the exoskeleton by setting mercury_list = [] to 0 in turn

For example:

```
"mercury_list = [arm_data[1], 0, 0, 0, 0, 0, 0]"
```

After running the program, if joint 2 of the MyArmM750 rotates in the opposite direction to joint 1 of the exoskeleton, it
Other joints are judged in turn

```

# Control script
import threading
import time
from pymycobot import Mercury, MyArmM
from pymycobot import ExoskeletonSocket, Exoskeleton

# Initialize Devices
# obj = Exoskeleton(port="COM15") # Exoskeleton serial port
obj = ExoskeletonSocket("192.168.4.1", 80) # Exoskeleton Wi-Fi connection
ml = MyArmM("COM5", 1000000) # Left M750 arm serial port
mr = MyArmM("COM5", 1000000) # Right M750 arm serial port

# ===== Joint Limit Configuration =====
# Fill in according to the provided parameters (Units: degrees)
# Format: (Minimum Value, Maximum Value)
JOINT_LIMITS = [
    (-163, 163), # Joint 1
    (-58, 93), # Joint 2
    (-90, 73), # Joint 3
    (-150, 148), # Joint 4
    (-89, 85), # Joint 5
    (-148, 148), # Joint 6
    (-116, 0), # Joint 7 (Gripper)
]

# Gripper Angle Definitions
GRIPPER_CLOSE = -116 # Closed
GRIPPER_OPEN = 0 # Open

def clamp_angles(angles):
    """
    Clamps a list of angles within the safe operating range.
    Angles exceeding the limits are clamped to the nearest boundary value.
    """
    clamped = []
    for i, angle in enumerate(angles):
        if i >= len(JOINT_LIMITS):
            clamped.append(angle)
            continue

        low, high = JOINT_LIMITS[i]

        if angle < low:
            print(f"Joint {i + 1} angle {angle:.1f}° is below the lower limit of {low}°; clamped to {low}°.")
            clamped.append(low)
        elif angle > high:
            print(f"Joint {i + 1} Angle {angle:.1f}° exceeds upper limit {high}°; clamped to {high}°")
            clamped.append(high)

```

```

else:
    clamped.append(angle)

return clamped

def check_joints_limits(angles):
    """
    Checks if all joint angles are within their limits.
    Returns: (is_all_valid, list_of_clamped_angles)
    """
    all_valid = True
    clamped = []

    for i, angle in enumerate(angles):
        if i >= len(JOINT_LIMITS):
            clamped.append(angle)
            continue

        low, high = JOINT_LIMITS[i]

        if angle < low or angle > high:
            print(f"Joint {i + 1} angle {angle:.1f}° exceeds limit range [{low}°, {high}°]")
            all_valid = False

        # Clamp the angle
        clamped_angle = max(min(angle, high), low)
        clamped.append(clamped_angle)

    return all_valid, clamped

def control_arm(arm):
    """
    arm: 1 for Left Arm, 2 for Right Arm
    """
    while True:
        try:
            if arm == 1:
                arm_data = obj.get_arm_data(1)
                print("1: ", arm_data)
                mc = ml

            # Calculate target angles based on exoskeleton data
            # Adjust according to the actual mapping relationship
            mercury_list = [
                arm_data[1] + 70,
                -arm_data[0],
                arm_data[3],
                arm_data[4]-90,
                -arm_data[5] + 50,

```

```

    arm_data[6],
    0
]

# ===== Gripper Control =====
if arm_data[10] == 1:
    mercury_list[6] = GRIPPER_CLOSE
    print("Gripper Closing")
elif arm_data[11] == 1:
    mercury_list[6] = GRIPPER_OPEN
    print("Gripper Opening")

# If both buttons are pressed simultaneously, skip this iteration
if len(arm_data) > 7 and arm_data[6] == 0 and arm_data[7] == 0:
    print("Both buttons pressed simultaneously; skipping this iteration.")
    time.sleep(0.01)
    continue

elif arm == 2:
    arm_data = obj.get_arm_data(2)
    print("r: ", arm_data)
    mc = mr

# Calculate target angles based on exoskeleton data
mercury_list = [
    -arm_data[1] - 50,
    -arm_data[0],
    arm_data[3]-10,
    arm_data[4]+90,
    -arm_data[5] + 50,
    arm_data[6], 0
]

if arm_data[10] == 1:
    mercury_list[6] = GRIPPER_CLOSE
    print("Gripper closing.")
elif arm_data[11] == 1:
    mercury_list[6] = GRIPPER_OPEN
    print("Gripper opening.")

# If both buttons are pressed simultaneously, skip this iteration
if len(arm_data) > 7 and arm_data[6] == 0 and arm_data[7] == 0:
    print("Both buttons pressed simultaneously; skipping this iteration.")
    time.sleep(0.01)
    continue
else:
    raise ValueError("Invalid arm: 1 for Left Arm, 2 for Right Arm.")

```

```

# ===== Limit Check =====
all_valid, clamped_list = check_joints_limits(mercury_list)

# Always send the corrected angles (to ensure limits are not exceeded)
mc.set_joints_angle(clamped_list, 10)

if all_valid:
    print(f"Command sent: {[round(a, 1) for a in clamped_list]}")
else:
    print(f"Sent after correction: {[round(a, 1) for a in clamped_list]}")

time.sleep(0.01)

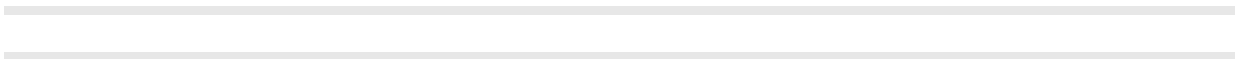
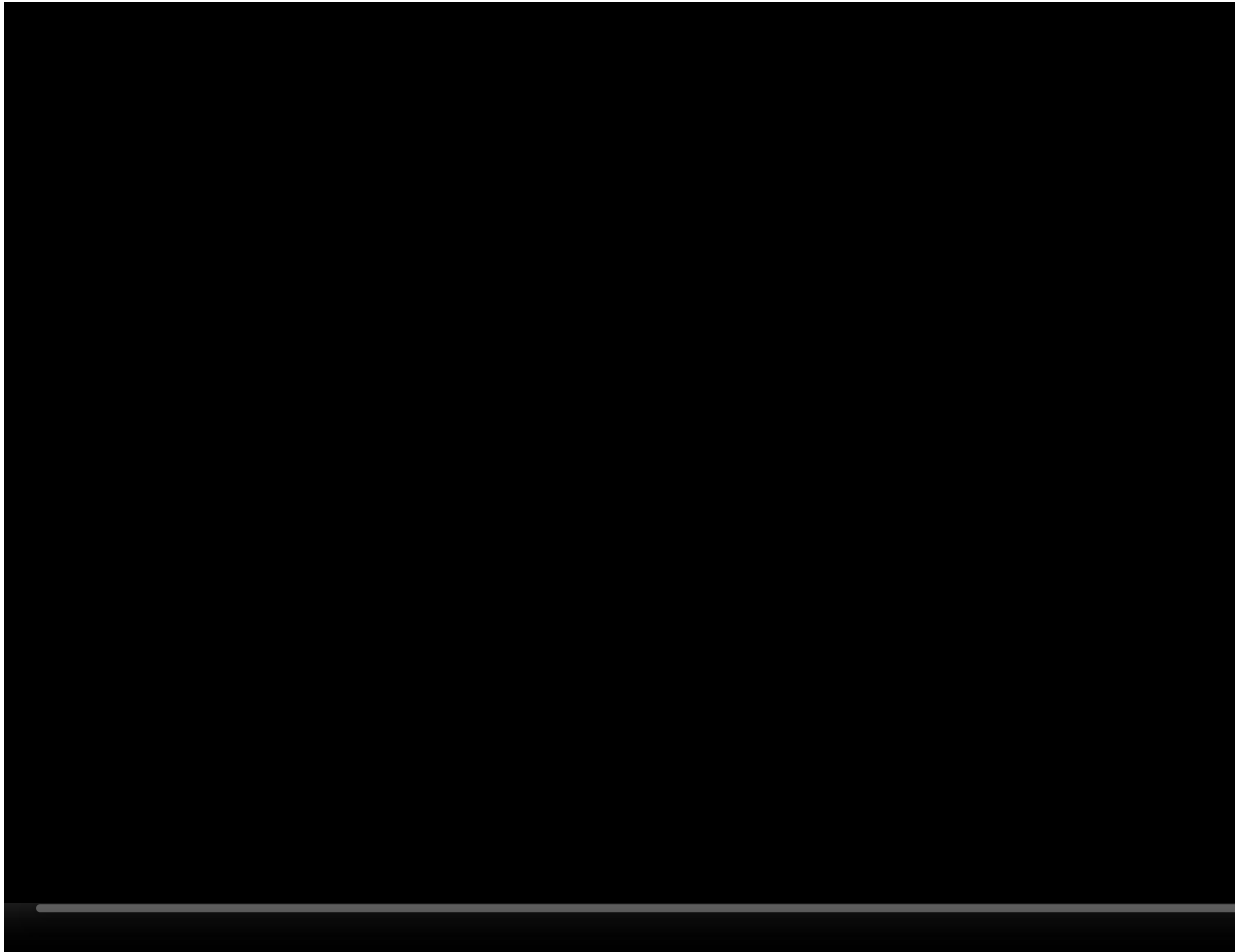
except Exception as e:
    print(f"Control loop exception: {e}")
    time.sleep(0.1)

# Start control threads for the left and right arms
threading.Thread(target=control_arm, args=(1,), daemon=True).start() # Left arm
threading.Thread(target=control_arm, args=(2,), daemon=True).start() # Right arm

# Keep the main thread running
try:
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("Program terminated.")

