

Table of Contents

Product Information

Introduction	1.1
1 Product Introduction	1.2
2 Product Specifications	1.3

Initial Setup

3 User Instructions	2.1
4 Initial Installation	2.2

Features and Applications

5 Basic Function	3.1
5.1 Minirobot	3.1.1
1 Minirobot User Guide	3.1.1.1
2 Drag-and-drop teaching	3.1.1.2
3 Zero Calibrate	3.1.1.3
4 Communication Transponder	3.1.1.4
5 Status Information	3.1.1.5
6 Burn Function	3.1.1.6
5.2 myStudio	3.1.2
myStudio Environment Setup	3.1.2.1
Installation of drivers	3.1.2.2
Burning and updating firmware	3.1.2.3
5.3 Firmware Version Description	3.1.3
6 Software Development Guide	3.2
6.1 Python	3.2.1
API	3.2.1.1
Related Examples	3.2.1.2
6.2 ROS1	3.2.2
ROS workspace code and usage	3.2.2.1
Common ROS tool commands	3.2.2.2
ROS architecture and communication	3.2.2.3
6.3 ROS2	3.2.3
ROS workspace code and usage	3.2.3.1
Common ROS tool commands	3.2.3.2
ROS architecture and communication	3.2.3.3

6.4 Communication Packages	3.2.4
7 Successful Cases	4.1
1 myarm mc remote operation case	4.1.1
2 Keyboard control case	4.1.2
3 Palletizing and handling case	4.1.3

Support and Services

8 About Us	5.1
------------	-----

Acknowledgments

9 Acknowledgments	6.1
-------------------	-----

myArm Master 750

Universal Intelligent Six Degree of Freedom Robotic Arm

Core Document

This document contains comprehensive information ranging from product introduction, detailed technical specifications to user instructions and initial installation guidance. We delve into the fundamental functionalities of the myArm M750 robotic arm, provide software development guidelines, and showcase successful application cases to help you understand how to effectively integrate the myArm M750 into various applications. Additionally, we offer rich support and service information to ensure you receive necessary assistance when encountering any technical challenges.

gitbook-en

English Version: https://docs.elephantrobotics.com/docs/myarm-master_750-en/

gitbook-cn

Chinese Version: https://docs.elephantrobotics.com/docs/myarm-master_750-cn/

Document Overview

Based on your needs and the professional level of myArm M750 application development, you can choose to follow this sequence from start to finish or use it as an independent reference. You can navigate to any section using the sidebar navigation on the left at any time. The document is divided into the following five major sections:

Product Information

The Product Information section provides you with a basic overview of the robotic arm, including design intent, key features, and application scenarios, helping you quickly understand the basic characteristics and usage environment of the product. Additionally, this section will detail application examples and supported extension development, providing you with necessary development guidelines and resources. Relevant purchase links and channels will be provided at the end for your convenience.

Product Parameters

The Product Parameters section will showcase detailed technical specifications including machine specifications, control core parameters, structural dimensions, and electrical characteristics, which are crucial for understanding the technical standards and performance indicators of the product. Additionally, Cartesian coordinate system information regarding the robotic arm's working range and precision will be provided as a reference for users who require precise operations.

Features and Applications

The Features and Applications section elaborates on the basic functionalities of the robotic arm and software usage methods, including system usage instructions and firmware functionalities. The software development guide provides guidance based on different development environments such as Python and ROS, supporting technical

developers in application extensions. By showcasing successful application cases and providing supporting resources, we aim to offer practical references and necessary support materials for you to better understand and utilize the product.

Support and Services

The Support and Services section will provide you with comprehensive troubleshooting guides 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. Additionally, the 'About Us' section enhances users' understanding of the design and manufacturer of the myArm series products, aiming to build trust and brand loyalty.

Acknowledgments

We greatly appreciate your time spent reading the myArm M750 user manual. We hope this document will help you better understand and effectively use this robot, inspiring 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 use the myArm M750 for innovative projects and welcome you to join our rapidly growing developer community.

Product Overview

myArm Master 750



General -purpose intelligent six free degree robotic arm

Product introduction

The device of Myarm Master 750 Six Freedom Movement Control Verification It is a universal intelligent robotic arm with an advanced Master main controller. As a device for exercise control verification, it is suitable for a variety of scenarios that require complex motion control, accurate positioning and highly programmed, which can realize end -to -end data training and duplication, research and business display.

The robotic arm has a six -degree -free precision operation ability. It is designed for education and research, suitable for machine learning, AI, and visual -based tasks. The robotic wing is 750mm and the standard load is 500g. Under specific conditions, it can carry a maximum of 1kg, which can meet the needs of various scientific research and experimental needs.

Design Philosophy

The design concept of the MyARM M750 universal intelligent robotic arm comes from a deep understanding of the current demand for scientific and technological education and research. The rapid development of high -tech fields such as machine learning and artificial intelligence requires experimental equipment not only to have common

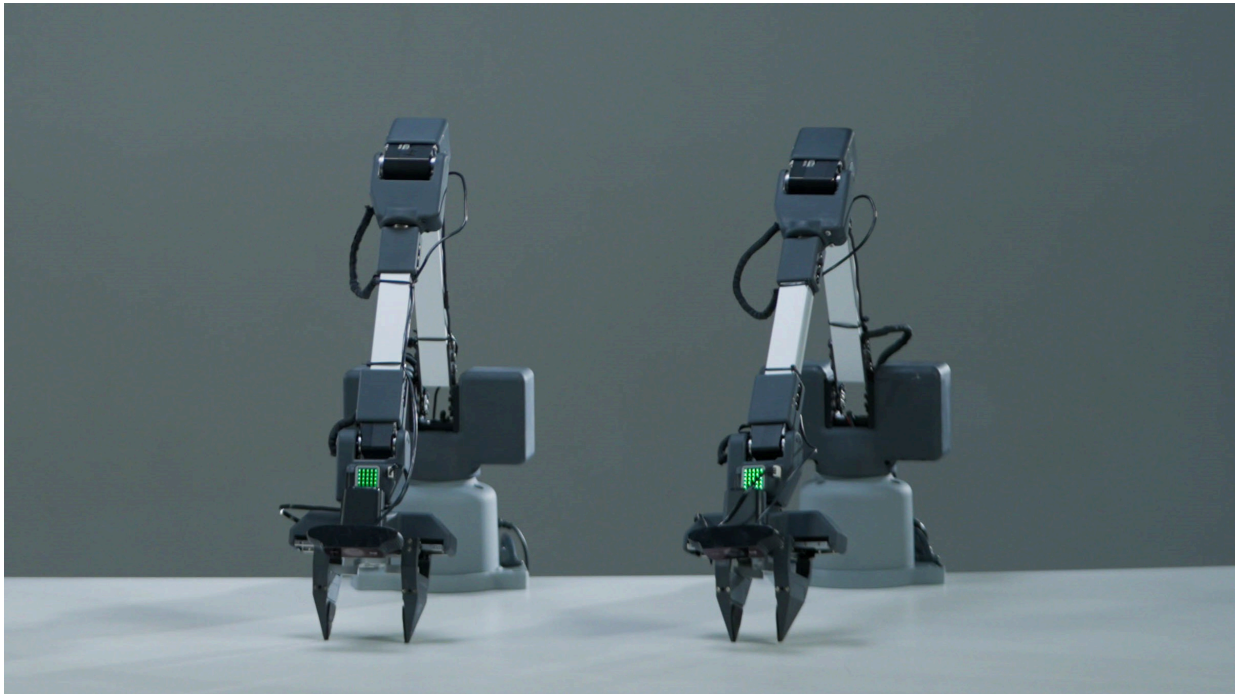
functionality, but also to have flexibility and accuracy. The design aims to meet these complex needs and provide a solution that is both flexible and accurate and adapted to various application scenarios.

Design Goals



Design Goals	Functional realisation
Universal multifunctional platform	The MyARM M750 is suitable for various application scenarios such as education, research and business display. Its six -degree of freedom and 750mm wingspan support to perform complex motion control in various working environments, such as precise positioning and path planning.
Scientific research and education support	The MyARM M750 is suitable for machine learning and artificial intelligence research, which can perform high -precision experiments and technical demonstrations. It supports end -to -end data training and reproduction, and basic visual tasks, which is an ideal device in the laboratory.
Programming and scalability	The highly programmer of the MyARM M750 allows users to customize and programming according to emerging technology to meet future technical needs and achieve optimized operations and experimental results.
Technical innovation and knowledge spread	In business display, the MyARM M750 can be used as a platform carrier to show the latest scientific and technological achievements. It aims to improve the public's understanding and interest in science and technology and promote the transformation of scientific and technological innovation to commercialization.

Features



Feature description	content
6 Freedom modular design	The perfect combination of flexibility and scalability is suitable for complex operations.
Industrial -grade digital servo motor	Ensure the precise control and long term stable operation of the robotic arm.
High-precision encoder	Provide accurate location, speed and acceleration information to optimize performance.
Support multiple development environments	Including Python and ROS to meet different development needs.
Support localization dragging teaching	Intuitive learning and operation methods do not rely on external devices.
High-speed data interface	Open a variety of state interfaces to support complex applications.
Customized embedded software	Provide user friendly operation interfaces to simplify complex tasks.
Built in 2 inch display	Real time display robotic arm status and operation feedback.
Multi -connection method support	Flexible scene application, seamlessly access the existing system.
Central symmetrical format	Ensure balanced and stable and improve operating accuracy.

Product Value

Enhance experiment and research capabilities	The MyARM M750 provides researchers with a platform that can perform high precision operations to help complex data analysis and algorithm verification.
Improve the quality of education	In the educational environment, the device can provide opportunities for practical operations to help students better understand theory and cultivate practical ability.
Increase business and display opportunities	The M750 is not limited to laboratory use, its application in technical exhibitions and public demonstrations can attract audiences and potential customers, enhance the interaction and attractiveness of technical display.

Industry Contribution

Promoting STEM Education	By providing high -level teaching tools, it will stimulate students' interest in science, technology, engineering and mathematics, and cultivate future innovators.
Promoting technology adopts	By providing easy -to -use and highly flexible robot platforms, reducing the technical threshold, so that more institutions can contact and use advanced robotics technology.
Stimulate innovation and personal development	Provide developers and engineers with an open platform, support cross -disciplinary learning and innovation, and cultivate key talents for the future development of robotics technology and related fields.

Application



Client

Higher education institutions and research laboratories	myArm M750 is a teaching and research tool designed for high-precision experiments and technology demonstrations. It can effectively support complex data analysis, algorithm development and validation activities, significantly improving research quality and educational effectiveness.
Higher education institutions and research laboratories	Supporting both Python and ROS development environments, the myArm M750 is suitable for professionals who need personalised programming and system integration. Its modular design and high programmability make it the ideal platform for developing and testing new control algorithms or robotics applications.
Organiser of commercial displays and public exhibitions	The myArm M750 has become the preferred equipment for science and technology demonstrations and product presentations due to its precision operation and display advantages. Dynamic demonstration not only attracts the audience, but also enhances the sense of participation and effectively promotes technological innovations and products.
Innovative enterprises and start-ups	The myArm Master 750 provides powerful support for companies seeking to integrate cutting-edge robotics to enhance product functionality or optimise production processes. Its outstanding performance and adaptability make it ideal for exploring new technologies and solutions.

Application Scenario

User Group	Application Scenario	Aims of Strength
Teachers and students in the field of education	- STEM Education - Robotics Programmes - Interdisciplinary Research Programmes - Teaching Aids for Robotics Subjects	- Increase students' interest in science and technology - Enhance hands-on and problem-solving skills - Promote creative thinking and teamwork - Serve as a construction tool for disciplines such as robotics structural design, circuit design, and mechanics analysis
Technology developers and engineers	- Prototyping - Experimental studies - Algorithm testing and validation - Robot kinematics validation - Robot composite scenario application validation	- Accelerating Research Progress - Connecting Theory and Practice - Advancing Technological Innovation - Validating Robot Kinematics and Composite Scenario Applications

User Group	Application Scenario	Aims of Strength
Business Presentation and Marketing Professionals	- Exhibition - Technology Demonstration - Brand Promotion	- Attract potential customers and investors - Demonstrate the company's technological strength and innovative products - Enhance brand influence
Engineers and technology developers	- Robot Kinematics Validation - Robot Remote Control Scenario Exploration and Development - Machine Learning, AI and Vision Based Tasks	- Perform validated machine operation of the C650 - Explore and develop robotic telecontrol applications - Extend the utility of robotics applications and scenarios

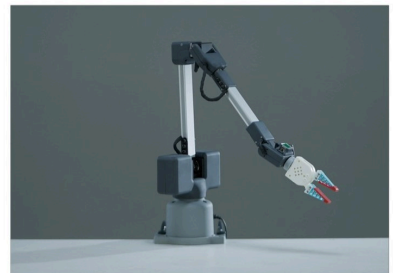
Peripheral Accessories



myCobot Adaptive Jaws



myCobot Parallel Jaws



myCobot Flexible Clamping Jaws



myCobot Vertical Suction Pump V2.0



myCobot Double Head Suction Pump



myCobot Camera Flange V2.0

A wide range of peripheral accessories provides myArm M750 arm with extensive functionality extensions, making it suitable for a variety of industrial, research and educational scenarios. By combining these accessories, users can significantly increase the flexibility and utility of the myArm M750 arm.

myCobot Adaptive jaws

The gripper jaws are self-adjusting to accommodate objects of different shapes and sizes and are designed to perform complex gripping tasks such as handling irregularly shaped or fragile objects. The flexibility and adaptability of the gripper significantly enhances the arm's operational efficiency and application versatility. The upgraded Adaptive Gripper provides even more gripping power and is compatible with a wide range of programming environments, making it suitable for a wide range of industrial-grade robotic arms.

myCobot Parallel jaws

Designed for tasks requiring precise control, such as the assembly and testing of electronic equipment. The gripper's two fingertips move in parallel to ensure object stability during gripping, making it ideal for handling uniformly shaped objects. Its compact design and multiple connection holes are designed to meet different mounting requirements, and it supports both IO and serial control, making it compatible with a wide range of industrial-grade robotic arms.

myCobot Flexible jaws

Suitable for handling sensitive or fragile objects, the flexible gripping jaws have a soft and resilient gripping surface that effectively reduces pressure on the object and reduces the risk of damage. Ideal for gripping fragile objects such as glass, plastic or porcelain. The fingertips are made of rubber and use pneumatic deformation for gripping. Widely used in a variety of automation and robotics applications, these grippers are favoured for their flexibility, adaptability and efficiency.

myCobot Vertical Suction Pump V2.0

The Vertical Suction Pump V2.0 is an upgraded version of the vacuum adsorption system for vertical lifting of smooth surface objects such as sheet metal, glass or plastic. The new version offers greater suction capacity and higher durability and is particularly suitable for fast and repetitive handling operations.

myCobot Double head suction pumps

The dual-head suction pump is equipped with two independent suction cups, allowing the arm to handle two objects simultaneously or provide greater stability and support, enhancing efficiency in assembly line tasks or applications where multiple parts are handled simultaneously.

myCobot Camera Flange V2.0

Camera Flange V2.0 enables a standard camera to be mounted on a robotic arm to provide the necessary visual feedback to the arm, especially for tasks that require visual recognition, such as quality inspection, colour sorting or precision placement.

myCobot Mobile phone holder

This accessory allows the myCobot robotic arm to hold and operate a smartphone or similar device and is suitable for automated testing of mobile phone applications, for educational presentations, or for using the phone's camera in specific scenarios.

Supported Extended Development



The myArm series of robotic arms are extremely valuable in education and research, especially in Python and ROS (Robot Operating System), two widely used development environments. These environments provide strong support for the myArm series in a wide range of machine learning, artificial intelligence research, complex motion control, and vision processing tasks.

Python	- Provides interfaces for joint angle control, servo motor control, servo motor parameter configuration, and Cartesian coordinate system control. - Supports more advanced motion control and precise positioning for sophisticated robotics research and application development. - The driver library can be downloaded and installed via PyPI, providing comprehensive interface support for more complex educational or research projects.
ROS	- It supports basic MoveIt functions such as keyboard control and basic path planning, and provides dual version support for ROS1 and ROS2. - Enhanced real-time robot arm display and state information acquisition through RVIZ simulation environment, suitable for more advanced robot path planning and operation research.
Hardware Interface	- Includes IO, USB, etc. for easy connection to various sensors and actuators.
Software Library	- Provides rich open source libraries and APIs to simplify the development process.
System Compatibility	- Compatible with Windows, Linux, MacOS, adapt to a variety of development environments.

Purchase Address

Purchase Address:

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

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

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

[Next chapter →](#)

Parameters of the bot

In the first chapter, we explore the selling points of the product and its design philosophy, giving you a panoramic view of the high-level understanding of the product. Now, let's move on to the second chapter – Robot Parameter Description. This section will be the key to understanding the technical details of the product. A detailed understanding of these technical specifications will not only help you fully understand the advanced and practical nature of our products, but also ensure that you can use these technologies more effectively to meet your specific needs.

1 Product Specifications

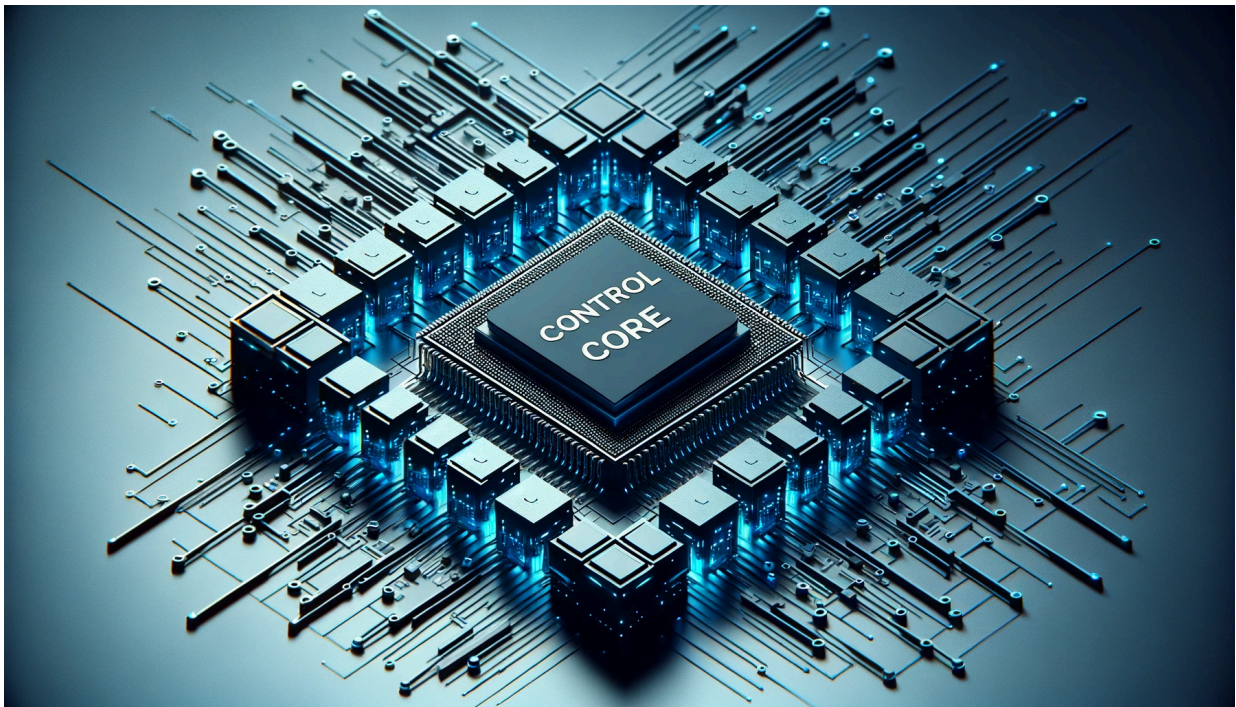
1.1 Machine Parameters

Specifications	Parameter
Model	myArm M750
DOF	6+1
Horizontal Reach Range	750
Total span	1500mm
Self-weight	3.2kg
Power Specifications	24V9.2A
Repeatability	±1mm
Accuracy	5 - 8mm
Working load	Rated 500g, peak 1Kg
Number of servos	8
Servo type	Industrial-grade high-precision digital servo motor
Rotation capacity	+/- 180°
End effector	Parallel gripper, optional camera adaptation
USB connection	Type-C
Atom end	5*5 LED light matrix
Communication frame rate	>50Hz

1.2 Software Basic Function Support

Function/Development Environment	Usage
Joint Movement	Yes
Cartesian Movement	Yes
Wireless control	Yes
Emergency stop	Yes
Windows	Yes
Linux	Yes
MAC	Yes
ROS	Yes
Python	Yes
myblockly	Yes
mystudio	Yes
Serial Port Control Protocol	Yes

2 Control core parameters



2.1 Master Controller Spec Sheet

Indicators	Parameters
Master Control	M5Stack-basic
Master Model	ESP32
CPU	240MHz dual core. 600 DMIPS、520KB SRAM。 Wi-Fi, dual-mode Bluetooth
Bluetooth	2.4G/5G
Wireless	2.4G 3D Antenna
Enter	IN1, IN2, IN3, IN4, IN5, IN6
Output	OUT1, OUT2, OUT3, OUT4, OUT5, OUT6

2.2 Secondary Controller 1 Spec Sheet

Indicators	Parameters
Auxiliary Control	Atom
Auxiliary Control Model	ESP32
Auxiliary controller core parameter	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

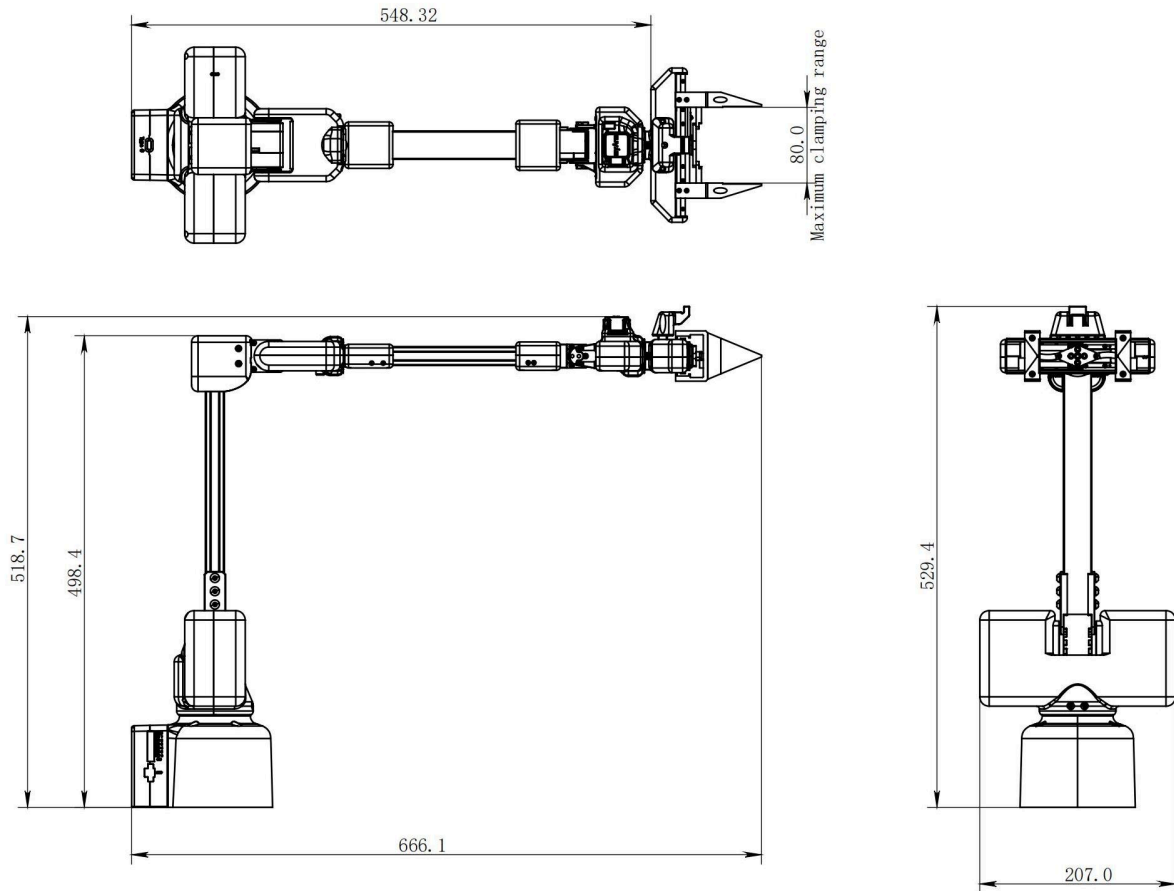
2.3 Secondary Controller 2 Spec Sheet

Indicators	Parameters
Auxiliary Control	Pico
Auxiliary Control Model	ESP32
Auxiliary controller core parameter	240MHz dual-core. 600 DMIPS, 520KB SRAM。 Wi-Fi, dual-mode Bluetooth
Secondary controller flash	4MB
TypeC	*1

3 Structural Dimension Parameters

! This chapter is in millimeters of distance and degrees of angle.

3.1 Product Dimensions and Workspace



3.2 Base Mounting Dimensions

- The base needs to be flanged and can be fixed to the corresponding mounting base using M6 screws.
- Before use, please confirm that the installed base can bear 3 times the weight of the body to prevent damage to the product caused by loosening of the product due to the increase in movement speed during use.

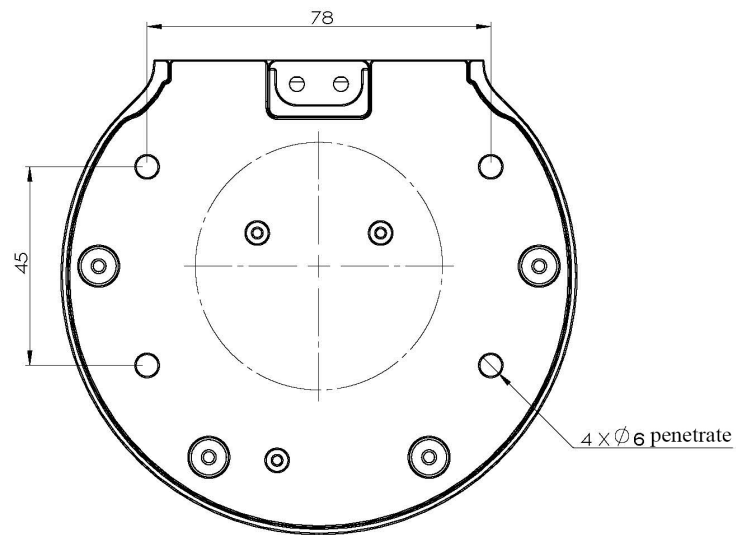
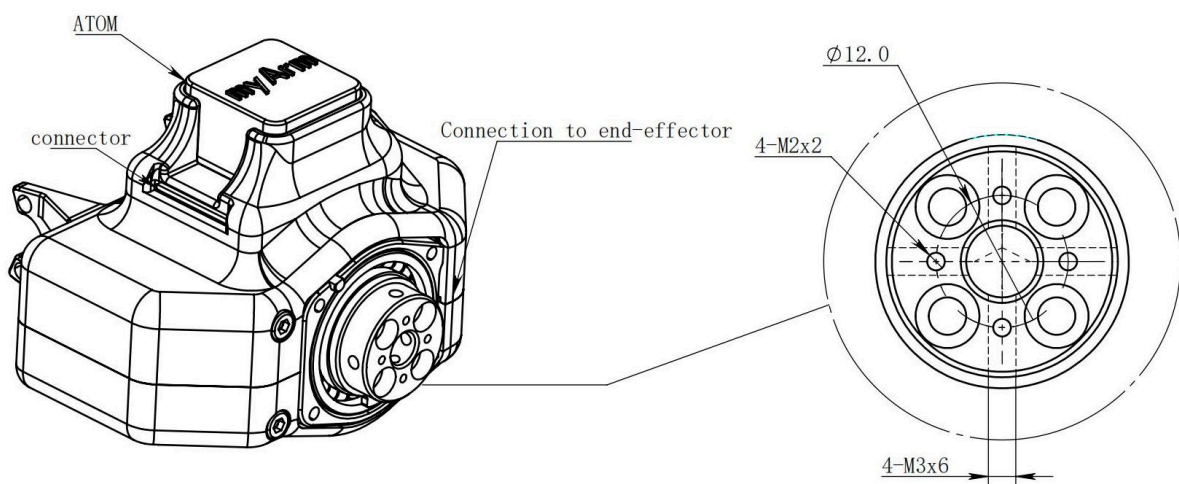


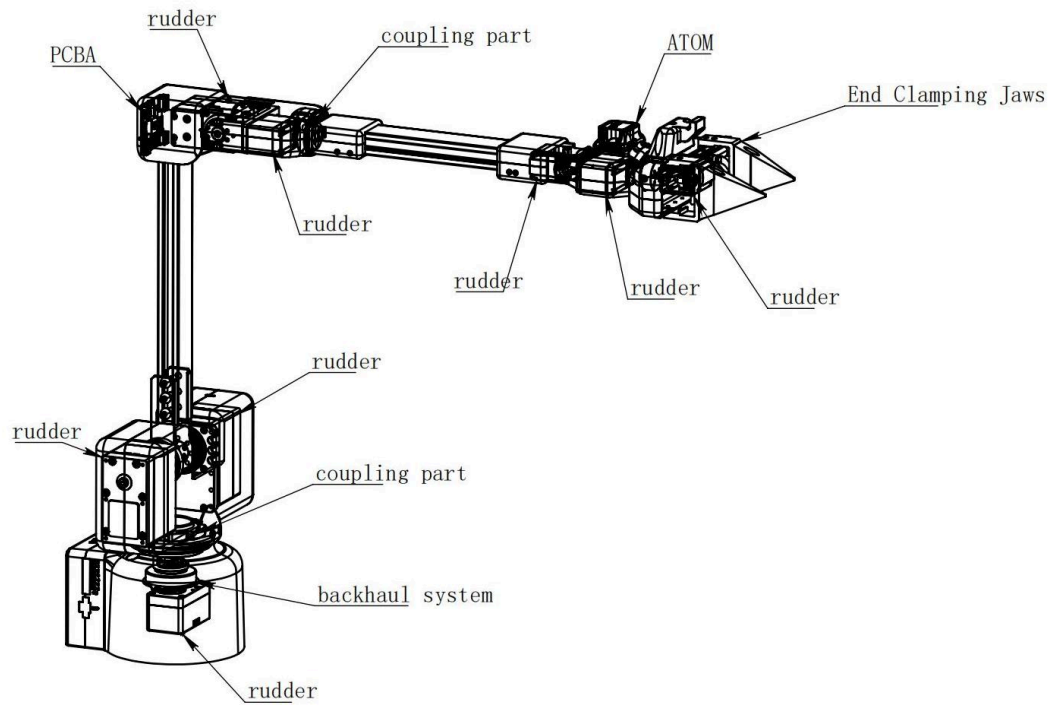
Figure 1 Front view of the base

3.3 End of arm

- The end of the arm is compatible with LEGO component holes and threaded holes.



3.4 Products



3.5 3D Model Download

[M750 model download](#)

4 Electrical Characteristic Parameters

4.1 Overview of the Electrical Interface of the Base



Figure 1 Front view of the base



Figure 2 Left side of the base



Figure 3 Right side of the base

No.	Interface name	Define	Features	Note
1	Type C	Communication Interface	Communicate with a PC	Development and use
2	Screen	Show	Use	with buttons
3	Press	Press the key A	Use	with the display
4		Press the B		
5		Press the C		
6	Switch	Power switch	Control the input power on and off	With light (light on)
7	DC/IO interface	GND	GND	
		IN6	Digital input signal 1~6	only in NPN mode
		IN5		
		IN4		
		IN3		
		IN2		
		IN1		
		24V	DC24V	
8	Type C	Communication Interface	Communicate with a PC	Development and use
9	Power DC input interface	DC24V input	DC24V input	
10	DC/IO interface	24V	DC24V	
		OUT1	Digital output signal 1~6	only in PNP mode
		OUT2		
		OUT3		
		OUT4		
		OUT5		
		OUT6		
		GND	GND	
11	E-stop interface	STOP	Emergency stop circuit interface	

4.1.1 Type C: The Type-C interface is used to connect and communicate with PCs and can be used by developers.

4.1.2 Screen: The screen is used to display the communication status of myArm and calibrate the robot to the starting point through a 2-inch IPS screen.

4.1.3 Keys A, B, and C are used to operate the screen in a coordinated manner.

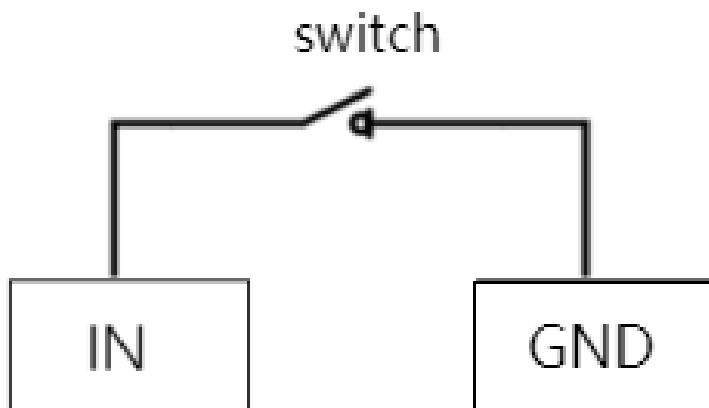
4.1.4 Power Switch: The power switch is used to control the main power input. If it is turned off, the controller will also lose power.

4.1.5 24V output: Built-in DC24V, available for user use.

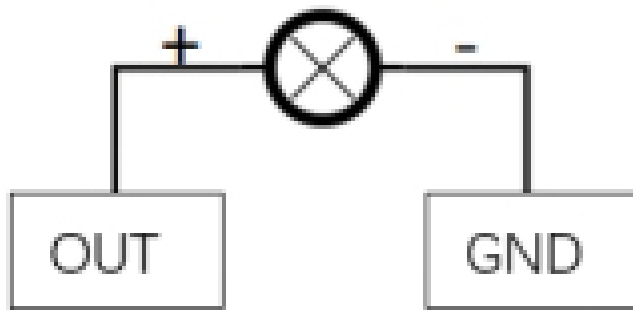
4.1.6 Digital Inputs/Outputs: Includes 6 digital input signals and 6 digital output signals for interaction with other devices and together with other devices form an important part of the automation system.

Digital Inputs/Outputs: Includes 6 digital input signals and 6 digital output signals to interact with other devices and together with other devices form an important part of the automation system.

It is important to note that the output signal is in PNP form and the input signal is in NPN form. Here is the external wiring diagram:



Example of Input Signal Wiring



Example of Output Signal Wiring

4.1.7 Power DC input interface: KPPX-4P R7BFDC power socket. The 24V 9.2A DC power adapter from the manufacturer can also be used to power the myCobot320.

4.1.8 The emergency stop circuit terminal is connected to the emergency stop button box and can be used to control the emergency stop of the robot.

NOTE: The emergency stop switch must be connected when using the robot and ensure that the emergency stop switch circuit is always connected.

4.2 Mechanical Wall End Electrical Interface

4.2.1 Introduction to the end of the robotic arm

A. The side interface at the end of the robotic arm is shown in Figure 2-1.



Figure 2-1 End of the robotic arm

No.	Interface name	Define	Features	Note
9	End IO interface	End Tool IO Interface	Interact with external devices	Development and use
10	End Grove Interface			
11	Type C interface		It can be used to connect and communicate with the PC and update the firmware	
12	End Atom	LED	For 5X5 RGB LED (G27) display and key function (G39)	
13	Servo interface	Connect the servo	Connect an external device servo	

4.2.2 End Interface Description

A. Table 2-1 defines the terminal IO port.

Tag name	Signal name	Features	Note
5V0	5V	5V Power Supply	
GND	GND	Motherboard power signal ground	
3V3	3V3	3.3V Power Supply	
G22	G22	3.3V-OUT-PIN output/3.3V-INT input	
G19	G19	3.3V-OUT-PIN output/3.3V-INT input	
G23	G23	3.3V-OUT-PIN output/3.3V-INT input	
G33	G33	3.3V-OUT-PIN output/3.3V-INT input	

Table 2-1 Terminal IO port

B. End Grove Interface: The Grove interface 4 definition is shown in Figure 2-2

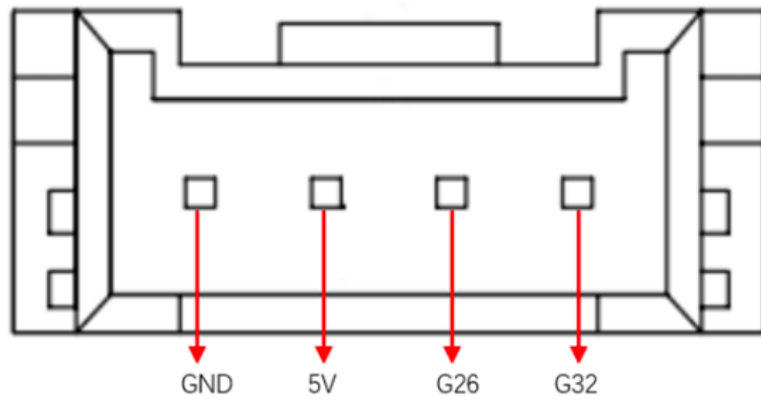


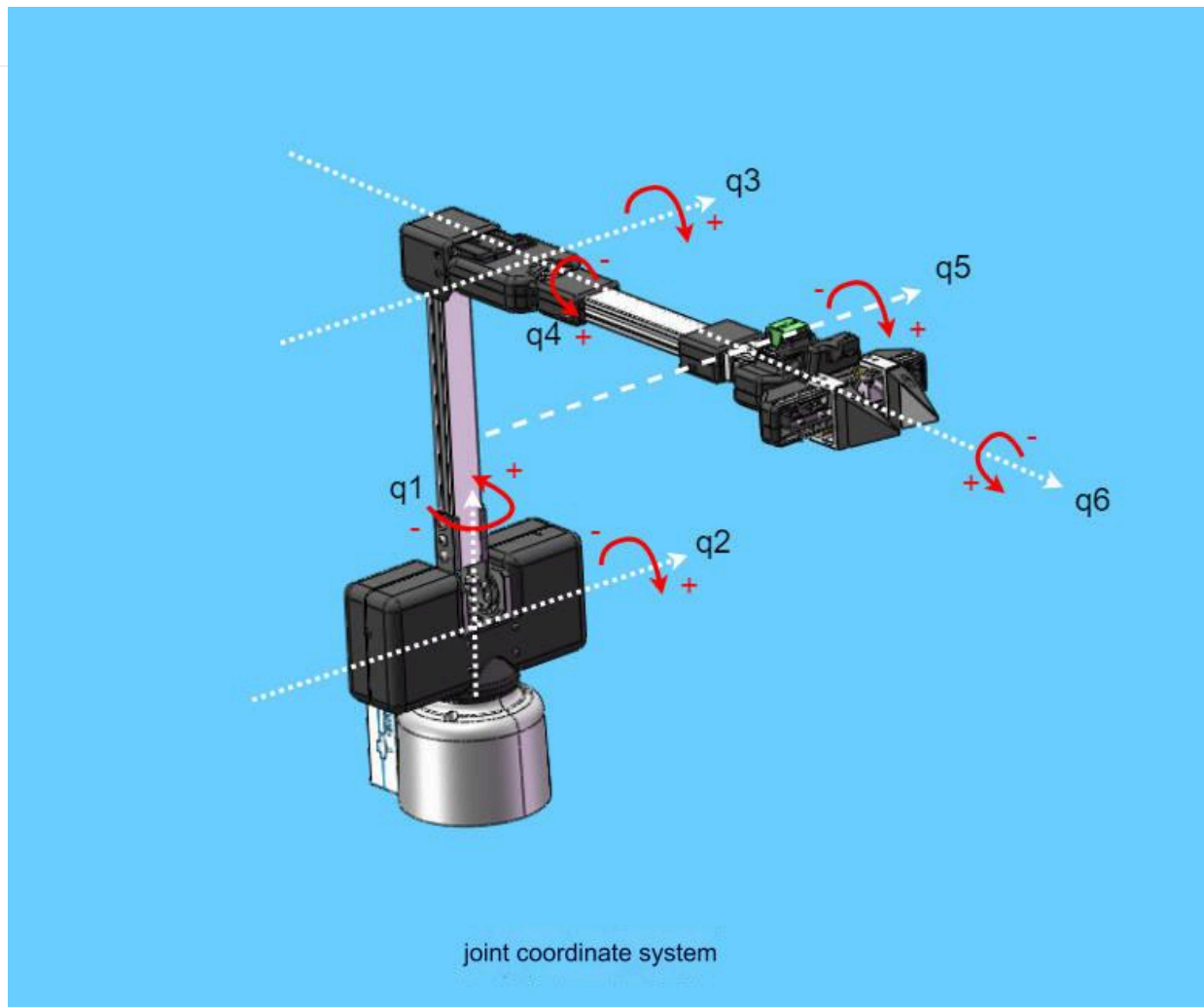
Figure 2-2 End Grove Interface

- C. Type C interface: It can be used to connect and communicate with the PC and update the firmware.
- D. Atom: for 5X5 RGB LED (G27) display and key function (G39)
- E. Servo interface: used for terminal expansion grippers, currently supporting the use of matching adaptive grippers.

5 Cartesian Coordinate System

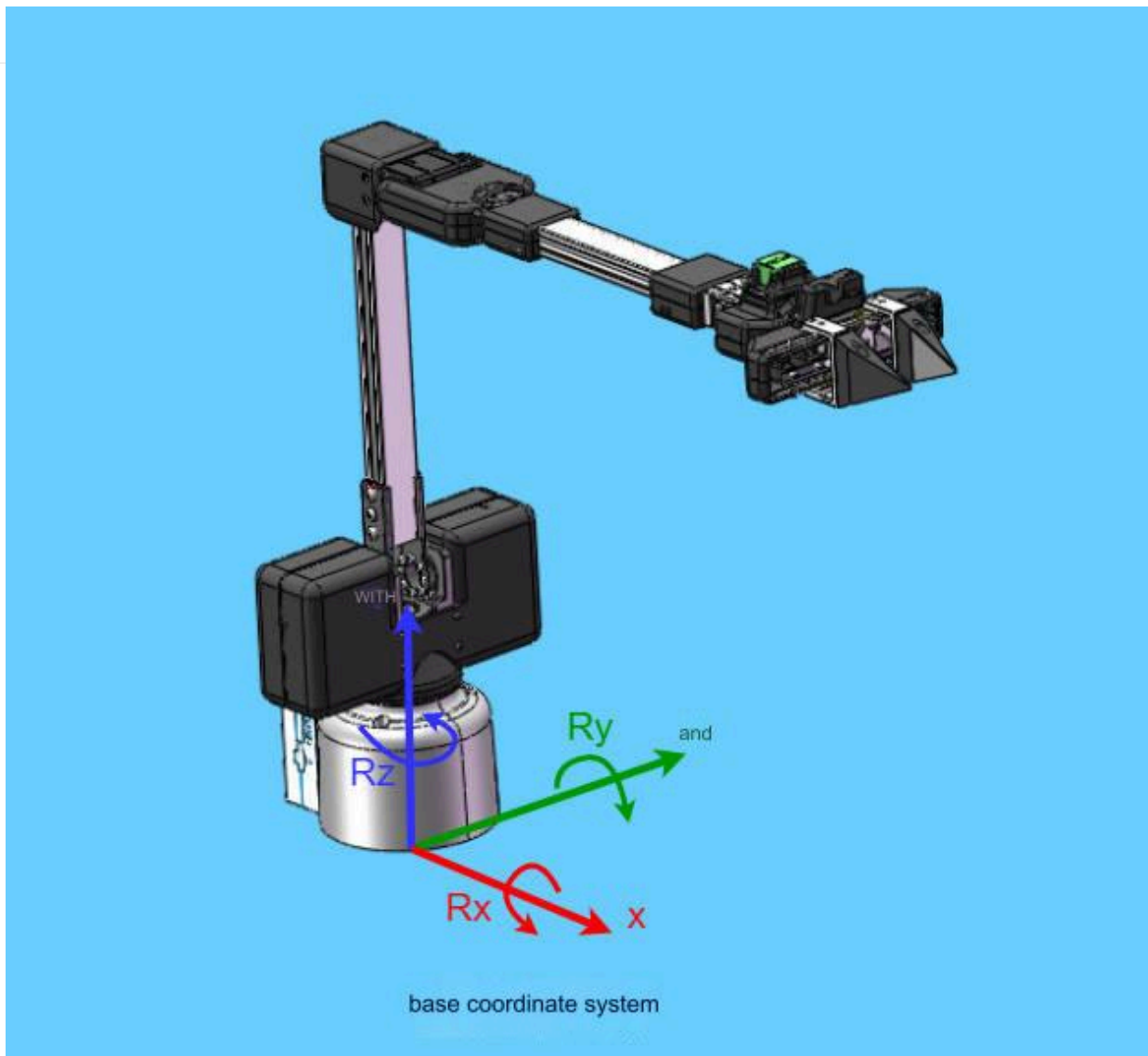
5.1 Joint Coordinate System

The joint coordinate system is a coordinate system based on the rotation joints of the robot arm. The white dotted lines in the figure below represent the rotation axes of each joint, the red arrows represent the rotation directions of the joints, and q_1 - q_6 represent the 1-6 joint coordinate systems.



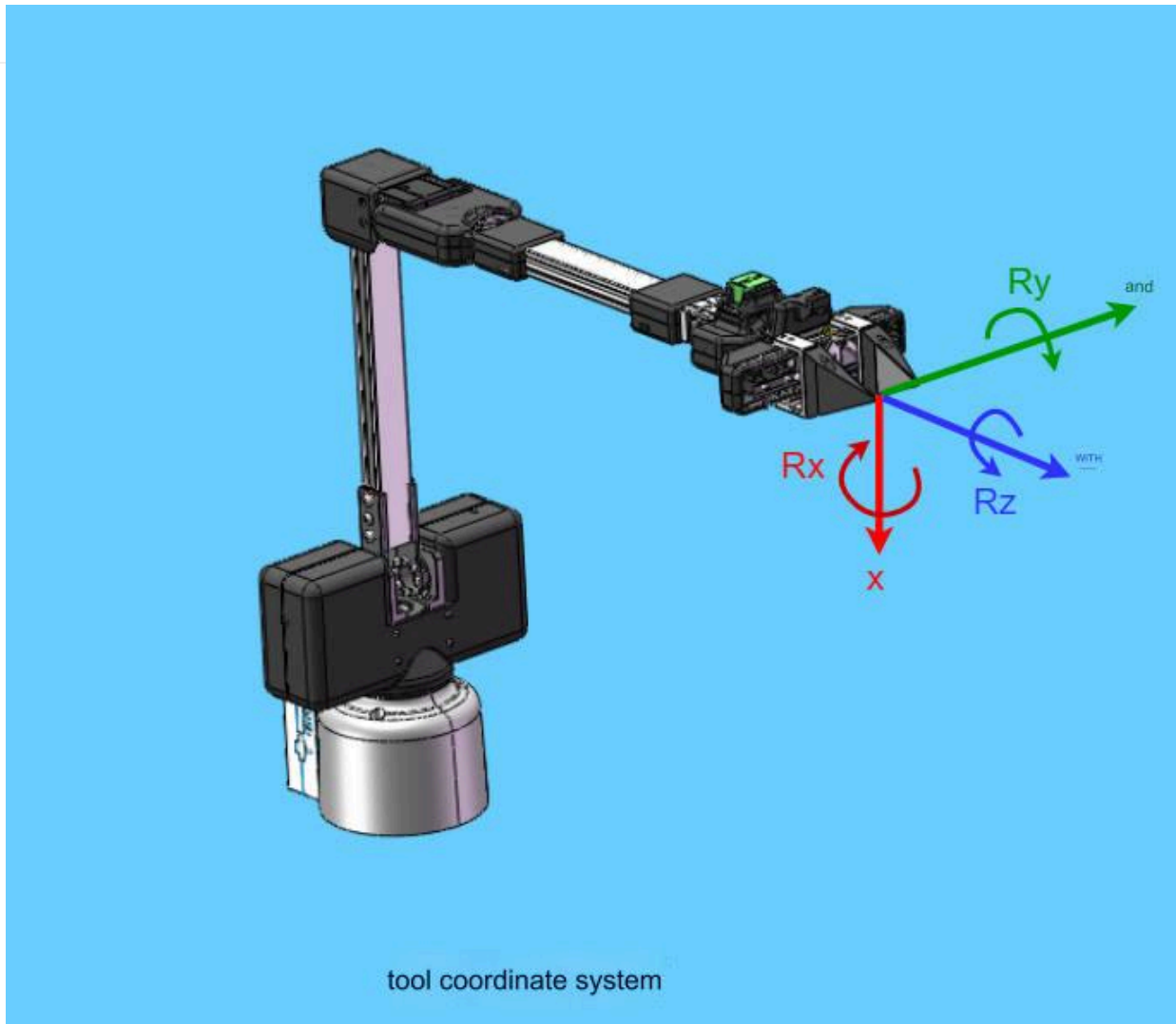
5.2 Base Coordinate System

The base coordinate system is a coordinate system fixed at the bottom of the robot arm. Its origin and coordinate axis directions are determined when the kinematic algorithm is modeled. Generally, the origin is set at the center point of the base.



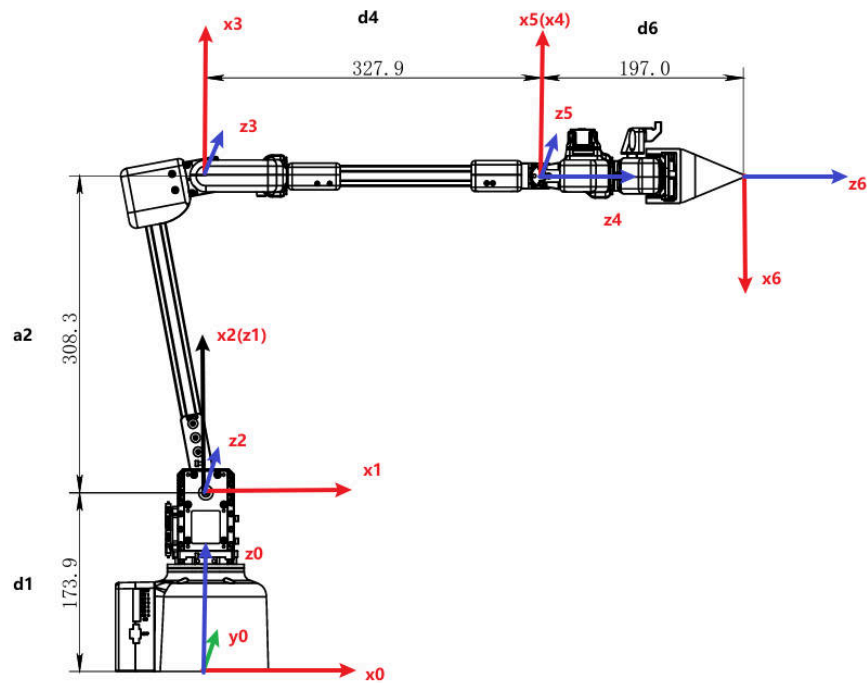
5.3 Tool Coordinate System

The tool coordinate system is a coordinate system fixed at the end of the robot arm. Its origin and coordinate axis direction are determined during the kinematic algorithm modeling. Generally, the origin is set at the center point of the flange at the end of the robot arm.



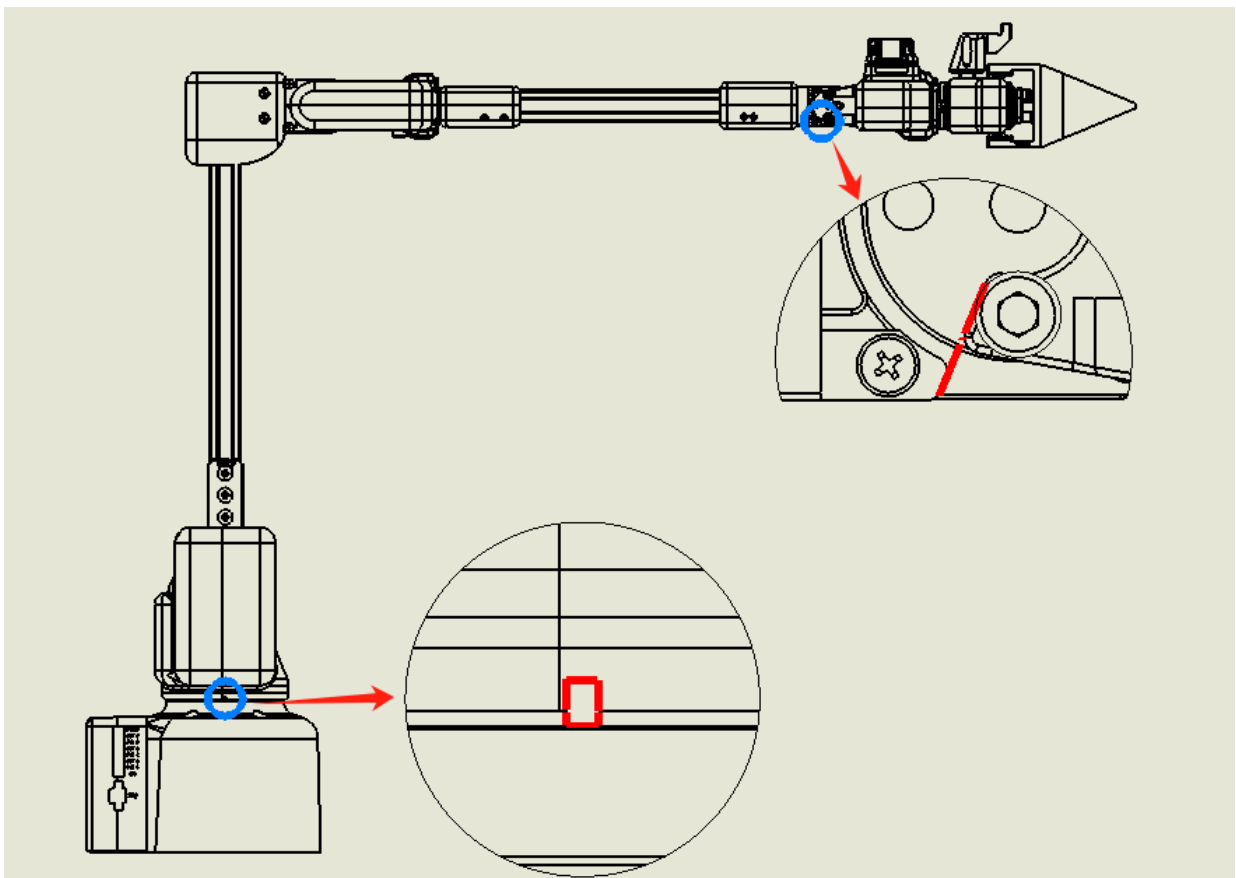
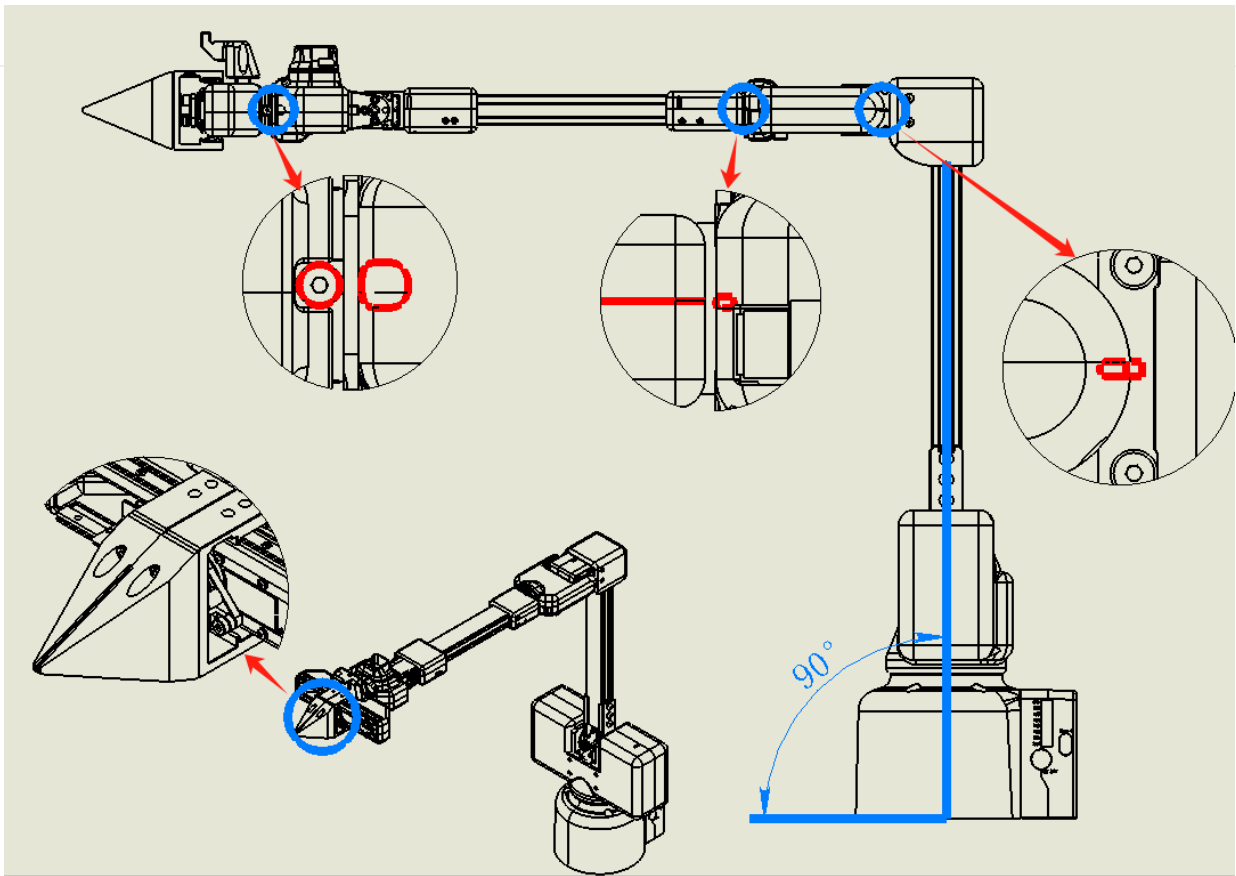
5.4 Kinematic model

The figure below shows the kinematic model of the robot arm. The position shown is the zero point on the algorithm model



5.4.1 Zero point calibration

The zero position posture is shown in the figure below. Please note that if the actual posture is found to be inconsistent with the model posture in the figure below after the software sets the joint to return to zero, please refer to 5.1.3 Zero Position Calibration Method to perform zero position calibration to avoid the robot arm collision caused by inconsistent reading and writing motion range due to abnormal joint zero position.



5.4.2 MDH parameters

DH parameters are used to describe the relative relationship between adjacent links:

- a_i : along x_i : the distance from z_i to z_{i+1}
- α_i : around x_i : from z_i to z_{i+1}
- d_i : represents the distance from x_{i-1} to x_i along z_i
- θ_i : around z_i : from x_{i-1} to The angle of x_i

5.4.3 MDH parameter list

Joint	α_{i-1}	a_{i-1}	d	θ	offset
1	0	0	173.9	θ_1	0
2	$-\pi/2$	0	0	θ_2	$-\pi/2$
3	0	308.315	0	θ_3	0
4	$-\pi/2$	0	327.91	θ_4	0
5	$\pi/2$	0	0	θ_5	0
6	$-\pi/2$	0	197.1	θ_6	π

5.4.4 Joint Limitation

- Shake Control Firmware Version

- **Limit Version Prior to 13 June 2025**

| Joint | Min | Max | | :--- | :---- | :---- | 1 | -170 | 170 | 2 | -83 | 83 | 3 | -90 | 84 | 4 | -155 | 153 | 5 | -91 | 88 | 6 | -153 | 153 | 7 | -118 | 2 |

- **Limit versions after 13 June 2025**

| Joint | Min | Max | | :--- | :---- | :---- | 1 | -165 | 165 | 2 | -80 | 100 | 3 | -100 | 80 | 4 | -160 | 160 | 5 | -90 | 120 | 6 | -180 | 180 | 7 | -118 | 2 |

Note: Prior to firmware version 1.1.3, the maximum limit for Joint 7 was 0; post-1.1.3 it is 2.

Regarding pymycobot versions: 3.9.8 and below apply to the old limits; subsequent versions apply to the new limits.

- Coordinate Firmware Version

| Joint | Min | Max | | :--- | :---- | :---- | 1 | -165 | 165 | 2 | -80 | 100 | 3 | -100 | 80 | 4 | -160 | 160 | 5 | -90 | 120 | 6 | -180 | 180 |

5.4.5 Joint Potential Value Limiting

- Shake Control Firmware Version

- **Limiting Version Prior to 13 June 2025**

| Servo | Min | Max | | :--- | :---- | :---- | | 1 | 114 | 3981 | 2 | 1200 | 2991 | 3 | 1200 | 2991 | 4 | 1024 | 3002 | 5 | 286
| 3788 | 6 | 1013 | 3048 | 7 | 307 | 3788 | 8 | 706 | 2068 |

- **Limited Edition Version After 13 June 2025**

| Servo | Min | Max | | :--- | :---- | :---- | | 1 | 175 | 3910 | 2 | 1370 | 3120 | 3 | 1370 | 3120 | 4 | 1015 | 2890 | 5
| 350 | 3740 | 6 | 1013 | 3048 | 7 | 350 | 3740 | 8 | 706 | 2068 |

Note: Regarding pymycobot versions, versions up to and including 3.9.8 apply to the old potential values, while subsequent versions apply to the new potential values.

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

User Notice

Chapter 3 "User Notice" is essential for ensuring that users can achieve the specified safety standards and efficiency when using the product, making it a must-read for every user. This chapter not only provides basic information on product use, transportation, storage, and maintenance to ensure the safe operation and maximization of efficiency of the product but also elaborates on the liability issues that may arise from failure to comply with these guidelines.

1. Safety Instructions

Introduction

This chapter provides detailed information on general safety for personnel performing installation, maintenance, and repair work on Elephant Robotics. Please read and understand the contents and precautions of this chapter thoroughly before handling, installing, and using the product.

Hazard Identification

The safety of collaborative robots depends on correct configuration and usage. Even when all safety instructions are followed, injuries or damages caused by operators can still occur. Therefore, understanding the safety hazards associated with robot use is crucial to prevent accidents.

Tables 1-1 to 1-3 below outline common safety hazards that may exist when using the robot:

Table 1-1 High-Level Hazard

 Hazard
1 Personal injury or robot damage caused by incorrect operation during robot handling.
2 Failure to secure the robot as required, such as missing screws or insufficient locking ability of the base, resulting in robot tipping over, causing personal injury or robot damage.
3 Failure to correctly configure robot safety features or insufficient installation of safety protective tools, resulting in the failure of safety functions and potential hazards.

Table 1-2 Warning-Level Hazard

**Warning**

- 1 Do not stay within the robot's range of motion when debugging programs, as inappropriate safety configurations may not prevent collisions that may cause personal injury.
- 2 Connecting the robot to other devices may pose new risks, requiring a comprehensive risk assessment.
- 3 Scratches or punctures caused by sharp surfaces of other devices or end effectors in the work environment.
- 4 The robot is a precision machine; stepping on it may cause robot damage.
- 5 Improper clamping or failure to remove clamped objects before turning off robot power or air supply (uncertainty of whether the end effector is firmly holding the object due to loss of power) may cause hazards, such as end effector damage or injury from falling objects.
- 6 There is a risk of unexpected movement of the robot; never stand under any axis of the robot under any circumstances!
- 7 The robot is a precision machine; unstable placement during handling may cause vibration, which may damage internal components of the robot.
- 8 Compared to conventional mechanical equipment, robots have more degrees of freedom and a larger range of motion; failure to meet the motion range may cause unexpected collisions.

Table 1-3 Potential Electric Shock Hazards**Danger Electric Shock**

- 1 Using non-original cables may pose unknown dangers.
- 2 Contact of electrical equipment with liquids may cause electric shock hazards.
- 3 Electrical shock hazards may exist when electrical connections are incorrect.
- 4 Be sure to replace operations after turning off the controller and related devices and unplugging the power plug. If operations are performed with power on, electric shock or failure may occur.

Safety Precautions

The following safety rules should be followed when using the robotic arm:

- The robotic arm is an electrified device, and non-professionals are not allowed to change the circuit at will, as this may cause damage to the equipment or personal injury.
- When operating the robotic arm, local regulations and standards should be strictly followed. The safety precautions and descriptions of "danger," "warning," and "caution" in the manual are only supplementary to

local safety regulations.

- Please use the robotic arm within the specified environmental range. Exceeding the specifications and load conditions of the robotic arm can shorten the product's service life or even damage the equipment.
- Personnel responsible for installing, operating, and maintaining the myArm robotic arm must undergo rigorous training, understand various safety precautions, and master the correct operation and maintenance methods before operating and maintaining the robot.
- Please do not use this product for long periods in a humid environment. This product contains precision electronic components, and working in a humid environment for a long time can damage the equipment.
- Please do not use this device in a high-temperature environment. The outer surface of this device is made of photosensitive resin, and higher temperatures can damage the device's shell, leading to equipment failure.
- High-corrosive cleaning is not suitable for cleaning the robotic arm, and anodized parts are not suitable for immersion cleaning.
- Please do not use this product without installing the base to avoid damage to the equipment or accidents. The product should be used in a fixed environment with no obstacles.
- Please do not use other power adapters for power supply. If the equipment is damaged due to the use of non-standard adapters, it will not be covered by after-sales service.
- Please do not disassemble or open the screws or casing of the robotic arm. If disassembled, warranty service cannot be provided.
- Unauthorized personnel are not allowed to repair faulty products or dismantle robotic arms without authorization. If the product fails, please contact myArm technical support engineers promptly.
- If the product is scrapped, please comply with relevant laws to properly dispose of industrial waste and protect the environment.
- Children must be monitored by an adult during use, and the device should be turned off promptly when the operation is completed.
- Please do not stretch your hands into the range of motion of the robotic arm during operation to avoid injury.
- It is strictly prohibited to change, remove, or modify the nameplates, descriptions, icons, and markings of the robotic arm and related equipment.
- Please handle with care during handling and installation. Follow the instructions on the packaging box to handle it gently and place the robot correctly according to the arrow direction; otherwise, the robot may be damaged.
- **Do not burn other product drivers or use unofficial firmware burning methods without authorization.** Damage to the device caused by the user burning other firmware will not be covered by after-sales service.

If you have any questions or suggestions regarding the content of this manual, please visit the Elephant Robotics official website to submit relevant information:

<https://www.elephantrobotics.com>

Do not use the robotic arm for the following purposes:

- In medical and life-critical applications.
 - In environments where explosions may occur.
 - Direct use without undergoing a risk assessment.
 - Use with insufficient safety functional levels.
 - Use that does not meet the performance parameters of the robot.
-

2. Transportation and Storage

Packing

When packaging the robot product, please ensure the use of packaging materials and boxes specifically designed for it, which can provide sufficient cushioning and support to prevent impact and vibration during transportation. Check whether all parts are properly fixed to avoid damage caused by looseness. For fragile or sensitive components, additional shock protection materials should be used for reinforcement. Finally, make sure that the outer packaging box is clearly labeled with handling and warning labels, indicating the correct handling method and storage orientation.

Logistics Transportation

During transportation, use the original packaging to transport robot products. During transportation, ensure that the robot product is stable as a whole in the packaging box and is protected by appropriate measures. During transportation and long-term storage, the environmental temperature should be maintained between -20 to +55°C, humidity ≤95%, and no condensation.

Due to the precision nature of robots, care should be taken when removing the robot product from the packaging. If not stably placed during transportation, vibration may occur, damaging internal components of the robot.

Equipment Storage

After transportation is completed, the original packaging should be properly stored in a dry place, with the environmental temperature maintained between -20 to +55°C, humidity ≤95%, and no condensation, for future repackaging and transportation needs. Do not stack other items on the original packaging box of the robotic arm to prevent deformation of the packaging box and damage to the robotic arm.

3. Maintenance and Care

As a robot manufacturer, we attach great importance to ensuring that customers can maintain and upgrade their robot equipment correctly and safely. To this end, we provide the following detailed maintenance and care guidelines, including common maintenance items and sections for repairing or upgrading hardware. Please read carefully:

Common Maintenance Items and Recommended Cycles

Maintenance Item	Description	Recommended Cycle
Visual Inspection	Check for any obvious damage, foreign object accumulation, or wear on the robot.	Daily
Structural Cleaning	Clean robot structural components with a clean, dry cloth. Avoid moisture and corrosive cleaners.	Daily
Fastener Inspection	Check and tighten all bolts and connections.	Daily
Lubrication	Lubricate joints and moving parts using manufacturer-recommended lubricants.	Every 3 months
Cable and Wiring Inspection	Inspect cables and wiring for damage or wear.	Monthly
Electrical Connection Check	Ensure all electrical connections are secure, free of corrosion or damage.	Monthly
Software Updates	Check and update control software and applications.	Whenever updates are available
Software Data Backup	Regularly backup critical software configurations and data.	Quarterly
Firmware Updates	Regularly check and update firmware for the latest features and security patches.	Whenever updates are available
Sensor and Device Inspection	Check sensors and other critical devices to ensure proper operation.	Monthly
Emergency Stop Function Test	Periodically test the emergency stop function to ensure its reliability.	Monthly
Environmental Conditions Monitoring	Monitor temperature, humidity, dust, etc., in the work environment to ensure compliance with robot operating specifications.	Continuous monitoring
Safety Configuration Review	Regularly check and confirm the robot's safety configuration, such as speed limits and workspace settings.	Monthly

Guidelines for Independent Changes to Robot Hardware

We understand that customers may have a need to independently upgrade or repair robot hardware. Before performing any upgrade operations, please read the relevant specifications of the product in detail and confirm with our official personnel whether such operations are allowed. Unauthorized operations may result in product failures and are not covered by the warranty.

Material Requirements

Officially manufactured or recommended materials: All parts and components required for maintenance and upgrades must be manufactured or explicitly recommended by us. This includes but is not limited to electronic components, sensors, motors, connectors, and any other replaceable parts.

Material Acquisition: Customers can purchase the required maintenance and upgrade materials through our official channels. This ensures the quality and compatibility of the components.

Repair or Upgrade Process

Customer self-repair: Customers are responsible for completing the repair work. We will provide detailed repair guidance and manuals to guide customers through the repair steps.

Follow official guidance: Repair operations should strictly follow the official guidance we provide. Any deviation from the official guidance may result in equipment damage.

Responsibility and Warranty Policy

- Responsibility division:

Manufacturer: Provides official guidance for repairs and upgrades, official manufacturing or recommended materials, and handles issues caused by manufacturing defects. Customer: Responsible for completing repairs according to official guidance and using official components.

- Warranty Policy:

Valid Warranty: The warranty is only valid if repair operations fully comply with our guidance and official components are used. Invalid Warranty: Any damage caused by the customer's failure to follow official guidance or by using non-official components for repair or upgrade will not be covered by the warranty.

Precautions

- Safety First: Before performing any repairs or upgrades, ensure that all safety guidelines are followed, including power-off and the use of appropriate protective equipment.
- Technical Support: If you encounter any issues during the repair process, it is recommended to stop the operation and contact our technical support team for assistance.

We strongly recommend that customers strictly follow these guidelines to ensure the safe and effective operation of robot equipment. Improper repair operations may result in equipment damage and affect warranty status. For further guidance or support, please contact our professional technical team promptly.

4. Common Questions

How to Ask Questions Gracefully

1. Phenomena Encountered in Various Places of Inquiry:

- No response after asking a question.
- It takes a long time for someone to respond to the question.
- The other party always belittles oneself.

2. Before Asking Questions, Please Make Sure You Have Studied This Manual.

Many questions can be resolved during this process. Avoid randomly asking questions in QQ groups, forums, issues, or emails, as many questions that are explained in the documentation from the beginning may not receive timely responses from the community. To save everyone's time and create a better community environment for mutual growth, please understand each other.

3. When Asking Questions, Try to Do the Following to Greatly Increase the Chance of Quickly Solving the Problem:

Understand What Happened, What I Did, Including:

- What effect do I want to achieve, what function?
- How did I do it to achieve this effect, what is the specific process?
- What errors occurred during the implementation, what symptoms (e.g., what is the reported error, the **complete** error message)?
- Did I carefully read the error message? Does the error message provide the cause of the error and the solution?
- Based on these error messages, think carefully, can I solve the problem?
- Search the documentation, issues, and use a search engine to find solutions to the problem?

4. If I Can't Solve the Problem Myself and Need to Ask for Help, I Need to Consider:

- Whom to ask, where to ask, who is more likely to answer my question? And how timely is it?
- What data and phenomena should I provide to him so that he can quickly help me solve the problem?
 - Provide my purpose (to let the answerer know what you are doing)
 - Provide the complete implementation process, as well as the phenomena encountered in the process (to facilitate the answerer to follow your process to reproduce the problem)
 - Point out the places where errors occurred, indicate where the phenomena or results differ from expectations! (Let the answerer know where it did not meet expectations)
 - Provide error information, need to be complete, as many screenshots and logs as possible, don't be stingy with screenshots, or only provide a part of the logs (because the answerer may not have done this for a long time and may have forgotten some details, need screenshots and complete logs to quickly recall; and according to the detailed logs, the problem can be quickly located)
- How to ask questions in a sincere manner, even if I am a newbie, everyone is willing to answer

5. Question Template

When asking questions, try to ask elegantly, without adding unnecessary particles, complaining words, carefully consider each word and punctuation mark, think from the perspective of the person answering the question, how to help the person answering the question quickly solve the problem, the description is not clear if the number of words is too small, if the number of words is too large, people will not have patience.

6. Title

No matter where you ask questions (including `qq group`), give yourself a title of about 30 words, clarify the central idea of the question, including:

- The category of the question, whether it is a question, a BUG submission, or an experience sharing, etc. Let everyone know immediately on the screen full of text what you want to do
- A sentence to clarify the central idea of the question, such as `Running the camera sample program, reporting reset fail, could it be a hardware problem`

So the title can be like this:

- `【mycobot Question】Running the camera sample program, reporting reset fail, could it be a hardware problem?`

Titles like these should be **avoided**:

- `Ah ah ah why is my board not working again`
- `Why can't my code run`
- `Why is my screen black`
- `【mycobot question】Received the development board, the screen is red, one line of small characters, why`
- `I ran xxx program, there was a problem`

You can ask like this:

- `【mycobot question】My board cannot start after I connect the power supply reversely. How to judge which part of the board is burned out? If possible, how can I revive it?`

7. Content

First of all, think from the perspective of the person answering the question:

- First of all, I need to know what the other party wants to do, what goals to achieve
- In order to achieve this goal, what steps did they take, and then at which step the problem occurred, so that I can try to reproduce the phenomenon based on their steps. If this problem seems difficult to solve and there are no steps to reproduce, it may take a lot of time to reproduce, so let's put it aside and solve other problems first
- Specifically, what problems occurred, if he only mentioned the problem, how do I know what problem he encountered? Maybe he was physically uncomfortable? So this is very important, let him describe the phenomenon when the problem occurred, and indicate what is different from the expectation, otherwise, I have to guess what is different from the expectation, the length of solving the problem increases
- When there is a problem, I may need his log file, so that I can analyze the source code according to the log, otherwise it may be difficult to solve the problem, then this problem can be put aside and solved slowly
- In summary, you can ask like this:
- Clearly state your goals, what you want to do, what the phenomenon should be like
- Did I refer to any documents, code, or tutorials
- How to reproduce the error: what specific steps, write in detail, until the problem occurs
- Describe in detail the phenomenon when the error occurred, and what is different from the expected, need to prove that the problem really occurred
- Attach log files, screenshots, or even videos, logs and screenshots must be complete, do not just take a small part, the answerer may find some problems that you did not notice from your complete logs and screenshots, this is very important
- In addition, pay attention to the format when pasting the code, don't paste it after it is messy and unreadable, try to copy it directly so that it can run
- Finally, express gratitude to the community friends who answered the questions

Driver Related

1. About Python

Q: What do the parameters in the API `send_coords([x,y,z,rx,ry,rz], speed, 1)` mean, and what are the corresponding Euler angles rx, ry, rz? What is the rotation order of Euler angles? And what are the ranges of each parameter?

- A: The parameters in the array are the end coordinates of the robot arm, speed is the speed, and the last parameter is the motion mode. rx, ry, rz correspond to rpy, which stands for roll, pitch, and yaw respectively. The Euler angle order is zyx, where zyx is the body coordinate. X, Y, Z range from -300 to 300.00 (the range is undefined, exceeding the range will return "inverse kinematics no solution" prompt), and rx, ry, rz range from -180 to 180.

2. About ROS

Q: How does the myCobot based on microcontroller and the myCobot based on microprocessor run ROS?

- A: The myCobot based on the microcontroller is currently running on Ubuntu, and you need to download a virtual machine to set up a virtual environment and install the Ubuntu system in the virtual environment to run ROS. You can also develop your own ROS. The myCobot based on the microprocessor comes with a ROS environment and can be used directly.

Can the myCobot based on the microprocessor be connected to a PC for use with ROS and MoveIt?

- A: There is currently no direct communication control in the open-source materials. You can achieve it by modifying existing node files through ROS + socket.

Q: Can you provide the rviz model file and programming examples?

- A: They are available on our GitHub. "https://github.com/elephantrobotics/mycobot_ros"

Q: When running the rviz model file with ROS, why does it report permission denied: '/dev/ttyUSB0'?

- A: Because the serial port permission is not granted. You should enter `sudo chmod 777 port name` in the terminal. For example:

```
sudo chmod 777 /dev/ttyUSB0
```

Q: When running the ROS slider control and model following commands, why does it report init() takes exactly 2 arguments(3 given)?

- A: The pymycobot library is not installed and started.

Q: When using ROS, why do the arm angles and model angles in rviz not match?

- A: It is likely that the zero position of the robot arm has not been calibrated, and you need to calibrate the zero position of the robot arm.

Software Issues

Q: Why can't my compiler find the corresponding device?

- A: You need to set up the development environment and install the corresponding project library before you can develop the device.

1. About myStudio

Q: What is myStudio?

- A: It is a self-developed software by our company. It is a tool for firmware burning or modification for the existing robotic arms launched by our company. [More](#)

Q: Can the drag and teach in the firmware record gripper actions?

- A: Currently, drag-and-teach cannot record gripper actions because the gripper belongs to joint 7, and our drag-and-teach can only record and play back the movements of joints 1-6.

Q: Even after burning the minirobot firmware, why can't I use drag and teach?

- A: First, check if both the M5Stack-basic firmware and the atom firmware are burned, if the burned firmware corresponds to the requirements and if it is the latest version of the firmware. It is recommended to burn the minirobot firmware to version v2.1, and the top atommain firmware to version v4.1 or above (requires support for mystudio version v4.3.1 or above).

Q: What version of firmware should be burned for the myCobot phone controller app?

- A: You need to burn the atom firmware atommain2.5 version in mystudio.

Q: Why does myStudio not recognize the serial port of myCobot? What should I do?

- A: If your computer does not prompt that myCobot is connected, install the serial port driver program first. Also, note that the Raspberry Pi, Arduino, and Jetson Nano series robotic arms **cannot be connected to a laptop** via a data cable and need to use mystudio for firmware burning in the built-in system.

Q: Can the trajectories recorded by drag-and-teach be stored in the card?

- A: Currently, it cannot be stored in the memory card. Also, drag-and-teach can only store one path at a time, and the next recording will overwrite the previous action.

2. About RoboFlow

Q: Can RobotStudio software programming be used?

- A: Our own industrial programming software roboFlow can be used, but RobotStudio is from ABB and cannot be interoperable with us.

Q: What is the reason for RoboFlow software to quickly move beyond the limit?

- A: It may be that one or more joints exceed the limit.

Q: How to load the written program in roboFlow software?

- A: After logging in, select the robot program, and then click Load Program. You cannot directly click Run Program, you can only click pro600.

3. About myblockly

Q: Why does a dialog box always appear when running myblockly?

- A: Before running myblockly, please close the serial port occupation.

Q: When running myblockly, why does it display child process exited with code 0?

- A: This is not an error. The real error needs to be analyzed one by one. This string represents the end of the program and returns binary number 0, indicating termination.

4. About ROS1

Q: When switching the terminal to `~/catkin_ws/src` and using git to install and update `mycobot_ros`, why does it seem that the target path "`mycobot_ros`" already exists?

- A: There is already a `mycobot_ros` package in `~/catkin_ws/src`, so you need to delete it first and then rerun git.

****Q: When running `roslaunch`, why does the terminal report 'could not open port /dev/ttyUSB0'?**

0: Permission: '/dev/ttyUSB0'?

- A: Insufficient serial port permission. Enter `sudo chmod 777 /dev/ttyUSB0` to grant permission.

Q: Why can't ROS programs run in vscode?

- A: Because the vscode terminal cannot load the ROS environment, you need to run it on the system terminal.

Q: When running `roslaunch`, why does the terminal prompt 'Unable to register with master node'?

[http://localhost:11311]: master may not be running yet. Will keep trying ?

- A: Before running ROS programs, you need to start the ROS node by entering `roscore` in the terminal.

Q: When running `roslaunch`, why does the terminal report 'could not open port /dev/ttyUSB0: No such file or directory: '/dev/ttyUSB1' ?

- A: The serial port is incorrect. Confirm the actual serial port of the current robotic arm. You can use `ls /dev/tty*` to check.

Q: When building code in Ubuntu18.04 with `catkin_make`, the build fails with errors such as 'Project 'cv_bridge' specifies '/usr/include/opencv' as an include dir, which is not found. What is the reason?

- A: The opencv path in the configuration file does not match the actual path of the system. You need to use `sudo` to modify the configuration file (the path is `/opt/ros/melodic/share/cv_bridge/cmake/cv_bridgeConfig.cmake`), and the actual opencv path of the system is located in `/usr/include/`.

