

Table of Contents

| | |
|---|-----------|
| Introduction | 1.1 |
| Product Information | 1.2 |
| 1. Product Introduction | 1.2.1 |
| 2. Product Parameters | 1.2.2 |
| Basic Settings | 1.3 |
| 3. User Notice | 1.3.1 |
| 4. First Time Installation | 1.3.2 |
| 4.1 First-time self-check | 1.3.2.1 |
| 4.2 Software issues | 1.3.2.2 |
| 4.3 Hardware issues | 1.3.2.3 |
| 4.4 Accessories issues | 1.3.2.4 |
| 4.5 Others | 1.3.2.5 |
| Functions and applications | 1.4 |
| 5. Basic functions | 1.4.1 |
| 5.1 Introduction to the use of the robotic arm system | 1.4.1.1 |
| 5.2 myStudio instructions | 1.4.1.2 |
| 5.3 Hardware Interface | 1.4.1.3 |
| 5.4 PID control | 1.4.1.4 |
| 6. Software Development Guide | 1.4.2 |
| 6.1 Development and Use Based on Arduino | 1.4.2.1 |
| 6.1.1 Environment Construction | 1.4.2.1.1 |
| 6.1.2 Simple Use | 1.4.2.1.2 |
| 6.1.3 API description | 1.4.2.1.3 |
| 6.2 Development and use based on Python | 1.4.2.2 |
| 6.2.1 Environment construction | 1.4.2.2.1 |
| 6.2.2 API description | 1.4.2.2.2 |
| 6.2.3 Joint control | 1.4.2.2.3 |
| 6.2.4 Coordinate Control | 1.4.2.2.4 |
| 6.2.5 IO Control | 1.4.2.2.5 |
| 6.2.6 Gripper Control | 1.4.2.2.6 |
| 6.2.7 Handle Control | 1.4.2.2.7 |
| 6.2.8 Drawing Patterns | 1.4.2.2.8 |
| 6.2.9 Demonstration Code and Video | 1.4.2.2.9 |
| 6.3 Development and Use Based on ROS1 | 1.4.2.3 |
| 6.3.1 ROS1 Environment Building | 1.4.2.3.1 |
| 6.3.2 ROS1 Basics | 1.4.2.3.2 |
| 6.3.3 rrvz Introduction and Use | 1.4.2.3.3 |
| 6.3.4 Moveit Introduction and Use | 1.4.2.3.4 |

4.1 First-time self-check

| | |
|--|------------|
| 6.4 Development and Use Based on ROS2 | 1.4.2.4 |
| 6.4.1 ROS2 environment installation | 1.4.2.4.1 |
| 6.4.2 ROS2 basics | 1.4.2.4.2 |
| 6.4.3 rivz introduction and use | 1.4.2.4.3 |
| 6.4.4 Moveit2 Introduction and Use | 1.4.2.4.4 |
| 6.5 Development and use based on Blockly | 1.4.2.5 |
| 6.5.1 Initial use of myBlockly | 1.4.2.5.1 |
| 6.5.2 Control RGB light board | 1.4.2.5.2 |
| 6.5.3 Control the robotic arm to return to the origin | 1.4.2.5.3 |
| 6.5.4 Control single joint motion | 1.4.2.5.4 |
| 6.5.5 Control multiple joints | 1.4.2.5.5 |
| 6.5.6 Control the left and right swing of the robotic arm | 1.4.2.5.6 |
| 6.5.7 Control Robotic Arm Dance | 1.4.2.5.7 |
| 6.5.8 Gripper Usage | 1.4.2.5.8 |
| 6.5.9 Pump Usage | 1.4.2.5.9 |
| 6.5.10 Gripper Test | 1.4.2.5.10 |
| 6.5.11 IO Test | 1.4.2.5.11 |
| 6.5.12Q&A | 1.4.2.5.12 |
| 6.6 Development and use based on serial communication protocol | 1.4.2.6 |
| 7. Successful Cases | 1.4.3 |
| Robot gripper carrying wooden block example | 1.4.3.1 |
| 280Arduino handle remote control box | 1.4.3.2 |
| Robot suction pump to carry wooden blocks | 1.4.3.3 |
| Mega2560-gripper to carry wood blocks | 1.4.3.4 |
| Raspberry Pi 4B-Stag code visual tracking | 1.4.3.5 |
| Jetson Xavier NX-item classification example | 1.4.3.6 |
| 8. Supporting Resources | 1.4.4 |
| 8.1 Product Information | 1.4.4.1 |
| 8.2 Product Drawings | 1.4.4.2 |
| 8.3 Software Information and Source Code | 1.4.4.3 |
| 8.4 System Information | 1.4.4.4 |
| 8.5 Promotional Materials | 1.4.4.5 |
| Support and Service | 1.5 |
| Contact Us | 1.5.1 |
| Robot Arm Accessories | 1.5.2 |
| Flat Base | 1.5.2.1 |
| G-Stand | 1.5.2.2 |
| Adaptive Gripper | 1.5.2.3 |
| Parallel Gripper | 1.5.2.4 |
| Flexible Gripper - Open Leg Type | 1.5.2.5 |

4.1 First-time self-check

| | |
|--------------------------|----------|
| Vertical Suction Pump | 1.5.2.6 |
| Double-head suction pump | 1.5.2.7 |
| Integrated suction pump | 1.5.2.8 |
| Pen holder | 1.5.2.9 |
| Phone holder | 1.5.2.10 |
| Dexterous hand | 1.5.2.11 |
| USB camera | 1.5.2.12 |
| Bamboo flange | 1.5.2.13 |
| Acknowledgments | 1.5.3 |

myCobot 280 For Arduino(2023)

Algorithm Verification Practice Teaching

Core Document

This document contains comprehensive information from product introduction, detailed technical parameters to user instructions and product development guidance. The document will introduce the basic functions of the myCobot 280 For Arduino robot arm in depth, provide a software development guide, and show successful application cases to help you understand how to effectively integrate myCobot 280 For Arduino into various applications. In addition, we also provide a wealth of support and service information to ensure that you can get the necessary help when you encounter any technical challenges.

Document Description

Depending on your needs and your level of expertise in myCobot 280 For Arduino application development, you can choose to follow this order from beginning to end or use it as a standalone reference. You can always use the sidebar navigation on the left to jump to any part of this document. The full text is divided into the following five sections:

Product Information

The product information section will provide you with a basic overview of the robot arm, including detailed technical specifications such as main functions, product parameters and electrical characteristics, to help you quickly understand the basic characteristics and usage environment of the product. In addition, this section will detail the application examples and supported extension development of the product, providing you with the necessary development guides and resources. At the end of the article, relevant purchase links and channels will be given to facilitate your purchase.

Basic Settings

This section is an important part that every user of this product must read carefully. It covers key information about product use, transportation, storage and maintenance, aiming to ensure the safety and efficiency of users when operating the product. In addition, this section also details the division of responsibilities for product failure or damage that may occur due to failure to follow these guidelines.

Functions and Applications

The Functions and Applications section details the basic functions of the robot arm and how to use the software, including system instructions and firmware functions. The Software Development Guide provides guidance based on different development environments, such as Python and ROS, to support technical developers to expand applications. By showing successful application cases and providing supporting resources, it provides you with practical references and necessary support materials for a deeper understanding and use of 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, to help you quickly resolve problems when you encounter them, and ensure that you understand your rights and obligations after purchase. In addition, the 'About Us' section

strengthens the user's understanding of the design and manufacturer of the myCobot series products, aiming to build trust and brand loyalty.

Acknowledgements

We really appreciate you taking the time to read the myCobot 280 For Arduino User Manual. We hope this document will help you better understand and effectively use this robot, thereby 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 your innovative projects using myCobot 280 For Arduino and welcome you to join our fast-growing developer community.

Document Directory

Summary

- [Introduction](#)
- [Product Information](#)
 - [1. Product Introduction](#)
 - [2. Product Parameters](#)
- [Basic Settings](#)
 - [3. User Notice](#)
 - [4. First Time Installation](#)
 - [4.1 First-time self-check](#)
 - [4.2 Software issues](#)
 - [4.3 Hardware issues](#)
 - [4.4 Accessories issues](#)
 - [4.5 Others](#)
- [Functions and applications](#)
 - [5. Basic functions](#)
 - [5.1 Introduction to the use of the robotic arm system](#)
 - [5.2 myStudio instructions](#)
 - [5.3 Hardware Interface](#)
 - [5.4 PID control](#)
 - [6. Software Development Guide](#)
 - [6.1 Development and Use Based on Arduino](#)
 - [6.1.1 Environment Construction](#)
 - [6.1.2 Simple Use](#)
 - [6.1.3 API description](#)
 - [6.2 Development and use based on Python](#)
 - [6.2.1 Environment construction](#)
 - [6.2.2 API description](#)
 - [6.2.3 Joint control](#)
 - [6.2.4 Coordinate Control](#)
 - [6.2.5 IO Control](#)
 - [6.2.6 Gripper Control](#)
 - [6.2.7 Handle Control](#)
 - [6.2.8 Drawing Patterns](#)
 - [6.2.9 Demonstration Code and Video](#)
 - [6.3 Development and Use Based on ROS1](#)

- 6.3.1 ROS1 Environment Building
 - 6.3.2 ROS1 Basics
 - 6.3.3 rrvz Introduction and Use
 - 6.3.4 Moveit Introduction and Use
 - 6.4 Development and Use Based on ROS2
 - 6.4.1 ROS2 environment installation
 - 6.4.2 ROS2 basics
 - 6.4.3 rrvz introduction and use
 - 6.4.4 Moveit2 Introduction and Use
 - 6.5 Development and use based on Blockly
 - 6.5.1 Initial use of myBlockly
 - 6.5.2 Control RGB light board
 - 6.5.3 Control the robotic arm to return to the origin
 - 6.5.4 Control single joint motion
 - 6.5.5 Control multiple joints
 - 6.5.6 Control the left and right swing of the robotic arm
 - 6.5.7 Control Robotic Arm Dance
 - 6.5.8 Gripper Usage
 - 6.5.9 Pump Usage
 - 6.5.10 Gripper Test
 - 6.5.11 IO Test
 - 6.5.12Q&A
 - 6.6 Development and use based on serial communication protocol
 - 7. Successful Cases
 - Robot gripper carrying wooden block example
 - 280Arduino handle remote control box
 - Robot suction pump to carry wooden blocks
 - Mega2560-gripper to carry wood blocks
 - Raspberry Pi 4B-Stag code visual tracking
 - Jetson Xavier NX-item classification example
 - 8. Supporting Resources
 - 8.1 Product Information
 - 8.2 Product Drawings
 - 8.3 Software Information and Source Code
 - 8.4 System Information
 - 8.5 Promotional Materials
 - Support and Service
 - Contact Us
 - Robot Arm Accessories
 - Flat Base
 - G-Stand
 - Adaptive Gripper
 - Parallel Gripper
 - Flexible Gripper - Open Leg Type
 - Vertical Suction Pump
 - Double-head suction pump
 - Integrated suction pump
 - Pen holder
 - Phone holder
 - Dexterous hand
-

4.1 First-time self-check

- USB camera
 - Bamboo flange
-
- Acknowledgments

Chapter 1 Product Introduction

1. Product Overview

ER myCobot 280 For Arduino



Algorithm Verification Practice Teaching

Product Introduction

The mycobot – 280 for Arduino 2023 robot arm is based on the mycobot 280 collaborative robot arm series and uses the **M5STACK-ATOM ESP32** core control. It is an entry-level product of Elephant Robotics. The device can use **Arduino MKRWIFI1010 \ MEGA2560 \ UNO** and other **support UART serial communication** interface module development version expansion, download the product driver library provided by Elephant Robotics, no need to migrate the development board model, just a simple link, you can start expansion development.

myCobot weighs **860g**, has a payload of **250g**, and an effective working radius of **280mm**; it is small in size but powerful in function. It can be matched with a variety of end effectors to adapt to a variety of application scenarios, supports secondary development of multi-platform software, and meets the needs of **scientific research and education, smart home, commercial exploration** and other scenarios

Design concept



myCobot 280 For Arduino uses ATOM Matrix ESP32 Development Kit core development board as the main control. It is equipped with the robot motion control operating system firmware independently developed by Elephant Robotics, with built-in kinematic forward and inverse solutions. It supports matching with a variety of Arduino development boards for extended application development. Using the Arduino development board with a variety of sensors officially supported by Arduino, you can quickly build a robotic arm. It is the preferred assistant for machine vision and robot applications, control logic development, robot fusion development, and ROS simulation experiment classrooms, helping you quickly start learning and applying six-axis robotic arms.

Design Goal

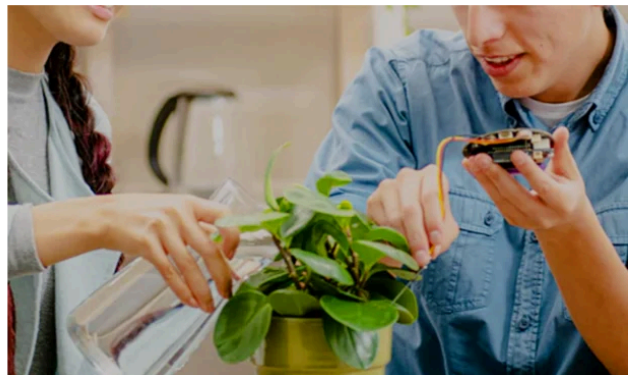
| Design Goal | Description | Application Scenarios and Features |
|--|--|---|
| Embedded Programming Classroom Learning | The robot arm has a built-in ATOM Matrix ESP32 Development Kit, which provides the device with powerful kinematic processing capabilities. Supports expansion of multiple models of development boards. | Users can use the official Arduino driver library to experience the latest robot control functions, and can also expand development boards with different functions to meet various scene applications. |
| High-level language programming teaching | Supports python2/3 driver library and ROS development framework. Users can use high-level development languages to develop the robot arm. | The product supports development boards with multi-serial port applications. Users can use high-level development languages to control the robot arm and verify the algorithm in a virtual environment, which reduces the requirements for the experimental environment and improves the experimental efficiency. |
| Creative Development | The myCobot 280 series robot arm has a built-in robot motion control algorithm independently developed by Elephant Robotics and supports myBlockly drag-and-drop programming. | The open source driver library for robotic arm products provides more than 90+ robot control interfaces, 1:1 restoration of the basic control interface of industrial robots, allowing users to experience programming closer to real industrial robot development and application. |
| Technology innovation and knowledge dissemination | myCobot 280 For Arduino can be used as a platform to showcase the latest scientific and technological achievements in commercial exhibitions, aiming to enhance the public's understanding and interest in science and technology, and promote the commercialization of scientific and technological innovation. | By displaying and demonstrating the latest scientific and technological achievements, increase public participation, promote the popularization of scientific and technological knowledge and the market acceptance of scientific and technological products. |

Product Features



| | |
|--|--|
| <p>Super extension support, unlimited possibilities</p> | <p>Standard UART serial port, compatible with a variety of development board controls. Support Arduino MKRWIFI1010\MEGA2560\UNO and other development version extensions. Provides basic firmware for Mega and MKR development boards, and burns to expand multiple development methods. Supports expansion of multiple development control methods such as Python, ROS, myBlockly, C++, C#, etc. Supports self-matching grippers, suction pumps and other different accessories to achieve more application scenarios. Supports expansion of artificial intelligence kits for robot education and teaching.</p> |
| <p>Unique industrial design, extremely compact</p> | <p>Integrated design, compact overall body structure, net weight of only 860g, very easy to carry. Modular design, few spare parts, low maintenance cost, can be quickly disassembled and replaced, and plug-and-play</p> |
| <p>Supports ROS1 and graphical programming</p> | <p>Supports ROS1 multi-version applications, so that development is no longer restricted. Supports graphical programming software, making robot programming applications within reach.</p> |
| <p>High configuration, convenient development</p> | <p>Contains 6 high-performance servo motors, fast response, small inertia, and smooth rotation. Built-in kinematics forward and inverse solution algorithm, open standard SDK</p> |
| <p>Graphical programming, support for industrial robot software</p> | <p>Using myBlockly visual programming software, easy programming on the palm of your hand, simple and easy to use. Support Arduino + ROS open source system</p> |
| <p>High configuration, with Lego interface</p> | <p>Contains 6 high-performance servo motors, fast response, small inertia, and smooth rotation. The base and the end are equipped with Lego technology parts interface, suitable for the development of various micro embedded devices.</p> |

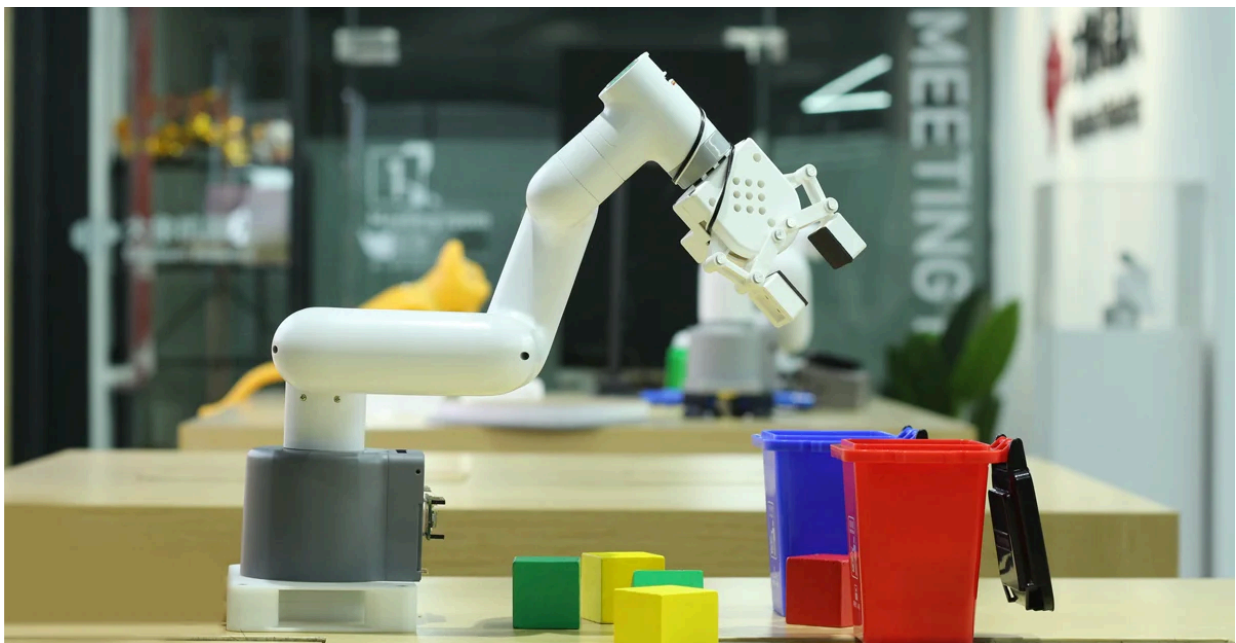
2. Product application



User group

| | |
|---|--|
| <p>Educational institution</p> | <p>The Arduino platform supports a large number of development expansion boards, whether it is basic entry, primary education, or advanced applications, you can find suitable development components. MyBlockly, an image-drag programming software that supports entry-level teaching, puzzle-style programming, and graphical debugging tools, allow novices to easily control the robot arm.</p> |
| <p>Technical developers and engineers</p> | <p>myCobot 280 For Arduino has a built-in robot motion control algorithm independently developed by Elephant Robotics, which supports multiple control modes such as angle, coordinate, potential value, and radian value, greatly meeting various control needs. It can effectively support complex data analysis, algorithm development, and verification activities.</p> |
| <p>Commercial display and public exhibition organizers</p> | <p>myCobot 280 For Arduino can be expanded to a variety of main controls such as PC, industrial computer, and embedded device, suitable for a variety of application scenarios; multiple accessories can be superimposed to complete complex project applications and meet commercial exhibition displays, such as robot application model display, education and teaching set display, and industrial 4.0 application scenario display.</p> |
| <p>Geek development enthusiasts</p> | <p>myCobot 280 For Arduino is based on the Arduino ecosystem and supports a large number of development boards and expansion components, which can greatly meet the development needs of geek enthusiasts.</p> |

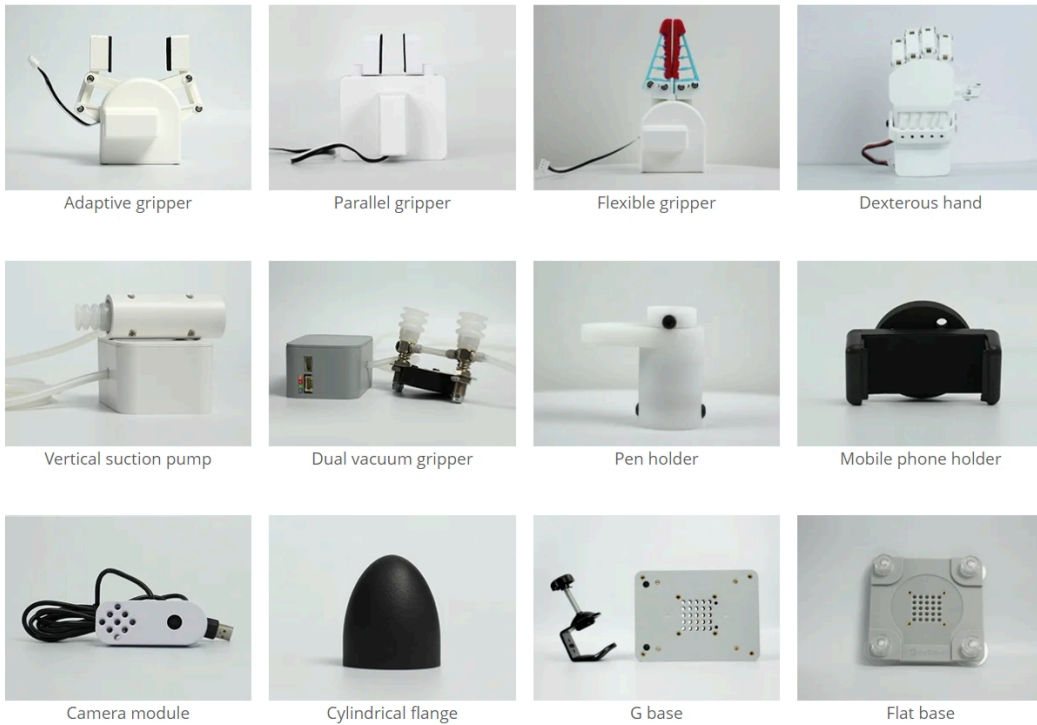
Application scenarios



4.1 First-time self-check


| User groups | Application scenarios | Advantage targets |
|--|--|---|
| Teachers and students in the field of education | <ul style="list-style-type: none"> - STEM education - Robotics projects - Interdisciplinary research projects - Education and research | <ul style="list-style-type: none"> - Improve students' interest in science and technology - Enhance hands-on skills and problem-solving skills - Promote innovative thinking and teamwork - Provide a practical platform for data collection and robotics |
| Makers and technical developers | <ul style="list-style-type: none"> - Prototype development - Experimental research - Algorithm testing and verification - Robot trial teaching | <ul style="list-style-type: none"> - Accelerate research progress - Connect theory and practice - Promote technological innovation - Support multiple programming languages and development environments |
| Business Presentation and Marketing Professionals | <ul style="list-style-type: none"> - Exhibition Display - Technology Demonstration - Brand Promotion | <ul style="list-style-type: none"> - Attract Potential Customers and Investors - Showcase Company's Technical Strength and Innovative Products - Enhance Brand Influence |

3. Supported Extension Development




The mycobot series of robotic arms are extremely valuable in the fields of education and scientific research, especially in Python and ROS (Robot Operating System), two widely used development environments. These environments provide strong support, allowing the mycobot series of products to be widely used in machine learning, artificial intelligence research, complex motion control, and visual processing tasks. At the same time, with dozens of accessories such as adaptive grippers, camera flanges, suction pumps, etc., you can give full play to myCobot's creative ideas. Mobile chassis + robotic arm development, low-cost start of robot collaboration, SLAM mapping, autonomous navigation, ROS development environment, six-axis collaborative robot control, visual recognition, inter-device communication and other development content.


PARAMETERS




Working Radius
280mm




Payload
250g




Arduino




ROS Development




Python




Open-source




Rich Sensors



Technical Support



Practical Course



LEGO Compatible

| | |
|---------------------------|--|
| Python | The robot supports Python and has a complete Python API library. The robot's joint angles, coordinates, grippers, etc. can be controlled through Python. |
| ROS | Supports both ROS1 and ROS2 versions, and provides RVIZ simulation environment support. Allows users to display the robot arm in real time and collect the status information of the robot arm, making myCobot 280 For Arduino suitable for ROS beginners and educational purposes. |
| Arduino | Supports providing the open source program MyCobotBasic sample program. At the same time, users can also modify the open source program according to their own needs. |
| Hardware interface | Including IO, USB, etc., to facilitate the connection of various sensors and actuators. |
| Software library | Provides a wealth of open source libraries and APIs to simplify the development process. |
| myBlockly | It is both a graphical programming software and a visualization tool. Users can drag and drop modules to create programs. This process is very similar to building blocks, which is convenient, fast and easy to use. |

4. Where to buy

If you are interested in purchasing the device, please click on the link below: Taobao:

<https://shop504055678.taobao.com> Shopify: <https://shop.elephantrobotics.com/> AliExpress:

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

[Next Chapter →](#)

Chapter 2 Robot Parameters

In Chapter 1, we explored the product's selling points and design concepts, providing you with a panoramic view of the product's high-level understanding. Now, let's move on to Chapter 2 - Robot Parameters. This chapter will be the key to your understanding of the product's technical details. A detailed understanding of these technical parameters will not only help you fully realize the advancement and practicality of our products, but also ensure that you can use these technologies more effectively to meet your specific needs.



1 Robot Specifications

1.1 Robot Arm Parameters

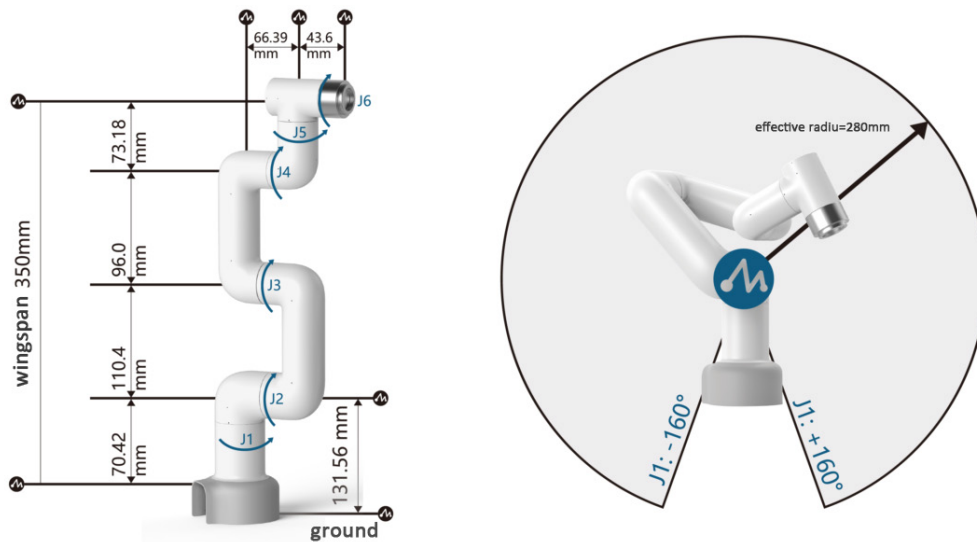
| Indicators | Parameters |
|---------------------|--|
| Name | Little Elephant Collaborative Robot Arm |
| Model | myCobot 280 Arduino 2023 |
| Degrees of Freedom | 6 |
| Payload | 250g |
| Working Radius | 280mm |
| Weight | 860g |
| Power Input | 12V,5A |
| Working Temperature | -5-45°C |
| Communication | USB (depending on the development board) |

2. Control Core Parameters

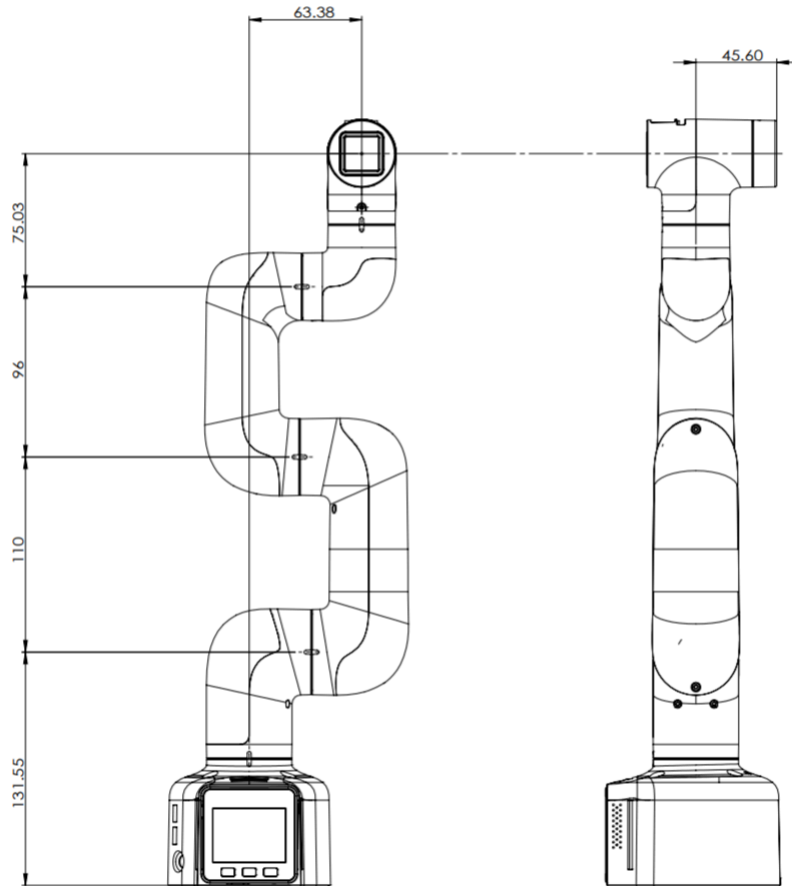
| Indicators | Parameters |
|--------------------|---|
| SOC | None |
| CPU | None |
| Bluetooth/Wireless | None |
| USB | None |
| Display | None |
| HDMI interface | None |
| Custom buttons | None |
| IO interface | Panel IO is a transfer function, depending on the expansion board |

3. Structural size parameters

3.1 Workspace



3.2 Specifications



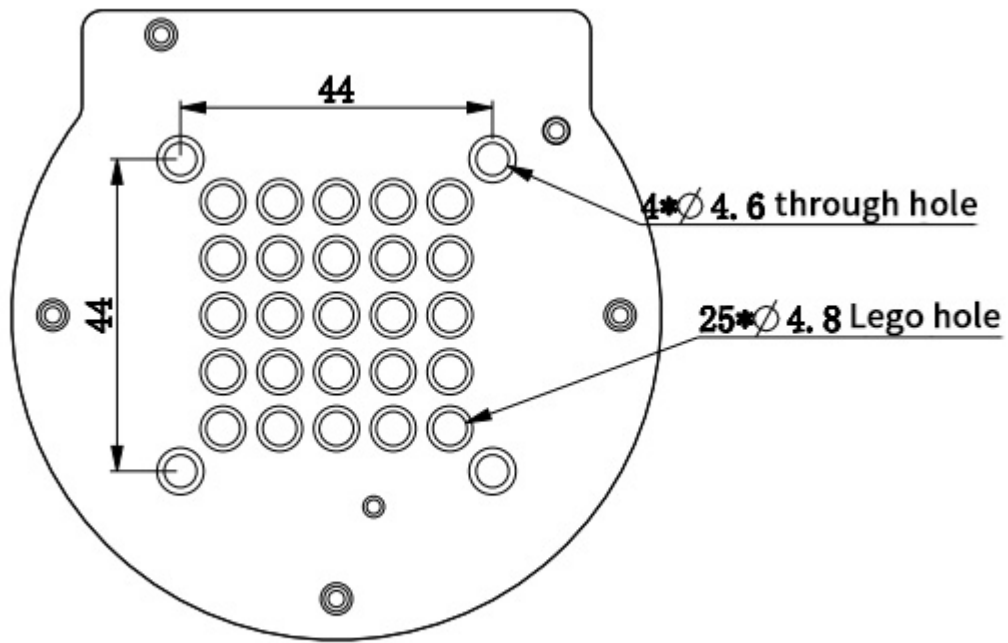
3.3 Joint motion range

Note: △ This joint limit information function is only available on Atom firmware ≥ 7.3 and pymycobot library $\geq 4.0.2$.

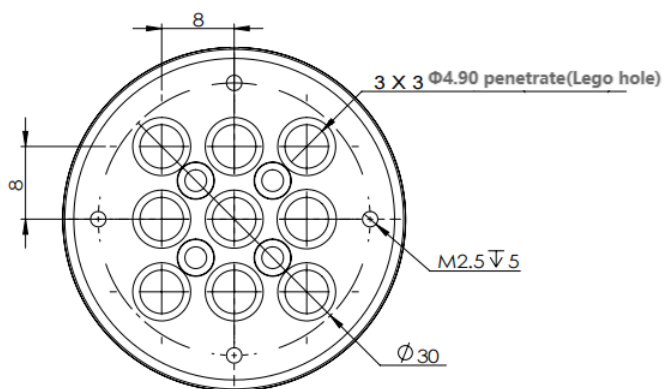
| Joint | Range |
|-------|-------------|
| J1 | -168 ~ +168 |
| J2 | -140 ~ +140 |
| J3 | -150 ~ +150 |
| J4 | -150 ~ +150 |
| J5 | -155 ~ +160 |
| J6 | -180 ~ +180 |

3.4 Hole Installation

- The robot base is mounted with a flange. The base is compatible with both LEGO technology parts and M4 screws.



- The robot end is mounted with a flange. The end of the robot arm is compatible with both LEGO technology parts and screw threaded holes.



4. Electrical characteristic parameters

Electrical interface of the robot arm

4.1 Introduction

Base

- A. The interface on the front of the base is shown in Figure 2.1.8.2-1:

4.1 First-time self-check



Figure 2.1.8.2-1 Front interface of F base

- ① Function interface group 1
- ② Function interface group 2
- ③ Function interface group 3
- ④ Function interface group 4
- B. The interface on the top of the base is shown in Figure 2.1.8.2-2:



Figure 2.1.8.2-2 Interface on the base

- ① Function interface group 5
- C. The interface on the right side of the base is shown in Figure 2.1.8.2-3:



Figure 2.1.8.2-3 Interface on the right side of the base

- ① Power DC interface

Base interface description

Note: The functional interface groups are all 2.54mm DuPont interfaces, and 2.54mm DuPont wires can be used externally.

A. Figure 2.1.8.2-4, Figure 2.1.8.2-5, Figure 2.1.8.2-6 and Figure 2.1.8.2-7 are the signal names of the four expansion interface groups on the front of the base. These interfaces are used to connect different Arduino development boards. Function interface group 1 and function interface group 4 can be used to expand development boards with the same pins as Arduino UNO, such as: Arduino UNO, Arduino MEGA 2560, etc.; Function interface group 2 and function interface group 3 can be used to expand development boards with the same pins as Arduino MKR WiFi 1010;

| | | | | | | | | | | | | | | | | | |
|----|----|---|---|---|---|---|---|---|---|----|----|----|----|-----|------|-----|-----|
| RX | TX | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | GND | AREF | SDA | SCL |
|----|----|---|---|---|---|---|---|---|---|----|----|----|----|-----|------|-----|-----|

Figure 2.1.8.2-4 Function interface group 1

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|----|----|----|----|----|----|----|------|
| 5 | 4 | 3 | 2 | 1 | 0 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | AREF |
|---|---|---|---|---|---|----|----|----|----|----|----|----|------|

Figure 2.1.8.2-5 Functional interface group 2

| | | | | | | | | | | | | | |
|---|---|---|---|----|----|----|----|----|-------|-----|-----|-----|----|
| 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | RESET | GND | VCC | VIN | 5V |
|---|---|---|---|----|----|----|----|----|-------|-----|-----|-----|----|

Figure 2.1.8.2-6 Functional interface group 3

| | | | | | | | | | | | | |
|----|----|----|----|----|----|-----|-----|-----|----|-----|-------|-------|
| A5 | A4 | A3 | A2 | A1 | A0 | VIN | GND | GND | 5V | VCC | RESET | IOREF |
|----|----|----|----|----|----|-----|-----|-----|----|-----|-------|-------|

Figure 2.1.8.2-7 Functional interface group 4

B. Figure 2.1.8.2-8 It is the signal name of the functional interface group on the base. This part of the interface corresponds to each functional interface of the connected Arduino development board.

| | | | | | | | | | | | | | | | |
|---|---|---|---|----|----|----|------|---|----|----|-------|-----|-------|-----|-----|
| 0 | 2 | 4 | 6 | A6 | A4 | A2 | A0 | 8 | 10 | 12 | 14 | SDA | RESET | VIN | GND |
| 1 | 3 | 5 | 7 | A5 | A3 | A1 | AREF | 9 | 11 | 13 | IOREF | SCL | VCC | VIN | |

4.1 First-time self-check

Figure 2.1.8.2-8 Functional interface group 5

C. Power DC interface: Use a DC power socket with an outer diameter of 6.5mm and an inner diameter of 2.0mm; use the 12V 5A DC power adapter provided by the manufacturer to power myCobot 280.

4.2 Electrical interface of the end of the robot

Introduction to the end of the robot

- A. The end of the robot is shown in Figure 2.1.8.2-9 and Figure 2.1.8.2-10:



Figure 2.1.8.2-9 End of the robot

- ① Servo interface
- ② Atom



Figure 2.1.8.2-10 End of the robot

- ① Function interface group 6
- ② Grove
- ③ Type C

Terminal interface description

- A. The definitions of each interface of functional interface group 6 are shown in Table 2.1.8.2-1:

Table 2.1.8.2-1

| Label | Signal name | Type | Function | Remarks |
|-------|-------------|------|----------|---------|
| 5V | 5V | P | DC 5V | |
| GND | GND | P | GND | |
| 3V3 | 3V3 | P | DC 3.3V | |
| G22 | G22 | I/O | GPIO22 | |
| G19 | G19 | I/O | GPIO19 | |
| G23 | G23 | I/O | GPIO23 | |
| G33 | G33 | I/O | GPIO33 | |

Note:

1. I: Input only.
2. I/O: This function signal includes input and output combination.
3. When the tube angle is set as the output end, it will output a voltage of 3.3V.
4. The pull current of a single tube angle decreases with the increase of the number of pins, from about 40mA to 29mA.
5. If a GPIO is set to output mode, it outputs a high-level signal, and the circuit connection is shown in Figure 2.1.8.2-11, and the LED light will light up.

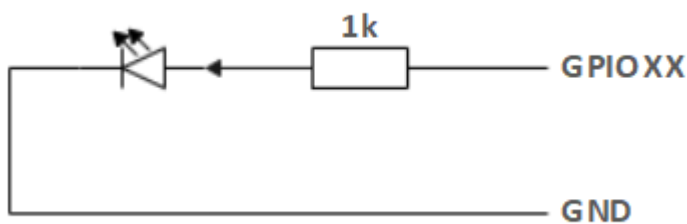


Figure 2.1.8.2-11

- B. Type C interface: can be used to connect and communicate with the PC end and update the firmware.
- C. Grove: Definition as shown in Figure 2.1.8.2-12

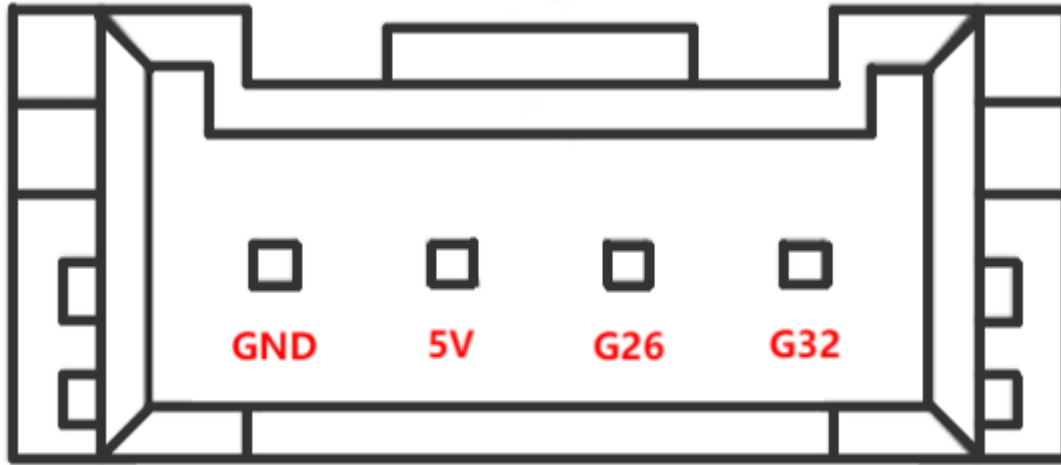
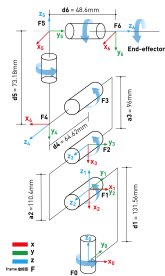


Figure 2.1.8.2-12 Grove

- D. Servo interface: used for end extension gripper, currently supports matching adaptive gripper.
- E. Atom: used for 5X5 RGB LED (G27) Display and key functions (G39)

5.DH parameters



SDH parameter table:

```
mycobot280:: 6 axis, RRRRRR, stdDH, slowRNE
```

| j | theta | d | a | alpha | offset |
|---|-------|--------|--------|---------|---------|
| 1 | q1 | 131.22 | 0 | 1.5708 | 0 |
| 2 | q2 | 0 | -110.4 | 0 | -1.5708 |
| 3 | q3 | 0 | -96 | 0 | 0 |
| 4 | q4 | 63.4 | 0 | 1.5708 | -1.5708 |
| 5 | q5 | 75.05 | 0 | -1.5708 | 1.5708 |
| 6 | q6 | 45.6 | 0 | 0 | 0 |

Chapter 3 User Notice

Chapter 3 User Notice is a must-read for every user to ensure that the user can achieve the established safety standards and efficiency when using the product. This chapter not only provides basic information on product use, transportation, storage and maintenance to ensure safe operation and maximized efficiency of the product, but also details the liability issues for product failure or damage that may result from failure to comply with these guidelines.

1. Safety Instructions

Introduction

This chapter details general safety information for personnel who perform installation, maintenance and repair work on the Elephant Robot. Please fully read and understand the contents and precautions of this chapter before handling, installing and using it.

Hazard Identification

The safety of collaborative robots is based on the premise of correctly configuring and using the robots. Moreover, even if all safety instructions are followed, injuries or damage to the operator may still occur. Therefore, it is very important to understand the safety hazards of robot use, which is conducive to preventing them before they happen.

Tables 1-1 to 3 below are common safety hazards that may exist when using robots:

Table 1-1 Dangerous safety hazards


| |
|---|
|  |
| 1 Personal injury or robot damage caused by incorrect operation during robot handling. |
| 2 Failure to fix the robot as required, such as lack of screws or screws not tightened, insufficient base locking capacity to stably support the robot for high-speed movement, etc., may cause the robot to fall over, resulting in personal injury or robot damage. |
| 3 Failure to correctly configure the robot's safety functions, or insufficient installation of safety protection tools, etc., may cause the robot's safety functions to fail to function, thus causing danger. |

Table 1-2 Warning-level safety hazards



| |
|---|
|  警告 |
| 1 Do not stay in the robot's motion range when debugging the program. Improper safety configuration may not be able to avoid collisions that may cause personal injury. |
| 2 The connection between the robot and other equipment may cause new hazards, and a comprehensive risk assessment needs to be re-performed. |
| 3 Scratches and punctures caused by sharp surfaces such as other equipment or the robot's end effector in the working environment. |
| 4 The robot is a precision machine, and stepping on it may cause damage to the robot. |
| 5 Failure to clamp the object in place or not removing the clamped object before turning off the robot power or air source (not confirming whether the end effector is firm and the clamped object falls due to power loss) may cause dangers, such as damage to the end effector and injuries to people. |
| 6 The robot may move unexpectedly. Do not stand under any axis of the robot under any circumstances! |
| 7 The robot is a precision machine. If it is not placed stably during transportation, it may cause vibration, which may cause damage to the robot's internal components. |
| 8 Compared with ordinary mechanical equipment, the robot has more degrees of freedom and a larger range of motion. Failure to meet the range of motion may cause unexpected collisions. |

Table 1-3 Safety hazards that may cause electric shock

| |
|---|
|  小心触电 |
| 1 Using non-original cables may cause unknown dangers. |
| 2 Electrical equipment in contact with liquid may cause leakage. |
| 3 There may be a risk of electric shock when the electrical connection is incorrect. |
| 4 Always turn off the power of the controller and related devices and unplug the power plug before replacing. If the work is carried out in the power-on state, it may cause electric shock or malfunction. |

2. Safety precautions

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

- The robot arm is a live device. Non-professionals are not allowed to change the circuit at will, otherwise it is easy to cause damage to the equipment or personal injury.

4.1 First-time self-check

- When operating the robot arm, local laws and regulations should be strictly observed. The safety precautions and "Danger", "Warning" and "Caution" items described in the manual are only used as supplements to local safety regulations.
- Please use the robot arm within the specified environmental range. Exceeding the robot arm specifications and load conditions will shorten the product life or even damage the equipment.
- Personnel responsible for installing, operating and maintaining the myCobot robot arm must first undergo rigorous training, understand various safety precautions, and master the correct operation and maintenance methods before operating and maintaining the robot.
- Do not use this product in a humid environment for a long time. This product is a precision electronic component. Long-term operation in a humid environment will damage the device.
- Do not use this device in a high temperature environment. The outer surface of this device is made of photosensitive resin as raw material. Higher temperatures will damage the outer shell of the device and cause equipment failure.
- Highly corrosive cleaning is not suitable for cleaning the robot arm, and anodized parts are not suitable for immersion cleaning.
- Do not use this product without a base installed to avoid damage to the device or accidents. This product should be used in a fixed environment with no obstacles around.
- Do not use other power adapters for power supply. If the device is damaged due to the use of an adapter that does not meet the standards, it will not be covered by after-sales service.
- Do not disassemble, disassemble, or unscrew the screws or casing of the robot arm. If disassembled, warranty service cannot be provided.
- Personnel who have not received professional training are not allowed to repair faulty products or disassemble the robot arm without authorization. If the product fails, please contact myCobot technical support engineers in time.
- 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 someone during use, and the device must be turned off in time when the operation is completed.
- When the robot is in motion, do not put your hand into the range of motion of the robot arm to avoid injury.
- It is strictly forbidden to change or remove and modify the nameplate, instructions, icons and markings of the robot arm and related equipment.
- Please be careful during transportation and installation. Please place the robot gently and correctly in the direction of the arrow according to the instructions on the packaging box, otherwise it is easy to damage the machine.
- **Do not burn other product drivers without authorization, or use unofficial recommended methods to burn firmware.** If the device is damaged due to the user's personal burning of other firmware, it will not be covered by after-sales service.

If you have any questions or suggestions about the contents of this manual, please log in to the official website of Elephant Robotics to submit relevant information:

<https://www.elephantrobotics.com>

Do not use the robot arm for the following purposes:

- Medical and life-critical applications.
- In an environment that may cause an explosion.
- Direct use without risk assessment.

4.1 First-time self-check

- Use with insufficient safety function level.
 - Use that does not meet the robot performance parameters.
-

3. Transportation and storage

Packing and packaging

When packing and packaging the robot product, please make sure to use packaging materials and boxes designed for it. These materials can provide sufficient cushioning and support to prevent impact and vibration during transportation. Be sure to check that all parts are properly fixed to avoid looseness and damage. For fragile or sensitive parts, additional anti-vibration protection materials should be used for reinforcement. Finally, make sure that the outside of the packaging box is marked with clear handling and warning labels to indicate the correct handling method and storage direction.

Logistics and Transportation

During transportation, the robot product should be transported in the original packaging. During transportation, it should be ensured that the robot product is stable as a whole in the packaging box and protected by appropriate measures. During transportation and long-term storage, the ambient temperature should be maintained in the range of -20 to +55°C, and the humidity should be ≤95% without condensation.

Because the robot is a precision machine, the robot product should be handled with care when it is removed from the packaging. During transportation, if it is not placed stably, it may cause vibration and damage the internal parts of the robot.

Equipment Storage

After transportation, the original packaging should be properly stored in a dry place, the ambient temperature should be kept within the range of -20 to +55°C, the humidity should be ≤95% and without condensation, in preparation for future repackaging and transportation needs. Do not stack other items on the original packaging box of the robot arm to prevent the packaging box from deforming and the robot arm from being damaged.

4. Maintenance and Care

As a robot manufacturer, we value ensuring that our customers can properly and safely maintain and upgrade their robot equipment. To this end, we provide the following detailed maintenance and care guide, including common maintenance items and parts for repairing or upgrading hardware. Please read carefully:

Common Maintenance Items and Recommended Cycles

| Maintenance Item | Description | Recommended Cycle |
|---------------------------------------|---|---------------------------------------|
| Visual Inspection | Check the robot for obvious damage, foreign matter accumulation or wear. | Daily |
| Structural Cleaning | Clean the robot structural parts with a clean, dry cloth. Avoid moisture and aggressive cleaning agents. | Daily |
| Fastener inspection | Check and tighten all bolts and connections. | Daily |
| Lubrication | Lubricate joints and moving parts with lubricants recommended by the manufacturer. | Every 3 months |
| Cable and wiring inspection | Check cables and wiring to ensure they are not damaged or worn. | Monthly |
| Electrical connection inspection | Ensure all electrical connections are secure and free of corrosion or damage. | Monthly |
| Software update | Check and update control software and applications. | Every update |
| Software data backup | Regularly back up critical software configurations and data. | Quarterly |
| Firmware update | Regularly check and update firmware to get the latest features and security patches. | Every update |
| Sensor and device inspection | Check sensors and other critical devices to ensure they are functioning properly. | Monthly |
| Emergency stop function test | Regularly test the emergency stop function to ensure its reliability. | Monthly |
| Environmental Condition Monitoring | Monitor the temperature, humidity, dust, etc. of the working environment to ensure that the robot's operating specifications are met. | Continuous Monitoring |
| Safety Configuration Review | Regularly check and confirm the robot's safety configuration, such as speed limit and working range settings. | Monthly |
| Preventive Maintenance Plan Execution | Perform regular inspections and maintenance according to the manufacturer's maintenance plan. | According to the manufacturer's guide |

Guidelines for Independently Changing Robot Hardware

We understand that customers may have the need to upgrade or repair robot hardware on their own. Before performing any upgrade operations, please be sure to read the relevant parameters of the product in detail and confirm with our official personnel whether such operations are allowed. Operations without official permission may cause product failure and are not covered by the warranty.

Material Requirements

4.1 First-time self-check

Officially manufactured or recommended materials: All accessories and components required for repairs and upgrades must be officially 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 repair and upgrade materials through our official channels. This ensures the quality and compatibility of the accessories.

Repair or Upgrade Process

Customer Self-Repair: The customer is responsible for completing the repair work. We will provide detailed repair instructions and manuals to guide customers through the repair steps.

Follow Official Instructions: Repair operations should strictly follow the official instructions we provide. Any deviation from the official instructions may cause damage to the device.

Liability and Warranty Policy

- Division of Responsibilities: Manufacturer: Provide official instructions for repairs and upgrades# Chapter 3 User Notice

Chapter 3 User Notice is a must-read for every user to ensure that the user can achieve the established safety standards and efficiency when using the product. This chapter not only provides basic information on product use, transportation, storage and maintenance to ensure safe operation and maximized efficiency of the product, but also details the liability issues for product failure or damage that may result from failure to comply with these guidelines.

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

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

Chapter 4 First Use

1. Product Standard List

1.1 Product List Image

Thank you for choosing the Elephant Robot myCobot 280 For Arduino Robotic Arm. This chapter is designed to help you easily get started with the Elephant Robotic product and enjoy every wonderful moment brought by the product.



1.2 Product Standard List Comparison Table

| Serial Number | Product |
|---------------|---|
| 1 | myCobot Robotic Arm (Model myCobot 280 For Arduino) |
| 2 | myCobot Robotic Arm-Product Brochure |
| 3 | myCobot Robotic Arm-Matching Power Cord |
| 4 | USB-Type C |
| 5 | Several Dupont Wires |

Note: After the packaging box arrives, please confirm that the robot packaging is intact. If there is any damage, please contact the logistics company and the supplier in your area in time. After unpacking, please check the actual items in the box according to the list of items.

2. Product unpacking guide

2.1 Product unpacking graphic guide

Why do you need to disassemble the product according to the steps

In this section, we strongly recommend disassembling the product according to the specified steps. This not only helps to ensure that the product is not damaged during transportation, but also minimizes the risk of unexpected failures. Please read the following graphic guide carefully to ensure the safety of your product during the unpacking process.

- **1** Check the packaging box for damage. If there is any damage or missing accessories, please contact the logistics company and the supplier in your area in time.
- **2** Open the box and take out the product brochure, sponge packaging cover, myCobot robot arm, matching power supply, flat base and accessory bag.
- **3** Make sure each step is completed before proceeding to the next step to prevent unnecessary damage or omissions.

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

3. Startup Inspection Guide

3.1 Structural Installation and Fixing

During the movement of the **robotic arm**, if the **bottom surface of myCobot is not connected to the desktop or other bottom surface**, myCobot will still **shake or overturn**.

There are three common ways to fix the robotic arm:

- 1) Use Lego keys to fix it on a base with a Lego interface

4.1 First-time self-check

We sell two types of bases: flat suction cup base and G-type clamp base



Flat base Applicable model: myCobot 280

- Install suction cups at the four corners of the base and tighten them.
- Use the included Lego technology parts to connect the flat base and the bottom of the robotic arm.
- Fix the four suction cups on a flat and smooth surface before starting to use.

4.1 First-time self-check

- Tips: You can add a small amount of non-conductive liquid under the suction cup to fill the gap between the suction cup and the desktop to achieve the best adsorption effect.



G-type base Applicable models: myCobot 280 series, myPalletizer 260



- Use the G-clip to fix the base to the edge of the table

4.1 First-time self-check

- Use the included Lego tech parts to connect the base and the bottom of the robot arm
-
- Make sure it is stable before starting to use



2 myCobot base screw hole connection

The robot needs to be fixed on a solid base before it can be used normally. Base weight requirements: fixed base, or mobile base.

Please make sure that there are corresponding threaded holes on the fixed base before installation.

Before formal installation, please confirm:

- The installation environment meets the requirements of the above "Working Environment and Conditions" table.
- The installation location is not less than the robot's working range, and there is enough space for installation, use, maintenance, and repair.
- Place the base in a suitable position.
- The installation-related tools are ready, such as screws, wrenches, etc.

After confirming the above content, please move the robot to the base installation table, adjust the robot position, and align the robot base fixing holes with the holes on the base installation table. After aligning the holes, align the screws with the holes and tighten them.

- Note: When adjusting the robot position on the base installation table, please try to avoid pushing and pulling the robot directly on the base installation table to avoid scratches. When manually moving the robot, please try to avoid applying external force to the fragile parts of the robot body to avoid unnecessary damage to the robot.

4. Power on and preliminary inspection

Power on the robot

Before operation, please make sure you have read and followed the contents of **Chapter 1 Safety Instructions** to ensure safe operation. At the same time, connect the power adapter to the robot arm and fix the base of the robot arm on the table. The connection method is shown in Figure 3-1:



Figure 3-1

myCobot **must be powered by an external power supply** to provide sufficient power:

- Rated voltage: 12V
- Rated current: 3-5A
- Plug type: DC 5.5mm x 2.1

Note that **it cannot be powered by just the TypeC plugged into the M5Stack-basic**. Use the official power adapter to avoid damage to the robot arm.

Connect external devices

- Connect to the computer via USB

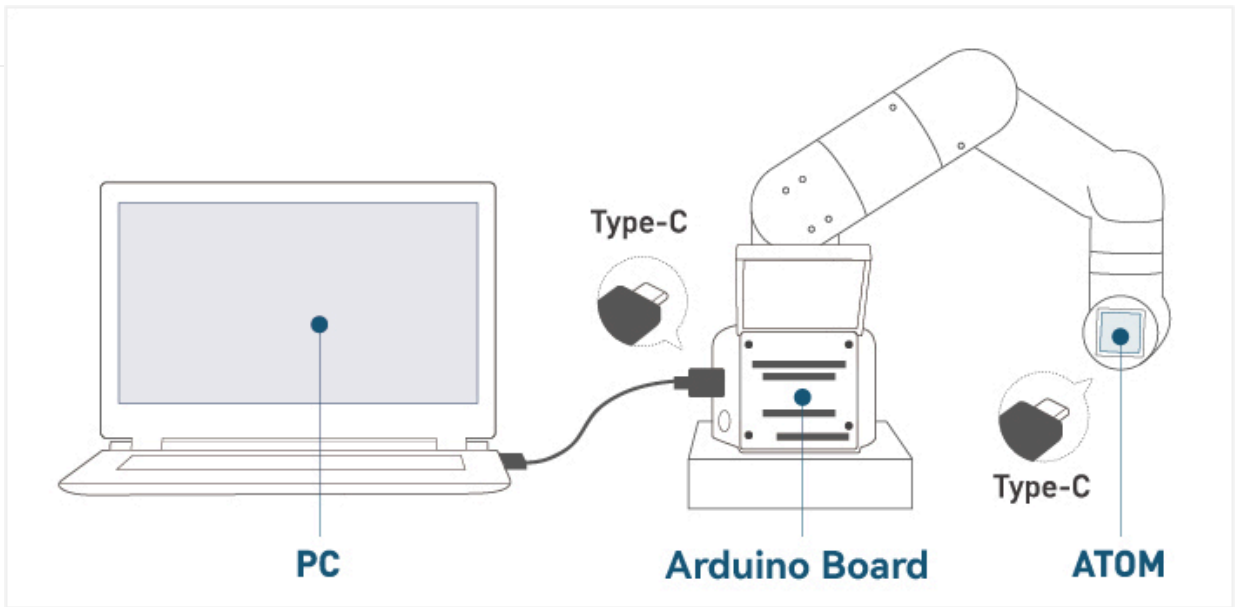


Figure 3-2 USB interface connection diagram

Robot working status detection

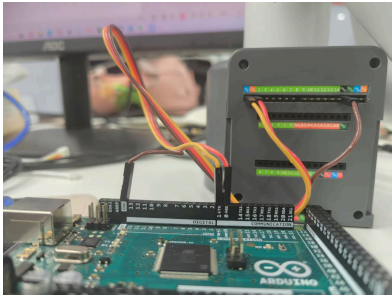
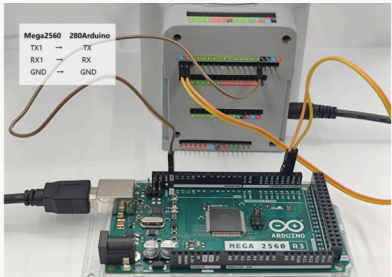
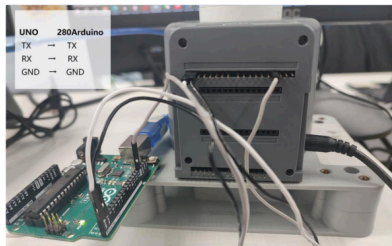
Unboxing video



Hardware connection

Note: The current 280AR serial port baud rate is changed to 1000000. The wiring of different development boards is different:

4.1 First-time self-check

| Development board type | Baud rate | Control method | Wiring method | Wiring diagram |
|------------------------|-----------|------------------|-------------------------------------|---|
| mega2560 | 1000000 | python/MyBlockly | TX0-->RX、 RX0-->TX、 GND-->GND |  |
| mega2560 | 1000000 | Arduino IDE | TX1-->RX、 RX1-->TX、 GND-->GND |  |
| UNO | 1000000 | Arduino IDE | tx-->tx RX-->RX、 GND-->GND |  |
| mkrwifi1010 | 1000000 | None | None | None |

How to perform zero calibration

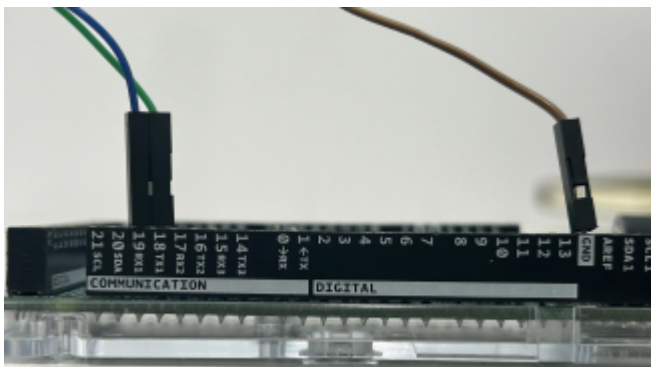
Note: Please strictly follow the steps below

1. Power on the robot arm first (if the green light is on, it means the robot arm is powered on)

4.1 First-time self-check

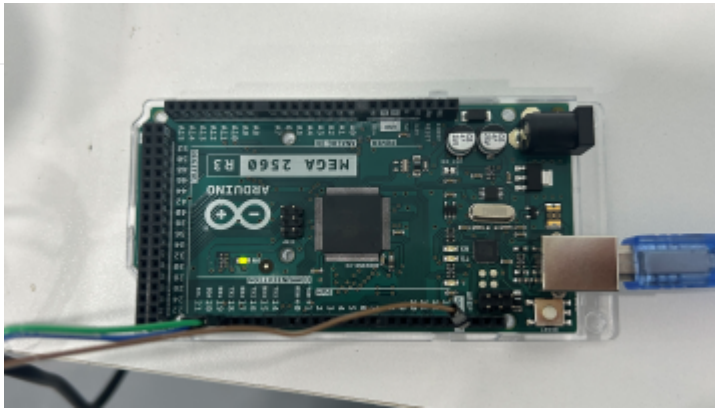


1. Connect the Arduino board (the connection position is shown in the figure below)



1. Connect the serial port to the computer (if the green light is on, it means the computer is successfully connected)

4.1 First-time self-check



After connecting, you can follow the steps below to view the connection port number

Click this computer --> Right-click to manage --> Click Device Manager --> Click Port



Calibration zero position example

4.1 First-time self-check

```
from pymycobot import MyCobot280

from pymycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables
import time

# The above needs to be written at the beginning of the code, meaning to import the project package

# MyCobot class initialization requires two parameters: serial port and baud rate

# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# windows: "COM6"
# The second is the baud rate:
# M5 version is: 1000000
# The following is as follows:
# mycobot-M5:
# linux:
# mc = MyCobot("/dev/ttyUSB0", 1000000)
# windows:
# mc = MyCobot("COM6", 1000000)
# mycobot-raspi:
# mc = MyCobot(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot object
# The following is the object code for the windows version
mc = MyCobot280("COM6", 1000000)
# Power off the robot and relax the joints
time.sleep(2)
mc.release_all_servos()
# After relaxing the joints, bend the robot to align the zero position scale of the robot
time.sleep(2)

# Calibrate the robot joints one by one
for i in range(1, 7):
    mc.set_servo_calibration(i)
    time.sleep(1)

# Power on the robot to lock the robot
mc.power_on()
time.sleep(2)

# Print all joint angles
print(mc.get_angles())
time.sleep(1)
```

Example diagram of robot arm calibration zero position (alignment of adjacent joint zero position scale lines is the joint zero position)

4.1 First-time self-check



Example diagram of robot arm joint zero position



5. Common Problem Solving

This section aims to help users solve common problems encountered during use, covering hardware, software, accessories, and how to self-check for the first time. If you encounter problems while using the robot arm, please read the contents of this section first to find solutions. If the listed problems cannot help you solve and you have more after-sales questions to consult, please add the after-sales butler WeChat.

[First-time self-check](#)

[Common software problems and solutions](#)

[Common hardware problems and solutions](#)

[Common accessories problems and solutions](#)

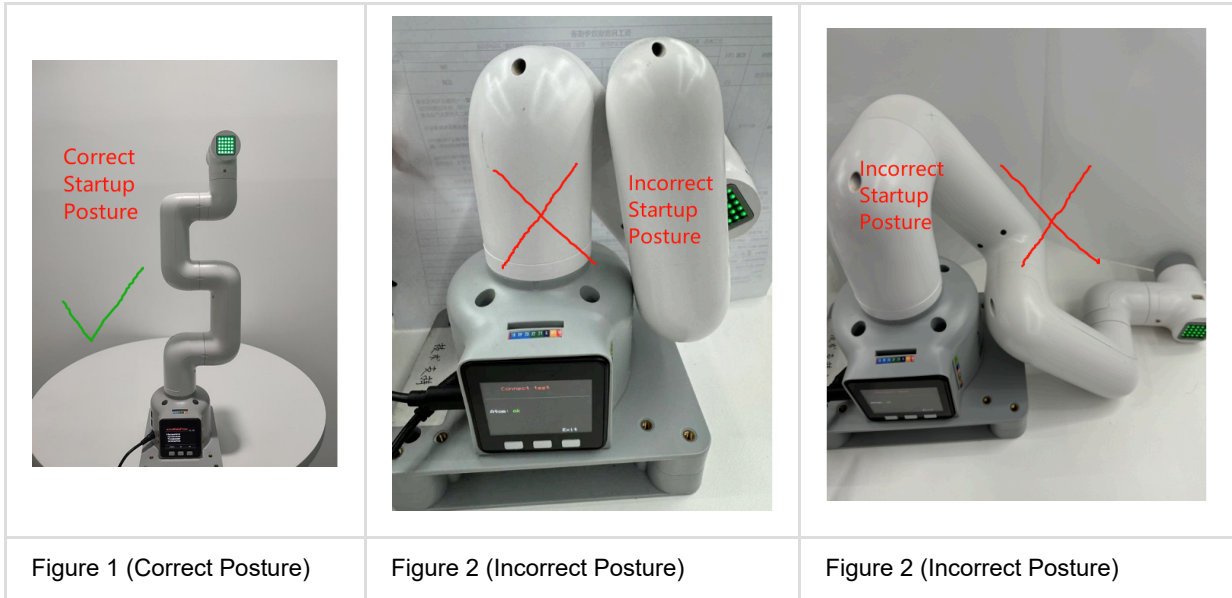
[Other problems and solutions](#)

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

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

First-time self-check- Machine Joint Function Verification

Note: When starting the robot arm, please be careful not to let the robot arm be in a curled-up or touching position between joints. It is recommended that the robot arm posture should be as shown in Figure 1 below when starting. Figures 2 and 3 are both incorrect starting postures:



Joint control method steps

1. Make hardware connections

- Hardware connections for AR series machines:

Make sure the AR series robot is connected to the power adapter and USB data cable.

2. Install and configure the software environment

You need to prepare a computer to use the M5 version of the machine. Install python, pymycobot library and USB serial port driver on the computer. For details, please refer to the environment configuration section of gitbook.

3.USB communication example

Please use myblockly or python source code examples to verify the joint motion of the robotic arm.

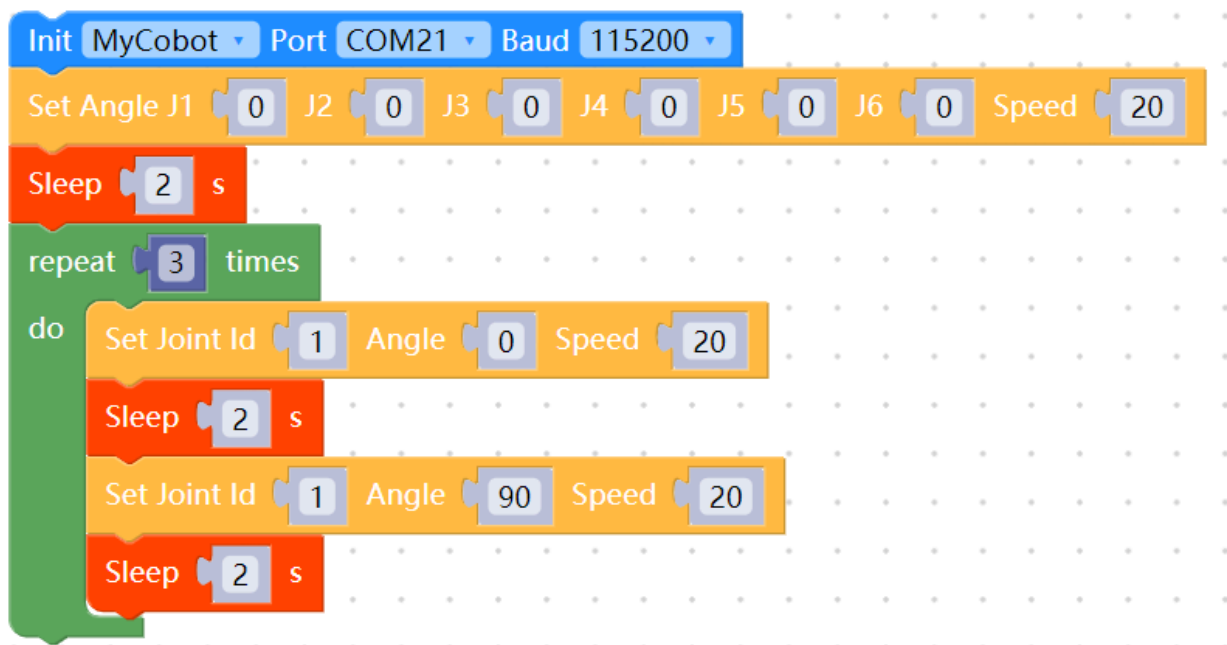
Pay special attention to the need to select the corresponding serial port and baud rate when using the USB serial port opening method so that the robot arm can communicate with the computer normally and thus control the robot arm normally:

4.1 First-time self-check

| Machine model | Serial port number | Baud rate |
|----------------|------------------------------|-----------|
| 260 M5 | Win: COM; Linux: /dev/ttyUSB | 115200 |
| 270 M5 | Win: COM; Linux: /dev/ttyUSB | 115200 |
| 280 M5 | Win: COM; Linux: /dev/ttyUSB | 115200 |
| 280 AR | Win: COM; Linux: /dev/ttyUSB | 1000000 |
| 280 PI | /dev/ttyAMA0 | 1000000 |
| 280 JetsonNano | /dev/ttyTHS1 | 115200 |
| 320 M5 | Win: COM; Linux: /dev/ttyUSB | 115200 |
| 320 PI | /dev/ttyAMA0 | 115200 |
| 280 PI | /dev/ttyAMA0 | 1000000 |
| 270 PI | /dev/ttyAMA0 | 1000000 |

Note: Regarding the selection of the COM port of the AR series machine, it is necessary to make a real-time selection based on the port number recognized by the current personal computer, because the COM port number recognized by each person's computer may be different and not fixed. The specific selection scheme can be viewed in this document. "Q: Why is the connection rejected when selecting a certain COM port? Or how to find the corresponding COM port?" Answer

3.1 Robotic arm joint movement myblockly source code



When you see the robot arm joint 1 cycle 3 times from 0 to 90 degrees, it means that the robot arm joint 1 responds normally. You can try to change the joint ID to test other joints and learn other cases in gitbook step by step or use the robot arm to do various interesting things! It is worth mentioning that if you are not familiar with the code block

development method of myblockly, there is also a relatively quick way to verify the joints: use the myblockly fast movement tool to perform simple joint motion control. For specific usage, please refer to: [Myblockly fast movement tool usage](#)

Quick Move

Joints Control:

 Read Angles

| | | | | | | | |
|----|---|---|---|----|---|---|---|
| J1 | - | 0 | + | J2 | - | 0 | + |
| J3 | - | 0 | + | J4 | - | 0 | + |
| J5 | - | 0 | + | J6 | - | 0 | + |

Coordination Control:

 Read Coords

| | | | | | | | |
|---|---|---|---|----|---|---|---|
| X | - | 0 | + | Rx | - | 0 | + |
| Y | - | 0 | + | Ry | - | 0 | + |
| Z | - | 0 | + | Rz | - | 0 | + |

3.2 Robotic arm joint motion joint python source code

```
#The motion effect is that the robot arm moves around the zero position, and the 1-6 joints move one by one ±20 degrees
import time
from pymycobot.mycobot import MyCobot

if __name__ == "__main__":
    cobot = MyCobot('com22',115200)#Select the corresponding port number and baud rate according to the model
    cobot.set_fresh_mode(1)
    cobot.send_angles([0, 0, 0, 0, 0, 0], 20)
    time.sleep(2)
    print("start")
    for i in range(1,7):
        cobot.send_angle(i, (-30), 20)
        time.sleep(2)
        cobot.send_angle(i, (30), 20)
        time.sleep(2)
        cobot.send_angle(i, (0), 20)
        time.sleep(2)
```

When you see the robot arm around the zero-position posture, 1-6 joints move one by one ± 20 degrees, indicating that joints 1-6 respond normally, you can gradually learn to use other cases in gitbook or use the robot arm to do various interesting things!

If you do not see the corresponding effect when executing the case, please refer to the common problem solutions below. In addition, please make sure that you have checked the following 5 points before contacting technical support personnel:

4.1 First-time self-check

1. Can the robot arm lock normally after power-on? If it cannot be locked, please refer to FQA hardware-related questions: "Q: How to solve the problem that the robot arm cannot be locked after power-on?" for troubleshooting
2. If you have an M5 series robot arm, is your computer connected to the USB port on the side of the M5stack via type-c?
3. If you have an M5 series robot arm, is your screen LCD now stuck in the Atom: ok interface?
4. If you have an M5 series robot arm, the LCD interface shows Atom: no, please refer to "Q: How to solve the problem that the robot arm cannot be locked after power-on?" for troubleshooting
5. Is there any error message when running the code?

Please describe the usage details as detailed as possible. If convenient, please provide an operation video, which will help to quickly analyze and locate the problem. Thank you in advance!

Software Issues

1 myStudio related

Q: What is myStudio?

- A: It is our company's self-developed software. It is a tool for burning or modifying the firmware of the existing robot arm launched by our company.

Q: What is the method to troubleshoot the abnormal download of minirobot, Atom, and PICO firmware?

1. Check whether the network connection is normal. During the firmware download process, you need to connect to the network to download the firmware first.
2. Check whether the line has been connected. The details are as follows: In the M5/Arduino series machine, when burning the Atom firmware, you need to use a USB cable to connect the Atom interface at the end to the USB port of the computer; when the M5 series machine burns the minirobot firmware, use a USB cable to connect the side interface of the M5stack to the USB port of the computer.
3. Select the firmware of the corresponding model, and don't choose the wrong model.
4. Download and install the driver. If it still cannot be recognized after downloading the driver, try to replace the latest [ch340 driver](#). If the port number still cannot be displayed after installing the driver and the system is a win11 model, try [How to install the CH340 driver in Win11 system](#).
5. Try to change a USB cable, USB port or computer to download it to avoid abnormal firmware download caused by the cable not having data transmission function.
6. Uninstall mystudio and reinstall mystudio in a non-C drive location, such as installing mystudio in the D drive. When mystudio is installed in the C drive, the file permissions are relatively strict, and the firmware may not be burned.

Q: What to do if 280Arduino cannot burn firmware?

A: We have verified that the genuine Arduino mega2560 firmware can burn the mystudio firmware normally. The verification video is as follows: <https://drive.google.com/file/d/1C7OhUlgkG05WX0LcPSollQZ0Vj-yJZhS/view?usp=sharing>

For the situation that your arduino mega2560 board cannot burn the firmware, it is recommended that you check the following points:

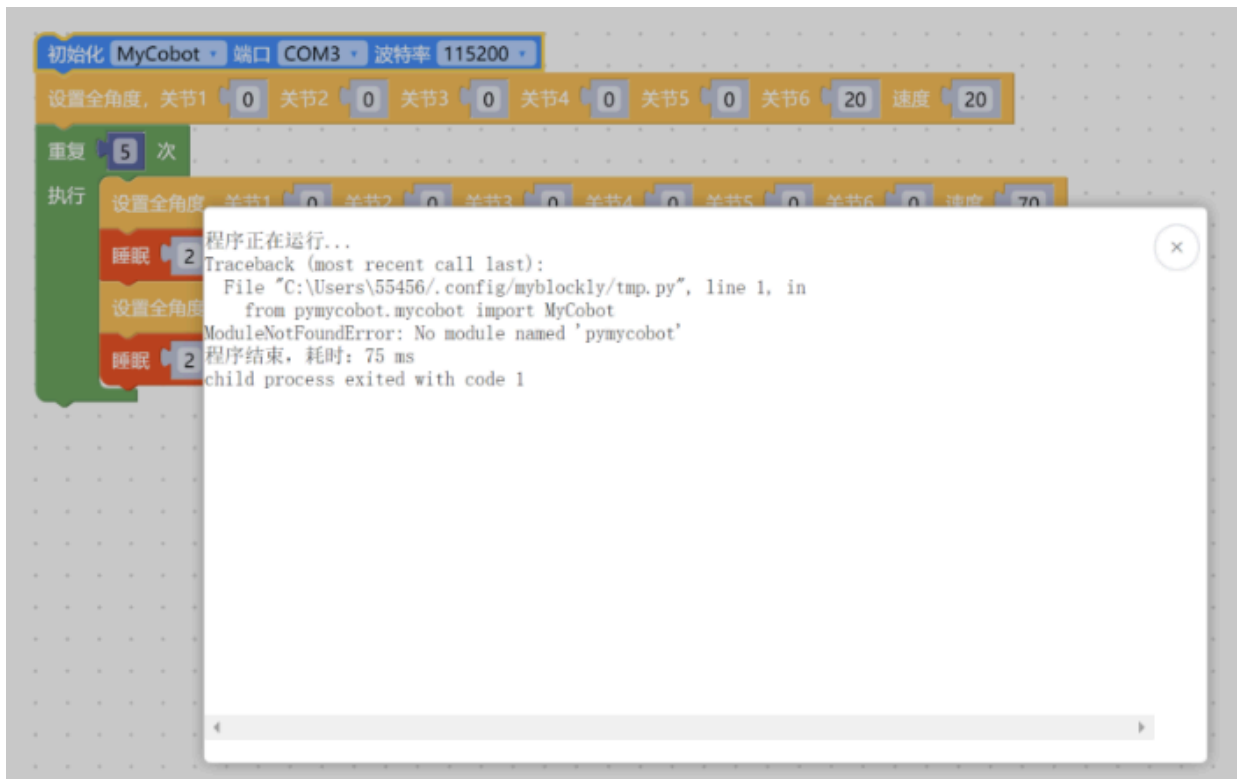
1. Check whether the network connection is normal. During the firmware download process, you need to connect to the network to download the firmware first.
2. When burning the firmware, disconnect the tx and rx pins between mycobot280Arduino and mega2560. There will be a serial port conflict when downloading the firmware, and there may be a situation where the firmware cannot be burned normally. Please connect the tx and rx pins after the burning is successful.
3. Try to unplug and plug the USB port again after entering mystudio and burn it again.
4. Try to press the reset button on the Arduino motherboard when burning the mega firmware.
5. Uninstall mystudio and try to reinstall mystudio in a non-C drive location, such as installing mystudio in the D drive. When mystudio is installed in the C drive, the file permissions are relatively strict, and the firmware may not be burned.

If the above methods do not work, it may be because your Arduino mega2560 motherboard is incompatible with the mycobot280Arduino firmware. You can consider replacing other mega2560 motherboards to download firmware or directly use the Arduino development method of the UNO motherboard to control the robotic arm.

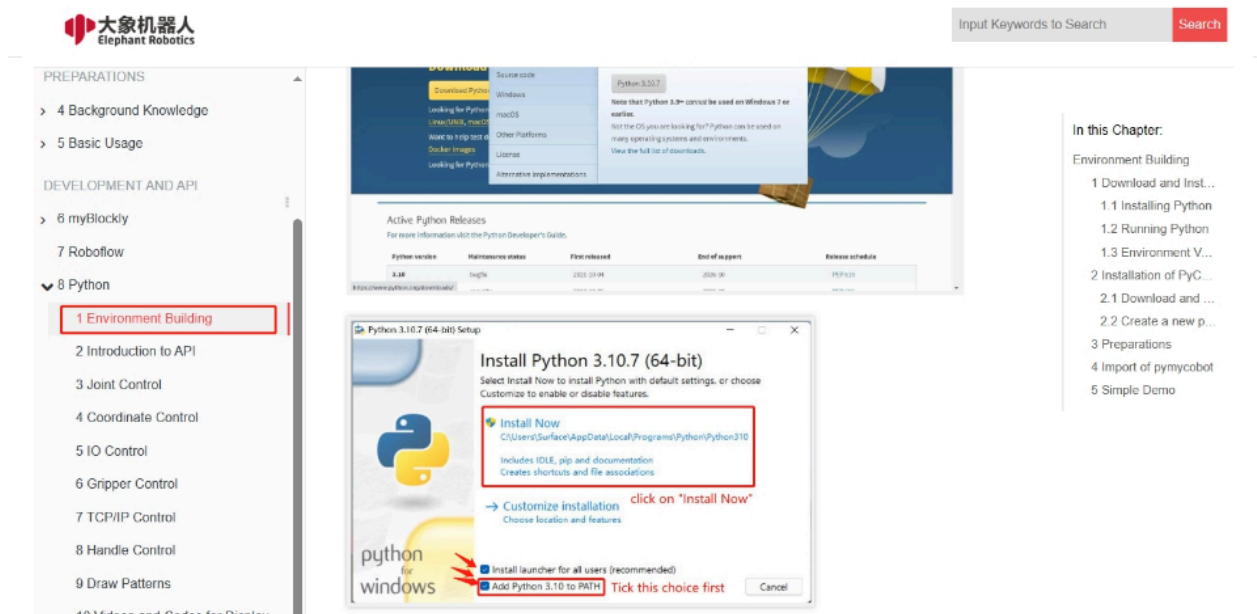
Note: When using the UNO motherboard, you do not need to use mystudio to download the firmware. You can directly use the Arduino IDE development method to burn the corresponding case code to control the robotic arm. Only when using the mega2560 and WIFI1010 motherboards to control the robotic arm do you need to use mystudio to download the corresponding firmware

2 myblockly related

Q: What should I do if I encounter the error message: ModuleNotFoundError: No module named "pymycobot"?



- A: The error message indicates that the pymycobot file is missing. Please refer to the following 3 points for the reasons and solutions: ①If pymycobot is not installed or pymycobot fails, the corresponding solution is to reinstall pymycobot. The command is pip3 install pymycobot --upgrade --user ②If it is an M5 or AR series machine, please note that the "Add Pythonx to PATH" in the figure below is not checked during the installation of python. You need to uninstall python and reinstall python, and check this option



③ If it is an M5 or AR series machine, please confirm whether there are multiple python versions in the PC. It is recommended to uninstall all python versions in the PC and reinstall a version higher than python3.8. Note that there is only one python version higher than python3.8 in the PC. If multiple python versions are required for actual use, please specify the python version used by pymycobot and specify the python version when calling the pymycobot library

Q: When using 280arduino with mega2560 board, myblockly cannot be used to control joints normally, and there is no response to movement and reading. How to deal with it?

- A: Download myblockly1.4.4 version, or when using the latest version of myblockly, please note that when using the development board Arduino MEGA2560 and the building block to control the robot arm, the user needs to manually add the delay building block and then send the command. When using the fast movement tool, you need to open the serial port and send the command after an interval of 1s



Q: How to deal with the fact that myblockly's express movement tool cannot display the real-time angle?

- A: This is generally caused by incorrect selection of device serial port information and pymycobot exception. It is recommended to check according to the "First Use Self-Check" solution in this article. If the robot arm cannot be controlled normally, please try to update pymycobot. The corresponding update solution is to enter the command `pip install pymycobot --upgrade --user` in cmd or terminal. Finally, if it still cannot be controlled normally, please try to update the myblockly software. Please refer to the following link for the update method: <https://drive.google.com/file/d/1yBWzhhSBUYsZPBI7PBdZKRwk3al71Dc7/view?usp=sharing>

Q: The result of running the program shows child process exited with code 1. Is it normal?

- A: This is not an error. All programs have finished running and returned the binary number 1. It means that all have been successfully completed.

Q: How to preset the code block content in myblockly, including the model, baud rate and other information after entering the system, which are all corresponding to the connected model?

- A: Currently, the default model for the first startup in myblockly is mycobot and the baud rate is 115200. There is no way to change the initial baud rate, but you can make and save an initialization json file yourself. Next time you enter myblockly, load this file to get the preset code block. For how to make and save json files, please refer to the following: https://drive.google.com/file/d/1g_dd933TK1tptnisUad4PBfwSRsWWFeQ/view?usp=sharing

3 Roboflow Related

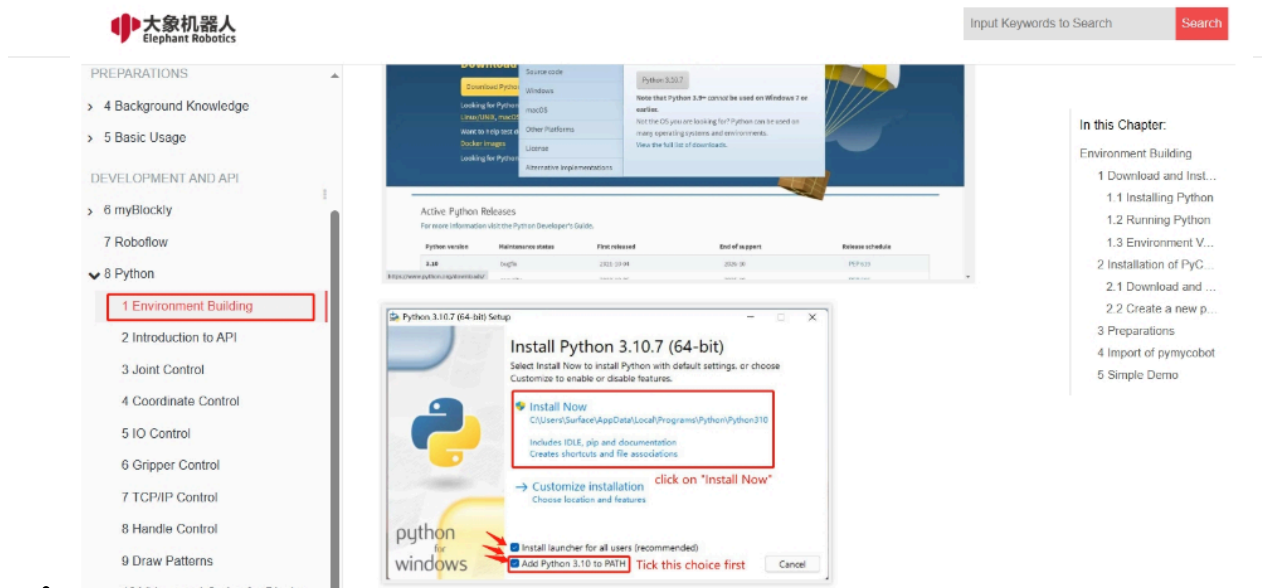
Q: What should I do if I cannot download the Roboflow software or if Roboflow fails to properly control the robot?

- A: Currently, the Roboflow software only supports the 600 Pro and 630 Pro (two professional collaborative robot models). It no longer supports the Mycobot collaborative series or other robot models. For Mycobot-series robots, it is recommended to use MyBlockly, Python, or ROS for control. Notably, MyBlockly is a software with a graphical interface similar to Roboflow. If you prefer visual, block-based programming, MyBlockly is the preferred choice.

4 Python related

Q: The running prompt is missing library filesQ: The error message: ModuleNotFoundError: No module named "pymycobot", how to deal with it?

- A1: Pymycobot is not installed. The corresponding solution is to reinstall pymycobot. The command is `pip3 install pymycobot --upgrade --user`
- A2: During the installation of Python, the "Add Pythonx to PATH" in the figure below was not checked. You need to uninstall Python and reinstall Python, and check this option.



- A3: If it is an M5 or AR series machine, please confirm whether there are multiple Python versions in the PC. It is recommended to uninstall all Python versions in the PC and reinstall a version higher than Python 3.8. Note that there is only one Python version higher than Python 3.8 in the PC. If multiple python versions are required for actual use, please specify the python version used by pymycobot and specify the python version when calling the pymycobot library.
- A4: It is recommended to use version 3.9 of python, as python12 will be incompatible.

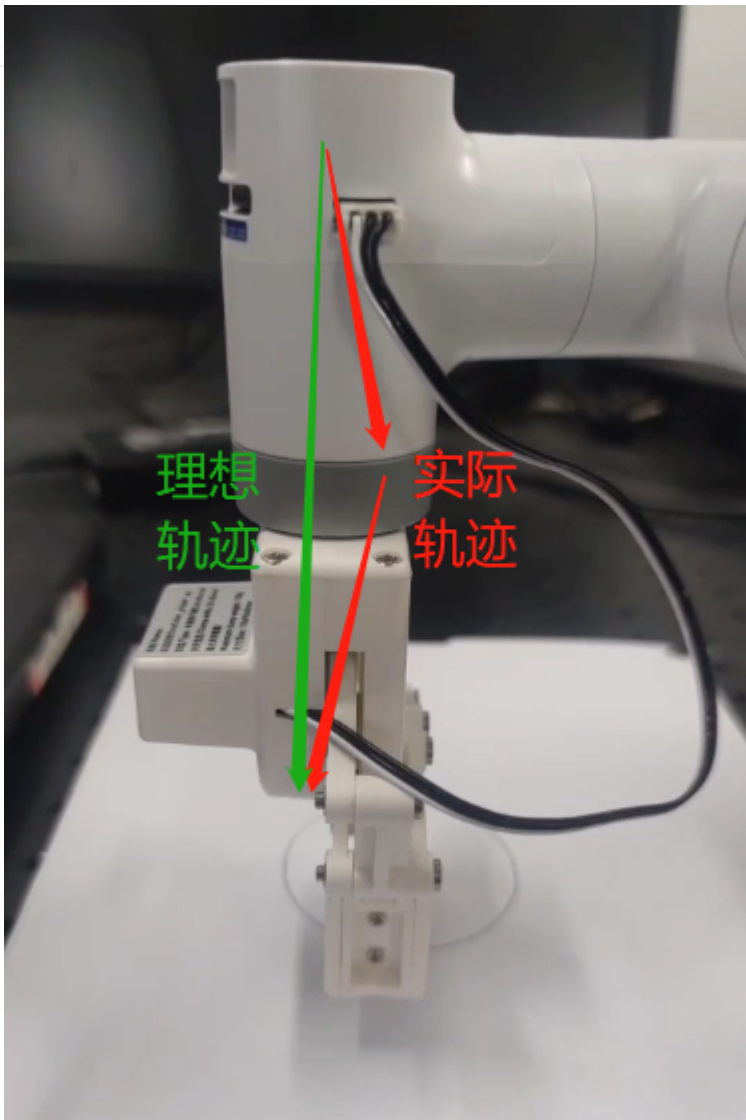
Q: Is there a more popular explanation for the mode in send_coords(coords, speed, mode)?

- A: Linear 1 means that the end of the robot reaches the target position in a straight line. If it cannot go in a straight line due to limitations, structure, etc., the command will not be fully executed; Linear 0 means that the end reaches the target position in an arbitrary posture. Since there is no straight line restriction, it is not easy for the command to not be executed.

Q: What is the difference between the interpolation and refresh modes of set_fresh_mode(mode)?

- A: Interpolation 0 means that many dense points are planned between the starting point and the end point, so as to achieve the effect of controlling the trajectory of the middle segment. How to achieve the effect of program parallelism: Non-interpolation 1 means that there is no planning of the middle segment, and the trajectory cannot be controlled, but the movement will be relatively smooth.

Q: Is it normal for the trajectory not to be straight up and down when only the Z-axis is changed, but the final landing point is adjusted only in the Z-axis? How can the middle trajectory be ensured to be straight?



- Turn on interpolation and walk in a straight line to ensure the trajectory

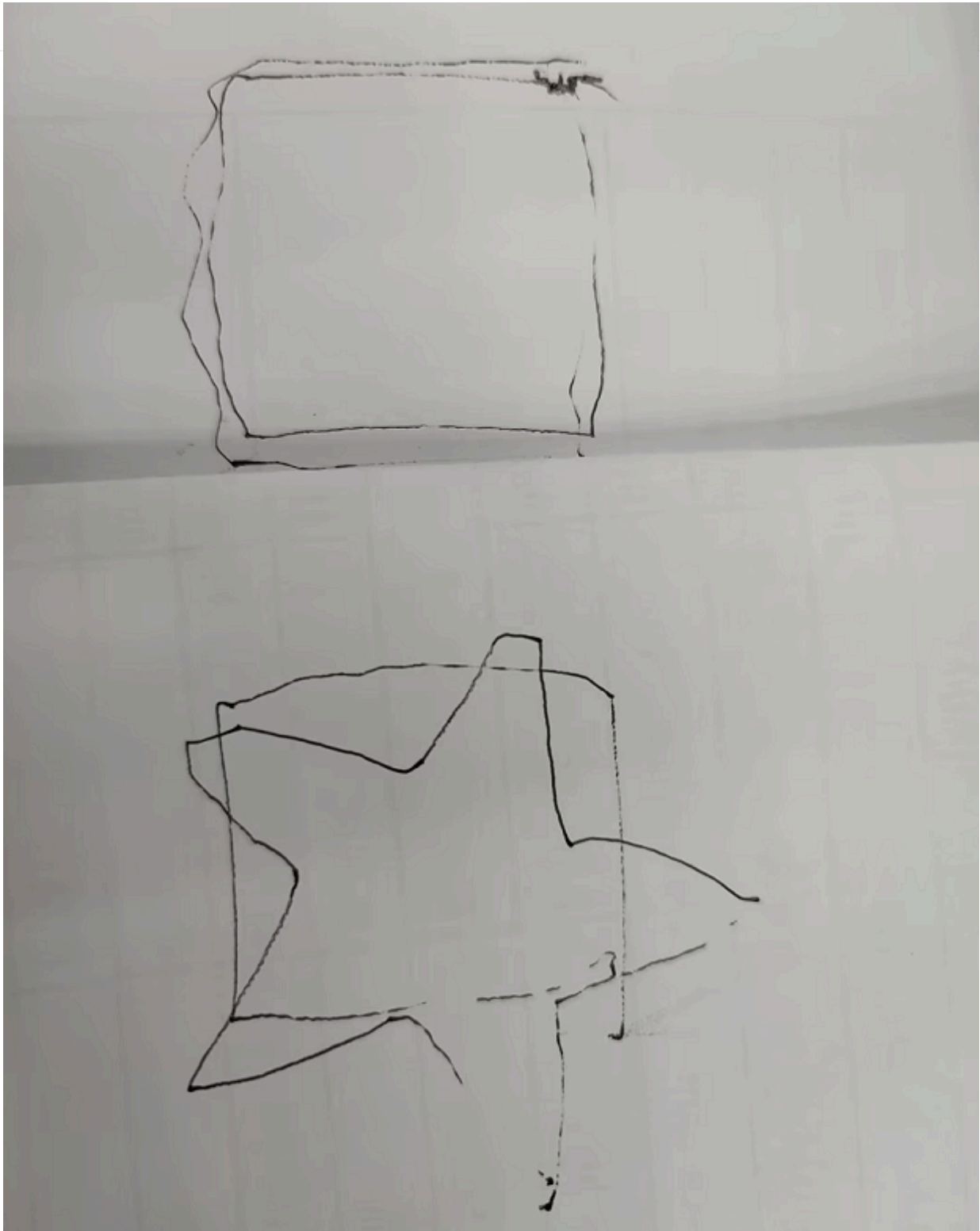
```
set_fresh_mode(0) # Turn on interpolation
send_coords(coords, speed, mode=1) # Walk in a straight line
```

Note that the intelligent planning route set in `send_coords` will only be useful after turning on interpolation. Interpolation means that many dense points are planned between the starting point and the end point, so as to achieve the effect of controlling the trajectory of the middle section. Non-interpolation means that there is no planning of the middle section, and the trajectory cannot be controlled.

Q: What does the return value of `get_error_information()` being -1 mean?

- A: The return value of `get_error_information()` is -1, indicating that communication is not possible. You need to check whether the power adapter and USB cable are connected, and whether the LCD screen stays on the Atom: ok interface. If the line is not connected successfully and does not display ok, communication abnormalities will occur. You need to reconnect and test again.

Q: In the case of drawing with a 280 machine, it is found that the shape trajectory is not very straight. Can it be optimized?



- A1: It is normal to get a deviation in the trajectory when using a signature pen or hard stationery to draw this case. There are two main reasons for this deviation. First, mycobot uses a servo motor, which has a certain accuracy deviation (if it is a machine that has been used for a long time, the deviation of its joints will be greater due to aging of the joints). Second, when using a hard pen to draw, the contact distance with the desktop is relatively demanding. If the distance is too high, the trajectory is prone to interruption. If the distance is too low, the pen tip resistance will be too large and the drawing effect is not ideal. It is currently recommended to use soft stationery for drawing, such as brushes and other tools, which will help improve the drawing effect.

4.1 First-time self-check

- A2: In addition, you can change the motion mode of the robot arm to interpolation mode, so that the motion trajectory will be relatively straight.

```
set_fresh_mode(0) # Enable interpolation
send_coords(coords, speed, mode=1) # Go straight
```

Note that the intelligent planning route set in `send_coords` will only be useful after interpolation is turned on. Interpolation means that many dense points are planned between the starting point and the end point to achieve the effect of controlling the trajectory of the middle section.

Q: The target position is identified, but the end cannot reach it. How to determine whether this coordinate can be reached and then process it?

- A: Use `solve_inv_kinematics(target_coords, current_angles)` to see if there is a solution.
`solve_inv_kinematics(target_coords, current_angles)`
- Function: Convert coordinates to angles.
- Parameters:
- `target_coords`: list A floating point list of all coordinates.
- `current_angles`: list A floating point list of all angles, the current angle of the robot
- Return value: list A floating point list of all angles.

5 Arduino related

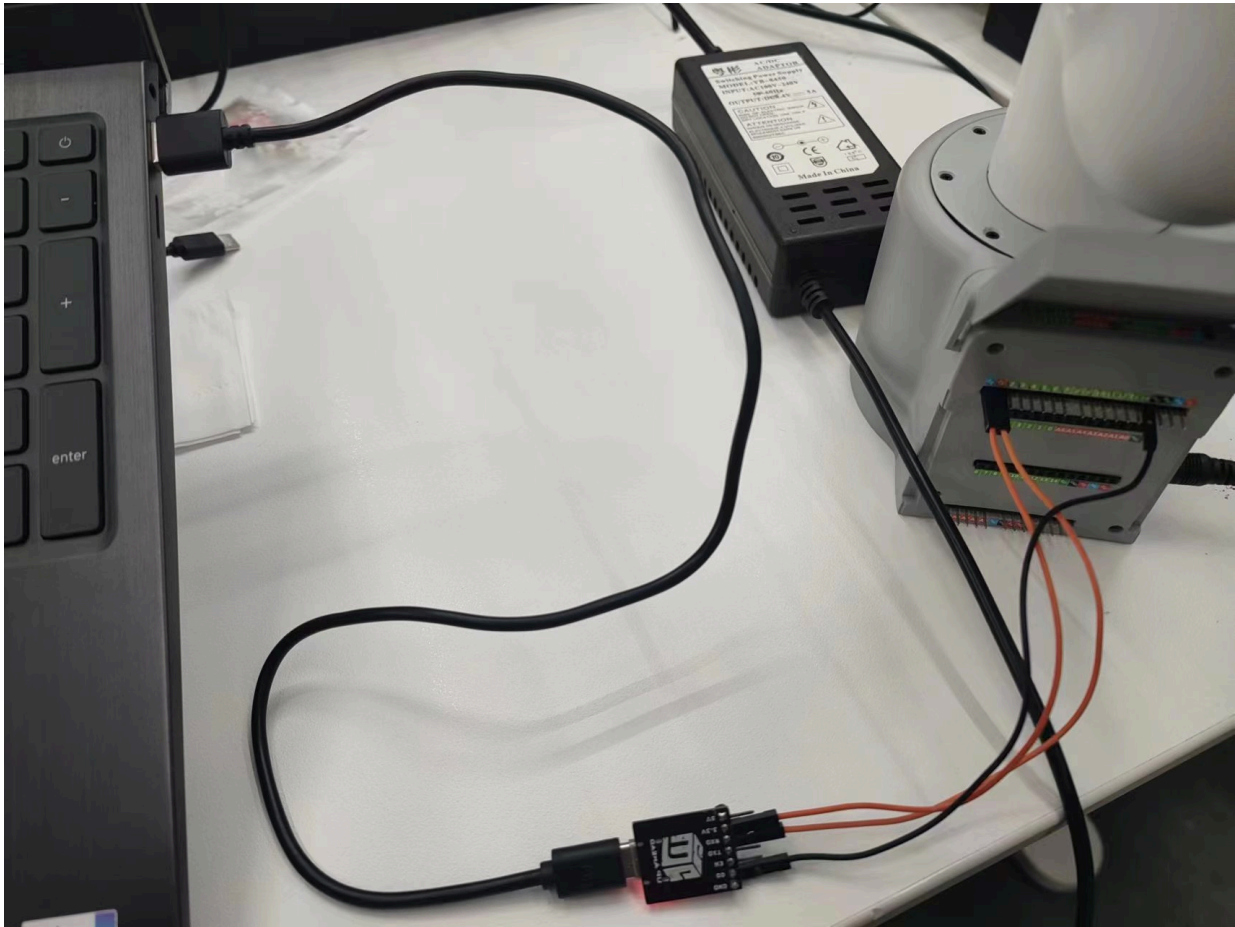
Q: Can Arduino be controlled by serial port using UNO

- A: Yes, connect TX and RX, and use serial port commands to control

Q: How to use serial port control on Arduino?

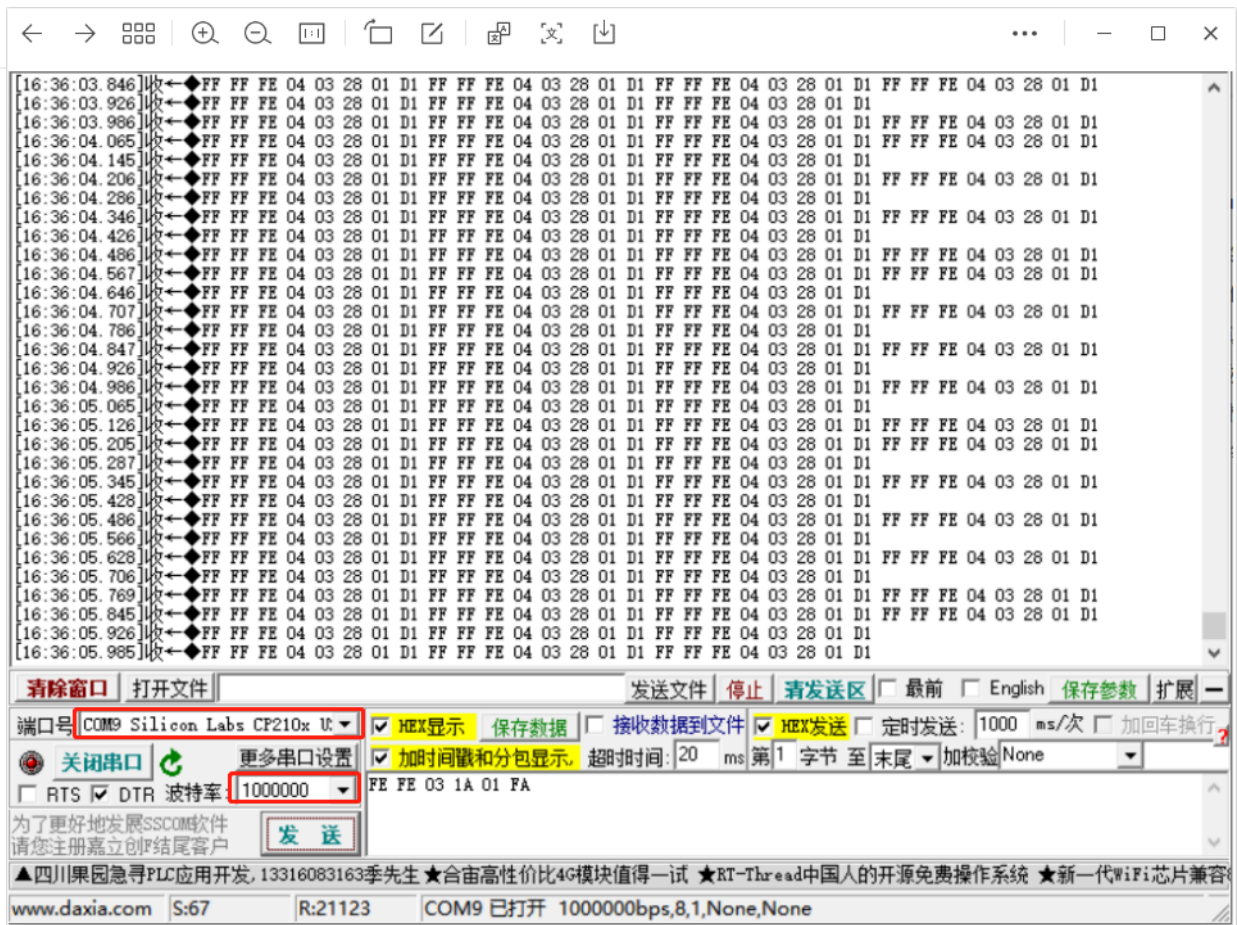
- A: Without any main control board, connect a USB to TTL module (TX, RX, GND), and use the serial port assistant tool to control the sending of the serial port commands we provide to achieve functions such as controlling joint movement

4.1 First-time self-check



When using, please note that the baud rate is 1000000 and the corresponding port

4.1 First-time self-check



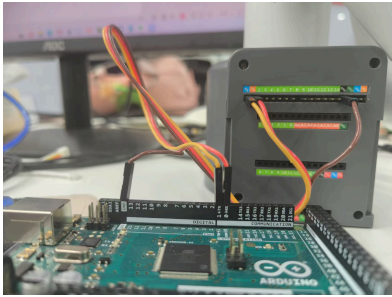
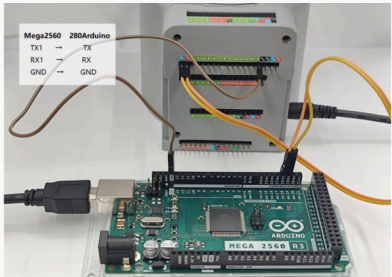
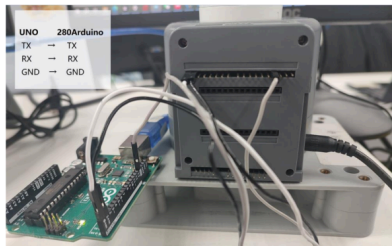
When using the robot free mode (turn off all torque outputs), the serial port sends an example: FE FE 03 1A 01 FA. Note that when the Atom light turns yellow, you need to press and hold the Atom button to move the joint.



Q: Control methods supported by 280Arduino

- A: 280Arduino supports: Arduino mkrwifi1010, mega2560, UNO, and other development board extensions that support UART serial communication interface modules. Among them, UNO cannot be controlled by Python due to limited hardware interfaces. It needs to be controlled by Arduino IDE, and the UNO board is connected to the Arduino board via Dupont wires and cannot be directly plugged in.

Note: The current 280AR serial port baud rate is changed to 1000000. The wiring of different development boards is different:

| Development board type | Baud rate | Control method | Wiring method | Wiring diagram |
|------------------------|-----------|------------------|-------------------------------------|--|
| mega2560 | 1000000 | python/MyBlockly | TX0-->RX、 RX0-->TX、 GND-->GND |  |
| mega2560 | 1000000 | Arduino IDE | TX1-->RX、 RX1-->TX、 GND-->GND |  |
| UNO | 1000000 | Arduino IDE | tx-->tx RX-->RX、 GND-->GND |  |
| mkrwifi1010 | 1000000 | None | None | None |

6 ROS related

Q: Is there a virtual machine image with a configured environment?

- A: We have provided a virtual machine environment with configured ROS1 and ROS2 environments and built-in ROS source code. Users can download it through the following link and import the virtual machine file into VirtualBox, saving the trouble of configuring the environment by themselves. When testing ROS cases, it is recommended to use the virtual machine environment we have configured for verification to avoid some case operation errors due to environmental configuration reasons Please refer to the operation steps video of importing virtual machine files into virtual machine software:

https://drive.google.com/file/d/1KeYk_CUgDE46rVn7zbd0Ehralbgt3qZt/view?usp=sharing

[ROS1 virtual machine file download](#)

[Download ROS2 virtual machine file](#)

[Download virtual machine software VirtualBox](#)

Q: How to deal with errors when importing ROS2 virtual machine files?



- A: This is because the version of the virtual machine software Oracle VM VirtualBox is too low, and the virtual machine software version needs to be updated.

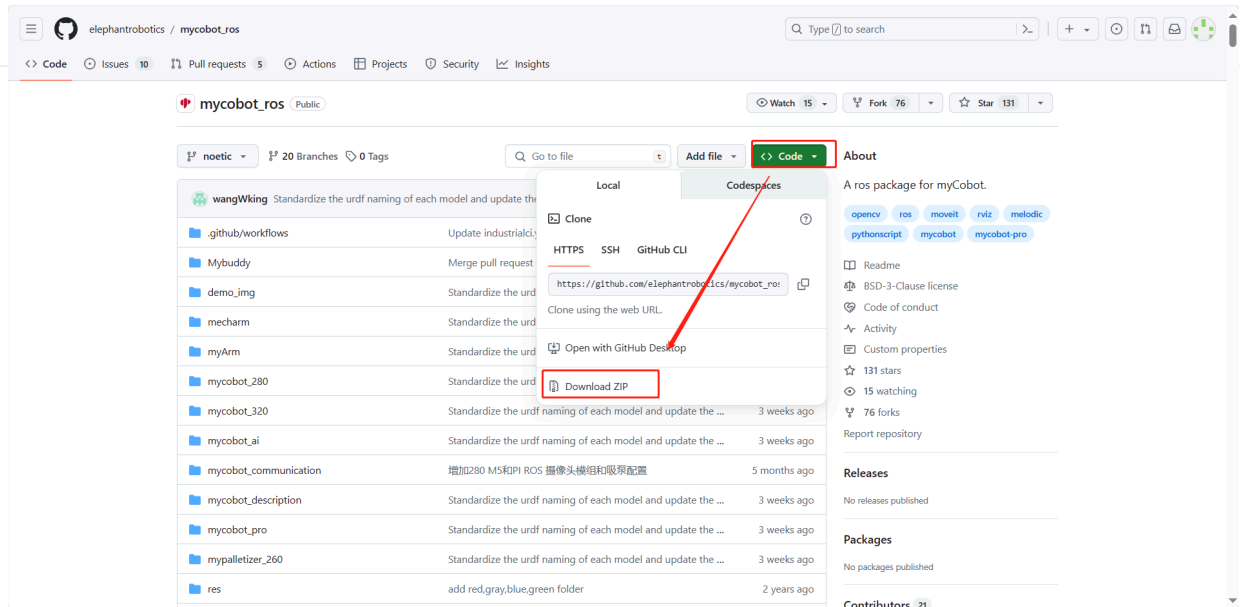
Q: How to re-download the ROS source code package?

- A: Use the command to pull:

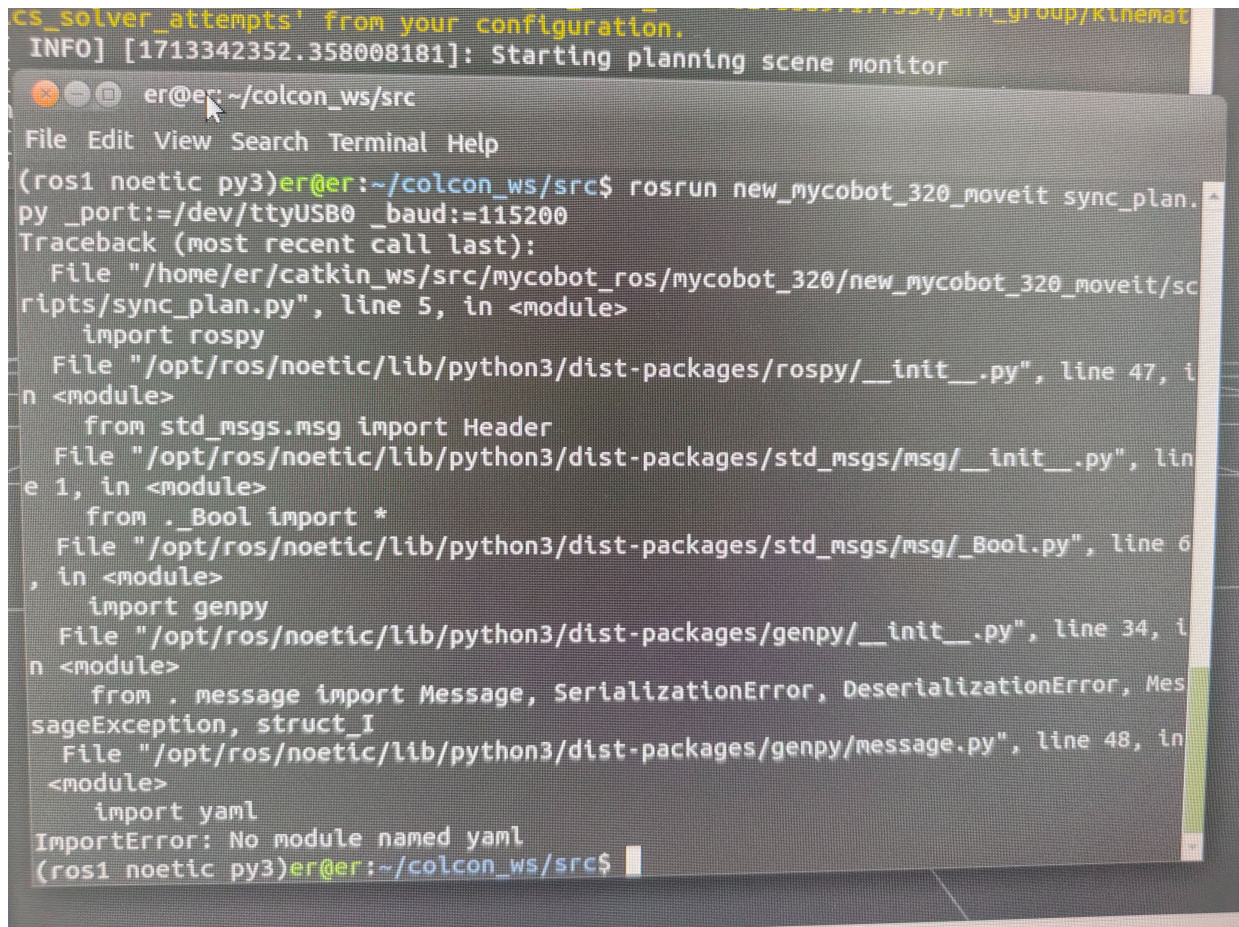
```
git clone https://github.com/elephantrobotics/mycobot_ros.git
```

Or download manually. The download method is to enter the ROS source code package address and follow the steps below. The source code package address is: https://github.com/elephantrobotics/mycobot_ros

4.1 First-time self-check



Q: What should I do if I run the ROS moveit case and get an error ImportError: No module named yaml?



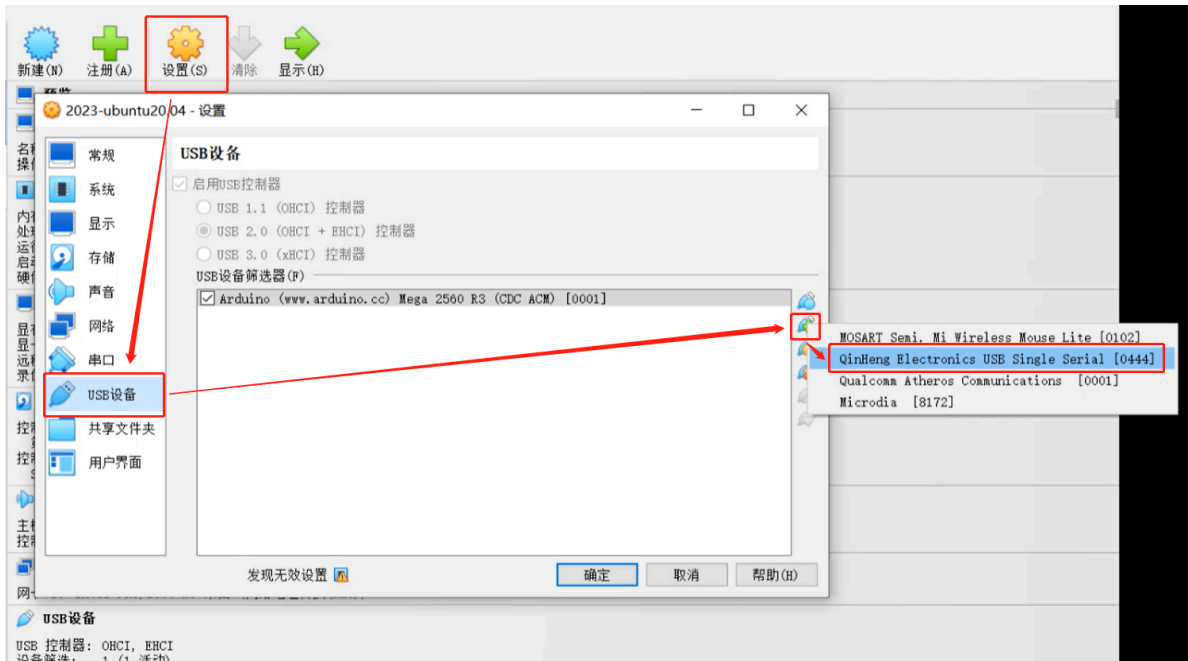
- A: In the first line of this script, change the Python interpreter to python3

Q: What should I do if the serial port cannot be found when running the virtual machine?

- A: Use a USB cable to connect the M5 robot to the PC, open the virtual machine settings → USB device → Add USB device → Select the serial port number QinHeng xxxxx, which is the serial port device of the machine. If there is no such device number, you can get the corresponding USB device number by re-plugging the device.

4.1 First-time self-check

The serial port number corresponding to the machine serial port device number is the one that changes when plugging and unplugging.



Q: Using a mujoco-based environment for simulation training, the robot's xml file is required

- A: Currently, there is only the 280JN xml file on GitHub: [280JN](#)
- Provide customers with methods on how to convert dae and urdf files into xml files, and let customers use [meshlab](#) to convert by themselves.

Q: When the terminal switches to `~/catkin_ws/src` and uses `git` to install and update `mycobot_ros`, the target path "`mycobot_ros`" already exists. What is the reason?

- A: This means that there is already a `mycobot_ros` package in `~/catkin_ws/src`. You need to delete it in advance and then re-execute the `git` operation.

Q: When `roslaunch` is running, the terminal reports an error message `could not open port /dev/ttyUSB0: Permission: '/dev/ttyUSB0'`, why?

- A: The serial port permissions are insufficient. Enter `sudo chmod 777 /dev/ttyUSB0` in the terminal to grant permissions.

Q: When `roslaunch` is running, the terminal prompts `Unable to register with master node [http://localhost:11311]: master may not be running yet. Will keep trying`. Why?

- A: Before running the `roslaunch` program, you need to open the `roslaunch` node. Enter `roslaunch` in the terminal.

Q: When `roslaunch` is running, the terminal reports an error message `could not open port /dev/ttyUSB0: No such file or directory: '/dev/ttyUSB1'`, why?

- A: The serial port is incorrect. You need to confirm the actual serial port of the current robot. You can check it through `ls /dev/tty*`.

Q: In Ubuntu 18.04, `catkin_make` failed to build the code, and the terminal prompted `Project 'cv_bridge' specifies '/usr/include/opencv' as an include dir, which is not found` and other error messages

- A: The `opencv` path in the configuration file does not match the actual system path. 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 system `opencv` path is under the `/usr/include/` path.