```


Once the program runs successfully, the MyArm M750 can be controlled with the exoskeleton



MyController S570 Control AGV+280M5 program case

Connect the exoskeleton to the Jetson Nano system on the AGV via USB and save and run the following script file.

Note: Make sure each serial number corresponds to the correct device

Recommended to start this case: atom version 6.5; The pymycobot version 3.6.8

```

# The configuration file is mercury_ros_api.py
import time
import threading
import time
import sys
import subprocess
import os
import glob
import rospy
import actionlib
import tf2_ros
import tf_conversions

from std_msgs.msg import Float32
from actionlib_msgs.msg import *
from actionlib_msgs.msg import GoalID
from actionlib_msgs.msg import GoalStatusArray
from tf.transformations import quaternion_from_euler
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
from geometry_msgs.msg import Point
from geometry_msgs.msg import Point
from geometry_msgs.msg import Twist
from geometry_msgs.msg import PoseWithCovarianceStamped
from geometry_msgs.msg import PoseStamped, PoseArray

# MapNavigation
class MapNavigation:
    def __init__(self):
        # Initialize the ROS node
        rospy.init_node('map_navigation', anonymous=False)

        # ros publisher
        self.pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
        self.pub_setpose = rospy.Publisher('/initialpose', PoseWithCovarianceStamped, queue_size=10)
        self.pub_cancel = rospy.Publisher('/move_base/cancel', GoalID, queue_size=10)
        self.pub_tempPose = rospy.Publisher('move_base_simple/goal_temp', PoseStamped, queue_size=1)
        self.goal_pub = rospy.Publisher('move_base_simple/goal', PoseStamped, queue_size=1)

        # ros subscriber
        #self.voltage_subscriber = rospy.Subscriber("/PowerVoltage", Float32, self.voltage_callback) #Create a battery-vol
        self.initialpose_subscriber = rospy.Subscriber('/initialpose', PoseWithCovarianceStamped, self.initial_pose_callback)
        self.goal_subscriber = rospy.Subscriber('move_base/status', GoalStatusArray, self.goalCallback)

        self.RobotVersion = 1.0
        self.SystemVersion = 1.0
        self.is_running = False
        self.thread = None
        self.LastError=None
        self.directory = "~/mercury_x1_ros/src/turn_on_mercury_robot/map"

```

```

self.pose = PoseStamped()
self.pose_array = PoseArray()
self.pose_stamped = PoseStamped()

# getPosition
self.tf_buffer = tf2_ros.Buffer()
self.tf_listener = tf2_ros.TransformListener(self.tf_buffer)

self.point_count = 0
self.flag_HomePosition = True
self.flag_Parking = True
self.flag_Charge = True
self.flag_Load = True
self.flag_Unload = True

def getRobotVersion(self):
    return self.RobotVersion

def getSystemVersion(self):
    return self.SystemVersion

#voltage Callback
def voltage_callback(self, msg):
    return msg.data

def initial_pose_callback(self,msg):
    #rospy.loginfo("Received Initial Pose:\n%s", msg)
    return msg

def goalCntCallback(self,goal_msg):

    self.pose_stamped.header = goal_msg.header
    self.pose_stamped.pose = goal_msg.pose

    # self.pose_array.header.append(pose_stamped.pose)
    self.pose_array.poses.append(self.pose_stamped.pose)

def setpoint(self,identifier,xGoal, yGoal, orientation_z, orientation_w):
    self.pose = PoseStamped()
    self.pose.header.seq = self.point_count
    self.pose.header.stamp = rospy.Time.now()
    self.pose.header.frame_id = 'map'
    self.pose.pose.position.x = xGoal
    self.pose.pose.position.y = yGoal
    self.pose.pose.position.z = 0.0
    self.pose.pose.orientation.x = 0.0
    self.pose.pose.orientation.y = 0.0
    self.pose.pose.orientation.z = orientation_z
    self.pose.pose.orientation.w = orientation_w

```

```

    rospy.sleep(1)

    self.pub_tempPose.publish(self.pose)

    self.point_count += 1

    rospy.loginfo(' {} 对应 {} 号点位 pose: {}'.format(identifier,self.point_count, self.pose))

    return self.point_count

def clearPosition(self):
    self.pose_array.poses = []

    self.point_count = 0

    self.flag_HomePosition = True

    self.flag_Parking = True

    self.flag_Charge = True

    self.flag_Load = True

    self.flag_Unload = True

    rospy.loginfo('clearPosition')

def getPose(self,identifier,point_index):
    if point_index > 0:
        if point_index <= len(self.pose_array.poses):
            curGoalIdx_ = point_index % len(self.pose_array.poses)

            pose = self.pose_array.poses[curGoalIdx_-1]

            rospy.loginfo('{} Retrieving {} point pose: {}'.format(identifier,point_index, pose))

def goTopoint(self,identifier,point_index):
    if point_index > 0:
        if point_index <= len(self.pose_array.poses):
            curGoalIdx_ = point_index % len(self.pose_array.poses)

            goal = PoseStamped()

            goal.header = self.pose_array.header

            goal.pose = self.pose_array.poses[curGoalIdx_-1]

            self.goal_pub.publish(goal)

            rospy.loginfo('Going to {} point:{}'.format(identifier, goal.pose))

def setHomePosition(self,xGoal, yGoal, orientation_z, orientation_w):
    global point_HomePositionCount

    if self.flag_HomePosition == True:
        self.flag_HomePosition = False

        # Returns the navigation point count

        point_HomePositionCount = self.setpoint("HomePosition",xGoal, yGoal, orientation_z, orientation_w)
    else:
        print("The navigation point has been set. clearPosition() is required to change the navigation.")

def setParking(self,xGoal, yGoal, orientation_z, orientation_w):
    global point_ParkingCount

    if self.flag_Parking == True:
        self.flag_Parking = False

        # Returns the navigation point count

        point_ParkingCount = self.setpoint("Parking",xGoal, yGoal, orientation_z, orientation_w)
    else:
        print("The navigation point has been set. clearPosition() is required to change the navigation.")