Q: After compilation, when running launch instructions in a new terminal, why does it show the following error?

```
u20@u20-VirtualBox:~/catkin_ws$ roslaunch mybuddy_socket slider_control.launch
RLEException: [slider_control.launch] is neither a launch file in package [mybuddy_socket] nor is [mybuddy_socket] a launch file name
The traceback for the exception was written to the log file
```

- A1: The ROS environment variables are not added to the system, so you need to source them every time you open a new terminal:

```
cd ~/catkin_ws/
source devel/setup.bash
```

- A2: The ROS environment variables are added to the system, so you don't need to execute source every time you open a new terminal:

```
# The noetic is Ubuntu20.04 system
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

- A3: It may be that the file name in the instruction does not match the actual file name in the mycobot_ros package. Please carefully check if there is any error in the instruction.
-

Hardware Issues

Q: What controls the six servos?

A2: The servos are controlled by the atom on the top.

Q: What is the role of the atom in the robotic arm?

A3: It controls the robot through forward and inverse kinematics and coordinate transformation algorithms. The Atom is not open source yet.

Communication-related Issues

Q: Why, after connecting the robotic arm with an HDMI cable, does the screen not display? Is it necessary to download port driver program?

A1: Check if the connection is correct and if the power is turned on. Try using other interfaces and insert them steadily. There is no need to download the port driver program.

Q: Which versions of communication interfaces are supported by different robotic arms?

A2: The robotic arms based on microprocessors support socket communication TCP; the robotic arms based on microcontrollers can communicate via USB to serial port.

Q: What is the communication frequency of the robotic arm?

A3: 10-20Hz.

Parameter-related Issues

Q: What is the speed unit of the robotic arm?

A2: 180°/s.

End Effector Issues

Q1: Can the gripper not be completely closed?

A1: The gripper itself has a certain gap and cannot be completely closed. You can adjust it by increasing the thickness of the spacer.

Q2: What communication does the gripper use?

A2: It is TTL communication.

Other Issues

Q: Why does the motor power turn off during use?

A1: Because the motor temperature is too high to use. Please wait a few minutes before using it again.

Q: Does the robotic arm support Android-based development?

A2: We do not support Android development. If you want to develop it yourself, we can provide you with the port protocol.

If you have read all of this chapter, please continue to the next chapter.

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

1 List of product standards

1.1 Product List Pictures

Information is being updated...

1.2 Product Standards List Comparison Table

Serial Number	Product
1	myArm Master 750 robotic Arm
2	Product kit with USB-Type C, jumper cables, M6*35, cup hex head, fully threaded stainless steel screws
3	DC24V DC power supply
4	Fixed base plate
5	G-type nipple
6	Certificate of Conformity
7	emergency stop button

Note: Once the box is in place, please make sure the robot packaging is intact. In case of damage, please contact the logistics company and the supplier in your area. After opening the box, please check the actual contents of the box against the list of items.

2 Product Unboxing Guide

2.1 Product Unboxing Graphic Guide

Why you need to follow the steps to disassemble the product

In this section, we strongly recommend following the specified steps for disassembling the product. This not only helps to ensure that the product is not damaged during transport, but also minimises the risk of accidental failure. Please read the following illustrated guide carefully to ensure the safety of your product during unpacking.

- **1** Check the packing box for damage. In case of damage, please contact the logistics company and the supplier in your area.
- **2** Open the box and take out the product brochure, sponge packaging cover, myCobot robot arm, matching power supply, emergency stop switch, flat base and accessory bag.
- **3** Make sure each step is completed before moving on to the next to prevent unnecessary damage or omissions.

Note: After removing the product, please check the appearance of each item carefully. Please check the actual items in the box against the list of items.

2.2 Product unboxing video guide

Information is being updated...

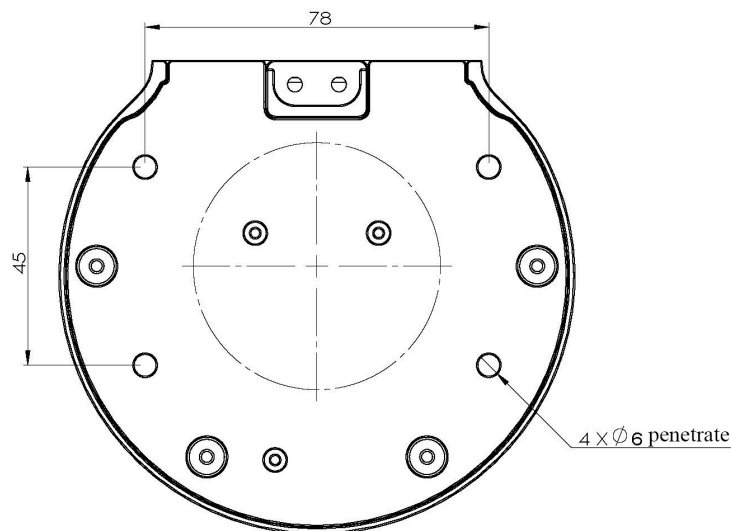
3 Structural installation and fixing

myArm weighs 3.2kg. Since the centre of gravity changes as the robot moves during use, it is required that the robot be fixed on a sturdy base from the outset. Either a fixed base or a mobile base is possible.

3.1 graphic guide

Basic Interface Dimensions

- The base mounting holes are the interface between the robot and other bases or planes. The holes are shown below. There are 4 countersunk holes with a diameter of 4.5 mm that can be secured with M6 bolts.



- The end mounts have flanges that are compatible with LEGO component holes and threaded holes. Ensure the appropriate threaded holes are present in the fixing base before installation.
-

Before installing, please make sure:

- Environmental conditions meet the requirements listed in section 2.2.4.3.1 above.
- The mounting position is not smaller than the working range of the robot and has sufficient space for installation, use, maintenance and repair.
- Place the base in a suitable position.
- Prepare tools related to installation, such as screws and spanners.
- After confirming the above steps, move the robot to the base mounting table, adjust the robot position, and align the fixing holes in the robot base with the holes in the base mounting table. After aligning the holes, align the screws with the holes and tighten them.

CAUTION: When adjusting the robot's position on the base mounting table, do not push or pull the robot directly on the base mounting table as this may cause scratches. When manually moving the robot, do not apply external force to the vulnerable parts of the body to avoid unnecessary damage to the robot.

- Once the robot is installed, it is ready for cable connection and use.

3.2 Video guide

For more installation details, scan the code to watch the video:

Information is being updated...

4 External cable connection

4.1 graphic guide

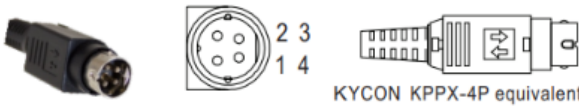
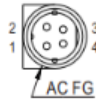
Before operation, make sure you have read **Chapter 3 Safety Instructions** to ensure safe operation. Meanwhile, connect the power adapter to the robot arm and fix the robot arm base on the table.

The myArm must be powered by an external power supply to provide adequate power:

- Rated Voltage: 24V
- Rated current: 9.4A
- Plug Type: R7B

■ Dc输出插头

● 标准插头: R7B

R7B		Pin脚定义		
	KYCON KPPX-4P equivalent	 外壳连接ACFG -V不与ACFG连接	PIN脚编号	输出
			1	+Vo
			2	-Vo
			3	-Vo
			4	+Vo

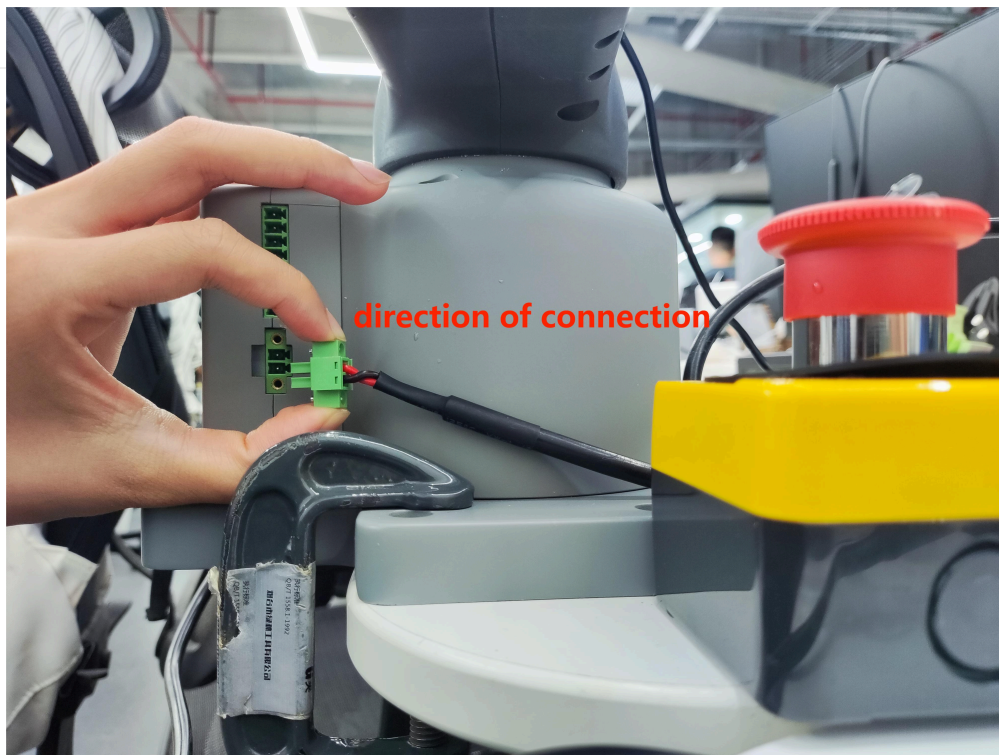
Please note that it is not possible to use only the plug-in TypeC power supply. Please use an officially approved power supply to avoid damaging the arm.

The use case diagram is shown below: (Please align the use case diagram carefully for connection)

Step 1:



Step 2:



Step 3:



Step 4:



Step 5:



Step 6:



5 Boot status display

5.1 Graphic guidance

5.1.1 Switching on the power supply

Ensure that the power adapter, emergency stop switch and HDMI monitor are connected and press the power switch **Start button (round)**.

5.1.2 Machine Startup

During startup, the screen appears as a delayed wait screen.

5.1.3 Status Display

The display shows information about the system desktop.



If you have read all of this chapter, please continue to read the next chapter.

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

Basic functions

This chapter explains the basic functional usage of the product and the use of basic software. This chapter is critical and should be read carefully. Before actually applying the robot, make sure that the operations described are properly understood.

- [5.1 Minirobot](#)

The MyArm C650 version comes with the miniRoboflow application software developed independently by Elephant Robotics. Functions such as drag-and-drop teaching, communication forwarding, status checking and zero calibration can be achieved through simple push-button interaction. The user interface of the built-in software is simple and easy to use. The communication forwarding function allows you to control the robot using a variety of programming languages and development environments, making robot control easy.

- [5.2 Instructions for use of the software](#)

myStudio is a one-stop platform for using robots such as myRobot/myCobot.

It is convenient for users to select different firmware and download it according to their own usage scenarios, while learning relevant teaching materials and browsing tutorial videos online.

- [5.3 Firmware Version Description](#)

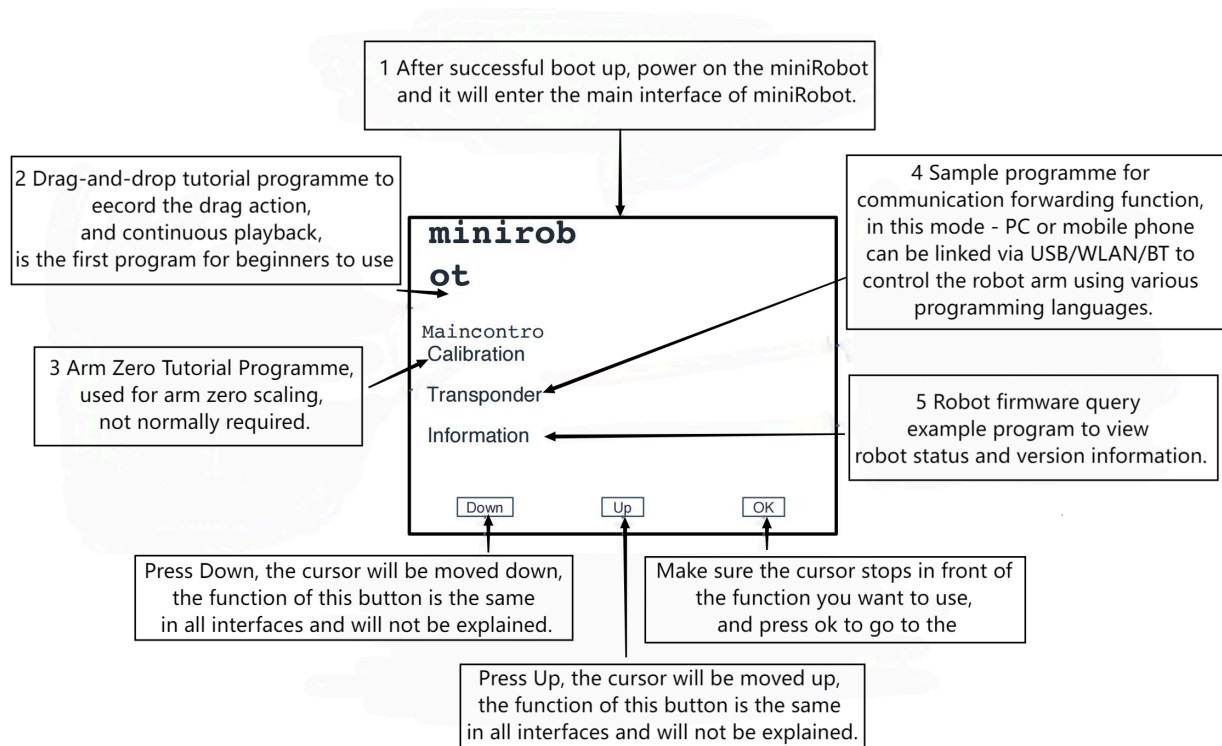
A table of firmware update dates is provided, illustrating the effects of firmware changes that have improved the results.

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

Factory Firmware Introduction

The MyArm M750 version comes with the miniRoboFlow application software developed independently by Elephant Robotics. Functions such as drag-and-drop teaching, communication forwarding, status checking and zero calibration can be achieved through simple push-button interaction.

The user interface of the built-in software is simple and easy to use. The communication forwarding function allows you to control the robot using a variety of programming languages and development environments, making robot control easy.



miniRoboFlow

miniRoboFlow It has six main chapters:

- **User Guide**
 - Minirobot is an application for controlling the MyArm robotic arm by combining the 3 buttons on the base of the robotic arm and the on-screen display. (It has functions such as drag and drop demonstration, communication forwarding, zero calibration, status messages, etc.) to facilitate the operator's interaction with and proper use of the Elephant Robot.
- **Drag-and-drop teaching**
 - Robot drag-and-drop teaching: the operator can directly drag the robot joints to make the desired posture, and then save the movement in the robot by **button** operation. cobot is an early system with this function. This type of teaching method avoids the disadvantages of traditional teaching and has a broad development prospect. This teaching method avoids the disadvantages of traditional teaching methods and is a forward-looking technology for robot applications.
- **Zero Calibration**

- Calibrating the robot arm is a prerequisite for precise control of the robot arm, while setting the joint zeros and initialising the motor potentials are basic tasks for subsequent advanced development.
 - **Communication Transponder**
 - For microcontroller robotic arms, the timeliness of communication is critical. For this type of robotic arm, we often send control commands to the **M5Stack-basic** at the bottom. Through communication forwarding, the end-effector analyses the commands and then executes the target action. Currently, the **MyArm M750** has three communication methods: **Serial, Bluetooth, and WIFI**.
 - **Status Information**
 - Link Test is a detection function that utilises the connection status of the motors in the robot arm and the **Atom**. This function allows the user to easily troubleshoot the device. During the link test, the connection status of the robotic arm device can be seen, including the connection of the **Server** and the communication status of the **Atom**. In microcontroller devices, M5Stack-basic displays its current firmware version.
 - **Burn Function**
 - Describes how to use the built-in software's burn function, which is mainly used to update the firmware version.
-

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

Minirobot User Guide

1 Introduction to Minirobot

Minirobot is an application that controls the MyArm robot arm by combining the 3 buttons of the robotic arm base with an on-screen display. Its functions include drag teaching, communication forwarding, zero calibration, status information, etc.), which is convenient for operators to interact with the elephant robot and use the elephant robot correctly.

Applicable Equipment:

-MyArm M750

Be sure to download and update the Atom, Pico and Basic firmware before use

2 How to use

After you have updated the basic firmware, plug it in, press the power switch and the Minirobot will start. The boot will automatically run the POST, and Robot Checking will be displayed on the screen, wait for about 20 seconds before entering the main interface. The main interface displays Maincontrol, Transponder, Calibration, Information and Flash, and you can control the selection through the three buttons at the bottom, which are up, down, and ok. If you're going to use Transponder, connect Type-C to your device first. (When using the key, if there is no response, please press and pause for a while and then release to ensure that the key is pressed) Before the motion control, you need to confirm whether the zero position is normal, relax the robotic arm and move it to the theoretical zero position, and use the communication protocol to read the angle value through Transponder, if the zero deviation is too large, you need to use calibration to calibrate the zero position of each joint first, and use test to check whether the zero position is normal after the calibration is completed. **Note: If there is an abnormal motor communication during the POST, the end LED will be red, and the end LED will be green under normal circumstances. When there is a communication abnormality, the motion control cannot be controlled, at this time, it is necessary to call the clear_servo_error in the communication protocol through the serial port, and ignore the abnormality to move**

2.1 Drag teach-in

Drag the show to control the movement of the robot arm according to the trajectory you drag while recording.

2.2 Communication Forwarding

Enter the communication forwarding interface, you can control the robotic arm by sending commands in the communication protocol through the serial port assistant on your device.

2.3 Zero calibration

The zero calibration interface allows you to calibrate the joint zero and test the zero position.

2.4 Status Information

The status information interface allows you to view the motor connection status and firmware version number.

2.5 Burning

When you need to burn the PICO firmware of the main controller, go to the programming interface and select Flash to enter the programming preparation stage.

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

Drag the teach-in

Robot drag teaching means that the operator can directly drag the joints of the robot to make an ideal posture, and then record accordingly.

Cobot was one of the earliest systems to have this feature. This teaching method avoids the disadvantages of traditional teaching and is a forward-looking technology for robot application.

In this chapter, we'll teach you how to get started with it and experience the fun of cobot dragging.

The approximate steps for them are as follows: **Step 1:** Select Maincontrol and click OK to enter the drag teaching interface.



Step 2: Press the Record button.



Step 3: Select the storage path and press Ram.



Step 4: Drag the joints of the robotic arm to move to your desired position to complete a set of movements.



Step 5: Press any key to stop recording and save the record.

Step 6: Press the play button Play.



Step 7: Press Ram and the robotic arm will start playing the saved recording.



Step 8: Press Pause to pause, Stop to stop, Play to resume.

Step 9: Click Exit to exit this feature.

Zero calibration

This was done before the default bot was delivered. There is no need to repeat the operation. Using this feature incorrectly may damage the bot. If your robot is working without abnormality, do not use it. Thank you for your cooperation.

Calibrating the robotic arm is the prerequisite for precise control of the robotic arm, while setting the joint zero point and initializing the motor potential are the basic tasks of subsequent advanced development.

In this chapter, we'll teach you how to calibrate robotic arms as well as test and validate joints. **Step 1:** Select Calibration, click OK to enter the zero calibration interface.



Step 2: Press the A button to start calibrating the robotic arm.



Step 3: First drag the robotic arm to bring the No. 1 joint to the zero position (zero tick alignment).



Step 4: Follow the joint serial number (1-7) as prompted on the screen, drag the robotic arm to make each joint reach the zero position (zero tick alignment). **Step 5:** Press NEXT in turn to enter the next motor calibration until

Already Calibrate all appears!! , that is, the calibration is complete.



Step 6: Press EXIT to exit the calibration.

Step 7: Press the B button to test the zero position of each joint of the robotic arm, and the robotic arm will perform a small movement near the zero position of each joint. Only after the zero calibration has been completed can the joints be tested with the test.





Step 8: Press the C key to exit this function.

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

Communication forwarding

This chapter describes how to use the extended development capabilities of the built-in software. This feature allows you to expand the connectivity options of your devices and supports different development environments and methods.

Currently, there are three connection methods supported for communication forwarding: USB, Wi-Fi, and Bluetooth. They are suitable for different functional development scenarios.

- USB and Wi-Fi are used to connect to other master devices, and the robot can be controlled to execute the corresponding commands through the robot communication protocol.
- Bluetooth is used to connect to mobile control software, which can be controlled quickly by mobile phone.
- **Step 1:** Confirm that Type-C is correctly connected to your device, select Transponder and click OK to enter the communication forwarding interface.





Step 2: Use the serial port connection, select the USB UART and click OK to enter the serial port interface. The serial interface detects the connection of the Atom (ok indicates that the connection is normal, otherwise no is displayed).



Step 3: To use the WiFi connection, you need to set the WiFi name to: MyCobotWiFi2.4G and the password to: mycobot123. Select WiFi and click OK to automatically search for WiFi and connect.







Step 4: Click Exit to exit this feature.

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

Status information

This chapter describes how to check the status of a device using basic software. Please make sure that both the software version and firmware version of the device remain unchanged (factory default version) before use. If you make any modifications to the software or firmware, ensure that your operation meets the requirements listed in this chapter.

At present, the information that can be viewed is divided into motor connection status and the built-in firmware version number of the robotic arm.

Step 1: Select Information and click OK to enter the status information interface.





Step 2: Press the A key to start connection detection. The screen shows the connection status of the Atom and the motor.



Step 3: Press the B key to start detecting the version information, and the screen will display the firmware version.





Step 4: Press the C key to exit this function.

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

Burning

This chapter describes how to use the programming function of the built-in software, which is mainly used to update the firmware version.

Step 1: Select Flash and click OK to enter the burning interface.



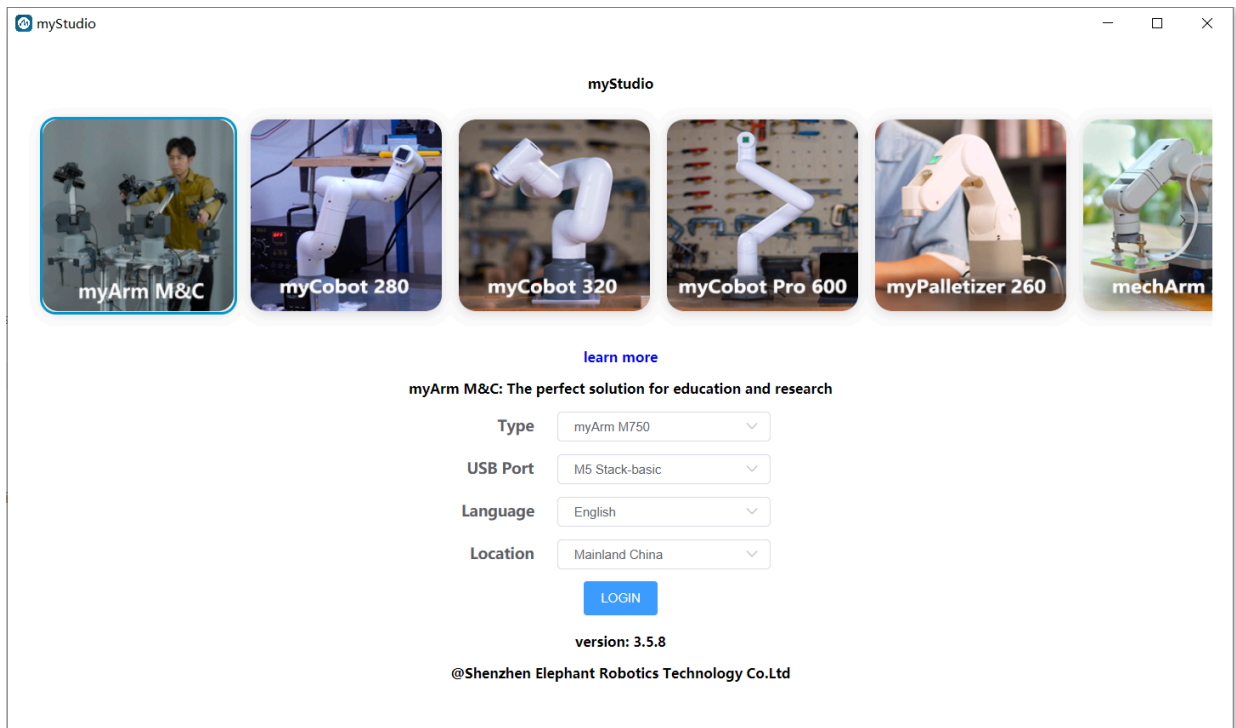


Step 2: Press the A button to prepare for burning. The machine will reboot within a period of time, during which the pico firmware will be burned, note that the pico firmware will start when the flash is pressed.

Step 3: Press the C key to exit this function.

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

myStudio



1 The original design intention of myStudio

- myStudio is a one-stop platform for using robots such as myRobot/myCobot.
- It is convenient for users to select different firmware and download it according to their own usage scenarios, while learning relevant teaching materials and browsing tutorial videos online.

2 myStudio latest version and supported platforms

- Latest version: V3.5.8
- Available on: Windows, Mac, Linux

3 myStudio function

- Burn and update firmware
- Provide robot usage tutorials, such as user manuals, video tutorials, Q&A, etc.
- Information on maintenance and repairs

Jump to sections:

- [myStudio environment setup](#)
- [Install driver](#)
- [Burn and update firmware](#)

1 myStudio Environmental construction

myStudio Download

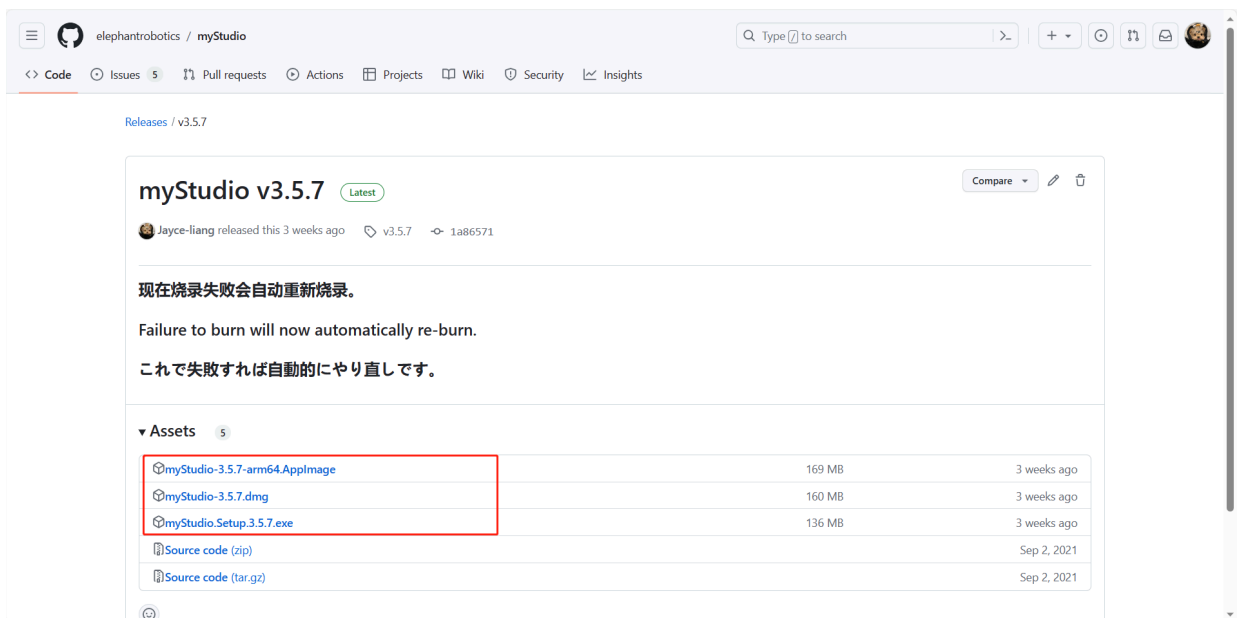
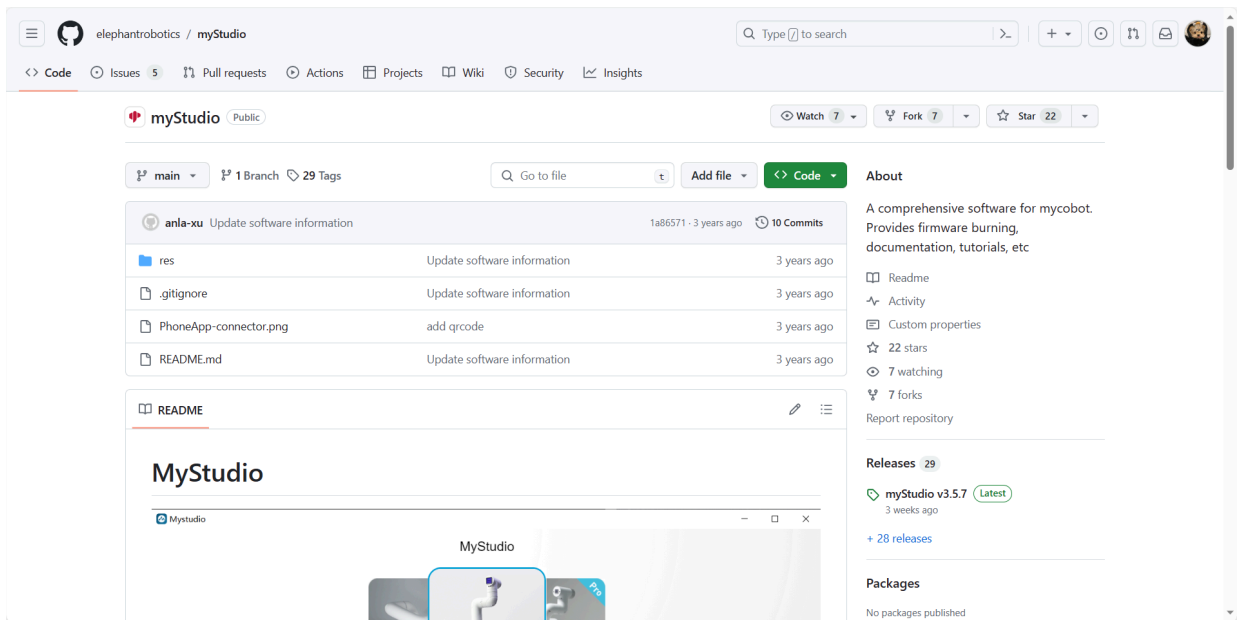
Note: The installation path of MyStudio should not have any space during installation

download link:

1. GitHub

-After entering the download address, click on the right `myStudio`, select the corresponding version to download.

Take myStudio v3.5.7 as an example



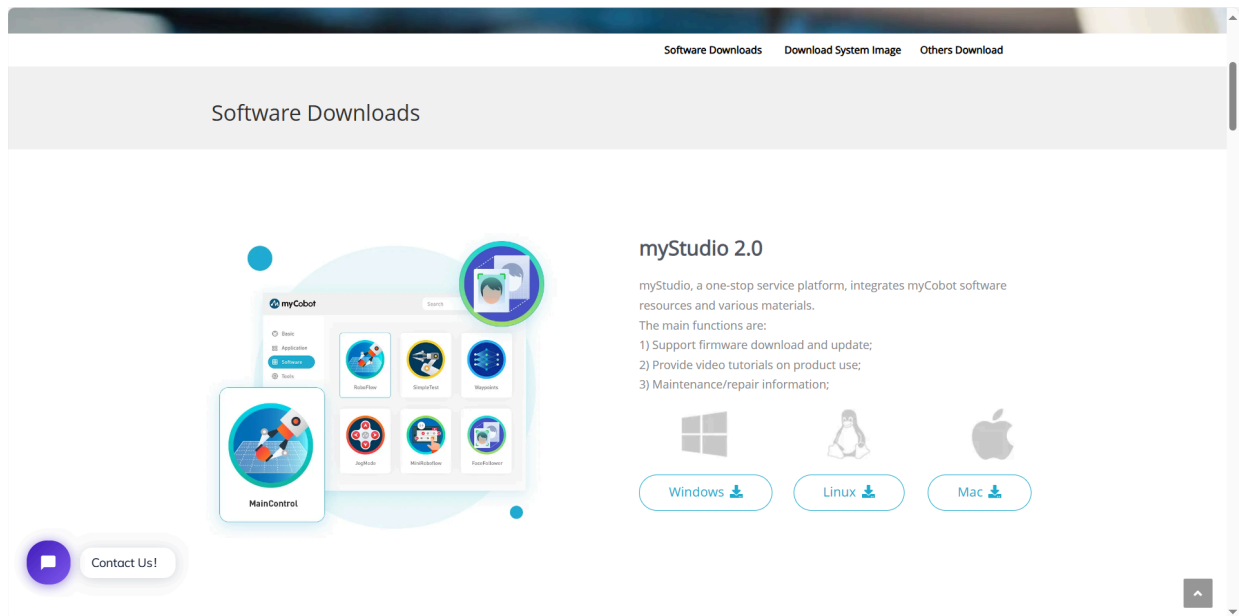
-Different suffixes represent different systems, please download the corresponding version:

- *. APPIMAGE -Linux system
- *. DMG -MAC system

- *. Exe -Window system

2. Official website address

According to the computer system, select `myStudio 2.0`.



Install

Install mystudio for Linux system

Download the `myStudio` version of the linux version from the official website to get a installation package as shown below



Select the `myStudio-Latest.appimage` Right-click on the mouse, click `properTies` to open





myStudio Application

Open

Open With Other Application...

Cut

Copy

Make Link

Rename...

Copy to

Move to

Move to Trash

Send to...

File's Emblem

Edit as Administrator

Encrypt...

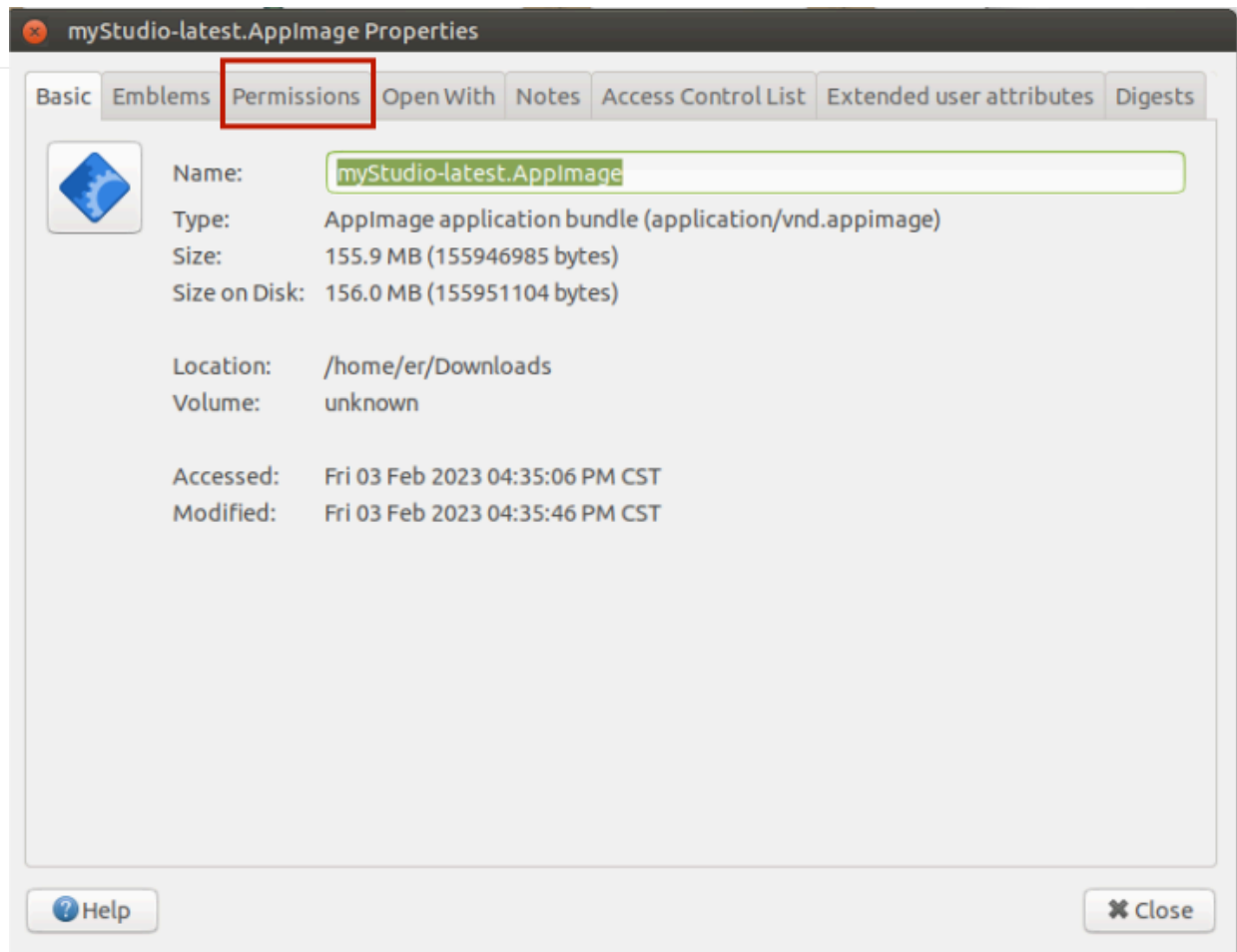
Sign

Compress...

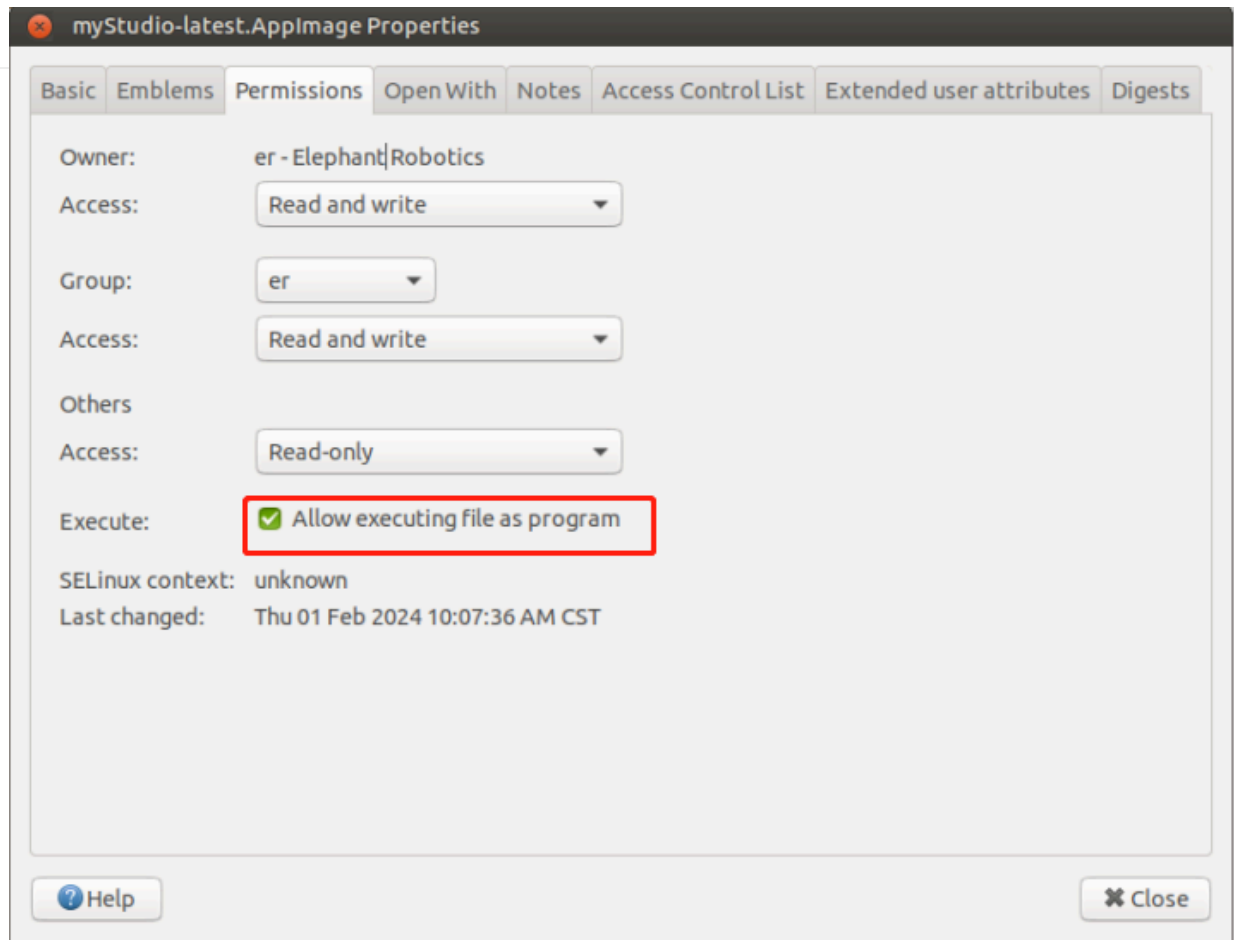
Properties

click enter

Permissions



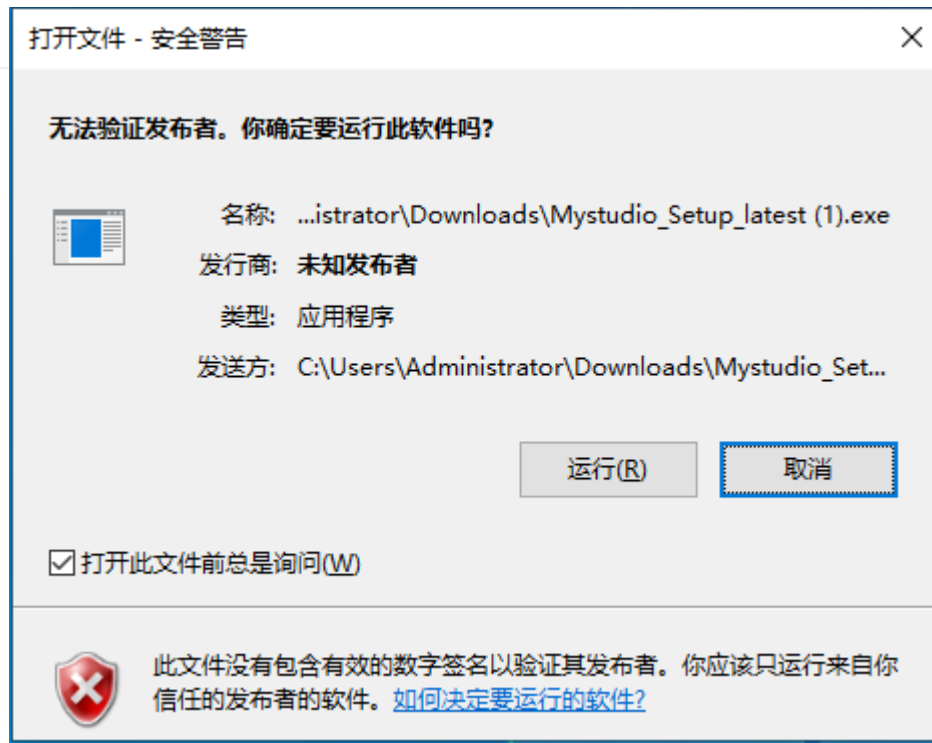
Permissions page checks Allow Executing File as Program , then click the CLOSE button to close the pop -up window



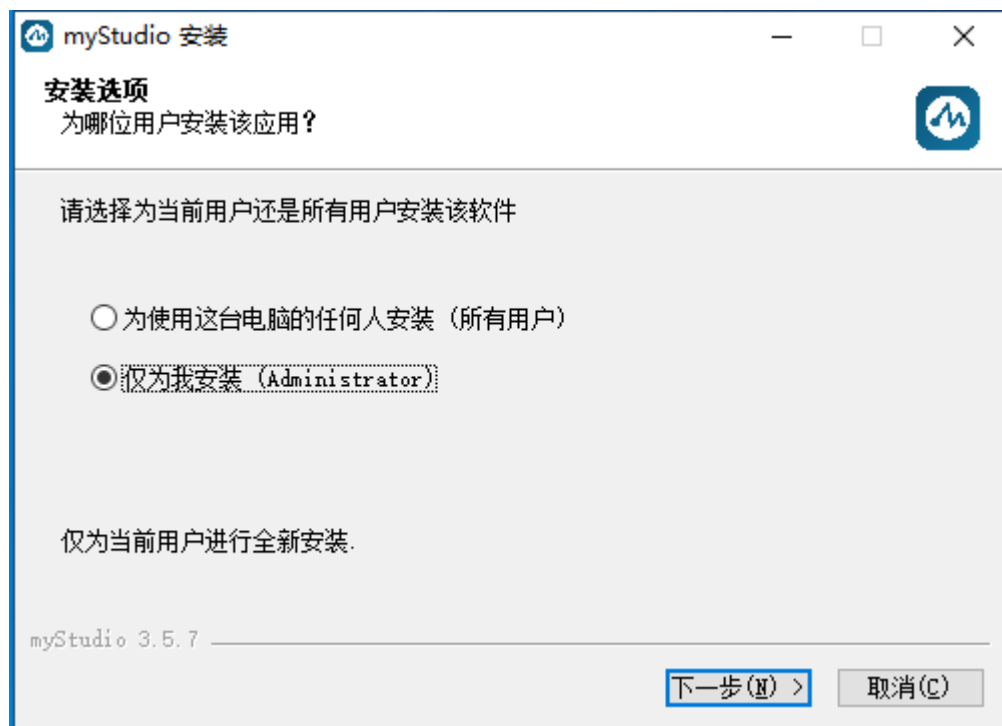
After the pop-up window is turned off, double-click the mouse to install the package `myStudio-latest.appimage` to open myStudio

Install myStudio for Windows system

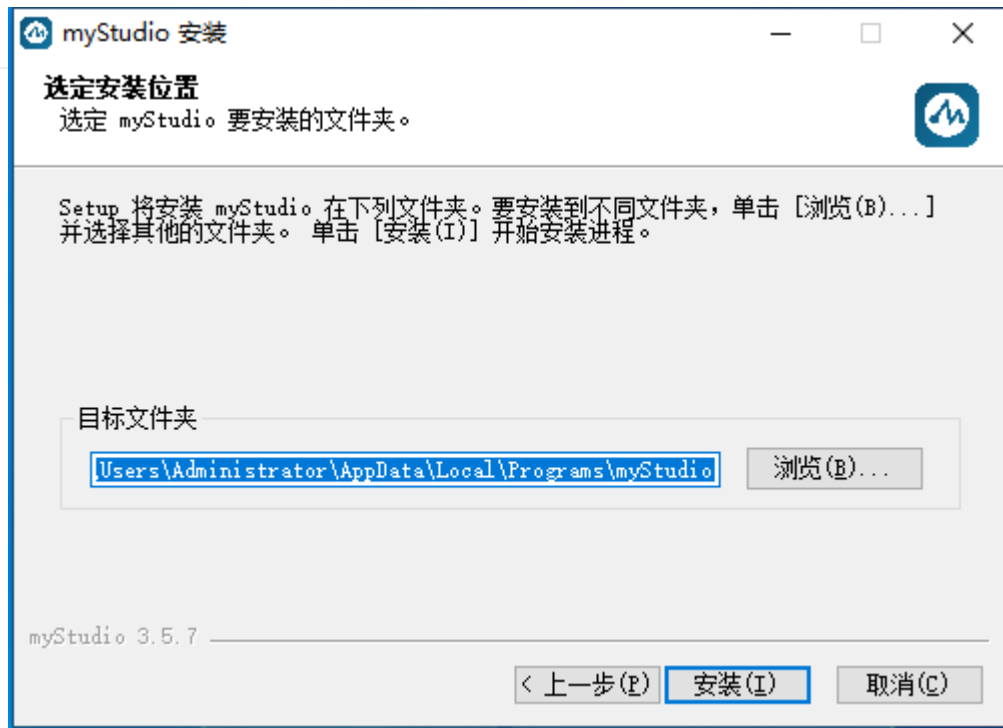
Double -click to open the file named `Mystudio_Setup_latest.exe` , and then click `Run`



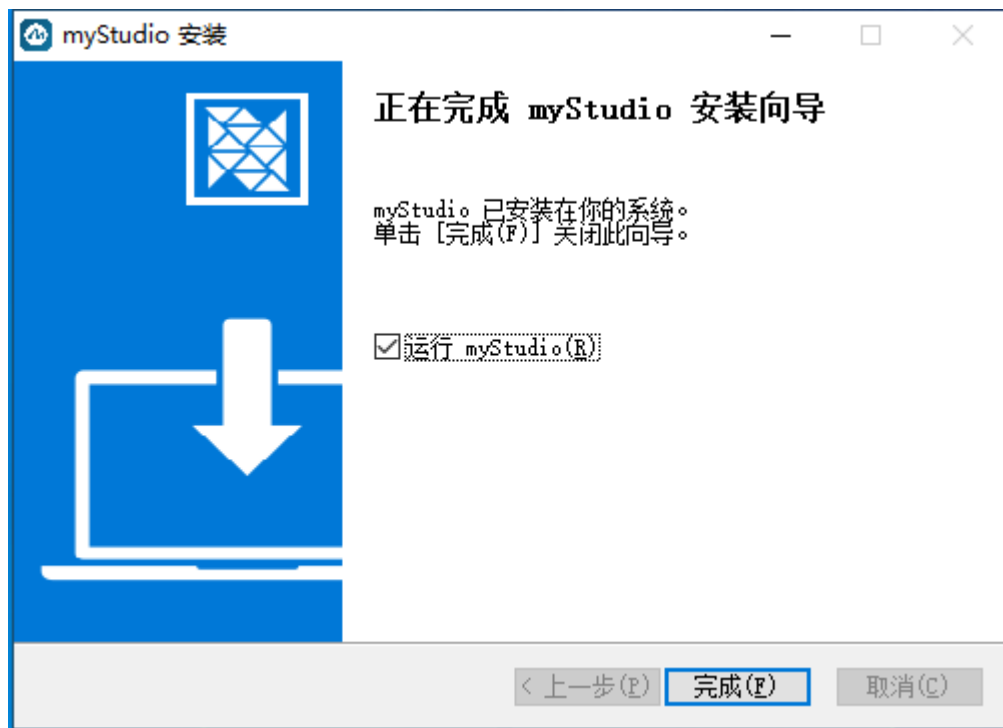
click **Next**

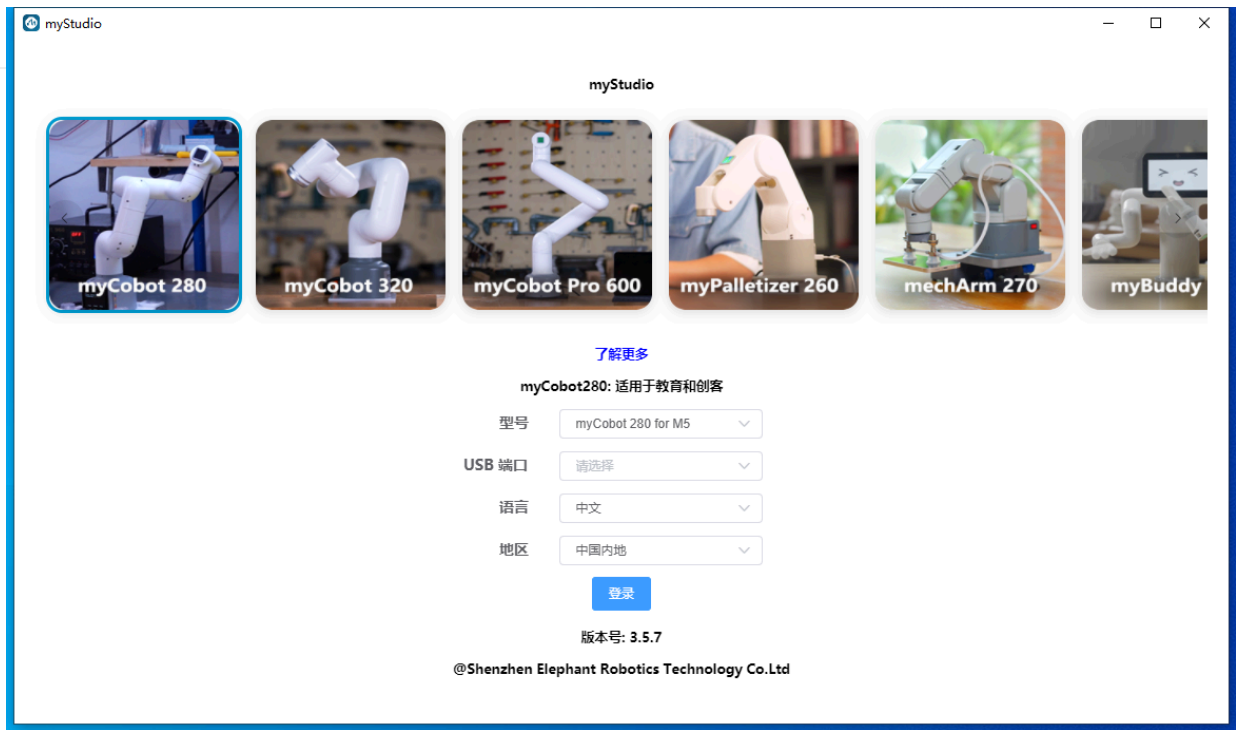


After clicking the install, wait for myStudio installation to complete



After the installation is complete, click the `button to open and run myStudio





Install myStudio on Mac

Download the Mac version of myStudio from the official website to get an installation package as shown below. Double-click to open it.



Note: For MacOS, make sure system "Preferences->Security & Privacy->General" and Allow Apps from App Store and Recognized Developers are enabled before installing.

Uninstall

For Linux systems uninstall myStudio

Just delete the installation package directly

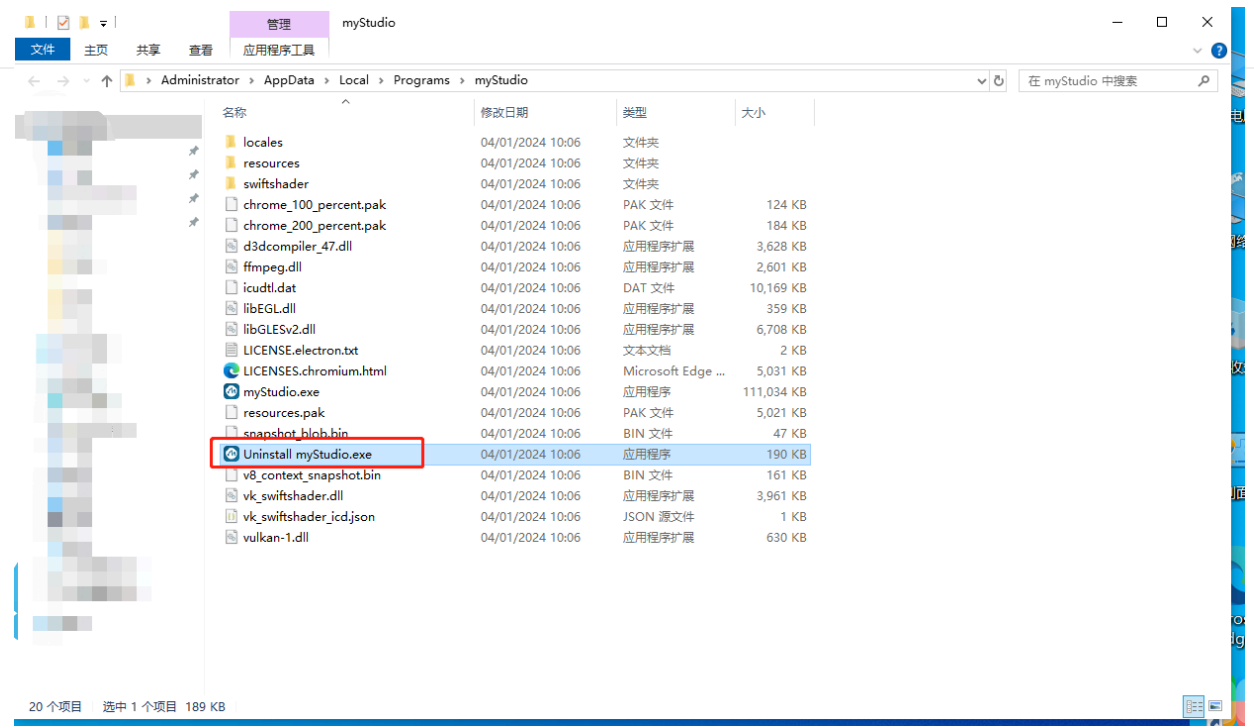
The default name of the installation package is `myStudio-latest.AppImage`

Uninstall myStudio for Mac systems

Just move myStudio to the Trash in the application

Uninstall myStudio for Windows systems

Enter the file directory of myStudio and click to run `Uninstall myStudio.exe`



Click **Next**



myblockly has been uninstalled, click **Finish** to exit



Update

In myStudio you can click the **Update** button to update



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

Install driver

Users can click the button below to download the corresponding **CP210X** or **CP34X** driver compressed package according to the operating system they are using. After decompressing the compressed package, select the installation package corresponding to the bit number of the operating system to install.

There are currently two driver chip versions, **CP210X** (applicable to CP2104 version) and **CP34X** (applicable to CH9102 version) driver compressed package. If you are not sure which USB chip your device uses, you can install both drivers at the same time. (**CH9102_VCP_SER_MacOS** An error may occur during the installation process, but it has actually been installed. Just ignore it.)

For Mac OS, make sure your system "Preferences -> Security & Privacy -> General" and allow access from the App Store and Approved Developers before installation.

- Download the **M5Stack-basic** serial port driver at the bottom

- **CP210X**

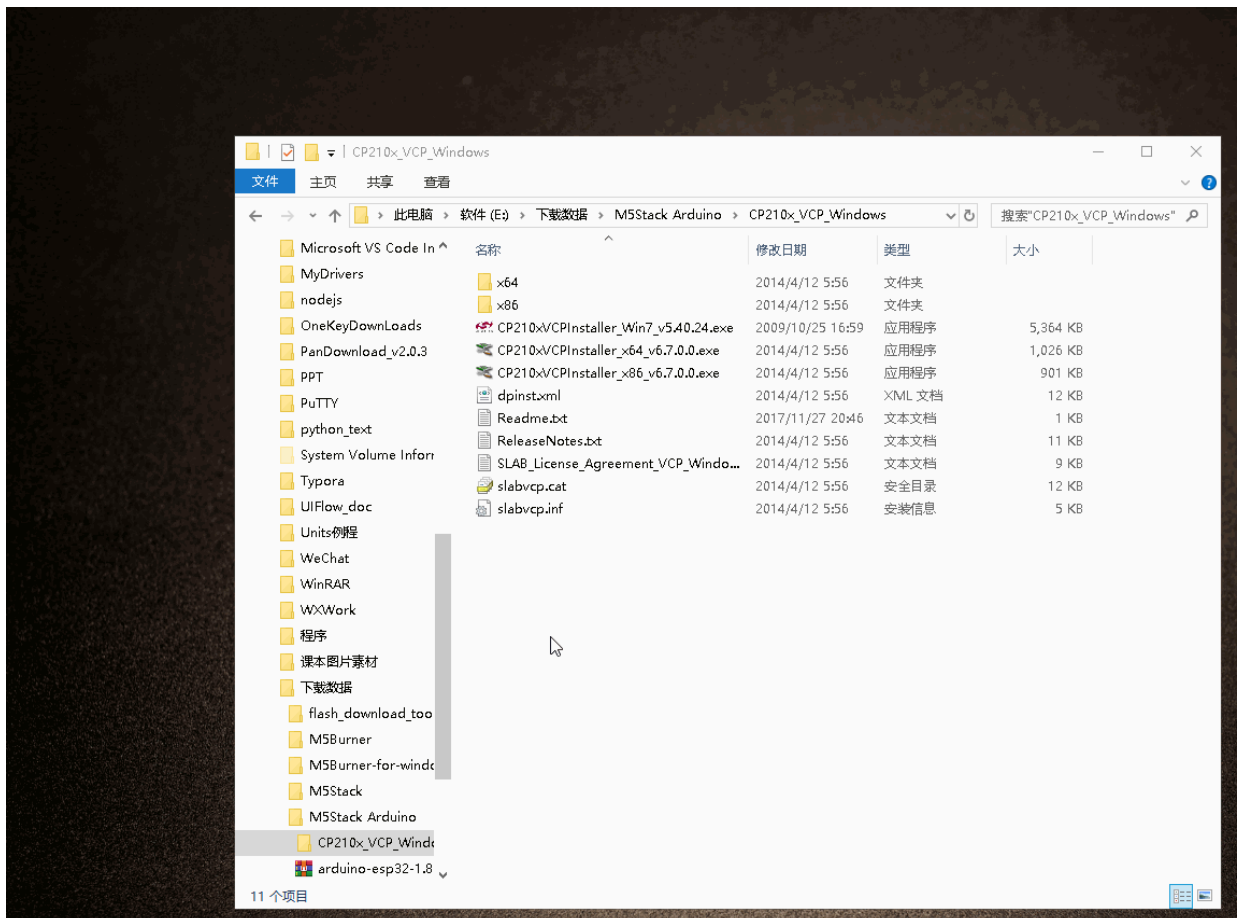
- [Windows10](#)
- [MacOS](#)
- [Linux](#)

- **CP34X**

- [Windows10](#)
- [MacOS](#)

-Finally download the **Atom** serial port driver

- [Windows10](#)

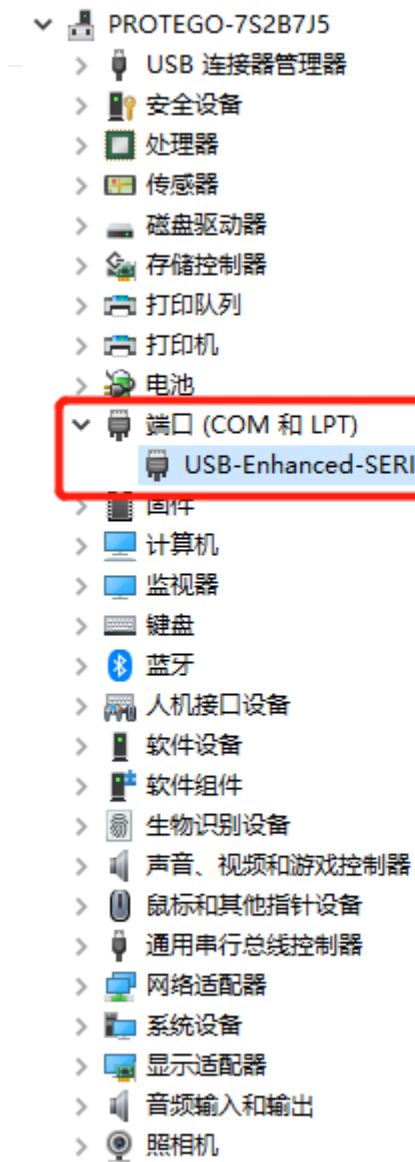


How to distinguish CP210X and CP34X chips

- As shown in the figure below, open **Device Manager** and view **Ports (COM and LPT)**



- If **Port (COM and LPT)** displays **USB-Enhanced-SERIAL CH9102**, it is **CP34X** chip



- If Port (COM and LPT) shows **Silicon Labs CP210x USB to UART Bridge**, it is **CP210X** chip

- ▼  PROTEGO-7S2B7J5
 - >  USB 连接器管理器
 - >  安全设备
 - >  处理器
 - >  传感器
 - >  磁盘驱动器
 - >  存储控制器
 - >  打印队列
 - >  打印机
 - >  电池
 - ▼  端口 (COM 和 LPT)
 -  Silicon Labs CP210x USB to UART Bridge (COM6)
 - >  鼠标
 - >  计算机
 - >  监视器
 - >  键盘
 - >  蓝牙
 - >  人机接口设备
 - >  软件设备
 - >  软件组件
 - >  生物识别设备
 - >  声音、视频和游戏控制器
 - >  鼠标和其他指针设备
 - >  通用串行总线控制器
 - >  网络适配器
 - >  系统设备
 - >  显示适配器
 - >  音频输入和输出
 - >  照相机

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

Burn and update firmware

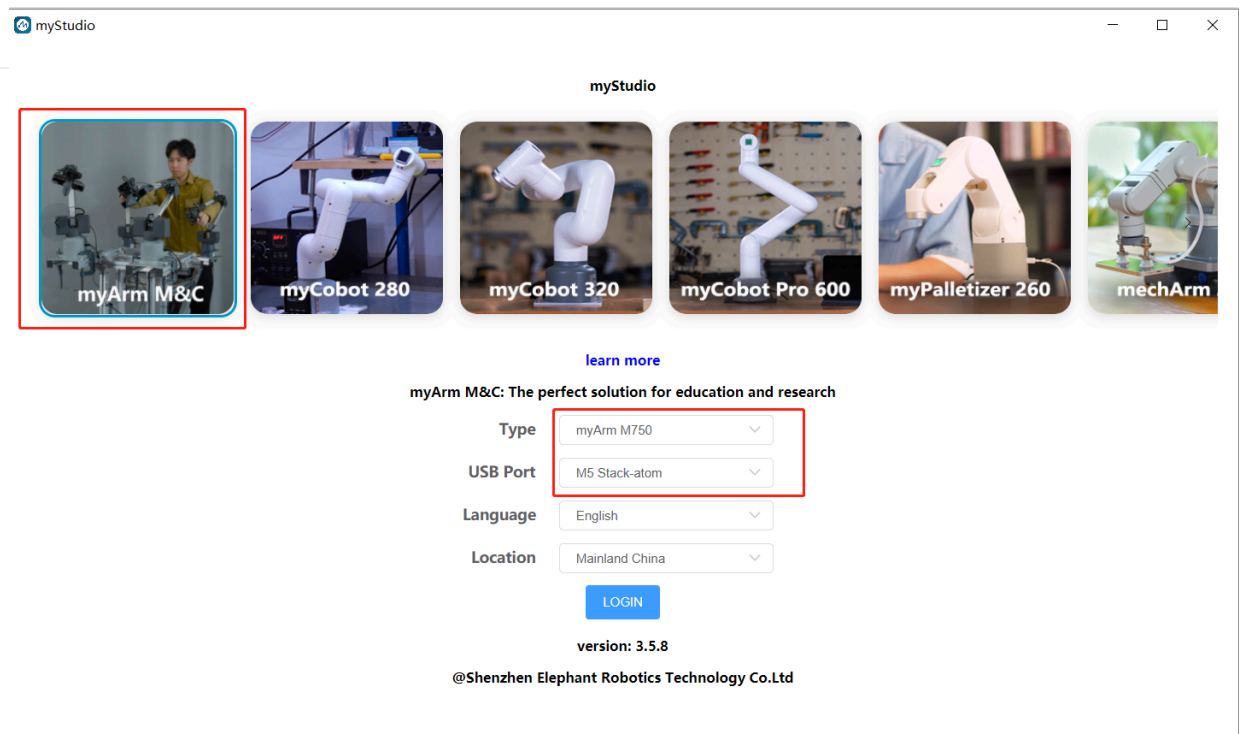
[myStudio video tutorial](#)

Burn Atom firmware

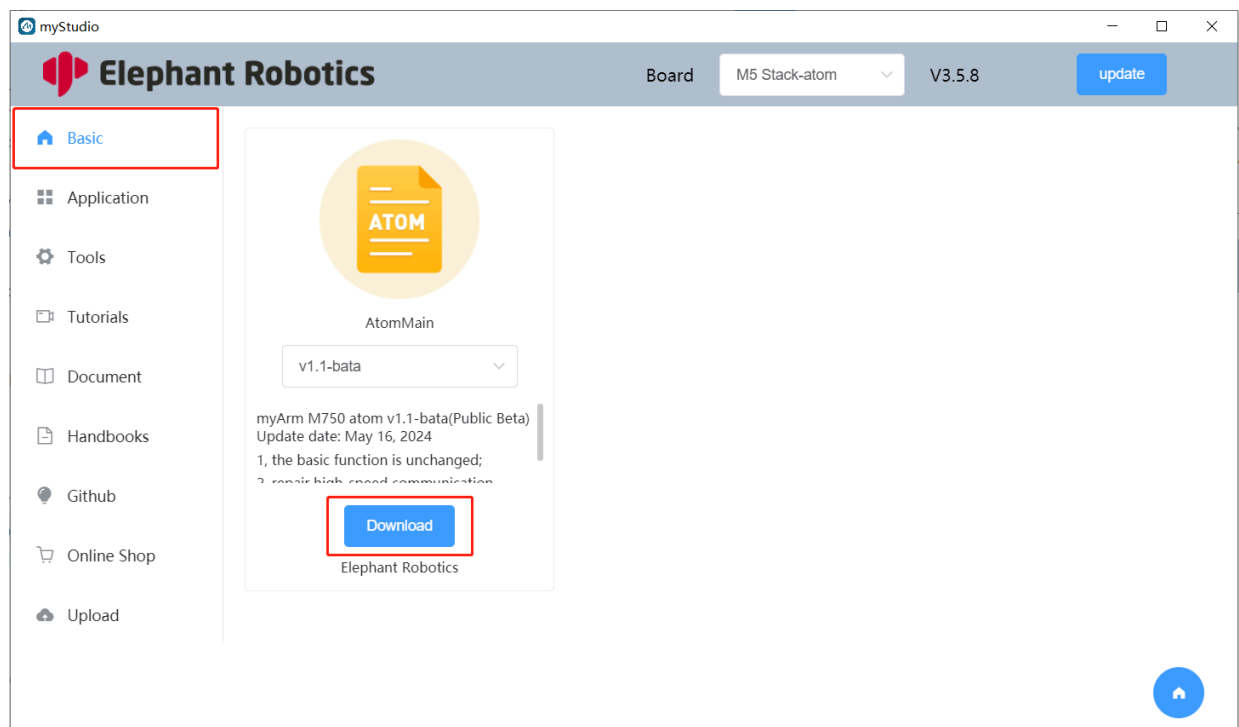
Step 1: Connect to the computer. Connect the terminal Atom with Type-C.



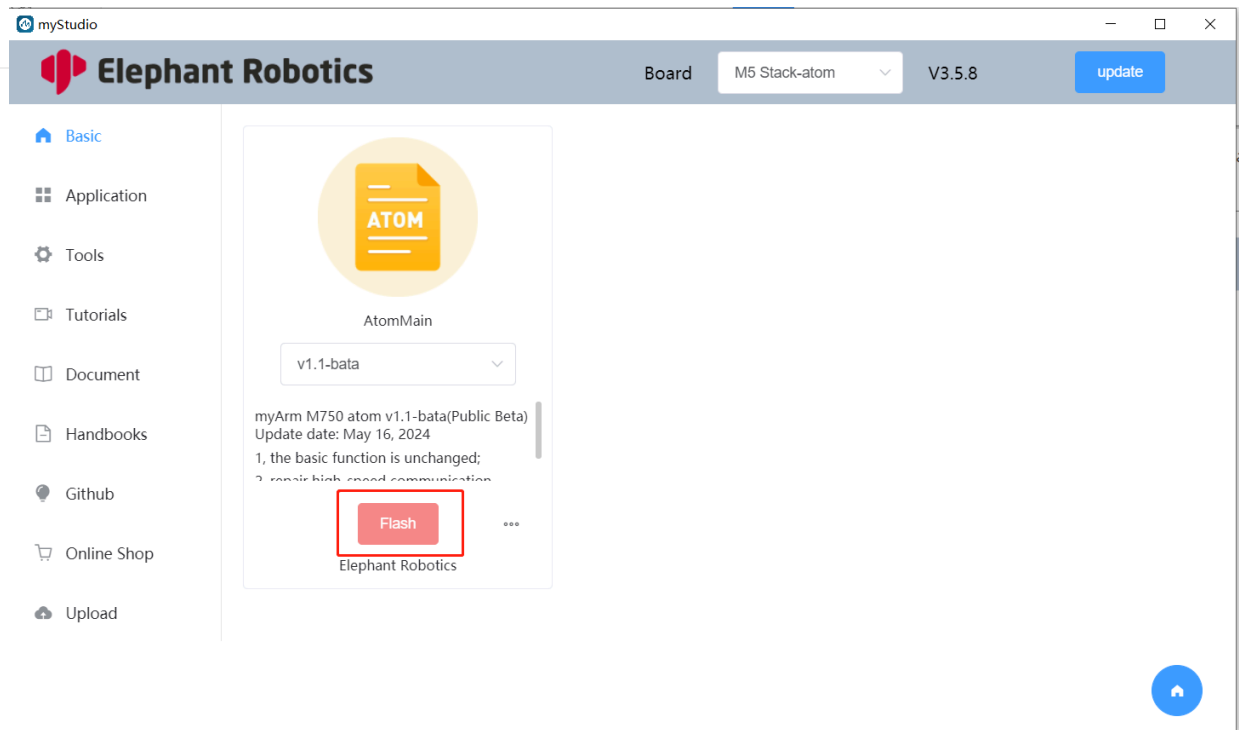
Step 2: Select the robot `myArm M750` and click the `Login` button



Step 2: Select **ATOM** in the **Board** column, and the Atom firmware will appear in the sidebar **Basic**. Atom only has one firmware, just click **Download**.



Step 3: After the download is completed, click the **Flash** button to start burning.



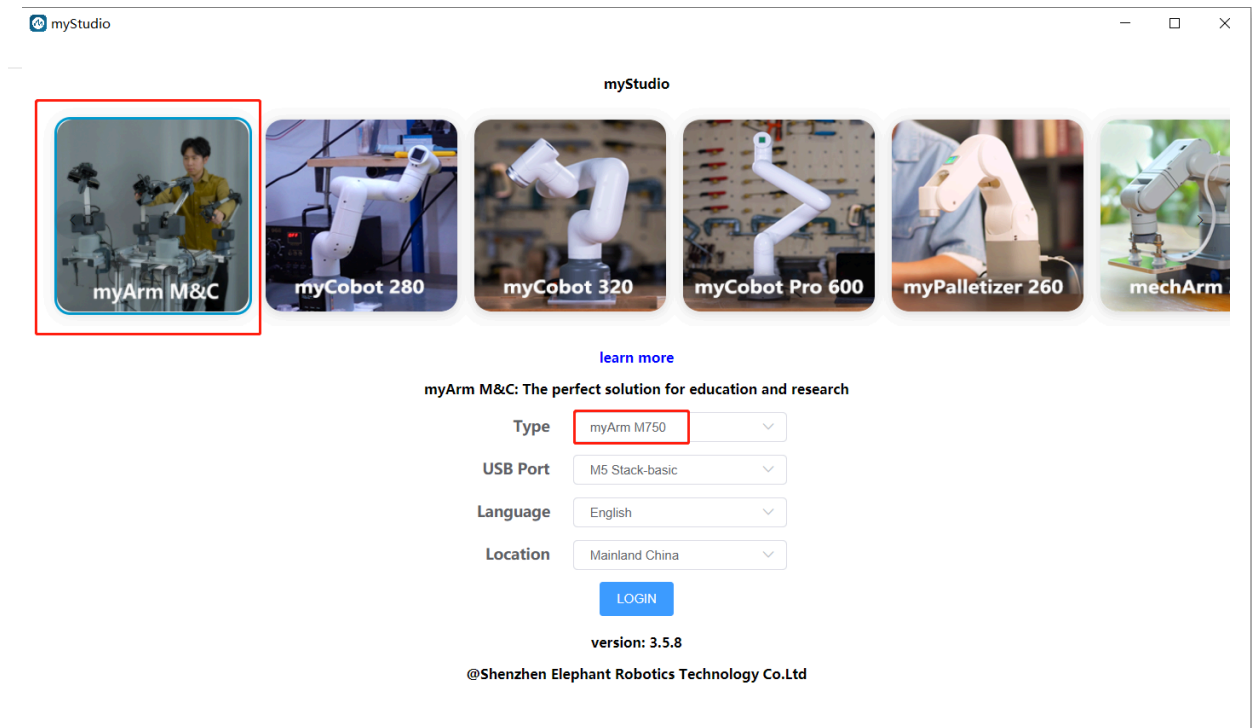
Burn basic firmware

Step 1: Connect M5Stack-Basic to PC using Type-C.

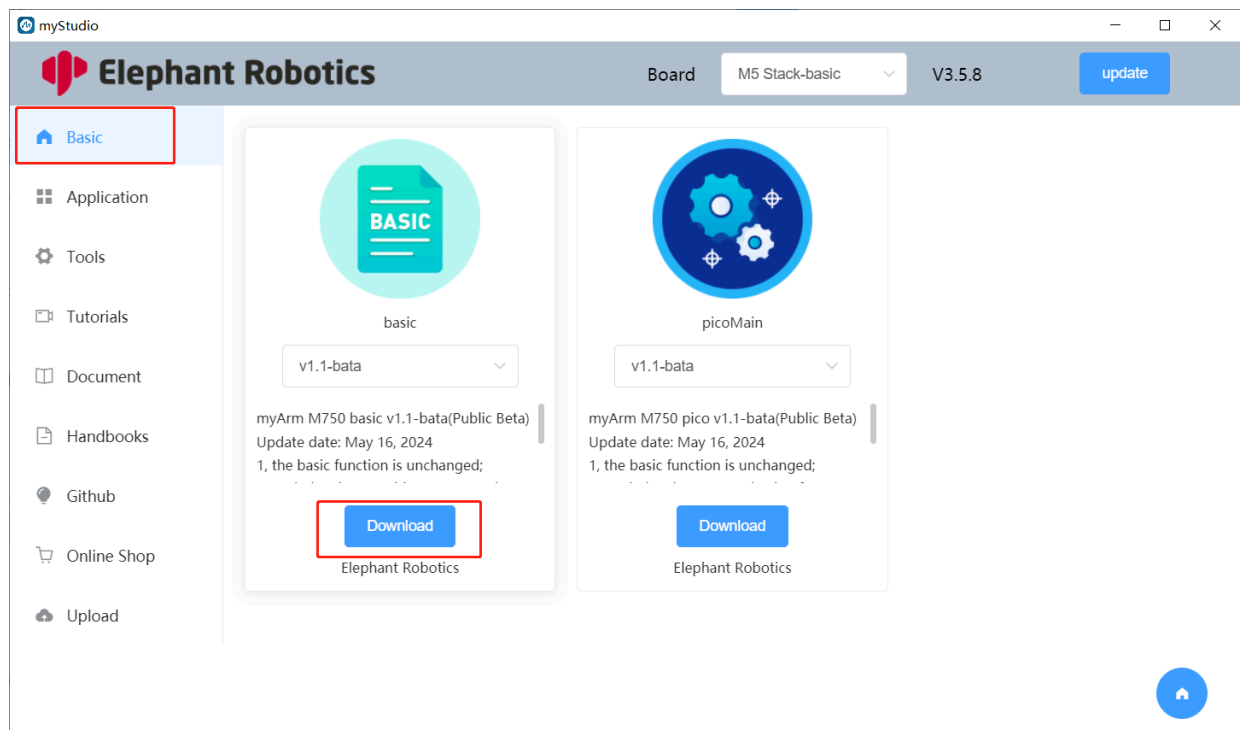


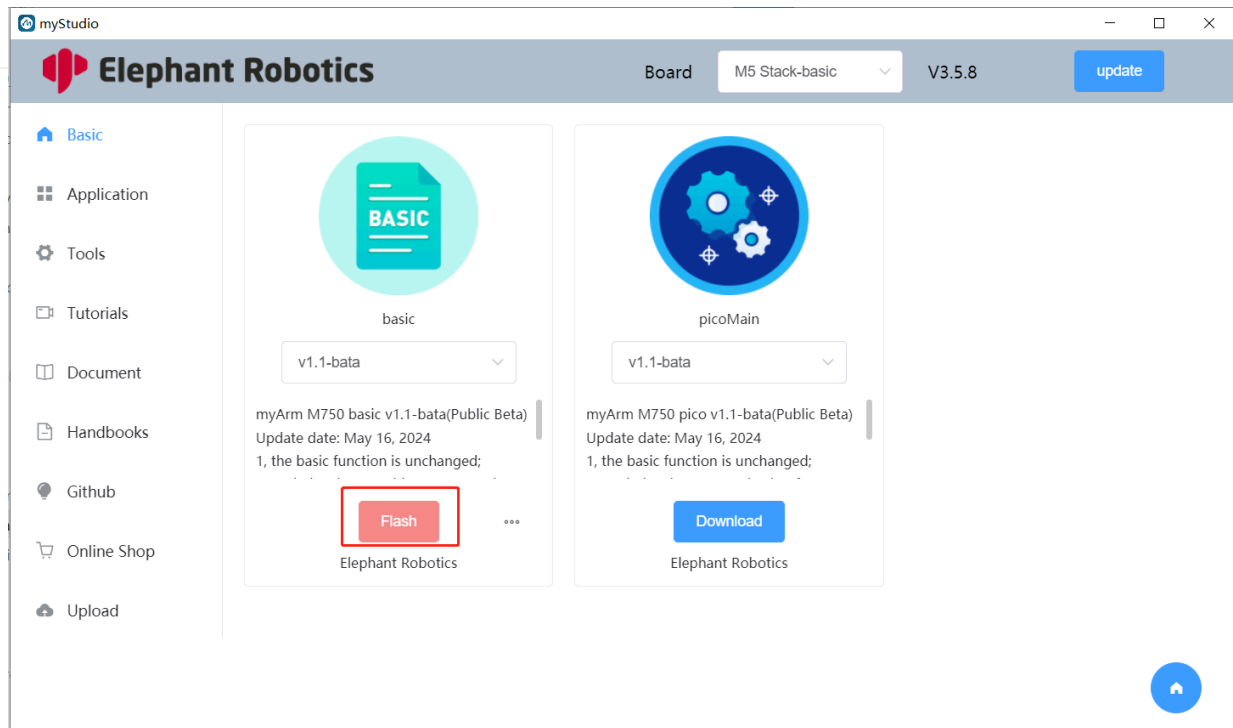


Step 2: Select the robot `myArm M750` and click the `Login` button.



Step 3: After logging in, click **Basic**, select **basic**, click **Download** and then **Flash** to burn.





Burn pico firmware

Step 1: Connect M5Stack-Basic to PC.



Step 2: Select Flash mode in the robot base (you can choose to press the "UP" key once or press the "Down" key four times), and then the base will sleep with a black screen for about 30 seconds.



Step 3: Enter the `myStudio` firmware burning page and select `picoMain` firmware burning.

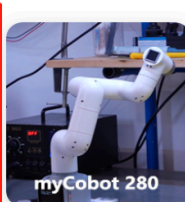
Note: Burning pico firmware requires selecting Flash mode, that is, burning in sleep state. If the sleep time is exceeded, the base monitor will reboot and then burning the pico firmware will fail.

- Select the robot `myArm M750` and click the `Login` button.

myStudio



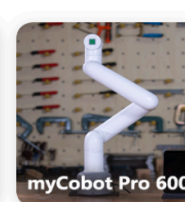
myArm M&C



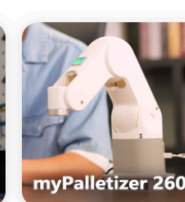
myCobot 280



myCobot 320



myCobot Pro 600



myPalletizer 260



mechArm

[learn more](#)

myArm M&C: The perfect solution for education and research

Type

myArm M750

USB Port

M5 Stack-basic

Language

English

Location

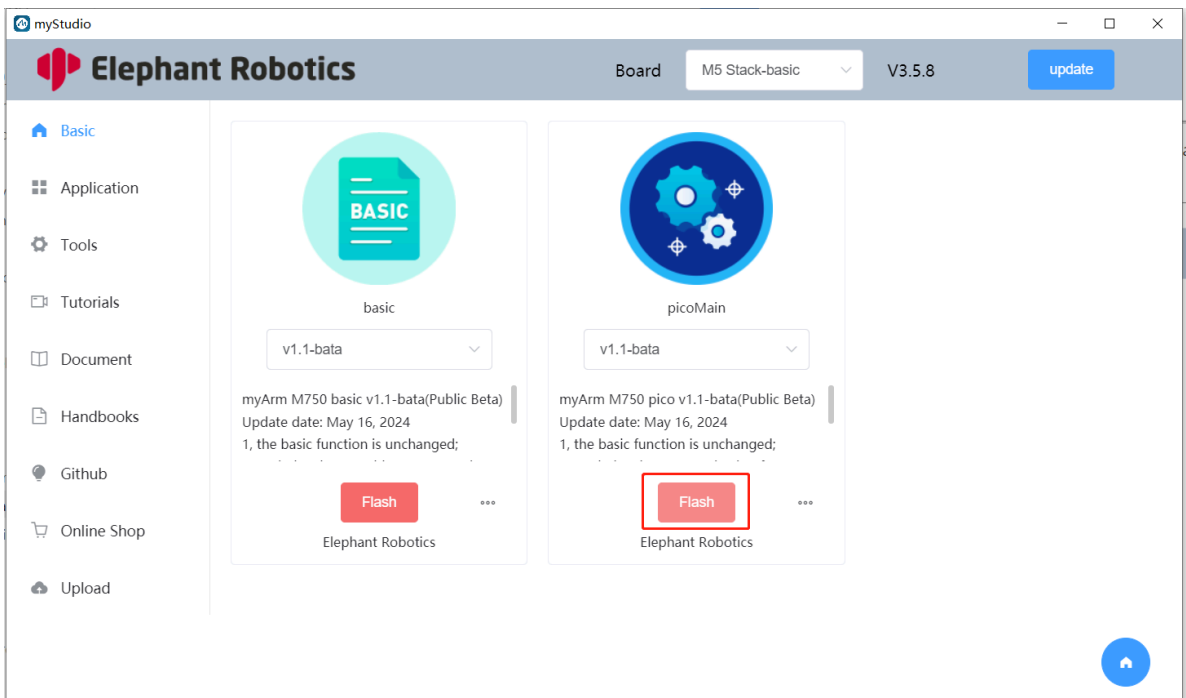
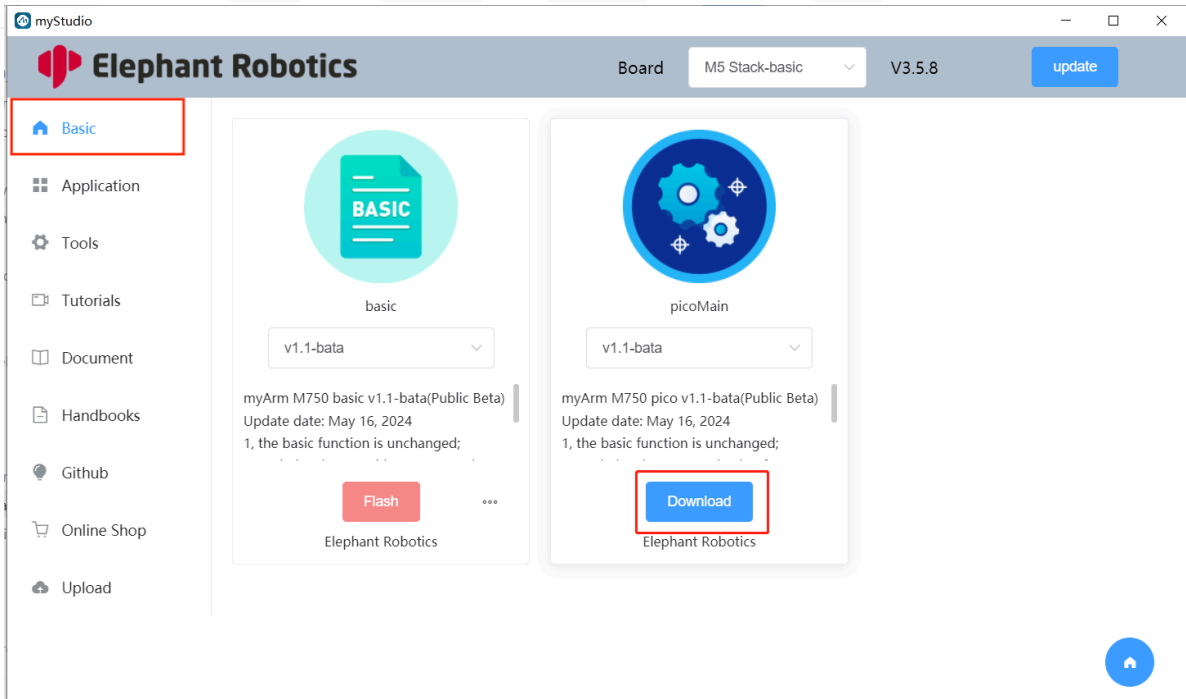
Mainland China

LOGIN

version: 3.5.8

@Shenzhen Elephant Robotics Technology Co.Ltd

- After logging in, click **Basic**, select **picoMain**, click **Download**, and then click **Flash** to burn.



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

Firmware release notes

1 Master Control(Pico)

Version	Features	Updated date
v1.0	1. Real-time detection of the status of each joint, timely alarm and movement stop when abnormal; 2. Support joint control, setting/reading system parameters, IO control (2 inputs, 2 outputs), etc.; 3. Automatically detect whether the joints are normal when turned on, and alarm in time when abnormal.	April 15, 2024
v1.1-beta	1. The basic functions remain unchanged; 2. Optimize the communication frequency and change it to 1M communication 3. Add the interface to read the current position and speed May	16, 2024

Applicable Equipment:

-MyArm M750

2 Screen (Basic)

Version	Features	Update Date
v1.0	Offline interface operation, drag teaching, zero calibration, communication (serial/Bluetooth/wifi2.4G), joint status & version viewing.	April 15, 2024
v1.1-beta	1. The basic functions remain unchanged; 2. Optimize drag teaching, real-time drag speed playback 3. Communication supports real-time speed control.	May 16, 2024

Applicable Equipment:

-MyArm M750

3 Tool (Atom)

Version	Features	Update Date
v1.0	1. Set the LED color; 2. Read the status of the LED button pressed/released, io control (2 inputs and 2 outputs).	April 15, 2024
v1.1-beta	1. The basic functions remain unchanged; 2. Fix the packet loss problem of high-speed communication.	May 16, 2024

Applicable Equipment:

-MyArm M750

SDK Development Guide

1 Environment

The myArm C650 is developed and used on a PC. Since there is no built-in system in the robot arm, it is necessary to combine the robot arm with a PC during use. Please prepare a PC before use.

2 Development Environment

In order to meet the diverse application requirements of the robot in different scenarios, we have adapted the robot to multiple programming languages. So far, we have adapted the following mainstream programming languages, and we think you can use any of the following languages for development. Please be sure to follow the instructions exactly. Any missing steps may result in the corresponding language not working successfully. We wish you the best of luck with your robot.

If you wish to use one of the following programming languages, make sure that your robot is configured for USB/Wi-Fi mode in the Transponder section and that the connection is correct.

- [6.1 Python](#)

If you wish to use one of the following programming languages, make sure that your robot is configured for USB/Wi-Fi mode in the Transponder section and that the connection is correct.。

- [6.2 ROS1](#)

As an open-source robot operating system, ROS (Robot Operating System) provides unlimited possibilities for the development and control of robots. Our robot can be controlled in a modular way through the rich control functions of ROS. Whether it's joint control, path planning, or perceptual feedback, ROS provides tools and libraries to make the control process more flexible and efficient.

- [6.3 ROS2](#)

ROS 2 (Robot Operating System 2) is a flexible software framework designed specifically for robot software development. Our robots make application development more efficient and modular by providing hardware abstraction, device drivers, library functions, visualization tools, messaging, and package management.

- [6.3 Communication](#)

If you know anything about information theory, coding and robot communication functions, then you should understand that all communication stems from data transfer. To make it easier for you to operate your robot, we have opened up a communication protocol based on serial port communication. You can control the robot using the Serial Assistant or encapsulate it into any programming language with which you are familiar.

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

Environment Building

pymycobot is a Python package used for serial communication with myCobot/myArm. It supports Python2, Python3.5 and later versions.

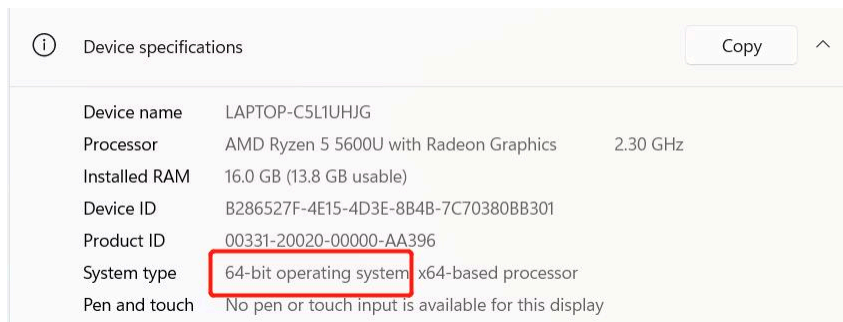
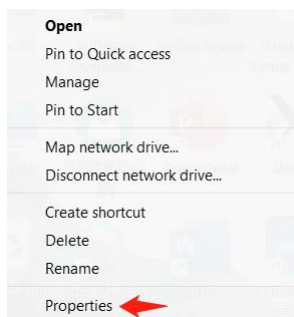
Before using pymycobot, make sure to build a Python environment. Follow the steps below to install Python.

1 Download and Installation of Python

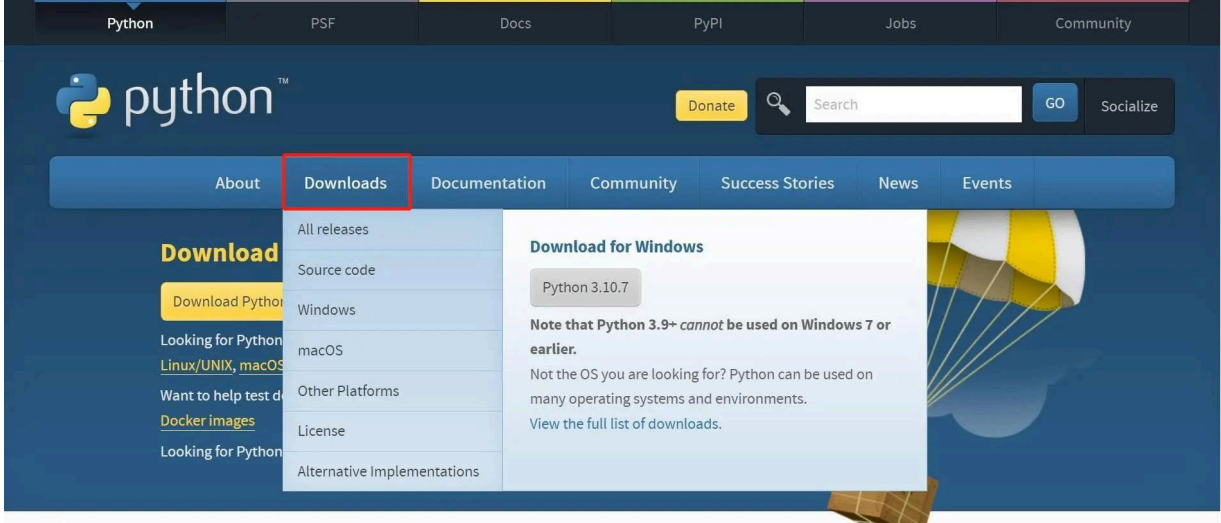
At present, Python has two versions: 2.x and 3.x. These two versions are incompatible with each other. This section takes the version 3.x as an example due to its increasing popularity.

1.1 Installing Python

Notice: Before installation, check the operation system of PC. Press right button on the `My Computer` icon and then select `Properties`. Install the corresponding Python.



- Go to <http://www.python.org/download/> to download Python.
- Click on `Downloads`, and then download begins. Tick `Add Python 3.10 to PATH`. Click on `Install Now`, and then installation begins.



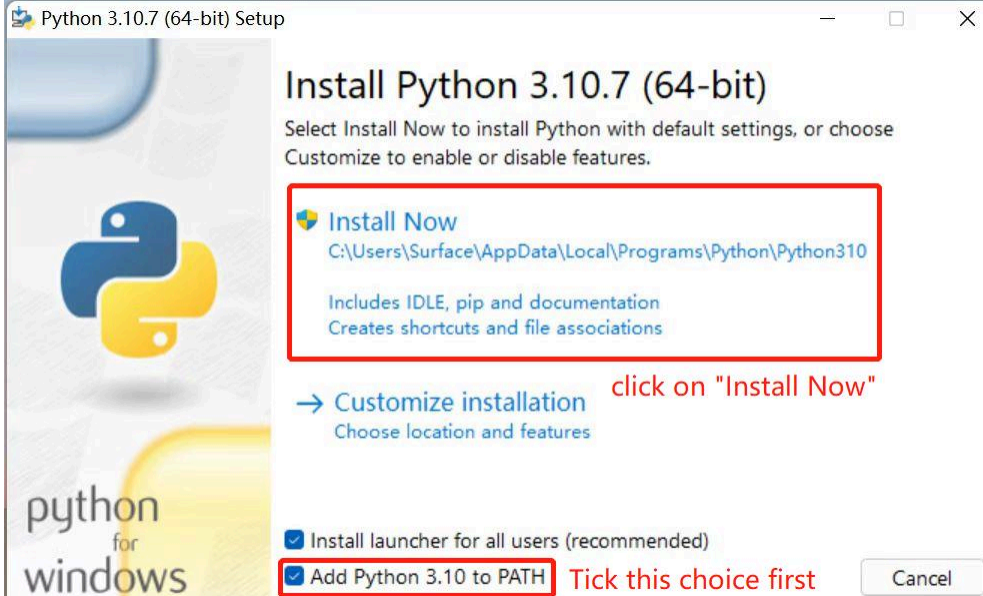
The screenshot shows the Python.org website. The 'Downloads' tab is highlighted in the top navigation bar. A dropdown menu is open, showing options: All releases, Source code, Windows, macOS, Other Platforms, License, and Alternative Implementations. The 'Windows' option is selected, leading to the 'Download for Windows' section. This section features a button for 'Python 3.10.7' and a note stating: 'Note that Python 3.9+ cannot be used on Windows 7 or earlier. Not the OS you are looking for? Python can be used on many operating systems and environments. View the full list of downloads.'

Active Python Releases

For more information visit the Python Developer's Guide.

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

<https://www.python.org/downloads/>

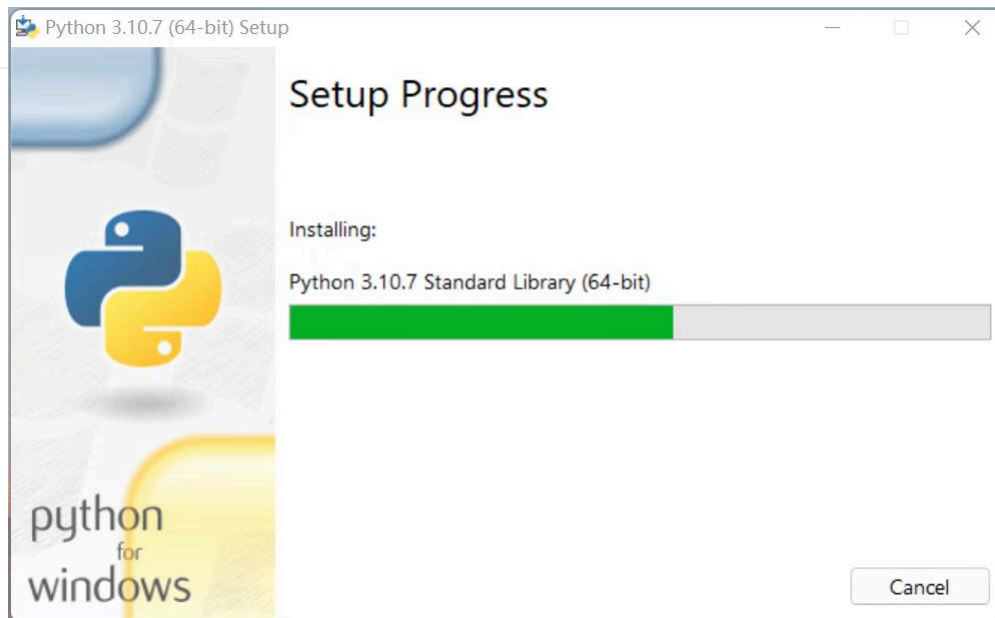


The screenshot shows the 'Python 3.10.7 (64-bit) Setup' window. The title bar reads 'Python 3.10.7 (64-bit) Setup'. The main heading is 'Install Python 3.10.7 (64-bit)'. Below it, the text says: 'Select Install Now to install Python with default settings, or choose Customize to enable or disable features.'

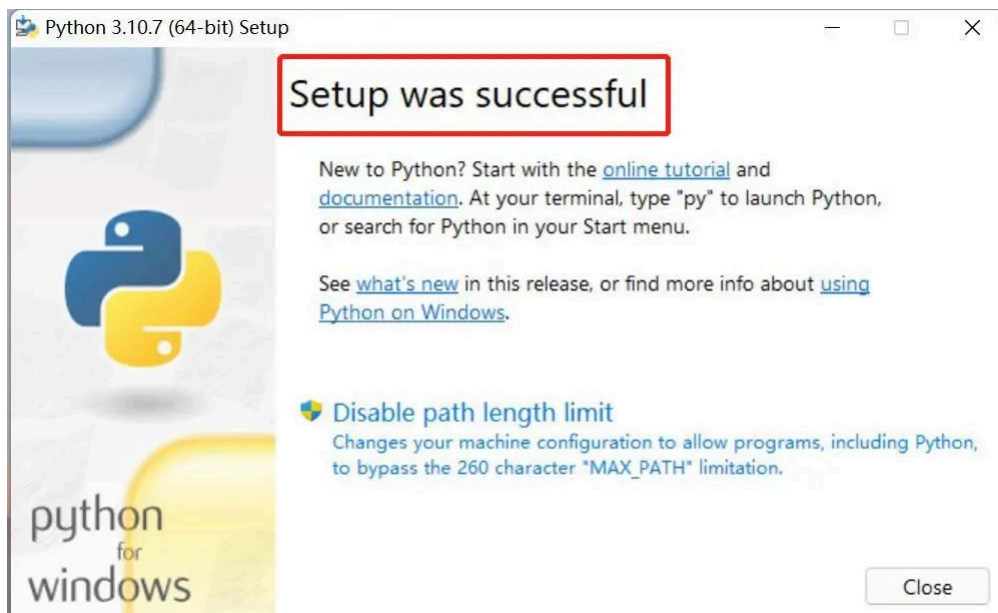
The 'Install Now' option is highlighted with a red box. It shows the installation path: 'C:\Users\Surface\AppData\Local\Programs\Python\Python310'. Below this, it lists features: 'Includes IDLE, pip and documentation' and 'Creates shortcuts and file associations'.

Below the 'Install Now' option, there is a link to 'Customize installation' with the text 'Choose location and features'.

At the bottom, there are two checked options: 'Install launcher for all users (recommended)' and 'Add Python 3.10 to PATH'. The 'Add Python 3.10 to PATH' option is highlighted with a red box. To the right of this box, the text says 'Tick this choice first'. A 'Cancel' button is located at the bottom right.



- Download and installation complete.



1.2 Running Python

Open the command prompt window (Win+R, input `cmd` and press `Enter`). Type `Python` .

Successful Installation:


```

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.
>>>

```

This on-screen instruction means that Python is successfully installed. The prompt `>>>` means Python interactive environment. If you input a Python code to get the execution result immediately.

Error Report:

If a wrong instruction is typed, for example "pythonn", the system may report an error.

```

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

C:\Users\12599>pythonn
'pythonn' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\12599>

```

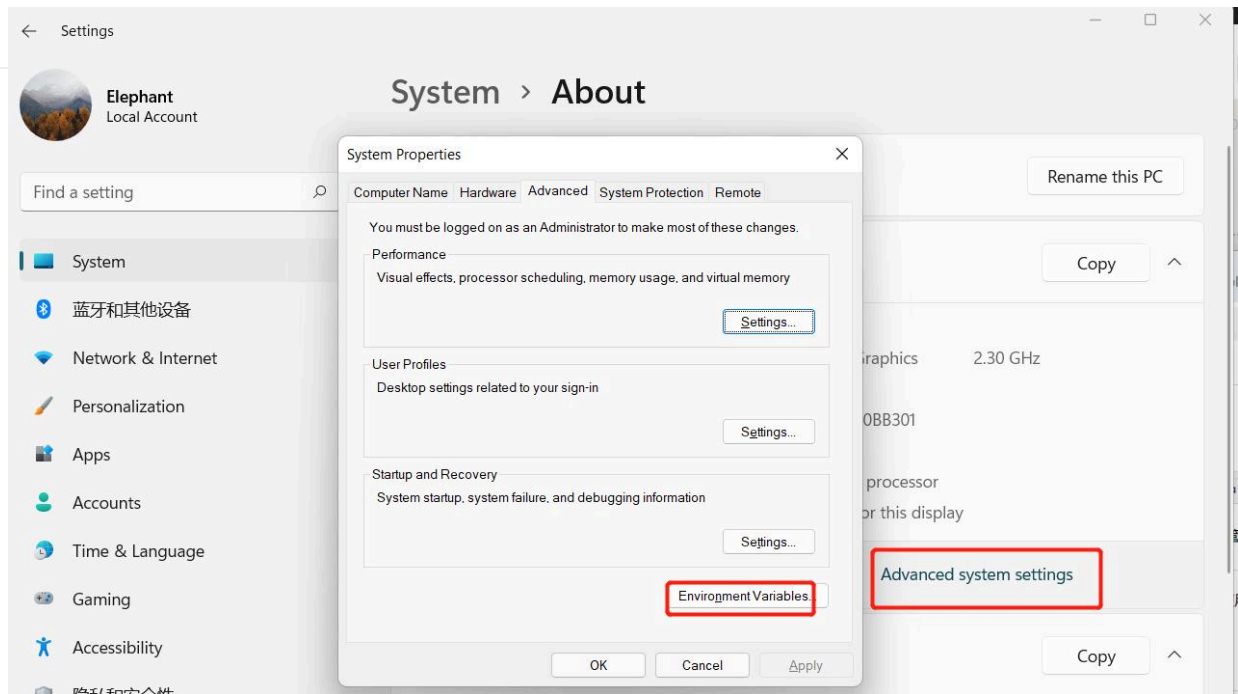
Notice: Generally, the error results from lack of environment configuration. Refer to **1.3 Environment Configuration** to solve problems.

1.3 Environment Variable Configuration

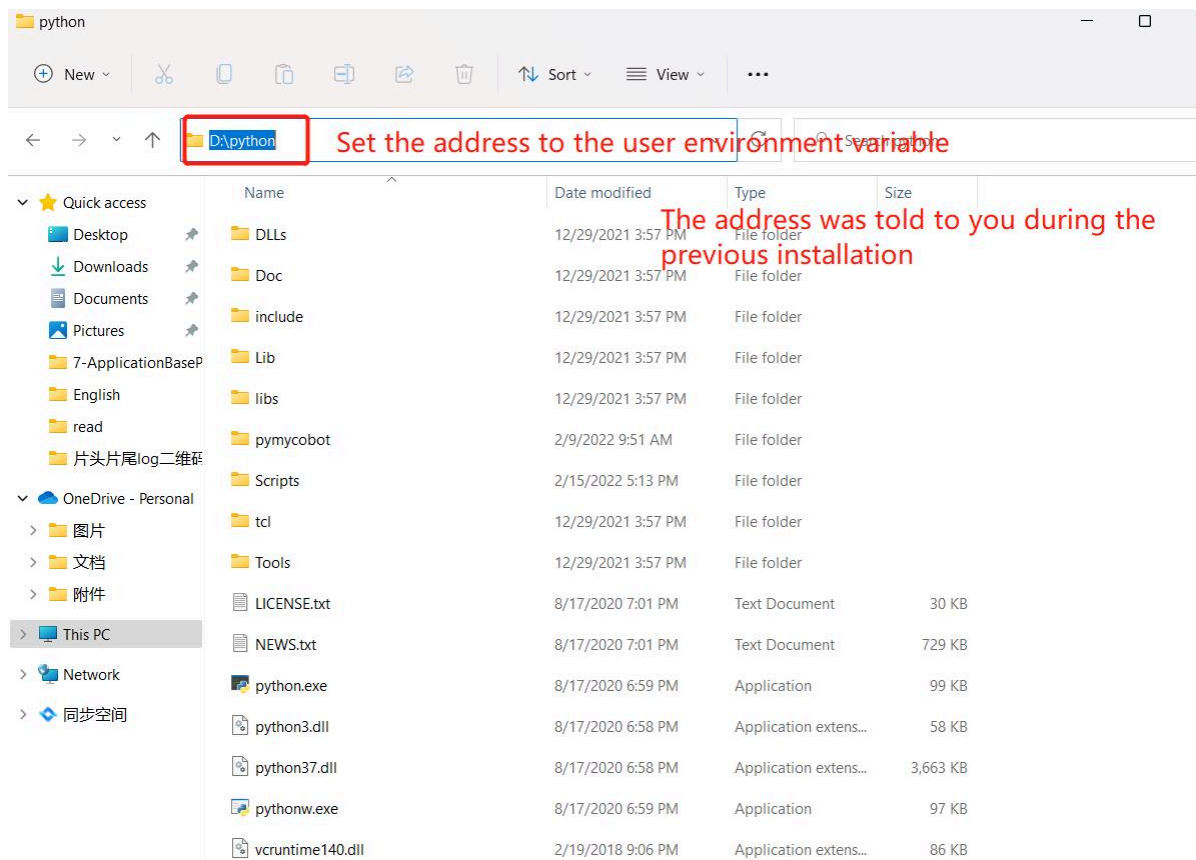
Windows follows the path set by a Path environment variable in search of **python.exe**. Otherwise, an error will be reported. If you fail to tick **Add Python 3.9 to PATH** during installation, you need to manually add the path where python.exe is located into environment variable or download python again. Remember to tick **Add Python 3.9 to PATH**.

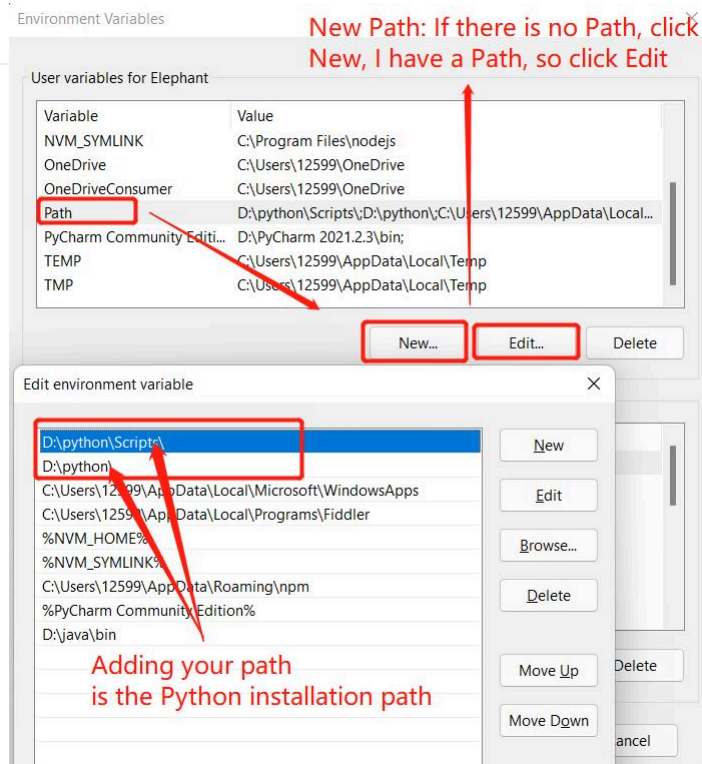
Follow the steps below to add python into environment variable manually.

- Right click on **My Computer** icon --> Properties -> Advanced System Settings -> Environment Variables

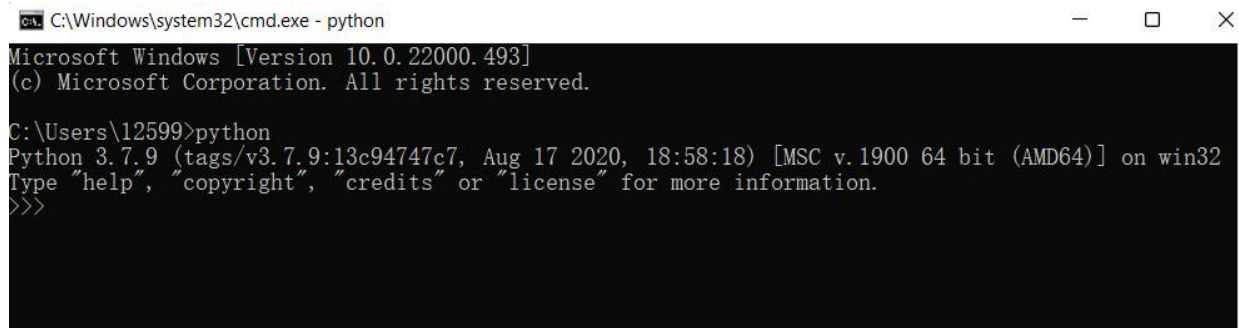


- The environment variables include user variables and system variables. For user variables, users can utilize their own downloaded programs via `cmd` command. Write the absolute path of the target program into the user variables.





- After the configuration, open the command prompt window (Win+R; input `cmd` and press `Enter`), and type `Python`.



2 Installation of PyCharm

PyCharm is a powerful python editor with the nature of cross-platform. Follow the steps below to download and install PyCharm.

Go to [PyCharm](#) to download PyCharm.

2.1 Download and Installation

Official website view:



Version: 2022.2.3
Build: 222.4345.23
11 October 2022

[System requirements](#)

[Installation instructions](#)

[Other versions](#)

[Third-party software](#)

Download PyCharm

[Windows](#)

[macOS](#)

[Linux](#)

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)

Free 30-day trial available

Community

For pure Python development

[Download](#)

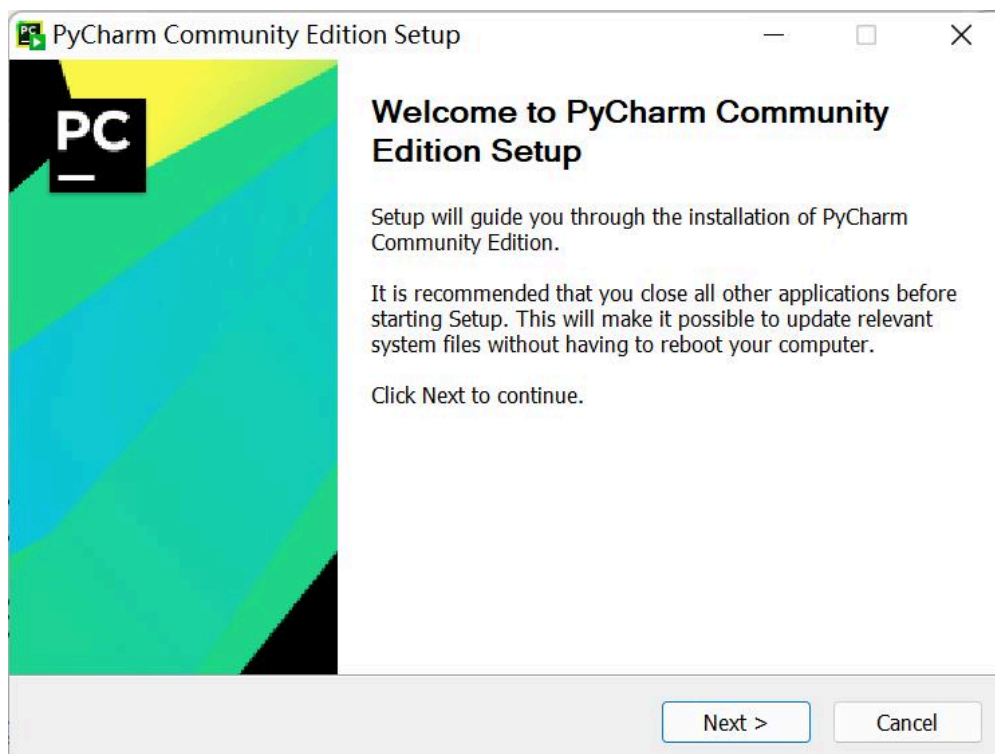
Free, built on open-source



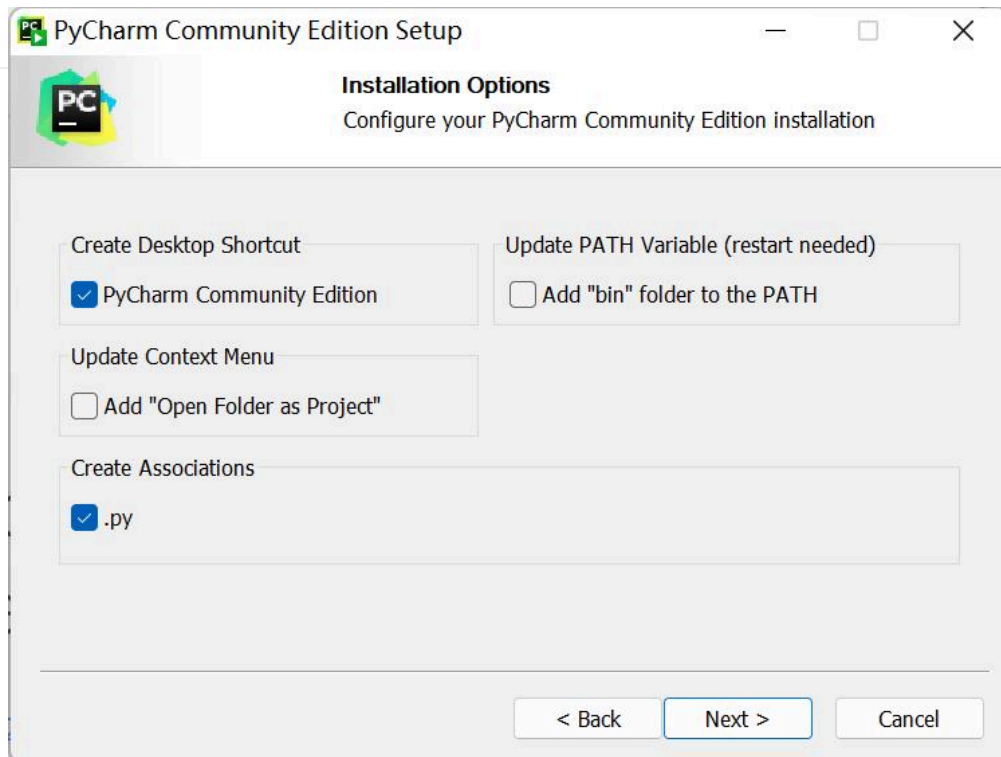
Get the Toolbox App to download PyCharm and its future updates with ease

It is recommended to install the free version.

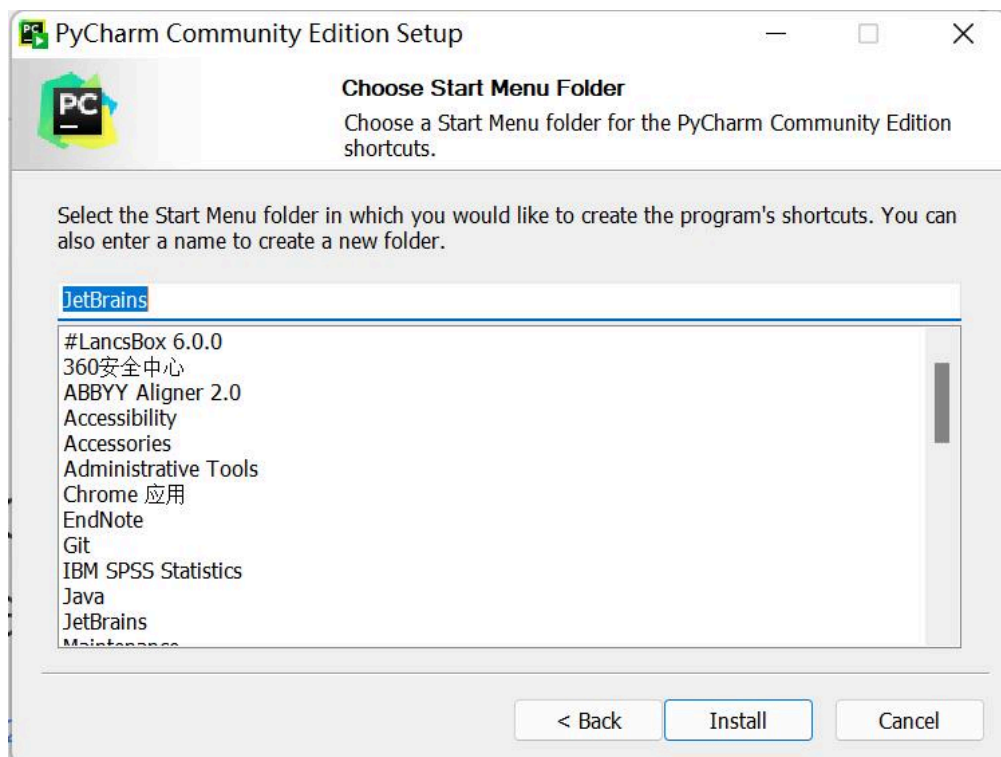
- Click on **Next** :



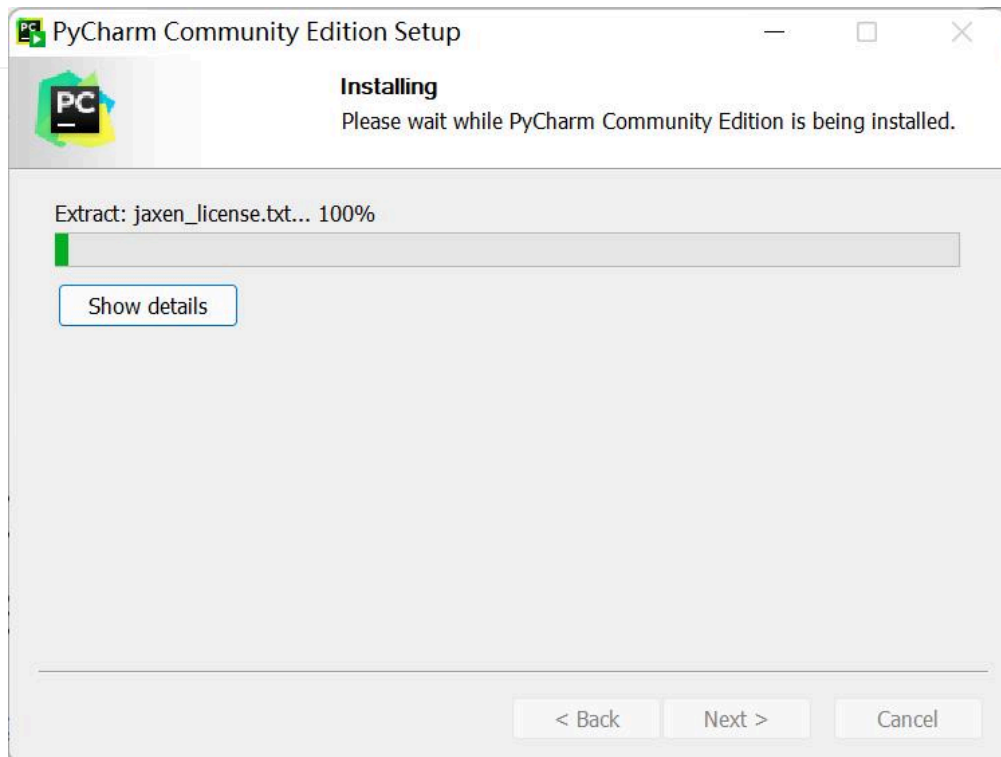
- Select options according to your needs and then select **Next** :



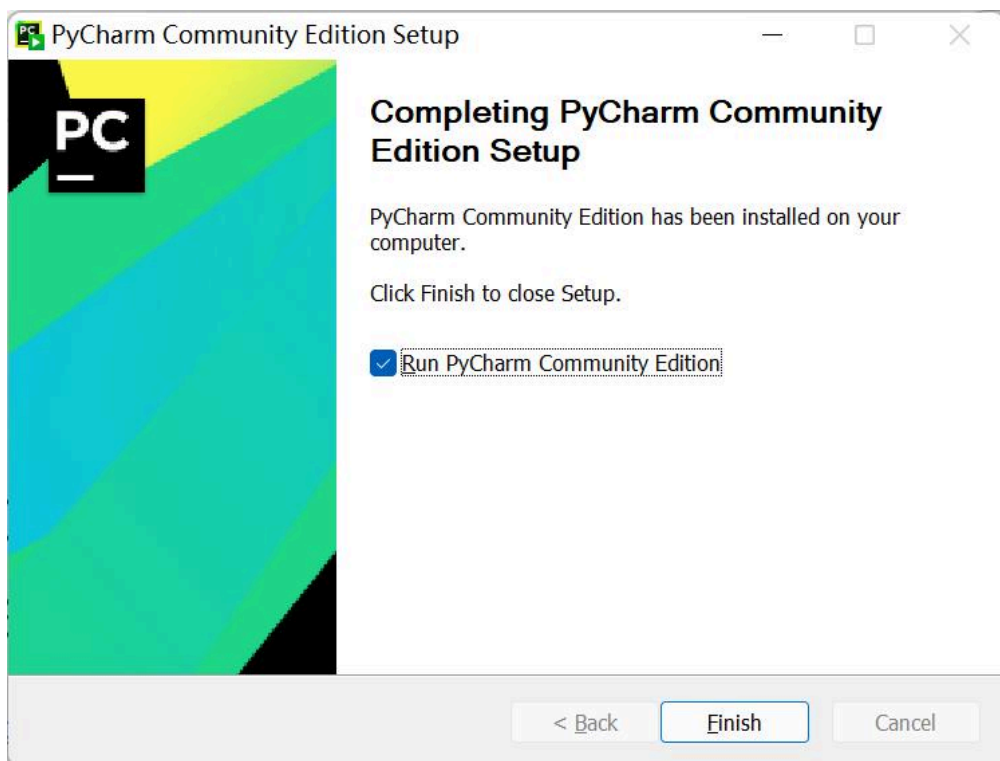
- Tap **Install** :



- Installing:

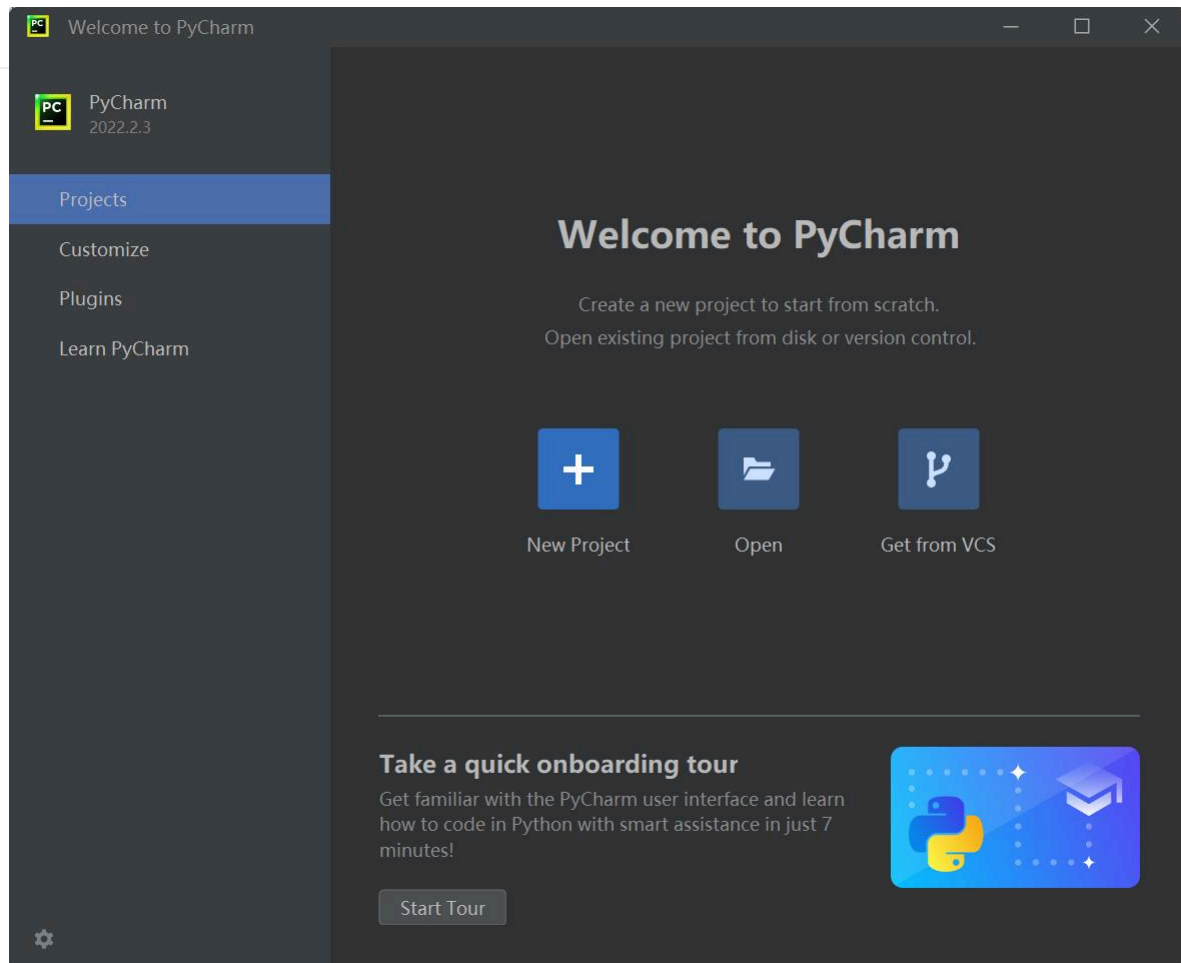


- Tap **Finish**

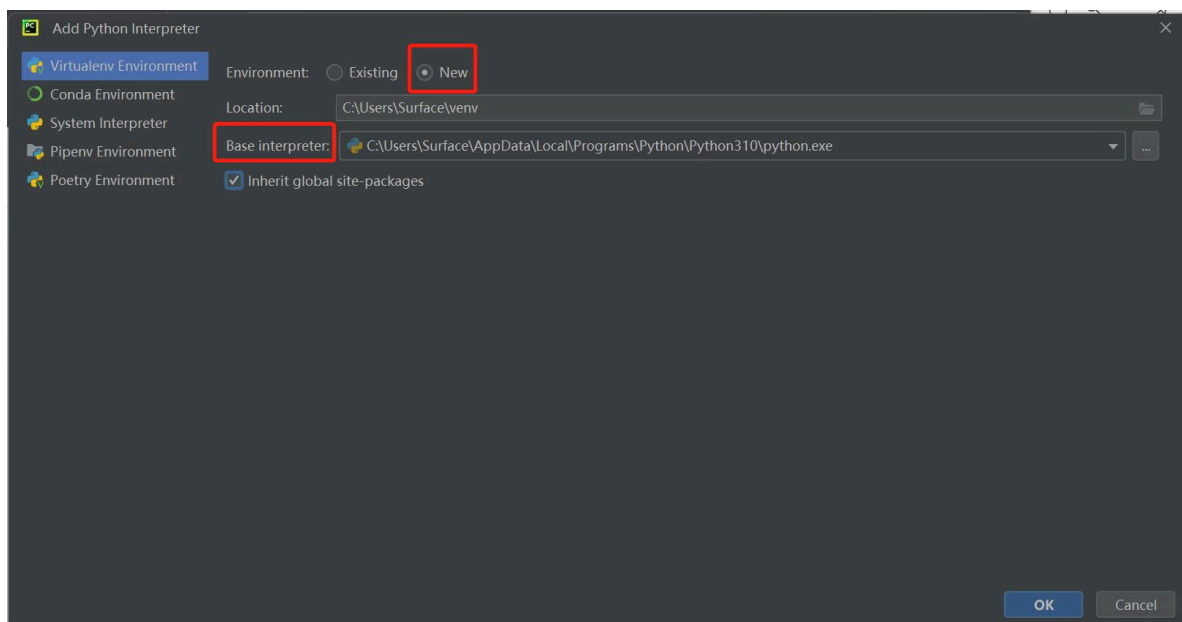
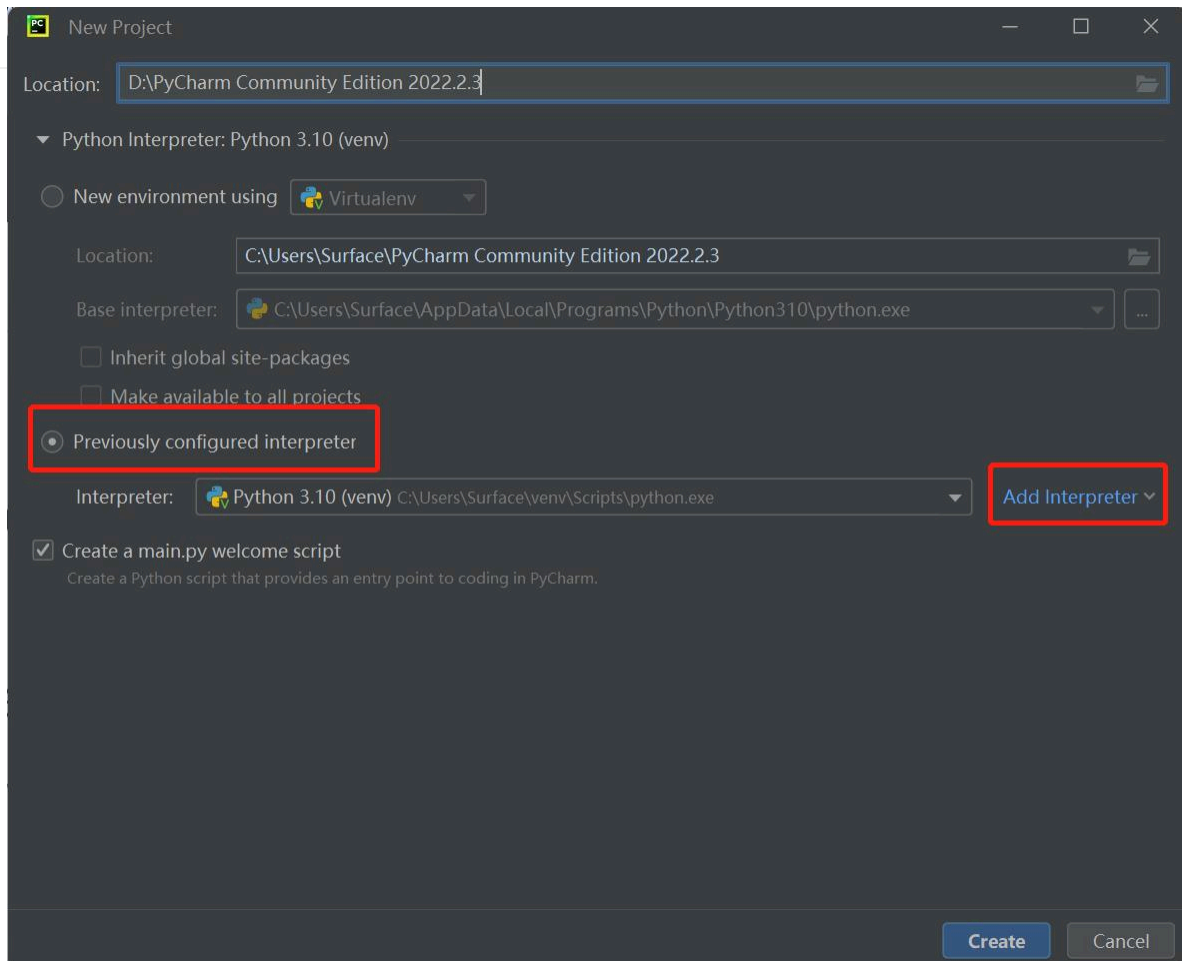


2.2 Create a new project

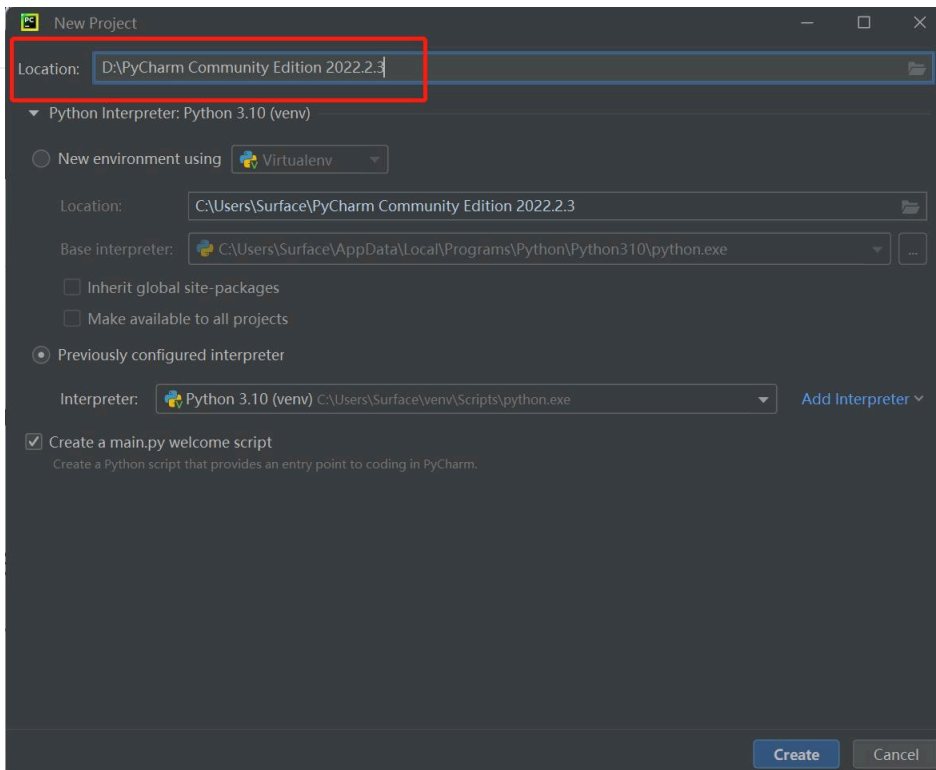
- Click **+New Project** :



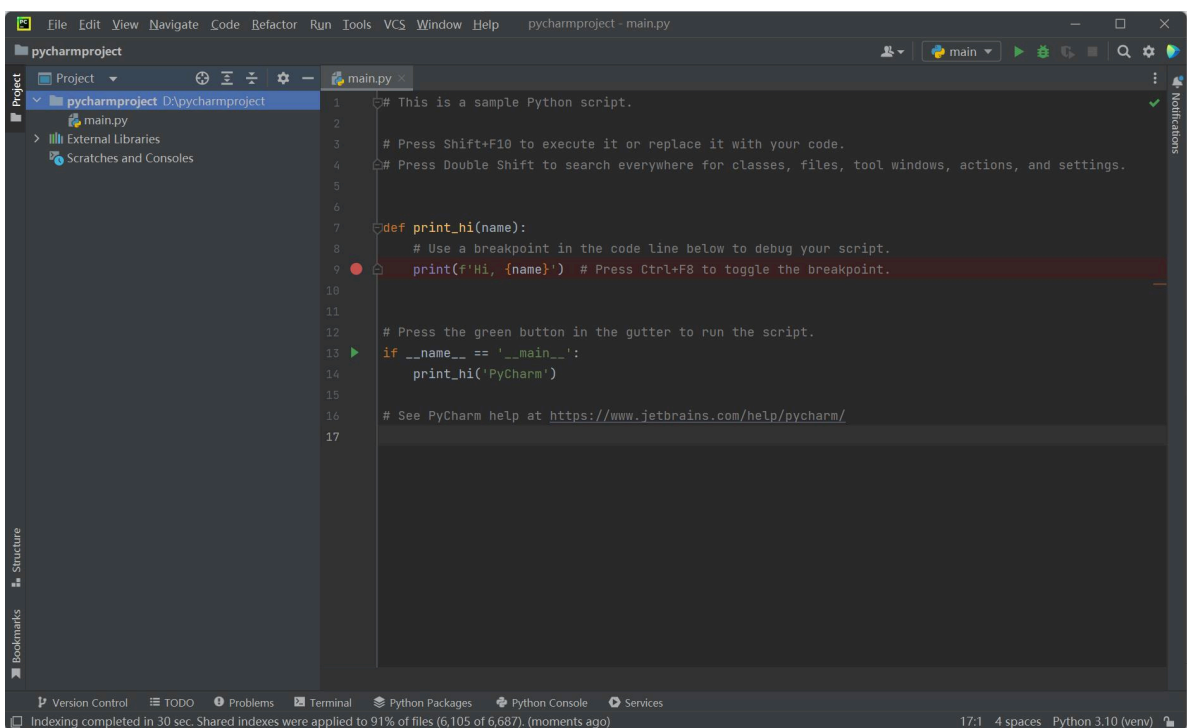
- The `Interpreter` is used to interpret python programs. Select `Add Interpreter` -> `New` to add base interpreter.



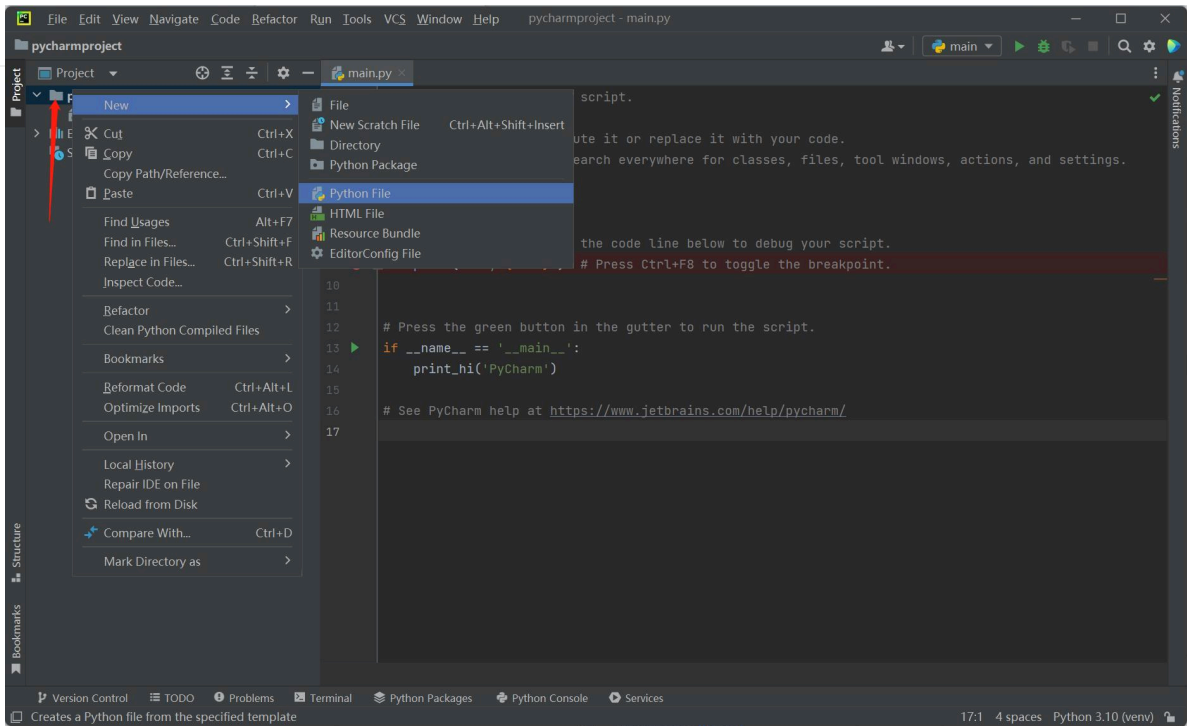
- **Location** refers to the place where to save python file. Choose a file to put your programs.



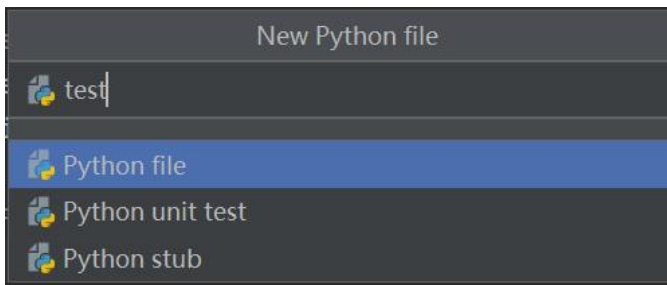
- Click on **Create** and a sample appears:



- Right click on the selection that the red arrow points, and create a new python file.



- Type name for the new file.



3 Preparations

- Firmware burning. Firmware serves as a driver for systems to control robots.
 - **M5Stack version** Make sure to burn `minirobot` for **Basic** at the bottom. And then choose `Transponder` function (to receive instructions from Basic), Press `Press A`. `Atom: OK` means connect successfully. Refer to [MyStudio](#)) for more information about firmware burning.
- `pymycobot` installation. Type `pip install pymycobot --upgrade --user` via terminal (Win+R) `cmd` command.

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



- Source code installation. Open a terminal (Win+R, input `cmd`), and type the command below to install.


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

cd <your-path>/pymycobot
#Go to the pymycobot folder of the downloaded package.

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

- Update pymycobot

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

4 Import of pymycobot

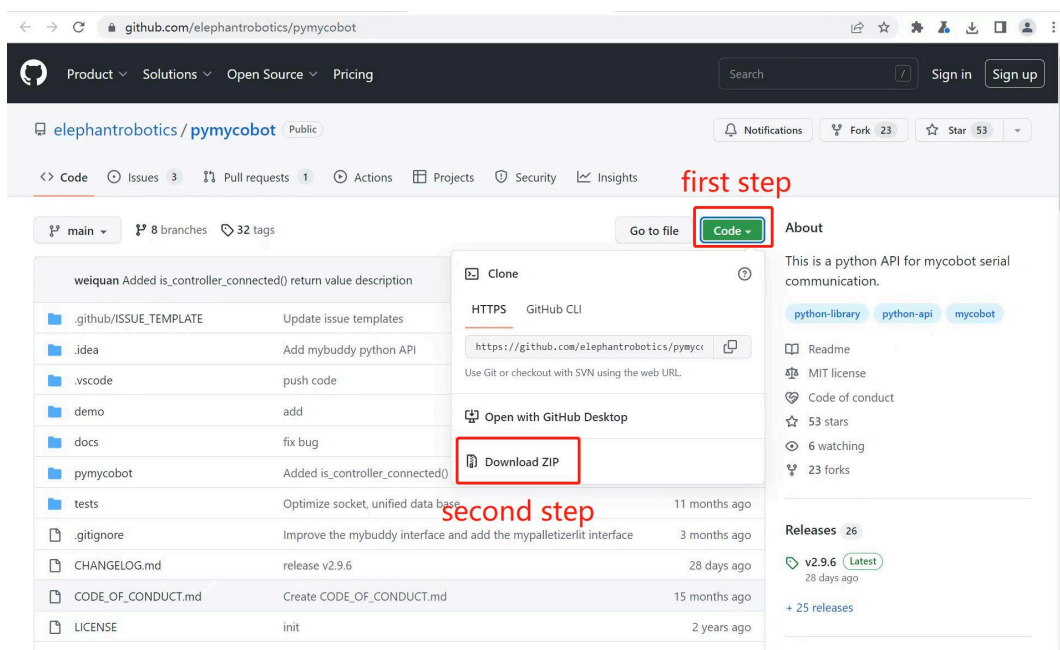
After the above preparations are completed, the control of the robotic arm begins with Python code. Here's an example of the myArm Master 750 version.

- Enter the following code to import our library:

```
from pymycobot import MyArmM
```

Notice:

1. If no red wavy line appears below the codes, pymycobot is successfully installed.
2. if a red wavy line appears, got to the address <https://github.com/elephantrobotics/pymycobot> to download pymycobot manually and put it into python library.



5 Simple Demo

Create a new Python file, and type the following codes to set the color of RGB light panel.

Notice: The baud rates are different according to types of devices. Refer to [calculator device manager](#) to check the corresponding number.

- **Codes for MyArmM:**

```
# demo.py

from pymycobot import MyArmM
import time

# Initiate MyCobot
# Create object code here for windows version
myarmm = MyArmM("COM4", 115200)

i = 7
#loop 7 times
while i > 0:
    myarmm.set_tool_led_color(0, 0, 255) # blue light on
    time.sleep(2) # wait for 2 seconds
    myarmm.set_tool_led_color(255, 0, 0) #red light on
    time.sleep(2) # wait for 2 seconds
    myarmm.set_tool_led_color(0, 255, 0) #green light on
    time.sleep(2) # wait for 2 seconds
    i -= 1
```

Run the example file:

```
python3 demo.py
```

The blue, red, and green lights on the top of the robot flash 7 times continuously at an interval of 2 seconds.

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

Introduction to API

API or Application Programming Interface refers to a number of preset programs. Before utilization, it is required to import API library

The M750 firmware supports the following interfaces:

- [Shake Firmware API](#)
- [Coordinate Firmware API](#)

Shake Firmware API

```
from pymycobot import MyArmM

# Example
m750 = MyArmM("COM3")

# Gets the current angle of all joints
angles = m750.get_joints_angle()
print(angles)

# Set joint 1 to move to 40 and set the speed to 20
m750.set_joint_angle(1, 40, 20)
```

1. Query the status of the bot

1.1 `get_robot_modified_version(self)`

- **function:** Get the bot correction version number
- **Return value:**
 - **version (int):** the bot correction version number

1.2 `get_robot_firmware_version(self)`

- **function:** Obtaining the Robot Firmware Version (Major and Minor Versions)
- **Return value:**
 - **version (int):** the robot firmware

1.3 `get_robot_tool_modified_version(self)`

- **function:** Get the remediation version of the bot tool

1.4 `get_robot_tool_firmware_version(self)`

- **function:** Get the Robot Tool Firmware Version (End Atom)

1.5 set_robot_err_check_state(self, status)

- **function:** Set Error Detection Status You can turn off error detection, but do not turn it off unless necessary
- **Parameters:**
 - **status (int):** 1 open; 0 close

1.6 get_robot_err_check_state(self)

- **function:** Read error detection status

1.7 get_robot_error_status(self)

- **function:** Get the bot error status
- **Return value:**
 - **No error:** [0,0,0,0,0,0,0,0]
 - **assuming that error 1 and 3 are reported in section 1, it should return:** [[1,3],0,0,0,0,0,0]

1.8 get_robot_power_status(self)

- **function:** Get the robot power status
- **Return value:**
 - **power_status (int):** 0: power off, 1: power on

1.9 set_robot_power_on(self)

- **function:** Set the robot to power on

1.10 set_robot_power_off(self)

- **function:** Set the robot to power off

1.11 clear_robot_err(self)

- **function:** Clear the robot abnormality Ignore the error joint and continue to move

1.12 get_recv_queue_max_len(self)

- **function:** The total length of the read command queue, the default length is 100

1.13 set_recv_queue_max_len(self, max_len)

- **function:** Set the total length of the receiving command queue

1.14 clear_recv_queue(self)

- **function:** Clear the queue for receiving commands

1.15 get_recv_queue_len(self)

- **function:** The current length of the read receive queue

2. Joint servo control

2.1 `get_joint_angle(self, joint_id)`

- **function:** Obtain the current angle of the specified joint, when the obtained angle is 200,
 - it means that the joint has no communication
- **Parameters:**
 - `joint_id (int): 0 - 254`
- **Return value:**
 - `angle (int)`

2.2 `set_joint_angle(self, joint_id, angle, speed)`

- **function:** Sets the individual joints to move to the target angle
- **Parameters:**
 - `joint_id (int) : 0 - 254`
 - `angle (int) : 0 - 254`
 - `speed (int) : 1 - 100`

2.3 `get_joints_angle(self)`

- **function:** Obtain the current angle of all joints, if one of the angles is 200??,
 - it means that the joint has no communication
- **Return value:**
 - `angles list(int): 0 - 254`

2.4 `set_joints_angle(self, angles, speed)`

- **function:** Sets all joints to move to the target angle
- **Parameters:**
 - `angles (list[int]): 0 - 254`
 - `speed (int): 0 - 100`

2.5 `get_joints_coord(self)`

- **function:** Get the coordinates
- **Return value:**
 - `list[float] * 6: joints angle`

2.6 `is_robot_moving(self)`

- **function:** See if the robot is moving
 - **Return value:**
 - `1: moving`
 - `0: not moving`
-

2.7 stop_robot(self)

- **function:** The robot stops moving
-

2.8 get_joints_max(self)

- **function:** Read the maximum angle of all joints

2.9 get_joints_min(self)

- **function:** Read the minimum angle of all joints

3. Servo motor control

3.1 set_servo_calibrate(self, servo_id)

- **function:** Sets the zero position of the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0 - 254

3.2 get_servo_encoder(self, servo_id)

- **function:** Gets the current encoder potential value for the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0 - 254
- **Return value:**
 - **encoder (int):** 0-4095

3.3 set_servo_encoder(self, servo_id, encoder, speed)

- **function:** Sets the individual motor motion to the target encoder potential value
- **Parameters:**
 - **servo_id: (int)** 0 - 254
 - **encoder: (int)** 0 - 4095
 - **speed: (int)** 1 - 100

3.4 get_servos_encoder(self)

- **function:** Obtain the current encoder potential values for multiple servo motors

3.5 set_servos_encoder(self, positions, speed)

- **function:** Set the encoder potential value for multiple motors moving to the target
 - **Parameters:**
 - **positions (list[int * 8]):** 0 - 4095:
 - **speed (int):** 1 - 100:
-

3.6 get_servos_speed(self)

- **function:** Gets the current movement speed of multiple servo motors
-

3.7 set_servos_encoder_drag(self, encoders, speeds)

- **function:** Set multiple servo motors with a specified speed to the target encoder potential value
- **Parameters:**
 - **encoders (list[int * 8]):** 0 - 4095:
 - **speeds (list[int * 8]):** -10000 - 10000:

3.8 get_servos_encoder_drag(self)

- **function:** Reads the current encoder value and operating speed of all servo motors
- **Return value:**
 - **encoders (list[int * 8]):** 0-4095
 - **speeds (list[int * 8]):** 0-4095

3.9 is_all_servos_enabled(self)

- **function:** Get the connection status of multiple servo motors
- **Return value:**
 - **status: list[int*8]** 0: The connection failed 1: The connection is successful

3.10 get_servos_temp(self)

- **function:** Obtain the temperature of multiple servo motors

3.11 get_servos_voltage(self)

- **function:** Get the voltage of multiple servo motors

3.12 get_servos_current(self)

- **function:** Obtain the current of multiple servo motors

3.13 get_servos_status(self)

- **function:** Get all the statuses of multiple servo motors

3.14 get_servos_protect_current(self)

- **function:** Obtain multiple servo motor protection currents

3.15 set_servo_enabled(self, servo_id, state)

- **function:** Set the servo motor torque switch
 - **Parameters:**
 - **servo_id (int):** 0-254 254-all
-

- **state:** 0/1
 - **1-focus**
 - **0-release**
-

4. Servo motor system parameter modification

4.1 `set_servo_p(self, servo_id, data)`

- **function:** Sets the proportionality factor of the position loop P of the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0-254
 - **data (int):** 0-254

4.2 `get_servo_p(self, servo_id)`

- **function:** Reads the position loop P scale factor of the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0-254

4.3 `set_servo_d(self, servo_id, data)`

- **function:** Sets the proportional factor for the position ring D of the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0-254
 - **data (int):** 0-254

4.4 `get_servo_d(self, servo_id)`

- **function:** Reads the position ring D scale factor for the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0-254

4.5 `set_servo_i(self, servo_id, data)`

- **function:** Set the proportional factor of the position ring I of the specified servo motor
- **Parameters:**
 - **servo_id (int):** 0 - 254
 - **data (int):** 0 - 254

4.6 `get_servo_i(self, servo_id)`

- **function:** Reads the position loop I scale factor of the specified servo motor

4.7 `set_servo_cw(self, servo_id, data)`

- **function:** Sets the clockwise insensitivity zone of the encoder for the specified servo motor
-

- **Parameters:**

- **servo_id (int): 0 - 254**
- **data (int): 0 - 32**

4.8 `get_servo_cw(self, servo_id)`

- **function:** Reads the clockwise insensitive area of the encoder for the specified servo motor

- **Parameters:**

- **servo_id (int): 0 - 254**

4.9 `set_servo_cw(self, servo_id, data)`

- **function:** Sets the counterclockwise insensitive zone of the encoder for the specified servo motor

- **Parameters:**

- **servo_id (int): 0 - 254**
- **data (int): 0 - 32**

4.10 `get_servo_cw(self, servo_id)`

- **function:** Reads the counterclockwise insensitive area of the encoder of the specified servo motor

- **Parameters:**

- **servo_id (int): 0 - 254**

4.11 `restore_servo_system_param(self, servo_id)`

- **function:** Restore servo motor system parameters

- **Parameters:**

- **servo_id: 0 - 255**

4.12 `set_servo_system_data(self, servo_id, addr, data, mode)`

- **function:** Set the system parameters for the specified servo motor

- **Parameters:**

- **servo_id (int): 0 - 254:**
- **addr (int):**
- **data (int): 0 - 4096**
- **mode (int): 1 - data 1byte. 2 - data 2byte**

4.13 `get_servo_system_data(self, servo_id, addr, mode)`

- **function:** Read the system parameters of the specified servo motor

- **Parameters:**

- **servo_id (int): 0 - 254**
- **addr (int):**
- **mode (int): 1 - data 1byte. 2 - data 2byte**

5. Master IO Control

5.1 `set_master_out_io_state(self, pin_number, status = 1)`

- **function:** Set the host I/O pin status
- **Parameters:**
 - **pin_number:** M750(1 -6), C650(1-2)
 - **status:** 0/1; 0: low; 1: high. default: 1

5.2 `get_master_in_io_state(self, pin_number)`

- **function:** get the host I/O pin status
- **Parameters:**
 - **pin_number (int):** M750(1 -6), C650(1-2)
- **Return value:**
 - 0 or 1. 1: high 0: low

6. Slave IO Control

6.1 `set_tool_out_io_state(self, io_number, status = 1)`

- **function:** Set the Atom pin status
- **Parameters:**
 - **io_number (int):** 1 - 2
 - **status:** 0 or 1; 0: low; 1: high. default: 1

6.2 `get_tool_in_io_state(self, pin)`

- **function:** Get the Atom pin status
- **Parameters:**
 - **pin (int):** pin
- **Return value:**
 - 0 or 1. 1: high 0: low

6.3 `set_tool_led_color(self, r, g, b)`

- **function:** Set the Atom LED color
- **Parameters:**
 - **r:** 0-255
 - **g:** 0-255
 - **b:** 0-255

6.4 `is_tool_btn_clicked(self)`

- **function:** get the end button status
 - **Return value:**
 - **int:** 0/1, 1: press, 0: no press
-

Coordinate Firmware API