4.1 First-time self-check

Q: I just cloned the mycobot_ros package, and then ran the rosrn program directly. The error `package`

`'mycobot_280' not found` **or the file could not be found appeared?**

- A: The mycobot_ros that I just cloned needs to build the code for ros environment compilation. Terminal input

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

Q: After the compilation is completed, why does the following error appear when the launch command is run in a new terminal?

```
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 system does not add ros environment variables, so you need to source each time you open a new terminal:

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

- A2: The system adds ros environment variables, and you do not need to execute source each time you open a new terminal:

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

- A3: The file name in the command may be inconsistent with the actual file name in the mycobot_ros package. Please check the command carefully for errors.

7 C++ related

Q: What should I do if I can't find various dll files?

- A1: If myCobotCpp.dll is missing, put myCobotCpp.dll in the lib directory to the directory where mycobotcppexample.exe is located.
- A2: If QT5Core.dll is missing, open qt command (search QT in the menu bar), select msvc2017 64-bit, and execute windeployqt--release to the directory where myCobotCppExample.exe is located (such as: windeployqt --release D:\vs2019\myCobotCpplout\build\64-Release\bin). If the vs installation path cannot be found after executing the command here, please check the settings of the vs environment variables.

After executing the above steps, if the qt5serialport.dll file is missing, move this file in the qt installation directory (path such as: D:\qt5.12.10\5.12.10\msvc2017_64\bin), copy it to the directory where myCobotCppExample.exe is located

Q: Generate the myCobotCppExample.exe executable file, what could be the problem? Select the start



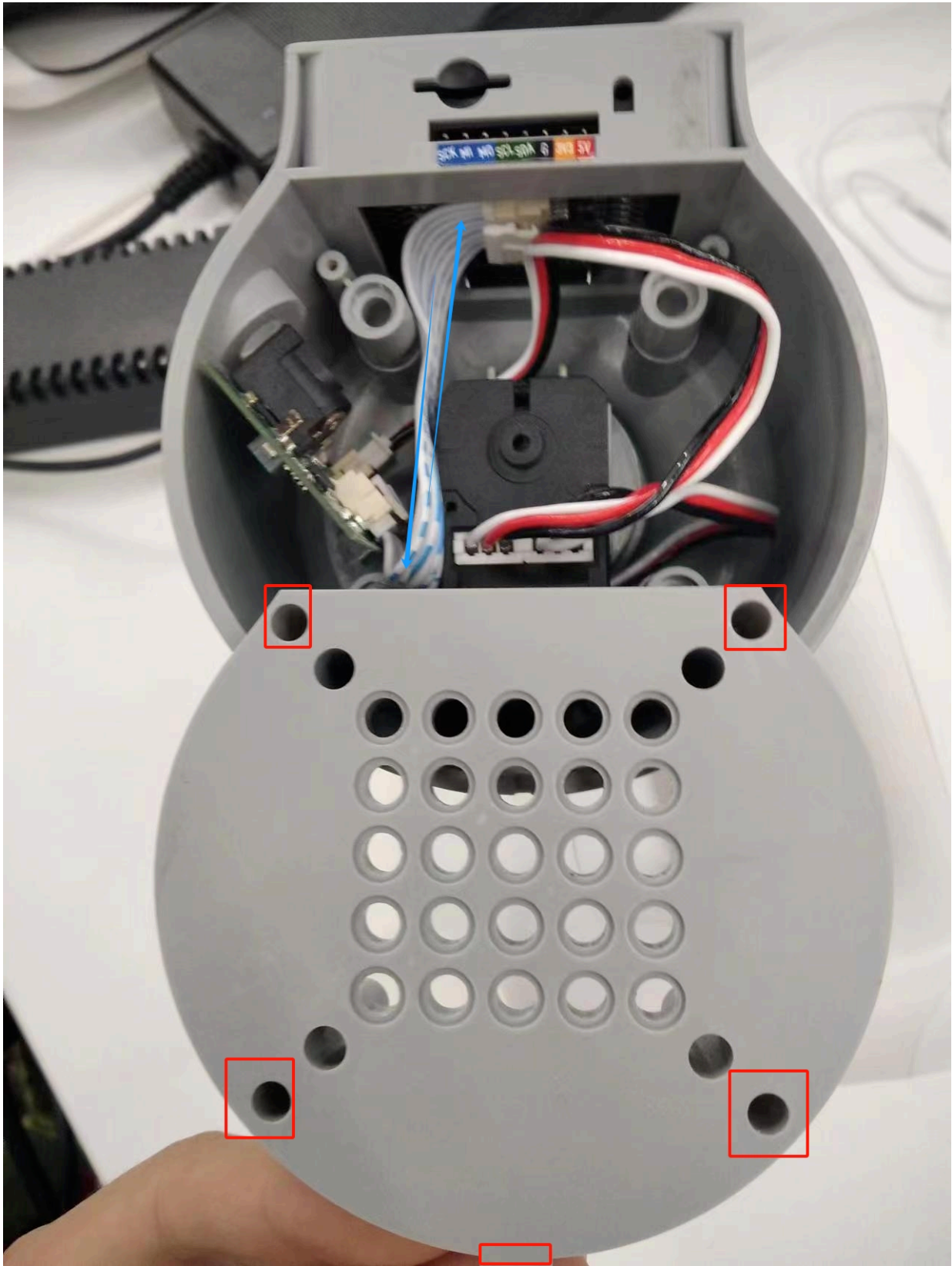
Hardware Issues

Q: How to resolve the issue if the M5 screen is not displaying anything?

1. Check if the power adapter is properly connected to the machine. You can try unplugging and replugging the power adapter.
2. Gently press on the corners of the screen to ensure that the M5stack is making good contact with the internal expansion board.



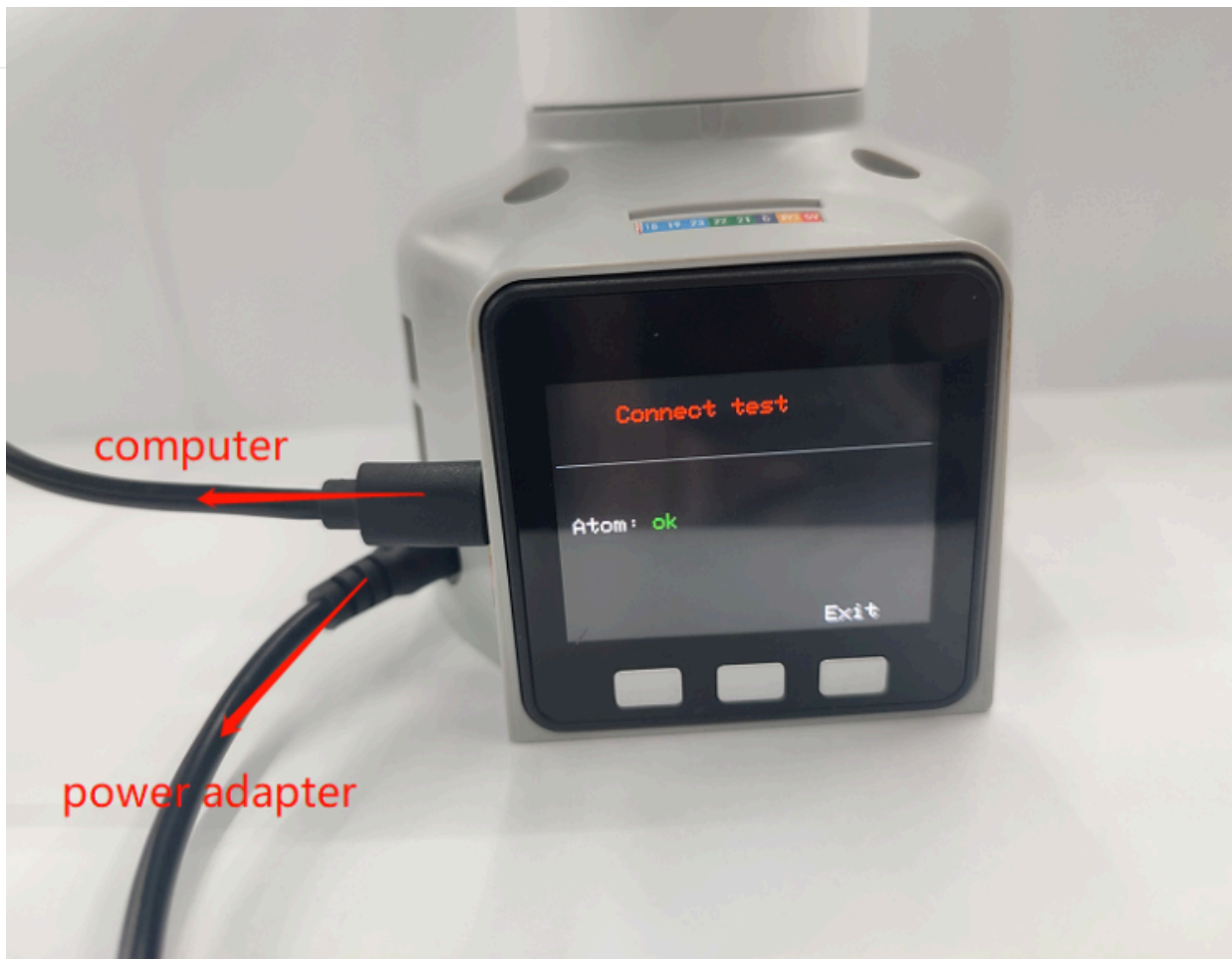
1. Check the GitBook and download the corresponding MiniRobot firmware.
2. You can remove the base screws to inspect whether any internal cables are disconnected. If so, reconnect them before use.



Q: How to solve the problem that the robot arm cannot be locked when powered on?

1. Check whether the original power adapter is connected or whether the adapter is in good contact. You can try to re-plug the power adapter

4.1 First-time self-check



1. Check whether the joint can rotate normally when the power is off, whether there is too much or too little resistance, and preliminarily determine whether the internal structure is physically broken. If there is no physical break, continue to check.
2. Check the Atom firmware as follows:

Under normal circumstances, the robot arm will self-lock after power is turned on, and the Atom will light up green, as shown in the figure below (note that mechArm has no light display)



After the robot arm is powered on, the Atom does not light up green or the joint cannot self-lock. You can check according to the following points: ① Gently press the Atom screen to make Atom contact well with the internal plate of the robotic arm.

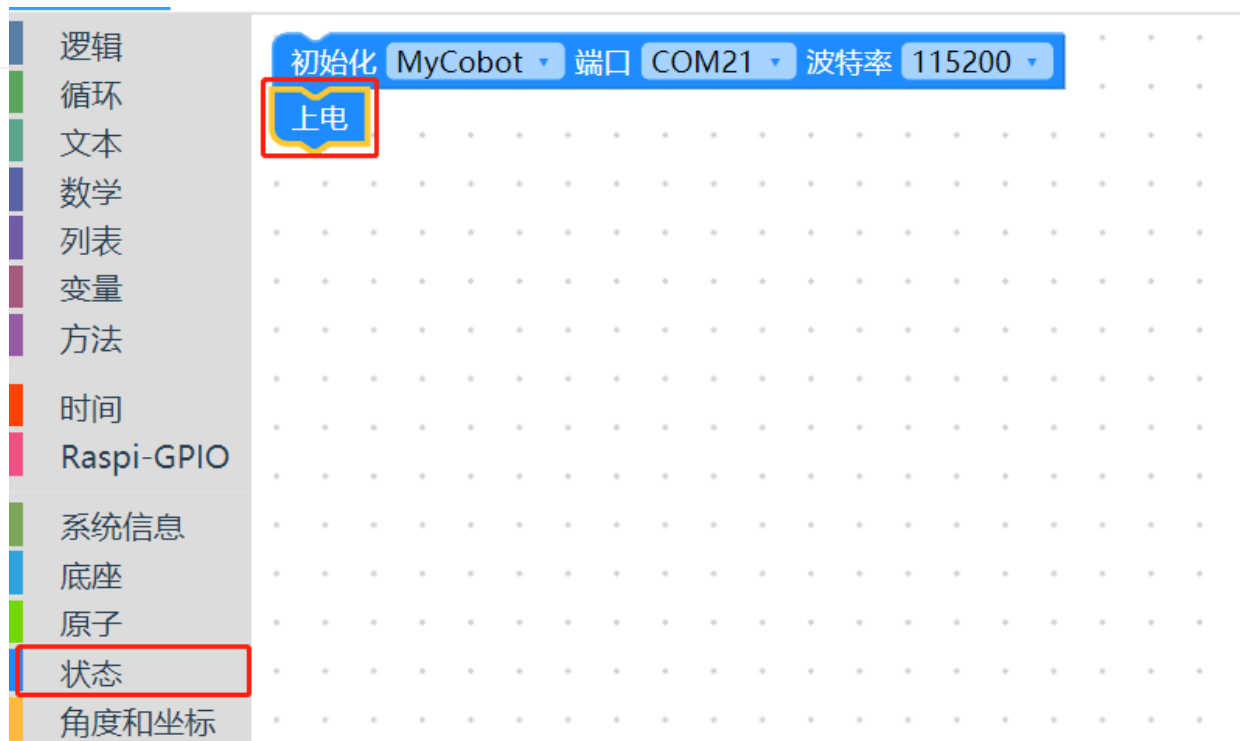
② Check gitbook to get the method of using mystudio. According to the model and version information, use mystudio to download the corresponding Atom firmware. If you encounter any problems during the burning process, please refer to the relevant "Firmware download exception" in this article mystudio to obtain the troubleshooting steps.

③ When the Atom firmware is successfully burned and the robotic arm is not connected to the power supply, use type-c to connect Atom. If the green light of Atom is on, but the green light of Atom is off after unplugging type-c, it is judged that Atom is normal, but there is a problem of circuit detachment or damage inside the robotic arm, and it is necessary to contact the technician to deal with it.

④ When the Atom firmware is successfully burned and the robotic arm is not connected to the power supply, use type-c to connect Atom but the green light is not on, it is judged that the Atom hardware is damaged and it is necessary to contact the technician to replace it.

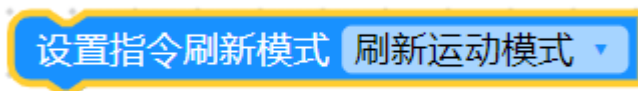
Q: After pressing the emergency stop, the emergency stop cannot be locked after release. How to lock the robotic arm again?

You need to power on the machine again, for example, power on the machine with myblockly



Q: How to optimize joint jitter, excessive joint angle deviation or joint weakness?

1. Refer to the robot parameter introduction section to check whether the actual load is within the effective load range of the robot arm. Excessive load will cause joint jitter. The load of the actual joint can be appropriately reduced.
2. Change the motion mode to refresh mode, so that the running trajectory of the robot arm will be relatively smooth. For specific APIs, please refer to `set_fresh_mode(1)`



1. Check the following link to adjust pid:
https://drive.google.com/file/d/1UWhaaSTuwLFIImuEGY1J2tvgxTQDwWxK_/view?usp=sharing
2. Check the gitbook section and use mystudio to download the corresponding version of Atom firmware. It is recommended to download the latest
3. Check Chapter 5 of the gitbook to calibrate the robot arm at zero position. You can also refer to the calibration steps in the following link: https://drive.google.com/file/d/1XtKH-ykKWP0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing
4. For machines that have been used for a long time (more than 3 months), joints may age and produce joint gaps. You can follow the following video to manually bend the joints to check if there is any joint gap:
<https://drive.google.com/file/d/1tXDUALmfw1z0u6IM9uH5hOHivjbRoWxW/view?usp=sharing>
5. If there is a joint gap problem due to joint aging, this kind of jitter is inevitable due to the natural aging of the machine.

Q: What is the joint zero position?

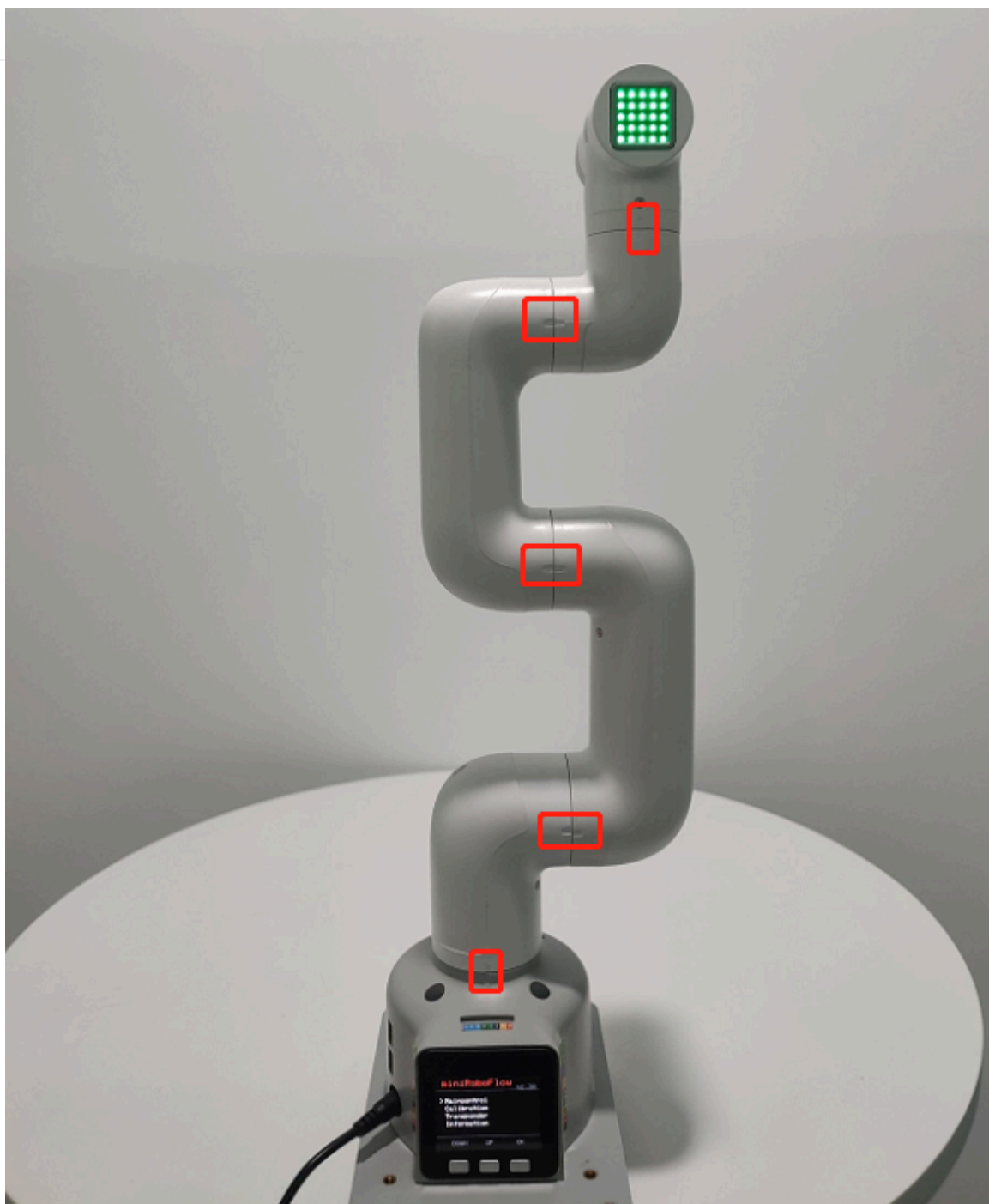
Take the following figure as an example, there is an arched groove designed between the joint and the edge of the joint shell, which is the joint zero point

4.1 First-time self-check

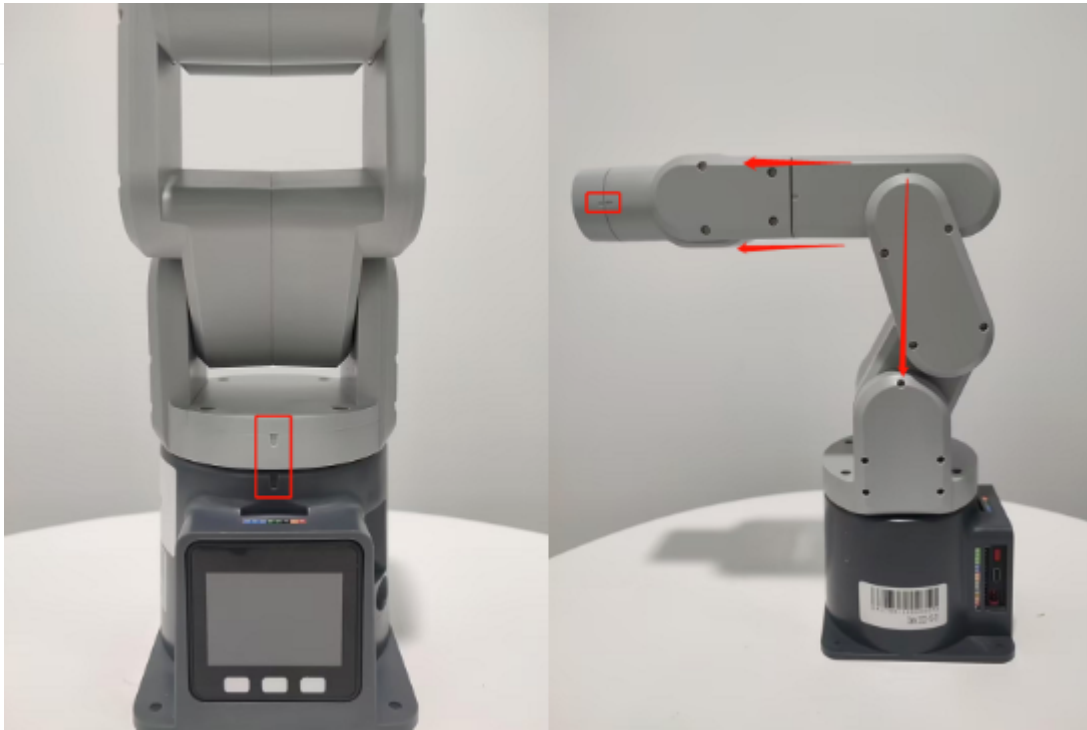


Generally, the zero point posture after calibration is as follows:

4.1 First-time self-check



Pay special attention to the zero position posture of the 270 joint as follows:



Q: Is there a method for zero point calibration?

Please refer to Chapter 5 of gitbook or the following link:

https://drive.google.com/file/d/1XtKH-ykKWP0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing

Q: Is the IO port of G36 unusable? Why is there no voltage return when measuring with a multimeter?

- A: 36 on basic can be used and is defined as an input pin. If the customer needs to test whether the IO can be used, the customer can modify the basic firmware code by himself. The firmware is open source. The customer writes a program by himself, single control IO

Q: What are the limits of myCobot's joints?

- A: One axis and five axes have limits. One axis is about 165° clockwise and about 165° counterclockwise. The five axes can rotate about 165° clockwise and counterclockwise

Note: When rotating the robot arm, it should be rotated at a small angle and gently. After reaching the limit, it cannot be rotated forcefully.

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

- A: Atom is mainly used in the robot arm to control the kinematic algorithm of the robot arm: including forward and inverse kinematics, solution selection, acceleration and deceleration, speed synchronization, multi-square interpolation, coordinate conversion, etc., and requires real-time control and multithreading. The relevant programs of atom are not open source yet.

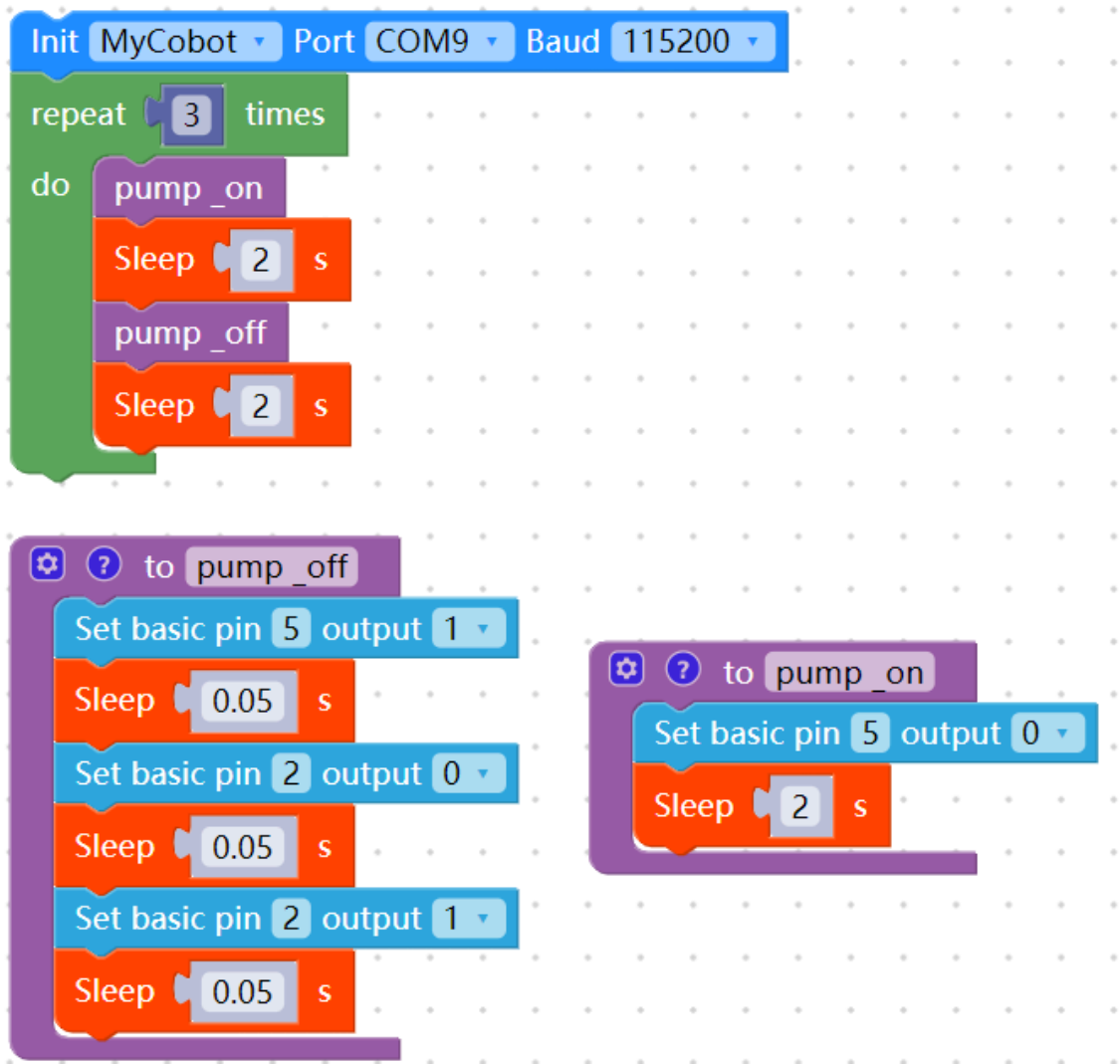
Q: What communication interfaces do different versions of the robot arm support?

- A: The robot arm based on the microprocessor supports socket communication TCP; the robot arm based on the microcontroller can communicate via USB to serial port.

Accessory related questions

Q: 280M5 and suction pump 2.0 io connection diagram and quick use source code

- A: The new version v2.0 version of myblockly source code is as follows:



The G5 pin on the suction pump is the suction pump switch control pin, and the G2 label is the solenoid valve control pin. Both are low level valid. The function of the solenoid valve is to make the suction pump more rapid when released. If the solenoid valve is not used, the suction pump can also work normally, but the speed of releasing the object when the suction pump is closed is relatively slow. The source code here uses the G5 pin to control the opening and closing of the suction pump, and the G2 pin to control the opening and closing of the solenoid valve. The opening and closing of the solenoid valve mainly works in the stage of closing the suction pump. Note that the pins designated for connecting the suction pump and the machine end are not fixed. On the machine end, the G2 and G5 pins are not the only options. They can be replaced with any other two common GPIOs on the machine for control. However, when changing the control pins, you need to pay attention to the G2 and G5 pin function descriptions on the suction pump end - the pin corresponding to the G5 label on the suction pump end is the suction pump switch control pin, and the G2 label is the solenoid valve control pin, both of which are low-level valid. As shown in the following source code, GPIO18 and 19 are used to control the suction pump. G19 is selected to connect the suction pump's G5 to control the suction pump opening and closing, and the other G18 is connected to the suction pump's G2 pin to control the solenoid valve.

The image displays three Scratch code blocks on a grid background. The top block is an 'Init' block for 'MyCobot' with 'Port' set to 'COM9' and 'Baud' set to '115200'. Below it is a 'repeat 3 times' loop containing a 'do' block with four sub-blocks: 'pump_on', 'Sleep 2 s', 'pump_off', and 'Sleep 2 s'. The bottom-left block is a function 'to pump_off' consisting of five sub-blocks: 'Set basic pin 19 output 1', 'Sleep 0.05 s', 'Set basic pin 18 output 0', 'Sleep 0.05 s', and 'Set basic pin 18 output 1', followed by 'Sleep 0.05 s'. The bottom-right block is a function 'to pump_on' consisting of two sub-blocks: 'Set basic pin 19 output 0' and 'Sleep 2 s'.

Q: What is the pin sequence and connection method of mycobot adaptive gripper?

Please refer to the following figure for the pin introduction of mycobot adaptive gripper:

4.1 First-time self-check



Gripper connection method:



Q: Parallel gripper usage source code

The image shows a sequence of Scratch-style code blocks for controlling a robot gripper. It starts with an 'Init' block for 'MyCobot' on 'Port COM22' with a 'Baud' rate of '115200'. This is followed by a 'repeat 3 times' loop. Inside the loop, the 'do' block contains four steps: 1) 'Set Gripper State' to 'Open' with a 'Speed' of '30' and 'Type' 'Parallel Gripper'; 2) 'Sleep' for '3 s'; 3) 'Set Gripper State' to 'Close' with a 'Speed' of '30' and 'Type' 'Parallel Gripper'; 4) 'Sleep' for '3 s'.

Q: Is there anything to pay attention to between the gripping object and the movement of the robot arm?

When the load is > 500g, the speed needs to be less than 50%.

Other Issue

Q: How to use the mycobot test tool?

A: The "mycobot test tool" is intended for factory use only, and we do not recommend users to use it. Using this tool may lead to abnormalities in the zero position or PID, resulting in damage to the robot. Please delete this tool directly. If you have already used this tool and it has caused abnormal movement of the robot, please refer to the following instructions to readjust the PID and zero position. Furthermore, refrain from continuing to use this factory test tool in future operations. Resetting PID method reference link:

https://drive.google.com/file/d/1UWhaaSTuwLFImuEGY1J2tvgxTQDwWxK_/view?usp=sharing Zero position calibration method reference link: https://drive.google.com/file/d/1XtKH-ykKWP0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing

Q: After the vertical suction pump is turned off, there is no air leakage and the adsorption can be done normally. However, the vertical suction pump is still in the state of adsorption when the program ends. What is the reason?

A: 1. First check if the GPIO port is connected incorrectly. The correct connection method is shown in the figure
Pi connection method



M5 connection method:



1. If the above is correct but still not working, you can try to swap the high and low levels in the code to find out the reason

4.1 First-time self-check

```
1 from pymycobot.mycobot import MyCobot
2 from pymycobot import PI_PORT, PI_BAUD # 当使用树莓派版本的mycobot时, 可以引
3 import time
4 import RPi.GPIO as GPIO
5
6 # 初始化一个MyCobot对象
7 mc = MyCobot(PI_PORT, PI_BAUD)
8
9 # 初始化
10 GPIO.setmode(GPIO.BCM)
11 # 引脚20/21分别控制电磁阀和泄气阀门
12 GPIO.setup(20, GPIO.OUT)
13 GPIO.setup(21, GPIO.OUT)
14
15 # 开启吸泵
16 def pump_on():
17     # 打开电磁阀
18     GPIO.output(20,0)
19
20 # 停止吸泵
21 def pump_off():
22     # 关闭电磁阀
23     GPIO.output(20,1)
24     time.sleep(0.05)
25     # 打开泄气阀门
26     GPIO.output(21,0)
27     time.sleep(1)
28     GPIO.output(21,1)
29     time.sleep(0.05)
30
31 pump_off()
32 time.sleep(3)
33 pump_on()
34 time.sleep(3)
35 pump_off()
36 time.sleep(3)
37
38 GPIO.cleanup() # 释放 pin channel
```

Figure 1 (source code)

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD # 当使用树莓派版本的mycobot时,
import time
import RPi.GPIO as GPIO

# 初始化一个MyCobot对象
mc = MyCobot(PI_PORT, PI_BAUD)

# 初始化
GPIO.setmode(GPIO.BCM)
# 引脚20/21分别控制电磁阀和泄气阀门
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# 开启吸泵
def pump_on():
    # 打开电磁阀
    GPIO.output(20,0)

# 停止吸泵
def pump_off():
    # 关闭电磁阀
    GPIO.output(20,1)
    time.sleep(0.05)
    # 打开泄气阀门
    GPIO.output(21,1)
    time.sleep(0.05)
    GPIO.output(21,1)
    time.sleep(0.05)

pump_off()
time.sleep(3)
# pump_on()
# time.sleep(3)
# pump_off()
# time.sleep(3)

GPIO.cleanup() # 释放 pin channel
```

Figure 2 (modified)

According to the modified code, it can be checked whether the closed solenoid valve and the vent valve are high-level venting or low-level venting, and so on.

Q: How to reset to factory settings when the machine is abnormal?

Restoring to factory settings mainly depends on resetting the firmware, image, pid and zero position. The following is the reset solution:

- **About resetting the firmware:** It is recommended to ensure that mystudio is updated to the latest version, and then download the corresponding latest Atom version firmware, minirobot firmware (only available in M5 series machines) and pico firmware (only available in 320 series models). For the method of resetting the firmware, please refer to the mystudio chapter of gitbook
- **About resetting the image:** When resetting the image, all contents in the original system will be cleared. If there are important files, please save them in advance. For the method of resetting the image, please refer to the system usage chapter of gitbook
- **About resetting pid:** Generally, when the machine has severe joint shaking, abnormal joint movement speed, and joints curled up, the pid can be reset. For the reset method, refer to: https://drive.google.com/file/d/1UWhaaSTuwLFImuEGY1J2vtgxTQDwWxK/_view?usp=sharing
- **About resetting zero position:** Generally, when the machine has an incorrect zero position or the joint limit is abnormal, the zero position can be recalibrated. For the reset method, refer to: https://drive.google.com/file/d/1XtKH-ykKWPH0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing

Q: Where is the download path for the urdf file?

- A: Please refer to the following path. The urdf of all mycobot models is in this path: https://github.com/elephantrobotics/mycobot_ros/tree/noetic/mycobot_description/urdf

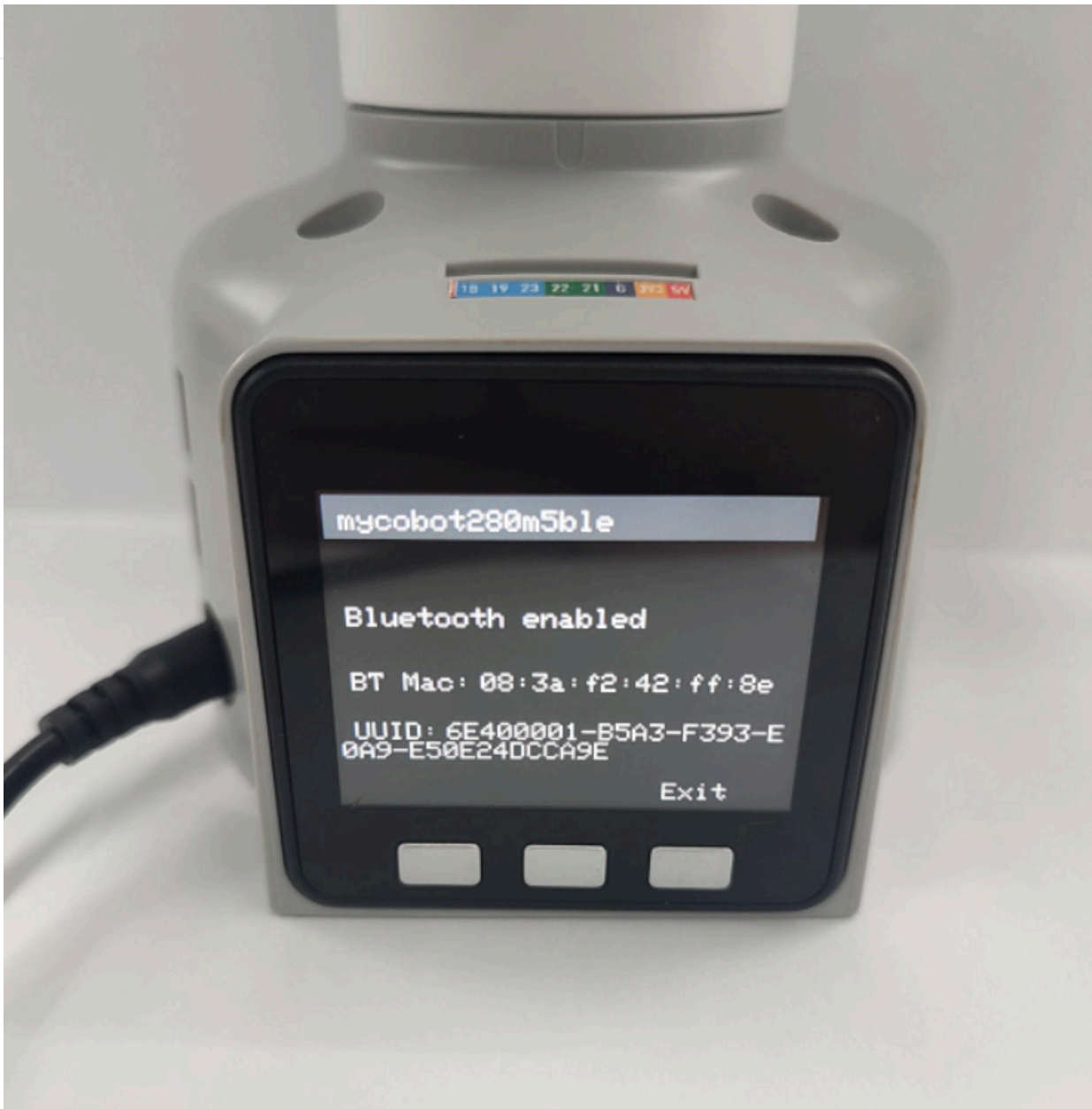
Q: How to troubleshoot if the robot arm cannot be controlled when using Bluetooth or WiFi?

A: The M5 robot arm supports three communication methods, namely USB serial port, WiFi and Bluetooth communication. The most commonly used USB serial port communication method is required. Before using each communication method, you need to ensure that the LCD screen of the M5 is adjusted to the corresponding mode and maintain this communication state to control the robot arm normally.

wifi communication mode: When using the TCP/IP example in the python chapter, you need to keep the M5's LCD screen in the wifi communication interface, as shown below:



bluetooth communication mode: When using the mobile phone APP control, you need to keep the M5's LCD screen in the Bluetooth communication interface, as shown below:

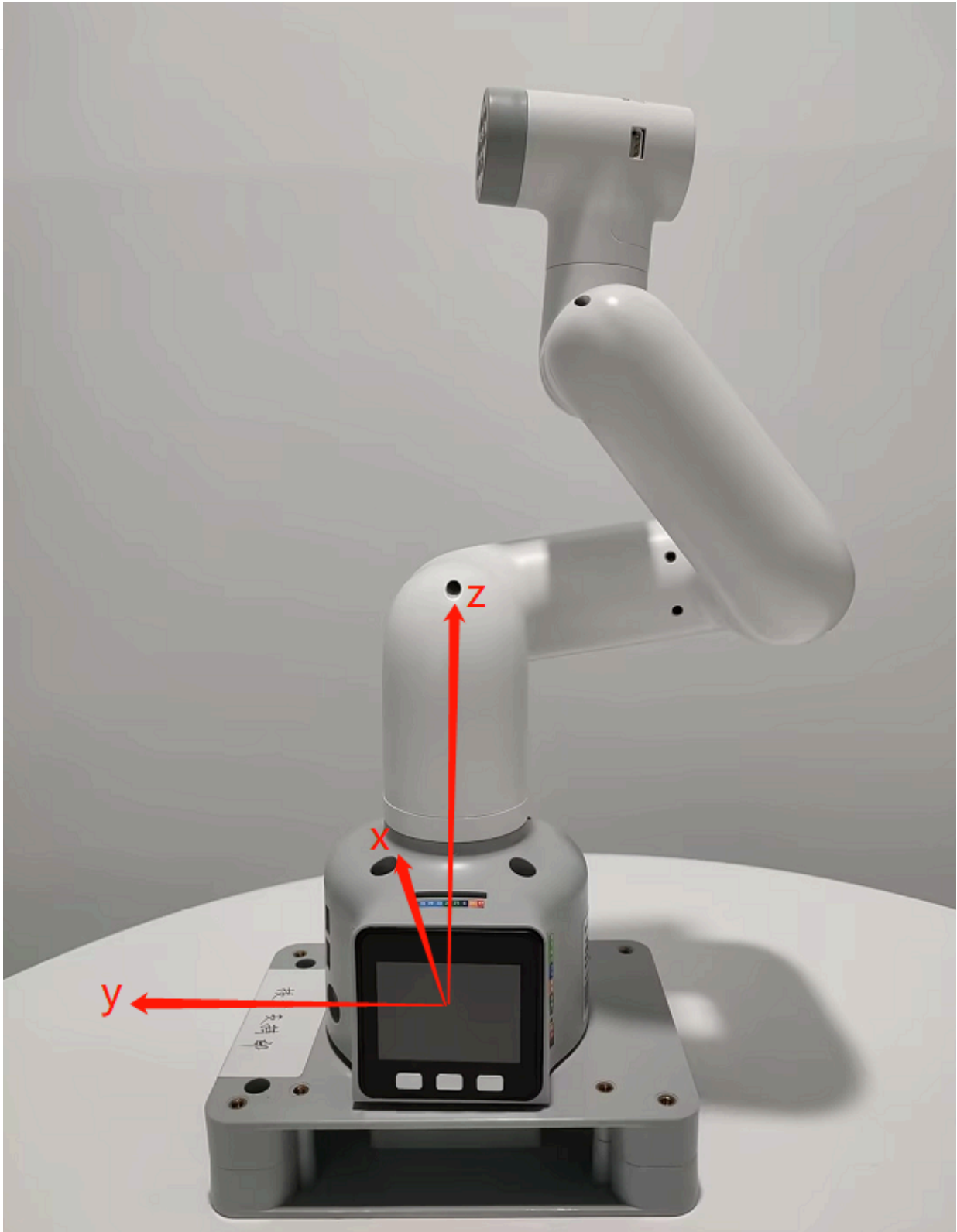


After ensuring that the communication mode status is selected correctly, try to control the robot arm again

Q: How long is the command transmission delay when controlling the motor through the robot's controller via serial port or socket communication? Is there a communication timing diagram? How about real-time performance?

There is no delay test data for serial or socket communication. According to the feedback from our development and use, the real-time performance is still quite high and there will be no lag.

Q: What is the base coordinate system of the 280M5 robot?



Q: Are the joints of 280 controlled by the serial bus?

A: Yes

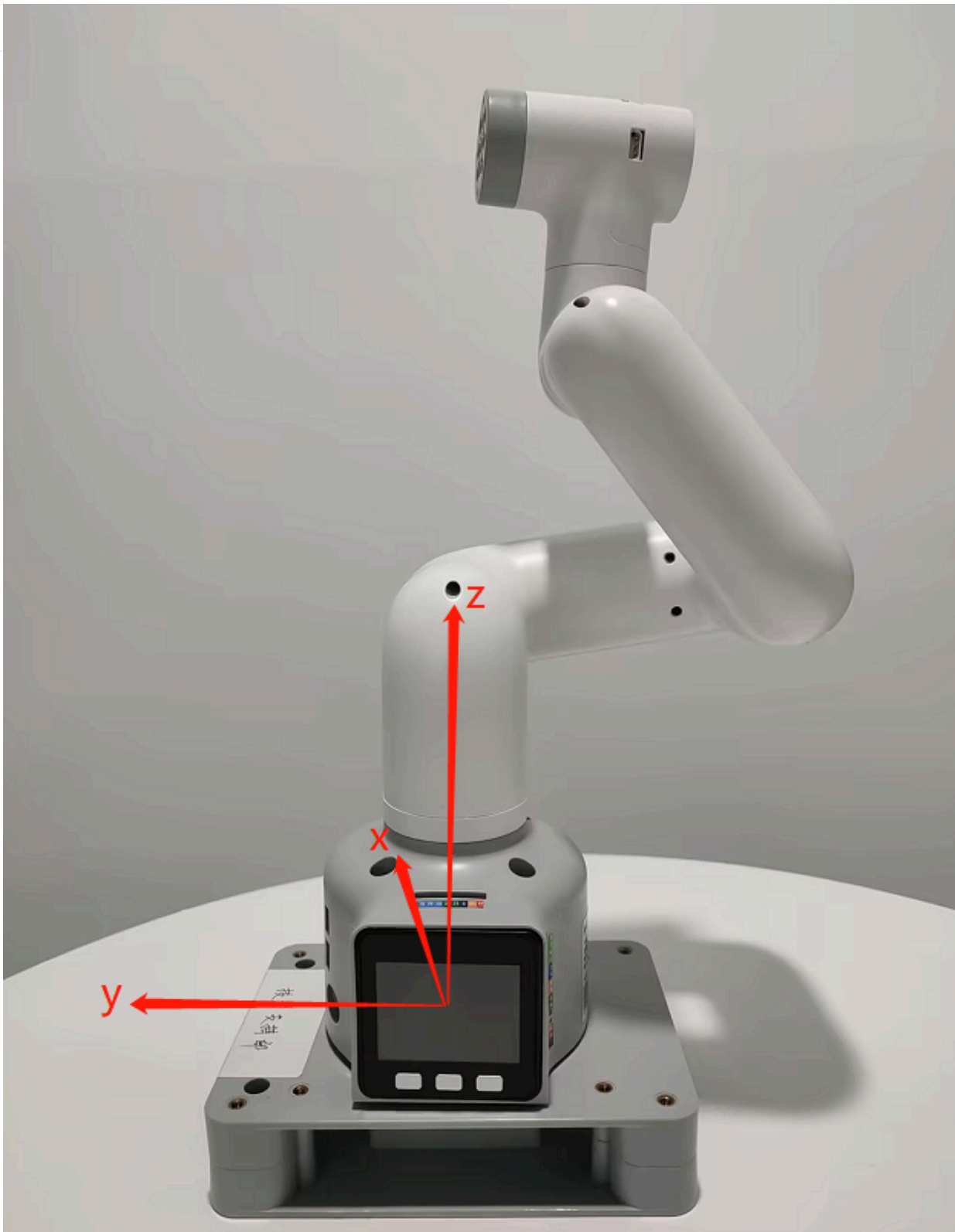
Q: Is there more explanation about the understanding of coordinates?

A: The API for controlling coordinate movement is `send_coords([x,y,z,rx,ry,rz], speed)` **x, y, z coordinates:** Control the position of the end effector of the robot in space. Changing these coordinate values will move the robot to different spatial positions, thereby achieving positioning in three-dimensional space. **rx, ry, rz attitude angles:**

4.1 First-time self-check

Control the attitude or orientation of the end effector of the robot. These values are usually given in the form of Euler angles, describing the rotation of the end effector of the robot relative to the base coordinate system, and the order of Euler angles is zyx. Changing these values will rotate the end effector of the robot to different angles or directions. For example: When you adjust +X, this means that the position of the end effector of the current robot arm moves a certain distance along the positive direction of the X axis of the current end effector. This action will cause the robot to move in a certain direction as a whole. And when you adjust RX, this means that the attitude of the end effector of the current robot arm rotates a certain angle around the X axis of the current end effector. This action will cause the robot to rotate as a whole and the direction of the end effector will change. In general, the adjustment of +X and RX will directly affect the motion state of the robot arm. +X controls the movement of the position, while RX controls the change of attitude. If you want to see the changes more intuitively, we recommend that you use myblockly's serial control tool to adjust a parameter at a time and observe its changes in the coordinate system. Please note that when observing rx, ry, and rz, if you want to be more intuitive, please pay attention to adjusting x and ry when the J1 joint is 0, and adjusting y and rx when the joint is 90. You can refer to the coordinate system diagram below:





Q: Is there more explanation about the Offset of the DH parameter? Is the Offset rotated around z?

A: The DH parameter describes the geometric and kinematic relationship between adjacent links in the robot arm. In the DH parameter table, the Offset parameter indicates the effect of the previous link rotating around its z-axis on the position of the next link, that is, the offset when connecting two links. For the Offset parameter in the robot arm, it generally indicates the effect of the previous link rotating around its own z-axis on the position of the next link, rather than rotating around the z-axis of the next link. Therefore, Offset is not a rotation around z, but a displacement when connecting two links.

4.1 First-time self-check

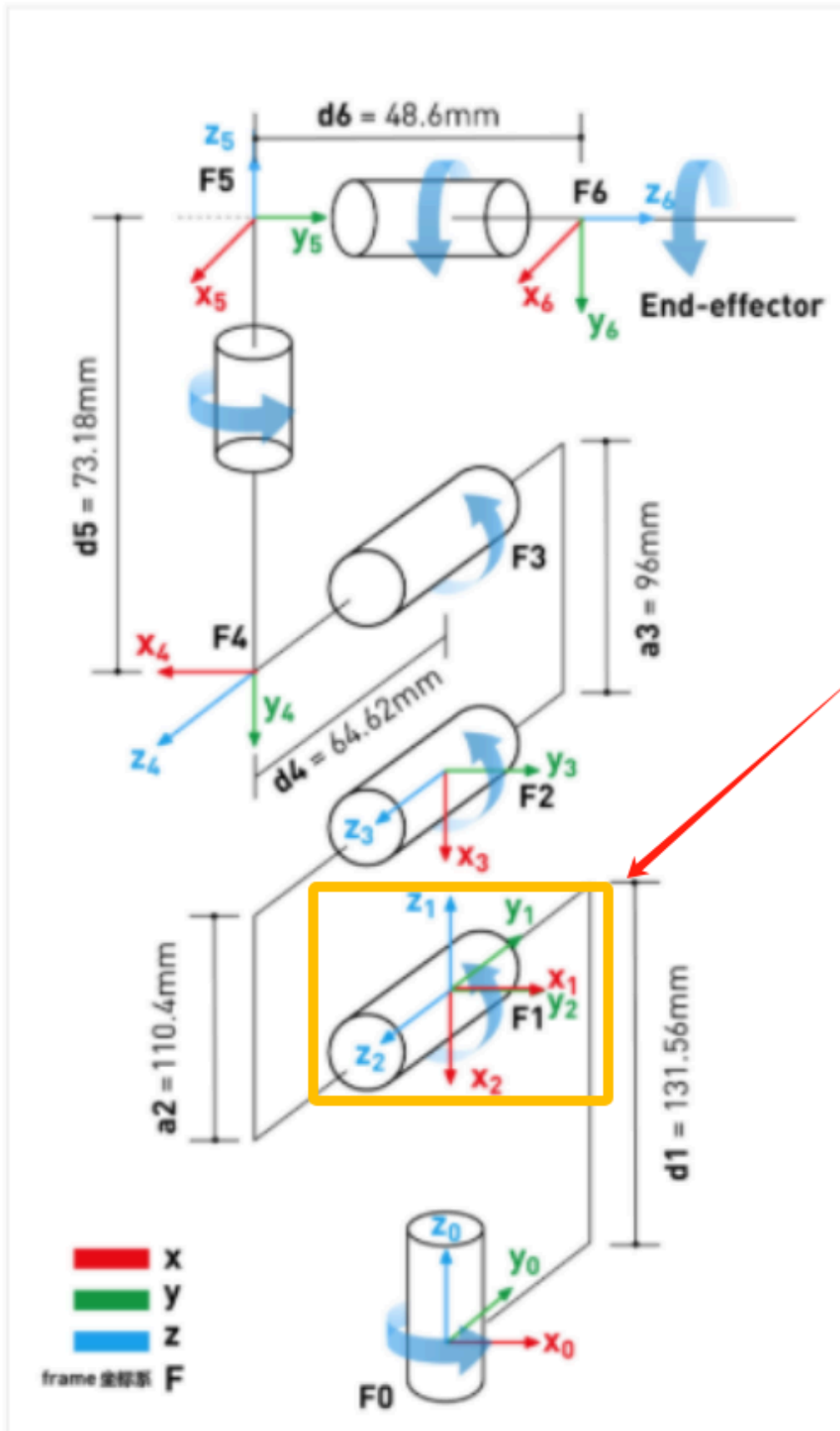
Q: What is the voltage range of the 280 robot arm power supply? How much is the instantaneous current?

A: 12V plus or minus 10%, 5A

Q: What is the inner diameter of the J1 joint of the 280M5? A: 70mm

Q: Why are two coordinate systems defined at F1 in this DH model of the 280m5?

3 DH参数



Q: Following the above question, why are the physical distances of axis 1-2 different, but the origin settings are overlapping?

A: Because the rotation axes intersect, DH modeling has regulations. If the rotation axes of adjacent joints intersect, the origin must be set at the intersection

Q: If the servos of each axis are controlled and feedback is obtained, what is the shortest communication cycle?

A: This needs to be determined according to the speed. The minimum response time is 50ms

Q: Does the mycobot series machine have collision detection?

A: 280 has algorithmic collision self-interference, which has been integrated into the API for setting joint angles and coordinates

Q: What are the input parameters of Atom's USB interface?

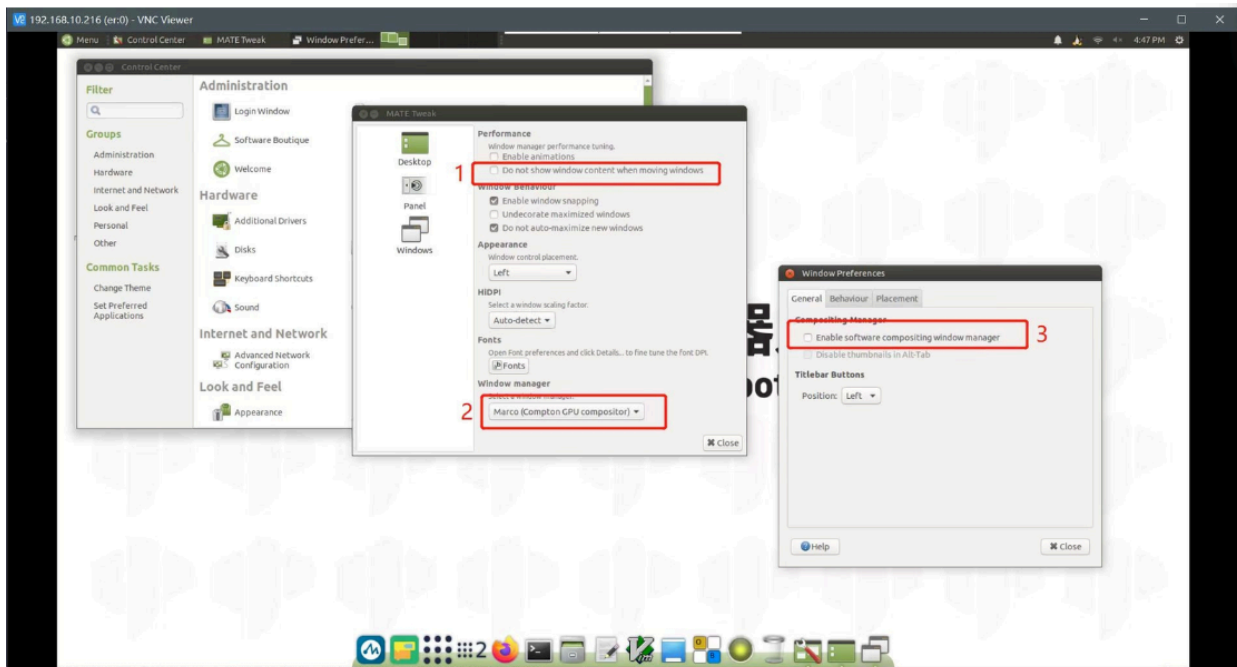
A: 5V @ 500mA

Q: How to deal with the 270 j3 joint not being able to display joint values in real time?

A: Reference link: https://drive.google.com/drive/folders/1BrvMxJltcLsr8T8-4kKOB7SH0D_qZkIP?usp=sharing The corresponding firmware has been replaced, and the corresponding python file can be run directly

Q: How to deal with the VNC dragging jam?

A: If the jam is caused by dragging any window in VNC, you can make some configurations according to the picture below. The options need to be consistent with the picture below. After successful setting, the problem of VNC disconnection caused by dragging the window will be solved.



Q: When replacing the second joint of 280, I found that 4 screws were stripped. How to remove them?

Regarding the replacement of joints, the 4 screws do not need to be removed. Please remove the large screw in the middle, then fix the J2 joint body back, and then use force to pull out the entire coupling. I recorded a video for you to refer to for specific operations

Q: Is the joint torque information provided?

A: Our machines only provide the overall information of the entire joint, and do not provide the internal torque, voltage and current of the servo and motor actuator. The overall parameters of the robot arm are disclosed, such as repeatability, power supply voltage, etc.

Q: How do you understand the relationship between the two coordinates in the following figure?

- **查看两个坐标系之间的关系**

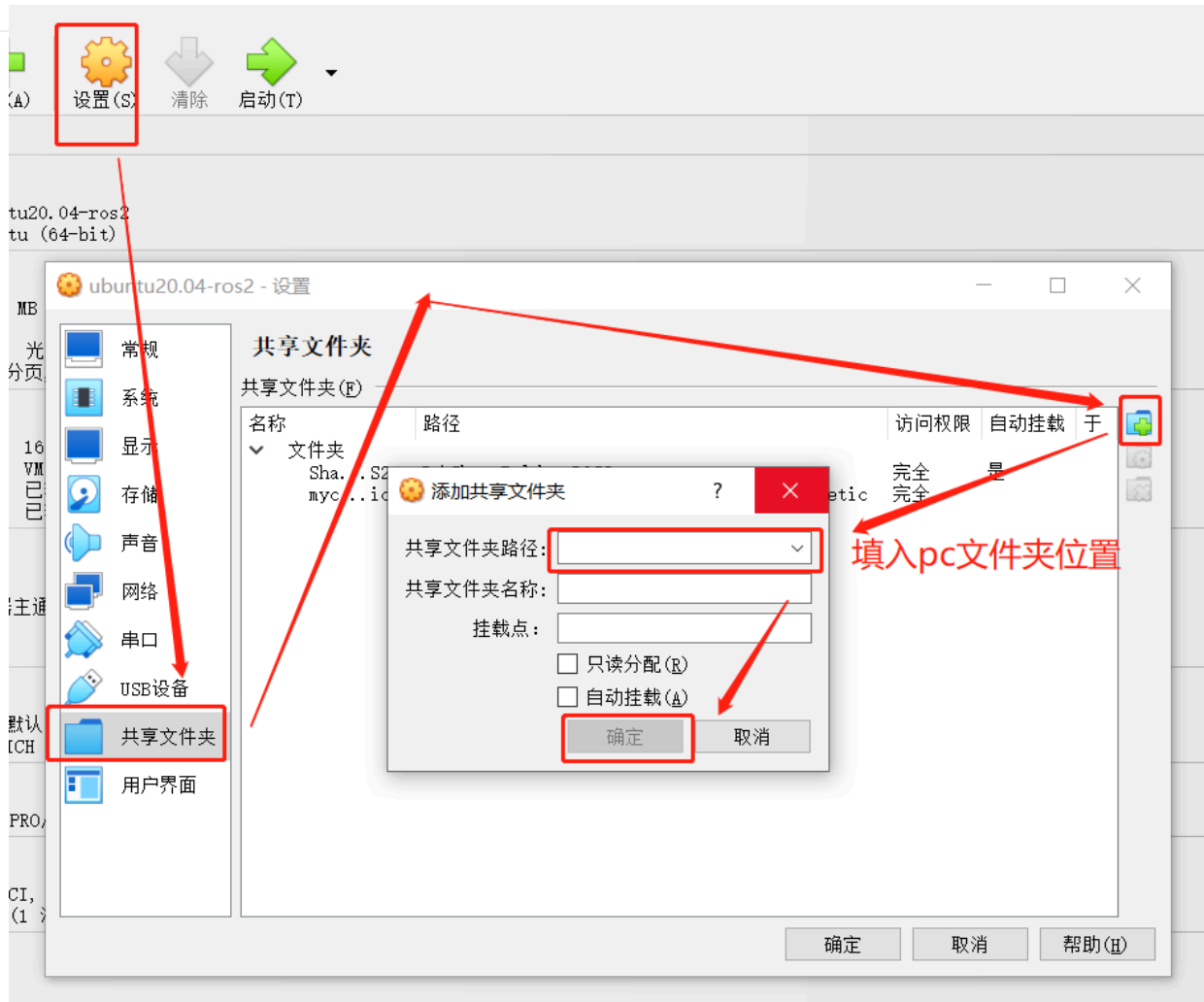
```
ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]
ros2 run tf2_ros tf2_echo turtle2 turtle1
```

A: If you want to view the transformation relationship between the coordinate system named "turtle1" and the coordinate system named "turtle2", you can use this command. In layman's terms, when you run this command, it will tell you the position and direction information of an object ("turtle1") relative to another object ("turtle2"). Just like you can know the position of a city relative to another city on a map

Q: The environment of ROS2 has been accidentally changed. Can I just delete the 280pi ROS and reinstall it myself? A: Regarding the issue of reinstalling ROS, we do not recommend users to reinstall it themselves, because the construction of the ROS environment is relatively complex and prone to errors. If you need to reset the ROS environment, we recommend users to re-write the system image. For specific methods, please refer to [Development and Use Based on ROS](#)

Q: How to transfer files from the host to the virtual machine

A: Set up a shared folder as shown below to transfer files from the PC to the virtual machine



Q: How to solve the problem of excessive repeated positioning deviation after the robot arm is in place at the same position?

Both new and old machines can adjust pid to reduce deviation as much as possible.

Appendix: <https://docs.qq.com/doc/DU0VhT2JNVUdNUEJS>, https://drive.google.com/file/d/1UWhaaSTuwLFImuEGY1J2tvngxTQDwWxK/_view?usp=sharing However, the old version of the machine has gear gaps in the 2nd and 4th joints of the robot arm, which is easy to produce joint deviations under the action of gravity, which ultimately affects the end precision. The forces of the 2nd and 4th joints in these four sets of joint values are inconsistent, so the precision is also different. It is currently recommended to adjust through the program. When the machine reaches the point, you can read the point again at this point to check if there is a deviation. On this basis, adjust the specific deviation value of the single joint to achieve the effect of reaching the specified point.

Q: What is the difference between API and serial port instructions to directly control joints?

API provides a simplified and abstract interface to make development more efficient and easy, suitable for rapid development and integration. Serial port commands provide direct, low-level control, suitable for scenarios that require fine-tuning or development of custom functions, but are usually more complex to develop and debug. In general: Using serial port commands to directly control the robot arm is more flexible, but also more complex, requiring a deep understanding of the communication protocol; while using API control is simpler and more convenient, but may be limited by the functions and performance provided by the API.

Q: Windows runs git commands and reports errors

4.1 First-time self-check

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.19045.2965]
(c) Microsoft Corporation。保留所有权利。

C:\Users\xia'yu>git clone https://github.com/elephantrobotics/pymycobot.git
'git' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\Users\xia'yu>_
```

A: This is caused by not installing git. You need to install git first and then use git commands

Q: When using Arduino with 280M5, if you use a vertical suction pump, you cannot use the 2 and 5 IO ports. These two IO ports are occupied by the serial port. You can use other IO ports to control it. You can use the following code to control it.

```
#include <MyCobotBasic.h>

void setup()
{
  pinMode(1,OUTPUT);
  pinMode(3,OUTPUT);
}

void loop()
{
  delay(100);
  digitalWrite(1,1);
  delay(100);
  digitalWrite(3,0);
  delay(100);
  digitalWrite(1,0);
  delay(100);
  digitalWrite(3,1);
  delay(1000);
}
```

Q: What is the difference between MDI and JOG?

A: MDI (Manual Data Input) is called the set value direct given operation mode. That is, after the upper controller directly sets the target position, speed, acceleration and deceleration, the axis automatically moves to the target position. MDI is also the most commonly used positioning function in practical applications. JOG moves continuously in a certain direction.

Q: What is the latest supported version of pymycobot for each model?

| 机型 | 最新支持版本 |
|-----------|---|
| 260 | 3.9.7 |
| 270 | |
| 280 | |
| 320 | |
| myAVG | |
| myArm | |
| myArm M&C | |
| UltraArm | |
| 630-pico | |
| 水星6轴 | 一层闭环(不支持头部两个舵机): 3.5.0dev5 无闭环(不支持头部两个舵机): 3.5.0dev6 完整闭环: 3.5.0dev15 一层闭环(支持头部两个舵机): 3.5.0dev8 (待测试, 未发布, 可以找开发要) |
| 水星7轴 | 无闭环: 3.5.0b19 一层闭环 (到位指令为0) : 3.5.0b14 完整闭环: 3.9.7 |

Q: How to distinguish between standard and improved DH tables

sdh, std, standard mdh, modify, improved We provide standard DH tables. Customers can convert them by themselves if necessary. They are just two different description methods.

SDH参数表

```
mycobot280:: 6 axis, RRRRRR, stdDH, slowRNE
```

| j | theta | d | a | alpha | offset |
|---|-------|--------|--------|---------|---------|
| 1 | q1 | 131.22 | 0 | 1.5708 | 0 |
| 2 | q2 | 0 | -110.4 | 0 | -1.5708 |
| 3 | q3 | 0 | -96 | 0 | 0 |
| 4 | q4 | 63.4 | 0 | 1.5708 | -1.5708 |
| 5 | q5 | 75.05 | 0 | -1.5708 | 1.5708 |
| 6 | q6 | 45.6 | 0 | 0 | 0 |

Q: How to deal with the error of missing opencv_camera?

4.1 First-time self-check

```
auto-starting new master
process[master]: started with pid [2258]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 7ae8779a-4a36-11ef-a6aa-cd6ac7cb4b28
process[rosout-1]: started with pid [2268]
started core service [/rosout]
process[robot_state_publisher-2]: started with pid [2271]
process[joint_state_publisher_gui-3]: started with pid [2272]
process[rviz-4]: started with pid [2274]
ERROR: cannot launch node of type [mycobot_280pi/opencv_camera]: Cannot locate n
ode of type [opencv_camera] in package [mycobot_280pi]. Make sure file exists in
package path and permission is set to executable (chmod +x)
process[detect_marker-6]: started with pid [2275]
process[following_marker-7]: started with pid [2277]
^C[following_marker-7] killing on exit
```

A: The error message shows that the executable permission is missing. You may need to add the permission.



Change to using `mycobot_280` instead of `pi` itself, because the `m5` side has occupied the file. Both sides cannot occupy it at the same time, otherwise it will cause the subsequent compilation to fail.

```
<node name="opencv_camera" pkg="mycobot_280pi" type="opencv_camera" args="$(arg num)"/>
```

Q: How to check the data transmission speed? A: Use the following code:

```
import time
from pymycobot.mycobot import MyCobot
mc = MyCobot("COM8", 115200, debug = True)
while 1:
    mc.get_angles()
```

4.1 First-time self-check

```
13:39:38.670 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:38.764 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:38.764 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:38.856 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:38.857 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:38.949 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:38.950 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.042 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.043 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.138 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.139 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.233 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.234 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.327 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.328 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.422 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.422 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.517 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.518 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.611 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.611 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.703 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.704 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.796 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.797 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.891 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.892 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:39.986 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:39.986 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:40.080 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:40.080 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:40.173 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:40.174 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:40.267 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
13:39:40.268 DEBU [pymycobot.generate] _write: fe fe 2 20 fa
13:39:40.363 DEBU [pymycobot.generate] _read : fe fe e 20 34 a1 1c 41 2e 50 11 68 cc 81 ee 5b fa
```

Write indicates the acquisition command sent, read indicates the returned message, and the left side indicates the time. Here, 518write + 611read are displayed, indicating that a get_angles read was completed in about 100ms, with a frequency of 10hz.

Q: The indicator light of the adapter is not on

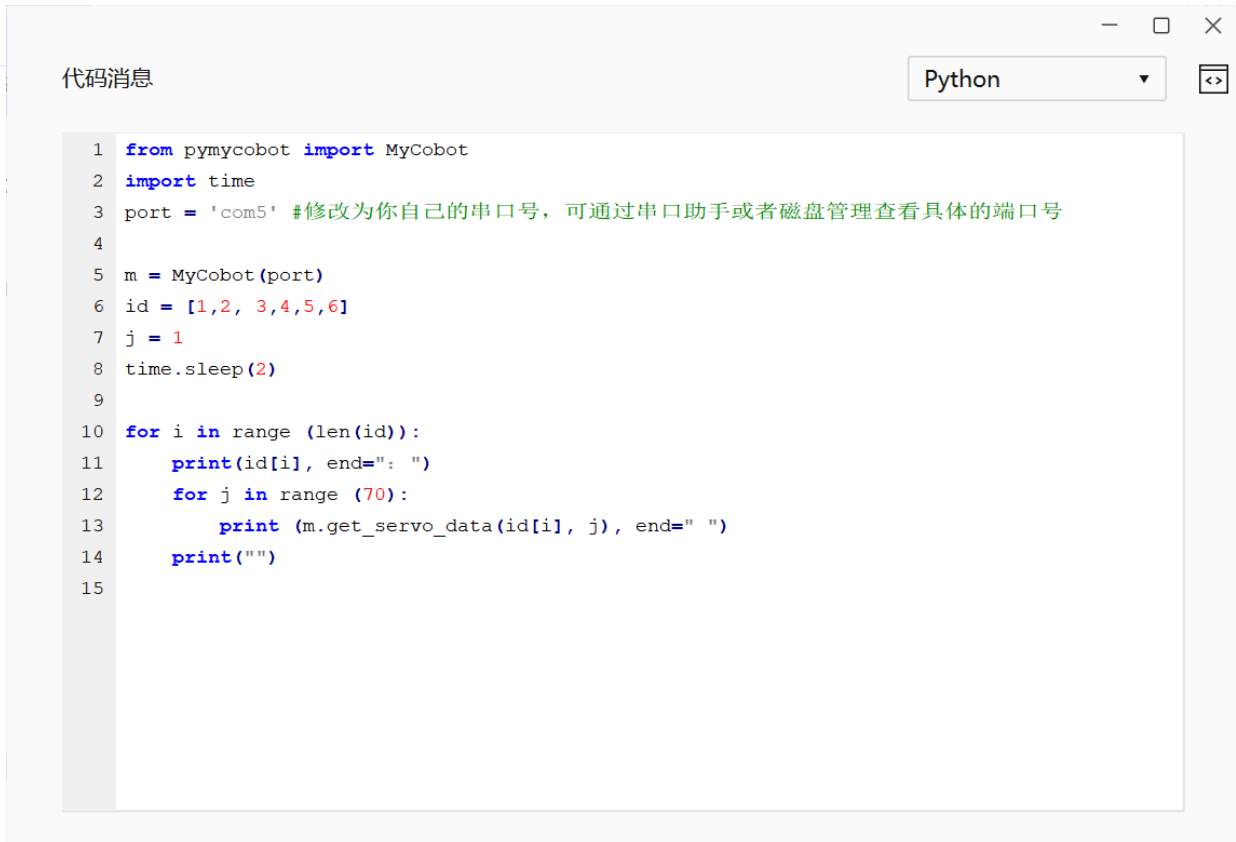
A: It is possible that the adapter is powered off for self-protection after a short circuit. Disconnect the adapter for a few minutes before using it. If it does not work after a few minutes, wait a little longer. After 15 minutes, power on the adapter separately to see if it lights up.

Q: A joint of the robot arm cannot move. How to deal with it?

A: You can use a python script to read the angle in a loop, and then manually turn the joint to see if the angle changes.

If there is a return value, check the following points and return the information to the technical support staff.

1. Use get_servo_status to check if the J2 servo has hardware problems such as undervoltage/overvoltage
2. Manually turn J2 to see if there is obvious resistance, and compare it with other joints; enable focus_servo(2) on J2 separately
3. Use the script to check if there is any problem with the parameters



```
1 from pymycobot import MyCobot
2 import time
3 port = 'com5' #修改为你自己的串口号，可通过串口助手或者磁盘管理查看具体的端口号
4
5 m = MyCobot(port)
6 id = [1,2, 3,4,5,6]
7 j = 1
8 time.sleep(2)
9
10 for i in range (len(id)):
11     print(id[i], end=" ")
12     for j in range (70):
13         print (m.get_servo_data(id[i], j), end=" ")
14     print("")
15
```

Q: End zero position abnormality

A: After using the adaptive gripper to grip objects for a long time, the gripper and end zero position abnormality will occur, and the gripper needs to be stationary.

Q: What is forward kinematics and inverse kinematics?

A: Forward kinematics refers to solving the position and posture of the robot's end effector (such as the gripper of the robot arm) in Cartesian space when the angles (or displacements) of each joint of the robot are known. It is implemented in the `get_coords()` API, but the specific algorithm is not public. Inverse kinematics is the opposite of forward kinematics. It refers to solving the angles (or displacements) of each joint of the robot when the position and posture of the robot's end effector in Cartesian space are known. `write_coords()`, `send_coords()`

Chapter 5 Basic Functions

This chapter mainly explains the basic functions and usage of the product and the use of basic software. This chapter is very important and should be read carefully in order. Before actually applying the robot, please make sure you understand the described operations correctly.

The myCobot 280 for Arduino version is developed and used based on PC and development board. There is no built-in system inside the robot arm, so the robot arm, PC and development board need to be combined during use. So please prepare a PC and development board before use

myCobot 280 For Arduino uses ATOM Matrix ESP32 Development Kit core development board as the main control. It is equipped with the robot motion control operating system firmware independently developed by Elephant Robot, with built-in kinematic forward and inverse solutions. It supports the combination of various Arduino development boards for extended application development. Using the Arduino development board with a variety of sensors officially supported by Arduino, you can quickly build a robot arm. The preferred assistant for machine vision and robot applications, control logic development, robot fusion development, and ROS simulation experiment classrooms, helping you quickly start learning and applying six-axis robot arms.

Please follow the following chapters to learn the basic functions of the myCobot 280 For Arduino version of the robot arm.

[System Usage Introduction](#) This section will briefly introduce the development environment of the 280 For Arduino robot arm. Because the myCobot 280 for Arduino version does not have a built-in development environment, you need to use a PC to install the development environment for the robot arm.

[MyStudio Introduction](#) The one-stop service platform myStudio integrates myCobot software resources and various materials, and supports firmware downloads and updates. Provides product usage video tutorials and maintenance/repair information.

[Hardware Interface Description](#) This section will introduce the common hardware interface of the 280 For Arduino version robot

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

Development Environment Construction

1 How to Build the Environment

1 Arduino IDE Download



Arduino IDE Download Address

- [Arduino Official Website](#)
- [Windows X64](#)
- [Mac OS X](#)
- [Linux ARM 64](#)

2 Installed Drivers

Before burning the program, M5Core host (including M5Stack-basic/GRAY/M5GO/FIRE/FACES) / According to the operating system you use, for the device user's **microcontroller type**, please download the corresponding driver package. After unzipping the package, select the installation package for the corresponding OPERATION system bit.

For Mac OS, make sure the system settings are correct **Preferences --> Security and Privacy --> General** and allow users to obtain from the App Store or approved developers.

Download **M5Stack-basic** serial port driver **CP210X** or **CP34X**

CP210X

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

After unzipping the zip package, select the corresponding installation package to be installed according to the **operating system** of the computer (select x64 or x86 for win10 and win11 for installation).

4.1 First-time self-check



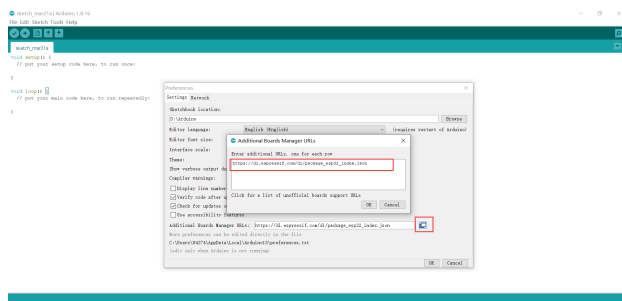
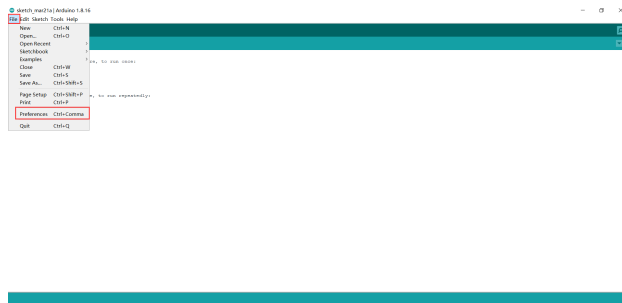
CP34X

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

3 Add board

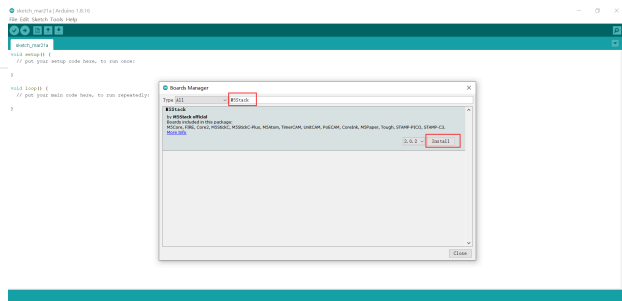
- Open Arduino IDE and select **File --> Preferences --> Settings** to add the following url to the additional board manager:

https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json

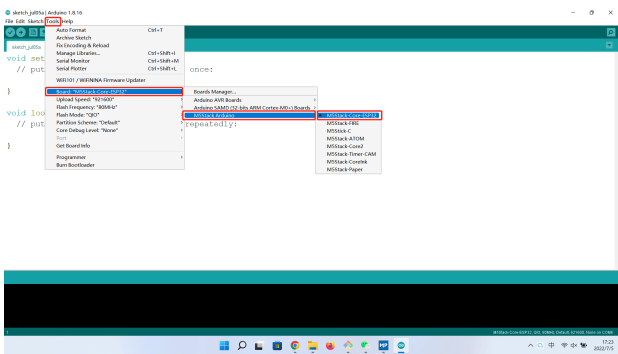


- After adding, click **Tools --> Board --> Boards Manager**, in the new pop-up dialog box, Enter and search for **M5Stack**, click "Install" (if the search fails, you can try to restart the **Arduino** program), as shown below:

4.1 First-time self-check



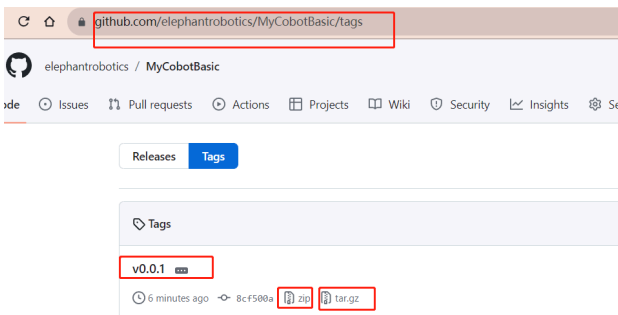
- After adding, click **Tools --> Board**, check if it is successful, as shown below:



4 Add related libraries

Note: Please download the latest library, the first version is v0.0.1.

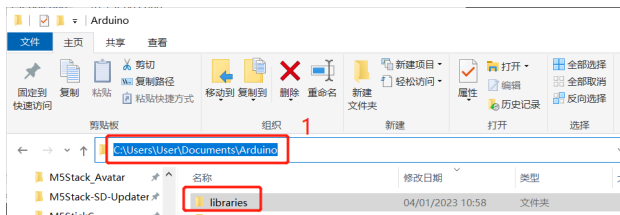
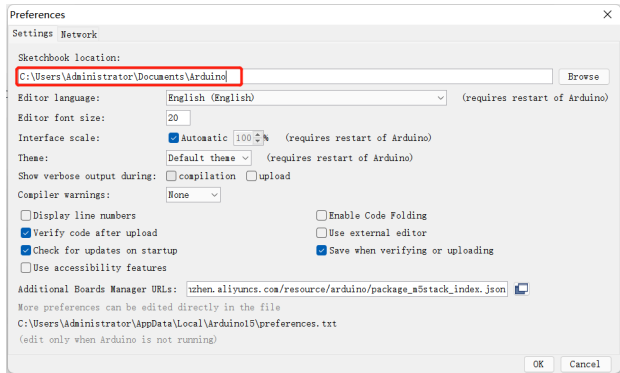
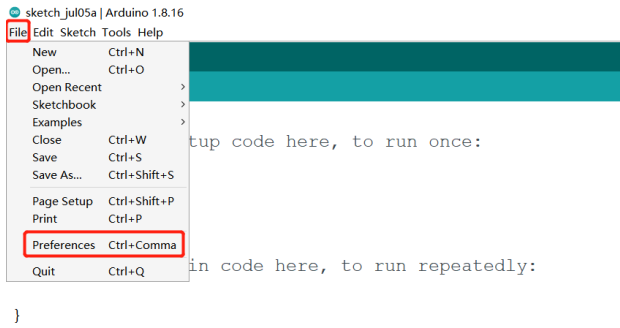
- Click to download related dependent libraries **-MycobotBasic**(After importing the Mycobot280-Arduino model, you can refer to the **arduino_use** section for use). Please see the following figure for details. .zip is for Windows system, and .tar.gz is for Linux system:



- Library installation instructions

4.1 First-time self-check

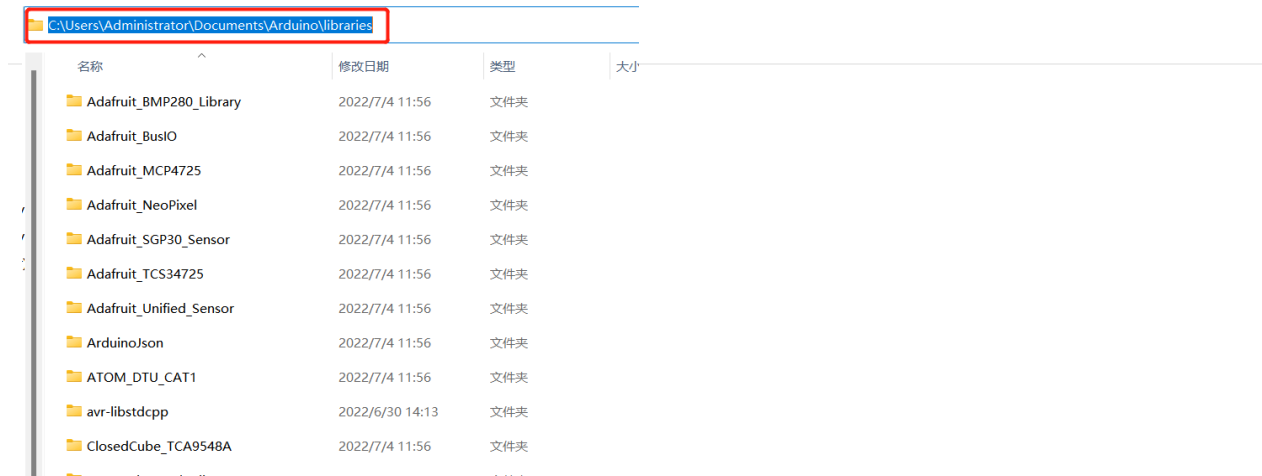
First check the location of the Arduino project folder by clicking **File --> Preferences** (you can copy the path to your hard drive to find the library folder)



1 Copy the path here and press enter to find the library folder

Unzip it to the appropriate folder in the library directory. If you are using **Arduino**, do not overwrite it, just add to the existing **library**.

4.1 First-time self-check

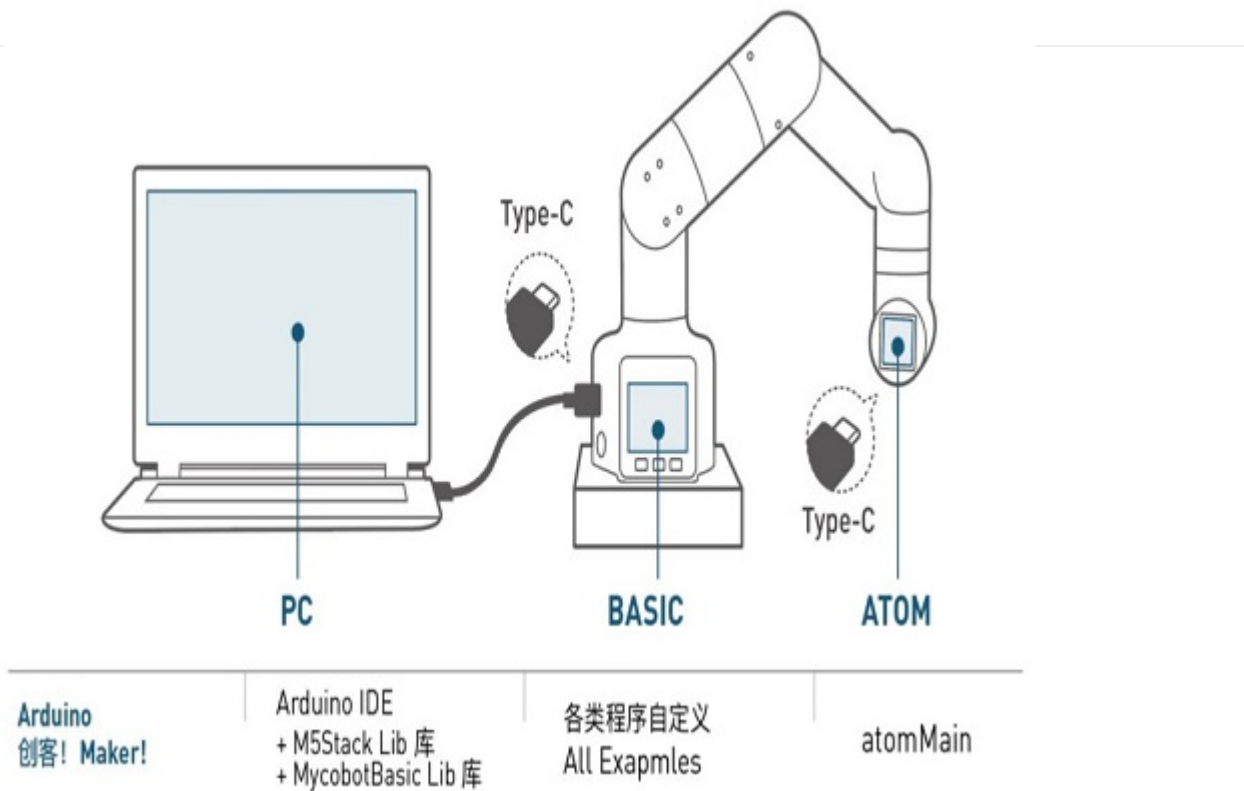


Congratulations, you have built an **Arduino** related development environment.

Note: Arduino environment configuration and case compilation can be seen in our B station video (<https://www.bilibili.com/video/BV1Vi4y1c7DQ/>).

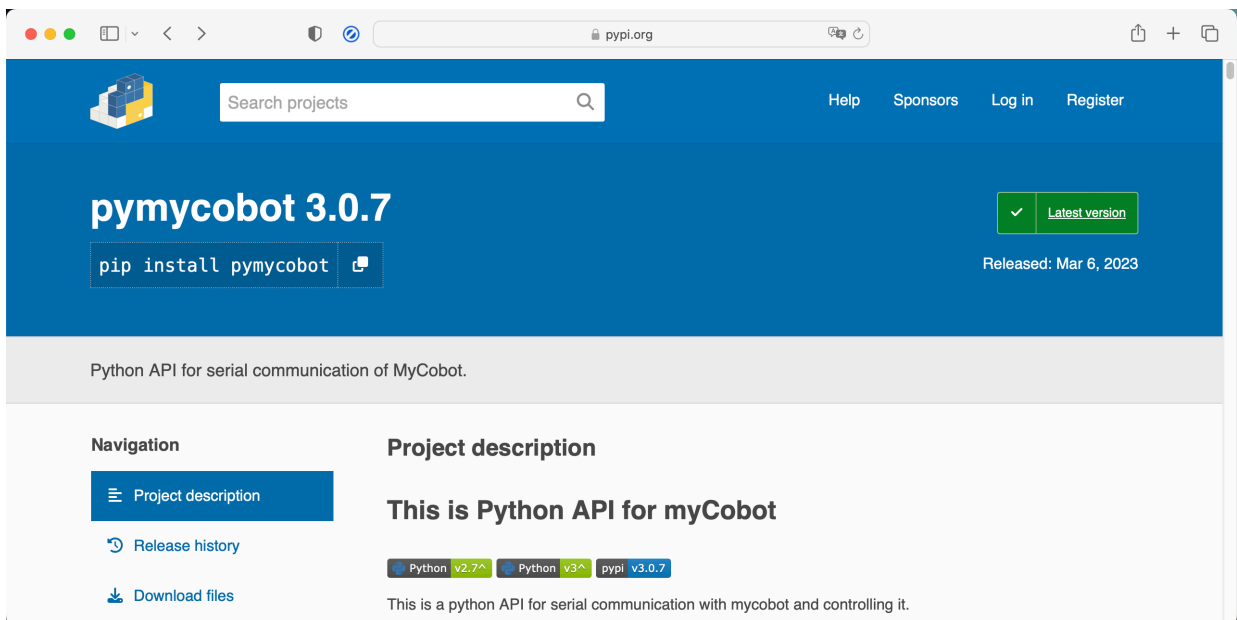
2 Development and use based on Arduino

Arduino is an easy-to-use, easy-to-use open source electronic prototyping platform, including hardware (various development boards that meet Arduino specifications) and software (Arduino IDE and related development packages). The hardware part (or development board) consists of a microcontroller (MCU), flash memory, and a set of general input/output interfaces (GPIO). You can think of it as a microcomputer motherboard. The software part mainly consists of the Arduino IDE on the PC side, the related board support package (BSP), and a rich third-party function library. Users can easily download the BSP and required function libraries related to the development board you have through the Arduino IDE to write your program. We have an open source library MyCobotBasic, which is an open source robot control library developed by our company. It can only be used with robots developed by our company. Using this library, you can control our robot through Bluetooth, WiFi, serial port, etc., and it also supports external sensors, IIC communication, LED lights and other functions. You can DIY different application scenarios according to your needs, or refer to the MiniRobot sample code or angle, coordinate, gripper and other control cases we provide. The MiniRobot sample code contains control-related content such as Bluetooth, WiFi, drag teaching, and distance sensors. After the user installs the Arduino environment, you can directly view the simple use of Arduino and related interfaces. For details, please refer to the Arduino Development Guide section.



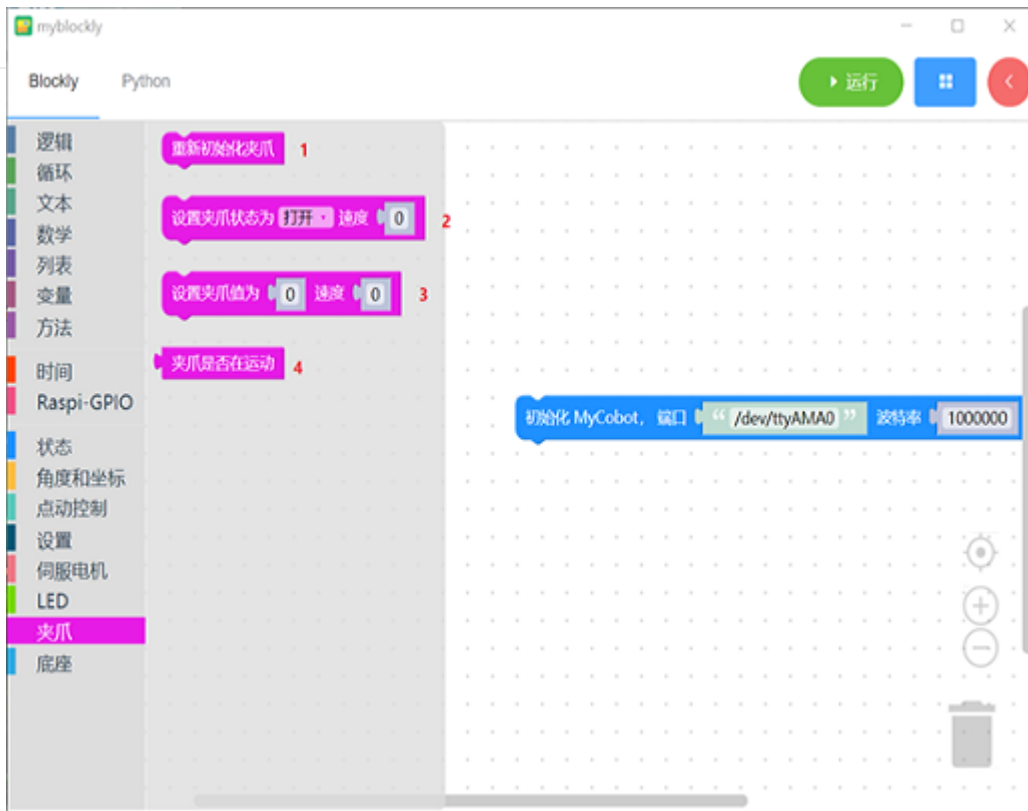
3 Development and use based on Python

Our robots support Python, and the development of Python API libraries is becoming more and more complete. You can control the robot's joint angles, coordinates, grippers, etc. through Python. Refer to **python development related chapters** for more information.



4 Development and use based on Blockly

myBlockly is a fully visual modular programming software, which belongs to a graphical programming language. For details, please refer to the relevant chapters based on myBlockly development.



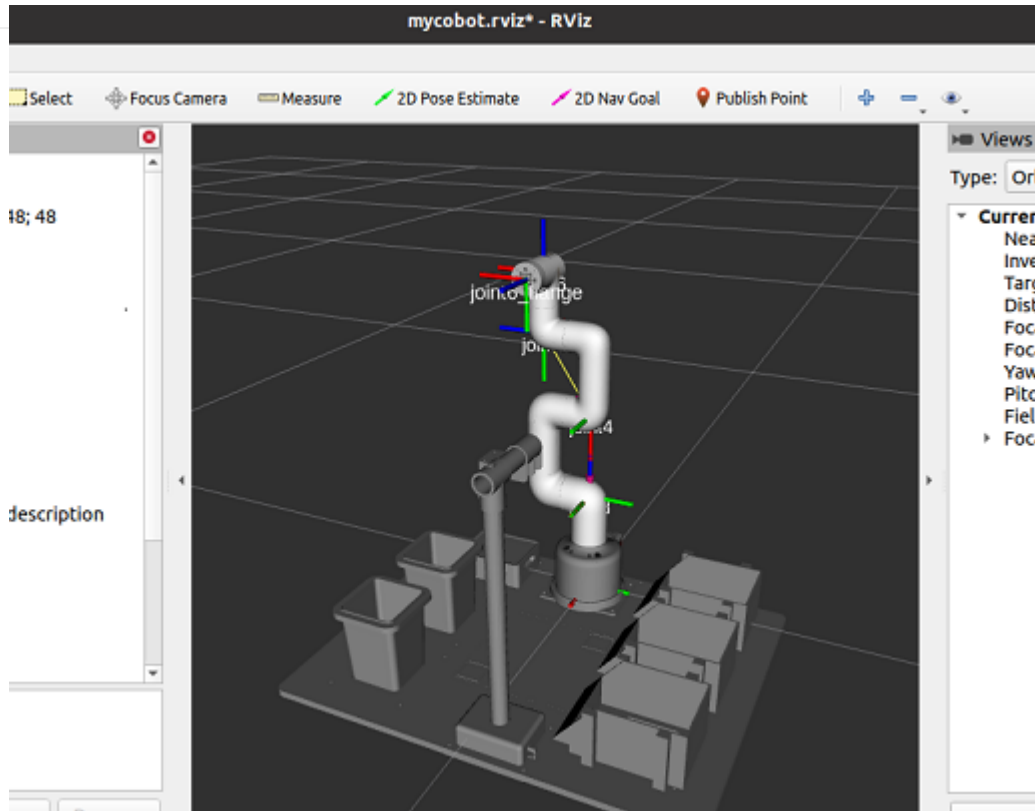
5 Development based on ROS

ROS is open source and is a post-operating system, or secondary operating system, for robot control. Through ROS, we can achieve simulation control of the robot arm in a virtual environment. We will use the rviz platform to visualize the robot arm and use a variety of methods to operate our robot arm; use the moveit platform to plan and execute the robot arm's motion path to achieve free control of the robot. The effect of the arm.

The emergence of ROS solved the communication problem of various parts of the robot. Later, more and more robot algorithms were integrated into ROS. **ROS2** inherited **ROS** and is more powerful and better than **ROS**. Compared with **ROS**, which only supports Linux systems, **ROS2** also supports **windows**, **mac**, and even **RTOS**

4.1 First-time self-check

platforms. For more details about ROS and ROS2 development, please refer to the relevant chapters.



myStudio



[myStudio video tutorial](#)

1 Original 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 them according to their own usage scenarios, and at the same time learn related teaching materials and browse tutorial videos online.

2 Latest version of myStudio and supported platforms

- Latest version: V3.5.8
- Applicable to: Windows, Mac, Linux

3 myStudio functions

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

4 Applicable devices for myStudio

- myCobot 280
- myCobot 280 M5

4.1 First-time self-check

- myCobot 280 PI
 - myCobot 280 Jetson Nano
 - **myCobot 280 for Arduino**
-

5 Firmware version recommendation

Different models of robotic arms require different firmware to be burned. The following are the firmware versions recommended for burning different models of robotic arms.

myCobot 280 series

The myCobot 280 series has 4 versions: M5 version, PI version, arduino version and jetsonnano version. Different versions have different core models, and the firmware and versions required to be burned are also different.

4.1 First-time self-check

| Robot version number | Core | Required firmware to be burned | Recommended firmware and version |
|----------------------|----------------|--------------------------------|--|
| M5 version | M5Stack-Basic | miniRobot firmware | Recommended to burn v2.1 version, you can use drag teaching, wifi, Bluetooth and other functions |
| | Atom | atomMain firmware | For products with serial numbers ER28001202200415 and earlier, or products without serial numbers, it is recommended to burn v4.1; for products with serial numbers ER28001202200416 and later, it is recommended to burn v5.1 |
| PI version | RaspberryPI 4B | ubuntu firmware | Recommend burning V18.04. version |
| | Atom | atomMain firmware | For products with serial numbers ER28001202200415 and earlier, or products without serial numbers, it is recommended to burn v4.1; for products with serial numbers ER28001202200416 and later, it is recommended to burn v5.1 |
| Arduino version | mega2560 | transponder firmware | Recommend burning v1.0 version |
| | mkrwifi1010 | transponder firmware | Recommend burning v1.0 version |
| | Atom | atomMain firmware | For products with serial numbers of ER28001202200415 and earlier, or products without serial numbers, it is recommended to burn v4.1; for products with serial numbers of ER28001202200416 and later, it is recommended to burn v5.1 |
| Jetson nano version | JestonNano | ubuntu firmware | Recommend burning V18.04. version |
| | Atom | atomMain firmware | For products with serial numbers of ER28001202200415 and earlier, or products without serial numbers, it is recommended to burn v4.1; for products with serial numbers of ER28001202200416 and later, it is recommended to burn v5.1 |

myStudio environment setup

myStudio download and installation

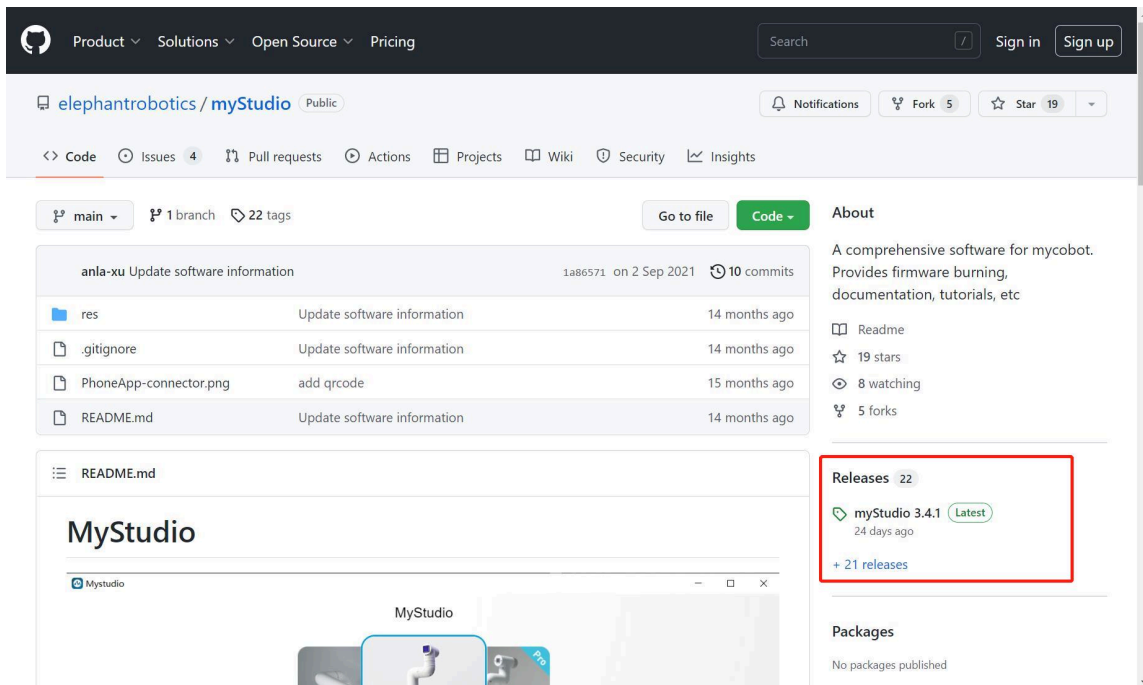
Note: The installation path of myStudio cannot have any spaces

Download address:

1. [GitHub address](#)

4.1 First-time self-check

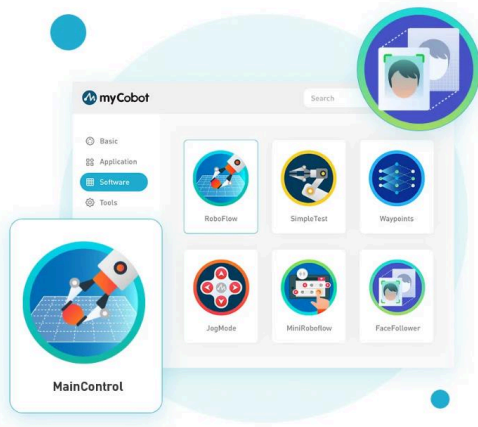
- After entering the download address, click `myStudio` on the right and select the corresponding version to download.



- Different suffixes represent different systems, please download the corresponding version:
- `*.tra.xz` — Linux system
- `*.dmg` — Mac system
- `*.exe` — Window system

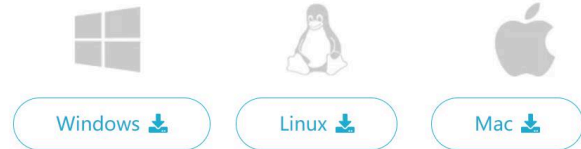
2. Official website address

You can download it according to your computer system.

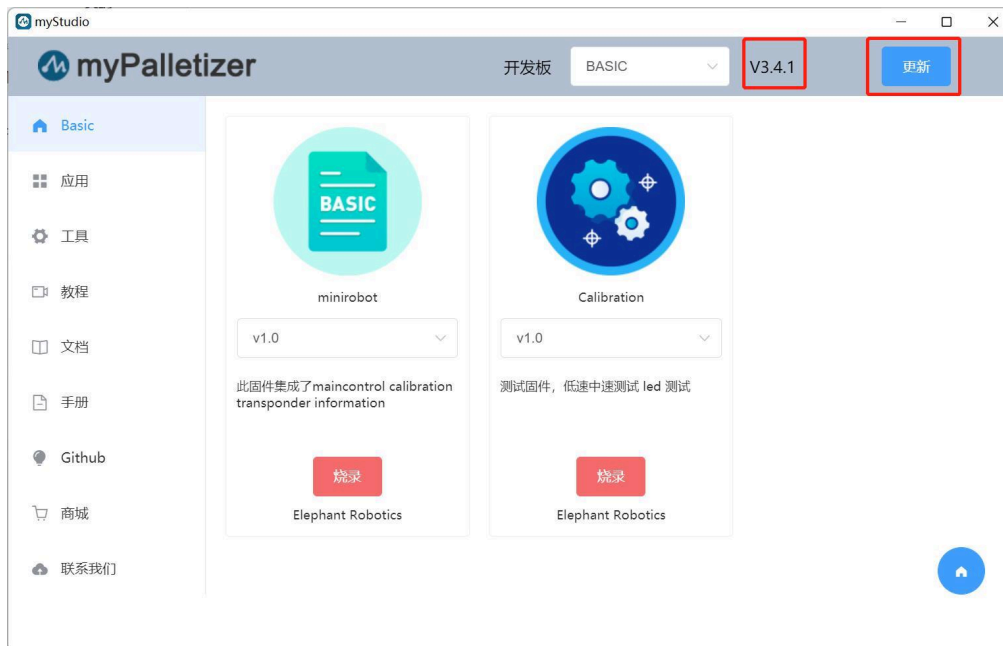


myStudio

myStudio是一个一站式的机器人的使用平台。
myStudio整合了myCobot的软件资源及各类资料，主要功能为：
1) 下载更新固件；
2) 查看机器人使用视频教程；
3) 维护和维修方面的信息(如视频教程、Q&A等)



Note: Please download the latest version. You can view the current version in the downloaded myStudio interface and update to the latest version.



Install the driver

Users can click the button below to download the corresponding **CP210X** or **CP34X** driver package according to their operating system. After decompressing the package, select the installation package corresponding to the operating system bit number for installation.

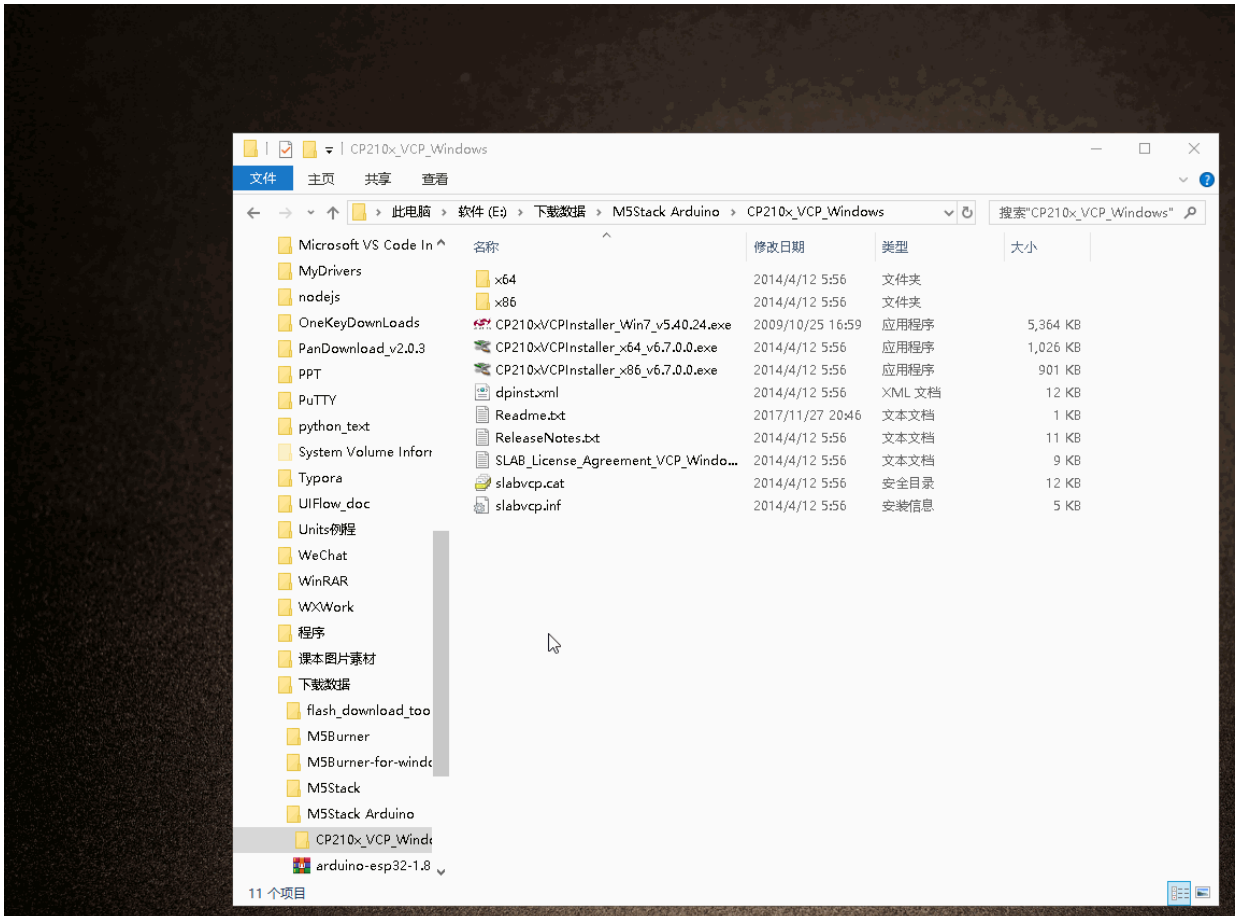
There are currently two driver chip versions, **CP210X** (for CP2104 version) and **CP34X** (for CH9102 version) driver compression packages. If you are not sure which USB chip your device uses, you can install both drivers at the same time. (**CH9102_VCP_SER_MacOS** may show an error during installation, but it has actually been installed, so just ignore it.)

For Mac OS, make sure the system "Preferences->Security and Privacy->General" is set before installation, and allow downloads from the App Store and approved developers.

- Download the bottom **M5Stack-basic** serial port driver
- **CP210X**

4.1 First-time self-check

- [Windows10](#)
- [MacOS](#)
- [Linux](#)
- [CP34X](#)
- [Windows10](#)
- [MacOS](#)
- Download the terminal **Atom** serial port driver
- [Windows10](#)



How to distinguish CP210X and CP34X chips

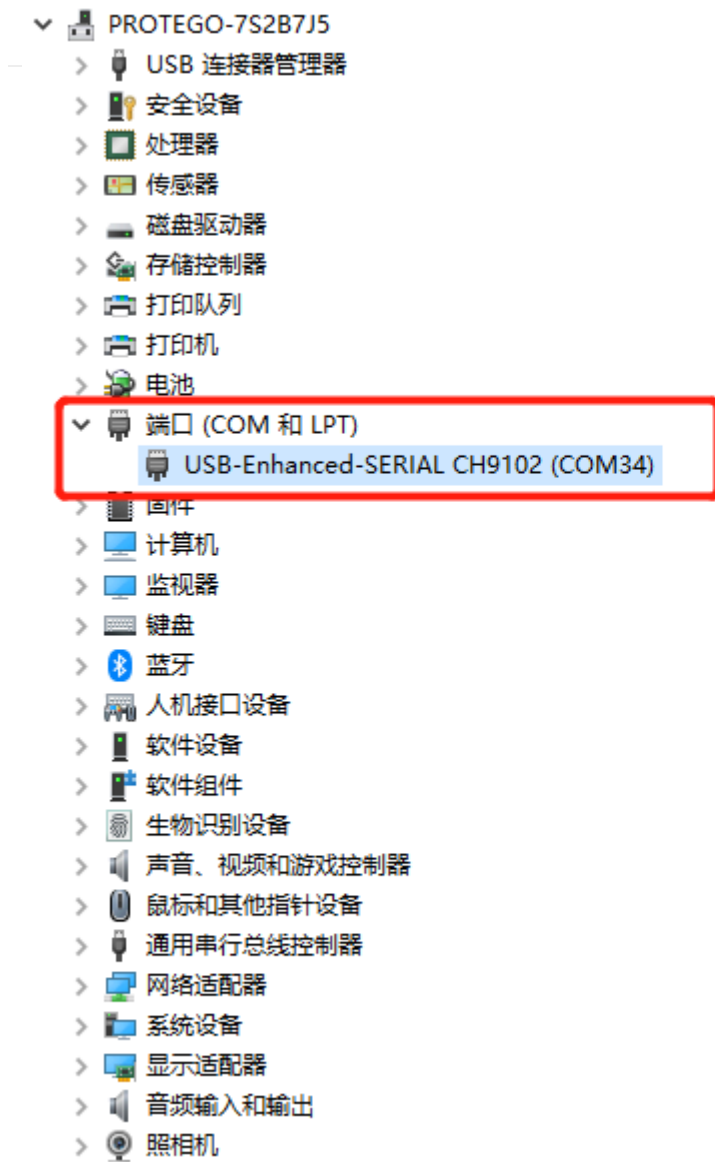
- As shown in the figure below, open **Device Manager** and check **Ports (COM and LPT)**

4.1 First-time self-check

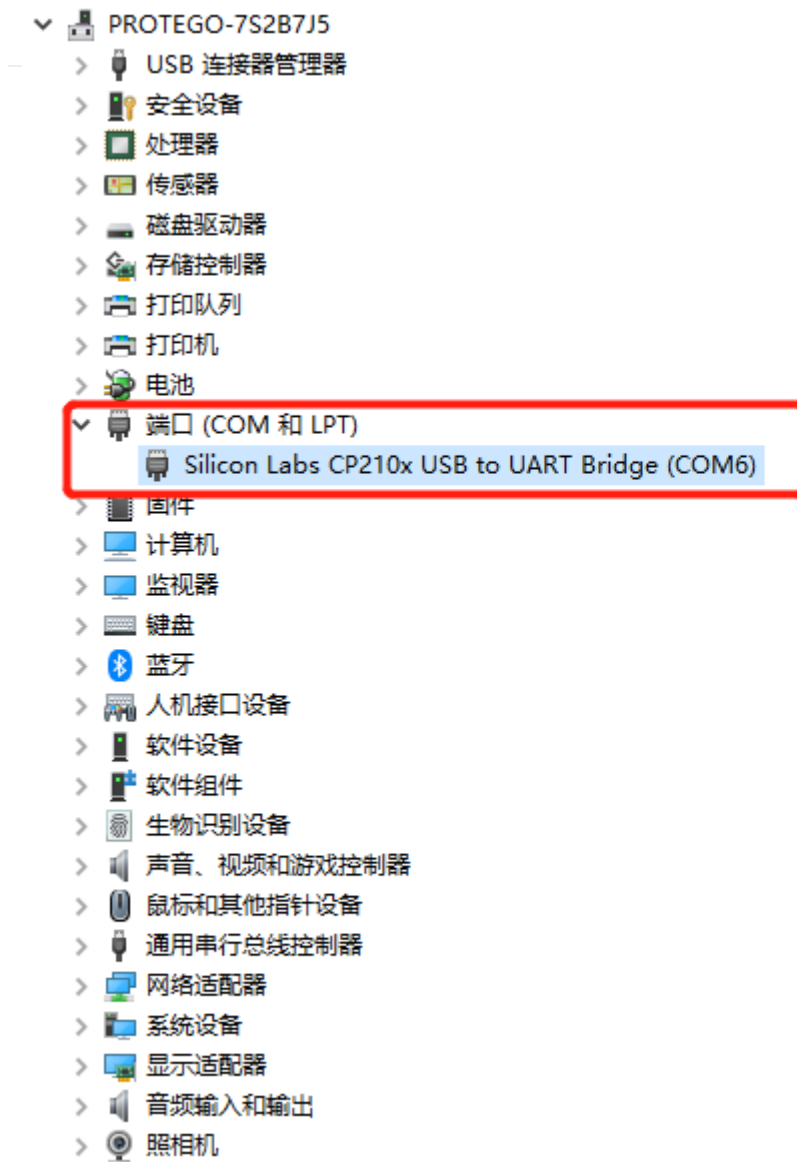


- If Port (COM and LPT) shows **USB-Enhanced-SERIAL CH9102**, it is **CP34X** chip

4.1 First-time self-check



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



Burn and update firmware

[myStudio video tutorial](#)

Introduction to Motor PID

Modification of the PID parameters can be a trade-off between the control accuracy of the robotic arm and the stability of the movement, we provide two sets of PID parameters, which are suitable for occasions requiring high stability of the movement, and for occasions requiring high precision of the movement.

Applies to PID parameters with smooth motion, where motion accuracy is affected by false positives:

```

...

This document is applicable to the myCobot 280 series of robotic arms.
The function is to modify the joint motor configuration parameters
and verify the modification results. Please ensure that the robot
control port is not occupied when using it. If you encounter failure,
please run the file again.

#Import dependent library files.
import time
from pymycobot import *

#Define data address and data.

#Define the serial port of the robotic arm, Please check your robot model and fill in the corresponding content.
# _port = '/dev/ttyAMA0'
_port = 'COM30'
_baud = 115200

#Instantiate a hardware serial port.
try:
    mc = MyCobot280(_port, _baud)
except Exception as e:
    print(e)
    print("Error: The current serial port can not be used, please check whether the serial port serial number is correct,
    exit()

#Loop modification and display modification results.
#joint_id = 1 - 6.
for i in range(1,6):
    mc.set_servo_data(i,23,0)
    time.sleep(1)
    print(mc.get_servo_data(i,23))
input("The program ends, please press any key to exit.")
exit()

```

PID parameters suitable for high-precision scenarios will cause the arm to continuously adjust the steady state error:

```
...

This document is applicable to the myCobot 280 series of robotic arms.
The function is to modify the joint motor configuration parameters
and verify the modification results. Please ensure that the robot
control port is not occupied when using it. If you encounter failure,
please run the file again.

#Import dependent library files.
import time
from pymycobot import *

#Define data address and data.