```

```

def setCharge(self,xGoal, yGoal, orientation_z, orientation_w):
    global point_ChargeCount
    if self.flag_Charge == True:
        self.flag_Charge = False
        # Returns the navigation point count
        point_ChargeCount = self.setpoint("Charge",xGoal, yGoal, orientation_z, orientation_w)
    else:
        print("The navigation point has been set. clearPosition() is required to change the navigation.")

def setLoad(self,xGoal, yGoal, orientation_z, orientation_w):
    global point_LoadCount
    if self.flag_Load == True:
        self.flag_Load = False
        # Returns the navigation point count
        point_LoadCount = self.setpoint("Load",xGoal, yGoal, orientation_z, orientation_w)
    else:
        print("The navigation point has been set. clearPosition() is required to change the navigation.")

def setUnload(self,xGoal, yGoal, orientation_z, orientation_w):
    global point_UnloadCount
    if self.flag_Unload == True:
        self.flag_Unload = False
        # Returns the navigation point count
        point_UnloadCount = self.setpoint("Unload",xGoal, yGoal, orientation_z, orientation_w)
    else:
        print("The navigation point has been set. clearPosition() is required to change the navigation.")

# HomePosition
def getHomePosition(self):
    self.getPose("HomePosition",point_HomePositionCount)

def goToHomePosition(self):
    self.goTopoint("HomePosition",point_HomePositionCount)

# Parking
def getParking(self):
    self.getPose("Parking",point_ParkingCount)

def goToParking(self):
    self.goTopoint("Parking",point_ParkingCount)

# Charge
def getCharge(self):
    self.getPose("Charge",point_ChargeCount)

def goToCharge(self):
    self.goTopoint("Charge",point_ChargeCount)

# Load

```

```

def getLoad(self):
    self.getPose("Load",point_LoadCount)

def goToLoad(self):
    self.goTopoint("Load",point_LoadCount)

# Unload
def getUnload(self):
    self.getPose("Unload",point_UnloadCount)

def goToUnload(self):
    self.goTopoint("Unload",point_UnloadCount)

# init robot pose AMCL
def set_pose(self, xGoal, yGoal, orientation_z, orientation_w,covariance):
    pose = PoseWithCovarianceStamped()
    pose.header.seq = 0
    pose.header.stamp.secs = 0
    pose.header.stamp.nsecs = 0
    pose.header.frame_id = 'map'
    pose.pose.pose.position.x = xGoal
    pose.pose.pose.position.y = yGoal
    pose.pose.pose.position.z = 0.0
    q = quaternion_from_euler(0, 0, 1.57)
    pose.pose.pose.orientation.x = 0.0
    pose.pose.pose.orientation.y = 0.0
    pose.pose.pose.orientation.z = orientation_z
    pose.pose.pose.orientation.w = orientation_w
    pose.pose.covariance = [0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.25, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0,0.0, 0.0, 0.0, covariance]
    rospy.sleep(1)
    self.pub_setpose.publish(pose)
    rospy.loginfo('Published robot pose: %s' % pose)

# move_base
def moveToGoal(self, xGoal, yGoal, orientation_z, orientation_w):
    ac = actionlib.SimpleActionClient("move_base", MoveBaseAction)
    while(not ac.wait_for_server(rospy.Duration.from_sec(5.0))):

        sys.exit(0)

    goal = MoveBaseGoal()
    goal.target_pose.header.frame_id = "map"
    goal.target_pose.header.stamp = rospy.Time.now()
    goal.target_pose.pose.position = Point(xGoal, yGoal, 0)
    goal.target_pose.pose.orientation.x = 0.0
    goal.target_pose.pose.orientation.y = 0.0
    goal.target_pose.pose.orientation.z = orientation_z
    goal.target_pose.pose.orientation.w = orientation_w

```

```

    rospy.loginfo("Sending goal location ...")
    ac.send_goal(goal)

    ac.wait_for_result(rospy.Duration(60))

    if(ac.get_state() == GoalStatus.SUCCEEDED):
        rospy.loginfo("You have reached the destination")
        return True
    else:
        rospy.loginfo("The robot failed to reach the destination")
        return False

def shutdown(self):
    rospy.loginfo("Quit program")
    rospy.sleep()

# speed command
def pub_vel(self, x, y, theta):
    twist = Twist()
    twist.linear.x = x
    twist.linear.y = y
    twist.linear.z = 0
    twist.angular.x = 0
    twist.angular.y = 0
    twist.angular.z = theta
    self.pub.publish(twist)

def vel_control(self, direction=[0,0,0], speed=0.0, control_time=0.0):
    """
    Function to control velocity.

    Parameters:
        - direction (list): An array containing three elements representing the direction of motion.
        - speed (float): Speed value.
        - time (float): Time value.

    Raises:
        - ValueError: If the direction is not a list containing three elements.

    Usage:
        >>> vel_control([1, 0, 0], 0.2, 5)
    """
    duration, vel = control_time, speed
    start_time = time.time()
    # Motion control
    if isinstance(direction, list) and len(direction) == 3:
        while (time.time() - start_time) < duration:
            self.pub_vel(direction[0] * abs(vel), direction[1] * abs(vel), direction[2] * abs(vel))
        self.pub_vel(0,0,0)

```



```

    else:
        print("Direction should be a list containing three elements")

def turnRight(self,speed,control_time):
    self.vel_control([0, 0, -1], speed, control_time)

def turnLeft(self,speed,control_time):
    self.vel_control([0, 0, 1], speed, control_time)

def goStraight(self,speed,control_time):
    self.vel_control([1, 0, 0], speed, control_time)

def goBack(self,speed,control_time):
    self.vel_control([-1, 0, 0], speed, control_time)

def stop(self):
    self.pub_vel(0,0,0)

def startMapping(self):
    try:
        launch_command = "roslaunch turn_on_tringai_robot mapping.launch"
        subprocess.run(['gnome-terminal', '-e', f"bash -c '{launch_command}; exec $SHELL'"])
    except subprocess.CalledProcessError as e :
        self.LastError = sys.exc_info()
        print(e)

def stopMapping(self):
    try:
        # Kill the corresponding process
        close_command = "ps -ef | grep -E " + "mapping.launch" + \
            " | grep -v 'grep' | awk '{print $2}' | xargs kill -2"
        subprocess.run(close_command, shell=True)
    except subprocess.CalledProcessError as e :
        self.LastError = sys.exc_info()
        print(e)

def deleteMap(self):
    files_to_delete = glob.glob(os.path.join(self.directory, "map.pgm")) + \
        glob.glob(os.path.join(self.directory, "map.yaml"))
    try:
        for file_to_delete in files_to_delete:
            os.remove(file_to_delete)
            print(f"Deleted file: {file_to_delete}")
    except Exception as e:
        self.LastError = sys.exc_info()
        print(e)

def agvOn(self):
    try:
        # Start lidar and odometer communication