```
from pymycobot import MyArmMControl

# Example
arm_control = MyArmMControl('COM3')

# Gets the current angle of all joints
angles = arm_control.get_angles()
print(angles)

# Set joint 1 to move to 40 and set the speed to 20
arm_control.write_angle(1, 40, 20)
```

1. System Status

1.1 `get_robot_modify_version(self)`

- **function:** Get the bot correction version number

1.2 `get_robot_system_version(self)`

- **function:** Obtaining the Robot Firmware Version (Major and Minor Versions)

1.3 `get_robot_tool_modify_version(self)`

- **function:** Get the remediation version of the bot tool

1.4 `get_robot_tool_system_version(self)`

- **function:** Get the Robot Tool Firmware Version (End Atom)

2. Overall Status

2.1 `power_on(self)`

- **function:** The robotic arm turns on the power

2.2 `power_off(self)`

- **function:** The robotic arm turns off the power

2.3 `is_powered_on(self)`

- **function:** Control core connection status queries
 - **Return value:**
 - 1 - power on
 - 0 - power off
 - -1 - error data
-

2.4 release_all_servos(self, data)

- **function:** The robot turns off the torque output
- **Parameters:**
 - **data:** 1 - Un damping (The default is damping)

2.5 set_fresh_mode(self, mode)

- **function:** Set command refresh mode
- **Parameters:**
 - **mode:** int.
 - **1** - Always execute the latest command first.
 - **0** - Execute instructions sequentially in the form of a queue.

2.6 get_fresh_mode(self)

- **function:** Query sports mode

2.7 get_robot_status(self)

- **function:** Get robot status

3. MDI mode and operation**3.1 get_angles(self)**

- **function:** Get the angle of all joints.
- **Return value:**
 - **list:** A float list of all angle.

3.2 write_angle(self, joint_id, degree, speed)

- **function:** Send the angle of a joint to robot arm.
- **Parameters:**
 - **joint_id:** (int) 1 ~ 6
 - **degree:** (float) -150 ~ 150
 - **speed :** (int) 1 ~ 100

3.3 write_angles(self, angles, speed)

- **function:** Send the angles of all joints to robot arm.
- **Parameters:**
 - **angles:** (list) A float list of all angle.
 - **speed :** (int) 1 ~ 100

3.4 get_coords(self)

- **function:** Get the coordinates of all joints.

- **Return value:**

-
- **list:** A float list of all coordinates.

3.5 `write_coord(self, coord_id, coord, speed)`

- **function:** Send the coordinates of a joint to robot arm.

- **Parameters:**

- **coord_id:** (int) 1 ~ 6
- **coord:** (float) -150 ~ 150
- **speed :** (int) 1 ~ 100

3.6 `write_coords(self, coords, speed, mode)`

- **function:** Send the coordinates of all joints to robot arm.

- **Parameters:**

- **coords:** (list) A float list of all coordinates.
- **speed :** (int) 1 ~ 100
- **mode:** (int) 0 - normal, 1 - low, 2 - high

3.7 `pause(self)`

- **function:** Pause movement

3.8 `is_paused(self)`

- **function:** Judge whether the manipulator pauses or not.

- **Return value:**

- 1 - paused
- 0 - not paused
- -1 - error

3.9 `resume(self)`

- **function:** Recovery movement

3.10 `stop(self)`

- **function:** Stop moving

3.11 `is_in_position(self, data, mode = 0)`

- **function:** Judge whether in the position.

- **Parameters:**

- **data:** A data list, angles or coords.
- **for myArm:** angles len 7, coords len 6.
- **mode:** 1 - coords, 0 - angles

- **Return value:**
-

- 1 - True
- 0 - False
- -1 - Error

3.12 `is_moving(self)`

- **function:** Detect if the robot is moving
- **Return value:**
 - 0 - not moving
 - 1 - is moving
 - -1 - error data

4. JOG mode and operation

4.1 `jog_angle(self, joint_id, direction, speed)`

- **function:** Jog control angle.
- **Parameters:**
 - **joint_id:** int
 - **for myArm:** Joint id 1 - 7.
 - **direction:** 0 - decrease, 1 - increase
 - **speed:** int (0 - 100)

4.2 `jog_rpy(self, end_direction, direction, speed)`

- **function:** Rotate the end around a fixed axis in the base coordinate system
- **Parameters:**
 - **end_direction (int):** Roll, Pitch, Yaw (1-3)
 - **direction (int):** 1 - forward rotation, 0 - reverse rotation
 - **speed (int):** 1 ~ 100

4.3 `jog_coord(self, coord_id, direction, speed)`

- **function:** Jog control coord.
- **Parameters:**
 - **coord_id:** int 1 ~ 6
 - **direction:** 0 - decrease, 1 - increase
 - **speed:** int (1 - 100)

4.4 `jog_increment(self, joint_id, increment, speed)`

- **function:** step mode
- **Parameters:**
 - **joint_id(int):**
 - **for myArm:** Joint id 1 - 6.
 - **increment(int):** incremental
 - **speed(int):** int (0 - 100)

5. Motor potentiometer control mode

5.1 `get_encoder(self, joint_id)`

- **function:** Obtain the specified joint potential value.
- **Parameters:**
 - `joint_id`: (int)
 - for myArm: Joint id 1 - 7.

5.2 `get_encoders(self)`

- **function:** Get the six joints of the manipulator
- **Return value:**
 - The list of encoders

6. Software joint limit

6.1 `get_joint_min(self, joint_id)`

- **function:** Gets the minimum movement angle of the specified joint
- **Parameters:**
 - `joint_id`:
 - for myArm: Joint id 1 - 7.
- **Return value:**
 - angle value(float)

6.2 `get_joint_max(self, joint_id)`

- **function:** Gets the maximum movement angle of the specified joint
- **Parameters:**
 - `joint_id`:
 - for myArm: Joint id 1 - 7.
- **Return value:**
 - angle value(float)

6.3 `set_joint_min(self, joint_id, angle)`

- **function:** Set the joint minimum angle
- **Parameters:**
 - `joint_id`: int.
 - for myArm: Joint id 1 - 7.
 - `angle`: 0 ~ 180

6.4 `set_joint_max(self, joint_id, angle)`

- **function:** Set the joint maximum angle
 - **Parameters:**
-

- **joint_id:** int.
 - **for myArm:** Joint id 1 - 7.
 - **angle:** 0 ~ 180
-

7. Servo control

7.1 `is_servo_enable(self, servo_id)`

- **function:** To detect the connection state of a single joint
- **Parameters:**
 - **servo_id:**
 - **for myArm:** Joint id 1 - 8.
- **Return value:**
 - **0** - disable
 - **1** - enable
 - **-1** - error

7.2 `is_all_servo_enable(self)`

- **function:** Detect the connection status of all joints
- **Return value:**
 - **0** - disable
 - **1** - enable
 - **-1** - error

7.3 `set_servo_data(self, servo_id, data_id, value, mode)`

- **function:** Set the data parameters of the specified address of the steering gear
- **Parameters:**
 - **servo_id:** Serial number of articulated steering gear.
 - **for myArm:** joint id 1 - 8
 - **data_id:** Data address.
 - **value:** 0 - 4096
 - **mode:** 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.

7.4 `get_servo_data(self, servo_id, data_id, mode)`

- **function:** Read the data parameter of the specified address of the steering gear.
 - **Parameters:**
 - **servo_id:** Serial number of articulated steering gear. 1 ~ 7
 - **data_id:** Data address.
 - **mode:** 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.
 - **Return value:**
 - **values** 0 - 4096
-

7.5 `release_servo(self, servo_id, mode)`

- **function:** Power off designated servo
-

- **Parameters:**

- **servo_id:** int 1 ~ 8
- **mode:** Default damping, set to 1, cancel damping

7.6 `focus_servo(self, servo_id)`

- **function:** Power on designated servo

- **Parameters:**

- **servo_id:** int 1 ~ 7

7.7 `get_servo_speeds(self)`

- **function:** Get the joint speed .

- **Return value:**

- **speeds:** list[float * 8] +- 3000 step/s

7.8 `get_servo_currents(self)`

- **function:** Get the joint current

- **Return value:**

- **currents:** list[float * 8] 0 ~ 3250

7.9 `get_servo_voltages(self)`

- **function:** Get the joint voltages

- **Return value:**

- **voltages:** list[float] voltage 0 ~ 240

7.10 `get_servo_status(self)`

- **function:** Get the joint status.

- **Return value:**

- **status:** list[int] 0 ~ 255
- 0 - normal,
- other - error

7.11 `get_servo_temps(self)`

- **function:** Get joint temperature

- **Return value:**

- **temperatures:** list[float] 0 ~ 255
-

8. Atom IO

8.1 `set_digital_output(self, pin_no, pin_signal)`

- **function:** Set the terminal atom io status
- **Parameters:**
 - `pin_no (int):`
 - `pin_signal (int): 0 / 1`

8.2 `get_digital_input(self, pin_no)`

- **function:** Get the terminal atom io status
- **Return value:**

9. Atom Button

9.1 `set_led_color(self, r = 0, g = 0, b = 0)`

- **function:** Set the light color on the top of the robot arm.
- **Parameters:**
 - `r (int): 0 ~ 255`
 - `g (int): 0 ~ 255`
 - `b (int): 0 ~ 255`

9.2 `is_tool_btn_clicked(self)`

- **function:** Judge whether the button on the top of the robot arm is clicked or not.
- **Return value:**
 - `0 - not clicked`
 - `1 - is clicked`
 - `-1- error data`

10. Gripper API

10.1 `set_gripper_enabled(self)`

- **function:** Enable gripper

10.2 `get_gripper_value(self)`

- **function:** Get the value of gripper.
- **Return value:**
 - `gripper value (int)`

10.3 `set_gripper_state(self, flag, speed)`

- **function:** Set gripper switch state
 - **Parameters:**
-

- **flag (int):** 0 - open, 1 - close, 254 - release
 - **speed (int):** 1 ~ 100
-

10.4 `set_gripper_value(self, gripper_value, speed)`

- **function:** Set gripper value
- **Parameters:**
 - **gripper_value (int):** 0 ~ 100
 - **speed (int):** 1 ~ 100

10.5 `set_gripper_calibration(self)`

- **function:** Set the current position to zero, set current position value is 2048 .

10.6 `is_gripper_moving(self)`

- **function:** Judge whether the gripper is moving or not
- **Return value:**
 - 0 - not moving
 - 1 - is moving
 - -1- error data

11. Cartesian space parameter setting

11.1 `set_tool_reference(self, coords)`

- **function:** Set tool coordinate system
- **Parameters:**
 - **coords:** a list of coords value(List[float]), [x(mm), y, z, rx(angle), ry, rz]

11.2 `get_tool_reference(self)`

- **function:** Get tool coordinate system

11.3 `set_world_reference(self, coords)`

- **function:** Set the world coordinate system
- **Parameters:**
 - **coords:** a list of coords value(List[float]), [x(mm), y, z, rx(angle), ry, rz]

11.4 `get_world_reference(self)`

- **function:** Get the world coordinate system

11.5 `set_reference_frame(self, rftype)`

- **function:** Set the base coordinate system
 - **Parameters:**
-

- **rftype: 0 - base 1 - tool.**

11.6 `get_reference_frame(self)`

- **function:** Get the base coordinate system
- **Return value:**
 - **0 - base 1 - tool.**

11.7 `set_movement_type(self, move_type)`

- **function:** Set movement type
- **Parameters:**
 - **move_type: 1 - moveI, 0 - moveJ**

11.8 `get_movement_type(self)`

- **function:** Get movement type
- **Return value:**
 - **1 - moveI, 0 - moveJ**

11.9 `get_end_type(self)`

- **function:** Get end coordinate system
- **Return value:**
 - **0 - flange, 1 - tool**

11.10 `set_end_type(self, mode)`

- **function:** Set end coordinate system
- **Parameters:**
 - **mode: int, 0 - flange, 1 - tool**

12. Basic IO Control

12.1 `set_basic_output(self, pin_no, pin_signal)`

- **function:** Set the base IO output
- **Parameters:**
 - **pin_no: pin port number.**
 - **pin_signal: 0 / 1**

12.2 `get_basic_input(self, pin_no)`

- **function:** Read the base IO output
- **Parameters:**
 - **pin_no: (int) pin port number.**

13. WLAN Setting

13.1 `set_ssid_pwd(self, account: str, password: str)`

- **function:** Change connected wi-fi.
- **Parameters:**
 - **account: (str)** new wi-fi account.
 - **password: (str)** new wi-fi password.

13.2 `get_ssid_pwd(self)`

- **function:** Get connected wi-fi account and password.
- **Return value:**
 - **(account, password)**

13.3 `set_server_port(self, port)`

- **function:** Change the connection port of the server.
 - **Parameters:**
 - **port: (int)** The new connection port of the server.
-

[← 上一页](#) | [下一页 →](#)

Videos and Codes for Display

Videos given below are for reference.

Notice: The baud rates are different depending on the type of device. Before using them, refer to the information related thereto. The serial port number can be viewed through [calculator device manager](#) or a serial helper.

Please make sure the robot power on.

1. Get the joint angle

1.1 Shake the firmware to read the angle

```
from pymycobot import MyArmM
import time

myarmmm = MyArmM("COM3")

# Gets the current angle of all joints
angles = myarmmm.get_joints_angle()
print(f"All current joint angles are: {angles}")

time.sleep(0.1)

# Get the angle of joint 1
angle = myarmmm.get_joint_angle(1)
print(f"The current angle of joint 1 is {angle}")

# Get the angle of joint 2
angle = myarmmm.get_joint_angle(2)
print(f"The current angle of joint 2 is {angle}")

# Get the angle of joint 3
angle = myarmmm.get_joint_angle(3)
print(f"The current angle of joint 3 is {angle}")
```

1.2 Coordinate the firmware to read the angle

```
from pymycobot import MyArmMControl
import time

myarmmm = MyArmMControl("COM3")

# Gets the current angle of all joints
angles = myarmmm.get_angles()
print(f"All current joint angles are: {angles}")
```

2. Control the joint to move five points

2.1 Shake the firmware to control the joint to move five points

```
from pymycobot import MyArmM
import time

myarmm = MyArmM("COM3")

# Reset all joints at a speed of 40
myarmm.set_joints_angle([0, 0, 0, 0, 0, 0], 40)
#Wait for all joints to move to the specified position
time.sleep(3)

# Move all joints at the specified angle at a speed of 40
myarmm.set_joints_angle([90, 45, -90, 90, -90, 90], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Reset all joints at a speed of 40
myarmm.set_joints_angle([0, 0, 0, 0, 0, 0], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Move all joints at the specified angle at a speed of 40
myarmm.set_joints_angle([90, 45, -90, 90, -90, 90], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Reset all joints at a speed of 40
myarmm.set_joints_angle([0, 0, 0, 0, 0, 0], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Move all joints at the specified angle at a speed of 40
myarmm.set_joints_angle([90, 45, -90, 90, -90, 90], 40)
# Wait for all joints to move to the specified position
time.sleep(3)
```

2.2 Coordinate the firmware to control the joint to move five points

```

from pymycobot import MyArmMControl
import time

myarmmm = MyArmMControl("COM3")

# Reset all joints at a speed of 40
myarmmm.write_angles([0, 0, 0, 0, 0, 0], 40)
#Wait for all joints to move to the specified position
time.sleep(3)

# Move all joints at the specified angle at a speed of 40
myarmmm.write_angles([90, 45, -90, 90, -90, 90], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Reset all joints at a speed of 40
myarmmm.write_angles([0, 0, 0, 0, 0, 0], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Move all joints at the specified angle at a speed of 40
myarmmm.write_angles([90, 45, -90, 90, -90, 90], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Reset all joints at a speed of 40
myarmmm.write_angles([0, 0, 0, 0, 0, 0], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

# Move all joints at the specified angle at a speed of 40
myarmmm.write_angles([90, 45, -90, 90, -90, 90], 40)
# Wait for all joints to move to the specified position
time.sleep(3)

```

3 The case for control procedures

Program address

https://github.com/elephantrobotics/pymycobot/tree/main/demo/myArm_M&C_demo

Install dependencies

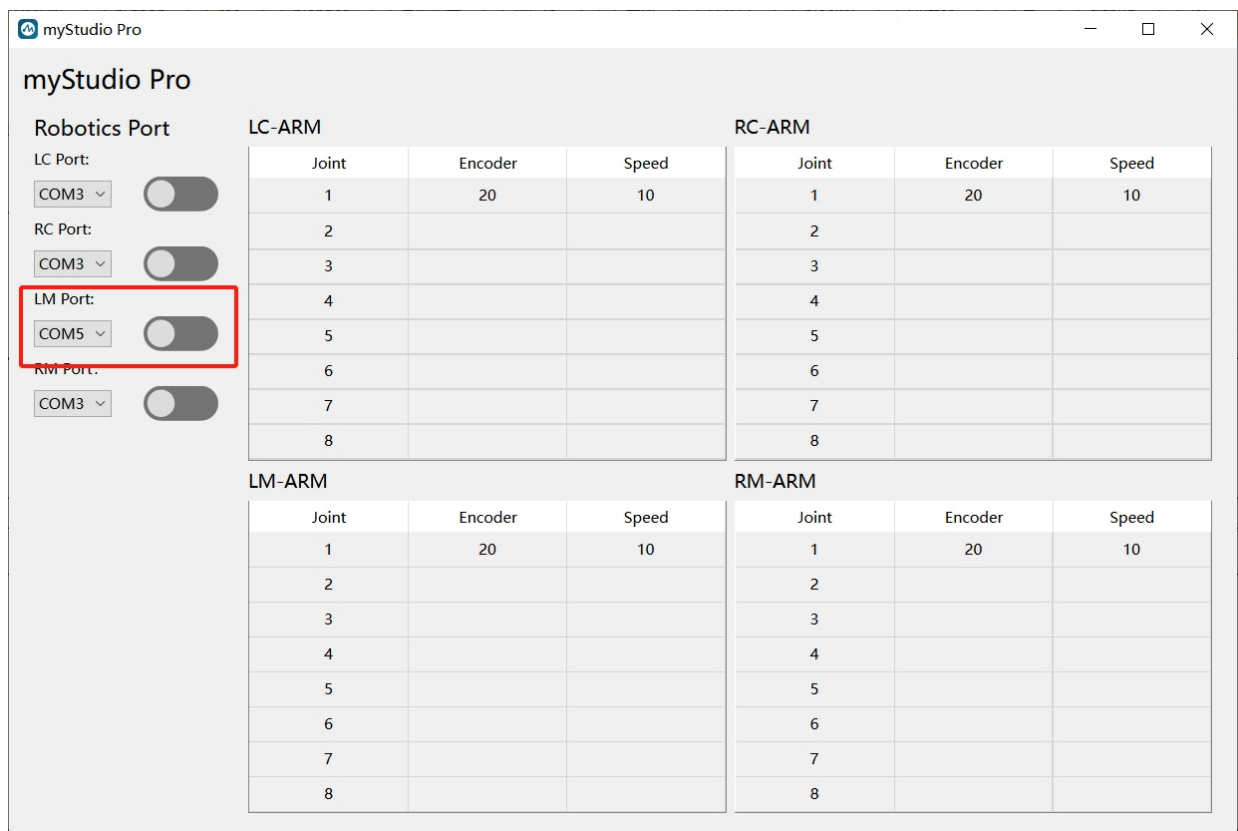
```
pip install -r requirement.txt
```

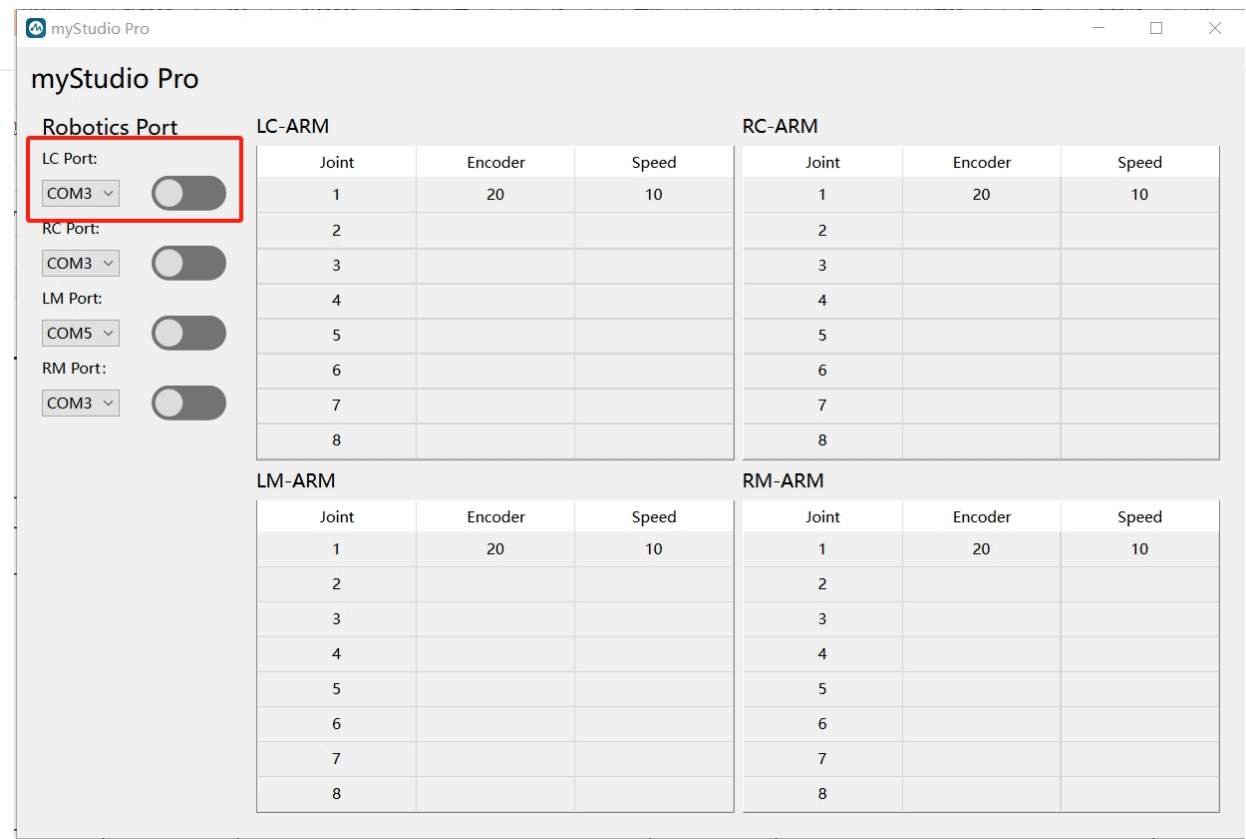
Run the program

```
python main.py
```

Instructions for use of the program

There are sequential requirements for the opening of the serial port: first open the serial port connection of myArmM, and then open the serial port connection of myArmC.





After both serial ports are turned on, you can control the `myArmM` movement by moving `myArmC` .

Myarm M750 version:

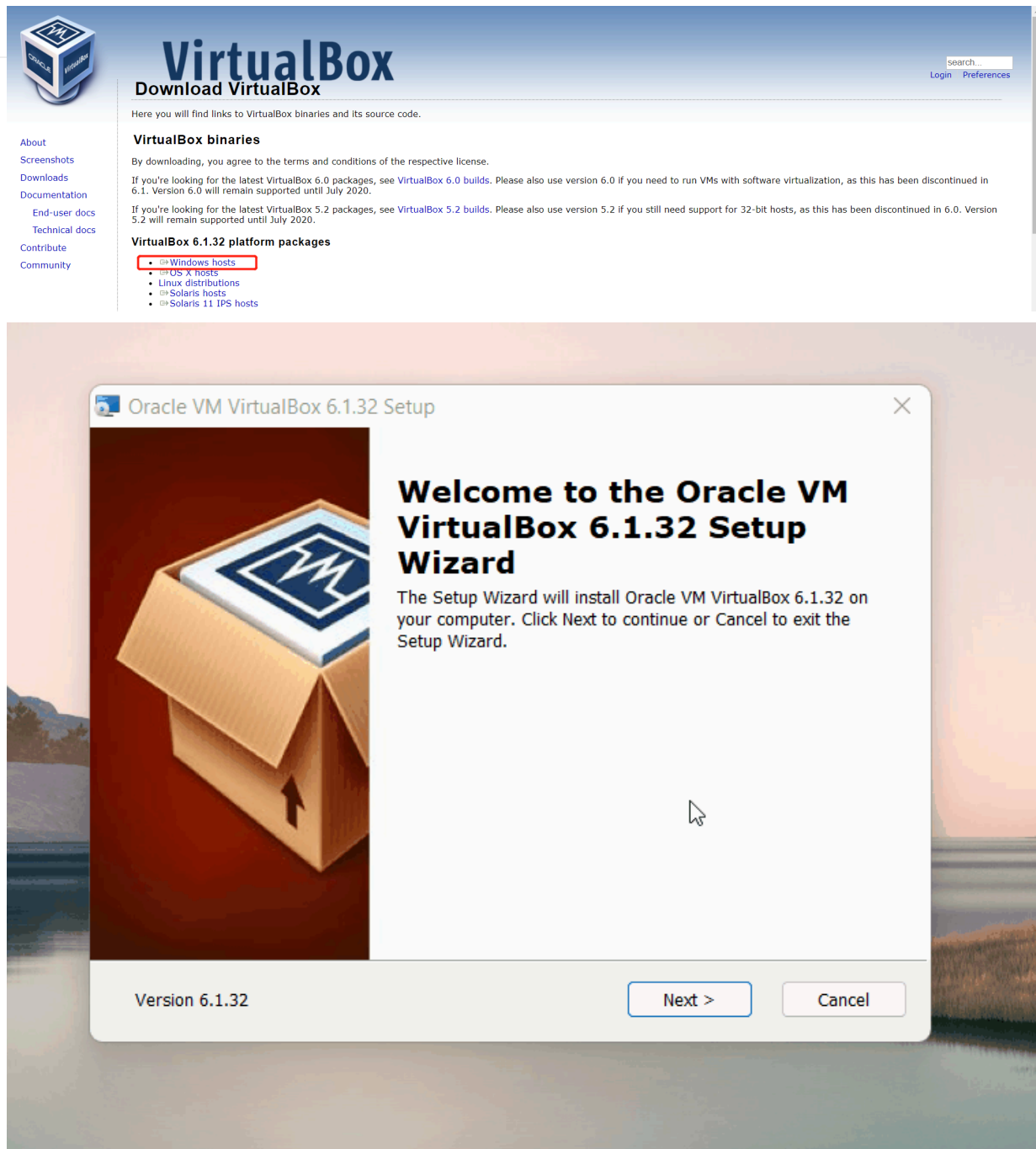
Install different versions of ubuntu in Linux

1 Virtual machine installation

Go to [Official website](#) to download the virtual machine Vltul albox or go to [the official website](#) download the virtual machine Wimwal

Of course, if you already have your virtual machine, you can skip that step.

We chose to download the Virtual box because it is free.

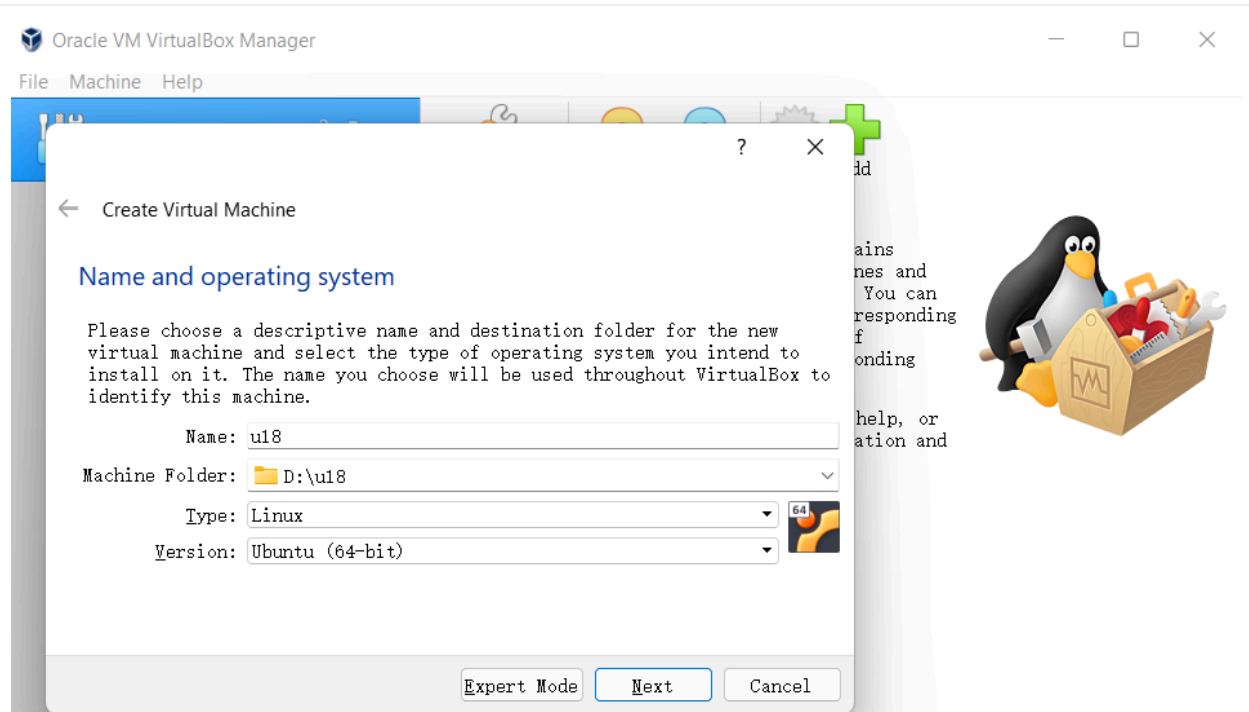


2 Create a new virtual machine

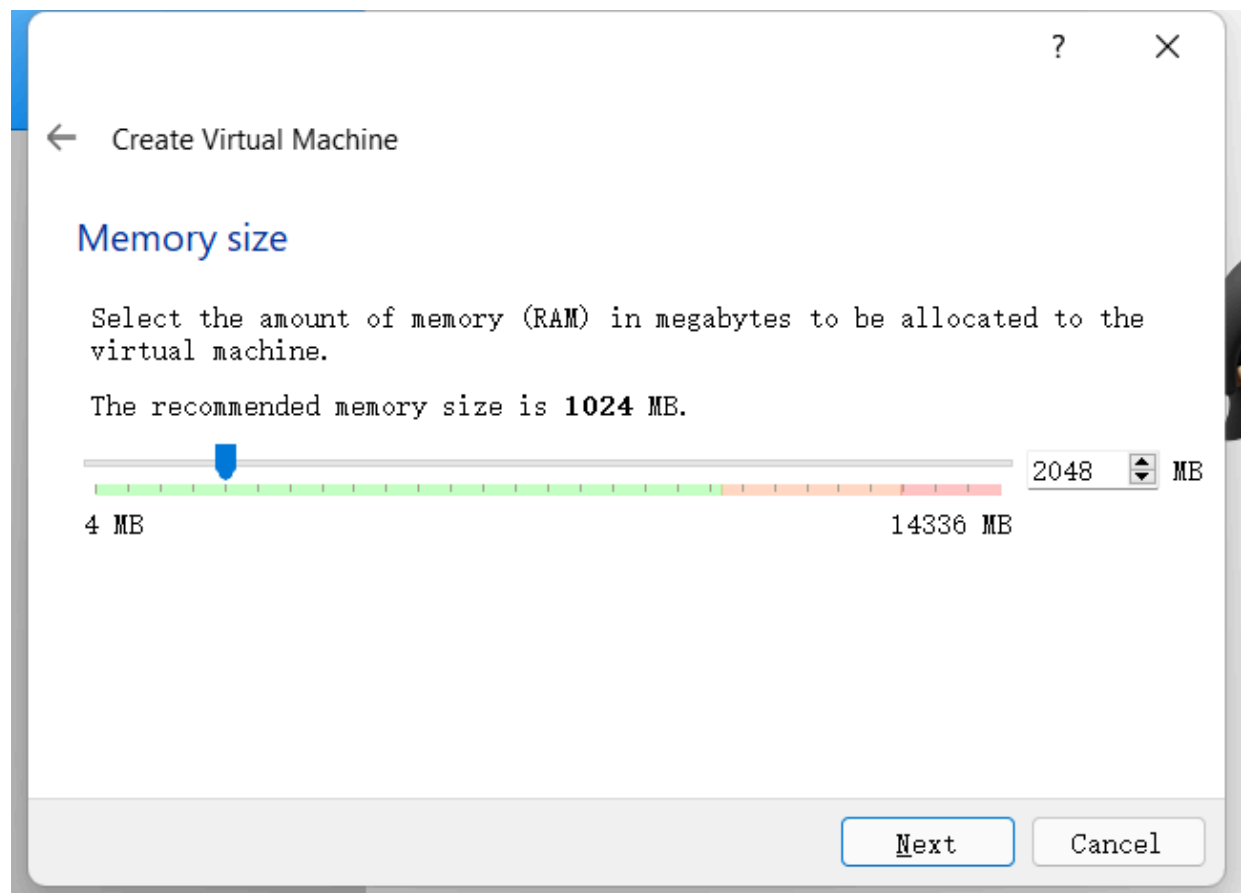
2.1 Create a virtual machine

In Control, select **New**

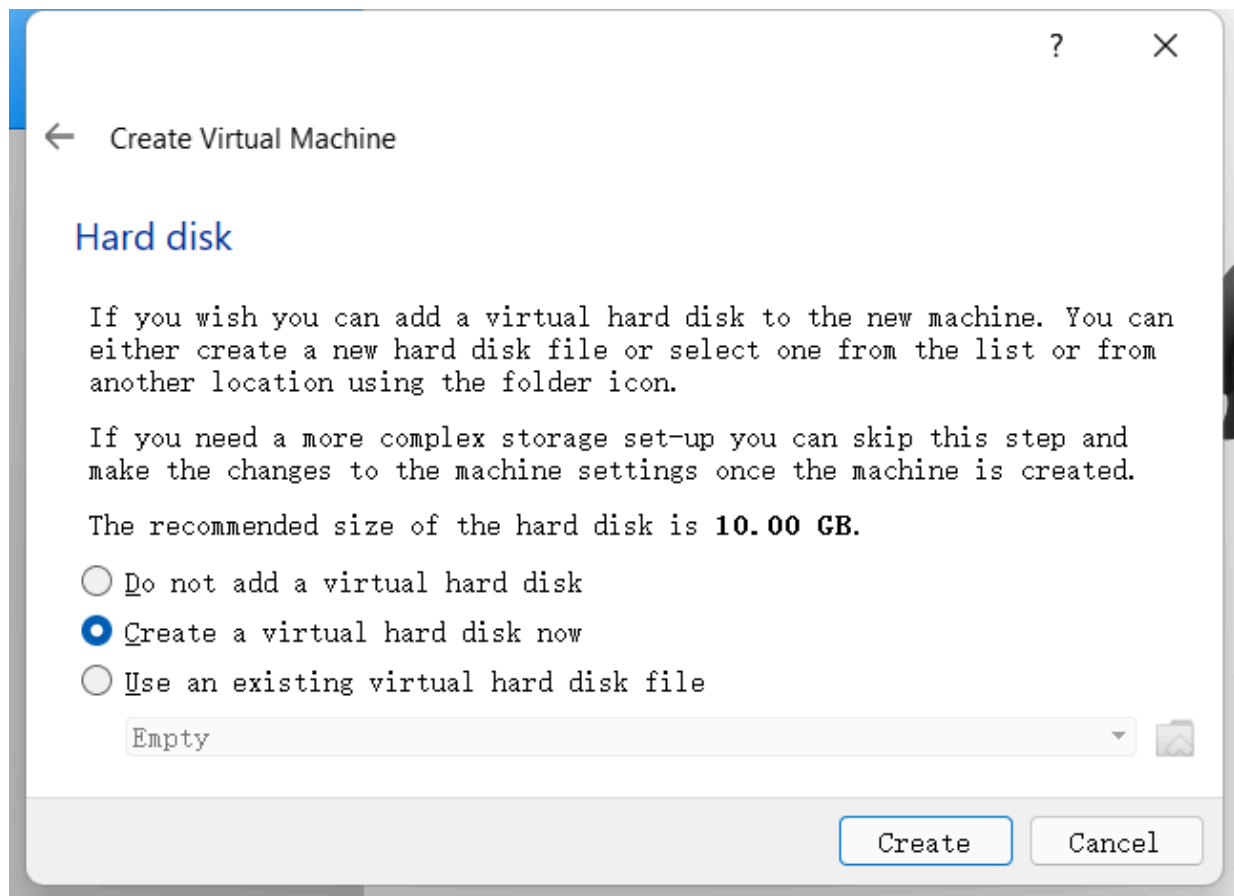
Enter the name of the VM and the location where the VM is stored, select the VM type as Linux, and select the ubuntu 64-bit version to proceed to the next step.



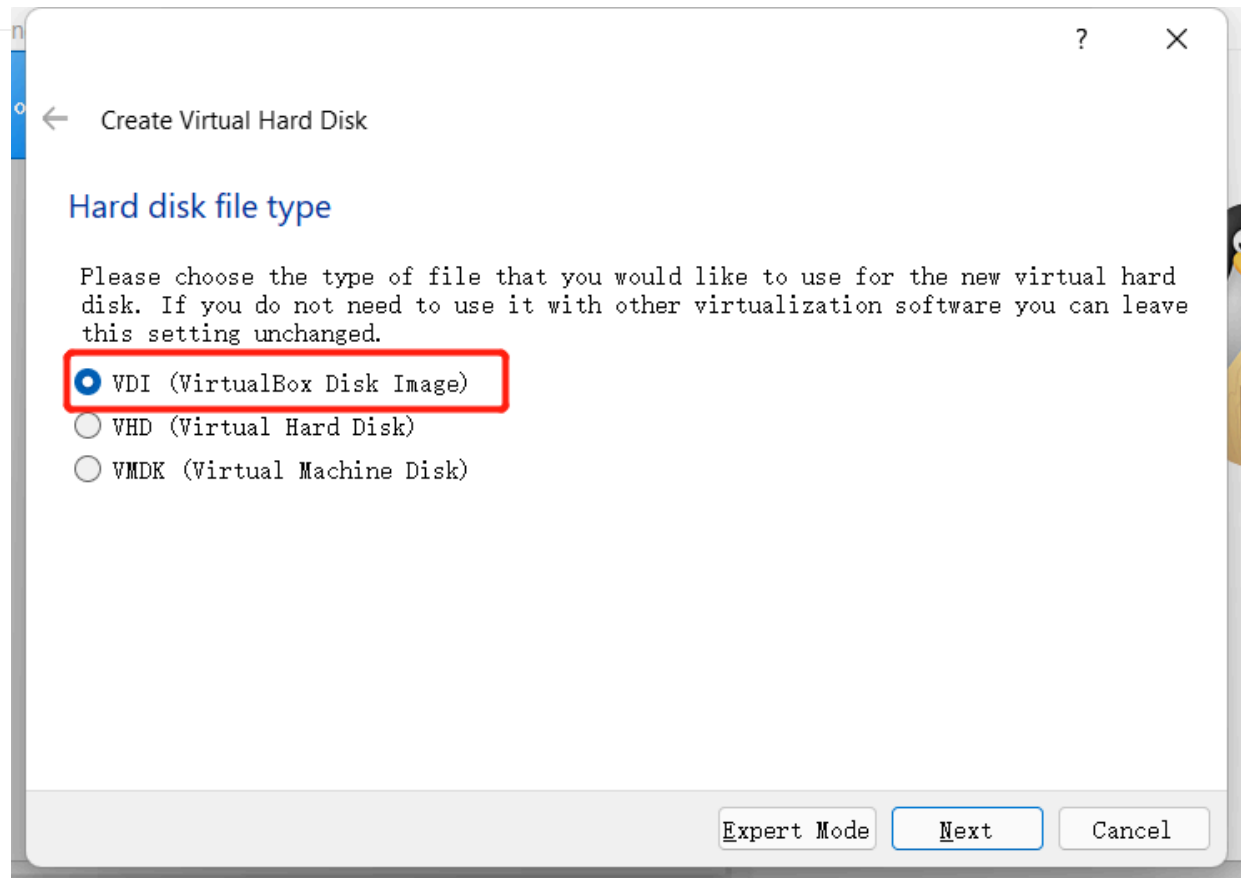
Configure the memory size according to your needs and proceed to the next step.



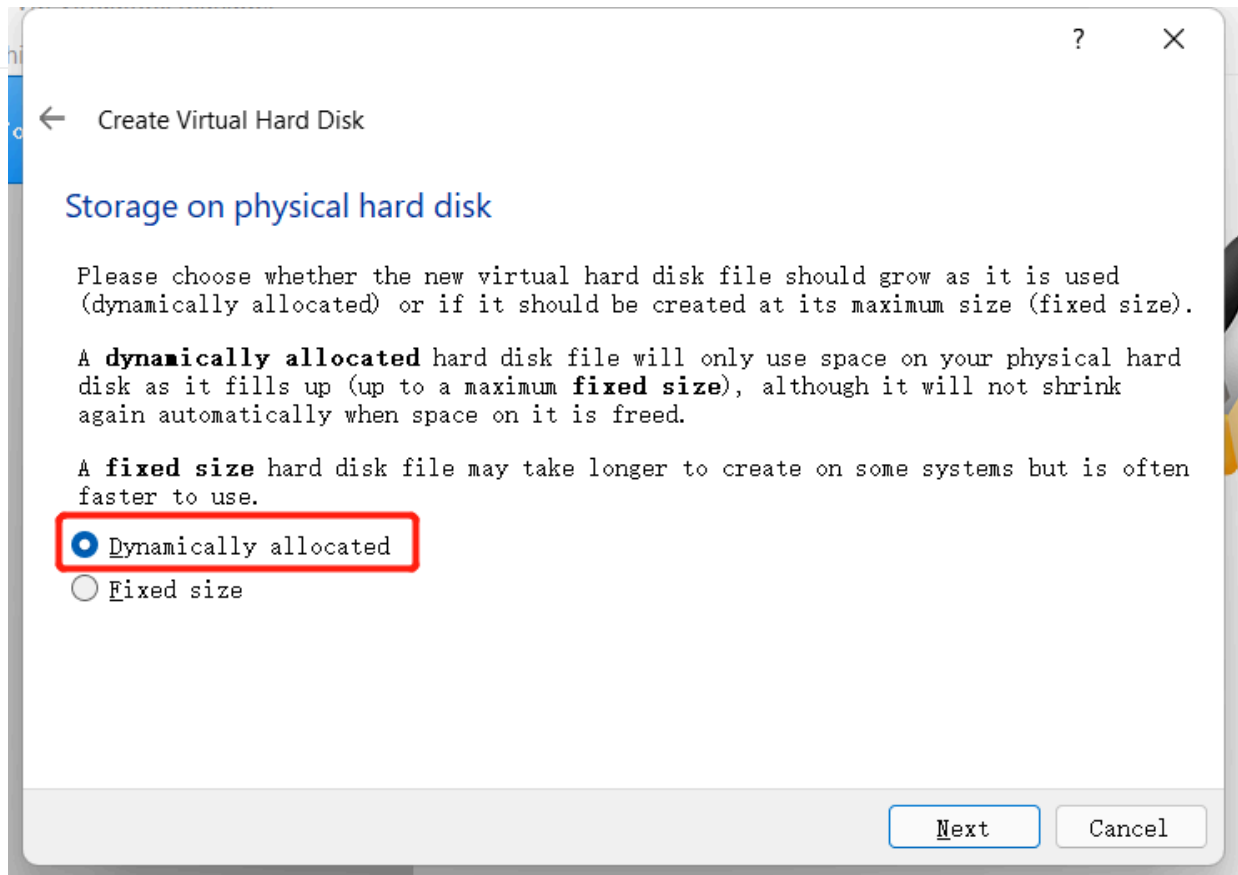
Select **Create virtual hard disk** now to create it.



Select **VDI** for Virtual Hard Disk Type and proceed to the next step.



Allocate the size of the virtual hard disk, since the ubuntu system needs to be installed, and it will also operate in the system, it is recommended that the size should not be less than 20G.



Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.



2.2 Import the Ubuntu system

2.2.1 Download the Ubuntu system

Please choose the Ubuntu version according to your needs to install it.

Note: ROS2 requires version 20.04 to be downloaded.

16.04 version

18.04 version

20.04 version

The installation method and process are the same for all three versions, and version 18.04 is used as an example for installation

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

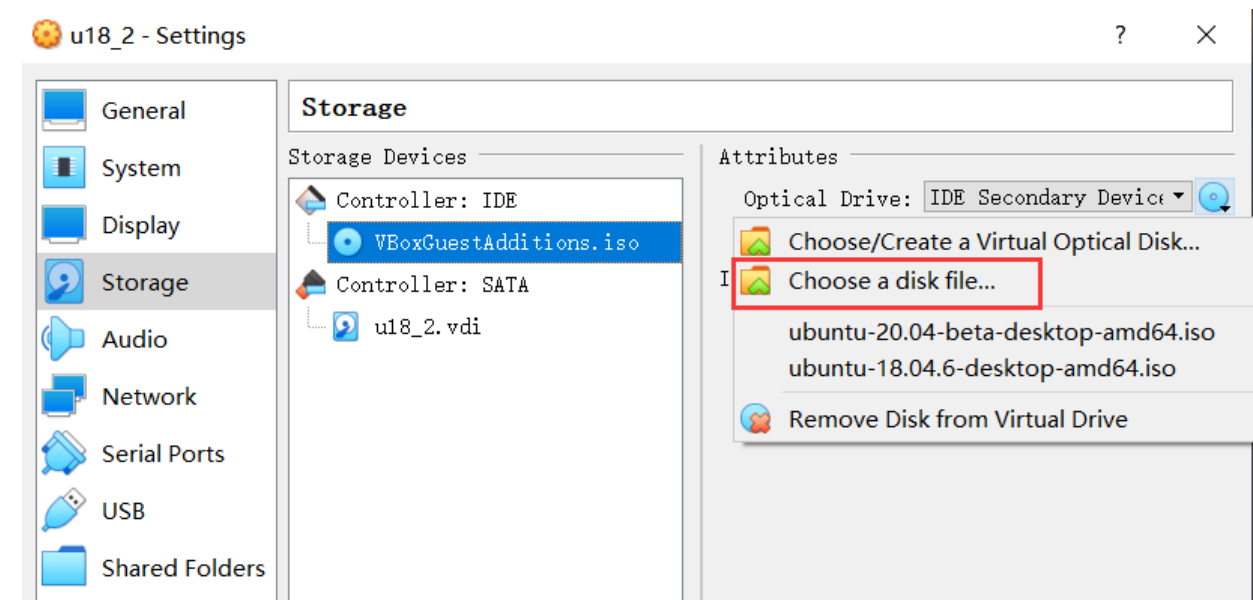
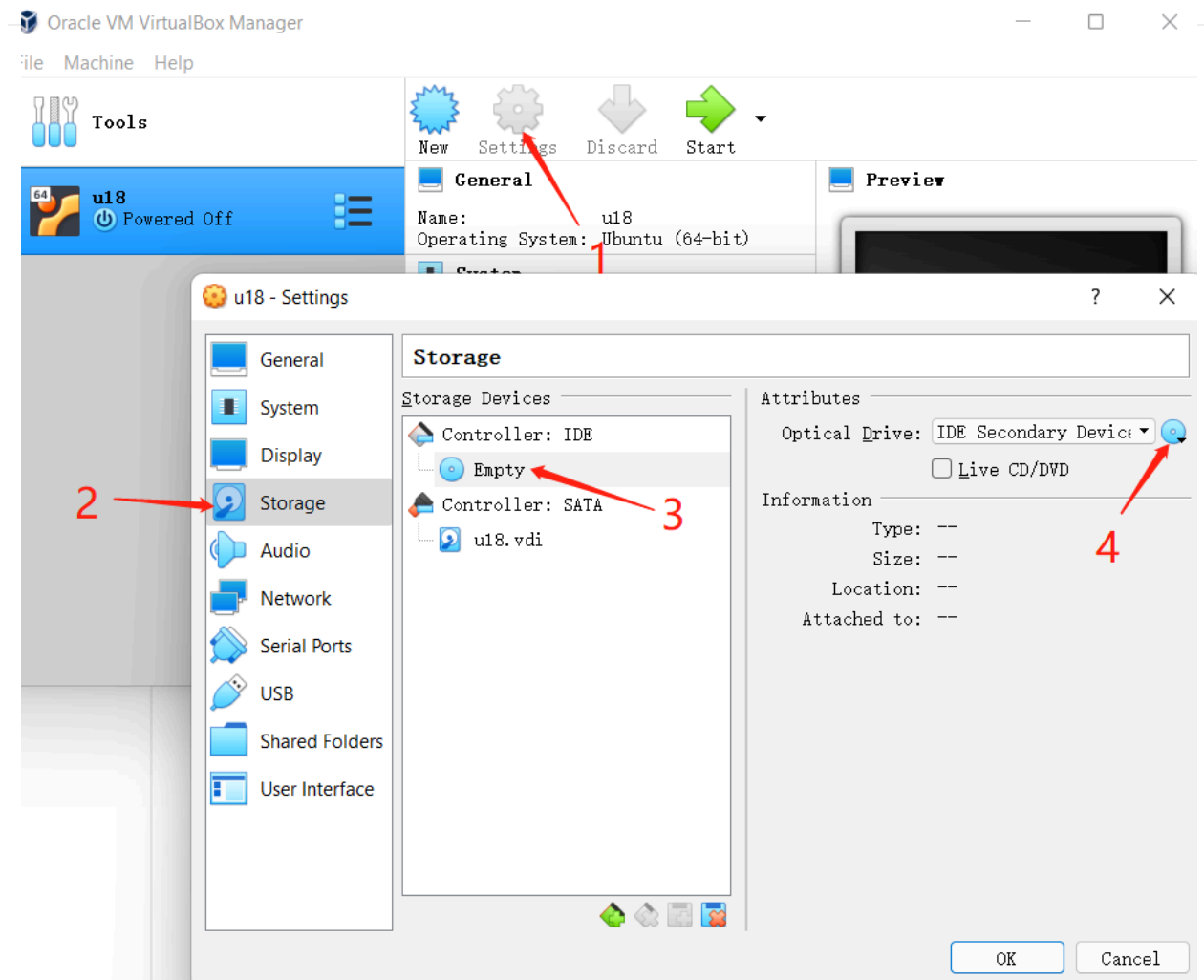
Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (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)

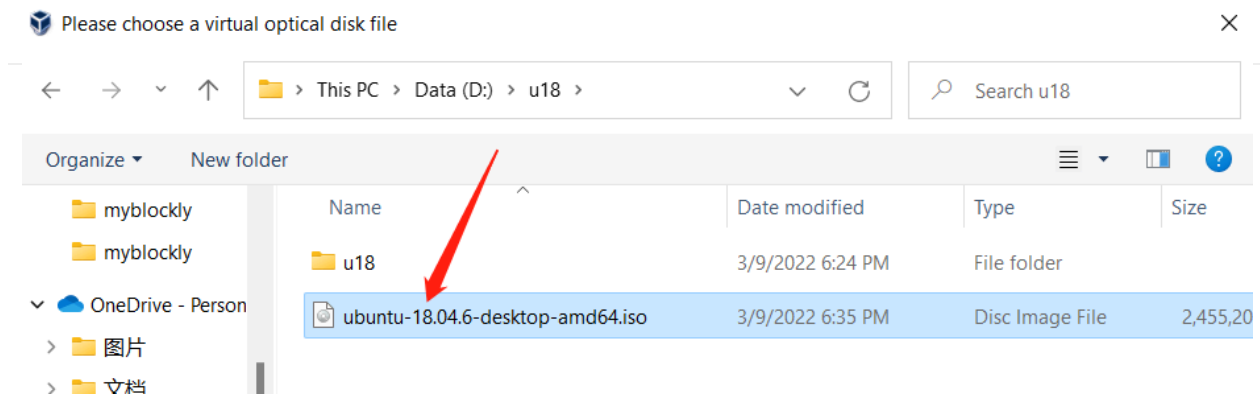
After the download is completed, there is a file as shown in the figure:

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 to the virtual machine

Locate the previously installed virtual machine in the virtual box, go to **Settings**, and assign a disc to the controller in **Storage**:





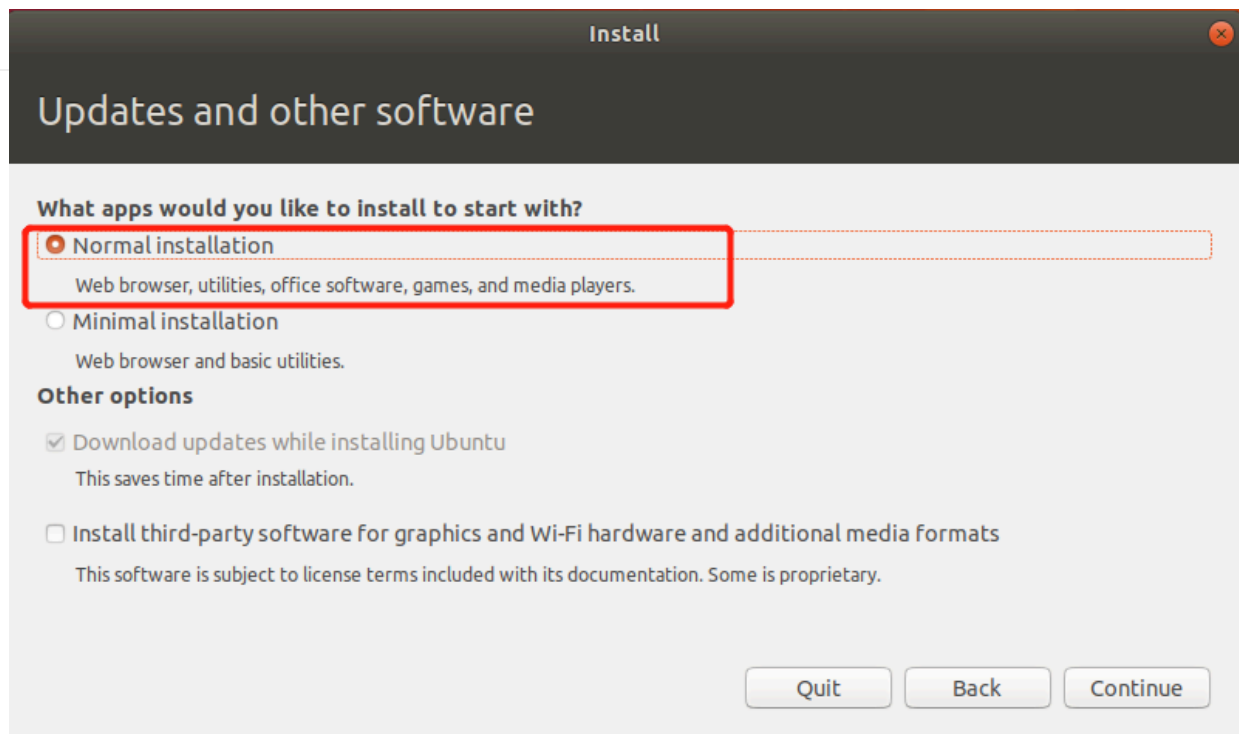
Then open the virtual machine for ubuntu installation and click start.

2.2.3 ubuntu installation

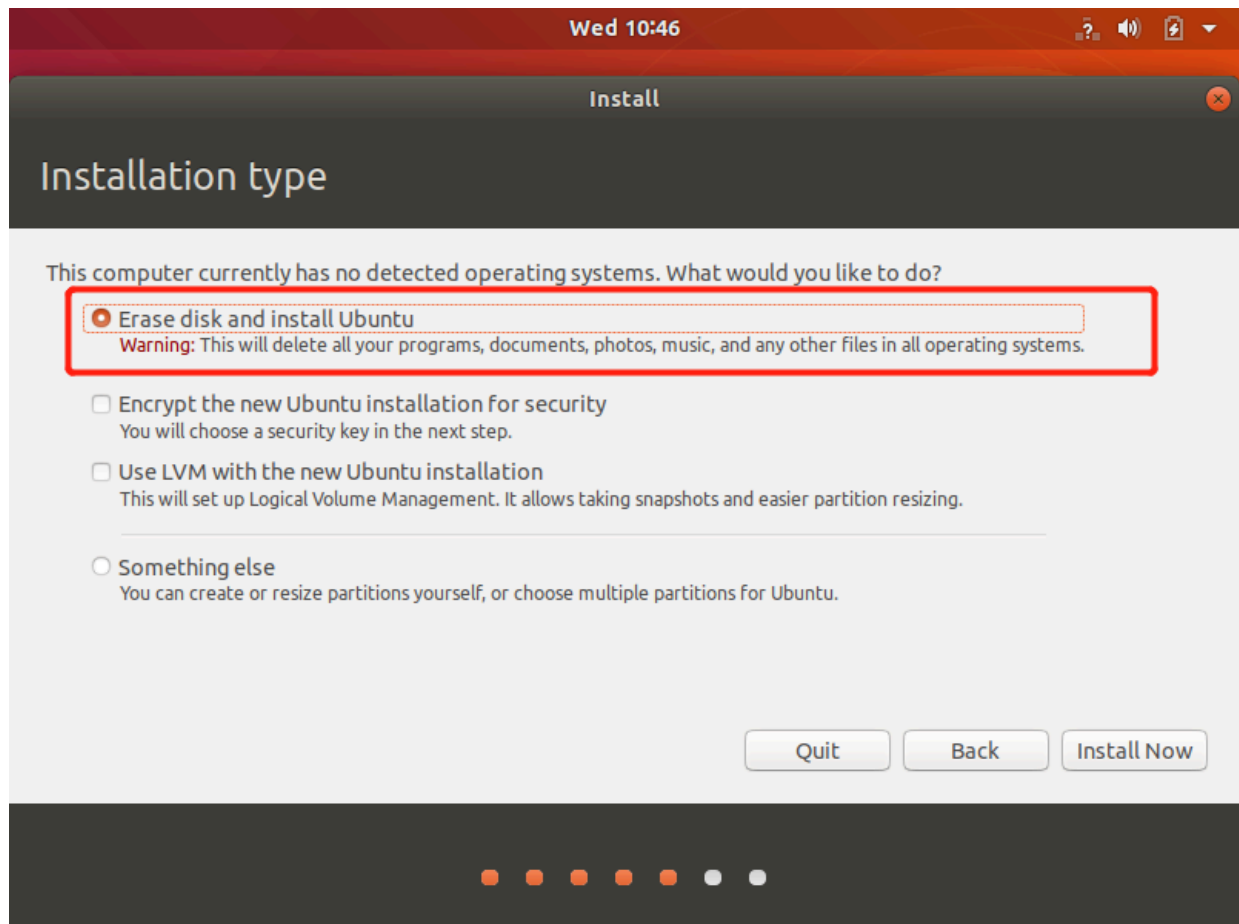
Wait for the system to boot, enter the **Welcome** interface, select "Chinese (Simplified)", and click the "Install Ubuntu" button;



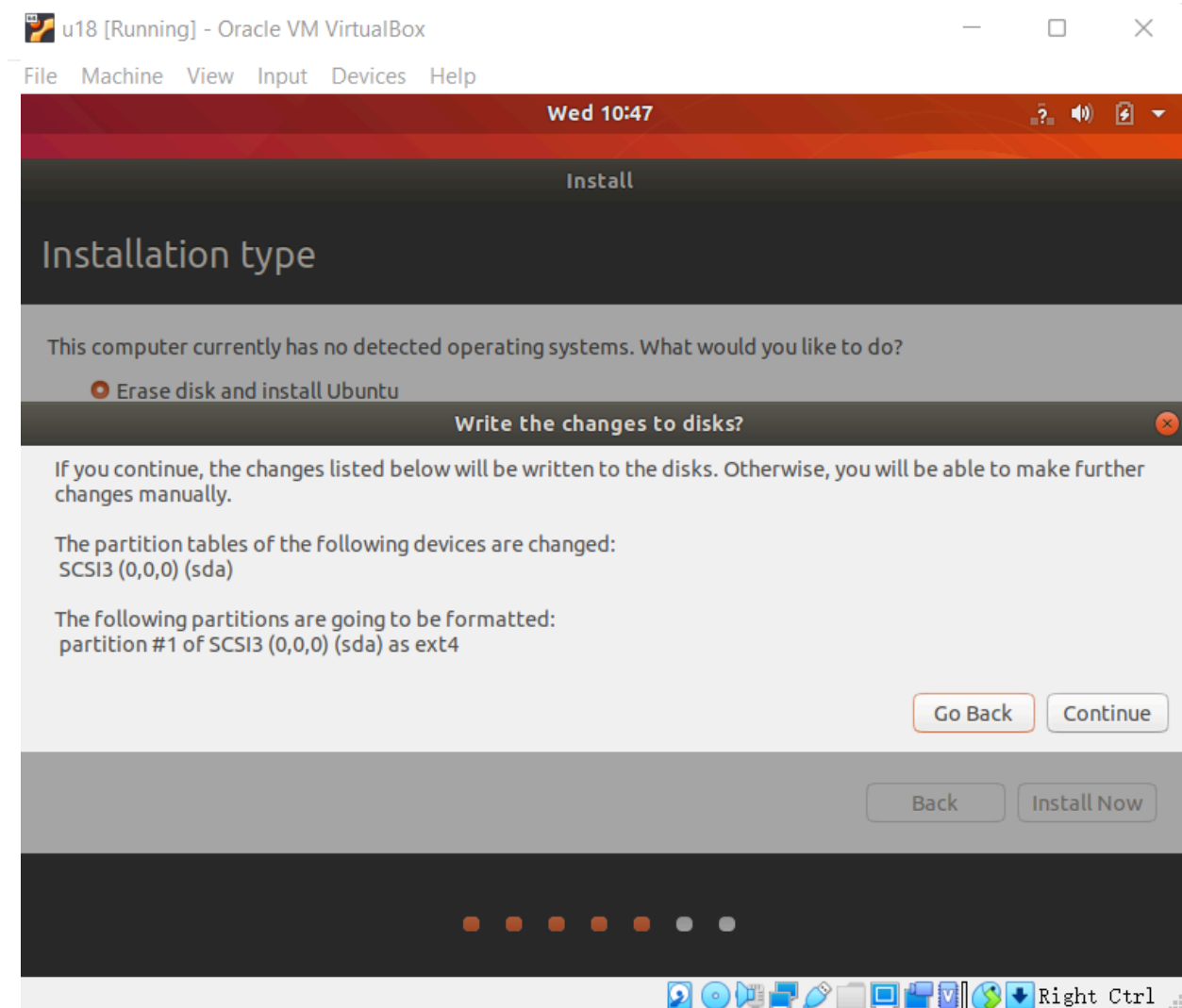
Click "Continue" button;



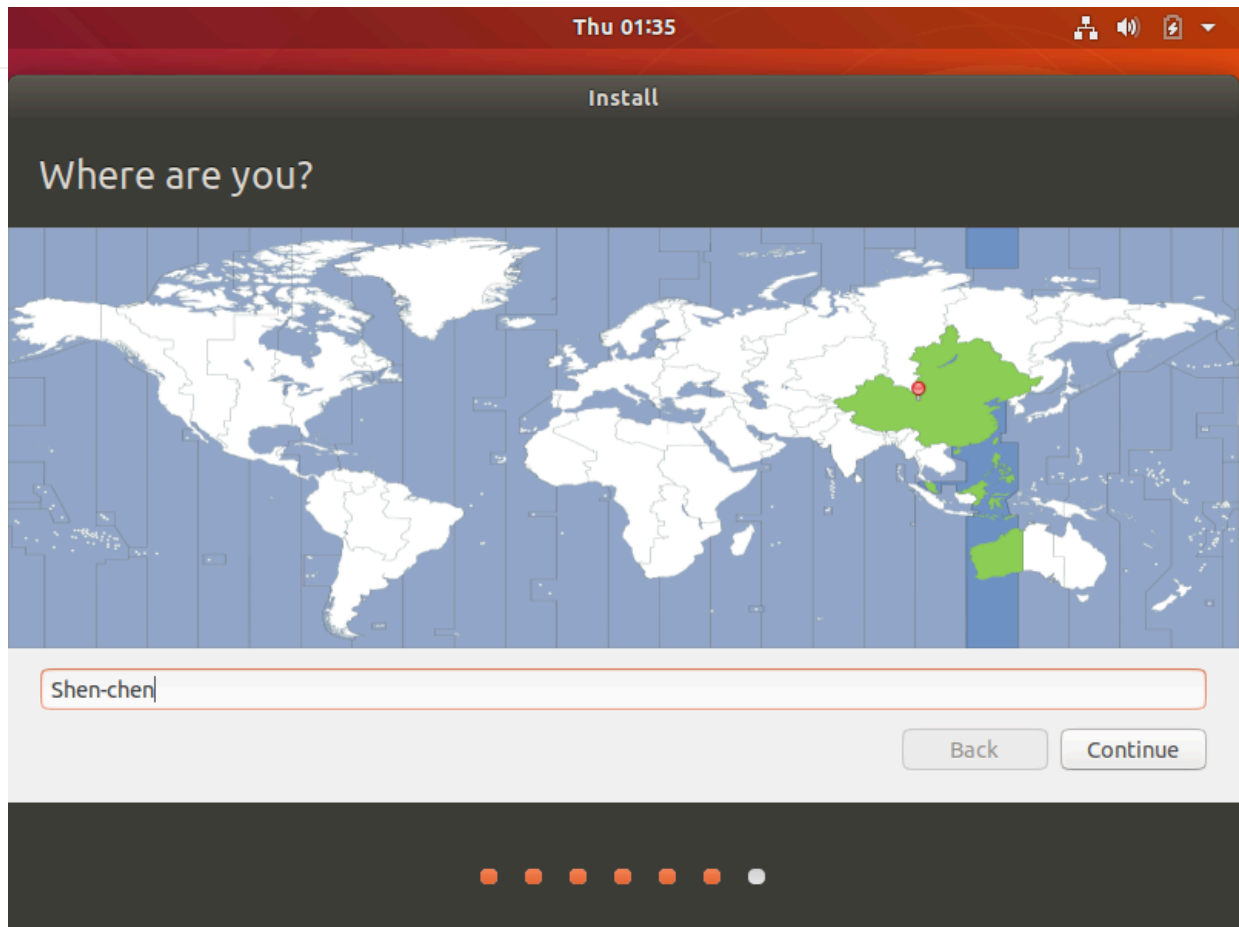
Select the "Clear entire disk and install Ubuntu" option, and click the "Install Now" button;



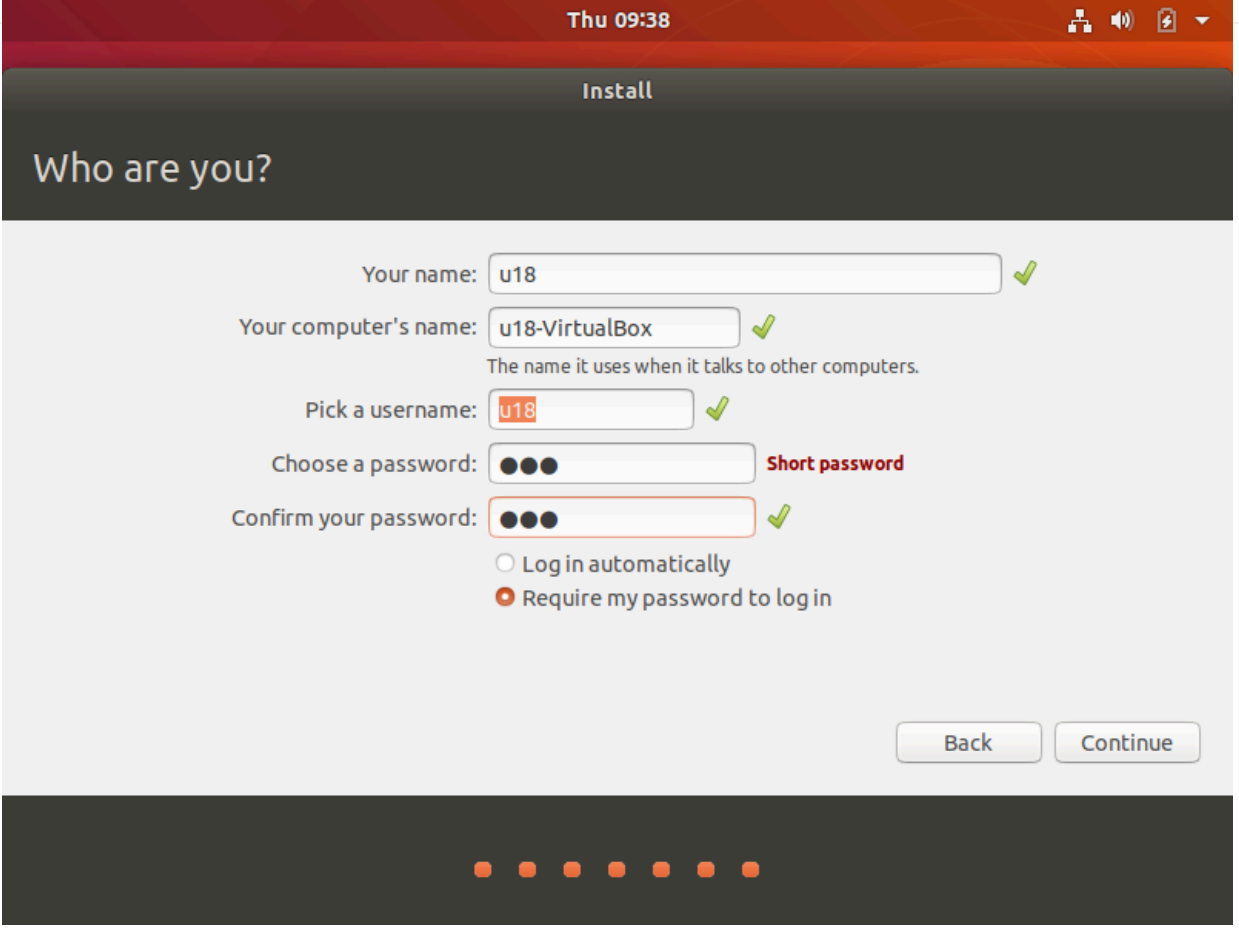
Click the "Continue" button in the pop-up dialog box;



Set a geographic position and click the "Continue" button;



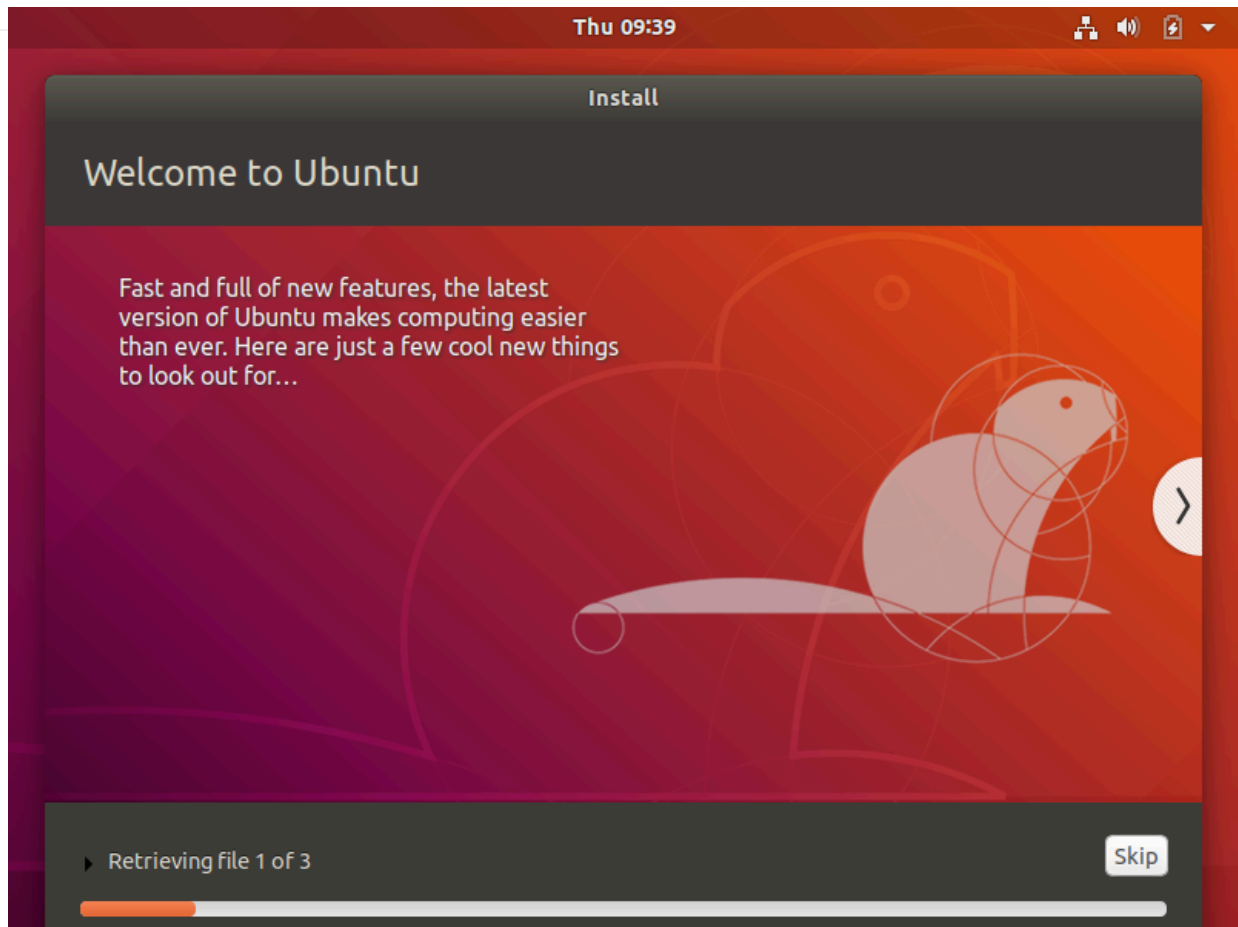
Set a user name and password, and click the "Continue" button;



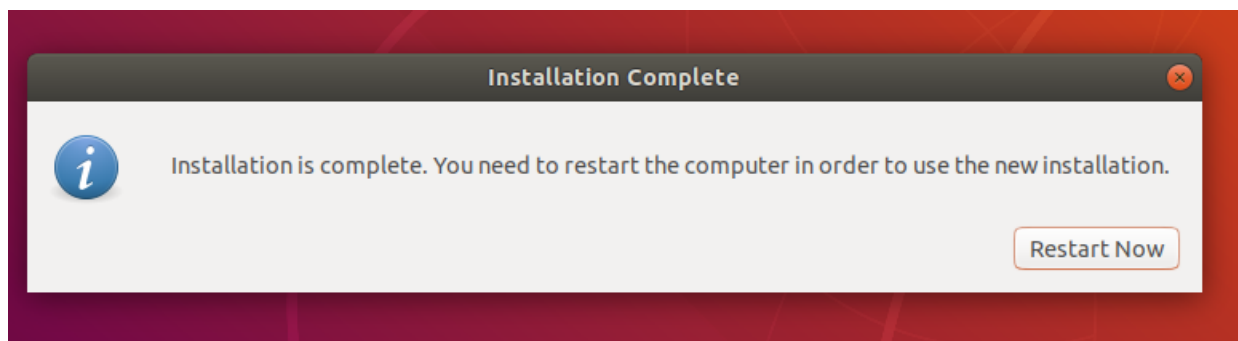
The image shows the 'Who are you?' screen of the Ubuntu installer. The window title is 'Install'. The top bar shows the time 'Thu 09:38' and system icons. The main heading is 'Who are you?'. The form contains the following fields and options:

- Your name:** A text box containing 'u18' with a green checkmark to its right.
- Your computer's name:** A text box containing 'u18-VirtualBox' with a green checkmark to its right. Below it is the text 'The name it uses when it talks to other computers.'
- Pick a username:** A text box containing 'u18' with a green checkmark to its right.
- Choose a password:** A password box with three dots and a red error message 'Short password' to its right.
- Confirm your password:** A password box with three dots and a green checkmark to its right.
- Log in options:** Two radio buttons: 'Log in automatically' (unselected) and 'Require my password to log in' (selected).
- Buttons:** 'Back' and 'Continue' buttons at the bottom right.
- Progress bar:** A horizontal bar at the bottom with seven orange dots, where the first four are filled.

Enter the system installation interface, and wait patiently;



After installation is complete, click the "Restart now" button in the pop-up dialog box to complete the installation.



1 ROS Environment building

1.1 Installing ROS

Building a basic development environment requires the installation of the robot operating system (short for ROS), MoveIt and a git version manager. The installation methods and procedures are described respectively below.

1.1.1 Selecting a version

There is a one-to-one relationship between ROS and ubuntu. Different versions of ubuntu correspond to different versions of ROS. The reference website is as follows: <http://wiki.ros.org/Distributions>

- Here are the ROS versions supported by Ubuntu:
 - Ubuntu 16.04 / ROS Kinetic
 - Ubuntu 18.04 / ROS Melodic
 - Ubuntu 20.04 / ROS Noetic

Install the ROS version corresponding to the Ubuntu version you have installed.

If the versions are different, the downloading will fail. The system we choose here is Ubuntu 18.04, so the corresponding ROS version is ROS Melodic.

- NOTE: At present, we do not provide any reference for installing ROS in windows. If necessary, refer to <https://www.ros.org/install/>

1.1.2 Installing ROS

1 Adding a software source

There is no ROS software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS. Open a console terminal (shortcut key: Ctrl+Alt+T) and input 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, it is recommended to select a nearby mirror source to replace the above 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'
```

Here you will be asked to input a user password. Just input the user password set when Ubuntu is installed.

2 Setting a key

Configuring a public key. This step is to let the system confirm that our path is safe, so that there is no problem in downloading the file. Otherwise it will be deleted immediately after downloading:

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

The execution result is displayed as follows:

```

u18@u18-VirtualBox:~$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
Executing: /tmp/apt-key-gpghome.ncwvc7DwDj/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:80 --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
gpg: key F42ED6FBAB17C654: public key "Open Robotics <info@osrfoundation.org>" imported
gpg: Total number processed: 1
gpg:         imported: 1
u18@u18-VirtualBox:~$

```

3 Installation

After adding a new software source, you need to update the software source list. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

```
sudo apt-get update
```

Execute **Installing ROS**. Open a console terminal (shortcut key: Ctrl+Alt+T) and input 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
```

It is recommended to install the complete ROS here to prevent the lack of libraries and dependencies.

The installation process takes a long time, so please wait with patience.

- If the following error message appears on the console terminal during the installation process, you need to replace the software source list in /etc/apt/sources.list.

```

Fetched 323 MB (in 24min 34s (3,401 KB/s))
E: Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/universe/w/wxwidgets
3.0/libwxbase3.0-0v5_3.0.4+dfsg-3_amd64.deb Undetermined Error [IP: 91.189.91.3
9 80]

```

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

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

- Replace all the official repositories in **sources.list** with the following Alibaba Cloud repositories:

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

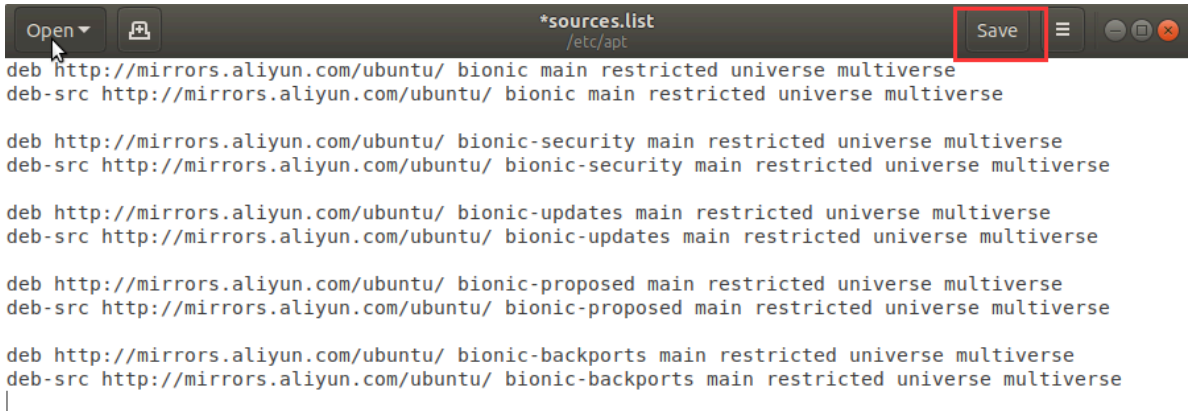
deb <http://mirrors.aliyun.com/ubuntu/> focal-proposed main restricted universe multiverse

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 is complete, the contents of the `sources.list` file are as follows, click Save and Exit.



- To update the list of software sources, 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, so please wait with patience.

4 Configuring the ROS environment to the system

rosdep allows you to easily install the source codes you want to compile, or the system dependencies required by some ROS core components. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: Ctrl+Alt+T).

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

If the system of the Ubuntu installed by you is version 20.04, use the command `sudo apt install python3-rosdep` to it install. After completion, execute the rosdep initialization command.

```
u18@u18-VirtualBox:~$ sudo rosdep init
[sudo] password for u18:
sudo: rosdep: command not found
u18@u18-VirtualBox:~$ sudo apt install python-rosdep
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  docutils-common docutils-doc libpython-stdlib python python-catkin-pkg
  python-catkin-pkg-modules python-chardet python-dateutil python-docutils
  python-minimal python-olefile python-pil python-pkg-resources
  python-pygments python-pyparsing python-roman python-rosdep-modules
  python-roscdistro python-roscdistro-modules python-rospkg
  python-rospkg-modules python-setuptools python-six python-yaml python2.7
  python2.7-minimal sgml-base xml-core
Suggested packages:
  python-doc python-tk fonts-linuxlibertine | ttf-linux-libertine
  texlive-lang-french texlive-latex-base texlive-latex-recommended
  python-pil-doc python-pil-dbg ttf-bitstream-vera python-pyparsing-doc
  python-setuptools-doc python2.7-doc binfmt-support sgml-base-doc debhelper
The following NEW packages will be installed:
  docutils-common docutils-doc libpython-stdlib python python-catkin-pkg
  python-catkin-pkg-modules python-chardet python-dateutil python-docutils
  python-minimal python-olefile python-pil python-pkg-resources
  python-pygments python-pyparsing python-roman python-rosdep
  python-rosdep-modules python-roscdistro python-roscdistro-modules
  python-rospkg python-rospkg-modules python-setuptools python-six
  python-yaml python2.7 python2.7-minimal sgml-base xml-core
0 upgraded, 29 newly installed, 0 to remove and 119 not upgraded.
```

initialize rosdep:

```
sudo rosdep init
```

If the following error message 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.
```

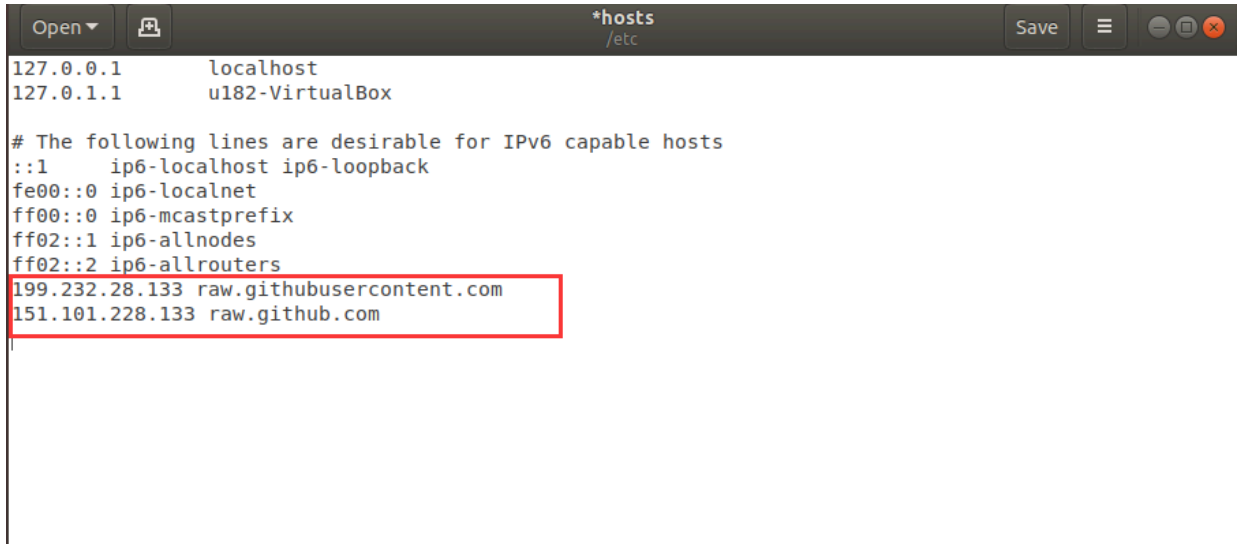
Solution: Modify the hosts file and enter the following command in the console terminal:

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

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

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

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



```
*hosts
/etc

127.0.0.1    localhost
127.0.1.1    u182-VirtualBox

# The following lines are desirable for IPv6 capable hosts
::1         ip6-localhost ip6-loopback
fe00::0     ip6-localnet
ff00::0     ip6-mcastprefix
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
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 is completed, in order to avoid re-validating the ROS function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: Ctrl+Alt+T):

5 Set up the ros environment

- Bash

Execute the following commands:

```
# Ubuntu 16.04
```

```
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
```

- Zsh If you replace bash with zsh, then:

```
# Ubuntu16.04
```

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.zshrc
```

```
## Ubuntu18.04
```

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

```
## Ubuntu20.04
```

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

```
source ~/.zshrc
```

6 Installing a ROS extra dependency

Input the following command in the terminal to install a ROS extra dependency, and open a console terminal (shortcut key: Ctrl+Alt+T):

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

If your Ubuntu system is version 20.04, please execute the following command to install:


```
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
```

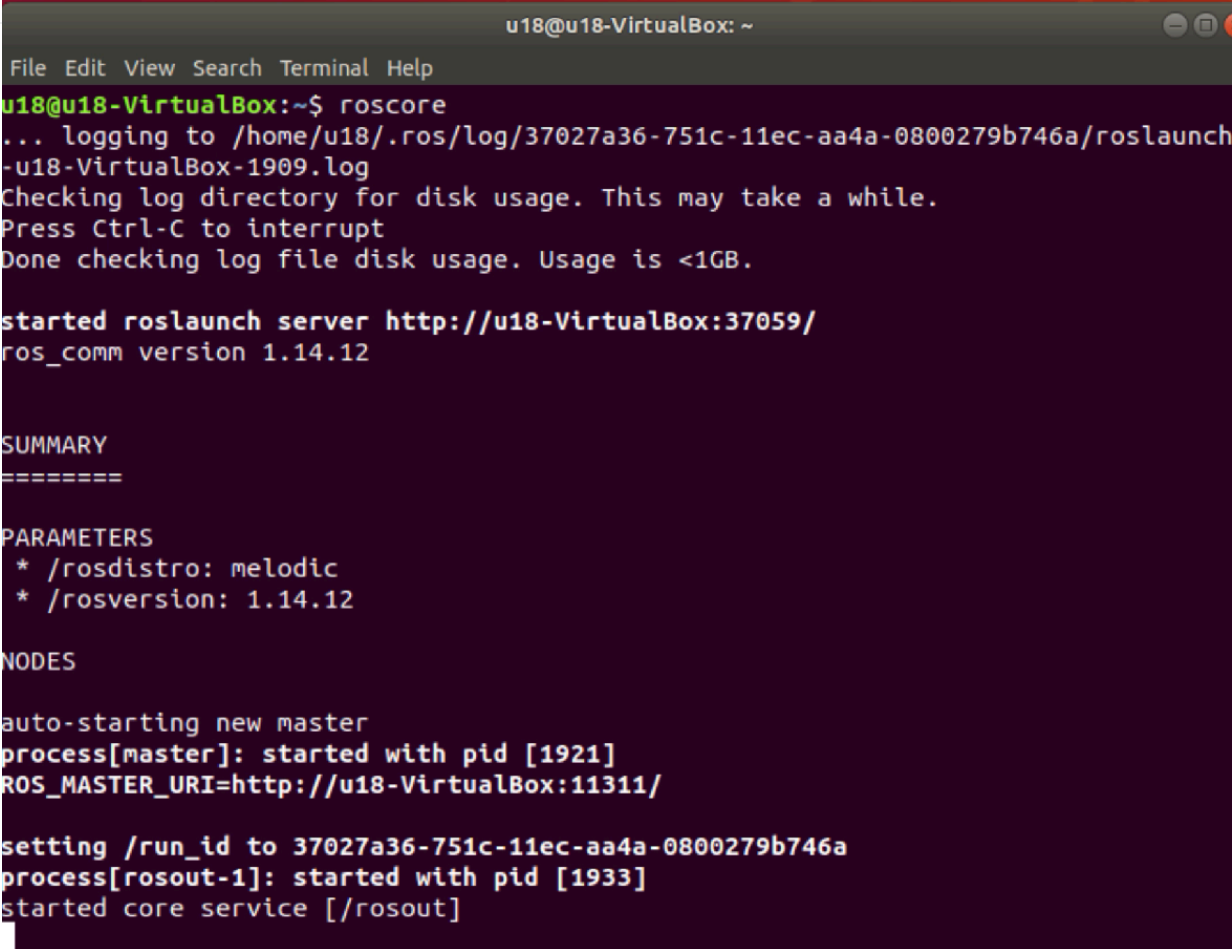
1.1.3 Verifying ROS

The startup of the ROS system requires a ROS Master, that is, a node manager. We can input the `roscore` command in the terminal to start the ROS Master.