#Define the serial port of the robotic arm, Please check your robot model and fill in the corresponding content.
# _port = '/dev/ttyAMA0'
_port = 'COM30'
_baud = 115200

#Instantiate a hardware serial port.
try:
    mc = MyCobot280(_port, _baud)
except Exception as e:
    print(e)
    print("Error: The current serial port is not available, please check whether the serial port serial number is correct")
    exit()

#Loop modification and display modification results.
#joint_id = 1 - 6.
for i in range(1,6):
    mc.set_servo_data(i,23,4)
    time.sleep(1)
    print(mc.get_servo_data(i,23))
input("The program ends, please press any key to exit.")
exit()
```

Chapter 6 Software Development Guide

1 Usage Environment

The myCobot 280 for Arduino version is developed and used based on PC and development board. There is no built-in system inside the robot arm, so the robot arm, PC and development board need to be combined during use. Please prepare a PC and development board before use. At the same time, the myCobot 280 for Arduino version does not have a built-in development environment, so you need to use a PC to install the development environment for the robot arm.

2 Development Environment

In order to meet the diverse application needs of robots in different scenarios, we have adapted the robot to multiple programming languages. So far, we have adapted the following mainstream programming languages, and we believe that you can use any of the following languages for development. Please be sure to follow the instructions strictly. Any omitted steps may cause the corresponding language to fail to run successfully. I wish you a smooth use of the robot.

6.1 Arduino

Arduino is an easy-to-use, open-source electronic prototyping platform. Users can use the Arduino IDE to easily download the BSP and required function libraries related to the development board you have to write your program. MyCobotBasic library is an Arduino open-source robot control library developed by our company. This library can be used to control our robot through Bluetooth, WiFi, serial port, etc.

6.2 Python

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

6.3 ROS1

ROS (Robot Operating System), as an open-source 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 is joint control, path planning or sensor feedback, ROS provides corresponding tools and libraries to make the control process more flexible and efficient.

6.4 ROS2

ROS 2 (Robot Operating System 2) is a flexible software framework designed for robot software development. Our robot can make application development more efficient and modular through a series of services and functions such as hardware abstraction, device drivers, library functions, visualization tools, messaging, and package management.

6.5 Blockly

myBlockly is a puzzle-style programming software developed by the R&D team of Shenzhen Elephant Robot Company based on the python environment and the pymycobot dependency library, which allows users to program and control the mycobot robot in a building block-like way.

6.6 Serial Communication

If you have a certain understanding of information theory, coding, and robot communication functions, then you should understand that all communications originate from data transmission. In order to facilitate users to operate the robot, we have opened a communication protocol based on serial communication. You can use the serial assistant or encapsulate it into any programming language you are familiar with to control the robot.

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

Arduino



What is Arduino?

Arduino is an easy-to-use, easy-to-use open source electronic prototyping platform, which includes hardware (various development boards that meet Arduino specifications) and software (Arduino IDE and related development kits). The hardware part (or development board) consists of a microcontroller (MCU), flash memory (Flash) and a set of general input/output interfaces (GPIO), etc. You can think of it as a microcomputer motherboard. The software part mainly consists of the PC-side Arduino IDE and related board support packages (BSP) and rich third-party function libraries. Users can easily download the BSP and required function libraries related to the development board you have through Arduino IDE to write your program.

What can MyCobotBasic library do?

MyCobotBasic library is an open source robot control library developed by our company. It can only be used with robots developed by our company. With this library, you can control our robots through Bluetooth, WiFi, serial ports, etc., and it also supports external sensors, IIC communication, LED lights and other functions. You can DIY different application scenarios according to your needs, or refer to the MiniRobot sample code or angle, coordinate, gripper and other control cases we provide. The MiniRobot sample code contains control-related content such as Bluetooth, WiFi, drag teaching, and distance sensors.

Applicable devices:

- myCobot 280
- myCobot 280 M5
- myCobot 280 for Arduino

Arduino development guide

You can use it according to the following instructions Arduino develops our robotic arm

1. [Environment setup](#)

2. [Simple use](#)
 3. [API description](#)
-

Arduino development board connection guide

The myCobot 280 for Arduino version is developed and used based on PC and development board. **There is no built-in system inside the robot**, so the robot, PC and development board need to be combined during use, so please prepare a PC and development board before use. **Please connect the PC and development board before development.**

1. Power on the robot first (if the green light is on, it means the robot is powered on)



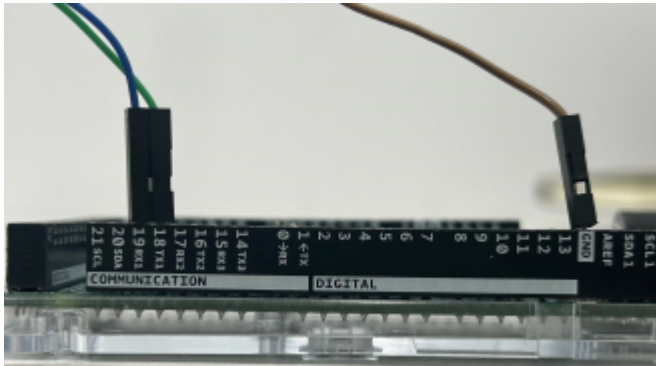
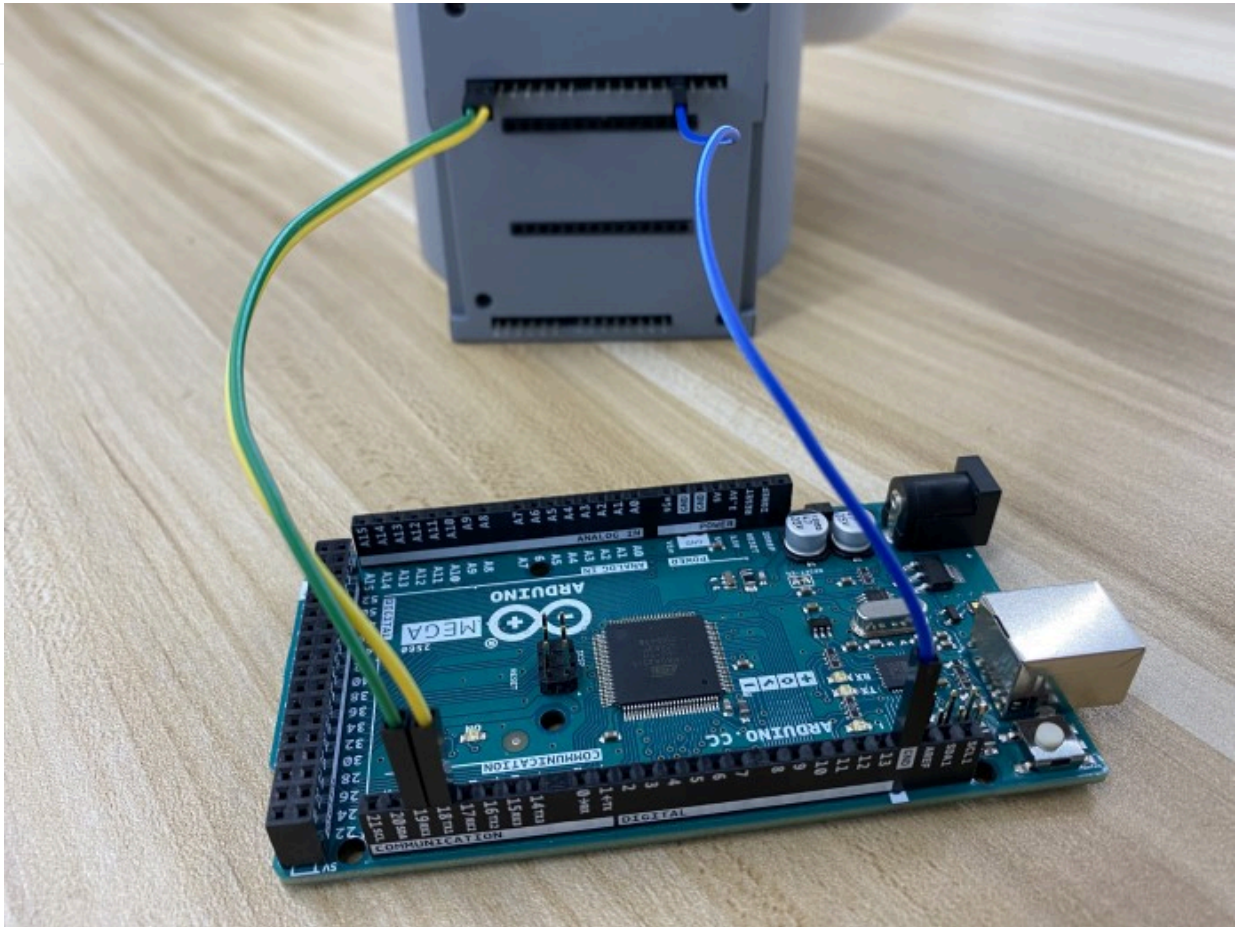
myCobot 280 for Arduino connection

4.1 First-time self-check



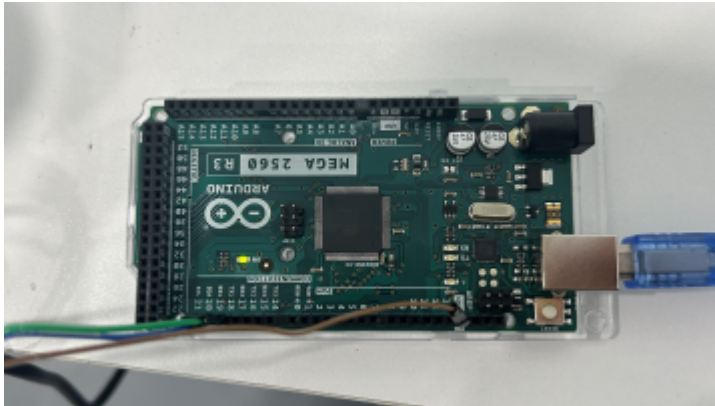
1. Connect the Arduino board (the connection position is shown in the figure below, with the development board Arduino MEGA2560 as an example)

4.1 First-time self-check



4.1 First-time self-check

1. Connect the serial port to the computer (if the green light is on, it means the computer is successfully connected)
-



After connecting, you can follow the steps below to view the connection port number

Click this computer --> Right-click to manage --> Click Device Manager --> Click on the port



If the port number is displayed normally, it means that the development board has been successfully connected to the computer and development can begin. If no port number is displayed, please check whether the connection lines are loose or have other poor connection conditions.

Arduino environment setup

Arduino IDE download



Downloads



Arduino IDE 1.8.16

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

- Windows** Win 7 and newer
- Windows** ZIP file
- Windows app** Win 8.1 or 10 
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits
- Mac OS X** 10.10 or newer

[Release Notes](#) [Checksums \(sha512\)](#)

Download **Arduino IDE** You can click [Arduino official website](#) to download and install the version corresponding to the computer system.

- [Windows X64](#)
- [Mac OS X](#)
- [Linux ARM 64](#)

Install the driver

Before burning the program, users of M5Core host (including BASIC/GRAY/M5GO/FIRE/FACES)/ **microcontroller devices** please click the button below to download the corresponding **CP210X** driver package according to your operating system. After decompressing the package, select the installation package corresponding to the operating system bit number to install.

For **Mac OS**, before installing, make sure **System Preferences-->Security & Privacy-->General** and allow **CP2104** drivers to be downloaded from the App Store and approved developers

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

4.1 First-time self-check

After unzipping the compressed package, select the corresponding installation package according to the **operating system** of the computer for installation (select x64 or x86 for win10 and win11).

此电脑 > 下载 > CP210x_VCP_Windows (1) > CP210x_VCP_Windows

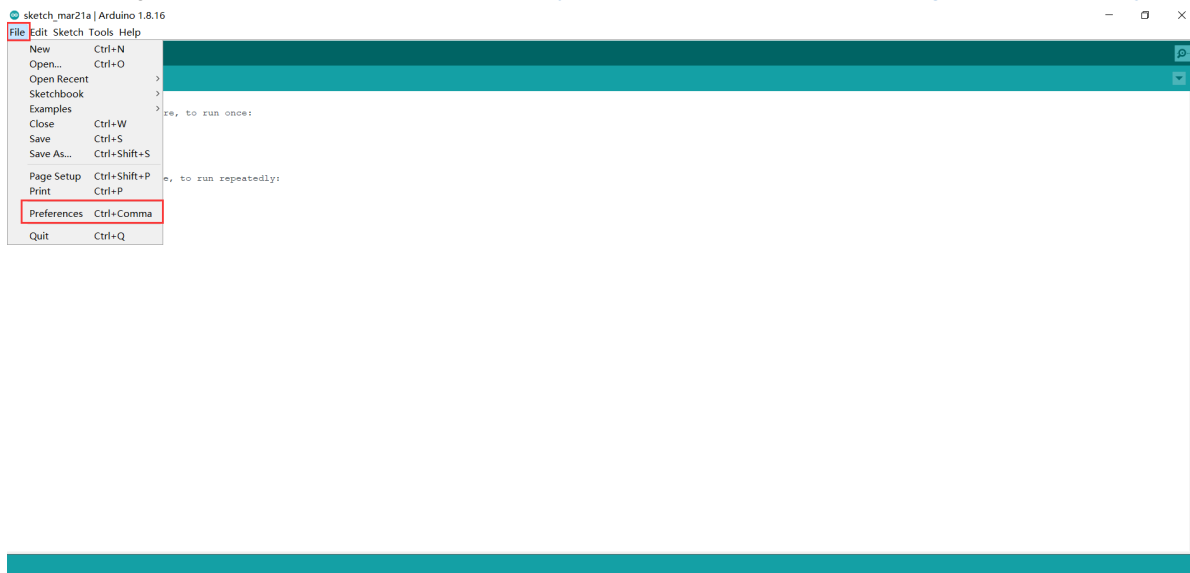
| 名称 | 修改日期 | 类型 | 大小 |
|--------------------------------------|------------------|--------|----------|
| x64 | 2014/4/12 5:56 | 文件夹 | |
| x86 | 2014/4/12 5:56 | 文件夹 | |
| CP210xVCPInstaller_Win7_v5.40.24.exe | 2009/10/25 16:59 | 应用程序 | 5,364 KB |
| CP210xVCPInstaller_x64_v6.7.0.0.exe | 2014/4/12 5:56 | 应用程序 | 1,026 KB |
| CP210xVCPInstaller_x86_v6.7.0.0.exe | 2014/4/12 5:56 | 应用程序 | 901 KB |
| dpinst.xml | 2014/4/12 5:56 | XML 文档 | 12 KB |
| Readme.txt | 2017/11/27 20:46 | 文本文档 | 1 KB |
| ReleaseNotes.txt | 2014/4/12 5:56 | 文本文档 | 11 KB |
| SLAB_License_Agreement_VCP_Wind... | 2014/4/12 5:56 | 文本文档 | 9 KB |
| slabvcp.cat | 2014/4/12 5:56 | 安全目录 | 12 KB |
| slabvcp.inf | 2014/4/12 5:56 | 安装信息 | 5 KB |

CP34X

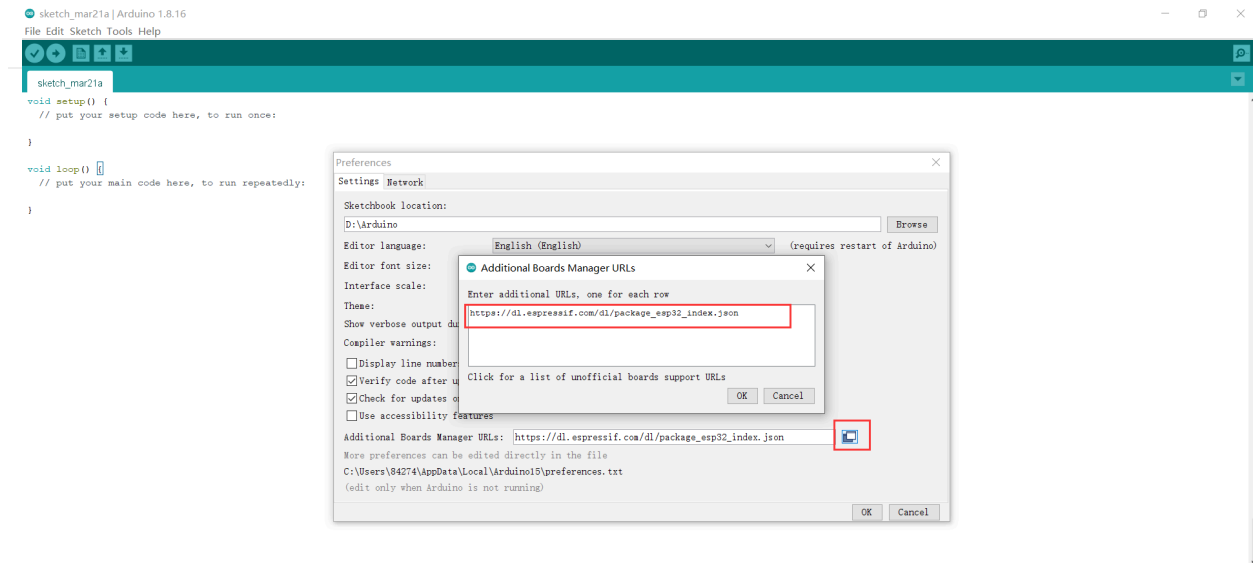
- [Windows10](#)
- [MacOS](#)

Add development board

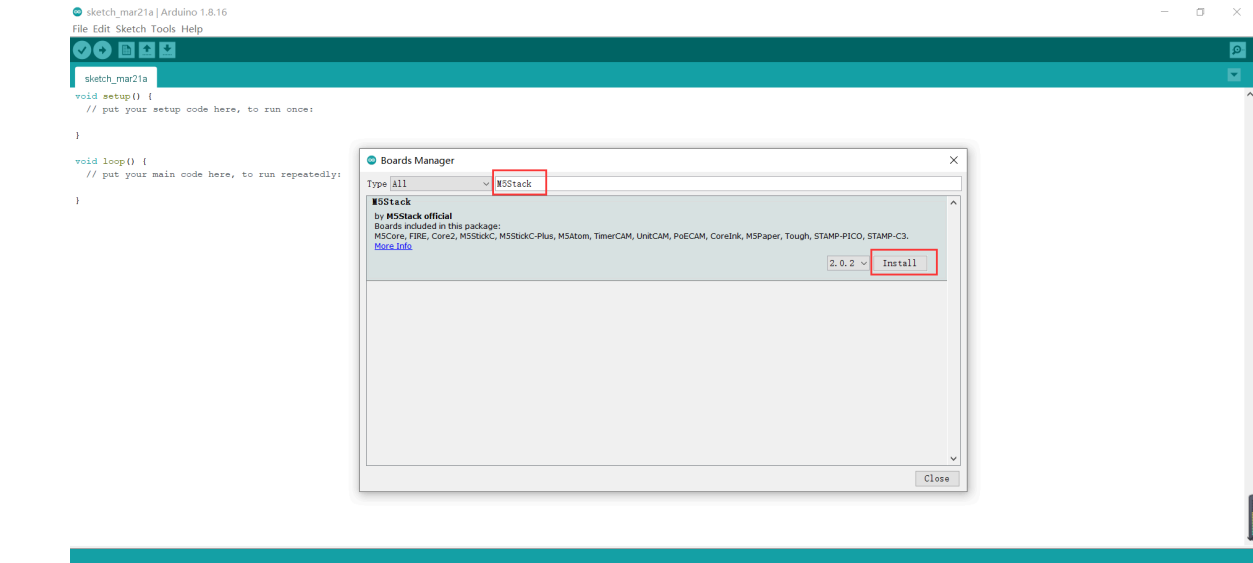
- Open Arduino IDE, select **File --> Preferences --> Settings**, add the URL below to the additional development board manager https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json



4.1 First-time self-check

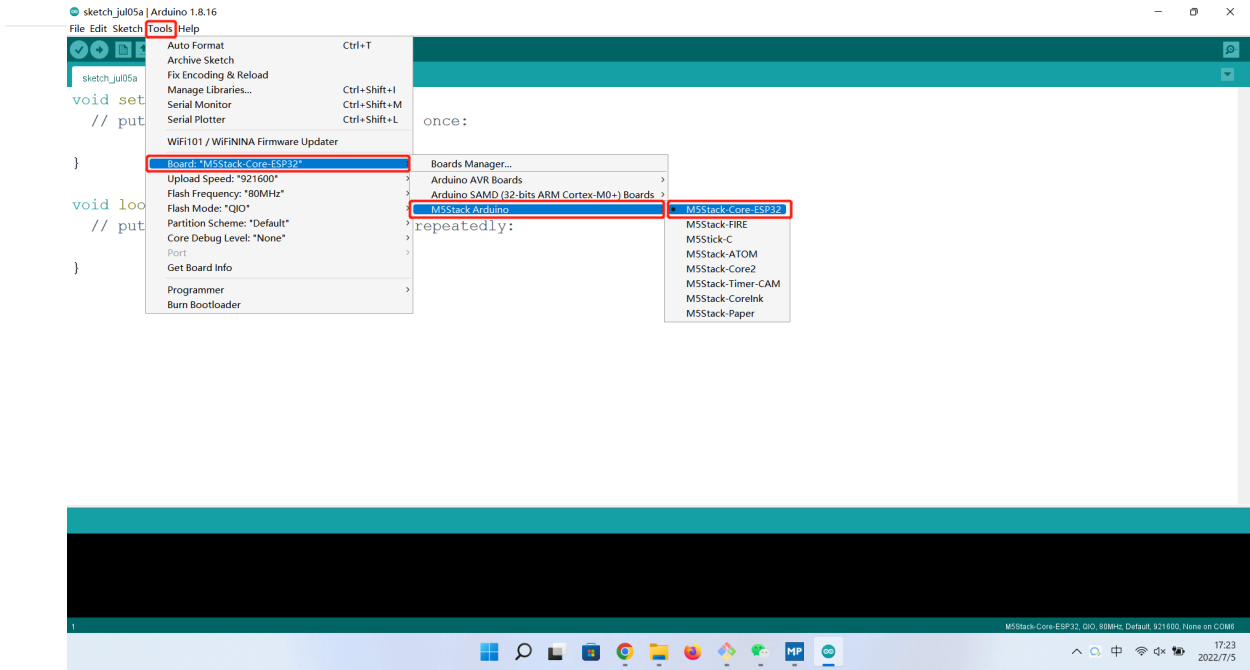


- After adding, select **Tools --> Development Edition --> Development Board Manager**, enter and search **M5Stack** in the new pop-up dialog box, and click Install (if the search fails, you can try restarting the **Arduino** program; if an error occurs during downloading, click Install again), as shown below:



4.1 First-time self-check

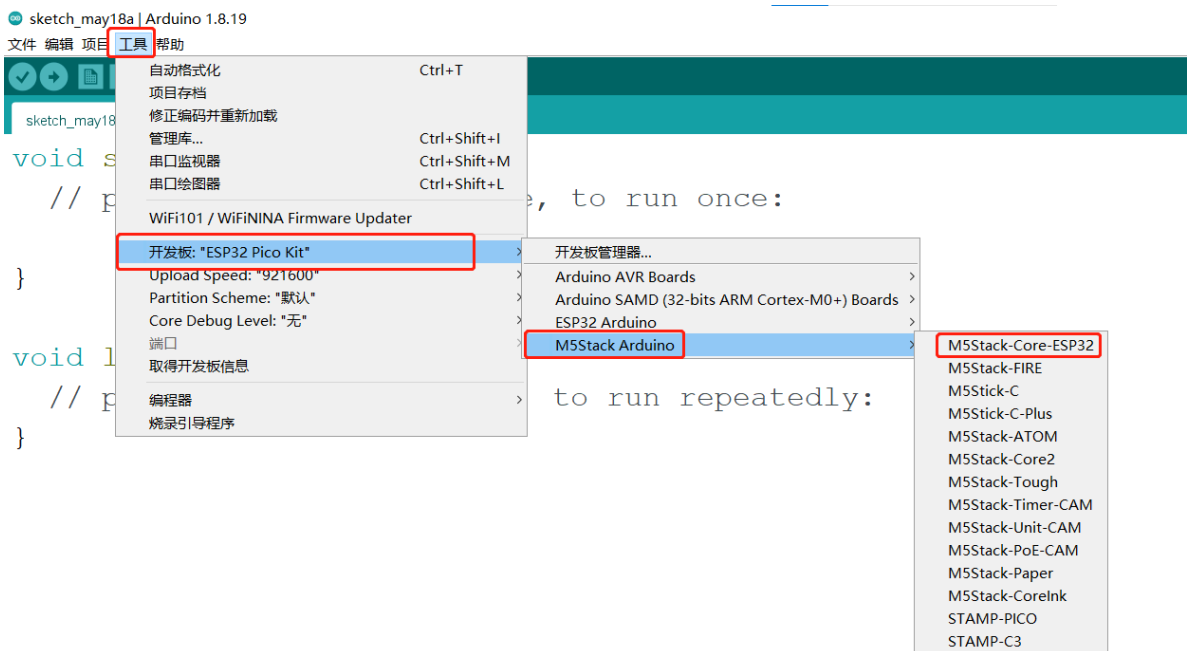
- After adding, select **Tools --> Development Board**, check if it is successful, as shown below:



Add related libraries

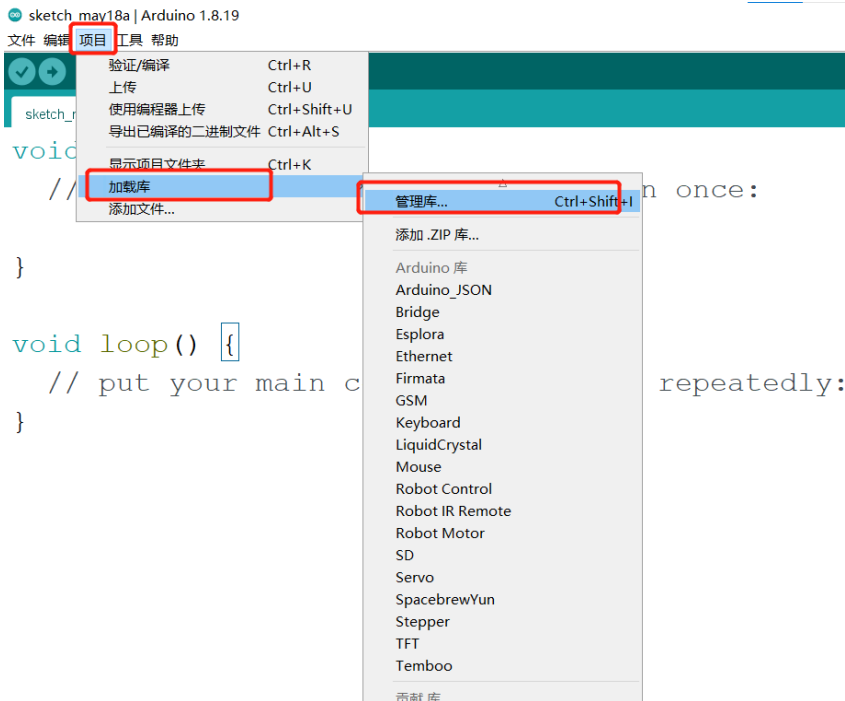
4.1 Install M5Stack library

1. Tools --> Development Board --> M5Stack Arduino Select **M5Stack-Core-ESP32**, as shown below:

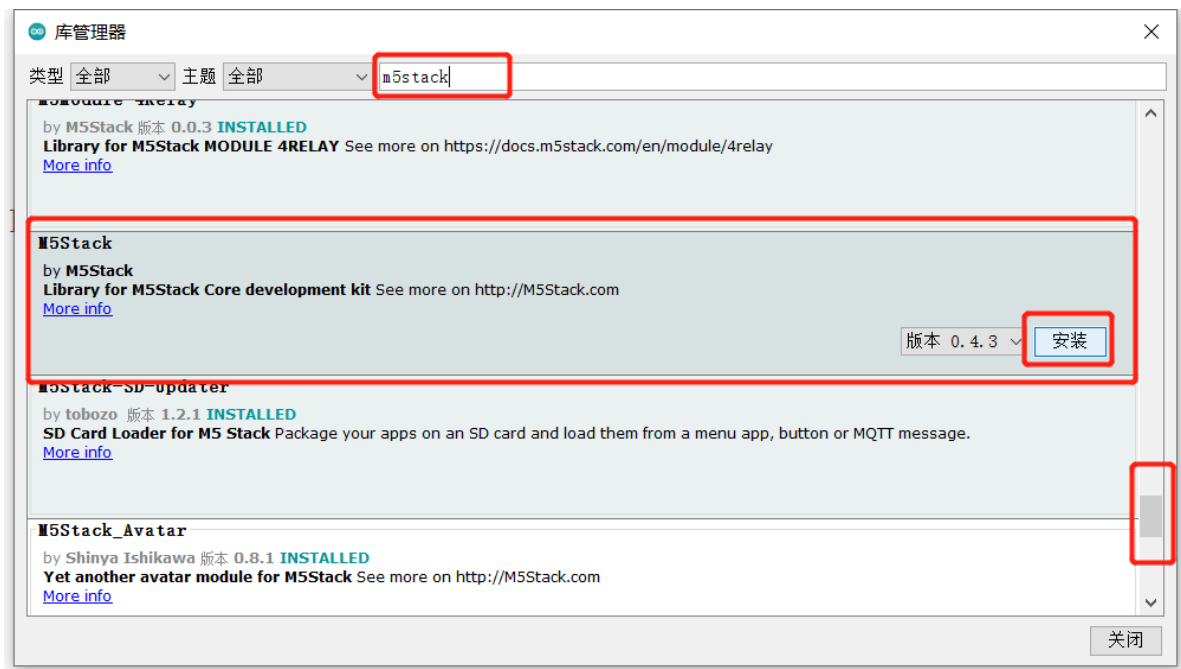


4.1 First-time self-check

2. Project --> Load Library --> Manage Library Enter **M5Stack** in the search box, as shown in the following figure:



3. Click Install after finding it, scroll down, **M5Stack** is at the back, and you can see the location of the drop-down slider in the picture, as shown in the following figure:

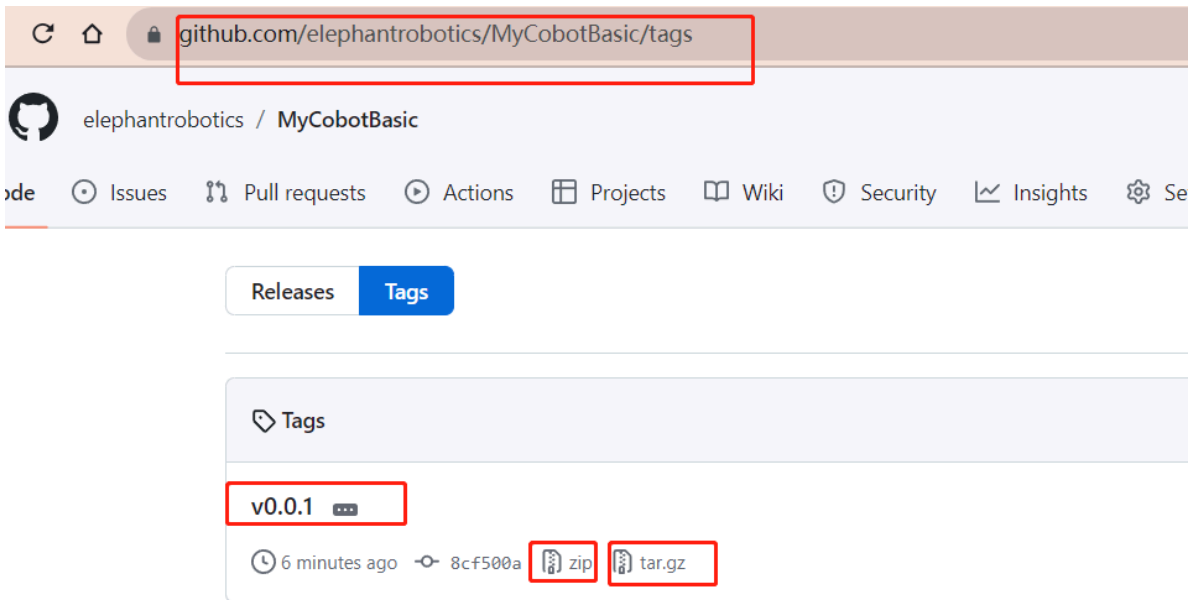


4.2 Install MyCobotBasic library

4.1 First-time self-check

Note: Please download the latest library, the first version is v0.0.1.

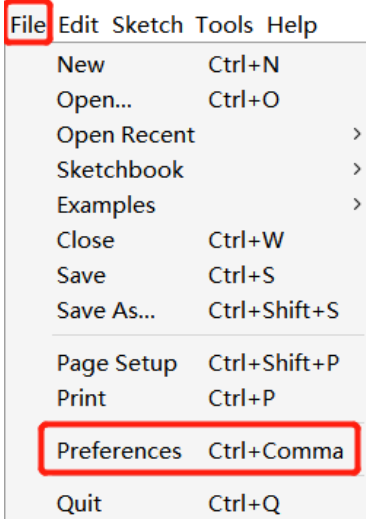
- Click to download related dependency libraries
- **MycobotBasic**(After importing the Mycobot280-Arduino model, you can refer to arduino_use for use). Please see the figure below for details. .zip is suitable for Windows system, and .tar.gz is suitable for Linux system:



- Dependent library installation instructions

First check the location of the Arduino project folder. You can view it by clicking File --> Preferences (you can copy the path to the hard disk path to find the libraries folder)

sketch_jul05a | Arduino 1.8.16

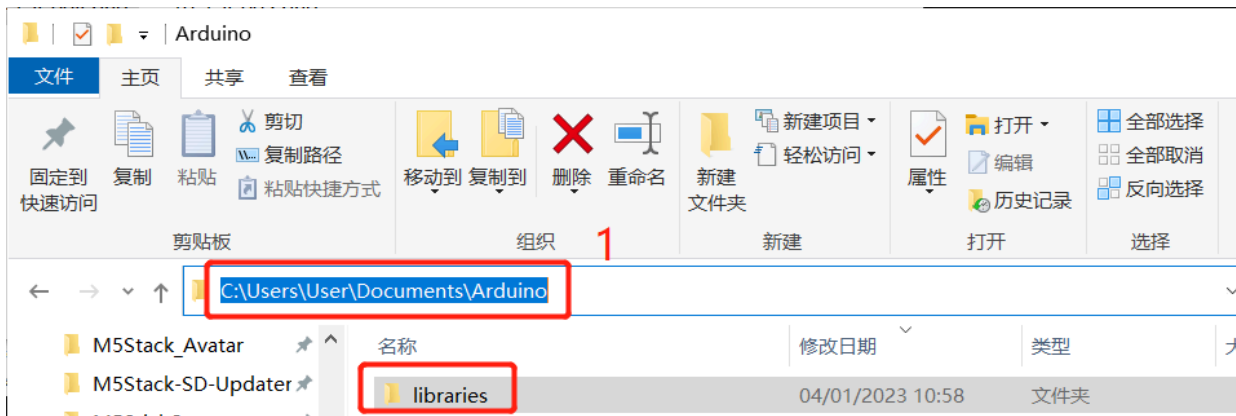
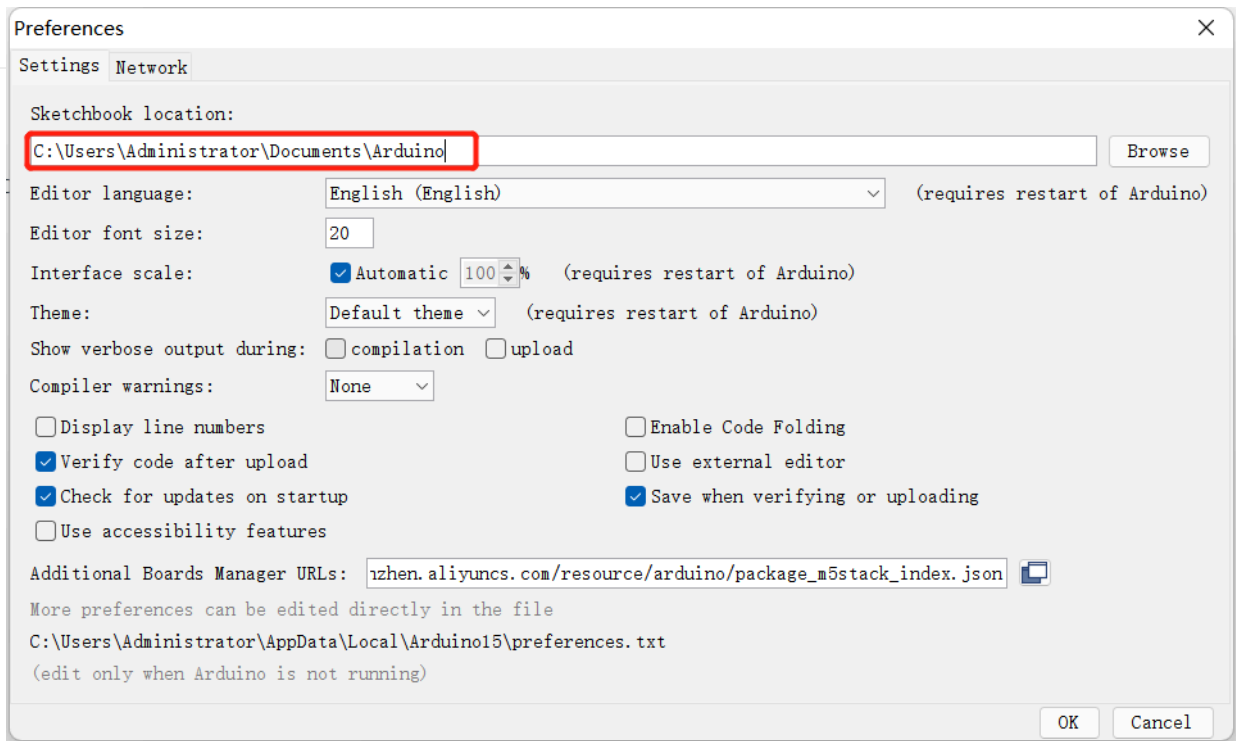


Setup code here, to run once:

in code here, to run repeatedly:

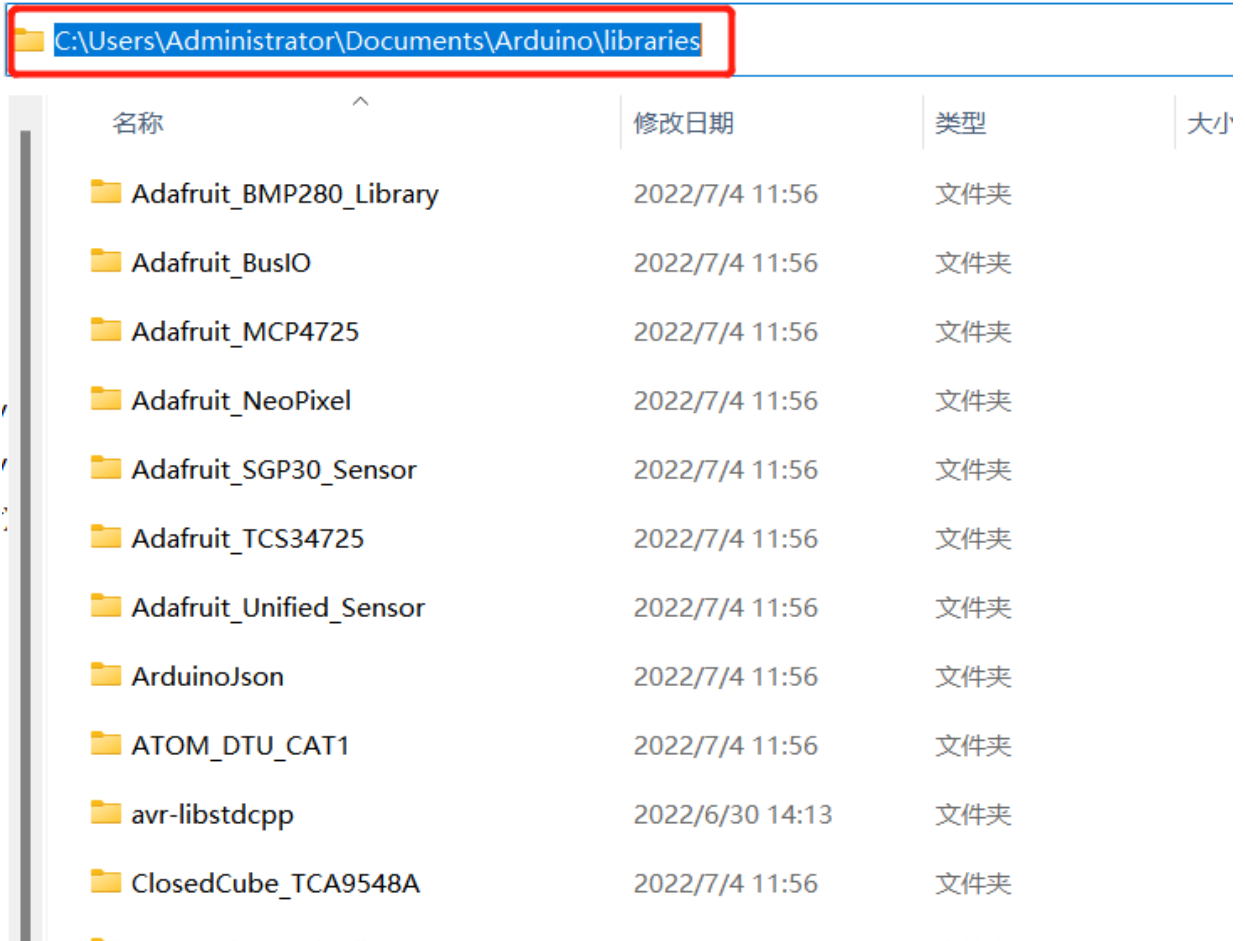
}

4.1 First-time self-check



1 Copy the path here and press enter to find the libraries folder

Unzip to the corresponding folder **libraries** directory. If you are using **Arduino**, please do not overwrite, just add it to the existing **Library**.



At this point, congratulations, you have built the **Arduino** related development environment.

Note: For Arduino environment configuration and case compilation, you can watch our video on Bilibili (<https://www.bilibili.com/video/BV1Vi4y1c7DQ/>).

Use of Arduino library

Supported robot arm type: **myCobot280-Arduino**

Use case: For example, open

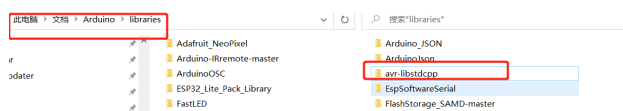
C:\Users\User\Documents\Arduino\libraries\MyCobotBasic\examples\MyCobot280\MyCobot280_Arduino\Me ga\AnglesControl\AnglesControl.ino. This case requires burning the development board first and then connecting it to the robot arm, otherwise the upload will fail.

Using the basic library at the bottom, you can freely develop under Arduino and control our company's robot arm.

1 Modifications before compiling

Import library files. If your robot arm is myCobot280-Arduino, please check whether the development board is Mega2560 or Uno. If it is:

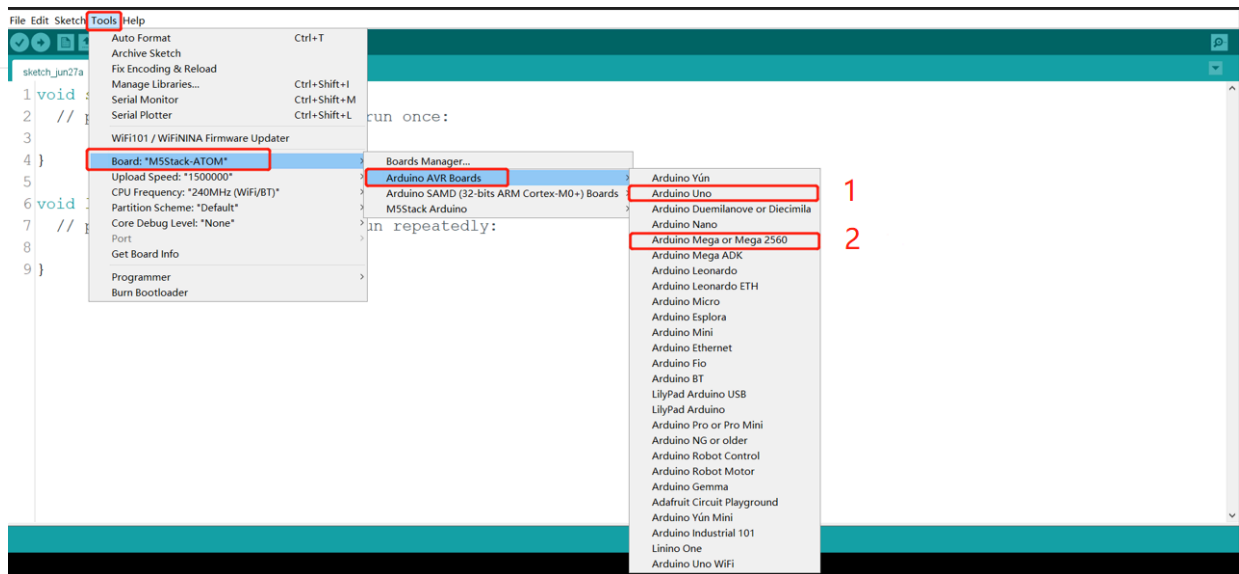
1.1 Please put **MyCobotBasic\lib\avr-libstdcpp** under **C:\Users\User\Documents\Arduino\libraries**:



2 Select the development board before compiling

2.1 The development board is Uno or Mega2560, Tools --> Development Board --> Arduino AVR Boards --> Arduino Uno (or Arduino MEAG or Mega2560), see the following figure for details:

4.1 First-time self-check

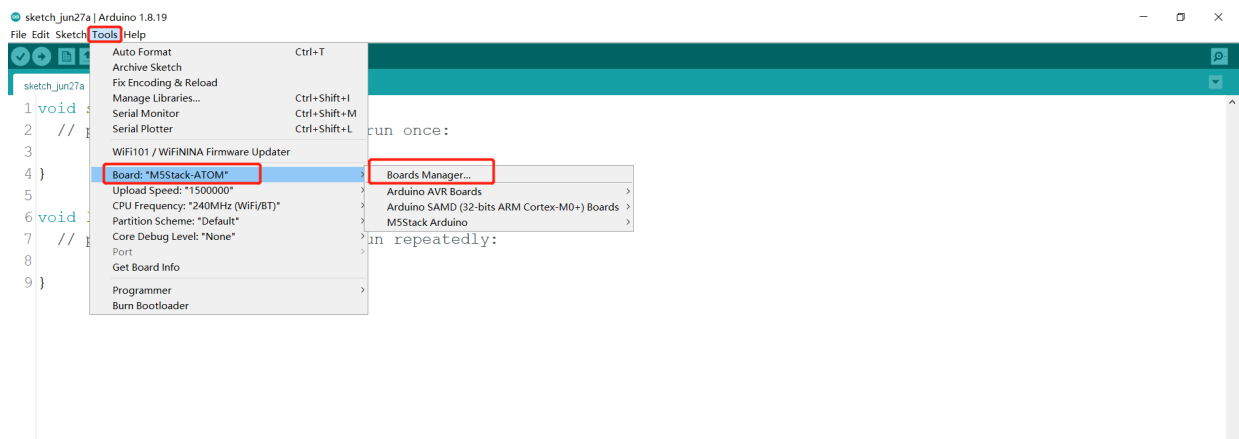


1 When using uno,select

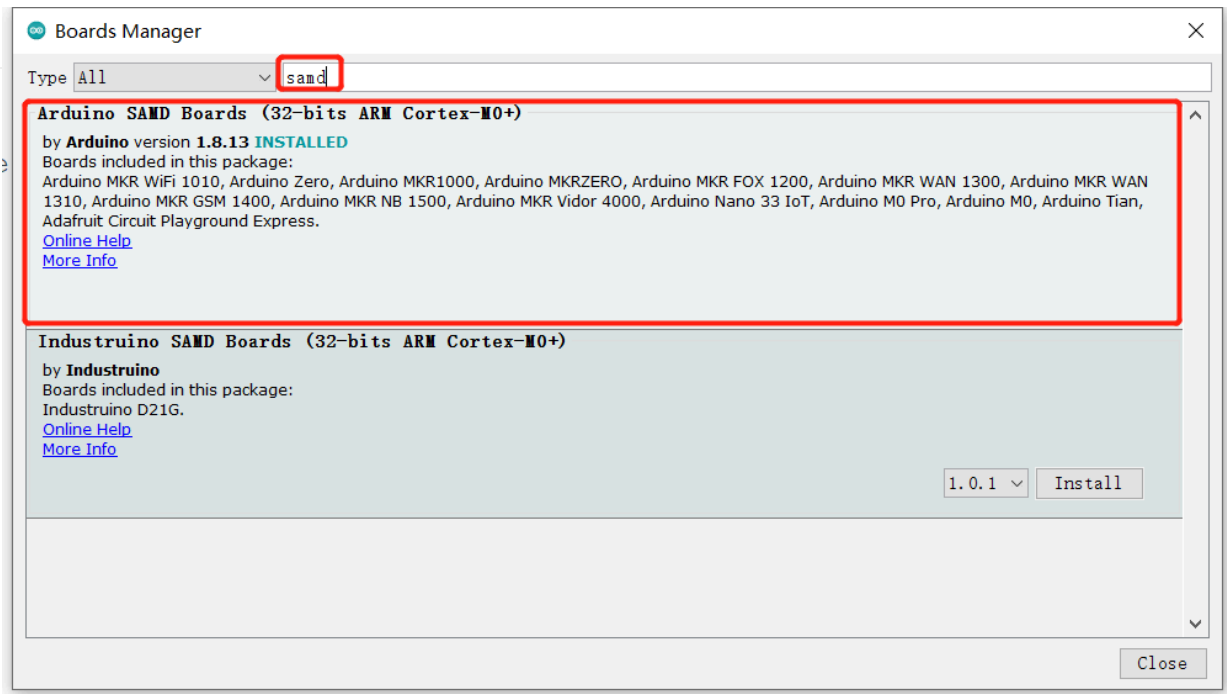
2 When using Mega2560, select

2.2 The development board is mkr wifi1010

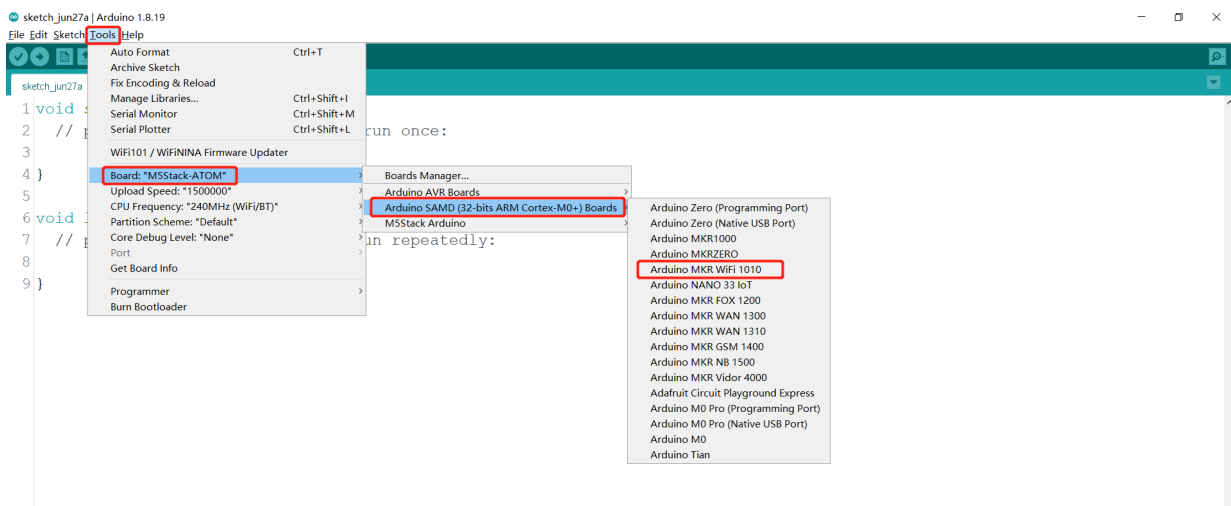
Search for samd in the development board manager. If it is not installed, install it. First, Tools --> Development Board --> Development Board Manager, then search for samd, see the following figure for details:



4.1 First-time self-check



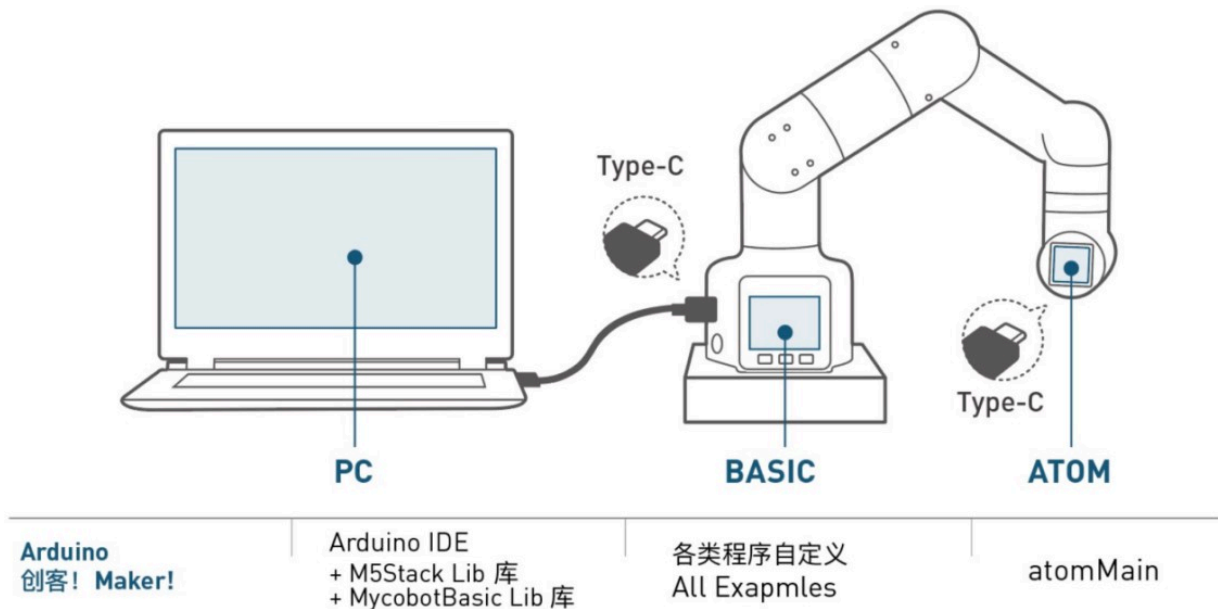
Development board selects mkr wifi1010, tools --> development board --> Arduino SAMD --> Arduino MKR WiFi1010



Simple use of Arduino

Connect the device

Take **myCobot 280-M5** as an example, use the Type-C data cable to connect the M5Stack-basic on the base of the robot arm to the PC.



Firmware requirements

- ATOM: Use **MyStudio** to burn the latest version of AtomMain
- Basic: No requirements

Check the link

Open the computer device manager to check if there is any device. If the device is not detected, please change the USB cable. If it is not available, please install and click to download **CP210X driver**. After downloading, unzip and install the required driver version to use it.

Open **Arduino IDE --> Tools --> Port** to check if there is a device. If the device is not detected, please change the USB cable to test, or check whether the driver is installed successfully.

Start development

Take burning an official demo as an example. Open **Arduino IDE --> File --> Examples --> MyCobotBasic** to see all project examples (if you don't see the example, you can restart Arduino). Choose to burn a simple demo, for example --> **MyCobot280--> MyCobot280-M5--> AnglesControl**.

4.1 First-time self-check

Open AnglesControl.ino from the example file



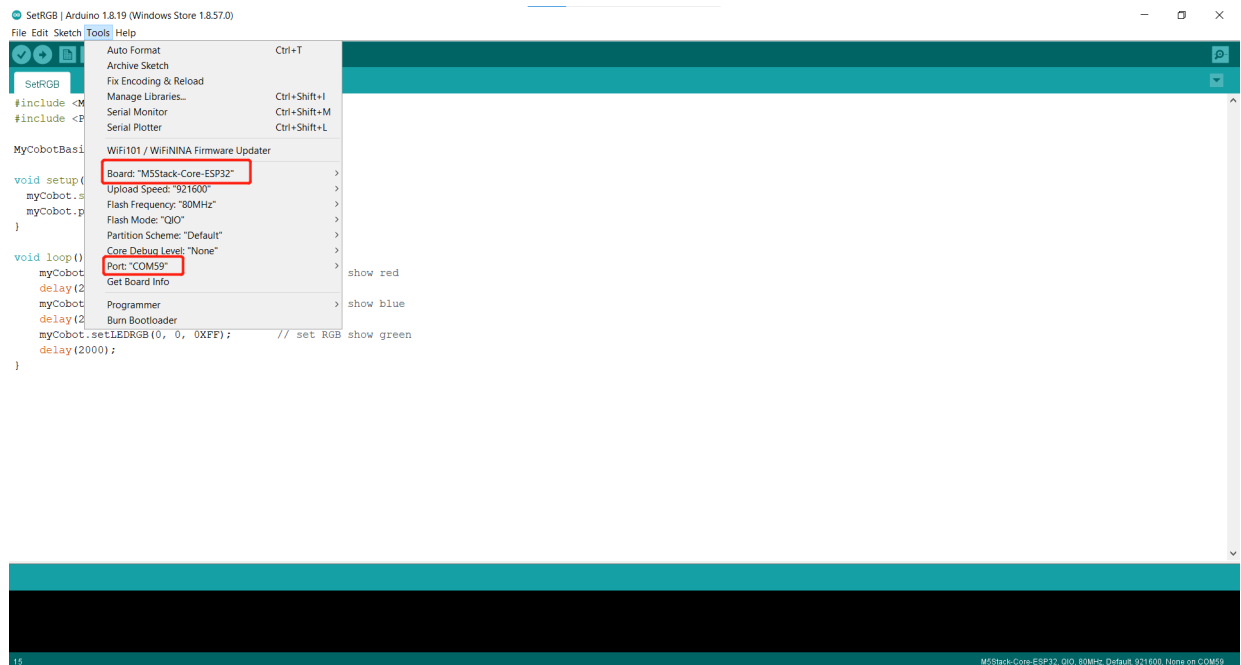
```
SetRGB | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
SetRGB
#include <MyCobotBasic.h>
#include <ParameterList.h>

MyCobotBasic myCobot;

void setup() {
  myCobot.setup();
  myCobot.powerOn();
}

void loop() {
  myCobot.setLEDRGB(0xFF, 0, 0); // set RGB show red
  delay(2000);
  myCobot.setLEDRGB(0, 0xFF, 0); // set RGB show blue
  delay(2000);
  myCobot.setLEDRGB(0, 0, 0xFF); // set RGB show green
  delay(2000);
}
```

Note: Select the development board as **M5Stack-Core-ESP32** and the corresponding **COM port**.



```
SetRGB | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WIFININA Firmware Updater
Board: "M5Stack-Core-ESP32"
Upload Speed: "921600"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Partition Scheme: "Default"
Core Debug Level: "None"
Port: "COM59"
Get Board Info show red
Programmer show blue
Burn Bootloader
myCobot.setLEDRGB(0, 0, 0xFF); // set RGB show green
delay(2000);
}
```

If you are using myCobot280-M5, please use **ParameterList.h** in the **MyCobot280-M5** folder to replace **ParameterList.h** in the **MyCobotBasic** folder. Please see the following figure for details:

Arduino > libraries > MyCobotBasic > examples > MyCobot280 > MyCobot280_M5 >

| 名称 | 修改日期 | 类型 |
|-----------------|------------------|------|
| AnglesControl | 29/07/2023 14:32 | 文件夹 |
| CoordsControl | 29/07/2023 14:32 | 文件夹 |
| GripperControl | 29/07/2023 14:32 | 文件夹 |
| MiniRobot | 29/07/2023 14:32 | 文件夹 |
| ParameterList.h | 29/07/2023 14:32 | H 文件 |


Arduino > libraries > MyCobotBasic >

| 名称 | 修改日期 | 类型 | 大小 |
|---------------------------------|------------------|-----------|-------|
| .git | 29/07/2023 16:40 | 文件夹 | |
| examples | 29/07/2023 16:11 | 文件夹 | |
| lib | 29/07/2023 14:40 | 文件夹 | |
| res | 29/07/2023 14:32 | 文件夹 | |
| CommunicateDefine.h | 29/07/2023 14:32 | H 文件 | 11 KB |
| mycobot.json | 29/07/2023 14:32 | JSON File | 9 KB |
| mycobot_24px.h | 29/07/2023 14:32 | H 文件 | 87 KB |
| MyCobotBasic.cpp | 29/07/2023 14:32 | CPP 文件 | 78 KB |
| MyCobotBasic.h | 29/07/2023 14:32 | H 文件 | 9 KB |
| MyCobotLanguage.cpp | 29/07/2023 14:32 | CPP 文件 | 3 KB |
| MyCobotLanguage.h | 29/07/2023 14:32 | H 文件 | 1 KB |
| MyCobotSaver.cpp | 29/07/2023 14:32 | CPP 文件 | 5 KB |
| MyCobotSaver.h | 29/07/2023 14:32 | H 文件 | 2 KB |
| MyPalletizerBasic.cpp | 29/07/2023 14:32 | CPP 文件 | 57 KB |
| MyPalletizerBasic.h | 29/07/2023 14:32 | H 文件 | 5 KB |
| MyPalletizerCommunicateDefine.h | 29/07/2023 14:32 | H 文件 | 8 KB |
| ParameterList.h | 29/07/2023 16:39 | H 文件 | 5 KB |
| README.md | 29/07/2023 14:32 | MD 文件 | 3 KB |

Note: When using different models, please use the "ParameterList.h" file in the respective case directory to replace the "MyCobotBasic\ParameterList.h" file

Click upload and wait for the progress bar in the lower right corner to finish running

4.1 First-time self-check

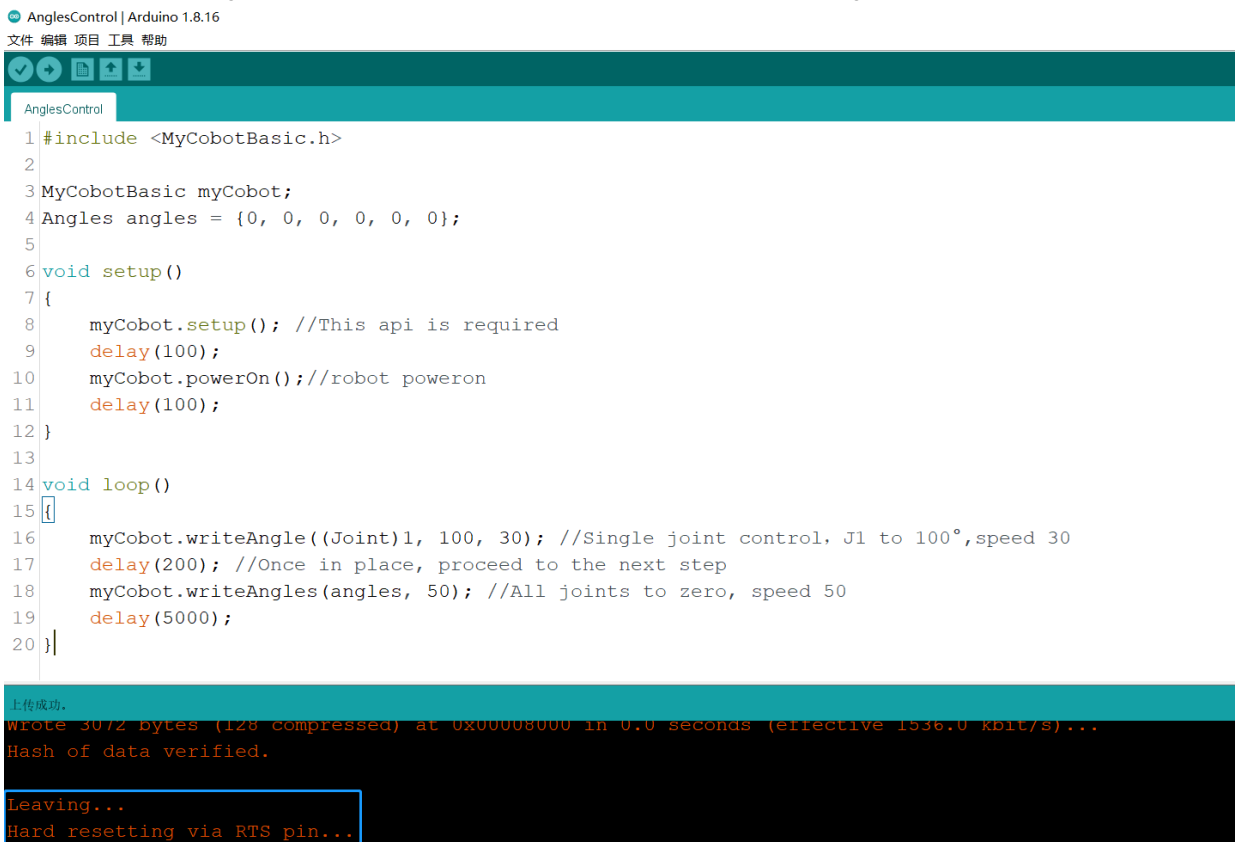


AnglesControl | Arduino 1.8.16
文件 编辑 项目 工具 帮助

```
1 #include <MyCobotBasic.h>
2
3 MyCobotBasic myCobot;
4 Angles angles = {0, 0, 0, 0, 0, 0};
5
6 void setup()
7 {
8     myCobot.setup(); //This api is required
9     delay(100);
10    myCobot.powerOn();//robot poweron
11    delay(100);
12 }
13
14 void loop()
15 {
16    myCobot.writeAngle((Joint)1, 100, 30); //Single joint control, J1 to 100°, speed 30
17    delay(200); //Once in place, proceed to the next step
18    myCobot.writeAngles(angles, 50); //All joints to zero, speed 50
19    delay(5000);
20 }
```

正在编译项目...

Wait until the lower right corner shows that the upload is successful, and the program has been downloaded



AnglesControl | Arduino 1.8.16
文件 编辑 项目 工具 帮助

```
1 #include <MyCobotBasic.h>
2
3 MyCobotBasic myCobot;
4 Angles angles = {0, 0, 0, 0, 0, 0};
5
6 void setup()
7 {
8     myCobot.setup(); //This api is required
9     delay(100);
10    myCobot.powerOn();//robot poweron
11    delay(100);
12 }
13
14 void loop()
15 {
16    myCobot.writeAngle((Joint)1, 100, 30); //Single joint control, J1 to 100°, speed 30
17    delay(200); //Once in place, proceed to the next step
18    myCobot.writeAngles(angles, 50); //All joints to zero, speed 50
19    delay(5000);
20 }
```

上传成功。
wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1536.0 kbit/s)...
Hash of data verified.
Leaving...
Hard resetting via RTS pin...

At this time, we can see the **robot** start working.

For the interface and driver of **basic buttons and screen**, please refer to the following documents:

Button: <https://docs.m5stack.com/en/api/core/button>

Screen: <https://docs.m5stack.com/en/api/core/lcd>

Introduction to some cases

Currently, different models have angle, coordinate, and gripper control. **Transponder:** MyCobot280-Arduino uno and Mega2560 development board use case, the main function is communication, on this basis, use RoboFlow, python, myblockly, etc. to control the robot arm and perform io control.

AnglesControl: This demo is used to control the angle control of the robot arm about its joints.

CoordsControl: This demo is used to control the angle control and coordinate control of the robot arm about its joints.

GripperControl: This demo is used to control the opening and closing action and angle of the gripper of a robot arm called MyCobot.

Note: Arduino environment configuration and case compilation can be seen in our gitbook document or the video on our Bilibili (<https://space.bilibili.com/2126215657>).

arduino API

Overall status of the robot

1.1 `powerOn();`

- Function: Power on the robot, and the robot can be controlled only after powering on
- Return value: None

1.2 `powerOff();`

- Function: Power off the robot
- Return value: None

1.3 `isPoweredOn();`

- Function: Atom status query, return atom connection status
- Return value: TRUE for open, FALSE for close

1.4 `getAtomVersion();`

- Function: Get the Atom firmware version
- Return value: int type value, data needs /10, for example, the version number read is 12, the actual number needs to be divided by 10, and the final version number is 1.2

1.5 `setFreeMoveMode(bool mode);`

- Function: Set the free movement mode. After the free movement mode is turned on, the end LED is yellow. Long press the end atom to manually move the robot

Parameter description:

- mode: mode, 0/1, 0--turn off free movement, 1--turn on free movement
- Return value: None

Input program control mode MDI Mode and Robot Control (Manual Data Input)

2.1 `getAngles();`

- Function: Read all joint angles. When using, you should define an Angles angles to receive the read angles. Angles is a built-in variable or function definition of the library function. You can define a storage space angles with a memory of 6 to store angle variables. The method of use is the same as that of an array.
- Return value: Angles type array

2.2 `writeAngle(int joint, float value, int speed);`

- Function: Send single joint angle
- Parameter description: Joint number = joint, value range 1-6; Specified angle value = value, value range about -170°- +170°; Specified speed = speed, value range 1-100;
- Return value: None

4.1 First-time self-check

2.3 `writeAngles(Angles angles, int speed);`

- Function: Joint angles are executed synchronously, and the angles of six joints are sent to the actuator at the same time. Angles is the definition type declared by the library function. It specifies that angles is a container with a capacity of 6 data, which can be understood as an array. When assigning values, you can use for loop assignment or assign values individually.
- Parameter description: Angles[0] = specific angle, Angles[2] = specific angle, and so on. The value range is 0-90 (the range is defined, and the value range should be the same as writeAngle) Unit: degree Symbol: °
Movement speed = speed, value range 1-100 Unit %
- Return value: None

2.4 `getCoords();`

- Function: Read the x, y, z, rx, ry, rz of the current robot end. When using, a Coords tempcoords should be defined to receive the read angle. Coords is a built-in variable or function definition of the library function. You can define a storage space tempcoords with a memory of 6 to store angle variables. The method of use is the same as that of an array.
- Return value: an array of Coords type, you need to define the variable of Coords type

2.5 `writeCoord(Axis axis, float value, int speed);`

- Function: send the specific value of the individual coordinate parameters X/ Y/ Z/ RX/ RY/ RZ, the end will move in a single direction,
- Parameter description: Moving path coordinate value = value value range -300-300 (axis=Axis::X, aixs=Axis::Y and axis=Axis::Z are position coordinates X, Y, Z respectively, unit mm, position coordinate value range is not uniform, axis=Axis::RX, aixs=Axis::RY and axis=Axis::RZ are RX, RY, RZ respectively, value range is -180°-180°, if it exceeds the value range, it will return the prompt of inverse kinematics no solution) Specified speed = speed value range 1-100 Unit %
- Return value: None

2.6 `writeCoords(Coords coords, int speed);`

- Function: Send the specified coordinate parameter. The parameter type should be Coords. You need to declare a variable of type Coords. The usage of this variable is the same as that of an array.
- Parameter description: coords[0] = X, coords[1] = Y, coords[2] = Z, X,Y,Z value range -300.00-300.00 (the value range is undefined. If it exceeds the range, the prompt "inverse kinematics no solution" will be returned) Unit mm coords[3] = RX, coords[4] = RY, coords[5] = RZ, RX,RY,RZ value range -180-180 Specified speed = speed value range 1-100 Unit %
- Return value: None

2.7 `checkRunning();`

- Function: Check if the device is moving
- Return value: Return TRUE if it is moving, otherwise return FALSE

2.8 `setEncoder(int joint, int encoder);`

- Function: Set a single joint to rotate to a specified potential value
- Parameter description: Joint number = joint value range 1-7 (No. 7 joint is generally a gripper, and the gripper potential value range is: 1325-2048); Servo potential value = encoder value range 0-4096 (this range should be positively correlated with the range of each joint)

4.1 First-time self-check

- Return value: None

2.9 `getEncoder(int joint);`

- Function: Get the specified joint potential value
- Parameter description: Servo number = joint value range 1-7
- Return value: int type, reference value range 0-4096

2.10 `setEncoders(Angles angleEncoders, int speed);`

- Function: Set the six joints of the robot arm to execute synchronously to the specified position
- Parameter description: It is necessary to define a variable `angleEncoders` of the `Angles` type. The use of `angleEncoders` is equivalent to an array. Assign values to the array `angleEncoders`, with a value range of 0-4096 (the range should be positively correlated with the range of each joint). The length range of the array is 6. Specify `speed = speed`, with a value range of 1-100 units %
- Return value: None

2.11 `getEncoders();`

- Function: Get all joint potential values
- Return value: `Angles` type array, reference value range 0-4096

2.12 `getServoSpeeds();`

- Function: Get all servo speeds
- Return value: `Angles` type array, the speed is 0 when not moving

JOG Mode

3.1 `jogAngle(int joint, int direction, int speed);`

- Function: Control a single joint of the device to move in one direction
- Parameter description: Joint servo number = joint Value range 1-6 Joint movement direction = Direction Value range -1/1 Specified speed = speed, value range 1-100 unit %
- Return value: None

3.2 `jogCoord(Axis axis, int direction, int speed);`

- Function: Control the device to move in one direction in Cartesian space
- Parameter description: Device direction selection = axis Value X,Y,Z,RX,RY,RZ Joint movement direction = Direction Value -1/1 Specified speed = speed, value range 1-100 unit %
- Return value: None

3.3 `jogStop();`

- Function: Stop the movement in the specified direction that has already started
- Return value: None

3.4 `ProgramPause();`

- Function: Pause the program

4.1 First-time self-check

- Return value: None

3.5 `ProgramResume();`

- Function: Resume the program
- Return value: None

3.6 `TaskStop();`

- Function: Stop the program
- Return value: None

Running auxiliary information Running Status and Settings

4.1 `getSpeed();`

- Function: Read the current running speed of the device
- Return value: int type, value range 1-100, unit %

4.2 `setSpeed(int percentage);`

- Function: Set the running speed of the device
- Parameter description: percentage value range 1-100, unit %

4.3 `getJointMin(int joint);`

- Function: Read the minimum limit angle of the joint
- Parameter description: Joint servo number = joint, value range 1-6
- Return value: float type value

4.4 `getJointMax(int joint);`

- Function: Read the maximum limit angle of the joint
- Parameter description: Joint servo number = joint, value range 1-6
- Return value: float type value

4.5 `setMovementType(MovementType movement_type);`

- Function: Set the movement mode
- Parameter description: The movement modes are nonlinear path movement (movej) and linear path movement (moveL)
- Return value: None

4.6 `getMovementType();`

- Function: Read the movement mode
- Return value: Nonlinear mode returns 0; linear mode returns 1

Joint motor settings Joint Servo Control

5.1 `isServoEnabled(int joint);`

- Function: Check the connection status of a single joint
- Parameter description: Joint servo number = joint, value range 1-6
- Return value: connection status, 0/1, 1--connected, 0--unconnected

5.2 `isAllServoEnabled();`

- Function: Detect the connection status of all joints
- Return value: connection status, 0/1, 1--connected, 0--unconnected
- `3 setServoData(int joint, byte data_id, byte data);`
- Function: Set servo system parameters
- Parameter description: Joint servo number = joint, value range 1-6 Data address = data_id, value range please refer to the address in the figure above Data = value range in the figure above
- Return value: None

• `4 setServoCalibration(int joint);`

- Function: Joint zero position calibration, corresponding potential value is 2048
- Parameter description: Joint servo number = joint, value range 1-6

• `5 releaseServo(byte servo_no);`

- Function: Relax/disable a joint of the robot
- Parameter description: servo_no is 1-6
- Return value: None

• `6 focusServo(byte servo_no);`

- Function: Enable a joint of the robot
- Parameter description: servo_no is 1-6
- Return value: None

• `7 getServoVoltages();`

- Function: Get all servo voltages
- Return value: Angles type array, reference value range 8.4-12.0

• `8 getServoStatus();`

- Function: Get all servo status
- Return value: 0 means everything is normal; 1 means voltage overvoltage/undervoltage; 2 means magnetic encoding status is abnormal; 4 means temperature overheating; 8 means current overcurrent; 32 means load overload; when the number is not equal to the above abnormal number, for example: 3 means voltage overvoltage/undervoltage and magnetic encoding status is abnormal, 7 means voltage overvoltage/undervoltage, magnetic encoding status is abnormal and temperature is overheating

• `9 getServoTemps();`

4.1 First-time self-check

- Function: Get all servo temperatures
-
- Return value: Angles type array, reference value range 0-255

Atom terminal IO control Atom IO Control

6.1 `setPinMode(byte pin_no, byte pin_mode);`

- Function: Set the state mode of the atom specified pin
- Parameter description: Pin number = `pin_no` Value range: 19, 22, 23, 26, 32, 33 Output mode = `pin_mode` Value range: 0, 1
- Return value: None

6.2 `setLEDRGB(byte r, byte g, byte b);`

- Function: Set the color of the RGB light of the atom screen:
- Parameter description: Parameter value corresponding to red light = r, value range 0x00-0xFF; Parameter value corresponding to green light = g, value range 0x00-0xFF; Parameter value corresponding to blue light = b, value range 0x00-0xFF;
- Return value: None

6.3 `setGripperState(byte mode, int sp);`

- Function: Set the gripper opening and closing state
- Parameter description:
- mode, gripper opening and closing mode, range 0/1, 0--gripper opens to the maximum, 1--gripper closes to the minimum
- sp, gripper opening and closing speed, range 1-100

6.4 `setGripperValue(int data, int sp);`

- Function: Set the gripper opening and closing angle
- Parameter description:
- data, gripper opening and closing angle, range 0-100, 0--close to the minimum angle, 100--open to the maximum angle
- sp, gripper opening and closing speed, range 1-100
- Return value: None

6.5 `setGripperIni();`

- Function: Set the gripper zero point
- Return value: None

6.6 `getGripperValue();`

- Function: Get the current gripper angle
- Return value: Return the current gripper angle, range 0-100

6.7 `isGripperMoving();`

4.1 First-time self-check

- Function: Detect whether the gripper is moving

-
- Return value: 0 not moving, 1 moving

6.8 `void setElectricGripper(bool mode);`

- Note: This interface is only available for MyCobot320 robot
- Function: Control the opening and closing of the electric gripper
- Parameter description:
- mode: mode, 0/1, 0--the gripper opens to the maximum, 1--the gripper closes to the minimum
- Return value: None

6.9 `void InitElectricGripper();`

- Note: This interface is only available for MyCobot320 robot
- Function: Initialize the opening and closing of the electric gripper. Each time the electric gripper is plugged in, it needs to be initialized before it can be controlled. After successful initialization, the gripper will open and close once
- Return value: None

6.10 `void setGripperMode(bool mode);`

- Note: This interface is only available for MyCobot320 robot
- Function: Set the adaptive gripper control mode
- Parameter description:
- mode: mode, 0/1, 0--485 communication control, 1--io control (in io mode, it can only be turned on or off, and the angle cannot be set. When turning on or off, pins 23 and 33 need to be set to different states, one high and one low)
- Return value: None

6.11 `bool getGripperMode();`

- Note: This interface is only available for MyCobot320 robots
- Function: Set the adaptive gripper control mode
- Return value: Adaptive gripper control mode, 0/1, 0--485 communication control, 1--io control

6.12 `setDigitalOutput(byte pin_no, byte pin_state);`

- Function: Set the working state of the IO pin
- Parameter description: 0 input; 1 output; 2 `pull_up_input`
- Return value: None

6.13 `getDigitalInput(byte pin_no);`

- Function: Read input
- Parameter description: Pin number = pin_no Value range: 19, 22, 23, 26, 32, 33
- Return value: None

6.14 `setPWMOutput(byte pin_no, int freq, byte pin_write);`

4.1 First-time self-check

- Function: Set the ATOM terminal IO to output a PWM signal with a specified duty cycle
-
- Parameter description:
 - pin_no: IO number
 - freq: clock frequency
 - pin_write: Duty cycle 0-256; 128 means 50%
 - Return value: None

Coordinate control mode

7.1 `setToolReference(Coords coords);`

- Function: Set tool coordinate system
- Parameter description: X, Y, Z value range -300.00-300.00 (the value range is undefined, and it will be returned if it exceeds the range inverse kinematics no solution prompt) Unit mm RX,RY,RZ value range -180.00-180.00
- Return value: None

7.2 `setWorldReference(Coords coords);`

- Function: Set the world coordinate system
- Parameter description: X,Y,Z value range -300.00-300.00 (the value range is not defined, and the inverse kinematics no solution prompt will be returned if it exceeds the range) Unit mm RX,RY,RZ value range -180.00-180.00
- Return value: None

7.3 `getToolReference();`

- Function: Get the tool coordinate system
- Return value: X,Y,Z value range -300.00-300.00 (the value range is not defined, and the inverse kinematics no solution prompt will be returned if it exceeds the range) Tip) Unit mm RX,RY,RZ Value range -180.00-180.00

7.4 `getWorldReference();`

- Function: Get the world coordinate system
- Return value: X,Y,Z Value range -300.00-300.00 (the value range is undefined, and the inverse kinematics no solution prompt will be returned if it exceeds the range) Unit mm RX,RY,RZ Value range -180.00-180.00

7.5 `setReferenceFrame(RFType rftype);`

- Function: Set the flange coordinate system
- Parameter description:

RFType::BASE is to use the robot base as the base coordinate, and RFType::WORLD is to use the world coordinate system as the base coordinate.

- Return value: None

7.6 `getReferenceFrame();`

- Function: Get flange coordinate system
- Return value: X,Y,Z range -300.00-300.00 (the range is undefined, and the inverse kinematics no solution prompt will be returned if it exceeds the range) Unit mm RX,RY,RZ range -180.00-180.00

7.7 `setEndType(EndType end_type);`

- Function: Set the end coordinate system
- Parameter description: EndType::FLANGE is to set the end to flange, EndType::TOOL is to set the end to tool end.
- Return value: None

7.8 `getEndType();`

- Function: Get the end coordinate system
- Return value: X,Y,Z range -300.00-300.00 (the range is undefined, and the inverse kinematics no solution prompt will be returned if it exceeds the range) Unit mm RX,RY,RZ range -180.00-180.00

What is Python?

Our products are very friendly to Python, and the development of Python API library is also improving day by day. Through Python, the robot's joint angles, coordinates, grippers and other aspects can be controlled. There are many options. If you want to control our robot arm through Python programming, you can learn this chapter.



Python was designed by Guido van Rossum of the Netherlands Institute for Mathematical and Computer Science Research in the early 1990s as a replacement for a language called ABC.

Python provides efficient high-level data structures and simple and effective object-oriented programming.

Python syntax and dynamic typing, as well as the nature of interpreted languages, make it a programming language for writing scripts and rapidly developing applications on most platforms. With the continuous update of versions and the addition of new language functions, it is gradually used for the development of independent and large projects.

Python interpreter is easy to extend, and can be extended with new functions and data types using C or C++ (or other languages that can be called from C).

Python can also be used as an extension language in customizable software. **Python** has a rich standard library that provides source code or machine code for all major system platforms.

Python development and use guide

You can use Python to develop our robot arm according to the following guidelines

1. [Environment construction](#)
2. [API description](#)
3. [Joint control](#)
4. [Coordinate control](#)
5. [IO control](#)
6. [Gripper control](#)
7. [Handle control](#)
8. [Drawing patterns](#)
9. [Demonstration code](#)

Environment setup

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

Before using pymycobot to control the robot arm, you need to build a Python environment. The following is a detailed description of Python download and installation.

Download and install Python

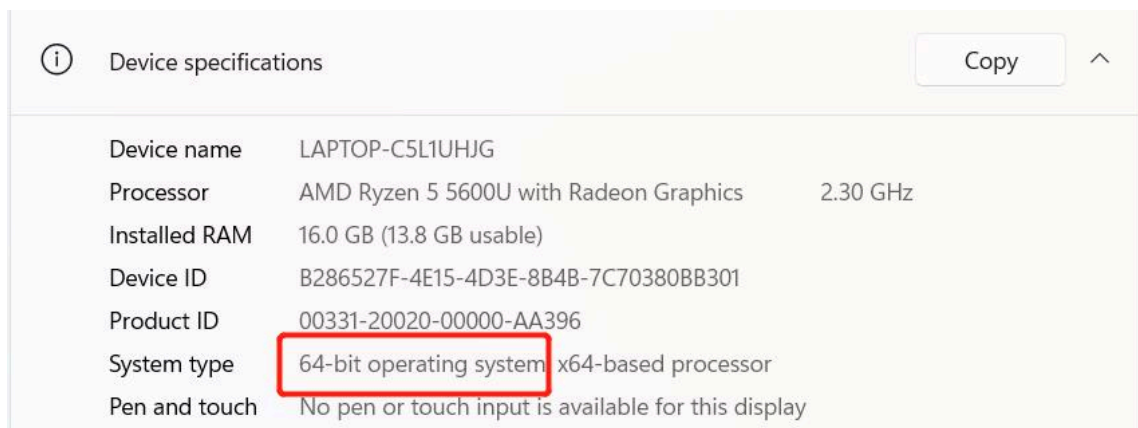
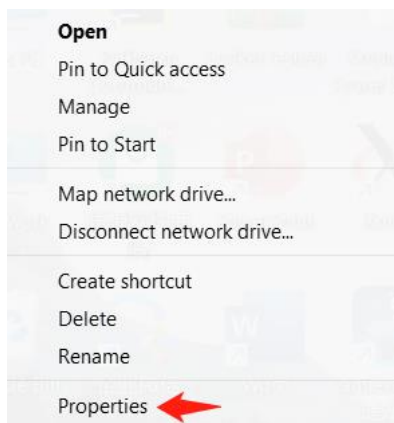
Applicable devices:

- myCobot 280:
- myCobot 280 Arduino
- myCobot 280 PI
- myCobot 280 Jetson Nano
- myCobot 280 Arduino

Currently, there are two versions of Python, one is 2.x version and the other is 3.x version. These two versions are incompatible. As 3.x version is becoming more and more popular, our tutorial will take the latest 3.10.7 version as an example.

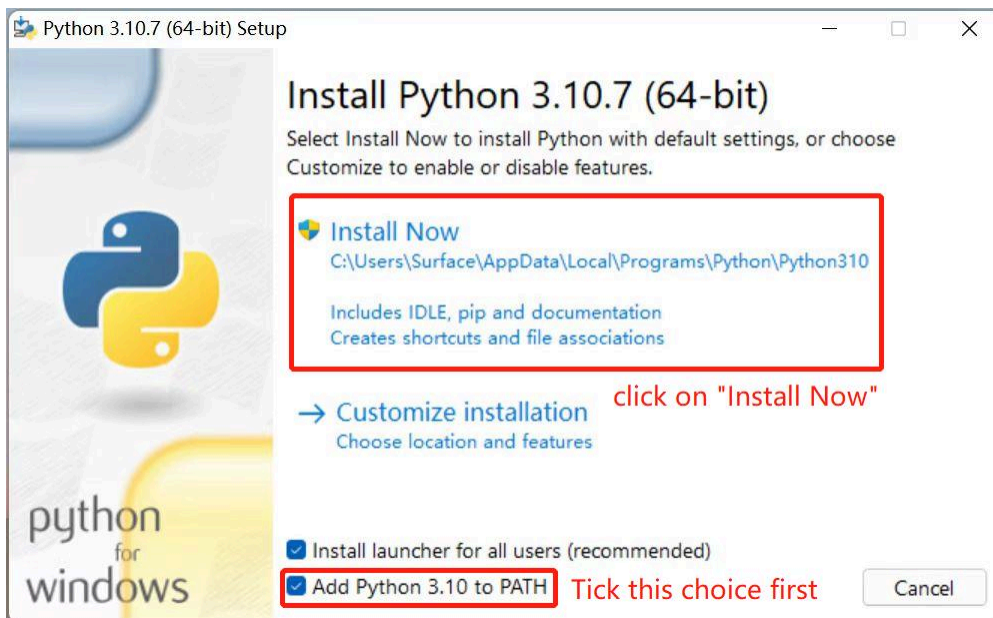
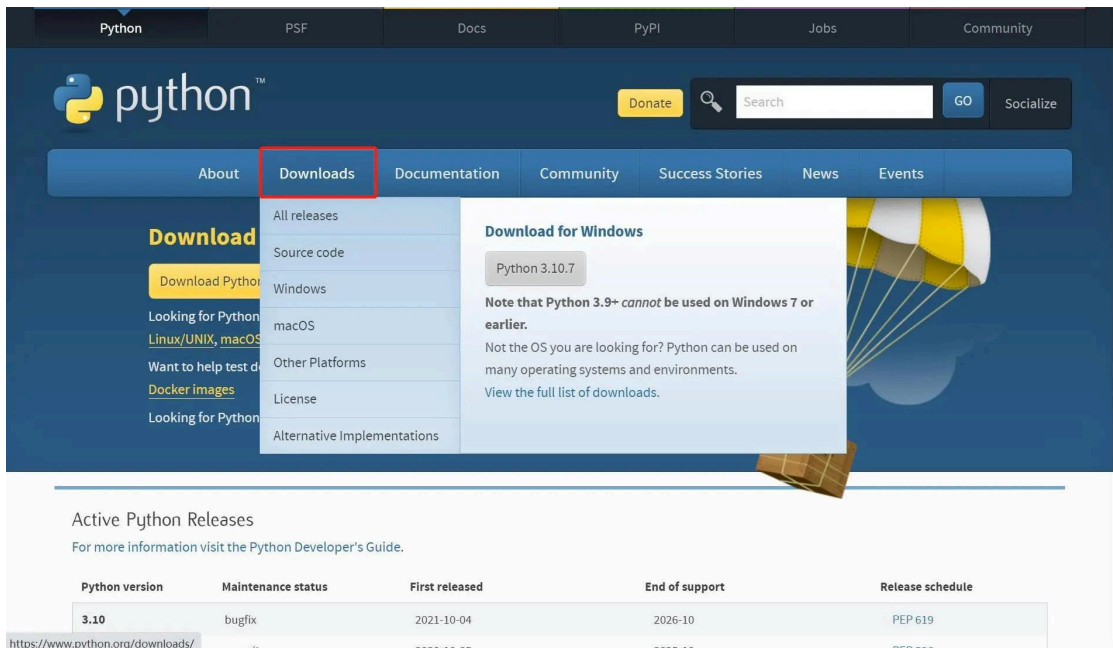
Install Python

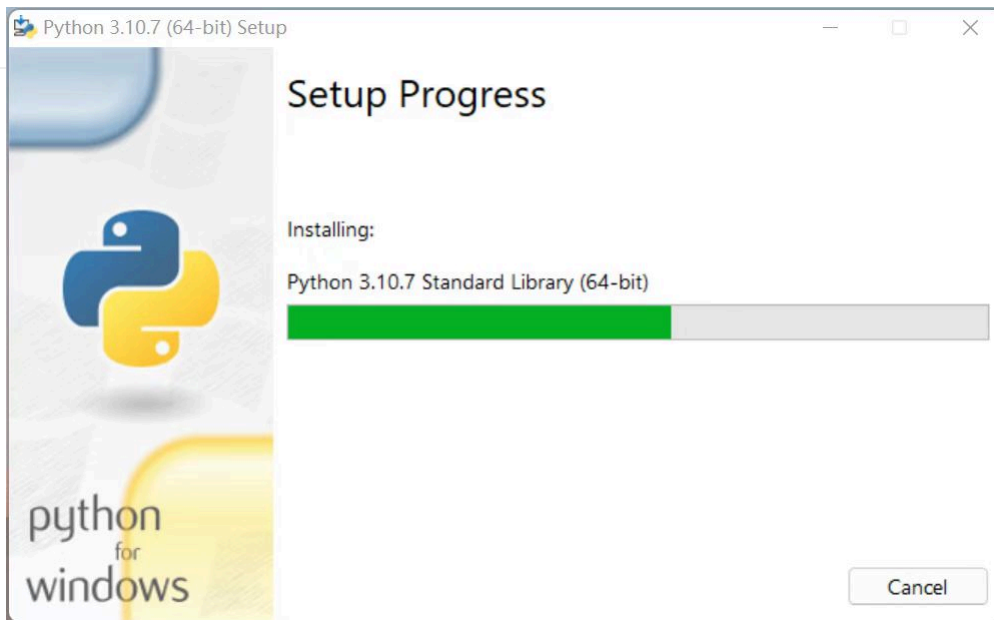
Note: Before installing, please confirm whether your computer is 64-bit or 32-bit. Right-click `My Computer` and select `Properties`. As shown in the figure below, it is a 64-bit operating system, so select the 64-bit Python installation package.



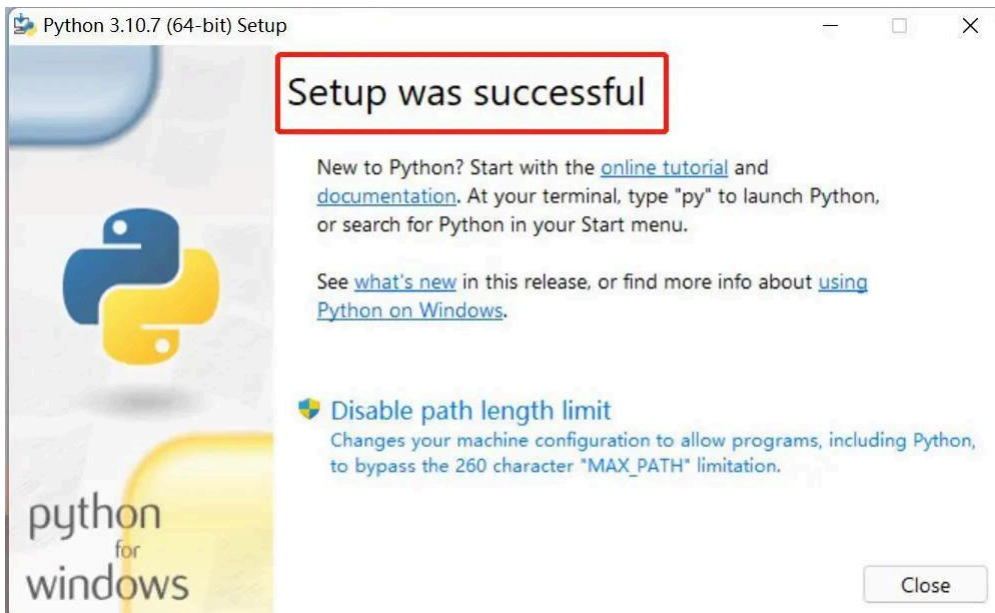
4.1 First-time self-check

- Python official download address: <https://www.python.org/downloads/>
- Click the **Downloads** option to start downloading Python, click **Add Python 3.10 to PATH**, click **Install Now** to start installing Python





- The prompt "Setup was successful" appears, indicating that the installation is complete

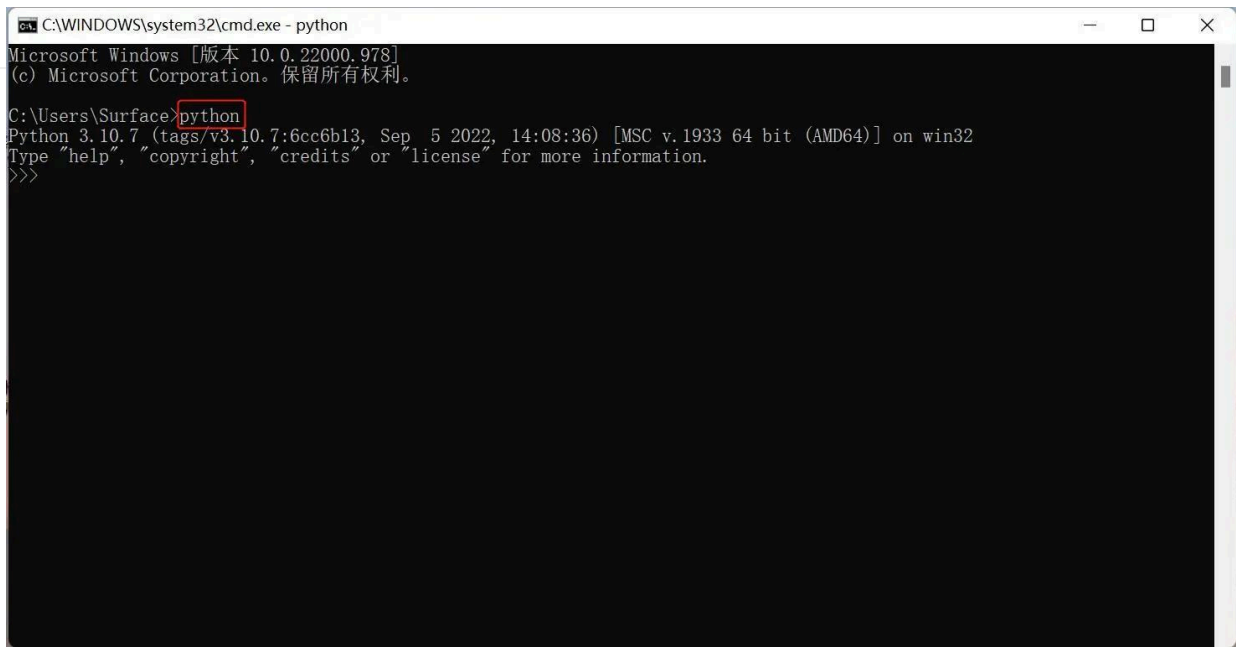


Run Python

After successful installation, open the command prompt window (Win+R, enter cmd and press Enter), and type `python`. Two situations will occur.

Situation 1:

4.1 First-time self-check



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

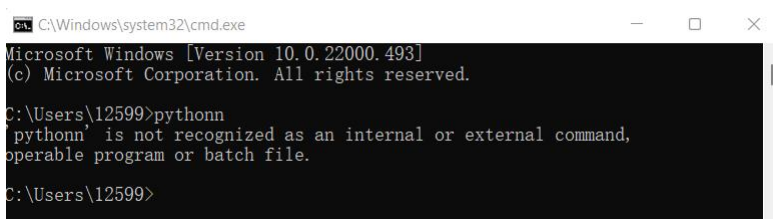
C:\Users\Surface>python
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep  5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

The prompt in the picture indicates that Python has been successfully installed.

The prompt `>>>` indicates that we are already in the Python interactive environment. We can enter any Python code and get the execution result immediately after pressing Enter.

Case 2:

If the input is wrong (for example, enter `pythonn`), an error message will appear:



```
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>
```

Note: The error message is generally caused by not configuring the environment variables. You can refer to [1.3 Configure environment variables](#) to modify the environment variables.

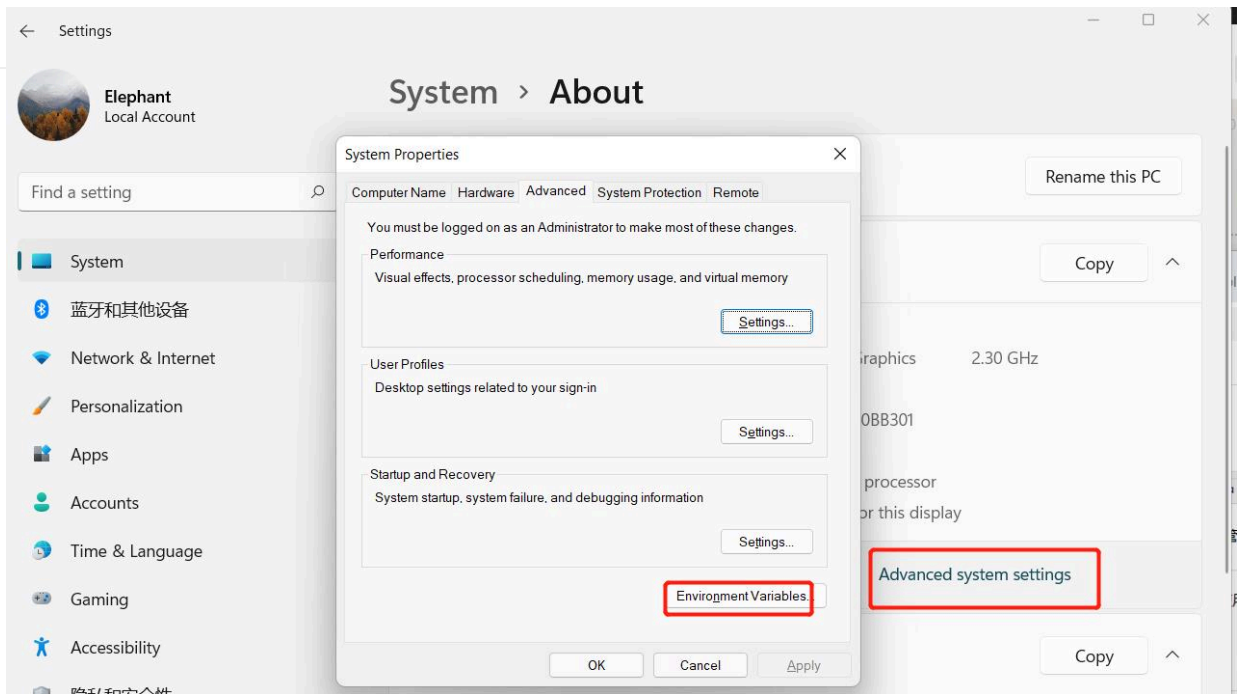
Configure environment variables

Since Windows will search for `python.exe` according to the path set by a Path environment variable, if it is not found, an error will be reported. Therefore, if you miss checking `Add Python 3.10 to PATH` during installation, you need to manually add the path where `python.exe` is located to Path, or reinstall Python and remember to check the `Add Python 3.10 to PATH` option.

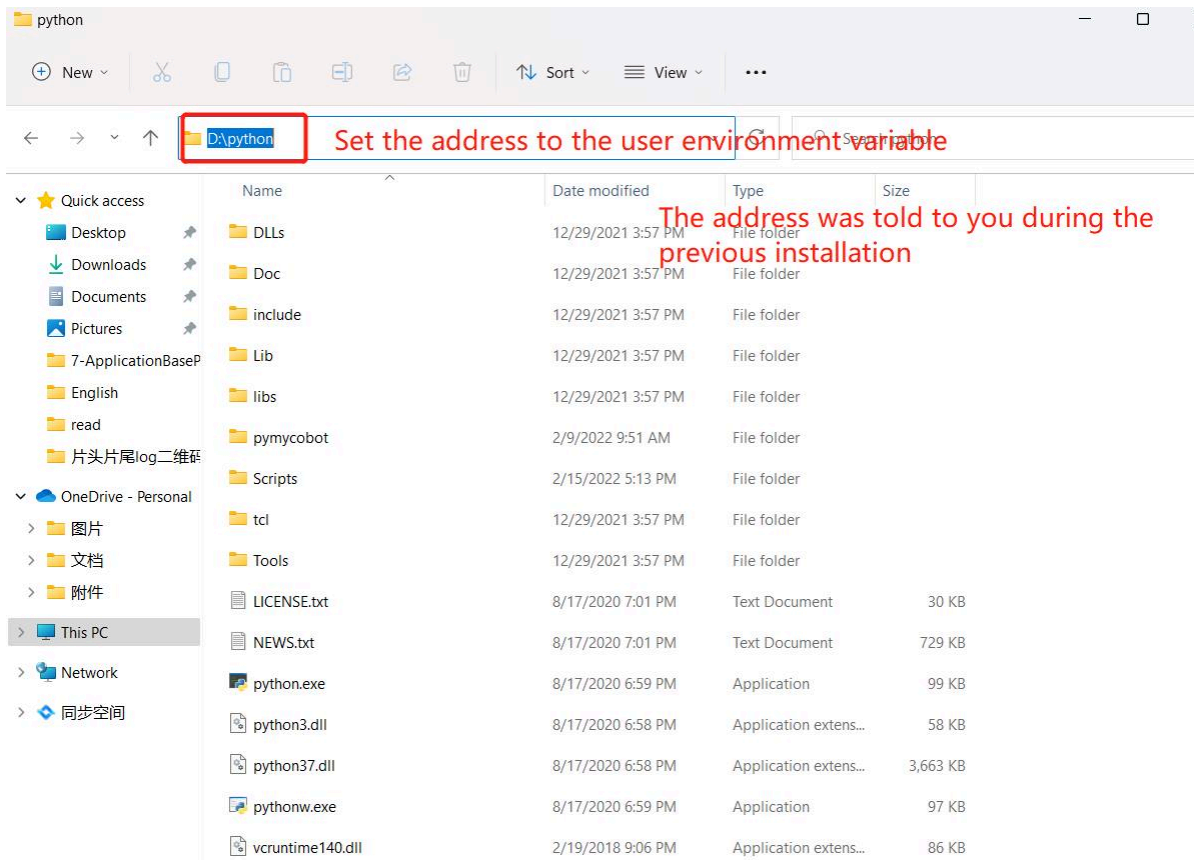
The following are the steps to manually add the path where `python.exe` is located.

- Right-click My Computer → Select Properties → Select Advanced System Settings → Select Environment Variables in the lower right corner:

4.1 First-time self-check

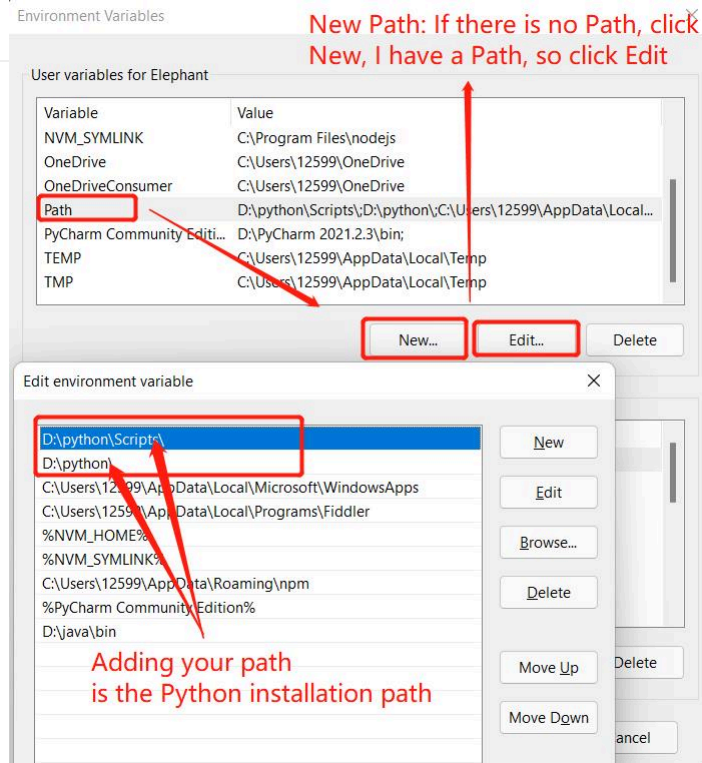


- Environment variables mainly include user variables and system variables. The environment variables that need to be set are in these two variables. As shown in the figure below:



- User variables are used to download programs that can be used in cmd commands. Write the absolute path of the program to the user variable and you can use it, as shown in the figure below:

4.1 First-time self-check



- After completing the above steps, open the command prompt window (Win+R, then enter cmd, press Enter), type Python, and the prompt in the figure below indicates success:

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

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

PyCharm installation and use

PyCharm is a powerful Python editor with cross-platform capabilities. First, let's introduce the installation steps of PyCharm in Windows system.

Download address: <https://www.jetbrains.com/pycharm/download/#section=windows>

Download and install

- After entering the website, we will see the following interface:



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

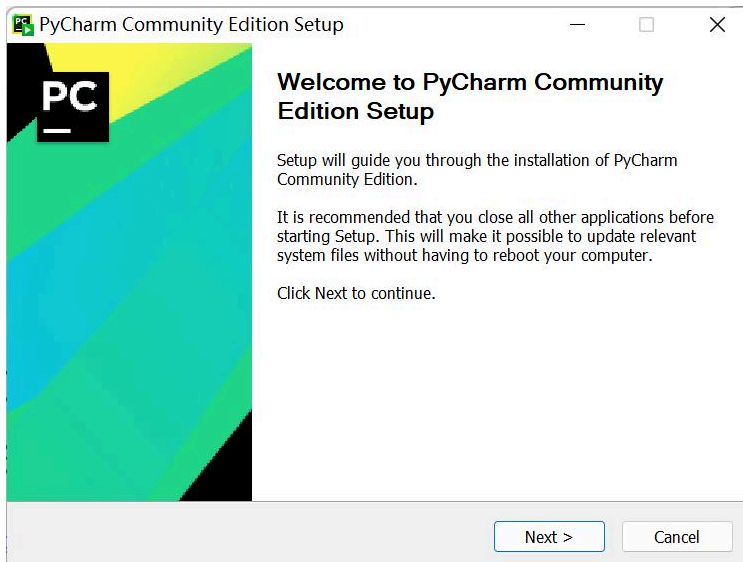
Free, built on open-source



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

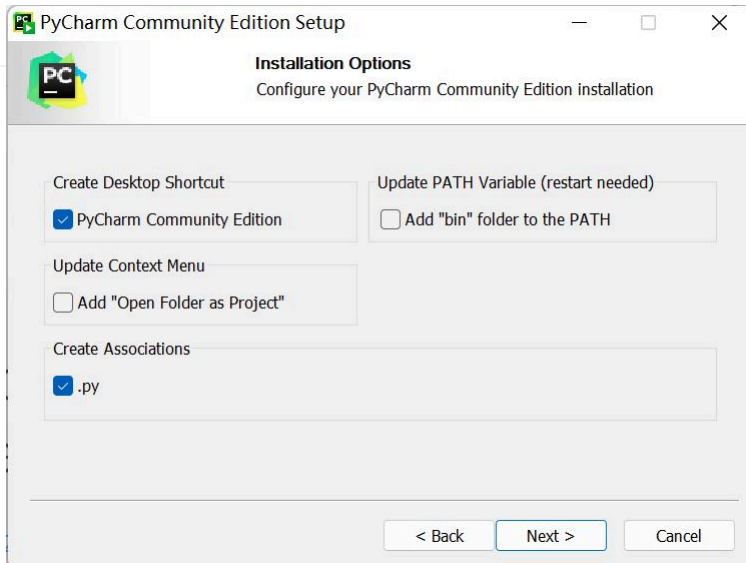
Download the file according to the interface introduction. Professional means professional version, and Community means community version. It is recommended to install the community version because it is free to use.

- After downloading, start installing and click **Next** :

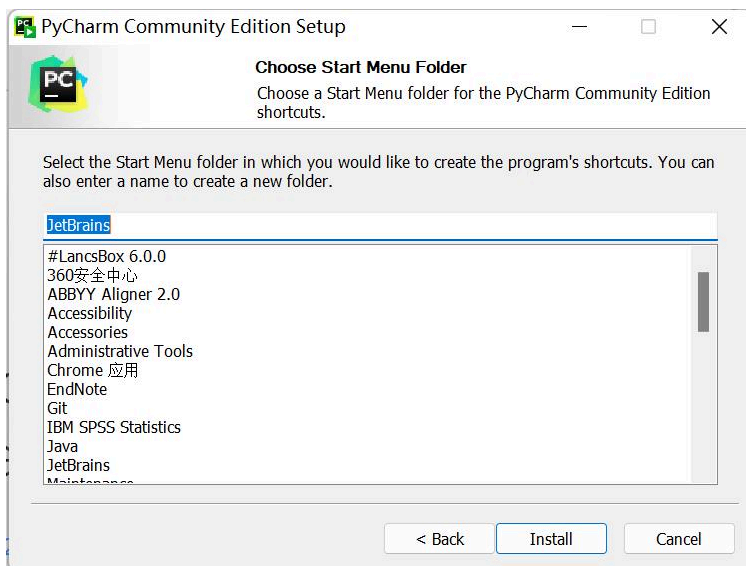


- Select the corresponding options according to your personal preferences, and then click **Next** :

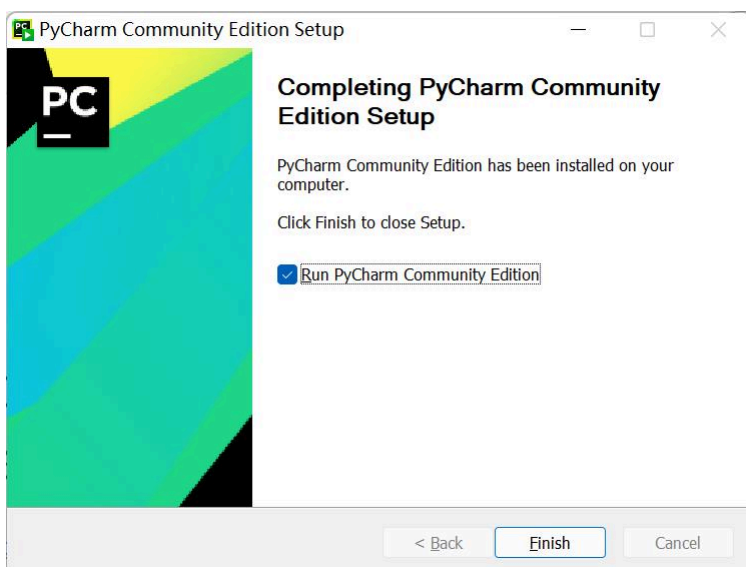
4.1 First-time self-check



- The following interface appears and continue to click **Next** :



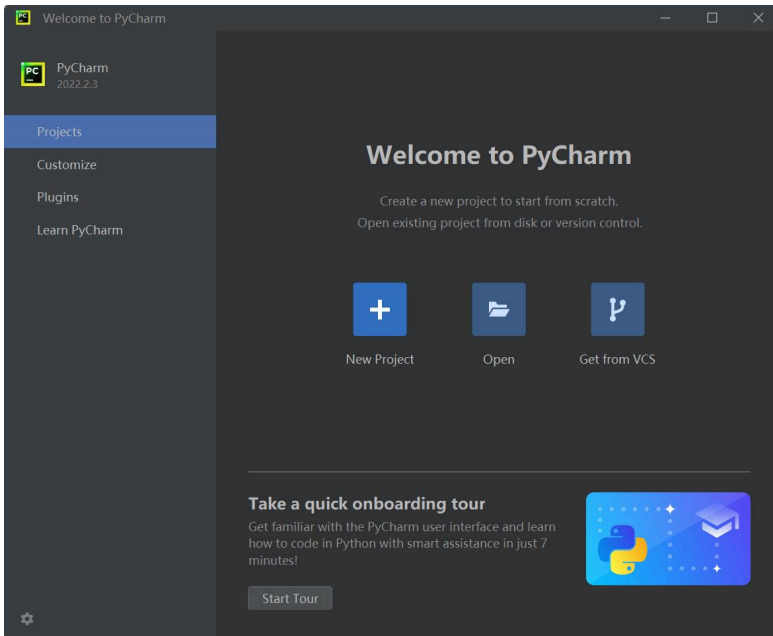
- Click **Finish** to complete the installation:



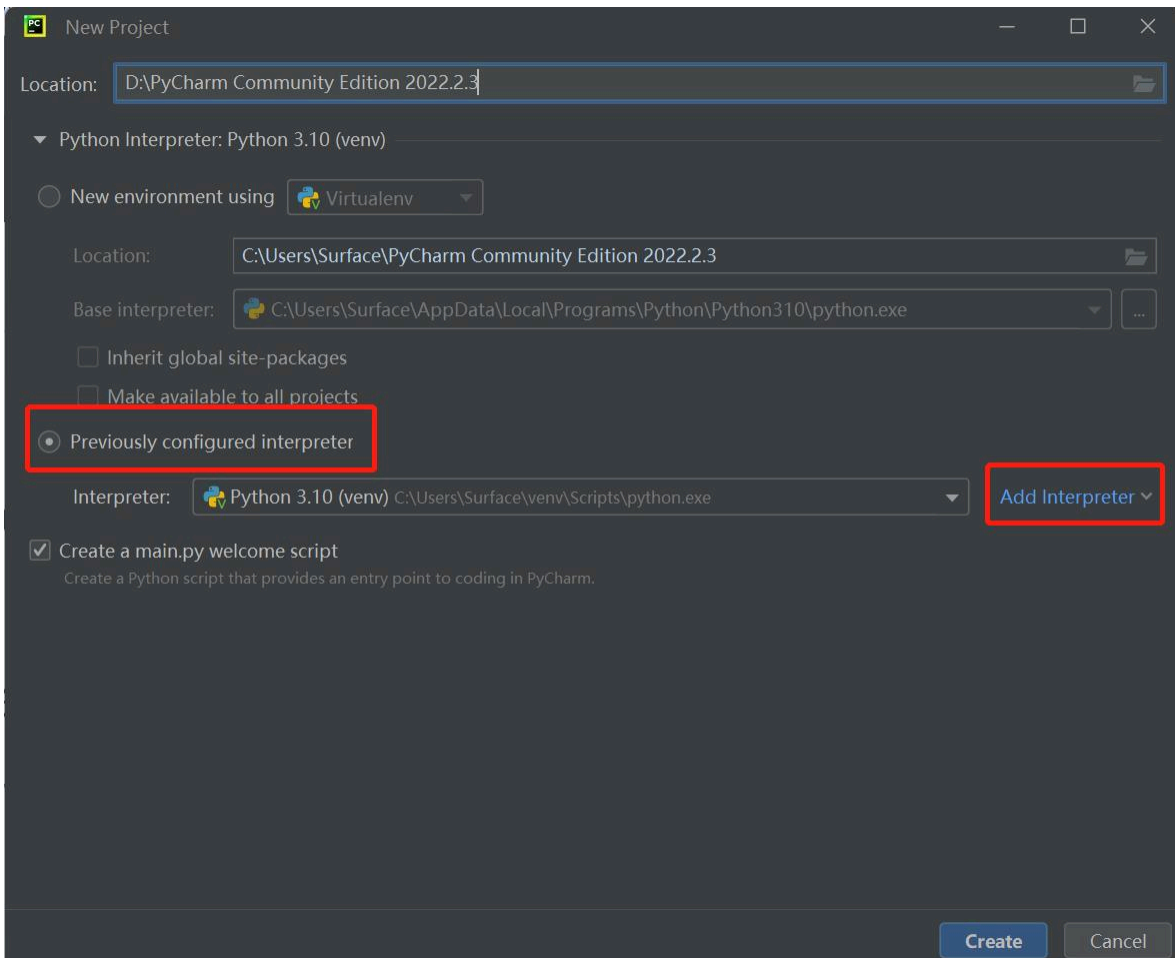
Create a project

After PyCharm is installed, enter the software and create the first program.

- Click the PyCharm icon on the desktop to enter PyCharm, as shown in the figure below, and click **New Project** :

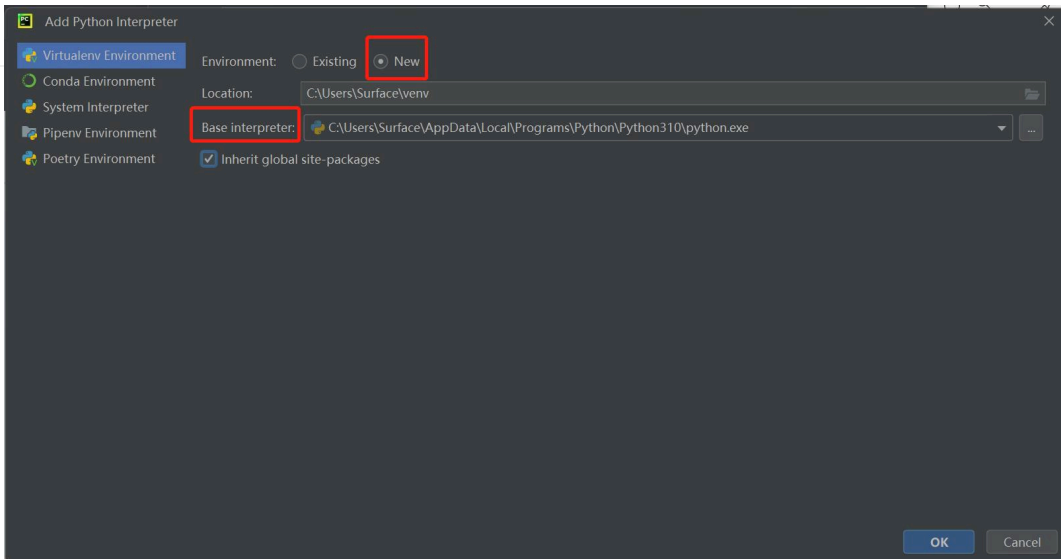


- After clicking, find **Interpreter** , start setting the interpreter, and click **Add Interpreter** :

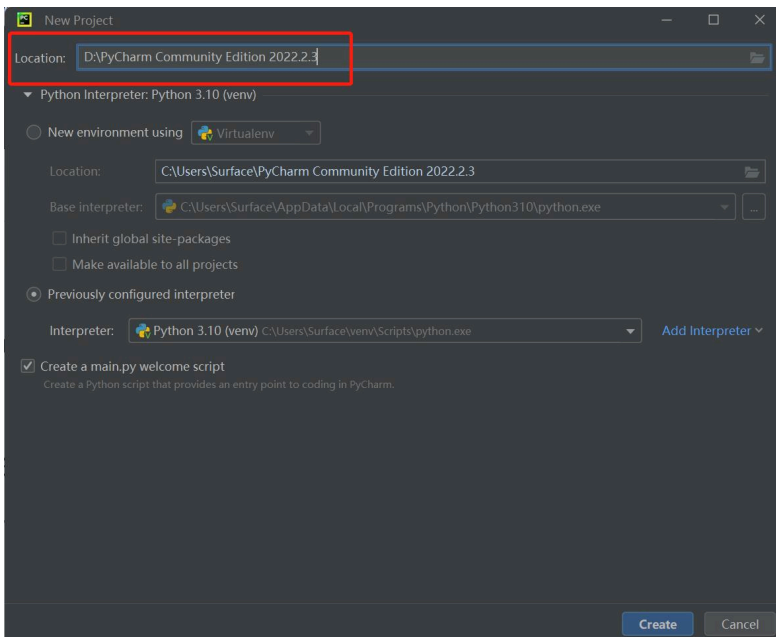


- Click **New** , find the python.exe storage location, and check the **Inherit global site-package** option:

4.1 First-time self-check

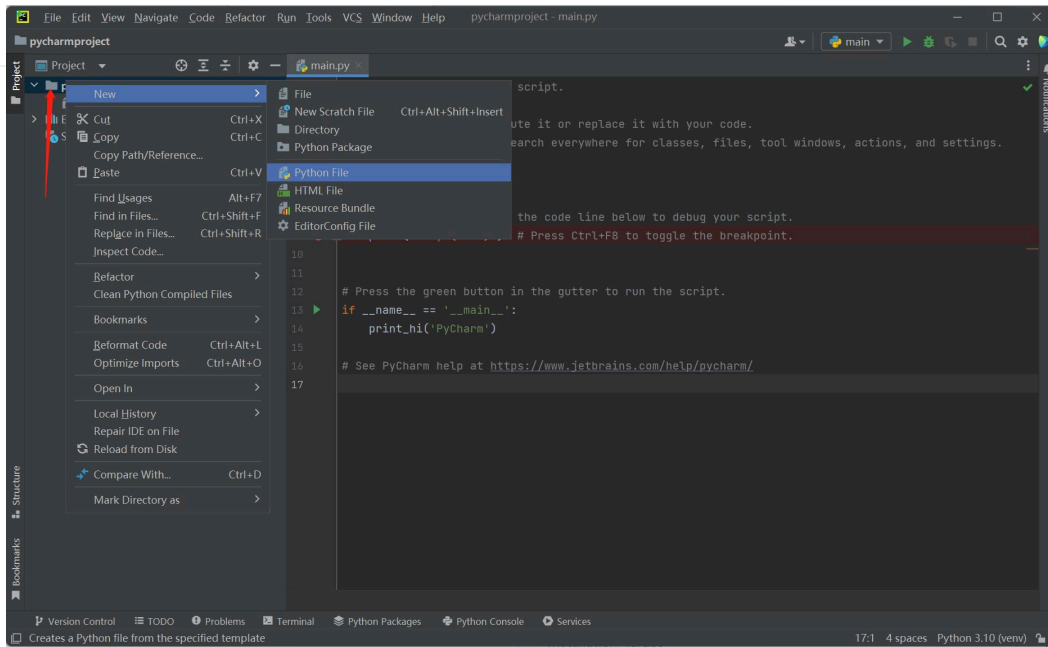


- Set `Location`. Location is where the PyCharm project is stored. You can choose it according to your needs.

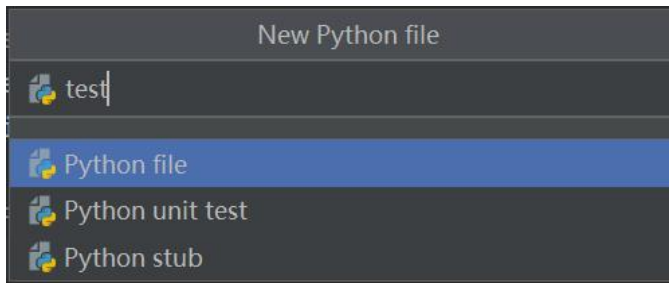


- Create a new PyCharm file. Right-click the document icon pointed by the arrow, click `New`, click `Python File`, and the new file is created successfully.

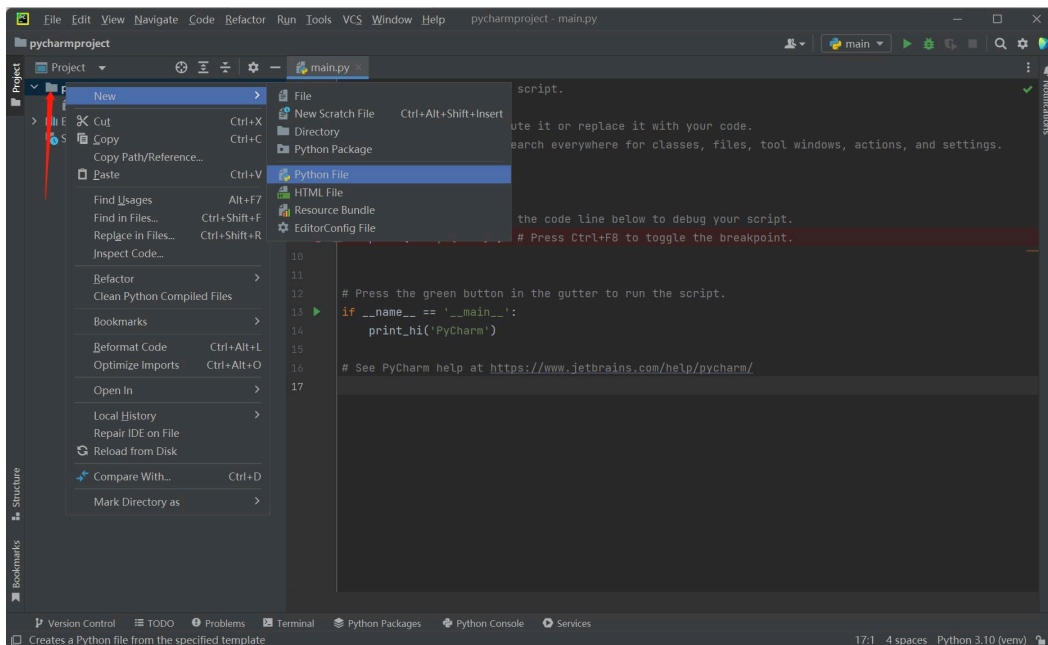
4.1 First-time self-check



- Name Python File:



- After the file is successfully created, you will enter the following interface and you can write your own program



Before use

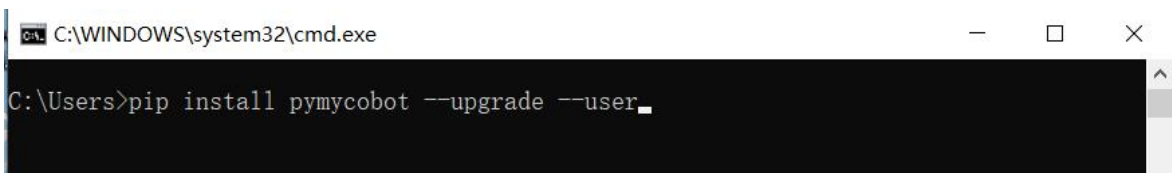
- Firmware burning. Firmware refers to the device "driver" stored inside the device. Only through firmware can the operating system implement the operation of a specific machine according to the standard device driver.

4.1 First-time self-check

Different versions of the robot arm need to burn different firmware (refer to the **MyStudio** chapter).

- **Arduino version** The Basic at the bottom needs to burn minirobot. After the burning is completed, select the **Transponder** function (this function is used to receive and forward the instructions sent by the Basic at the bottom to perform the target action), click `Press A`, and the **Atom: OK** prompt message appears, which means success. In addition, the latest version of atomMain is burned in the Atom at the end of the Arduino version. It is burned by default at the factory, and there is no need to burn it yourself.
- **Pi \ jetsonnano version** The latest version of atomMain is burned in the Atom at the end. It is burned by default at the factory, and there is no need to burn it yourself.
- **pymycobot installation.** Open a console terminal (shortcut Win+R, enter cmd to enter the terminal), and enter the following command:

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



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

```
git clone https://github.com/elephantrobotics/pymycobot.git <your-path>
#Where <your-path> fills in your installation address, if not filled in, the current path is used by default

cd <your-path>/pymycobot
#Enter the pymycobot folder of the download package

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

Simple use of Python

After the above preparations are completed, start to control the robot arm through Python code. Here, the MyCobot 280 Arduino version is used as an example for demonstration.