```

```

        launch_command = "roslaunch turn_on_mercury_robot turn_on_mercury_robot.launch"
        subprocess.run(['gnome-terminal', '-e', f"bash -c '{launch_command}'; exec $SHELL'"])
    except Exception as e:
        self.LastError = sys.exc_info()
        print(e)

def agvOff(self):
    try:
        # Kill the corresponding process
        close_command = "ps -ef | grep -E " + "turn_on_mercury_robot.launch" + \
            " | grep -v 'grep' | awk '{print $2}' | xargs kill -2"
        subprocess.run(close_command, shell=True)
    except Exception as e:
        self.LastError = sys.exc_info()
        print(e)

def isAgvOn(self):
    try:
        # Check whether there is a corresponding process
        process_check_command = "ps -ef | grep -E 'turn_on_tringai_robot.launch' | grep -v 'grep'"
        result = subprocess.run(process_check_command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        if len(result.stdout) > 0:
            return True
        else:
            return False
    except Exception as e:
        self.LastError = sys.exc_info()
        print(e)

def batteryState(self):
    voltage_data = self.voltage_callback(None)
    return voltage_data

def initPosition(self, x_goal, y_goal, orientation_z, orientation_w, covariance):
    try:
        self.set_pose(x_goal, y_goal, orientation_z, orientation_w, covariance)
    except Exception as e:
        self.LastError = sys.exc_info()

def getPosition(self):
    # Position = self.initial_pose_callback(None)
    # return Position
    try:
        transform = self.tf_buffer.lookup_transform("map", "base_up", rospy.Time(0), rospy.Duration(1.0))
        translation = transform.transform.translation
        rotation = transform.transform.rotation

        x = translation.x
        y = translation.y

```

```

        euler = tf_conversions.transformations.euler_from_quaternion([rotation.x, rotation.y, rotation.z, rotation.w])

        tw = euler[2] # Yaw

        rospy.loginfo("X position{}".format(x))
        rospy.loginfo("Y position{}".format(y))
        rospy.loginfo("TW orientation{}".format(tw))

        return x,y,tw

    except (tf2_ros.LookupException, tf2_ros.ConnectivityException, tf2_ros.ExtrapolationException) as e:
        rospy.logerr("TF transformation query failed:{}".format(e))

def goToPosition(self,goal_x, goal_y, orientation_z, orientation_w):
    flag_feed_goalReached = self.moveToGoal(goal_x, goal_y , orientation_z, orientation_w)
    return flag_feed_goalReached

def pause(self):
    if not self.is_running:
        self.is_running = True
        self.thread = threading.Thread(target=self.stop_navigation, daemon=True)
        self.thread.start()

def stop_navigation(self):
    while self.is_running:
        goal_id = GoalID()
        self.pub_cancel.publish(goal_id)

        twist = Twist()
        twist.linear.x = 0.0
        twist.linear.y = 0.0
        twist.linear.z = 0.0
        twist.angular.x = 0.0
        twist.angular.y = 0.0
        twist.angular.z = 0.0
        self.pub.publish(twist)
        rospy.sleep(0.1)

def unpause(self):
    self.is_running = False
    if self.thread is not None:
        self.thread.join()

def getLastError(self):
    if self.LastError is not None:
        exc_type, exc_value, exc_traceback = self.LastError
        print(f"Exception Type: {exc_type}")
        print(f"Exception Value: {exc_value}")
        print("Exception Traceback:")
        traceback_str = "\n".join(self.LastError.format_tb(exc_traceback))

```

```

        print(traceback_str)
    else:
        print("No recent errors.")
    return self.LastError

def startNavigation(self):
    try:
        # Start lidar and odometer communication
        launch_command = "roslaunch myagv_navigation navigation_active.launch"
        subprocess.run(['gnome-terminal', '-e', f"bash -c '{launch_command}; exec $SHELL'"])
    except Exception as e:
        self.LastError = sys.exc_info()
        print(e)

if __name__ == '__main__':
    #init navigation
    map_navigation = MapNavigation()

    print(map_navigation.getRobotVersion())
    print(map_navigation.getSystemVersion())

    print(map_navigation.isAgvOn())

    # map_navigation.getLastError()

    # map_navigation.initPosition(1.1, -0.22, 0.7, 0.99,0.55)

    # print(map_navigation.getPosition())

    # map_navigation.getLastError()

    # map_navigation.turnRight(0.2,5)

    # map_navigation.goStraight(0.2,5)

```

```

import threading
import time

import serial
from pymycobot import *
from pymycobot import Exoskeleton
from mercury_ros_api import MapNavigation
from pymycobot import MyCobot280
from pymycobot.utils import get_port_list

mr = MyCobot280("/dev/ttyACM3",115200)

obj = Exoskeleton(port="/dev/ttyACM4")

map_navigation = MapNavigation()
mr.power_on()

def Pump_testing_open():
    import RPi.GPIO as GPIO
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(2, GPIO.OUT)
    GPIO.setup(3, GPIO.OUT)

    # open
    GPIO.output(3, GPIO.LOW)
    GPIO.output(2, GPIO.HIGH)

def Pump_testing_close():
    import RPi.GPIO as GPIO
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(2, GPIO.OUT)
    GPIO.setup(3, GPIO.OUT)
    # close
    GPIO.output(3, GPIO.HIGH)
    GPIO.output(2, GPIO.LOW)
    time.sleep(0.05)
    GPIO.output(2, GPIO.HIGH)

def move(x, y):
    if y < 50:
        map_navigation.pub_vel(0.25, 0, 0)
    elif y > 200:
        map_navigation.pub_vel(-0.25, 0, 0)
    elif x < 50:
        map_navigation.pub_vel(0, 0, -0.5)