To verify whether ROS has been installed successfully, open a console terminal (shortcut key: Ctrl+Alt+T) and execute the following command in the terminal:

```
roscore
```

When the following interface is displayed, it means that ROS has been installed successfully.



```

u18@u18-VirtualBox: ~
File Edit View Search Terminal Help
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
* /roscdistro: melodic
* /rosversion: 1.14.12

NODES

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

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

```

The roscore command starts a node manager, which is used for node management. In a ros system, there is and only one node manager, which is the prerequisite for the operation of the ros node, so before executing the start of the ros node, roscore needs to be executed in the first step.

For more detailed installation instructions, you can refer to the official installation instructions by visiting the website: <http://wiki.ros.org/ROS/Installation>

1.2 MoveIt Installation

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

1.2.1 Updating the software source list

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to update the software source list:

```
sudo apt-get update
```

1.2.2 Installing MoveIt

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to execute the installation of MoveIt:

```
# Ubuntu16.04
```

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

```
# Ubuntu 18.04
```

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

```
# Ubuntu20.04
```

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

1.3 git Installation

1.3.1 Adding a software source

Add the software source for installing git to the software source list of ubuntu, open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window:

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

1.3.2 Updating the software source list

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to update the software source list:

```
sudo apt-get update
```

1.3.3 Installing git

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to execute the installation of git:

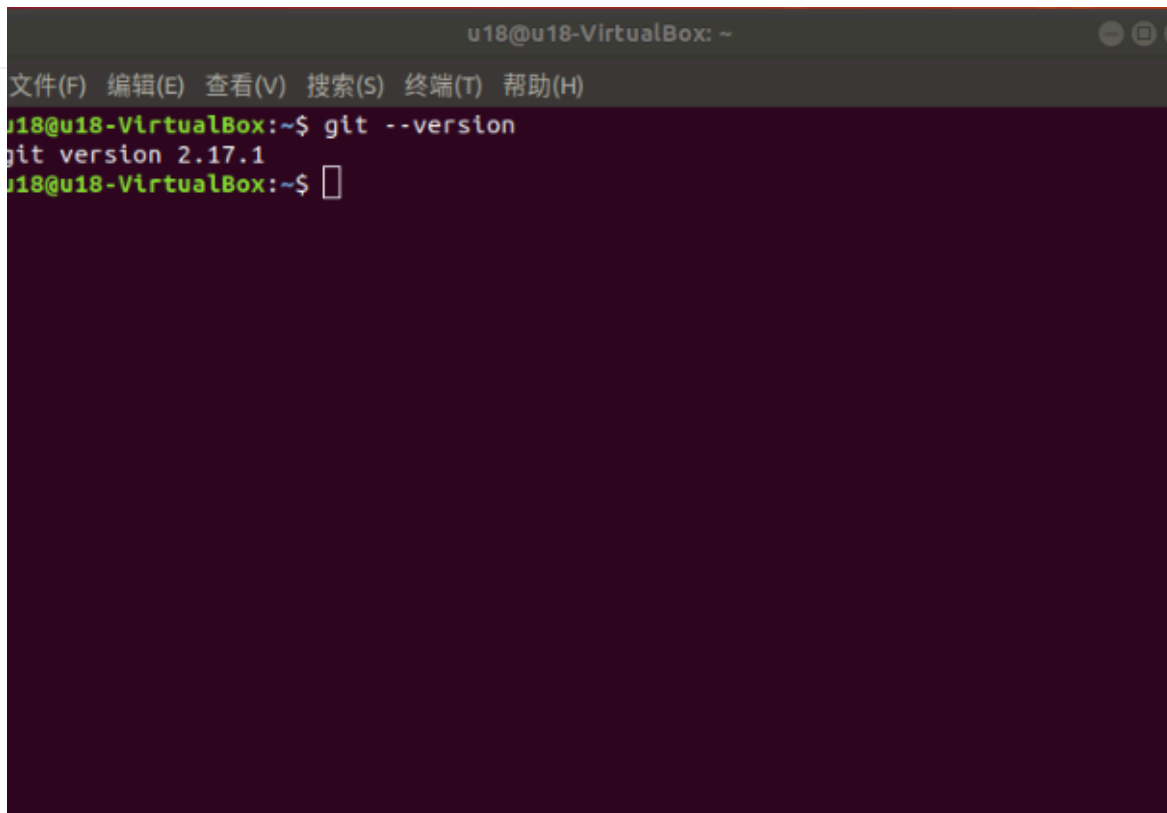
```
sudo apt-get install git
```

1.3.4 Verifying git

Read the git version, open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window:

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, 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 on the next 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:~$
```

1.3.5 Use

During the subsequent downloading of the ros package, you need to use git. For the use of git, refer to the following link:

<https://git-scm.com/book/zh/v2>

<https://www.runoob.com/git/git-tutorial.html>

2 MyarmM750 Installation

The MyarmM750 is a ROS package launched by ElephantRobotics that adapts to its desktop six-axis robotic arm Myarm series.

Project Address: https://github.com/elephantrobotics/mycobot_ros/tree/myarm-c650/myArm/myarm_m

2.1 precondition

Before installing the package, make sure you have a ROS workspace.

Here we give a sample command to **create a workspace**, The default is `catkin_ws`, Open a console terminal (Ctrl+Alt+T), On the command line, enter the following command:

```
mkdir -p ~/catkin_ws/src # Create folders
```

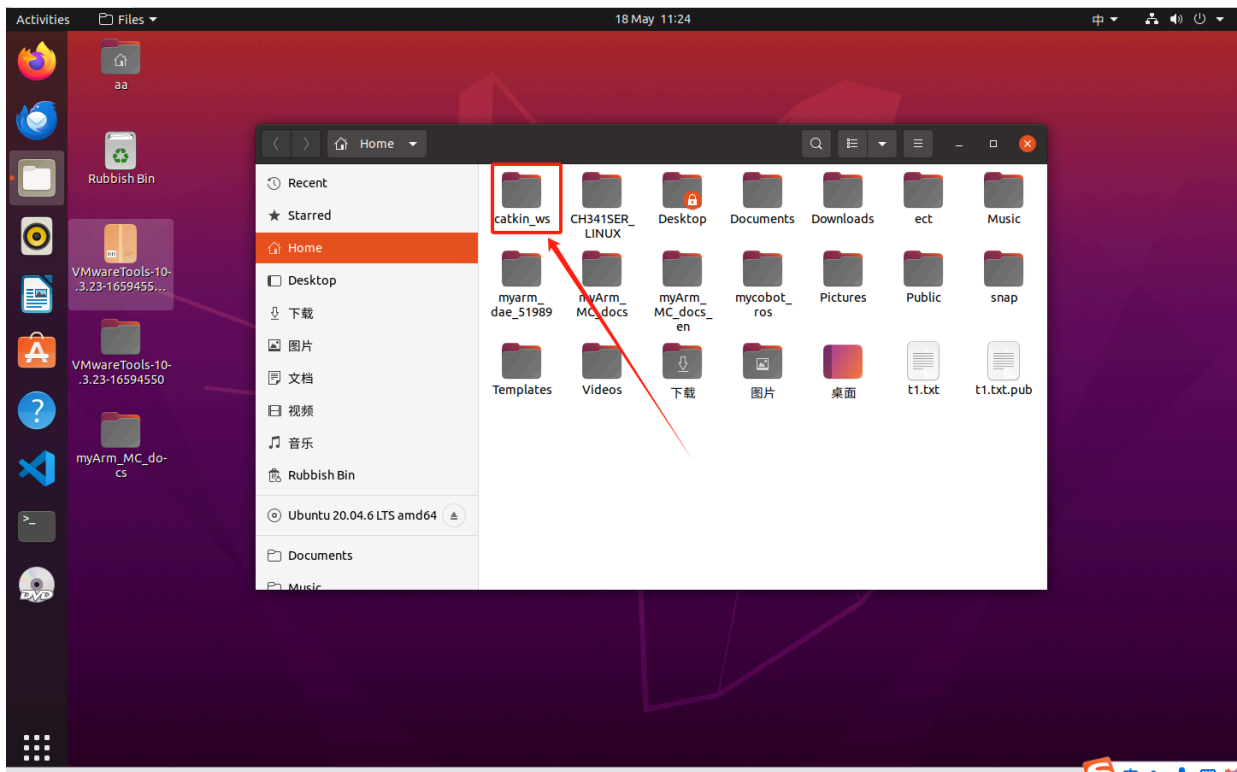
```
cd ~/catkin_ws/src # Go to the folder
```

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

```
cd .. # Return to the parent directory
```

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

Once created, a folder will be generated



Add the environment for the workspace

Bash

The official 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

source ~/.bashrc
```

2.2 Installation

NOTE:

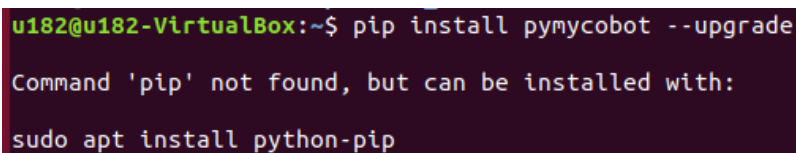
This package relies on ROS and MoveIT, make sure to install ROS and MoveIT successfully before use.

The interaction between this package and the real robotic arm relies on the PythonApi - pymycobot

The API project locations are: <https://github.com/elephantrobotics/pymycobot>

Quick Installation: `pip install pymycobot --upgrade`

When executing the Pip Instal Pimicobo-Upgrad command, if the following error message appears:



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

Enter the following command to install pip when prompted:

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

If your Ubuntu system is version 20.04, run the `sudo apt install python3-pip` installation and run the terminal again after the installation is complete

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

The installation method depends on Git, so make sure you have Git installed on your computer.

The official default ROS1 workspace is `catkin_ws`.

Open the terminal:

```
cd ~/catkin_ws/src # Go to the src folder in your workspace
```

```
git clone https://github.com/elephantrobotics/mycobot\_ros.git # Clone the code on github
```

```
cd .. # Return to the workspace
```

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

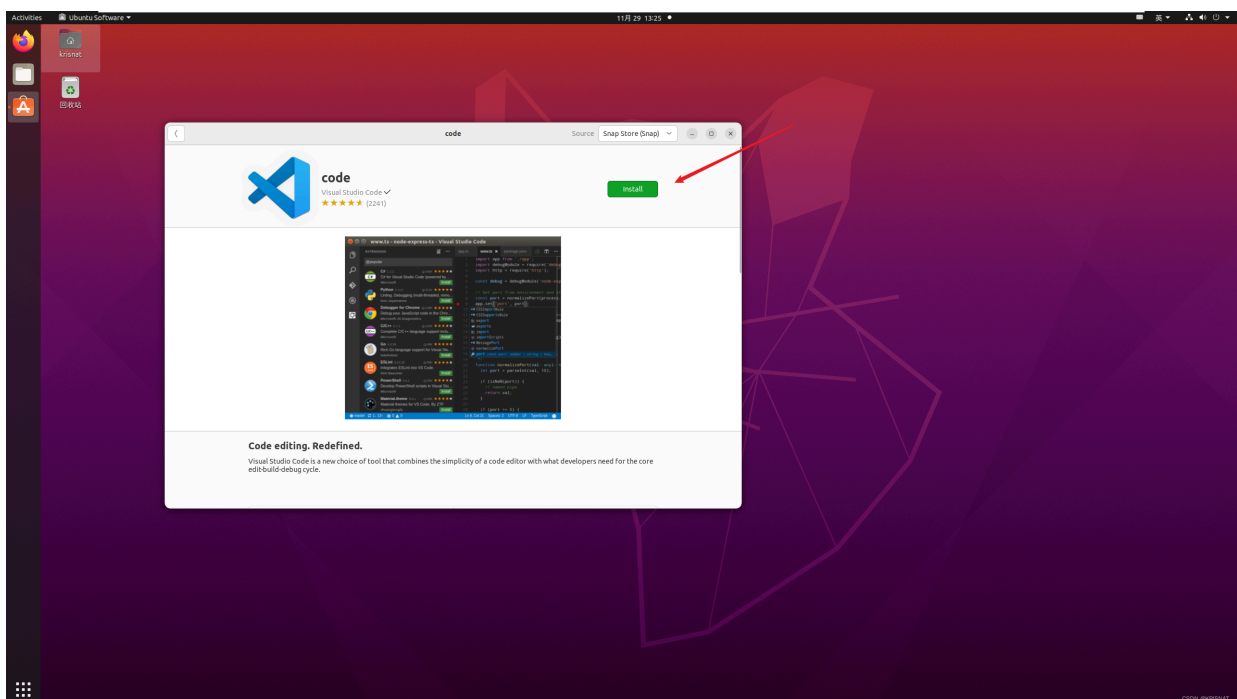
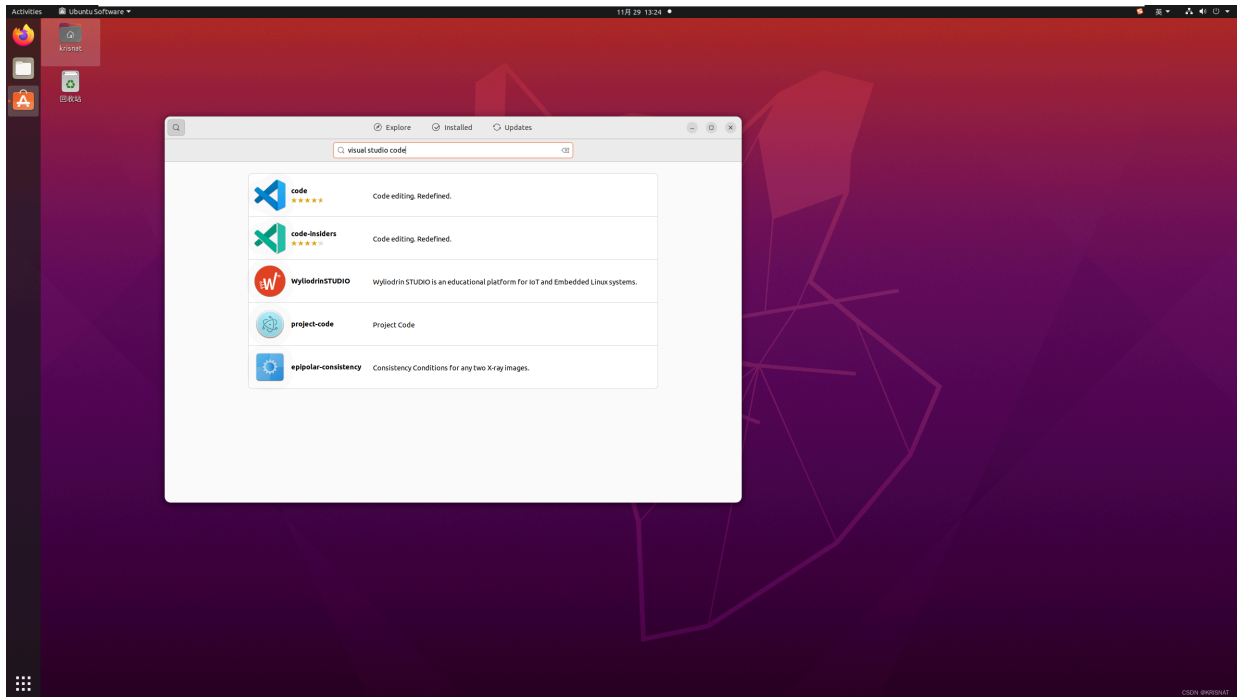
```
cd ..
```

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

At this point, the environment construction is partially completed

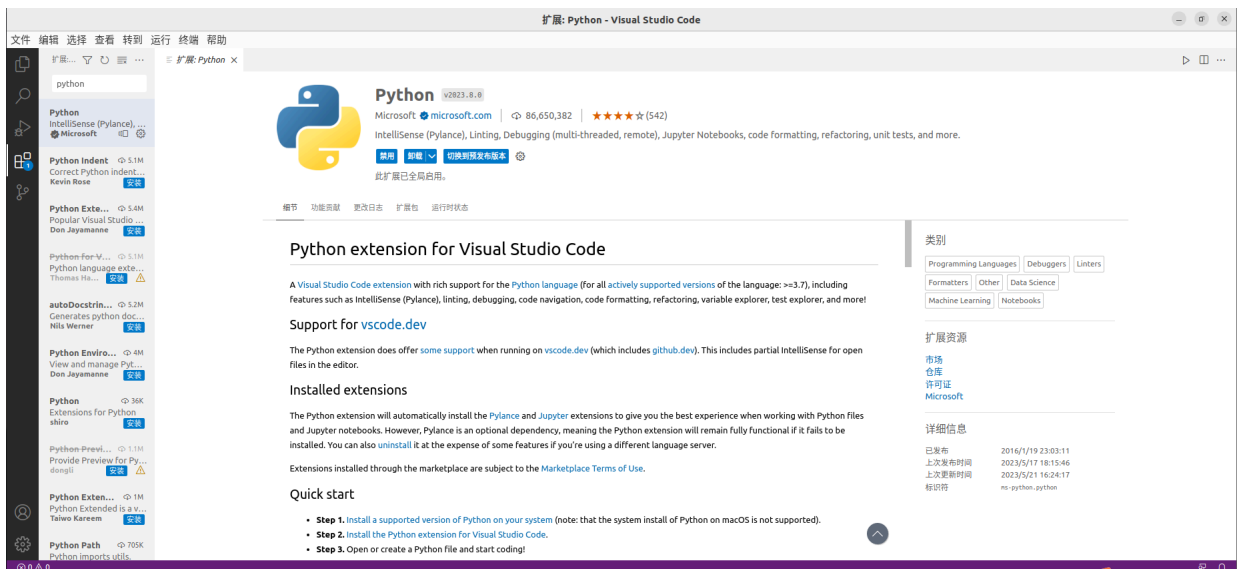
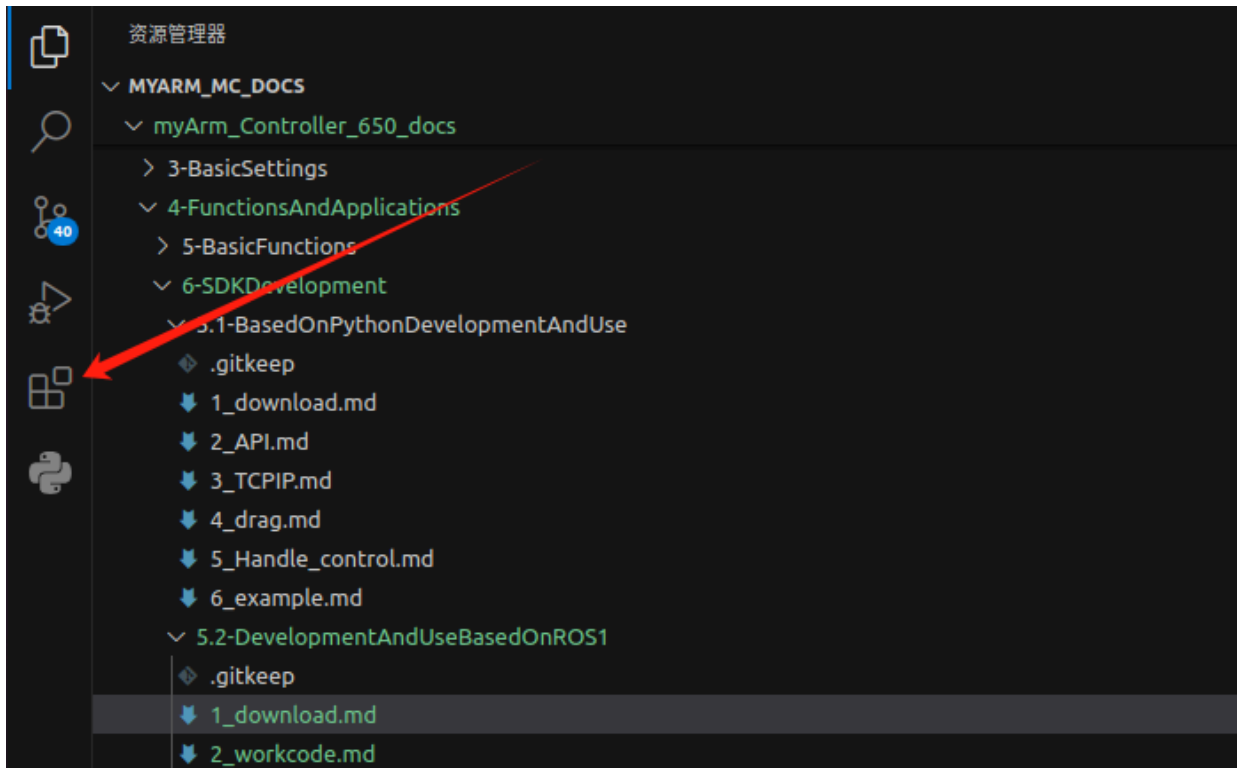
For the subsequent programming, we also need to download one VS code

Here we will choose the most convenient method: Installing in Ubuntu Software [Easy & Convenient]



After installation, we also need to install the python environment

To run a python file, click on the left extension (ctrl+shift+X) - search for python, download and install



After installation, you can click on the upper left file, select New File, select the file type as Python, write your own Python code, and then click Run in the upper menu bar

3 A brief introduction and use of RVIZ

rviz is a 3D visualization platform in ROS, which can realize the graphical display of external information on the one hand, and release control information to objects through rviz, so as to realize the monitoring and control of the robot.

3.1 Introduction to the installation and interface of RVIZ

When installing ROS, if the full installation is performed, RVIZ is already installed, you can directly try to run it; If it is not fully installed, you can install rviz separately:

- **#Ubuntu16.04**
- `sudo apt-get install ros-kinetic-rviz`
- **#Ubuntu18.04**
- `sudo apt-get install ros-melodic-rviz`
- **#Ubuntu20.04**
- `sudo apt-get install ros-noetic-rviz`

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

```
roscore
```

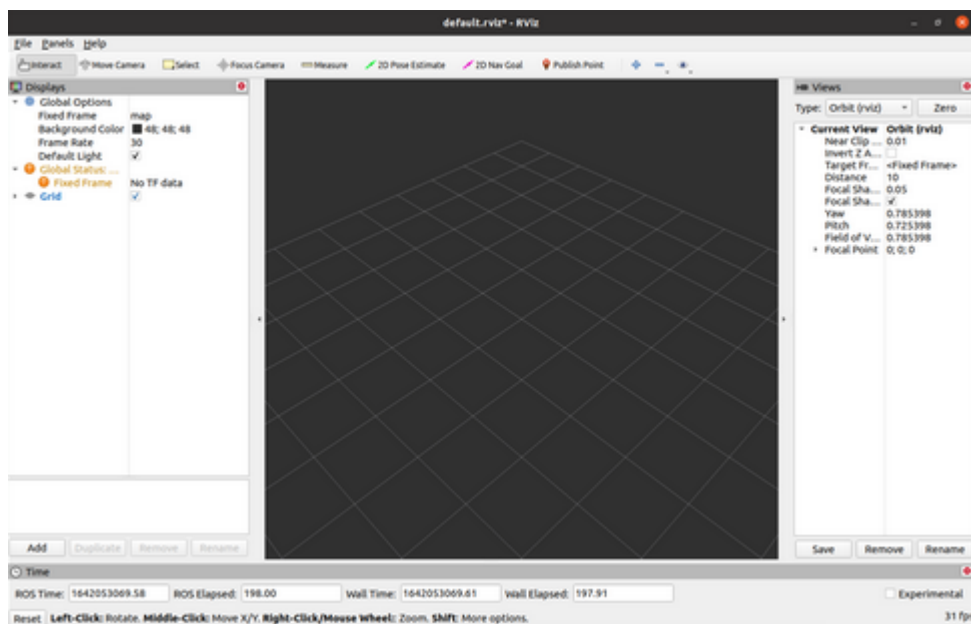
Then open a new terminal (shortcut Ctrl+Alt+T) and enter the command to open rviz :

```
roslaunch rviz rviz
```

or

```
rviz
```

Open RVIZ and display the following interface:



3.1.1 Introduction to each area

- On the left is a list of monitors, which are things that draw something in the 3D world and may have some options available in the display list.
- At the top is the toolbar, which allows the user to select a tool with a variety of functions with various function buttons
- The middle part is the 3D view: it is the main screen where you can view various data in 3D. The background color, fixed frame, mesh, etc. of the 3D view can be set in detail in the Global Options and Grid items displayed on the left.
- The lower part is the time display area, including the system time and ROS time.
- On the right is the observation angle setting area, which can be set to different observation angles.

We only give a rough introduction in this part, if you want to know more details, you can go to [User Guide](#) to review.

[← last section](#) | [next page →](#)

1、Emulation

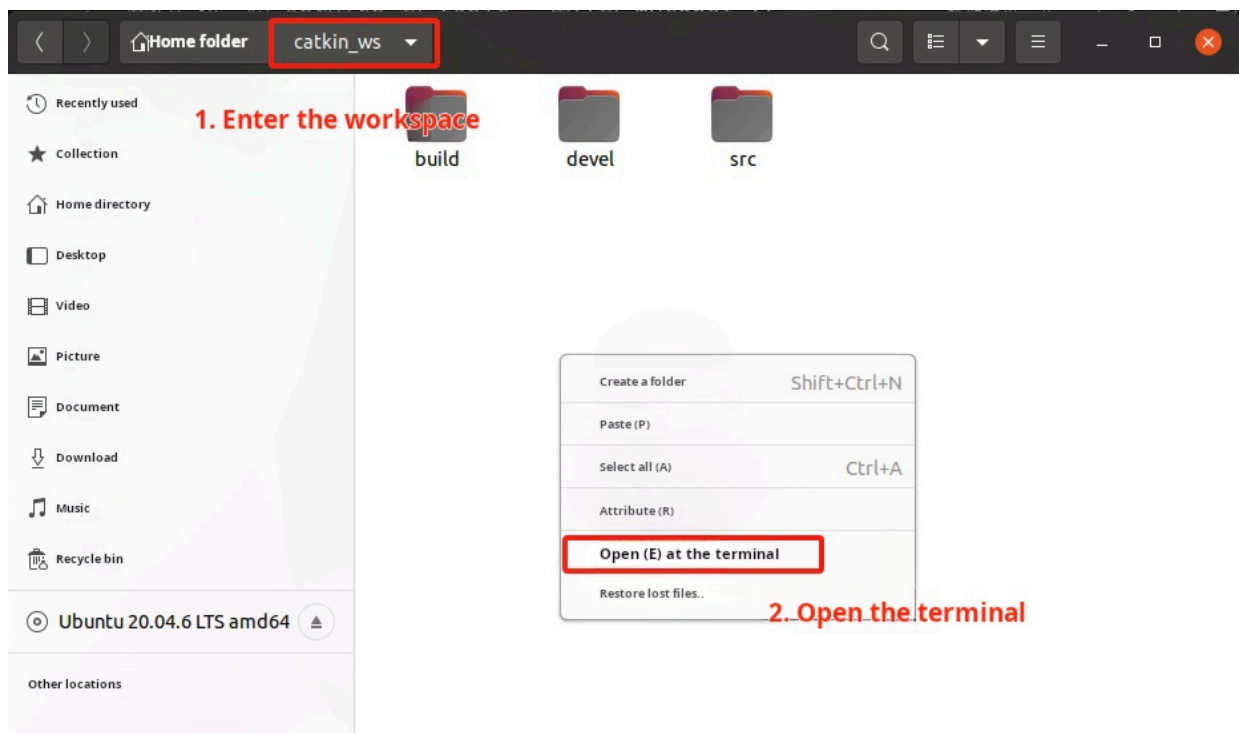
We provide a number of **static and dynamic simulations** to interact with the MyarmM750.

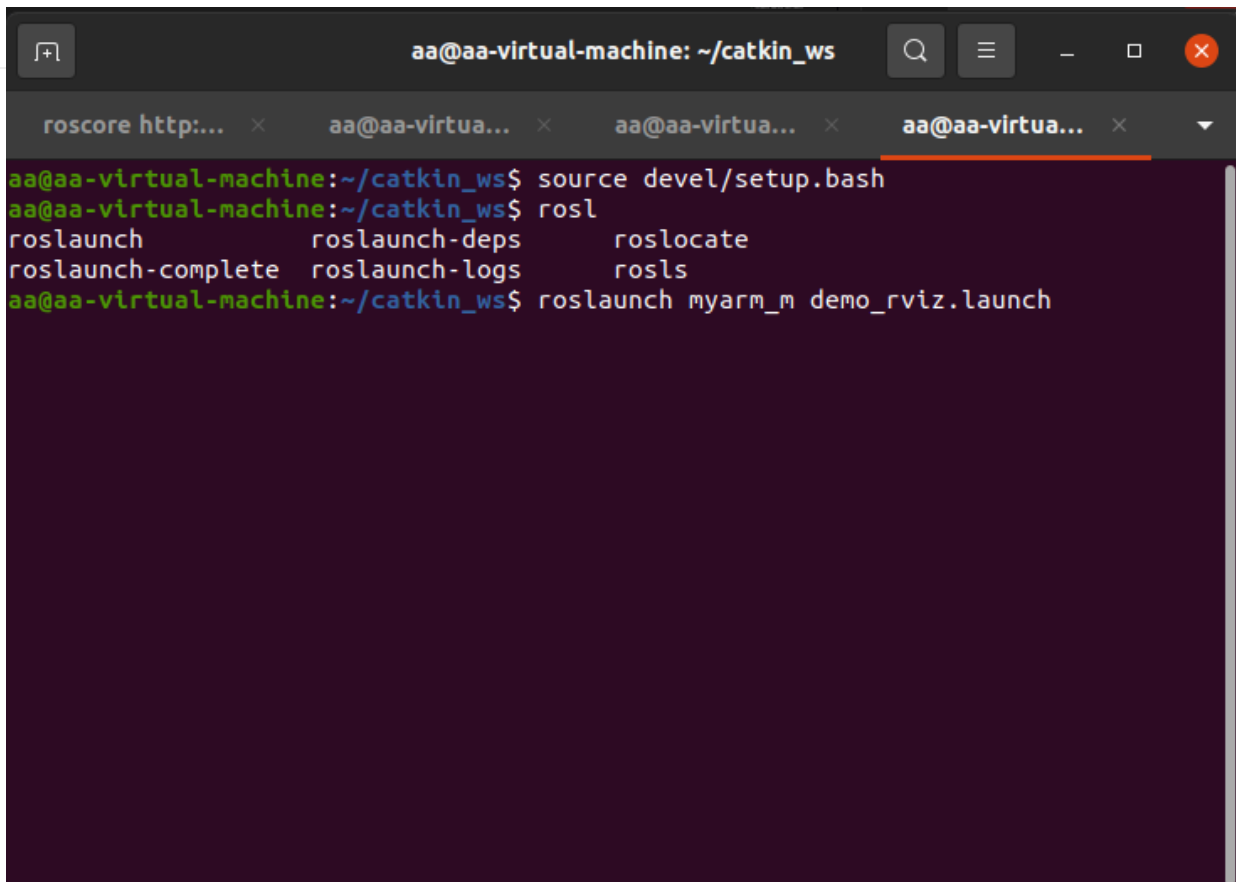
1.1 Static simulation

Static simulation here refers to the use of the sliding module in rviz to interact with the MyarmC650 in the simulation

In the Workspace Open Terminal command line, enter:

```
roscore  
  
source devel/setup.bash  
  
roslaunch myarm_m demo_rviz.launch
```

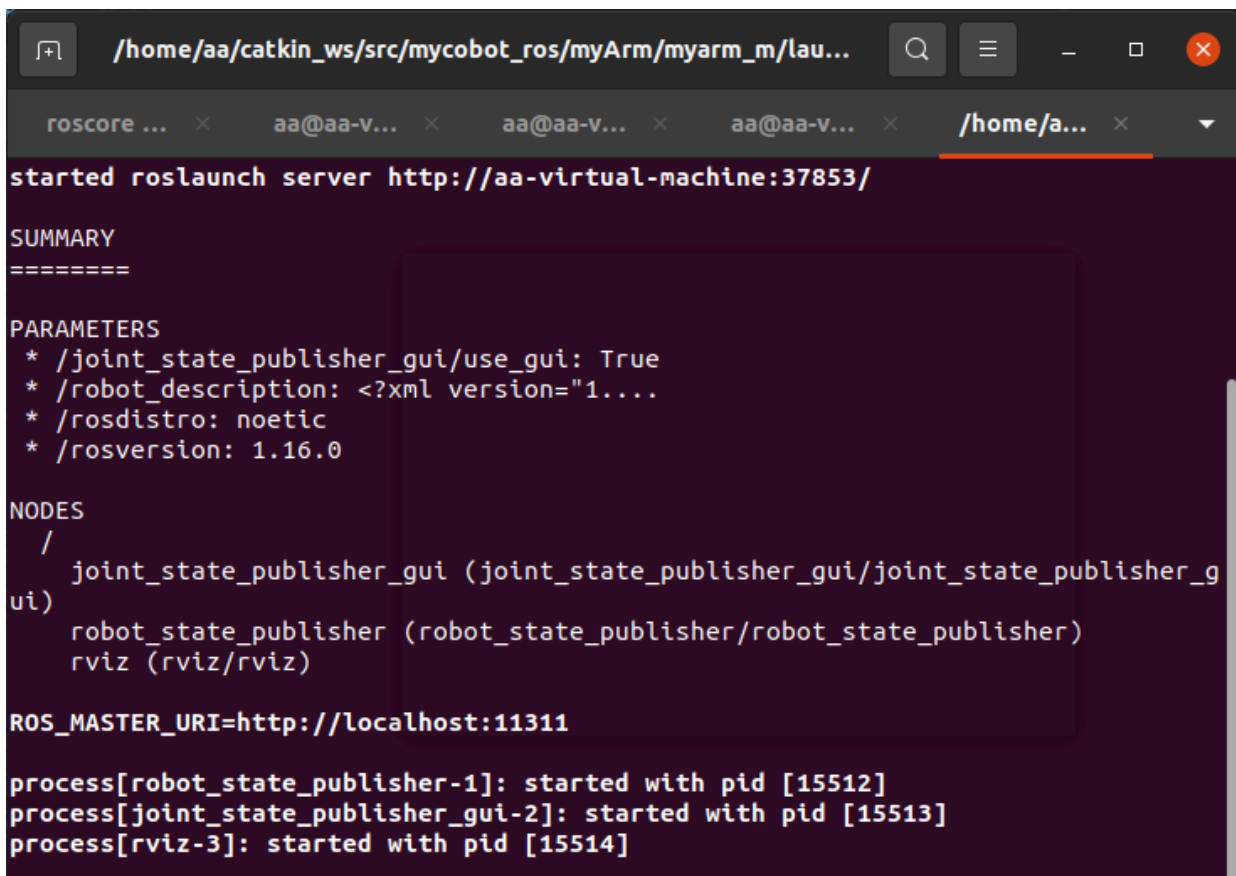




A terminal window titled 'aa@aa-virtual-machine: ~/catkin_ws'. The window has four tabs: 'roscore http:...', 'aa@aa-virtua...', 'aa@aa-virtua...', and 'aa@aa-virtua...'. The terminal shows the following commands and output:

```
aa@aa-virtual-machine:~/catkin_ws$ source devel/setup.bash
aa@aa-virtual-machine:~/catkin_ws$ rosl
roslaunch          roslaunch-deps      roslocate
roslaunch-complete roslaunch-logs      rosls
aa@aa-virtual-machine:~/catkin_ws$ roslaunch myarm_m demo_rviz.launch
```

After successfully running the launch file, the terminal will display:



A terminal window titled '/home/aa/catkin_ws/src/mycobot_ros/myArm/myarm_m/lau...'. The window has five tabs: 'roscore ...', 'aa@aa-v...', 'aa@aa-v...', 'aa@aa-v...', and '/home/a...'. The terminal displays the following output:

```
started roslaunch server http://aa-virtual-machine:37853/

SUMMARY
=====

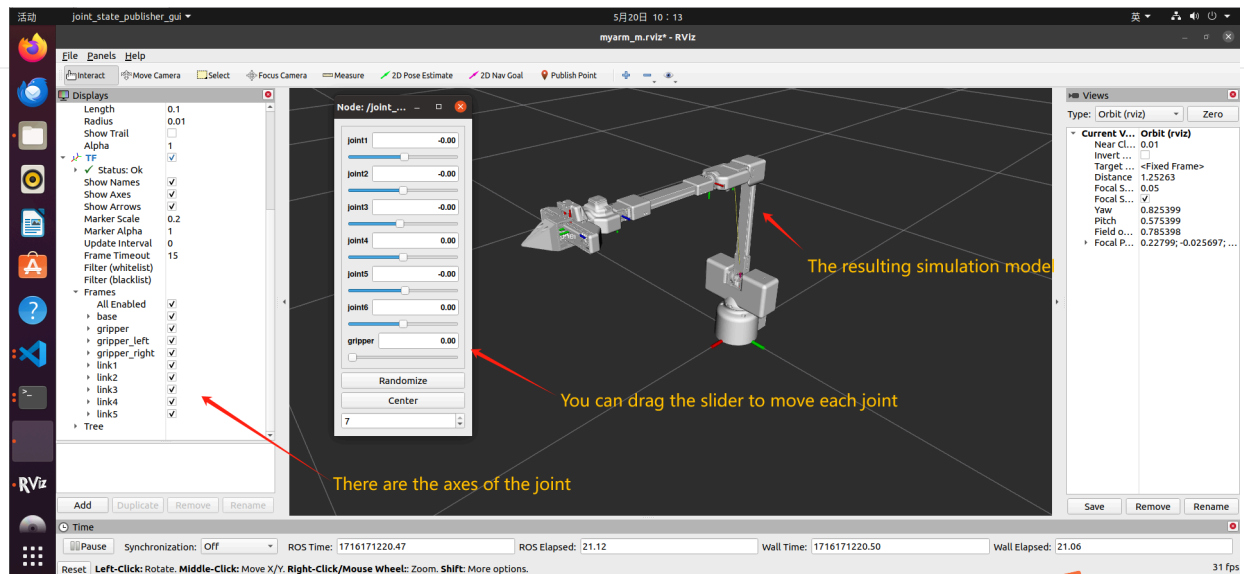
PARAMETERS
* /joint_state_publisher_gui/use_gui: True
* /robot_description: <?xml version="1...
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES
/
  joint_state_publisher_gui (joint_state_publisher_gui/joint_state_publisher_g
ui)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)
  rviz (rviz/rviz)

ROS_MASTER_URI=http://localhost:11311

process[robot_state_publisher-1]: started with pid [15512]
process[joint_state_publisher_gui-2]: started with pid [15513]
process[rviz-3]: started with pid [15514]
```

At the same time, rviz will be opened to generate a simulation model of MyarmM750



1.2 Dynamic simulation

Dynamic simulation here refers to the interaction of the MyarmM750 in motion reality with the MyarmM750 in the simulation

First of all, we need to connect the MyarmM750 robotic arm to our computer via a USB to TypeC cable and power it on

Select the **Transponder** button and press the **OK** button



Then the screen will be displayed:



We can see that the arrow points to **"USB UART"**, then press the **"OK"** button, after entering **"NO"**, then press the **"Exit"** button, go back to the arrow pointing to **"USB UART"**, and then press the **"OK"** button, then **"OK"** will be displayed **"OK"**

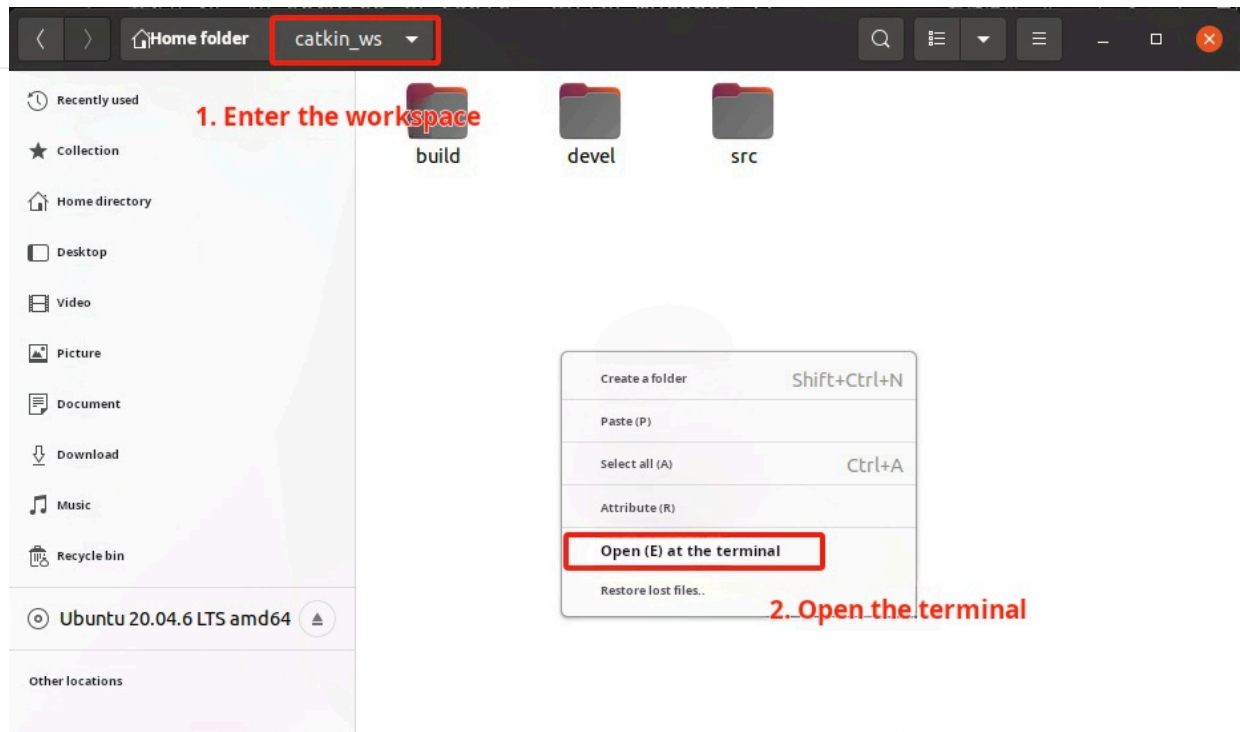


At this time, our MyarmM750 has been successfully connected to the computer

Next, enter it in the Workspace Open Terminal command line:

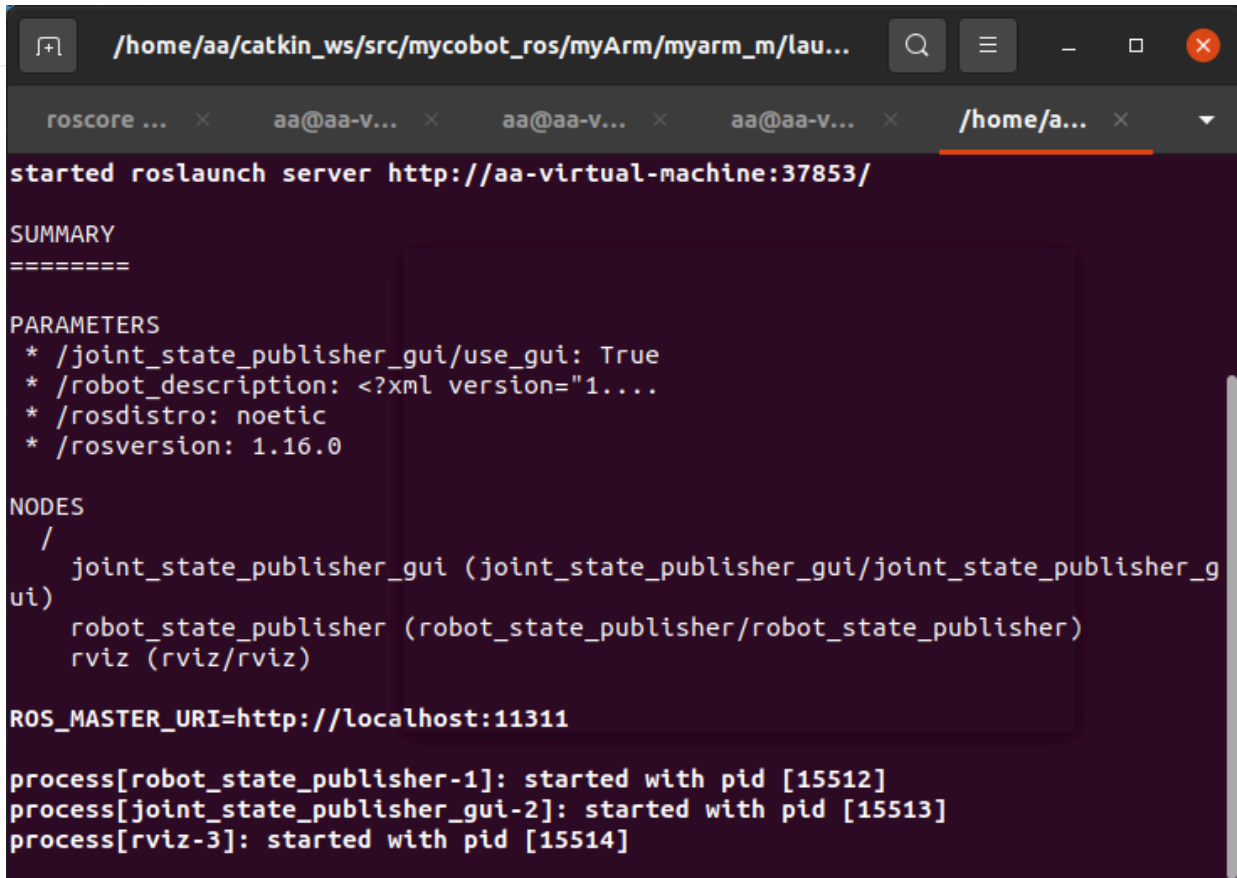
```
source devel/setup.bash
```

```
roslaunch myarm_m demo_rviz.launch
```



```
aa@aa-virtual-machine: ~/catkin_ws
roscore http:... x aa@aa-virtua... x aa@aa-virtua... x aa@aa-virtua... x
aa@aa-virtual-machine:~/catkin_ws$ source devel/setup.bash
aa@aa-virtual-machine:~/catkin_ws$ rosl
roslaunch          roslaunch-deps      roslocate
roslaunch-complete roslaunch-logs      rosls
aa@aa-virtual-machine:~/catkin_ws$ roslaunch myarm_m demo_rviz.launch
```

After successfully running the launch file, the terminal will display:



```

/home/aa/catkin_ws/src/mycobot_ros/myArm/myarm_m/lau...
roscore ... x aa@aa-v... x aa@aa-v... x aa@aa-v... x /home/a... x

started roslaunch server http://aa-virtual-machine:37853/

SUMMARY
=====

PARAMETERS
* /joint_state_publisher_gui/use_gui: True
* /robot_description: <?xml version="1...
* /rostdistro: noetic
* /rosversion: 1.16.0

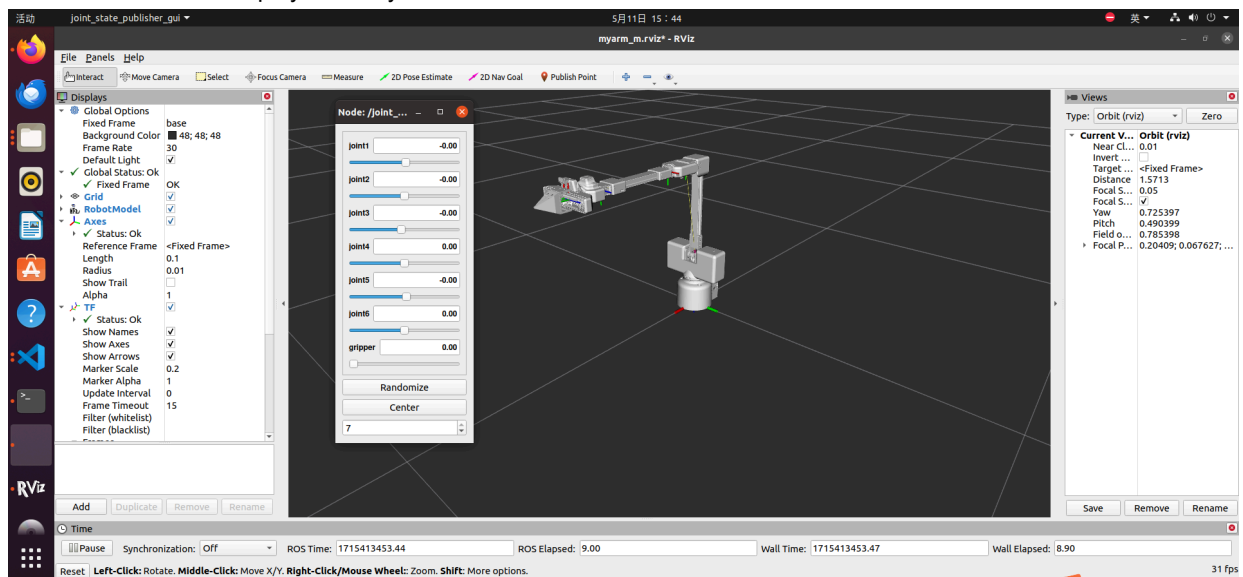
NODES
/
  joint_state_publisher_gui (joint_state_publisher_gui/joint_state_publisher_g
ui)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)
  rviz (rviz/rviz)

ROS_MASTER_URI=http://localhost:11311

process[robot_state_publisher-1]: started with pid [15512]
process[joint_state_publisher_gui-2]: started with pid [15513]
process[rviz-3]: started with pid [15514]

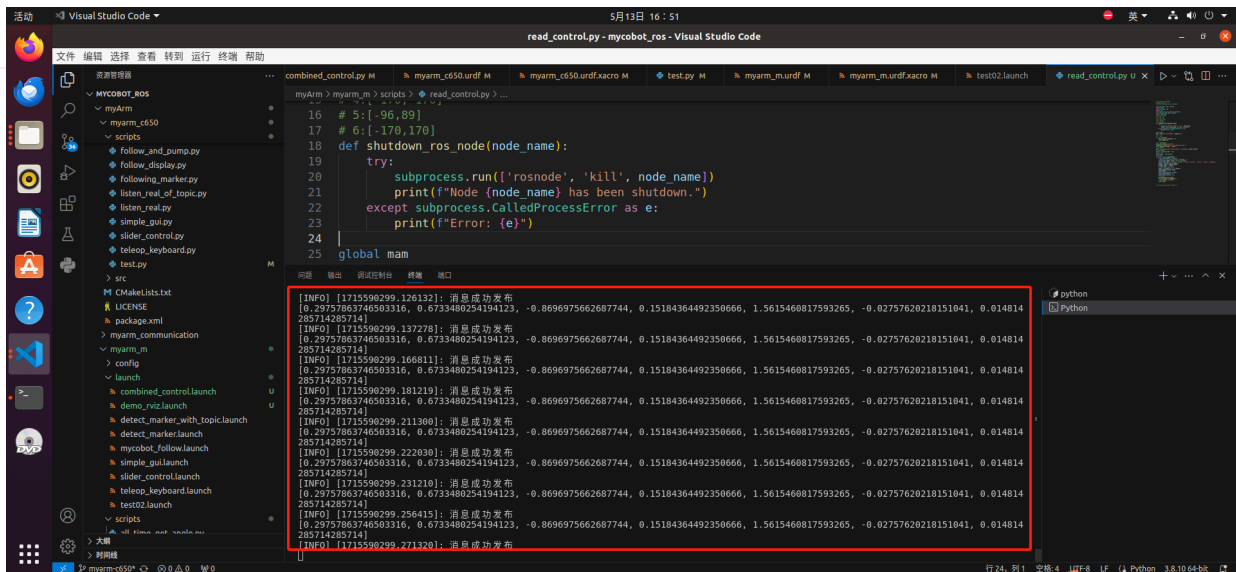
```

The rviz file will also display normally



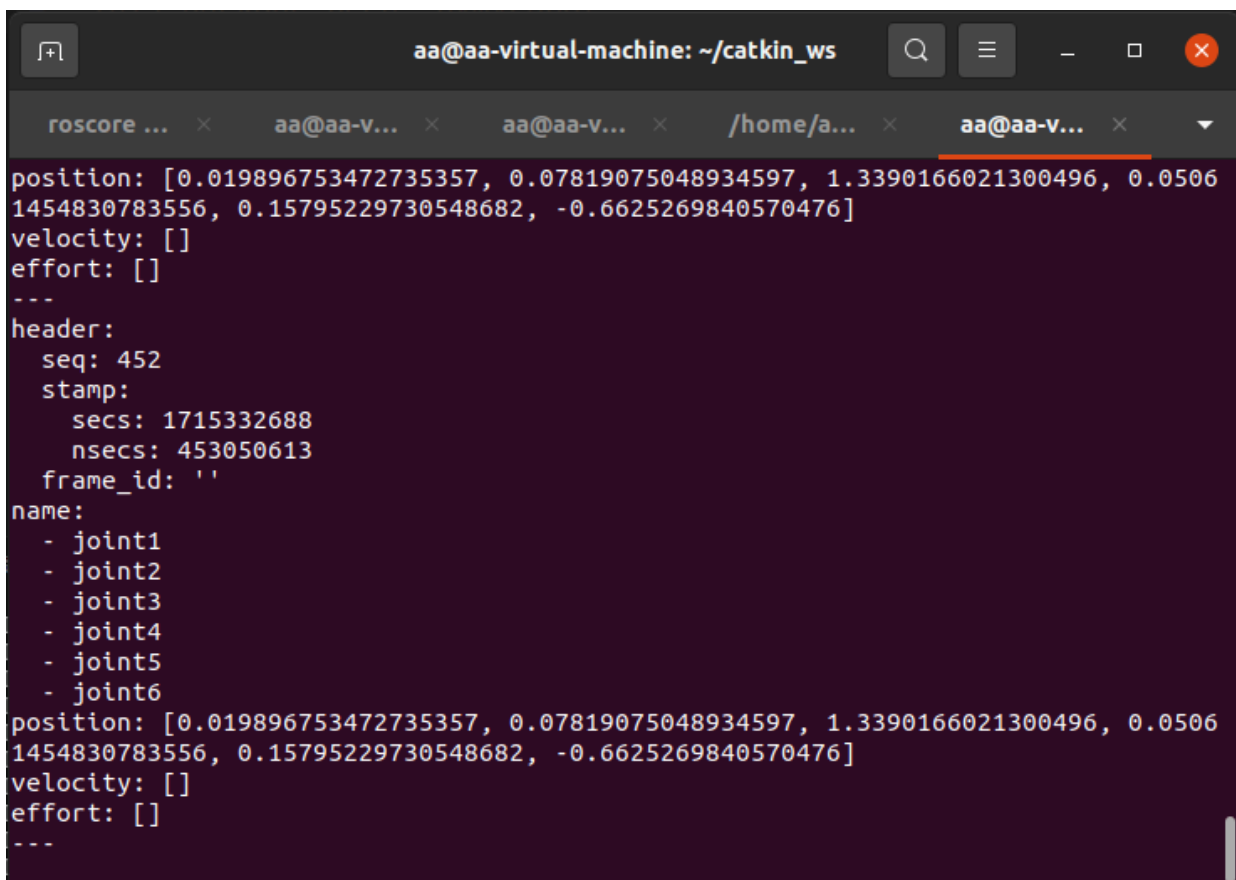
Open the project in VS code and find the **read_control.py** file myArm/myarm_m/scripts/read_control.py

- Open the .py file, right-click, select "Run Python", and then click "Run Python file in the terminal"

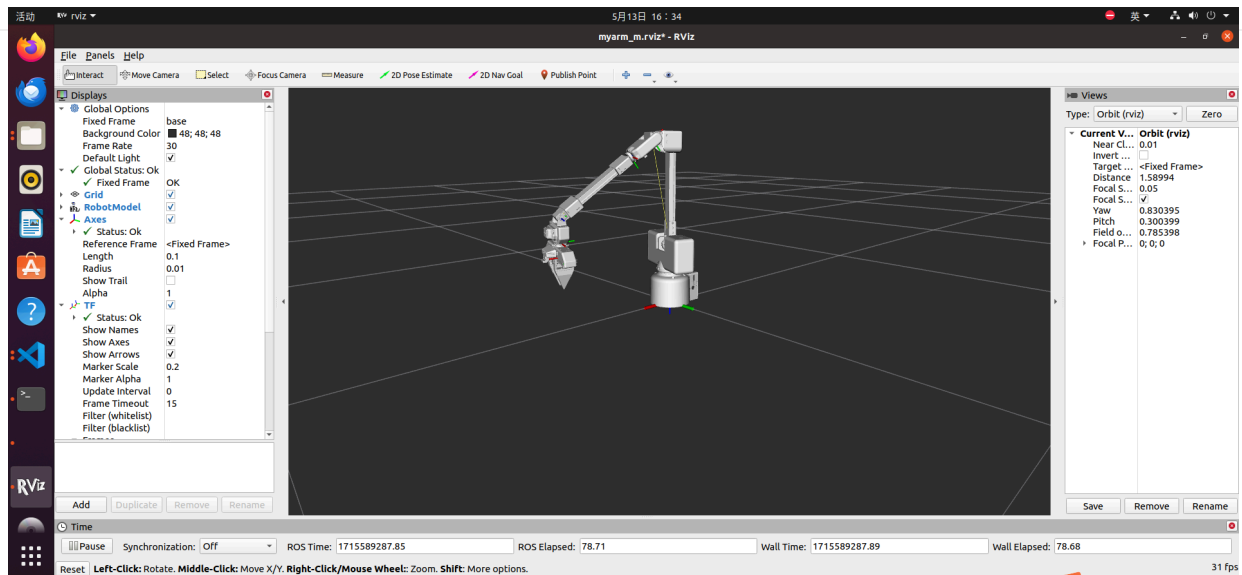


We can also continue to enter the following commands in the terminal in order to see the angle change of each joint of the MyarmM750:

```
rostopic echo /joint_state
```



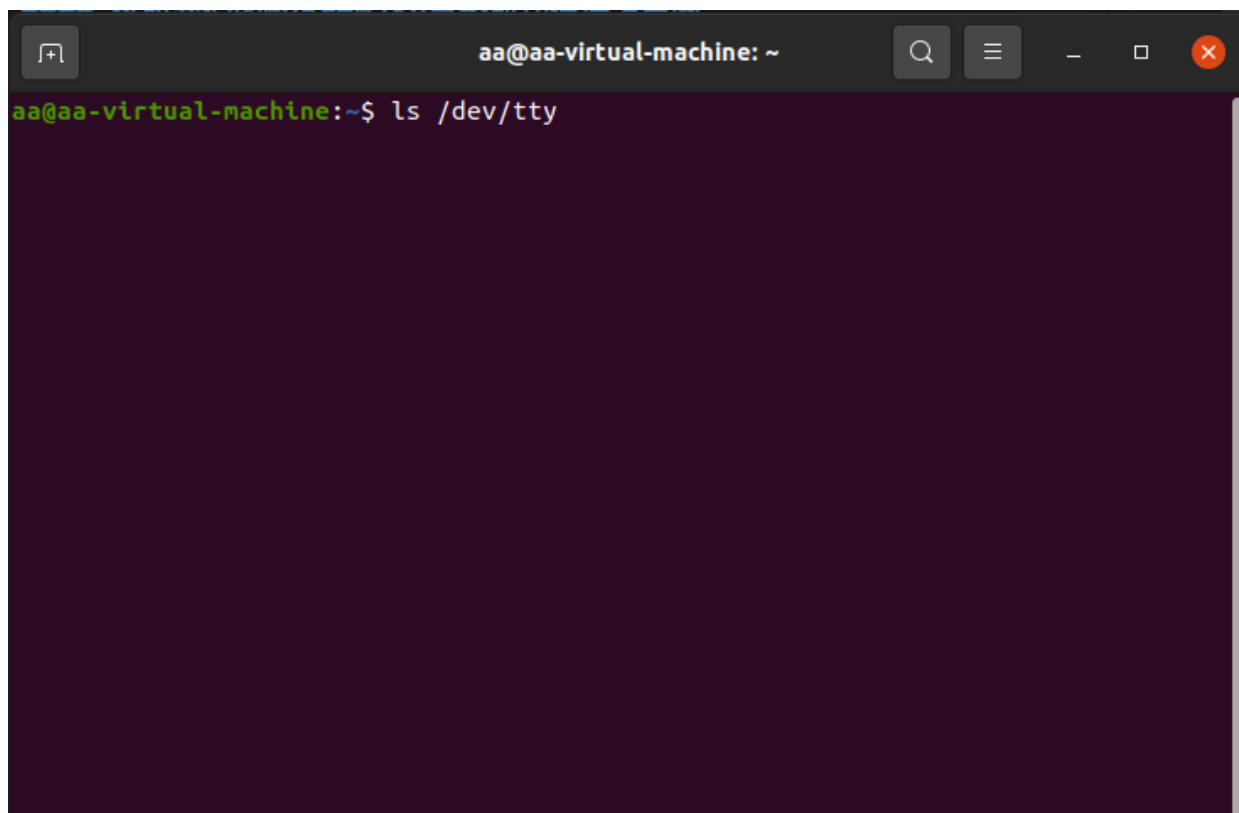
Next, we move the real-life MyarmM750 robotic arm with our hands, and the robotic arm in rviz will also follow:



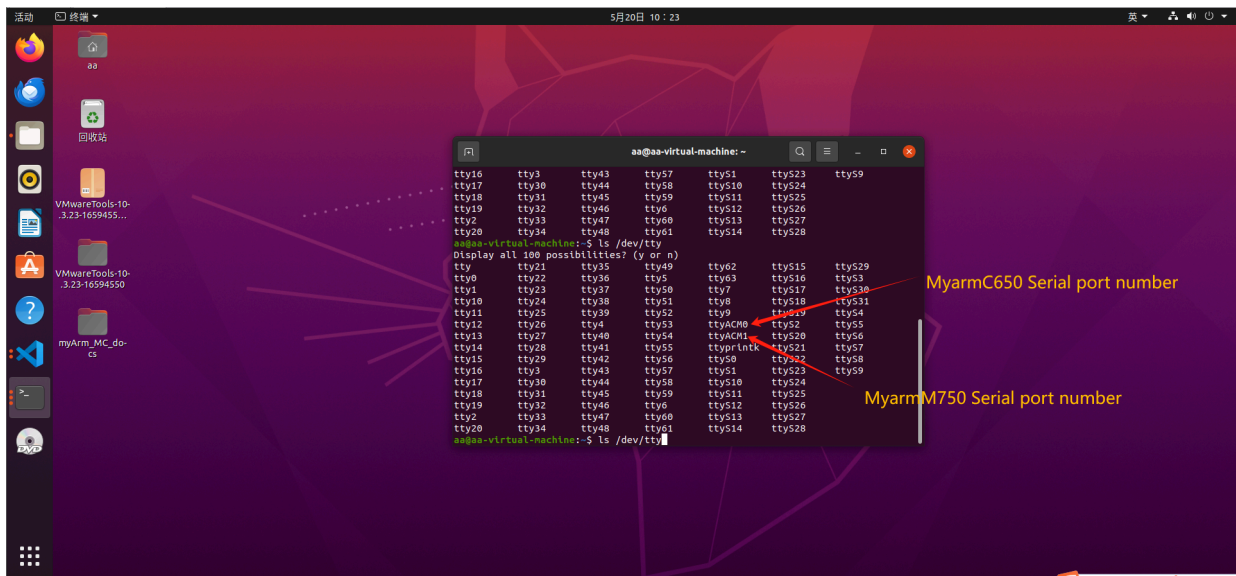
At this point, the interaction with the MyarmM750 robotic arm has been completed

2. MyarmC650 controls the movement of MyarmM750

This function requires two robotic arms to be connected to our computer at the same time through USB, we need to distinguish how much each robotic arm is connected to our serial port, and create a new terminal:

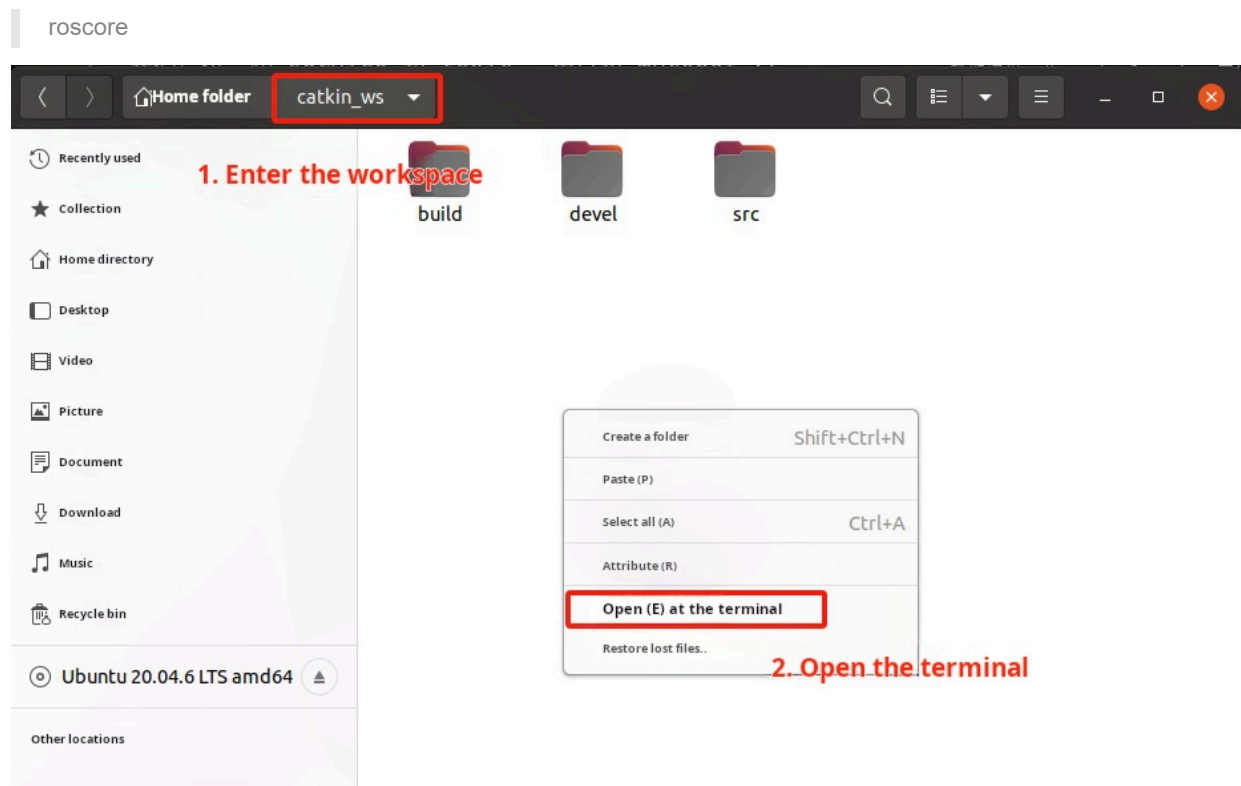


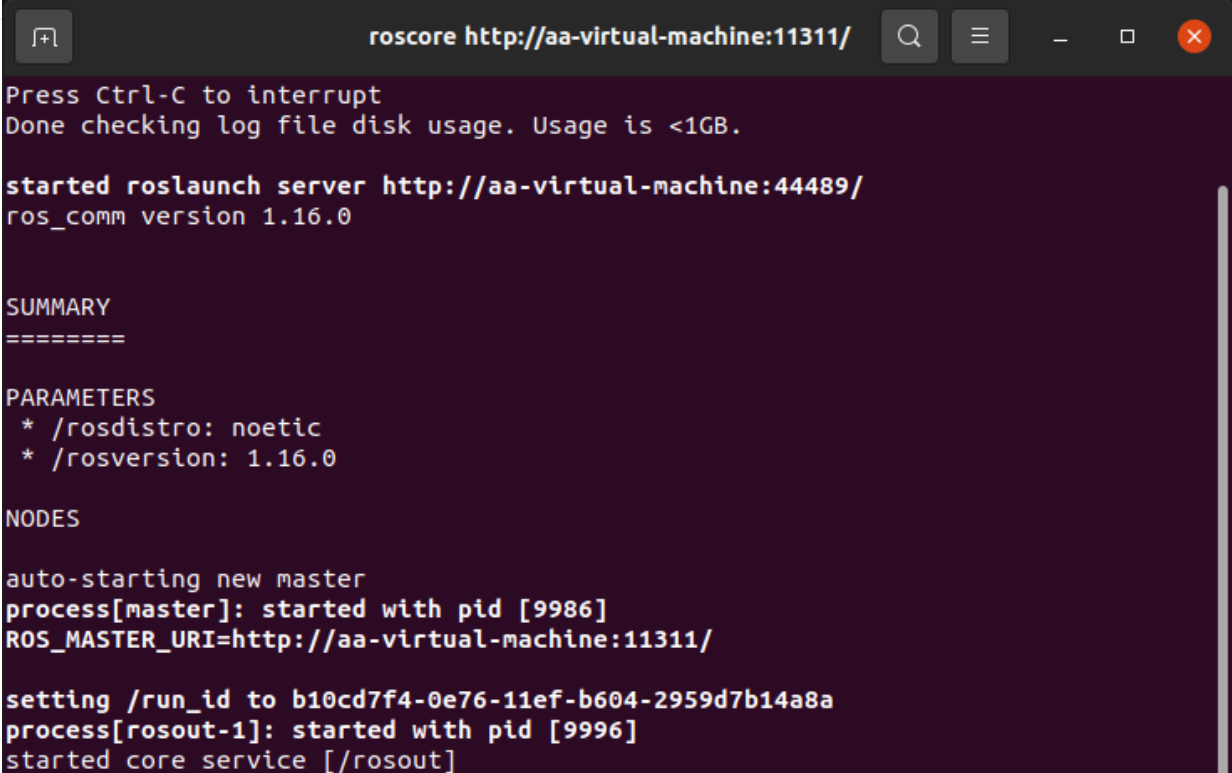
Press **TAB** key



How to distinguish the string slogans: connect a robotic arm and then enter "ls /dev/tty" to view the current string slogan, and when the robotic arm is not opened, connect another robotic arm and enter the command to view the string slogan

Create a terminal input in the workspace:



A terminal window with a dark background and light text. The title bar reads 'roscore http://aa-virtual-machine:11311/'. The terminal output shows the process of starting a ROS master and a node. It includes messages about disk usage, the start of the roslaunch server, a summary of parameters, and the start of the master and roscout processes.

```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://aa-virtual-machine:44489/
ros_comm version 1.16.0

SUMMARY
=====

PARAMETERS
* /rosdistro: noetic
* /rosversion: 1.16.0

NODES

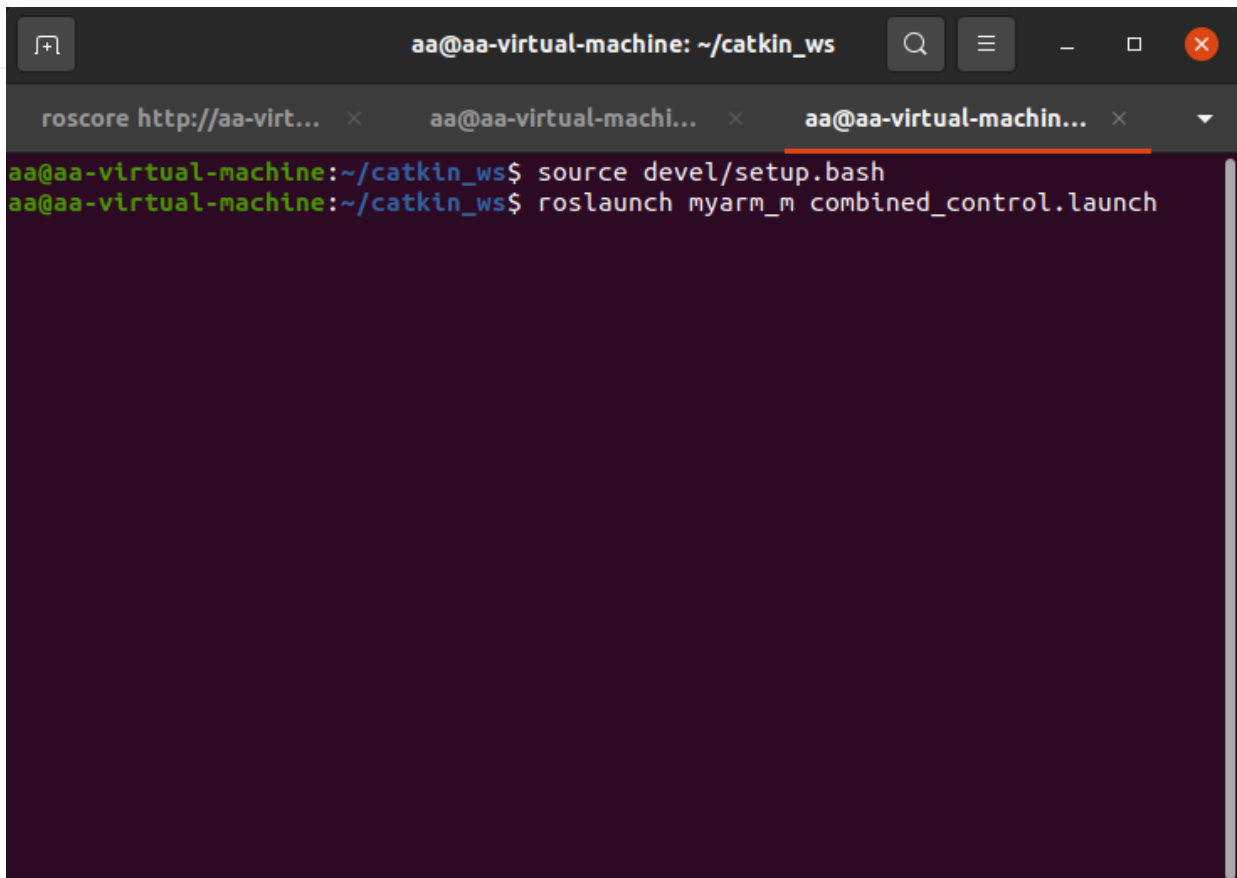
auto-starting new master
process[master]: started with pid [9986]
ROS_MASTER_URI=http://aa-virtual-machine:11311/

setting /run_id to b10cd7f4-0e76-11ef-b604-2959d7b14a8a
process[roscout-1]: started with pid [9996]
started core service [/roscout]
```

Create another terminal and enter:

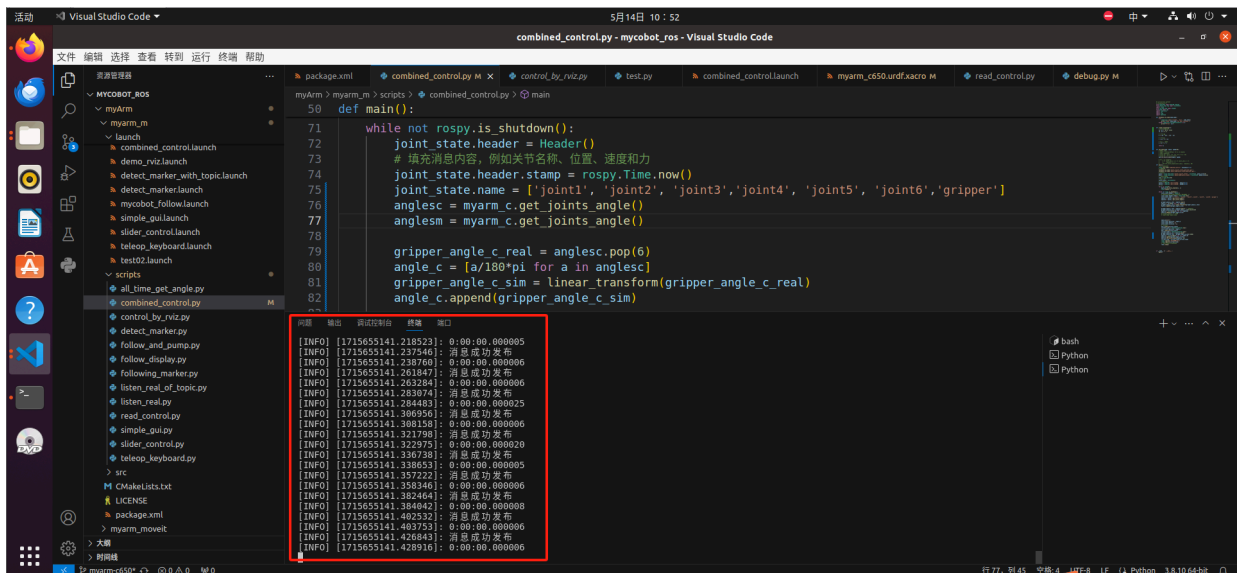
```
source devel/setup.bash
```

```
roslaunch myarm_m combined_control.launch
```

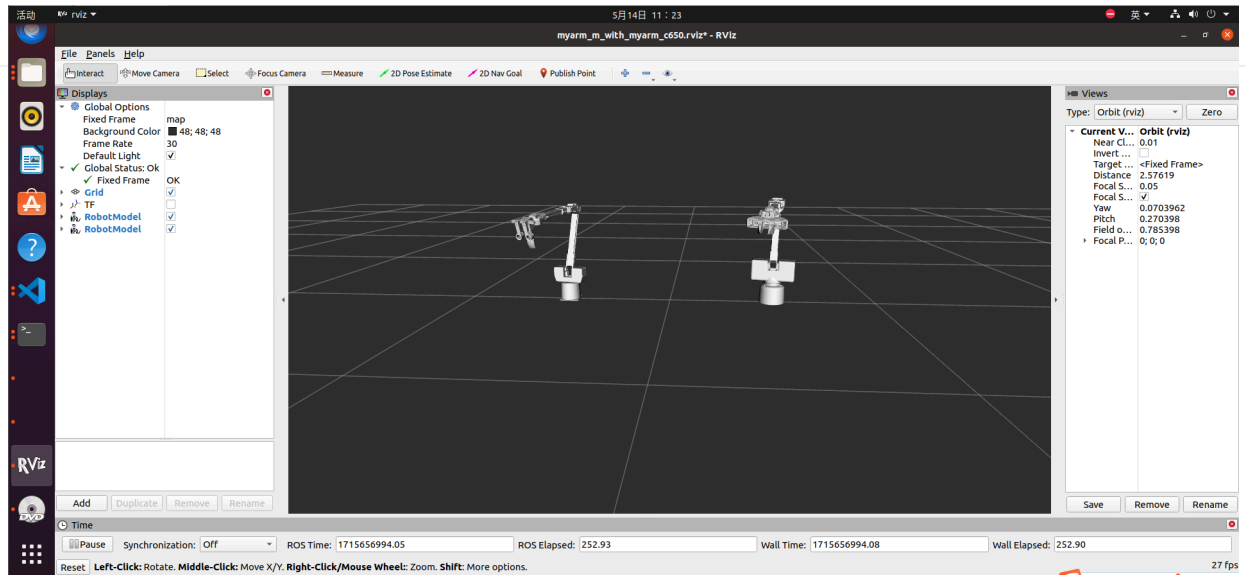


After starting the rviz emulation, open the .py file in the **myArm/myarm_m/scripts/combined_control.py** directory

- Open the .py file, right-click, select "Run Python", and then click "Run Python file in the terminal"



At this time, we can use our hands to move the MyarmC650 robotic arm, the two robotic arms in the rviz follow the movement of the real-life MyarmC650, and at the same time, the MayrmM750 robotic arm in the real world will also follow the movement of the MyarmC650 (the gripping function can also move)



At this point, the MyarmC650 control MyarmM750 function is fully realized

[← last page](#) | [next page →](#)

1、 Introduction

ROS also provides some practical command-line tools, which can be used to obtain various information of different nodes, and the common commands are as follows:

roscall : Operational nodes

rostopic : Operational topics

rosservice : Operational Services

rosmmsg : Manipulating MSG messages

rossrv : Manipulate SRV messages

rosparm : Operational parameters

[Official reference URL](#)

2、 Rosnod

roscall is a command used to obtain node information

```
1 | roscall ping      Test the connection status to the node
2 | roscall list      List active nodes
3 | roscall info      Print node information
4 | roscall machine   Lists nodes on a specified device
5 | roscall kill      Kill a node
6 | roscall cleanup   Clear unconnectable nodes
```

3、rostopic

rostopic contains the rostopic command-line tool to display debugging information about ROS topics, including publishers, subscribers, publishing frequency, and ROS messages. It also contains an experimental Python library for dynamically fetching information about topics and interacting with them.

```

1 | rostopic bw      Displays the bandwidth used by the topic
2 | rostopic delay  Show topic latency with header
3 | rostopic echo   Print messages to the screen
4 | rostopic find   Find topics by type
5 | rostopic hz     Display topic publishing frequency
6 | rostopic info   Display topic-related information
7 | rostopic list   Displays topics in all active states
8 | rostopic pub    Publish data to topics
9 | rostopic type   Print theme type

```

3.1rostopic list

You can call it directly, and the console will print the name of the topic in the current running state

rostopic list -v : Get thread details (e.g. list: number of publishers and subscribers...)