First, open the PyCharm you installed, create a new Python file, enter the following code, and import our library:

```
from pymycobot.mycobot280 import MyCobot280
```

Note:

1. If you enter `from pymycobot.mycobot280 import MyCobot280`, there is no red wavy line under the font, which proves that it has been successfully installed and can be used. If a red wavy line appears, you can refer to [How to install the API library](#), [How to call the API library](#).
2. If you do not want to install the API library through the above command, you can download the project to your local computer through the following github.

4.1 First-time self-check

First, go to the project address: <https://github.com/elephantrobotics/pymycobot>. Then click the Code button on the right side of the webpage, and then click Download ZIP to download it locally. Put the pymycobot folder in the compressed package pymycobot file project into your python dependency library directory, and you can directly import and use it.



Simple Demonstration

Create a new Python file in PyCharm and enter the following code to execute the LED flashing.

Note: The corresponding baud rates of various devices are different. Please refer to the information to understand their baud rates when using them. The serial port number can be viewed through [Calculator Device Manager](#) or the serial port assistant.

The following are the corresponding codes for myCobot.

- **myCobot**

4.1 First-time self-check

```
# demo.py
from pymycobot.mycobot280 import MyCobot280

import time
#The above codes are required to be written, which means importing the project package

# MyCobot280 class initialization requires two parameters:
# The first is the serial port string, such as:
#     linux: "/dev/ttyUSB0"
#     windows: "COM3"
# The second is the baud rate::
#     Arduino version is: 115200
#
# Example:
#     mycobot-Arduino:
#         linux:
#             mc = MyCobot280("/dev/ttyUSB0", 115200)
#         windows:
#             mc = MyCobot280("COM3", 115200)

# Initiate MyCobot280
# Create object code here for windows version
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
time.sleep(2)

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

If the following error message appears when executing the code, please check carefully whether the serial port number in the program is correct. You can solve this error message by checking the serial port number of your computer and changing it to the serial port number you found in the program. If the program runs normally but the robot arm does not respond, please check whether the baud rate is entered correctly.

4.1 First-time self-check

Traceback (most recent call last):

```
File "D:/python/Tms-GCN-edit/Tms-GCN-PyTorch/JointControl.py", line 24, in <module>
```

```
    mc = MyCobot("COM7", 115200)
```

```
File "C:\Users\Lenovo\AppData\Roaming\Python\Python38\site-packages\pymycobot\mycobot.py", line 69, in __init__
```

```
    self._serial_port.open()
```

```
File "C:\Users\Lenovo\AppData\Roaming\Python\Python38\site-packages\serial\serialwin32.py", line 64, in open
```

```
    raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))
```

```
serial.serialutil.SerialException: could not open port 'COM7': FileNotFoundError(2, 'The system cannot find the file speci
```

MyCobot 280

[toc]

Python API usage instructions

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

```
# Example
from pymycobot import MyCobot280

mc = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the actual situation
time.sleep(2)
# Gets the current angle of all joints
angles = mc.get_angles()
print(angles)

# Set 1 joint to move to 40 and speed to 20
mc.send_angle(1, 40, 20)
```

Note: Some function interfaces have return values, but if you enter the code directly, the result returned is no return value. You need to use the print function to print out the result. For example, if you want to get the current angle value of the robot arm, you can use get_angles(), but directly entering this function will not give you any result. The correct way to write it is: print(get_angles()) to print out the speed value. If the API description below indicates that there is no return value, you do not need to use the print function. Otherwise, you need to use the print function to print the result.

1. System Status

1.1 get_system_version()

- **function:** get system version
- **Return value:** system version

1.2 get_basic_version()

- **function:** Get basic firmware version for M5 version
- **Return value:** float firmware version

1.3 get_error_information()

- **function:** Obtaining robot error information
- **Return value:**
 - 0: No error message.
 - 1 ~ 6: The corresponding joint exceeds the limit position.

4.1 First-time self-check

- o 16 ~ 19: Collision protection.
- o 32: Kinematics inverse solution has no solution.
- o 33 ~ 34: Linear motion has no adjacent solution.

1.4 `clear_error_information()`

- **function:** Clear robot error message

2. Overall Status

2.1 `power_on()`

- **function:** atom open communication (default open)
- **Return value:**
 - o 1 - Power on completed.

2.2 `power_off()`

- **function:** Power off of the robotic arm
- **Return value:**
 - o 1 - Power on completed.

2.3 `is_power_on()`

- **function:** judge whether robot arms is powered on or not
- **Return value:**
 - o 1 : power on
 - o 0 : power off
 - o -1 : error

2.4 `release_all_servos()`

- **function:** release all robot arms
 - o **Attentions:** After the joint is disabled, it needs to be enabled to control within 1 second
- **Parameters:** `data` (optional): The way to relax the joints. The default is damping mode, and if the 'data' parameter is provided, it can be specified as non damping mode (1- Undamping).
- **Return value:**
 - o 1 - release completed.

2.5 `focus_servo(servo_id)`

- **function:** Power on designated servo
- **Parameters:**
 - o `servo_id`: int, 1-6
- **Return value:**
 - o 1 : complete

2.6 is_controller_connected()

- **function:** Whether connected with Atom
- **Return value:**
 - 1 : succeed
 - 0 : failed
 - -1 : error data

2.7 read_next_error()

- **function:** Robot Error Detection
- **Return value:** list len 6
 - 0 : No abnormality
 - 1 : Communication disconnected
 - 2 : Unstable communication
 - 3 : Servo abnormality

2.8 get_fresh_mode()

- **function:** Query sports mode
- **Return value:**
 - 0 : Interpolation mode
 - 1 : Refresh mode

2.9 set_fresh_mode()

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

2.10 set_free_mode()

- **function:** set to free mode
- **Parameters:**
 - 1 : open free mode
 - 0 : close free mode
- **Return value:**
 - 1 : complete

2.11 is_free_mode()

- **function:** Check if it is free mode

- **Return value:**

- 1 : free mode
- 0 : on-free mode

2.12 focus_all servos()

- **Function:** All servos are powered on

- **Return value:**

- 1 : complete

2.13 set_vision_mode()

- **Function:** Set the vision tracking mode, limit the attitude flip of send_coords in refresh mode. (Applicable only to vision tracking function)

- **Parameter:**

- 1 : open
- 0 : close

- **Return value:**

- 1 : complete

3.MDI Mode and Operation

3.1 get_angles()

- **function:** get the degree of all joints
- **Return value:** list a float list of all degree

3.2 send_angle(id, degree, speed)

- **function:** send one degree of joint to robot arm
- **Parameters:**

- id : Joint id, range int 1-6
- degree : degree value(float)

Note: ⚠ This joint limit information function is only available on Atom firmware ≥ 7.3 and pymycobot library $\geq 4.0.2$.

| Joint Id | Range |
|----------|------------|
| 1 | -168 ~ 168 |
| 2 | -140 ~ 140 |
| 3 | -150 ~ 150 |
| 4 | -150 ~ 150 |
| 5 | -155 ~ 160 |
| 6 | -180 ~ 180 |

4.1 First-time self-check

- o `speed` : the speed and range of the robotic arm's movement 1~100
- **Return value:**
 - o `1` : complete

3.3 `send_angles(angles, speed)`

- **function:** Send all angles to all joints of the robotic arm
- **Parameters:**
 - o `angles` : a list of degree value(`List[float]`), length 6
 - o `speed` : (`int`) 1 ~ 100
- **Return value:**
 - o `1` : complete

3.4 `get_coords()`

- **function:** Obtain robot arm coordinates from a base based coordinate system
- **Return value:** a float list of coord:[x, y, z, rx, ry, rz]

3.5 `send_coord(id, coord, speed)`

- **function:** send one coord to robot arm
- **Parameters:**
 - o `id` :send one coord to robot arm, 1-6 corresponds to [x, y, z, rx, ry, rz]
 - o `coord` : coord value(`float`)

| Coord ID | Range |
|----------|------------------|
| x | -281.45 ~ 281.45 |
| y | -281.45 ~ 281.45 |
| z | -70 ~ 412.67 |
| rx | -180 ~ 180 |
| ry | -180 ~ 180 |
| rz | -180 ~ 180 |

- o `speed` : (`int`) 1-100
- **Return value:**
 - o `1` : complete

3.6 `send_coords(coords, speed, mode)`

- **function:** Send overall coordinates and posture to move the head of the robotic arm from its original point to your specified point
- **Parameters:**
 - o `coords` : a list of coords value [x,y,z,rx,ry,rz] ,length6
 - o `speed` (`int`) : 1 ~ 100
 - o `mode`: (`int`) 0 - angular, 1 - linear
- **Return value:**

- o 1 : complete
-

3.7 pause()

- **function:** Control the instruction to pause the core and stop all movement instructions
- **Return value:**
 - o 1 - stopped
 - o 0 - not stop
 - o -1 - error

3.8 sync_send_angles(angles, speed, timeout=15)

- **function:** Send the angle in synchronous state and return when the target point is reached
- **Parameters:**
 - o angles : a list of degree value(List[float]), length 6
 - o speed : (int) 1 ~ 100
 - o timeout : default 15 seconds
- **Return value:**
 - o 1 : complete

3.9 sync_send_coords(coords, speed, mode=0, timeout=15)

- **function:** Send the coord in synchronous state and return when the target point is reached
- **Parameters:**
 - o coords : a list of coord value(List[float]), length 6
 - o speed : (int) 1 ~ 100
 - o mode : (int) 0 - angular (default) , 1 - linear
 - o timeout : default 15 seconds
- **Return value:**
 - o 1 : complete

3.10 get_angles_coords()

- **function:** Get joint angles and coordinates
- **Return value:** A list with a length of 12. The first six digits are angle information, and the last six digits are coordinate information.

3.11 is_paused()

- **function:** Check if the program has paused the move command
- **Return value:**
 - o 1 - paused
 - o 0 - not paused
 - o -1 - error

3.12 resume()

- **function:** resume the robot movement and complete the previous command
 - **Return value:**
 - o 1 - complete
-

3.13 stop()

- **function:** stop all movements of robot
- **Return value:**
 - 1 - stopped
 - 0 - not stop
 - -1 - error

3.14 is_in_position(data, flag)

- **function :** judge whether in the position.
- **Parameters:**
 - data: Provide a set of data that can be angles or coordinate values. If the input angle length range is 6, and if the input coordinate value length range is 6
 - flag data type (value range 0 or 1)
 - 0 : angle
 - 1 : coord
- **Return value:**
 - 1 - true
 - 0 - false
 - -1 - error

3.15 is_moving()

- **function:** judge whether the robot is moving
- **Return value:**
 - 1 moving
 - 0 not moving
 - -1 error

3.16 angles_to_coords(angles)

- **Function :** Convert angles to coordinates.
- **Parameters:**
 - angles : list List of floating points for all angles.
- **Return value:** list List of floating points for all coordinates.

3.17 solve_inv_kinematics(target_coords, current_angles)

- **Function :** Convert coordinates to angles.
- **Parameters:**
 - target_coords : list List of floating points for all coordinates.
 - current_angles : list List of floating points for all angles, current angles of the robot
- **Return value:** list List of floating points for all angles.

4. JOG Mode and Operation

4.1 jog_angle(joint_id, direction, speed)

- **function:** jog control angle
- **Parameters:**

4.1 First-time self-check

- `joint_id` : Represents the joints of the robotic arm, represented by joint IDs ranging from 1 to 6
- `direction(int)` : To control the direction of movement of the robotic arm, input `0` as negative value movement and input `1` as positive value movement
- `speed` : 1 ~ 100
- **Return value:**
 - `1` : complete

4.2 `jog_coord(coord_id, direction, speed)`

- **function:** jog control coord.
- **Parameters:**
 - `coord_id` : (`int`) Coordinate range of the robotic arm: 1~6
 - `direction` : (`int`) To control the direction of machine arm movement, `0` - negative value movement, `1` - positive value movement
 - `speed` : 1 ~ 100
- **Return value:**
 - `1` : complete

4.3 `jog_rpy(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`) To control the direction of machine arm movement, `1` - forward rotation, `0` - reverse rotation
 - `speed` : (`int`) 1 ~ 100
- **Return value:**
 - `1` : complete

4.4 `jog_increment_angle(joint_id, increment, speed)`

- **Function:** Angle stepping, single joint angle increment control
- **Parameter:**
 - `joint_id` : 1-6
 - `increment` : Incremental movement based on the current position angle
 - `speed` : 1~100
- **Return value:**
 - `1` : Completed

4.5 `jog_increment_coord(id, increment, speed)`

- **Function:** Coordinate stepping, single coordinate increment control
- **Parameter:**
 - `id` : Coordinate axis 1-6
 - `increment` : Incremental movement based on the current position coordinate
 - `speed` : 1~100
- **Return value:**
 - `1` : Completed

4.6 `set_encoder(joint_id, encoder, speed)`

- **function:** Set a single joint rotation to the specified potential value
- **Parameters**
 - `joint_id` : (int) 1-6
 - `encoder` : 0 ~ 4096
 - `speed` : 1 ~ 100
- **Return value:**
 - `1` : complete

4.7 `get_encoder(joint_id)`

- **function:** Set a single joint rotation to the specified potential value
- **Parameters**
 - `joint_id` : (int) 1-6
- **Return value:** (int) Joint potential value

4.8 `set_encoders(encoders, speed)`

- **function:** Set the six joints of the manipulator to execute synchronously to the specified position.
- **Parameters**
 - `joint_id` : (int) 1-6
 - `encoder` : 0 ~ 4096
 - `speed` : 1 ~ 100
- **Return value:**
 - `1` : complete

4.9 `get_encoders()`

- **function:** Get the six joints of the manipulator.
- **Return value:** (list) the list of encoders

5. Running status and Settings

5.1 `get_joint_min_angle(joint_id)`

- **function:** Gets the minimum movement angle of the specified joint
- **Parameters:**
 - `joint_id` : Enter joint ID (range 1-6)
- **Return value:** float Angle value

5.2 `get_joint_max_angle(joint_id)`

- **function:** Gets the maximum movement angle of the specified joint
- **Parameters:**
 - `joint_id` : Enter joint ID (range 1-6)
- **Return value:** float Angle value

5.3 `set_joint_min(id, angle)`

- **function:** Set minimum joint angle limit
- **Parameters:**
 - `id` : Enter joint ID (range 1-6)
 - `angle` : Refer to the limit information of the corresponding joint in the `send_angle()` interface, which must not be less than the minimum value
- **Return value:**
 - `1` : complete

5.4 `set_joint_max(id, angle)`

- **function:** Set maximum joint angle limit
- **Parameters:**
 - `id` : Enter joint ID (range 1-6)
 - `angle` : Refer to the limit information of the corresponding joint in the `send_angle()` interface, which must not be greater than the maximum value
- **Return value:**
 - `1` : complete

6. Joint motor control

6.1 `is_servo_enable(servo_id)`

- **function:** Detecting joint connection status
- **Parameters:** `servo_id` 1-6
- **Return value:**
 - `1` : Connection successful
 - `0` : not connected
 - `-1` : error

6.2 `is_all_servo_enable()`

- **function:** Detect the status of all joint connections
- **Return value:**
 - `1` : Connection successful
 - `0` : not connected
 - `-1` : error

6.3 `set_servo_calibration(servo_id)`

- **function:** The current position of the calibration joint actuator is the angle zero point
- **Parameters:**
 - `servo_id` : 1 - 6
- **Return value:**
 - `1` : complete

6.4 `release_servo(servo_id)`

- **function:** Set the specified joint torque output to turn off
- **Parameters:**

4.1 First-time self-check

- `servo_id` : 1 ~ 6
- **Return value:**
 - 1 : release successful
 - 0 : release failed
 - -1 : error

6.5 `focus_servo(servo_id)`

- **function:** Set the specified joint torque output to turn on
- **Parameters:** `servo_id` : 1 ~ 6
- **Return value:**
 - 1 : focus successful
 - 0 : focus failed
 - -1 : error

6.6 `set_servo_data(servo_id, data_id, value, mode=None)`

- **function:** Set the data parameters of the specified address of the steering gear
- **Parameters:**
 - `servo_id` : (int) joint id 1 - 6
 - `data_id` : (int) Data address
 - `value` : (int) 0 - 4096
 - `mode` : 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.
- **Return value:**
 - 1 : complete

6.7 `get_servo_data(servo_id, data_id, mode=None)`

- **function:** Read the data parameter of the specified address of the steering gear.
- **Parameters:**
 - `servo_id` : (int) joint id 1 - 6
 - `data_id` : (int) Data address
 - `mode` : 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.
- **Return value:** 0 ~ 4096

6.8 `joint_brake(joint_id)`

- **function:** Make it stop when the joint is in motion, and the buffer distance is positively related to the existing speed
- **Parameters:**
 - `joint_id` : (int) joint id 1 - 6
- **Return value:**
 - 1 : complete

7. 9g Servo

7.1 `move_round()`

- **function:** Drive the 9g steering gear clockwise for one revolution
- **Return value:**

- 1 : complete
-

7.2 set_four_pieces_zero()

- **function:** Set the zero position of the four-piece motor
- **Return value:**
 - 1 : success
 - 0 : failed

8. Servo state value

8.1 get_servo_speeds()

- **function:** Get the movement speed of all joints
- **Return value:** A list unit step/s

8.2 get_servo_voltages()

- **function:** Get joint voltages
- **Return value:** A list volts < 24 V

8.3 get_servo_status()

- **function:** Get the movement status of all joints
- **Return value:** A list,[voltage, sensor, temperature, current, angle, overload], a value of 0 means no error, a value of 1 indicates an error

8.4 get_servo_temps()

- **function:** Get joint temperature
- **Return value:** A list unit °C

9. Robotic arm end IO control

9.1 set_color(r, g, b)

- **function:** Set the color of the end light of the robotic arm
- **Parameters:**
 - r (int) : 0 ~ 255
 - g (int) : 0 ~ 255
 - b (int) : 0 ~ 255
- **Return value:**
 - 1 : complete

9.2 set_digital_output(pin_no, pin_signal)

- **function:** set IO statue
 - **Parameters**
 - pin_no (int): Pin number
-

4.1 First-time self-check

- `pin_signal` (int): 0 / 1
- **Return value:**
 - `1` : complete

9.3 `get_digital_input(pin_no)`

- **function:** read IO status
- **Parameters:** `pin_no` (int)
- **Return value:** signal

9.4 `set_pin_mode(pin_no, pin_mode)`

- **function:** Set the state mode of the specified pin in atom.
- **Parameters**
 - `pin_no` (int): Pin number
 - `pin_mode` (int): 0 - input, 1 - output, 2 - input_pullup
- **Return value:**
 - `1` : complete

10. Robotic arm end gripper control

10.1 `set_gripper_state(flag, speed, _type_1=None, is_torque=None)`

- **function:** Adaptive gripper enable
- **Parameters:**
 - `flag` (int) : 0 - open 1 - close, 254 - release
 - `speed` (int) : 0 ~ 100
 - `_type_1` (int) :
 - `1` : Adaptive gripper (default state is 1)
 - `2` : A nimble hand with 5 fingers
 - `3` : Parallel gripper
 - `4` : Flexible gripper
 - `is_torque` (int) : Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-end Atom firmware version is ≥ 6.5**)
 - `0` : Non-force-controlled gripper
 - `1` : Force-controlled gripper
- **Return value:**
 - `1` : complete

10.2 `set_gripper_value(gripper_value, speed, gripper_type=None, is_torque=None)`

- **function:** Set the gripper value
- **Parameters:**

4.1 First-time self-check

- `gripper_value (int)` : 0 ~ 100

- `speed (int)` : 0 ~ 100
- `gripper_type (int)` :
 - 1 : Adaptive gripper (default state is 1)
 - 3 : Parallel gripper
 - 4 : Flexible gripper
- `is_torque (int)` : Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-end Atom firmware version is ≥ 6.5**)
 - 0 : Non-force-controlled gripper
 - 1 : Force-controlled gripper
- **Return value:**
 - 1 : complete

10.3 `set_gripper_calibration()`

- **function:** Set the current position of the gripper to zero
- **Return value:**
 - 1 : complete

10.4 `is_gripper_moving()`

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

10.5 `get_gripper_value()`

- **function:** Get the value of gripper
- **Parameters:**
 - `gripper_type` : (int) default 1
 - 1: Adaptive gripper
 - 3: Parallel gripper
 - 4: Flexible gripper
- **Return value:** gripper value (int)

10.6 `set_pwm_output(channel, frequency, pin_val)`

- **function:** PWM control
- **Parameters:**
 - `channel` : (int): IO number.
 - `frequency` : (int): clock frequency
 - `pin_val` : (int) Duty cycle 0 ~ 256; 128 means 50%
- **Return value:**
 - 1 : complete

10.7 set HTS gripper torque(torque)

- **function:** Set new adaptive gripper torque
- **Parameters:**
 - torque : (int): 150 ~ 980
- **Return value:**
 - 0 : Set failed
 - 1 : Set successful

10.8 get HTS gripper torque()

- **function:** Get gripper torque
- **Return value:** (int) 150 ~ 980

10.9 get gripper protect current()

- **function:** Get the gripper protection current
- **Return value:** (int) 1 ~ 500

10.10 set gripper protect current(current)

- **function:** Set the gripper protection current
- **Parameters:**
 - current : (int): 1 ~ 500
- **Return value:**
 - 1 : complete

10.11 init gripper()

- **function:** Initialize gripper
- **Return value:**
 - 1 : complete

10.12 gripper stop()

Note: Atom firmware version 7.3 or higher is required.

- **Function:** Stop gripper movement
- **Return value:**
 - 1 : Completed

10.13 is torque gripper()

- **Function:** Determines if the gripper is a force-controlled gripper type
 - **Return Value:**
 - 40 : Force-controlled gripper
 - 9 : Non-force-controlled gripper
-

11. Set bottom IO input/output status

11.1 `set_basic_output(pin_no, pin_signal)`

- **function:** Set Base IO Output
- **Parameters:**
 - `pin_no` (`int`) Pin port number
 - `pin_signal` (`int`): 0 - low. 1 - high

11.2 `get_basic_input(pin_no)`

- **function:** Read base IO input
- **Parameters:**
 - `pin_no` (`int`) pin number
- **Return value:** 0 - low. 1 - high

12. TOF

12.1 `get_tof_distance()`

- **function:** Get the detected distance (Requires external distance detector)
- **Return value:** (int) The unit is mm.

13. Communication mode

13.1 `set_transponder_mode(mode)`

- **function:** Set basic communication mode
- **Parameters:**
 - `mode` : 0 - Turn off transparent transmission, 1 - Open transparent transmission
- **Return value:**
 - 1 : complete

13.2 `get_transponder_mode()`

- **function:** Get basic communication mode
- **Parameters:**
- **Return value:**
 - 1 : Open transparent
 - 0 : Turn off transparent transmission

14. Cartesian space coordinate parameter setting

14.1 `set_tool_reference(coords)`

- **function:** Set tool coordinate system.
- **Parameters:**
 - `coords` : (`list`) [X, y, Z, rx, ry, rz].
- **Return value:**
 - 1 : complete

14.2 `get_tool_reference(coords)`

- **function:** Get tool coordinate system.
- **Return value:** (`list`) [x, y, z, rx, ry, rz]

14.3 `set_world_reference(coords)`

- **function:** Set world coordinate system.
- **Parameters:**
 - `coords` : (`list`) [x, y, z, rx, ry, rz].
- **Return value:**
 - `1` : complete

14.4 `get_world_reference()`

- **function:** Get world coordinate system.
- **Return value:** `list` [x, y, z, rx, ry, rz].

14.5 `set_reference_frame(rftype)`

- **function:** Set base coordinate system.
- **Parameters:** `rftype` : 0 - base 1 - tool.
- **Return value:**
 - `1` : complete

14.6 `get_reference_frame()`

- **function:** Get base coordinate system.
- **Return value:** (`list`) [x, y, z, rx, ry, rz].

14.7 `set_movement_type(move_type)`

- **function:** Set movement type.
- **Parameters:**
 - `move_type` : 1 - moveI, 0 - moveJ.
- **Return value:**
 - `1` : complete

14.8 `get_movement_type()`

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

14.9 `set_end_type(end)`

- **function:** Set end coordinate system
- **Parameters:**
 - `end` (int) : `0` - flange, `1` - tool
- **Return value:**
 - `1` : complete

14.10 `get_end_type()`

- **function:** Obtain the end coordinate system
- **Return value:**
 - `0` - flange
 - `1` - tool

15. `utils` (module)

This module supports some helper methods. Use the code entered at the beginning of the file to import the module:

```
from pymycobot import utils
```

16.1 `utils.get_port_list()`

- **Function:** Get a list of all current serial port numbers
- **Return value:** Serial port list (`list`)

16.2 `utils.detect_port_of_basic()`

- **Function:** Return the first detected serial port number of M5 Basic. (Only one serial port number will be returned)
- **Return value:** Return the detected port number. If no serial port number is detected, it will return: `None`

Joint Control

For serial multi-joint robots, the control of joint space is to control the variables of each joint of the robot, and the goal is to make each joint of the robot reach the target position at a certain speed.

Note: When setting the angle, the limit of different series of robot arms is different. For details, please refer to the parameter introduction of the corresponding model.

Single joint control

send_angle(id, degree, speed)

- **Function:** Send the specified single joint movement to the specified angle
- **Parameter description:**
 - `id` : Represents the joint of the robot arm. Use numbers 1-6 to represent it.
 - `degree` : Represents the angle of the joint
 - `speed` : Represents the speed of the robot arm movement, ranging from 1 to 100
- **Return value:** 1

set_encoder(joint_id, encoder, sp)

- **Function:** Send the specified single joint movement to the specified potential value
- **Parameter description:**
 - `joint_id` : Represents the joint of the robot arm. Use numbers 1-6 to represent it.
 - `encoder` : represents the potential value of the robot arm, the value range is 0 ~ 4096
 - `sp` : Indicates the speed of the robot arm movement, ranging from 1 to 100
- **Return value:** 1

Multi-joint control

get_angles()

- **Function:** Get all joint angles
- **Return value:** `list` A list of floating point values, representing the angles of all joints

send_angles(degrees, speed)

- **Function:** Send all angles to all joints of the robot arm
- **Parameter description:**
 - `degrees` : (`List[float]`) contains the angles of all joints. the representation is: `[20,20,20,20,20,20]`
 - `speed` : Indicates the speed of the robot arm, the value range is 1-100
- **Return value:** 1

set_encoders(encoders, sp)

- **Function:** Send potential values to all joints of the robot arm
 - **Parameter description:**
 - `encoder` : Indicates the potential value of the robot arm, the value range is 0 ~ 4096, the length is 6, representation method is: `[2048,2048,2048,2048,2048,2048]`
 - `sp` : Indicates the speed of the robot arm, the value range is 1-100
 - **Return value:** 1
-

sync_send_angles(degrees, speed, timeout=15)

- **Function:** Synchronously send angles and return when reaching the target point
- **Parameter description:**
 - `degrees` : List of angle values of each joint `List[float]`
 - `speed` : (`int`) The speed of the robot arm, the value range is 1-100
 - `timeout` : The default time is 15s
- **Return value:** 1

get_radians()

- **Function:** Get the radians of all joints
- **Return value:** `list` contains a list of all joint radian values

send_radians(radians, speed)

- **Function:** Send radian values to all joints of the robot arm
- **Parameter description:**
 - `radians` : `list` Indicates the radian value of the robot arm, the value range is -5~5
- **Return value:** 1

Example use

```

from pycobot.pycobot280 import MyCobot280
import time

# MyCobot280 Class initialization requires two parameters:
# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# windows: "COM3"
# The second is the baud rate:
# Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it according
# For example:
# mycobot-Arduino:
# linux:
# mc = MyCobot280("/dev/ttyUSB0", 115200)
# windows:
# mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# Here is the object code for the windows version
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according
time.sleep(2)
# By passing the angle parameter, each joint of the robot arm moves to the corresponding position [0, 0, 0, 0, 0, 0]
mc.send_angles([0, 0, 0, 0, 0, 0], 50)

# Set the waiting time to ensure that the robot has reached the specified position
time.sleep(2.5)

# Move joint 1 to the position of 90
mc.send_angle(1, 90, 50)

# Set the waiting time to ensure that the robot has reached the specified position
time.sleep(2)

# The following code can make the robot swing left and right
# Set the number of loops
num=5
while num > 0:
    # Move joint 2 to the position of 50
    mc.send_angle(2, 50, 50)

    # Set the waiting time to ensure that the robot has reached the specified position
    time.sleep(1.5)

    # Move joint 2 to the position of -50
    mc.send_angle(2, -50, 50)

    # Set the waiting time to ensure that the robot has reached the specified position
    time.sleep(1.5)

```

4.1 First-time self-check

```
num -= 1

# Retract the robot arm. You can swing the robot arm manually, and then use the get_angles() function to get the coordinates
# Use this function to make the robot arm reach the position you want.
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)

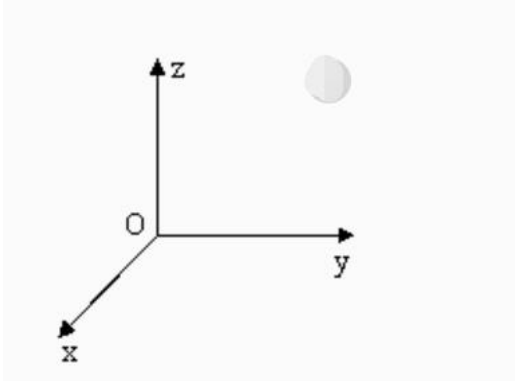
# Set the waiting time to ensure that the robot arm has reached the specified position
time.sleep(2.5)

# Relax the robot arm and swing it manually
mc.release_all_servos()
```

Coordinate control

It is mainly used to realize intelligent route planning to let the robot arm move from one position to another specified position. It is divided into $[x, y, z, rx, ry, rz]$, where $[x, y, z]$ represents the position of the robot head in space (the coordinate system is the [rectangular coordinate system](#)), and $[rx, ry, rz]$ represents the posture of the robot head at this point (the coordinate system is the Euler coordinate system). The implementation of the algorithm and the representation of Euler coordinates require certain academic knowledge. We will not explain it too much here. As long as we understand the rectangular coordinate system, we can use this function well.

Note: When setting coordinates, different series of robot arm joint structures are different. For the same set of coordinates, different series of robot arms will show different postures.



Single parameter coordinate

send_coord(id,coord,speed)

- **Function:** Send a single coordinate value to the robot for movement
- **Parameter description:**
 - **id** : Represents the coordinate of the robot. Represented by numbers 1-6, for example, the X axis can be filled in 1, the Y can be filled in 2, and so on
 - **coord** : Enter the coordinate value you want to reach
 - **speed** : Indicates the speed of the robot movement, the range is 1-100
- **Return value:** 1

Multi-parameter coordinates

get_coords()

- **Function:** Get current coordinates and posture
- **Return value:** `list` contains a list of coordinates and postures, length is 6, in order $[x, y, z, rx, ry, rz]$

send_coords(coords, speed, mode)

- **Function:** Send the overall coordinates and posture, and let the robot head move from the original point to the point you specify.
- **Parameter description:**
 - **coords** : $[x,y,z,rx,ry,rz]$ coordinate value, length is 6
 - **speed** : Indicates the speed of the robot movement, the range is 1-100

4.1 First-time self-check

- `mode` : (`int`): The value is limited to 0 and 1.
 - 0 means that the path of the robot head is nonlinear, that is, the route is randomly planned, as long as the robot head moves to the specified point while maintaining the specified posture.
 - 1 means that the path of the robot head is linear, that is, the intelligent planning route allows the robot head to move to the specified point in a straight line.
- **Return value:** 1

`set_tool_reference(coords)`

- **Function:** Set the tool coordinate system.
- **Parameter description:**
 - `coords` :
 - Six axes: [x,y,z,rx,ry,rz] coordinate values, length 6, x,y,z range -280 ~ 280, rx,ry,yz range -314 ~ 314
- **Return value:** 1

`get_tool_reference()`

- **Function:** Get the tool coordinate system.
- **Return value:** Return a coordinate list with a length of 6

`get_world_reference()`

- **Function:** Get the world coordinate system.
- **Return value:** Return a coordinate list with a length of 6

`set_world_reference(coords)`

- **Function:** Set the world coordinate system.
- **Parameter description:**
 - `coords` :
 - Six axes: [x,y,z,rx,ry,rz] coordinate values, length is 6, x,y,z range is -280 ~ 280, rx,ry,yz range is -314 ~ 314
- **Return value:** 1

`set_reference_frame(rftype)`

- **Function:** Set the base coordinate system.
- **Parameter description:**
 - `rftype` : 0 - base coordinate system (default), 1 - world coordinate system
- **Return value:** 1

`get_reference_frame()`

- **Function:** Get the base coordinate system.
- **Return value:** 0 - base coordinate system, 1 - world coordinate system, -1 - communication failed

`set_end_type(end)`

- **Function:** Set the end coordinate system.
- **Parameter description:**
 - `end` : 0 - flange (default), 1 - tool
- **Return value:** 1

`get_end_type()`

- **Function:** Get the end coordinate system.
- **Return value:** 0 - flange (default), 1 - tool, -1 - communication failed

Example use

```

from pymycobot.mycobot280 import MyCobot280
import time

# MyCobot280 class initialization requires two parameters:
# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# windows: "COM3"
# The second is the baud rate:
# Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it accordingly
# For example:
# mycobot-Arduino:
# linux:
# mc = MyCobot280("/dev/ttyUSB0", 115200)
# windows:
# mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# The following is the object code for the Windows version
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accordingly
time.sleep(2)

if mc.get_fresh_mode() != 1:
    mc.set_fresh_mode(1)

# Get the current head coordinates and posture
coords = mc.get_coords()
print(coords)
# # Intelligently plan the route, so that the head reaches the coordinates [57.0, -107.4, 316.3] in a linear manner, and maintain
mc.send_coords([57.0, -107.4, 316.3, -93.81, -12.71, -163.49], 80, 1)

# Set the waiting time to 1.5 seconds
time.sleep(1.5)

# Intelligently plan the route, let the head reach the coordinates [-13.7, -107.5, 223.9] in a linear manner, and maintain
mc.send_coords([-13.7, -107.5, 223.9, 165.52, -75.41, -73.52], 80, 1)

# Set the waiting time to 1.5 seconds
time.sleep(1.5)

# Only change the x coordinate of the head, set the x coordinate of the head to -40. Let it intelligently plan the route a
mc.send_coord(1, -40, 70)

```

IO control

IO is the input and output of data. There are multiple pins on the Basic and Atom of our robot arm. The input and output modes can be set through the following function interface.

- **myCobot:**



Basic IO

get_basic_input(pin_no)

- **Function:** Get the working status of the bottom pin number
- **Parameter description:** `pin_no` : Indicates the specific pin number at the bottom of the robot
- **Return value:** `pin_signal (int)` When the returned value is 0, it means it is running in the working state, and 1 means it is stopped

set_basic_output(pin_no, pin_signal)

- **Function:** Set the working status of the bottom pin number
- **Parameter description:**
 - `pin_no (int)` The number marked on the bottom of the device only takes the digital part
 - `pin_signal (int)`: Input 0 means set to running state, input 1 means stop state
- **Return value:** 1

get_tof_distance()

- **Function:** Get the detected distance (external distance detector is required)
- **Return value:** Detected distance value, unit is mm

Atom IO

set_pin_mode(pin_no, pin_mode)

- **Function:** Set the state mode of the specified pin in atom
- **Parameter description:**
 - `pin_no (int)`: The specific pin number on the top of the robot
 - `pin_mode (int)`: Limited to 0~2
 - 0 is set to running state

4.1 First-time self-check

- 1 is set to stop state
 - 2 is set to pull-up mode
-

- **Return value:** 1

set_digital_output(pin_no, pin_signal)

- **Function:** Set the working state of the end pin number
- **Parameter description:**
 - `pin_no (int)` The number marked at the end of the device only takes the digital part
 - `pin_signal (int)`: Enter 0 to set it to running state, enter 1 to stop state
- **Return value:** 1

get_digital_input(self, pin_no)

- **Function:** Get the working state of the end pin number
- **Parameter description:** `pin_no` : Indicates the specific pin number at the end of the robot
- **Return value:** `pin_signal (int)` When the returned value is 0, it means running in working state, and 1 means stop state

Case use

- **MyCobot 280-Arduino version:**

4.1 First-time self-check

```
from pycobot.mycobot280 import MyCobot280
import time

#Enter the above code to import the required packages for the project

# MyCobot280 class initialization requires two parameters:
# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# windows: "COM3"
# The second is the baud rate:
# Arduino version: 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it acc

# For example:
# mycobot-Arduino:
# linux:
# mc = MyCobot280("/dev/ttyUSB0", 115200)
# windows:
# mc = MyCobot280("COM3", 115200)

# Initialize a MyCobo280 object
# The following is the object creation code for the windows version
arm = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it acco

# Turn on the pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

# Stop the pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)

for i in range(2):
    pump_on()
    time.sleep(2)
    pump_off()
    time.sleep(2)
```

Gripper control

Before using Python to control the gripper, you need to install and connect the gripper on the robot arm. Different grippers are compatible with different robots (for specific adaptation information, please refer to [Product accessories](#).)

Note:

MyCobot 280 adaptive gripper inserts the gripper into the pins on the Atom, see the following figure:



Gripper control

`is_gripper_moving()`

- **Function:** Determine whether the gripper is running
- **Return value:**
 - `0` : Indicates that the gripper of the robot arm is not running
 - `1` : Indicates that the gripper of the robot arm is running
 - `-1` : Indicates an error

`set_gripper_state(flag, speed, _type_1=None, is_torque=None)`

- **function:** Adaptive gripper enable
- **Parameters:**
 - `flag (int)` : 0 - open 1 - close, 254 - release
 - `speed (int)` : 0 ~ 100
 - `_type_1 (int)` :
 - `1` : Adaptive gripper (default state is 1)
 - `2` : A nimble hand with 5 fingers
 - `3` : Parallel gripper
 - `4` : Flexible gripper
 - `is_torque (int)` : Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-end Atom firmware version is ≥ 6.5**)
 - `0` : Non-force-controlled gripper
 - `1` : Force-controlled gripper
- **Return value:**
 - `1` : complete

`set_gripper_value(gripper_value, speed, gripper_type=None, is_torque=None)`

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

4.1 First-time self-check

- `gripper_type (int)` :
 - `1` : Adaptive gripper (default state is 1)
 - `3` : Parallel gripper
 - `4` : Flexible gripper
- `is_torque (int)` : Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-end Atom firmware version is ≥ 6.5**)
 - `0` : Non-force-controlled gripper
 - `1` : Force-controlled gripper
- **Return value:**
 - `1` : complete

`get_gripper_value(gripper_type=None)`

- **Function:** Get the current position data information of the gripper
- **Parameter description:**
 - `gripper_type` : gripper type, the default is adaptive gripper
 - `1` : adaptive gripper
 - `3` : parallel gripper
 - `4` : flexible gripper
- **Return value:** Gripper data information

`set_HTS_gripper_torque(torque)`

- **Function:** Set adaptive gripper torque
- **Parameter description:**
 - `torque` : 150 ~ 900
- **Return value:** 0 - Setting failed; 1 - Setting successful

`get_HTS_gripper_torque()`

- **Function:** Get adaptive gripper torque
- **Return value:** 150 ~ 900

`get_gripper_protect_current()`

- **Function:** Get the gripper protection current
- **Return value:** 1 ~ 500

`init_gripper()`

- **Function:** Initialize the gripper
- **Return value:** 0 - Initialization failed; 1 - Initialization successful

`set_gripper_protect_current(current)`

- **Function:** Set the gripper protection current
- **Parameter description:**
 - `current` : 1 ~ 500
- **Return value:** 0 - Initialization failed; 1 - Initialization successful

Example

```

from pymycobot.mycobot280 import MyCobot280
import time

#Enter the above code to import the packages required for the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)
    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 and let it rotate to the position 2048, speed 20
    mc.set_encoder(1, 2048, 20)
    time.sleep(2)
    # Set six joint positions and let the robot arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)

    # Let the gripper reach the 100 state at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the 0 state at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to open the claw quickly at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)

    # Set the state of the gripper to close the claw quickly at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())

if __name__ == "__main__":
    # MyCobot280 class initialization requires two parameters:
    # The first is a serial string, such as:
    # linux: "/dev/ttyUSB0"
    # windows: "COM3"
    # The second is the baud rate:
    # Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it acc

```

4.1 First-time self-check

```
# Example:
# mycobot-Arduino:
# linux:
# mc = MyCobot280("/dev/ttyUSB0", 115200)
# windows:
# mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# Below is the object code for Arduino version
mc = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the board
time.sleep(2)
# Move it to zero
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```

Handle control

You can use the game controller to control the movement of the machine and use the gripper or suction pump to grab objects.

Note: The handle needs to be purchased separately, please contact the official customer service for details



The corresponding functions of the handle buttons are as follows:

myCobot:

- 1: RX coordinate value increases
- 2: RX coordinate value decreases
- 3: RY coordinate value increases
- 4: RY coordinate value decreases
- 5: X coordinate value increases
- 6: X coordinate value decreases

4.1 First-time self-check

- **7:** Y coordinate value decreases
- **8:** Y coordinate value increases
- **9:** Z coordinate value decreases
- **10:** Z coordinate value increases
- **11:** RZ coordinate value decreases
- **12:** RZ Coordinate value increases
- **13:** Wake up the handle. If the handle is not used for a long time after connection, it will enter sleep mode. You need to press this button to continue using it.
- **X:** Click the button to open the jaws
- **Y:** Click the button to close the jaws
- **A:** Click the button to turn on the suction pump
- **B:** Click the button to turn off the suction pump
- **Left 1:** Press and hold for 2s to initialize the robot to the joint zero position state.
- **Left 2:** Press and hold for 2s, the robot stops torque output and relaxes all joints.
- **Right 1:** Press and hold for 2s to initialize the robot to the initial point of movement.
- **Right 2:** Press and hold for 2s, the robot turns on torque output and all joints are locked.

Instructions for use

Connect the device

Connect MyCobot280 and the handle to the computer.

Install the required packages

Download code: <https://github.com/elephantrobotics/pymycobot>

Open the terminal, switch the path to the `pymycobot/demo/handle_control` folder, and run the following command:

```
pip3 install -r requirements.txt
```

Change the port number

myCobot

Edit the `myCobot280_handle_control.py` file

4.1 First-time self-check

```
import pygame
import time
from pymycobot import MyCobot280
import threading

# Change com7 to the actual port number detected by your computer, and modify the suction pump control logic
mc = MyCobot280("com7", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
time.sleep(1)
# start pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

# stop pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)
...

```

Run the program.

```
python3 myCobot280_handle_control.py
```

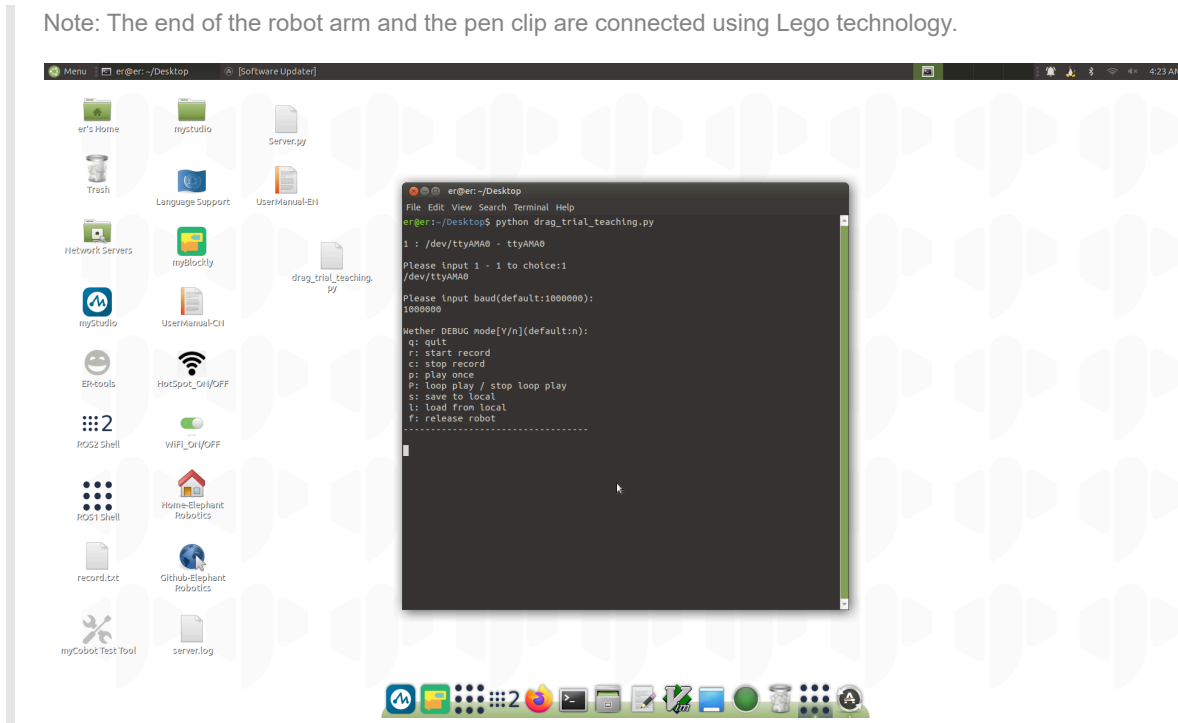
Note: After running the program, first click the **Right 1** button. After the machine reaches the initial point, other operations can be performed.

Draw a pattern

You can control the movement of the robot arm and realize the drawing operation by parsing the instructions in a gcode file.

Installation diagram

Note: The end of the robot arm and the pen clip are connected using Lego technology.



Instructions

1. Connect the device

Connect MyCobot to the computer, install the pen clip to the end of the robot arm, put the signature pen in the pen clip and tighten the screws to fix it.

Note: Use the G-type base 2.0 to fix the robot arm on the desktop, and place the A4 white paper on the desktop for drawing patterns.

2. Install the required packages

Download code: <https://github.com/elephantrobotics/pymycobot>

Open the terminal, switch the path to the `pymycobot/demo/myCobot_280_demo` folder, and run the following command:

```
pip install pyserial pymycobot
```

3. Change the port number

Edit the `280_draw_gcode.py` file

4.1 First-time self-check

```
# Change COM14 to the actual port number detected by your computer
import time
from pymycobot.mycobot280 import MyCobot280 # import mycobot library,if don't have, first 'pip install pymycobot'

# use PC and Arduino control
mc = MyCobot280('COM14', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the actual situation
# mc = MyCobot280('/dev/ttyUSB0',115200) # VM linux use
time.sleep(2)
...
```

Just run the program.

```
python 280_draw_gcode.py
```

Then, according to the terminal prompt, enter different numbers to select different patterns:

```
1-square
2-triangle
3-five point star
4-quit
```

Note: The initial point of the robot arm can be changed by yourself, but the posture of the J6 joint must be vertically downward, and the speed can also be changed by yourself, the default is 100 mm per second.

```
bash ...
# Send the initial point angle of the robot arm, the customized is 50,
# it can be and modified, as long as the end is facing down mc.send_angles([0, -40, -130, 80, 0, 50], 50)
# Wait 3 seconds for the robot arm to move to the specified angle time.sleep(3)
# Get the current coordinates of the robot arm get_coords = speed mc.get_coords() time.sleep(1.5)
# Save the parsed coordinates data_coords = []
# Set the drawing speed to 100, and the speed range is 0~100 draw_speed = 100 ...
```

Demonstration code

The following are various use cases and operation result videos. You can copy the code for use or modification (the robot arm model used in the following cases is MyCobot280 280. The parameters of different series of robot arms are different. Please pay attention to the modification).

Note: The corresponding baud rates of various devices are different. Please refer to the information to understand their baud rates when using them. The serial port number can be viewed through [Calculator Device Manager](#) or the serial port assistant.

1 Control RGB light board

```

from pymycobot import MyCobot280
import time

#The above needs to be written at the beginning of the code, which means importing the project package

# MyCobot280 class initialization requires two parameters: serial port and baud rate
# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# windows: "COM3"
# The second is the baud rate:
# Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it acc
# The following is such as:
# mycobot-Arduino:
# linux:
# mc = MyCobot280("/dev/ttyUSB0", 115200)
# windows:
# mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# The following is the object code for the Windows version
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
time.sleep(2)

i = 7
# Loop 7 times
while i > 0:
mc.set_color(0,0,255) #Blue light on
time.sleep(2) #Wait 2 seconds
mc.set_color(255,0,0) #Red light on
time.sleep(2) #Wait 2 seconds
mc.set_color(0,255,0) #Green light on
time.sleep(2) #Wait 2 seconds
i -= 1

```

2 Control the machine to return to the origin

```
from pymycobot import MyCobot280
# MyCobot280 class initialization requires two parameters:
# The first is the serial port string, such as:
#   linux: "/dev/ttyUSB0"
#   windows: "COM3"
# The second is the baud rate:
# Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it acc
# The following is such as:
# mycobot-Arduino:
#   linux:
#   mc = MyCobot280("/dev/ttyUSB0", 115200)
#   windows:
#   mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# The following is the object code for Arduino version
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
time.sleep(2)

# Check if the robot can be programmed

if mc.is_controller_connected() != 1:

print("Please connect the robot correctly to write the program")

exit(0)

# Fine-tune the robot to ensure that all the ports are aligned

# The alignment of the robot port is the standard. This is just an example

mc.send_angles([0, 0, 0, 0, 0, 0], 30)

# Calibrate the position at this time. The calibrated angle position is [0,0,0,0,0,0] and the potential value is [2048,204

# This for loop is equivalent to the set_gripper_ini() method

#for i in range(1, 7):

#mc.set_servo_calibration(i)
```

3 Single joint movement

```
from pymycobot import MyCobot280
import time

# The MyCobot280 class requires two parameters to be initialized:
# The first is the serial port string, such as:
# linux: "/dev/ttyUSB0"
# windows: "COM3"
# The second is the baud rate:
# Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it acc
# The following is such as:
# mycobot-Arduino:
# linux:
# mc = MyCobot280("/dev/ttyUSB0", 115200)
# windows:
# mc = MyCobot280("COM3", 115200)

# Create object code for Arduino version
mc=MyCobot280('COM3',115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according
time.sleep(2)

# Robot arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 40)
time.sleep(3)

# Control joint 3 to move 70°
mc.send_angle(Angle.J3.value,70,40)
time.sleep(3)

# Control joint 4 to move -70°
mc.send_angle(Angle.J4.value,-70,40)
time.sleep(3)

# Control joint 1 to move 90°
mc.send_angle(Angle.J1.value,90,40)
time.sleep(3)

# Control joint 5 to move -90°
mc.send_angle(Angle.J5.value,-90,40)
time.sleep(3)

# Control joint 5 to move 90°
mc.send_angle(Angle.J5.value,90,40)
time.sleep(3)
```

4 Multi-joint exercise

```
import time
from pymycobot import MyCobot280

# The MyCobot280 class requires two parameters to be initialized:
# The first is the serial port string, such as:
#     linux: "/dev/ttyUSB0"
#     windows: "COM3"
# The second is the baud rate:
#     Arduino version: 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according
#
# Example:
#     mycobot-Arduino:
#         linux:
#             mc = MyCobot280("/dev/ttyUSB0", 115200)
#         windows:
#             mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# 280-Arduino version object code
mc=MyCobot280('COM3',115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according
time.sleep(2)
# Robot arm reset to zero
mc.send_angles([0,0,0,0,0,0],50)
time.sleep(2.5)
# Control the different angles of rotation of multiple joints
mc.send_angles([90,45,-90,90,-90,90],50)
time.sleep(2.5)
# Robot arm reset to zero
mc.send_angles([0,0,0,0,0,0],50)
time.sleep(2.5)
# Control the different angles of rotation of multiple joints
mc.send_angles([-90,-45,90,-90,90,-90],50)
time.sleep(2.5)
```

5 Control the robot arm to swing left and right

```

from pymycobot import MyCobot280
import time

# Arduino version
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
time.sleep(2)
# Get the coordinates of the current position
angle_datas = mc.get_angles()
print(angle_datas)

# Use a sequence to pass coordinate parameters to move the robot to the specified position
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
print(mc.is_paused())
# Set the waiting time to ensure that the robot has reached the specified position
# while not mc.is_paused():
time.sleep(2.5)

# Move joint 1 to position 90
mc.send_angle(Angle.J1.value, 90, 50)

# Set the waiting time to ensure that the robot has reached the specified position
time.sleep(2)

# Set the number of loops
num = 5

# Let the robot swing left and right
while num > 0:
    # Move joint 2 to position 50
    mc.send_angle(2, 50, 50)
    # Set the waiting time to ensure that the robot has reached the specified position
    time.sleep(1.5)
    # Move joint 2 to position -50
    mc.send_angle(2, -50, 50)
    # Set the waiting time to ensure that the robot has reached the specified position
    time.sleep(1.5)
    num -= 1

# Retract the robot arm. You can swing the robot arm manually, and then use the get_angles() function to get the coordinat
# Use this function to make the robot arm reach the position you want.
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)

# Set the waiting time to ensure that the robot arm has reached the specified position
time.sleep(2.5)
# Relax the robot arm and swing it manually
mc.release_all_servos()

```

6 Controlling the robotic arm to dances

```

from pycobot import MyCobot280
import time

if __name__ == '__main__':
    # MyCobot280 class initialization requires two parameters:
    # The first is the serial port string, such as:
        # linux: "/dev/ttyUSB0"
        # windows: "COM3"
        # The second is the baud rate:
        # Arduino version: 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it acc
    # The following is such as:
        # mycobot-Arduino:
            # linux:
            # mc = MyCobot280("/dev/ttyUSB0", 115200)
            # windows:
            # mc = MyCobot280("COM3", 115200)

    # Initialize a MyCobot280 object
    # Arduino version
    mc = MyCobot280("COM3",115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
    time.sleep(2)
    # Set the start time
    start = time.time()
    # Let the robot reach the specified position
    mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
    # Determine whether it has reached the specified position
    while not mc.is_in_position([-1.49, 115, -153.45, 30, -33.42, 137.9], 0):
        # Let the robot resume movement
        mc.resume()
        # Let the robot move for 0.5s
        time.sleep(0.5)
        # Pause the movement of the robot
        mc.pause()
        # Determine whether the movement has timed out
        if time.time() - start > 3:
            break
    # Set the start time
    start = time.time()
    # Let the movement last for 30 seconds
    while time.time() - start < 30:
        # Let the robot reach this position quickly
        mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
        # Set the color of the light to [0,0,50]
        mc.set_color(0, 0, 50)
        time.sleep(0.7)
        # Let the robot reach this position quickly
        mc.send_angles([-1.49, 55, -153.45, 80, 33.42, 137.9], 80)

```

4.1 First-time self-check

```
# Set the color of the light to [0,50,0]
mc.set_color(0, 50, 0)
time.sleep(0.7)
```

7 Gripper control

```

from pymycobot import MyCobot280
import time
def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to position 2048, speed 20
    mc.set_encoder(1, 2048, 20)
    time.sleep(2)
    # Set six joint positions and let the robot arm rotate to the position at a speed of 20

    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    # mc.set_encoders([2048, 2900, 2048, 2048, 2048, 2048], 20)
    # mc.set_encoders([2048, 3000,3000, 3000, 2048, 2048], 50)
    time.sleep(3)
    # Get the position information of joint point 1
    print(mc.get_encoder(1))
    # Set the gripper to rotate to the position of 2048, speed 20
    mc.set_encoder(7, 2048, 20)
    time.sleep(3)
    # Set the gripper to rotate to the position of 1300, speed 20
    mc.set_encoder(7, 1300, 20)
    time.sleep(3)

    # Set the state of the gripper to open the claw quickly at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Set the state of the gripper to close the claw quickly at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    num=5
    while num>0:
        # Set the state of the gripper to open the claws quickly at a speed of 70
        mc.set_gripper_state(0, 70)
        time.sleep(3)
        # Set the state of the gripper to close the claws quickly at a speed of 70
        mc.set_gripper_state(1, 70)
        time.sleep(3)

```

4.1 First-time self-check

```
num-=1

# Get the value of the gripper
print("")
print(mc.get_gripper_value())
# mc.release_all_servos()

if __name__ == "__main__":
# MyCobot280 class initialization requires two parameters:
# The first is the serial port string, such as:
#     linux: "/dev/ttyUSB0"
#windows: "COM3"
# The second is the baud rate:
#     Arduino version is: 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it ac
#
# Example:
#     mycobot-Arduino:
#         linux:
#             mc = MyCobot280("/dev/ttyUSB0", 115200)
#         windows:
#             mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# Arduino version
mc = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord
time.sleep(2)
# Move it to zero position

mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```

8 Suction pump control

280-Arduino

4.1 First-time self-check

```
from pycobot import MyCobot280
import time

# The MyCobot280 class requires two parameters to initialize:
# The first is the serial port string, such as:
#     linux: "/dev/ttyUSB0"
#     windows: "COM3"
# The second is the baud rate:
#     Arduino version: 115200 # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the board
# Example:
#     mycobot-Arduino:
#         linux:
#             mc = MyCobot280("/dev/ttyUSB0", 115200)
#         windows:
#             mc = MyCobot280("COM3", 115200)

# Initialize a MyCobot280 object
# The following is the object code for the Arduino version
mc = MyCobot280('COM3',115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the board
time.sleep(2)
# The position of the robot arm movement
angles = [
    [92.9, -10.1, -60, 5.8, -2.02, -37.7],
    [92.9, -53.7, -83.05, 50.09, -0.43, -38.75],
    [92.9, -10.1, -87.27, 5.8, -2.02, -37.7]
]

# Turn on the suction pump
def pump_on():
    # Open the solenoid valve
    mc.set_basic_output(5, 0)
    time.sleep(0.05)

# Stop the suction pump
def pump_off():
    # Close the solenoid valve
    mc.set_basic_output(5, 1)
    time.sleep(0.05)
    # The exhaust valve starts working
    mc.set_basic_output(2, 0)
    time.sleep(1)
    mc.set_basic_output(2, 1)
    time.sleep(0.05)

# Robot arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
time.sleep(3)

# Turn on the suction pump
```

4.1 First-time self-check

```
pump_on()
mc.send_angles(angles[2], 30)
time.sleep(2)

# Suction small objects
mc.send_angles(angles[1], 30)
time.sleep(2)
mc.send_angles(angles[0], 30)
time.sleep(2)
mc.send_angles(angles[1], 30)
time.sleep(2)
#Turn off the suction pump
pump_off()
mc.send_angles(angles[0], 40)
time.sleep(1.5)
```

Introduction to ROS

ROS is an open source meta-operating system for robots. It provides the services that an operating system should have, including hardware abstraction, low-level device control, implementation of common functions, messaging between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run code across computers.

The "graph" of the ROS runtime is a loosely coupled point-to-point process network based on the ROS communication infrastructure. ROS implements several different communication methods, including services based on synchronous RPC-style communication, topics based on asynchronous streaming data, and parameter servers for data storage.

ROS is not a real-time framework, but ROS can be embedded in real-time programs. Willow Garage's PR2 robot uses a system called `pr2_etherCAT` to send or receive ROS messages in real time. ROS can also be seamlessly integrated with the Orocos real-time toolkit.

Design goals and features of ROS

Many people ask "What is the difference between ROS and other robotics software platforms?" This is a difficult question to answer. Because ROS is not a framework that integrates most functions or features. In fact, the main goal of ROS is to support code reuse for robotics research and development. ROS is a framework of distributed processes (i.e. nodes) that are packaged in packages and function packages that are easy to share and distribute.

ROS also supports a federated system similar to a code repository, which also enables collaboration and distribution of projects. This design allows the development and implementation of a project to be completely independent of decisions (not limited by ROS) from the file system to the user interface. At the same time, all projects can be integrated with the basic tools of ROS.

In order to support the main goal of sharing and collaboration, the ROS framework has several other characteristics:

- **Lean:** ROS is designed to be as lean as possible so that code written for ROS can be used with other robotics software frameworks. A corollary to this is that ROS can be easily integrated with other robotics software platforms: ROS has been integrated with OpenRAVE, Orocos and Player.
- **ROS-insensitive libraries:** The preferred development model of ROS is written in clean library functions that do not depend on ROS.
- **Language independence:** The ROS framework can be easily implemented in any modern programming language. ROS has implemented Python version, C++ version and Lisp version. It also has Java and Lua version experimental libraries.
- **Loose coupling:** The function modules in ROS are encapsulated in independent function packages or meta-function packages for easy sharing. The modules in the function package run as nodes and use ROS standard IO as the interface. Developers do not need to pay attention to the internal implementation of the module. As long as they understand the interface rules, they can reuse it, realizing point-to-point loose coupling connection between modules
- **Convenient testing:** ROS has a built-in unit/integration test framework called `roscpp`, which can easily install or uninstall test modules.
- **Scalable:** ROS can be applied to large runtime systems and large development processes.
- **Free and open source:** many developers and many function packages

Why use ROS

Through ROS, we can realize the simulation control of the robot arm in a virtual environment.

We will use the **rviz** platform to realize the visualization of the robot arm and use a variety of methods to operate our robot arm; through the **moveit** platform to plan and execute the robot arm's action path, we can achieve the effect of free control of the robot arm.

In the next chapter, we will learn how to control our products through the platform in ros.

ROS1 Tutorial Guide

[Environment Building](#)

[ROS1 Basics](#)

[rviz Introduction and Use](#)

[moveit Introduction and Use](#)

M5 version:

Install different versions of Ubuntu system in Linux

Virtual machine installation

Go to the [official website](#) to download the virtual machine Virtual Box

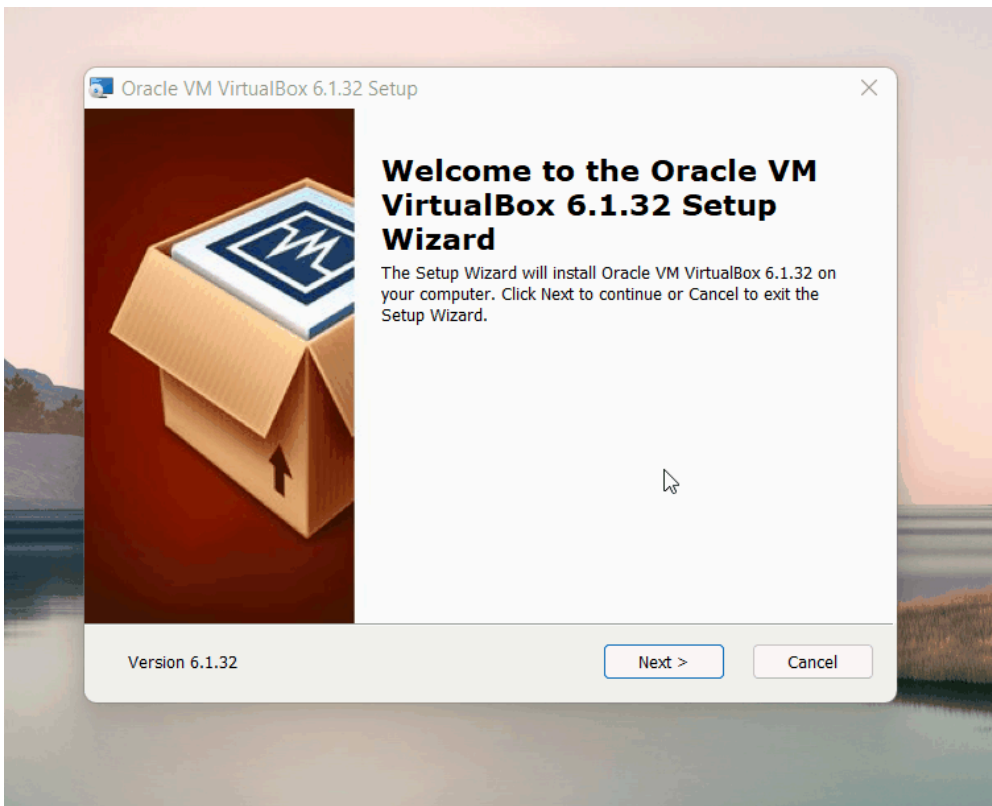
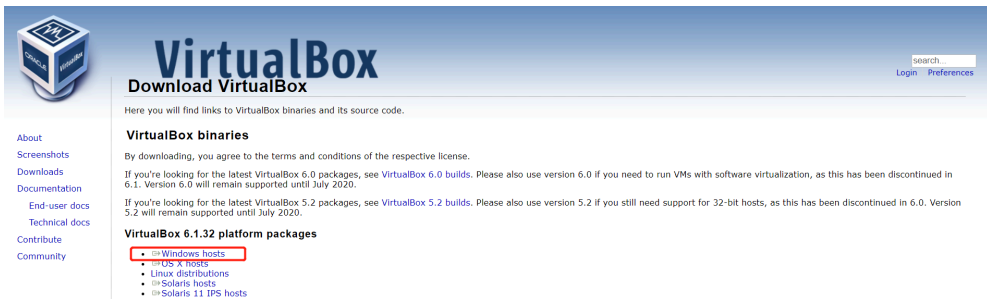
Or go to the [official website](#) to download the virtual machine VM ware

VirtualBox installation package: [Windows hosts](#)

VirtualBox expansion package: [VirtualBox 7.0.10 Oracle VM VirtualBox Extension Pack](#)

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

We chose to download Virtual box because it is free.

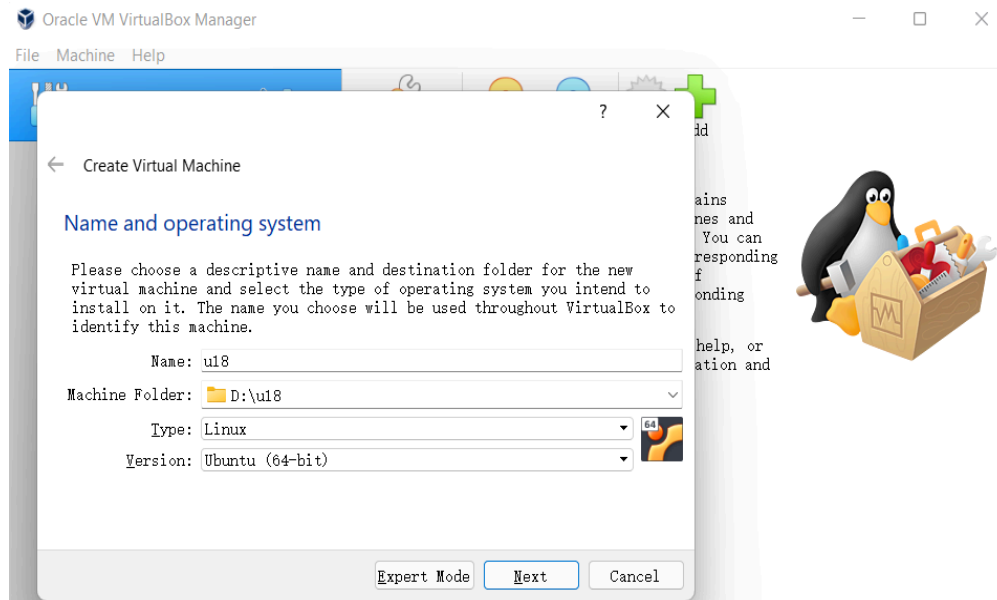


Create a new virtual machine

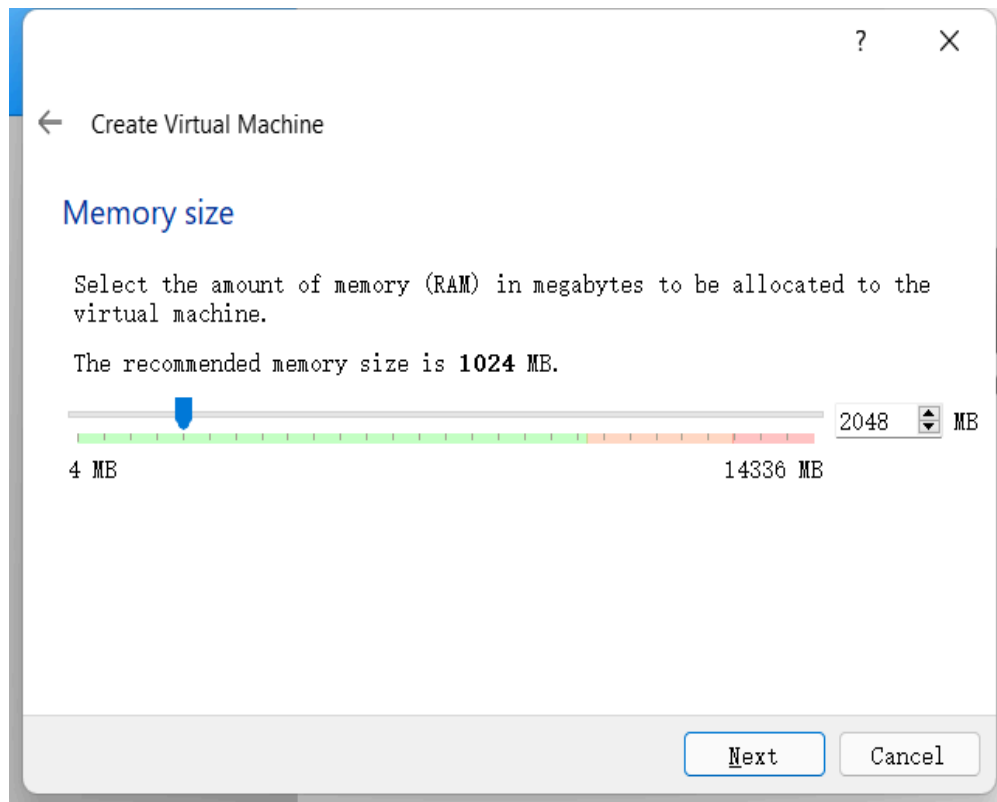
Create a virtual machine

Select New in the control

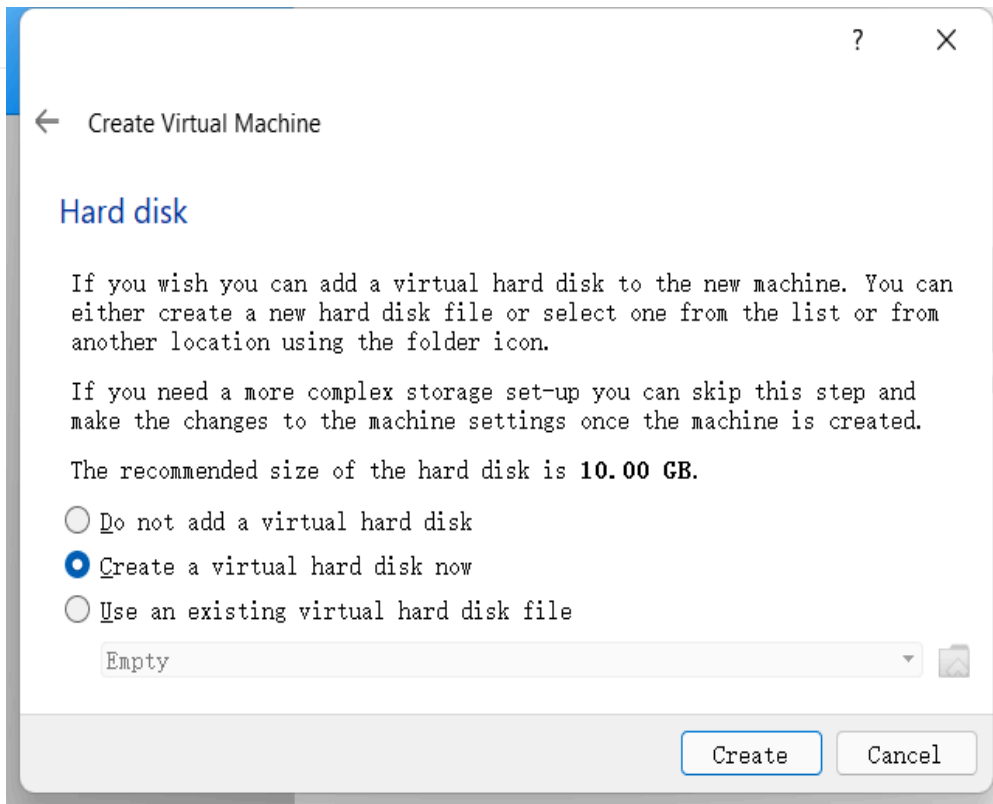
Enter the virtual machine name and the location where the virtual machine is stored, select the virtual machine type as **Linux**, select Ubuntu 64-bit version, and proceed to the next step.



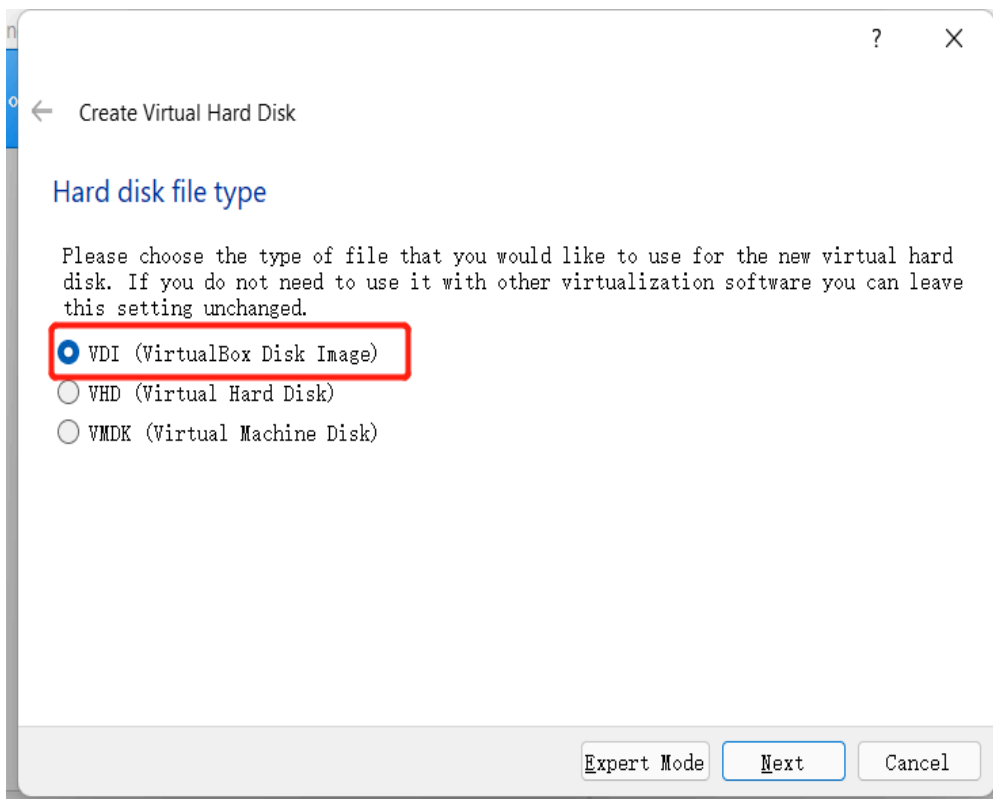
Configure the memory size according to your needs and proceed to the next step.



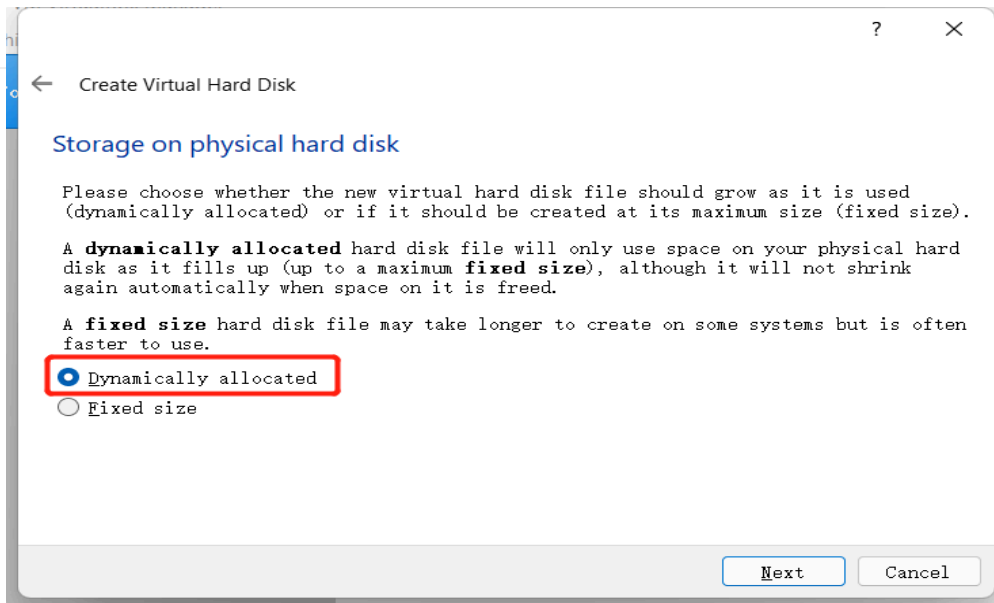
Select **Create a virtual hard disk now** and create it.



Select the **VDI** type for the virtual hard disk type and proceed to the next step.



Allocate the size of the virtual hard disk. Since you need to install the Ubuntu system and will also operate in the system, it is recommended that the size should not be less than 20G.



选择虚拟硬盘的大小。此大小为虚拟硬盘文件在实际硬盘中能用的极限大小。



Import Ubuntu system

Download Ubuntu system.

Please choose the Ubuntu version to install according to your needs

Note: ROS2 needs to download **20.04 version**.

- [16.04 version](#)
- [18.04 version](#)
- [20.04 version](#)

The installation methods and processes of the three versions are the same. Here, the 18.04 version is used as an example for installation

4.1 First-time self-check

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

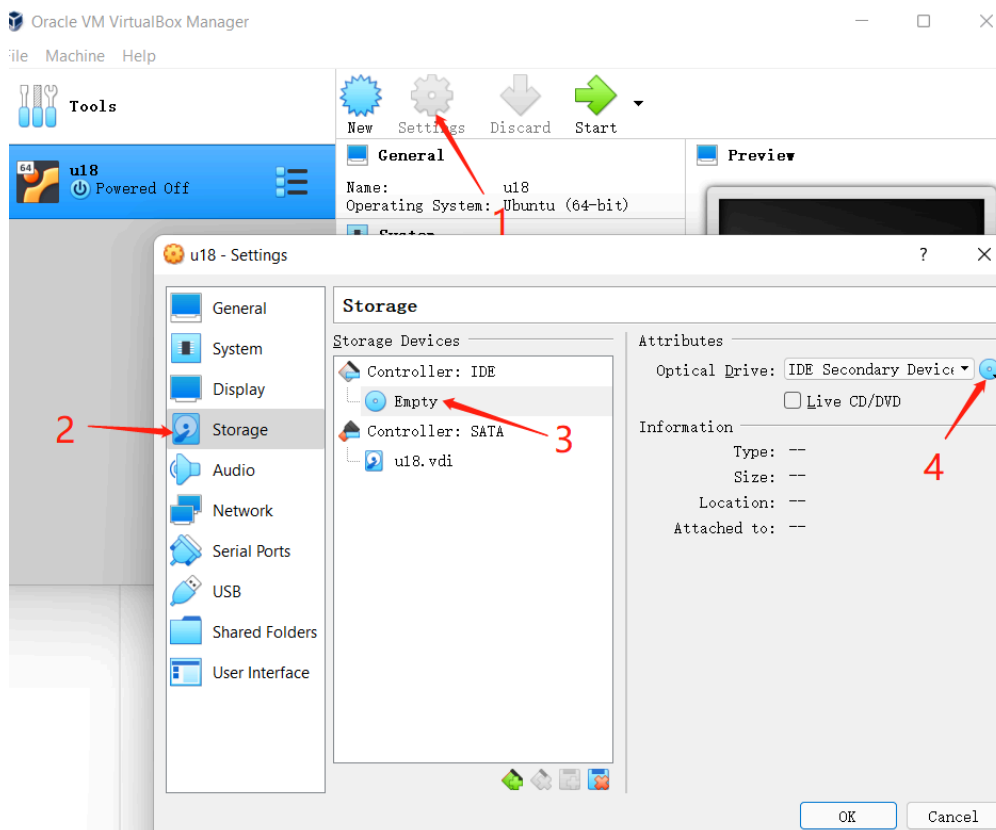
| Name | Last modified | Size | Description |
|--|------------------|------|---|
| Parent Directory | | - | |
| MD5SUMS-metalink | 2020-02-12 13:42 | 296 | |
| MD5SUMS-metalink.gpg | 2020-02-12 13:42 | 916 | |
| SHA256SUMS | 2021-09-16 21:58 | 202 | |
| SHA256SUMS.gpg | 2021-09-16 21:58 | 833 | |
| ubuntu-18.04.6-desktop-amd64.iso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

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

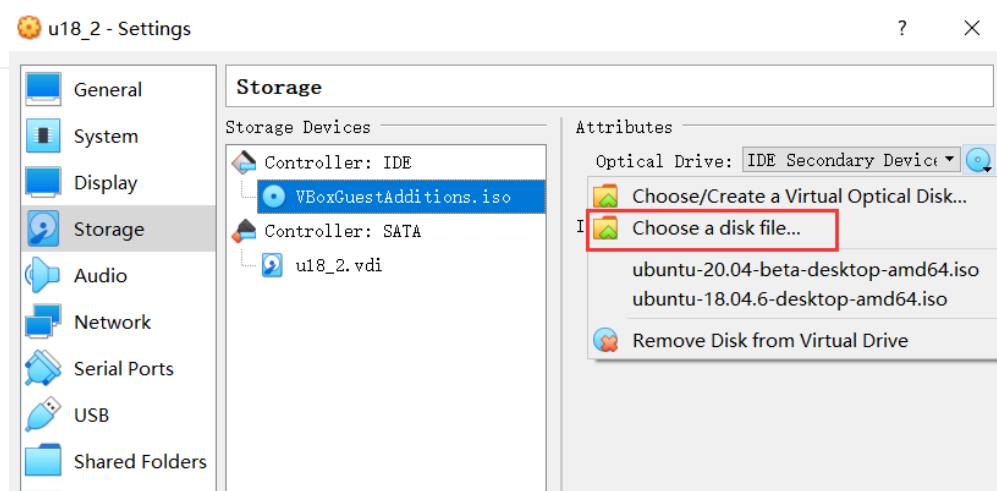
| 名称 | 修改日期 | 类型 | 大小 |
|----------------------------------|-----------------|--------|----------|
| ubuntu-20.04.3-desktop-amd64.iso | 2022/1/12 15:08 | ISO 文件 | 2,999,93 |
| ubuntu-18.04.6-desktop-amd64.iso | 2022/1/12 14:45 | ISO 文件 | 2,455,20 |

Import Ubuntu into the virtual machine

Find the previously installed virtual machine in Virtual Box, enter **Settings**, and assign the CD to the controller in **Storage**:



4.1 First-time self-check



Then open the virtual machine to install Ubuntu and click Start.

2.2.3 Ubuntu installation

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



Click the "Continue" button;



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



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

4.1 First-time self-check



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



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

4.1 First-time self-check



Enter the system installation interface, please wait patiently;



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



ROS environment setup

ROS installation

The basic development environment setup requires the installation of the robot operating system ROS, MoveIt, and git version manager. The following describes their installation methods and processes.

For **myCobot 280-M5** and **myCobot 320-M5** devices, please refer to the installation methods and processes described below. **myCobot 280-PI** and **myCobot 320-PI** devices only need to execute the installation of the **mycobot_ros** installation package.

Version selection

ROS has a one-to-one correspondence with Ubuntu. Different versions of Ubuntu correspond to different versions of ROS. For reference, see the following website: <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

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

If the versions are different, the download will fail. Here we choose Ubuntu 18.04, and the corresponding ROS version is ROS Melodic

NOTE: We currently do not provide any reference for installing ROS on Windows. If necessary, please refer to <https://wiki.ros.org/Installation>

Start installation

1 Add source

There is no ROS software source in the software source list of Ubuntu itself, so you need to **configure the ROS software source to the software list warehouse** before you can download ROS. Open a console terminal (shortcut key **Ctrl+Alt+T**), enter the following command:

- Official source:

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

4.1 First-time self-check

- If the download speed is slow, it is recommended to select a mirror source nearby 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
```

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

2 Set up the secret key

Configure the public network secret key. This step is to let the system confirm that our path is safe, so that there is no problem 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 results are shown below.:

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

3 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 enter the following command:

```
sudo apt-get update
```

Execute **Install ROS**, open a console terminal (shortcut key `Ctrl+Alt+T`), and enter the following command according to your Ubuntu version:

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

It is recommended to install the complete ROS to prevent missing libraries and dependencies.

The installation process takes a long time, please be patient

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

4.1 First-time self-check

```
Retrieved 525 MB in 2min 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`), and enter the following command:

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

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

Ubuntu 16.04:

```
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:

```
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:

4.1 First-time self-check

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

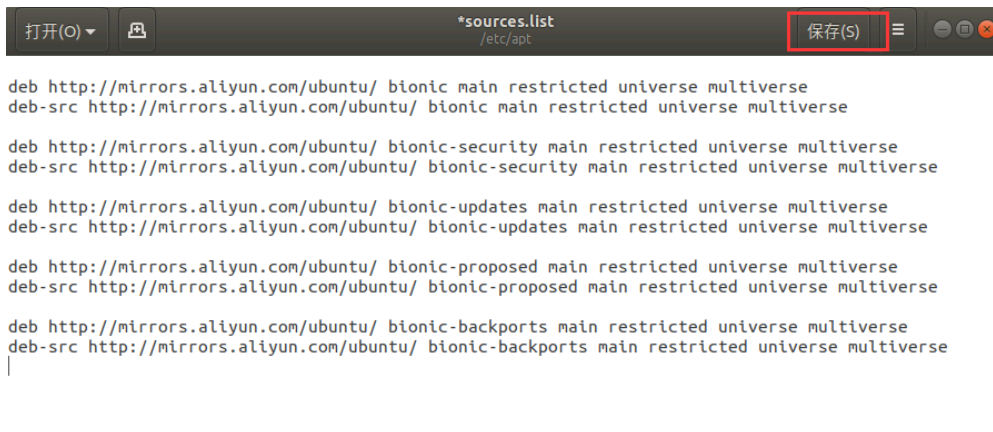
deb http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ focal-security main restricted universe multiverse

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.



```
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
```

- Update the software source list and enter in the console terminal:

```
sudo apt-get update
```

- Enter the command 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, please wait patiently

4 Configure ROS environment to the system

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

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

If your installed Ubuntu system is version 20.04, please use the command `sudo apt install python3-rosdep` to install it, and execute the rosdep initialization command after completion.

```
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.
```

Initialize rosdep:

```
sudo rosdep init
```

If the error message shown below appears:



```
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.github.com
```

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 websites to access:

```
199.232.28.133 raw.githubusercontent.com
151.101.228.133 raw.github.com
```

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

```
sudo rosdep init
```

```
rosdep update
```

After initialization is completed, in order to avoid the need to re-validate the ROS function path every time the terminal window is closed, we can **configure the path to the environment variable**, so that the ROS function path can be automatically validated every time a new terminal is opened. Execute the following commands in the terminal in sequence to open a console terminal (shortcut key `Ctrl+Alt+T`):

1.1.3 Set up the ros environment

Bash

Execute the following command:

```
# Ubuntu 16.04
# 将 ros 环境加入到当前控制台的环境变量
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Install ROS additional dependencies

Enter the following command in the terminal to install ROS additional dependencies 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 Unbutu system is version 20.04, please execute the following command to install it:

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

```
# Ubuntu 16.04
sudo apt install ros-kinetic-joint-state-publisher-gui
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-joint-state-publisher-gui
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-joint-state-publisher-gui
```

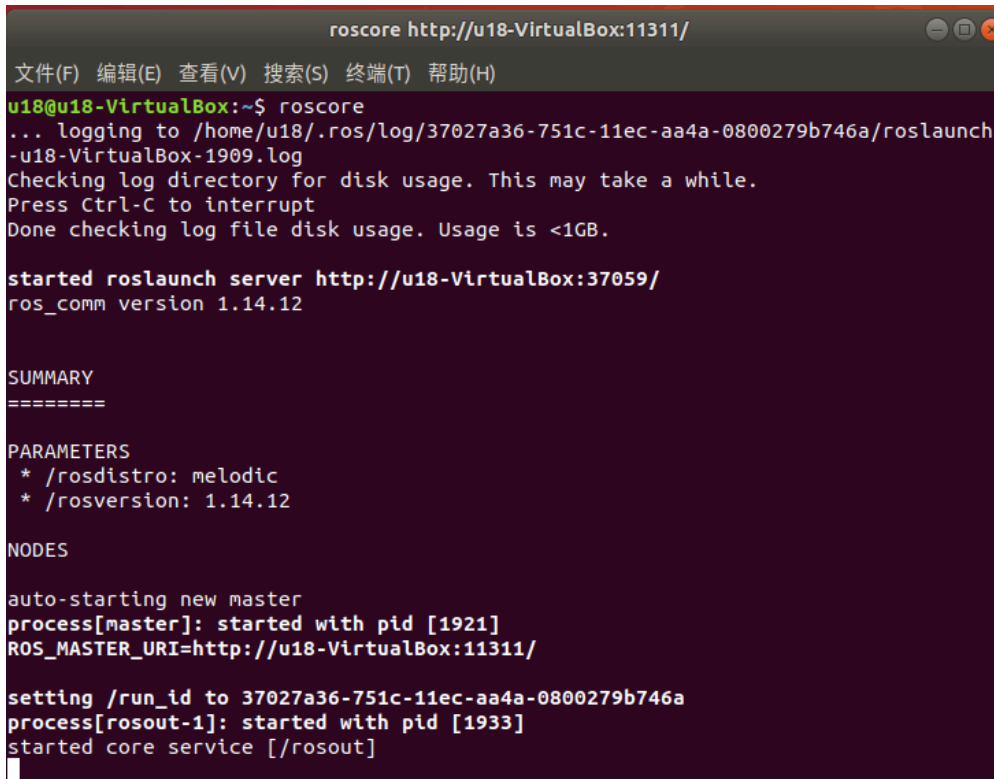
1.1.4 Verify the installation

The startup of the ROS system requires a ROS Master, i.e., a node manager. We can start the ROS Master by entering the `roscore` command in the terminal.

To verify whether ROS is successfully installed, 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 is installed successfully



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

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

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.12

NODES

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

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

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

_For more detailed installation instructions, please refer to the official installation guide at: <http://wiki.ros.org/ROS/Installation>

MoveIt Installation

MoveIt is a functional package of a series of mobile operations in ROS, mainly including motion planning, collision detection, kinematics, 3D perception, operation control and other functions.

Update the software source list

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

```
sudo apt-get update
```

Install MoveIt

Open a console terminal (shortcut key `Ctrl+Alt+T`), enter the following command in the terminal window to execute **MoveIt installation**:

4.1 First-time self-check

```
# 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
```

git installation

Add software source

Add the software source installed by git to the software source list of Ubuntu, open a console terminal (shortcut key `Ctrl+Alt+T`), and enter the following command in the terminal window:

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

Update the software source list

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

```
sudo apt-get update
```

Install git

Open a console terminal (shortcut key `Ctrl+Alt+T`), enter the following command in the terminal window, **execute git installation**:

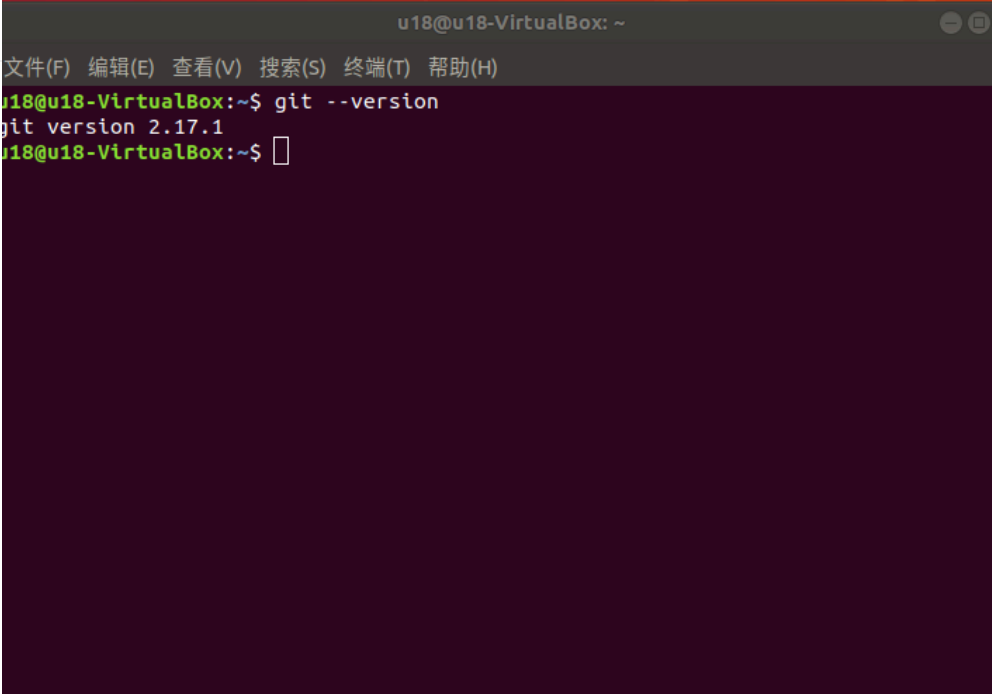
```
sudo apt-get install git
```

Verify installation

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

```
git --version
```

The git version number can be displayed in the terminal, as shown below, indicating a successful installation

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

```
u18@u18-VirtualBox:~$ git --version
git version 2.17.1
u18@u18-VirtualBox:~$
```

Usage

You will need to use git to download the ros package later. For details on how to use git, please refer to the following link:

- <https://git-scm.com/book/zh/v2>
- <https://www.runoob.com/git/git-tutorial.html>

mycobot_ros installation

`mycobot_ros` is a ROS package launched by ElephantRobotics, which is compatible with its desktop six-axis robot arm mycobot series.

项目地址: http://github.com/elephantrobotics/mycobot_ros

Prerequisites

Before installing the package, please ensure that you have a ros workspace.

Here we give a sample command for creating a workspace, the default is `catkin_ws`, open a console terminal (shortcut key `Ctrl+Alt+T`), and enter the following command in the command line:

```
mkdir -p ~/catkin_ws/src # 创建文件夹
cd ~/catkin_ws/src      # 进入文件夹
catkin_init_workspace    # 把当前目录初始化为一个ROS工作空间
cd ..                   # 返回上级目录
catkin_make              # 构建工作区中的代码。
```

Add workspace environment

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
```

Installation

NOTE:

- This package depends on ROS and MoveIT. Please make sure that ROS and MoveIT are installed successfully before use.
- The interaction between this package and the real robot arm depends on PythonApi - `pymycobot`
- The Api project is: <https://github.com/elephantrobotics/pymycobot>
- Quick installation: `pip install pymycobot --upgrade`

When executing the `pip install pymycobot --upgrade` 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 according to the prompt

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

- If your Ubuntu system is version 20.04, please execute the command `sudo apt install python3-pip` to install pip
After pip is installed, execute it again in the terminal

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

- The installation method depends on Git, please make sure Git is installed on your computer.

The official default ROS1 workspace is `catkin_ws` .

4.1 First-time self-check

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

Import system image for M5 version/How to use ros for Pro600:

Note: In order to reduce the difficulty of environment construction, we will provide Linux system image (Ubuntu 20.04), Virtual Box installation package and its extension package. Next, we will teach you how to install Virtual Box and import the Linux system image (the default password is 123). **Built-in environment:** ROS1 + Moveit + Git + pymycobot + mycobot_ros

Install the virtual machine

Go to the [official website](#) to download the virtual machine Virtual Box

Or go to the [official website](#) to download the virtual machine VM ware

VirtualBox installation package: [Windows hosts](#)

VirtualBox expansion package: [VirtualBox 7.0.10 Oracle VM VirtualBox Extension Pack](#)

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

We chose to download Virtual box because it is free.



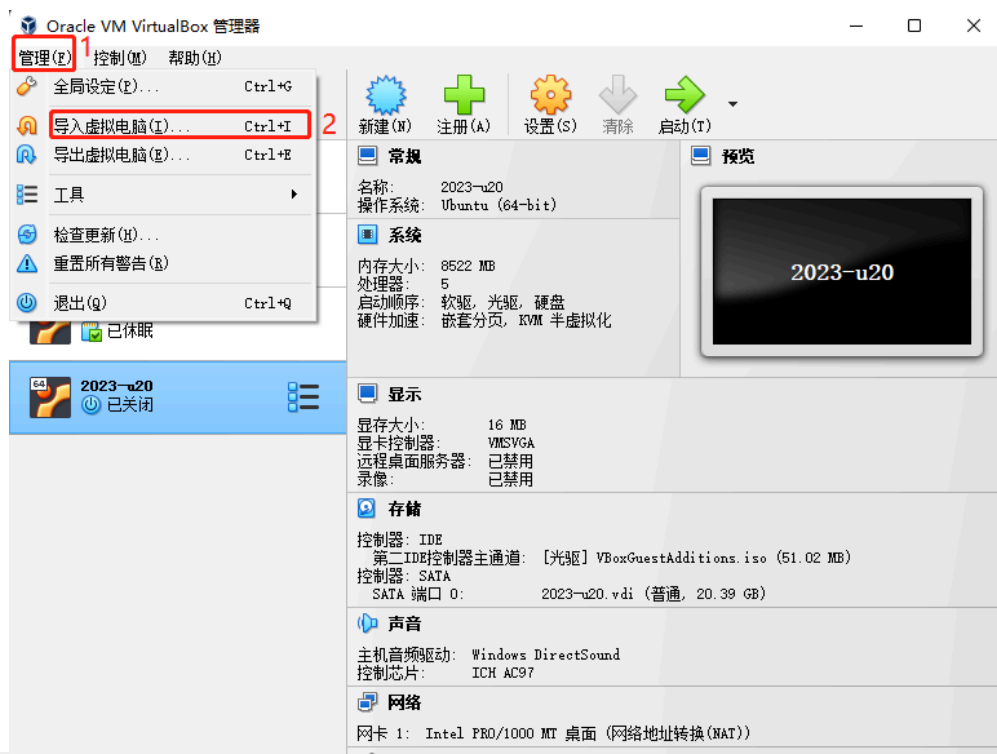


Download Linux system image

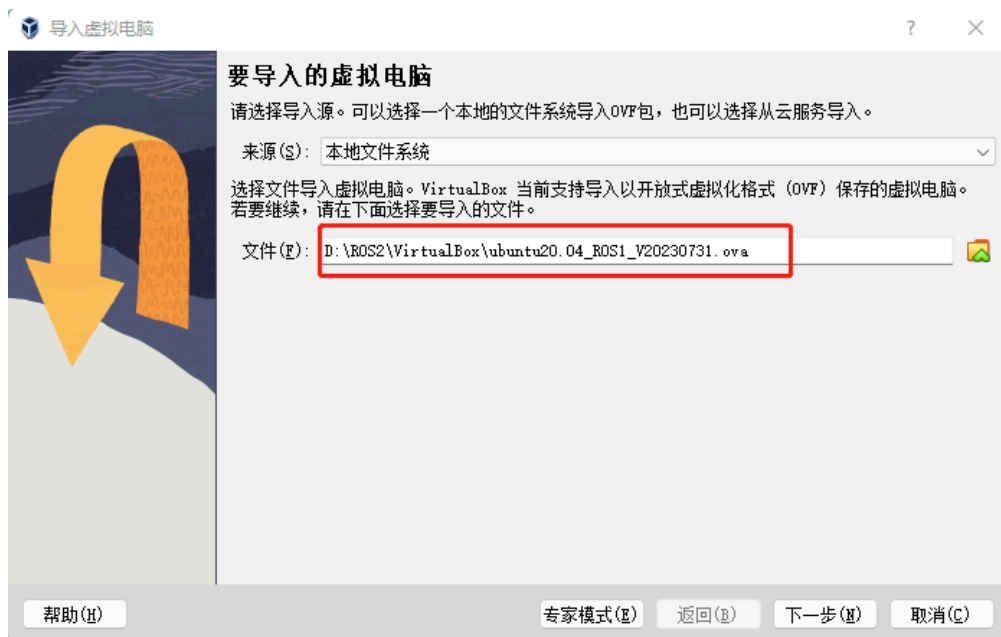
Click to download: [Linux ubuntu20.04](#)

Import Linux system image

Click Manage in the Virtual Box interface -> Import virtual machine -> Select virtual image -> Select installation path and import, and install as follows.



4.1 First-time self-check



4.1 First-time self-check



Wait for the image to be imported. The installation is successful as shown below.



Then start the system. The default password is **123**

The ROS1 environment is now complete. For the use of ROS1, please refer to Introduction and Use of Rviz chapter.

Service and Topic

Service

Note: myCobot Pro 600 and myCobot Pro 630 devices do not support Service usage.

Service is a request-response communication mechanism. One node can provide a service, and another node can request the service and wait for a response. Services are usually used to perform time-consuming operations or tasks that require interaction, such as performing calculations and obtaining data.

Related commands and instructions:

| Command | Detailed description |
|--|--|
| <code>rosservice list</code> | Display active service information |
| <code>rosservice info [service name]</code> | Display information of a specified service |
| <code>rosservice type [service name]</code> | Display service type |
| <code>rosservice find [service type]</code> | Find services of a specified service type |
| <code>rosservice uri [service name]</code> | Display ROSRPC URI services |
| <code>rosservice args [service name]</code> | Display service parameters |
| <code>rosservice call [service name] [parameters]</code> | Request services with input parameters |

Topic

Topic is a communication mechanism in publish-subscribe mode. Nodes can publish messages to a topic or subscribe to a topic to receive messages. Multiple nodes can publish and subscribe to the same topic at the same time to broadcast and receive information. Topic is mainly used for data transmission with low real-time requirements, such as sensor data, status information, etc.

Related commands and instructions:

| Command | Detailed description |
|---|--|
| rostopic list | Display the active topic directory |
| rostopic echo [topic name] | Display the message content of the specified topic in real time |
| rostopic find [type name] | Display topics using the specified type of message |
| rostopic type [topic name] | Display the message type of the specified topic |
| rostopic bw [topic name] | Display the message bandwidth of the specified topic |
| rostopic hz [topic name] | Display the message data publishing cycle of the specified topic |
| rostopic info [topic name] | Display the information of the specified topic |
| rostopic pub [topic name] [message type] [parameters] | Publish a message with the specified topic name |

Difference between service and topic:

| | topic | service |
|----------------------|-------------------|--------------------------|
| Synchronicity | Asynchronous | Synchronous |
| Communication model | Publish/Subscribe | Request/Response |
| Underlying protocol | ROSTCP/ROSUDP | ROS service protocol RPC |
| Feedback mechanism | No | Yes |
| Buffer | Yes | No |
| Real-time | Weak | Strong |
| Node relationship | Many-to-many | One-to-many |
| Applicable scenarios | Data transmission | Logical processing |

You can go to [service](#) and [topic](#) to learn more about the use of these two functions

Introduction to msg and srv

- msg: msg files are simple text files that describe the fields of ROS messages. They are used to generate source code for messages in different languages (c++ or python, etc.).
- srv: srv files are used to describe services. It consists of two parts: request and response. msg files are stored in the msg directory of the package, while srv files are stored in the srv directory.

rosmmsg

rosmmsg is a command-line tool for displaying information about ROS message types.

4.1 First-time self-check

rosmg demo:

```
rosmg show # Display message description
rosmg info # Display message information
rosmg list # List all messages
rosmg md5 # Display md5 encrypted messages
rosmg package # Display all messages under a certain function package
rosmg packages # List function packages containing messages
```

- rosmg list Will list all msg in the current ROS
- rosmg packages List all packages containing messages
- rosmg package List all msg under a certain package

```
//rosmg package # Package name
rosmg package turtlesim
```

- rosmg show Display message description

```
//rosmg show # Message name
rosmg show turtlesim/Pose
# Result:
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

- rosmg info Same as rosmg show
- rosmg md5 A verification algorithm to ensure the consistency of data transmission

rossrv

rossrv is a command line tool used to display information about ROS service types, and its syntax is highly similar to rosmg.

```
rossrv show # Display service message details
rossrv info # Display service message related information
rossrv list # List all service information
rossrv md5 # Display md5 encrypted service message
rossrv package # Display all service messages under a certain package
rossrv packages # Display all packages containing service messages
```

- rossrv list Will list all srv messages in the current ROS
- rossrv packages List all packages containing service messages
- rossrv package List all msg under a certain package

4.1 First-time self-check

```
//rossrv package # Package name  
rossrv package turtlesim
```

- `rossrv show` Display message description

```
//rossrv show # Message name  
rossrv show turtlesim/Spawn  
# Result:  
float32 x  
float32 y  
float32 theta  
string name  
---  
string name
```

- `rossrv info` The same function as `rossrv show`
- `rossrv md5` Use md5 verification (encryption) for service data

Introduction to URDF

- Unified Robot Description Format, abbreviated as URDF. The `urdf` function package in ROS contains a C++ parser for URDF. The URDF file uses XML format to describe the robot model.
- URDF cannot be used alone and needs to be combined with Rviz or Gazebo. URDF is just a file that needs to be rendered into a graphical robot model in Rviz or Gazebo.

urdf file description

Code example:

Here only part of the code is intercepted for display:

4.1 First-time self-check

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="mycobot_ai_with" >

  <xacro:property name="width" value=".2" />

  <link name="env">
    <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="10"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
    </inertial>
    <visual>
      <geometry>
        <!-- 0.0 0 -0.04 1.5708 3.14159-->
        <mesh filename="package://mycobot_description/urdf/mycobot/suit_env.dae"/>
      </geometry>
      <origin xyz = "0 0 0.0" rpy = "1.5708 0 -1.5708"/>
    </visual>
  </link>

  <link name="joint1">
    <inertial>
      <origin xyz="0 0 0.1" rpy="0 0 0"/>
      <mass value="0.2"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
    </inertial>
    <visual>
      <geometry>
        <!-- 0.0 0 -0.04 1.5708 3.14159-->
        <mesh filename="package://mycobot_description/urdf/mycobot/joint1.dae"/>
      </geometry>
      <origin xyz = "0.0 0 0.02 " rpy = " 0 0 -1.5708"/>
    </visual>
    <collision>
      <geometry>
        <!-- 0.0 0 -0.04 1.5708 3.14159-->
        <mesh filename="package://mycobot_description/urdf/mycobot/joint1.dae"/>
      </geometry>
      <origin xyz = "0.0 0 0.02 " rpy = " 0 0 -1.5708"/>
    </collision>
  </link>

  <joint name="vision_joint" type="fixed">
    <axis xyz="0 0 0"/>
```

4.1 First-time self-check

```
<limit effort = "1000.0" lower = "-3.14" upper = "3.14159" velocity = "0"/>
<parent link="env"/>
<child link="joint1"/>
<origin xyz= "0 0 0" rpy = "0 0 0"/>
</joint>

<link name="world"/>
<joint name="fixed" type="fixed">
  <parent link="world"/>
  <child link="env"/>
</joint>

</robot>
```

It can be seen that the urdf file is not complicated, mainly composed of the two parts of `link` and `joint` that are repeated continuously.

Link section

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

Attributes

name:

Name used to describe the link itself

Elements

- `<inertial>` (optional)
- Inertial properties of the link
- `<origin>` (optional, defaults to identity if not specified)
- Defines the reference coordinates of the inertial reference frame relative to the link coordinate system. The coordinates must be defined at the center of gravity of the link, and the coordinate axes may not be parallel to the principal axes of inertia.
- `xyz` (optional, defaults to zero vector) Represents the offsets in the x, y, z directions, in meters.
- `rpy` (optional: defaults to identity if not specified) Represents the rotation of the coordinate axis in the RPY direction, in radians.
- `<mass>` Mass properties of the link
- `<inertia>` 3×3 rotational inertia matrix, consisting of six independent quantities: $ixx, ixy, ixz, iyy, iyz, izz$.
- `<visual>` (optional)
- Visual properties of the link. Used to specify the shape of the link display (rectangle, cylinder, etc.). The same link can have multiple visual elements, and the shape of the link is formed by two of the multiple elements. In general, if the model is more complex, it can be drawn by solidworks and then generate stl calls. Simple shapes such as adding end effectors can be written directly. At the same time, the position of the geometric shape can be adjusted here according to the gap between the theoretical model and the actual model.
- `<name1>` (optional) The name of the connecting rod geometry.
- `<origin>` (optional, defaults to identity if not specified)

4.1 First-time self-check

- The coordinate system of the geometric shape relative to the coordinate system of the connecting rod.
- xyz (optional: defaults to zero vector) Indicates the offset in the x, y, z x,y,zx,y,z direction, in meters.
- rpy (optional: defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
- `<geometry>` (required)
- The shape of the visual object, which can be one of the following:
 - `<box>` Rectangle, elements include length, width, and height. The origin is at the center.
 - `<cylinder>` Cylinder, elements contain radius, length. Origin center.
 - `<sphere>` Sphere, elements contain radius. Origin at center.
 - `<mesh>` Mesh, determined by file, and scale is provided to define its boundaries. Collada .dae files are recommended, .stl files are also supported, but must be a local file.
 - `<material>` (optional)
 - Material of the visual component. Can be defined outside the link tag, but must be in the robot tag. When defined outside the link tag, the link name must be referenced.
 - `<color>` (optional) Color, composed of red/green/blue/alpha, size range [0,1].
 - `<texture>` (optional) Material properties, defined by file.
 - `<collision>` (optional)
 - Collision properties of the link. Collision properties are different from the visual properties of the link, and a simple collision model is often used to simplify calculations. The same link can have multiple collision attribute tags. The collision attribute representation of the link is composed of the set of geometric shapes it defines.
 - `<name>` (optional) Specifies the name of the link geometry
 - `<origin>` (optional, defaults to identity if not specified)
 - The reference coordinate system of the collision component relative to the reference coordinate system of the link coordinate system.
 - xyz (optional, defaults to zero vector) Represents the offset in the x, y, z x,y,zx,y,z direction, 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 the geometry element description above

For detailed elements and the functions of each element, please go to the [official document](#) for viewing

joint section

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

Properties of joint:

name:

Specifies a unique name for the joint

type:

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

- revolute - A hinge joint that rotates along an axis with a range specified by upper and lower limits.
- continuous - A continuous hinge joint that rotates around an axis with no upper and lower limits.
- prismatic - A sliding joint that slides along an axis with a range specified by upper and lower limits.
- fixed - This is not a true joint as it cannot move. All degrees of freedom are locked. This type of joint does not require axis, calibration, dynamics, limits or safety_controller.
- floating - This joint allows motion in all 6 degrees of freedom.

- planar - This joint allows motion in a plane perpendicular to the axis.

Elements of joint

- `<origin>` (optional, defaults to identity if not specified) Transformation from parent link to child link, joint is located at the origin of child link. Modifying this parameter can adjust the position of the link, which can be used to adjust the error between the actual model and the theoretical model, but it is not recommended to modify it drastically, because this parameter affects the position of the link stl, which is easy to affect the collision detection effect.
- xyz (optional: defaults to zero vector) Represents the offset in the x, y, z x, y, z axis direction, in meters.
- rpy (optional: defaults to zero vector) Represents the angle of rotation around a fixed axis: roll around the x axis, pitch around the y axis, yaw around the z axis, expressed in radians.
- `<parent>` (required)
- The name of the parent link is a mandatory attribute.
- link The name of the parent link is the name of this link in the robot structure tree.
- `<child>` (required)
- The name of the child link is a mandatory attribute.
- link The name of the child link, which is the name of this link in the robot structure tree.
- `<axis>` (optional: defaults to (1,0,0))
- The axis of the joint in the joint coordinate system. This is the axis of rotation (revolute joint), the axis along which the prismatic joint moves, and the standard plane of the planar joint. This axis is specified in the joint coordinate system. Modifying this parameter can adjust the axis around which the joint rotates. It is often used to adjust the direction of rotation. If the model's rotation direction is opposite to the actual direction, just multiply by -1. Fixed and floating joints do not need this element.
- xyz (required) The x, y, z components of the axial vector, as a normalized vector.
- `<calibration>` (optional)
- The reference point of the joint, used to calibrate the absolute position of the joint.
- rising (optional) The reference point triggers a rising edge when the joint moves in the positive direction.
- falling (optional)

The reference point triggers a falling edge when the joint moves in the forward direction.

- `<dynamics>` (optional)
- This element is used to specify the physical properties of the joint. Its value is used to describe the modeling properties of the joint, especially during simulation.

`<limit>` (required when the joint is a revolute or translation joint)

- This element is the kinematic constraint of the joint.
- lower (optional, defaults to 0)

Attribute that specifies the lower limit of the joint's range of motion (in radians for revolute joints and meters for prismatic joints). This attribute is ignored for continuous joints.

- upper (optional, defaults to 0)

4.1 First-time self-check

Attribute that specifies the upper limit of the joint's range of motion (in radians for revolute joints and meters for prismatic joints). This attribute is ignored for continuous joints.

- effort (required)

Attribute that specifies the maximum force with which the joint is applied.

- velocity (required)

Attribute that specifies the maximum velocity with which the joint is applied.

`<mimic>` (optional)

- This tag is used to specify a defined joint to mimic an existing joint. The value of this joint can be calculated using the following formula:

```
value = multiplier * other_joint_value + offset
```

- joint(required) The name of the joint to be mimicked.
- multiplier(optional) Specifies the multiplier factor in the above formula.
- offset(optional) Specifies the offset term in the above formula. The default value is 0

`<safety_controller>` (optional)

- This element is a safety control limit. The data under this element will be read into `move_group`, but it is actually invalid. `move_group` will skip this limit and directly read the parameter content under `limit`. At the same time, setting this element may cause planning failure.
- `soft_lower_limit` (optional, default is 0) This attribute specifies the lower limit of the joint safety control boundary, which is the starting limit of the joint safety control. This value needs to be greater than the lower value in the limit above.
- `soft_upper_limit` (optional, default is 0) This attribute specifies the upper limit of the joint safety control boundary, which is the starting limit of the joint safety control. This value needs to be less than the upper value in the limit above.
- `k_position` (optional, default is 0) This attribute is used to describe the relationship between position and velocity.
- `k_velocity` (required) This attribute is used to describe the relationship between force and velocity.

For detailed elements and their functions, please visit <http://wiki.ros.org/urdf/XML/joint> for more information.

A brief introduction and use of rviz

rviz is a 3D visualization platform in ROS. On the one hand, it can realize the graphical display of external information. On the other hand, it can also release control information to objects through rviz, thereby realizing the monitoring and control of robots.

Introduction to the installation and interface of rviz

When installing ros, if you perform a complete installation, rviz has been installed, and you can try to run it directly; if it is not fully installed, you can install rviz separately:

```
# Ubuntu20.04
sudo apt-get install ros-noetic-rviz
```

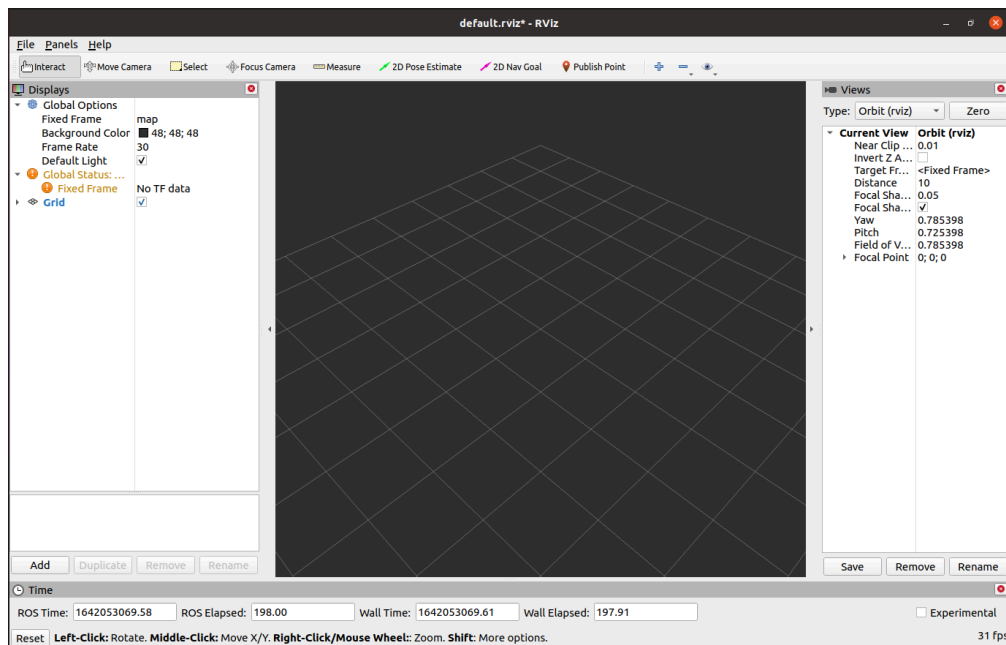
After the installation is complete, please open a new terminal (shortcut key `Ctrl+Alt+T`) and enter the following command:

```
roscore
```

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

```
roslaunch rviz rviz
# or
rviz
```

Open rviz and the following interface will be displayed:



Introduction to each area

- On the left is the display list. A display is something that draws something in the 3D world and may have some options available in the display list.

4.1 First-time self-check

- On the top is the toolbar, which allows users to select multiple functions with various function keys
- In the middle is the 3D view: it is the main screen that allows users to view various data in three dimensions. The background color, fixed frame, grid, 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.
- On the right is the observation angle setting area, which can set different observation angles.

In this section, we will only give a rough introduction. If you want to know more details, you can go to the [User Guide](#) to check it out.

mycobot_ros installation and update

`mycobot_ros` is a ROS package launched by ElephantRobotics that is compatible with various types of desktop robotic arms.

Project address: https://github.com/elephantrobotics/mycobot_ros

The official default workspace is `catkin_ws`.

enter the following command:

```
cd ~/catkin_ws/src # Enter the src folder in the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd .. # Return to the workspace
catkin_make # Build the code in the workspace
source devel/setup.bash # Add environment variables
```

Note: If the `mycobot_ros` folder already exists in the `/home/er/catkin_ws/src` (equivalent to `~/catkin_ws/src`) directory, you need to delete the original `mycobot_ros` first, and then execute the above command. Among them, `er` in the directory path is the user name of the virtual machine. If it is inconsistent, please modify it.

Simple use

Start through the launch file

This example is based on the premise that you have completed [Environment Construction](#) and successfully copied the company's code from GitHub.

Open a console terminal (shortcut key `Ctrl+Alt+T`) Enter the following command to **ROS environment configuration**.

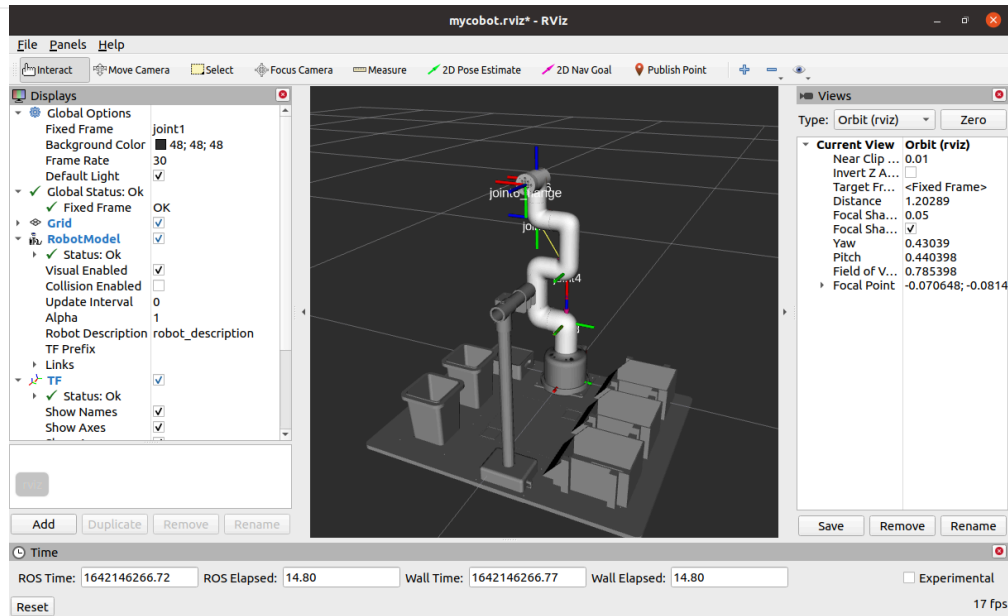
```
cd ~/catkin_ws/
source devel/setup.bash
```

Enter again:

```
roslaunch mycobot_280arduino test.launch
```

4.1 First-time self-check

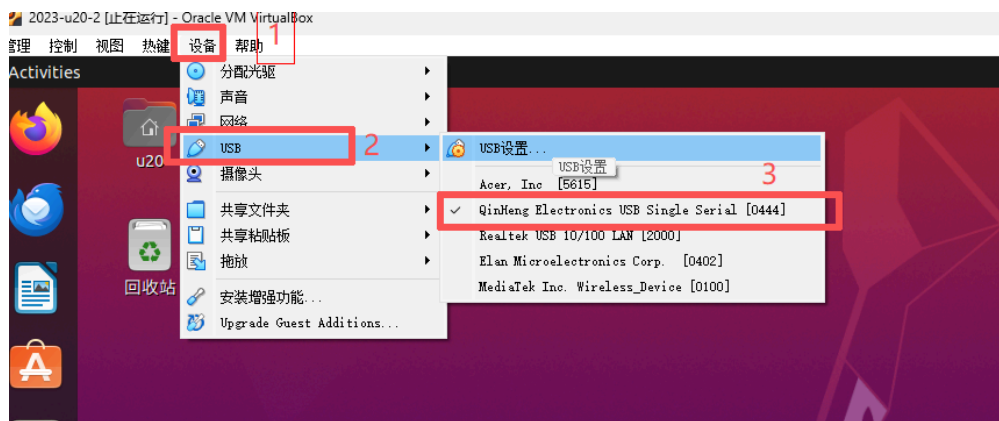
Open rviz and get the following result:



If you want to learn more about rviz, you can go to the [official document](#) to view it

M5/Arduino version prerequisites

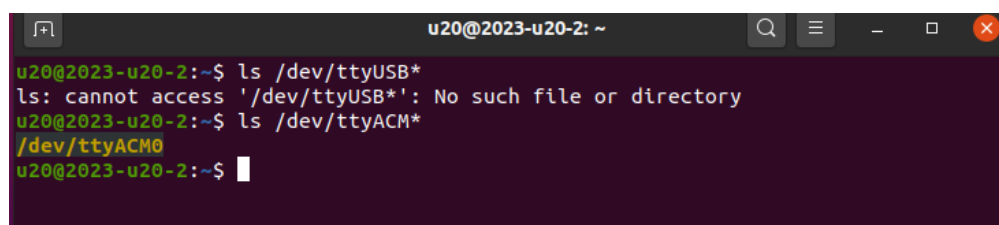
- Device connection, select **Device** —> **USB** —> **Check the device serial port name**



- Open the console terminal (shortcut key **Ctrl+Alt+T**), open the terminal window to view the device name:

```
# View the device name of the robot
ls /dev/ttyUSB* # Old version myCobot280 M5

# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # New version myCobot280 M5
```



4.1 First-time self-check

- Grant serial port permissions to the robot:

```
# The default device name is /dev/ttyUSB0. If the device name is not the default value, it needs to be modified.
sudo chmod 777 /dev/ttyUSB0 # Old version myCobot280 M5

sudo chmod 777 /dev/ttyACM0 # New version myCobot280 M5
```

Then enter the user password (**Note:** The password will not be displayed, just enter it correctly).

280 series rviz user guide

Robot arm control

Note: Before using the following example, you need to configure the ROS package environment after opening the terminal command line:

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

Then proceed with the example.

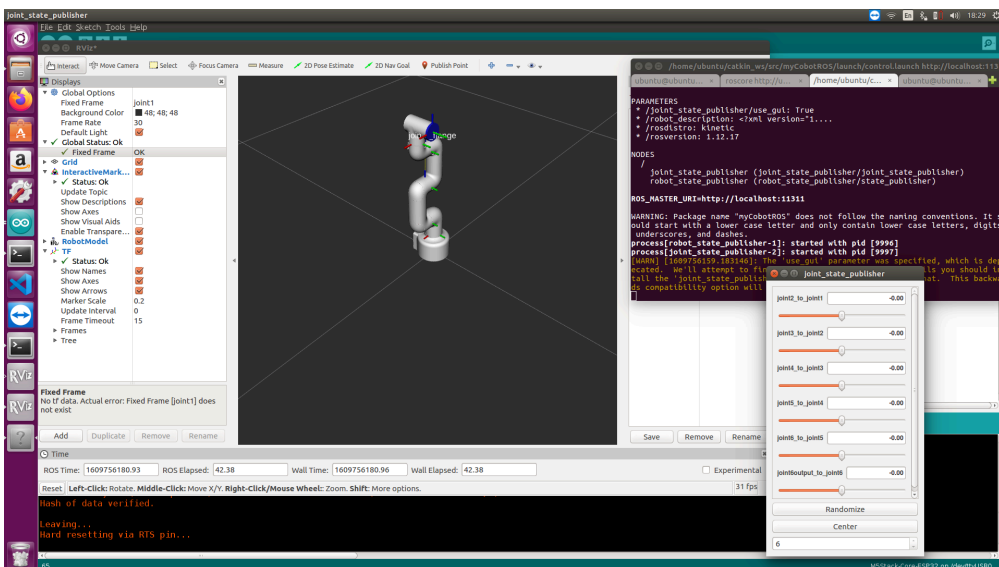
1 Slider control

Open a command line and run:

- mycobot 280-Arduino version:

```
# mycobot 280-Arduino version default serial port name is "/dev/ttyACM0", baud rate is 1000000.
roslaunch mycobot_280arduino slider_control.launch port:=/dev/ttyACM0 baud:=1000000
```

It will open rviz and a slider component, you will see the following screen:



4.1 First-time self-check

Then you can **control the movement of the model in rviz by dragging the slider**. If you want the real mycobot to move with you, you need to open another command line and run:

- mycobot 280-Arduino version:

```
# mycobot 280-Arduino version default serial port name is "/dev/ttyACM0", baud rate is 1000000.  
roslaunch mycobot_280arduino slider_control.py _port:=/dev/ttyACM0 _baud:=1000000
```

Please note: due to the commandThe robot arm will move to the current position of the model while inputting. Before you use the command, please make sure that the model in rviz does not have any penetration Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm

2 Model following

In addition to the above controls, we can also **let the model follow the real robot arm movement**. Open a command line and run:

- mycobot 280-Arduino version:

```
# mycobot 280-Arduino version default serial port name is "/dev/ttyACM0", baud rate is 1000000.  
roslaunch mycobot_280arduino follow_display.py _port:=/dev/ttyACM0 _baud:=1000000
```

Then open another command line and run:

- mycobot 280-Arduino version:

```
roslaunch mycobot_280arduino mycobot_follow.launch
```

It will **open rviz to show the model following effect**.

3 GUI control

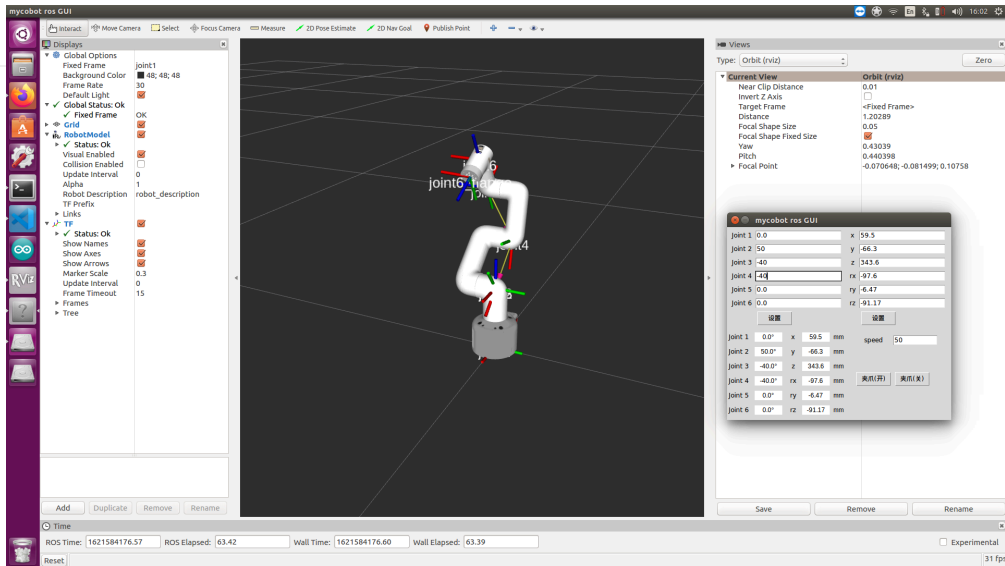
Based on the previous, this package also **provides a simple Gui control interface**. This method is intended for real robotic arms to interact with each other, please connect mycobot.

Open the command line:

- mycobot 280-Arduino version:

```
# mycobot 280-Arduino version default serial port name is "/dev/ttyACM0", baud rate is 1000000.  
roslaunch mycobot_280arduino simple_gui.launch port:=/dev/ttyACM0 baud:=1000000
```

4.1 First-time self-check



4 Keyboard control

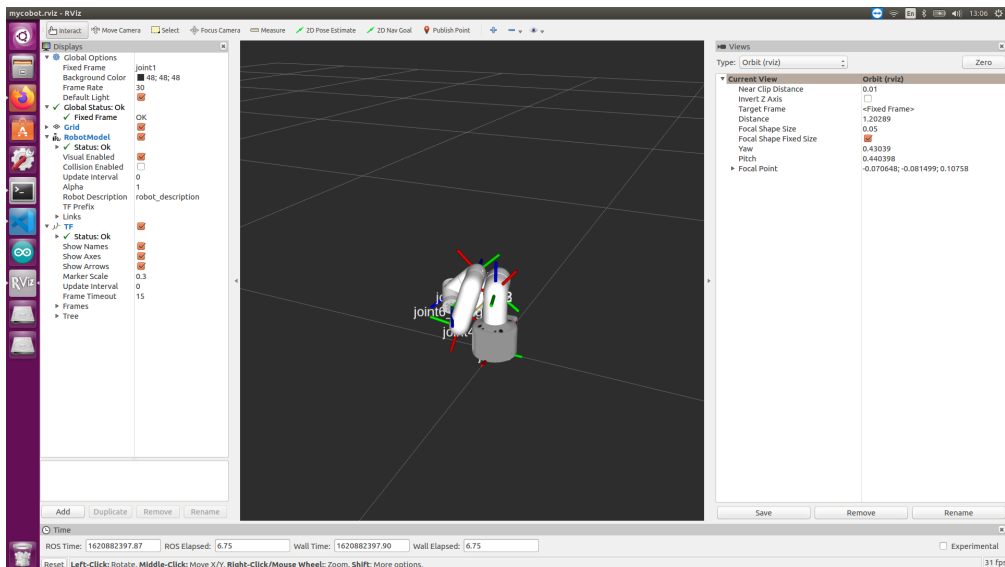
Added the keyboard control function in the `mycobot_280arduino` package, and synchronized it in real time in rviz. This function depends on `pythonApi`, so make sure it is connected to the real robot arm.

Open a command line and run:

- mycobot 280-Arduino version:

```
# mycobot 280-Arduino version default serial port name is "/dev/ttyACM0", baud rate is 1000000.  
roslaunch mycobot_280arduino teleop_keyboard.launch port:=/dev/ttyACM0 baud:=1000000
```

The running effect is as follows:



The command line will output mycobot information as follows:

4.1 First-time self-check

SUMMARY

=====

PARAMETERS

```
* /mycobot_services/ baud: 115200
* /mycobot_services/ port: /dev/ttyACM0
* /robot_description: <?xml version="1....
* /roscdistro: noetic
* /rosversion: 1.16.0
```

NODES

```
/
  mycobot_services (mycobot_communication/mycobot_topics.py)
  real_listener (mycobot_280arduino/listen_real_of_topic.py)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)
  rviz (rviz/rviz)
```

auto-starting new master

process[master]: started with pid [217714]

ROS_MASTER_URI=http://localhost:11311

setting /run_id to 01e863c8-5642-11f0-a2da-335dad9016e7

process[rosout-1]: started with pid [217728]

started core service [/rosout]

process[robot_state_publisher-2]: started with pid [217731]

process[rviz-3]: started with pid [217733]

process[mycobot_services-4]: started with pid [217738]

process[real_listener-5]: started with pid [217740]

current pymycobot library version: 3.9.9

pymycobot library version meets the requirements!

ls: cannot access '/dev/ttyUSB*': No such file or directory

[INFO] [1751350196.024298]: /dev/ttyACM0,115200

MyCobot Status

Joint Limit:

joint 1: -168 ~ +168

joint 2: -135 ~ +135

joint 3: -150 ~ +150

joint 4: -145 ~ +145

joint 5: -165 ~ +165

joint 6: -180 ~ +180

Connect Status: True

Servo Infomation: all connected

Servo Temperature: unknown

4.1 First-time self-check

```
Atom Version: 7.2
```

Next, open another command line and run:

- mycobot 280-Arduino version:

```
roslaunch mycobot_280arduino teleop_keyboard.py
```

You will see the following output in the command line:

```
[INFO] [1751342043.889285]: Waiting to receive current coordinates...
Mycobot Teleop Keyboard Controller (ROS1 - Topic Version)
-----
Movement (Cartesian):
      w (x+)
a (y-)  s (x-)  d (y+)
      z (z-)  x (z+)

Rotation (Euler angles):
      u (rx+)  i (ry+)  o (rz+)
      j (rx-)  k (ry-)  l (rz-)

Movement Step:
+ : Increase movement step size
- : Decrease movement step size

Gripper:
g - open  h - close

Pump:
b - open  m - close

Other:
1 - Go to init pose
2 - Go to home pose
3 - Save current pose as home
q - Quit

currently:  speed: 50  change percent: 5
[INFO] [1751342044.199244]: Current moving step: position 12.5 mm, angle attitude 9.0°
```

In this terminal, you can control the state of the robot and move the robot through the keys in the command line.

Parameters supported by this script:

- `_speed`: robot movement speed.
- `_change_percent`: movement distance percentage.

5 End effector

- **Supported end effectors:** myCobot Adaptive Gripper, myCobot Vertical Suction Pump V2.0, Camera Flange
- **Applicable devices:** myCobot 280 M5, myCobot 280 PI

Note: myCobot Adaptive Gripper only supports myCobot 280 M5 devices

5.1 myCobot Adaptive Gripper

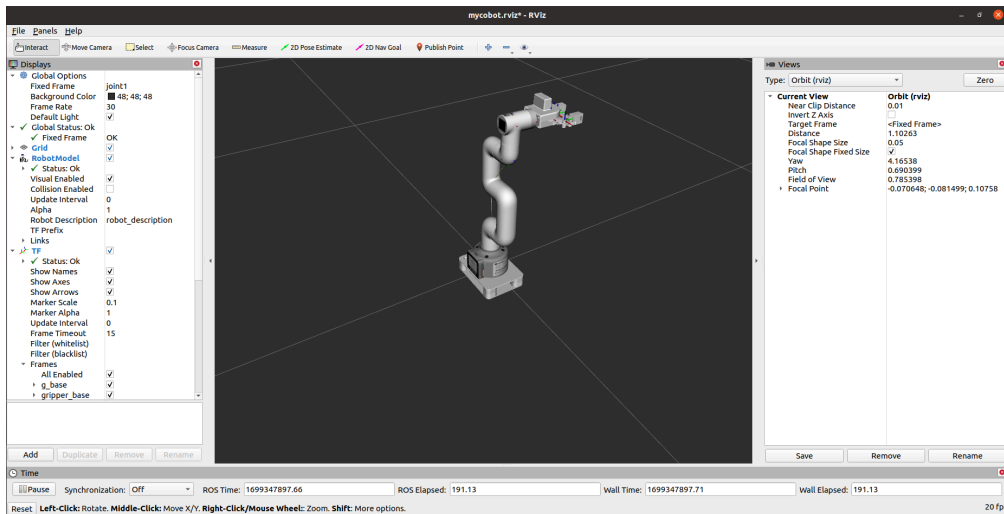
1 Load the model

Open a command line and run:

- myCobot 280-M5 version:

```
roslaunch mycobot_280 test_gripper.launch
```

It will open `rviz`, and you will see the following screen:



2 Slider Control

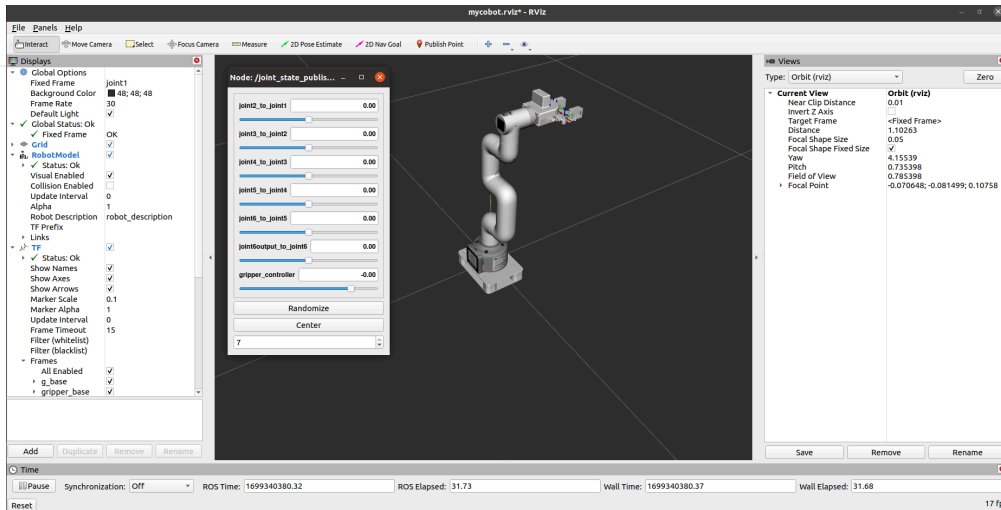
Open a command line and run:

- myCobot 280-M5 version:

```
# The default serial port name of myCobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is
roslaunch mycobot_280 slider_control_gripper.launch port:=/dev/ttyUSB0 baud:=115200
```

It will open `rviz` and a slider component, and you will see the following screen:

4.1 First-time self-check



Then you can **control the movement of the model in rviz by dragging the slider**. If you want the real myCobot to move with you, you need to **open another command line** and run:

- myCobot 280-M5 version:

```
# The default serial port name of myCobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is
rosrun mycobot_280 slider_control_gripper.py _port:=/dev/ttyUSB0 _baud:=115200
```

Please note: Since the robot arm will move to the current position of the model while the command is input, please make sure that the model in rviz does not have a model penetration phenomenon before you use the command. Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm.

3 Model following

In addition to the above control, we can also **let the model follow the movement of the real robot arm**. Open a command line and run:

- mycobot 280-M5 version:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is
rosrun mycobot_280 follow_display_gripper.py _port:=/dev/ttyUSB0 _baud:=115200
```

Then open another command line and run:

- mycobot 280-M5 version:

```
roslaunch mycobot_280 mycobot_follow_gripper.launch
```

It will open rviz to show the model following effect.

4 GUI control

Based on the previous, this package also provides a simple Gui control interface. This method is intended to allow real robotic arms to interact with each other. Please connect myCobot.

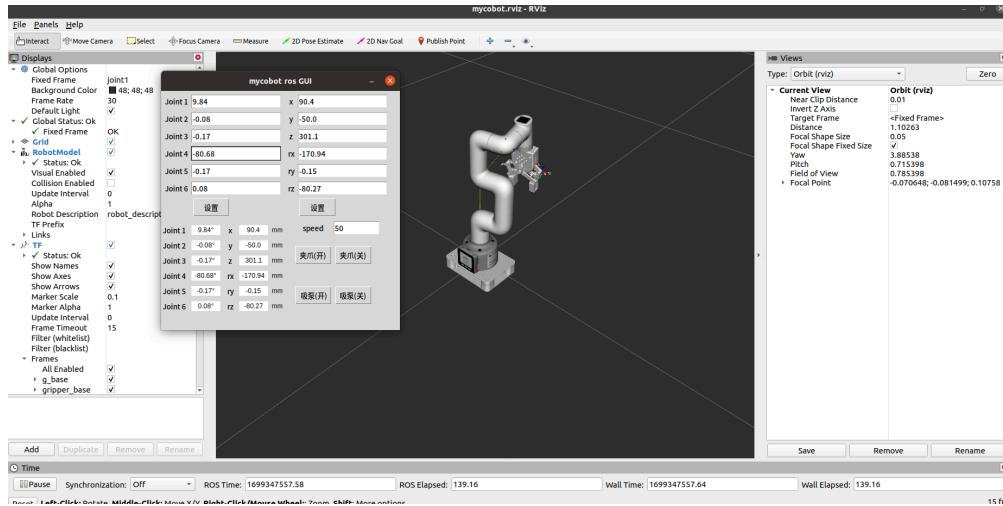
Open the command line:

4.1 First-time self-check

- mycobot 280-M5 version:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is  
roslaunch mycobot_280 simple_gui_gripper.launch port:=/dev/ttyUSB0 baud:=115200
```

It will open rviz and a GUI interface, and you will see the following screen:



5 Keyboard control

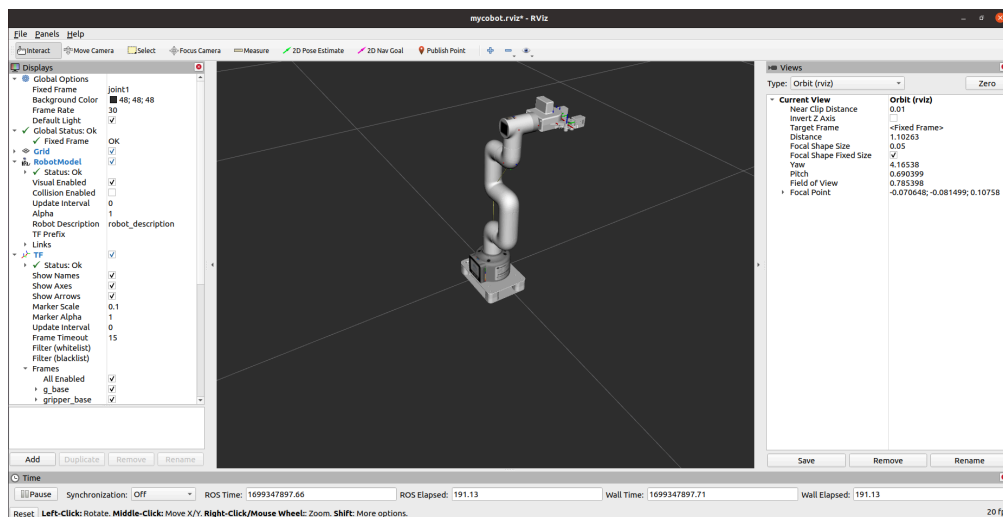
Keyboard control function has been added to the `mycobot_280` package, and it is synchronized in real time in rviz. This function relies on pythonApi, so make sure it is connected to the real robot arm.

Open a command line and run:

- mycobot 280-M5 version:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is  
roslaunch mycobot_280 teleop_keyboard_gripper.launch port:=/dev/ttyUSB0 baud:=115200
```

The running effect is as follows:



The command line will output mycobot information, as follows:

4.1 First-time self-check

SUMMARY

=====

PARAMETERS

```
* /mycobot_services/ baud: 115200
* /mycobot_services/port: /dev/ttyACM0
* /robot_description: <?xml version="1....
* /roscdistro: noetic
* /rosversion: 1.16.0
```

NODES

```
/
  mycobot_services (mycobot_communication/mycobot_topics.py)
  real_listener (mycobot_280/listen_real_of_topic.py)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)
  rviz (rviz/rviz)
```

auto-starting new master

process[master]: started with pid [217714]

ROS_MASTER_URI=http://localhost:11311

setting /run_id to 01e863c8-5642-11f0-a2da-335dad9016e7

process[rosout-1]: started with pid [217728]

started core service [/rosout]

process[robot_state_publisher-2]: started with pid [217731]

process[rviz-3]: started with pid [217733]

process[mycobot_services-4]: started with pid [217738]

process[real_listener-5]: started with pid [217740]

current pymycobot library version: 3.9.9

pymycobot library version meets the requirements!

ls: cannot access '/dev/ttyUSB*': No such file or directory

[INFO] [1751350196.024298]: /dev/ttyACM0,115200

MyCobot Status

Joint Limit:

joint 1: -168 ~ +168

joint 2: -135 ~ +135

joint 3: -150 ~ +150

joint 4: -145 ~ +145

joint 5: -165 ~ +165

joint 6: -180 ~ +180

Connect Status: True

Servo Infomation: all connected

Servo Temperature: unknown

4.1 First-time self-check

```
Atom Version: 7.2
```

Next, open another command line and run:

- mycobot 280-M5 version:

```
roslaunch mycobot_280 teleop_keyboard.py
```

You will see the following output in the command line:

```
[INFO] [1751342043.889285]: Waiting to receive current coordinates...
Mycobot Teleop Keyboard Controller (ROS1 - Topic Version)
-----
Movement (Cartesian):
      w (x+)
a (y-)  s (x-)  d (y+)
      z (z-)  x (z+)

Rotation (Euler angles):
      u (rx+)  i (ry+)  o (rz+)
      j (rx-)  k (ry-)  l (rz-)

Movement Step:
+ : Increase movement step size
- : Decrease movement step size

Gripper:
g - open  h - close

Pump:
b - open  m - close

Other:
1 - Go to init pose
2 - Go to home pose
3 - Save current pose as home
q - Quit

currently:  speed: 70  change percent: 5
[INFO] [1751342044.199244]: Current moving step: position 12.5 mm, angle attitude 9.0°
```

In this terminal, you can control the state of the robot and move the robot by pressing keys in the command line.

Parameters supported by this script:

- `_speed`: robot movement speed.
- `_change_percent`: percentage of moving distance.

5.2 myCobot vertical pump V2.0

1 Load the model

Open a command line and run:

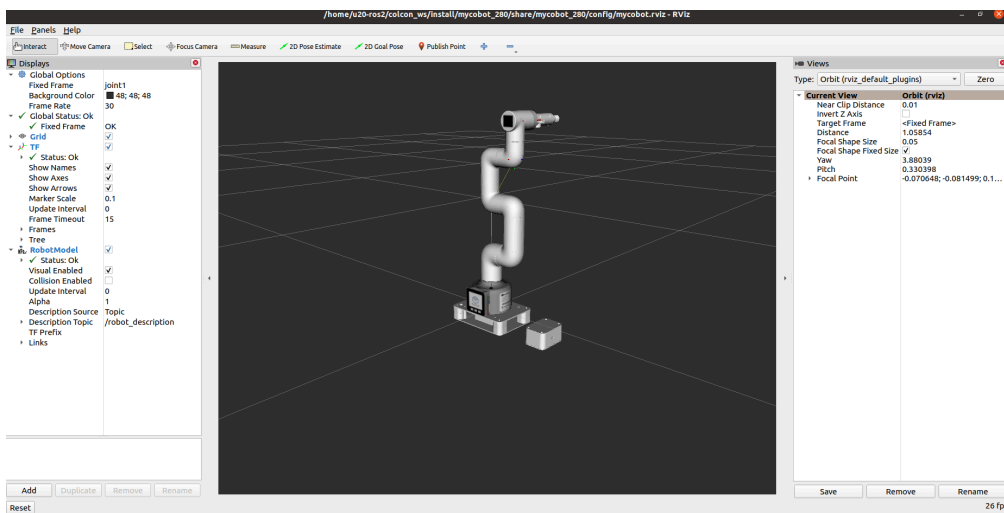
- myCobot 280-M5 version:

```
roslaunch mycobot_280 test_pump.launch
```

- myCobot 280-Pi version:

```
roslaunch mycobot_280pi test_pump.launch
```

It will **open rviz** and you will see the following screen:



2 Slider control

Note: This function only supports the control of the robot

Open a command line and run:

- myCobot 280-M5 version:

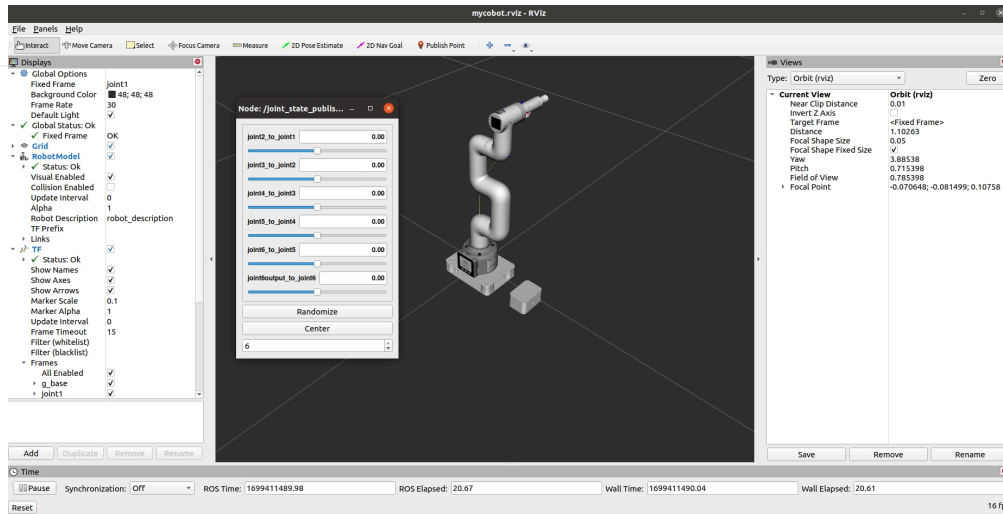
```
### The default serial port name of myCobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port
roslaunch mycobot_280 slider_control1_pump.launch port:=/dev/ttyUSB0 baud:=115200
```

- myCobot 280-Pi version:

```
### The default serial port name of myCobot 280-Pi version is "/dev/ttyAMA0" and the baud rate is 1000000.
roslaunch mycobot_280pi slider_control1_pump.launch port:=/dev/ttyAMA0 baud:=1000000
```

It will **open rviz** and a **slider component**, and you will see the following screen:

4.1 First-time self-check



Then you can **control the movement of the model in rviz by dragging the slider**. If you want the real myCobot to move with you, you need to **open another command line** and run:

- myCobot 280-M5 version:

```
### The default serial port name of myCobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port  
roslaunch mycobot_280 slider_control.py _port:=/dev/ttyUSB0 _baud:=115200
```

- myCobot 280-Pi version:

```
### The default serial port name of myCobot 280-Pi version is "/dev/ttyAMA0" and the baud rate is 1000000.  
roslaunch mycobot_280pi slider_control.py _port:=/dev/ttyAMA0 _baud:=1000000
```

Please note: Since the robot will move to the current position of the model when the command is input, please make sure that the model in rviz does not have any penetration before you use the command. Do not drag the slider quickly after connecting the robot to prevent damage to the robot.

3 GUI control

Based on the previous, this package also **provides a simple Gui control interface**. This method is intended for real robots to interact with each other, please connect myCobot.

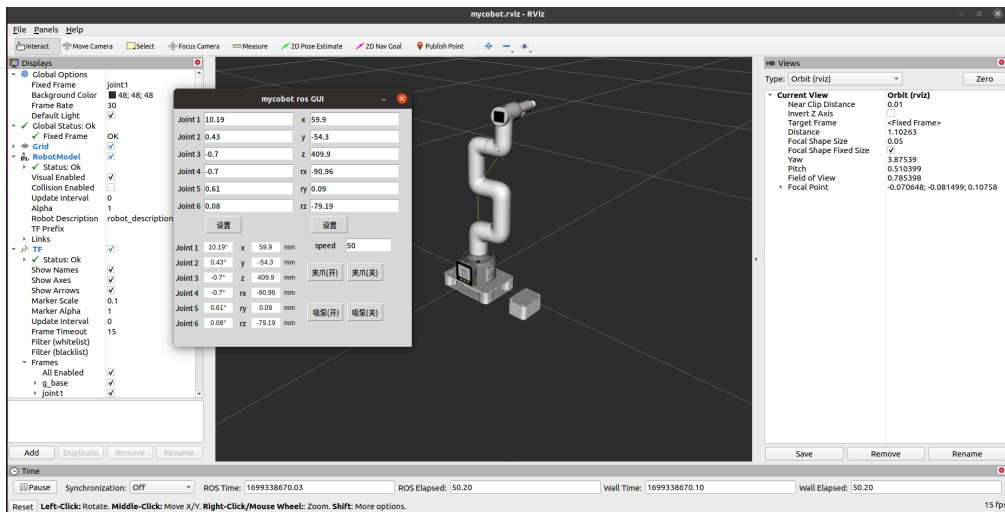
Open the command line:

- mycobot 280-M5 version:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port na  
roslaunch mycobot_280 simple_gui_pump.launch port:=/dev/ttyUSB0 baud:=115200
```

It will **open rviz and a GUI interface**, and you will see the following screen:

4.1 First-time self-check



4 Keyboard control

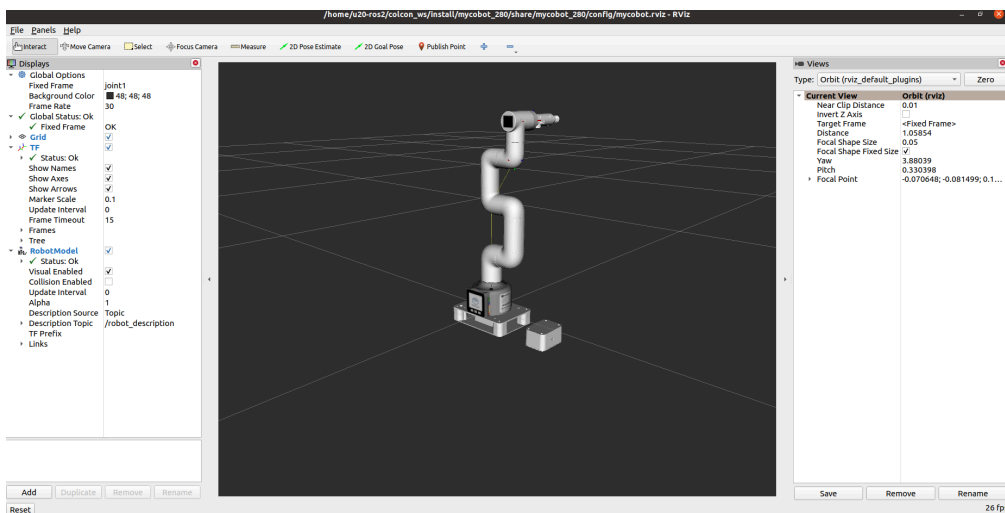
Added keyboard control function in the `mycobot_280` package, and synchronized in real time in rviz. This function depends on python API, so make sure to connect to the real robot arm.

Open a command line and run:

- mycobot 280-M5 version:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is  
roslaunch mycobot_280 teleop_keyboard_pump.launch port:=/dev/ttyUSB0 baud:=115200
```

The running effect is as follows:



Next, open another command line and run:

- mycobot 280-M5 version:

```
roslaunch mycobot_280 teleop_keyboard.py
```

You will see the following output in the command line:

4.1 First-time self-check

```
[INFO] [1751342043.889285]: Waiting to receive current coordinates...
Mycobot Teleop Keyboard Controller (ROS1 - Topic Version)
-----
Movement (Cartesian):
      w (x+)
  a (y-)  s (x-)  d (y+)
      z (z-)  x (z+)

Rotation (Euler angles):
  u (rx+)  i (ry+)  o (rz+)
  j (rx-)  k (ry-)  l (rz-)

Movement Step:
  + : Increase movement step size
  - : Decrease movement step size

Gripper:
  g - open  h - close

Pump:
  b - open  m - close

Other:
  1 - Go to init pose
  2 - Go to home pose
  3 - Save current pose as home
  q - Quit

currently:  speed: 50  change percent: 5
[INFO] [1751342044.199244]: Current moving step: position 12.5 mm, angle attitude 9.0°
```

In this terminal, you can control the state of the robot and move the robot through the keys in the command line.

Parameters supported by this script:

- `_speed`: robot movement speed.
- `_change_percent`: movement distance percentage.

5.3 Camera Flange

1 Load the model

Open a command line and run:

- myCobot 280-M5 version:

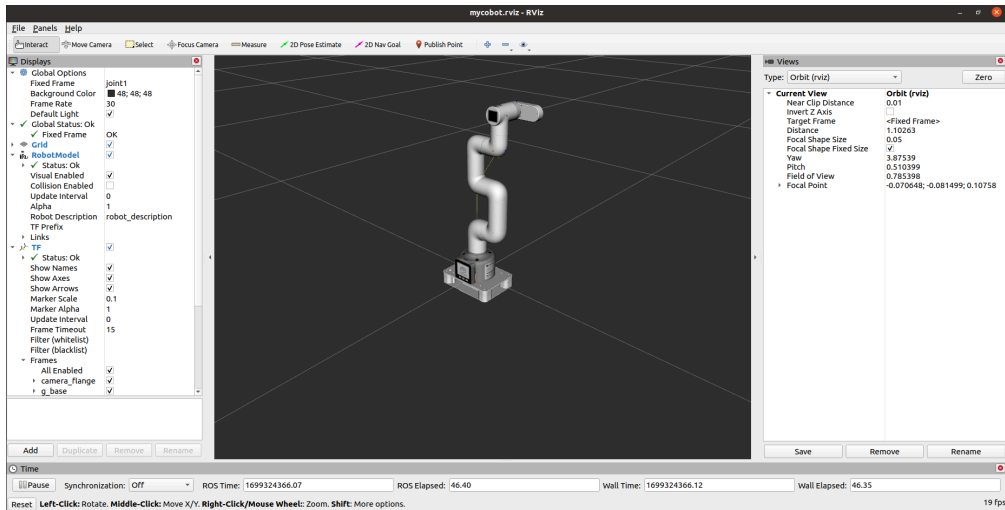
```
roslaunch mycobot_280 test_camera_flange.launch
```

- myCobot 280-PI version:

4.1 First-time self-check

```
roslaunch mycobot_280pi test_camera_flange.launch
```

It will open rviz and you will see the following screen:



2 Slider Control

Note: This function only supports the control of the robot

Open a command line and run:

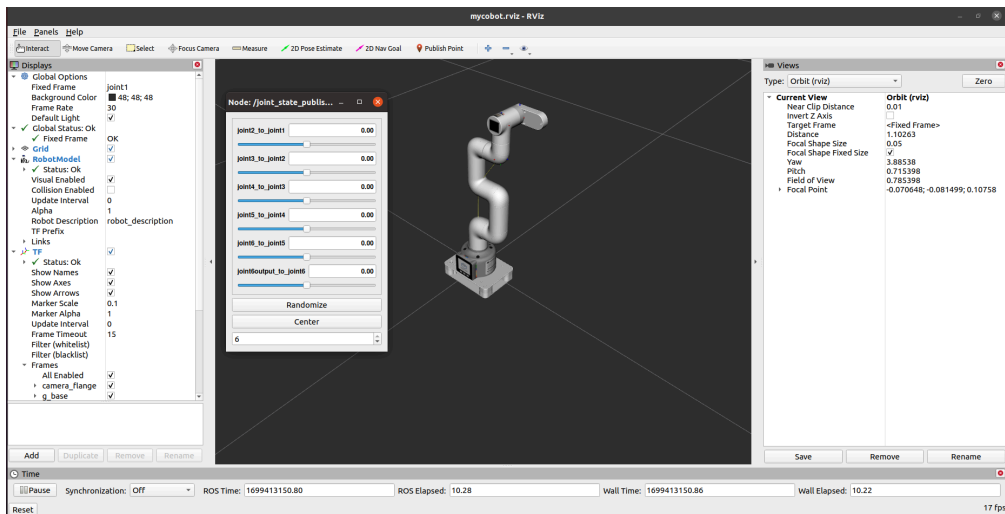
- myCobot 280-M5 version:

```
# The default serial port name of myCobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is  
roslaunch mycobot_280 slider_control_camera_flange.launch port:=/dev/ttyUSB0 baud:=115200
```

- myCobot 280-Pi version:

```
# The default serial port name of myCobot 280-Pi version is "/dev/ttyAMA0" and the baud rate is 1000000.  
roslaunch mycobot_280pi slider_control_camera_flange.launch port:=/dev/ttyAMA0 baud:=1000000
```

It will open rviz and a slider component, and you will see the following screen:



4.1 First-time self-check

Then you can **control the movement of the model in rviz by dragging the slider**. If you want the real myCobot to move with you, you need to **open another command line** and run:

- myCobot 280-M5 version:

```
# The default serial port name of myCobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is
rosrun mycobot_280 slider_control.py _port:=/dev/ttyUSB0 _baud:=115200
```

- myCobot 280-Pi version:

```
# The default serial port name of myCobot 280-Pi version is "/dev/ttyAMA0" and the baud rate is 1000000.
rosrun mycobot_280pi slider_control.py _port:=/dev/ttyAMA0 _baud:=1000000
```

Please note: Since the robot will move to the current position of the model when the command is input, please make sure that the model in rviz does not have any penetration before you use the command. Do not drag the slider quickly after connecting the robot to prevent damage to the robot.

5.4 Camera Flange & Pump

1 Load the model

Open a command line and run:

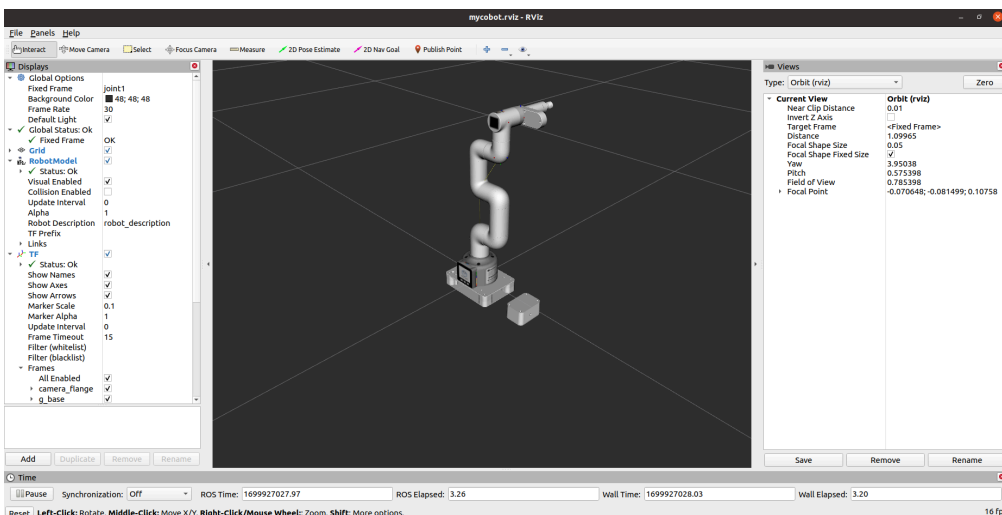
- myCobot 280-M5 version:

```
roslaunch mycobot_280 test_camera_flange_pump.launch
```

- myCobot 280-Pi version:

```
roslaunch mycobot_280pi test_camera_flange_pump.launch
```

It will **open rviz** and you will see the following:



5.5 URDF model address

1 myCobot adaptive gripper

- [myCobot 280-M5 version](#)

2 myCobot vertical suction pump V2.0

- [myCobot 280-M5 version](#)
- [myCobot 280-PI version](#)

3 Camera flange

- [myCobot 280-M5 version](#)
- [myCobot 280-PI version](#)

4 Camera flange && suction pump

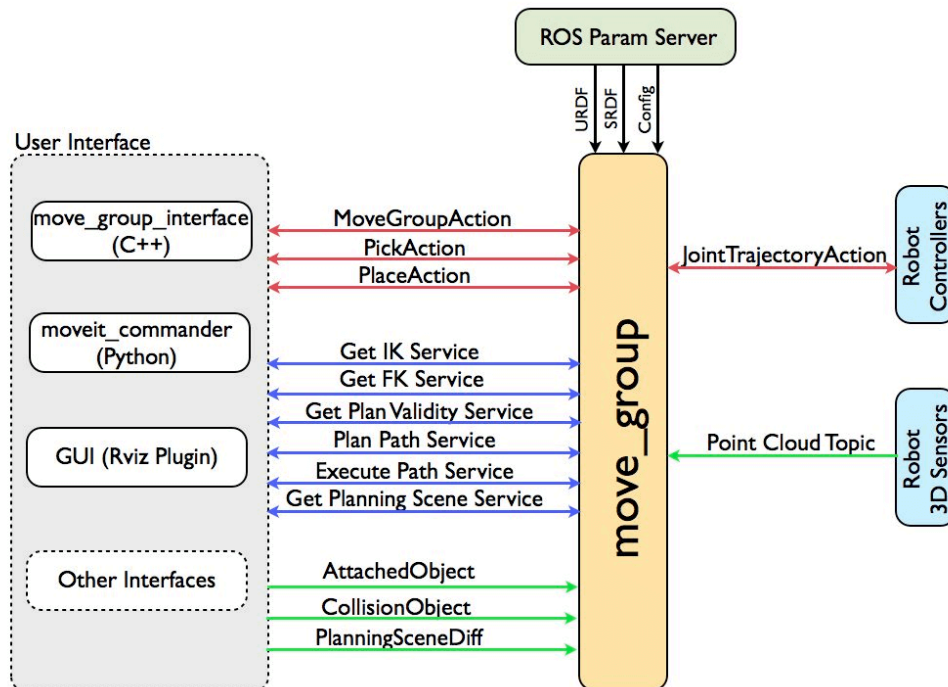
- [myCobot 280-M5 version](#)
- [myCobot 280-PI version](#)

MoveIt

Introduction to MoveIt

MoveIt is an integrated development platform in ROS, consisting of a variety of function packages for manipulating robotic arms, including: motion planning, manipulation, control, inverse kinematics, 3D perception, and collision detection.

The figure below shows the high-level structure of the main node `move_group` provided by MoveIt. It is like a combiner: it integrates all individual components together and provides a series of actions and services for users to use.



User Interface

Users can access the operations and services provided by `move_group` in three ways:

- In C++: Use the `move_group_interface` package to facilitate the practical use of `move_group`.
- In Python: Use the `moveit_commander` package.
- Via GUI: Use Rviz (ROS Visualizer) from Motion-commander.

`move_group` can be configured using the ROS parameter server, from which it can also get the robot's URDF and SRDF.

Configuration

`move_group` is a ROS node. It uses the ROS parameter server to get three kinds of information:

1. URDF - `move_group` looks up the `robot_description` parameter on the ROS parameter server to get the robot's URDF.
2. SRDF - `move_group` looks up the `robot_description_semantic` parameter on the ROS parameter server to get the robot's SRDF. The SRDF is usually created by the user using the MoveIt Setup Assistant.

4.1 First-time self-check

3. MoveIt Configuration - move_group will look up additional configuration specific to MoveIt on the ROS parameter server, including joint limits, kinematics, motion planning, perception, and other information. Configuration files for these components are automatically generated by the MoveIt setup assistant and stored in the configuration directory of the corresponding MoveIt configuration package for the robot.

How to use MoveIt

Note: For better motion effects, the Atom firmware version of the end arm is 6.5, and the python driver library pymycobot version is 3.5.3

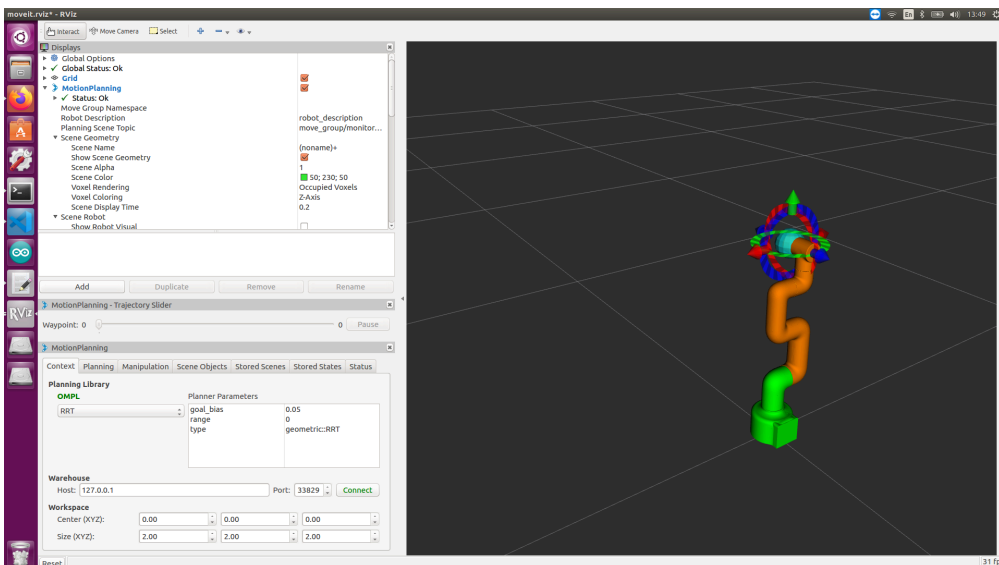
mycobot_ros now has MoveIt integrated.

Open the command line and run:

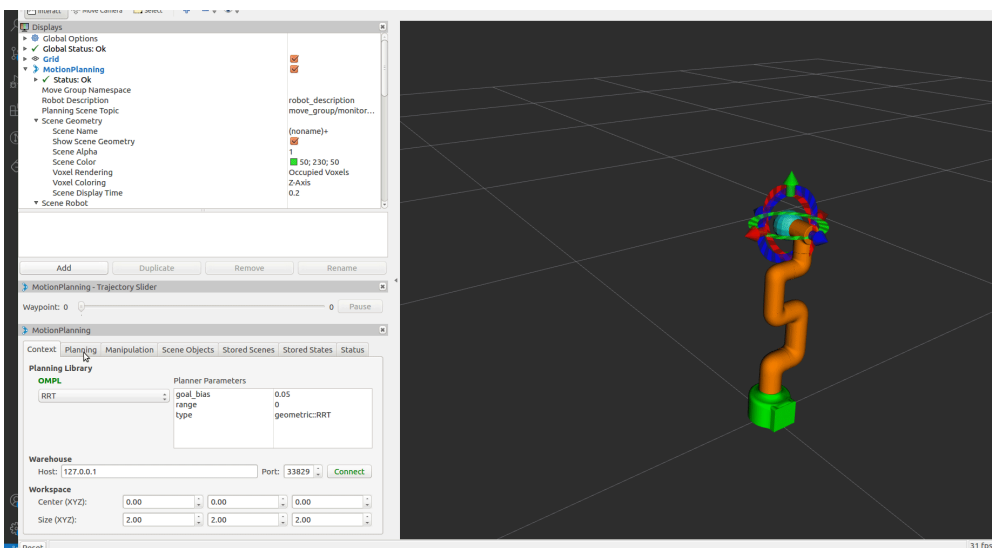
- mycobot 280-Arduino version:

```
roslaunch mycobot_280arduino_moveit mycobot_moveit.launch
```

The running effect is as follows:



You can plan and execute, demonstration effect:



4.1 First-time self-check

If you need to let the real robot arm execute the plan synchronously, you need to open another command line and run:

- mycobot 280-Arduino version:

```
# The default serial port name of mycobot 280-Arduino version is "/dev/ttyACM0" and the baud rate is 1000000.  
roslaunch mycobot_280arduino_moveit sync_play.py _port:=/dev/ttyACM0 _baud:=1000000
```

Introduction to ROS2

ROS2's predecessor is ROS, which stands for Robot Operating System. However, ROS itself is not an operating system, but a software library and tool set. The emergence of ROS solved the communication problem of various robot components. Later, more and more robot algorithms were integrated into ROS. ROS2 inherited ROS and is more powerful and easier to use than ROS.

Design goals and features of ROS2

ROS2 shoulders 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 system:** In the future, robots are no longer independent individuals, and robots also need to communicate and collaborate. ROS2 provides standard methods and communication mechanisms for the application of multi-robot systems.
- **Cross-platform:** The control platforms used will be very different in different robot application scenarios. In order to enable all robots to run ROS2, ROS2 can run cross-platform on Linux, Windows, MacOS, and RTOS.
- **Real-time:** Robot motion control and many behavioral strategies require robots to be real-time. For example, a robot must reliably detect pedestrians in front of it within 100 milliseconds, or complete kinematic and dynamic calculations within 1 millisecond. ROS2 provides basic requirements in real time like this.
- **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 R&D stage, but can also be directly installed in products and enter 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. Project management tools and mechanisms for the entire process of design, development, debugging, testing, and deployment will also be reflected in ROS2, making it easier for us to develop robots.

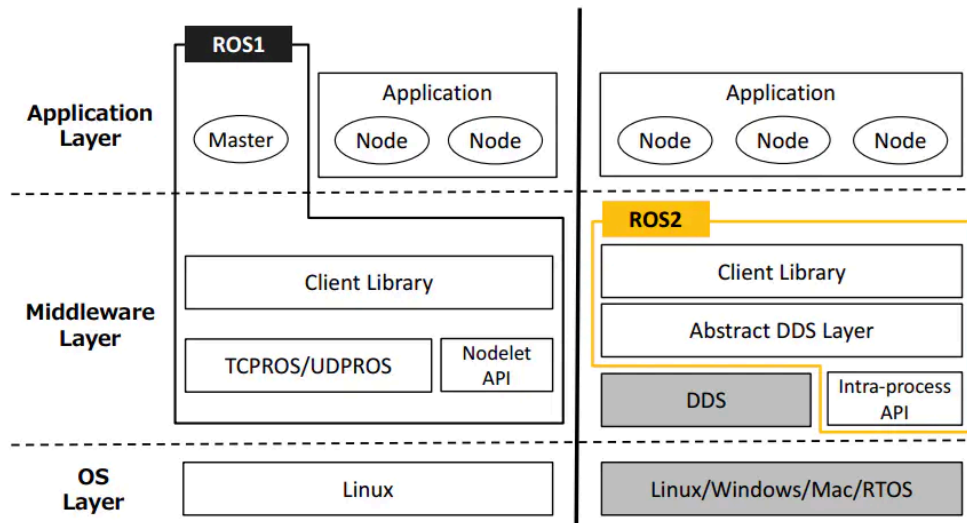
Release version

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

| ROS2 version | Release date | Maintenance period | Ubuntu version |
|--------------------------|--------------|--------------------|-------------------------------|
| Dashing | 2019.5 | 2021.5 | Ubuntu 18.04 (Bionic Beaver) |
| Eloquent | 2019.11 | 2020.11 | Ubuntu 18.04 (Bionic Beaver) |
| 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) |

Comparison between ROS and ROS2

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



- **OS Layer:** OS layer. In ROS2, it can be built on Linux or other systems, even bare metal without an operating system.
- **Middleware Layer:** 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/subscription in distributed real-time systems.
- **Application Layer:** Application layer. ROS1 relies on ROS Master, while in ROS2, a discovery mechanism called "Discovery" is used between nodes to help each other establish connections.

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 designs 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 autonomous driving. In addition, there are the following disadvantages:

- TCP-based communication has poor real-time performance and high system overhead
- Not friendly to python3 support
- Incompatible message mechanism
- No encryption mechanism, low security

ROS2 first removes the master node in ROS. After removing the master node, each node can discover each other through the DDS node. Each node is equal and can achieve one-to-one, one-to-many, and many-to-many communication. After using DDS for communication, reliability and stability are enhanced.

Compared with **ROS** that only supports Linux systems, **ROS2** also supports Windows, Mac and even RTOS platforms.

ROS2 Development Guide

[ROS2 Installation Guide](#)

[ROS2 Basic Use](#)

[rviz Use Guide](#)

Environment Installation

M5/Arduino Version:

Note:When installing the virtual machine system, please install the **Ubuntu 20.04 version of the system**. The installation method is the same as Ubuntu 18.04. If you want to use the `moveit2` function, you need to install **Ubuntu 22.04 version system**.

To install different versions of Ubuntu systems in Linux, please refer to the [13.1.1 Environment Construction](#) section.

ROS2 Environment Construction

ROS2 Installation

The basic development environment construction requires the installation of the robot operating system ROS2 and the git version manager. The following describes their installation methods and processes respectively.

For **myCobot 280-M5**, **myCobot 280-Arduino**, **myCobot 320-M5**, **myPalletizer 260-M5** and **mechArm 270-M5** devices, please refer to the following installation methods and processes.

myCobot 280-PI, **myCobot 320-PI**, **myPalletizer 260-PI** and **mechArm 270-PI** devices only need to execute the `mycobot_ros2` installation package.

Version selection

ROS2 has a one-to-one correspondence with Ubuntu. Different versions of Ubuntu correspond to different versions of ROS2. For reference, see the following website: <http://docs.ros.org/en/foxy/Releases.html>

Here are the ROS2 versions supported by Ubuntu:

| ROS2 version | Release date | Maintenance deadline | Ubuntu version |
|--------------|--------------|----------------------|-------------------------------|
| Foxy | June 5, 2020 | May 2023 | Ubuntu 20.04(Focal Fossa) |
| Galactic | May 23, 2021 | November 2022 | Ubuntu 20.04(Focal Fossa) |
| Humble | May 23, 2022 | May 2027 | Ubuntu 22.04(Jammy Jellyfish) |

Please install the corresponding ROS2 version according to the Ubuntu version you installed

If the versions are different, the download will fail. Here we choose Ubuntu 20.04 (recommended), and the corresponding ROS2 version is ROS2 Foxy

NOTE: Currently we do not provide any reference for installing ROS2 on Windows. If necessary, please refer to <http://docs.ros.org/en/foxy/Installation/Alternatives/Windows-Development-Setup.html>

Start installation

1 Add source

There is no ROS2 in the software source list of Ubuntu itself Software source, so you need to first **configure the ROS2 software source to the software list warehouse** before you can download ROS2. Open a console terminal (shortcut key `Ctrl+Alt+T`), enter the following command:

4.1 First-time self-check

- Official source:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.or
```

- If the download speed is slow, it is recommended to select a mirror source nearby 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 te
```

2 Set the key

Configure the public network key, this step is to let the system confirm that our path is safe, so that there is no problem 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 the new software source, you need to **update the software source list**, open a console terminal (shortcut key `Ctrl+Alt+T`), and enter the following command:

```
sudo apt-get update
```

Execute **install ROS2**, open a console terminal (shortcut key `Ctrl+Alt+T`), please enter 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, please wait patiently

After the installation is complete, refresh the environment variables:

```
source /opt/ros/foxy/setup.bash
```

1.1.3 Set up the ros2 environment

In order to avoid the need to re-validate the ROS2 function path every time the terminal window is closed, we can **configure the path to the environment variable**, so that the ROS2 function path can be automatically validated every time a new terminal is opened. Execute the following commands in the terminal in sequence, open a console terminal (shortcut key `Ctrl+Alt+T`) and execute the following commands:

```
# Ubuntu 20.04 foxy version
# Add the ros 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
```

Install ROS2 additional dependencies

Enter the following command in the terminal **Install ROS2 additional dependencies**, open a console terminal (shortcut `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
```

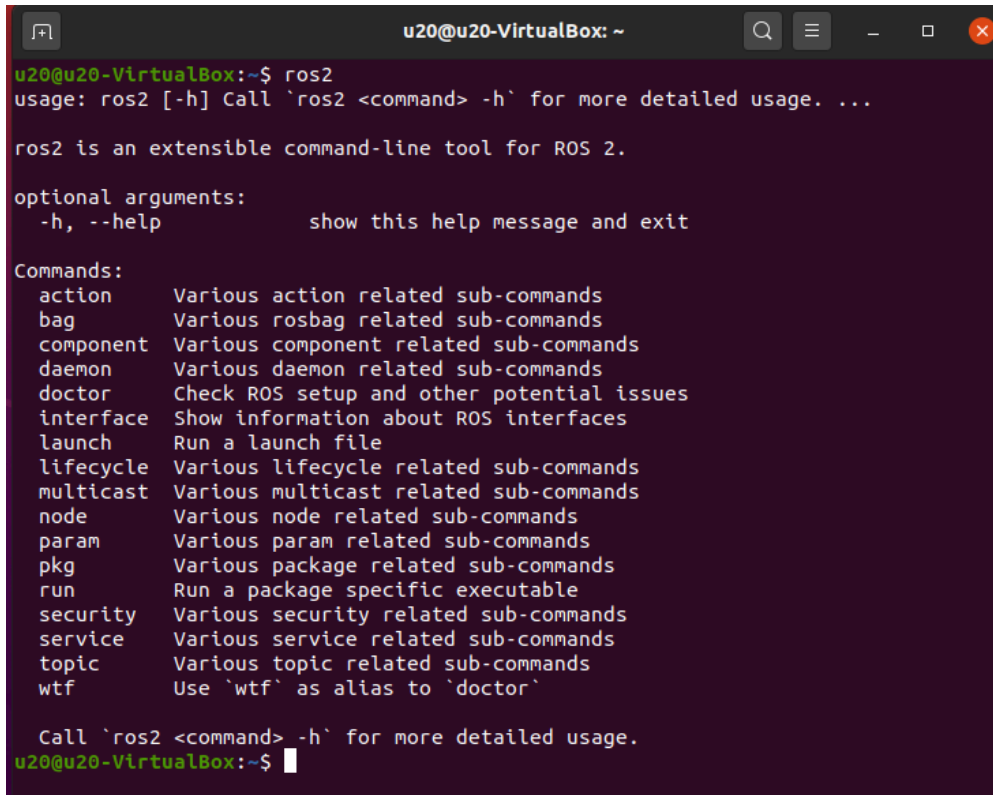
1.1.4 Verify installation

To verify ROS2 To check whether the installation is successful, open a console terminal (shortcut key `Ctrl+Alt+T`), and execute the following command in the terminal:

4.1 First-time self-check

```
ros2
```

When the following interface is displayed, it means that ROS2 is 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:~$
```

2 git installation

3.1 Update the software source list

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

```
sudo apt-get update
```

3.2 Installation git

Open a console terminal (shortcut key `Ctrl+Alt+T`), enter the following command in the terminal window, **execute git installation**:

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

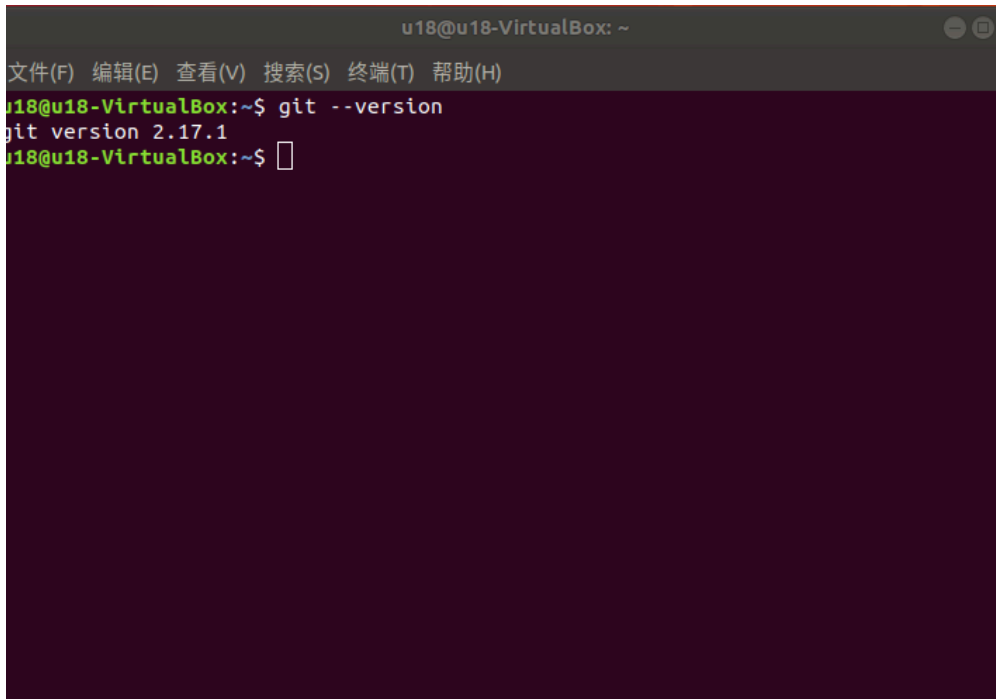
3.3 Verify installation

Read git version, open a console terminal (shortcut key `Ctrl+Alt+T`), enter the following command in the terminal window:

4.1 First-time self-check

```
git --version
```

The git version number can be displayed in the terminal, as shown below, which means the installation is successful.

A screenshot of a terminal window titled 'u18@u18-VirtualBox: ~'. 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 before and after the command. The terminal interface includes a menu bar at the top with options in Chinese: '文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)'.

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

3 mycobot_ros2 installation

`mycobot_ros2` is a ROS2 package launched by ElephantRobotics, which is compatible with its desktop six-axis robot arm mycobot series.

Project address: http://github.com/elephantrobotics/mycobot_ros2

Supported ROS2 versions:

- Ubuntu 20.04 / ROS2 Foxy - foxy branch
- Ubuntu 20.04 / ROS2 Galactic - galactic branch
- Ubuntu 22.04 / ROS2 Humble - humble branch (default is main branch)

3.1 Prerequisites

Before installing the package, please ensure that you have a ros2 workspace.

Here we give **sample commands for creating a workspace**. Open a console terminal (shortcut key `Ctrl+Alt+T`), and enter the following command in the command line:

```
mkdir -p ~/colcon_ws/src # Create a folder
```

Add workspace environment

The official default ROS2 workspa

4.1 First-time self-check

```
echo "source ~/colcon_ws/install/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

3.2 Installation

NOTE:

- This package depends on ROS2 and MoveIT2. Make sure to install ROS2 and MoveIT2 successfully before using it.
- The interaction between this package and the real robot arm depends on PythonApi - `pymycobot`
- The Api project is: <https://github.com/elephantrobotics/pymycobot>
- Quick installation: `pip install pymycobot --upgrade`

When executing the `pip install pymycobot --upgrade` 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 according to the prompt

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

- If your Ubuntu system is version 20.04, please execute the command `sudo apt install python3-pip` to install pip. After pip is installed, execute it again in the terminal

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

- The installation method depends on Git, please make sure Git is installed on your computer.

The official default ROS2 workspace is `colcon_ws`.

```
cd colcon_ws/src # Enter the src folder in 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 recompiling every time you adju
source install/setup.bash # Add environment variables
```

4 Update the mycobot_ros2 package

To ensure that users can use the latest official package in a timely manner, you can enter the `/home/er/colcon_ws/src` folder through the file manager, open the console terminal (shortcut `Ctrl+Alt+T`), and enter the following command to update:

4.1 First-time self-check

```
# Clone the code on github
cd colcon_ws/src # Enter the src folder in 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 recompiling every time you adjust
source install/setup.bash # Add environment variables
```

Note: If the `mycobot_ros2` folder already exists in the `/home/er/colcon_ws/src` (equivalent to `~/colcon_ws/src`) directory, you need to delete the original `mycobot_ros2` first, and then execute the above command. Among them, `er` in the directory path is the system user name. If there is any inconsistency, please modify it.

To reduce compilation time, you can compile a certain package separately, where `package_name` is the name of the specific package. Please modify it according to your actual situation.

```
cd ~/colcon_ws
colcon build --packages-select package_name
source install/setup.bash
```

M5/Arduino version import system image using ROS2/Movelt2:

Note: To simplify the environment setup, we will provide a Linux system image (Ubuntu 22.04), Virtual Box installation package, and its extension package. Next, we will teach you how to install Virtual Box and import the Linux system image (the default password is 123). **Built-in environment:** ROS2 humble + Movelt2 + Git + pymycobot + mycobot_ros2

Install the virtual machine

Go to the [official website](#) to download the virtual machine Virtual Box

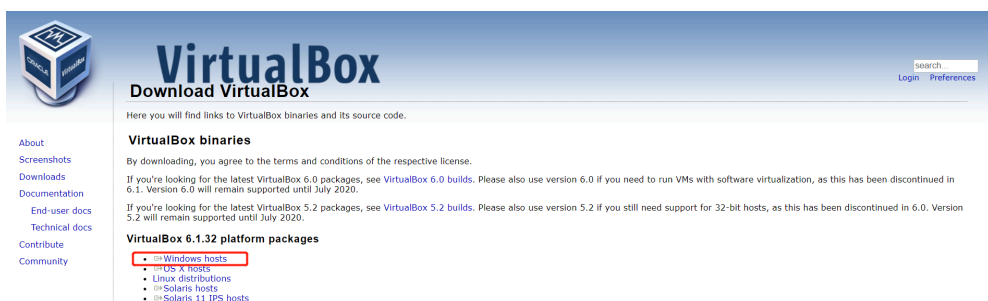
Or go to the [official website](#) to download the virtual machine VM ware

VirtualBox installation package: [Windows hosts](#)

VirtualBox Extension Pack: [VirtualBox 7.0.10 Oracle VM VirtualBox Extension Pack](#)

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

We chose to download Virtual box because it is free.





Download Linux system image

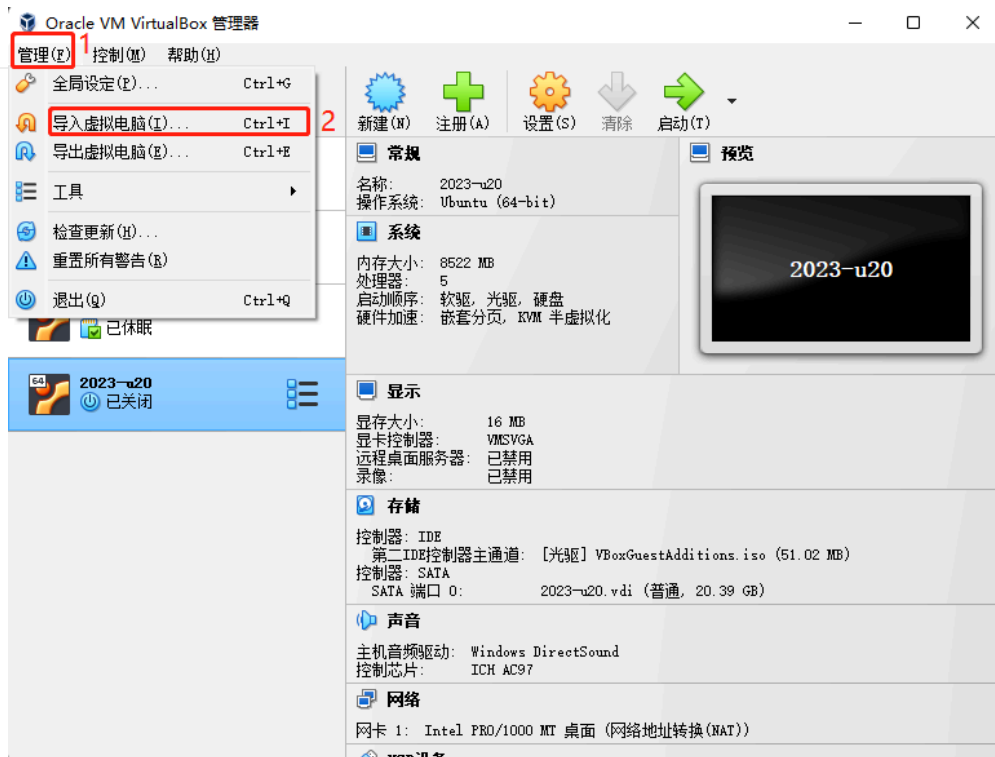
Click to download:[Linux ubuntu22.04](#)

Import Linux system image

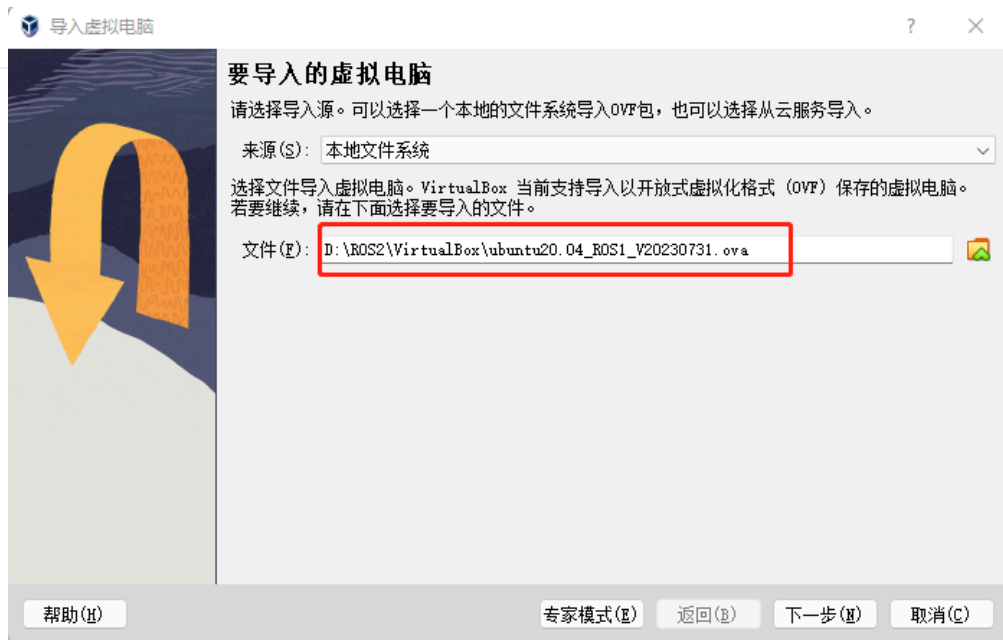
Note: For the import method, please refer to the import method of Ubuntu 20.04 system

In the Virtual Box interface, click Management -> Import Virtual Computer -> Select Virtual Image -> Select the installation path and import, and then install it as follows.

4.1 First-time self-check



4.1 First-time self-check



Just wait for the image to be imported. The installation is successful as shown below.

4.1 First-time self-check



Then start the system, the default password is **123**

At this point, the ROS2 environment has been set up. For the use of ROS2, please refer to Rviz Introduction and Usage.

Basic Tools

In this chapter, you will learn about the common command tools of ROS2.

Topics

ROS 2 breaks down complex systems into many modular nodes. Topics are an important element of the ROS graph, acting as a bus for nodes to exchange messages. Topics are one of the main ways data moves between nodes, and therefore between different parts of the system.

Specific reference: [Official tutorial](#)

- **topics help**

```
ros2 topics -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node  
ros2 run turtlesim turtle_teleop_key
```

- **Node relationship graph**

```
rqt_graph
```

- **Understand topics related commands**

```
ros2 topics -h
```

- **Topic list**

```
ros2 topic list  
ros2 topic list -t # Display the corresponding message type
```

- **View topic content**

```
ros2 topic echo <topic_name>  
ros2 topic echo /turtle1/cmd_vel
```

- **Display topic related information, type**

```
ros2 topic info <topic_name>  
# Output /turtle1/cmd_vel topic interface related information  
ros2 topic info /turtle1/cmd_vel
```

- **Display interface related information**

4.1 First-time self-check

```
ros2 interface show <msg_type>
# Output geometry_msgs/msg/Twist interface related information
ros2 interface show geometry_msgs/msg/Twist
```

- **Publish command**

```
ros2 topic pub <topic_name> <msg_type> '<args>'
# Publish 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: 0.0}"
# Publish 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: 0.0}"
```

- **View the frequency of topic publishing**

```
ros2 topic hz <topic_name>
#Output/turtle1/cmd_vel publishing 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: 0.0}"
```

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 laser rangefinders, etc.). Each node can send and receive data to other nodes through topics, services, actions, or parameters. A complete robotic system consists of many nodes working together. In ROS 2, a single executable file (C++ program, Python program, etc.) can contain one or more nodes.

Specific reference: [Official tutorial](#)

- **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 the node relationship graph**

```
rqt_graph
```

- **Remap**

4.1 First-time self-check

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=my_turtle
ros2 node list
```

- **View node information**

```
ros2 node info <node_name>
ros2 node info /my_turtle
```

Services

Services are another way for nodes in the ROS graph to communicate. Services are based on a call and response model, rather than the publisher-subscriber model of topics. While topics allow nodes to subscribe to a stream of data and get continuous updates, services only provide data when specifically called by a client.

Specific reference: [Official tutorial](#)

- **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 type received by the service**

```
ros2 service type <service_name>
ros2 service type /clear
```

- **Find services that use a certain message type**

```
ros2 service find <type_name>
ros2 service find std_srvs/srv/Empty
```

- **View service message type definition**

```
ros2 interface show <type_name>.srv
ros2 interface show std_srvs/srv/Empty.srv
```

- **Call service command, clear walking track**

4.1 First-time self-check

```
ros2 service call <service_name> <service_type>
ros2 service call /clear std_srvs/srv/Empty
```

- **Generate a new turtle**

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle2'}"
```

Parameters

Parameters are configuration values for a node. You can think of parameters as node settings. Nodes can store parameters as integers, floating point numbers, Boolean values, strings, and lists. In ROS 2, each node maintains its own parameters. For more background on parameters, see the concepts documentation.

Specific reference: [Official tutorial](#)

- **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 parameter value**

```
ros2 param get <node_name> <parameter_name>
ros2 param get /turtlesim background_g
```

- **Set parameter value**

```
ros2 param set <node_name> <parameter_name> <value>
ros2 param set /turtlesim background_r 150
```

- **Export parameter values**

```
ros2 param dump <node_name>
ros2 param dump /turtlesim
```

- **Import parameters independently**

4.1 First-time self-check

```
ros2 param load <node_name> <parameter_file>
ros2 param load /turtlesim ./turtlesim.yaml
```

- **Start node and import parameters**

```
ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>
ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```

Actions

Actions are a type of communication in ROS 2 used for long-running tasks. They consist of three parts: a goal, a response, and a result.

Actions are based on topics and services. They function like services, except that actions are preemptible (you can cancel them while they are executing). They also provide steady feedback, rather than services that return a single response.

Actions use a client-server model, similar to the publisher-subscriber model (described in the topic tutorial). An "action client" node sends a target to an "action server" node, which confirms the target and returns a stream of feedback and results.

Specific reference: [Official tutorial](#)

- **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 rotate, press F to cancel

- **View the server and client of the node action**

```
ros2 node info /turtlesim
```

- **View the action list**

```
ros2 action list
ros2 action list -t # DisplayAction type
```

- **View action information**

```
ros2 action info <action>
ros2 action info /turtle1/rotate_absolute
```

- **View action message content**

4.1 First-time self-check

```
ros2 interface show turtlesim/action/RotateAbsolute
```

- **Send action target information**

```
ros2 action send_goal <action_name> <action_type>
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
```

RQt

RQt is a graphical user interface framework that implements various tools and interfaces in the form of plugins. You can run all your existing GUI tools as dockable windows in RQt! The tools can still be run in the traditional standalone way, but RQt makes it easier to manage all the different windows in a single screen layout.

Specific reference: [Official Tutorial](#)

You can easily run any RQt tool/plugin by:

```
rqt
```

- **rqt help**

```
rqt -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- Action browser: / Plugins -> Actions -> Action Type Browser
- Parameter reconfiguration: / Plugins -> configuration -> Parameter Reconfigure
- Node graph: /Node Graph
- Control steering: /Plugins -> Robot Tools -> Robot Steering
- Service call: /Plugins -> Services -> Service Caller
- Service type browser: Plugins -> Services -> Service Type Browser
- Message publishing: Plugins -> Topics -> Message Publisher
- Message type browser: Plugins -> Topics -> Message Type Browser
- Topic list: Plugins -> Topics -> Topic Monitor
- Draw a curve: Plugins -> Visualization -> Plot
- **View log: rqt_console**

4.1 First-time self-check

```
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
```

TF2

tf2 It is a transformation library that allows users to track multiple coordinate systems over time. tf2 maintains the relationship between coordinate systems in a time-buffered tree structure and lets users transform points, vectors, etc. between any two coordinate systems at any desired time point.

Specific reference: [Official Tutorial](#)

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 two turtles, the first turtle automatically follows the second

```
ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

- Control the movement of the first 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 the two coordinate systems**

```
ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]
ros2 run tf2_ros tf2_echo turtle2 turtle1
```

- **View TF relations on rviz**

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz
```

URDF

URDF is the Unified Robot Description Format, which is used to specify robot geometry and organization in ROS.

Specific reference: [Official tutorial](#)

- **Full syntax**

4.1 First-time self-check

```
<robot>
  # describe:
  # Parameters: name=""
  # Child node:
  <link>
    # Description:
    # Parameters: name=""
    # Child node:
    <visual>
      # describe:
      # Parameters:
      # child nodes:
      <geometry>
        # description
        # parameters
        # Child node:
        <cylinder />
          # Description:
          # Parameters:
            # length="0.6"
            # radius="0.2"
        <box />
          # description
          # Parameters:size="0.6 0.1 0.2"
        <mesh />
          # Description
          #Parameters: filename="package://urdf_tutorial/meshes/l_finger_tip.dae"
      <collision>
        # Description: collision element, prioritized
        # parameters
        # child node
        <geometry>
      <inertial>
        # description
        # parameters
        # Child nodes:
        <mass />
          # description: mass
          # Parameters: value=10
        <inertia />
          # Description: Inertia
          # Parameters: i+"Cartesian product of xyz" (9 in total)="0.4"
    <origin />
      # Description:
      # Parameters:
        # rpy="0 1.5 0"
        # xyz="0 0 -0.3"
    <material />
      # Description
      # Parameters: name="blue"
```

4.1 First-time self-check

```
<joint>
  # Description
  # Parameters:
    # name=""
    # type=""
      # fixed
      # prismatic
  # child node
    <parent />
      # Description
      # Parameters: link=""
    <child />
      # Description:
      # Parameters: link=""
    <origin />
      # Description:
      # Parameters: xyz="0 -0.2 0.25"
    <limit />
      # Description
      # Parameters:
        # effort="1000.0"    maximum effort
        # lower="-0.38"     Joint upper limit (radians)
        # upper="0"         Joint lower limit (radians)
        # velocity="0.5"    Maximum velocity
    <axis />
      # Description: Press ? axis rotation
      # Parameters: xyz="0 0 1", along the Z axis
</material>
  # Description:
  # Parameters: name="blue"
  # child node:
    <color />
      # description:
      # Parameters: rgba="0 0 0.8 1"
```

- **Download source code**

```
cd ~/dev_ws
git clone -b ros2 https://github.com/ros/urdf_tutorial.git src/urdf_tutorial
```

- **Compile source code**

```
colcon build --packages-select urdf_tutorial
```

- **Run the example**

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf
```

Launch

The launch system in ROS 2 is responsible for helping users describe the configuration of their system and then executing it accordingly. The configuration of the system consists of the programs to run, where to run them, the arguments to pass to them, and ROS-specific conventions that make it easy to reuse components throughout the system by providing different configurations for each component. It is also responsible for monitoring the status of launched processes and reporting and/or responding to changes in the status of those processes.

Launch files written in Python, XML, or YAML can start and stop different nodes, as well as trigger and handle various events.

Specific reference: [Official Tutorial](#)

Setup

Create a new directory to store your launch file:

```
``bash mkdir launch
```

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel'),
            ]
        )
    ])

```

Run the ros2 launch file

To run the launch file created above, go to the directory you created earlier and run the following command:

The syntax format is:

```
ros2 launch <package_name> <launch_file_name>
```

4.1 First-time self-check

```
cd launch
ros2 launch turtlesim_mimic_launch.py
```

- **launch help**

```
ros2 launch -h
```

- **Run Node**

```
ros2 launch turtlesim multisim.launch.py
```

- **Check the parameters of the launch file**

```
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py -s
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py --show-arguments
ros2 launch turtlebot3_bringup robot.launch.launch.py -s
```

- **Run launch file with parameters**

```
ros2 launch turtlebot3_bringup robot.launch.launch.py usb_port:=/dev/ opencr
```

- **Run the node and debug**

```
ros2 launch turtlesim turtlesim_node.launch.py -d
```

- **Only output node description**

```
ros2 launch turtlesim turtlesim_node.launch.py -p
```

- **Run component**

```
ros2 launch composition composition_demo.launch.py
```

Run

run is used to run a single node, component program.

- **run help**

```
ros2 run -h
```

- **run node**

```
ros2 run turtlesim turtlesim_node
```

- **run node with parameters**

4.1 First-time self-check

```
ros2 run turtlesim turtlesim_node --ros-args -r __node:=turtle2 -r __ns:=/ns2
```

- **Run component container**

```
ros2 run rclcpp_components component_container
```

- **Run component** ```bash ros2 run composition manual_composition`

Package

A package can be thought of as a container for your ROS 2 code. If you want to be able to install your code or share it with others, you need to organize it in a Packages allow you to publish your ROS 2 work and allow others to easily build and use it.

Package creation in ROS 2 uses ament as its build system and colcon as its build tool. You can create packages using the officially supported CMake or Python, but other build types do exist.

Specific parameters: [Official tutorial](#)

Creating a Workspace

Create a new directory for each new workspace. The name is not important, but it helps indicate the purpose of the workspace. Let's choose the directory name `ros2_ws` for the "development workspace":

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
```

- **pkg help**

```
ros2 pkg -h
```

- **List function packages**

```
ros2 pkg executable turtlesim
```

- **Output a function package executable program**

```
ros2 pkg executable turtlesim
```

- **Create Python Packages**

Before running the package creation command, make sure you are in the `src` folder.

```
cd ~/ros2_ws/src
```

The command syntax for creating a new package in ROS 2 is:

4.1 First-time self-check

```
ros2 pkg create --build-type ament_python <package_name>
# You will use the optional parameter -- node-name Create a simple Hello World type executable file in the package.
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

- **Build package**

Put the package in the workspace Especially valuable because you can build many packages at once by running `colcon build` in the workspace root. Otherwise, you would have to build each package individually.

```
# Return to the workspace directory:
cd ~/ros2_ws# Now you can build your package:
colcon build
```

- **Source setup file**

To use your new package and executable, first open a new terminal and source your main ROS 2 installation.

Then, from the `ros2_ws` directory, run the following command to source your workspace:

```
source install/setup.bash
```

Now that your workspace is added to your path, you will be able to use your new package's executable.

- **Use package**

To run the executable you created during package creation using the `--node-name` parameter, enter the following command:

```
ros2 run my_package my_node
```

A brief introduction and use of rviz2

rviz is a three-dimensional visualization platform in ROS. On the one hand, it can realize the graphical display of external information. On the other hand, it can also release control information to objects through rviz, thereby realizing the monitoring and control of robots.

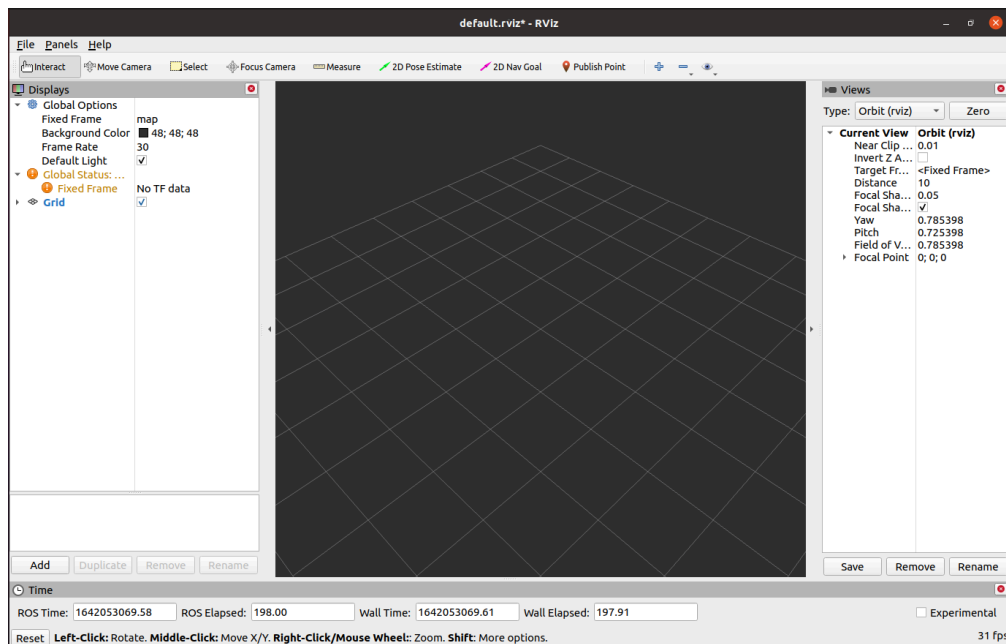
Introduction to rviz2

The successful installation of ros2 indicates that rviz2 is also successfully installed, because the installation of ros2 includes rviz2.

Open a new terminal (shortcut `Ctrl+Alt+T`) and enter the command to open rviz2

```
rviz2
```

Open rviz2 and the following interface will be displayed:



Introduction to each area

- On the left is the display list. The display is something that draws something in the 3D world and may have some options available in the display list.
- Above is the toolbar, which allows the user to select multiple functions with various function keys
- The middle part is the 3D view: it is the main screen for viewing various data in three dimensions. The background color, fixed frame, grid, 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.
- On the right is the observation angle setting area, where different observation angles can be set.

In this section, we only give a rough introduction. If you want to know more details, you can go to the [User Guide](#) to check it out.

mycobot_ros2 installation and update

`mycobot_ros2` is a ROS package launched by ElephantRobotics that is compatible with various types of desktop robotic arms.

Project address: https://github.com/elephantrobotics/mycobot_ros2

The official default workspace is `colcon_ws` .

Open the terminal and enter the following command:

```
cd ~/colcon_ws/src # Enter the src folder in the workspace
# Clone the code on github
git clone 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 recompiling every time you adju
source install/setup.bash # Add environment variables
```

Note: If `/home/er/colcon_ws/src` (equivalent to `~/colcon_ws/src`) directory, you need to delete the original `mycobot_ros2` first, and then execute the above command.

Simple use

Start through the launch.py file

This example is based on the fact that you have completed [Environment Setup](#) and successfully copied the company's code from GitHub.

Open a console terminal (shortcut key `Ctrl+Alt+T`) Enter the following command to **ROS2 environment configuration**.

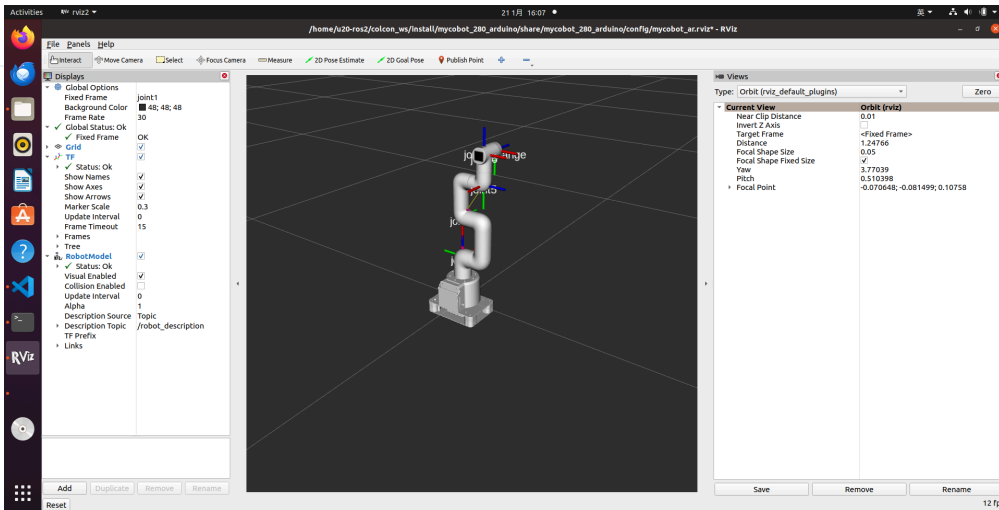
```
cd ~/colcon_ws
colcon build --symlink-install
source install/setup.bash
```

Then enter the following command to run:

```
ros2 launch mycobot_280_arduino test.launch.py
```

Open `rviz2` and get the following result:

4.1 First-time self-check



If you want to learn more about rviz, you can go to the [official document](#) to view it

M5/Arduino version prerequisites

- Open the console terminal (shortcut key `Ctrl+Alt+T`), open the terminal window to view the device name:

```
# View the device name of the robot arm
ls /dev/ttyUSB* # Old version myCobot280 M5

# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # New version myCobot280 M5
```

- Grant serial port permissions to the robot:

```
# The default device name is /dev/ttyUSB0. If the device name is not the default value, you need to modify it.
sudo chmod 777 /dev/ttyUSB0 # Old version myCobot280 M5

sudo chmod 777 /dev/ttyACM0 # New version myCobot280 M5
```

Then enter the user password (**Note:** The password is not displayed, just enter it correctly).

280 series rviz user guide

Robot arm control

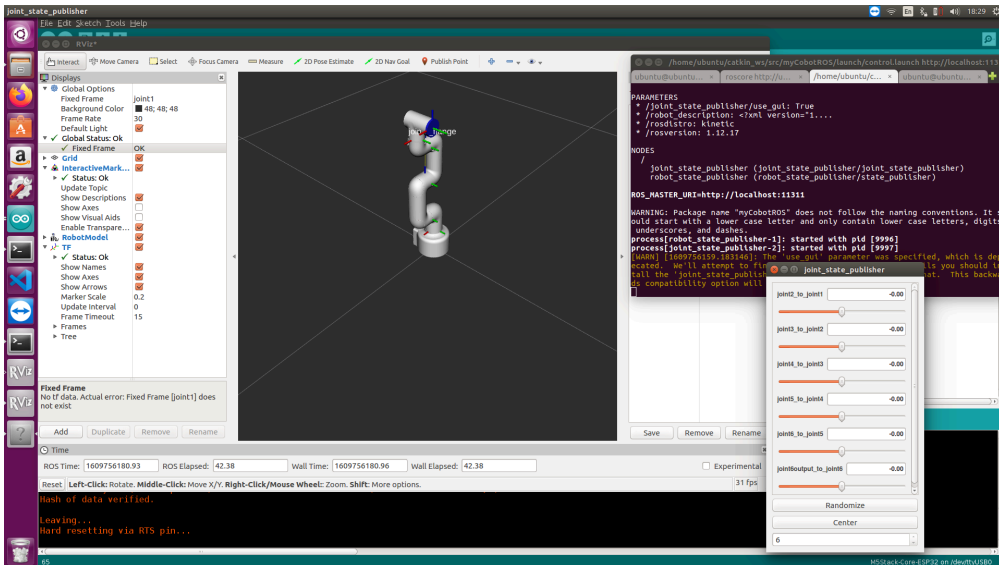
1 Slider control

Open a command line and run:

```
# The default serial port name of mycobot 280-Arduino version is "/dev/ttyUSB0" and the baud rate is 1000000. The serial p
ros2 launch mycobot_280_arduino slider_control.launch.py
```

4.1 First-time self-check

It will **open rviz** and a **slider component**, and you will see the following screen (the screen of the Raspberry Pi version is slightly different, which does not affect the use):



Then you can **control the movement of the model in rviz by dragging the slider**. The real mycobot will move with it.

Please note: Since the robot arm will move to the current position of the model when the command is entered, please make sure that the model in rviz does not appear to be through the model before you use the command. Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm.

2 Model following

In addition to the above control, we can also **let the model follow the movement of the real robot arm**. Open a command line and run:

```
# The default serial port name of mycobot 280-Arduino version is "/dev/ttyUSB0" and the baud rate is 1000000. The serial p
ros2 launch mycobot_280_arduino mycobot_follow.launch.py
```

It will **open rviz to show the model following effect**. At this time, manually drag the real robot arm, and the model in rviz will follow the movement.

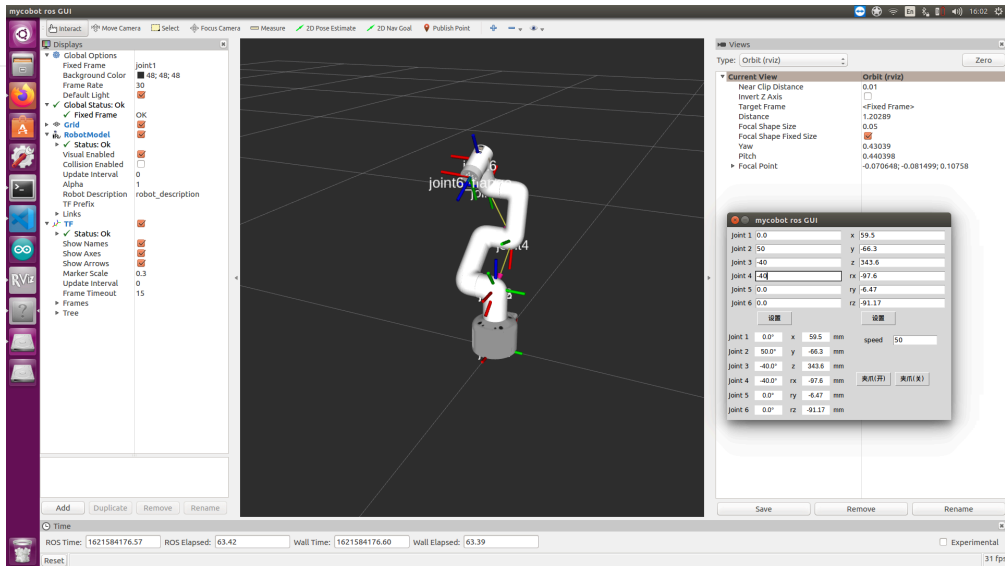
3 GUI control

Based on the previous, this package also **provides a simple Gui control interface**. This method is intended to allow real robotic arms to interact with each other. Please connect mycobot.

Open the command line:

```
# The default serial port name of mycobot 280-Arduino version is "/dev/ttyUSB0" and the baud rate is 1000000. The serial p
ros2 launch mycobot_280_arduino simple_gui.launch.py
```

4.1 First-time self-check



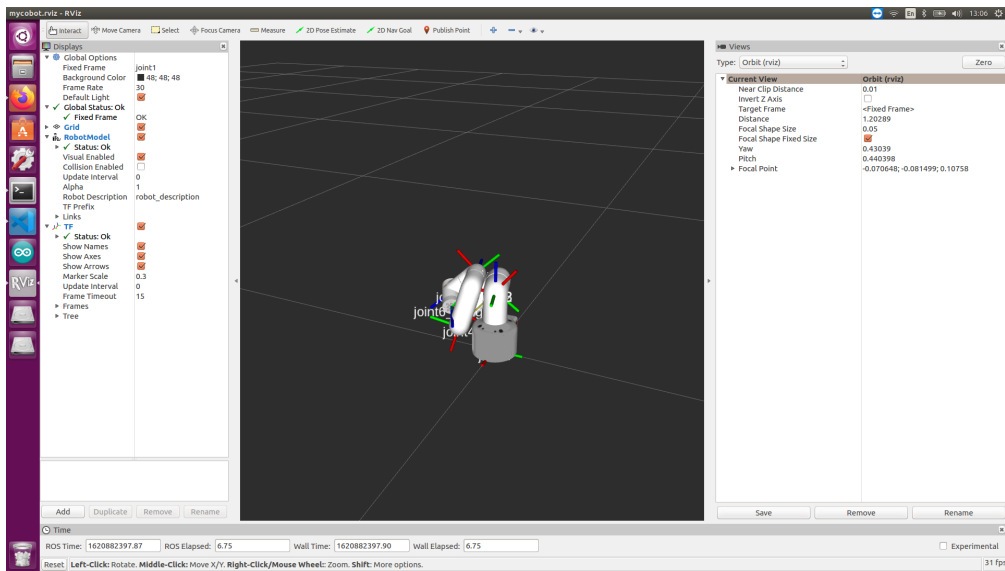
4 Keyboard control

Keyboard control function has been added to the `mycobot_280_arduino` package, and it is synchronized in real time in rviz. This function relies on pythonApi, so make sure it is connected to the real robot arm.

Open a command line and run:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 1000000. The serial port name is  
ros2 launch mycobot_280_arduino teleop_keyboard.launch.py
```

The running effect is as follows:



Mycobot information will be output on the command line, as follows:

4.1 First-time self-check

```
[INFO] [launch]: All log files can be found below /home/elephant/.ros/log/2022-05-19-16-25-45-949761-elephant-virtual-mac
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [robot_state_publisher-1]: process started with pid [19114]
[INFO] [rviz2-2]: process started with pid [19116]
[robot_state_publisher-1] Parsing robot urdf xml string.
[robot_state_publisher-1] Link joint2 had 1 children
[robot_state_publisher-1] Link joint3 had 1 children
[robot_state_publisher-1] Link joint4 had 1 children
[robot_state_publisher-1] Link joint5 had 1 children
[robot_state_publisher-1] Link joint6 had 1 children
[robot_state_publisher-1] Link joint6_flange had 0 children
[robot_state_publisher-1] [INFO] [1737443676.433771864] [robot_state_publisher]: got segment joint1
[robot_state_publisher-1] [INFO] [1737443676.433899995] [robot_state_publisher]: got segment joint2
[robot_state_publisher-1] [INFO] [1737443676.433912168] [robot_state_publisher]: got segment joint3
[robot_state_publisher-1] [INFO] [1737443676.433921515] [robot_state_publisher]: got segment joint4
[robot_state_publisher-1] [INFO] [1737443676.433930673] [robot_state_publisher]: got segment joint5
[robot_state_publisher-1] [INFO] [1737443676.433939720] [robot_state_publisher]: got segment joint6
[robot_state_publisher-1] [INFO] [1737443676.433948907] [robot_state_publisher]: got segment joint6_flange
[rviz2-2] [INFO] [1737443677.010176632] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] [INFO] [1737443677.010780016] [rviz2]: OpenGL version: 3.1 (GLSL 1.4)
[rviz2-2] [INFO] [1737443677.134247803] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] Parsing robot urdf xml string.
[listen_real-3] [INFO] [1737443677.525803631] [listen_real]: port://dev/ttyUSB0, baud:1000000
```

Then run the command:

```
ros2 run mycobot_280_arduino teleop_keyboard
```

You will see the following output in the command line:

4.1 First-time self-check

```
Mycobot Teleop Keyboard Controller
-----
Moving options(control coordinations [x,y,z,rx,ry,rz]):
w(x+)

a(y-) s(x-) d(y+)

z(z-) x(z+)

u(rx+) i(ry+) o(rz+)

j(rx-) k(ry-) l(rz-)

Gripper control:
g - open
h - close

Other:
1 - Go to init pose
2 - Go to home pose
3 - Resave home pose
q - Quit

currently: speed: 10 change percent: 2
```

In this terminal, you can control the state of the robot and move the robot by pressing keys in the command line.

5 End effector

- **Supported end effectors:** myCobot vertical suction pump V2.0, camera flange
- **Applicable devices:** myCobot 280 M5, myCobot 280 PI

5.1 myCobot vertical suction pump V2.0

1 Load the model

Open a command line and run:

- mycobot 280-M5 version:

```
ros2 launch mycobot_280 test_pump.launch.py
```

- mycobot 280-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the bar below the desktop to open the ROS2 environment terminal:

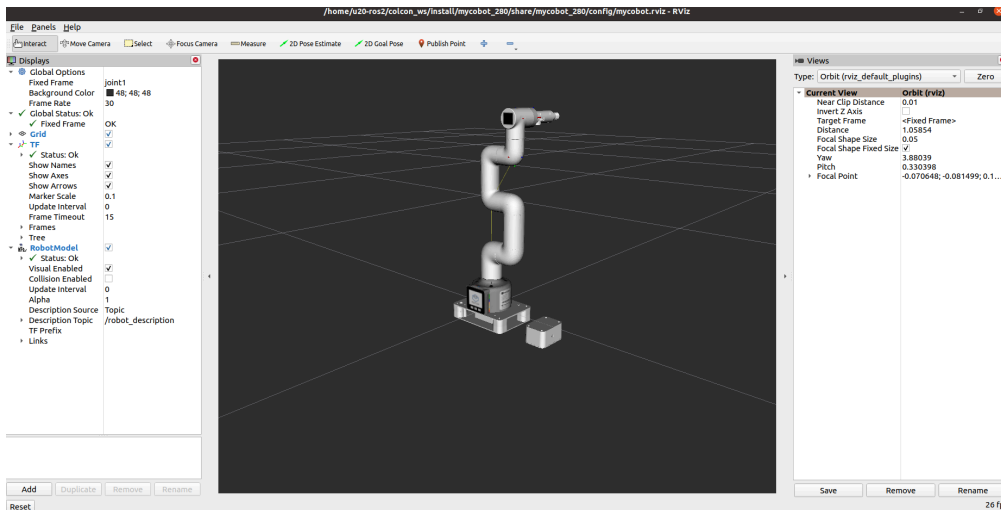
4.1 First-time self-check



Then run the command:

```
ros2 launch mycobot_280pi test_pump.launch.py
```

It will **open rviz**, and you will see the following screen (the screen of the Raspberry Pi version is slightly different, which does not affect the use):



2 Slider control

Note: This function only supports the control of the robot arm

Open a command line and run:

- mycobot 280-M5 version:

```
# mycobot The default serial port name of the 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port  
ros2 launch mycobot_280 slider_control_pump.launch.py
```

- mycobot 280-PI version:

4.1 First-time self-check

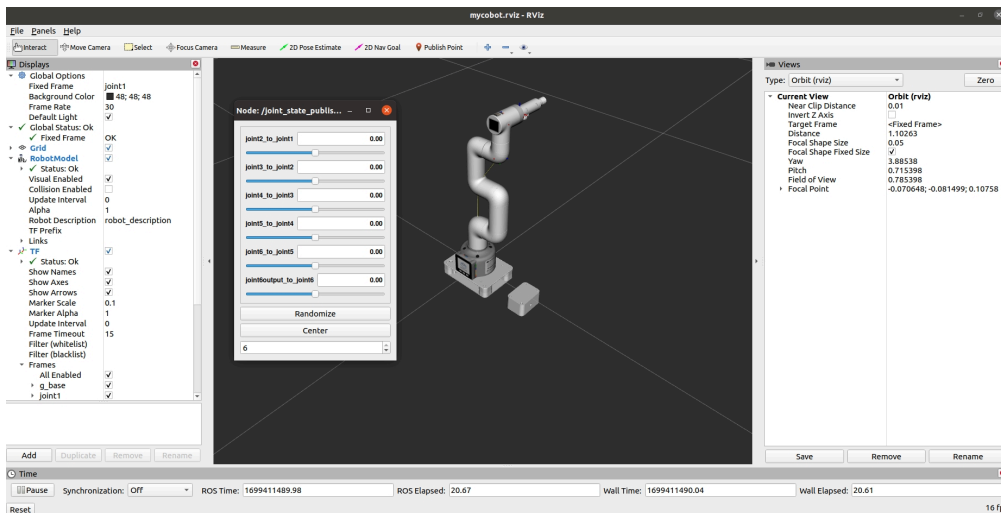
Click the ROS2 Shell icon on the desktop or the corresponding icon in the bar below the desktop to open the ROS2 environment terminal:



Then run the command:

```
# The default serial port name of mycobot 280-PI version is "/dev/ttyAMA0" and the baud rate is 1000000.  
ros2 launch mycobot_280pi slider_control_pump.launch.py
```

It will open rviz and a slider component, you will see the following screen (the Raspberry Pi version screen is slightly different, which does not affect the use):



Then you can control the movement of the model in rviz by dragging the slider. The real mycobot will move with it.

Please note: Since the robot arm will move to the current position of the model while the command is input, please make sure that the model in rviz does not appear to be through the model before you use the command Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm

4.1 First-time self-check

3 GUI control

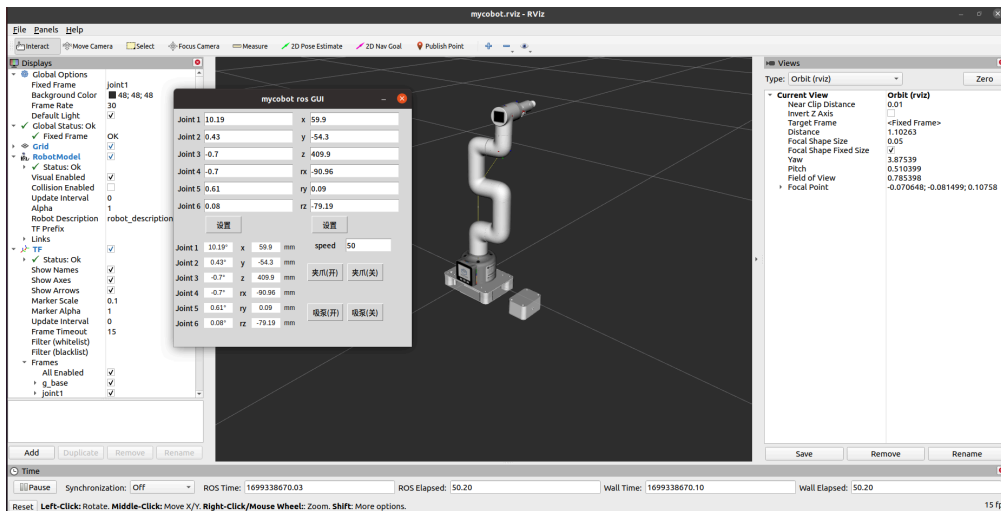
Based on the previous, this package also **provides a simple Gui control interface**. This method is intended for real robot arms to be linked to each other. Please connect mycobot.

Open the command line:

- mycobot 280-M5 version:

```
# The default serial port name of mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port name is  
ros2 launch mycobot_280 simple_gui_pump.launch.py
```

It will open rviz and a GUI interface, and you will see the following screen:



5.2 Camera flange

1 Load the model

Open a command line and run:

- mycobot 280-M5 version:

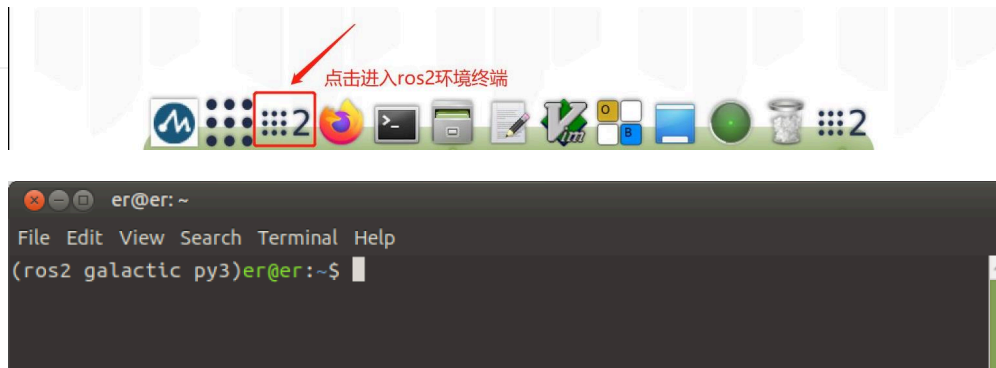
```
ros2 launch mycobot_280 test_camera_flange.launch.py
```

- mycobot 280-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the bar below the desktop to open the ROS2 environment terminal:



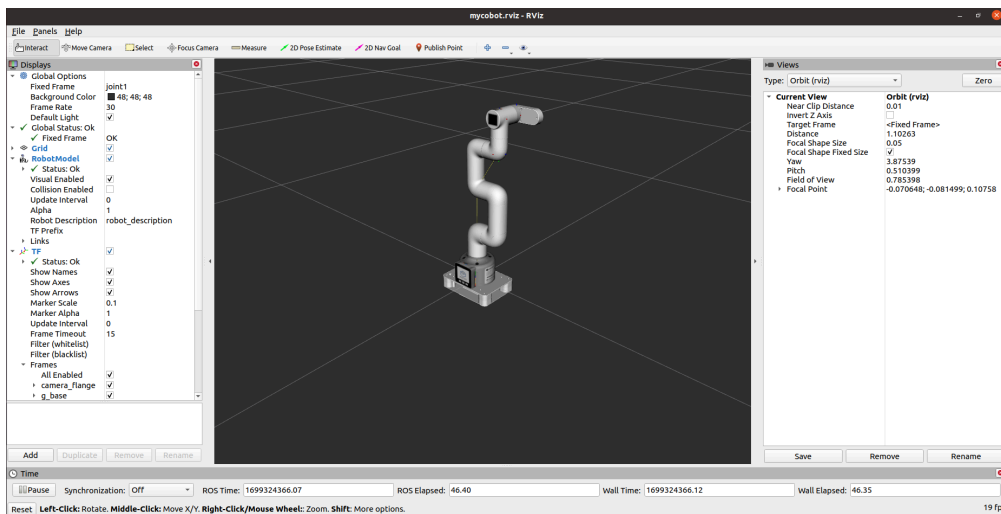
4.1 First-time self-check



Then run the command:

```
ros2 launch mycobot_280pi test_camera_flange.launch.py
```

It will **open rviz**, you will see the following screen (the screen of the Raspberry Pi version is slightly different, which does not affect the use):



2 Slider control

Note: This function only supports the control of the robot arm

Open a command line and run:

- mycobot 280-M5 version:

```
# The default serial port name of the mycobot 280-M5 version is "/dev/ttyUSB0" and the baud rate is 115200. The serial port  
ros2 launch mycobot_280 slider_control_camera_flange.launch.py
```

- mycobot 280-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the bar below the desktop to open the ROS2 environment terminal:

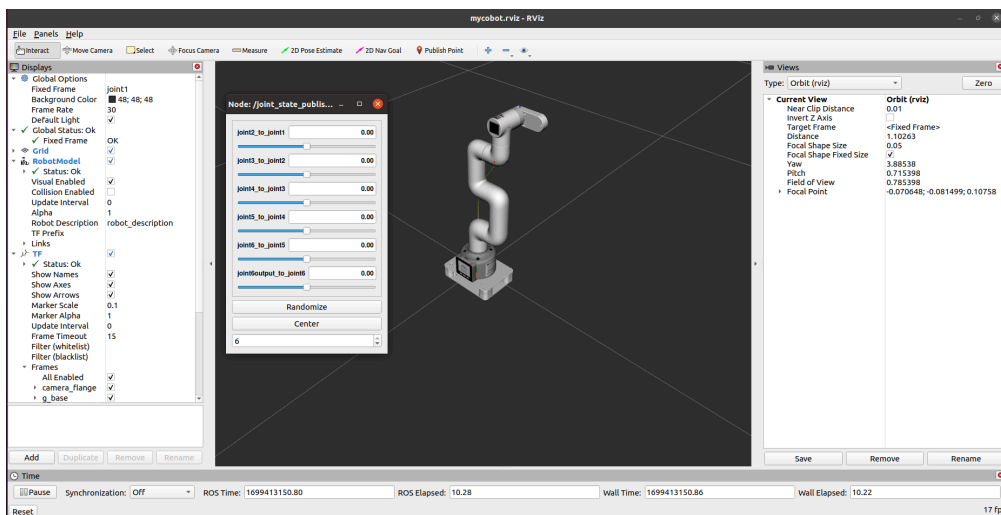
4.1 First-time self-check



Then run the command:

```
# The default serial port name of mycobot 280-PI version is "/dev/ttyAMA0" and the baud rate is 1000000.  
ros2 launch mycobot_280pi slider_control_camera_flange.launch.py
```

It will open rviz and a slider component, and you will see the following screen (the screen of the Raspberry Pi version is slightly different, which does not affect the use):



Then you can control the movement of the model in rviz by dragging the slider. The real mycobot will move with it.