```

```

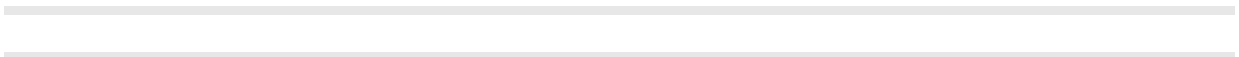
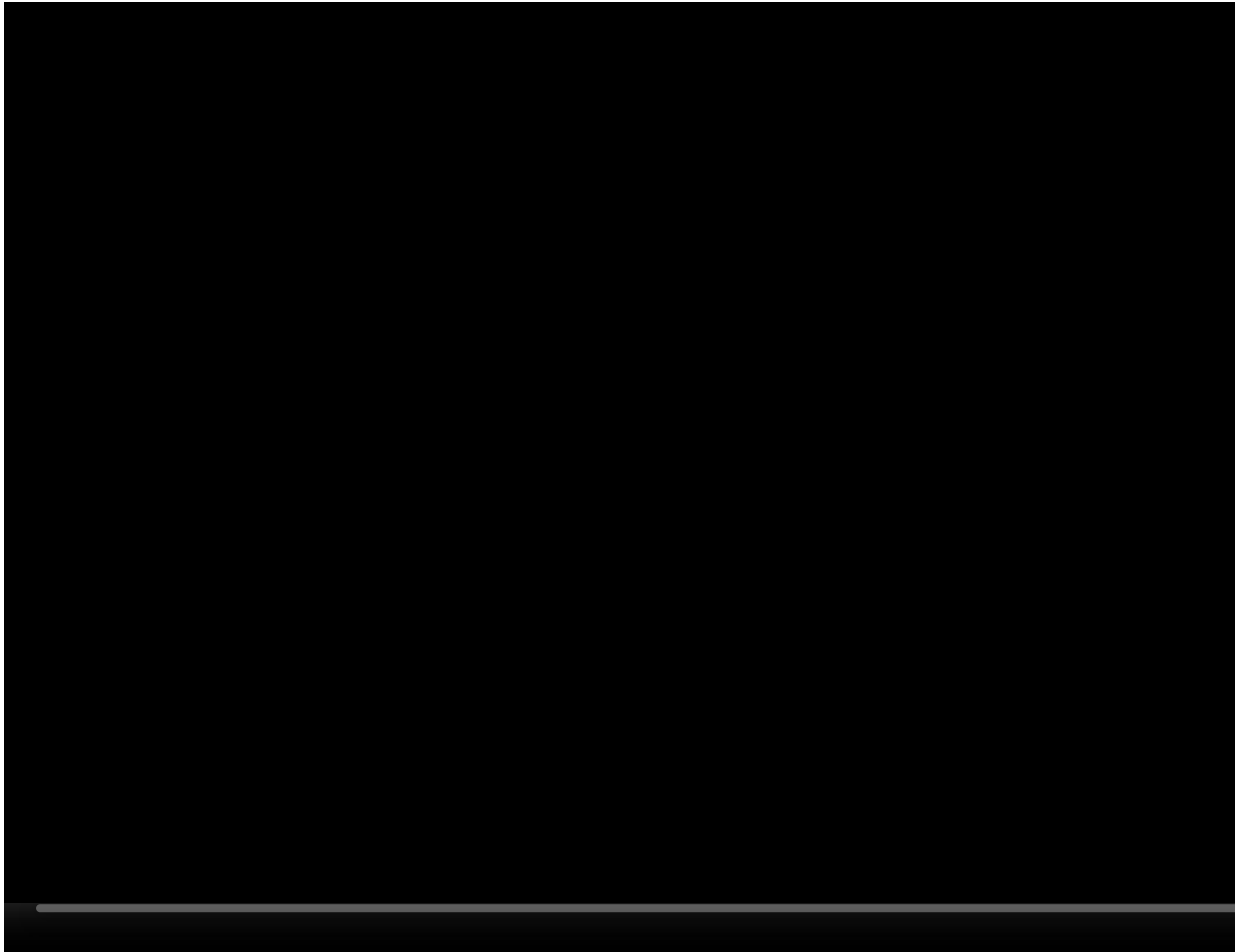
elif x > 200:
    map_navigation.pub_vel(0, 0, 0.5)
else:
    map_navigation.pub_vel(0, 0, 0)
    #map_navigation.stop()

# 0 left arm, 1 right arm
def control_arm(arm):
    while True:
        if arm == 0:
            arm_data = obj.get_data(0)
            x, y = arm_data[9], arm_data[10]
            print("l: ", arm_data)
            # mc = ml
            mc = mr
        elif arm == 1:
            arm_data = obj.get_data(1)
            x, y = arm_data[9], arm_data[10]
            print("r: ", arm_data)
            mc = mr
            threading.Thread(target=move, args=(x, y,)).start()
        else:
            raise ValueError("error arm")
    mercury_list = [
        -arm_data[1]-40, arm_data[0] - 90, -arm_data[3], arm_data[5],
        arm_data[4], arm_data[6]
    ]
    #mercury_list = [
    #    0, 0, 0, 0, 0,
    #    arm_data[6]
    #]
    if arm_data[7] == 0:
        Pump_testing_open()
    elif arm_data[8] == 0:
        Pump_testing_close()
    mc.send_angles(mercury_list, 100)

# The right arm of the exoskeleton controls AGV+280M5
threading.Thread(target=control_arm, args=(1,)).start()

```

The AGV+280M5 can be controlled with the exoskeleton after the program has been successfully run



MyController S570 Control X1 program case

Connect the exoskeleton to Mercury via USB and save the following script file. **Note: Make sure each serial number corresponds to the correct device**

1.Use example

Operation tutorial:

Turn on the side switch before inserting Type-C



Turn on wifi:

Press button A, IP is displayed for use, wifi account number: elephant, wifi password elephant

Turn off wifi: press button C

Turn on Bluetooth:

Press button B, display BT can be used, Bluetooth name BLE

Turn off Bluetooth: press button C



1.1 Serial communication

```
# example
from pymycobot import Exoskeleton

# Connect the serial port
obj = Exoskeleton(port="COM5")

# Get dual-arm data
all_data = obj.get_all_data()
print(all_data)

# Get left arm data
left_data = obj.get_arm_data(1)
print(left_data)

# Get right arm data
right_data = obj.get_arm_data(2)
print(right_data)

# Obtain single arm and single joint data
joint_data = obj.get_joint_data(1, 1)
print(joint_data)

# Set the current position to zero on right arm J7
obj.set_zero(2, 7)

# Set the left arm atom screen color
obj.set_color(1, 0, 255, 0)
```

1.2 socket communication

```
# example
from pymycobot import ExoskeletonSocket

# Connection server
obj = ExoskeletonSocket()

# Get dual-arm data
all_data = obj.get_all_data()
print(all_data)

# Get left arm data
left_data = obj.get_arm_data(1)
print(left_data)

# Get right arm data
right_data = obj.get_arm_data(2)
print(right_data)

# Obtain single arm and single joint data
joint_data = obj.get_joint_data(1, 1)
print(joint_data)

# Set the current position to zero on right arm J7
obj.set_zero(2, 7)

# Set the left arm atom screen color
obj.set_color(1, 0, 255, 0)
```

2 Remote control robot arm case

mercury X1(7-axis)

```
# The version of pymycobot should be 4.0.0 or above
# This is a control file named MercuryControl.py
import threading
from pymycobot import Mercury
from pymycobot import Exoskeleton

obj = Exoskeleton(port="/dev/ttyACM4")
ml = Mercury("/dev/left_arm")
mr = Mercury("/dev/right_arm")

ml.set_vr_mode(1)
mr.set_vr_mode(1)
# 设置双臂为速度融合模式
ml.set_movement_type(3)
mr.set_movement_type(3)
# 设置夹爪运行模式
ml.set_gripper_mode(0)
mr.set_gripper_mode(0)

# 1 左臂, 2 右臂
def control_arm(arm):
    while True:
        if arm == 1:
            arm_data = obj.get_arm_data(1)
            print("l: ", arm_data)
            mc = ml
        elif arm == 2:
            arm_data = obj.get_arm_data(2)
            print("r: ", arm_data)
            mc = mr
        else:
            raise ValueError("error arm")
    # 由于外骨骼各关节转向、零点与部分机械臂构型不同, 在此设置映射关系(根据各关节实际控制转向、位置设置)
    mercury_list = [
        arm_data[0], -arm_data[1], arm_data[2], -arm_data[3], arm_data[4],
        135 + arm_data[5], arm_data[6]
    ]
    if arm_data[9] == 0:
        mc.set_gripper_state(1, 100)
    elif arm_data[10] == 0:
        mc.set_gripper_state(0, 100)
    mc.send_angles(mercury_list, 6, _async=True)
```

```
# 左臂
threading.Thread(target=control_arm, args=(1, )).start()

# 右臂
threading.Thread(target=control_arm, args=(2, )).start()
```