3.2rostopic pub

Commands can be invoked directly to publish messages to subscribers

Publish a string for the Subscriber in the Publish/Subscribe model case that Roboware auto-generated

	1.topic name	2. message type	3. message content
1	rostopic pub /主题名称	消息类型	消息内容
2	rostopic pub /liaotian	std_msgs	huluwa

Take the baby turtle as an example to post a sports message:

```

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 publish the campaign information only once
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 the motion information is sent in a
    cycle at a frequency of 10Hz

```

3.3rostopic echo

Gets the current message posted on a specified topic

3.4rostopic info

Get information on current topics

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

4、 rosservice

rosservice contains the rosservice command-line tool for listing and querying ROSServices.

When you call some services, if you do not configure a path for the related workspace, you need to go to the workspace to call the service **source devel/setup.bash**

```
1 | rosservice args print service parameters
2 | rosservice call      Use the provided parameters to call the service
3 | rosservice find      Find services by service type
4 | rosservice info      Print information about the service
5 | rosservice list      List all active services
6 | rosservice type      Print Service type
7 | rosservice uri       The ROSRPC uri of the print service
```

4.1 rosservice args

Print service parameters

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

4.2 rosservice call

Invoke 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 Generate a turtle named XXX
8 |
```

5、 rosmmsg

rosmmsg is a command-line tool used to display information about the type of ROS message.

```
1 | rosmmsg show      Display message description
2 | rosmmsg info      Display message information
3 | rosmmsg list      List all messages
4 | rosmmsg md5       Displays md5 encrypted messages
5 | rosmmsg package   Displays all messages in a Feature Package
6 | rosmmsg packages  List feature packages that contain messages
```

5.1 rosmmsg package

Lists all MSGs under a package

```
1 | rosmmsg package 包名 package name
```

5.2 rosmmsg show

Displays a description of the message

```
1 | //rosmmsg show 消息名称 Message name
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 for displaying information about the type of ROS service, and is highly similar to the rosmmsg syntax.

```
1 | rossrv show      Show service message details
2 | rossrv info      Displays information about service messages
3 | rossrv list      List all service information
4 | rossrv md5       Displays md5 encrypted service messages
5 | rossrv package   Displays all service messages under a package
6 | rossrv packages  Displays all packages that contain service messages
```

6.1 rossrv package

```
1 | rossrv package 包名 package name
```

6.2 rossrv show

Displays a description of the message


```

1 //rossrv show 消息名称 message name
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 includes the rosparam command-line tool for obtaining and setting ROS parameters on a parameter server using a YAML encoding file.

```

1 rosparam set      Set parameters
2 rosparam get      GET parameters
3 rosparam load      Load parameters from an external file
4 rosparam dump      Write parameters to an external file
5 rosparam delete    Delete parameters
6 rosparam list      List all parameters

```

7.1rosparam set

Set the parameters

```
rosparam set name xxx
```

7.2rosparam get

Get the parameters

```
rosparam get name
```

7.3rosparam delete

Delete the parameter

```
rosparam delete name
```

7.4rosparam load(Prepare the yaml file first)

Load parameters from an external file

```
rosparam load xxx.yaml
```

7.5rosparam dump

Write out the parameters to an external file

8、 Introduction to the URDF

- Unified Robot Description Format, The Unified Robot Description Format, or URDF for short. The urdf feature pack in ROS contains a C++ parser for URDF, and URDF files describe the robot model in XML format.
- URDF cannot be used on its own and needs to be combined with Rviz or Gazebo, URDF is just a file that needs to be rendered as a graphical robot model in Rviz or Gazebo.

8.1 urdf file description

Code examples:

Only some of the codes are screenshotted for display:

```
<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>
</link>
```

```

<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 you can see, the urdf file is not complicated, and is mainly composed of two parts, **link** and **joint**, which are constantly repeated.

8.2 link section

The link element describes a rigid body with inertia, visual features, and collision properties

8.2.1 attribute

name: The name used to describe the link itself

8.2.2 element

- **visual**
 - Visualization properties of the linkage. Used to specify the shape of the linkage display (rectangle, cylinder, etc.), the same link can have multiple visual elements, and the shape of the link is formed by two elements. In general, if the model is more complex, you can generate STL calls after drawing through solidwork, and simple shapes such as adding end effectors can be written directly. At the same time, the position of the geometry can be adjusted here according to the gap between the theoretical and practical models.
 - **name1** (Optional) The name of the geometry of the connecting rod.
 - **origin** (Optional, defaults to identity if not specified)
 - Geometry coordinate system relative to the coordinate system of the linkage.
 - xyz (optional: defaults to zero vector) Indicates the offset in the direction of gram, yi, zik, yi, z, yi, z, and z, in meters.

- rpy (optional: defaults to identity if not specified) Represents the rotation of the coordinate axis in the RPY direction, in radians.

- **geometry** (Required)

- The shape of the visualization, which can be one of the following:
- **box** A rectangle with elements that contain length, width, and height. The origin is in the center.
- **cylinder** A cylinder, an element containing radius, length. Origin Center.
- **sphere** A sphere, an element containing a radius. The origin is in the center.
- **mesh** The grid, which is determined by the file, also provides a scale to delimit its boundaries. Collada .dae files are recommended, and .stl files are also supported, but they must be a local file.

- **collision** (Optional)

- Collision properties of the linkage. Collision properties are different from the visual properties of the linkage, and simple collision models are often used to simplify calculations. The same link can have multiple collision attribute labels, and the collision attribute representation of the linkage is made up of the set of geometry it defines.
- **name** (Optional) Specifies the name of the linkage geometry
- **origin** (Optional, defaults to identity if not specified)
 - The reference coordinate system of the collision component is relative to the reference coordinate system of the linkage coordinate system.
 - xyz (Optional, Default zero vector) Indicates the offset in the direction of gram, yi, zik, yi, z, yi, z, and z, in meters.
 - rpy (Optional, defaults to identity if not specified) Represents the rotation of the coordinate axis in the RPY direction, in radians.
- **geometry** Same as described above for the geometry element

Details of the elements and what each element does can be found in the [official documentation](#) to review

8.3 Joyn part

The joint section describes the kinematics and dynamics of the joint and specifies safety limits for the joint.

8.3.1 Properties of joint:

name:

Specifies a unique name for the joint

type:

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

- revolute - A hinged joint head that rotates along an axis, the extent of which is specified by the upper and lower limits.
- 连续 - A continuous hinged link head that rotates around an axis with no upper and lower limits.
- Prismatic - A slip joint that slides along an axis, the range of which is specified by the upper and lower limits.
- fixed - It's not really a joint because it can't move. All degrees of freedom are locked. This type of joint does not require shafts, calibration, dynamics, limits, or safety_controller.
- float - This joint allows movement of all 6 degrees of freedom.

- **plane** - This joint allows movement in a plane perpendicular to the axis.

8.3.2 joint elements

- **origin** (Optional, defaults to identity if not specified) For the transformation from parent link to child link, the joint is located at the origin of the child link, and the position of the connecting rod can be adjusted by modifying this parameter, which can be used to adjust the error between the actual model and the theoretical model, but it is not recommended to modify it significantly, because this parameter affects the position of the connecting rod STL and is easy to affect the collision detection effect.
 - **xyz** (Optional: The default is a zero vector) Represents the offset in the direction of the axis of gram, yi, zik, yi, z, z, z, z
 - **rpy** (Optional: The default is a zero vector) Represents the angle of rotation around a fixed axis: Roll around the x-axis, Pitch around the Y-axis, and Yaw around the Z-axis, expressed in radians.
- **parent** (Required)
 - The name of the parent link is a mandatory property.
 - The name of the link parent link is the name of the link in the robot structure tree.
- **child** (Required)
 - The name of the child link is a mandatory attribute.
 - The name of the link child link is the name of the link in the robot structure tree.
- **axis**(Optional: Default is (1,0,0))
 - The axis of the joint is in the coordinate system of the joint. This is the axis of rotation (revolute joint), the axis on which the prismatic joint moves, and is the standard plane of the Planar joint. This axis is specified in the joint coordinate system. Modify this parameter to adjust the axis around which the rotation of the joint is rotated, which is often used to adjust the direction of rotation, if the rotation direction of the model is opposite to the actual one, you only need to multiply -1. Fixed and floating joints do not need this element.
 - **xyz**(required) The gram, yi, zik, yi, z, yi, yi, yi, zi, zi, zi, yi, zi, yi, z
- **calibration** (Optional)
 - The reference point of the joint, which is used to correct the absolute position of the joint.
 - **rising** (Optional) When the joint is moving forward, the reference point triggers a rising edge.
 - **falling** (Optional) When the joint is moving forward, the reference point triggers a descending edge.
- **dynamics**(Optional)
 - This element is used to specify the physical properties of the joint. Its value is used to describe the modeling performance of the joint, especially when simulating.
- **limit** (Required when the joint is rotated or moved)
 - This element is a joint kinematic constraint.
 - **lower** (Optional, The default value is 0) Specifies the property of the lower bound of the joint's range of motion (in radians for revolute joints and meters for prismatic joints), which is ignored for continuous joints.
 - **upper** (Optional, The default value is 0) Specifies the property of the upper bound of the joint's range of motion (in radians for revolute joints and meters for prismatic joints), which is ignored for continuous joints.
 - **effort** (Required) This property specifies the maximum force at which the joint runs.
 - **velocity** (required) This property specifies the maximum speed at which the joint runs.

Details of the elements and what each element does can be found in [Documentation](#) to review

correspondence

Our MyarmM750 mainly uses a **Topic** method for communication

Topic is one of the most commonly used communication mechanisms in ROS, which is based on the publisher-subscriber model, in which one node publishes messages as publishers and other nodes receive messages as subscribers. A publisher can publish messages to multiple subscribers at the same time, and subscribers can receive messages from multiple publishers. This approach is ideal for situations that require real-time data updates, such as processing and real-time control of sensor data.

Start by creating a Publisher in the .py file to publish our message to the MyarmM750

```

37
38 pub = rospy.Publisher('/joint_states', JointState, queue_size=10)
39 # 设置发布频率
40 rate = rospy.Rate(50) # 50Hz
41 # 消息实例
42 joint_state = JointState()
43 # 发布消息
44 while not rospy.is_shutdown():
45     joint_state.header = Header()
46     # 填充消息内容, 例如关节名称、位置、速度和力
47     joint_state.header.stamp = rospy.Time.now()
48     joint_state.name = ['joint1', 'joint2', 'joint3', 'joint4', 'joint5', 'joint6', 'gripper']
49     angles = mam.get_joints_angle()
50     gripper_angle = angles.pop(6)
51     gripper_angle /= -3500
52     # angles: 角度 angles: 角度
53     angle = [a/180*pi for a in angles]
54     angle.append(gripper_angle)
55     print(angle)
56     joint_state.position = angle
57     joint_state.effort = []
58     joint_state.velocity = []
59     # 发布消息
60     pub.publish(joint_state)
61     rospy.loginfo('消息成功发布')
62     # 等待, 允许其他节点处理
63     rate.sleep()
64
65

```

Next, open the terminal in the workspace and start ROS:

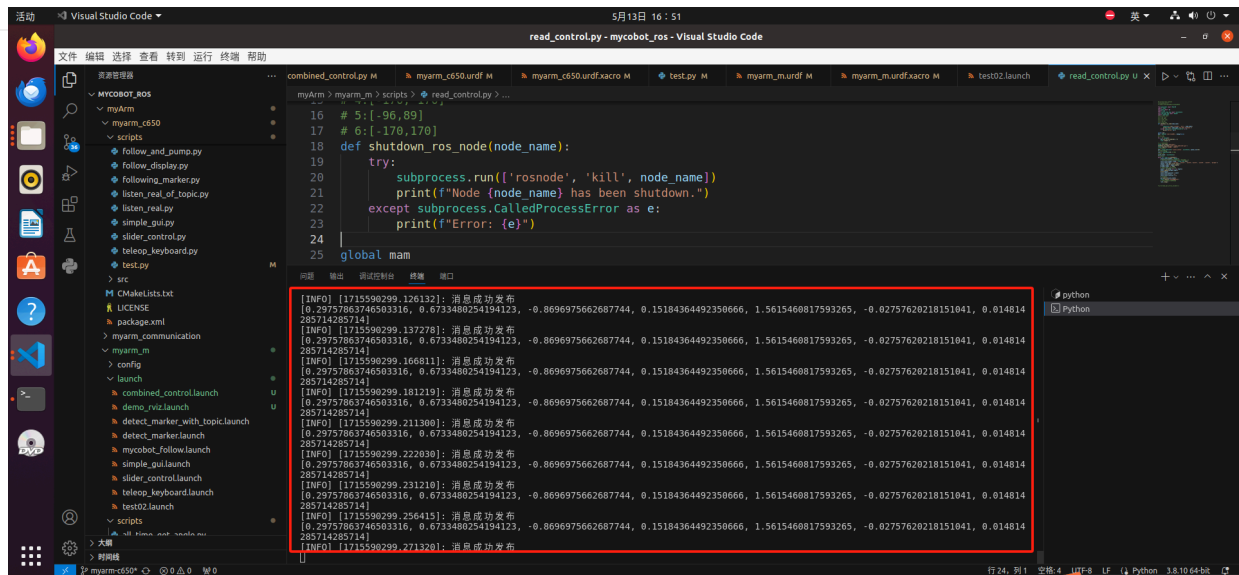
```
roscore
```

Create another terminal and enter:

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

```
roslaunch myarm_m read_control.launch
```

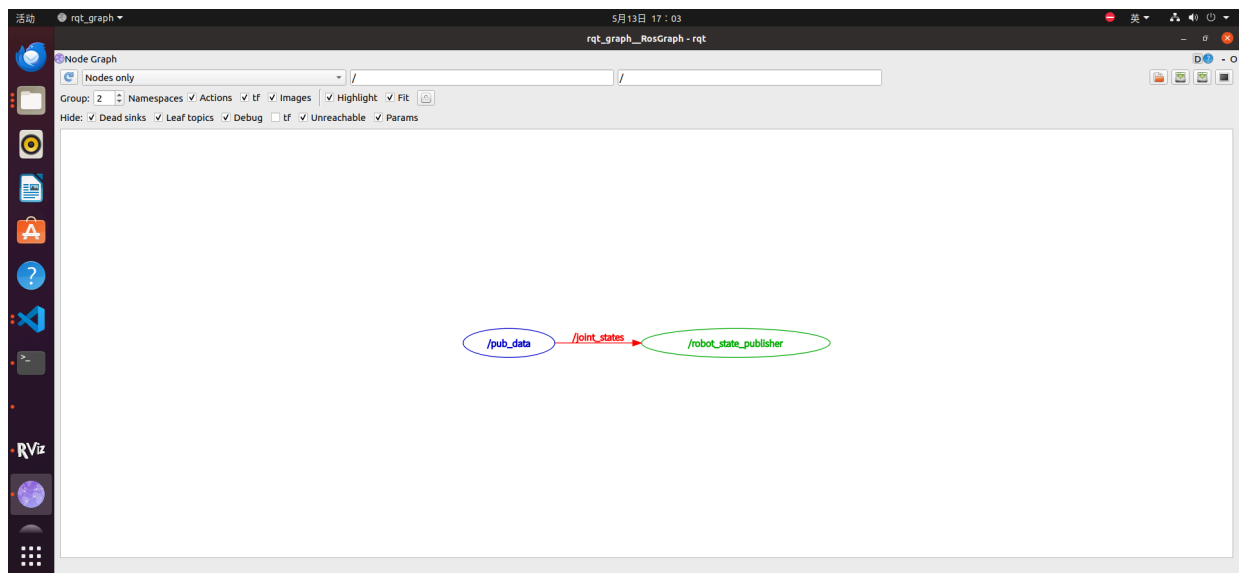
Open RVIZ and then launch our **read_control.py** file



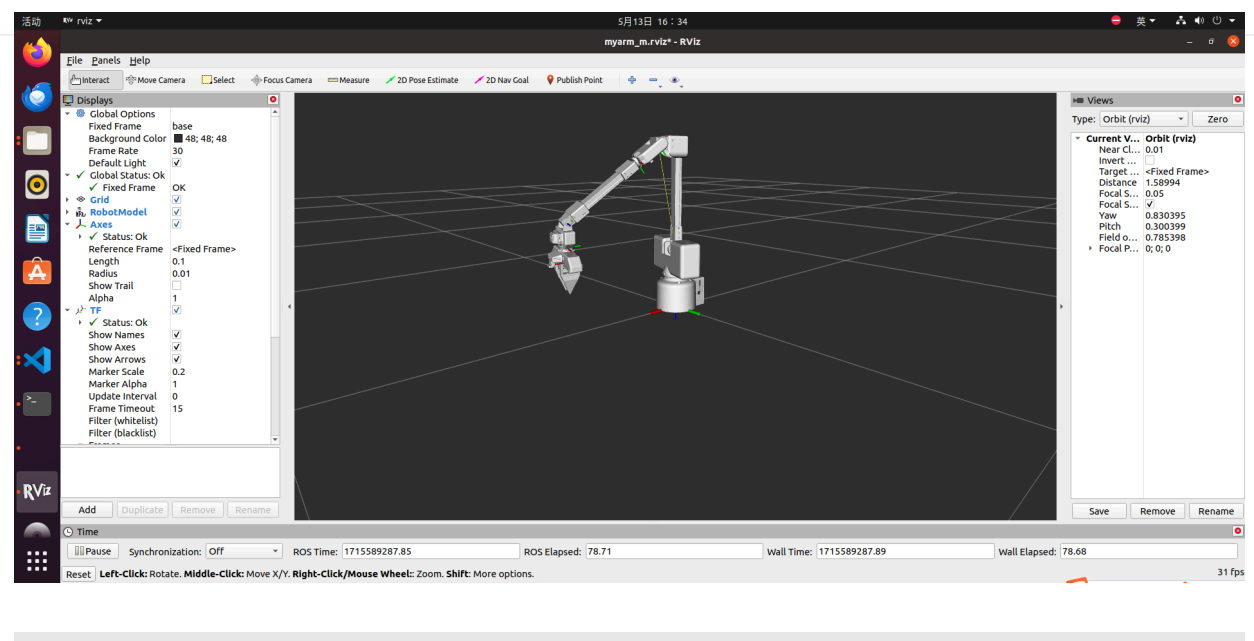
Finally, we open a new terminal and enter:

```
rqt_graph
```

We can see all the information about the node



The MyarmM750 is also in a state where it can be manually moved



[← last page](#) | [next section →](#)

ROS2 introduction

The predecessor of ROS2 is ROS, and ROS is the Robot Operating System (Robot Operating System). But ROS itself is not an operating system, but a software library and toolset. The emergence of Ros solved the communication problem of each component of the robot. Later, more and more robot algorithms were integrated into ROS. ROS2 inherited ROS, which is more powerful and better than ROS.

1 Design goals and features of ROS2

ROS2 has the historical mission of changing the era of intelligent robots. At the beginning of the design, it was considered to meet the needs of various robot applications.

- **Multi-Robot Systems:** In the future, robots will not be independent individuals, and communication and collaboration between robots are also required. ROS2 provides standard methods and communication mechanisms for the application of multi-robot systems.
- **Cross-platform:** Robot application scenarios are different, and the control platforms used will also be very different. In order to allow all robots to run ROS2, ROS2 can run on Linux, Windows, MacOS, and RTOS across platforms.
- **Real time:** Robot motion control and many behavior strategies require the robot to be real-time. For example, the robot must reliably detect pedestrians in front of it within 100ms, or complete kinematics and dynamics calculations within 1ms. ROS2 is a real-time like this Basic requirements are provided.
- **Productization:** A large number of robots have entered our lives, and there will be more and more in the future, ROS2 can not only be used in the robot research and development stage, but also can be directly installed in the product and go to the consumer market. This also poses a huge challenge to the stability and robustness of ROS2.
- **Project management:** Robot development is a complex system engineering. The project management tools and mechanisms for the whole process of design, development, debugging, testing, and deployment will also be reflected in ROS2, making it easier for us to develop a robot.

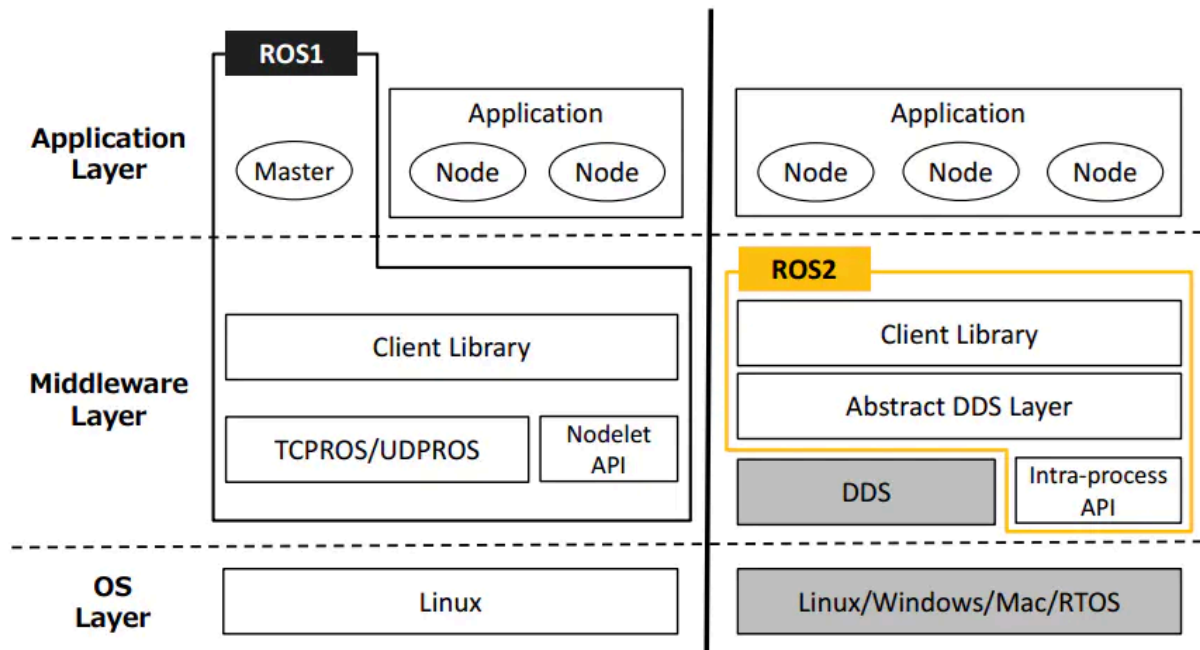
2 Release Version

The release version and maintenance cycle corresponding to ROS2 and Ubuntu.

ROS2 version	release date	Maintenance deadline	Ubuntu version
2019.5	2021.5	2023.5	Ubuntu 18.04 (Bionic Beaver)
2020.6	2022.5	2024.5	Ubuntu 20.04(Focal Fossa)
2022.5	2024.5	2026.5	Ubuntu 22.04(Jammy Jellyfish)

3 Comparison of ROS and ROS2

ROS2 redesigned the system architecture. The architecture changes between the two generations of ROS are as follows:



- **OS Layer:** In ROS2, it can be built on linux or other systems, even bare metal without an operating system.
- **Middleware Layer:** The communication system of ROS1 is based on TCPROS/UDPROS, while the communication system of ROS2 is based on DDS. DDS is a standard solution for data publishing/subscribing in distributed real-time systems.
- **Application Layer:** ROS1 relies on ROS Master, while in ROS2, a discovery mechanism called "Discovery" is used between nodes to help establish connections with each other.

ROS has designed a complete set of communication mechanisms (topics, services, parameters, actions) to simplify robot development. Through this mechanism, the various components of the robot can be connected. This mechanism has designed a node called Ros Master, and the communication of all other components must go through the master node. Once the master node hangs up, it will cause the communication of the entire robot system to collapse! Therefore, the instability of Ros cannot be used to make some high-risk robots such as automatic driving. In addition, there are the following disadvantages:

- Communication based on TCP has poor real-time performance and high system overhead
- Unfriendly to python3 support
- Messaging mechanism is not compatible
- No encryption mechanism, low security

ROS2 first removes the master node that exists in ROS. After removing the master node, each node can discover each other through the DDS node, each node is equal, and can realize one-to-one, one-to-many, and many-to-many communication. After using DDS for communication, reliability and stability have been enhanced.

Compared with **ROS** that only supports Linux systems, **ROS2** also supports windows, mac, and even RTOS platforms

Myarm M750 version:

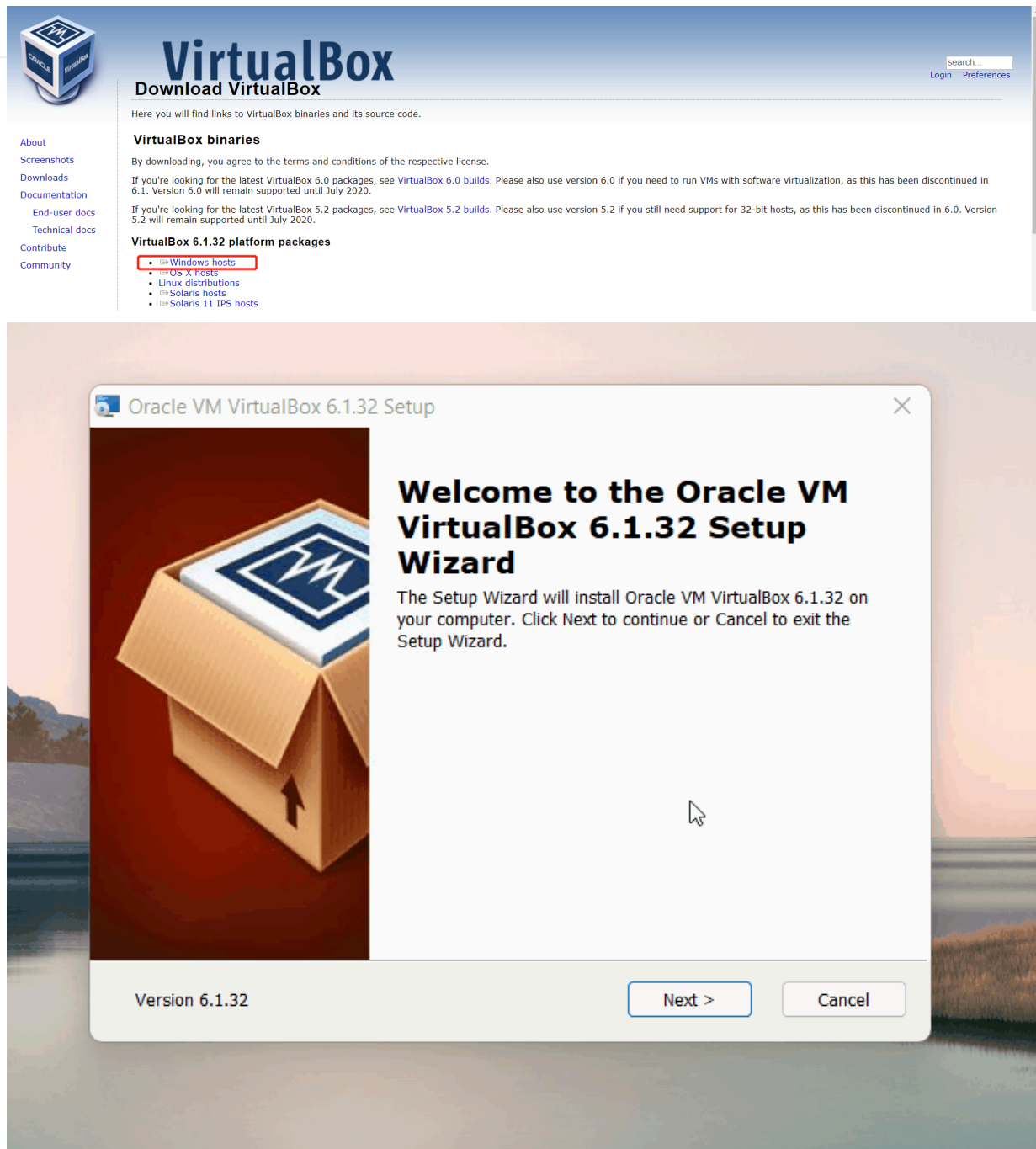
Install different versions of ubuntu in Linux

1 Virtual machine installation

Go to [Official website](#) to download the virtual machine Vltul albox or go to [the official website](#) download the virtual machine Wimwal

Of course, if you already have your virtual machine, you can skip that step.

We chose to download the Virtual box because it is free.

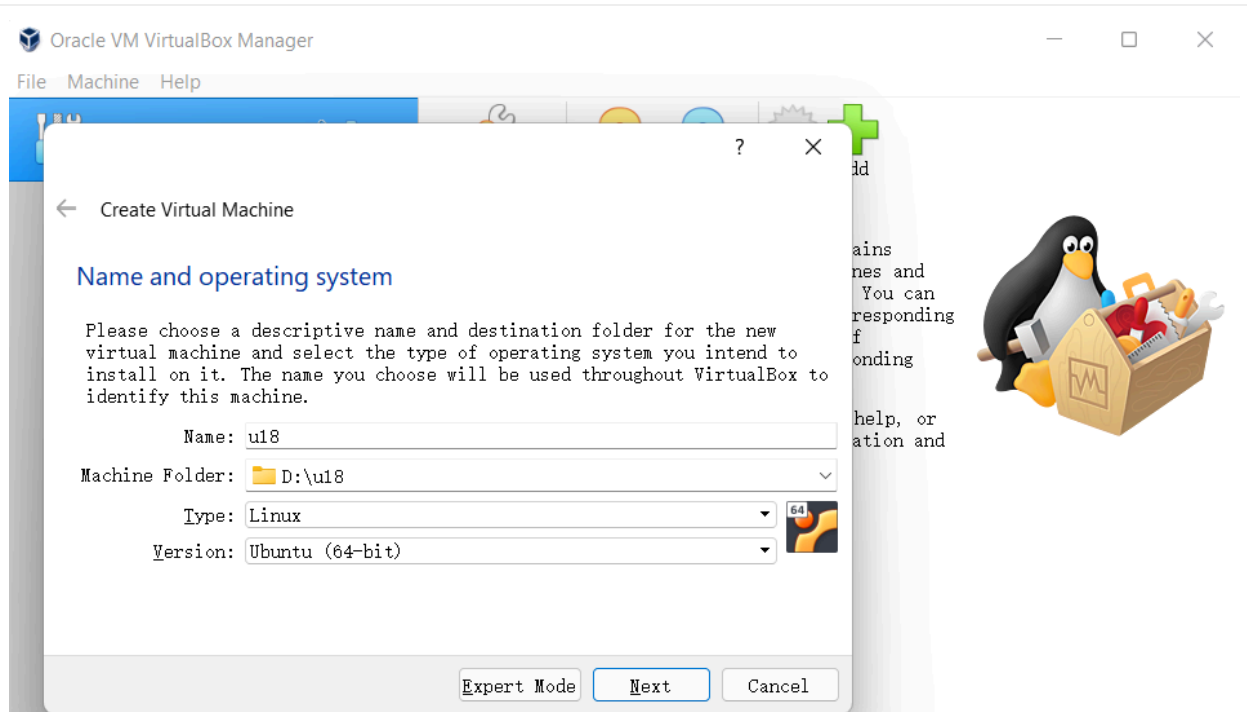


2 Create a new virtual machine

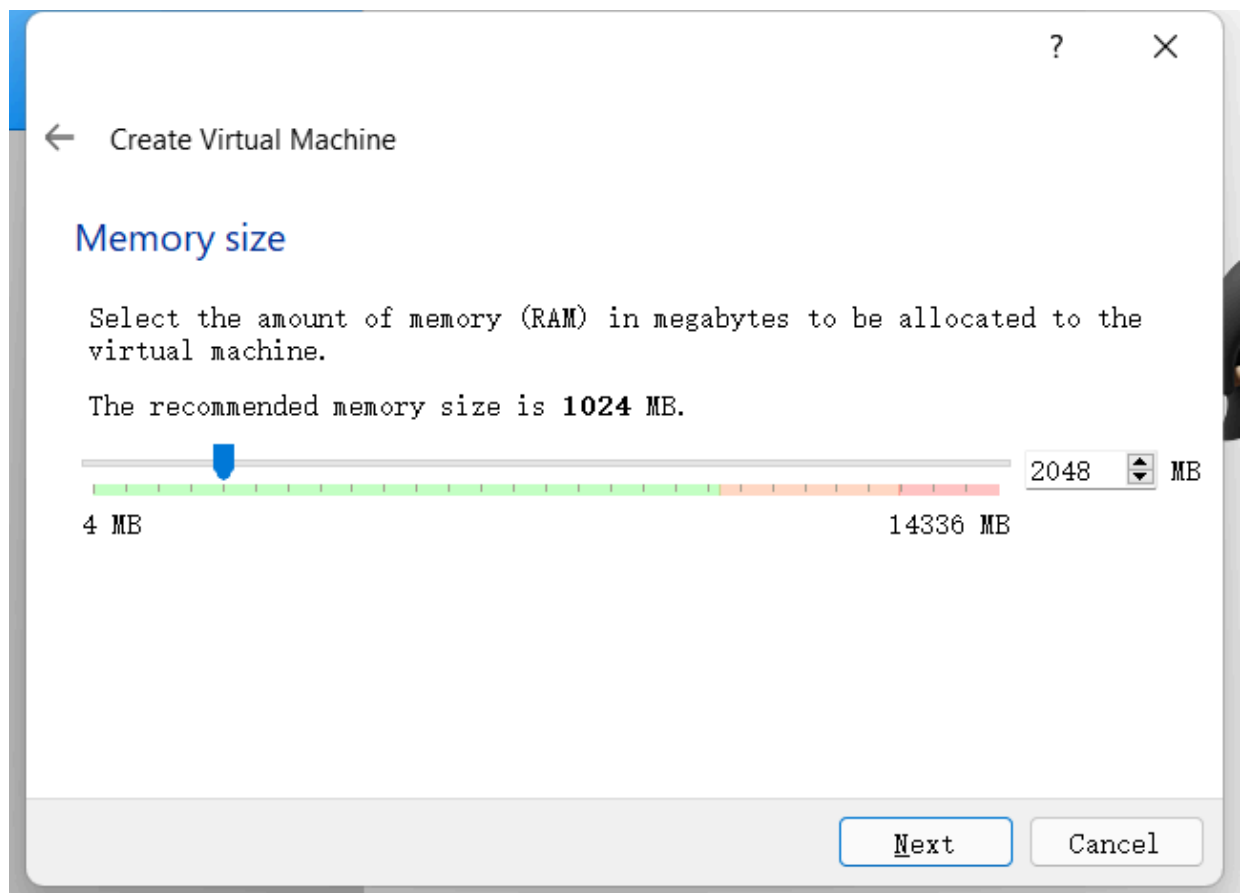
2.1 Create a virtual machine

In Control, select **New**

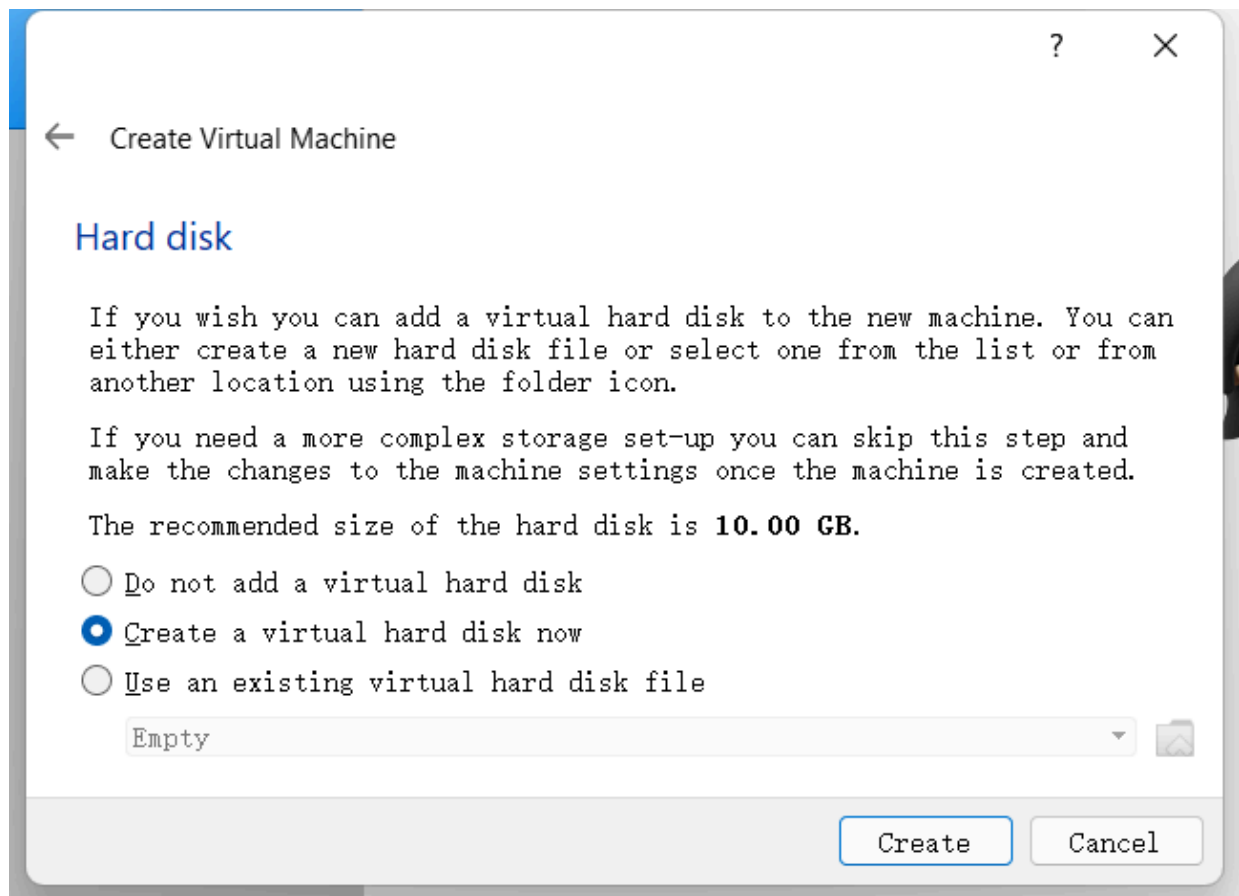
Enter the name of the VM and the location where the VM is stored, select the VM type as Linux, and select the ubuntu 64-bit version to proceed to the next step.



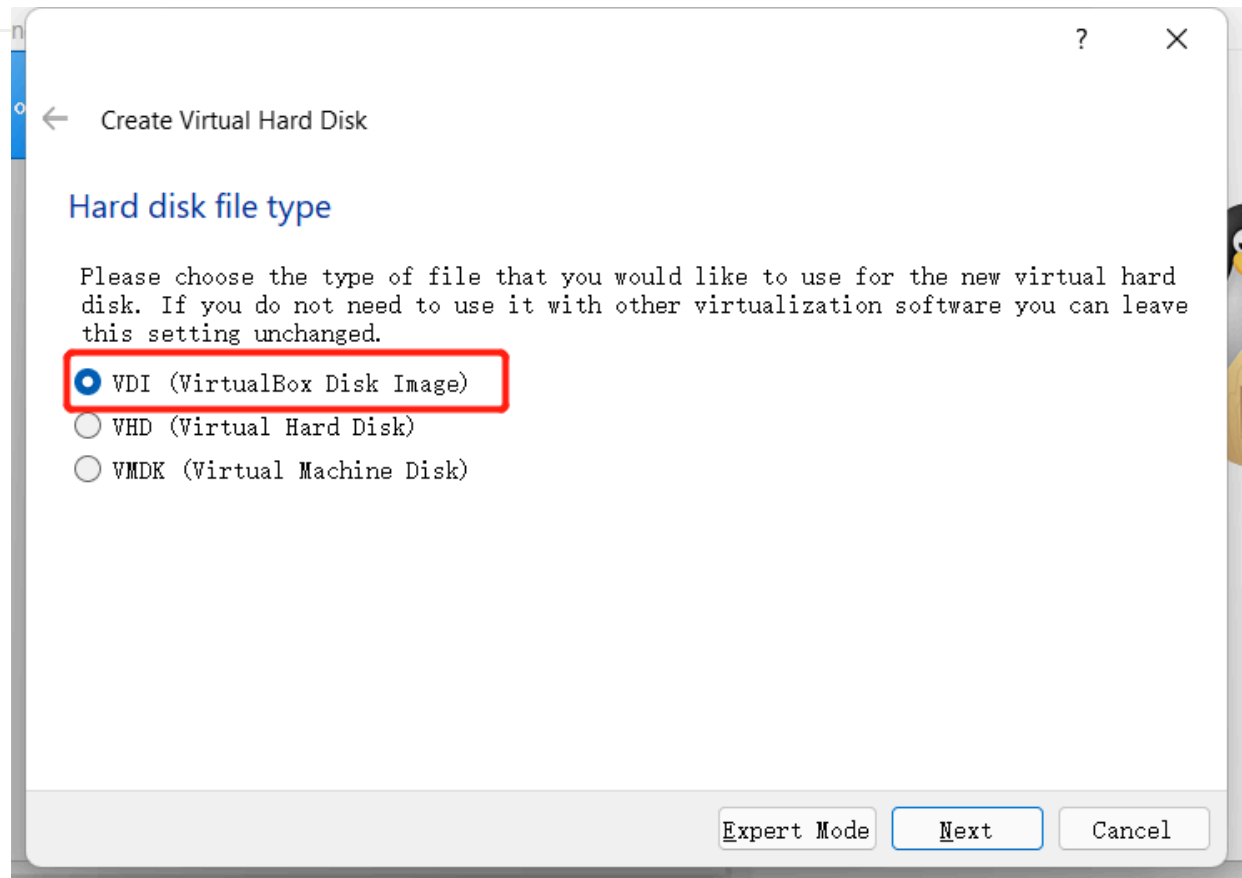
Configure the memory size according to your needs and proceed to the next step.



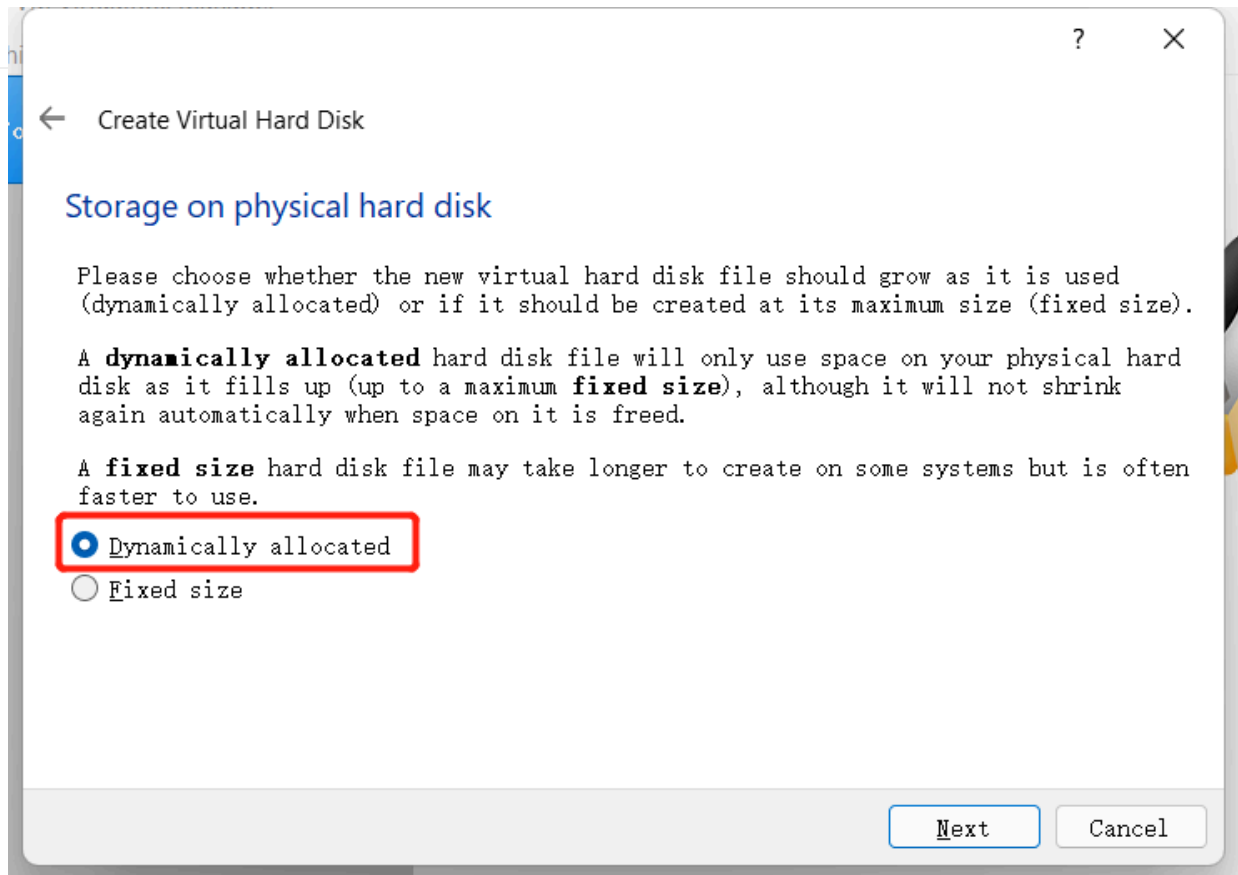
Select **Create virtual hard disk** now to create it.



Select **VDI** for Virtual Hard Disk Type and proceed to the next step.



Allocate the size of the virtual hard disk, since the ubuntu system needs to be installed, and it will also operate in the system, it is recommended that the size should not be less than 20G.



Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.



2.2 Import the Ubuntu system

2.2.1 Download the Ubuntu system

Please choose the Ubuntu version according to your needs to install it.

Note: ROS2 requires version 20.04 to be downloaded.

[16.04 version](#)[18.04 version](#)[20.04 version](#)

The installation method and process are the same for all three versions, and version 18.04 is used as an example for installation

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

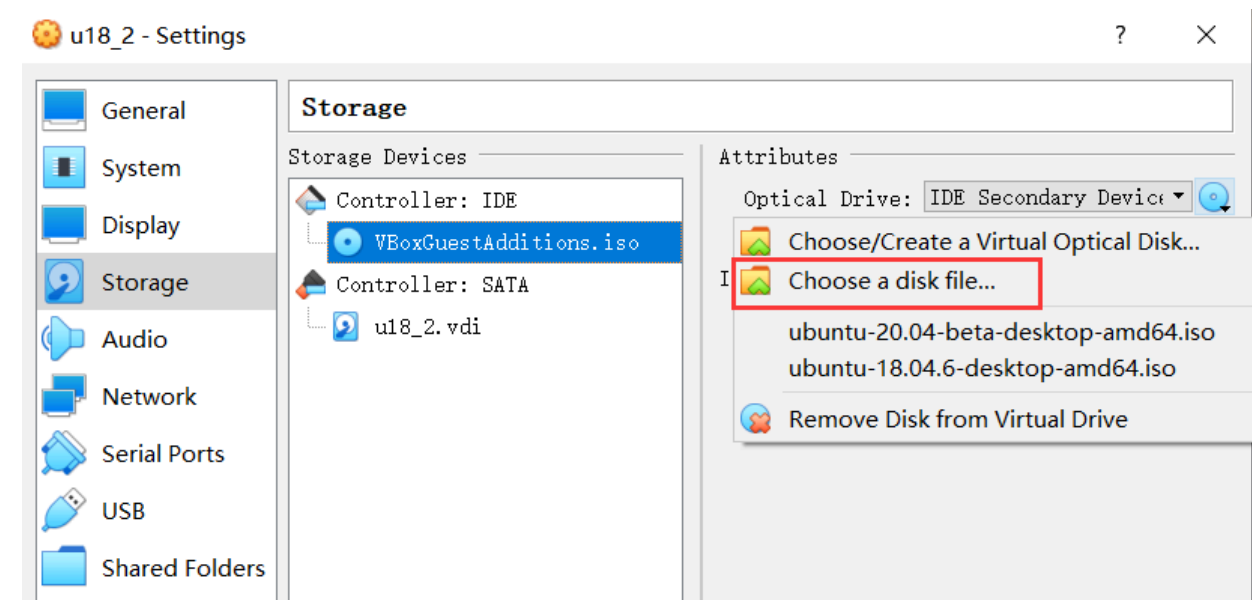
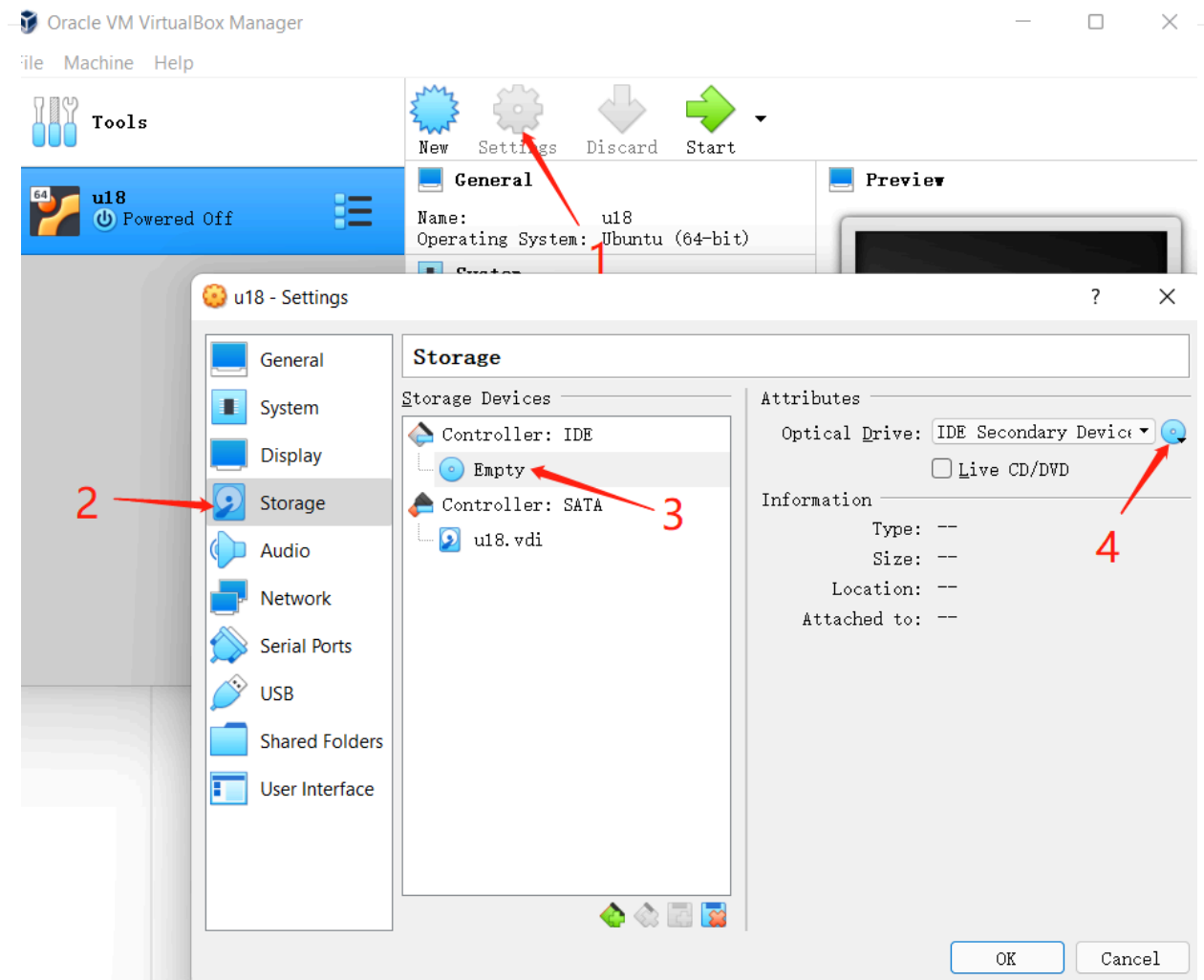
Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers (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)

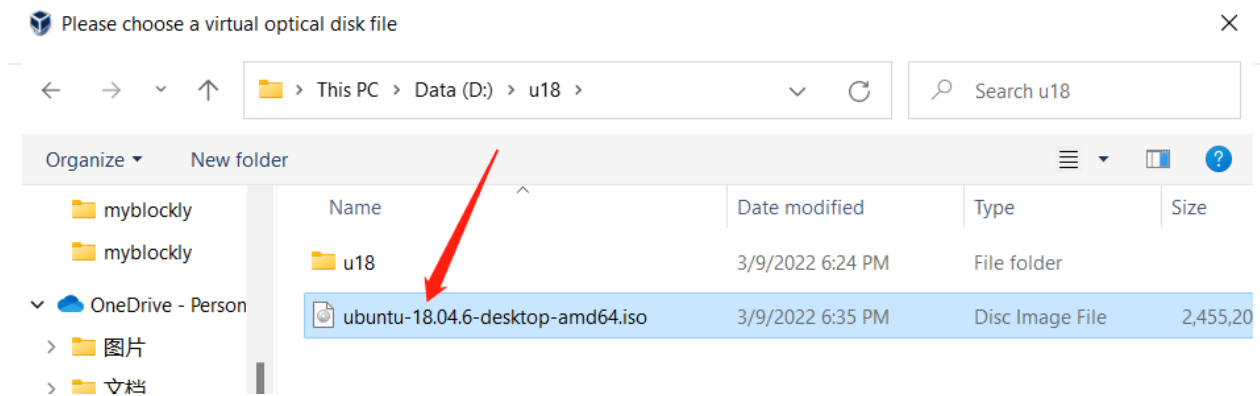
After the download is completed, there is a file as shown in the figure:

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 to the virtual machine

Locate the previously installed virtual machine in the virtual box, go to **Settings**, and assign a disc to the controller in **Storage**:





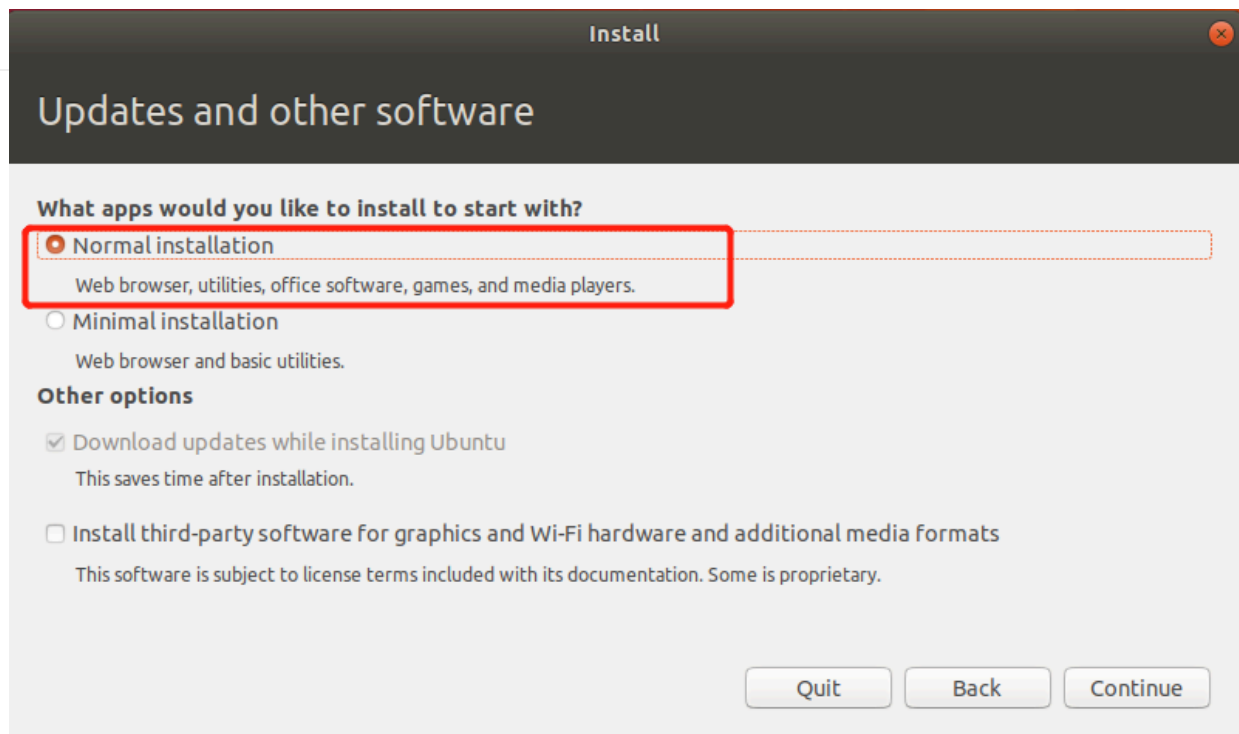
Then open the virtual machine for ubuntu installation and click start.

2.2.3 ubuntu installation

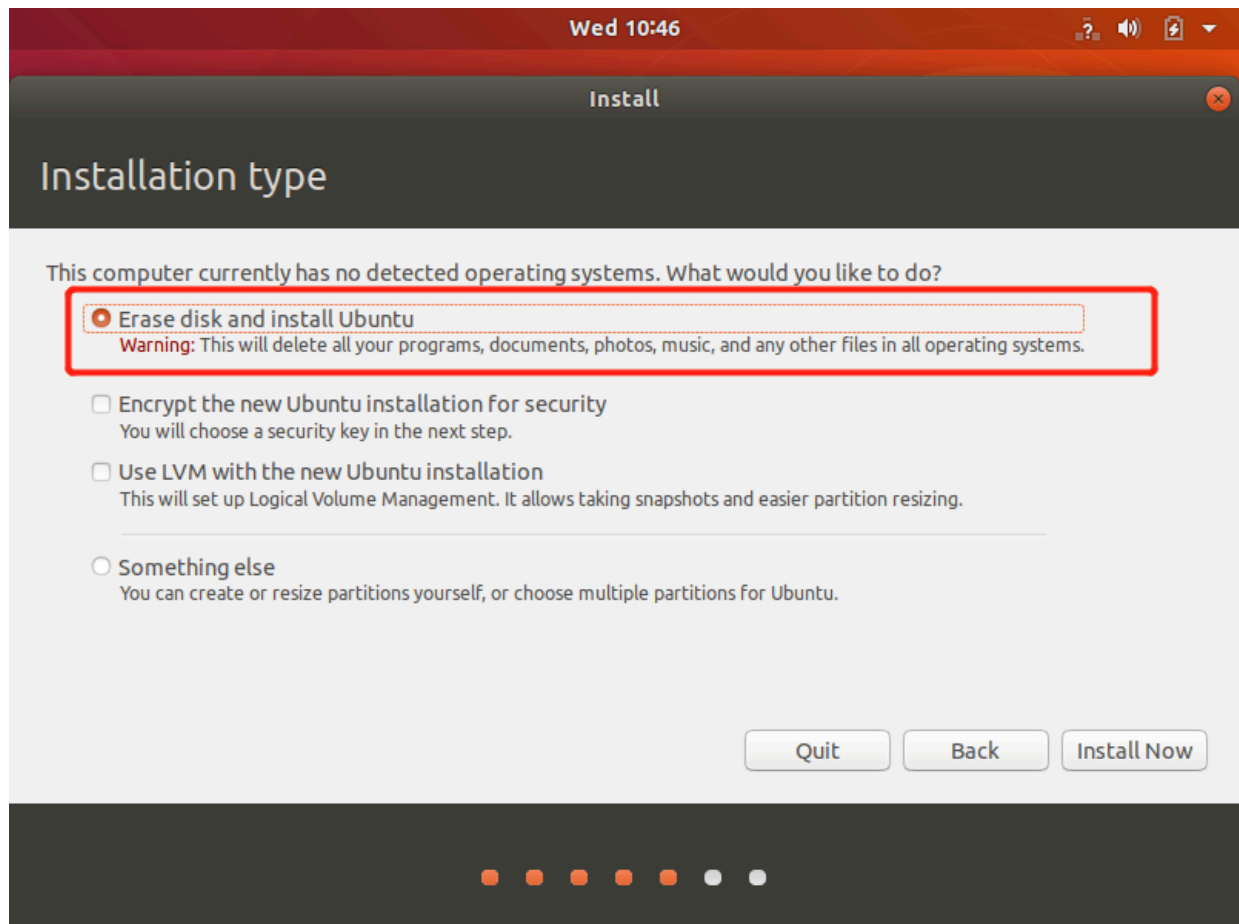
Wait for the system to boot, enter the **Welcome** interface, select "Chinese (Simplified)", and click the "Install Ubuntu" button;



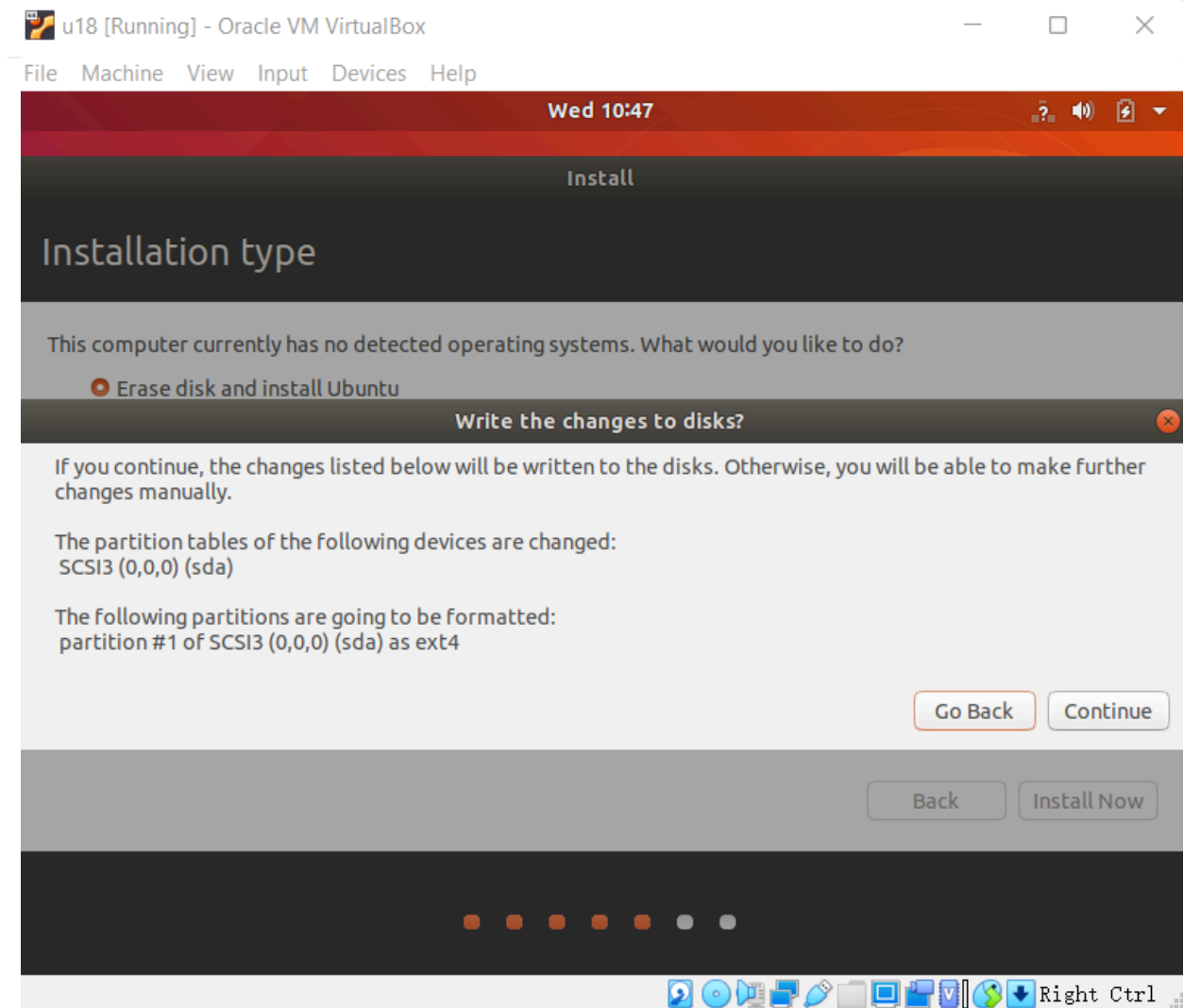
Click "Continue" button;



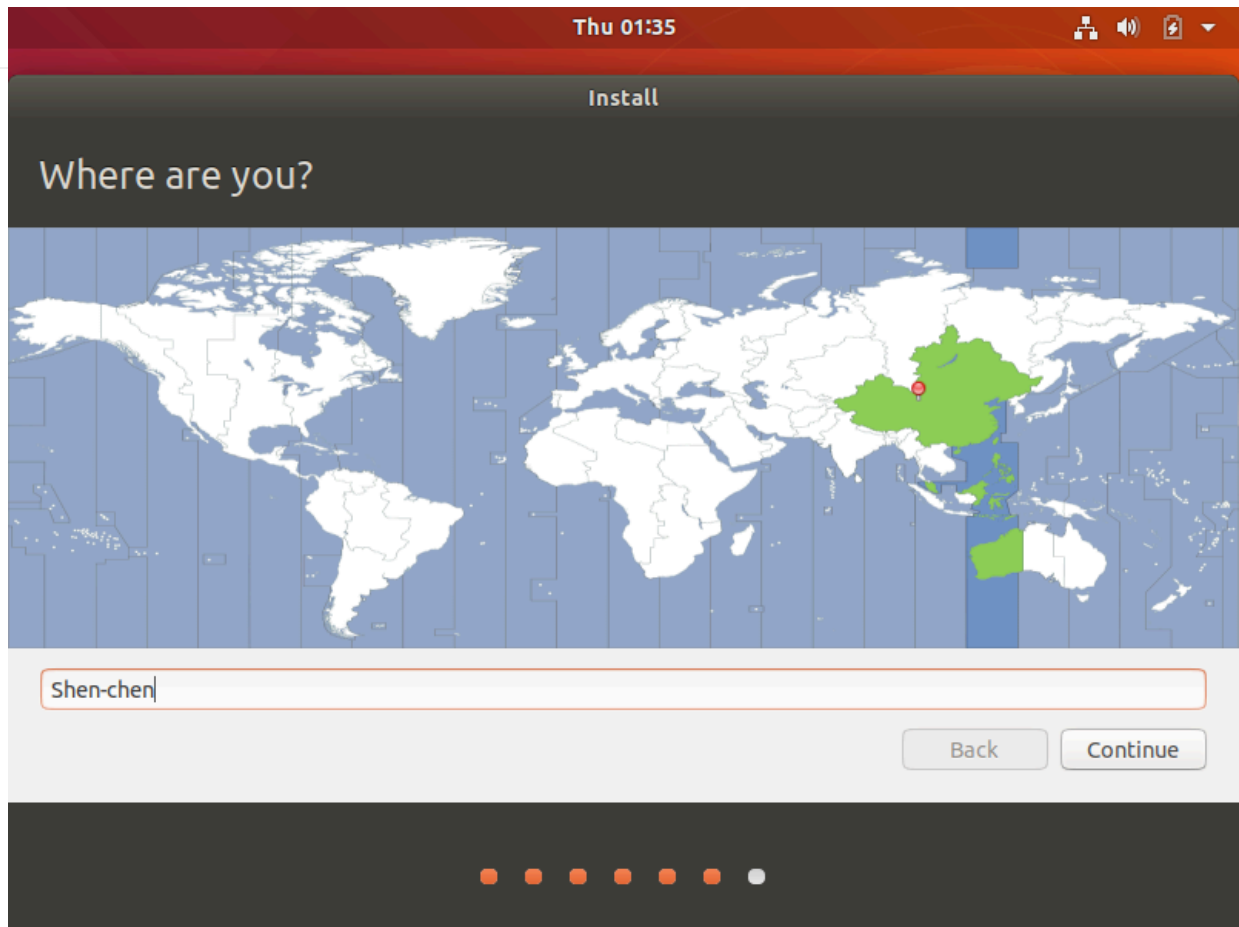
Select the "Clear entire disk and install Ubuntu" option, and click the "Install Now" button;



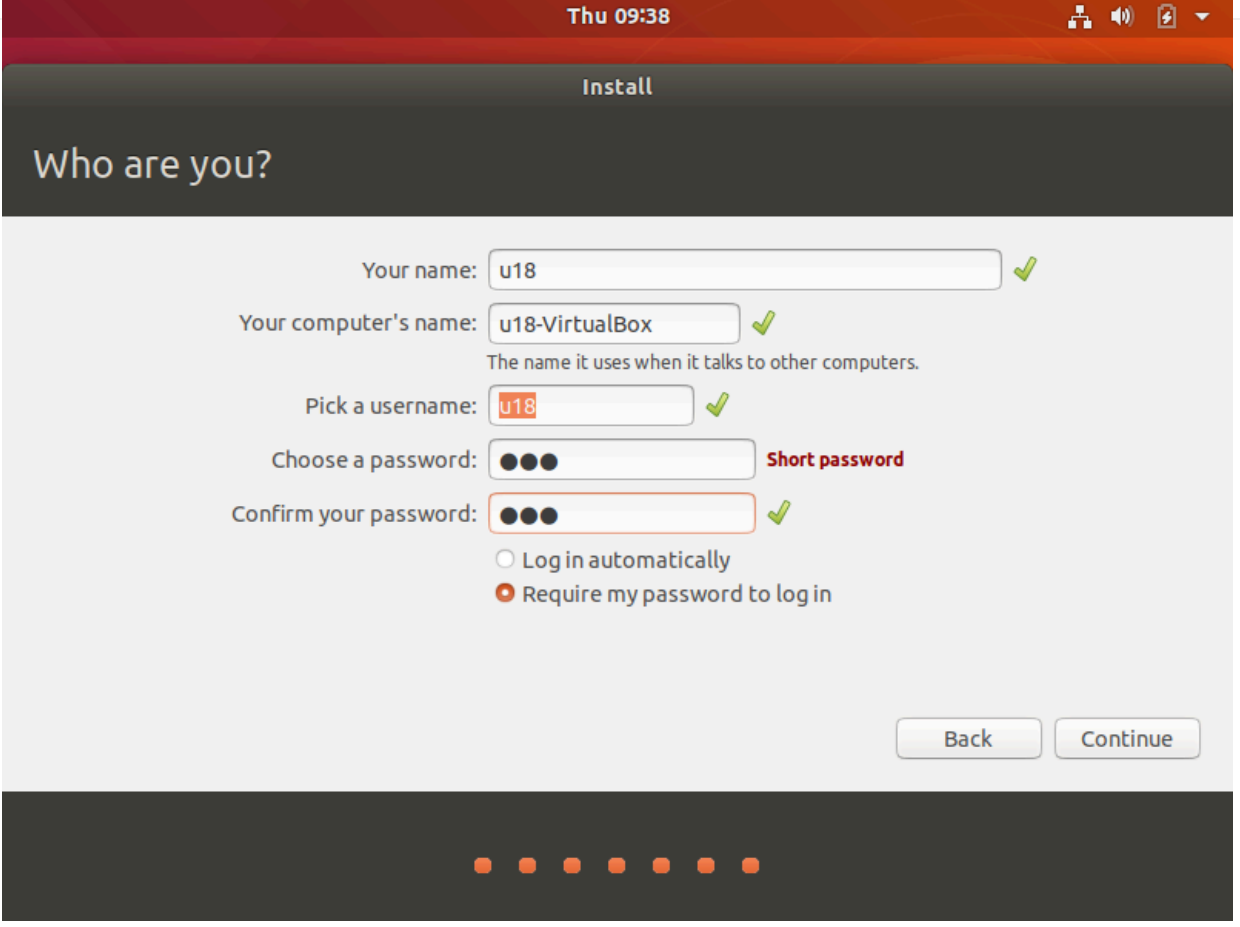
Click the "Continue" button in the pop-up dialog box;



Set a geographic position and click the "Continue" button;



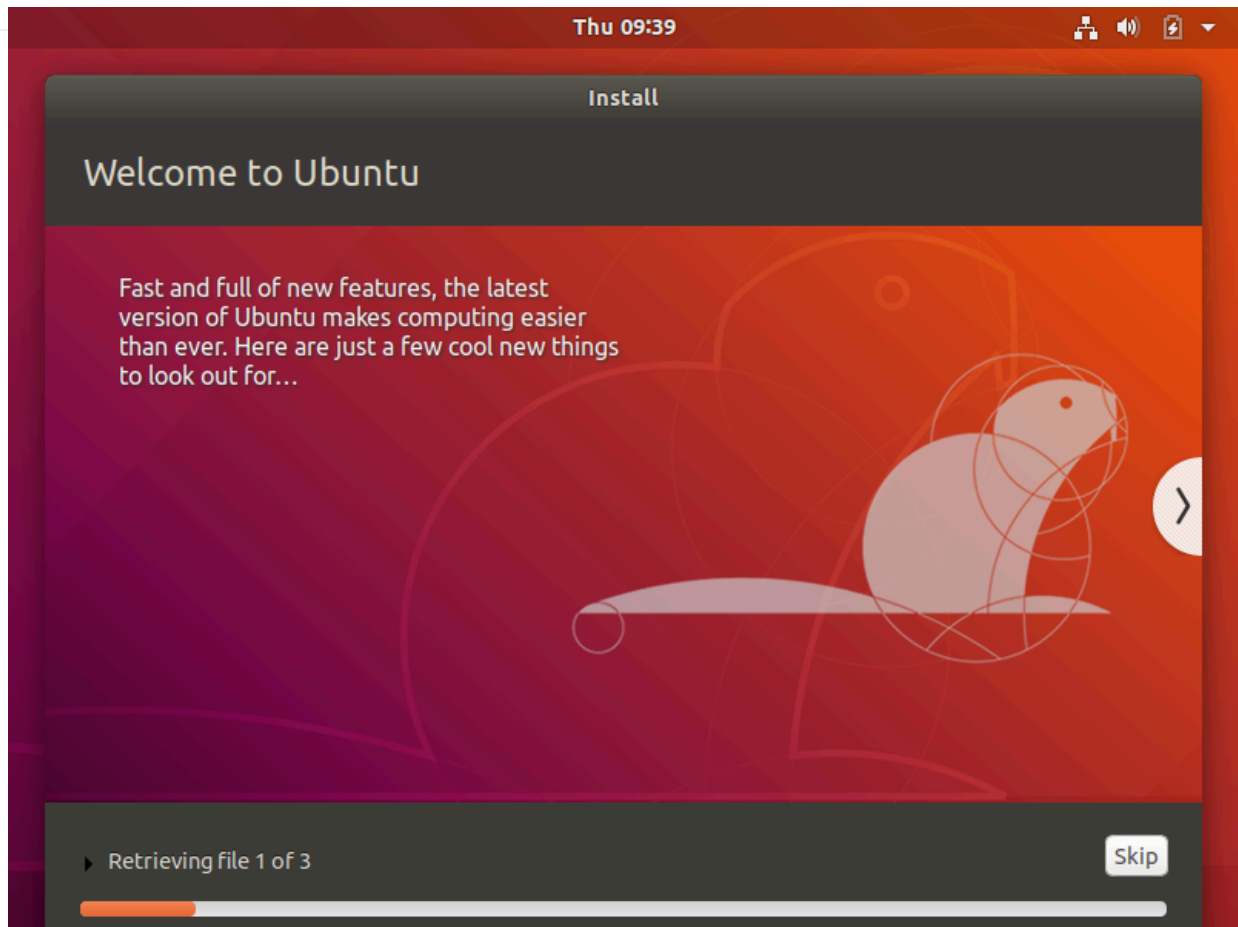
Set a user name and password, and click the “Continue” button;



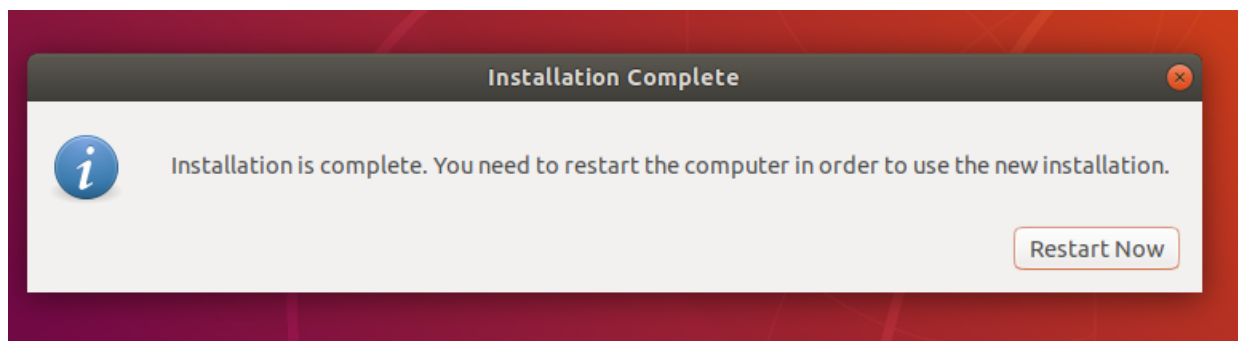
The image shows the 'Who are you?' screen of the Ubuntu installer. The window title is 'Install'. The top bar shows the time 'Thu 09:38' and system icons. The main heading is 'Who are you?'. The form contains the following fields and options:

- 'Your name:' with the value 'u18' and a green checkmark.
- 'Your computer's name:' with the value 'u18-VirtualBox' and a green checkmark. Below it is the text 'The name it uses when it talks to other computers.'
- 'Pick a username:' with the value 'u18' and a green checkmark.
- 'Choose a password:' with three black dots, a red border, and the text 'Short password' in red.
- 'Confirm your password:' with three black dots, a red border, and a green checkmark.
- Two radio buttons: 'Log in automatically' (unselected) and 'Require my password to log in' (selected).
- 'Back' and 'Continue' buttons at the bottom right.
- A progress bar at the very bottom with seven orange dots, the second one from the left being filled.

Enter the system installation interface, and wait patiently;



After installation is complete, click the "Restart now" button in the pop-up dialog box to complete the installation.



1 ROS2 Environment building

1.1 Installing ROS2

Building a basic development environment requires the installation of the robot operating system (short for ROS2), MoveIt2 and a git version manager. The installation methods and procedures are described respectively below.

1.1.1 Selecting a version

There is a one-to-one relationship between ROS2 and ubuntu. Different versions of ubuntu correspond to different versions of ROS2. The reference website is as follows: <http://docs.ros.org/en/foxy/Releases.html>

Here are the ROS versions supported by Ubuntu: | ROS2 version | release date | Maintenance deadline | Ubuntu version | | :---: | :---: | :---: | :---: | | [Foxy](#) | 2020.6 | 2023.5 | Ubuntu 20.04(Focal Fossa) | | [Galactic](#) | 2021.5 | 2022.11 | Ubuntu 20.04(Focal Fossa) | | [Humble](#) | 2022.5 | 2027.5 | Ubuntu 22.04(Jammy Jellyfish) |

Install the ROS version corresponding to the Ubuntu version you have installed.

If the versions are different, the downloading will fail. The system we choose here is Ubuntu 20.04(recommend), so the corresponding ROS2 version is ROS2 Foxy.