Please note: Since the robot arm will move to the current position of the model when the command is entered, please make sure that the model in rviz does not appear to be through the model before you use the command. Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm.

5.3 Camera flange & pump

1 Load the model

Open a command line and run:

4.1 First-time self-check

- mycobot 280-M5 version:

```
ros2 launch mycobot_280 test_camera_flange_pump.launch.py
```

- mycobot 280-PI version:

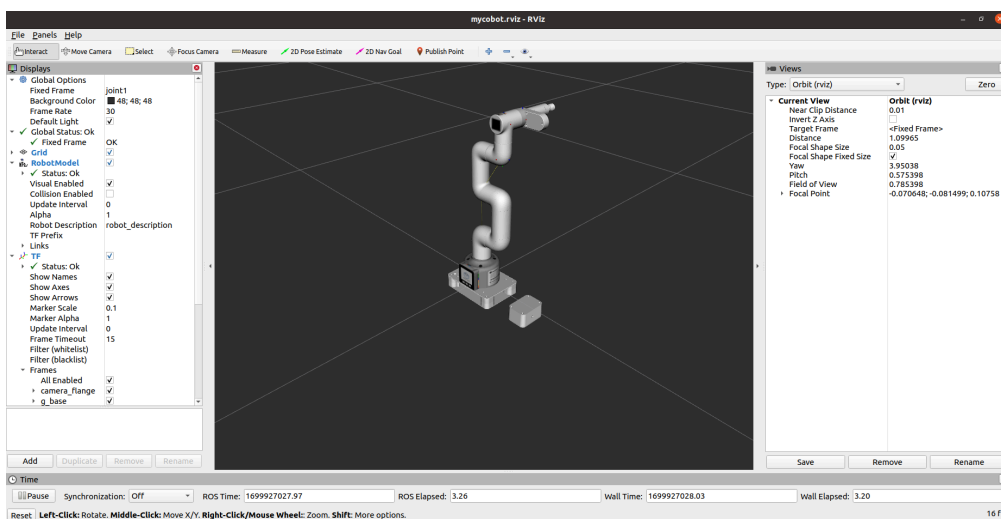
Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the bar below the desktop to open the ROS2 environment terminal:



Then run the command:

```
ros2 launch mycobot_280pi test_camera_flange_pump.launch.py
```

It will **open rviz**, you will see the following screen (the screen of the Raspberry Pi version is slightly different, which does not affect the use):



5.3 URDF model address

1 myCobot vertical suction pump V2.0

- [myCobot 280-M5 version](#)
- [myCobot 280-PI version](#)

2 Camera flange

- [myCobot 280-M5 Version](#)
- [myCobot 280-PI Version](#)

3 Camera Flange && Pump

- [myCobot 280-M5 Version](#)
- [myCobot 280-PI version](#)

Movelt2

Introduction to Movelt2

Movelt2 is an integrated development platform in ROS2, consisting of a variety of function packages for manipulating robotic arms, including: motion planning, manipulation, control, inverse kinematics, 3D perception, and collision detection.

Core functions

1. **Motion Planning** Movelt 2 provides motion planning capabilities based on OMPL (Open Motion Planning Library) and other third-party libraries, supporting a variety of planning algorithms and constraints.
2. **Inverse Kinematics (IK)** Movelt 2 uses a plug-in mechanism to support different IK solvers, which can quickly calculate the target pose of the robotic arm.
3. **Collision Detection & Avoidance** Movelt 2 has built-in powerful collision detection capabilities to ensure that the robot does not collide with the surrounding environment when planning and executing movements.
4. **Dynamic Scene Awareness** Movelt 2 supports dynamic updates of the environment model and can sense changes in obstacles in real time.
5. **Control & Execution** Movelt 2 provides a motion control interface that is tightly integrated with the robot hardware to ensure that the planned path can be accurately executed.
6. **Visualization Tools** Integrated with RViz 2, it supports intuitive interaction and debugging functions, and can display the motion planning and execution process in real time.

Advantages of Movelt 2

- **Real-time support based on ROS 2** The DDS communication architecture of ROS 2 enables Movelt 2, which has significantly improved real-time performance and reliability.
- **Modular design** Movelt 2 adopts a modular architecture, which supports users to load or replace modules as needed, providing great flexibility.
- **Cross-platform support** Movelt 2 supports running on multiple operating systems (such as Ubuntu, Windows) and hardware platforms.
- **Active Community Support** Movelt 2 has a global developer community that continuously provides updates, feature extensions, and technical support.

Configuration

1. URDF - Universal Robot Description Format.
2. SRDF - includes the robot's joint groups, virtual and passive joints, robot poses, self-collisions, and is usually created by the user using the Movelt2 Setup Assistant.
3. Movelt2 Configuration - includes joint limits, kinematics, motion planning, perception, and other information. Configuration files for these components are automatically generated by the Movelt2 Setup Assistant ([Movelt2 Configuration Assistant](#)) and stored in the configuration directory of the robot's corresponding Movelt2 configuration package.

How to use MoveIt2

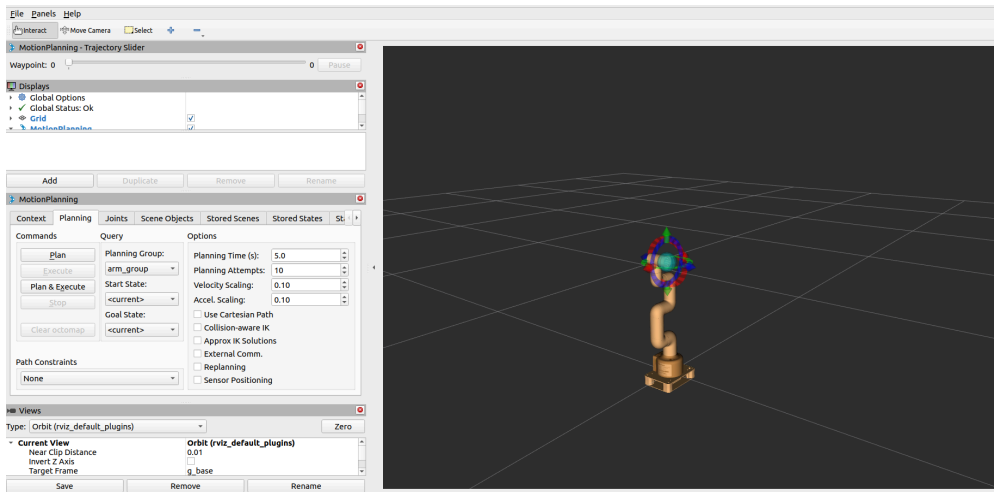
Note: For better motion planning, the Atom firmware version of the end-arm is 6.5, and the python driver library pymycobot version is 3.5.3

mycobot_ros2 has now integrated the MoveIt2 part.

Open the command line and run:

```
ros2 launch mycobot_280arduino_moveit2 demo.launch.py
```

The operation effect is as follows:



If you need to let the real robot arm execute the plan synchronously, you need to open another command line and run:

```
# The default serial port name of mycobot 280-Arduino version is "/dev/ttyUSB0" and the baud rate is 1000000. The serial p  
ros2 run mycobot_280_moveit2_control sync_plan_arduino
```

Case effect demonstration reference 280-M5 version:

4.1 First-time self-check



What is myBlockly?



myBlockly is a fully visual modular programming software, a graphical programming language.

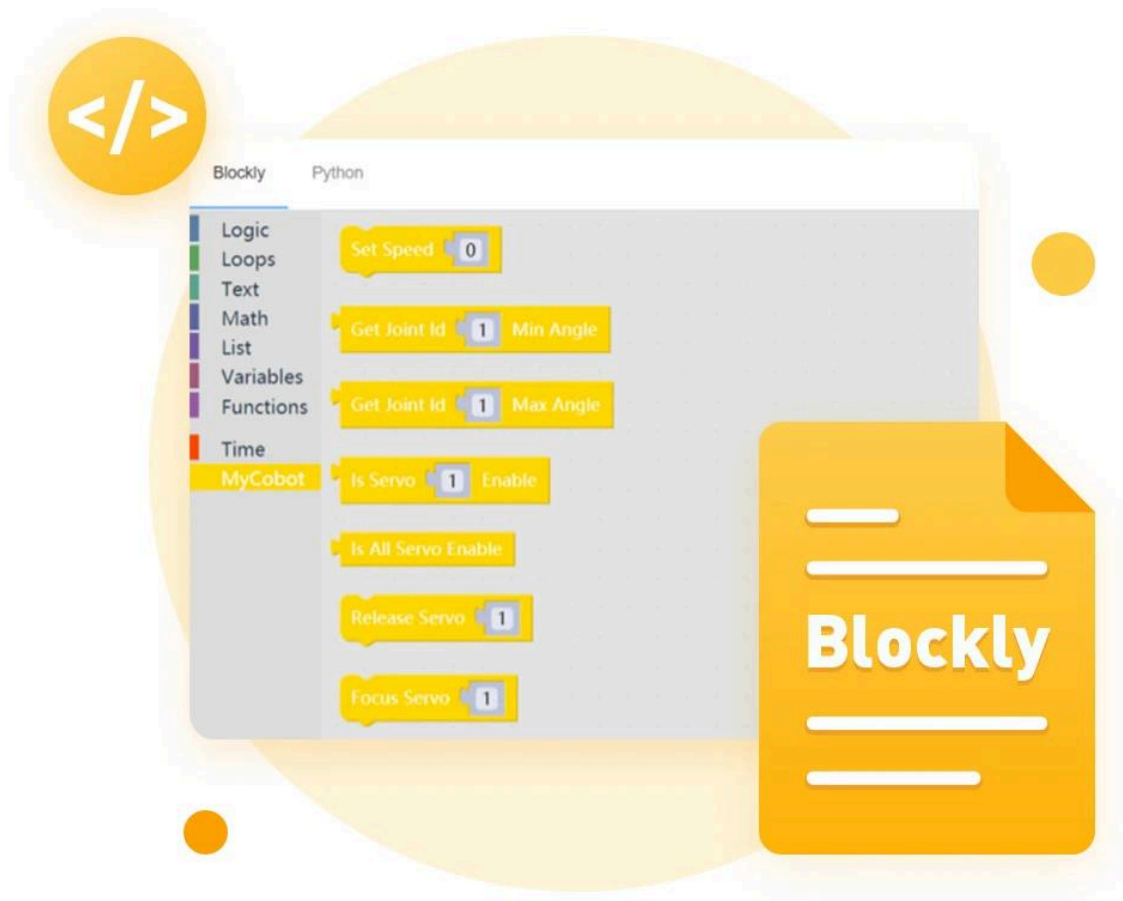
myBlockly is similar to MIT's children's programming language Scratch in terms of function/design.

When using **myBlockly**, users can drag and drop modules to build code logic, which is very similar to building blocks.

From the user's perspective, **myBlockly** is a simple and easy-to-use visual tool for generating code. From the developer's perspective, **myBlockly** is a text box that contains the code entered by the user.

The process of generating code into the text box is the process of the user dragging in **myBlockly**.

myBlockly installation



myBlockly download address:

- GitHub address: <https://github.com/elephantrobotics/myblockly-package/releases>
- Official website address: <https://www.elephantrobotics.com/download/>

Applicable devices:

- myCobot 280
- myCobot 280 M5
- myCobot 280 PI
- myCobot 280 Jetson Nano
- myCobot 280 for Arduino
- myCobot 320
- myCobot 320 M5
- myCobot 320 PI
- myPalletizer 260
- mechArm-270
- mechArm-270 M5
- mechArm-270 PI

Prerequisites:

4.1 First-time self-check

- **M5** series version, **M5Stack-basic** burn **miniRobot** at the bottom, select **Transponder** function, **ATOM** burn the latest version of **atomMain** at the end (factory default burned)
- **Pi \ jetsonnano** series, **ATOM** burn the latest version of **atomMain** (factory default burned)
- Configure **Python environment**, please refer to the [Python ▶](#) chapter

myBlockly Development and Use Guide

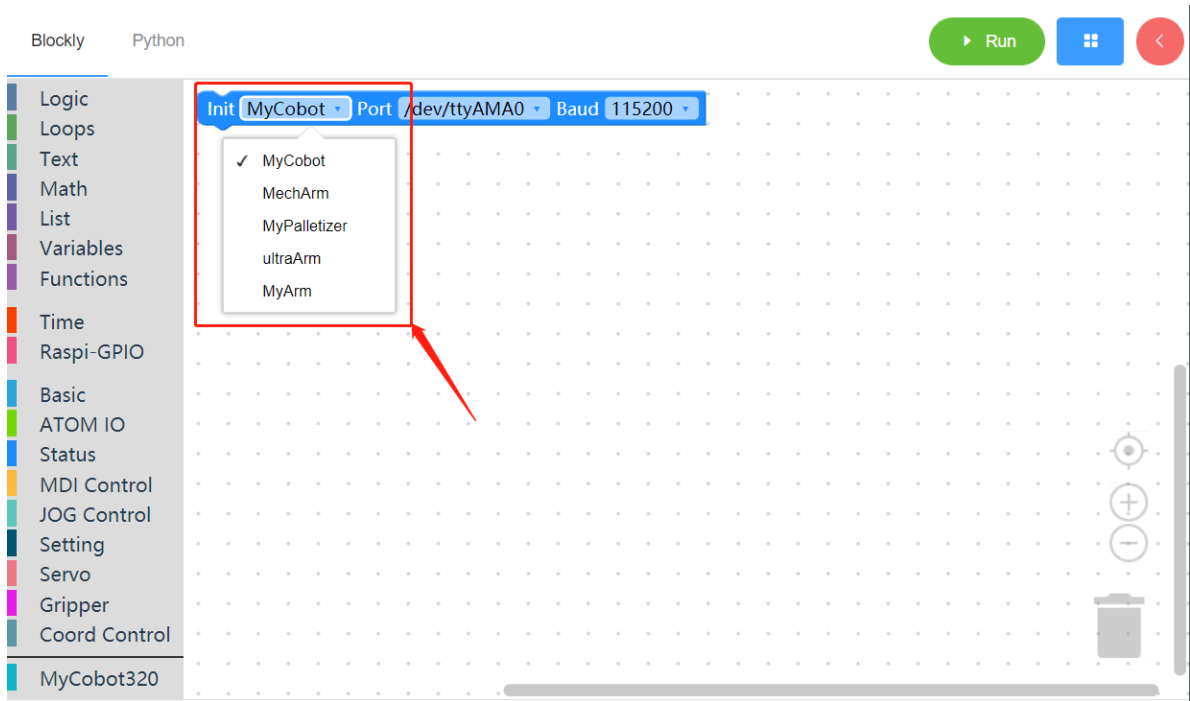
You can use myBlockly to develop our robotic arm according to the following guidelines

1. [MyBlockly Initial Use](#)
2. [Control RGB Light Board](#)
3. [Control the Robotic Arm to Return to Origin](#)
4. [Control Single Joint Movement](#)
5. [Control Multiple Joints](#)
6. [Control the robot arm to swing left and right](#)
7. [Control the robot arm to dance](#)
8. [Use of gripper](#)
9. [Use of suction pump](#)
10. [Gripper test](#)
11. [IO test](#)
12. [Q&A](#)

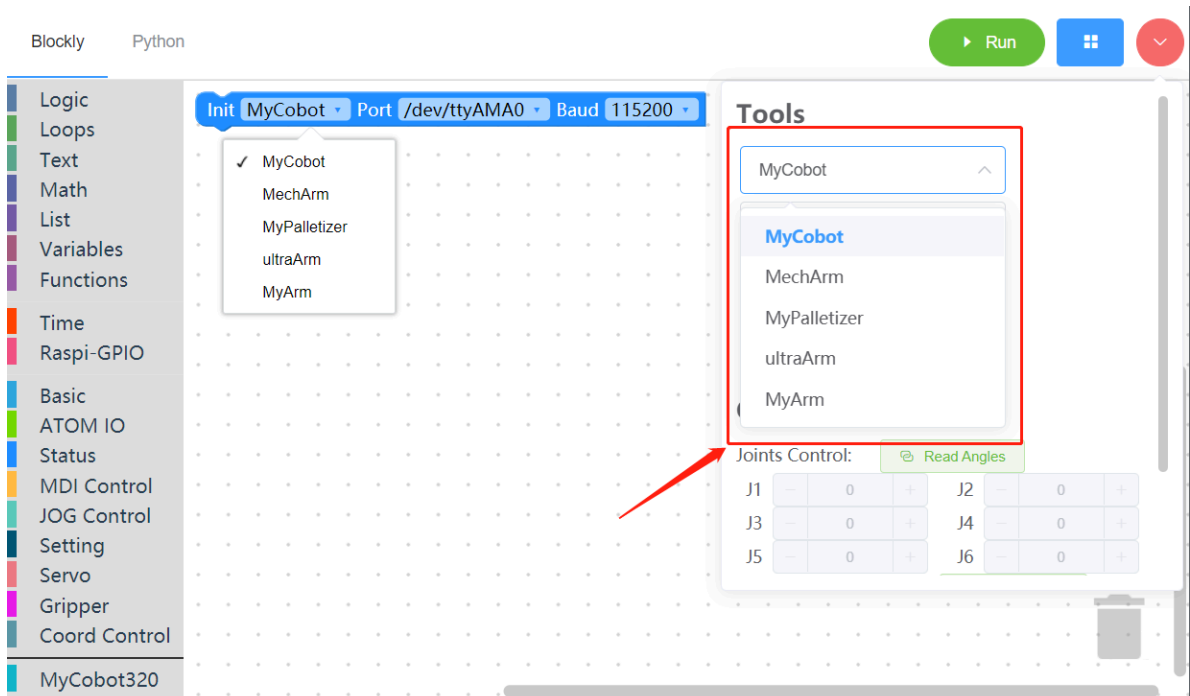
Initial use of myBlockly

Prerequisites

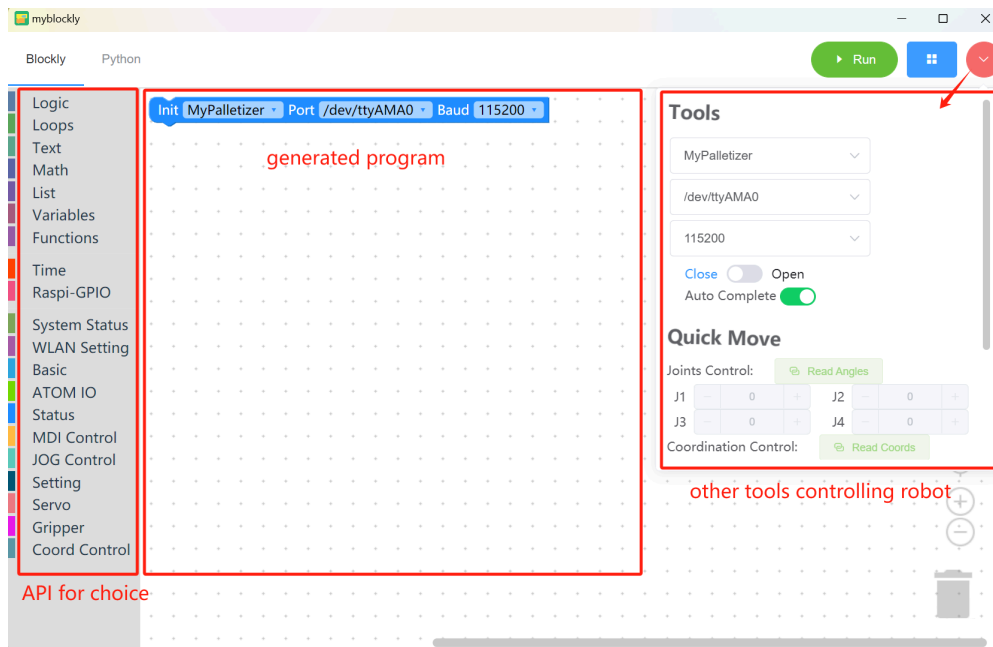
- Before you start programming, you must select the corresponding **machine model**, otherwise it is easy to cause hardware damage



- When controlling the machine with the control panel, you must select the corresponding **machine model**, otherwise it is easy to cause hardware damage



myBlockly interface display



- Module bar:
- Contains the method modules required for program writing, which can be placed in the program editing area for splicing by mouse
- MyCobot320 module
- Small toolbar:

Click the pink button in the upper right corner to display a small toolbar, where you can select the correct model, serial port number and baud rate. You can also get the real-time joint angle and coordinates of the robot arm by clicking the "Read Angle" or "Read Coordinate" button. Click "+/-" in the joint control or coordinate control bar to control the movement of the robot arm.

- Program editing area:
- Before running the program, you need to select the correct model, port and baud rate in the initialization module or the small toolbar, otherwise the program will not run normally.
- Drag the required module methods to this area and splice them to realize your own program.
- If the baud rate and serial port have been modified in the right toolbar, but it is still /dev/ttyAMA0, it is due to the myBlockly version. You need to update the software version on the official website first (the latest version will change the information in the editing area after selecting the serial port and baud rate in the toolbar).

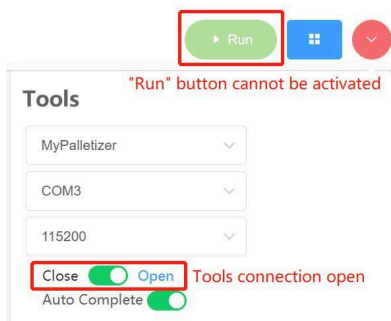
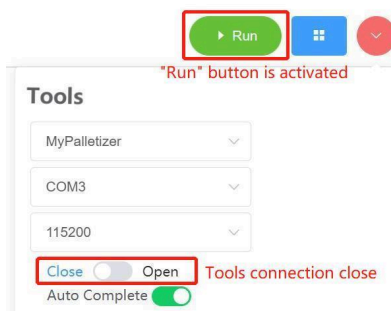
Note:

1. The baud rate of the M5Stack series is generally 115200, and the baud rate of the Raspberry Pi series is generally 1000000.

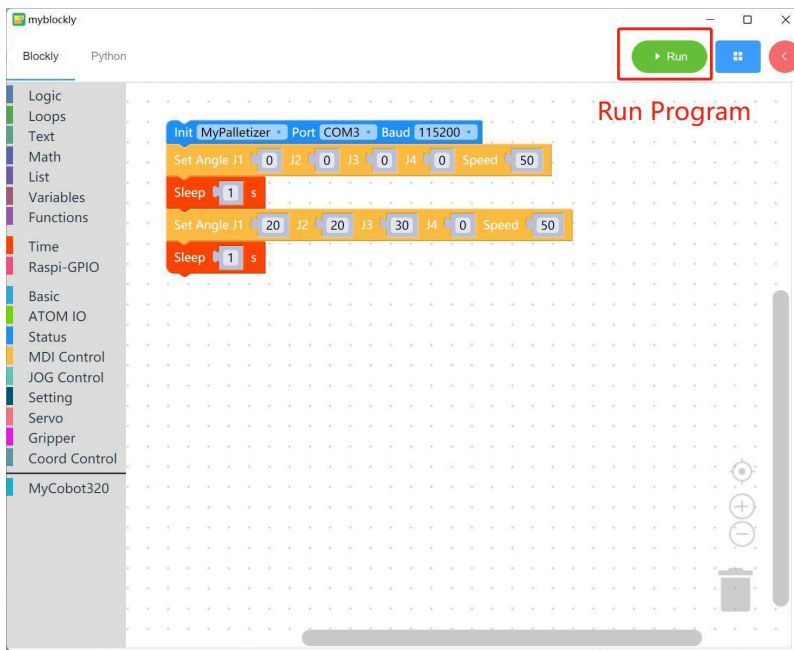
4.1 First-time self-check

| Machine model | Serial port number | Baud rate |
|-----------------|---|-----------|
| 260 M5 | Win: COM; <i>Linux: /dev/ttyUSB;</i> | 115200 |
| 270 M5 | Win: COM; <i>Linux: /dev/ttyUSB;</i> | 115200 |
| 280 M5 | Win: COM; <i>Linux: /dev/ttyUSB;</i> | 115200 |
| 320 M5 | Win: COM; <i>Linux: /dev/ttyUSB;</i> | 115200 |
| 260 PI | /dev/ttyAMA0 | 1000000 |
| 270 PI | /dev/ttyAMA0 | 1000000 |
| 280 PI | /dev/ttyAMA0 | 1000000 |
| 320 PI | /dev/ttyAMA0 | 115200 |
| 280 Jetson Nano | /dev/ttyTHS1 | 1000000 |
| 280 Arduino | Win: COM; <i>Linux: /dev/ttyUSB or /dev/ttyACM* ;</i> | 1000000 |

1. To check the serial port number and baud rate of the machine, please go to **myStudio Driver Installation** section.
2. When the program cannot run, please check whether the small toolbar is disconnected (as shown in the figure below).



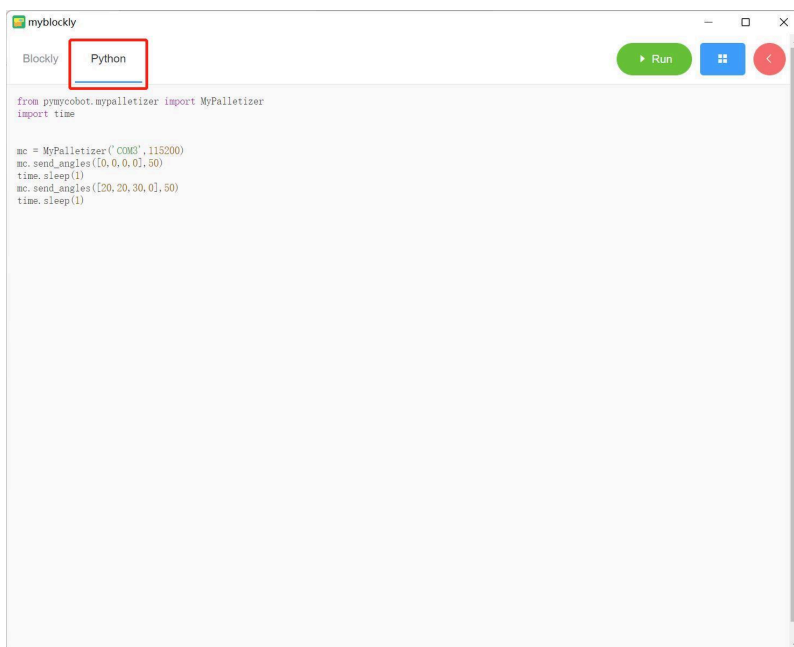
Program Run



Drag the desired method module, edit your own program (as shown above), combine each module structure together (there is a ki sound), and then click "Run" to upload the code to the robot arm to run.

Note: The program that operates the robot arm movement takes time to complete, so after an action, you need to connect a `sleep` module to give the robot arm time to move before the next movement. (Decide the required time according to the situation. The robot arm is set by default to run myBlockly with a minimum sleep time of no less than 0.5s) Otherwise, the robot arm will not be able to achieve the desired movement.

Click the "Python" option in the upper left corner to view the corresponding Python code, as shown in the figure below.



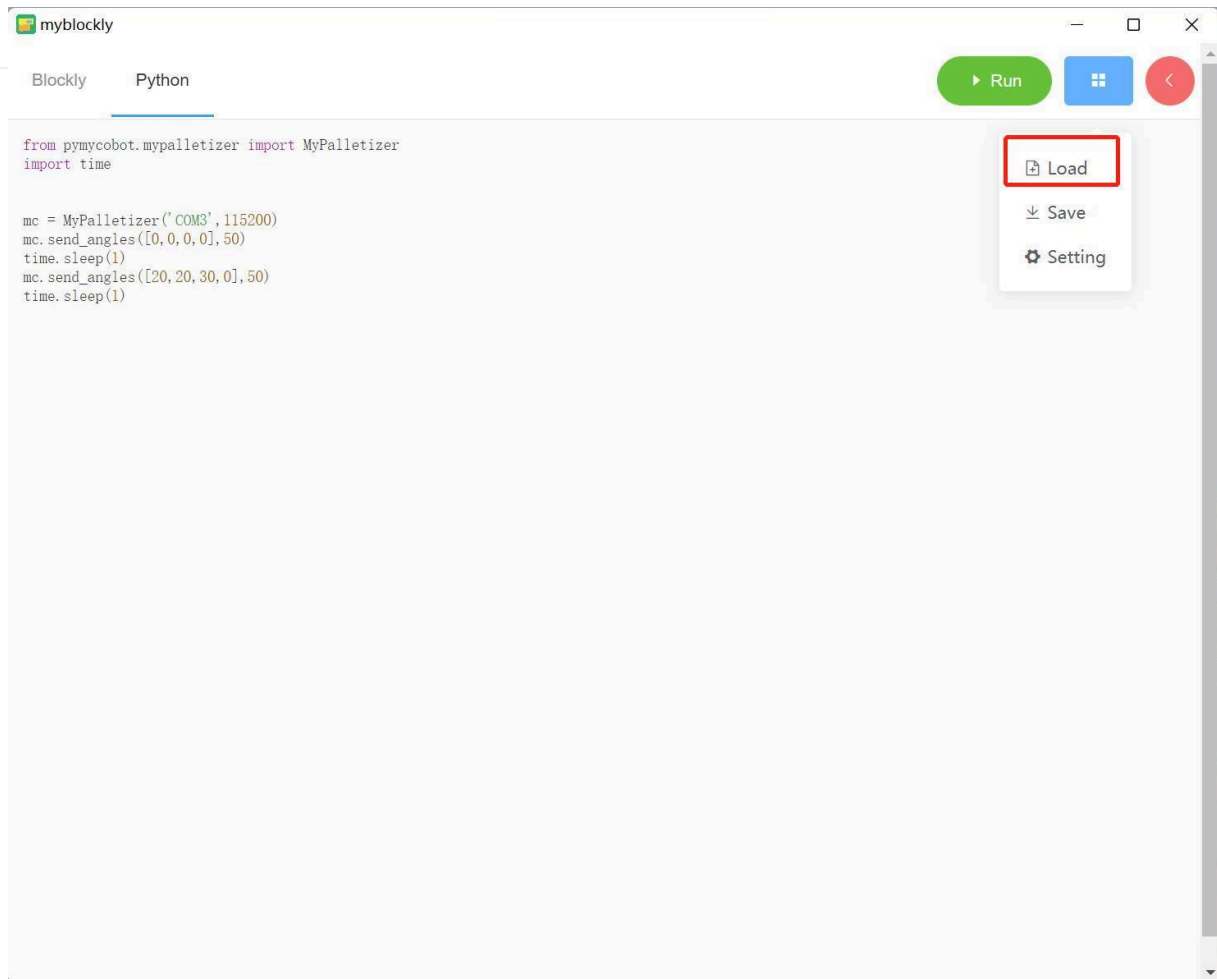
Program Saving and Loading

MyBlockly programs are saved in *.json format. Click the blue box in the upper right corner of the interface. Click the "Save" option to save the program.



Similarly, click the blue box and click the "Load" option to import the saved program.

4.1 First-time self-check

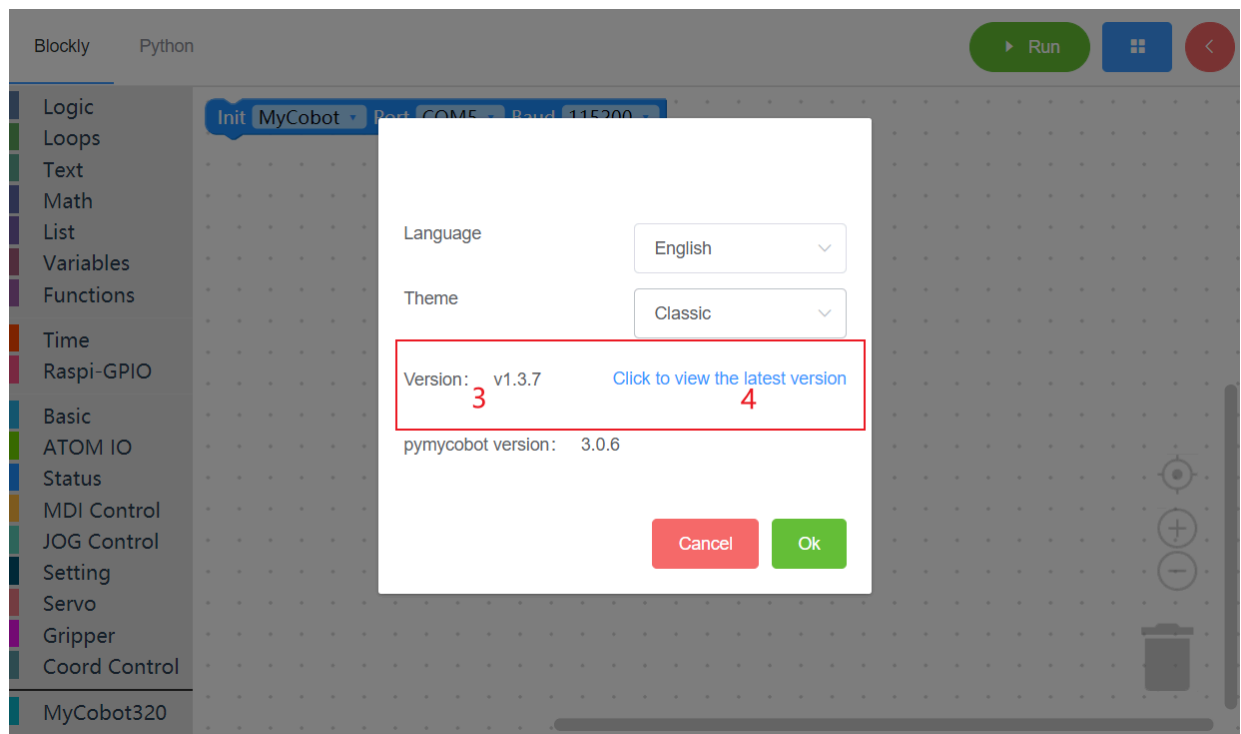
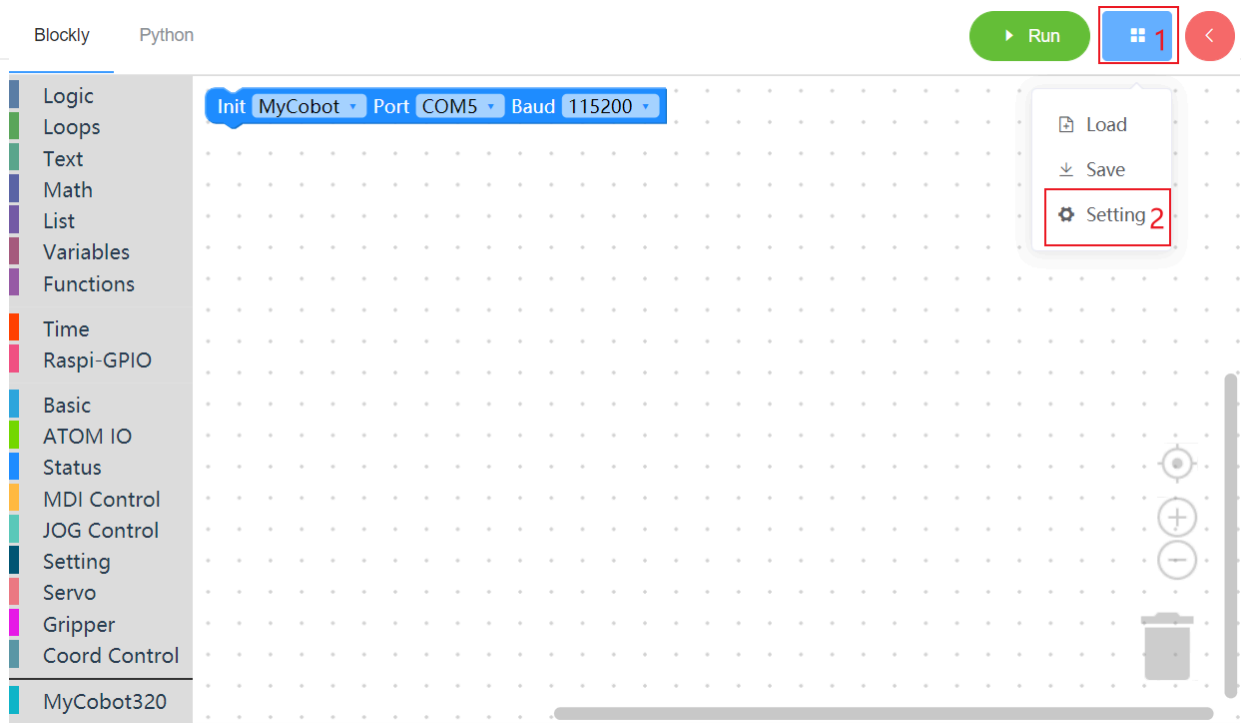


How to install and update the software

The installation and update of myBlockly need to go to the [official website](#) to download the latest version.

Inside the software, you can update the software through the following steps.

4.1 First-time self-check



Control RGB light board

Preparation before starting

M5Stack series: Make sure the robot arm is connected to the computer

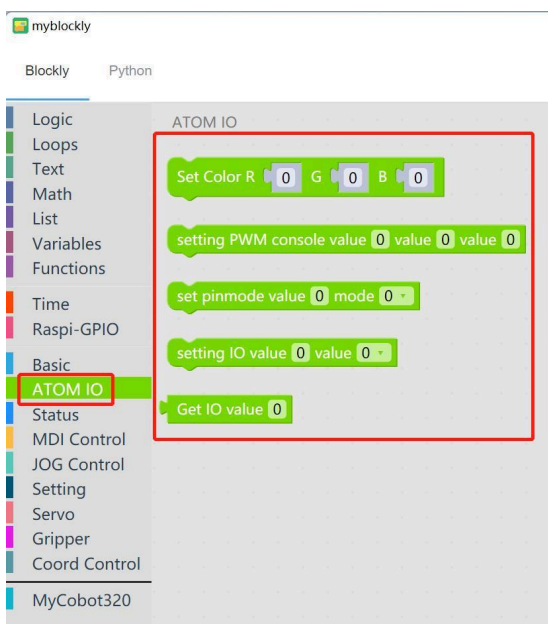
Other series: Make sure the machine is normal

Learning content in this chapter

How to use myBlockly to control the RGB light board

API introduction

- Method module: Color setting



- Scope of application: myCobot 280 series, myCobot 320 series, myCobot Pro 600 series and myPalletizer 260 series



- Parameter introduction:
 - The parameters to be set are $R(x)$, $G(x)$, $B(x)$, and different values represent different colors.
 - Parameter range (for details, please refer to the RGB parameter table):

R: 0~255

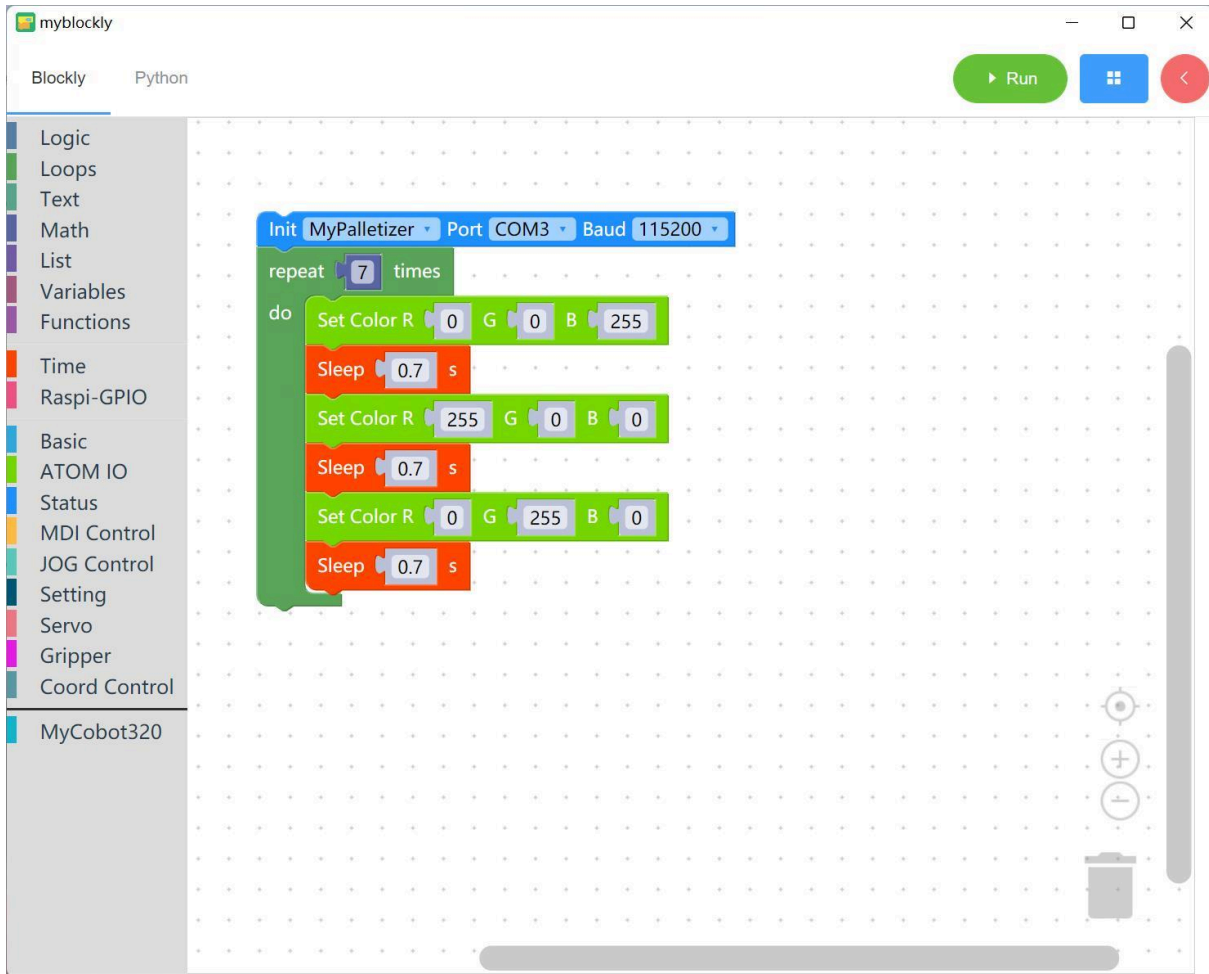
G: 0~255

B: 0~255

- Purpose: Control the color of the RGB light board.

Simple demonstration

- Graphic code is as follows:



- Implemented content:

Control the color of the robot arm RGB light board to change from "blue-red-green" in sequence, and the whole process is repeated seven times.

Control the robot arm to return to the origin

Preparation before starting

M5Stack series: Make sure the robot arm is connected to the computer

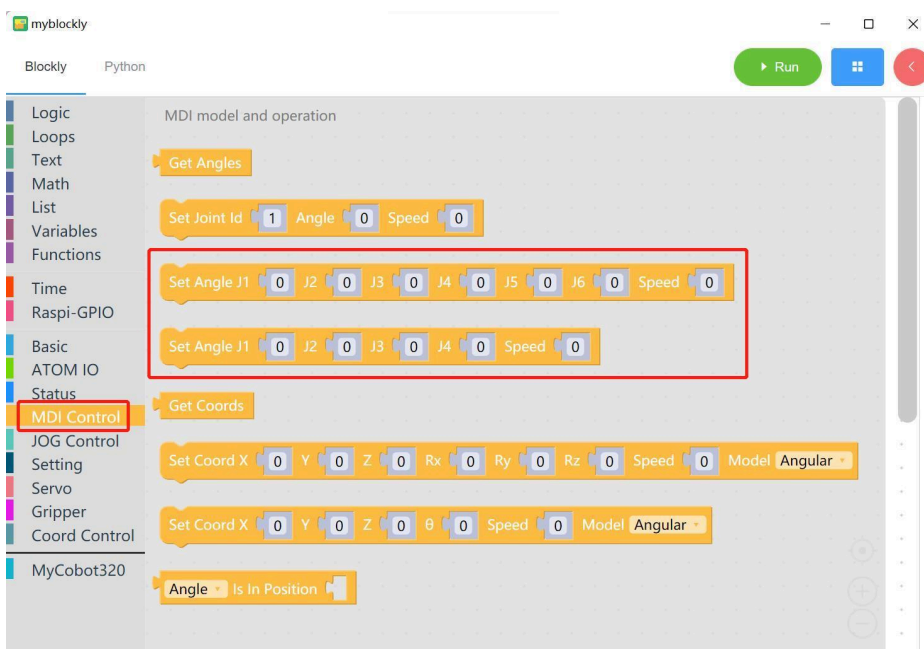
Other series: Make sure the machine is normal

Learning content in this chapter

How to use myBlockly to control the robot arm to return to the origin

API introduction

- Method module: Set angle



- Scope of application: This module is applicable to the 6-axis myCobot 280 series, mechArm series, myCobot 320 series



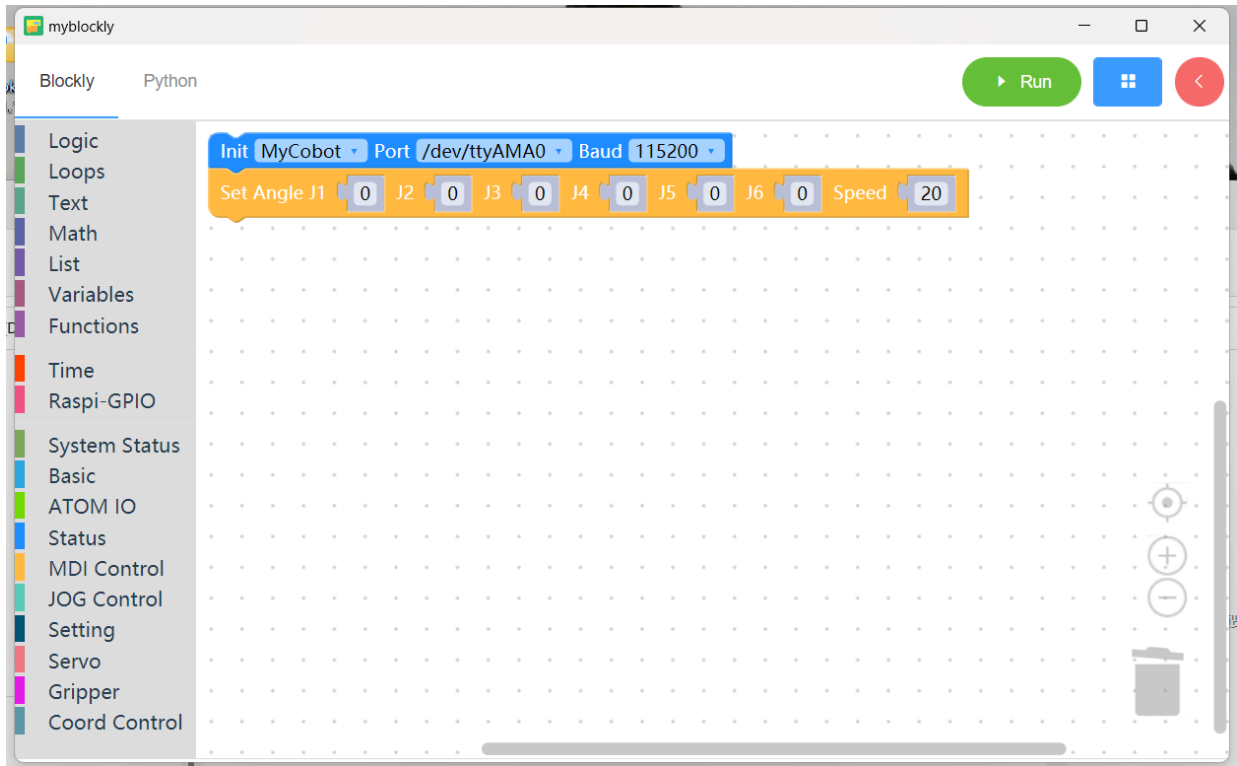
- Scope of application: This module is applicable to the 4-axis myPalletizer series
- Parameter introduction:

This module has two parameters that can be adjusted:

- Joint angle parameters: If the robot arm is returned to the origin, all joint angle parameters must be set to 0
- Speed parameters: Please refer to the robot parameter introduction section of **Product Introduction**
- Purpose: Control the robot arm and return the angles of all axes of the robot arm to the origin (angle is 0)

myCobot

Simple demonstration



- Implementation content: Control the robot arm to move back to the origin, so that the angles of all axes of the robot arm are 0

Control single joint motion

Preparation before starting

M5Stack series: Make sure the robot is connected to the computer

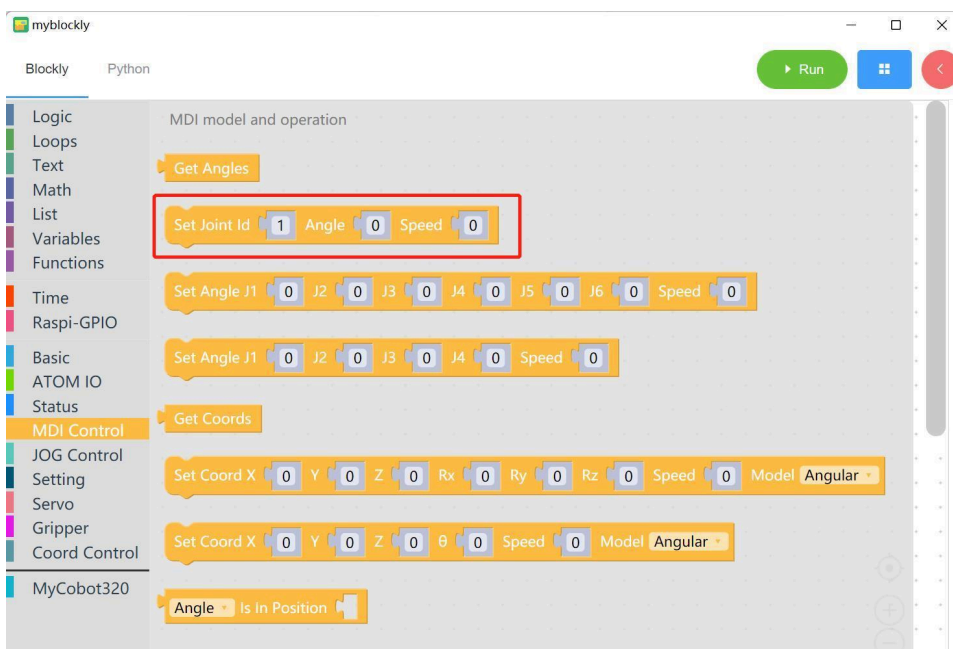
Other series: Make sure the machine is normal

Learning content in this chapter

How to use myBlockly to control the single joint motion of the robot

API introduction

- Method module: Set joint



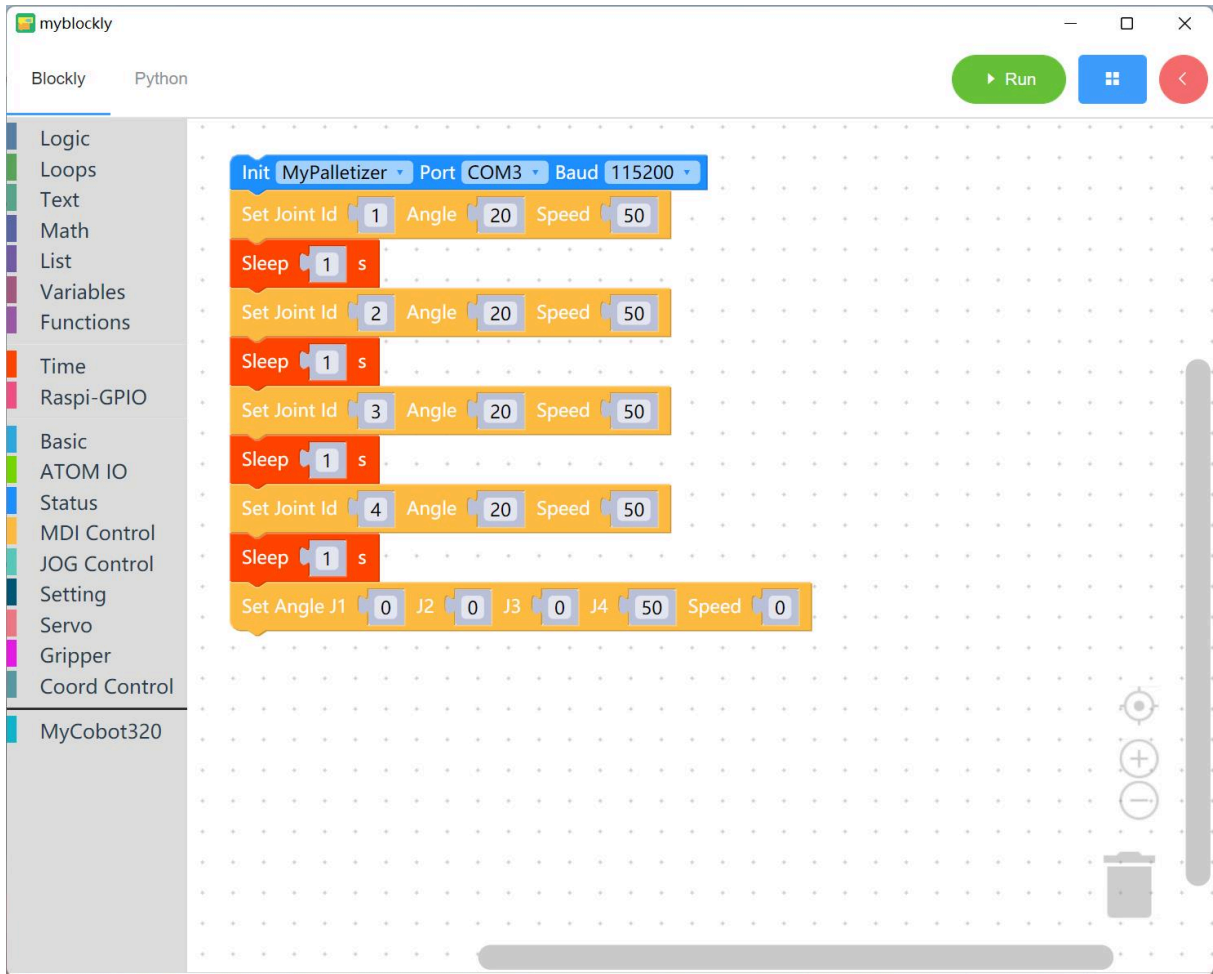
- Applicable scope: myCobot280 series, myCobot320 series, mechArm series, myPalletizer series
- Parameter introduction:

This method has three parameters that can be adjusted:

- Joint parameters: The parameter range of myCobot280 series, myCobot320 series and mechArm series is: 1-6; the parameter range of myPalletizer series is: 1-4 (corresponding to the joints of the robot arm)
- Angle parameters: refer to the parameters of the corresponding model
- Speed: Control the speed of the robot arm movement, the parameter range is: 0~100
- Purpose: Control the movement of a single joint of the robot arm

Simple demonstration

- The graphic code is as follows:



- Implementation content:

Control the robot arm 1 joint, run to the position of 1 joint angle 20 at a speed of 50, after one second,

Control the robot arm 2 joint, run to the position of 2 joint angle 20 at a speed of 50, after one second,

Control the robot arm 3 joint, run to the position of 3 joint angle 20 at a speed of 50, after one second,

Control the robot arm 4 joint, run to the position of 4 joint angle 20 at a speed of 50, after one second,

Return all joints of the robot arm to the origin at a speed of 60, and end the program.

Control multiple joints

Preparation before starting

M5Stack series: Make sure the robot is connected to the computer

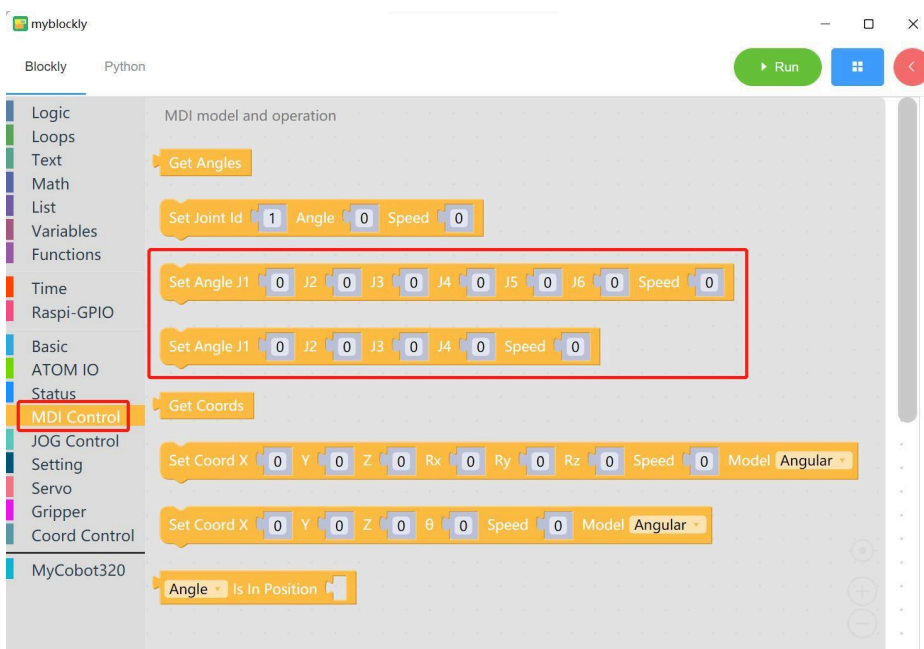
Other series: Make sure the machine is normal

Learning content in this chapter

How to use myBlockly to control the movement of multiple joints of the robot

API introduction

- Method module: Set full angle



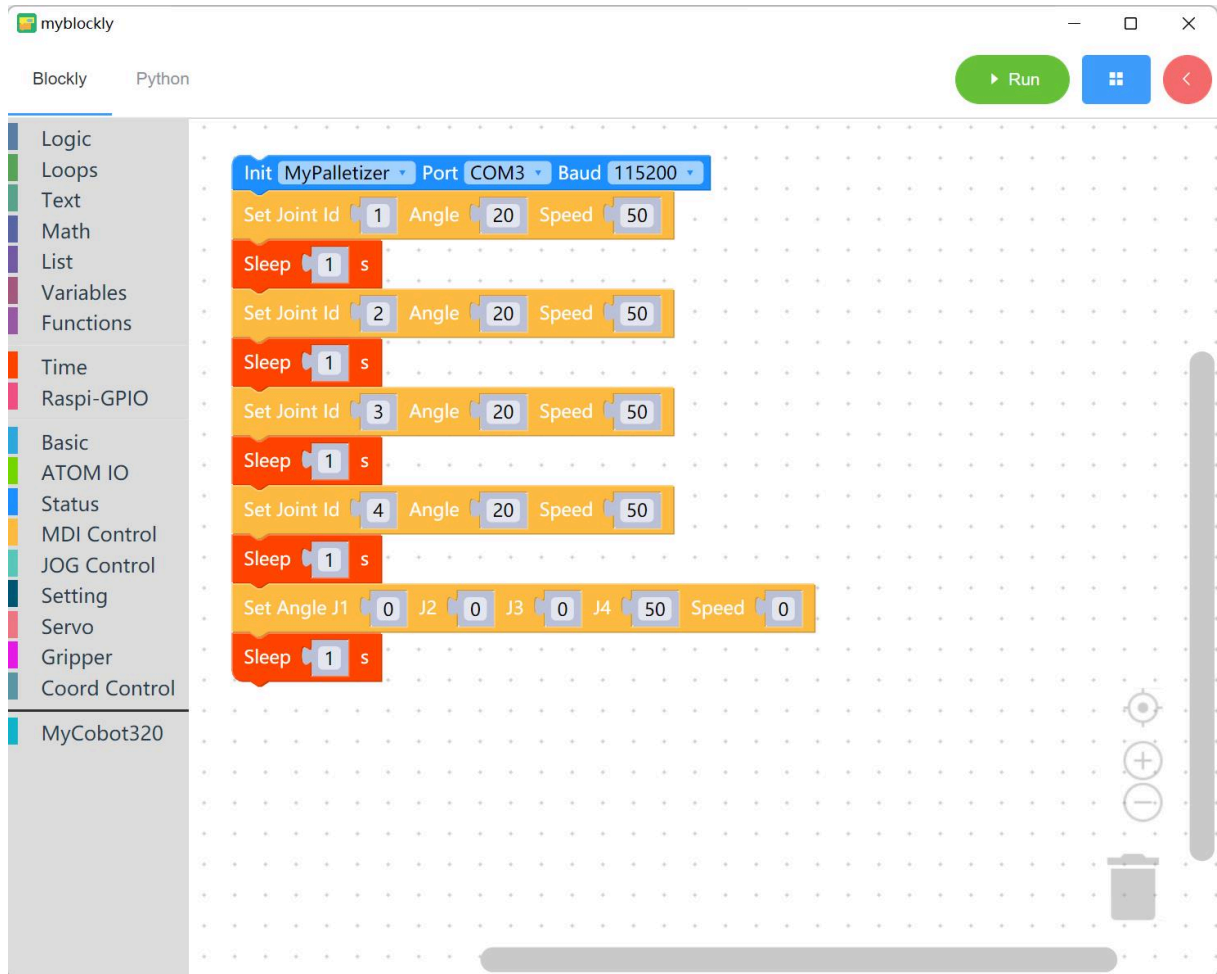
- Scope of application: This module is applicable to the 6-axis myCobot 280 series, mechArm series, myCobot 320 series
- Parameter introduction:

This module has two parameters that can be adjusted:

- Joint angle parameters: You can set the parameters within the range of joint motion of the robot arm as needed (for details of the joint motion range, please refer to Product Introduction section).
- Speed parameters: You can set the parameters within the range of robot arm motion speed as needed (for the maximum motion speed, please refer to Product Introduction section).
- Purpose: Control the motion of multiple joints of the robot arm

Simple demonstration

- The graphic code is as follows:



- Implementation content:

Control all joints of the robot to return to the origin. After two seconds,

Control joints 1, 2, 3 and 4 of the robot to run at a speed of 50 to 30 degrees, 30 degrees, -30 degrees and 50 degrees respectively. After two seconds,

Control all joints of the robot to return to the origin at a speed of 50. After two seconds,

Control joints 1, 2, 3 and 4 of the robot to run at a speed of 50 to -30 degrees, 0 degrees, 30 degrees and -50 degrees respectively. After two seconds, end the program.

Control the robot arm to swing left and right

Preparation before starting

M5Stack series: Make sure the robot arm is connected to the computer

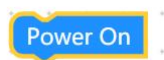
Other series: Make sure the machine is normal

Learning content in this chapter

How to use myBlockly to control the robot arm to swing left and right

API display

- Method module 1: Power on



- Applicable scope: myCobot280 series, myCobot320 series, mechArm series, myPalletizer series
- Purpose: Start the system
- Method module 2: Release joints

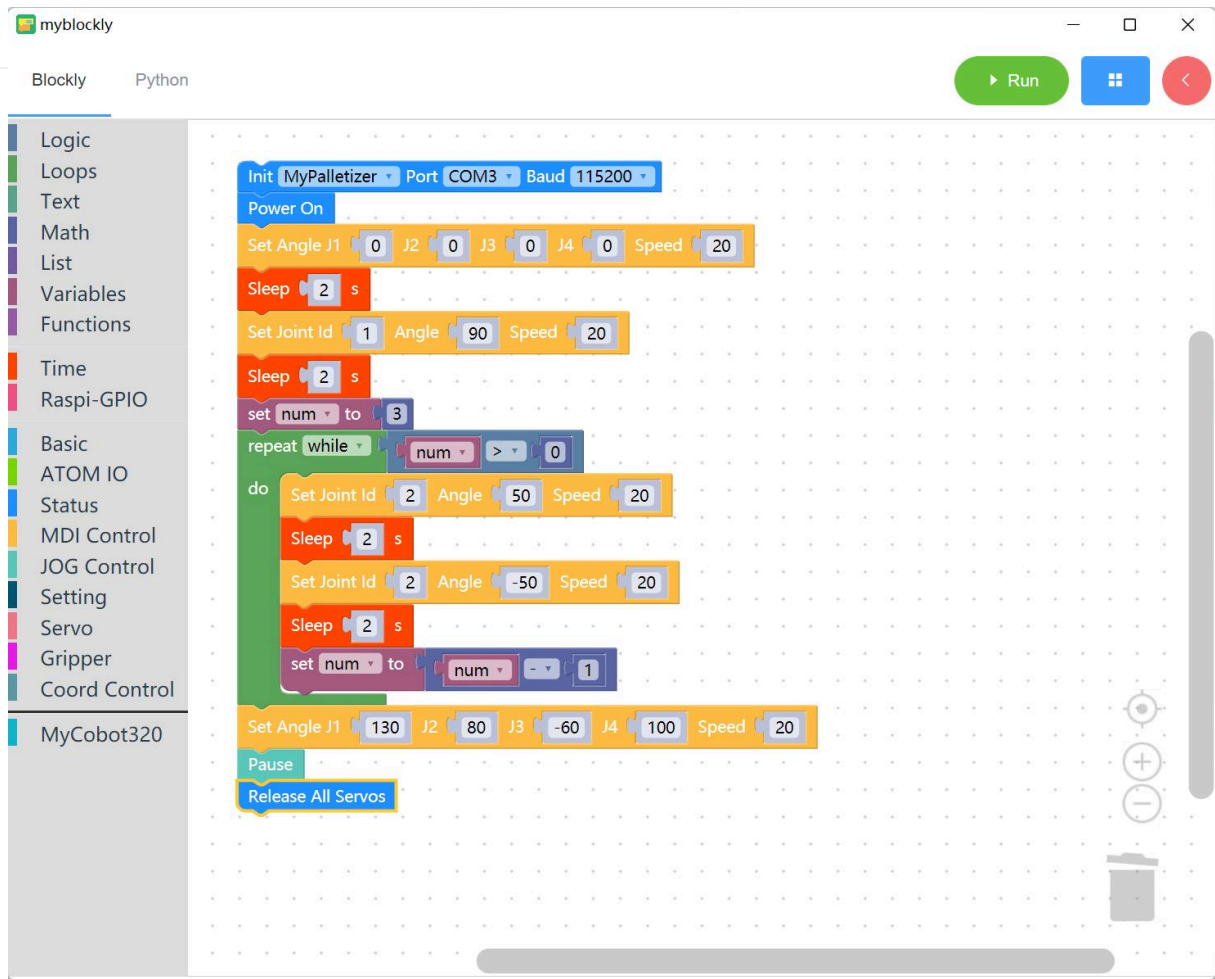


- Applicable scope: myCobot280 series, myCobot320 series, mechArm series, myPalletizer series
- Purpose: Stop the movement of the robot arm and lock each joint

Simple demonstration

- The graphic code is as follows:

4.1 First-time self-check



- Implementation content:

Power on the robot arm and control the robot arm to move back to the origin. After two seconds, Control the robot arm 1 joint to run to the angle 90 position at a speed of 20. After two seconds, Control the robot arm 2 joint to run to the angle 50 position at a speed of 20. After two seconds, Control the robot arm 2 joint again to run to the angle -50 position at a speed of 20. After two seconds, Loop the control of the 2 joints twice. After the loop ends, Run the 1st joint, 2nd joint, 3rd joint and 4th joint at a speed of 20 to the angles of 130, 80, -60 and 100 respectively. Finally, the robot arm movement is paused, all joints are released, and the program ends.

Control the robot to dance

Preparation before starting

M5Stack series: Make sure the robot is connected to the computer

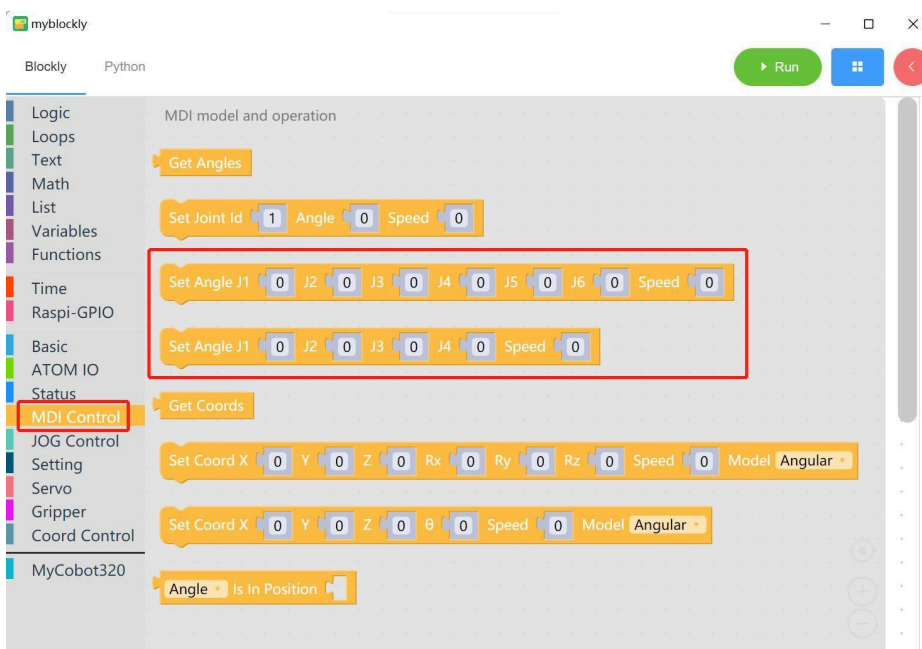
Other series: Make sure the machine is normal

Learning content in this chapter:

How to use myBlockly to control the robot to dance

API display

- Method module: Set full angle



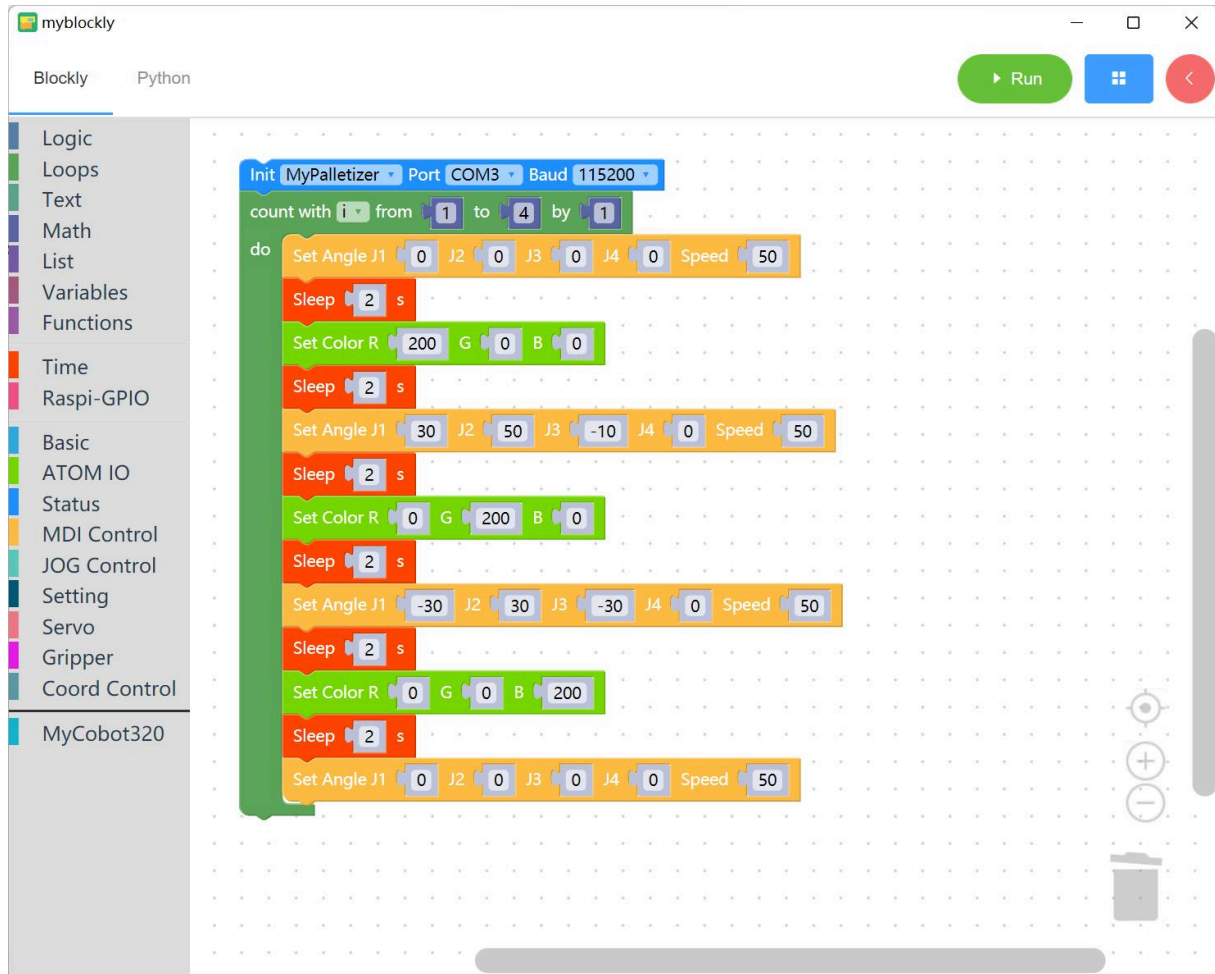
- Scope of application: This module is suitable for 6-axis myCobot 280 series, mechArm series, myCobot 320 series
- Parameter introduction:

This module has two parameters that can be adjusted:

- Joint angle parameters: You can set the parameters within the range of joint motion of the robot arm as needed (for details of the joint motion range, please refer to Product Introduction section).
- Speed parameters: You can set the parameters within the range of robot arm motion speed as needed (for the maximum motion speed, please refer to Product Introduction section).
- Purpose: Control the motion of multiple joints of the robot arm

Simple demonstration

- The graphic code is as follows:



- Implementation content:

Control all joints of the robot arm to return to their original positions, and the RGB light board turns red.

After one second, control joints 1, 2, and 3 to run at a speed of 50 to 30 degrees, 50 degrees, and -10 degrees respectively.

After one second, the RGB light board turns green.

After one second, control joints 1, 2, and 3 to run at a speed of 50 to -30 degrees, 30 degrees, and -30 degrees respectively.

After one second, the RGB light board turns blue.

After one second, control all the robot arms to shut down and return to their original positions.

Use of grippers

Preparation before starting

M5Stack series: Make sure the robot is connected to the computer

Other series: Make sure the machine is normal

Grippers include adaptive grippers, electric grippers, and pneumatic grippers. Different grippers are compatible with different robot arm models. For details, please refer to [Accessories](#). Here, we take the adaptive gripper and myPalletizer 260 M5Stack robot arm as an example to explain how to use myBlockly to control the gripper.

Learning content in this chapter

How to use myBlockly to control the adaptive gripper connected to the myPalletizer 260 M5Stack robot

API display

- Method module 1: Reinitialize the gripper

Set Gripper Ini.

- Applicable scope: myCobot 280 series, myPalletizer 260 series, mechArm 270 series
- Purpose: Set the initial position of the gripper
- Method module 2: Set the gripper state

Set Gripper State Speed

- Applicable scope: myCobot 280 series, myPalletizer 260 series, mechArm 270 series
- Parameter introduction:

This module has two parameters that can be adjusted:

- Gripper state parameter: 1 indicates the gripper is closed, 0 indicates the gripper is open
- Speed parameter: indicates the speed of rotation, the value range is 0~100
- Purpose: Make the gripper enter the specified state (open or closed) at the specified speed
- Method module 3: Set the value of the gripper

Set Gripper Value Speed

- Applicable scope: myCobot 280 series, myPalletizer 260 series, mechArm 270 series
- Parameter introduction:

This module has two parameters that can be adjusted:

- Gripper value parameter: indicates the position to be reached by the gripper, with a value range of 0~256.

4.1 First-time self-check

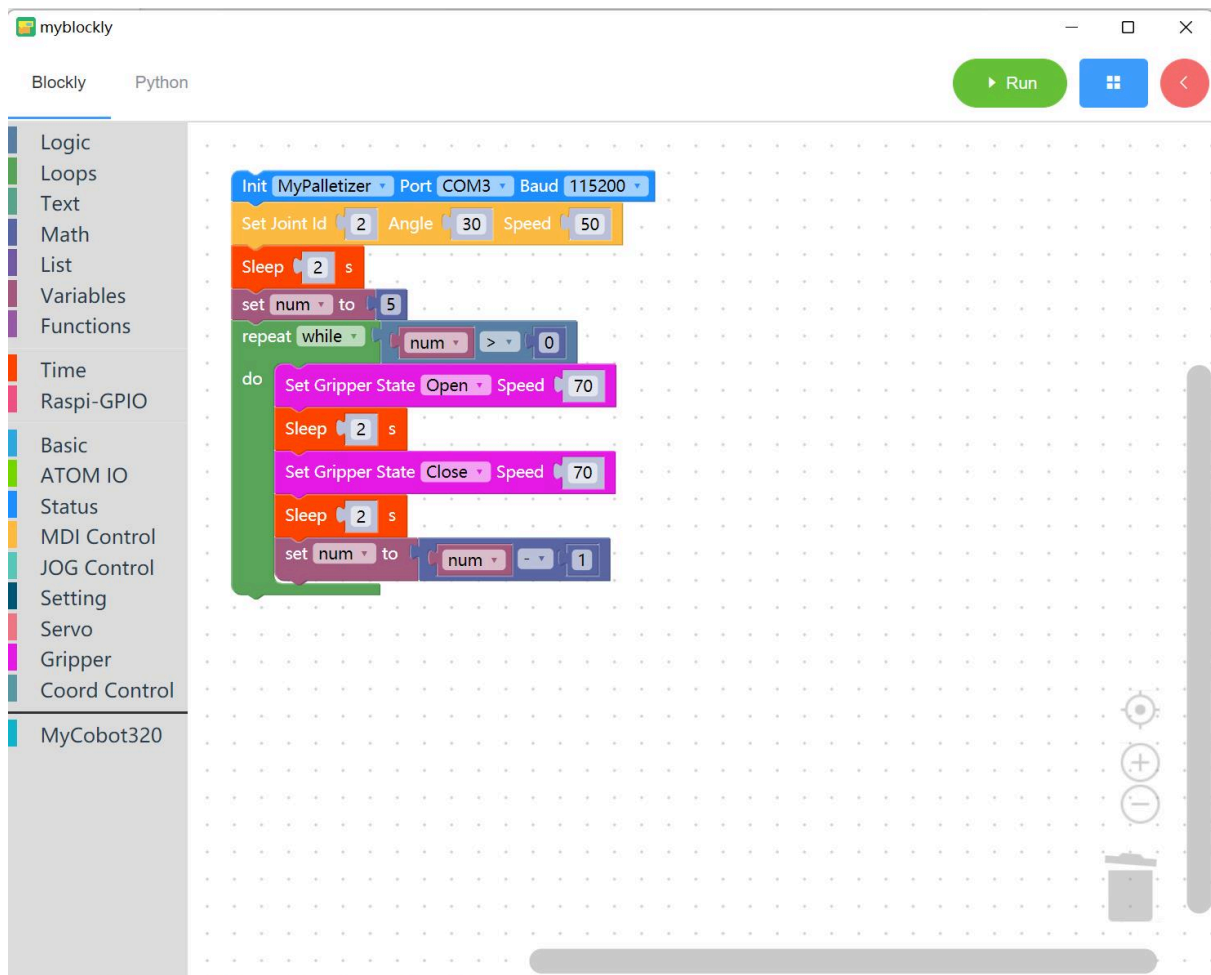
- Speed parameter: indicates the speed of rotation, with a value range of 0~100.
- Purpose: Make the gripper rotate to the specified position at a specified speed.
- Method module 4: Is the gripper moving

Is Gripper Moving

- Applicable scope: myCobot 280 series, myPalletizer 260 series, mechArm 270 series
- Purpose: Determine whether the gripper is running

Simple demonstration

- The graphic code is as follows:



- Implementation content:

The robot arm 2 joint runs at a speed of 50 to 30 degrees. After one second, the gripper opens at a speed of 70. After one second, the gripper closes at a speed of 70. After the gripper opens and closes 5 times, the program ends.

Use of suction pump

Preparation before starting

M5Stack series: Make sure the robot is connected to the computer (

Other series: Make sure the machine is normal

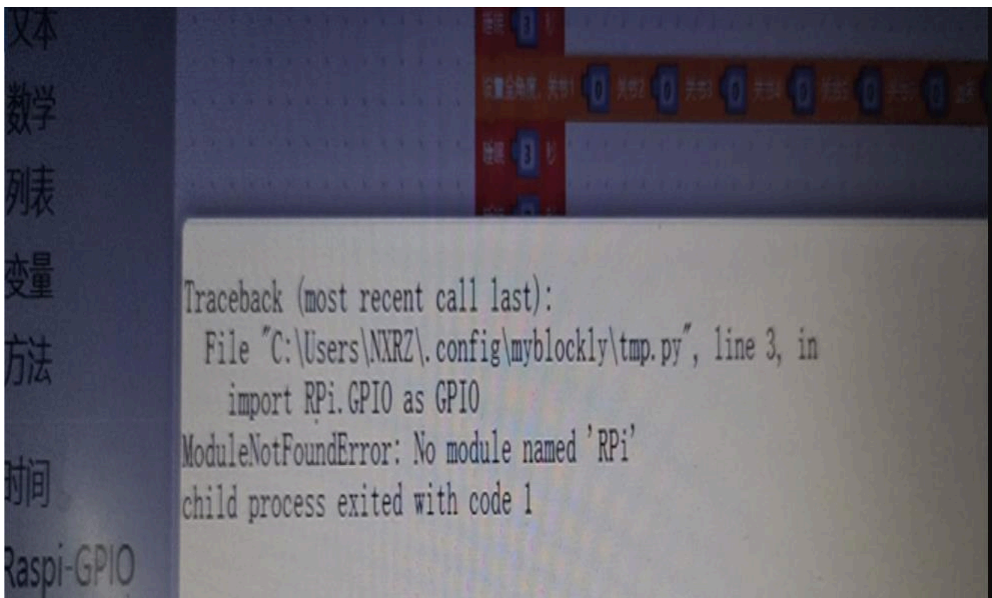
For the introduction and installation of the suction pump, please refer to **Accessories**. The robot models that are compatible with the suction pump include myCobot 280, myPalletizer 260 and mechArm 270. Here, the myPalletizer 260 M5Stack robot is used as an example.

Learning content in this chapter

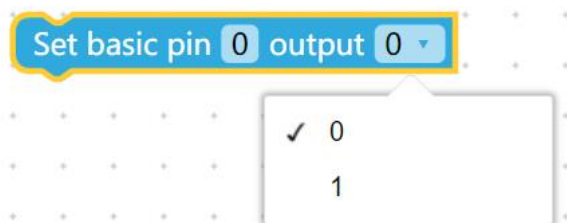
How to use myBlockly to control the suction pump connected to the myPalletizer 260 M5Stack robot

API display

Note: The M5 version and the Raspberry Pi version require different method modules to control the suction pump. The M5 version robot cannot use the Raspberry Pi interface. If the module does not match the model, the program will report an error (as shown below).



- Method module **1** (for M5 version): Set the bottom pin number status



- Applicable scope: myCobot 280 series, myPalletizer 260 series and mechArm 270 series
- Parameter introduction:

This module has two parameters that can be adjusted:

4.1 First-time self-check

- Bottom pin number parameter: the specific pin number at the bottom of the device (only the numeric part is taken)
- Running state parameter: 0 means set to running state, 1 means stop state
- Purpose: Set the working state of the bottom pin number
- Method module 2 (for M5 version): `Get bottom pin number`

Get basic pin 0 input

- Applicable scope: myCobot 280 series, myPalletizer 260 series and mechArm 270 series M5 version
- Parameter introduction:
- Bottom pin number parameter: the specific pin number at the bottom of the device (only the numeric part is taken)
- Purpose: Get the working state of the bottom pin number
- Method module 3 (for Raspberry Pi version): `Set Mode`

Set mode BCM

- Applicable to: Raspberry Pi versions of myCobot 280 series, myPalletizer 260 series and mechArm 270 series
- Parameter introduction:
- Mode parameter: can enter "BCM" or "BOARD" mode
- Purpose: Set the Raspberry Pi GPIO pin mode
- Method module 4 (Raspberry Pi version): `Pin signal setting`

Set pin 0 mode OUT

- Applicable to: Raspberry Pi versions of myCobot 280 series, myPalletizer 260 series and mechArm 270 series
- Parameter introduction:

This module has two parameters that can be adjusted:

- Pin number: The specific pin number at the bottom of the device (only the digital part is taken)
- Level status: IN is set to input signal, OUT is set to output signal
- Purpose: Set pin signal
- Method module 5 (Raspberry Pi version): `Pin level setting`

Set pin 0 output HIGH

- Applicable scope: Raspberry Pi version of myCobot 280 series, myPalletizer 260 series and mechArm 270 series
- Parameter introduction:

4.1 First-time self-check

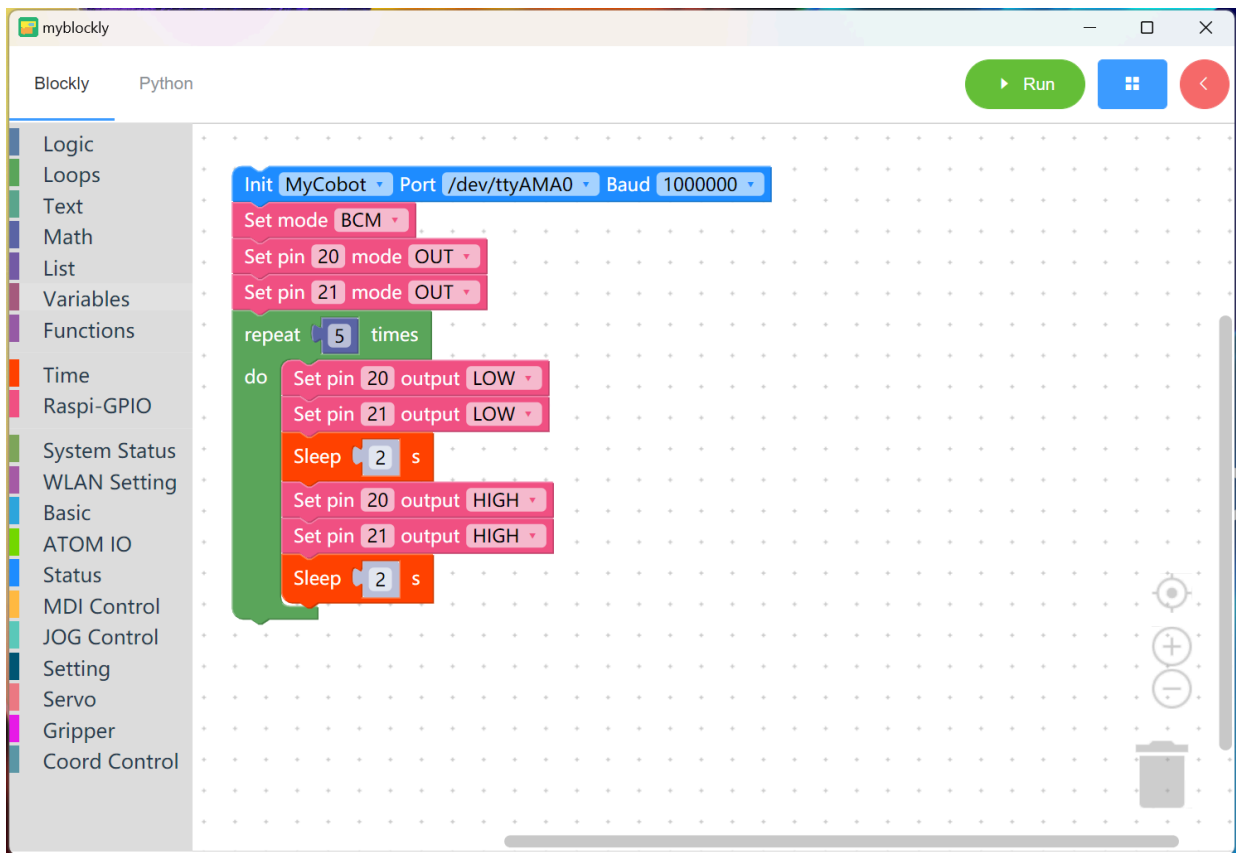
This module has two parameters that can be adjusted:

- Pin number: the specific pin number at the bottom of the device (only the digital part is taken)
- Level state: 0 is set to low level, 1 is set to high level (low level of suction pump starts working, high level stops working)
- Purpose: Set the pin to high or low level

Simple demonstration

Vertical suction pump V1.0

- The graphic code is as follows: (for M5 version)



```
myblockly
Blockly Python
Init MyCobot Port /dev/ttyAMA0 Baud 1000000
Set mode BCM
Set pin 20 mode OUT
Set pin 21 mode OUT
repeat 5 times
do
Set pin 20 output LOW
Set pin 21 output LOW
Sleep 2 s
Set pin 20 output HIGH
Set pin 21 output HIGH
Sleep 2 s
```

4.1 First-time self-check

- Graphic code is as follows: (for Raspberry Pi version)



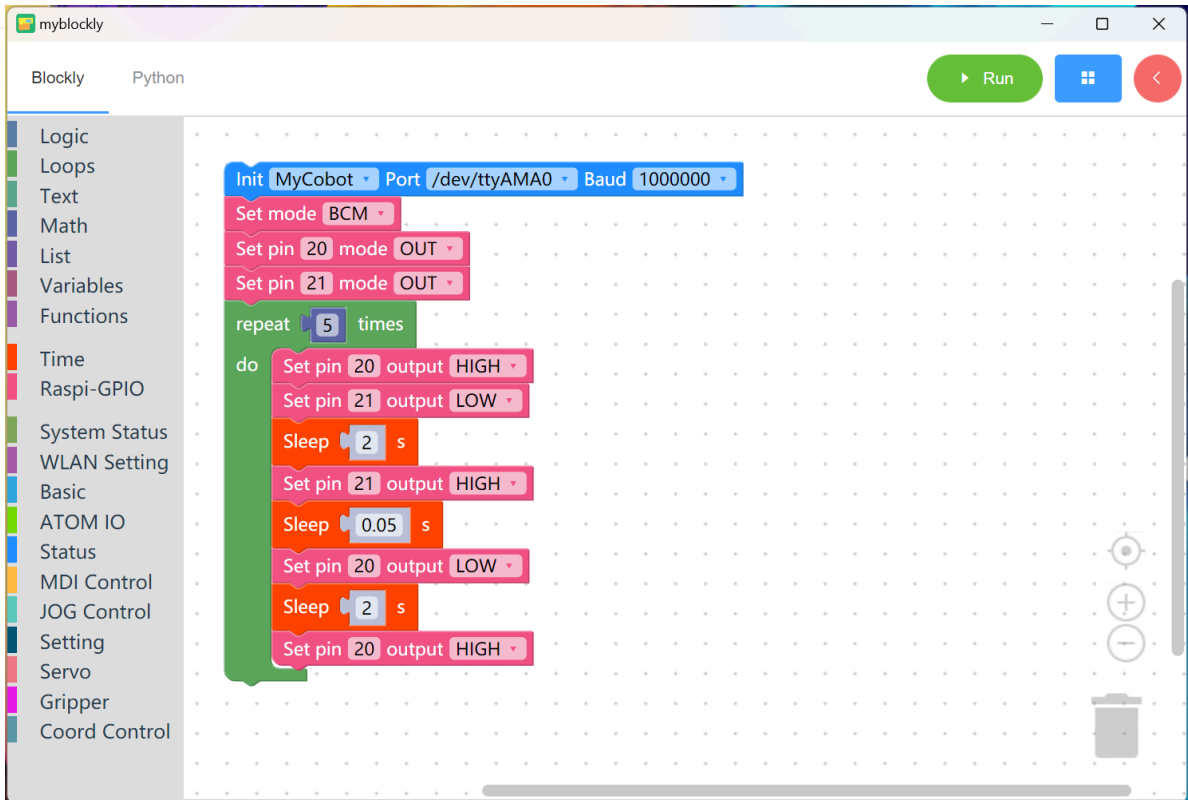
- Implementation content: The suction pump vibrates and starts working. The suction pump sucks up the object, puts it down after two seconds, and repeats the previous action after another two seconds until the program runs to the end.

Vertical Suction Pump V2.0

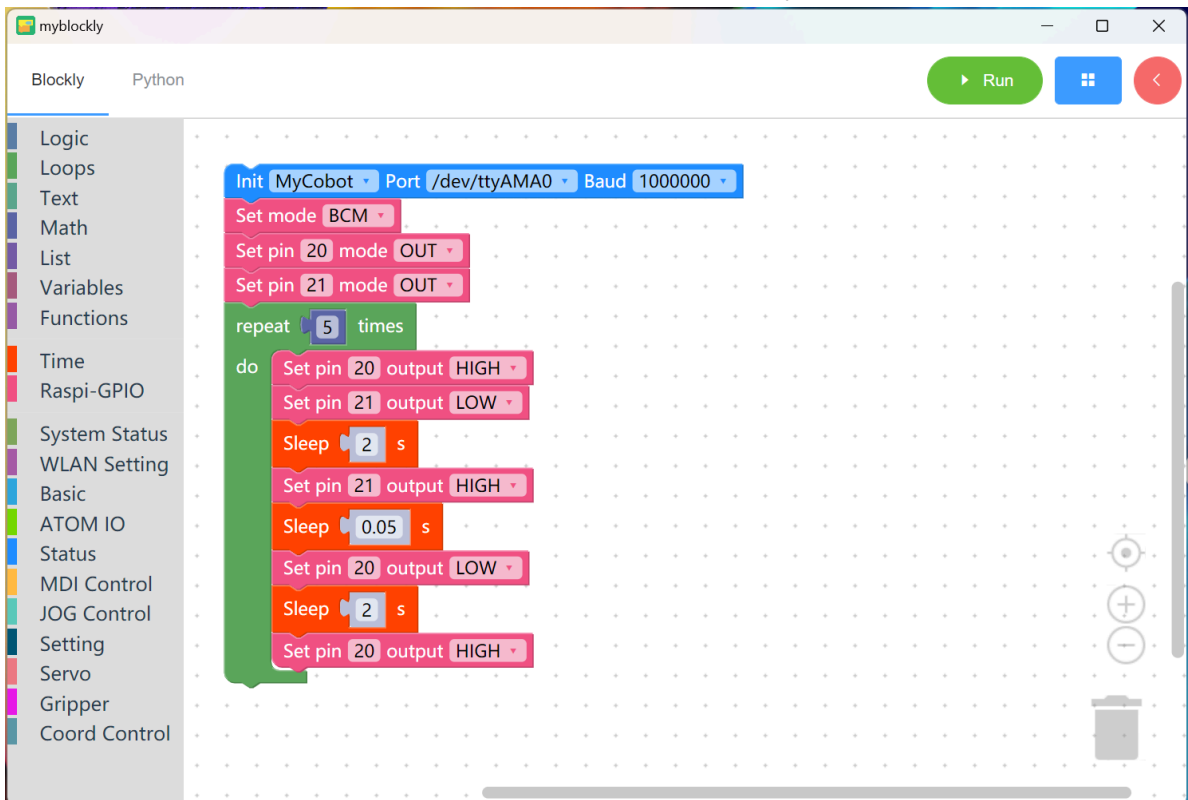
- Graphic code is as follows: (for M5 version)

4.1 First-time self-check

- Pin 5/2 controls the solenoid valve and the air release valve respectively



- Graphic code is as follows: (for Raspberry Pi version)
- Pin 20/21 controls the solenoid valve and the air release valve respectively



- Implementation content: First close the solenoid valve and open the air release valve to prepare for starting the suction pump; after two seconds, close the air release valve. After 0.05 seconds, open the solenoid valve, the suction pump vibrates and starts working. The suction pump picks up the object and puts it down after two

4.1 First-time self-check

seconds. Close the air release valve and repeat the previous actions until the program ends.

Gripper test

Before you start

M5Stack series: Make sure the robot arm is connected to the computer

Others: Make sure the machine is normal

To use the gripper test function, myblockly needs to be updated to v3.3.5

Learning content in this chapter

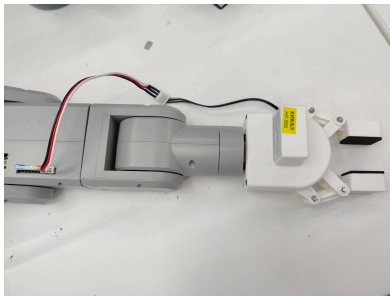
How to use myBlockly to quickly test whether the gripper is normal.

Currently, the gripper test function is only compatible with the following models: myCobot 280 series, mecharm 270 series, myPalletizer 260 series, myArm 300 Pi.

Gripper includes adaptive gripper, electric gripper and pneumatic gripper. Different grippers are compatible with different robot arm models. For details, please refer to [Accessories](#).

Here, mecharm 270 M5 and adaptive gripper are used as an example.

First install the gripper on the machine



myblockly connects to the machine serial port

myblockly

Blockly Python

Run

COM4

115200

Close Open

Auto Complete

Quick Move

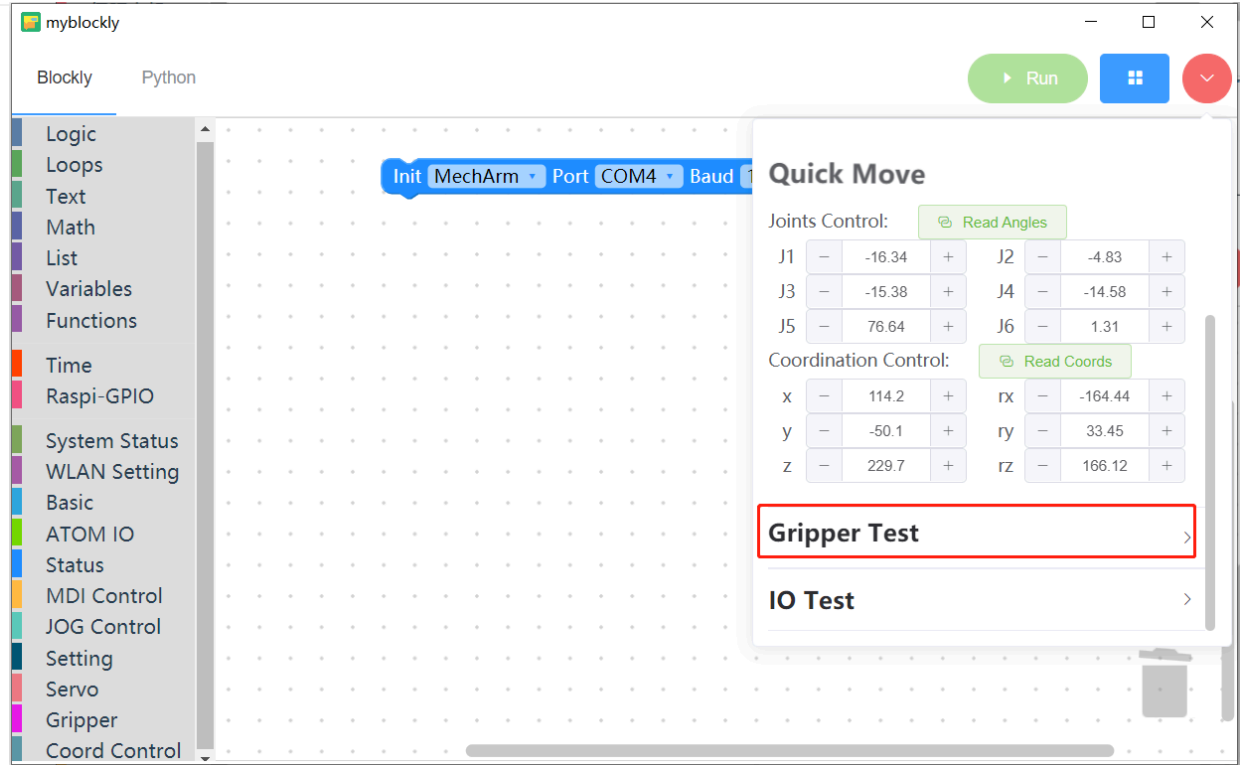
Joints Control:

| | | | | | | | |
|----|---|--------|---|----|---|--------|---|
| J1 | - | -16.34 | + | J2 | - | -4.83 | + |
| J3 | - | -15.38 | + | J4 | - | -14.58 | + |
| J5 | - | 76.64 | + | J6 | - | 1.31 | + |

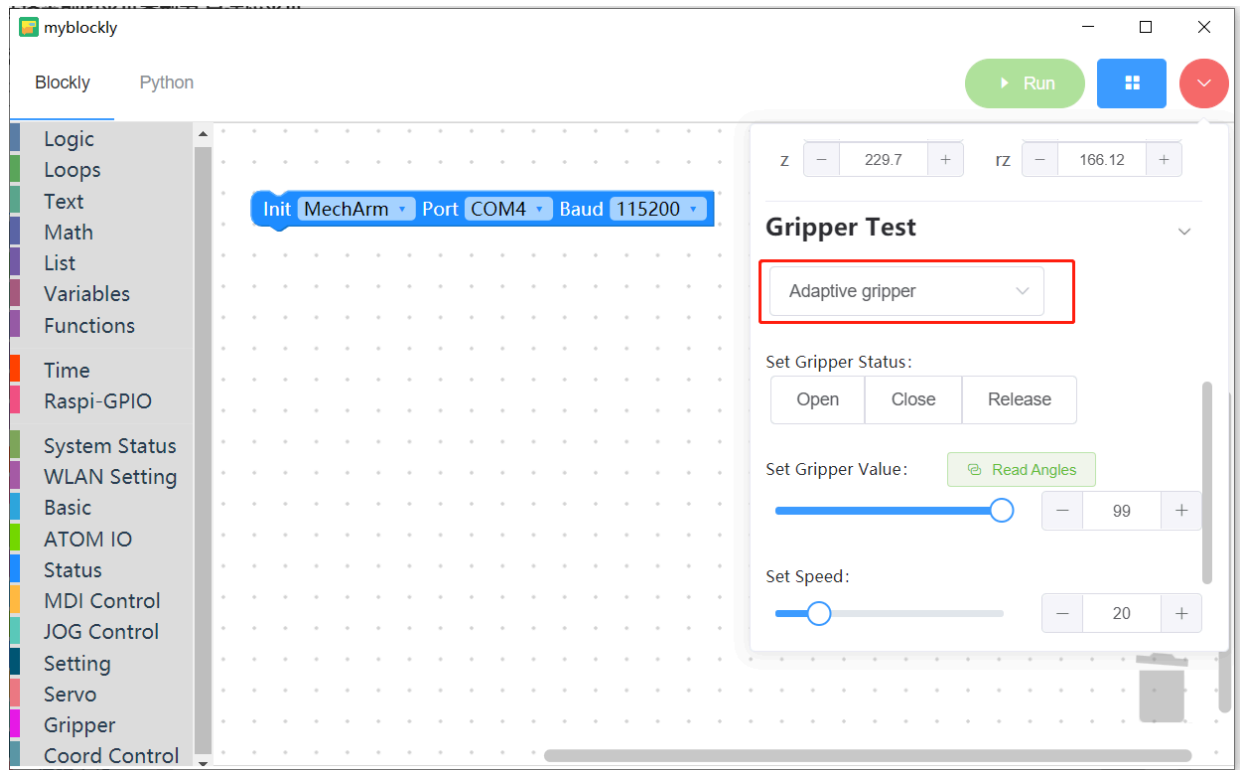
Coordination Control:

| | | | | | | | |
|---|---|-------|---|----|---|---------|---|
| x | - | 114.2 | + | rx | - | -164.44 | + |
| y | - | -50.1 | + | ry | - | 33.45 | + |
| z | - | 229.7 | + | rz | - | 166.12 | + |

Open the gripper test



Select the current gripper type as Adaptive Girpper



Then you can try the following operations:

- Set the gripper state: open, closed, released. Pay attention to the movement of the gripper.
- Click the Read Angle button to get the current gripper value.
- You can fine-tune the gripper value.

4.1 First-time self-check

- You can set the gripper's moving speed.

Tips: When the gripper type is Dexterous Hand, you cannot set the gripper value, you can only set its status.

IO test

Preparation before starting

M5Stack series: Make sure the robot arm is connected to the computer

Others: Make sure the machine is normal

To use the IO test function, myblockly needs to be updated to v3.3.5

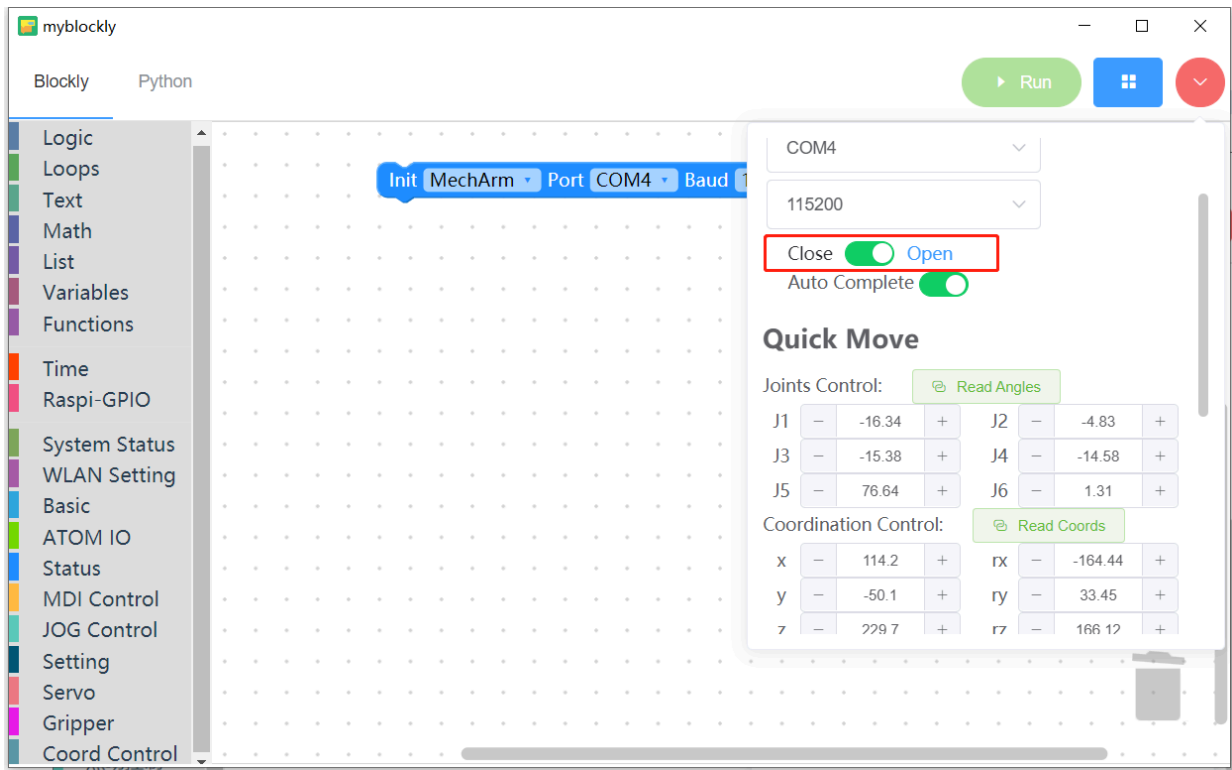
Learning content in this chapter

How to use myBlockly to quickly test whether io is normal.

Currently, the gripper test function is only compatible with the following models: myCobot 280 series, mecharm 270 series, myPalletizer 260 series, myArm 300 Pi.

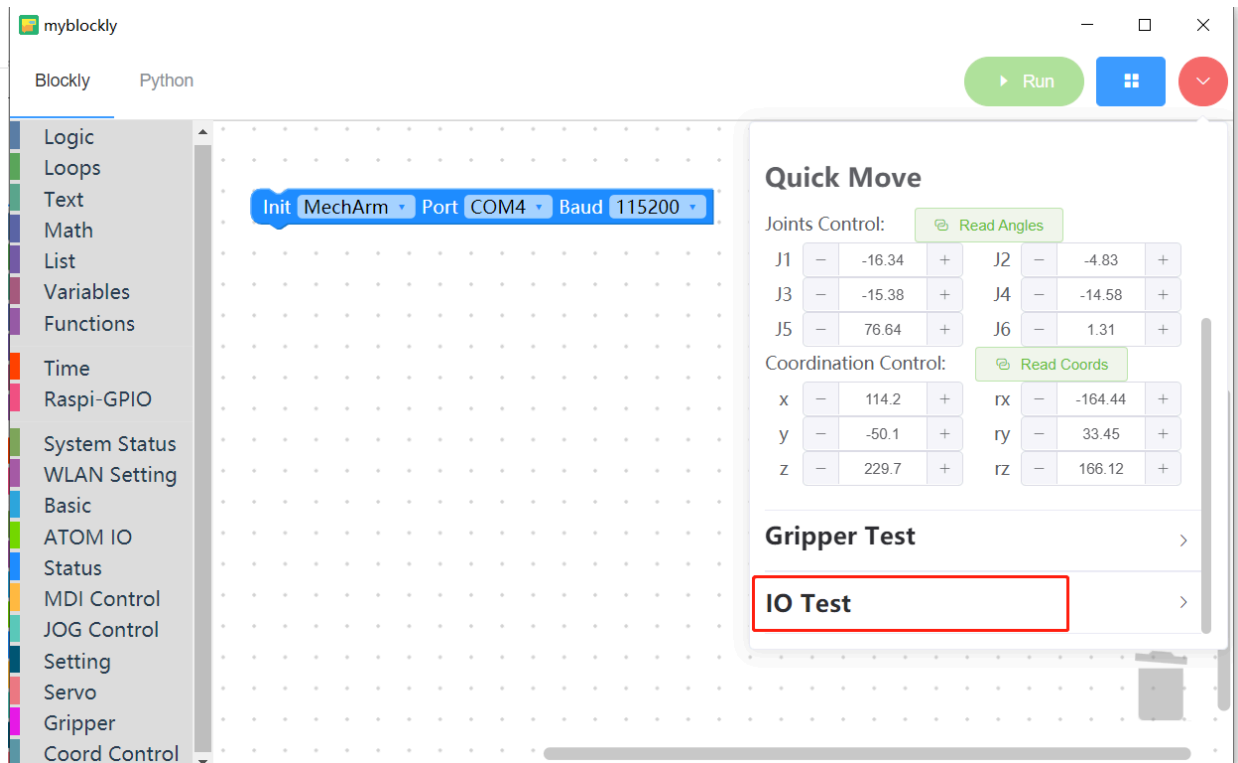
Here, take mecharm 270 M5 as an example.

myblockly connects to the machine serial port



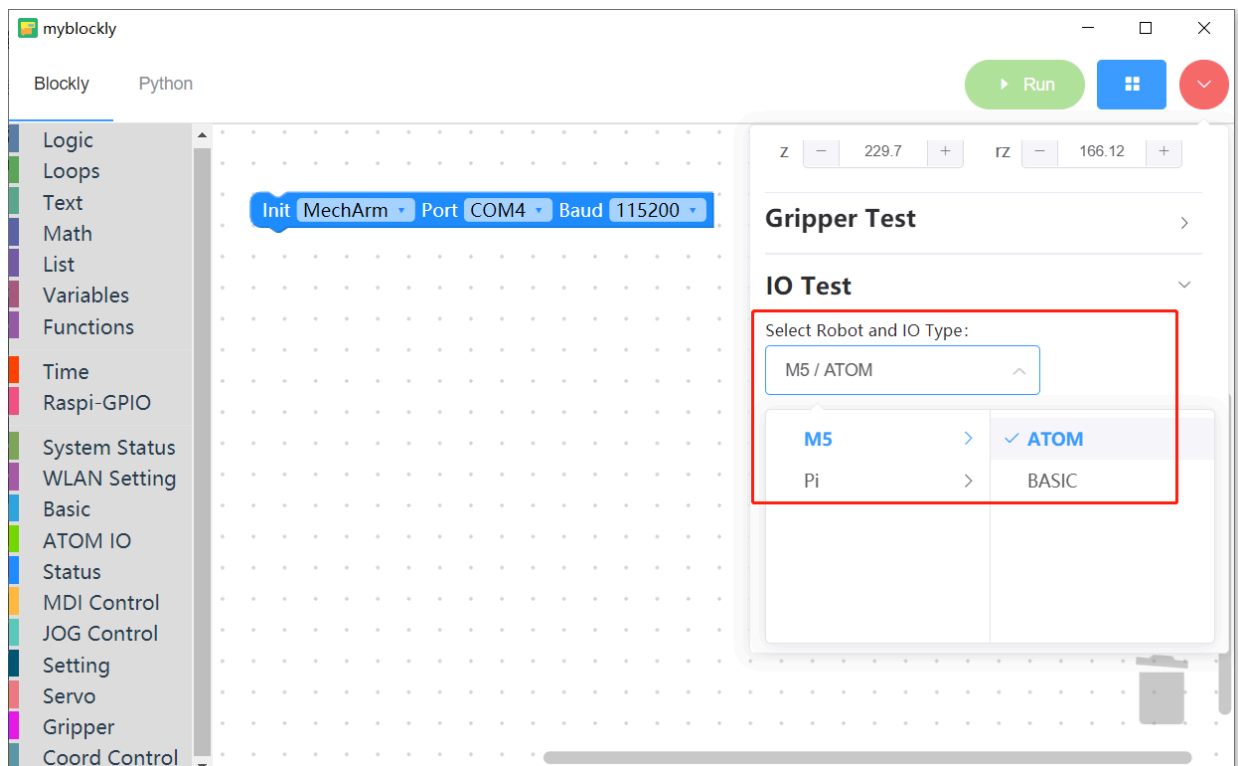
Open IO test

4.1 First-time self-check



Select IO type

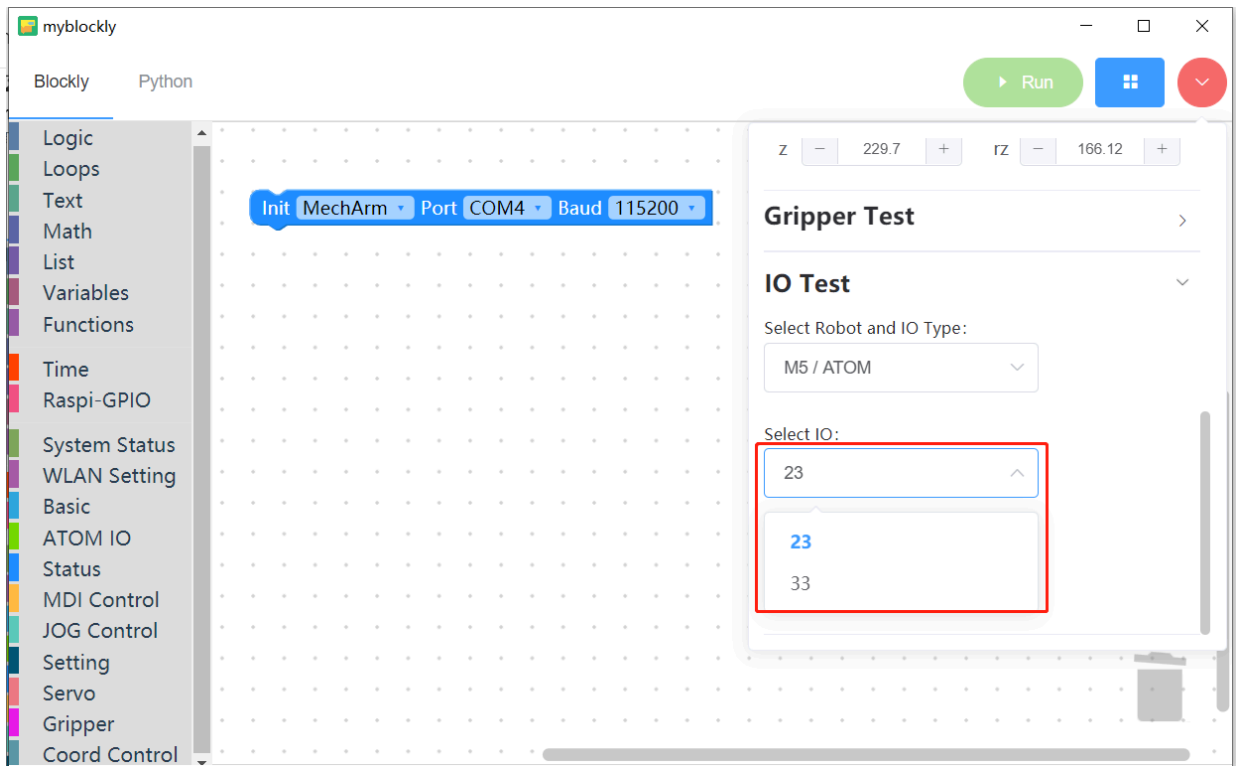
Our current model is M5, so select M5 type. You can choose ATOM and BASIC in the M5 category. Here we choose ATOM



Select ATOM IO

meacharm 270 M5 Atom IO ports are currently only open 23 and 33

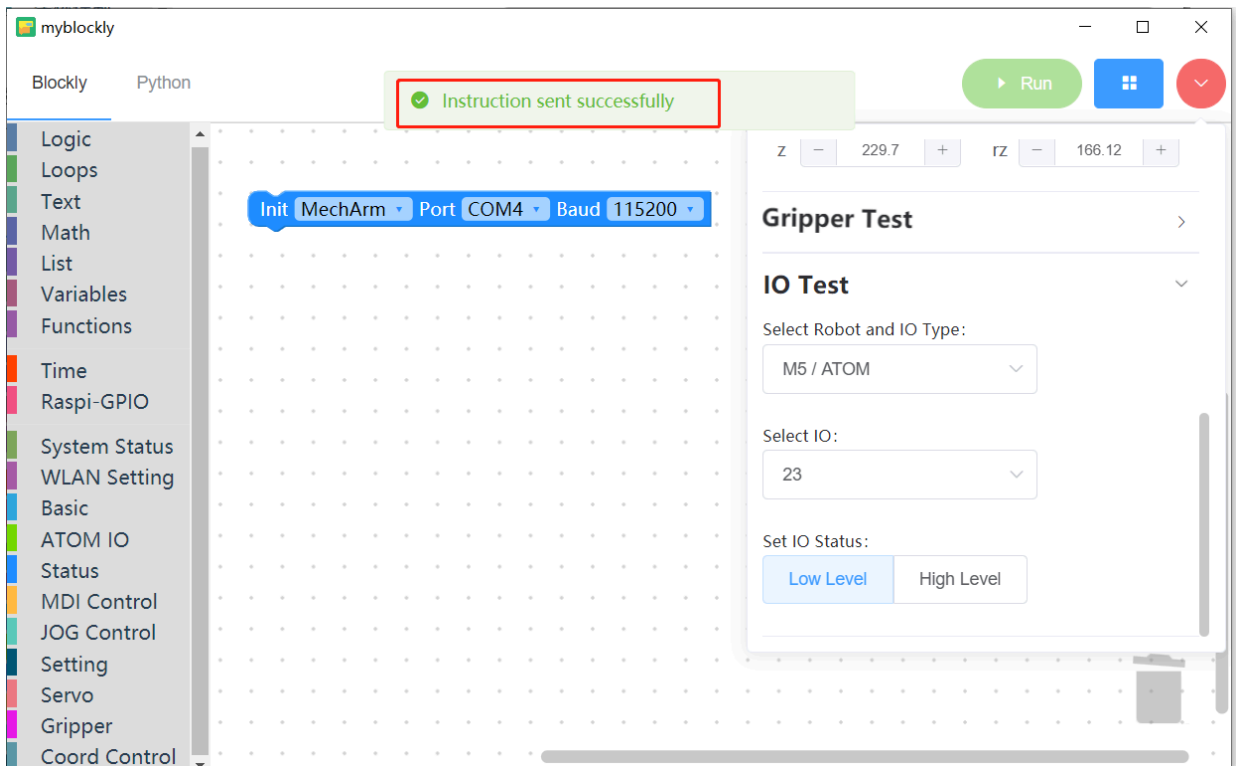
4.1 First-time self-check



Operate IO status

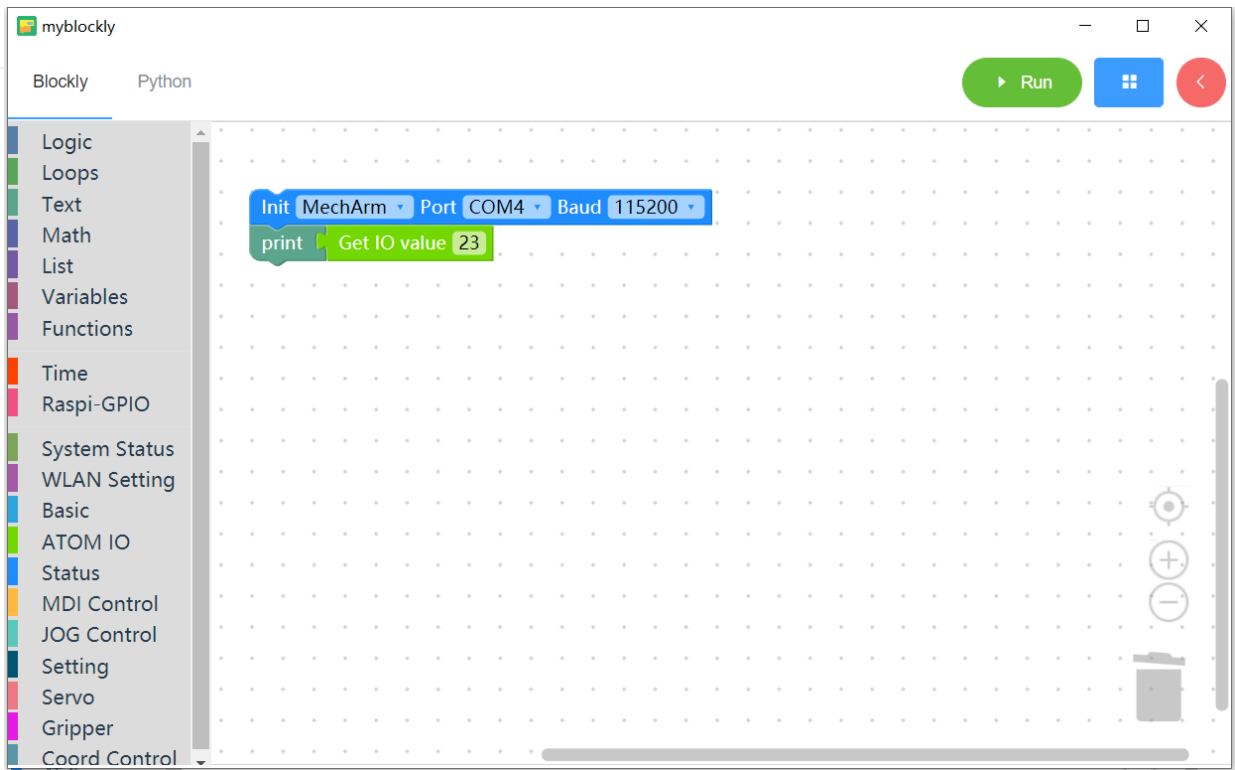
You can click the `Low Level` `High Level` buttons to change the current selection io port status

Set high and low levels and send the command successfully



After changing the io status, you can read the Atom io high and low levels through the building block to check whether it is successful

4.1 First-time self-check



Q&A

This chapter lists common problems in using myBlockly to control the robot arm for reference.