Note: Keep the two files in the same path

After successful operation of the program, Mercury X1 can be controlled with the exoskeleton



Exoskeleton control Mercury X1 specific case

Case 1: Exoskeleton controls Mercury X1+ three-finger dexterous hand to drill holes with electric drill

Source code

```

# This is a control file named MercuryControl.py
import threading
from pymycobot import Mercury
import time
from pymycobot import Exoskeleton
import os

os.system("sudo chmod 777 /dev/ttyACM*")

obj = Exoskeleton(port="/dev/ttyACM4")

ml = Mercury("/dev/left_arm")
mr = Mercury("/dev/right_arm")

l_control_mode = 1
r_control_mode = 1

l_last_mode = 0
r_last_mode = 0

ml.set_movement_type(3)
mr.set_movement_type(3)

ml.set_vr_mode(1)
mr.set_vr_mode(1)

ml.set_hand_gripper_angle(14, 4, 90)
ml.set_hand_gripper_angle(14, 5, 20)
ml.set_hand_gripper_angle(14, 6, 30)
ml.set_hand_gripper_angle(14, 1, 50)
ml.set_hand_gripper_angle(14, 3, 90)

def jointlimit(angles):
    max = [165, 120, 175, 0, 175, 180, 175]
    min = [-165, -50, -175, -175, -175, 60, -175]
    for i in range(7):
        if angles[i] > max[i]:
            angles[i] = max[i]
        if angles[i] < min[i]:
            angles[i] = min[i]

def gripper_control_open_2(mc):
    mc.set_hand_gripper_angle(14, 2, 0)

def gripper_control_close_2(mc):
    mc.set_hand_gripper_angle(14, 2, 90)

```

```

# 1 left, 2 right
def control_arm(arm):
    global l_control_mode, l_last_mode, r_control_mode, r_last_mode
    st = 0
    while True:
        try:
            if arm == 1:
                arm_data = obj.get_arm_data(1)
                print("l: ", arm_data)
                mc = ml
            elif arm == 2:
                arm_data = obj.get_arm_data(2)
                print("r: ", arm_data)
                mc = mr
            else:
                raise ValueError("error arm")
            time_err = 1000 * (time.time() - st)
            st = time.time()
            if arm == 1:
                mercury_list = [
                    arm_data[0] - 10, -arm_data[1], arm_data[2], -arm_data[3], 105, 38, 0
                ]
                jointlimit(mercury_list)
                if arm_data[7] == 0 and l_last_mode == 0:
                    print(6)
                    l_last_mode = 1
                    l_control_mode += 1
                    if l_control_mode > 3:
                        l_control_mode = 1

                    if l_control_mode == 1:
                        obj.set_color(arm, 0, 255, 0)
                    elif l_control_mode == 2:
                        obj.set_color(arm, 0, 0, 255)
                    else:
                        obj.set_color(arm, 255, 0, 0)

                if arm_data[7] == 1:
                    l_last_mode = 0

                if l_control_mode == 1:
                    TI = 10
                elif l_control_mode == 2:
                    TI = 5
                else:
                    TI = 3
            if arm_data[9] == 0:
                threading.Thread(target=gripper_control_close_2, args=(mc,)).start()

```

```

elif arm_data[10] == 0:
    threading.Thread(target=gripper_control_open_2, args=(mc,)).start()
if arm_data[9] == 0 and arm_data[10] == 0:
    time.sleep(0.01)
    continue

else:
    mercury_list = [
        arm_data[0], -arm_data[1], -0.524, -41.862, -90.686, 101.273, 25.257
    ]
    jointlimit(mercury_list)
    if arm_data[7] == 0 and r_last_mode == 0:
        print(6)
        r_last_mode = 1
        r_control_mode += 1
        if r_control_mode > 3:
            r_control_mode = 1
        if r_control_mode == 1:
            obj.set_color(arm, 0, 255, 0)
        elif r_control_mode == 2:
            obj.set_color(arm, 0, 0, 255)
        else:
            obj.set_color(arm, 255, 0, 0)

    if arm_data[7] == 1:
        r_last_mode = 0
    if r_control_mode == 1:
        TI = 10
    elif r_control_mode == 2:
        TI = 5
    else:
        TI = 3

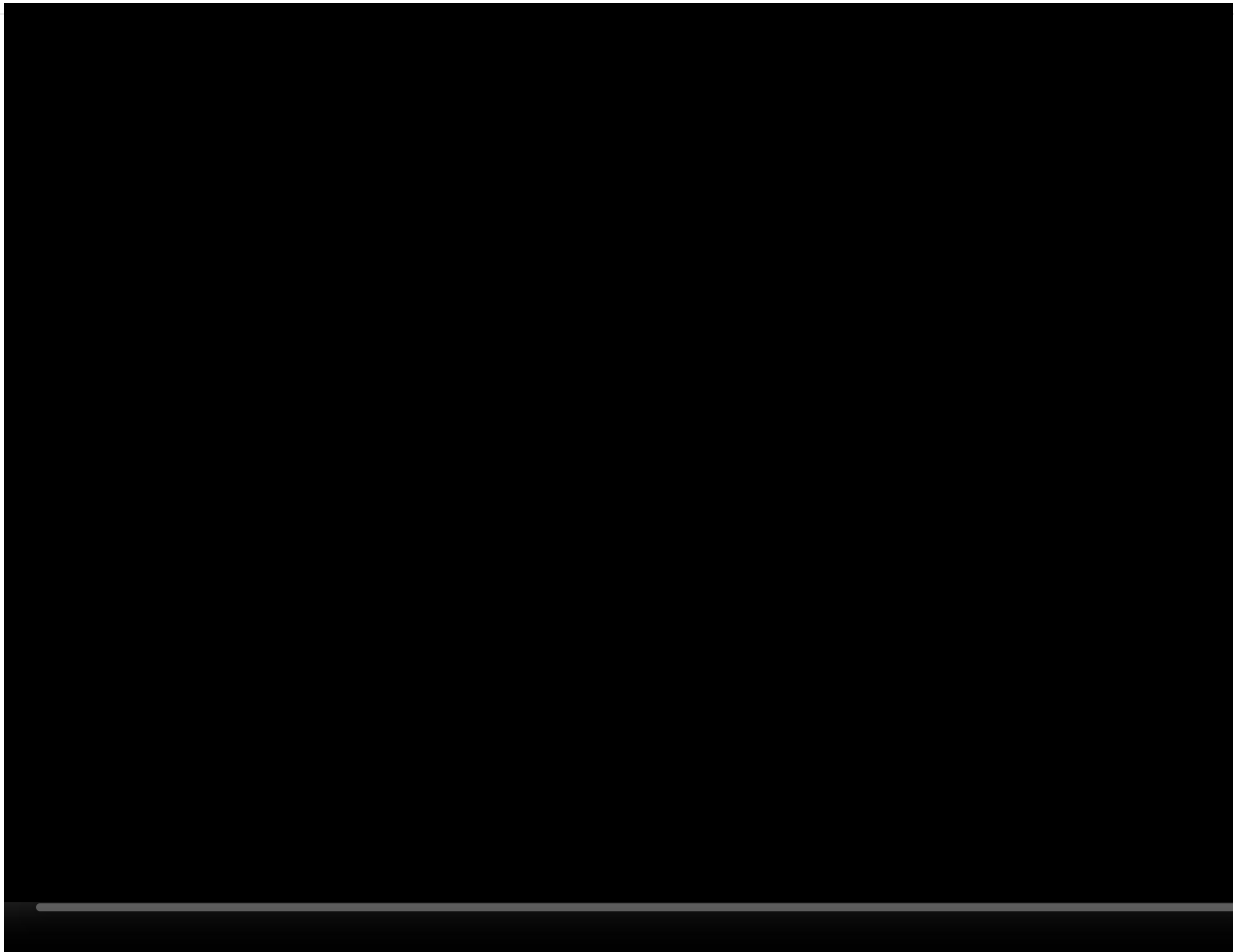
    mc.send_angles(mercury_list, TI, _async=True)
    time.sleep(0.01)

except Exception as e:
    print(e)
    pass

threading.Thread(target=control_arm, args=(1,)).start()
threading.Thread(target=control_arm, args=(2,)).start()