- NOTE: At present, we do not provide any reference for installing ROS in windows. If necessary, refer to <http://docs.ros.org/en/foxy/Installation/Alternatives/Windows-Development-Setup.html>

1.1.2 Installing ROS2

1 Adding a software source

There is no ROS2 software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS2. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

- official source:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg]
http://packages.ros.org/ros2/ubuntu $(source /etc/os-release && echo $UBUNTU_CODENAME) main" |
sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, huawei cloud is:

```
echo "deb [arch=$(dpkg --print-architecture)] https://repo.huaweicloud.com/ros2/ubuntu/ $(lsb_release -
cs) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

2 Setting a key

Configuring a public key. This step is to let the system confirm that our path is safe, so that there is no problem in downloading the file. Otherwise it will be deleted immediately after downloading:

```
sudo apt install curl gnupg2 -y
```

```
curl -s https://gitee.com/ohhuo/rosdistro/raw/master/ros.asc | sudo apt-key add -
```

3 Installation

After adding a new software source, you need to update the software source list. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

```
sudo apt-get update
```


Execute **Installing ROS2**. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command according to your Ubuntu version:

```
# Ubuntu 20.04 foxy version
```

```
sudo apt install ros-foxy-desktop
```

```
# Ubuntu 20.04 galactic version
```

```
sudo apt install ros-galactic-desktop
```

```
# Ubuntu 22.04 humble version
```

```
sudo apt install ros-humble-desktop
```

The installation process takes a long time, so please wait with patience.

Refresh environment variables after installation is complete:

```
source /opt/ros/foxy/setup.bash
```

1.1.3 Set up the ros2 environment

In order to avoid re-validating the ROS2 function path every time the terminal window is closed, we can set the path to an environment variable, so that each time you open a new terminal, the ROS2 function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: Ctrl+Alt+T):

Execute the following commands:

```
# Ubuntu 20.04 foxy version
```

```
# Add the ros2 environment to the environment variables of the current console
```

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

```
# Ubuntu 20.04 galactic version
```

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

```
# Ubuntu 22.04 humble version
```

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

```
source ~/.bashrc
```

Installing a ROS2 extra dependency

Input the following command in the terminal to install a ROS2 extra dependency, and open a console terminal (shortcut key: **Ctrl+Alt+T**):

```
sudo apt install python3-argcomplete -y
```

```
sudo apt install ros-foxy-xacro
```

```
sudo apt-get install python3-colcon-common-extensions
```

```
# Ubuntu 20.04 foxy version
```

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

```
# Ubuntu 20.04 galactic version
```

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

```
# Ubuntu 22.04 humble version
```

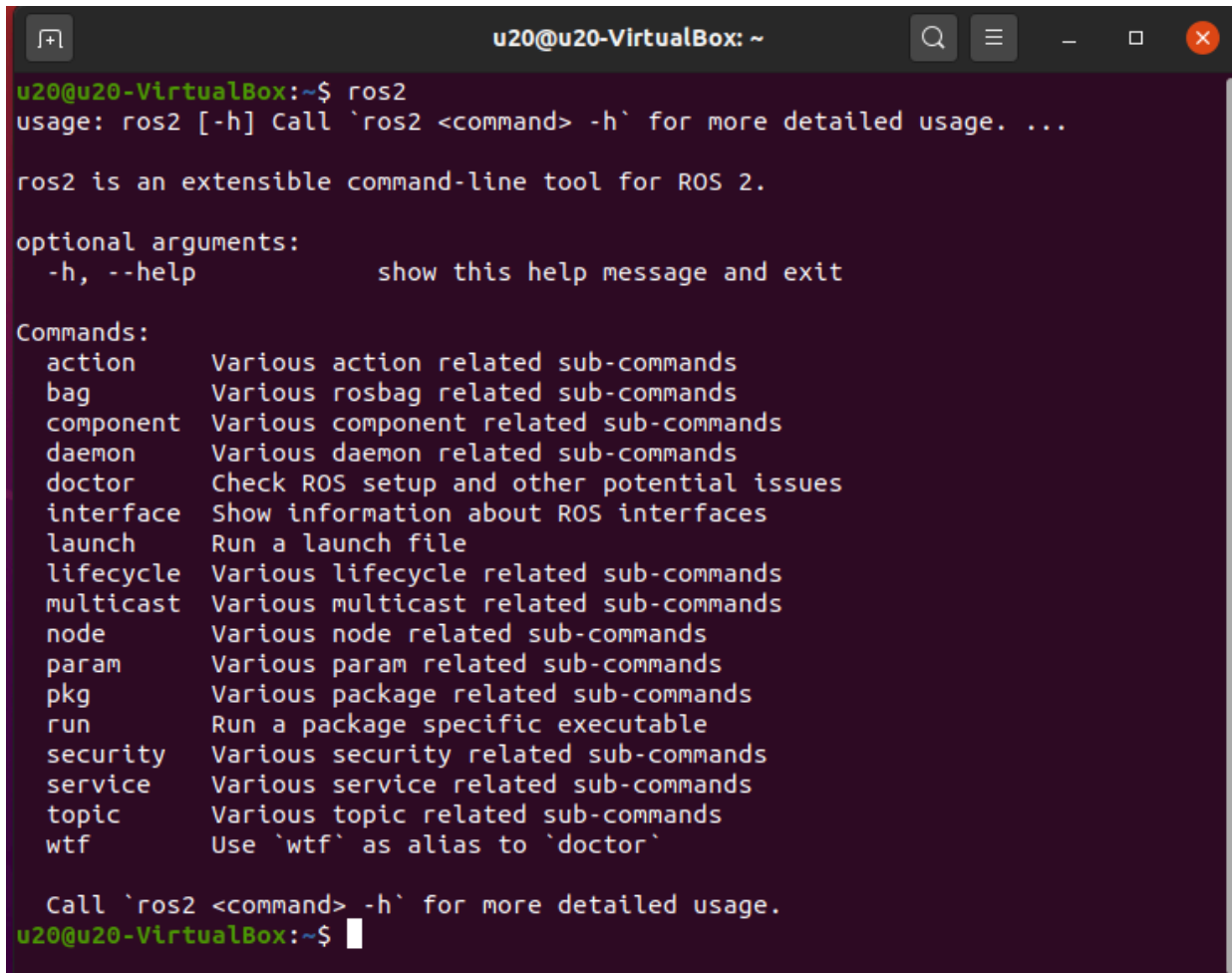
```
sudo apt install ros-humble-joint-state-publisher-gui
```

1.1.4 Verify installation

To verify whether ROS2 has been installed successfully, open a console terminal (shortcut key: Ctrl+Alt+T) and execute the following command in the terminal:

```
ros2
```

When the following interface is displayed, it means that ROS2 has been installed successfully.



```
u20@u20-VirtualBox: ~  
u20@u20-VirtualBox:~$ ros2  
usage: ros2 [-h] Call `ros2 <command> -h` for more detailed usage. ...  
  
ros2 is an extensible command-line tool for ROS 2.  
  
optional arguments:  
  -h, --help            show this help message and exit  
  
Commands:  
  action                Various action related sub-commands  
  bag                   Various rosbag related sub-commands  
  component             Various component related sub-commands  
  daemon               Various daemon related sub-commands  
  doctor               Check ROS setup and other potential issues  
  interface             Show information about ROS interfaces  
  launch               Run a launch file  
  lifecycle             Various lifecycle related sub-commands  
  multicast            Various multicast related sub-commands  
  node                 Various node related sub-commands  
  param                Various param related sub-commands  
  pkg                  Various package related sub-commands  
  run                  Run a package specific executable  
  security             Various security related sub-commands  
  service              Various service related sub-commands  
  topic                Various topic related sub-commands  
  wtf                  Use `wtf` as alias to `doctor`  
  
  Call `ros2 <command> -h` for more detailed usage.  
u20@u20-VirtualBox:~$
```

1.2 Movelt2 Installation

Movelt2 is the composition of a series of function packages for movement operations in ros, mainly including motion planning, collision detection, kinematics, 3D perception, operation control and other functions.

1.2.1 Updating the software source list

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to update the software source list:

```
sudo apt-get update
```

1.2.2 Installing MoveIt2

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to execute the installation of MoveIt2:

```
# Ubuntu 20.04 foxy version

sudo apt-get install ros-foxy-moveit

# Ubuntu 20.04 galactic version

sudo apt-get install ros-galactic-moveit

# Ubuntu 22.04 galactic version

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

1.3 git Installation

1.3.1 Updating the software source list

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to update the software source list:

```
sudo apt-get update
```

1.3.2 Installing git

Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window to execute the installation of git:

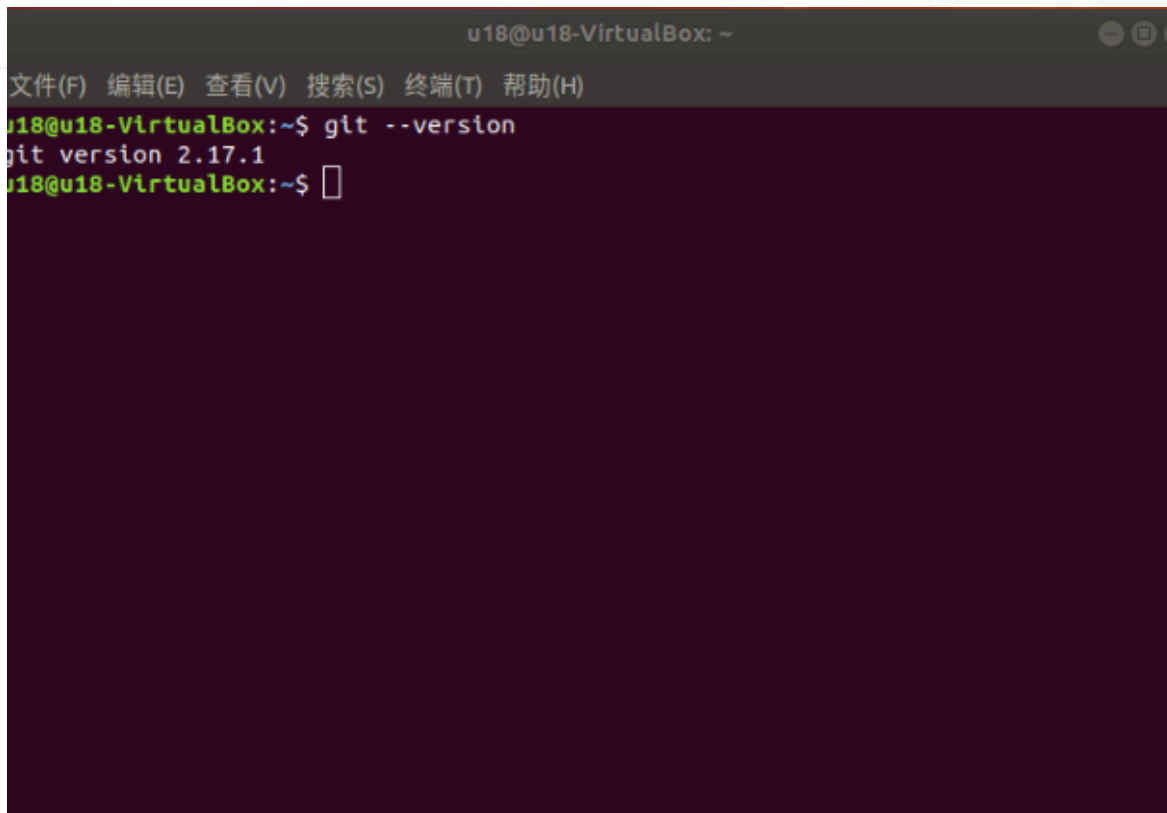
```
sudo apt-get install git
```

1.3.3 Verifying git

Read the git version, open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command on the terminal window:

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.

A terminal window titled 'u18@u18-VirtualBox: ~' with a menu bar in Chinese. The terminal shows the command 'git --version' being executed, resulting in the output 'git version 2.17.1'.

```
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:~$
```

2 MyarmM750 Installation

The MyarmM750 is a ROS package launched by ElephantRobotics that adapts to its desktop six-axis robotic arm Myarm series.

Project Address: https://github.com/elephantrobotics/mycobot_ros2/tree/myArmC650_M750

2.1 precondition

Before installing the package, make sure you have a ROS2 workspace.

Here we give a sample command to **create a workspace**, The default is colcon_ws, Open a console terminal (Ctrl+Alt+T), On the command line, enter the following command:

```
mkdir -p ~/colcon_ws/src # Create folders
```

```
cd ~/colcon_ws/src # Go to the folder
```

```
colcon build # Build the code in the workspace
```

Add the environment for the workspace

The official default ROS2 workspace is colcon_ws.

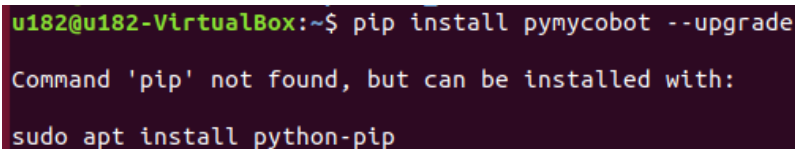
```
echo "source ~/colcon_ws/install/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

2.2 Installation

NOTE:

- This package relies on ROS2 and MoveIT2, make sure to install ROS and MoveIT successfully before use.
- The interaction between this package and the real robotic arm relies on the PythonApi - pymycobot
- The API project locations are:<https://github.com/elephantrobotics/pymycobot>
- Quick Installation: `pip install pymycobot --upgrade`

When executing the Pip Instal Pimicobo-Upgrad command, if the following error message appears:



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

Enter the following command to install pip when prompted:

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

- If your Ubuntu system is version 20.04, run the `sudo apt install python3-pip` installation pip installation and run the terminal again after the installation is complete

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

- The installation method depends on Git, so make sure you have Git installed on your computer.

The official default ROS1 workspace is colcon_ws.

Open the terminal:

```
cd ~/colcon_ws/src # Enter the src folder of the workspace
```

```
# Clone the code on github
```

```
git clone --depth 1 https://github.com/elephantrobotics/mycobot\_ros2.git
```

```
cd .. # return to the workspace
```

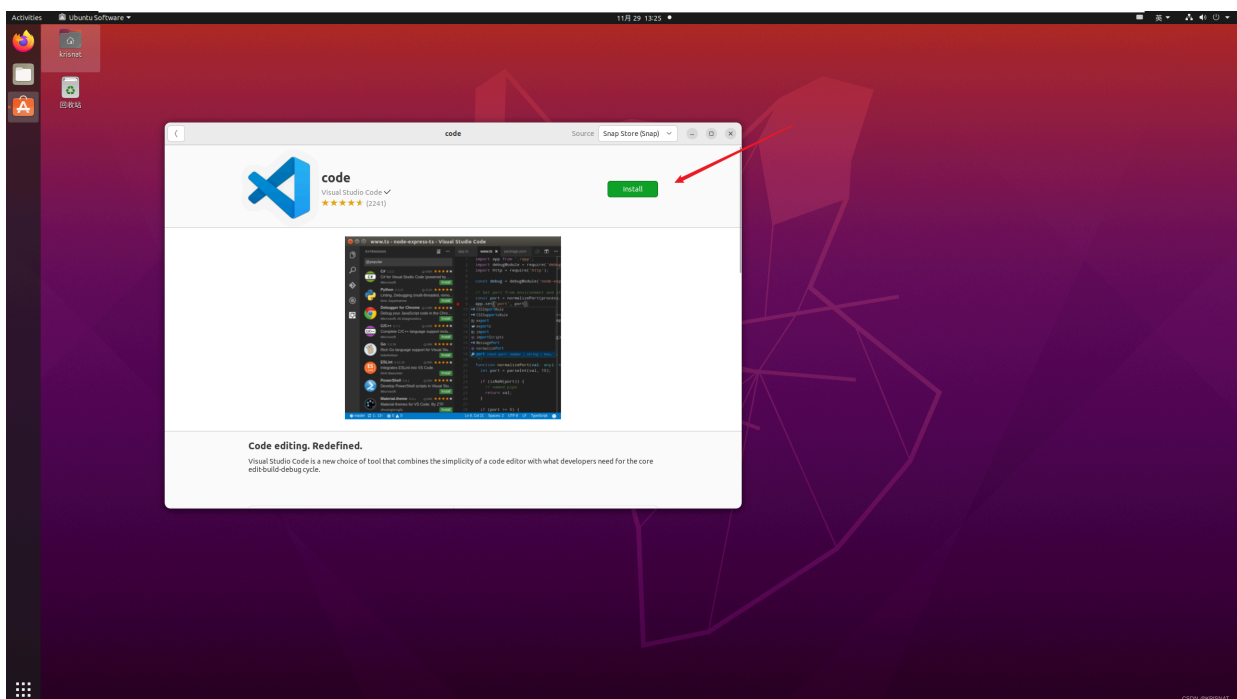
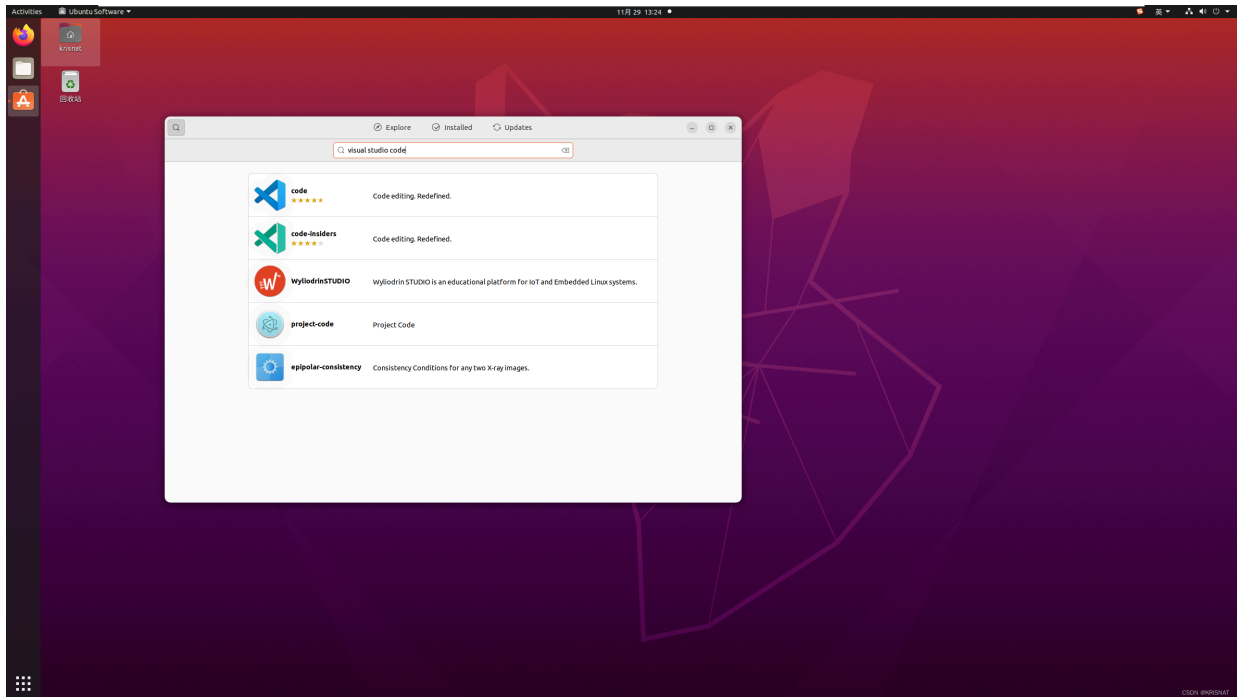
```
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile  
python scripts every time you adjust them
```

```
source install/setup.bash # add environment variables
```


At this point, the environment construction is partially completed

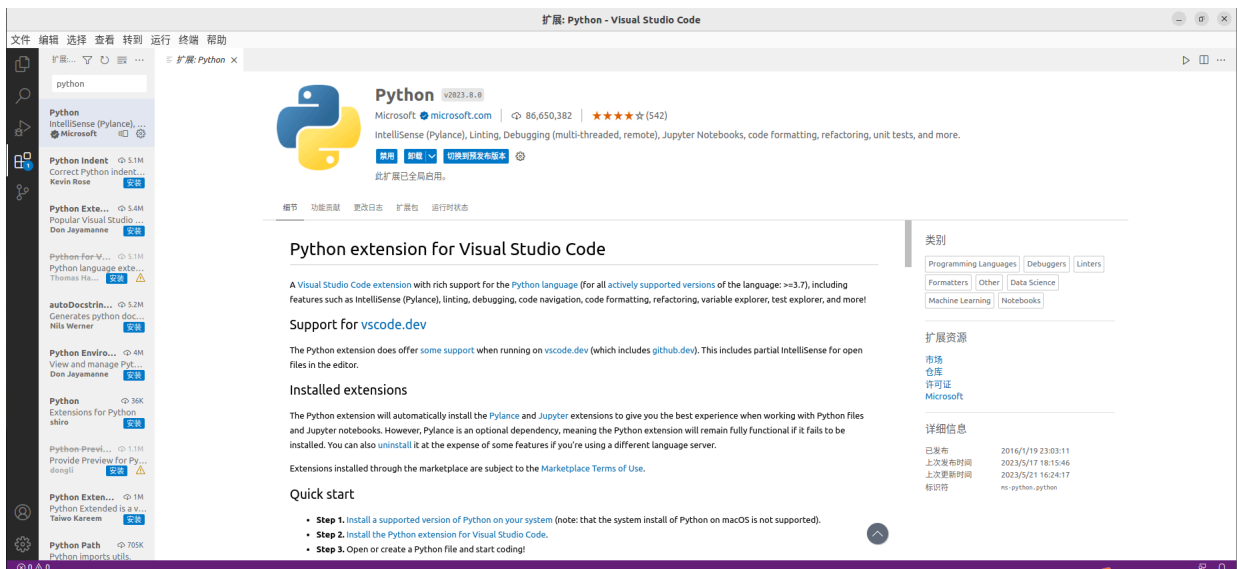
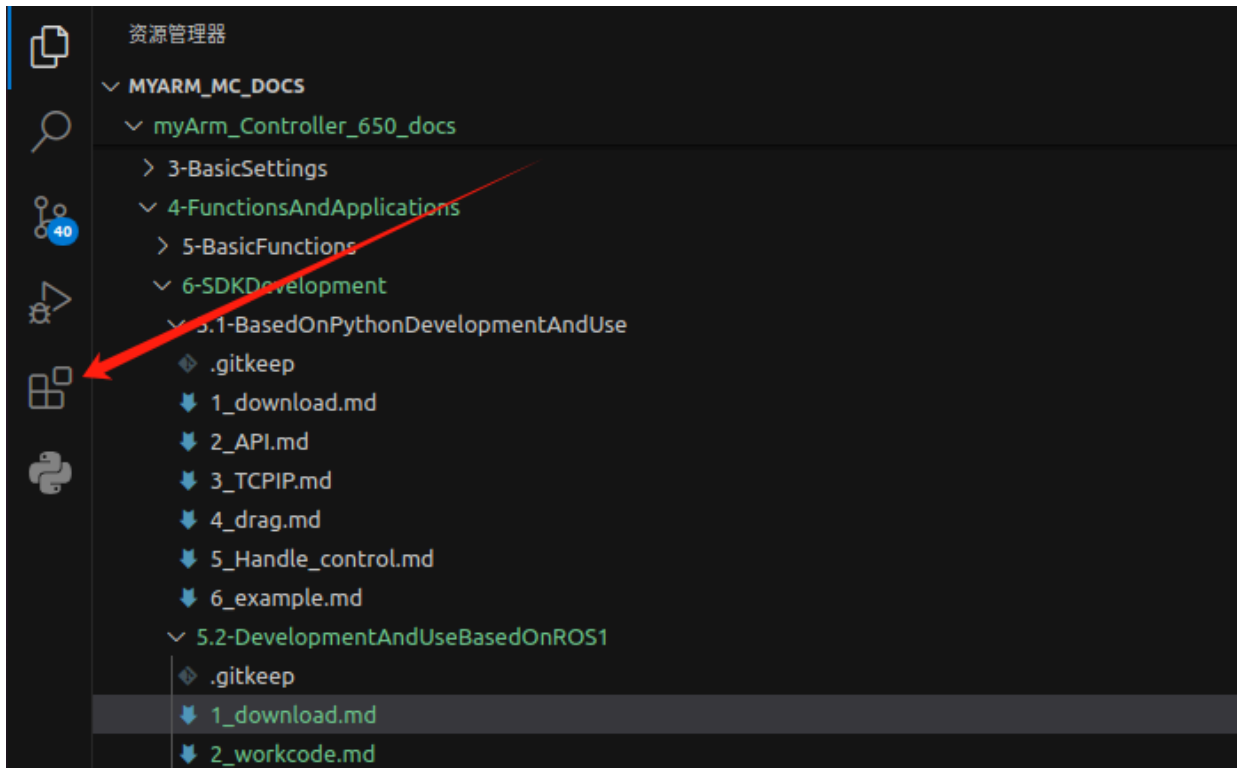
For the subsequent programming, we also need to download one VS code

Here we will choose the most convenient method: Installing in Ubuntu Software [Easy & Convenient]



After installation, we also need to install the python environment

To run a python file, click on the left extension (ctrl+shift+X) - search for python, download and install



After installation, you can click on the upper left file, select New File, select the file type as Python, write your own Python code, and then click Run in the upper menu bar

3 Brief introduction and use of rviz2

rviz2 is a three-dimensional visualization platform in ROS. On the one hand, it can realize the graphical display of external information, and on the other hand, it can also release control information to the object through rviz2, so as to realize the monitoring and control of the robot.

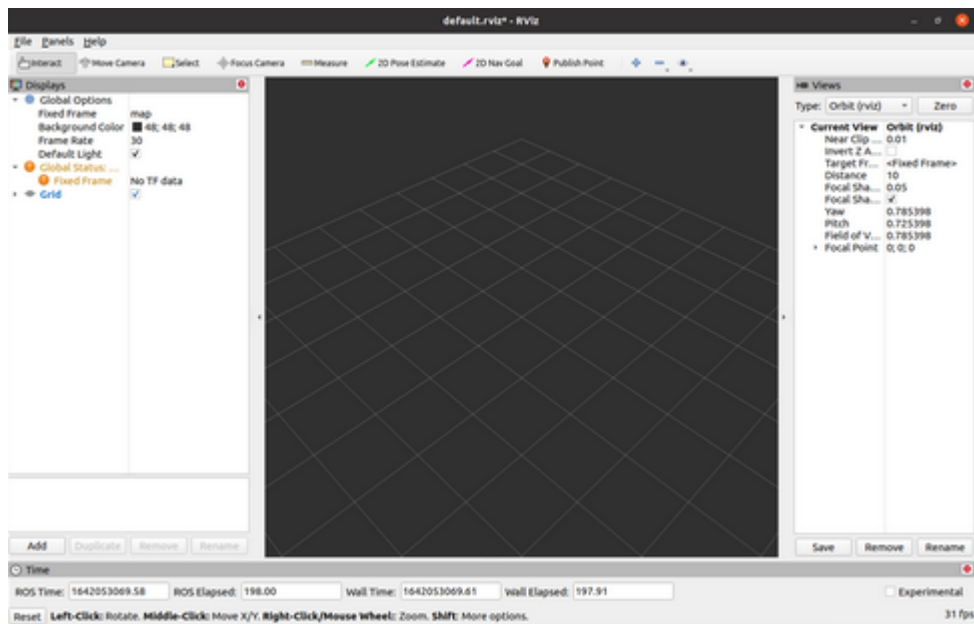
3.1 Introduction to rviz2

The successful installation of ros2 indicates that rviz2 is also successfully installed together, because the installation of ros2 includes rviz2.

Open a new terminal (shortcut Ctrl+Alt+T) enter the command to open rviz2

```
rviz2
```

Open RVIZ2 and display the following interface:



3.1.1 Introduction to each area

- On the left is a list of monitors, which are things that draw something in the 3D world and may have some options available in the display list.
- Above is the toolbar, which allows users to use various function buttons to select tools with multiple functions
- The middle part is the 3D view: it is the main screen where you can view various data in 3D. The background color, fixed frame, mesh, etc of the 3D view can be set in detail in the Global Options and Grid items displayed on the left.
- Below is the time display area, including system time and ROS time.
- The right side is the observation angle setting area, and different observation angles can be set.

We only give a rough introduction in this part, if you want to know more details, you can go to [User Guide](#) to review.

1、Emulation

We provide a number of **static and dynamic simulations** to interact with the MyarmM750.

1.1 Static simulation

Static simulation here refers to the use of the sliding module in rviz2 to interact with the MyarmM750 in the simulation

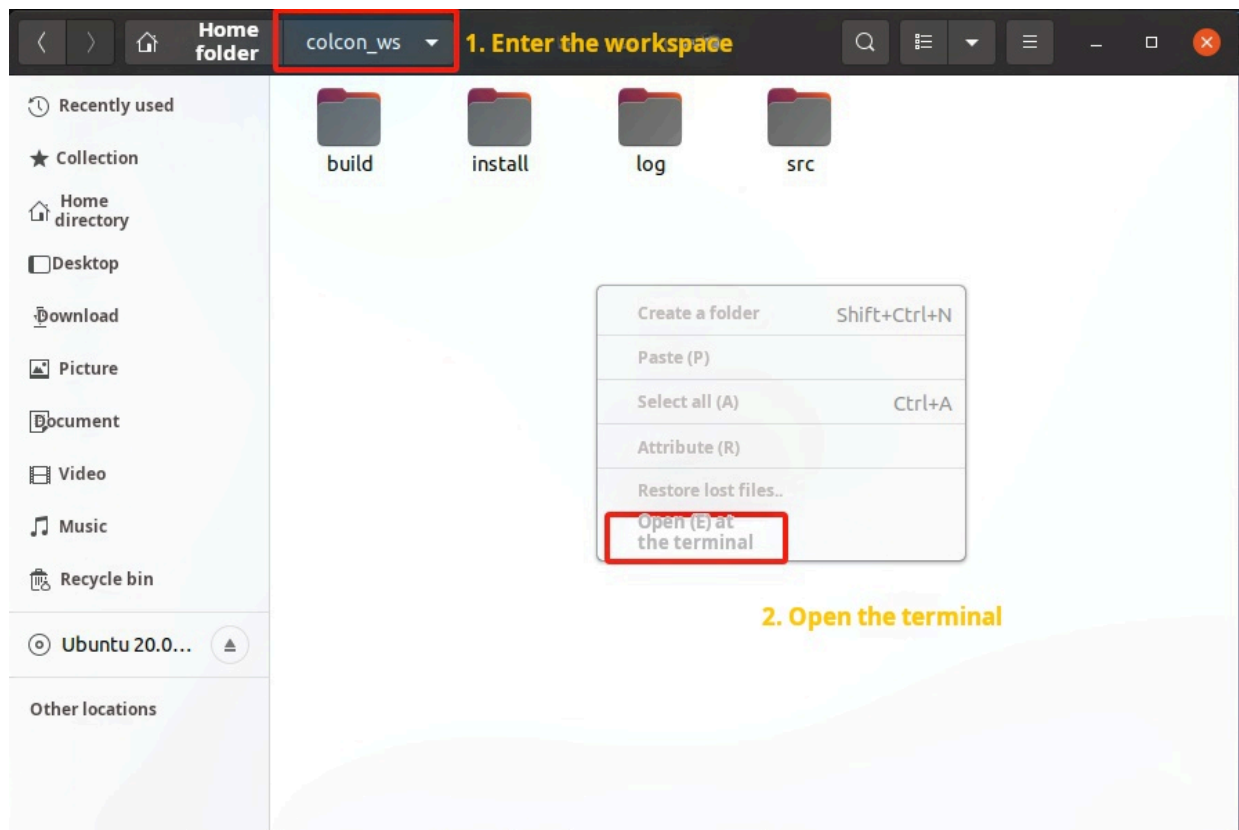
In the Workspace Open Terminal command line, enter:

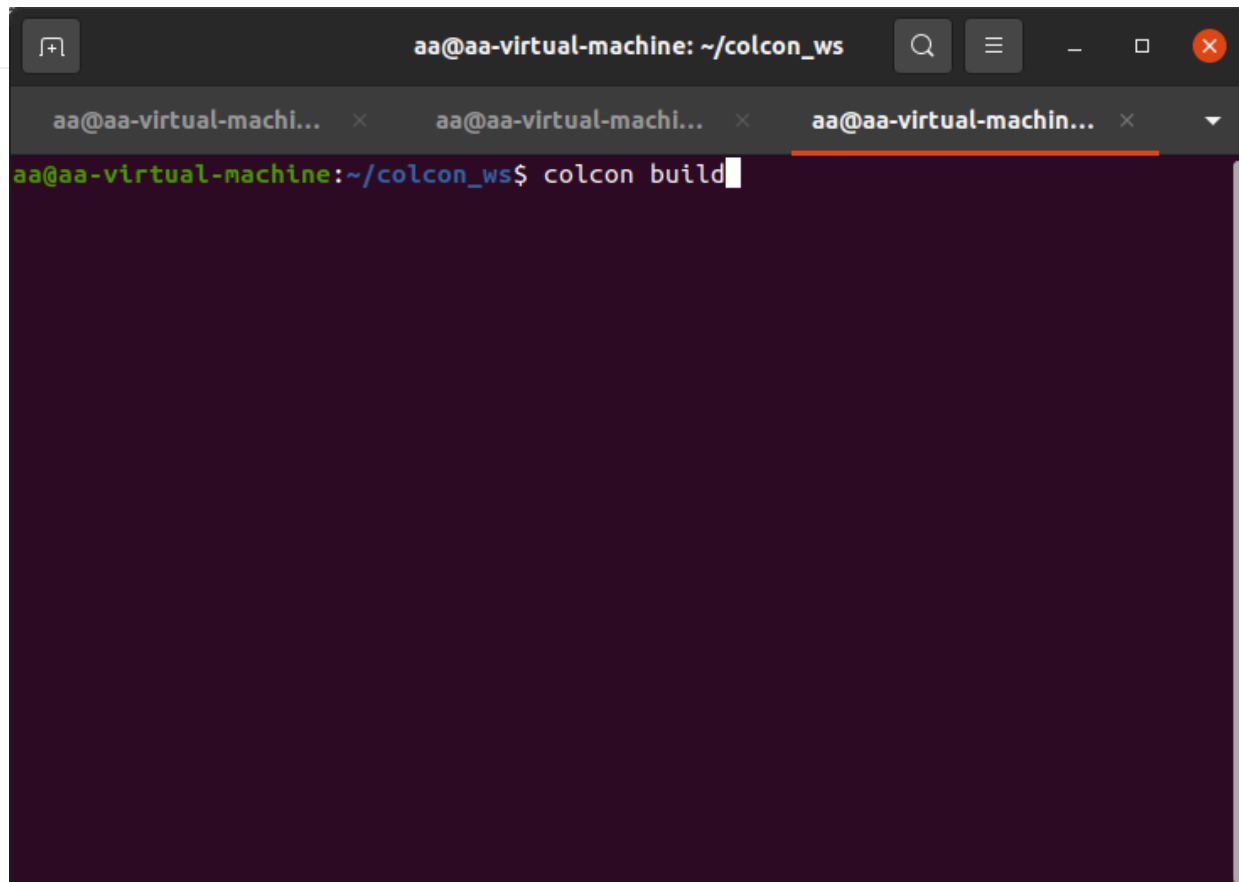
```
colcon build
```

Create a new terminal:

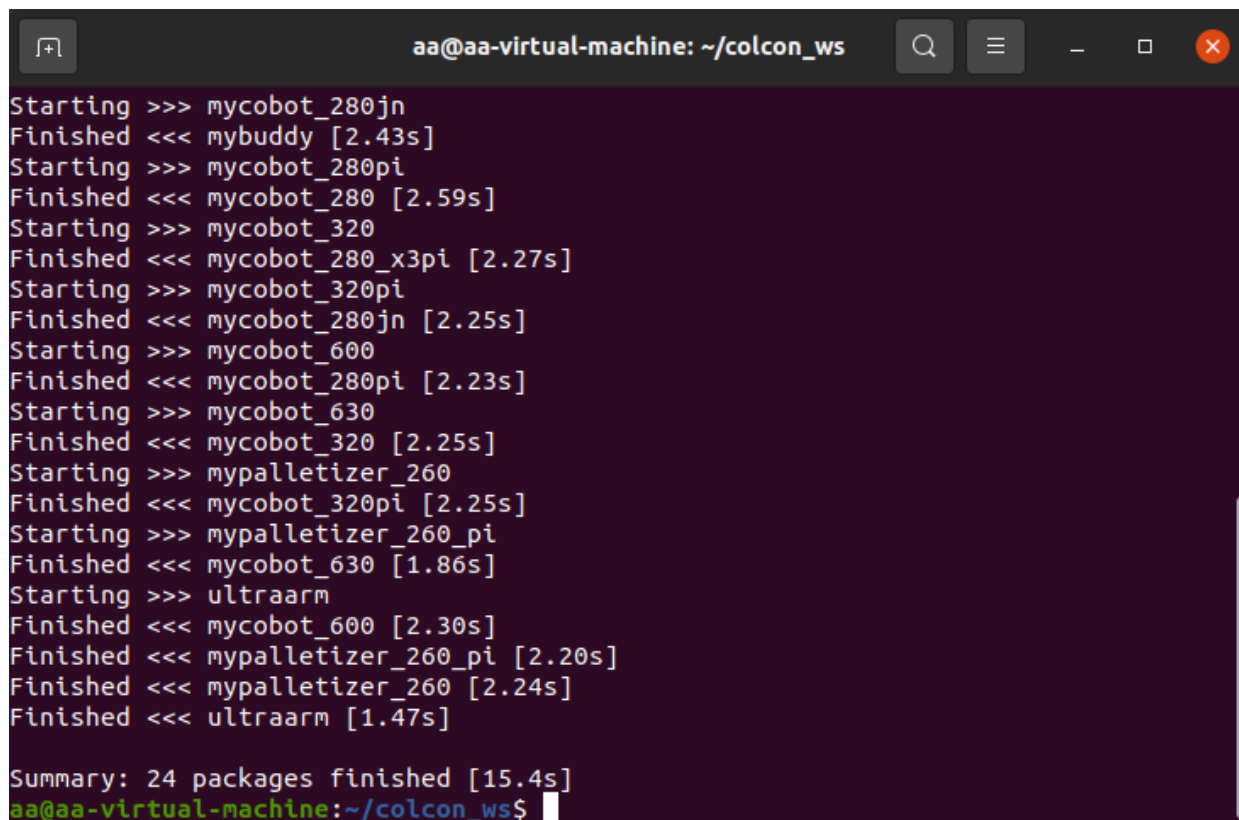
```
source install/setup.bash
```

```
ros2 launch myarm_m750 test.launch.py
```





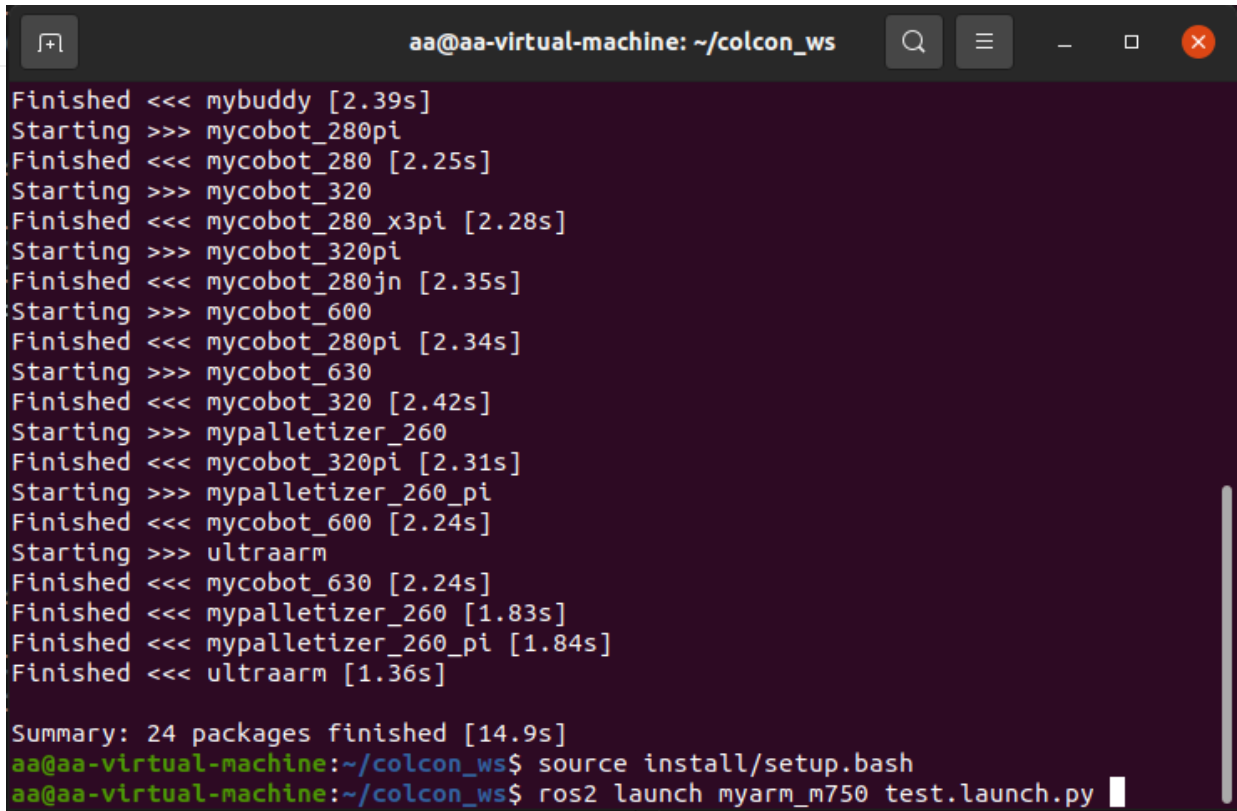
A terminal window titled 'aa@aa-virtual-machine: ~/colcon_ws'. The window has a dark background and a light-colored title bar. The terminal shows the prompt 'aa@aa-virtual-machine:~/colcon_ws\$' followed by the command 'colcon build' which has been entered but not yet executed.



A terminal window titled 'aa@aa-virtual-machine: ~/colcon_ws' showing the output of the 'colcon build' command. The output lists 24 packages being built, each with a 'Starting' and 'Finished' status and a duration in seconds. The packages are: mycobot_280jn, mybuddy, mycobot_280pi, mycobot_280, mycobot_320, mycobot_280_x3pi, mycobot_320pi, mycobot_280jn, mycobot_600, mycobot_280pi, mycobot_630, mycobot_320, mycobot_320pi, mypalletizer_260, mycobot_320pi, mypalletizer_260_pi, mycobot_630, ultraarm, mycobot_600, mypalletizer_260_pi, mypalletizer_260, and ultraarm. The summary shows 24 packages finished in 15.4s.

```
Starting >>> mycobot_280jn
Finished <<< mybuddy [2.43s]
Starting >>> mycobot_280pi
Finished <<< mycobot_280 [2.59s]
Starting >>> mycobot_320
Finished <<< mycobot_280_x3pi [2.27s]
Starting >>> mycobot_320pi
Finished <<< mycobot_280jn [2.25s]
Starting >>> mycobot_600
Finished <<< mycobot_280pi [2.23s]
Starting >>> mycobot_630
Finished <<< mycobot_320 [2.25s]
Starting >>> mypalletizer_260
Finished <<< mycobot_320pi [2.25s]
Starting >>> mypalletizer_260_pi
Finished <<< mycobot_630 [1.86s]
Starting >>> ultraarm
Finished <<< mycobot_600 [2.30s]
Finished <<< mypalletizer_260_pi [2.20s]
Finished <<< mypalletizer_260 [2.24s]
Finished <<< ultraarm [1.47s]

Summary: 24 packages finished [15.4s]
aa@aa-virtual-machine:~/colcon_ws$
```



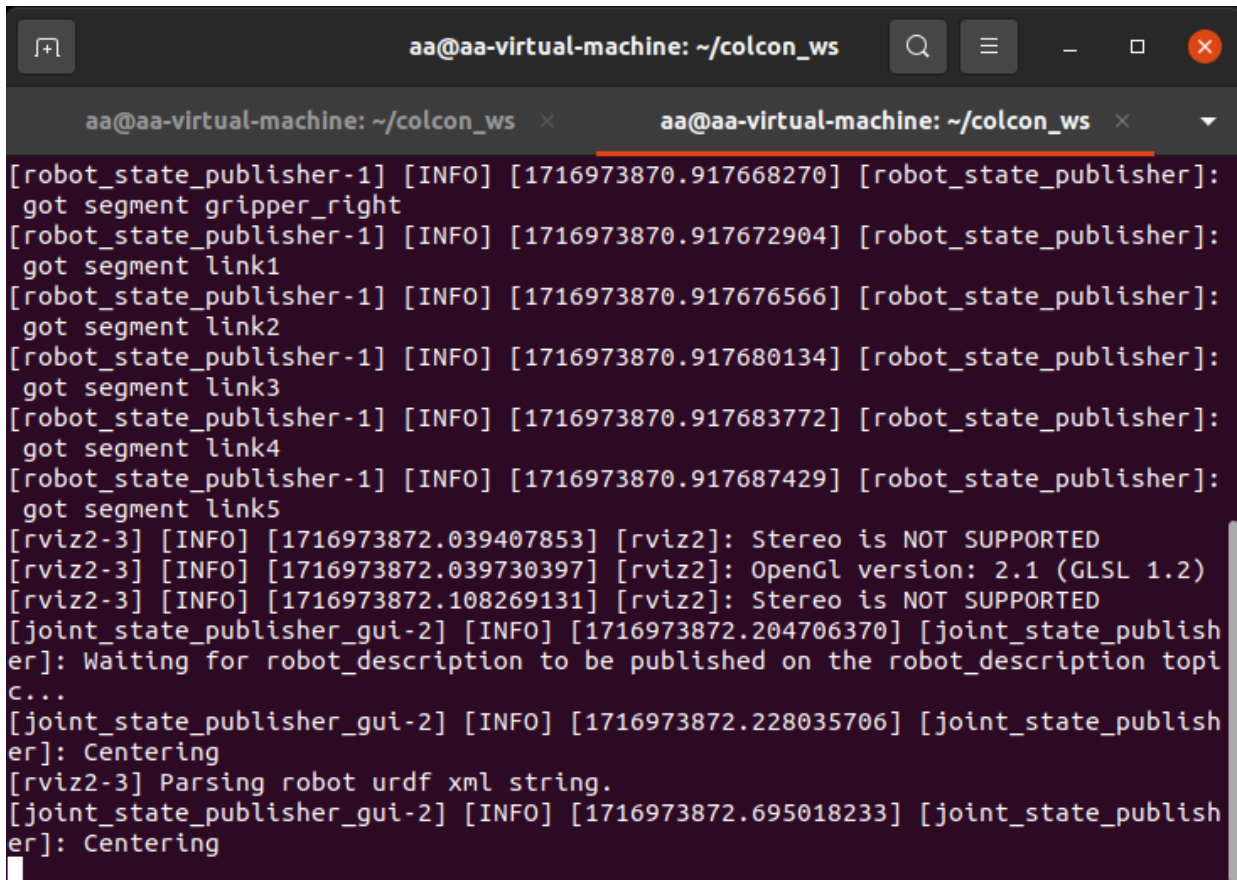
```

aa@aa-virtual-machine: ~/colcon_ws
Finished <<< mybuddy [2.39s]
Starting >>> mycobot_280pi
Finished <<< mycobot_280 [2.25s]
Starting >>> mycobot_320
Finished <<< mycobot_280_x3pi [2.28s]
Starting >>> mycobot_320pi
Finished <<< mycobot_280jn [2.35s]
Starting >>> mycobot_600
Finished <<< mycobot_280pi [2.34s]
Starting >>> mycobot_630
Finished <<< mycobot_320 [2.42s]
Starting >>> mypalletizer_260
Finished <<< mycobot_320pi [2.31s]
Starting >>> mypalletizer_260_pi
Finished <<< mycobot_600 [2.24s]
Starting >>> ultraarm
Finished <<< mycobot_630 [2.24s]
Finished <<< mypalletizer_260 [1.83s]
Finished <<< mypalletizer_260_pi [1.84s]
Finished <<< ultraarm [1.36s]

Summary: 24 packages finished [14.9s]
aa@aa-virtual-machine:~/colcon_ws$ source install/setup.bash
aa@aa-virtual-machine:~/colcon_ws$ ros2 launch myarm_m750 test.launch.py

```

After successfully running the launch file, the terminal will display:

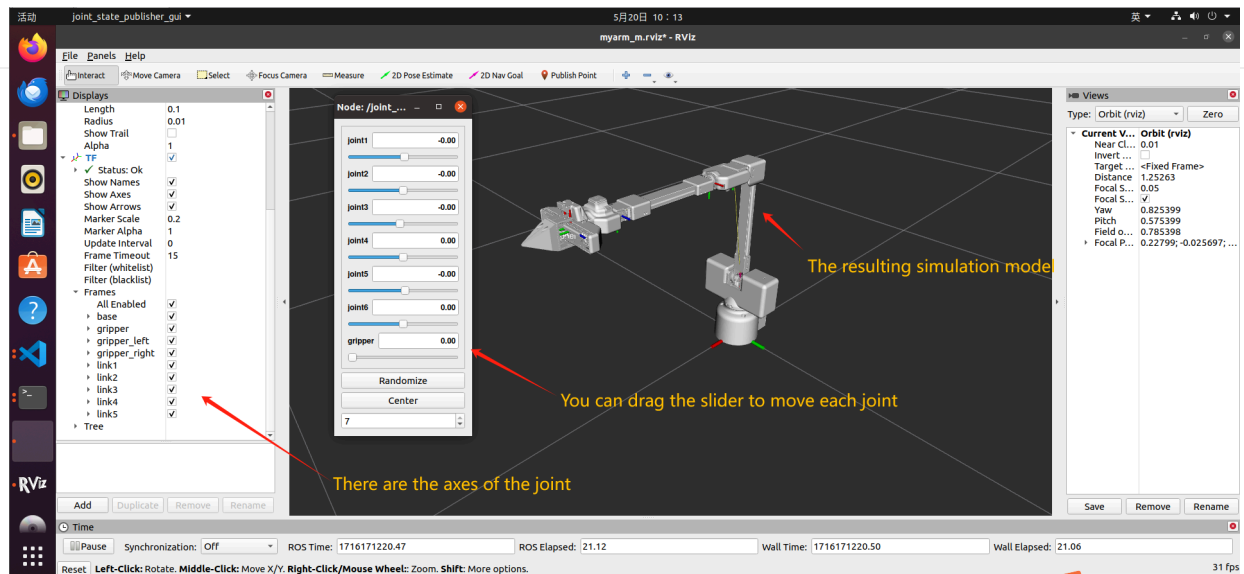


```

aa@aa-virtual-machine: ~/colcon_ws
aa@aa-virtual-machine: ~/colcon_ws
[robot_state_publisher-1] [INFO] [1716973870.917668270] [robot_state_publisher]:
got segment gripper_right
[robot_state_publisher-1] [INFO] [1716973870.917672904] [robot_state_publisher]:
got segment link1
[robot_state_publisher-1] [INFO] [1716973870.917676566] [robot_state_publisher]:
got segment link2
[robot_state_publisher-1] [INFO] [1716973870.917680134] [robot_state_publisher]:
got segment link3
[robot_state_publisher-1] [INFO] [1716973870.917683772] [robot_state_publisher]:
got segment link4
[robot_state_publisher-1] [INFO] [1716973870.917687429] [robot_state_publisher]:
got segment link5
[rviz2-3] [INFO] [1716973872.039407853] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-3] [INFO] [1716973872.039730397] [rviz2]: OpenGL version: 2.1 (GLSL 1.2)
[rviz2-3] [INFO] [1716973872.108269131] [rviz2]: Stereo is NOT SUPPORTED
[joint_state_publisher_gui-2] [INFO] [1716973872.204706370] [joint_state_publish
er]: Waiting for robot_description to be published on the robot_description topi
c...
[joint_state_publisher_gui-2] [INFO] [1716973872.228035706] [joint_state_publish
er]: Centering
[rviz2-3] Parsing robot urdf xml string.
[joint_state_publisher_gui-2] [INFO] [1716973872.695018233] [joint_state_publish
er]: Centering

```

At the same time, rviz will be opened to generate a simulation model of MyarmM750



1.2 Dynamic simulation

Dynamic simulation here refers to the interaction of the MyarmM750 in motion reality with the MyarmM750 in the simulation

First of all, we need to connect the MyarmM750 robotic arm to our computer via a USB to TypeC cable and power it on

Select the **Transponder** button and press the **OK** button



Then the screen will be displayed:



We can see that the arrow points to **"USB UART"**, then press the **"OK"** button, after entering **"NO"**, then press the **"Exit"** button, go back to the arrow pointing to **"USB UART"**, and then press the **"OK"** button, then **"OK"** will be displayed **"OK"**



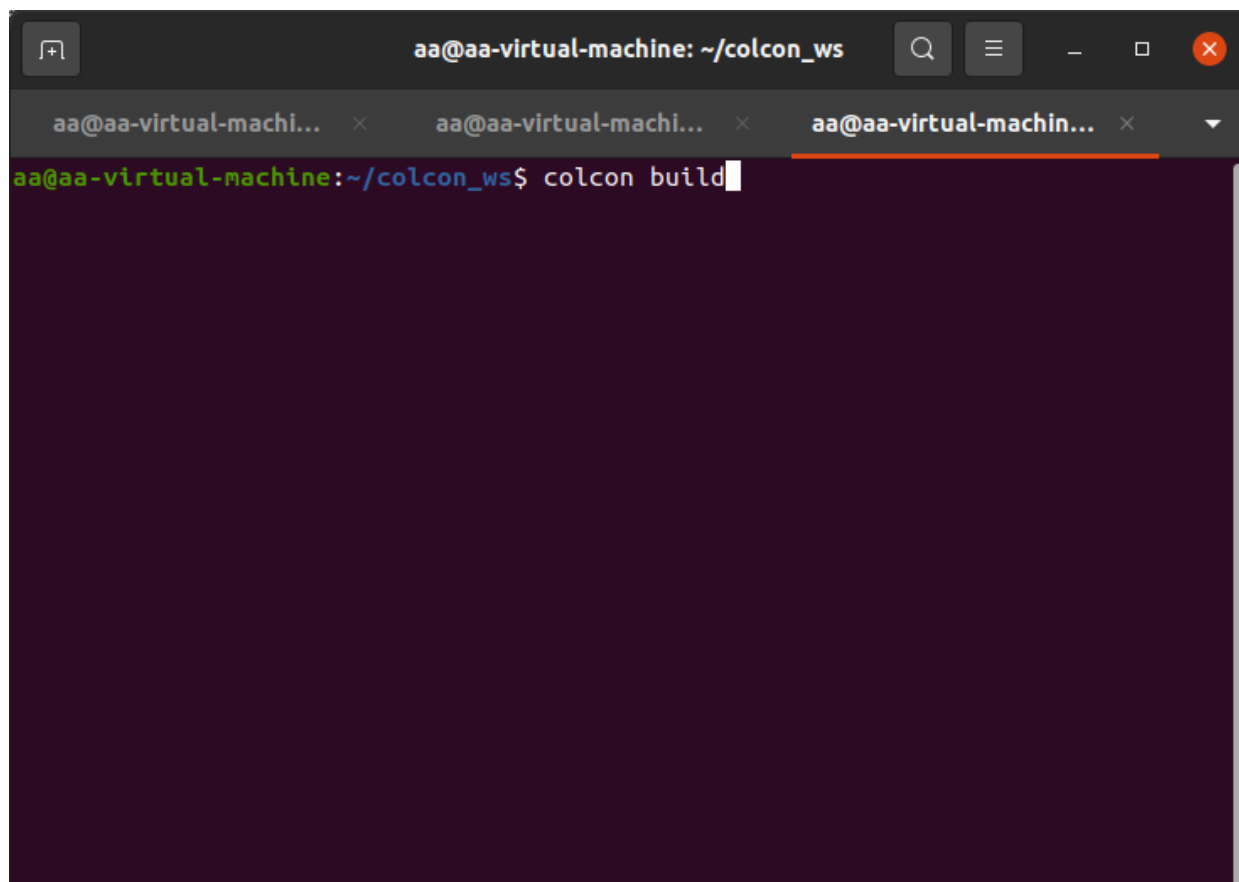
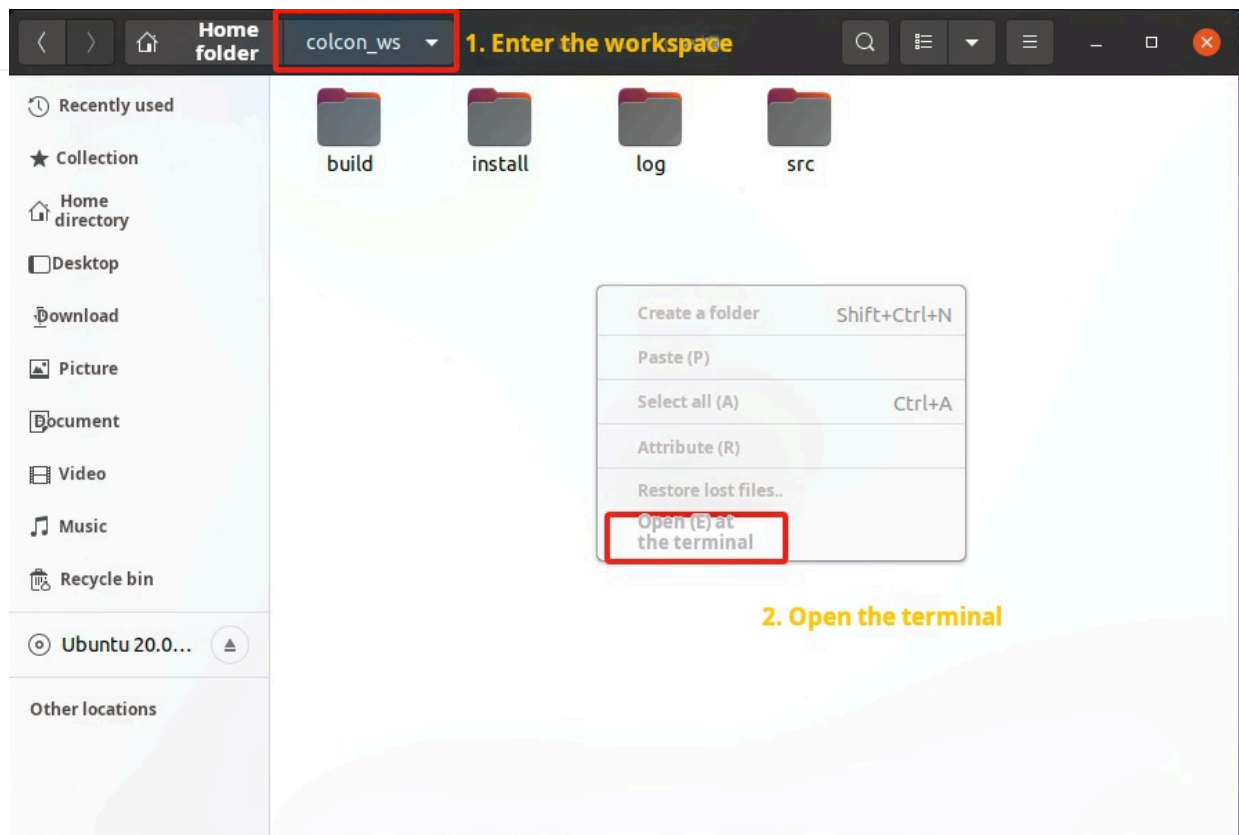
At this time, our MyarmM750 has been successfully connected to the computer

Next, enter it in the Workspace Open Terminal command line:

```
colcon build
```

```
source install/setup.bash
```

```
ros2 launch myarm_m750 test.launch.py
```




```

aa@aa-virtual-machine: ~/colcon_ws
Starting >>> mycobot_280jn
Finished <<< mybuddy [2.43s]
Starting >>> mycobot_280pi
Finished <<< mycobot_280 [2.59s]
Starting >>> mycobot_320
Finished <<< mycobot_280_x3pi [2.27s]
Starting >>> mycobot_320pi
Finished <<< mycobot_280jn [2.25s]
Starting >>> mycobot_600
Finished <<< mycobot_280pi [2.23s]
Starting >>> mycobot_630
Finished <<< mycobot_320 [2.25s]
Starting >>> mypalletizer_260
Finished <<< mycobot_320pi [2.25s]
Starting >>> mypalletizer_260_pi
Finished <<< mycobot_630 [1.86s]
Starting >>> ultraarm
Finished <<< mycobot_600 [2.30s]
Finished <<< mypalletizer_260_pi [2.20s]
Finished <<< mypalletizer_260 [2.24s]
Finished <<< ultraarm [1.47s]

Summary: 24 packages finished [15.4s]
aa@aa-virtual-machine:~/colcon_ws$

```

```

aa@aa-virtual-machine: ~/colcon_ws
Finished <<< mybuddy [2.39s]
Starting >>> mycobot_280pi
Finished <<< mycobot_280 [2.25s]
Starting >>> mycobot_320
Finished <<< mycobot_280_x3pi [2.28s]
Starting >>> mycobot_320pi
Finished <<< mycobot_280jn [2.35s]
Starting >>> mycobot_600
Finished <<< mycobot_280pi [2.34s]
Starting >>> mycobot_630
Finished <<< mycobot_320 [2.42s]
Starting >>> mypalletizer_260
Finished <<< mycobot_320pi [2.31s]
Starting >>> mypalletizer_260_pi
Finished <<< mycobot_600 [2.24s]
Starting >>> ultraarm
Finished <<< mycobot_630 [2.24s]
Finished <<< mypalletizer_260 [1.83s]
Finished <<< mypalletizer_260_pi [1.84s]
Finished <<< ultraarm [1.36s]

Summary: 24 packages finished [14.9s]
aa@aa-virtual-machine:~/colcon_ws$ source install/setup.bash
aa@aa-virtual-machine:~/colcon_ws$ ros2 launch myarm_m750 test.launch.py

```

After successfully running the launch file, the terminal will display:

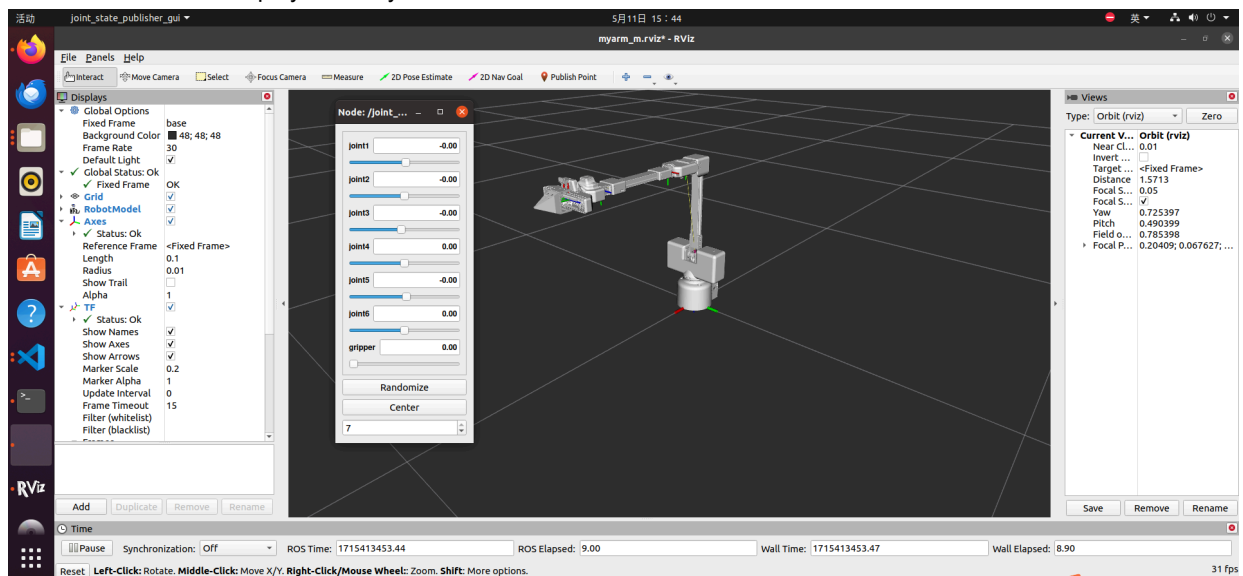
```

aa@aa-virtual-machine: ~/colcon_ws
aa@aa-virtual-machine: ~/colcon_ws

[robot_state_publisher-1] [INFO] [1716973870.917668270] [robot_state_publisher]:
got segment gripper_right
[robot_state_publisher-1] [INFO] [1716973870.917672904] [robot_state_publisher]:
got segment link1
[robot_state_publisher-1] [INFO] [1716973870.917676566] [robot_state_publisher]:
got segment link2
[robot_state_publisher-1] [INFO] [1716973870.917680134] [robot_state_publisher]:
got segment link3
[robot_state_publisher-1] [INFO] [1716973870.917683772] [robot_state_publisher]:
got segment link4
[robot_state_publisher-1] [INFO] [1716973870.917687429] [robot_state_publisher]:
got segment link5
[rviz2-3] [INFO] [1716973872.039407853] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-3] [INFO] [1716973872.039730397] [rviz2]: OpenGL version: 2.1 (GLSL 1.2)
[rviz2-3] [INFO] [1716973872.108269131] [rviz2]: Stereo is NOT SUPPORTED
[joint_state_publisher_gui-2] [INFO] [1716973872.204706370] [joint_state_publish
er]: Waiting for robot_description to be published on the robot_description topi
c...
[joint_state_publisher_gui-2] [INFO] [1716973872.228035706] [joint_state_publish
er]: Centering
[rviz2-3] Parsing robot urdf xml string.
[joint_state_publisher_gui-2] [INFO] [1716973872.695018233] [joint_state_publish
er]: Centering

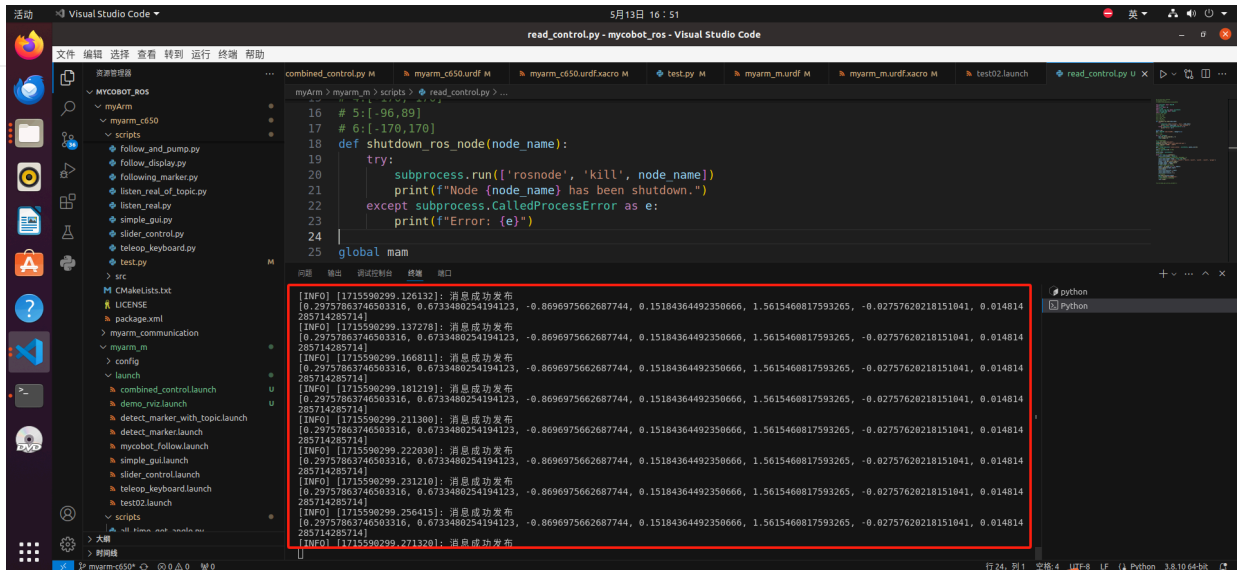
```

The rviz file will also display normally



Open the project in VS code and find the **read_control.py** file myArm/myarm_m750/myarm_m750/read_control.py

- Open the .py file, right-click, select "Run Python", and then click "Run Python file in the terminal"

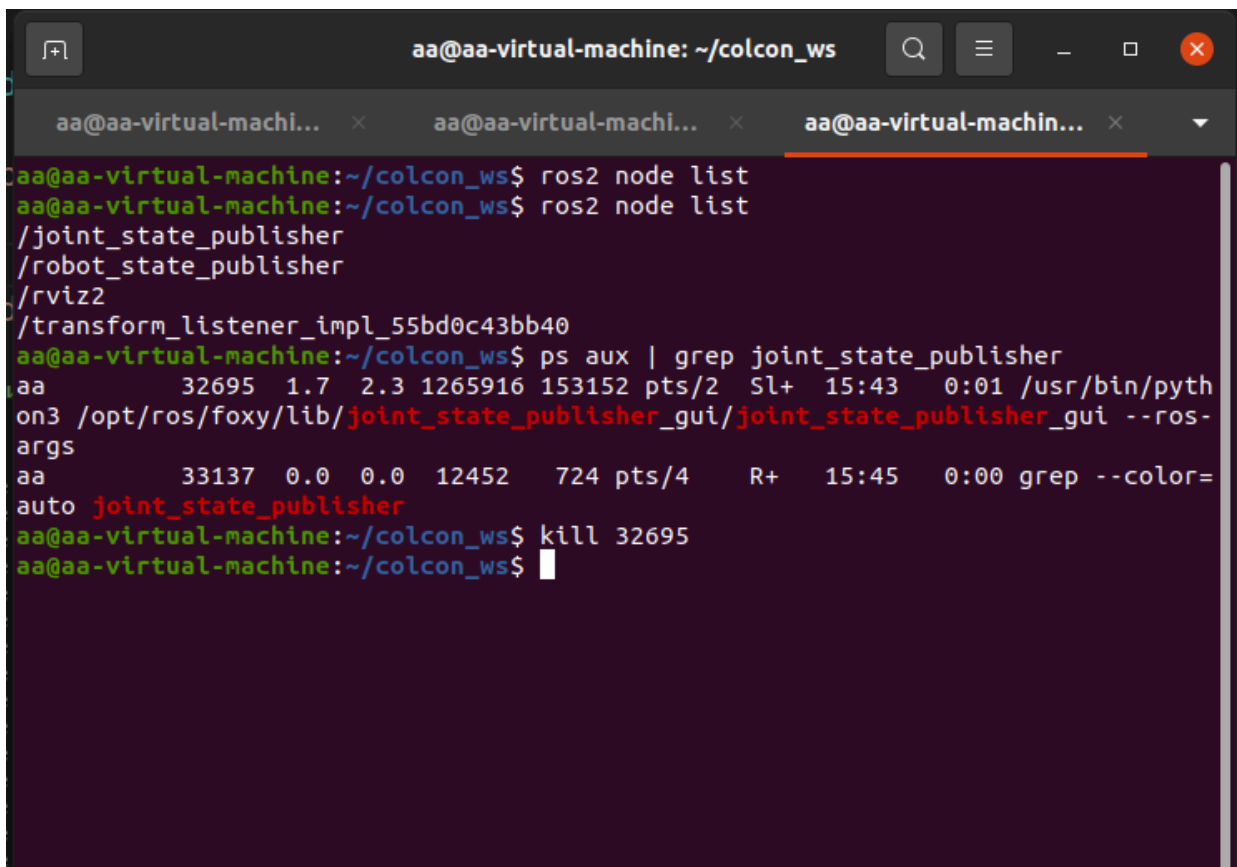


To create a new terminal in the workspace, enter the following command, first look at the running node, then look at the PID of the **joint_state_publisher** node, and finally kill the number string generated by this node:

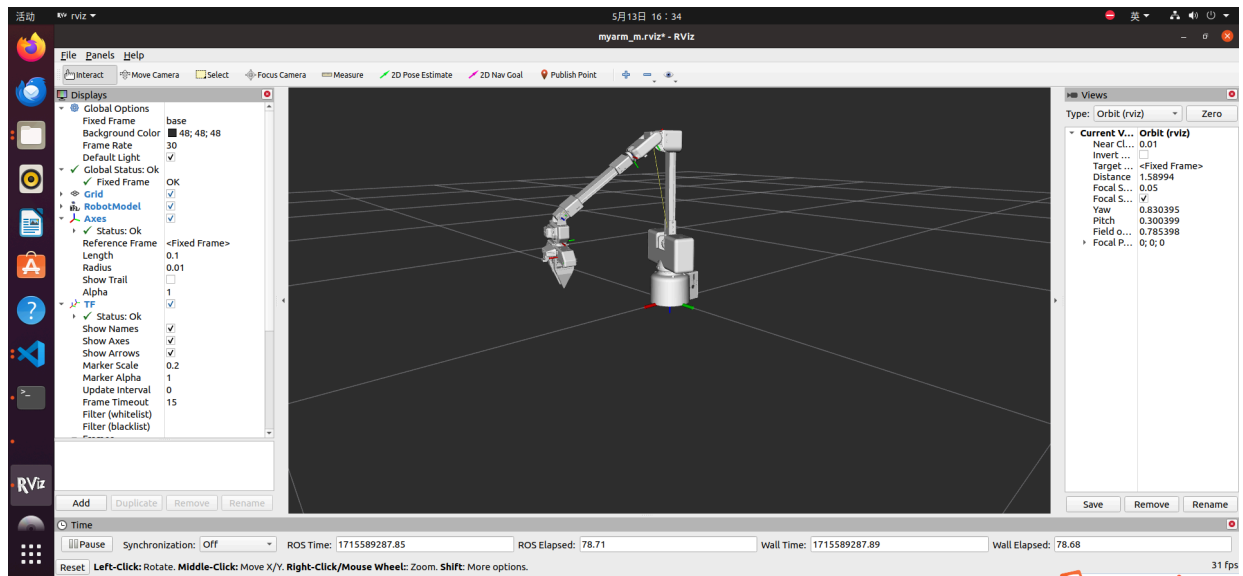
```
ros2 node list
```

```
ps aux | grep joint_state_publisher
```

```
kill 32695 #This string of numbers is generated at random
```



Next, we move the real-life MyarmM750 robotic arm with our hands, and the robotic arm in rviz will also follow:



At this point, the interaction with the MyarmM750 robotic arm has been completed

[← last page](#) | [next page →](#)

Basic tool

In this chapter, you will learn about the common command tools of ROS2.

1 Topics

ROS2 breaks complex systems down into many modular nodes. Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages. Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

Specific reference: [Official Tutorials](#)

- **topics help**

```
ros2 topics -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
```

```
ros2 run turtlesim turtle_teleop_key
```

- **Node Relationship Diagram**

```
rqt_graph
```

- **Learn about topic-related commands**

```
ros2 topics -h
```

- **topics list**

```
ros2 topic list
```

```
ros2 topic list -t # Display the corresponding message type
```

- **View topic content**

```
ros2 topic echo
```

```
ros2 topic echo /turtle1/cmd_vel
```

- **Display topic-related information, type**

```
ros2 topic info
```

```
# Output /turtle1/cmd_vel topic related information
```

```
ros2 topic info /turtle1/cmd_vel
```

- **Display interface related information**

```
ros2 interface show
```

```
# Output geometry_msgs/msg/Twist interface related information
```

```
ros2 interface show geometry_msgs/msg/Twist
```

- **Issue an order**

```
ros2 topic pub \ \ \
```

```
# Issue speed command
```

```
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

```
# Issue speed commands at a certain frequency
```

```
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 1.8}}"
```

2 Nodes

Each node in ROS should be responsible for a single, module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters. A full robotic system is comprised of many nodes working in concert. In ROS 2, a single executable (C++ program, Python program, etc.) can contain one or more nodes.

Specific reference: [Official Tutorials](#)

- **nodes help**

```
ros2 nodes -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
```

```
ros2 run turtlesim turtle_teleop_key
```

- **View the node list**

```
ros2 node list
```

- **View Node Relationship Diagram**

```
rqt_graph
```

- **Remapping**

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=my_turtle
```

```
ros2 node list
```

- **View node information**

```
ros2 node info \
```

```
ros2 node info /my_turtle
```

3 Services

Services are another method of communication for nodes in the ROS graph. Services are based on a call-and-response model, versus topics' publisher-subscriber model. While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.

Specific reference: [Official Tutorials](#)

- **services help**

```
ros2 service -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
```

```
ros2 run turtlesim turtle_teleop_key
```

- **View the service list**

```
ros2 service list
```

```
# Display service list and message type
```

```
ros2 service list -t
```

- **View the message types received by the service**

```
ros2 service type \
```

```
ros2 service type /clear
```

- **Find services that use a certain message type**

```
ros2 service find \
```

```
ros2 service find std_srvs/srv/Empty
```

- **View Service Message Type Definitions**

```
ros2 interface show \.srv
```

```
ros2 interface show std_srvs/srv/Empty.srv
```

- **Call the service command to clear the walking track**

```
ros2 service call \ \
```

```
ros2 service call /clear std_srvs/srv/Empty
```

- **Spawn a new turtle**

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle2'}"
```

4 Parameters

A parameter is a configuration value of a node. You can think of parameters as node settings. A node can store parameters as integers, floats, booleans, strings, and lists. In ROS 2, each node maintains its own parameters. For more background on parameters, please see the concept document.

Specific reference: [Official Tutorials](#)

- **parameters help**

```
ros2 param -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
```

```
ros2 run turtlesim turtle_teleop_key
```

- **View service list**

```
ros2 param list
```

- **Get the parameter value**

```
ros2 param get \ \
```

```
ros2 param get /turtlesim background_g
```

- **Set parameter values**

```
ros2 param set \ \ \
```

```
ros2 param set /turtlesim background_r 150
```

- **Export parameter values**

```
ros2 param dump \
```

```
ros2 param dump /turtlesim
```

- **Import parameters independently**

```
ros2 param load \ \
```

```
ros2 param load /turtlesim ./turtlesim.yaml
```

- **Start the node and import parameters at the same time**

```
ros2 run \ \ --ros-args --params-file \
```

```
ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```


5 Actions

Actions are one of the communication types in ROS 2 and are intended for long running tasks. They consist of three parts: a goal, feedback, and a result.

Actions are built on topics and services. Their functionality is similar to services, except actions are preemptable (you can cancel them while executing). They also provide steady feedback, as opposed to services which return a single response.

Actions use a client-server model, similar to the publisher-subscriber model (described in the topics tutorial). An “action client” node sends a goal to an “action server” node that acknowledges the goal and returns a stream of feedback and a result.

Specific reference: [Official Tutorials](#)

- **action help**

```
ros2 action -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
```

```
ros2 run turtlesim turtle_teleop_key
```

Press G|B|V|C|D|E|R|T to achieve rotation, press F to cancel

- **View the server and client of the node action**

```
ros2 node info /turtlesim
```

- **View action list**

```
ros2 action list
```

```
ros2 action list -t # show action type
```

- **view action info**

```
ros2 action info \
```

```
ros2 action info /turtle1/rotate_absolute
```

- **View action message content**

```
ros2 interface show turtlesim/action/RotateAbsolute
```

- **Send action target information**

```
ros2 action send_goal \ \
```

```
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
```

```
# With feedback information
```

```
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 0}" --feedback
```

6 RQt

RQt is a graphical user interface framework that implements various tools and interfaces in the form of plugins. One can run all the existing GUI tools as dockable windows within RQt! The tools can still run in a traditional standalone method, but RQt makes it easier to manage all the various windows in a single screen layout.

Specific reference: [Official Tutorials](#)

You can run any RQt tools/plugins easily by:

```
rqt
```

- **rqt help**

```
rqt -h
```

- **tart turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
```

```
ros2 run turtlesim turtle_teleop_key
```

- Action Type Browser: / Plugins -> Actions -> Action Type Browser
- parameter reconfiguration: / Plugins -> configuration -> Parameter Reconfigure
- Node graph: /Node Graph
- control steering: /Plugins -> Robot Tools -> Robot Steering
- service invocation: /Plugins -> Services -> Service Caller
- Service Type Browser: Plugins -> Services -> Service Type Browser
- message release: Plugins -> Topics -> Message Publisher
- Message Type Browser: Plugins -> Topics -> Message Type Browser
- topic list: Plugins -> Topics -> Topic Monitor
- draw a graph: Plugins -> Visualization -> Plot

- **View logs: rqt_console**

```
ros2 run rqt_console rqt_console
```

```
ros2 run turtlesim turtlesim_node
```

```
ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}"
```

7 TF2

tf2 is the transform library, which lets the user keep track of multiple coordinate frames over time. tf2 maintains the relationship between coordinate frames in a tree structure buffered in time and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time.

Specific reference: [Official Tutorials](#)

Let's start by installing the demo package and its dependencies.

```
sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools ros-foxy-tf-transformations
```

- **follow**
- **launch starts 2 little turtles, the first little turtle automatically follows the second one**

```
ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

- **Control the movement of the first little turtle through the keyboard**

```
ros2 run turtlesim turtle_teleop_key
```

- **View TF tree**

```
ros2 run tf2_tools view_frames.py
```

```
evince frames.pdf
```

- **View the relationship between two coordinate systems**

```
ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]
```

```
ros2 run tf2_ros tf2_echo turtle2 turtle1
```

- **View TF relationships on rviz**

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz
```

8、 Introduction to the URDF

- Unified Robot Description Format, The Unified Robot Description Format, or URDF for short. The urdf feature pack in ROS contains a C++ parser for URDF, and URDF files describe the robot model in XML format.
- URDF cannot be used on its own and needs to be combined with Rviz2 or Gazebo, URDF is just a file that needs to be rendered as a graphical robot model in Rviz2 or Gazebo.