Q1: myBlockly reports an error `ModuleNotFoundError: No module named 'pymycobot'` when running

A: This is because the `pymycobot` library is not installed when setting up the Python environment. To install the `pymycobot` library, open the terminal (Win key + R key), enter: `pip install pymycobot --upgrade --user`, and press Enter to display Successfully installed `pymycobot`.

```

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

C:\Users\Surface> pip install pymycobot
Collecting pymycobot
  Using cached pymycobot-2.9.6-py3-none-any.whl (48 kB)
Requirement already satisfied: pyserial in c:\users\surface\appdata\local\programs\python\python310\lib\site-packages
from pymycobot (3.5)
Installing collected packages: pymycobot
Successfully installed pymycobot-2.9.6

C:\Users\Surface>

```

Q2: The robot arm is unresponsive due to the lack of a `sleep` method module

A: The program that operates the robot arm movement takes time to complete, so a `sleep` module needs to be connected after an action to give the robot arm time to move before the next movement (the required time depends on the situation, and the robot arm is set by default to run myBlockly with a minimum sleep time of no less than 0.5s), otherwise the robot arm will not be able to achieve the desired movement.

Q3: The `Run` button in the upper right corner cannot be clicked, and it is gray-green.

A: The new version of myBlockly has added the function of detecting the serial communication of the robot arm. If the robot arm is currently connected to the computer, you need to check:

- (1) Is there a background program occupying the robot arm's serial port?
- (2) Is the toolbar under the red arrow on the right closed? If it is open, you need to manually close it.

Q4: Why do I get a lot of errors after running the program?

A: Before running the program, you need to confirm a few points:

- (1) Please make sure that your robot arm's serial port number and baud rate are correct.

How to check the serial port number:

- In Windows, find the device manager and check the port.

If the port (COM and LPT) shows USB-Enhanced-SERIAL CH9102, it is a CP34X chip.

4.1 First-time self-check

If the port (COM and LPT) shows Silicon Labs CP210x USB to UART Bridge, it is a CP210X chip. The ports with these two names are the serial port numbers of your robot arm.

- In Linux, open the terminal and enter `ls/dev/tty*` and press Enter. The serial port number of the robot arm is displayed. AMA0 or USB0 is the serial port number of your robot arm.
- Open the terminal on Mac, enter `cd /dev/` and press Enter, then `ls -al tty` to find it, for example `/dev/tty.usbserial-10`.

(2) Please make sure the baud rate is correct. The baud rates of M5, myCobot 320 Pi-2020, myCobot 320 Pi-2022, myCobot 280 Jetson nano, and myCobot 280 Arudino are all 115200. The baud rates of myCobot 280 Pi, myPalletizer 260 Pi, mecahrm 270 Pi, myBuddy 280, Pro 600, etc. are all 1000000.

(3) Please make sure that the model, serial port number, and baud rate in the blue box are consistent with those in the small toolbar on the right, and match the robot arm.

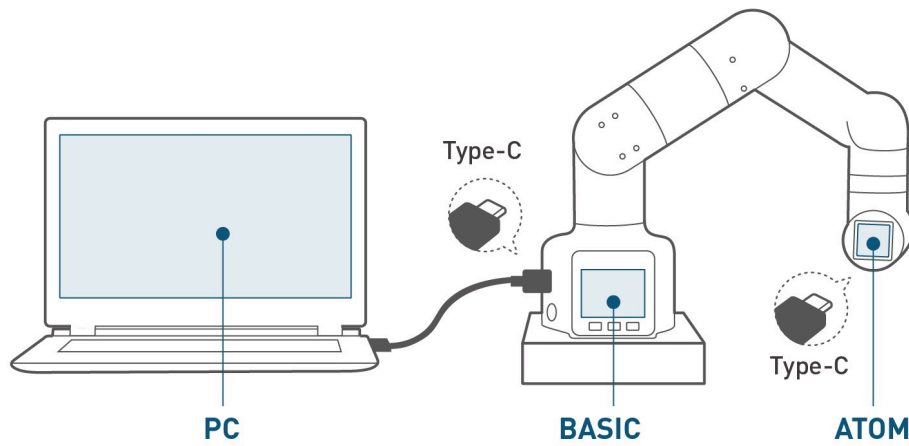
Q5: Error `MyCobot.int()` takes 2 positional arguments but 3 were given.

A: This error will appear in the old version of myBlockly because the versions of myBlockly and pymycobot do not match. Just update the versions of myBlockly and pymycobot driver library.

Q6: The result of running the program shows child process exited with code 1 A: This is not an error. All programs have finished running and returned the binary number 1. It means that everything has been successfully completed.

Communication and message commands

Note: To use the communication protocol for direct communication, you need to burn transponder in basic and the latest version of atomMain in atom



Robotic arm motion parameters

| Joint | *Joint minimum value ° | Joint maximum value ° | Joint maximum speed °/s | Joint maximum acceleration °/s ² |
|-------|------------------------|-----------------------|-------------------------|---|
| J1 | -168 | 168 | 150 | 200 |
| J2 | -135 | 135 | 150 | 200 |
| J3 | -150 | 150 | 150 | 200 |
| J4 | -145 | 145 | 150 | 200 |
| J5 | -165 | 165 | 150 | 200 |
| J6 | -180 | 180 | 150 | 200 |

| Axis | *Minimum value of coordinate mm | Maximum value of coordinate mm | Maximum velocity of coordinate mm/s | Maximum acceleration of coordinate mm/s ² |
|------|---------------------------------|--------------------------------|-------------------------------------|--|
| x | -281.45 | 281.45 | 100 | 400 |
| y | -281.45 | 281.45 | 100 | 400 |
| z | -70 | 412.76 | 100 | 400 |
| rx | -180° | 180° | 40° | 66°/s ² |
| ry | -180° | 180° | 40° | 66°/s ² |
| rz | -180° | 180° | 40° | 66°/s ² |

USB Communication Settings

Please make sure your communication settings are as follows

- Bus interface: USB Type-C connection
- Baud rate: 115200
- Data bits: 8
- Parity: None
- Stop bits: 1

Command frame description and single command analysis

The host Basic sends data to the slave, and the slave parses the data after receiving it. If the command contains a return value, the slave will return it to the host within 500ms.

Command frame sending and receiving format

All commands are in hexadecimal, and the sending and receiving formats are consistent.

Each communication command must contain the following 5 parts, of which 3 and 4 can be empty.

- **1 Command header:** 0xFE 0xFE
- Fixed
- Required
- **2 Effective command length:** 0x02 ~ 0x10
- Length of all the following commands
- Required
- **3 Command number:** 00 ~ 8F
- Multiple commands have been developed
- Can be empty
- **4 Command content:** Several
- Can be empty
- **5 Command end:** 0XFA
- Fixed
- Required

Command parsing

The host Basic sends data to the slave, and the slave parses the data after receiving it. If the command contains a return value, the slave will return it to the host within 500ms.

| Type | Data description | Data length | Description |
|---------------|------------------|-------------|--|
| Command frame | Header byte 0 | 1 | Frame header identification, 0XFE |
| | Header byte 1 | 1 | Frame header identification, 0XFE |
| | Data length byte | 1 | Different instructions correspond to different lengths of data |
| | Command byte | 1 | Depends on different commands |
| Data frame | Data | 0-16 | Data attached to the command, depending on different commands |
| End frame | End byte | 1 | Stop bit, 0XFA |

Single command analysis

Robot power on

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X10 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 10 FA

No return value

Robotic arm power off and disconnect

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X11 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 11 FA

No return value

Atom status query

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X12 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 12 FA

Return data structure

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return command frame | 0X12 |
| Data[4] | Power on/off | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Assume that Atom is powered on

Serial port return example: FE FE 03 12 01 FA**Robotic arm only powers off**

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X13 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 13 FA

No return value

Robot system detection is normal

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X14 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 14 FA

Return data structure

| Data field | Description | Data |
|------------|---------------------------------|-----------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return command frame | 0X14 |
| Data[4] | Normal connection/disconnection | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Assuming Atom connection is successful

Serial port return example: FE FE 03 14 01 FA

Command refresh mode switch (set interpolation/refresh motion mode)

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X16 |
| Data[4] | Command frame | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Set to refresh motion mode:

Serial port sending example: FE FE 03 16 01 FA

4.1 First-time self-check

Set to interpolation motion mode:

Serial port sending example: FE FE 03 16 00 FA

Robot free mode (turn off all torque output)

| Data field | Description | Data |
|------------|----------------------|-------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X1A |
| Data[4] | Open/Close | 01/00 |
| Data[5] | End frame | 0XFA |

Set to free movement mode:

Serial port sending example: FE FE 03 1A 01 FA

Check whether it is free mode

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X1B |
| Data[5] | End frame | 0XFA |

Serial port sending example: FE FE 02 1B FA

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|-------------------|--------------------------|-------------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return instruction frame | 0X1B |
| Data[4] | Open/Close | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Assuming Atom is in free movement mode

Serial port return example: FE FE 03 1B 01 FA

Read angle (read movement information)

| Data field | Description | Data |
|-------------------|----------------------|-------------|
| Data[0] | Identify frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X20 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 20 FA

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|------------|------------------------|-------------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X0E |
| Data[3] | Return command frame | 0X20 |
| Data[4] | No. 1 servo angle high | Angle1_high |
| Data[5] | No. 1 servo angle low | Angle1_low |
| Data[6] | Angle 2 high position | Angle2_high |
| Data[7] | Angle 2 low position | Angle2_low |
| Data[8] | Angle 3 high position | Angle3_high |
| Data[9] | Angle 3 low position | Angle3_low |
| Data[10] | Angle 4 high position | Angle4_high |
| Data[11] | Angle 4 low position | Angle4_low |
| Data[12] | Angle 5 high position | Angle5_high |
| Data[13] | Angle 5 low position | Angle5_low |
| Data[14] | Angle 6 high position | Angle6_high |
| Data[15] | Angle 6 low position | Angle6_low |
| Data[16] | End frame | 0XFA |

Serial port return example: FE FE 0E 20 00 8C 00 3D FF E6 FF 3F 00 AF FF 51 FA

How to get the angle of joint 1

$temp = angle1_low + angle1_high * 256$

$Angle1 = (temp \setminus 33000 \text{ ?}(temp - 65536) : temp) / 100$

Calculation method: angle value low + angle value high multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 100. If it is less than 33000, directly divide by 100

(The same applies to the rest)

Send a single angle

| Data field | Description | Data |
|------------|----------------------|------------|
| Data[0] | Identify frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X06 |
| Data[3] | Command frame | 0X21 |
| Data[4] | Servo serial number | joint_no |
| Data[5] | Angle value high | angle_high |
| Data[6] | Angle value low | angle_low |
| Data[7] | Specified speed | sp |
| Data[8] | End frame | 0XFA |

Move servo No. 1 to zero position at 20% speed

Serial port sending example: FE FE 06 21 01 00 00 14 FA

joint_no value range: 1~6

angle_high: data type byte

Calculation method: multiply the angle value by 100 and convert it to int format first Then take the high byte of the hexadecimal

angle_low: data type byte

Calculation method: multiply the angle value by 100, convert it to int format, and then take the low byte of the hexadecimal

No return value

Send all angles

| Data field | Description | Data |
|------------|---|-------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X0F |
| Data[3] | Command frame | 0X22 |
| Data[4] | No. 1 servo angle value high byte | Angle1_high |
| Data[5] | No. 1 servo angle value low byte | Angle1_low |
| Data[6] | No. 2 servo angle value high byte | Angle2_high |
| Data[7] | No. 2 servo angle value low byte | Angle2_low |
| Data[8] | No. 3 servo angle value high byte | Angle3_high |
| Data[9] | No. 3 servo angle value low byte | Angle3_low |
| Data[10] | High byte of the angle value of Servo No. 4 | Angle4_high |
| Data[11] | Low byte of the angle value of Servo No. 4 | Angle4_low |
| Data[12] | High byte of the angle value of Servo No. 5 | Angle5_high |
| Data[13] | Low byte of the angle value of Servo No. 5 | Angle5_low |
| Data[14] | High byte of the angle value of Servo No. 6 | Angle6_high |
| Data[15] | Low byte of the angle value of Servo No. 6 | Angle6_low |
| Data[16] | Specified speed | Sp |
| Data[17] | End frame | 0XFA |

Send all angles to zero/restore the machine to zero position and move at 30% speed

Serial port sending example: FE FE 0F 22 00 00 00 00 00 00 00 00 00 00 00 00 1E FA

angle1_high: data type byte

Calculation method: multiply the angle value of servo No. 1 by 100, convert it to int format first, and then take the high byte of hexadecimal

angle1_low: data type byte

Calculation method: multiply the angle value of servo No. 1 by 100, convert it to int format first, and then take the low byte of hexadecimal

(The same applies to the rest)

No return value

Read all coordinates

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X23 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 23 FA

Return data structure

| Data field | Description | Data |
|------------|---|---------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X0E |
| Data[3] | Return instruction frame | 0X23 |
| Data[4] | Specify x coordinate high | x_high |
| Data[5] | Specify x coordinate low | x_low |
| Data[6] | Specify y coordinate high | y_high |
| Data[7] | Specify y coordinate low | y_low |
| Data[8] | Specify z coordinate high | z_high |
| Data[9] | Specify z coordinate low | z_low |
| Data[10] | Specify rx coordinate high | rx_high |
| Data[11] | Specify the low bit of the rx coordinate | rx_low |
| Data[12] | Specify the high bit of the ry coordinate | ry_high |
| Data[13] | Specify the low bit of the ry coordinate | ry_low |
| Data[14] | Specify the high bit of the rz coordinate | rz_high |
| Data[15] | Specify the low bit of the rz coordinate | rz_low |
| Data[16] | End frame | 0XFA |

Serial port return example: FE FE 0E 23 01 BC FD A0 10 15 DC 66 FF 54 DE 21 FA

How to get the x coordinate

4.1 First-time self-check

$temp = x_low + x_high * 256$

$x \text{ coordinate} = (temp \setminus 33000 \ ?(temp - 65536) : temp) / 10$

Calculation method: x coordinate value low bit + x coordinate value high bit multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 10. If it is less than 33000, just divide by 10 directly

(The same applies to y coordinates and z coordinates)

How to get the rx coordinate

$temp = rx_low + rx_high * 256$

$rx \text{ coordinate} = (temp \setminus 33000 \ ?(temp - 65536) : temp) / 100$

Calculation method: x coordinate value low bit + x coordinate value high bit multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 100. If it is less than 33000, just divide by 100 directly

(The same applies to ry coordinates and rz coordinates)

Send individual coordinate parameters

| Data field | Description | Data |
|------------|-----------------------------------|-----------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X06 |
| Data[3] | Command frame | 0X24 |
| Data[4] | axis | x/y/z/rx/ry/rz |
| Data[5] | Specify xyz/rxryrz parameter high | xyz/rxryrz_high |
| Data[6] | Specify xyz/rxryrz parameter low | xyz/rxryrz_low |
| Data[7] | Specify speed | Sp |
| Data[8] | End frame | 0XFA |

Set X coordinate to 200 and target speed to 20

Serial port sending example: FE FE 06 24 01 07 D0 14 FA

Specify axis: data type byte

Value range: 1~6

xyz_high: data type byte

Calculation method: x/y/z coordinate value multiplied by 10 and then take the high byte of hexadecimal

xyz_low: data type byte

Calculation method: x/y/z coordinate value multiplied by 10 and then take the low byte of hexadecimal

4.1 First-time self-check

rxryrz_high: data type byte

Calculation method: rx/ry/rz multiplied by 100 and then take the high byte of hexadecimal

rxryrz_low: data type byte

Calculation method: rx/ry/rz multiplied by 100 and then take the low byte of hexadecimal

No return value

Send all coordinate parameters

| Data field | Description | Data |
|------------|---------------------------------------|---------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X10 |
| Data[3] | Command frame | 0X25 |
| Data[4] | Specify x coordinate high | x_high |
| Data[5] | Specify x coordinate low | x_low |
| Data[6] | Specify y coordinate high | y_high |
| Data[7] | Specify y coordinate low | y_low |
| Data[8] | Specify z coordinate high | z_high |
| Data[9] | Specify z coordinate low | z_low |
| Data[10] | Specify rx coordinate high | rx_high |
| Data[11] | Specify the low bit of rx coordinate | rx_low |
| Data[12] | Specify the high bit of ry coordinate | ry_high |
| Data[13] | Specify the low bit of ry coordinate | ry_low |
| Data[14] | Specify the high bit of rz coordinate | rz_high |
| Data[15] | Specify the low bit of rz coordinate | rz_low |
| Data[16] | Specify the speed | Sp |
| Data[17] | Mode | 0X01 |
| Data[18] | End frame | 0XFA |

Set the target position of the end of the robot arm (150.3, -68.7, 101.8, 10.18, 0, -90), target speed 10

Serial port sending example: FE FE 10 25 05 DF FD 51 03 FA BC 30 00 00 DC D8 0A 01 FA

4.1 First-time self-check

x_high: Data type byte

Calculation method: x coordinate multiplied by 10 and then take the high byte of hexadecimal

x_low: Data type byte

Calculation method: x coordinate multiplied by 10 and then take the low byte of hexadecimal

(The same applies to y-axis coordinates and z-axis coordinates)

rx_high: Data type byte

Calculation method: rx coordinate value multiplied by 100 and then take the high byte of hexadecimal

rx_low: Data type byte

Calculation method: rx coordinate value multiplied by 100 and then take the low byte of hexadecimal

(The same applies to ry-axis coordinates and rz-axis coordinates)

No return value

Program pause

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X26 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 26 FA

No return value

Is the program paused?

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X27 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 27 FA

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return command frame | 0X27 |
| Data[4] | Pause/unpause | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Assume the program is in pause state

Serial port return example: FE FE 03 27 01 FA

Program resume

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X28 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 28 FA

No return value

Program stop

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X29 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 29 FA

No return value

Whether the point is reached

| Data field | Description | Data |
|------------|--|---------------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X0E/0X0F |
| Data[3] | Command frame | 0X2A |
| Data[4] | Coordinate x high byte/No. 1 servo angle value high byte | x_high/Angle1_high |
| Data[5] | Coordinate x low byte/No. 1 servo angle value low byte | x_low/Angle1_low |
| Data[6] | Coordinate y high byte/No. 2 servo angle value high byte | y_high/Angle2_high |
| Data[7] | Coordinate y low byte/No. 2 servo angle value low byte | y_low/Angle2_low |
| Data[8] | Coordinate z high byte/No. 3 servo angle value high byte | z_high/Angle3_high |
| Data[9] | Coordinate z low byte/No. 3 servo angle value low byte | z_low/Angle3_low |
| Data[10] | Coordinate rx high bit/No. 4 servo angle value high byte | rx_high/Angle4_high |
| Data[11] | Coordinate rx low bit/No. 4 servo angle value low byte | rx_low/Angle4_low |
| Data[12] | Coordinate ry high bit/No. 5 servo angle value high byte | ry_high/Angle5_high |
| Data[13] | Coordinate ry low bit/No. 5 servo angle value low byte | ry_low/Angle5_low |
| Data[14] | Coordinate rz high bit/No. 6 servo angle value high byte | rz_high/Angle6_high |
| Data[15] | Coordinate rz low bit/No. 6 servo angle value low byte | rz_low/Angle6_low |
| Data[16] | Coordinate/angle | 0X01/0X00 |
| Data[17] | End frame | 0XFA |

Judge whether the robot has reached the origin

Serial port sending example: FE FE 0F 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 FA

x_high: data type byte

Calculation method: x coordinate multiplied by 10, first converted to int type, then take the hexadecimal high byte

x_low: data type byte

Calculation method: x coordinate multiplied by 10, first converted to int type, then take the hexadecimal low byte

(The same applies to y-axis coordinates and z-axis coordinates)

rx_high: data type byte

Calculation method: rx coordinate multiplied by 100, first converted to int type, then take the hexadecimal high byte

rx_low: data type byte

4.1 First-time self-check

Calculation method: rx coordinate multiplied by 100, first converted to int type Then take the low byte of hexadecimal

(The same applies to the ry axis coordinates and the rz axis coordinates)

angle_high: data type byte

Calculation method: multiply the angle value by 100, convert it to int format first, and then take the high byte of hexadecimal

angle_low: data type byte

Calculation method: multiply the angle value by 100, convert it to int format first, and then take the low byte of hexadecimal

Type: data type byte (not used yet)

Return data structure

| Data field | Description | Data |
|------------|-------------------------------|-----------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return instruction frame | 0X2a |
| Data[4] | Arrived point/unreached point | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Assume the robot arm has not reached the specified point

Serial port return example: FE FE 03 2A 00 FA

Robotic arm motion detection

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X2B |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 2B FA

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return command frame | 0X2B |
| Data[4] | Moving/Not Moving | 0X01/0X00 |
| Data[5] | End Frame | 0XFA |

Assuming the program is in motion

Serial port return example: FE FE 03 2B 01 FA

jog-Joint direction movement

| Data field | Description | Data |
|------------|-----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X05 |
| Data[3] | Command frame | 0X30 |
| Data[4] | Joint servo number | Joint |
| Data[5] | Joint servo direction | direction |
| Data[6] | Specified speed | sp |
| Data[7] | End Frame | 0XFA |

Set servo No. 1 to rotate clockwise at 20% speed

Serial port sending example: FE FE 05 30 01 01 14 FA

Joint number range: 1~6

di: Data type byte Value range 0 and 1

sp: Data type byte Value range 0-100

No return value

jod-absolute control

| Data field | Description | Data |
|------------|-------------------------------------|------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X06 |
| Data[3] | Command frame | 0X31 |
| Data[4] | Joint servo number | Joint |
| Data[5] | Joint servo angle value high byte | Angle_high |
| Data[6] | Low byte of joint servo angle value | Angle_low |
| Data[7] | Specified speed | sp |
| Data[8] | End frame | 0XFA |

Set servo No. 1 to 45°, speed 20

Serial port sending example: FE FE 06 31 01 11 94 14 FA

Joint number value range: 1~6

Angle_high: Data type byte

Calculation method: Multiply the angle value by 100, convert it to int format first, and then take the high byte of hexadecimal

Angle_low: Data type byte

Calculation method: Multiply the angle value by 100, convert it to int format first, and then take the low byte of hexadecimal

sp: Data type byte, value range 0-100

No return value

jog-coordinate direction movement

| Data field | Description | Data |
|------------|-----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X05 |
| Data[3] | Command frame | 0X32 |
| Data[4] | Specified coordinates | axis |
| Data[5] | Joint servo direction | di |
| Data[6] | Specified speed | sp |
| Data[7] | End frame | 0XFA |

Set the robot arm to move in the x direction, speed 20

Serial port sending example: FE FE 05 32 01 01 14 FA

axis value range: 1~6, representing x, y, z, rx, ry, rz respectively

di: data type byte value range 0 and 1

sp: data type byte value range 0-100

No return value

jog-stepping mode

| Data field | Description | Data |
|------------|-----------------------------------|------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X06 |
| Data[3] | Command frame | 0X33 |
| Data[4] | Joint servo serial number | Joint |
| Data[5] | Joint servo angle value high byte | Angle_high |
| Data[6] | Joint servo angle value low byte | Angle_low |
| Data[7] | Specified speed | sp |
| Data[8] | End frame | 0XFA |

Set the angle of servo No. 1 to increase by 45 and rotate at 20% speed

Serial port sending example: FE FE 06 33 01 11 94 14 FA

4.1 First-time self-check

Joint serial number value range: 1~6

Angle_high: Data type byte

Calculation method: Multiply the angle value by 100, convert to int format first, and then take the high byte of hexadecimal

Angle_low: Data type byte

Calculation method: Multiply the angle value by 100, convert to int format first, and then take the low byte of hexadecimal

sp: Data type byte Value range 0-100

No return value

Send potential value

| Data field | Description | Data |
|------------|---------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X06 |
| Data[3] | Command frame | 0X3A |
| Data[4] | Joint servo serial number | Joint |
| Data[5] | Potential value high | Encoder_high |
| Data[6] | Potential value low | Encoder_low |
| Data[7] | Specified speed | sp |
| Data[8] | End frame | 0XFA |

Example, set joint 5 to 2048 potential and rotate at 20% speed

Serial port sending example: FE FE 06 3A 05 08 00 14 FA

Joint number range: 1~6

Joint: Data type byte

Encoder_high: Data type byte

Calculation method: Take the high bit of the potential value (hexadecimal)

Encoder_low: Data type byte

Calculation method: Take the low bit of the potential value (hexadecimal)

No return value

Get potential value

| Data field | Description | Data |
|------------|-------------------|-------|
| Data[0] | Identify frame | 0XFE |
| Data[1] | Identify frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X3B |
| Data[4] | Joint number | joint |
| Data[5] | End frame | 0XFA |

Get the potential value of servo No. 2

Serial port sending example: FE FE 03 3B 02 FA

Joint number range: 1-6

Return data structure

| Data field | Description | Data |
|------------|-----------------------------|--------------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X04 |
| Data[3] | Return command frame | 0X3B |
| Data[4] | Servo potential value high | Encoder_high |
| Data[5] | Servo potential value low | Encoders_low |
| Data[6] | End frame | 0XFA |

Serial port return example: FE FE 04 3B 08 07 FA

How to calculate the potential value

Potential value = potential value low bit + potential value high bit * 256

Send the potential values of six servos

| Data field | Description | Data |
|------------|---|----------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X0F |
| Data[3] | Command frame | 0X3C |
| Data[4] | High byte of potential value of servo No. 1 | encoder_1_high |
| Data[5] | Low byte of potential value of servo No. 1 | encoder_1_low |
| Data[6] | High byte of potential value of servo No. 2 | encoder_2_high |
| Data[7] | Low byte of potential value of servo No. 2 | encoder_2_low |
| Data[8] | High byte of potential value of servo No. 3 | encoder_3_high |
| Data[9] | Low byte of potential value of servo No. 3 | encoder_3_low |
| Data[10] | No. 4 servo potential value high byte | encoder_4_high |
| Data[11] | No. 4 servo potential value low byte | encoder_4_low |
| Data[12] | No. 5 servo potential value high byte | encoder_5_high |
| Data[13] | No. 5 servo potential value low byte | encoder_5_low |
| Data[14] | No. 6 servo potential value high byte | encoder_6_high |
| Data[15] | No. 6 servo potential value low byte | encoder_6_low |
| Data[16] | Specified speed | Sp |
| Data[17] | End frame | 0XFA |

The potential value of all motors sent is 2048, and the speed is 20

Serial port sending example: FE FE 0F 3C 08 00 08 00 08 00 08 00 08 00 08 00 14 FA

(Refer to the above for sending a single potential value)

encoder_1_high: Data type byte

Calculation method: The potential value of servo No. 1 is first converted to int type and then the hexadecimal high byte is taken

encoder_1_low: Data type byte

Calculation method: The potential value of servo No. 1 is first converted to int type and then the hexadecimal low byte is taken

(The same applies to the rest)

Sp: Data type byte Value range: 0~100

No return value

Read the potential values of six servos

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X3D |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 3D FA

Return data structure

| Data field | Description | Data |
|------------|---|----------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X0E |
| Data[3] | Command frame | 0X3D |
| Data[4] | High byte of the potential value of Servo No. 1 | encoder_1_high |
| Data[5] | Low byte of the potential value of Servo No. 1 | encoder_1_low |
| Data[6] | High byte of the potential value of Servo No. 2 | encoder_2_high |
| Data[7] | Low byte of the potential value of Servo No. 2 | encoder_2_low |
| Data[8] | Servo 3 potential value high byte | encoder_3_high |
| Data[9] | Servo 3 potential value low byte | encoder_3_low |
| Data[10] | Servo 4 potential value high byte | encoder_4_high |
| Data[11] | Servo 4 potential value low byte | encoder_4_low |
| Data[12] | Servo 5 potential value high byte | encoder_5_high |
| Data[13] | Servo 5 potential value low byte | encoder_5_low |
| Data[14] | Servo 6 potential value high byte | encoder_6_high |
| Data[15] | Servo 6 potential value low byte | encoder_6_low |
| Data[16] | End frame | 0XFA |

Assume that all joints of the current robot arm are at 0 position

Serial port return example: FE FE 0E 3D 08 00 08 00 08 00 08 00 08 00 08 00 FA

4.1 First-time self-check

How to calculate the potential value

Potential value = potential value low bit + potential value high bit * 256

Set speed

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Instruction frame | 0X41 |
| Data[4] | Specified speed | sp |
| Data[5] | End frame | 0XFA |

Sp: Data type byte Value range: 0~100

Set the current speed to 50%

Serial port sending example: FE FE 03 41 32 FA

No return value

Read the minimum angle of the joint

| Data field | Description | Data |
|------------|---------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X4A |
| Data[4] | Joint servo serial number | Joint_number |
| Data[5] | End frame | 0XFA |

Read the minimum angle of joint No. 2

Serial port sending example: FE FE 03 4A 02 FA

joint_no value range: 1-6

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|------------|-----------------------------|--------------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X05 |
| Data[3] | Return command frame | 0X4A |
| Data[4] | Joint servo number | Joint_number |
| Data[5] | Servo angle value high | Angle_high |
| Data[6] | Servo angle value low | Angle_low |
| Data[7] | End frame | 0XFA |

Serial port return example: FE FE 05 4A 02 F9 F2 FA

How to get 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 value high multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 10. If it is less than 33000, divide by 10 directly

Read the maximum angle of the joint

| Data field | Description | Data |
|------------|---------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X4B |
| Data[4] | Joint servo serial number | joint_number |
| Data[5] | End frame | 0XFA |

joint_no value range: 1-6

Read the maximum angle of joint 2

Serial port sending example: FE FE 03 4B 02 FA

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|------------|-----------------------------|--------------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X05 |
| Data[3] | Return command frame | 0X4B |
| Data[4] | Joint servo number | joint_number |
| Data[5] | Servo angle value high | Angle_high |
| Data[6] | Servo angle value low | Angle_low |
| Data[7] | End frame | 0XFA |

Serial port return example: FE FE 05 4B 02 06 72 FA

How to get the maximum angle of the joint

temp = angle1_low+angle1_high*256

Angle1= (temp \ 33000 ?(temp – 65536) : temp) /10

Calculation method: angle value low bit + angle value high bit multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 10. If it is less than 33000, just divide by 10

Set the minimum angle of the joint

| Data field | Description | Data |
|------------|-----------------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X05 |
| Data[3] | Command frame | 0X4C |
| Data[4] | Joint servo number | Joint_number |
| Data[5] | Joint servo angle value high byte | Angle_high |
| Data[6] | Joint servo angle value low byte | Angle_low |
| Data[7] | End frame | 0XFA |

Set the minimum angle of joint 2 to 0

joint_no value range: 1-6

angle1_high: data type byte

4.1 First-time self-check

Calculation method: multiply the servo angle value by 100, convert it to int format first, and then take the high byte of hexadecimal

angle1_low: data type byte

Calculation method: multiply the servo angle value by 100, convert it to int format first, and then take the low byte of hexadecimal

Serial port sending example: FE FE 05 4C 02 00 00 FA

No return value

Set the maximum angle of the joint

| Data field | Description | Data |
|------------|-----------------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X05 |
| Data[3] | Command frame | 0X4D |
| Data[4] | Joint servo number | Joint_number |
| Data[5] | Joint servo angle value high byte | Angle_high |
| Data[6] | Joint servo angle value low byte | Angle_low |
| Data[7] | End frame | 0XFA |

Set the maximum angle of joint 2 to 45

Joint_no value range: 1-6

angle1_high: data type byte

Calculation method: Multiply the servo angle value by 100 and convert it to int format first Then take the high byte of hexadecimal

angle1_low: data type byte

Calculation method: Multiply the servo angle value by 100, convert it to int format first, and then take the low byte of hexadecimal

Serial port sending example: FE FE 05 4C 02 11 94 FA

No return value

View connection

| Data field | Description | Data |
|------------|---------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X50 |
| Data[4] | Joint servo serial number | Joint_number |
| Data[5] | End frame | 0XFA |

joint_no value range: 1-6

Check whether servo No. 1 is connected

Serial port sending example: FE FE 03 50 01 FA

Return data structure

| Data field | Description | Data |
|------------|-----------------------------|--------------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X04 |
| Data[3] | Command frame | 0X50 |
| Data[4] | Joint servo number | Joint_number |
| Data[5] | Connected/unconnected | 0X01/0X00 |
| Data[6] | End frame | 0XFA |

Servo No. 1 is connected normally

Serial port return example: FE FE 04 50 01 01 FA

Check if all servos are powered on

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X51 |
| Data[4] | End frame | 0XFA |

4.1 First-time self-check

Serial port sending example: FE FE 02 51 FA

Return data structure

| Data field | Description | Data |
|------------|-----------------------------|-----------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X03 |
| Data[3] | Command frame | 0X51 |
| Data[4] | Power on/off | 0X01/0X00 |
| Data[5] | End frame | 0XFA |

Not all servos are powered on Serial port return example: FE FE 03 51 01 FA

Read servo parameters

| Data field | Description | Data |
|------------|---------------------------|----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Command frame | 0X53 |
| Data[4] | Joint servo serial number | joint_no |
| Data[5] | Data address | data_id |
| Data[6] | End frame | 0XFA |

Read the proportional parameters of position P of servo No. 1

Serial port sending example: FE FE 04 53 01 15 FA

joint_no value range 1~6

Data_id: data type byte, value as shown in the following table

4.1 First-time self-check

| Address | Function | Value range | Initial value | Value analysis |
|---------|------------------------|-------------|---------------|---|
| 20 | LED alarm | 0-254 | 0 | 1\0 = Turn on or off LED alarm |
| 21 | Position loop P | 0-254 | 10 | Proportional coefficient of controlling motor |
| 22 | Position loop I | 0-254 | 0 | Differential coefficient of controlling motor |
| 23 | Position loop D | 0-254 | 1 | Integral coefficient of controlling motor |
| 24 | Minimum starting force | 0-1000 | 0 | Set the minimum output torque 1000 = 100% |

Return data structure

| Data field | Description | Data |
|------------|-----------------------------|------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X03 |
| Data[3] | Return instruction frame | 0X53 |
| Data[4] | Return data | data |
| Data[5] | End frame | 0XFA |

Serial port return example: FE FE 03 53 10 FA

Set servo parameters of steering gear

| Data field | Description | Data |
|------------|---------------------------|----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification 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[6] | Data | data |
| Data[7] | End frame | 0XFA |

Set the position P ratio parameter of servo No. 1 to 1

4.1 First-time self-check

Serial port sending example: FE FE 05 52 01 15 01 FA

joint_no value range: 1~6

No return value

data_id value is as follows

| Address | Function | Value range | Initial value | Value analysis |
|---------|------------------------|-------------|---------------|---|
| 20 | LED alarm | 0-254 | 0 | 1_0 = Turn LED alarm on or off |
| 21 | Position loop P | 0-254 | 10 | Proportional coefficient of the control motor |
| 22 | Position loop I | 0-254 | 0 | Differential coefficient of the control motor |
| 23 | Position loop D | 0-254 | 1 | Integral coefficient of the control motor |
| 24 | Minimum starting force | 0-1000 | 0 | Set the minimum output torque 1000 = 100% |

Set the servo zero point

| Data field | Description | Data |
|------------|---------------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X54 |
| Data[4] | Joint servo serial number | joint_number |
| Data[5] | End frame | 0XFA |

Set the zero position of servo No. 1

Serial port sending example: FE FE 03 54 01 FA

joint_number:1~6

No return value

Brake a single motor

| Data field | Description | Data |
|------------|----------------------|--------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X55 |
| Data[4] | Joint servo number | joint_number |
| Data[5] | End frame | 0XFA |

Brake servo No. 1

joint_number:1~6

Serial port sending example: FE FE 03 55 01 FA

No return value

Power off a single motor

| Data field | Description | Data |
|------------|----------------------|----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X56 |
| Data[4] | Servo serial number | Servo_no |
| Data[5] | End frame | 0XFA |

Power off servo No. 3

Serial port sending example: FE FE 03 56 03 FA

Servo_no: 1~6

No return value

Power on a single motor

| Data field | Description | Data |
|------------|----------------------|----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X57 |
| Data[4] | Servo number | Servo_no |
| Data[5] | End frame | 0XFA |

Power on servo No. 1

Serial port sending example: FE FE 03 57 01 FA

Servo_no:1~6

No return value

Set atom pin mode

| Data field | Description | Data |
|------------|----------------------|-------------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Command frame | 0X60 |
| Data[4] | Pin number | pin_no |
| Data[5] | Input/output | 00X00/00X01 |
| Data[6] | End frame | 0XFA |

Set atom pin22 to input mode

Serial port sending example: FE FE 04 60 16 00 FA

Pin_no: Data type byte

Pin_mode: 0/1

No return value

Set Atom IO (setDigitalOutput)

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Instruction frame | 0X61 |
| Data[4] | Pin number | Pin_no |
| Data[5] | Level signal | 0X00/0X01 |
| Data[6] | End frame | 0XFA |

Set pin P23 to high level

Serial port sending example: FE FE 04 61 17 01 FA

No return value

Read Atom IO (getDigitalInput)

| Data field | Description | Data |
|------------|----------------------|--------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Instruction frame | 0X62 |
| Data[4] | Pin number | pin_no |
| Data[5] | End frame | 0XFA |

Read the level signal of pin P22

Serial port sending example: FE FE 03 62 16 FA

Return data structure

4.1 First-time self-check

| Data field | Description | Data |
|------------|-----------------------------|-----------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X04 |
| Data[3] | Return instruction frame | 0X62 |
| Data[4] | Pin number | pin_no |
| Data[5] | Level signal | 0X00/0X01 |
| Data[6] | End frame | 0XFA |

Assume that pin P22 is high level

Serial port return example: FE FE 04 62 16 01 FA

Read the gripper angle

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X65 |
| Data[6] | End frame | 0XFA |

Serial port sending example: FE FE 02 65 FA

Return data structure

| Data field | Description | Data |
|------------|-----------------------|-------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X65 |
| Data[4] | Gripper opening range | value |
| Data[6] | End frame | 0XFA |

value: 0-100%

Assume the gripper is in full open state

Serial port return example: FE FE 03 65 64 FA

Gripper opening size = $6 * 16 + 4 = 100$

Set the gripper mode

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Command frame | 0X66 |
| Data[4] | Gripper open/close | 0X00/0X01 |
| Data[5] | Speed | Sp |
| Data[6] | End frame | 0XFA |

Set the gripper to open at a speed of 50

Serial port sending example: FE FE 04 66 00 32 FA

No return value

Set the gripper angle

| Data field | Description | Data |
|------------|-----------------------|-------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Command frame | 0X67 |
| Data[4] | Gripper opening range | value |
| Data[6] | Speed | Sp |
| Data[7] | End frame | 0XFA |

Assume the gripper is open 50% and the speed is 20

Serial port sending example: FE FE 04 67 32 14 FA

value can be directly converted to hexadecimal

No return value

Set the gripper to zero point

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Instruction frame | 0X68 |
| Data[4] | End frame | 0XFA |

Set the gripper's current position to zero point

Serial port sending example: FE FE 02 68 FA

Detect whether the gripper is moving

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X69 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 69 FA

Return data structure

| Data field | Description | Data |
|------------|----------------------|-------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X69 |
| Data[4] | Stop/Move | 00/01 |
| Data[5] | End frame | 0XFA |

Assume the gripper is in the stopped state

Serial port return example: FE FE 03 69 00 FA

Set the color of the RGB light on the atom screen

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification 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 |

Set RGB to blue

Serial port sending example: FE FE 05 6A 00 00 FF FA

No return value

Set the base IO output

| Data field | Description | Data |
|------------|----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Command frame | 0Xa0 |
| Data[4] | Pin number | Pin_no |
| Data[5] | Level signal | 0X00/0X01 |
| Data[6] | End frame | 0XFA |

Set pin 2 to output high level

Serial port sending example: FE FE 04 a0 02 01 FA

Read base IO output

| Data field | Description | Data |
|------------|----------------------|--------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0Xa1 |
| Data[4] | Pin number | Pin_no |
| Data[5] | End frame | 0XFA |

Serial port sending example: FE FE 03 a1 02 FA

Return data structure

| Data field | Description | Data |
|------------|-----------------------------|-----------|
| Data[0] | Return identification frame | 0XFE |
| Data[1] | Return identification frame | 0XFE |
| Data[2] | Return data length frame | 0X04 |
| Data[3] | Return instruction frame | 0Xa1 |
| Data[4] | Pin number | Pin_no |
| Data[5] | Level signal | 0X00/0X01 |
| Data[6] | End frame | 0XFA |

Assume that pin 2 is high level

Serial port return example: FE FE 04 a1 02 01 FA

Get WiFi account & password

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0Xb1 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 b1 FA

Serial port return example: ssid: MyCobotWiFi2.4G password: mycobot123

4.1 First-time self-check

ssid: WiFi account

password: WiFi password

Set port number

| Data field | Description | Data |
|------------|-----------------------|-----------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X04 |
| Data[3] | Command frame | 0Xb2 |
| Data[4] | Port number high byte | port_high |
| Data[5] | Port number low byte | port_low |
| Data[6] | End frame | 0XFA |

Assume that the port number is set to 7000

Serial port sending example: FE FE 04 b2 1b 58 FA

port_high: port number hexadecimal high byte

port_low: port number hexadecimal low byte

No return value

Set tool coordinate system

| Data field | Description | Data |
|------------|---|---------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X0E |
| Data[3] | Command frame | 0X81 |
| Data[4] | Specify the high bit of the x coordinate | x_high |
| Data[5] | Specify the low bit of the x coordinate | x_low |
| Data[6] | Specify the high bit of the y coordinate | y_high |
| Data[7] | Specify the low bit of the y coordinate | y_low |
| Data[8] | Specify the high bit of the z coordinate | z_high |
| Data[9] | Specify the low bit of the z coordinate | z_low |
| Data[10] | Specify the high bit of the rx coordinate | rx_high |
| Data[11] | Specify the low bit of the rx coordinate | rx_low |
| Data[12] | Specify the high bit of the ry coordinate | ry_high |
| Data[13] | Specify the low bit of the ry coordinate | ry_low |
| Data[14] | Specify the high bit of the rz coordinate | rz_high |
| Data[15] | Specify the low bit of the rz coordinate | rz_low |
| Data[16] | End frame | 0XFA |

Assume that (0, 0, 50, 0, 0, 0) is set as the tool coordinate system

Serial port sending example: FE FE 0E 81 00 00 00 00 13 88 00 00 00 00 00 00 FA

No return value

Get tool coordinate system

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X82 |
| Data[6] | End frame | 0XFA |

4.1 First-time self-check

Serial port sending example: FE FE 02 82 FA

Return data structure

| Data field | Description | Data |
|------------|---|---------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X0E |
| Data[3] | Return command frame | 0X82 |
| Data[4] | Specify the high bit of the x coordinate | x_high |
| Data[5] | Specify the low bit of the x coordinate | x_low |
| Data[6] | Specify the high bit of the y coordinate | y_high |
| Data[7] | Specify the low bit of the y coordinate | y_low |
| Data[8] | Specify the high bit of the z coordinate | z_high |
| Data[9] | Specify the low bit of the z coordinate | z_low |
| Data[10] | Specify the high bit of the rx coordinate | rx_high |
| Data[11] | Specify the low bit of the rx coordinate | rx_low |
| Data[12] | Specify the high bit of the ry coordinate | ry_high |
| Data[13] | Specify the low bit of the ry coordinate | ry_low |
| Data[14] | Specify the high bit of the rz coordinate | rz_high |
| Data[15] | Specify the low bit of the rz coordinate | rz_low |
| Data[16] | End frame | 0XFA |

Serial port return example: FE FE 0E 82 00 00 00 00 13 88 00 00 00 00 00 00 FA

How to get the x coordinate

$temp = x_low + x_high * 256$

$x \text{ coordinate} = (temp \setminus 33000 ? (temp - 65536) : temp) / 10$

Calculation method: x coordinate value low + x coordinate value high multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 10. If it is less than 33000, directly divide by 10

(The same applies to y coordinate and z coordinate)

How to get the rx coordinate

$temp = rx_low + rx_high * 256$

$x \text{ coordinate} = (temp \setminus 33000 ? (temp - 65536) : temp) / 100$

4.1 First-time self-check

Calculation method: x coordinate value low bit + x coordinate value high bit multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 100. If it is less than 33000, divide by 100 directly

(ry coordinate and rz coordinate are the same)

Set the world coordinate system

| Data field | Description | Data |
|------------|---|---------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X0E |
| Data[3] | Instruction frame | 0X83 |
| Data[4] | Specify x coordinate high bit | x_high |
| Data[5] | Specify x coordinate low bit | x_low |
| Data[6] | Specify y coordinate high bit | y_high |
| Data[7] | Specify the low bit of the y coordinate | y_low |
| Data[8] | Specify the high bit of the z coordinate | z_high |
| Data[9] | Specify the low bit of the z coordinate | z_low |
| Data[10] | Specify the high bit of the rx coordinate | rx_high |
| Data[11] | Specify the low bit of the rx coordinate | rx_low |
| Data[12] | Specify the high bit of the ry coordinate | ry_high |
| Data[13] | Specify the low bit of the ry coordinate | ry_low |
| Data[14] | Specify the high bit of the rz coordinate | rz_high |
| Data[15] | Specify the low bit of the rz coordinate | rz_low |
| Data[16] | End frame | 0XFA |

Assume that (0, 0, 50, 0, 0, 0) is set as the world coordinate system

Serial port sending example: FE FE 0E 83 00 00 00 00 13 88 00 00 00 00 00 00 FA

No return value

Get the world coordinate system

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Instruction frame | 0X84 |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 82 FA

Return data structure

| Data field | Description | Data |
|------------|--|---------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X0E |
| Data[3] | Return command frame | 0X84 |
| Data[4] | Specify x coordinate high | x_high |
| Data[5] | Specify x coordinate low | x_low |
| Data[6] | Specify y coordinate high | y_high |
| Data[7] | Specify y coordinate low | y_low |
| Data[8] | Specify z coordinate high | z_high |
| Data[9] | Specify z coordinate low | z_low |
| Data[10] | Specify rx coordinate high | rx_high |
| Data[11] | Specify rx coordinate low | rx_low |
| Data[12] | Specify ry coordinate high | ry_high |
| Data[13] | Specify ry coordinate low | ry_low |
| Data[14] | Specify rz coordinate high | rz_high |
| Data[15] | Specify the low bit of the rz coordinate | rz_low |
| Data[16] | End frame | 0XFA |

Serial port return example: FE FE 0E 84 00 00 00 00 13 88 00 00 00 00 00 00 FA

How to get the x coordinate

temp = x_low + x_high*256

4.1 First-time self-check

$$x \text{ coordinate} = (\text{temp} \setminus 33000 \ ?(\text{temp} - 65536) : \text{temp})/10$$

Calculation method: x coordinate value low bit + x coordinate value high bit multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 100. If it is less than 33000, directly divide by 10

(The same applies to the y coordinate and the z coordinate)

How to get the rx coordinate

$$\text{temp} = \text{rx_low} + \text{rx_high} * 256$$

$$x \text{ coordinate} = (\text{temp} \setminus 33000 \ ? (\text{temp} - 65536) : \text{temp}) / 100$$

Calculation method: x coordinate value low bit + x coordinate value high bit multiplied by 256 First determine whether it is greater than 33000 If it is greater than 33000, subtract 65536 and finally divide by 100. If it is less than 33000, directly divide by 100

(ry coordinate and rz coordinate are the same)

Set base coordinate system

| Data field | Description | Data |
|------------|----------------------------------|-------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Instruction frame | 0X85 |
| Data[4] | Base coordinate/world coordinate | 00/01 |
| Data[5] | End frame | 0XFA |

Assume that the coordinate system is set to the world coordinate system

Serial port sending example: FE FE 03 85 01 FA

No return value

Get the base coordinate system

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Instruction frame | 0X86 |
| Data[4] | End frame | 0XFA |

4.1 First-time self-check

Serial port sending example: FE FE 02 86 FA

Return data structure

| Data field | Description | Data |
|------------|------------------------------------|-------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return command frame | 0X86 |
| Data[4] | Base coordinates/world coordinates | 00/01 |
| Data[4] | End frame | 0XFA |

Serial port return example: FE FE 03 86 01 FA

Set end coordinate system

| Data field | Description | Data |
|------------|----------------------|-------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X03 |
| Data[3] | Command frame | 0X89 |
| Data[4] | Flange/tool | 00/01 |
| Data[5] | End frame | 0XFA |

Assume that the end coordinate system is set to tool

Serial port sending example: FE FE 03 89 01 FA

No return value

Get the end coordinate system

| Data field | Description | Data |
|------------|----------------------|------|
| Data[0] | Identification frame | 0XFE |
| Data[1] | Identification frame | 0XFE |
| Data[2] | Data length frame | 0X02 |
| Data[3] | Command frame | 0X8a |
| Data[4] | End frame | 0XFA |

Serial port sending example: FE FE 02 8a FA

Return data structure

| Data field | Description | Data |
|------------|----------------------|-------|
| Data[0] | Return frame header | 0XFE |
| Data[1] | Return frame header | 0XFE |
| Data[2] | Return length frame | 0X03 |
| Data[3] | Return command frame | 0X8a |
| Data[4] | Flange/Tool | 00/01 |
| Data[4] | End frame | 0XFA |

Serial port return example: FE FE 03 8a 01 FA

Appendix:

Added corresponding coordinate transformation programs in the ATOM library and kinematics library. The specific implementation methods are as follows:

1. Change the end coordinate system
2. The end coordinate system can be set through the setEndType and getEndType functions. EndType::FLANGE sets the end to the flange, and EndType::TOOL sets the end to the tool end.
3. The coordinate information of the tool can be set through the setToolReference and getToolReference functions. When setting, the flange coordinate system is used as the relative coordinate system, and the tool end information is relative to the flange coordinate system.
4. After setting EndType to FLANGE, the GetCoords and WriteCoords methods are calculated based on the flange position.
5. After setting EndType to TOOL, the GetCoords and WriteCoords methods are calculated based on the tool end position.
6. Change the base coordinate system

4.1 First-time self-check

7. The base coordinate system can be set through the `setReferenceFrame` function. `RType::BASE` uses the robot base as the base coordinate, and `RType::WORLD` uses the world coordinate system as the base coordinate. The `getReferenceFrame` function is used to read the current base coordinate system type.
8. The `setWorldReference` and `getWorldReference` functions can be used to set and read the base coordinate system information. When setting, the world coordinate system is used as the relative coordinate system, and the position information of the robot's base relative to the world coordinate system is input.
9. When the base coordinate system is the base, the `GetCoords` and `WriteCoords` methods both use the base as the reference coordinate system.
10. When the base coordinate system is the world coordinate system, the `GetCoords` and `WriteCoords` methods both use the world coordinate system as the reference coordinate system.

Communication related changes (temporary)

Now add the setting and reading of the end coordinate system, the setting and reading of the world coordinate system, the setting and reading of the current reference coordinate system, the setting and reading of the end type, the setting and reading of the movement method, and the sending and receiving of the robot information.

These communications are temporarily set to 0x80 to 0x8A

In the `ParameterList.h` file, add a new `roboticMessages` space for adding robot communication information. Now only temporarily add the prompt of "no inverse solution", which can be added later.

The simple design idea of `MOVEL` function is as follows:

Calculate the Euclidean distance between the initial point and the target point, and insert an interpolation point every 10mm based on the Euclidean distance. If there is no inverse solution for the interpolation point, search for an inverse solution in the adjacent space of positive and negative $\pi/30$ in the three directions of the unchanged position, mainly to avoid singular values and some special positions where the solution cannot be found.

The point sending interval of `MOVEL` and `JOG` is changed to dynamic time. The moving time is calculated according to the maximum joint moving distance between the two points, and then the moving time minus the specific time is used as the time interval.

Chapter 7 Successful Cases

myCobot 280 series robot arms support more than ten kinds of accessories, including bases, end extensions, peripheral products, etc. Multiple accessories can be stacked to complete complex project applications and meet the needs of commercial exhibitions, such as robot application model display, educational teaching package display, and industrial 4.0 application scenario display. Supports multiple mainstream programming languages such as python and C++ to meet the diverse needs of developers.

User case display video

[【Application Case】 Elephant Robot myCobot Elephant Robot Arm Creative Video Collection](#)

280 PLC IO Interactive Control Case

1. Functional Effect Description

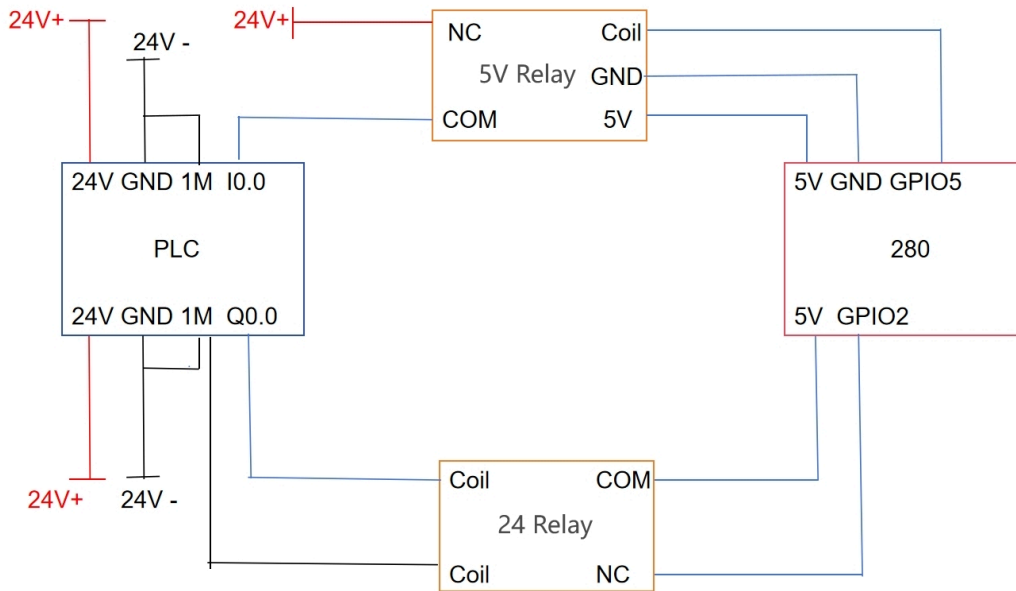
After receiving the IO signal from the PLC, the robot arm will perform an action to return each joint to zero position

2. Principle Description

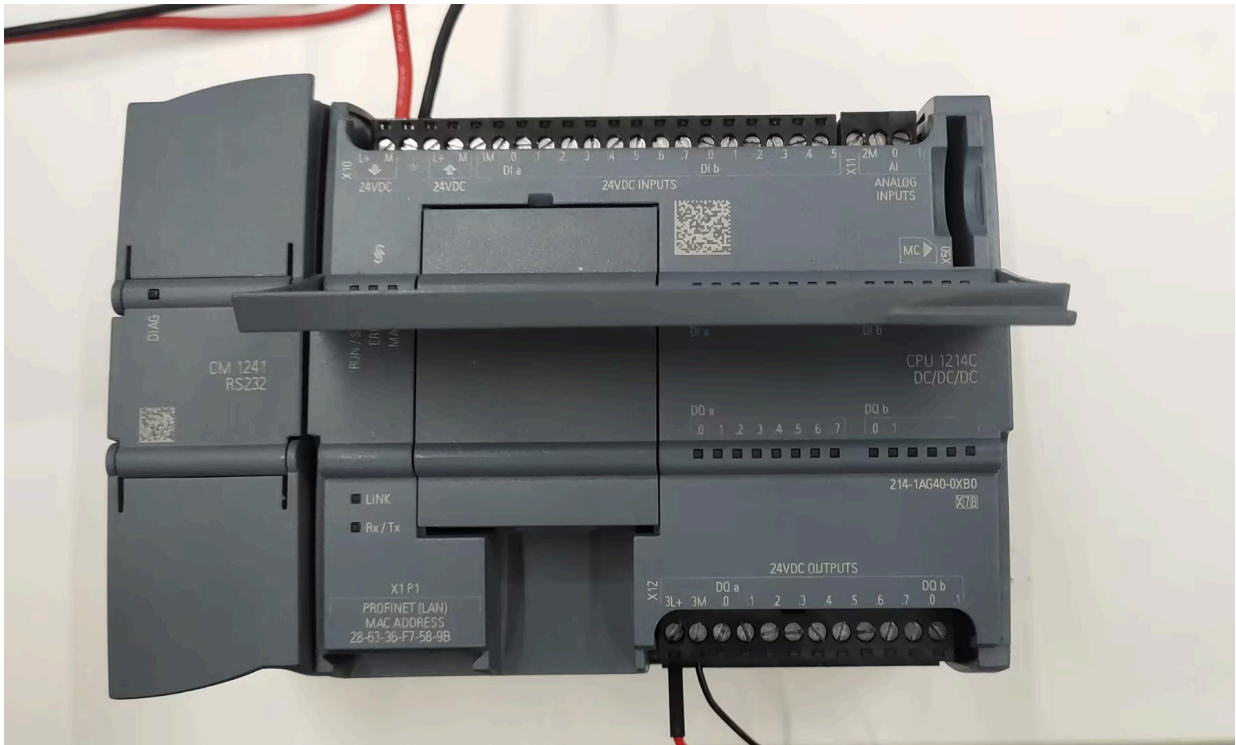
Since the input and output of the robot arm is 3.3V and the input and output of the PLC is 24V, a 5V relay and a 24V relay are required. The output end of the robot arm will first output a signal to energize the 5V relay coil, connect the normally open contacts, and pass the 24V signal to the input end of the PLC. After the PLC collects the input signal, the PLC output end will output a signal to energize the 24V relay coil, connect the normally open contacts, and pass the 3.3V signal to the input end of the robot arm. After the robot arm collects the input signal, it will perform an action of returning each joint to zero position

3. Hardware Link

Overall connection diagram



Wiring of the input of the robot arm and the output of the PLC First connect the PLC to a 24V power supply



Then connect the PLC output to the 24V relay coil

4.1 First-time self-check

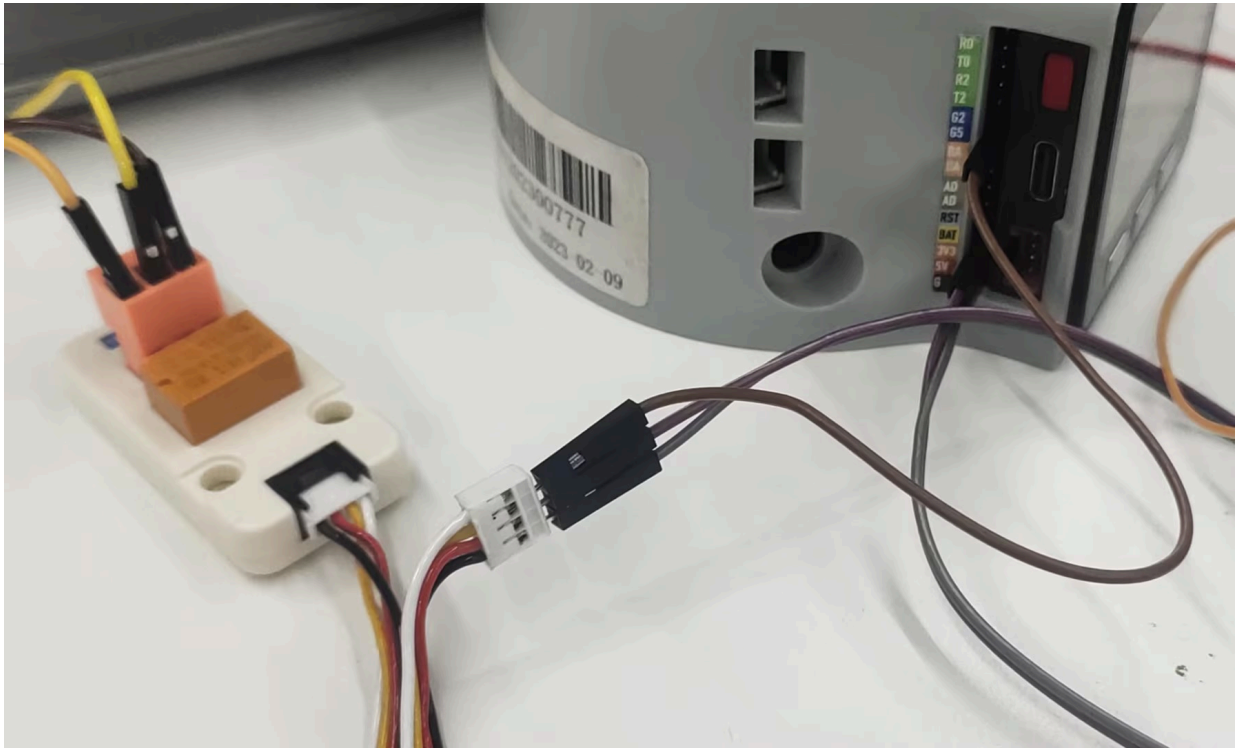


Connect the robot's GPIO2 and 3.3V to the normally open contact of the 24V relay

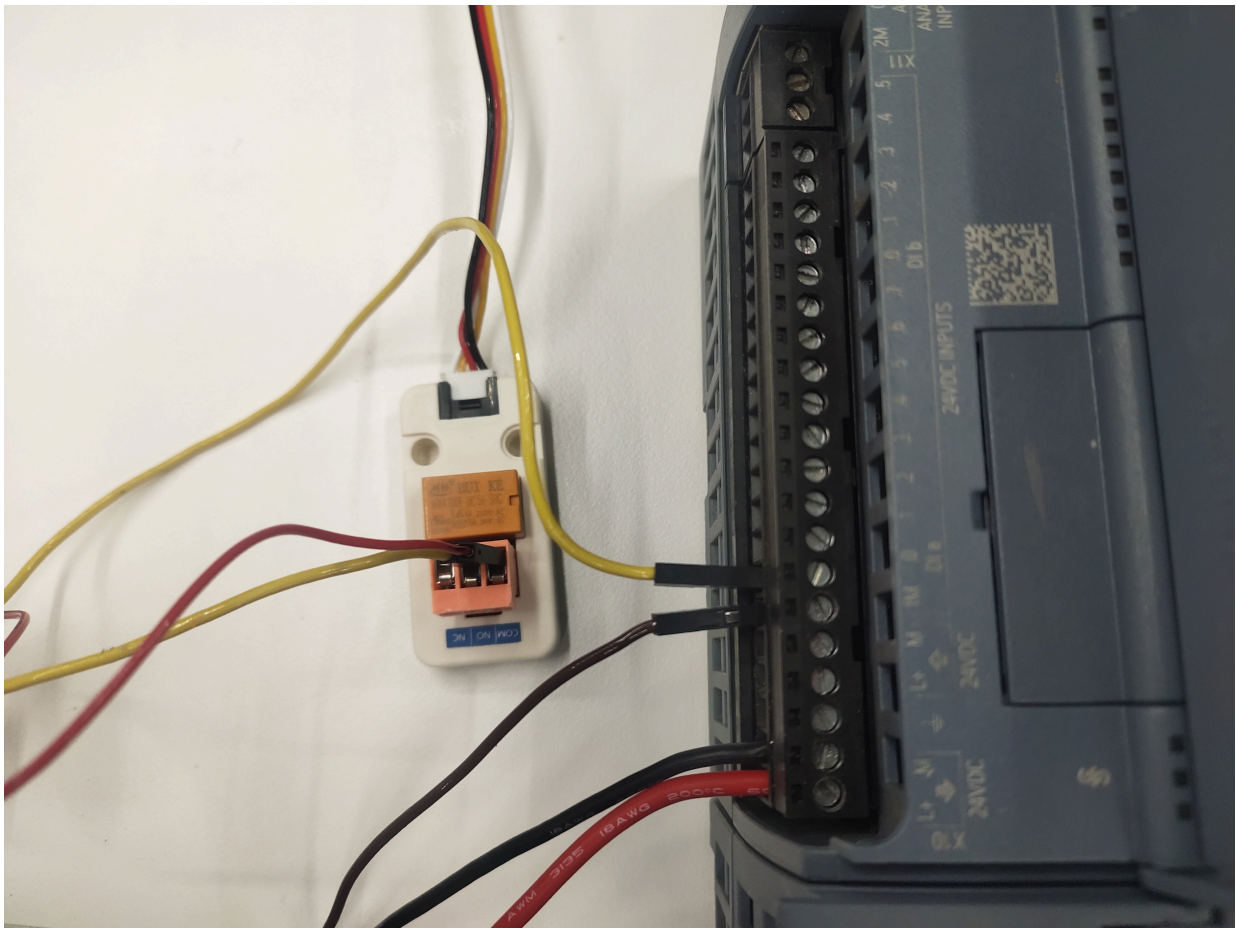


Connect the robot's output to the PLC's input Connect the 5V, GND and GPIO5 of the robot to the coil of the 5V relay

4.1 First-time self-check



Then connect the positive pole of 24V to the COM terminal of the 5V relay, the negative pole of 24V to the 1M terminal of the PLC, and NO to the input of the PLC



4 Software Programming

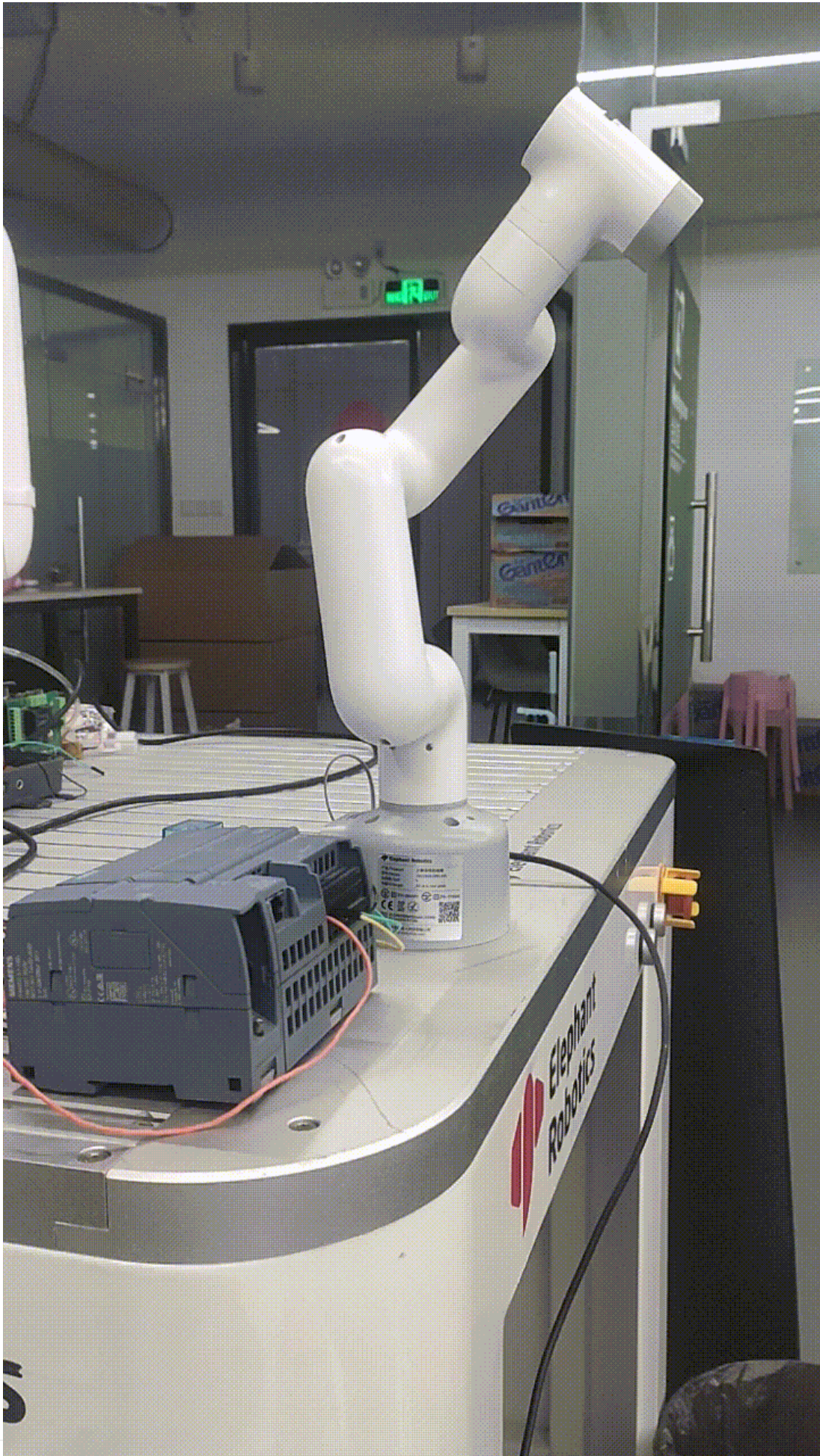
Robot Program

```
import time
mc=MyCobot("COM6")
mc.set_basic_output(5,1)
while 1:
if mc.get_basic_input(2)==1:
mc.sync_send_angles([0,0,0,0,0,0],50)
break
else:
pass
mc.set_basic_output(5,0)
```

PLC program

| 名称 | 数据类型 | 默认值 | 注释 |
|----------------|------|-----|---------------------------------------|
| 1 Input | | | |
| 2 Initial_Call | Bool | | Initial call of this OB |
| 3 Remanence | Bool | | =True, if remanent data are available |
| 4 Temp | | | |
| 5 | | | -注释- |
| 6 Constant | | | |
| 7 | | | -注释- |

5. Effect display



Robot gripper carrying wooden block example

1 Functional description

The robot will use the gripper to carry the wooden block from point A to point B

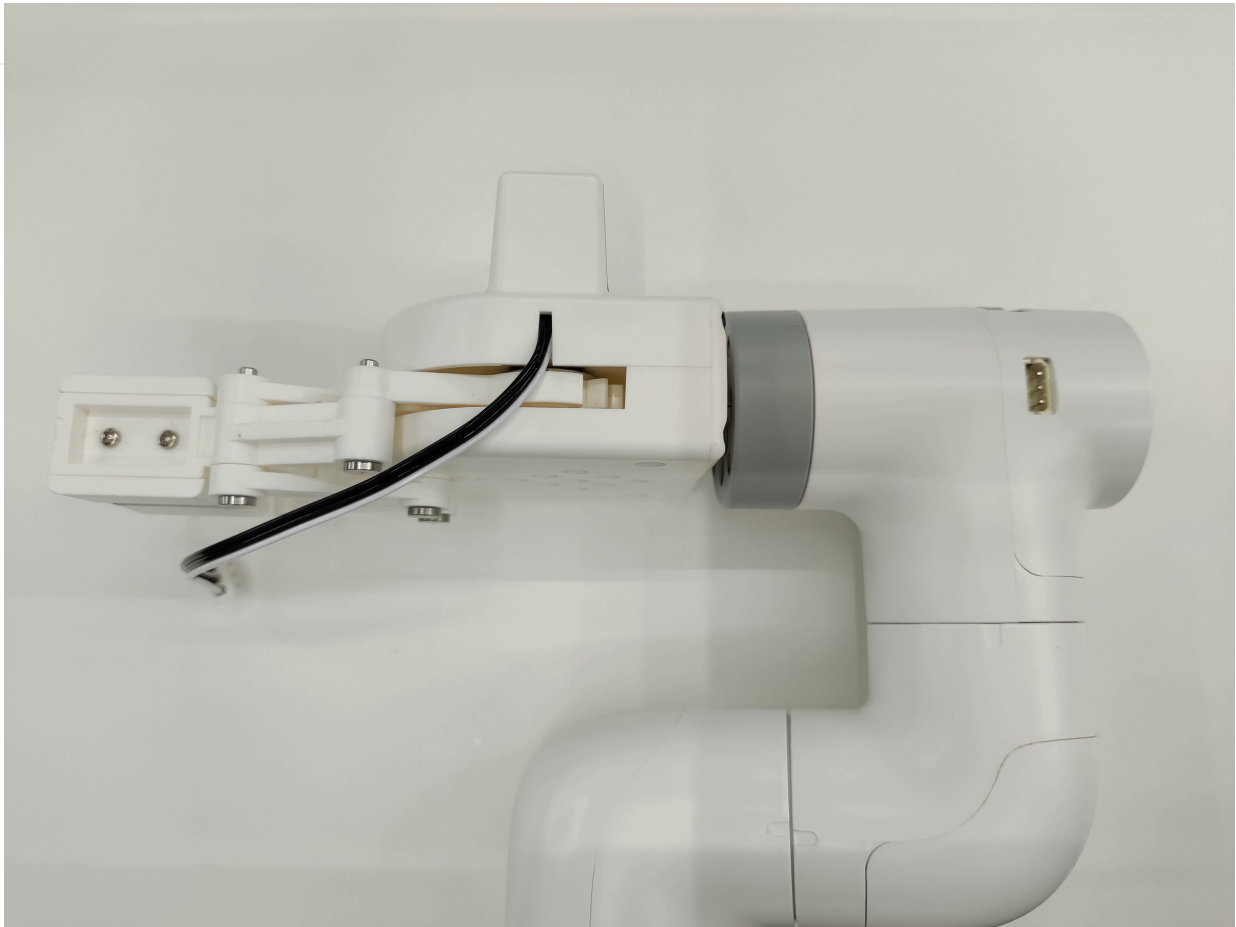
2 Hardware installation

Insert the Lego connector into the reserved socket of the gripper



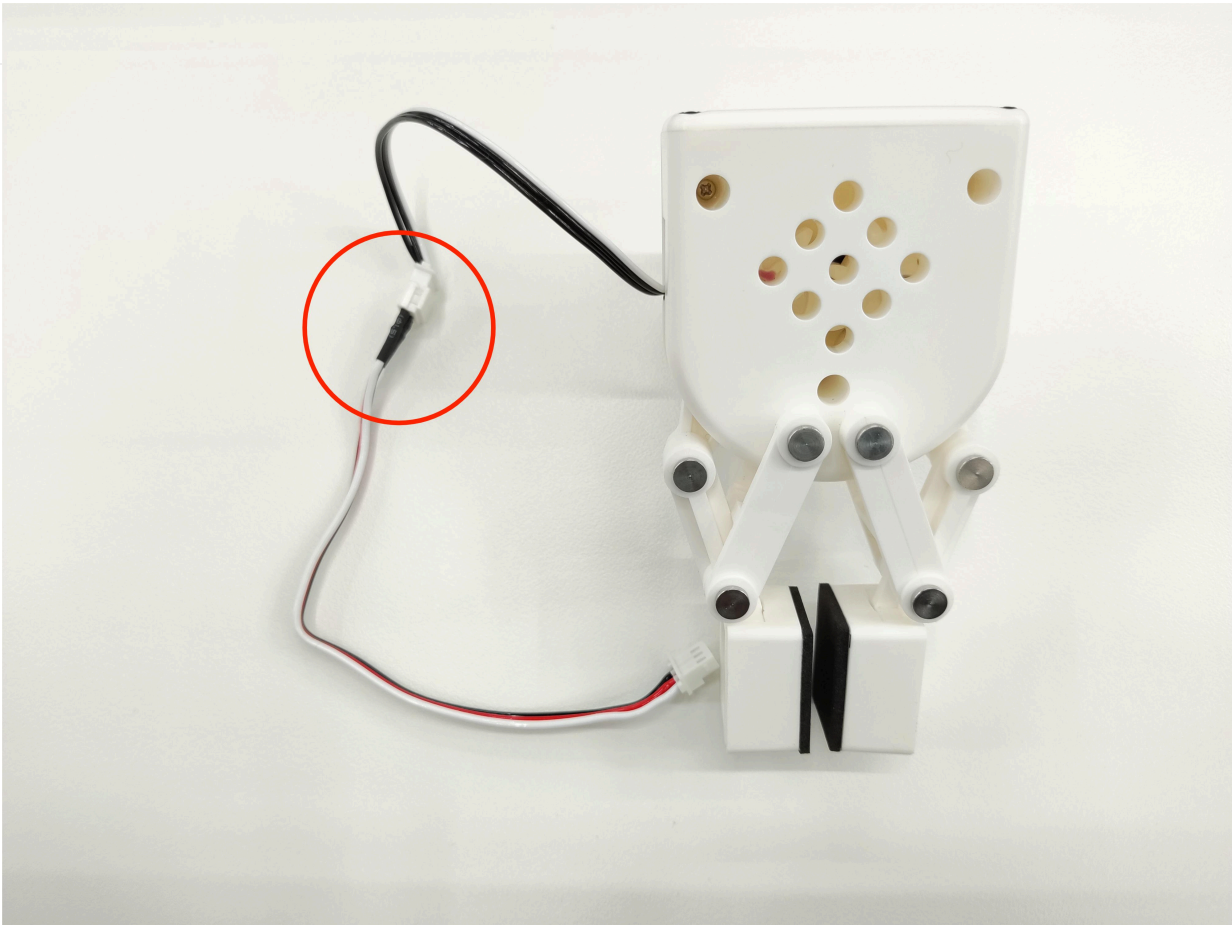
Align the gripper with the connector inserted into the socket at the end of the robot arm

4.1 First-time self-check

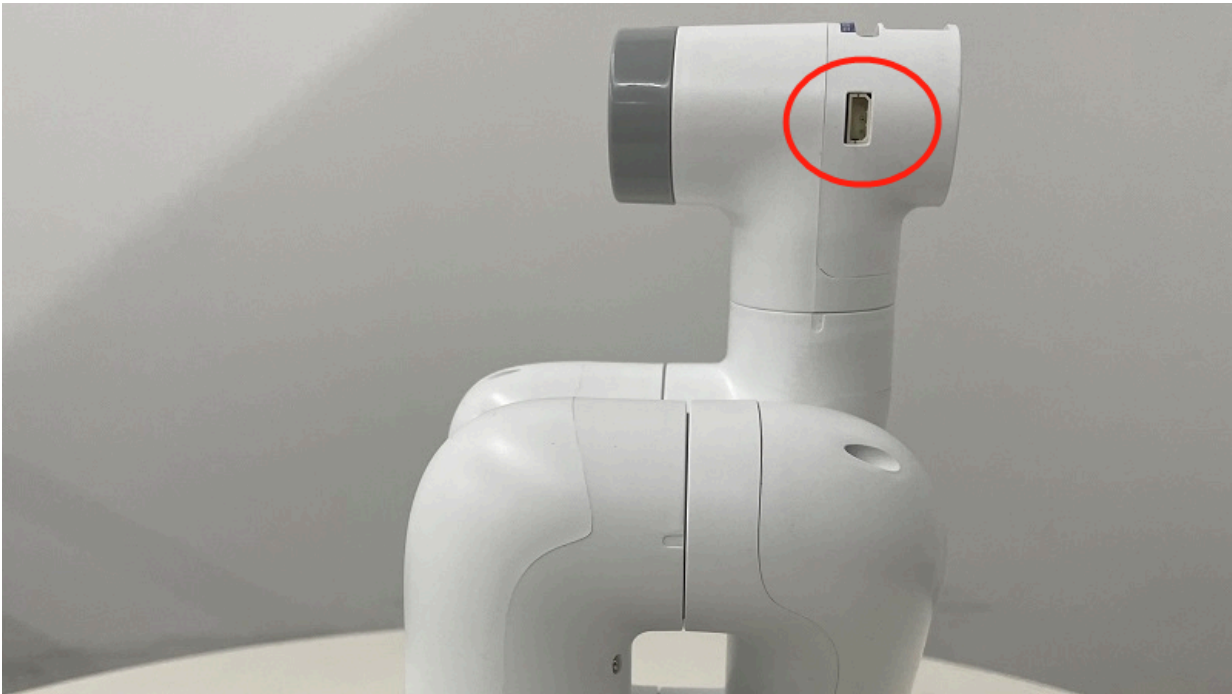


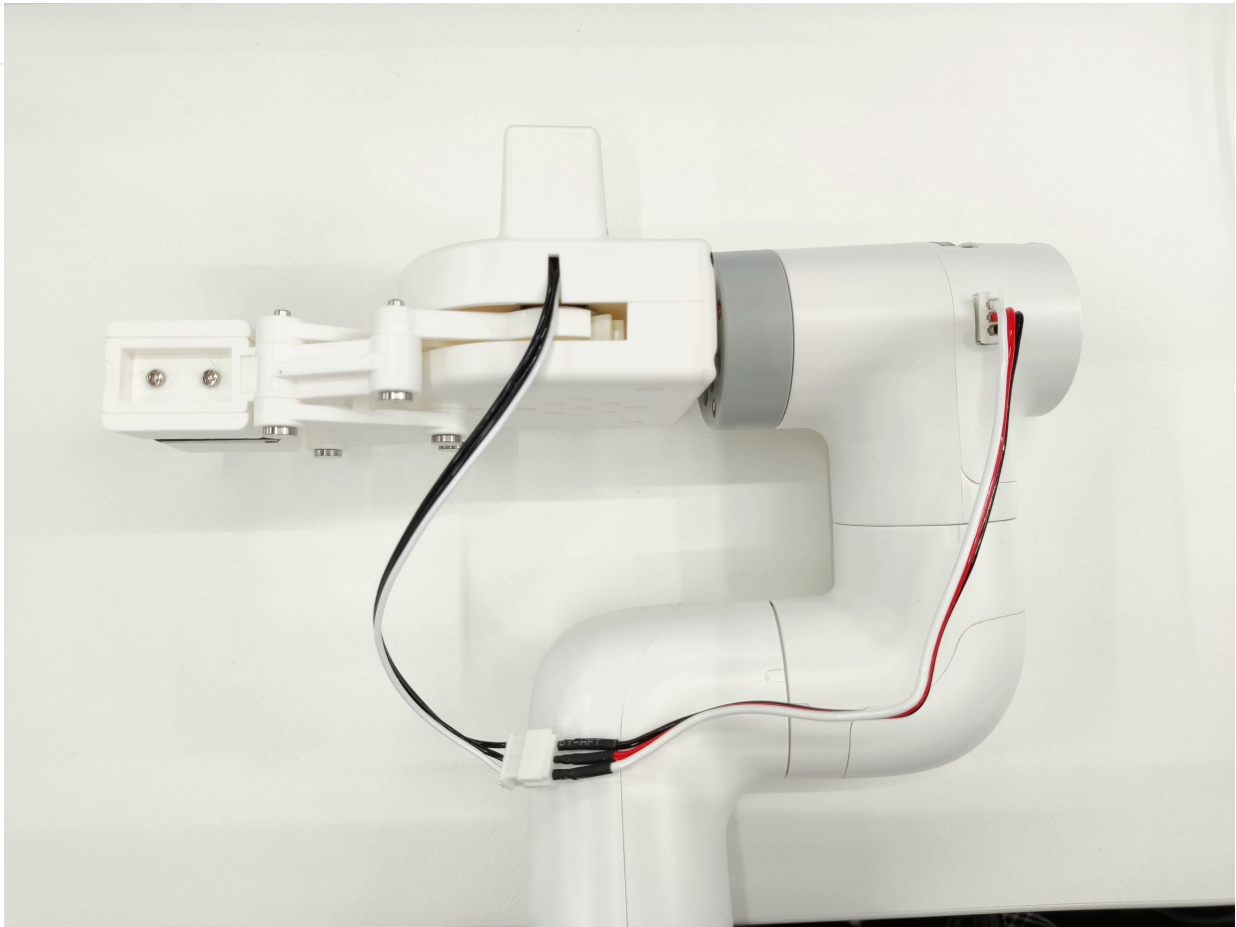
Connect the extension cable to the gripper

4.1 First-time self-check



Insert the robot arm control interface





3 Gripper test

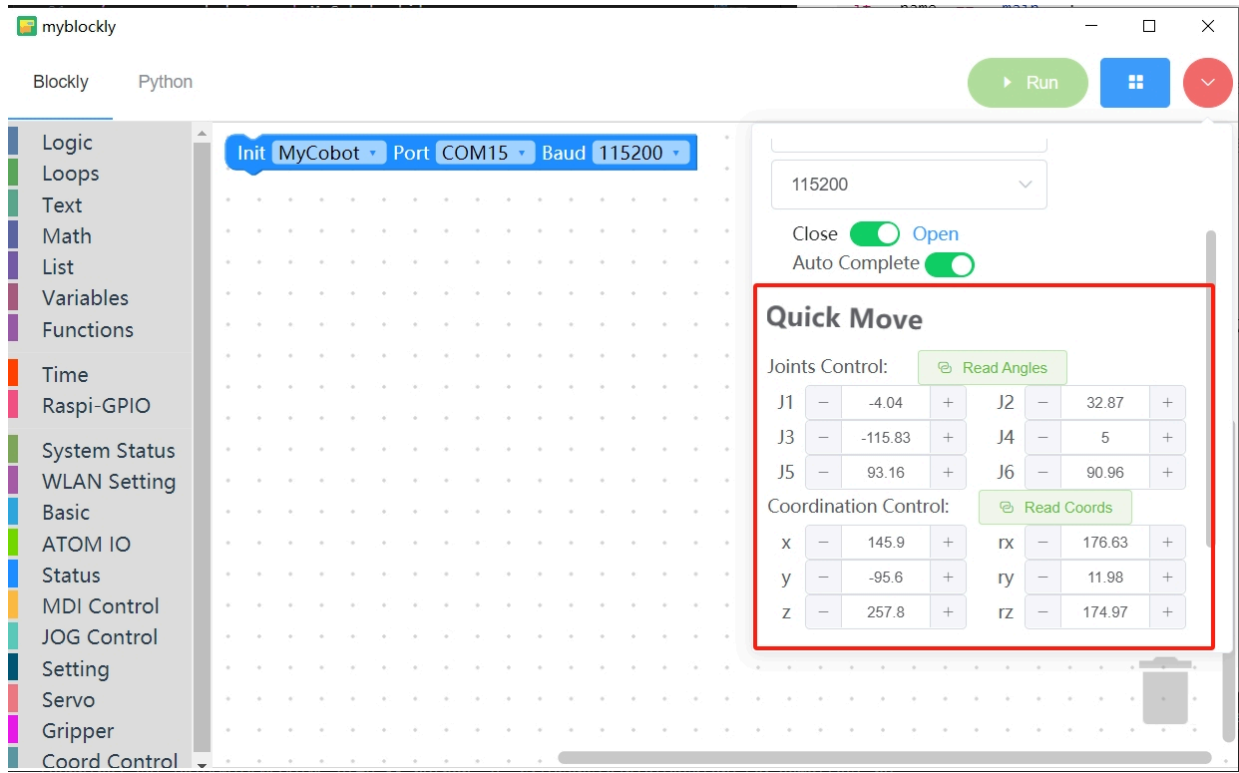
Run the following program, the gripper will repeat the closing and opening action twice

```
from pycobot import MyCobot280,utils
import time

arm=MyCobot280(utils.get_port_list()[0], 115200) # The baud rate is 115200 by default, and some boards are 1000000, please
time.sleep(2)
for i in range(2):
    arm.set_gripper_state(1,100)
    time.sleep(1)
    arm.set_gripper_state(1,100)
    time.sleep(1)
```

4 Software Usage

Use the fast moving function of myblockly to teach the grabbing and placing points of the wooden block, and record the position information. After teaching, you need to disconnect the serial port connection, otherwise the serial port will be reported when running the python script. The error is that the serial port is occupied.



5 Composite application

```

from pymycobot import MyCobot280,utils
import time

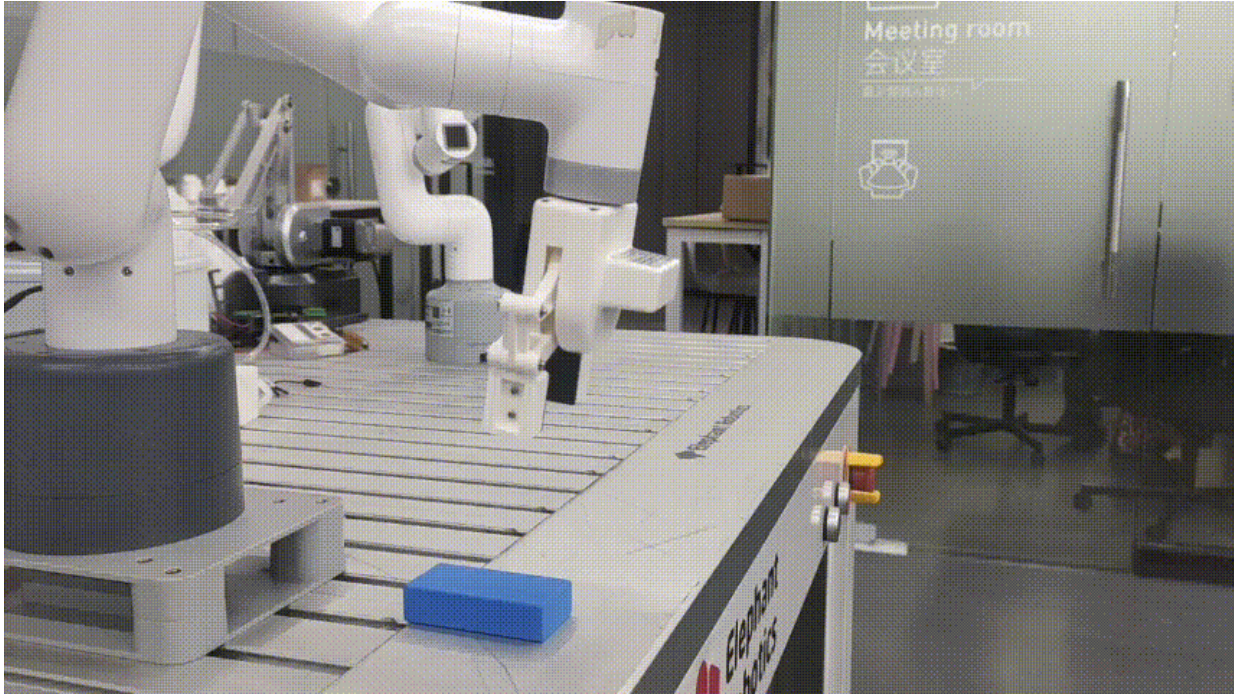
init_angles=[33.22, -15.55, -100.54, 25.48, 6.76, -13.35]#6 joint angles at the initial position
grab_point=[189.9, 12.1, 82.5, -178.15, 6.89, -43.47]#Coordinates of the grab point
place_point=[189.9, 120.1, 88.5, -178.15, 6.89, -43.47]#Coordinates of the placement point

arm=MyCobot280(utils.get_port_list()[0], 115200) # The baud rate is 115200 by default, and some boards are 1000000, please
time.sleep(2)
if __name__=="__main__":
    arm.set_gripper_state(0,100)#Open the gripper first
    time.sleep(1)
    arm.send_angles(init_angles,100)#Initial position of movement
    time.sleep(2)
    arm.send_coords([grab_point[0],grab_point[1],grab_point[2]+70,grab_point[3],grab_point[4],grab_point[5]],100,1)#Move t
    time.sleep(2)
    arm.send_coords([grab_point[0],grab_point[1],grab_point[2],grab_point[3],grab_point[4],grab_point[5]],100,1)#Move to t
    time.sleep(2)
    arm.set_gripper_state(1,100)#Clamp the gripper
    time.sleep(1)
    arm.send_coords([grab_point[0],grab_point[1],grab_point[2]+70,grab_point[3],grab_point[4],grab_point[5]],100,1)#Move t
    time.sleep(2)

    arm.send_coords([place_point[0],place_point[1],place_point[2]+70,place_point[3],place_point[4],place_point[5]],100,1)#
    time.sleep(2)
    arm.send_coords([place_point[0],place_point[1],place_point[2],place_point[3],place_point[4],place_point[5]],100,1)#Mov
    time.sleep(2)
    arm.set_gripper_state(0,100)#Open the gripper
    time.sleep(1)
    arm.send_coords([place_point[0],place_point[1],place_point[2]+70,place_point[3],place_point[4],place_point[5]],100,1)#
    time.sleep(2)

```

6 Effect display



280AR remote control case

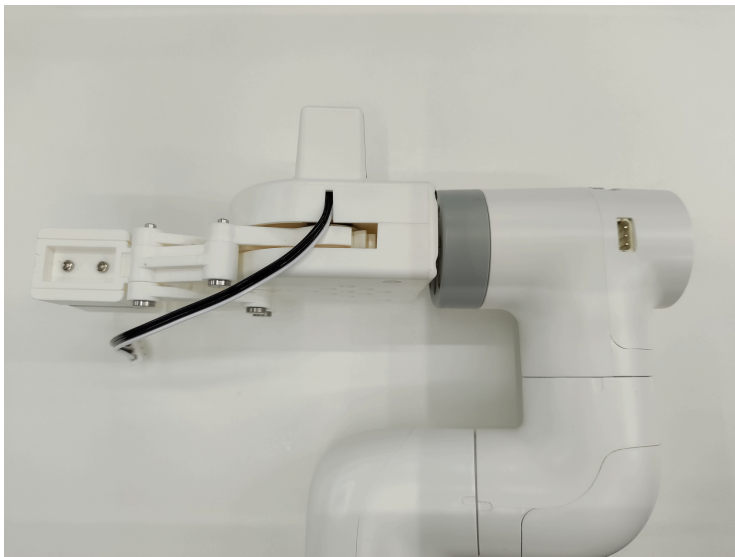
Function description: Use the game controller to control the robot to move coordinates and open and close the gripper

1 Gripper installation

Insert the Lego connector into the reserved socket of the gripper

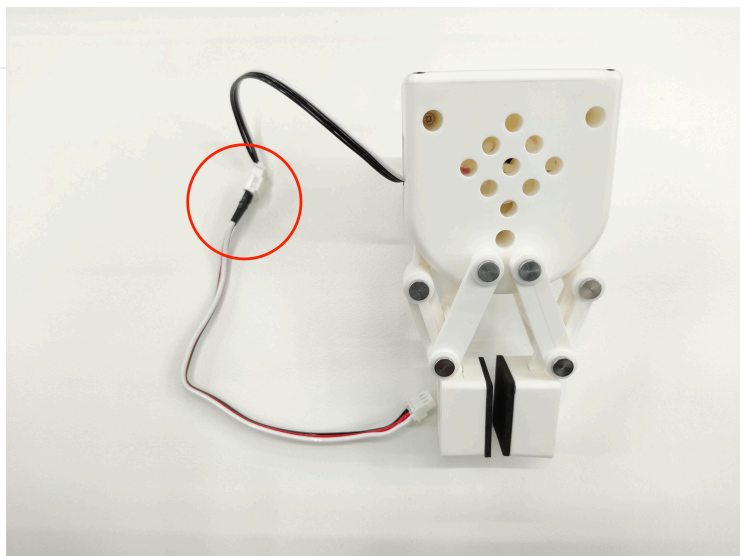


Align the gripper with the connector inserted into the socket at the end of the robot arm and insert it

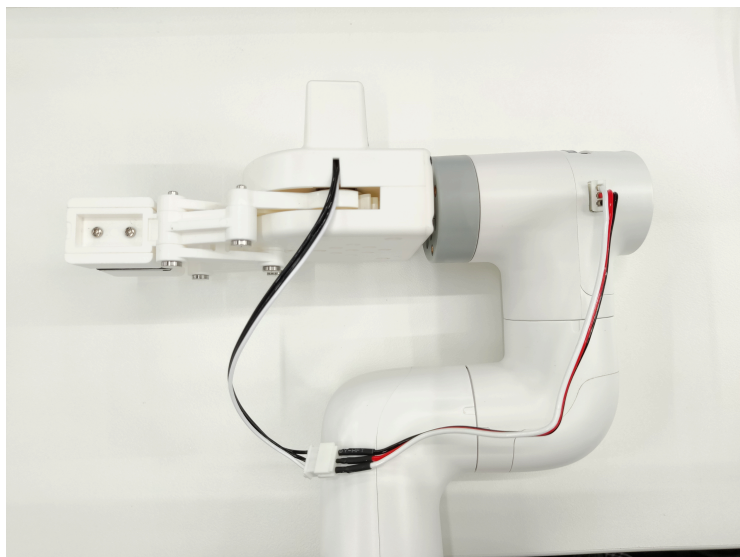
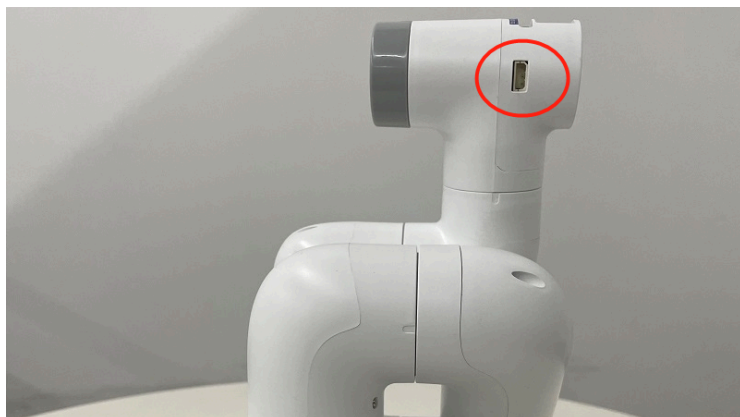


Connect the extension line to the gripper

4.1 First-time self-check



Insert the robot arm control interface



2 Gripper test

```
from pycobot import MyCobot280,utils
import time
arm=MyCobot280(utils.get_port_list()[0], 115200) # The baud rate is 115200 by default, and some boards are 1000000, please
time.sleep(1.5)
for i in range(2):
    arm.set_gripper_state(1,100)#Gripper clamped
    time.sleep(1)
    arm.set_gripper_state(0,100)#Gripper open
    time.sleep(1)
```

3 Handle function description

Note: The handle needs to be purchased separately, please consult the official customer service for details



Plug the receiver of the handle into the computer

| Button | Function |
|--------------------------|----------------------|
| Press the direction key↑ | RY positive movement |
| Press the direction key↓ | RY negative movement |
| Press the direction key← | RX positive movement |
| Press the direction key→ | RX negative movement |
| Push the left joystick↑ | X positive movement |
| Push the left joystick↓ | X negative movement |
| Push the left joystick← | Y positive movement |
| Push the left joystick→ | Y negative movement |
| Push the right joystick↑ | Z positive movement |
| Push the right joystick↓ | Z negative movement |
| Push the right joystick← | RZ positive movement |
| Push the right joystick→ | RZ negative movement |
| Press X key | Gripper opens |
| Press Y key | Gripper closes |
| Press A key | Start suction pump |
| Press B key | Stop suction pump |

Note: Some of the handle buttons are not used, so they will not have any effect on the robot arm; for more specific button usage, please refer to [Handle Control Usage](#)

4 Installation of handle dependency library

Open the terminal and enter the following command to install the handle driver library

```
pip install pygame
```

5 Preparation

Before connecting to the 12V power supply, you can manually adjust the robot arm to the zero position shown in the figure below, and then enter the 12V power supply and communication data line. There should be no debris around the robot arm to avoid collision

4.1 First-time self-check



Turn on the switch of the handle



Note whether the MODE LED of the handle is on

4.1 First-time self-check



Note: The robot arm can only be controlled when the MODE LED is on. If the handle is not used for a long time, it will enter the standby state. You can press the START button on the handle to activate it.



6 Example program

```

# coding:utf-8
import pygame
import sys
import time
from pymycobot import MyCobot280
import platform
import threading

if "linux" in platform.platform().lower():
    import RPi.GPIO as GPIO

    # Set GPIO mode
    GPIO.setmode(GPIO.BCM)

    # Set pin 20 (electromagnetic valve) and pin 21 (release valve) as output
    GPIO.setup(20, GPIO.OUT)
    GPIO.setup(21, GPIO.OUT)

# Initialize MyCobot280 with serial port and baud rate
mc = MyCobot280('com3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord

init_angles = [0, 0, -90, 0, 0, 0]
go_home = [0, 0, 0, 0, 0, 0]

# Initialize Pygame and joystick modules
pygame.init()
pygame.joystick.init()
button_pressed = False
hat_pressed = False
previous_state = [0, 0, 0, 0, 0, 0]

# Function to turn on the vacuum pump (electromagnetic valve)
def pump_on():
    # 打开电磁阀
    if platform.system() == "Linux":
        GPIO.output(20, 0)
    elif platform.system() == "Windows":
        mc.set_digital_output(33, 0)
        time.sleep(0.05)

# Function to turn off the vacuum pump (close electromagnetic valve and open release)
def pump_off():
    if platform.system() == "Linux":
        # Close valve
        GPIO.output(20, 1)
        time.sleep(0.05)

```

4.1 First-time self-check

```
# Open release valve
GPIO.output(21, 0)
time.sleep(1)
GPIO.output(21, 1)
time.sleep(0.05)
elif platform.system() == "Windows":
    # Close valve
    mc.set_digital_output(33, 1)
    time.sleep(0.05)
    # Open release valve
    mc.set_digital_output(23, 0)
    time.sleep(1)
    mc.set_digital_output(23, 1)
    time.sleep(0.05)

# Function to safely stop the robot (used in a thread)
def safe_stop():
    try:
        mc.stop()
        time.sleep(0.02)
    except Exception as e:
        print("stop error", e)
        previous_state[axis] = 0

# Handler for joystick input events
def joy_handler():
    global button_pressed
    global hat_pressed
    global previous_state
    # Joystick axis movement (analog stick)
    if event.type == pygame.JOYAXISMOTION:
        axis = event.axis
        value = round(event.value, 2)
        if abs(value) > 0.1:
            flag = True
            previous_state[axis] = value
            # jog_coord(index, direction, speed)
            # direction: 0 = negative direction, 1 = positive direction
            if axis == 0 and value == -1.00:
                mc.jog_coord(2, 1, 50)
            elif axis == 0 and value == 1.00:
                mc.jog_coord(2, 0, 50)
            if axis == 1 and value == 1.00:
                mc.jog_coord(1, 0, 50)
            elif axis == 1 and value == -1.00:
                mc.jog_coord(1, 1, 50)
            if axis == 3 and value == 1.00:
                mc.jog_coord(6, 1, 50)
```

4.1 First-time self-check

```
elif axis == 3 and value == -1.00:
    mc.jog_coord(6, 0, 50)
if axis == 4 and value == 1.00:
    mc.jog_coord(3, 0, 50)
elif axis == 4 and value == -1.00:
    mc.jog_coord(3, 1, 50)
# Axis 2 for servo release
elif axis == 2 and value == 1.00:
    mc.release_all_servos()
    time.sleep(0.03)
# Axis 5 to power on
elif axis == 5 and value == 1.00:
    mc.power_on()
    time.sleep(0.03)
else:
    if previous_state[axis] != 0:
        # mc.stop()
        threading.Thread(target=safe_stop).start()
        previous_state[axis] = 0
# Joystick button pressed
elif event.type == pygame.JOYBUTTONDOWN:
    if joystick.get_button(2) == 1:
        mc.set_gripper_state(0, 100)
    elif joystick.get_button(3) == 1:
        mc.set_gripper_state(1, 100)
    elif joystick.get_button(0) == 1:
        pump_on()
    elif joystick.get_button(1) == 1:
        pump_off()
    elif joystick.get_button(5) == 1:
        mc.send_angles(init_angles, 50)
        time.sleep(2)
    elif joystick.get_button(4) == 1:
        mc.send_angles(go_home, 50)
        time.sleep(3)
# D-Pad (HAT) movement
elif event.type == pygame.JOYHATMOTION:
    hat_value = joystick.get_hat(0)
    if hat_value == (0, -1):
        mc.jog_coord(4, 0, 50)
    elif hat_value == (0, 1):
        mc.jog_coord(4, 1, 50)
    elif hat_value == (-1, 0):
        mc.jog_coord(5, 0, 50)
    elif hat_value == (1, 0):
        mc.jog_coord(5, 1, 50)
    if hat_value != (0, 0):
        hat_pressed = True
else:
    if hat_pressed:
```

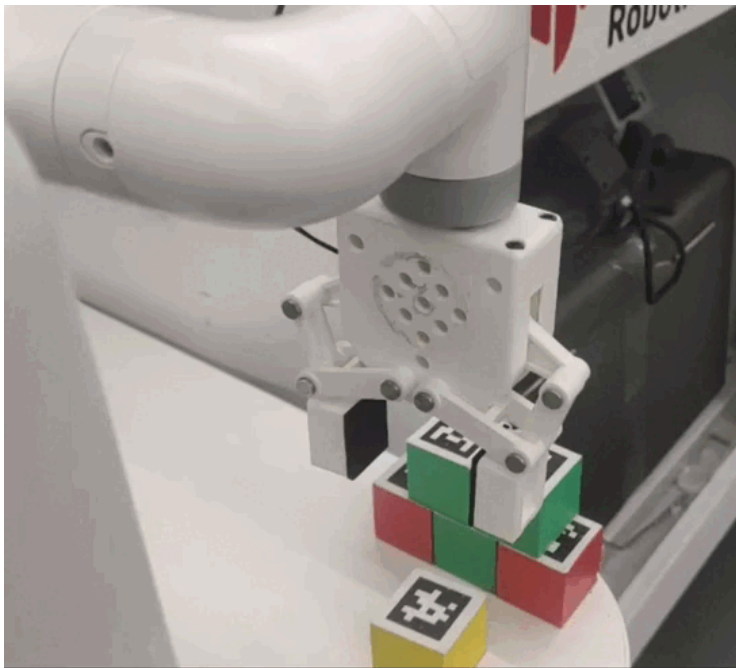
4.1 First-time self-check

```
# mc.stop()
threading.Thread(target=safe_stop).start()
hat_pressed = False

# Initialize joystick if detected
if pygame.joystick.get_count() > 0:
    joystick = pygame.joystick.Joystick(0)
    joystick.init()
else:
    print("Not handle detected")
    pygame.quit()
    sys.exit()

# Main loop to process events
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
            joy_handler()
    pygame.quit()
```

7 Effect display



Robot suction pump to carry wooden blocks

1 Functional description

The robot will use the suction pump to carry wooden blocks from point A to point B

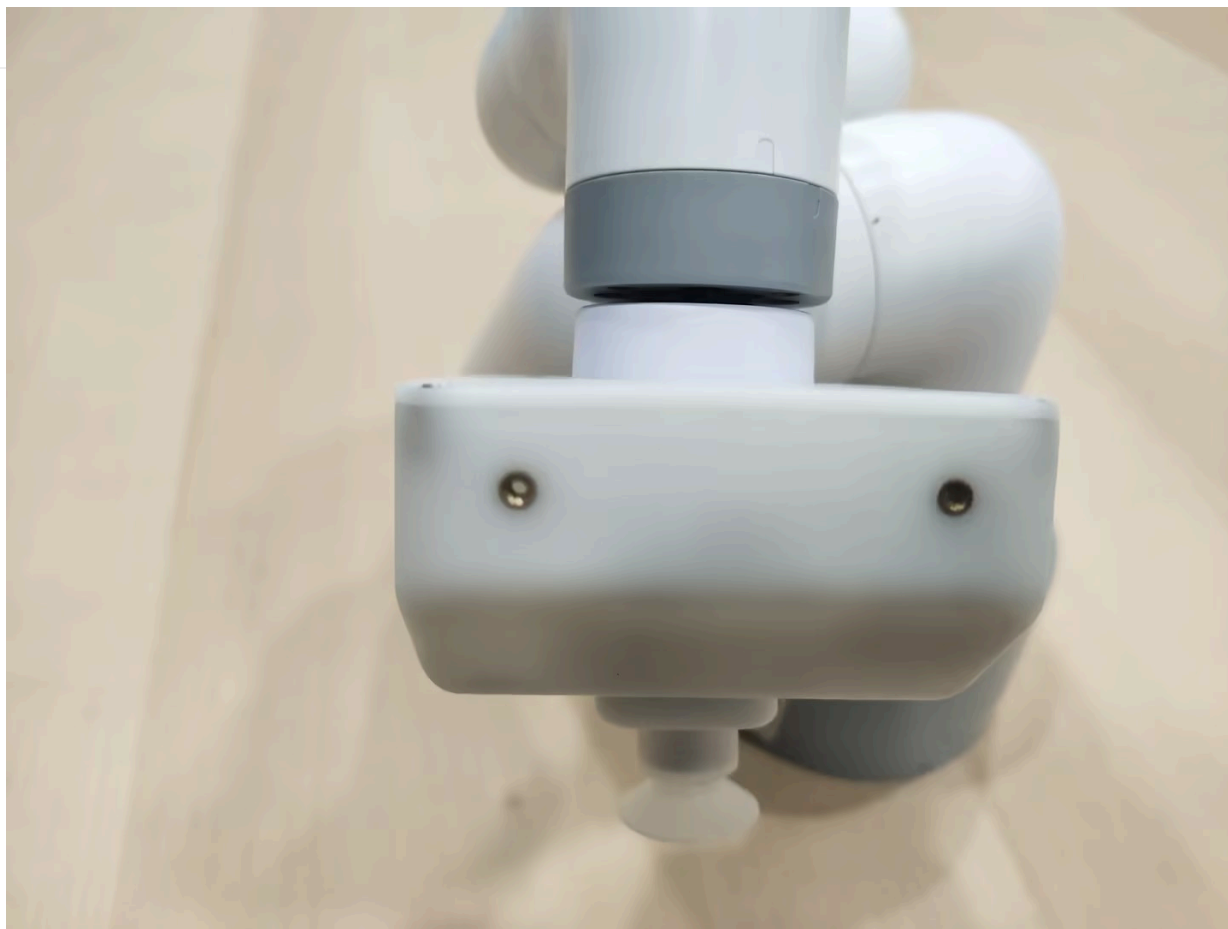
2 Hardware installation

Insert the Lego connector into the reserved socket on the suction pump

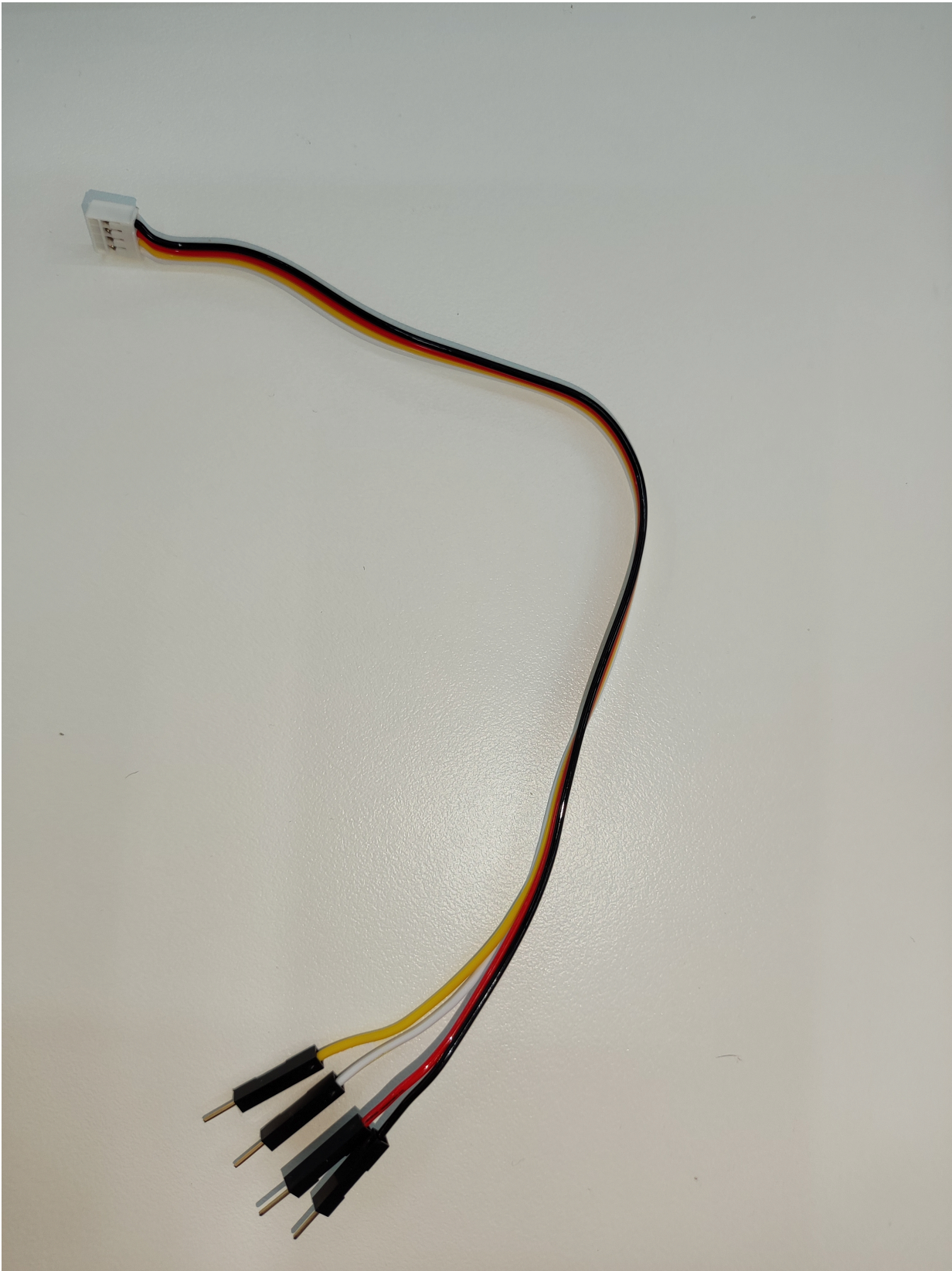


Align the suction pump with the connector inserted into the socket at the end of the robotic arm

4.1 First-time self-check



Select the male-female DuPont wire, and insert the female end into the socket marked with pins on the suction pump box



4.1 First-time self-check



Then connect the wire to the end IO of the robotic arm



The left side is the suction pump pin, and the right side is the robotic arm pin GND -> GND 5V -> 5V G2 -> 23
G5 -> 33

3 Pump test

Run the following program, the pump will repeat the opening and closing action twice

4.1 First-time self-check

```
from pycobot import MyCobot280,utils
import time

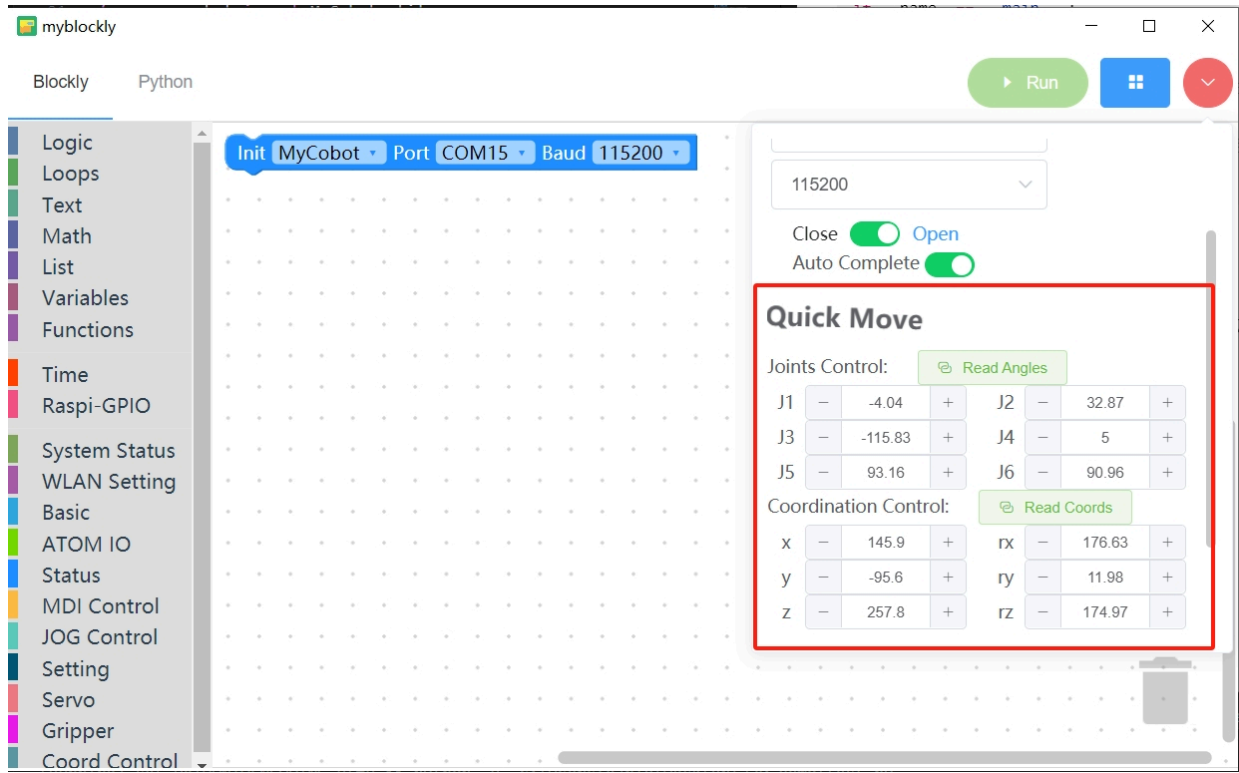
arm = MyCobot280(utils.get_port_list()[0], 115200) # The baud rate is 115200 by default, and some boards are 1000000, please
time.sleep(2)
# Turn on the pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

# Stop the pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)

for i in range(2):
    pump_on()
    time.sleep(2)
    pump_off()
    time.sleep(2)
```

4 Software Usage

Use the fast movement function of myblockly to teach the grab and place points of the wooden block, and record the position information. After teaching, you need to disconnect the serial port, otherwise the serial port will be reported when running the python script.