```

Case video



Case 2: Exoskeleton control Mercury X1+ force control claw+ the joystick controls the movement of Mercury

Note: When Mercury's voltage drops below 21v, movement cannot be controlled. Before use, please check whether the mapping relationship between the exoskeleton and Mercury's arms is correct

Source code

```

import os
import threading
import time
import serial
from pymycobot import *
from pymycobot.utils import get_port_list

```

```

os.system("sudo chmod 777 /dev/ttyACM*")
move = ChassisControl(port="/dev/wheeltec_controller")
obj = Exoskeleton(port="/dev/ttyACM4")
ml = Mercury("/dev/left_arm")
mr = Mercury("/dev/right_arm")

```

```

# set left and right arm mode
ml.set_movement_type(3)
mr.set_movement_type(3)
ml.set_vr_mode(1)
mr.set_vr_mode(1)
ml.set_pro_gripper_speed(14, 100)
mr.set_pro_gripper_speed(14, 100)

```

```

# set global value
last_move_cmd = None
last_rotation_cmd = None
move_lock = threading.Lock()

```

```

# control gripper
def control_gripper(mc, mode):
    if mode == 1:
        mc.set_pro_gripper_angle(14, 100)
    elif mode == 2:
        mc.set_pro_gripper_angle(14, 10)
    else:
        raise ValueError("error arm")

```

```

moving = False
moving_thread = None

```

```

def keep_moving(direction):
    global moving
    while moving:
        if direction == 'forward':
            move.go_straight(0.18)
        elif direction == 'backward':
            move.go_back(-0.15)
        elif direction == 'left':
            move.turn_left(0.3)

```

```

elif direction == 'right':
    move.turn_right(-0.3)
    time.sleep(0.05) # set the sending frequency to 20Hz

def control_move(x, y):
    global last_move_cmd, moving, moving_thread
    with move_lock:
        if y < 50:
            if last_move_cmd != 'forward':
                moving = False
                if moving_thread: moving_thread.join()
                moving = True
                moving_thread = threading.Thread(target=keep_moving, args=('forward',))
                moving_thread.start()
                last_move_cmd = 'forward'

        elif y > 200:
            if last_move_cmd != 'backward':
                moving = False
                if moving_thread: moving_thread.join()
                moving = True
                moving_thread = threading.Thread(target=keep_moving, args=('backward',))
                moving_thread.start()
                last_move_cmd = 'backward'

        elif x > 200:
            if last_move_cmd != 'left':
                moving = False
                if moving_thread: moving_thread.join()
                moving = True
                moving_thread = threading.Thread(target=keep_moving, args=('left',))
                moving_thread.start()
                last_move_cmd = 'left'

        elif x < 50:
            if last_move_cmd != 'right':
                moving = False
                if moving_thread: moving_thread.join()
                moving = True
                moving_thread = threading.Thread(target=keep_moving, args=('right',))
                moving_thread.start()
                last_move_cmd = 'right'

        elif 50 < y < 200 and 50 < x < 200:
            if last_move_cmd != 'stop':
                moving = False
                move.stop()
                last_move_cmd = 'stop'

```

```

def stop():
    global last_move_cmd
    with move_lock:
        if last_move_cmd != 'stop':
            move.stop()
            last_move_cmd = 'stop'

def control_arm(arm):
    global last_rotation_cmd
    while True:
        if arm == 1:
            arm_data = obj.get_arm_data(2)
            mc = mr
            mercury_list = [
                arm_data[0], -arm_data[1] + 10, arm_data[2],
                -arm_data[3], arm_data[4], 135 + arm_data[5], arm_data[6] - 20
            ]

            elif arm == 2:
                arm_data = obj.get_arm_data(1)
                x, y = arm_data[11], arm_data[12]
                mc = ml
                threading.Thread(target=control_move, args=(x, y,)).start()
                mercury_list = [
                    arm_data[0], -arm_data[1] + 10, arm_data[2],
                    -arm_data[3], arm_data[4], 135 + arm_data[5], arm_data[6] + 20
                ]

            else:
                raise ValueError("error arm")

            red_btn = arm_data[9]
            blue_btn = arm_data[10]
            atom_btn = arm_data[7]
            if red_btn == 0:
                threading.Thread(target=control_gripper, args=(mc, 1,)).start()

            elif blue_btn == 0:
                threading.Thread(target=control_gripper, args=(mc, 2,)).start()

            if red_btn == 0 and blue_btn == 0:
                time.sleep(0.01)
                continue

            if atom_btn == 0:
                exit()

            mc.send_angles(mercury_list, 6, _async=True)

```

```
        time.sleep(0.01)

def main():
    threading.Thread(target=control_arm, args=(1,)).start() # left arm
    threading.Thread(target=control_arm, args=(2,)).start() # right arm
    while True:
        time.sleep(1)

if __name__ == "__main__":
    main()
```