8.1 urdf file description

Code examples:

Only some of the codes are screenshotted for display:

```
<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>
</link>
```

```
<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>
```

- **Install dependent libraries**

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

```
sudo apt install ros-foxy-xacro
```

- **Download the source code**

```
cd ~/dev_ws
```

```
git clone -b ros2 https://github.com/ros/urdf\_tutorial.git src/urdf_tutorial
```

- **Compiling the source code**

```
colcon build --packages-select urdf_tutorial
```

- **Running the example**

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf
```

[← last page](#) | [next page →](#)

correspondence

Our MyarmM750 mainly uses a **Topic** method for communication

Topic is one of the most commonly used communication mechanisms in ROS, which is based on the publisher-subscriber model, in which one node publishes messages as publishers and other nodes receive messages as subscribers. A publisher can publish messages to multiple subscribers at the same time, and subscribers can receive messages from multiple publishers. This approach is ideal for situations that require real-time data updates, such as processing and real-time control of sensor data.

Start by creating a Publisher in the .py file to publish our message to the MyarmM750

```

12 import subprocess
13
14 class JointStatePublisher(Node):
15     def __init__(self):
16         super().__init__("pub_data")
17         self.pub = self.create_publisher(JointState, '/joint_states', 10)
18         self.timer = self.create_timer(0.02, self.publish_joint_states) # 50Hz
19         self.mam = MyArmM('/dev/ttyACM0', debug=False)
20
21         # 放松关节
22         for i in range(8):
23             self.mam.set_servo_enabled(i, 0)
24             sleep(0.2)
25
26         # 尝试关闭节点
27         self.shutdown_ros_node('joint_state_publisher_gui')
28         self.get_logger().info("已成功杀死节点")
29
30     def shutdown_ros_node(self, node_name):
31         try:
32             subprocess.run(['ros2', 'node', 'kill', node_name])
33             self.get_logger().info(f"Node {node_name} has been shutdown.")
34         except subprocess.CalledProcessError as e:
35             self.get_logger().error(f"Error: {e}")
36
37     def publish_joint_states(self):
38         joint_state = JointState()
39         joint_state.header = Header()
40         joint_state.header.stamp = self.get_clock().now().to_msg()
41         joint_state.name = ['joint1', 'joint2', 'joint3', 'joint4', 'joint5', 'joint6', 'gripper']
42         angles = self.mam.get_joints_angle()
43         gripper_angle = angles.pop(6)
44         gripper_angle /= -3500
45         # angle: 弧度 angles: 角度

```

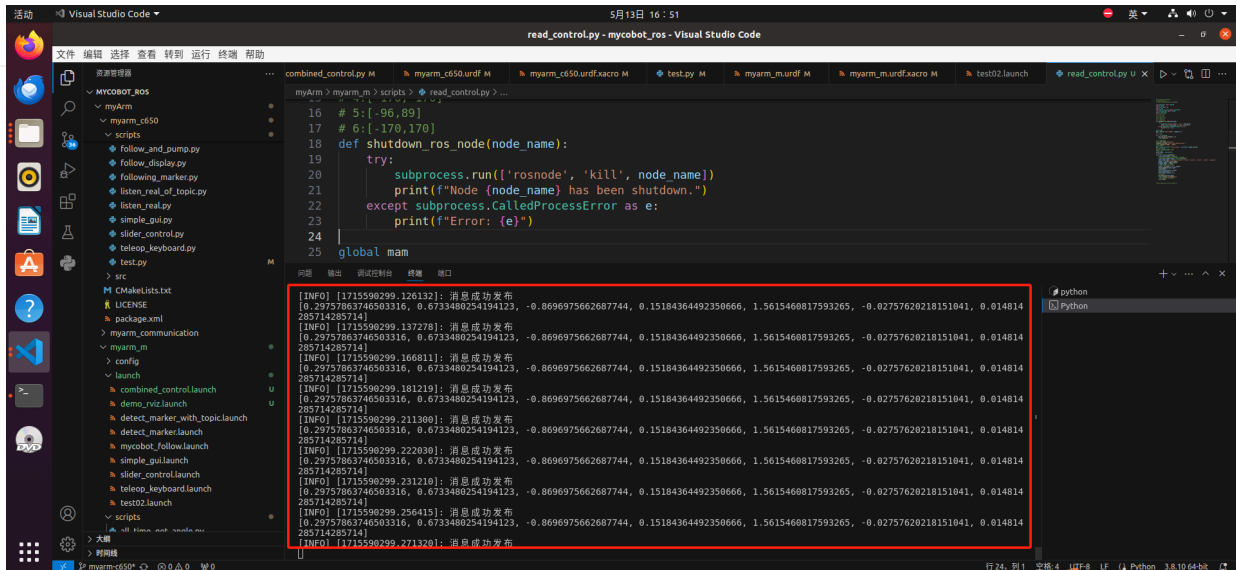
Next, open the terminal in the workspace and start ROS2:

```
colcon build
```

```
source install/setup.bash
```

```
ros2 launch myarm_m750 test.launch.py
```

Open RVIZ and then launch our **read_control.py** file

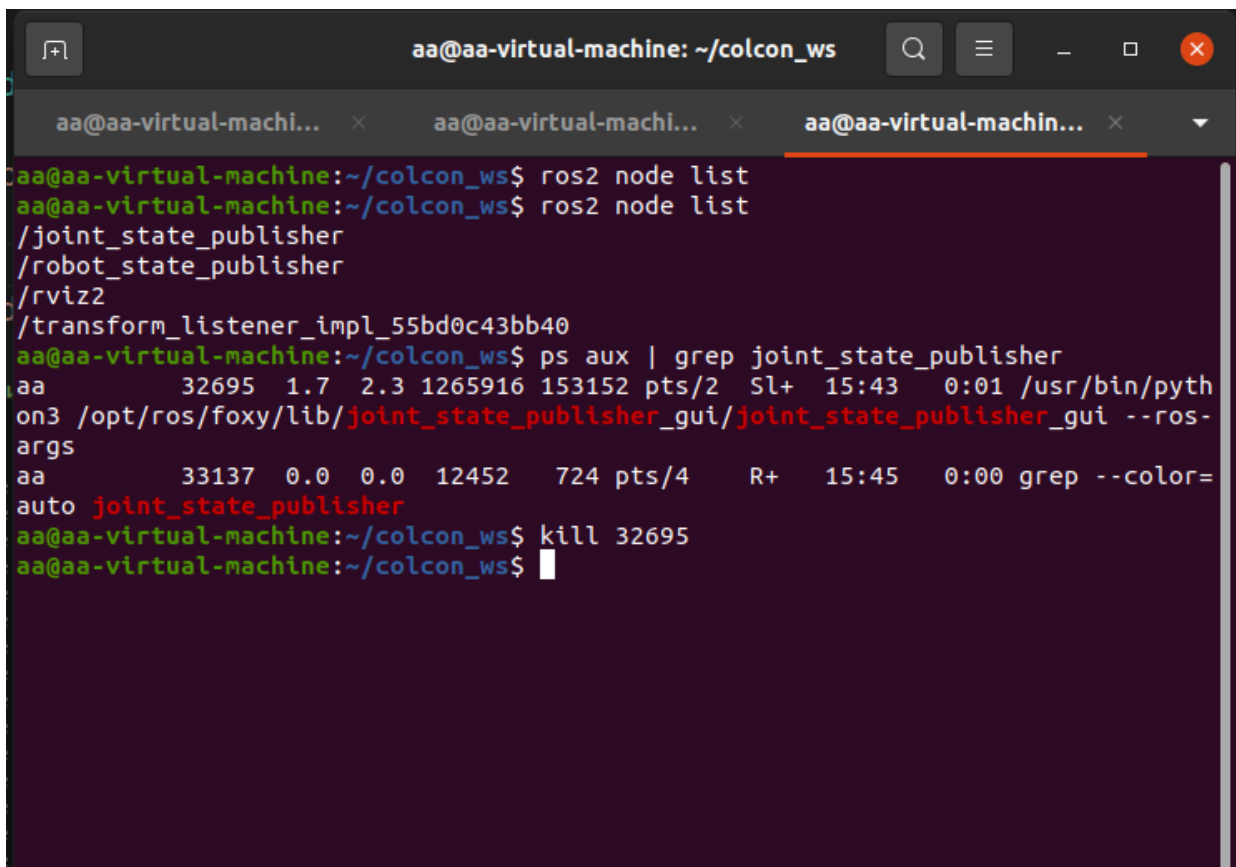


To create a new terminal in the workspace, enter the following command, first look at the running node, then look at the PID of the `joint_state_publisher` node, and finally kill the number string generated by this node:

```
ros2 node list
```

```
ps aux | grep joint_state_publisher
```

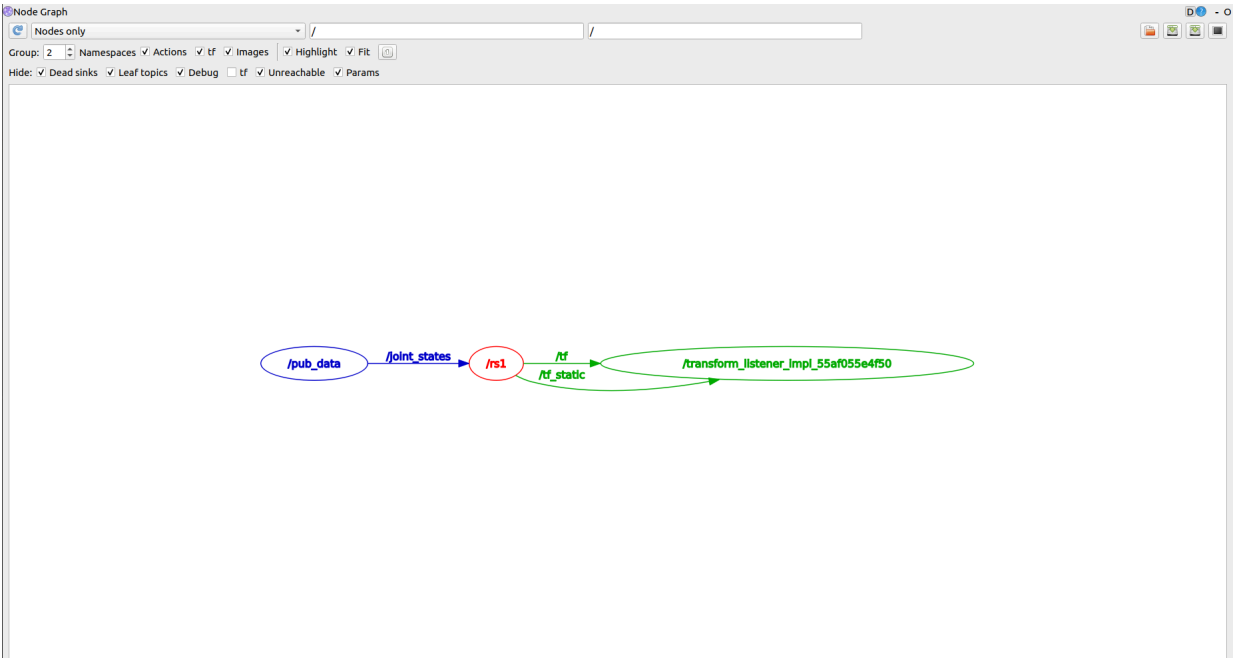
```
kill 32695 #This string of numbers is generated at random
```



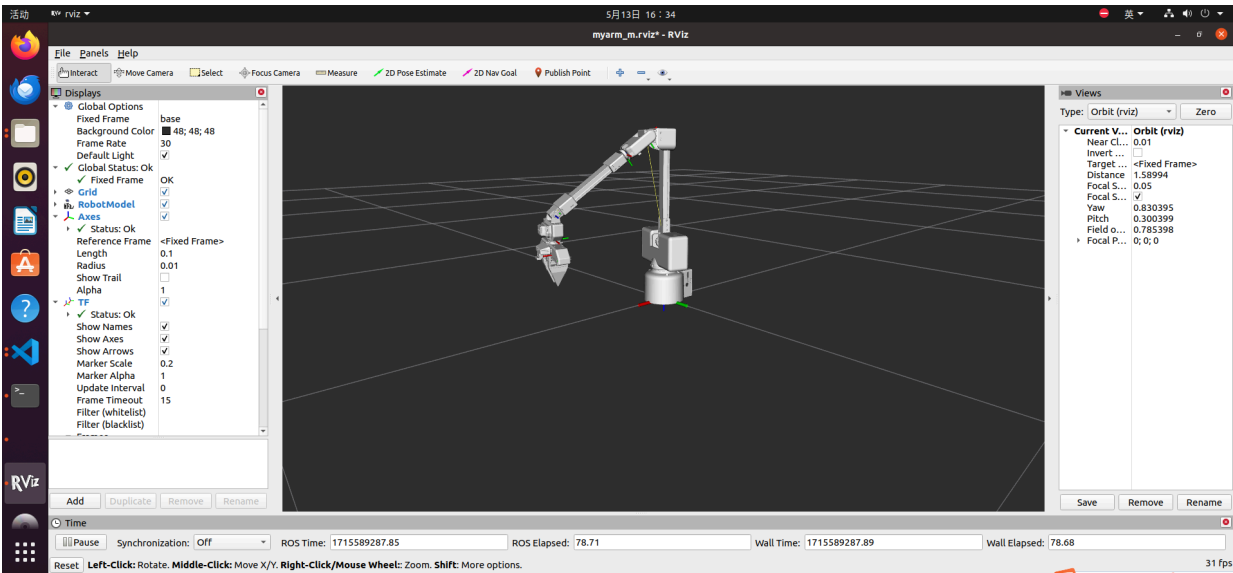
Finally, we open a new terminal and enter:

```
rqt_graph
```

We can see all the information about the node



The MyarmM750 is also in a state where it can be manually moved



Communication and Information Commands

Note: Before communicating directly via the communication protocol, you need to flash "Transponder" on M5Stack-basic and flash the latest atomMain on Atom.

Please read the interface protocol documentation carefully and continue with program development after testing.

1 USB Communication Settings

Make sure you have the following settings ready:

- Mainline Interface: USB Type-C to connect to a computer and basic
- Baud rate: 115200
- Data bits: 8
- Parity bit: none
- Stop bits: 1

2 Introduction to Instruction Frames & Guidance

The main PC transmits data to the peripheral PC via M5Stackc-basic. The peripheral PC decodes the data into a command with a return value, and then sends the results back to the primary PC within 500 milliseconds.

3 Format for sending and receiving information commands

Both sending and receiving should be expressed in hexadecimal. Each command should contain 5 sections as follows. Sections 3 and 4 can be left blank.

- 1 Command pin: 0xFE 0xFE
 - No change -indispensable
- 2 Effective Length:
 - The overall length includes pins, serial numbers, function codes, and endpoints -indispensable
- 3 Serial number: 00 ~ 8F
 - The corresponding number of developed instructions
 - Can be left blank.
- 4 Feature Codes
 - Be goal-oriented
 - Can be left blank.
- 5 End: 0xFA
 - No change -indispensable

4 Command Explanation

The main PC transmits data to the peripheral PC via M5Stackc-basic. The peripheral PC decodes the data into a command with a return value, and then sends the results back to the primary PC within 500 milliseconds.

Type	DATA	Length	FEATURES
Directive Framework	Start bits: 0	1	Start frame recognition, 0XFE
	Start bit: 1	1	Start frame recognition, 0XFE
	Bit data length	1	Different commands correspond to different data lengths
	Instruction bits	1	Depends on different command
Directive Framework	Data	0-16	Commands vs. data, depending on the command
End frame	End Bit	1	Stop bit, 0XFA

5 Single Command Command Description

Power on

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Instruction frame	0X10
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 10 FA

There is no return value

Power drops and connections are interrupted

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Instruction frame	0X11
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 11 FA

There is no return value

Check the status of Atom

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Instruction frame	0X12
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 12 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return value: Recognition frame	0X03
Data[3]	Return value: Command frame	0X12
Data[4]	Power on/off	0X01/0X00
Data[5]	End frame	0XFA

Example:

Assuming the Atom is plugged in:

Port returns: FE FE 03 12 01 FA

Joint enable

Data Field	Description	Data
Data[0]	Identification Frame	0XFE
Data[1]	Identification Frame	0XFE
Data[2]	Data Length Frame	0X04
Data[3]	Command Frame	0X13
Data[4]	Servo Serial Number	1~6/254
Data[5]	On/Off	0X01/0X00
Data[6]	End Frame	0XFA

示例:

串口传输: FE FE 02 13 FA

无返回值

Get bot error status

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X15
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 15 FA

Return Value: Data structure

Clear the bot exception

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X16
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 16 FA

The total length of the queue for read and receive instructions

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X17
Data[4]	End frame	0XFA

Example:

Port returns: FE FE 03 17 03 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Command frame	0X17
Data[4]	Total length of the received command queue	0X03
Data[5]	End frame	0XFA

Set the total length of the receiving command queue

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0X18
Data[4]	Total queue length	0X10
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 18 10 FA

Clear the queue for receiving commands

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X19
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 19 FA

Read the current length of the receive queue

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X08
Data[4]	End frame	0XFA

Example:

Port returns: FE FE 03 08 03 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Command frame	0X08
Data[4]	Total length of the received command queue	0X03
Data[5]	End frame	0XFA

Reading Single Joint Angle (Reading Movement Information)

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0X1c
Data[4]	Joint serial number	0X01
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 1c 01 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return Value: Start frame	0XFE
Data[1]	Return Value: Start frame	0XFE
Data[2]	Return Value: Data Length Frame	0X05
Data[3]	Return value: Command frame	0X1c
Data[4]	Serial number	0X01
Data[5]	Servo angle high	Angle_high
Data[6]	Servo angle low	Angle_low
Data[7]	End frame	0XFA

Example:

Return value of port: FE FE 05 1c 01 00 8C FA

temp = angle_low+angle_high*256

Angle= (temp 33000 ? (temp – 65536) : temp) /100

Explanation: If temp is greater than 33000, temp subtracts 65536 and then divides by 100, and if temp is less than 33000, temp is divided by 100.

Reading Angle (Reading Movement Information)

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X20
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 20 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return Value: Start frame	0XFE
Data[1]	Return Value: Start frame	0XFE
Data[2]	Return Value: Data Length Frame	0X0A
Data[3]	Return value: Command frame	0X20
Data[4]	Servo 1 angle high	Angle1_high
Data[5]	Servo 1 angle low	Angle1_low
Data[6]	Servo No. 2 angle high	Angle2_high
Data[7]	Servo No. 2 angle low	Angle2_low
Data[8]	Servo No. 3 angle high	Angle3_high
Data[9]	Servo 3 angle low	Angle3_low
Data[10]	Servo 4 angle high	Angle4_high
Data[11]	Servo 4 angle low	Angle4_low
Data[12]	Servo No. 5 angle high	Angle5_high
Data[13]	No. 5 servo angle low	Angle5_low
Data[14]	Servo No. 6 angle high	Angle6_high
Data[15]	Servo No. 6 angle low	Angle6_low
Data[16]	No. 7 servo angle high	Angle7_high
Data[17]	Servo No. 7 angle low	Angle7_low
Data[18]	End frame	0XFA

Example:

Return value of port: FE FE 0E 20 00 8C 00 3D FF E6 FF 3F 00 AF FF 51 00 3B 00 4F FA

How to get joint angle 1:

$\text{temp} = \text{angle1_low} + \text{angle1_high} * 256$

$\text{Angle1} = (\text{temp} > 33000 ? (\text{temp} - 65536) : \text{temp}) / 100$

Explanation: If temp is greater than 33000, temp subtracts 65536 and then divides by 100, and if temp is less than 33000, temp is divided by 100.

Other joint angles are calculated in a similar way.

Send angles

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X06
Data[3]	Command frame	0X21
Data[4]	Serial number	joint_no
Data[5]	High Angle	angle_high
Data[6]	Low angle	angle_low
Data[7]	Specify the speed	sp
Data[8]	End frame	0XFA

Example:

Move the servo 1 to zero

Serial transmission: FE FE 06 21 01 00 00 14 FA

Joint No.: 1-7

angle_high: byte

Calculation: Multiply the angular value by 100 and convert to integral form to get a hexadecimal high byte

angle_low: byte

Calculation: Multiply the angle value by 100 and convert to integral form to get the hexadecimal low byte

There is no return value

Send all angles

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X11
Data[3]	Command frame	0X22
Data[4]	Servo 1 angle high	Angle1_high
Data[5]	Servo 1 angle low	Angle1_low
Data[6]	Servo No. 2 angle high	Angle2_high
Data[7]	Servo No. 2 angle low	Angle2_low
Data[8]	Servo No. 3 angle high	Angle3_high
Data[9]	Servo 3 angle low	Angle3_low
Data[10]	Servo 4 angle high	Angle4_high
Data[11]	Servo 4 angle low	Angle4_low
Data[12]	Servo No. 5 angle high	Angle5_high
Data[13]	No. 5 servo angle low	Angle5_low
Data[14]	Servo No. 6 angle high	Angle6_high
Data[15]	Servo No. 6 angle low	Angle6_low
Data[16]	No. 7 servo angle high	Angle7_high
Data[17]	Servo No. 7 angle low	Angle7_low
Data[18]	Specify the speed	Sp
Data[19]	End frame	0XFA

Example:

Send an angle of 0 to the entire joint to move the servo to zero.

Serial transmission: FE FE 0F 22 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1E FA

angle1_high: byte

Calculation: Multiply the angular value by 100 and convert to integral form to get a hexadecimal high byte

angle1_low: byte

Calculation: Multiply the angle value by 100 and convert to integral form to get the hexadecimal low byte

Other angles are calculated in a similar way. _

no returns a value

The robot stops moving

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X29
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 26 FA

There is no return value

Mobile checks

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X2B
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 2B FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return Value: Header	0XFE
Data[1]	Return Value: Header	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Return value: Command frame	0X2B
Data[4]	Exercise/No Exercise	0X01/0X00
Data[5]	End frame	0XFA

Example:

Let's say the program is running:

port Return: FE FE 03 2B 01 FA

Send potential value

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X05
Data[3]	Command frame	0X3A
Data[4]	Serial number	Joint
Data[5]	The potential value is high	Encoder_high
Data[6]	The potential value is low	Encoder_low
Data[7]	End frame	0XFA

Example:

Set joint number 5 to potential 2048

Serial transmission: FE FE 05 3A 05 08 00 FA

Joint serial numbers from 0 to 5

Data type of Joint: byte

Encoder_high data type: byte

Calculation method: Get the high position of the potential value (hexadecimal form)

Encoder_low data type: byte

Calculation method: Get the low bit of the potential value (hexadecimal form)

There is no return value

Get the potential value

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0X3B
Data[4]	Joint serial number	joint
Data[5]	End frame	0XFA

Example:

Get the potential of the NO.2 servos

joint_no Value range: 1 to 7

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0X3B
Data[4]	The rudder potential value is high	Encoder_high
Data[5]	The rudder potential value is low	Encoders_low
Data[6]	End frame	0XFA

Example:

Serial transmission: FE FE 04 3B 08 07 FA

How to calculate the potential value:

Potential value = potential value low + potential value high * 256

23 servo direction is opposite, when it is not 2048 zero, you need to say No. 2: 1800; Then number 3 needs to be 2296

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X13
Data[3]	Command frame	0X3C
Data[4]	The high byte	. of servo 1 encoder_1_high
Data[5]	The low byte	of servo 1 encoder_1_low
Data[6]	The high byte	of the 2nd servo encoder_2_high
Data[7]	The low byte	of servo 2 encoder_2_low
Data[8]	The high byte	of the servo No. 3 encoder_3_high
Data[9]	The low byte	of servo 3 encoder_3_low
Data[10]	The high byte	of servo 4 encoder_4_high
Data[11]	The low byte	of the 4 servos encoder_4_low
Data[12]	The high byte	of servo 5 encoder_5_high
Data[13]	The low byte	of servo 5 encoder_5_low
Data[14]	The high byte	of the 6 servos encoder_6_high
Data[15]	The low byte	of the 6 servos encoder_6_low
Data[16]	The high byte	of the 7 servos encoder_7_high
Data[17]	The low byte	of the 7 servos encoder_7_low
Data[18]	The high byte	of the 8 servos encoder_8_high
Data[19]	The low byte	of the 8 servos encoder_8_low
Data[20]	Specify the speed	Sp
Data[21]	End frame	0XFA

Example:

The potential value of all the motors sent is 2048 and the speed is 20

Serial transmission: FE FE 13 3C 08 00 08 00 08 00 08 00 08 00 08 00 08 00 08 00 14 FA

Refer to the individual potential values sent above

encoder_1_high data type: byte

Calculation: Convert the potential of servo 1 into an integral form to obtain a high-byte hexadecimal number

encoder_1_low data type: byte

Calculation: Convert the potential of servo 1 into an integral form to obtain a low-byte hexadecimal number

sp: Byte data type, ranging from 0 to 100

There is no return value

Send the potential values of the seven servos

2\3 servo direction is opposite, when it is not 2048 zero, you need to say No. 2: 1800; Then number 3 needs to be 2296

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X22
Data[3]	Command frame	0X3E
Data[4]	The high byte	. of servo 1 encoder_1_high
Data[5]	The low byte	of servo 1 encoder_1_low
Data[6]	The high byte	of the 2nd servo encoder_2_high
Data[7]	The low byte	of servo 2 encoder_2_low
Data[8]	The high byte	of the servo No. 3 encoder_3_high
Data[9]	The low byte	of servo 3 encoder_3_low
Data[10]	The high byte	of servo 4 encoder_4_high
Data[11]	The low byte	of the 4 servos encoder_4_low
Data[12]	The high byte	of servo 5 encoder_5_high
Data[13]	The low byte	of servo 5 encoder_5_low
Data[14]	The high byte	of the 6 servos encoder_6_high
Data[15]	The low byte	of the 6 servos encoder_6_low
Data[16]	The high byte	of the 7 servos encoder_7_high
Data[17]	The low byte	of the 7 servos encoder_7_low
Data[18]	The high byte	of the 8 servos encoder_8_high
Data[19]	The low byte	of the 8 servos encoder_8_low
Data[20]	Servo No. 1 specifies speed	Sp
Data[21]	Designated speed of servo No. 2	Sp
Data[22]	No. 3 servo specified speed	Sp
Data[23]	Specified speed of servo 4	Sp
Data[24]	No. 5 servo specified speed	Sp
Data[25]	No. 6 servo specified speed	Sp
Data[26]	No. 7 servo specified speed	Sp
Data[27]	Servo No. 8 specified speed	Sp
Data[28]	End frame	0XFA

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X12
Data[3]	Command frame	0X3D
Data[4]	The high byte	of the bit value of the 1 rudder machine encoder_1_high
Data[5]	The low byte of the bit value of the 1 rudder	encoder_1_low
Data[6]	The high byte	of the potential value of the 2 rudder machine encoder_2_high
Data[7]	The low byte of the bit value of the 2 rudder	encoder_2_low
Data[8]	The high byte of the bit value of the 3rd rudder	encoder_3_high
Data[9]	The low byte of the bit value of the 3rd rudder	encoder_3_low
Data[10]	The high byte	of the rudder 4 potential value encoder_4_high
Data[11]	The low byte	of the bit value of the 4 rudder machine encoder_4_low
Data[12]	The high byte	of the bit value of the 5 rudder machine encoder_5_high
Data[13]	The low byte of the 5 rudder machine bit value	encoder_5_low
Data[14]	The high byte	of the 6 rudder electromechanical potential value encoder_6_high
Data[15]	The low byte of the 6 rudder electromechanical bit value	encoder_6_low
Data[16]	The high byte	of servo 5 encoder_7_high
Data[17]	The low byte	of servo 5 encoder_7_low
Data[18]	The high byte	of the 6 servos encoder_8_high
Data[19]	The low byte	of the 6 servos encoder_8_low
Data[20]	End frame	0XFA

Example:

Suppose that all joints of the robotic arm are currently in the 0 position

Serial port return value: FE FE 12 3D 08 00 08 00 08 00 08 00 08 00 08 00 FA

How to calculate the potential value

Potential value = potential value low + potential value high * 256

Read speed

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0XE1
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 E1 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0XE1
Data[4]	specified speed	Sp
Data[5]	End frame	0XFA

Example:

Let's say the speed is 50

port return value: FE FE 03 E1 32 FA

Read the minimum angle of all joints

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X4A
Data[4]	End frame	0XFA

Example:

Serial transmission: FE FE 02 4A FA

joint_no Value range: 1 to 7

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X0A
Data[3]	Return value: Command frame	0X4A
Data[4]	Joint 1 angle value high	Angle_high
Data[5]	The angle value of joint 1 is low	Angle_low
Data[6]	Joint 2 angle value high	Angle_high
Data[7]	Joint 2 angle value low	Angle_low
Data[8]	Joint 3 angle value high	Angle_high
Data[9]	Joint 3 angle value low	Angle_low
Data[10]	Joint 4 angle value high	Angle_high
Data[11]	Joint 4 angle value low	Angle_low
Data[12]	Joint 5 angle value high	Angle_high
Data[13]	Joint 5 angle value low	Angle_low
Data[14]	Joint 6 angle value high	Angle_high
Data[15]	Joint 6 angle value low	Angle_low
Data[16]	Joint 7 angle value high	Angle_high
Data[17]	Joint 7 angle value low	Angle_low
Data[18]	End frame	0XFA

Example:

port return value: FE FE 0A 4A F9 F2 F9 F2 F9 F2 F9 F2 F2 F9 F2 F9 F2 F9 F2 FA

How to arrive at the minimum angle of the joint

temp = angle1_low+angle1_high*256

Angle1= (temp 33000 ? (temp – 65536) : temp) /10

Calculation method: angle value low + angle high value multiplied by 256 first determine whether it is greater than 33000 if it is greater than 33000 then subtract 65536 and finally divide by 10 if it is less than 33000 directly divide by 10

Read the maximum angle of all joints

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X4B
Data[4]	End frame	0XFA

joint_no Value range: 1 to 7

Example:

Serial transmission: FE FE 02 4B FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X0A
Data[3]	Return value: Command frame	0X4B
Data[4]	Joint 1 angle value high	Angle_high
Data[5]	The angle value of joint 1 is low	Angle_low
Data[6]	Joint 2 angle value high	Angle_high
Data[7]	Joint 2 angle value low	Angle_low
Data[8]	Joint 3 angle value high	Angle_high
Data[9]	Joint 3 angle value low	Angle_low
Data[10]	Joint 4 angle value high	Angle_high
Data[11]	Joint 4 angle value low	Angle_low
Data[12]	Joint 5 angle value high	Angle_high
Data[13]	Joint 5 angle value low	Angle_low
Data[14]	Joint 6 angle value high	Angle_high
Data[15]	Joint 6 angle value low	Angle_low
Data[16]	Joint 7 angle value high	Angle_high
Data[17]	Joint 7 angle value low	Angle_low
Data[18]	End frame	0XFA

Example:

port return value: FE FE 0A 4B F9 F2 F9 F2 F9 F2 F9 F2 F9 F2 F9 F2 F2 F2 F2 F2 F2 FA

How to arrive at the maximum angle of the joint

temp = angle1_low+angle1_high*256

Angle1= (temp 33000 ? (temp – 65536) : temp) /10

Calculation method: angle value low + angle high value multiplied by 256 first determine whether it is greater than 33000 if it is greater than 33000 then subtract 65536 and finally divide by 10 if it is less than 33000 directly divide by 10

Sets the maximum angle of the joint

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X05
Data[3]	Command frame	0X4D
Data[4]	Joint servo serial number	Joint_number
Data[5]	highbyte of steering gear	Angle_high
Data[6]	lowbyte of steering gear	Angle_low
Data[7]	End frame	0XFA

Example:

Set the maximum angle of joint 2 to 45

joint_no Value range: 1 to 7

port return value: FE FE 03 41 32 FA

angle1_high: The data type is byte

Calculation: The servo angle value is multiplied by 100 and converted to int first, and then the hexadecimal high byte is taken

angle1_low: The data type is byte

Calculation: The servo angle value is multiplied by 100 and converted to int first, and then the hexadecimal low byte is taken

Serial transmission: FE FE 05 4C 02 11 94 FA

There is no return value

Check if all servos are powered on

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X51
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 51 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Return value: Command frame	0X51
Data[4]	power on/off	0X01/0X00
Data[5]	End frame	0XFA

Example:

All servos are powered on

Serial port return value: FE FE 03 51 01 FA

Get the temperature of all servo motors

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0XE5
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 E5 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X0A
Data[3]	Return value: Command frame	0XE5
Data[4]	Motor 1 temperature	servo1_temp
Data[5]	Motor 2 temperature	servo2_temp
Data[6]	Motor 3 temperature	servo3_temp
Data[7]	Motor 4 temperature	servo4_temp
Data[8]	Motor 5 temperature	servo5_temp
Data[9]	Motor 6 temperature	servo6_temp
Data[10]	Motor 7 temperature	servo7_temp
Data[11]	Motor 8 temperature	servo8_temp
Data[12]	End frame	0XFA

Example:

Serial port return value: FE FE 0A E5 14 14 14 14 14 14 14 14 FA

Get the voltage of all servo motors

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0XE3
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 E3 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X0A
Data[3]	Return value: Command frame	0XE3
Data[4]	Motor 1 voltage	servo1_vol
Data[5]	Motor 2 voltage	servo2_vol
Data[6]	Motor 3 voltage	servo3_vol
Data[7]	Motor 4 voltage	servo4_vol
Data[8]	Motor 5 voltage	servo5_vol
Data[9]	Motor 6 voltage	servo6_vol
Data[10]	Motor 7 voltage	servo7_vol
Data[11]	Motor 8 voltage	servo8_vol
Data[12]	End frame	0XFA

Example:

Serial port return value: FE FE 0A E5 00 00 00 00 00 00 00 00 00 FA

Get the current of all servo motors

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0XE2
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 E2 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X12
Data[3]	Return value: Command frame	0XE2
Data[4]	Motor 1 temperature	servo1_current_high
Data[5]	Motor 1 temperature	servo1_current_low
Data[6]	Motor 2 temperature	servo2_current_high
Data[7]	Motor 2 temperature	servo2_current_low
Data[8]	Motor 3 temperature	servo3_current_high
Data[9]	Motor 3 temperature	servo3_current_low
Data[10]	Motor 4 temperature	servo4_current_high
Data[11]	Motor 4 temperature	servo4_current_low
Data[12]	Motor 5 temperature	servo5_current_high
Data[13]	Motor 5 temperature	servo5_current_low
Data[14]	Motor 6 temperature	servo6_current_high
Data[15]	Motor 6 temperature	servo6_current_low
Data[16]	Motor 7 temperature	servo7_current_high
Data[17]	Motor 7 temperature	servo7_current_low
Data[18]	Motor 8 temperature	servo8_current_high
Data[19]	Motor 8 temperature	servo8_current_low
Data[20]	End frame	0XFA

Example:

[illegible]

Get all statuses of all servo motors

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0XE4
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 E3 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X0A
Data[3]	Return value: Command frame	0XE4
Data[4]	Motor 1 voltage	servo1_status
Data[5]	Motor 2 voltage	servo2_status
Data[6]	Motor 3 voltage	servo3_status
Data[7]	Motor 4 voltage	servo4_status
Data[8]	Motor 5 voltage	servo5_status
Data[9]	Motor 6 voltage	servo6_status
Data[10]	Motor 7 voltage	servo7_status
Data[11]	Motor 8 voltage	servo8_status
Data[12]	End frame	0XFA

Example:

Serial port return value: FE FE 0A E4 00 00 00 00 00 00 00 00 00 FA

Get all servo motor protection currents

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0XE6
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 02 E6 FA

Return Value: Data structure

Set the motor enabled

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X13
Data[4]	Serial number	servo_no
Data[6]	Enable/Disable Enable	0X01/0X00
Data[6]	End frame	0XFA

Example:

Enable the 1 servo

Serial transmission: FE FE 04 13 01 01 FA

servo_no Value range: 1-8

There is no return value

Set the servo zero point

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0X54
Data[4]	Joint servo serial number	servo_no
Data[5]	End frame	0XFA

Example:

Set the zero position of servo 1

Serial transmission: FE FE 03 54 01 FA

servo_no Value range: 1-8

There is no return value

Set the position loop P scale factor for the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X70
Data[4]	Serial number	servo_no
Data[5]	Serial number	data
Data[6]	End frame	0XFA

Example:

Serial transmission: FE FE 04 70 01 01 FA

servo_no Value range: 1-8

There is no return value

Set the proportional factor of the position ring D of the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X71
Data[4]	Serial number	servo_no
Data[5]	Serial number	data
Data[6]	End frame	0XFA

Example:

Serial transmission: FE FE 04 71 01 01 FA

servo_no Value range: 1-8

There is no return value

Set the ratio factor for the position ring I of the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X72
Data[4]	Serial number	servo_no
Data[5]	Serial number	data
Data[6]	End frame	0XFA

Example:

Serial transmission: FE FE 04 72 01 01 FA

servo_no Value range: 1-8

There is no return value

Reads the position loop P scale factor of the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0XE7
Data[4]	Motor serial number	servo_no
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 E7 01 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0XE7
Data[4]	Motor serial number	servo_no
Data[5]	Position loop P scale factor	P
Data[6]	End frame	0XFA

Example:

Serial port return value: FE FE 04 E7 01 01 FA

Reads the position loop D scale factor for the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0XE8
Data[4]	Motor serial number	servo_no
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 E8 01 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0XE8
Data[4]	Motor serial number	servo_no
Data[5]	Position loop P scale factor	D
Data[6]	End frame	0XFA

Example:

Serial port return value: FE FE 04 E8 01 01 FA

Reads the position loop I scale factor of the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0XE9
Data[4]	Motor serial number	servo_no
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 E9 01 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0XE7
Data[4]	Motor serial number	servo_no
Data[5]	Position loop P scale factor	1
Data[6]	End frame	0XFA

Example:

Serial port return value: FE FE 04 E9 01 01 FA

Sets the clockwise insensitivity zone of the encoder for the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X73
Data[4]	Serial number	servo_no
Data[5]	Clockwise insensitive zone value	data
Data[6]	End frame	0XFA

Example:

Serial transmission: FE FE 04 73 01 01 FA

servo_no Value range: 1-8 The value range of data is 0-32

There is no return value

Reads the clockwise insensitive area of the encoder for the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0XEA
Data[4]	Motor serial number	servo_no
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 E7 01 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0XEA
Data[4]	Motor serial number	servo_no
Data[5]	Clockwise insensitive zone value	data
Data[6]	End frame	0XFA

Example:

Serial port return value: FE FE 04 EA 01 01 FA

Sets the counterclockwise insensitivity zone of the encoder for the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X74
Data[4]	Serial number	servo_no
Data[5]	Counterclockwise insensitive zone value	data
Data[6]	End frame	0XFA

Example:

Serial transmission: FE FE 04 74 01 01 FA

servo_no Value range: 1-8 The value range of data is 0-32

There is no return value

Reads the counterclockwise insensitive area of the encoder for the specified servo motor

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0XEB
Data[4]	Motor serial number	servo_no
Data[5]	End frame	0XFA

Example:

Serial transmission: FE FE 03 EB 01 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0XEB
Data[4]	Motor serial number	servo_no
Data[5]	Counterclockwise insensitive zone value	data
Data[6]	End frame	0XFA

Example:

Serial port return value: FE FE 04 EB 01 01 FA

Read the servo parameters

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X05
Data[3]	Command frame	0X53
Data[4]	Joint servo serial number	Joint_number
Data[5]	Data address	data_id
Data[5]	Read length	data_len
Data[6]	End frame	0XFA

Example: Read the proportional parameter of the position P of the servo position No. 1

Serial transmission: FE FE 03 53 01 15 02 FA

joint_no Value range: 1-8

data_len: 1/2 1 byte/2 byte

Data_id: The data type is byte, and the values in the following table can be used

Address	FEATURES	Value range	Initial value	Value parsing
20	LED Alarm	0-254	0	1/0: Turn on/off LED siren
21	Position ring P	0-254	10	Control motor scale factor
22	Position Ring I	0-254	0	Controls the differential coefficient of the motor
23	Position Ring D	0-254	1	Control the integral factor of the motor
24	Minimum starting force	0-1000	0	Set the minimum output torque 1000 = 100%

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Return value: Command frame	0X53
Data[5]	Data address	data_id
Data[5]	Return value: data	data_high
Data[4]	Return value: data	data_low
Data[5]	End frame	0XFA

Set the servo parameters

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X05
Data[3]	Command frame	0X52
Data[4]	Joint servo serial number	Joint_no
Data[5]	Data address	data_id
Data[5]	Data high	data_high
Data[5]	Data low	data_low
Data[6]	End frame	0XFA

Example:

Set the Servo 1 Position P Scale parameter to 1

Serial transmission: FE FE 05 52 01 15 00 01 FA

joint_no Value range: 1-8

There is no return value

Data_id: The data type is byte, as shown in the following figure:

Address	FEATURES	Value range	Initial value	Value parsing
20	LED Alarm	0-254	0	1/0: Turn on/off LED siren
21	Position ring P	0-254	10	Control motor scale factor
22	Position Ring I	0-254	0	Controls the differential coefficient of the motor
23	Position Ring D	0-254	1	Control the integral factor of the motor
24	Minimum starting force	0-1000	0	Set the minimum output torque 1000 = 100%

Set Atom mode

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X60
Data[4]	Balloom	pin_no
Data[5]	Input/Output	00X00/00X01
Data[6]	End frame	0XFA

Example:

Set Atom Pin22 to input mode

Serial transmission: FE FE 04 60 16 00 FA

Pin_no: Data type byte

Pin_mode: 0/1

There is no return value

Set Atom IO(setDigitalOutput)

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X61
Data[4]	Balloom	pin_no
Data[5]	Level Signal	0X00/0X01
Data[6]	End frame	0XFA

Example:

Set pin P2 high

Serial transmission: FE FE 04 61 02 01 FA

pin_no: 1/2

There is no return value

Read Atom IO(getDigitalInput)

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0X62
Data[4]	Balloom	pin_no
Data[5]	End frame	0XFA

Example:

Read the level signal of pin P2

Serial transmission: FE FE 03 62 02 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Return value: Command frame	0X53
Data[4]	Balloom	pin_no
Data[5]	Level Signal	0X00/0X01
Data[6]	End frame	0XFA

Example:

Suppose pin P2 is high

Port return value: FE FE 04 62 02 01 FA

Set the master IO output

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0X65
Data[4]	Balloom	Pin_no
Data[5]	Level Signal	0X00/0X01
Data[4]	End frame	0XFA

Example:

Set pin 2 output high

Serial transmission: FE FE 04 65 02 01 FA

Read master IO output

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0X66
Data[4]	Balloom	Pin_no
Data[4]	End frame	0XFA

Serial transmission: FE FE 03 66 02 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0X66
Data[4]	Balloom	Pin_no
Data[5]	Level Signal	0X00/0X01
Data[6]	End frame	0XFA

Example:

Suppose pin 2 is high

Port return value: FE FE 04 66 02 01 FA

Set the color of the RGB lights on the atom screen

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X05
Data[3]	Command frame	0X6A
Data[4]	R	0X00/0XFF
Data[5]	G	0X00/0XFF
Data[6]	B	0X00/0XFF
Data[7]	End frame	0XFA

Example:

Set RGB to blue

Serial transmission: FE FE 05 6A 00 00 FF FA

There is no return value

Read the end LED pressed status

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X02
Data[3]	Command frame	0X6B
Data[4]	End frame	0XFA

Serial transmission: FE FE 02 6B FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X03
Data[3]	Return value: Command frame	0X6B
Data[5]	Pressed/unpressed	0X00/0X01
Data[6]	End frame	0XFA

Set the base IO output

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X04
Data[3]	Command frame	0Xa0
Data[4]	Balloom	Pin_no
Data[5]	Level Signal	0X00/0X01
Data[4]	End frame	0XFA

Example:

Set pin 2 output high

Serial transmission: FE FE 04 a0 02 01 FA

Read the base IO output

Data Field	Description	DATA
Data[0]	Recognize frame	0XFE
Data[1]	Recognize frame	0XFE
Data[2]	Data length frame	0X03
Data[3]	Command frame	0Xa1
Data[4]	Balloom	Pin_no
Data[4]	End frame	0XFA

Serial transmission: FE FE 03 a1 02 FA

Return Value: Data structure

Data Field	Description	DATA
Data[0]	Return value: Recognition frame	0XFE
Data[1]	Return value: Recognition frame	0XFE
Data[2]	Return Value: Data Length Frame	0X04
Data[3]	Return value: Command frame	0Xa1
Data[4]	Balloom	Pin_no
Data[5]	Level Signal	0X00/0X01
Data[6]	End frame	0XFA

Example:

Suppose pin 2 is high

Port return value: FE FE 04 a1 02 01 FA

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

Success Stories

The myArm M750 general-purpose intelligent 6-degree-of-freedom robot motion control verification device represents the latest advancement in robotics in terms of precision and flexibility.

It not only meets the demand for high-precision robot motion control, but is also particularly suitable for entry-level robot motion algorithm validation and hands-on teaching scenarios. Its standardised robotic arm structure provides an ideal learning platform for students and beginners to master the basic principles and applications of robot kinematics.



myArm M750 Robotic actuators (Master)

The myArm M750 is dedicated to precise motion control and verification, and excels especially for application scenarios that require strict operational accuracy, such as precision assembly, fine manipulation, and quality monitoring.

With a rated load capacity of 500g and a peak load of up to 1kg, it is equipped with industrial-grade high-precision digital servo motors and leading-edge control algorithms, which give the myArm M750 extraordinary torque control and precise positional positioning capabilities, making it a core element in achieving highly accurate robot motion control.



The versatility of the myArm M750 is emphasised by the design of its end-effector. Parallel gripper jaws and vision module as standard - providing the user with basic gripping and identification capabilities



In addition to this, the myArm M750 is compatible with a range of optional accessories that greatly broaden the arm's field of application and flexibility.



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

C650 control M750 program example

Program address

https://github.com/elephantrobotics/pymycobot/tree/main/demo/myArm_M%26C_demo_v1.1

Installation dependency

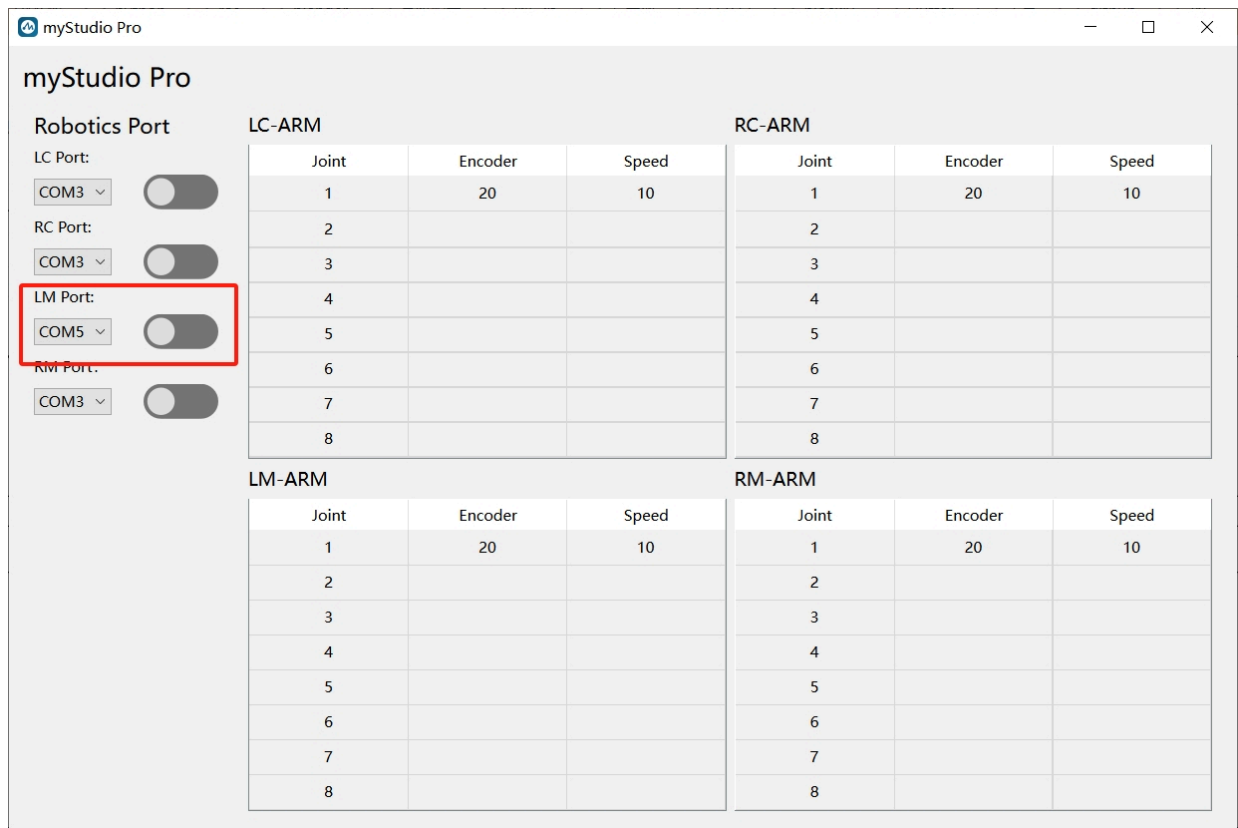
```
pip install -r requirement.txt
```

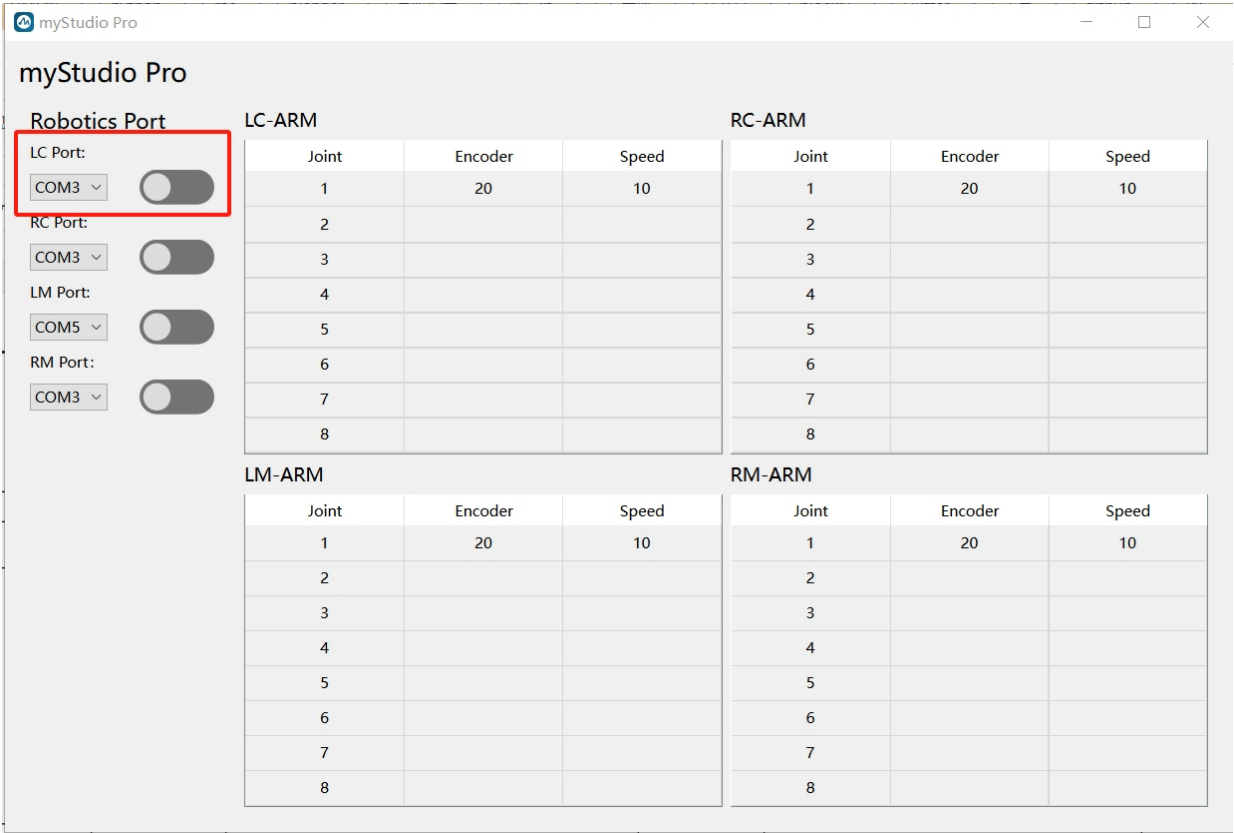
Run the program

```
python main.py
```

Program instructions

There is a sequence requirement for opening the serial port: open the serial port connection of myArmM first, and then open the serial port connection of myArmC.





After both serial ports are enabled, you can control the movement of myArmM by moving myArmC.

Keyboard control robot movement case

Function description: Based on Cartesian motion, control the robot arm to move in the XYZ direction through the keyboard

1 Firmware burning

Since the robot needs to use Cartesian motion, it is necessary to burn the pico firmware and basic firmware that support Cartesian motion. However, the firmware that supports Cartesian motion has not been released on mystudio and can only be obtained by contacting after-sales personnel.

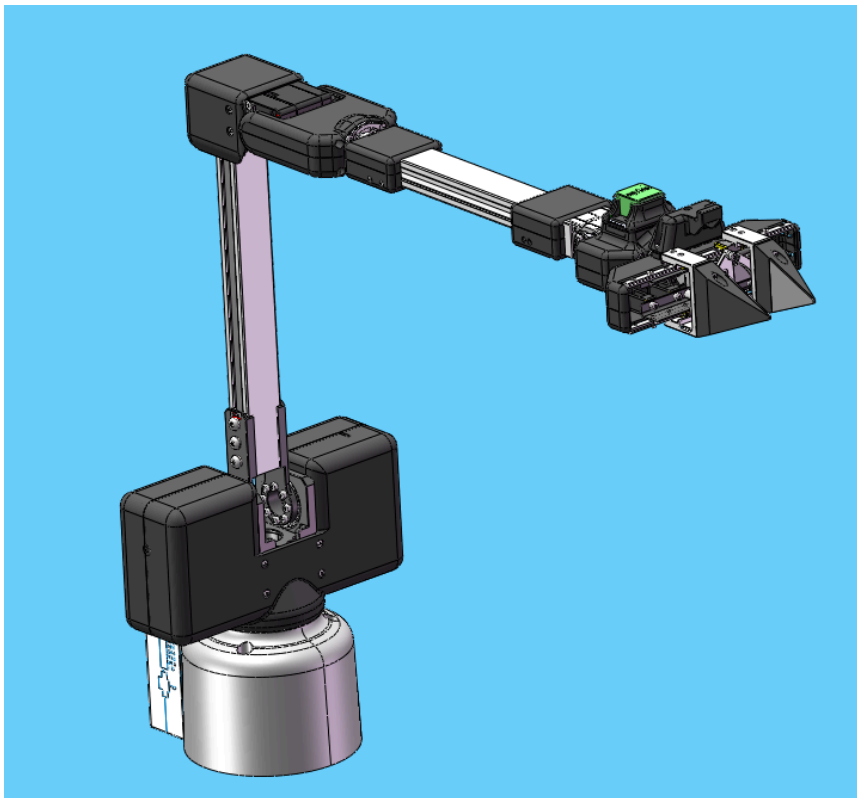
2 pymycobot installation

If you want to use Cartesian motion, you need to install or update pynycobot. Open the computer terminal and execute the following command to install or update pymycobot

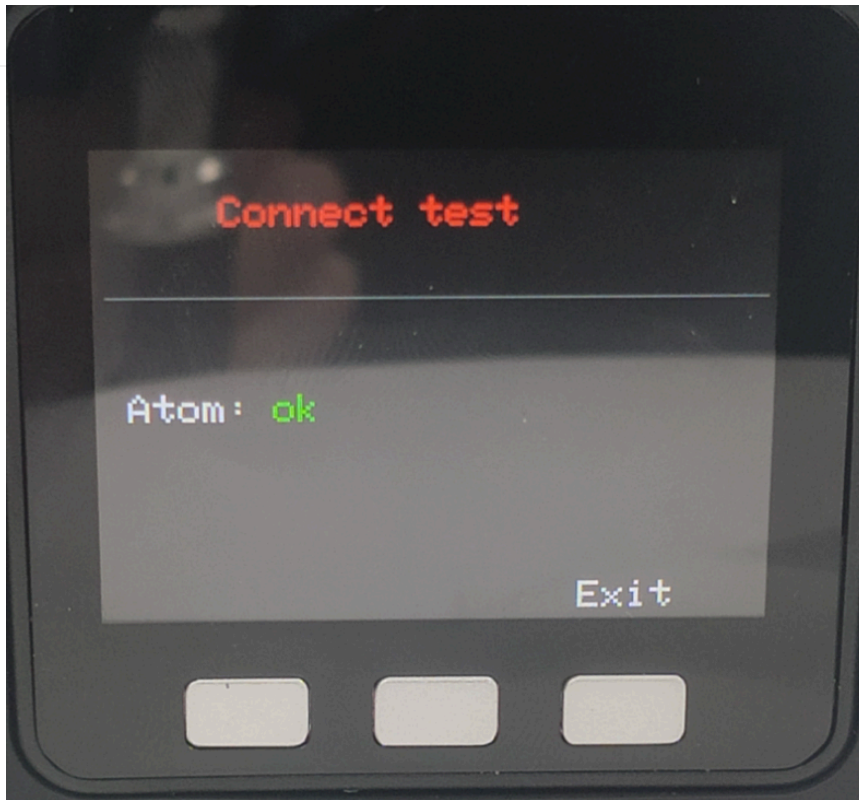
```
pip install pymycobot --upgrade
```

3 Preparation

Before connecting the robot arm to 24V, manually adjust the robot arm to the posture shown in the figure below, and then connect the 24V power supply and communication data cable in turn. There should be no debris around the robot arm to avoid collision



Make sure the base screen displays ok



4 Case reproduction

After running the following program, the robot arm will first move to an initial position, and then print the key prompt information in the terminal. Press the corresponding key according to the terminal information to control the robot movement




```
开始控制...
按下键盘W,往X+方向运动
按下键盘S,往X-方向运动
按下键盘D,往Y+方向运动
按下键盘A,往Y-方向运动
按下键盘Q,往Z+方向运动
按下键盘Z,往Z-方向运动
按下键盘X,往J6+方向运动
按下键盘E,往J6-方向运动
按下键盘F,获取当前坐标
按下键盘ESC,结束程序
当前坐标= [267.7, -14.1, -12.2, 178.74, -3.89, 173.93]
```

```

import threading
from pymycobot import MyArmMControl, utils
import keyboard
import time

m = MyArmMControl(utils.get_port_list()[0], 1000000)

def init():
    # 设置初始角度
    m.write_angles([-10.19, 8.62, 30.65, 2.19, 50.53, -4.83], 100)
    time.sleep(1)
    m.set_gripper_state(0, 100)
    time.sleep(1)

# 用于键盘输入检测的函数
def keyborad_ctrl():
    print("Start control...")
    print("Press the W keyboard to move in the X+ direction")
    print("Press the S keyboard to move in the X- direction")
    print("Press the D keyboard to move in the Y+ direction")
    print("Press the A keyboard to move in the Y- direction")
    print("Press the Q keyboard to move in the Z+ direction")
    print("Press the Z keyboard to move in the Z- direction")
    print("Press the X keyboard to move in the J6+ direction")
    print("Press the E keyboard to move in the J6- direction")
    print("Press the F keyboard to get the current coordinates")
    print("Press the ESC keyboard to end the program")
    blocked_keys = ['w', 'a', 's', 'd', 'q', 'z', 'f', 'e', 'x']
    for key in blocked_keys:
        keyboard.block_key(key)
    key_processed = {
        'w': False,
        'a': False,
        's': False,
        'd': False,
        'q': False,
        'z': False,
        'f': False,
        'e': False,
        'x': False,
    }
    else_executed = False
    try:
        while True:
            if keyboard.is_pressed('esc'):
                print("Exit control...")
                break
            for key in blocked_keys:
                if keyboard.is_pressed(key) and not key_processed[key]:
                    threading.Thread(target=handle_key, args=(key,)).start()
                    key_processed[key] = True
            else_executed = False

```

```

        if all(not keyboard.is_pressed(key) for key in blocked_keys) and not else_executed:
            m.stop()
            else_executed = True
        for key in key_processed:
            if not keyboard.is_pressed(key):
                key_processed[key] = False
        time.sleep(0.01)
    finally:
        keyboard.unhook_all()

def handle_key(key):
    if key == 'w':
        m.jog_coord(1, 1, 60)
    elif key == 'a':
        m.jog_coord(2, 0, 60)
    elif key == 'd':
        m.jog_coord(2, 1, 60)
    elif key == 's':
        m.jog_coord(1, 0, 60)
    elif key == 'q':
        m.jog_coord(3, 1, 60)
    elif key == 'z':
        m.jog_coord(3, 0, 60)
    elif key == 'e':
        m.jog_angle(6,0,50)
    elif key == 'x':
        m.jog_angle(6,1,50)
    elif key == 'f':
        print("Current coordinates=",m.get_coords())

if __name__ == "__main__":
    init()
    keyborad_ctrl()

```

5 Effect display



Robot palletizing and handling case

Function description: The robot moves the wood block from point A to point B

1 Firmware burning

Since the robot needs to use Cartesian motion, it is necessary to burn the pico firmware and basic firmware that support Cartesian motion. However, the firmware that supports Cartesian motion has not been released on mystudio and can only be obtained by contacting after-sales personnel.

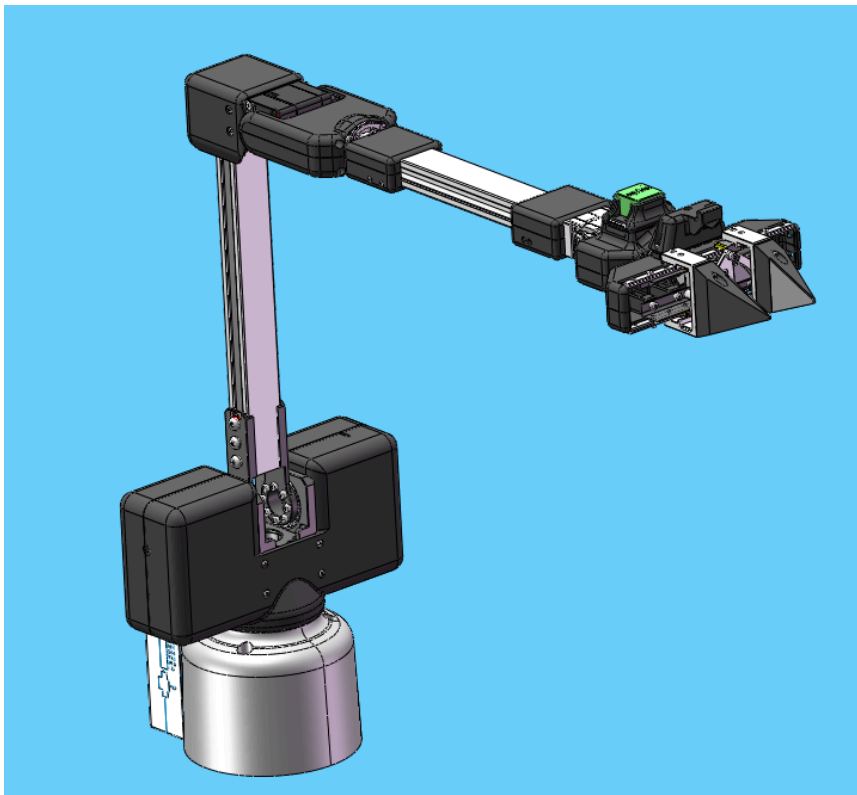
2 pymycobot installation

If you want to use Cartesian motion, you need to install or update pymycobot. Open the computer terminal and execute the following command to install or update pymycobot

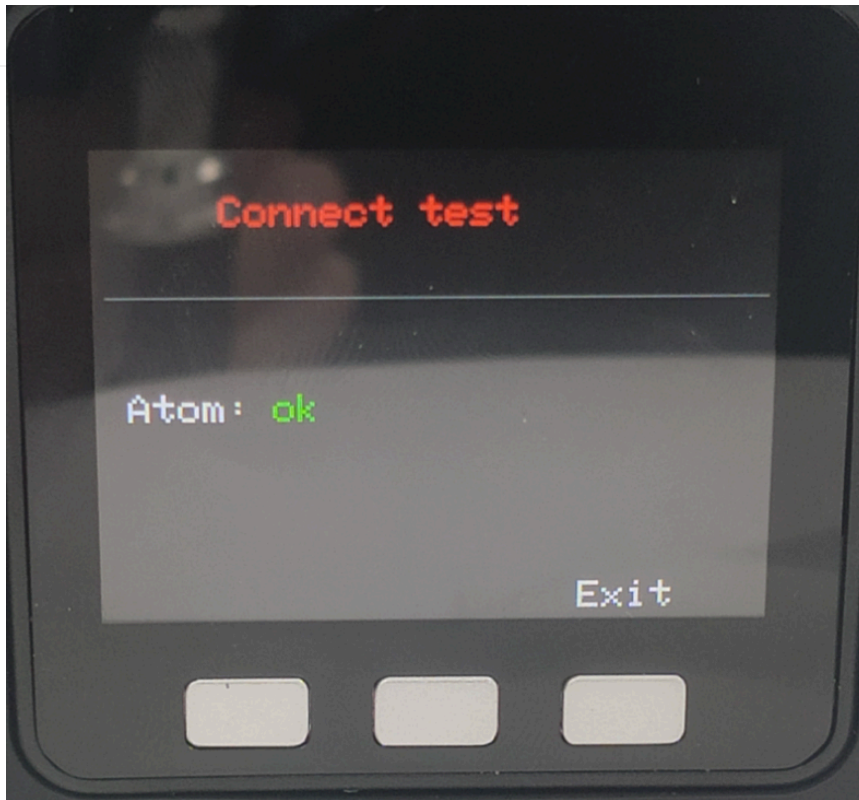
```
pip install pymycobot --upgrade
```

3 Preparation

Before connecting the robot arm to 24V, manually adjust the robot arm to the posture shown in the figure below, and then connect the 24V power supply and communication data cable in turn. There should be no debris around the robot arm to avoid collision



Make sure the base screen displays ok



4 Keyboard control

After running the following program, the robot arm will first move to an initial position, and then print the key prompt information in the terminal. Press the corresponding key according to the terminal information to control the robot movement



```

import threading
from pymycobot import MyArmMControl, utils
import keyboard
import time

m = MyArmMControl(utils.get_port_list()[0], 1000000)

def init():
    # 设置初始角度
    m.write_angles([-10.19, 8.62, 30.65, 2.19, 50.53, -4.83], 100)
    time.sleep(1)
    m.set_gripper_state(0, 100)
    time.sleep(1)

# 用于键盘输入检测的函数
def keyborad_ctrl():
    print("Start control...")
    print("Press the W keyboard to move in the X+ direction")
    print("Press the S keyboard to move in the X- direction")
    print("Press the D keyboard to move in the Y+ direction")
    print("Press the A keyboard to move in the Y- direction")
    print("Press the Q keyboard to move in the Z+ direction")
    print("Press the Z keyboard to move in the Z- direction")
    print("Press the X keyboard to move in the J6+ direction")
    print("Press the E keyboard to move in the J6- direction")
    print("Press the F keyboard to get the current coordinates")
    print("Press the ESC keyboard to end the program")
    blocked_keys = ['w', 'a', 's', 'd', 'q', 'z', 'f', 'e', 'x']
    for key in blocked_keys:
        keyboard.block_key(key)
    key_processed = {
        'w': False,
        'a': False,
        's': False,
        'd': False,
        'q': False,
        'z': False,
        'f': False,
        'e': False,
        'x': False,
    }
    else_executed = False
    try:
        while True:
            if keyboard.is_pressed('esc'):
                print("Exit control...")
                break
            for key in blocked_keys:
                if keyboard.is_pressed(key) and not key_processed[key]:
                    threading.Thread(target=handle_key, args=(key,)).start()
                    key_processed[key] = True
            else_executed = False

```

```

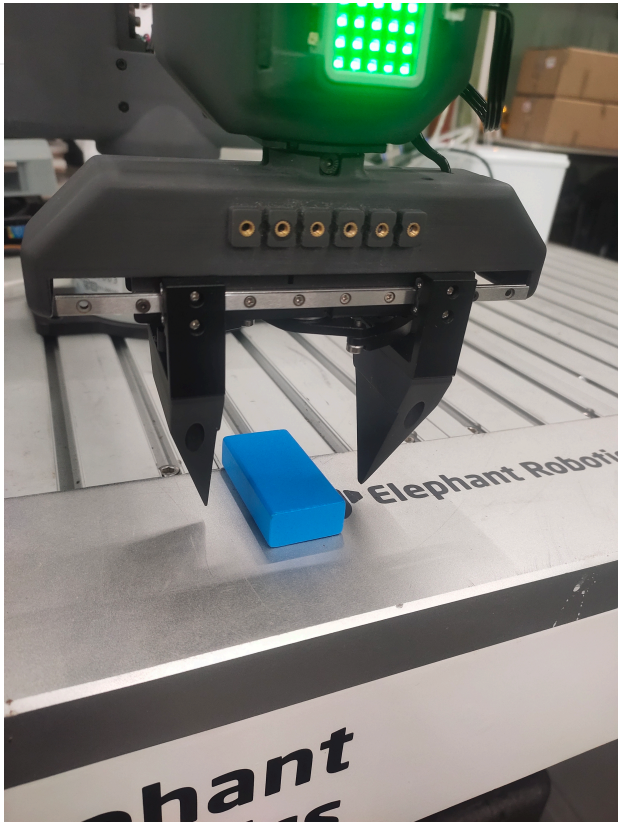
        if all(not keyboard.is_pressed(key) for key in blocked_keys) and not else_executed:
            m.stop()
            else_executed = True
        for key in key_processed:
            if not keyboard.is_pressed(key):
                key_processed[key] = False
        time.sleep(0.01)
    finally:
        keyboard.unhook_all()

def handle_key(key):
    if key == 'w':
        m.jog_coord(1, 1, 60)
    elif key == 'a':
        m.jog_coord(2, 0, 60)
    elif key == 'd':
        m.jog_coord(2, 1, 60)
    elif key == 's':
        m.jog_coord(1, 0, 60)
    elif key == 'q':
        m.jog_coord(3, 1, 60)
    elif key == 'z':
        m.jog_coord(3, 0, 60)
    elif key == 'e':
        m.jog_angle(6,0,50)
    elif key == 'x':
        m.jog_angle(6,1,50)
    elif key == 'f':
        print("Current coordinates=",m.get_coords())

if __name__ == "__main__":
    init()
    keyborad_ctrl()

```

When the robot reaches the target point, record the current coordinates and close the program



```
开始控制...
按下键盘W,往X+方向运动
按下键盘S,往X-方向运动
按下键盘D,往Y+方向运动
按下键盘A,往Y-方向运动
按下键盘Q,往Z+方向运动
按下键盘Z,往Z-方向运动
按下键盘X,往J6+方向运动
按下键盘E,往J6-方向运动
按下键盘F,获取当前坐标
按下键盘ESC,结束程序
当前坐标= [267.7, -14.1, -12.2, 178.74, -3.89, 173.93]
```


5 Case reproduction

```

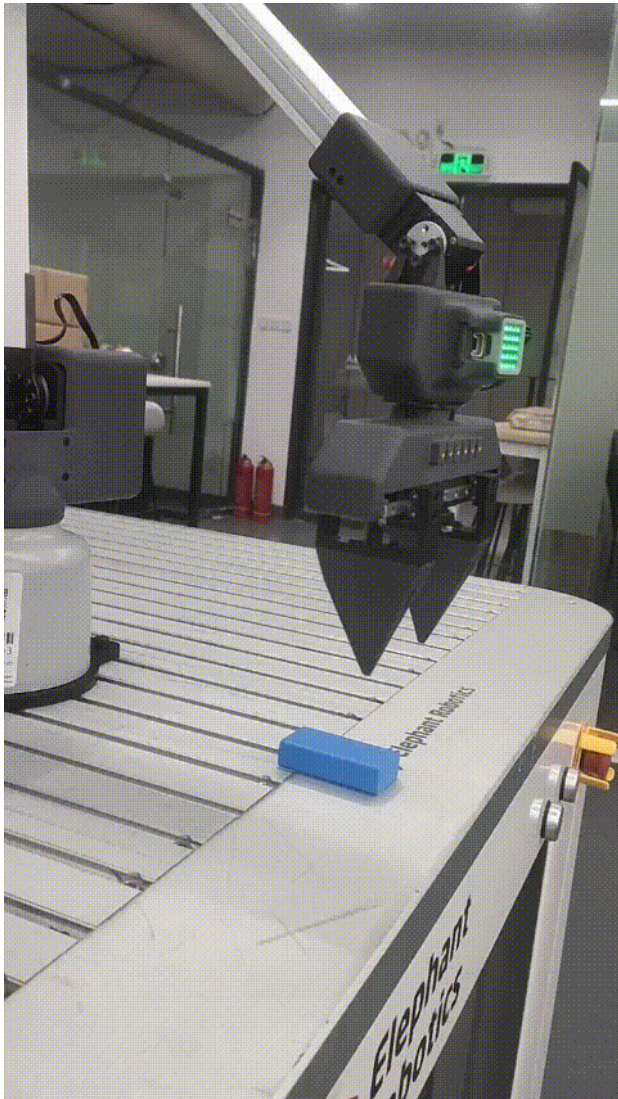
from pymycobot import MyArmMControl, utils
import time

init_angles=[-10.19, 8.62, 30.65, 2.19, 50.53, -4.83]#6 joint angles at the initial position
target_point=[277.7, -14.6, -7.5, 178.74, -2.04, 174.17]#Coordinate position of the gripping target
m = MyArmMControl(utils.get_port_list()[0],1000000)
if __name__=="__main__":
    m.set_gripper_state(0,100)#Open the gripper first
    time.sleep(1)
    m.write_angles(init_angles,100)#Move to the initial position
    time.sleep(2)
    m.write_coords([target_point[0],target_point[1],target_point[2]+50,target_point[3],target_point[4],target_point[5]],100,100,100)
    time.sleep(2)
    m.write_coords([target_point[0],target_point[1],target_point[2],target_point[3],target_point[4],target_point[5]],100,100,100)
    time.sleep(2)
    m.set_gripper_state(1,100)#Gripper closed
    time.sleep(1)
    m.write_coords([target_point[0],target_point[1],target_point[2]+50,target_point[3],target_point[4],target_point[5]],100,100,100)
    time.sleep(2)

    m.write_coords([target_point[0],target_point[1]+100,target_point[2]+50,target_point[3],target_point[4],target_point[5]],100,100,100)
    time.sleep(2)
    m.write_coords([target_point[0],target_point[1]+100,target_point[2],target_point[3],target_point[4],target_point[5]],100,100,100)
    time.sleep(2)
    m.set_gripper_state(0,100)#Gripper open
    time.sleep(1)
    m.write_coords([target_point[0],target_point[1]+100,target_point[2]+50,target_point[3],target_point[4],target_point[5]],100,100,100)
    time.sleep(2)

```

6 Effect display



Elephant Robotics



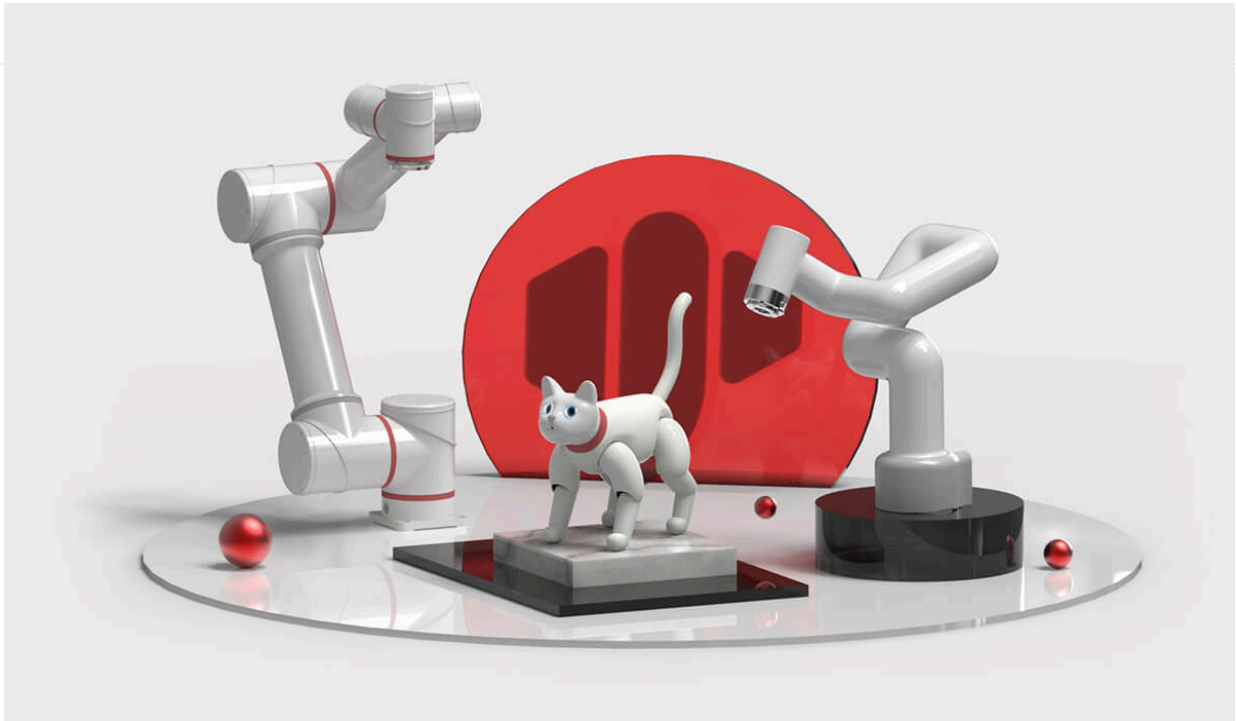
1. Company Introduction

Elephant Robotics, based in Shenzhen, China, is a high-tech enterprise specializing in robot research and development and automation solutions.

We are committed to providing highly flexible collaborative robots, easy-to-learn operating systems, and intelligent automation solutions for robot education and research institutions, commercial scenarios, and industrial production. Its product quality and intelligent solutions have been widely recognized and praised by factories of many Fortune 500 companies from South Korea, Japan, the United States, Germany, Italy, Greece, and other countries.

With the vision of "Enjoy Robots World," Elephant Robotics advocates for the collaboration between humans and robots, making robots a good helper for human work and life, helping people break free from simple, repetitive, and tedious work, fully leveraging the advantages of human-robot collaboration to improve work efficiency, and helping humans create a better new life.

In the future, Elephant Robotics hopes to promote the development of the robotics industry through cutting-edge technology and work together with customer partners to open a new era of automation and intelligence.



2. Development History

- 2016.08 ----- Elephant Robotics Co., Ltd. officially established
- 2016.08 ----- Entered HAX incubator and received SOSV seed round investment
- 2016.08 ----- Started developing Elephant S industrial collaborative robots
- 2017.01 ----- Awarded "CES China's Top 10 Most Innovative Enterprises"
- 2017.04 ----- Attended Hannover Messe and Korea Automation Exhibition
- 2017.07 ----- Two founders selected as "30 Under 30 Business Elites in Asia" by Forbes
- 2017.10 ----- The fifth-generation single-arm industrial collaborative robot Elephant S was launched
- 2018.04 ----- Received investment from "Angel Round of Cloud Angel Fund"
- 2018.06 ----- Made its debut at Hannover Messe 2018 for the first time
- 2018.06 ----- Won the "Smart Manufacturing Entrepreneur MBA Award" from Cheung Kong Graduate School of Business
- 2018.06 ----- Won the "X-elerator Award" from Tsinghua SEM "Entrepreneurship Accelerator"
- 2018.11 ----- Won the second prize in the Shenzhen Division of the Asia Intelligent Hardware Competition
- 2018.11 ----- Won the "Most Investable Enterprise Award" of Gao Gong Jin Qiu
- 2019.03 ----- Won the "Leadership Award" of Gao Gong Jin Qiu
- 2019.04 ----- Catbot won the "Industrial Robot Innovation Award" in March 2019
- 2019.09 ----- Attended Huawei Europe Eco-Connect (HCE) and officially became a member of the Huawei ecosystem
- 2019.11 ----- Elephant Robotics and Harbin Institute of Technology attended the IROS International Conference on Intelligent Robots and Systems
- 2019.12 ----- Elephant Robotics-South China University of Technology "Intelligent Robot Joint Development Laboratory" officially unveiled
- 2019.12 ----- Won the "Innovative Technology Award" of Gao Gong 2019
- 2019.12 ----- Won the "Top Ten Fast-Growing Enterprises" of Gao Gong 2019

- 2019.12 ----- Won the "New Enterprise Award" in the Shenzhen Equipment Industry-Industrial Robot Sub-Sector
- 2019.12 ----- The world's first bionic robot cat MarsCat was born
- 2020.05 ----- Founder won the "Shenzhen Robot Rising Star Award" for 2019
- 2020.10 ----- The world's lightest and smallest six-axis collaborative robot myCobot was born
- 2021.03 ----- The smallest collaborative robot myCobotPro 320 for scientific research was born
- 2021.05 ----- The MarsCat bionic cat received extensive coverage from Xinhua Finance, China Daily, Nanjing Daily, and Harbin Daily
- 2021.07 ----- Released the smallest composite robot chassis - Little Elephant Mobile Robot myAGV
- 2021.09 ----- The world's first fully enclosed four-axis mechanical arm - Little Elephant Palletizing Robot myPalletizer was born

3. Related 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>
 - Twitter: <https://twitter.com/CobotMy>
 - Discord: <https://discord.gg/2MAherp7nt>
 - Hackster: <https://www.hackster.io/elephant-robotics>

4. Contact Us

Our working hours are Chinese working days, from 10:00 a.m. to 6:00 p.m. Beijing time.

- If you have any other questions, please contact us via the following methods.

E-mail :

support@elephantrobotics.com

- If you have purchase intentions or any parameter questions, please send an email to this email address. [E-mail] (sales@elephantrobotics.com) :

`sales@elephantrobotics.com`

- If the listed questions cannot help you solve the problem, and you have more after-sales questions, please send an email to this email address.

E-mail :

`support@elephantrobotics.com`

We will reply within 1-2 working days;

WeChat:

We only provide one-on-one service for users who purchase myArm M&C series products through WeChat.



Acknowledgments

Here, we express our heartfelt gratitude to all those involved in the development, testing, and improvement of the myArm series products (including myArm Controller 650, myArm Master 750, and M&C kit). The meticulous refinement of every detail and the innovation of every feature would not have been possible without the hard work and dedication of the team behind it.

Special Thanks To:

Development Team: Thank you for your innovative thinking and countless days and nights of effort in transforming complex technology into user-friendly products.

Quality Assurance and Testing Team: Your strict control over every detail ensures the reliability of our products and the ultimate experience for users.

Customer Support Team: Thank you for providing professional support to our users, helping them solve every issue they encounter during the use of our products.

Partners and Suppliers: Your support and service are crucial to the success of our products. We appreciate your high-quality raw materials and components, as well as your readiness to support at all times.

Investors and Advisors: Without your trust and financial support, we would not have been able to bring these innovations to the market. Your insights and guidance have always been the driving force behind our progress.

User Appreciation:

We especially thank every user who chooses and trusts the myArm series products. Your feedback and suggestions drive us to continuously advance and improve. We promise to continue listening to your voices and continuously optimize our products and services.

Future Prospects:

We look forward to exploring and advancing further in the field of robotics technology together with all stakeholders. Let us join hands to create more possibilities and bring greater convenience and innovation to the world.

[← Previous Chapter](#)