Elephant Robotics



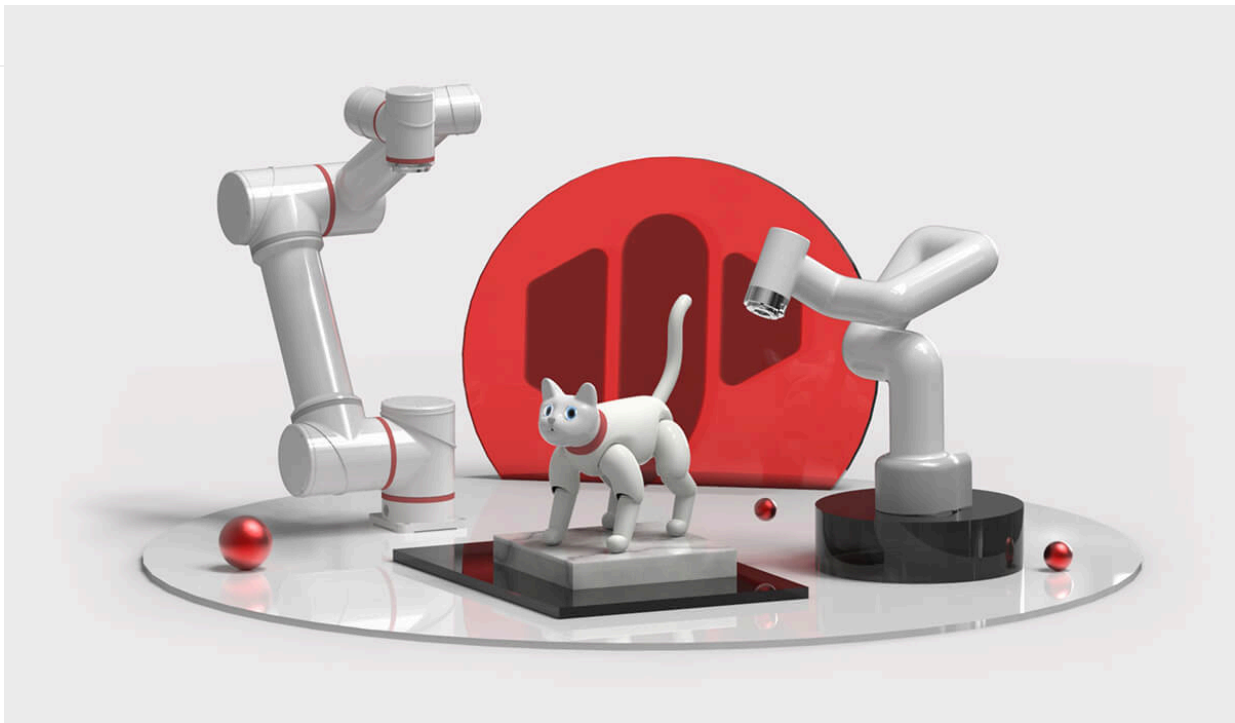
1. Company Introduction

Elephant Robotics, based in Shenzhen, China, is a high-tech enterprise focused on robotics R&D, design, and automation solutions.

We are committed to providing flexible collaborative robots, easy-to-learn operating systems, and intelligent automation solutions for robotic education and research institutions, commercial scenarios, and industrial production. Our product quality and smart solutions are highly recognized and praised by world-renowned Fortune 500 companies in countries like South Korea, Japan, the United States, Germany, Italy, and Greece.

Elephant Robotics adheres to the vision of "Enjoy Robots World," advocating for human-robot collaboration. Our goal is to make robots an excellent assistant to human work and life, helping people break free from simple, repetitive, and tedious tasks, fully leveraging the advantages of human-robot collaboration, thereby improving work efficiency and helping create a better life for humanity.

In the future, Elephant Robotics hopes to drive the development of the robotics industry through next-generation cutting-edge technology, working hand in hand with partners to usher in a new era of automation and intelligence.



2. Development History

- 2016.08 ----- Elephant Robotics Co., Ltd. was officially established
- 2016.08 ----- Entered HAX incubator, received SOSV seed funding
- 2016.08 ----- Started developing Elephant S industrial collaborative robots
- 2017.01 ----- Ranked among the "Top 10 Most Innovative Companies in China at CES"
- 2017.04 ----- Attended Hannover Industrial Expo and Korea Automation Exhibition
- 2017.07 ----- Two founders were selected for Forbes Asia's "30 Under 30"
- 2017.10 ----- Released the fifth-generation single-arm industrial collaborative robot, Elephant S
- 2018.04 ----- Received angel round investment from Yun Tian Shi Fund
- 2018.06 ----- Made its first public appearance at the 2018 Hannover World Industrial Expo
- 2018.06 ----- Awarded the "Smart Manufacturing Entrepreneur MBA Award" from Cheung Kong Graduate School of Business
- 2018.06 ----- Awarded the "Entrepreneur Accelerator X-elerator Award" from Tsinghua University School of Economics and Management
- 2018.11 ----- Ranked second in the Shenzhen division of the Asian Smart Hardware Competition
- 2018.11 ----- Won the "Most Investment-Worthy Enterprise Award" at the Gaogong Golden Globe Awards
- 2019.03 ----- Won the "Leading Talent Award" at the Gaogong Golden Globe Awards
- 2019.04 ----- Catbot won the "Industrial Robot Innovation Award" in March 2019
- 2019.09 ----- Attended Huawei European Ecosystem Conference (HCE), officially becoming a Huawei ecosystem partner
- 2019.11 ----- Elephant Robotics and Harbin Institute of Technology attended IROS International Symposium on Intelligent Robots and Systems
- 2019.12 ----- Officially opened the "Smart Robotics Joint Development Lab" with South China University of Technology
- 2019.12 ----- Won the "Innovative Technology Award" at Gaogong in 2019
- 2019.12 ----- Won the "Top 10 Rapidly Growing Enterprises" Award at Gaogong in 2019
- 2019.12 ----- Won the "Rising Star Enterprise Award" in the industrial robot sector at Shenzhen Equipment Industry
- 2019.12 ----- Released the world's first bionic robotic cat, MarsCat

- 2020.05 ----- Founder received the "Rising Star of Shenzhen Robotics 2019" Award
- 2020.10 ----- Released the world's lightest and smallest six-axis collaborative robot, myCobot
- 2021.03 ----- Released the smallest collaborative robot for scientific research, myCobotPro 320
- 2021.05 ----- MarsCat received extensive media coverage, including from Xinhua Finance, China Daily, Nanjing Daily, and Harbin Daily
- 2021.07 ----- Released the smallest composite robot chassis – myAGV mobile robot
- 2021.09 ----- Released the world's first fully enclosed four-axis robotic arm – myPalletizer

3. Relevant Links

Purchase Links

- Taobao: <https://shop504055678.taobao.com>
- Shopify: <https://shop.elephantrobotics.com/>
- AliExpress: <https://elephantrobotics.aliexpress.com/store/1101941423>

Other Information

- Official Website: <https://www.elephantrobotics.com>
- Videos
 - Bilibili: <https://space.bilibili.com/2126215657>
 - YouTube: <https://www.youtube.com/c/Elephantrobotics>
- Facebook: <https://www.facebook.com/mycobotcreator/>
- LinkedIn: <https://www.linkedin.com/company/18319865>
- X (Twitter): <https://twitter.com/CobotMy>
- Discord: <https://discord.gg/2MAherp7nt>
- Hackster: <https://www.hackster.io/elephant-robotics>

4. Contact Us

Our working hours are Monday to Friday, 10:00 AM to 6:00 PM (Beijing Time).

- If you have any other questions, please contact us through the following methods:

E-mail :

`support@elephantrobotics.com`

- If you have purchase intentions or any questions about parameters, please email us at: [E-mail]
(`sales@elephantrobotics.com`) :

`sales@elephantrobotics.com`

- If the listed problems don't resolve your issue, and you have additional after-sales questions, please email us at this address:

E-mail :

`support@elephantrobotics.com`

We will respond within 1-2 working days.

WeChat:

We provide one-on-one service only to users who purchase MyController S570 series products via WeChat.



[← Previous Chapter](#) | [Next Chapter →](#)

Acknowledgments

We would like to express our deep gratitude to all the individuals involved in the development, testing, and improvement of the MyController S570 series products . Every detail and every innovation in functionality is the result of the hard work and dedication of the team behind it.

Special Thanks:

R&D Team: Thank you for your innovative thinking and countless days and nights of effort, turning complex technology into user-friendly products.

Quality Assurance and Testing Team: Your meticulous attention to every detail has ensured the reliability of our products and the ultimate user experience.

Customer Support Team: Thank you for providing professional support to our users and helping them solve every issue during the usage process.

Partners and Suppliers: Your support and service have been crucial to the success of the product. We appreciate your high-quality materials and components, as well as your readiness to assist at any time.

Investors and Advisors: Without your trust and financial support, we could not have brought these innovations to the market. Your insights and guidance continue to be a driving force for us.

User Appreciation:

We especially thank every user who has chosen and trusted the myArm series products. Your feedback and suggestions are the driving force behind our continuous progress and improvement. We promise to keep listening to your voices and constantly optimize our products and services.

Looking Ahead:

We look forward to continuing to explore and advance in the field of robotics technology with all stakeholders. Let us work together to create more possibilities and bring greater convenience and innovation to the world.

[← Previous Chapter](#)