5 Composite Application

```

from pymycobot import MyCobot280,utils
import time
init_angles=[33.22, -15.55, -100.54, 25.48, 6.76, -13.35]#6 joint angles at the initial position
grab_point=[189.9, 12.1, 54.5, -178.15, 6.89, -43.47]#Coordinates of the grab point
place_point=[189.9, 120.1, 62.5, -178.15, 6.89, -43.47]# Coordinates of the placement point

arm = MyCobot280(utils.get_port_list()[0], 115200) # The baud rate is 115200 by default, and some boards are 1000000, please
time.sleep(2)

# Turn on the suction pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

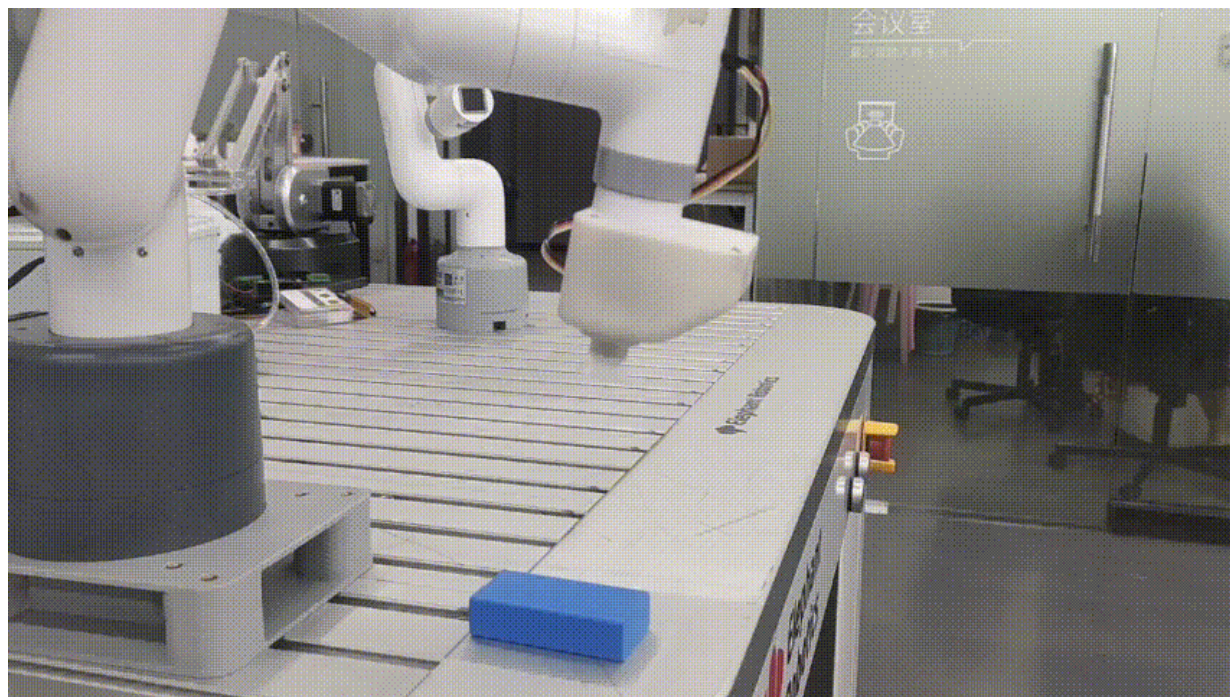
# Stop the suction pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)

if __name__=="__main__":
    pump_off()#Turn off the suction pump first
    time.sleep(1)
    arm.send_angles(init_angles,100)#Move to the initial position
    time.sleep(2)
    arm.send_coords([grab_point[0],grab_point[1],grab_point[2]+70,grab_point[3],grab_point[4],grab_point[5]],100,1)#Move to
    time.sleep(2)
    arm.send_coords([grab_point[0],grab_point[1],grab_point[2],grab_point[3],grab_point[4],grab_point[5]],100,1)#Move to t
    time.sleep(2)
    pump_on() #Turn on the suction pump
    time.sleep(1)
    arm.send_coords([grab_point[0],grab_point[1],grab_point[2]+70,grab_point[3],grab_point[4],grab_point[5]],100,1)#Move t
    time.sleep(2)

    arm.send_coords([place_point[0],place_point[1],place_point[2]+70,place_point[3],place_point[4],place_point[5]],100,1)#
    time.sleep(2)
    arm.send_coords([place_point[0],place_point[1],place_point[2],place_point[3],place_point[4],place_point[5]],100,1)#Mov
    time.sleep(2)
    pump_off() #Turn off the pump
    time.sleep(1)
    arm.send_coords([place_point[0],place_point[1],place_point[2]+70,place_point[3],place_point[4],place_point[5]],100,1)#
    time.sleep(2)

```

6 Effect display



Chapter 8 Supporting Resources

This chapter will introduce various supporting resources of the product in detail, aiming to help users fully understand and use our products efficiently. Whether it is product information, drawings, software information and source code, or system information and promotional materials, we provide detailed information and download links to ensure that users can make full use of these resources for product development, operation and promotion

Download Product Information

The product information includes detailed specifications, technical parameters and instructions for use of the 280 For Arduino robotic arm. This section aims to help users fully understand the performance and functions of the robotic arm and ensure the best experience during use

Product Drawings

The product drawings section provides detailed 3D and 2D drawings of the 280 For Arduino robotic arm. These drawings are particularly important for engineers who need to perform customized designs or maintenance, and can help them better understand the structure of the robotic arm.

Software Information and Source Code

The software information and source code section contains the software installation package, driver and related open source code that are compatible with the 280 For Arduino robotic arm. Users can use these materials to install, upgrade and perform secondary development of the software to enhance the functions and application scenarios of the robotic arm

System Information

The system information provides the system architecture and working principle of the 280 For Arduino robotic arm, covering the collaborative working mode of hardware and software. This section helps users to quickly locate and solve problems during integration and debugging to ensure stable operation of the system

(To be updated)

Promotional Materials

The promotional materials section contains the product brochure, demonstration video and customer cases of the 280 For Arduino robotic arm. These materials not only show the core advantages and application scenarios of the robotic arm, but also provide successful cases in actual applications to help potential customers understand the value of the product more intuitively.

-
- [Previous Chapter](#)- /[Next Chapter](#)
-

Product Information Download

The product information includes detailed specifications, technical parameters and instructions for use of the 280 For Arduino robotic arm. This section is designed to help users fully understand the performance and functions of the robotic arm and ensure the best experience during use

Download Link

You can download all relevant product information through the following link: [Product Information Download](#)

| Operating System | Download Link |
|------------------|-------------------------------|
| Ubuntu 18.04 | Download Link |
| ubuntu 20.04 | Download link |

Product Drawings

The product drawings section provides detailed 3D and 2D drawings of the 280 For Arduino robotic arm. These drawings are particularly important for engineers who need to make customized designs or perform maintenance, and can help them better understand the structure of the robotic arm.

Machine 3D Model

| Machine | 3D Model File |
|---------------------|--------------------------|
| myCobot 280 Arduino | Download |

Machine 2D Drawings

| Machine | Machine 2D Drawings |
|------------------|--------------------------|
| myCobot 280 Atom | Download |

Accessory 3D Model

- myCobot Series

| Accessory | 3D Model File |
|-----------------------------|--------------------------|
| Adaptive Gripper | Download |
| Parallel Gripper | Download |
| Flexible Gripper | Download |
| G-type base | Download |
| Large suction cup base | Download |
| Flat base | Download |
| Vertical Suction Pump | Download |
| Double-head suction pump | Download |
| Camera flange | Download |
| Spring bamboo shoots flange | Download |
| Dexterous Hand | Download |
| Pen holder | Download |
| Mobile Phone Gripper | Download |

Software information and source code

The software information and source code section includes the software installation package that is compatible with the 280 For Arduino robotic arm. Users can use these materials to install, upgrade and perform secondary development of the software to enhance the functions and application scenarios of the robotic arm

You can download all relevant product information through the following link: [Software Information Download](#)

The software download link includes the following software that may be used

- myStudio 2.0 The one-stop service platform myStudio integrates myCobot software resources and various materials. Main functions:
 - Support firmware download and update;
 - Provide product use video tutorials;
 - Maintenance/repair information;
-
- myBlockly myBlockly is a fully visual modular programming software, a graphical programming language, suitable for novice users to familiarize themselves with programming. Users can create simple and complex functions by dragging and dropping puzzles to develop applications.
-
- Meta Care Meta Care is an interesting pet simulation game that combines elements of pet raising, story and achievement collection.
 - The game requires an Internet connection throughout the game.
 - You need to own Meta Dog and connect it to your device via Bluetooth.
 - The game is only supported on Android devices.
-
- Elephant Luban Elephant Luban is a G-Code trajectory generation and usage platform that provides user-based use cases, selects writing, painting, and laser engraving usage scenarios, and quickly opens up a DIY creative space.
-
- MyCobot Controllerle MyCobot Controller is an APP that controls the MyCobot series of robotic arms via Bluetooth. You can use your phone to move the robotic arm. If you are an Android user, please go to [Google Play Store] If you are an IOS user, please wait for the software to be released before searching and downloading.
-

System Information

The system information provides the system architecture and working principle of the 280 For Arduino robotic arm, covering the collaborative working methods of hardware and software. This section helps users to quickly locate and solve problems during integration and debugging to ensure stable operation of the system (Information to be updated)

Promotional Materials

The promotional materials section includes the product brochure, demonstration video and customer cases of the 280 For Arduino robotic arm. These materials not only demonstrate the core advantages and application scenarios of the robotic arm, but also provide successful cases in actual applications to help potential customers understand the value of the product more intuitively.

| Machine | Product Brochure |
|----------------|--------------------------|
| myCobot 280 AR | Download |

Product unboxing video [Product unboxing video](#)

Product promotion video [Product promotion video](#)

User Case [User Case](#)

Elephant Robotics



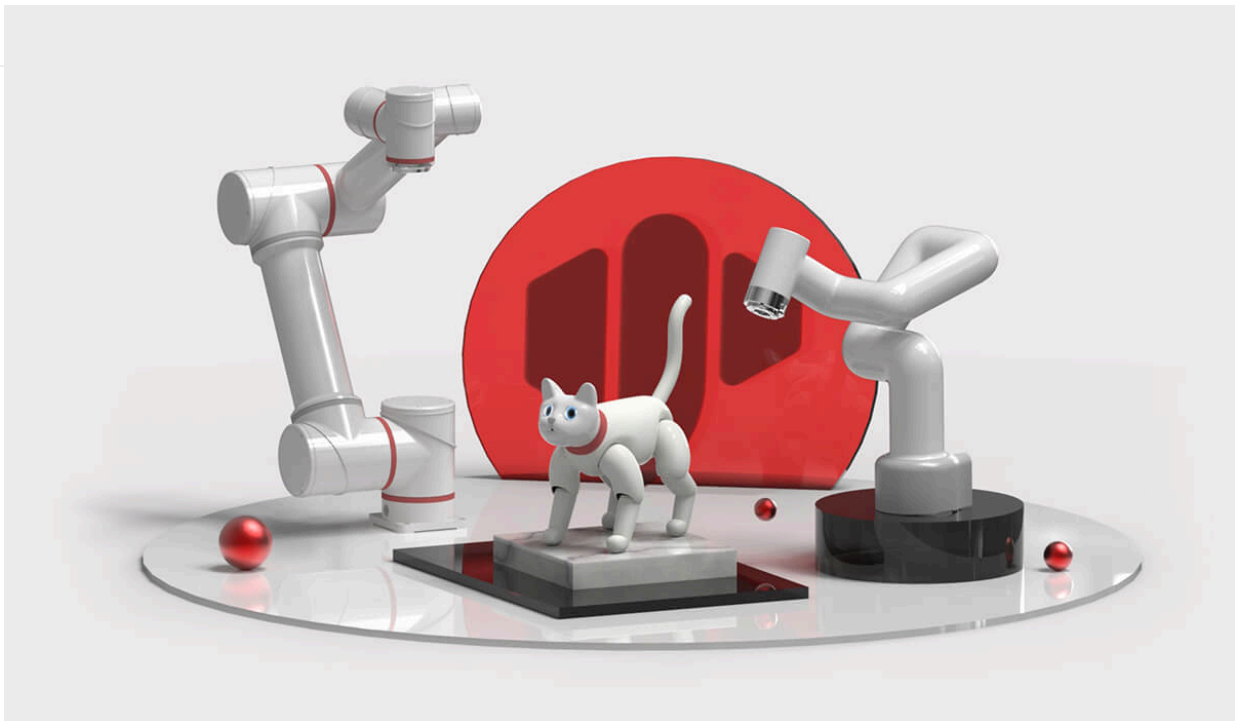
1. Company Introduction

Based in Shenzhen, China, Elephant Robotics is a high-tech enterprise focusing on robot R&D, design 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 scientific research institutions, commercial scenarios and industrial production. Its product quality and smart solutions have been unanimously recognized and praised by several factories from the world's top 500 companies in South Korea, Japan, the United States, Germany, Italy, Greece and other countries.

Elephant Robotics adheres to the vision of "Enjoy Robots World" and advocates the collaborative work of people and robots, making robots a good helper for human work and life, helping people to be liberated from simple, repetitive and boring work, giving full play to the advantages of human-machine collaboration, thereby improving work efficiency and helping humans create a better new life.

In the future, Elephant Robotics hopes to promote the development of the robot industry through a new generation of cutting-edge technology, and work with customers and partners to open a new era of automation and intelligence.



2. Development History

- 2016.08 -----Elephant Robotics Co., Ltd. was officially established
- 2016.08 -----Entered HAX incubator and received seed round investment from SOSV
- 2016.08 -----Started research and development of Elephant S industrial collaborative robot
- 2017.01 -----Rated as "Top 10 Most Innovative Companies in China at CES"
- 2017.04 -----Attended Hannover Industrial Fair and Korea Automation Exhibition
- 2017.07 -----Two founders were selected as "30 Business Elites Under 30" by Forbes Asia
- 2017.10 -----The fifth-generation single-arm industrial collaborative robot Elephant S was launched
- 2018.04 -----Received angel round investment from "Yun Angel Fund"
- 2018.06 -----First public appearance 2018 Hannover World Industrial Fair
- 2018.06 -----Won the "Smart Manufacturing Entrepreneurship MBA Award" from Cheung Kong Graduate School of Business
- 2018.06 -----Won the "Entrepreneurship Accelerator X-elerator Award" from Tsinghua School of Economics and Management
- 2018.11 -----Won the second place in the Shenzhen Division of the Asian Smart Hardware Competition
- 2018.11 -----Won the "Most Investment Enterprise Award" of the Gaogong Golden Globe Award
- 2019.03 -----Won the "Leadership Award" of the Gaogong Golden Globe Award
- 2019.04 -----In March 2019, Catbot won the "Industrial Robot Innovation Award"
- 2019.09 -----Attended the Huawei European Ecosystem Conference (HCE) and officially became a member of Huawei's ecological partner
- 2019.11 -----Elephant Robotics and Harbin Institute of Technology attended the IROS International Intelligent Robots and Systems Conference
- 2019.12 -----Elephant Robotics-South China University of Technology "Intelligent Robot Joint Development Laboratory" was officially unveiled
- 2019.12 -----Won the "Innovation Technology Award" of Gaogong in 2019

4.1 First-time self-check

- 2019.12 -----Won the "Top Ten Fast-Growing Enterprises" of Gaogong in 2019
- 2019.12 -----Won the "New Enterprise Award" in the Industrial Robot Segment of Shenzhen Equipment Industry
- 2019.12 -----The world's first bionic robot cat MarsCat was launched
- 2020.05 -----The founder won the 2019 Shenzhen Robotics Newcomer Award
- 2020.10 -----The world's lightest and smallest six-axis collaborative robot myCobot was launched
- 2021.03 -----The smallest collaborative robot for scientific research myCobotPro 320 was launched
- 2021.05 -----Mars bionic cat MarsCat Received coverage from Xinhua Finance, China Daily, Nanjing Daily, Harbin Daily and other media
- 2021.07 -----Released the smallest composite robot chassis - Elephant Mobile Robot myAGV
- 2021.09 -----The world's first fully enclosed four-axis robotic arm - Elephant palletizing robotic arm myPalletizer was launched

3. Related links

Purchase link

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

4. Contact Us

Our working hours are China working days, 10 am to 6 pm Beijing time.

- If you have any other questions, please contact us via the following methods. [E-mail](#) :

support@elephantrobotics.com

- If you have any purchase intention or any parameter questions, please send an email to this mailbox. [E-mail] (sales@elephantrobotics.com) :

sales@elephantrobotics.com

- If you encounter any problems in the use of this product, please read Chapter 9 of the manual first. If the problems listed cannot help you solve them, and you have more after-sales problems, please send an email to

4.1 First-time self-check

this mailbox. [E-mail](#) :

support@elephantrobotics.com

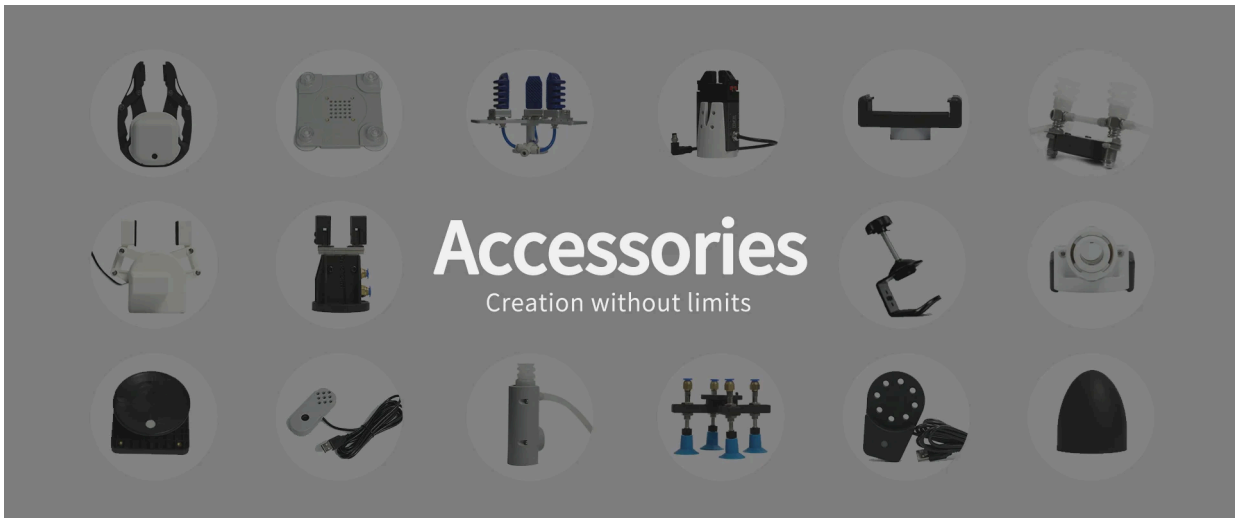
We will respond within 1-2 working days;

WeChat: We only provide one-to-one service for users who purchase mycobot products through WeChat.



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

Product Accessories



In the real world, different accessories can enhance the capabilities of robots in various ways. For example, accessories such as grippers, sensors, and tools can help robots perform various tasks, thereby increasing their versatility and flexibility.

Elephant Robotics is committed to making robots and these accessories easy for everyone to use, freeing users from the complexity of choosing the right accessories and enabling them to quickly start using robots.

Accessory Types

In order to meet the needs of customers in different scenarios, we have designed various types of accessories, including grippers, suction cups, camera modules, and other gripping devices, so that users can directly choose the right end effector

Base

- [Flat Base](#)

Suitable for fixing the robot arm on a flat and smooth surface.

- [G-Base](#)

Suitable for fixing the robot arm on the edge of a table.

Gripper

- [Adaptive Gripper](#)

The gripper is a robot component that can achieve functions similar to human hands. It has the advantages of complex structure, firm gripping of objects, not easy to fall, and easy operation. The gripper kit includes gripper accessories and Lego technology parts. The end effector of the robot arm is controlled by a programmable system to achieve functions such as object grasping and multi-point positioning.

- [Parallel Gripper](#)

Driven by a motor, the finger surface of the gripper makes a linear reciprocating motion to achieve the opening or closing action. The acceleration and deceleration of the electric gripper can be controlled, the impact on the workpiece can be minimized, the positioning point can be controlled, and the clamping can be controlled

- [Flexible Gripper-Open Foot Type](#)

The fingertips are made of rubber and rely on air pressure deformation to grasp objects. Pneumatic manipulators are widely used and are favored for their softness, adaptability and efficiency. These advantages make them powerful tools in automation and robotic applications, capable of effectively handling a wide range of objects and tasks

Suction Pumps

- [Vertical Suction Pumps](#)

With one suction nozzle and one exhaust nozzle, the suction pump kit is controlled as the end effector of the robot arm to perform the function of sucking objects.

- [Double-head Suction Pumps](#)

Compared with single-head suction pumps, it is more stable, with simple structure, small size, easy to use, low noise, and good self-priming ability.

- [Integrated Suction Pumps](#)
-

The integrated suction pump has the advantages of simple structure, small size, easy to use, low noise, and good self-priming ability.

Holder

- [Pen Holder](#)

The overall solid color design can be used for writing, drawing and other applications.

- [Phone Holder](#)

Suitable for devices that require physical clamping, such as photography, and can clamp a variety of mobile phones. It has a simple structure and is easy to install and disassemble.

Other functional accessories

- [Dexterous Hand](#)

A robot component that can achieve functions similar to human hands. It has the advantages of a complex structure, firm gripping of objects, not easy to fall, and easy operation. The gripper kit includes gripper accessories and Lego technology parts. The end effector of the robot arm is controlled by a programmable system to achieve functions such as object grasping and multi-point positioning.

- [USB Camera](#)

The USB high-definition camera can be used with suction pumps, adaptive grippers, artificial intelligence kits, etc., and eye in hand can achieve precise positioning and calibration.

- [Bamboo Flange](#)
-

4.1 First-time self-check

Suitable for devices with physical travel such as click buttons and keyboards. The overall solid color design, simple structure, and easy installation and disassembly.

Flat base

Applicable models: myCobot 280



Specifications:

| Name | Flat base |
|--------------------------|--------------------------------------|
| Model | myCobot_Fstand_grey |
| Color | Gray, black |
| Process | ABS compression molding |
| Size | 145×145×13 |
| Weight | 60g |
| Fixed | Lego connector/screw fixation |
| Environment requirements | Normal temperature and pressure |
| Applicable equipment Fit | ER myCobot 280 M5, ER myCobot 280 Pi |

Instructions for use:

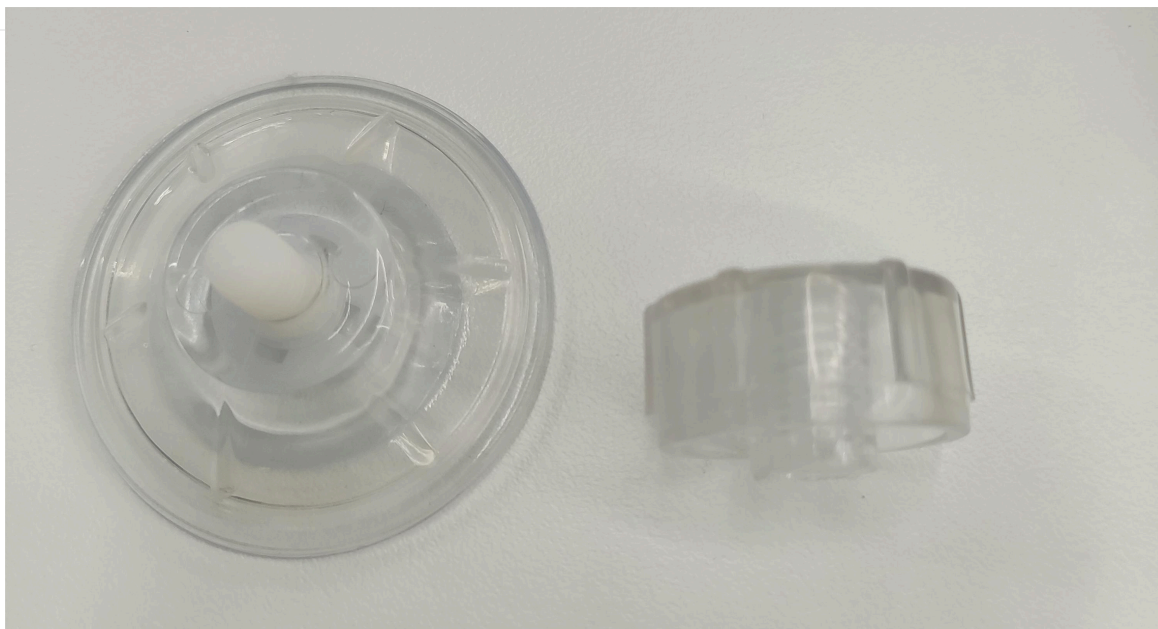
Applicable to flat and smooth surfaces

4.1 First-time self-check

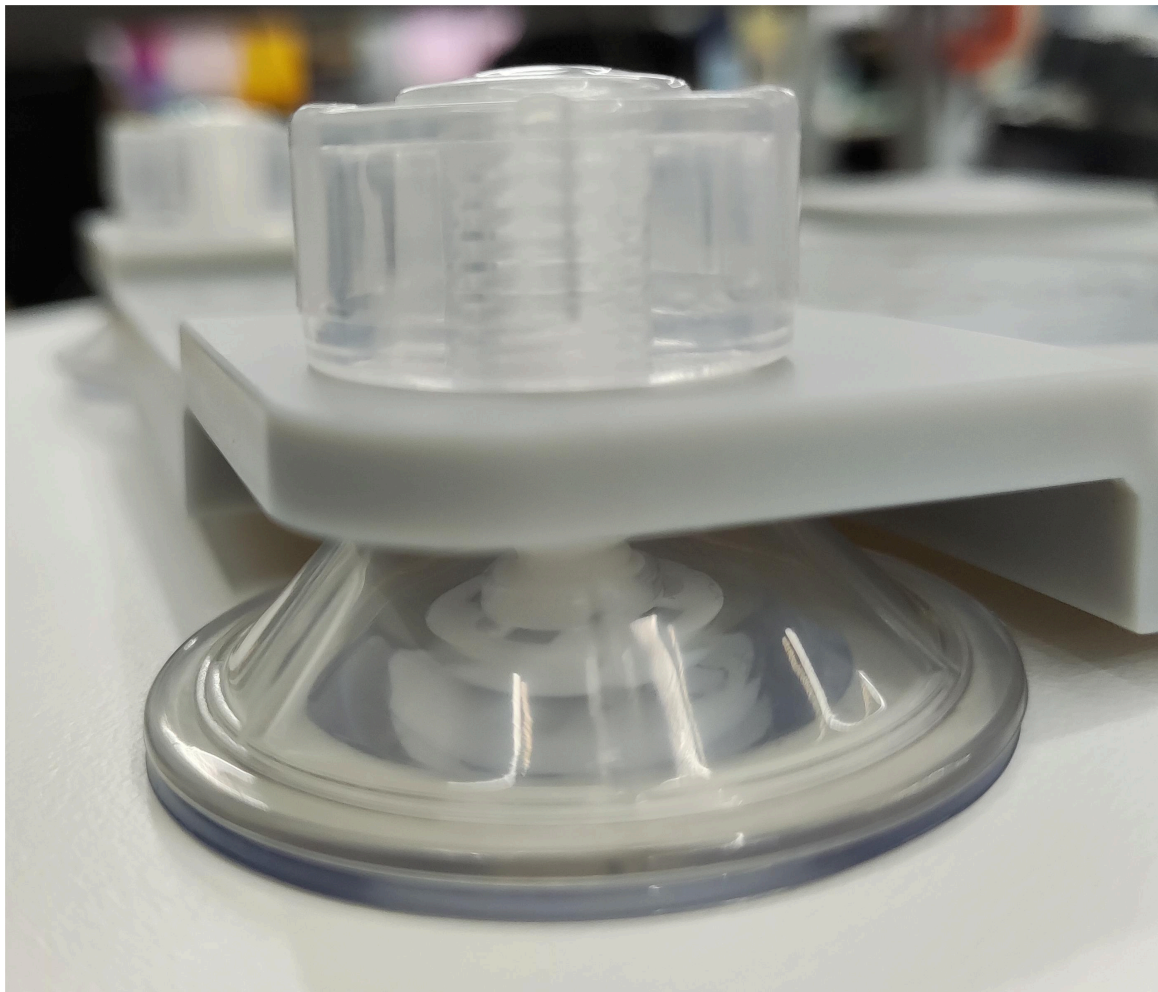
- 1. Install the suction cups at the four corners of the base and tighten them.
- 1. Use the included Lego tech parts to connect the flat base and the bottom of the robot arm.
- 1. Fix the four suction cups on a flat and smooth surface before starting to use.

4.1 First-time self-check

Remove the nut:



Tighten through the holes at the four corners:



4.1 First-time self-check

Adjust the number of Lego connectors as needed. It is recommended to use a sufficient number of connectors to ensure the stability of the machine:



Tips

You can add a small amount of **non-conductive** liquid under the suction cup to fill the gap between the suction cup and the desktop to obtain the best adsorption effect.

4.1 First-time self-check



G-type stand 2.0

Applicable models: myCobot 280, myPalletizer 260, mechArm 270



Specifications:

| Name | G-type stand 2.0 |
|--------------------------|---|
| Model | myCobot_Gstand_grey_V2 |
| Color | Gray, black |
| Process | ABS compression molding |
| Size | 174.8x166x31 |
| Fixed | Lego connector/screw fixation |
| Environment requirements | Normal temperature and pressure |
| Applicable equipment Fit | ER myCobot 280 series, ER myPalletizer 260 series, mecharm 270 series, myBuddy 280 series |

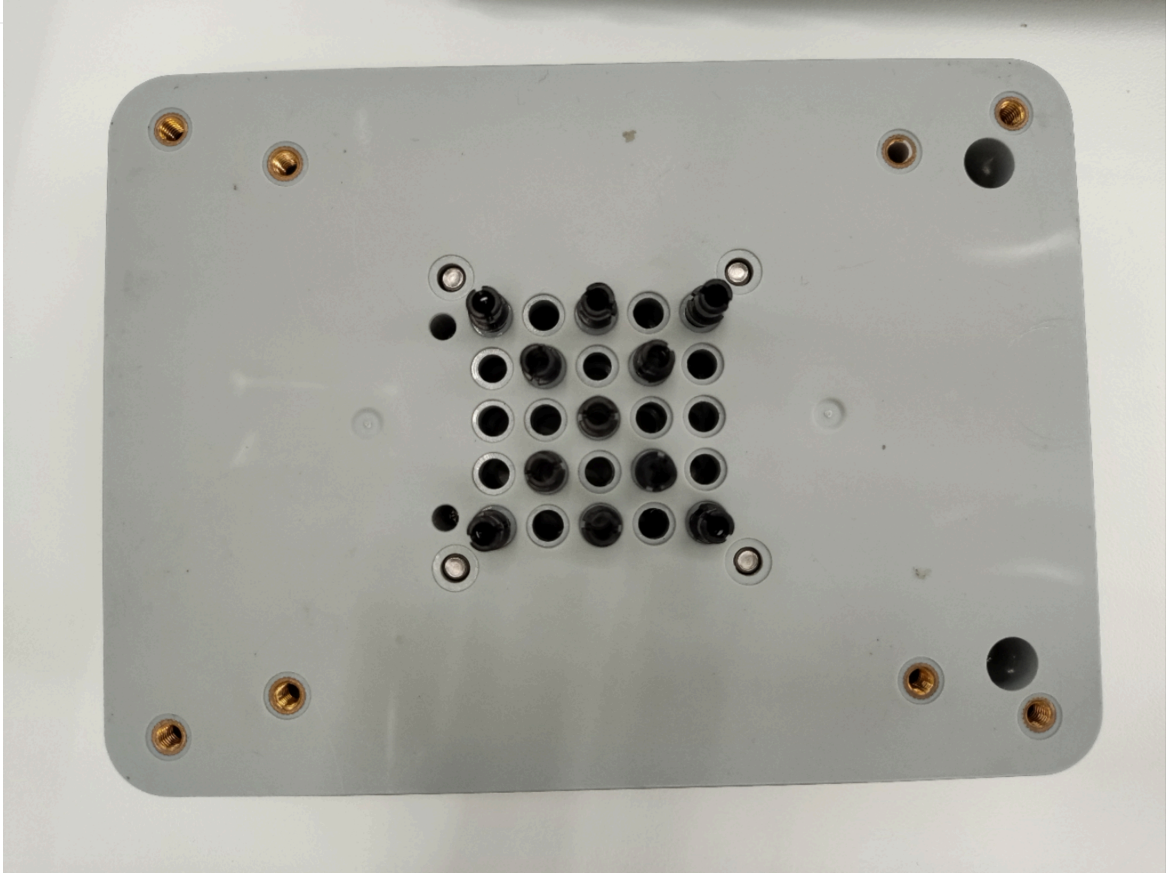
Instructions:

G-type base - fixed to the edge of the table

- 1. Use the G-shaped clamp to fix the base to the edge of the table
- 1. Use the included Lego connector to connect the base and the bottom of the robot arm
- 1. Make sure it is stable before starting to use

4.1 First-time self-check

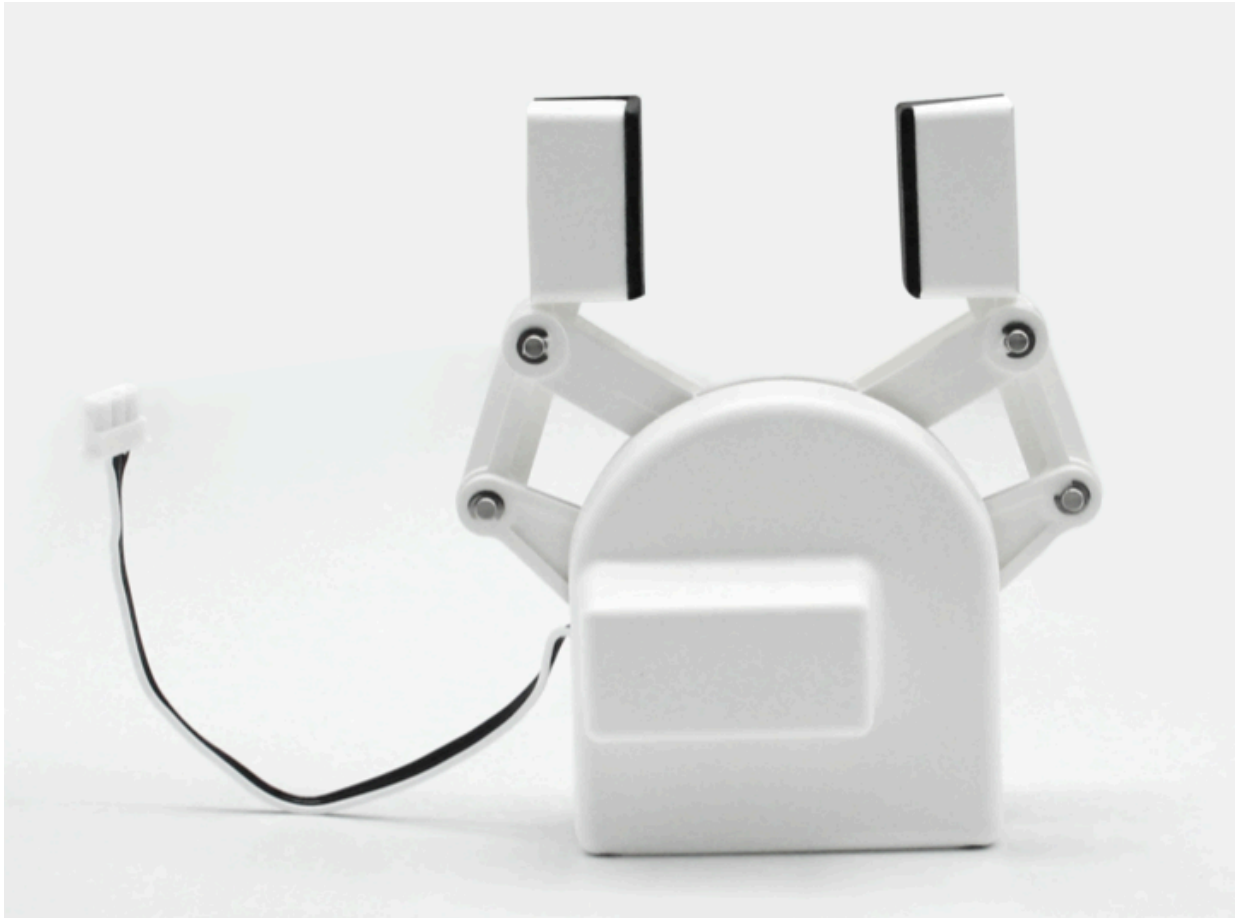
Insert the Lego connector as needed



Adaptive gripper

Applicable models: myCobot 280, myPalletizer 260, mechArm 270

Product image



Specifications:

4.1 First-time self-check

| Name | mycobot280 Adaptive Gripper |
|--------------------------|---|
| Model model | myCobot_gripperAg_white |
| Process | ABS injection molding |
| Color | White |
| Gripping range | 20-45mm |
| Maximum gripping force | 150g |
| Repeatability | 1mm |
| Service life | One year |
| Drive mode | Electric |
| Transmission mode | Gear + connecting rod |
| Dimensions | 112×94×50mm |
| Weight | 110g |
| Fixing method | Lego connector |
| Environment requirements | Normal temperature and pressure |
| Control interface | Serial control |
| Applicable equipment | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series |

Adaptive gripper: Used to grip objects

Introduction

- The gripper is a robot component that can achieve functions similar to human hands. It has the advantages of complex structure, firm gripping of objects, not easy to fall, and easy operation. The gripper kit includes gripper accessories and Lego technology parts. The end effector of the robot arm is controlled by a programmable system to realize functions such as object gripping and multi-point positioning. The gripper can be used in all development environments, such as ROS, Arduino, Roboflow, etc.

Working Principle

- Driven by a motor, the gripper's finger surface makes linear reciprocating motion to achieve opening or closing. The acceleration and deceleration of the electric gripper are controllable, the impact on the workpiece can be minimized, the positioning point is controllable, and the clamping is controllable.

Applicable objects

- Small cubes
- Small balls
- Long objects

Installation and use

4.1 First-time self-check

- Check whether the accessories package is complete: Lego connectors, grippers with connecting wires, and connecting wire extension wires
-



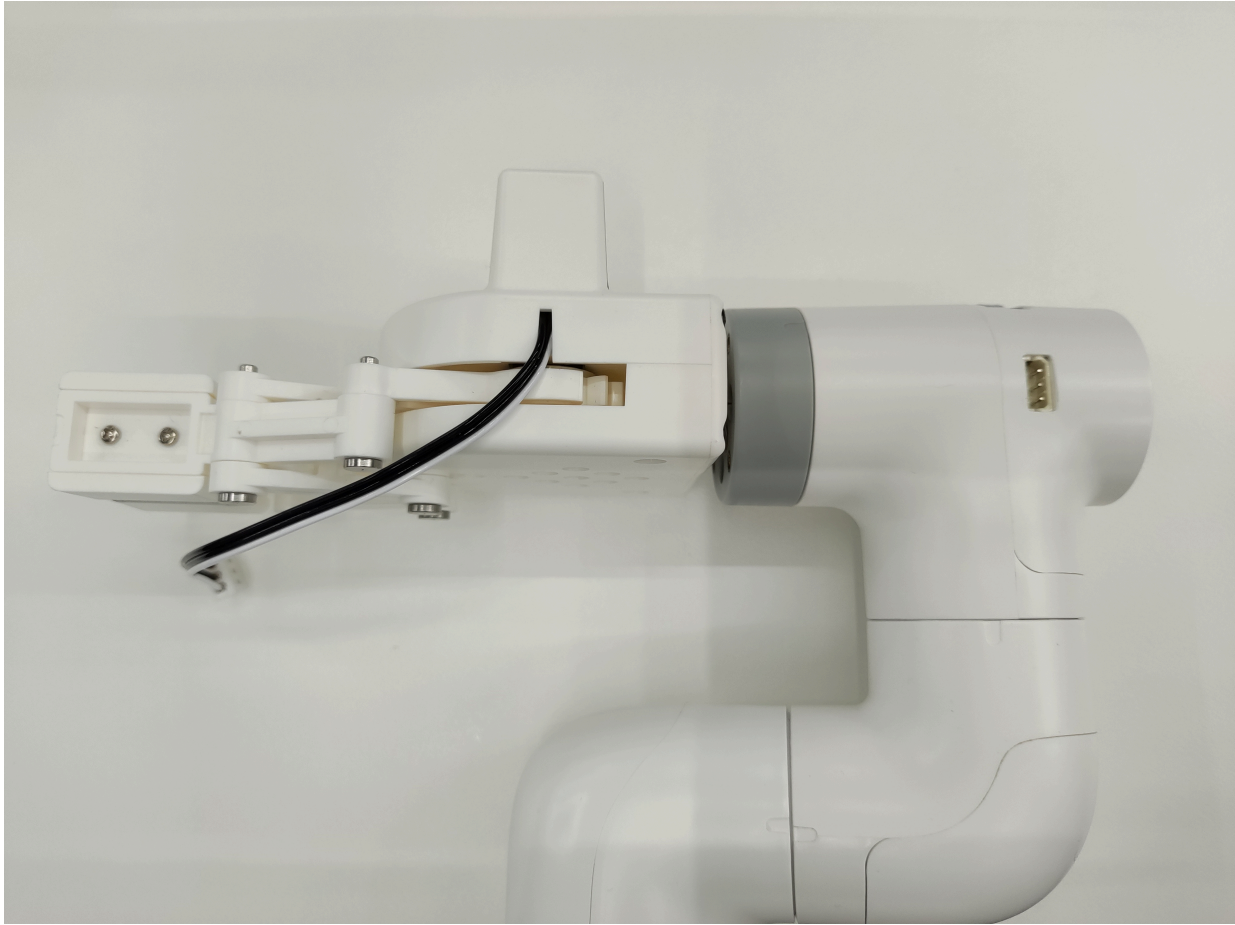
- Gripper installation:
- Structural installation:

Insert the Lego connector into the socket reserved for the gripper. You can choose two different directions for installation as needed:

4.1 First-time self-check

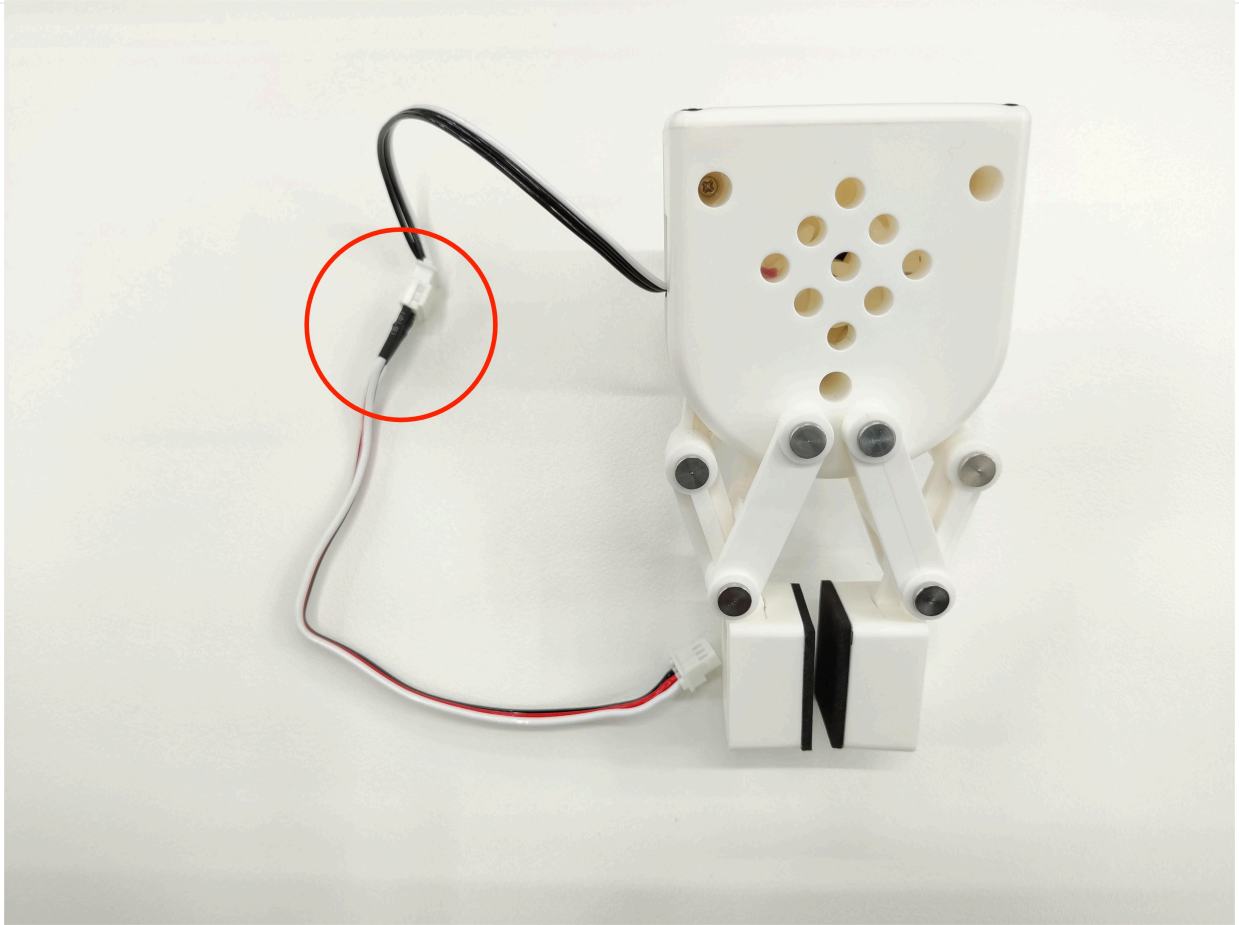


Align the gripper with the connector inserted into the socket at the end of the robot arm and insert it:



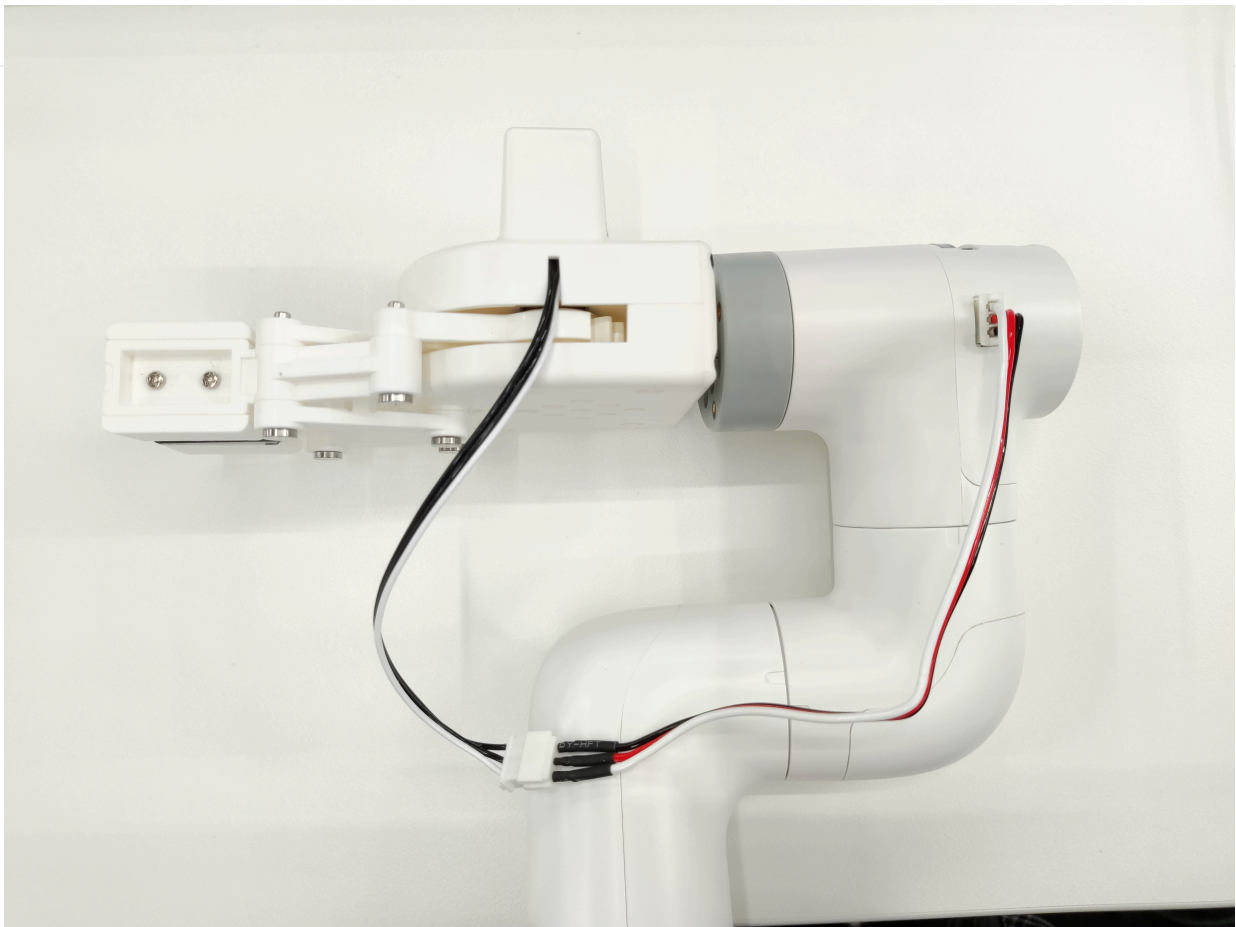
4.1 First-time self-check

- Electrical connection Connect the extension cord to the gripper:



Insert the robot control interface:





Programming development:

Use python to program the gripper

- AR version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
import time

# Initialize a MyCobot280 object
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord

# The following three methods can control the gripper to open-close-open
# Method 1:
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(1, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)
# Method 2:
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# mc.set_gripper_value(0, 80)
# time.sleep(3)
# mc.set_gripper_value(100, 80)
# time.sleep(3)

# Method three:
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
# mc.set_encoder(7, 1500, 20)
# time.sleep(3)
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
```

- Pi version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
from pmycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables
import time

# Initialize a MyCobot280 object
mc = MyCobot280(PI_PORT, PI_BAUD)

# The following three methods can control the gripper to open-close-open
# Method 1:
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(1, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)
# Method 2:
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# mc.set_gripper_value(0, 80)
# time.sleep(3)
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# Method 3:
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
# mc.set_encoder(7, 1500, 20)
# time.sleep(3)
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
```

Save the file and close it. Return to the command line terminal and enter:

```
python grip.py
```

You can see the gripper open-close-open

Parallel gripper

Applicable models: myCobot 280, myPalletizer 260, mechArm 270

Product image



Specifications:

4.1 First-time self-check

| Name | mycobot280 parallel gripper |
|--------------------------|---|
| Model model | myCobot_gripper_parallel |
| Process | ABS injection molding |
| Color | White |
| Gripping range | <20mm (effective 15mm) |
| Maximum gripping force | 150g |
| Repeatability | 1mm |
| Service life | One year |
| Drive mode | Electric |
| Transmission mode | Gear + connecting rod |
| Dimensions | 66×78×46mm |
| Weight | 84g |
| Fixing method | Lego connector |
| Environment requirements | Normal temperature and pressure |
| Control interface | Serial control |
| Applicable equipment | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series |

Parallel gripper: Used to grip objects

Introduction

- The gripper is a robot component that can achieve functions similar to human hands. It has the advantages of complex structure, firm gripping of objects, not easy to fall, and easy operation. The gripper kit includes gripper accessories and Lego technology parts. The end effector of the robot arm is controlled by a programmable system to realize functions such as object gripping and multi-point positioning. The gripper can be used in all development environments, such as ROS, Arduino, Roboflow, etc.

Working Principle

- Driven by a motor, the gripper's finger surface makes linear reciprocating motion to achieve opening or closing. The acceleration and deceleration of the electric gripper are controllable, the impact on the workpiece can be minimized, the positioning point is controllable, and the clamping is controllable.

Applicable objects

- Small cubes
- Small balls
- Long objects

Installation and use

4.1 First-time self-check

- Check whether the accessories package is complete: Lego connectors, grippers with connecting wires

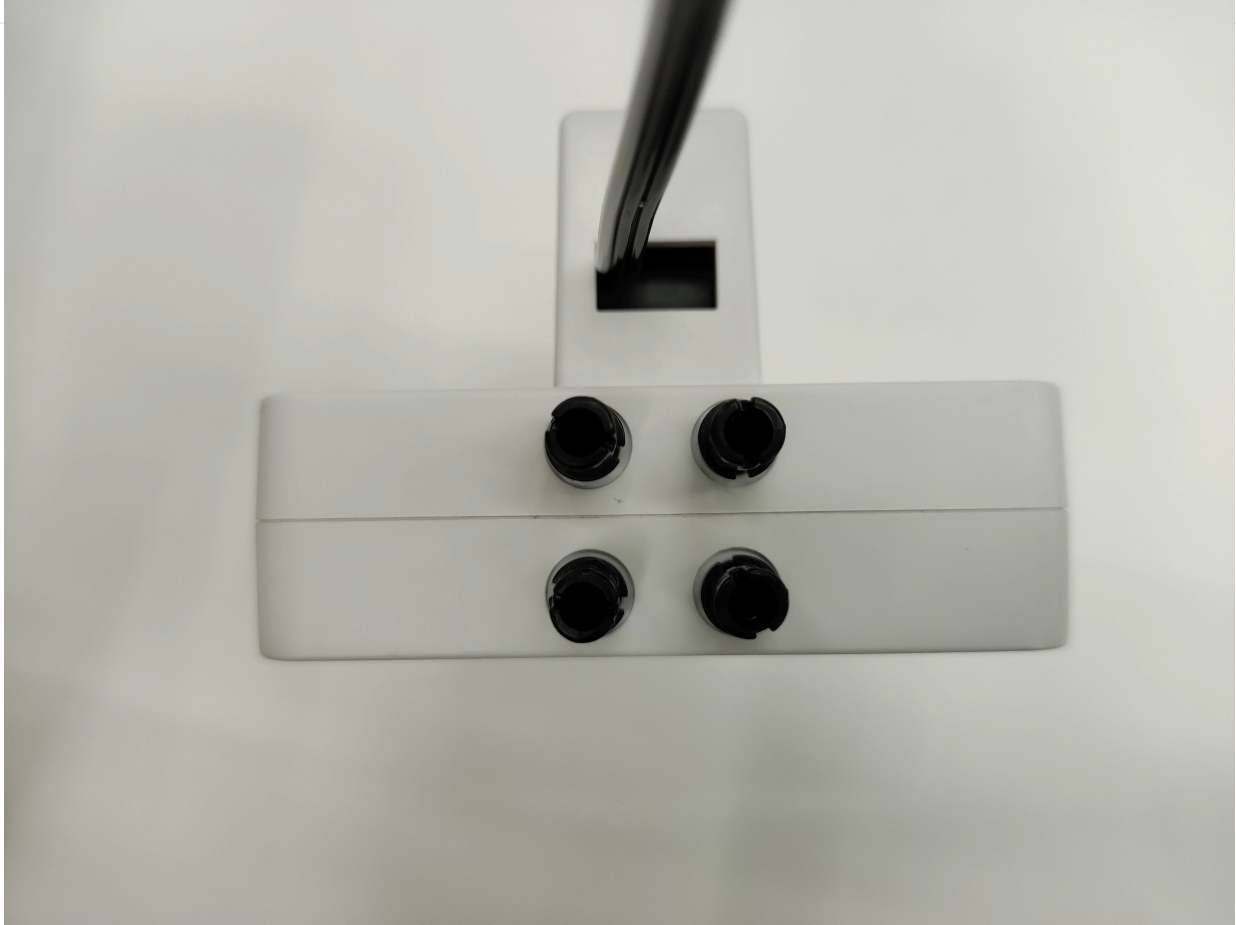


- Gripper installation:

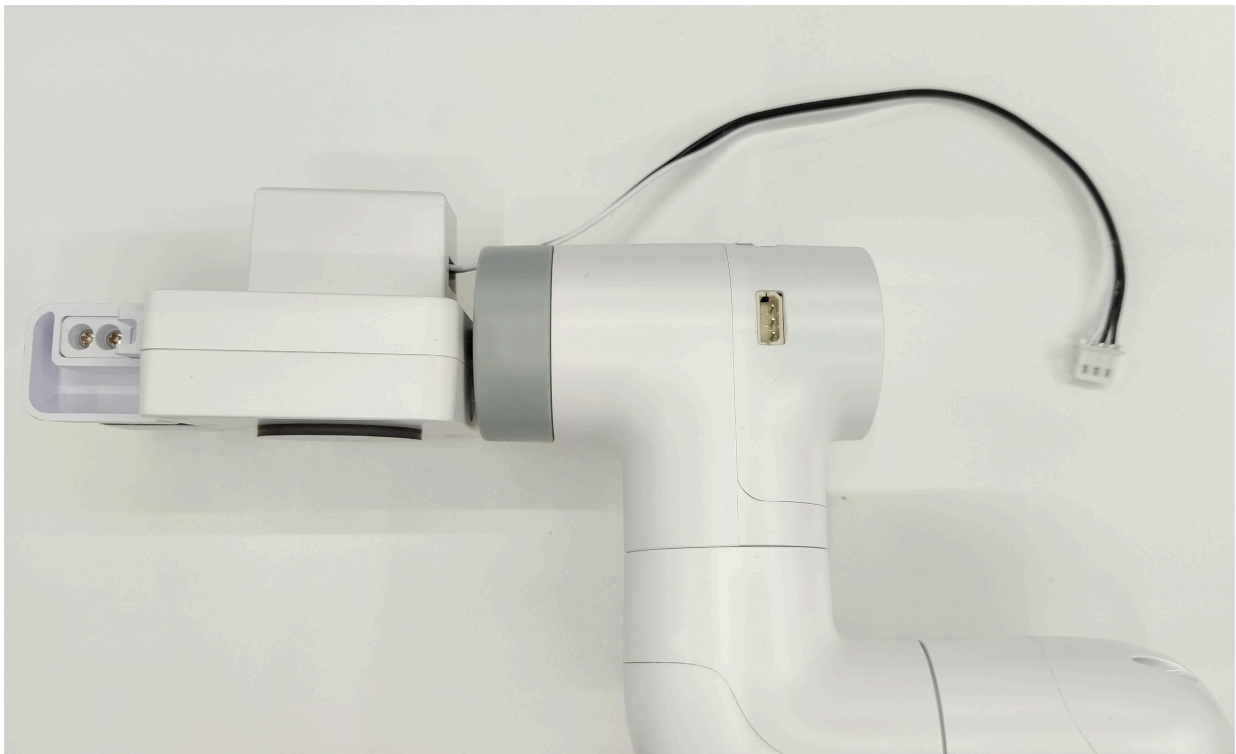
Structural installation:

4.1 First-time self-check

Insert the Lego connector into the reserved socket of the gripper:



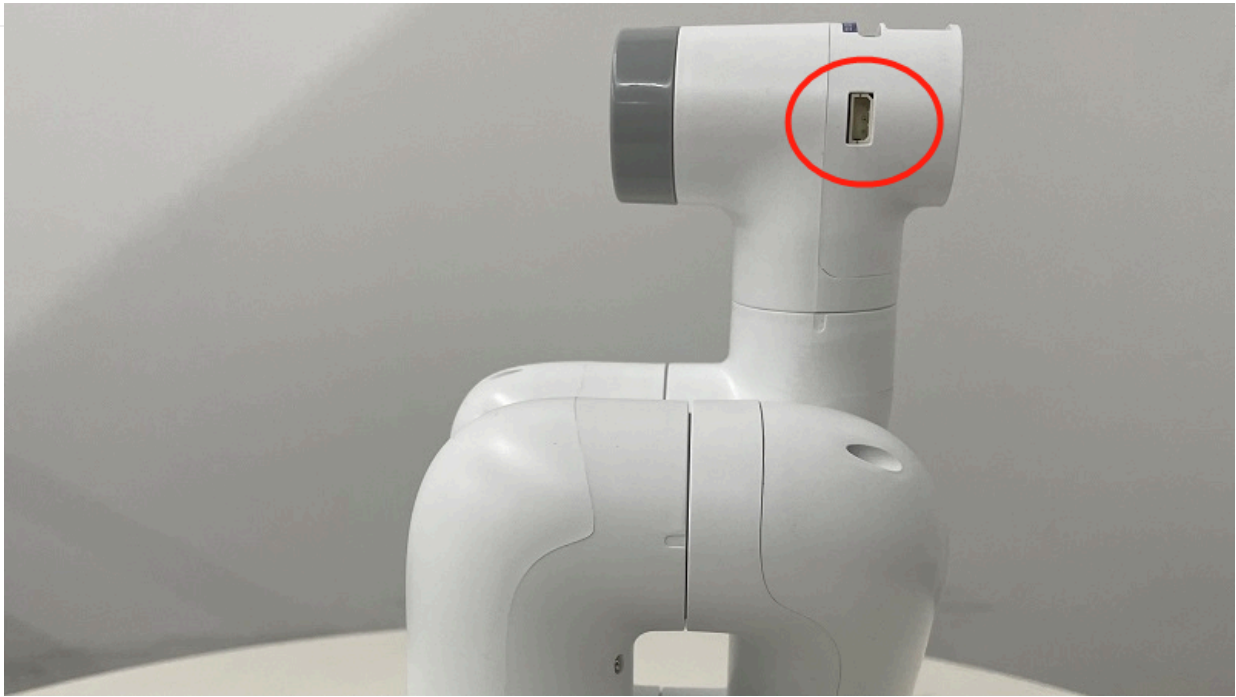
Align the gripper with the connector inserted into the socket at the end of the robot arm and insert it:



Electrical connection:

4.1 First-time self-check

Insert the robot arm control interface:



Programming and development:

Use python to program the gripper

4.1 First-time self-check

- AR version:

```
from pycobot import MyCobot280
import time

# Initialize a MyCobot280 object
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord

# The following three methods can control the gripper to open-close-open
# Method 1:
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(1, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)

# Method two:
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# mc.set_gripper_value(0, 80)
# time.sleep(3)
# mc.set_gripper_value(100, 80)
# time.sleep(3)

# Method three:
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
# mc.set_encoder(7, 1500, 20)
# time.sleep(3)
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
```

- Pi version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
from pmycobot import PI_PORT, PI_BAUD
# When using the Raspberry Pi version of mycobot, you can reference these two variables to initialize MyCobot280
import time

# Initialize a MyCobot280 object
mc = MyCobot280(PI_PORT, PI_BAUD)

# The following three methods can control the gripper to open-close-open
# Method 1:
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(1, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)

# Method 2:
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# mc.set_gripper_value(0, 80)
# time.sleep(3)
# mc.set_gripper_value(100, 80)
# time.sleep(3)

# Method 3:
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
# mc.set_encoder(7, 1500, 20)
# time.sleep(3)
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
```

Save the file and close it, return to the command line terminal, and enter:

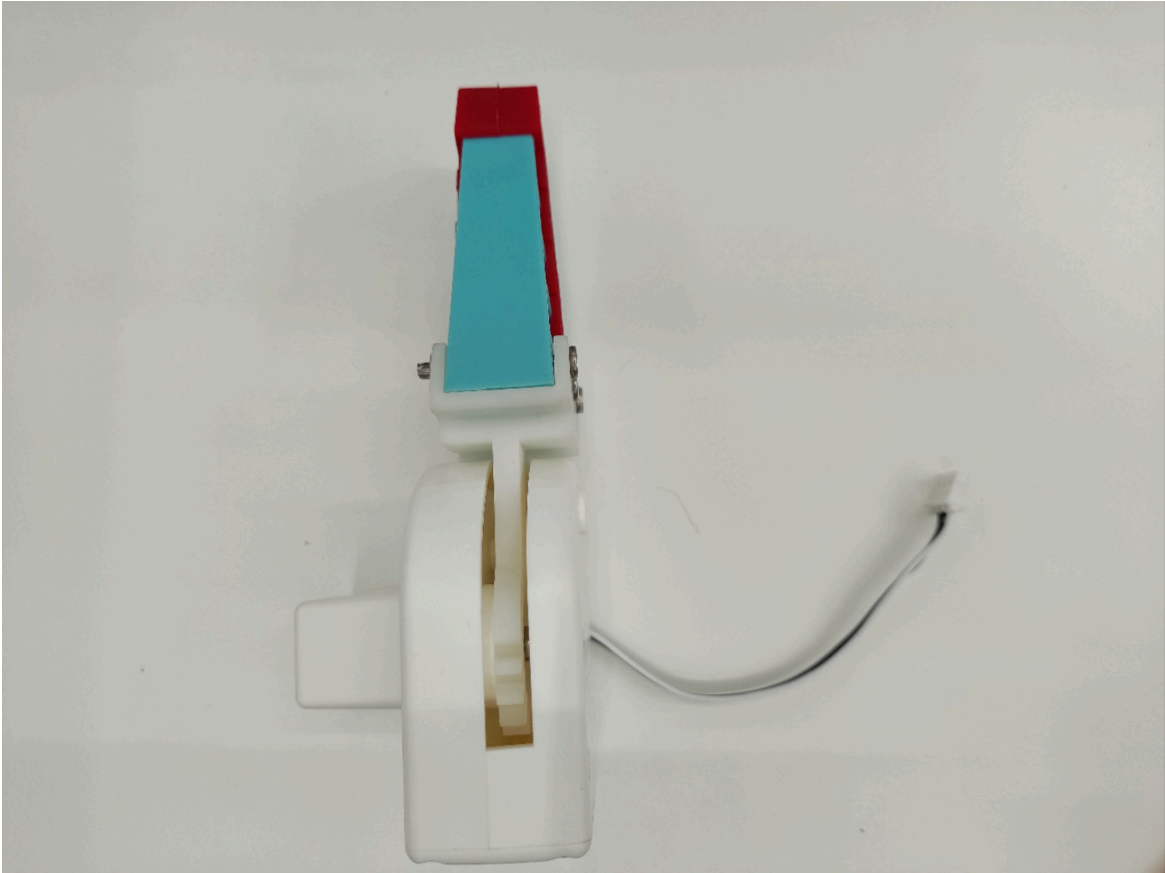
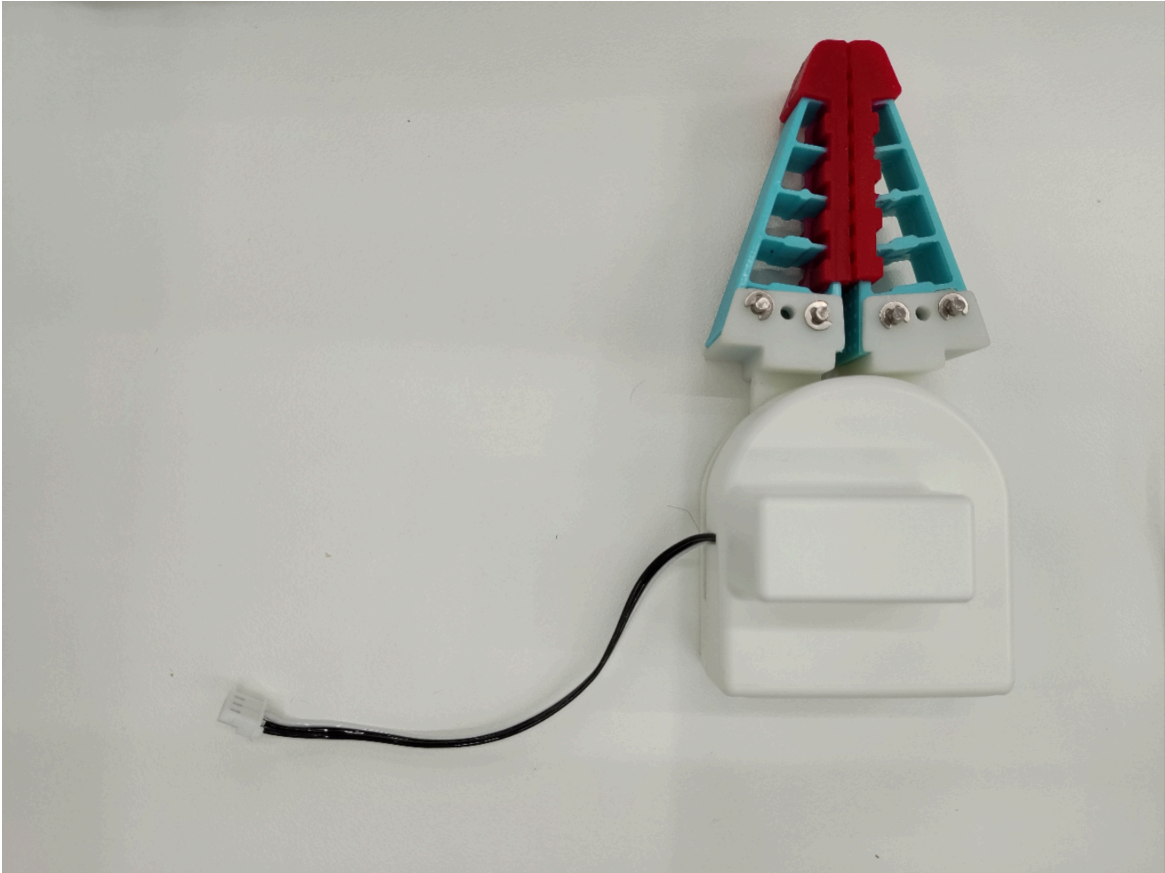
```
python grip.py
```

You can see the gripper open-close-open

Flexible Gripper - Open-leg Type

Applicable models: myCobot 280, myPalletizer 260, mechArm 270, myBuddy 280

Product Image



Specifications:

| Name | mycobot280 Open-leg Gripper |
|----------------------------|---|
| Model model | myCobot Open-leg Gripper |
| Process | Photosensitive resin |
| Color | White |
| Repeatability | ±1mm |
| Service life | One year |
| Drive mode | Electric |
| Fixing mode | Lego connector |
| Environmental requirements | Normal temperature and pressure |
| Control interface | Serial control |
| Applicable equipment | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series |

Flexible gripper: Used to grip objects

Introduction

- Traditional industrial suction cups need to suck the flat surface of the material. In more and more working conditions, the suction surface is easy to damage the panel or components. The soft-touch gripper grabs the edge and easily transports the panel without marks or damage, ensuring that the product surface is flawless and improving the yield rate.
- The modular design of the soft-touch gripper is light in weight and can be freely arranged and combined according to the size of the panel.
- The clamping force of traditional cylinders is generally large, and the force is difficult to control. The edge of the clamped panel is easy to be clamped and warped. The single-finger clamping force of the flexible gripper is precisely controllable and will not clamp fragile workpieces.

Working Principle

- The flexible gripper is an innovative bionic flexible gripper developed by researchers imitating the shape of the arms and legs of a starfish. The "fingers" of the soft gripper are made of flexible polymer silicone material, which can bend and deform by inflation. It can adaptively wrap around the target object like a starfish, and can complete the flexible and non-destructive grasping of irregular and fragile objects.

Applicable objects

- Any object of any shape within a reasonable size

Installation and use

- Check whether the accessories package is complete: Lego connector, gripper with connecting wire, extension wire

4.1 First-time self-check



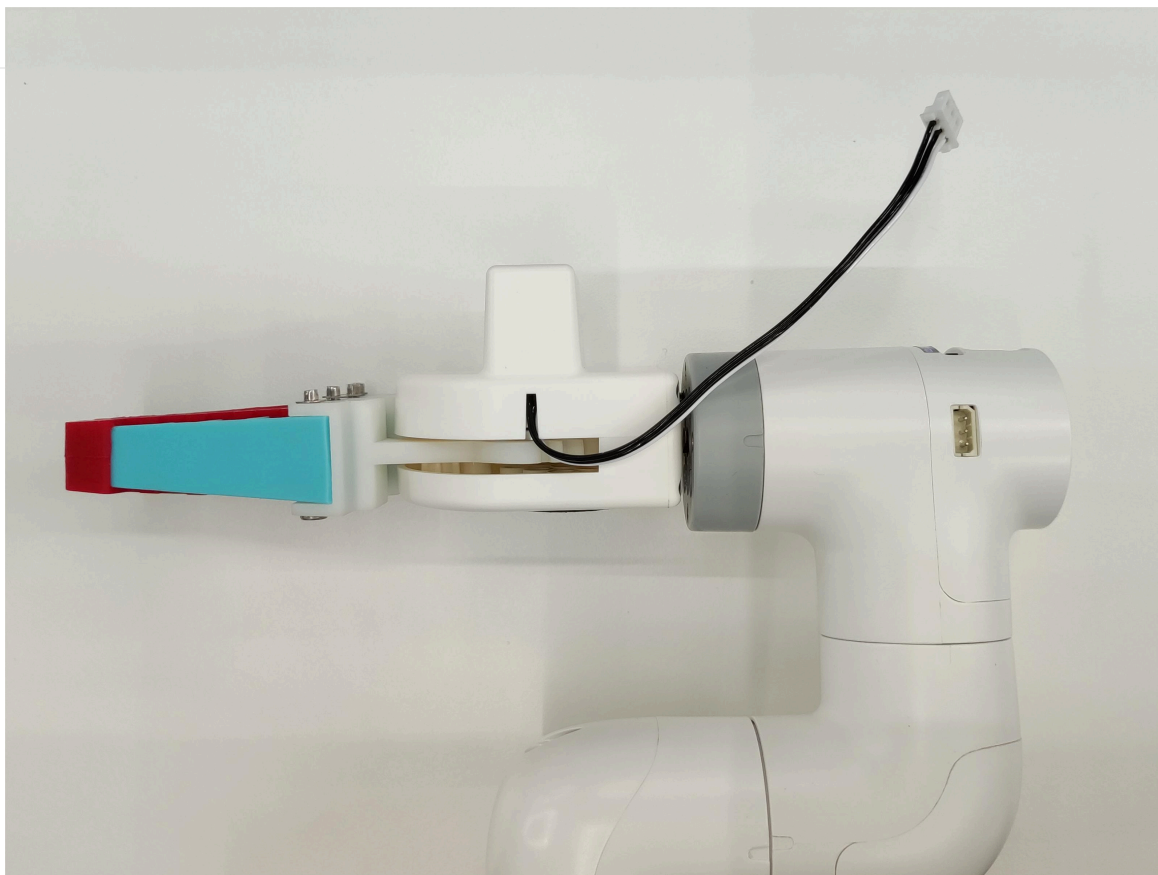
- Gripper installation:

Structural installation: Insert the Lego connector into the reserved socket of the gripper. You can choose two different directions for installation as needed:



Align the gripper with the connector inserted into the socket at the end of the robot arm and insert it:

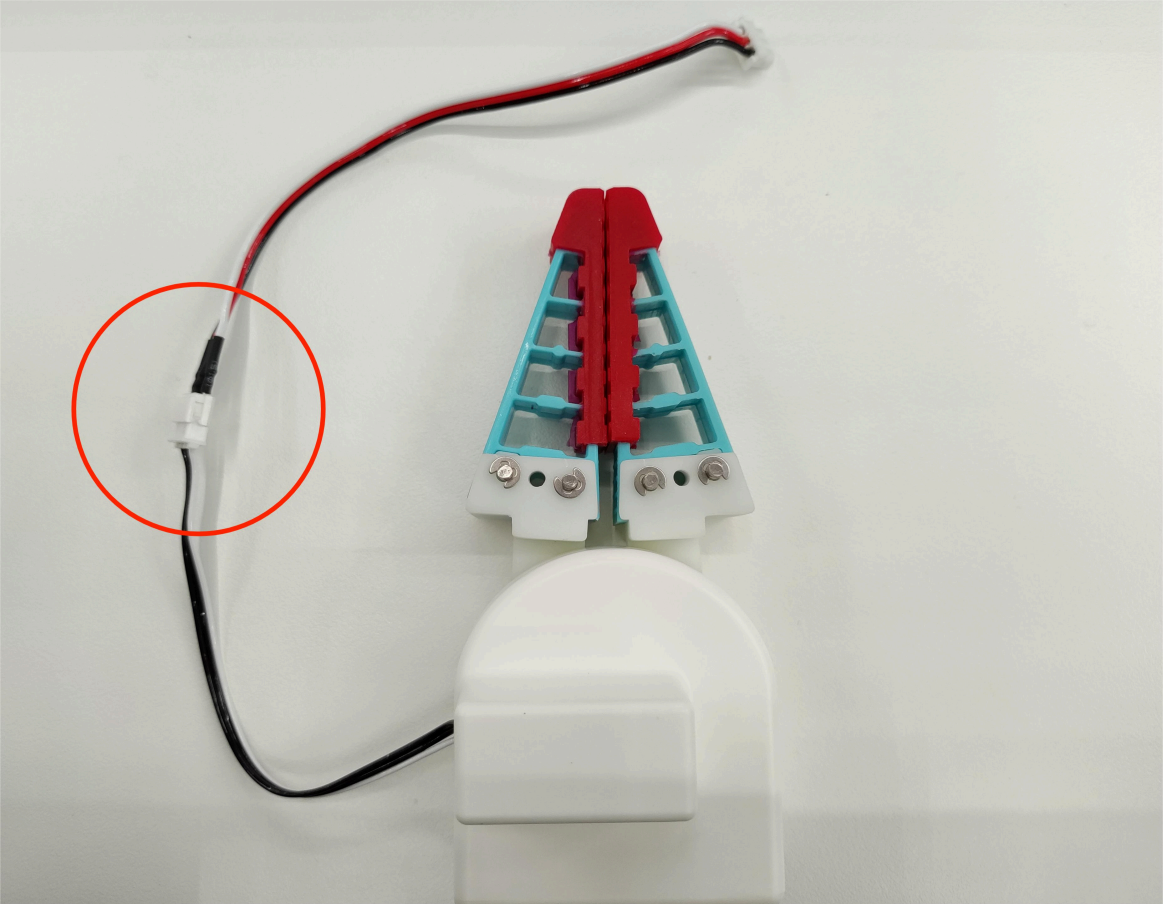
4.1 First-time self-check



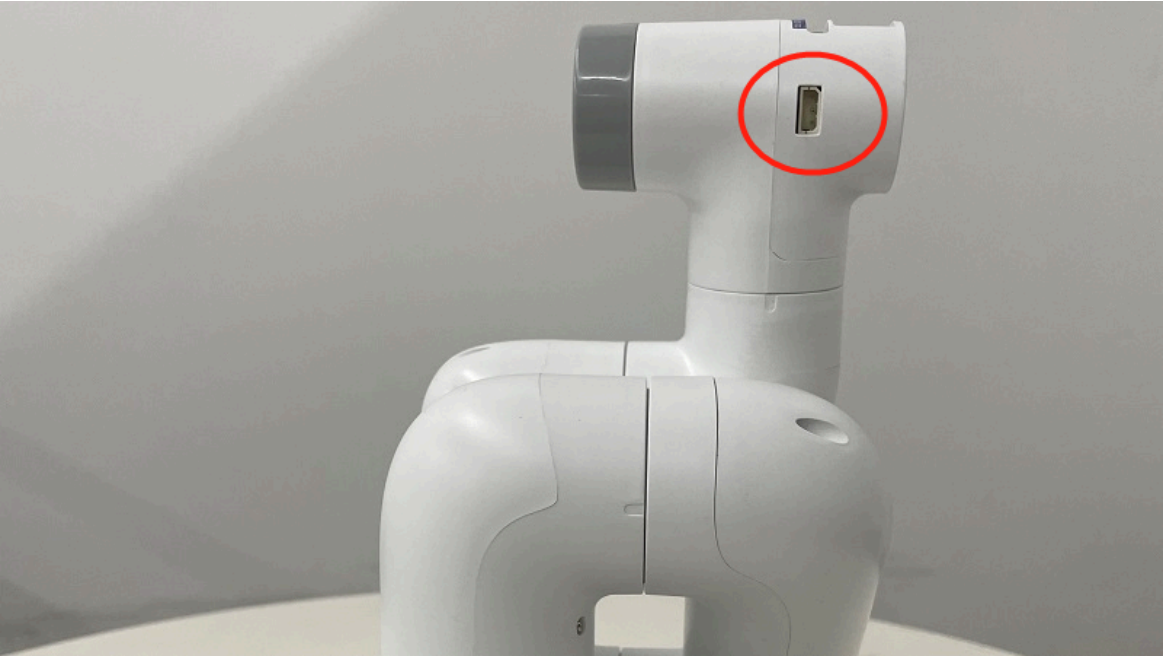
- Electrical connection:

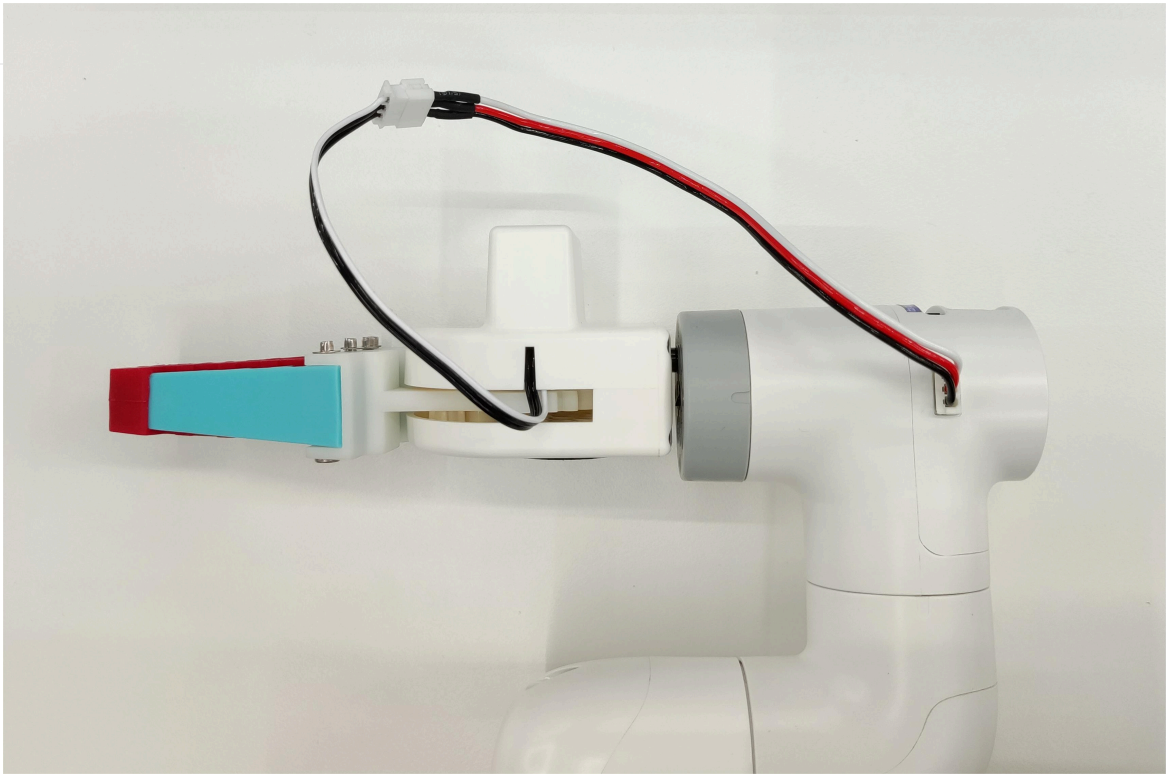
Connect the extension wire to the gripper:

4.1 First-time self-check



Insert the robot control interface:





Programming development

- AR version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
import time

# Initialize a MyCobot280 object
mc = MyCobot280("COM3", 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it accord

# The following three methods can control the gripper to open-close-open
# Method 1:
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(1, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)
# Method 2:
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# mc.set_gripper_value(0, 80)
# time.sleep(3)
# mc.set_gripper_value(100, 80)
# time.sleep(3)

# Method three:
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
# mc.set_encoder(7, 1500, 20)
# time.sleep(3)
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
```

- Pi version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
from pmycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables
import time

# Initialize a MyCobot280 object
mc = MyCobot280(PI_PORT, PI_BAUD)

# The following three methods can control the gripper to open-close-open
# Method 1:
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(1, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)
mc.set_gripper_state(0, 80)
time.sleep(3)
# Method 2:
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# mc.set_gripper_value(0, 80)
# time.sleep(3)
# mc.set_gripper_value(100, 80)
# time.sleep(3)
# Method 3:
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
# mc.set_encoder(7, 1500, 20)
# time.sleep(3)
# mc.set_encoder(7, 2048, 20)
# time.sleep(3)
```

Save the file and close it, return to the command line terminal, and enter:

```
python grip.py
```

You can see the gripper open-close-open

Vertical Suction Pump V2.0

Applicable models: myCobot 280, myPalletizer 260, mechArm 270

Product Image



Specifications

4.1 First-time self-check

| Name | myCobot Vertical Suction Pump V2.0 |
|---------------------------------|--|
| Model | myCobot_suctionPump_V2.0_grey |
| Material | ABS injection molding |
| Color | White |
| Dimensions | Suction Pump Box: 72x52x37 Suction Pump End: 63x24.5x26.7 |
| Number of Suction Cups | 1 |
| Suction Cup Size | Diameter 20mm |
| Suction Weight | 150g |
| Power Source Equipment | Suction pump box |
| Service life | One year |
| Fixing method | Lego connector |
| Control interface | IO control |
| Use environment requirements | Normal temperature and pressure |
| Applicable equipment | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series |

Suction pump: Used for adsorbing objects

Introduction

- Suction pump, that is, vacuum adsorption pump, has one suction nozzle and one exhaust nozzle. It has the advantages of simple structure, small size, easy use, low noise, and good self-priming ability. By controlling the suction pump kit as the end effector of the robot arm, the function of adsorbing objects is performed.
- Suction pump accessories: power cord x1, Dupont wire x10, one-input and two-output connection wire x1, several Lego tech parts

Working principle

- When sucking objects: the air pump starts to suck air and adsorbs the objects and then stops, and there will be no air leakage in a short time.
- When putting down the objects: the electronic valve starts, the air release valve opens, and air enters the vacuum suction cup to separate from the sucked objects.

Applicable objects

- Paper/plastic sheets
- Flat and smooth objects
- Cards, etc.

4.1 First-time self-check



Installation and use

- Check whether the accessories package is complete: Lego connectors, Dupont wires, double-head suction pump



4.1 First-time self-check

- Double-head suction pump installation:

Structural installation:

Insert the Lego connector into the reserved socket on the suction pump:



Align the suction pump with the connector plugged in with the socket at the end of the robotic arm and insert it:

4.1 First-time self-check

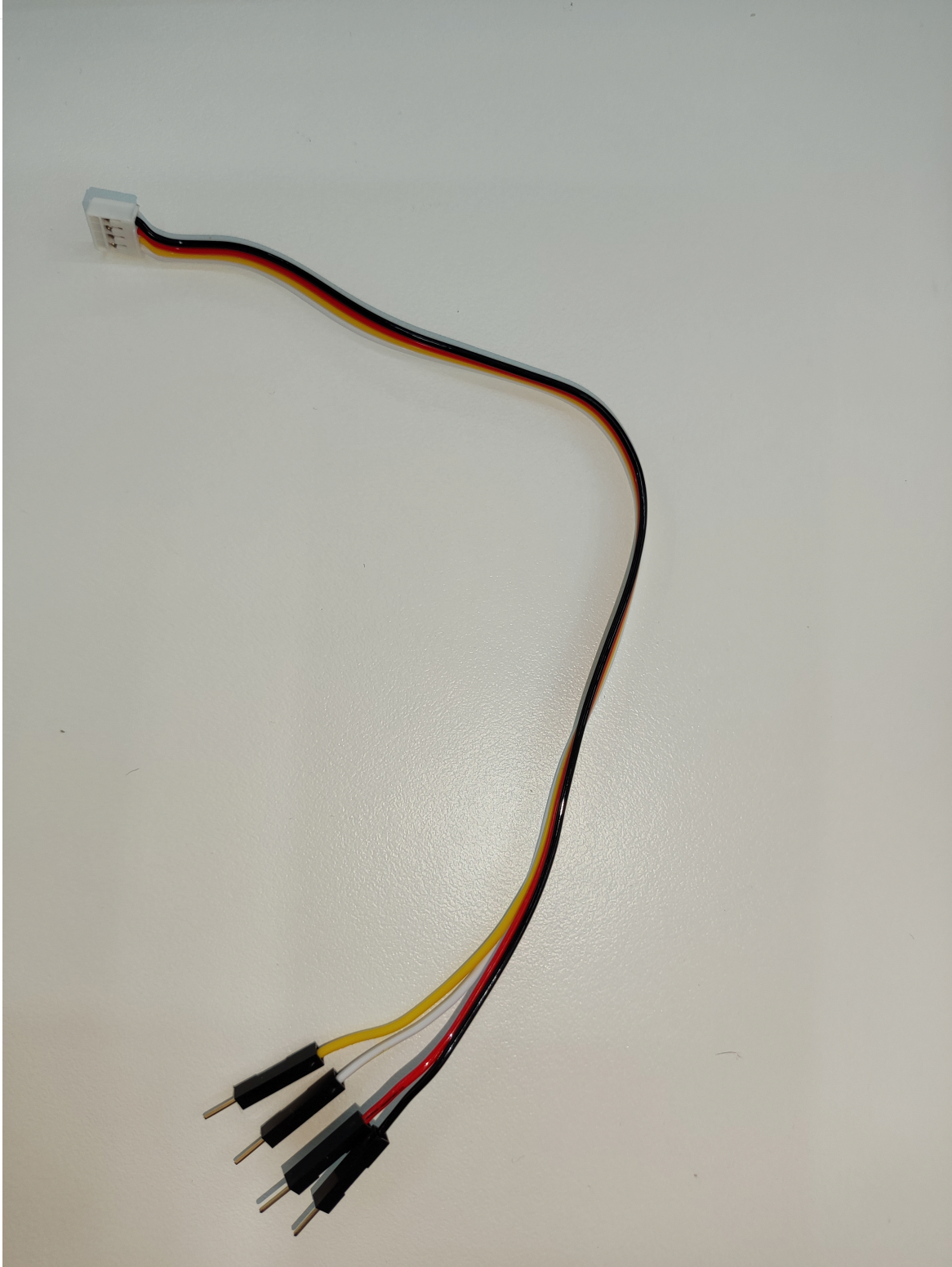


- Electrical connection:

Select the male-female DuPont wire and insert the female end into the socket marked with pins on the suction pump box:

4.1 First-time self-check

Male-female DuPont wire:

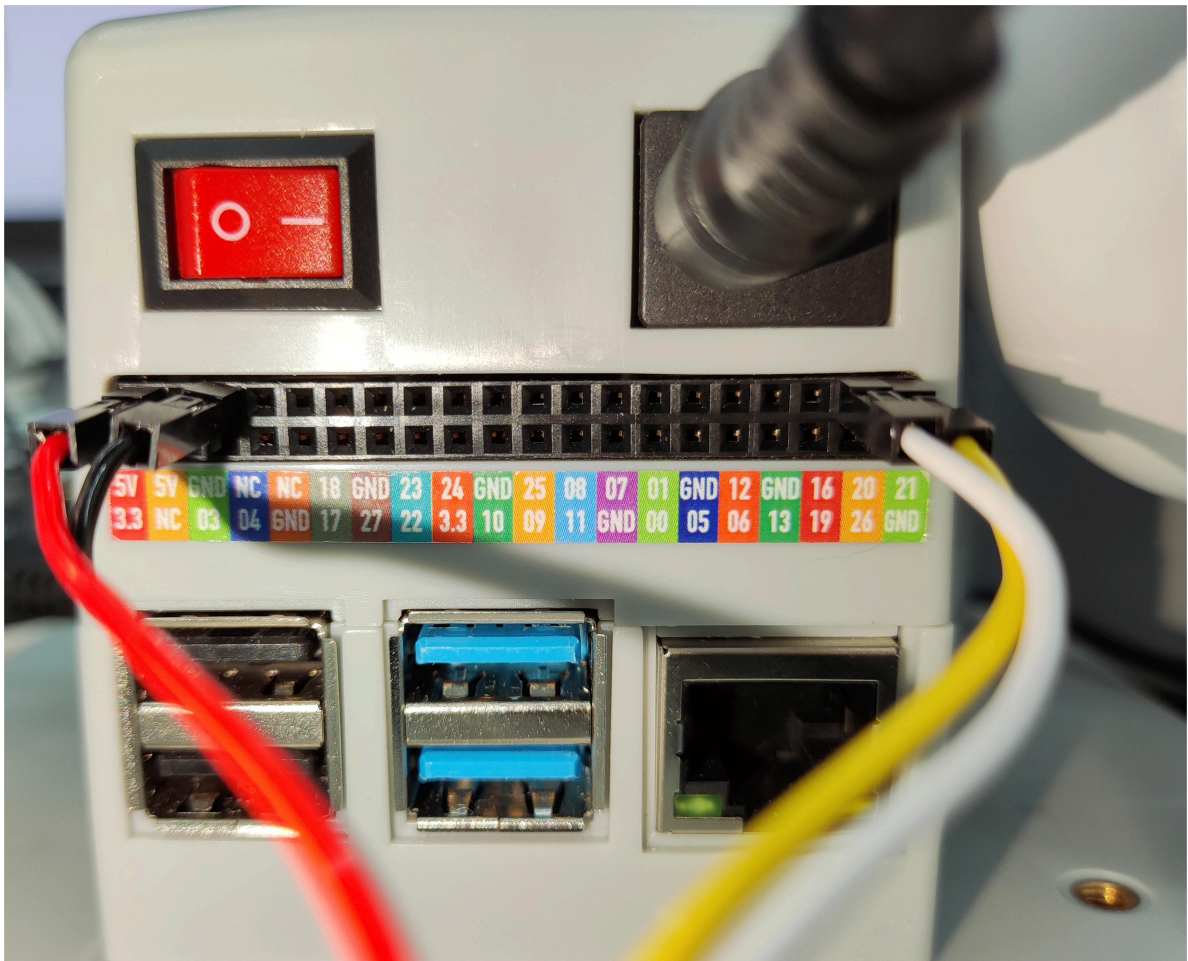


Note the correspondence between the DuPont wire colors and pins in the figure:

4.1 First-time self-check



1. Insert the male end into the robot base pin according to the given correspondence:



4.1 First-time self-check

The left side is the suction pump pin and the right side is the robot pin GND -> GND 5V -> 5V G2 -> 21 G5 -> 20

- **Programming development:**
 - Use python to program the suction pump

The code is as follows:

- **280-AR version:**

```
from pmycobot import MyCobot280
import time

# Initialize a MyCobot280 object
arm = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the actual situation

# Turn on the pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

# Stop the pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)

for i in range(2):
    pump_on()
    time.sleep(2)
    pump_off()
    time.sleep(2)
```

- **280-Pi version:**

4.1 First-time self-check

```
from pmycobot import MyCobot280
from pmycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables
import time
import RPi.GPIO as GPIO

# Initialize a MyCobot280 object
mc = MyCobot280(PI_PORT, PI_BAUD)

# Initialize
GPIO.setmode(GPIO.BCM)
# Pins 20/21 control the solenoid valve and the deflation valve respectively
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# Turn on the suction pump
def pump_on():
    # Open the solenoid valve
    GPIO.output(20, 0)

# Stop the suction pump
def pump_off():
    # Close the solenoid valve
    GPIO.output(20, 1)
    time.sleep(0.05)
    # Open the deflation valve
    GPIO.output(21, 0)
    time.sleep(1)
    GPIO.output(21, 1)
    time.sleep(0.05)

pump_off()
time.sleep(3)
pump_on()
time.sleep(3)
pump_off()
time.sleep(3)
GPIO.cleanup() # Release the pin channel
```

Dual-head suction pump

Applicable models: myCobot 280, myPalletizer 260, mechArm 270

Product image



Specifications

4.1 First-time self-check

| Name | Dual-head suction pump |
|--------------------------|---|
| Model | myCobot_DualPump_grey |
| Material | Photosensitive resin/nylon 7100 |
| Color | White+black |
| Size | Pump end: 63x24.5x26.7 |
| Number of suction cups | 2 |
| Suction cup size | Diameter 20mm |
| Suction weight | 150g |
| Power source equipment | Pump box |
| Service life | One year |
| Fixing method | Lego connector |
| Control interface | IO control |
| Environment requirements | Normal temperature and pressure |
| Applicable equipment | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series |

Suction pump: Used for adsorbing objects

Introduction

- Suction pump, that is, vacuum adsorption pump, has two suction nozzles and two exhaust nozzles for one inlet and one outlet. It is more stable than single-head suction pump. It has the advantages of simple structure, small size, easy use, low noise, and good self-priming ability. By controlling the suction pump kit as the end effector of the robot arm, the function of adsorbing objects is performed.
- Suction pump accessories: power cord x1, DuPont line x10, one-input and two-output connection line x1, Lego technology parts x several

Working principle

- When sucking objects: the air pump starts to suck air and adsorb objects and then stops, and there will be no leakage in a short time.
- When putting down the object: the electronic valve starts, the air release valve opens, and air enters the vacuum suction cup to separate from the sucked object.

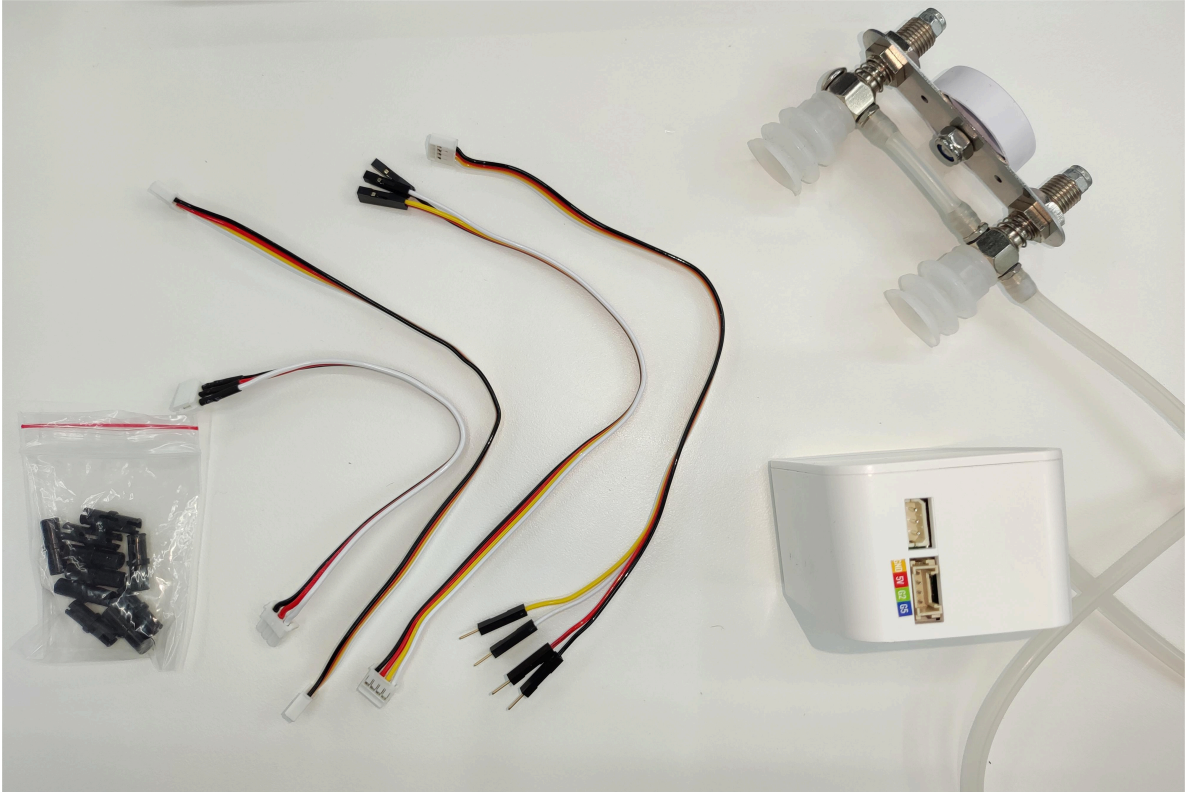
Applicable objects

- Paper/plastic sheets
- Flat and smooth objects
- Cards, etc.

Installation and use

4.1 First-time self-check

- Check whether the accessories package is complete: Lego connectors, Dupont wires, double-head suction pump



- Double-head suction pump installation:

Structural installation:

4.1 First-time self-check

Insert the Lego connector into the reserved socket on the suction pump:



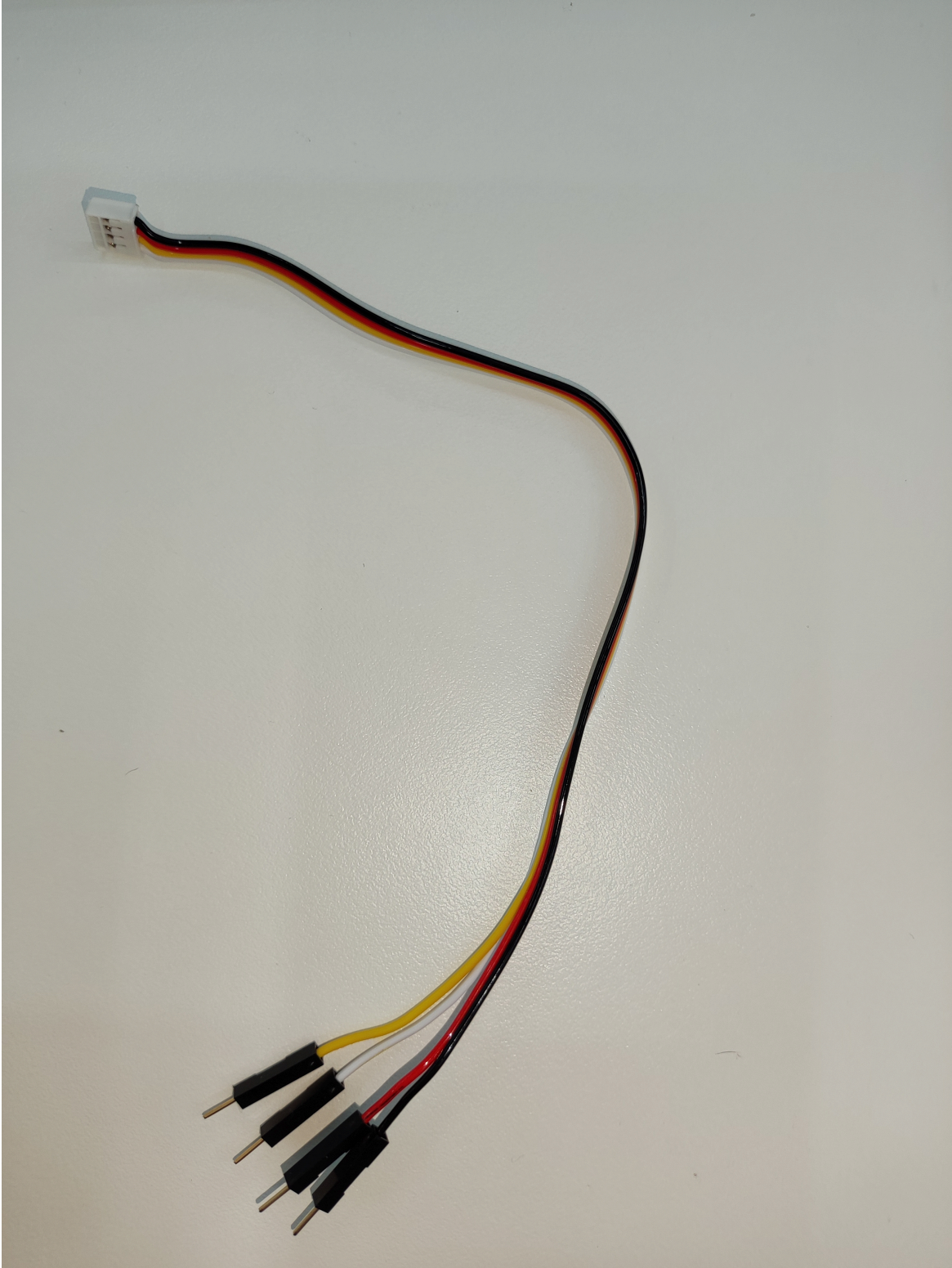
Align the suction pump with the connector plugged in with the socket at the end of the robotic arm and insert it:



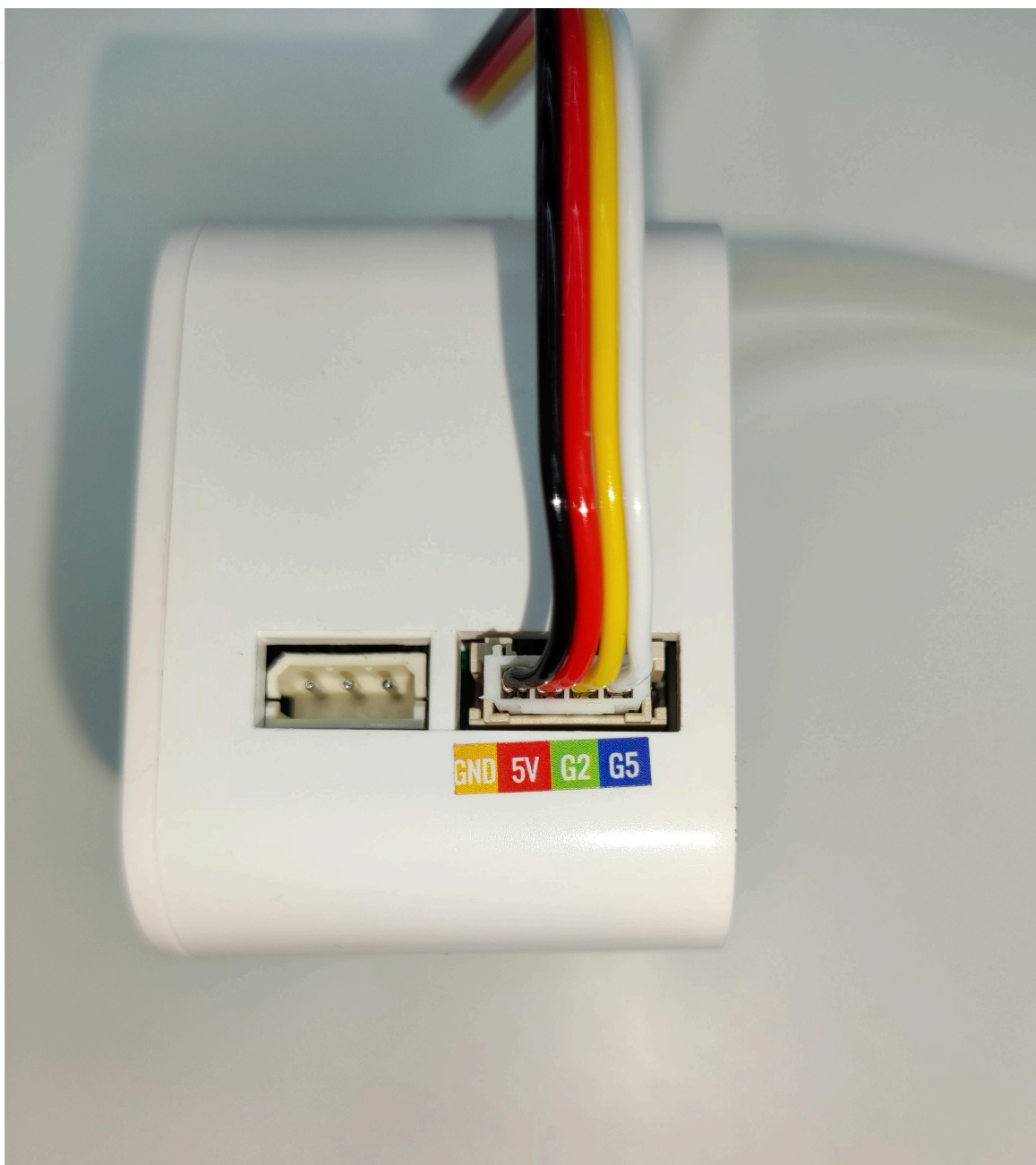
4.1 First-time self-check

- Electrical connection:

Select the male-female DuPont wire, and insert the female end into the socket marked with pins on the suction pump box:



Note the correspondence between the DuPont wire colors and pins in the figure:



Insert the male end into the robot base pin according to the given correspondence:



The left side is the suction pump pin, and the right side is the robot arm pin GND -> GND 5V -> 5V G2 -> 23 G5 -> 33

Programming development:

The code is as follows:

- 280-AR version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
import time

# Initialize a MyCobot280 object
arm = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the actual situation

# Turn on the pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

# Stop the pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)

for i in range(2):
    pump_on()
    time.sleep(2)
    pump_off()
    time.sleep(2)
```

- 280-Pi version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
from pmycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables
import time
import RPi.GPIO as GPIO

# Initialize a MyCobot280 object
mc = MyCobot280(PI_PORT, PI_BAUD)

# Initialization
GPIO.setmode(GPIO.BCM)
# Pins 20/21 control the solenoid valve and the exhaust valve respectively
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# Turn on the suction pump
def pump_on():
    # Open the solenoid valve
    GPIO.output(20,0)

# Stop the suction pump
def pump_off():
    # Close the solenoid valve
    GPIO.output(20,1)
    time.sleep(0.05)
    # Open the vent valve
    GPIO.output(21,0)
    time.sleep(1)
    GPIO.output(21,1)
    time.sleep(0.05)

pump_off()
time.sleep(3)
pump_on()
time.sleep(3)
pump_off()
time.sleep(3)

GPIO.cleanup() # Release the pin channel
```

Save the file and close it, return to the command line terminal, and enter:

```
python pump_double.py
```

You can see that the suction pump opens after 3 seconds and closes after working for 3 seconds

Integrated suction pump

Applicable models: myCobot 280, myPalletizer 260, mechArm 270



Specifications

4.1 First-time self-check

| Name | myCobot integrated suction pump |
|------------------------------|---|
| Model | myCobot integrated suction pump |
| Material | ABS injection molding |
| Color | White |
| Size | Diameter 20mm |
| Number of suction cups | 1 |
| Suction weight | 50g |
| Power source equipment | Suction pump box |
| Service life | One year |
| Fixing method | Lego connector |
| Control interface | IO control |
| Use environment requirements | Normal temperature and pressure |
| Applicable equipment | ER myCobot 280 Series, ER myPalletizer 260 Series, ER mechArm 270 Series, ER myBuddy 280 Series |

Suction pump: Used for adsorbing objects

Introduction

- Suction pump, that is, vacuum adsorption pump, has one suction nozzle and one exhaust nozzle. It has the advantages of simple structure, small size, easy use, low noise, and good self-priming ability. By controlling the suction pump kit as the end effector of the robot arm, the function of adsorbing objects is performed.
- Suction pump accessories: power cord x1, DuPont line x10, one-input and two-output connection line x1, Lego technology parts x several

Working principle

- When sucking objects: the air pump starts to suck air and adsorb objects and then stops, and there will be no air leakage in a short time.
- When putting down objects: the electronic valve starts, the air release valve opens, and air enters the vacuum suction cup to separate from the sucked objects.

Applicable objects

- Paper/plastic sheets
- Flat and smooth objects
- Cards, etc.

Installation and use

Check whether the accessories package is complete: Lego connectors, Dupont wires, double-head suction pump



Suction pump installation

Structural installation:

Insert the Lego connector into the reserved socket on the suction pump:

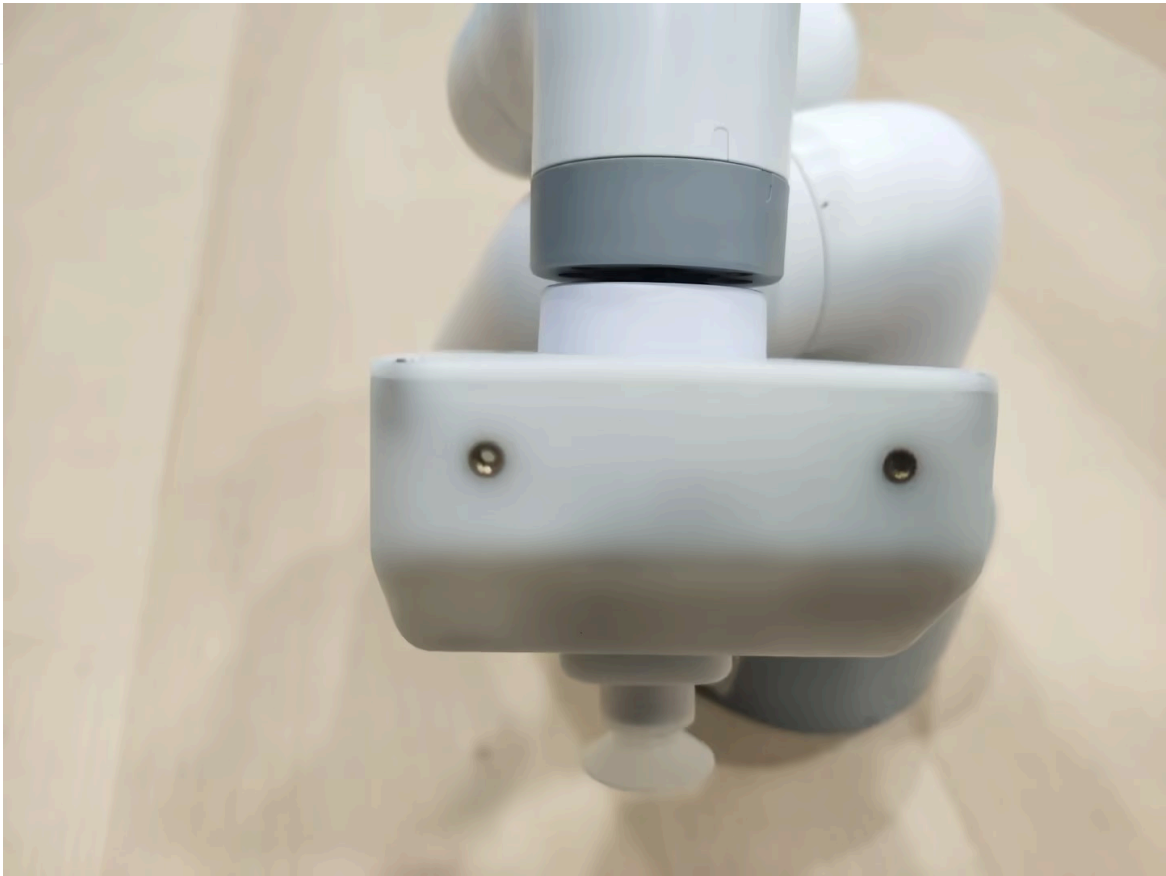
4.1 First-time self-check



1. Align the suction pump with the connector plugged in with the socket at the end of the robot arm and insert it:

>

4.1 First-time self-check

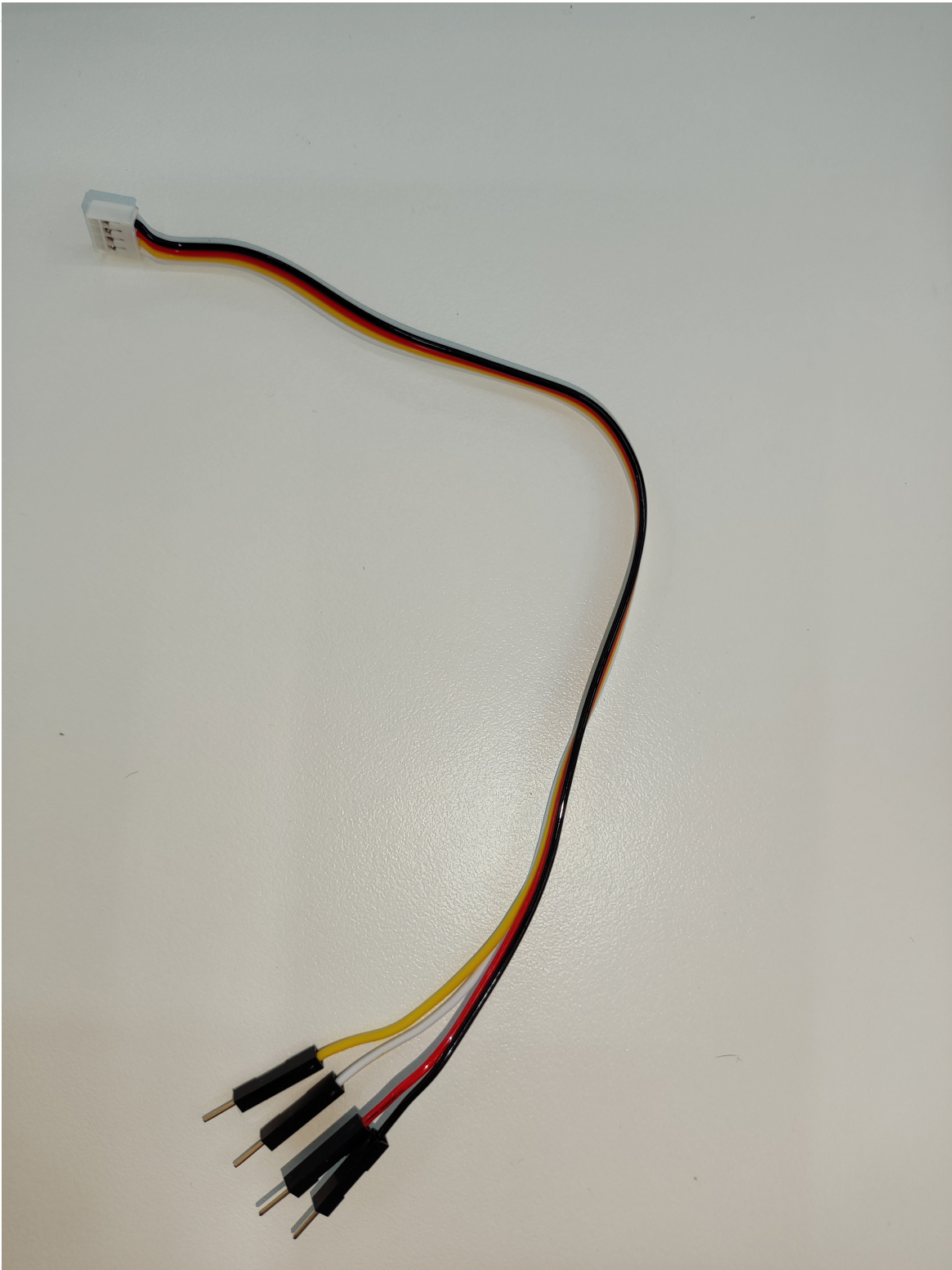


- Electrical connection:

Select the male-female DuPont wire, and insert the female end into the socket marked with pins on the suction pump box:

4.1 First-time self-check

Male-female DuPont wire:



Note the correspondence between the DuPont wire colors and pins in the figure:

4.1 First-time self-check



Insert the male end into the robot base pin according to the given correspondence:



The left side is the suction pump pin, and the right side is the robot arm pin GND -> GND 5V -> 5V G2 -> 21
G5 -> 20

4.1 First-time self-check

Programming development:

280-AR Version:

```
from pycobot import MyCobot280
import time

# Initialize a MyCobot280 object
arm = MyCobot280('COM3', 115200) # The baud rate is 115200 by default, and some boards are 1000000, please modify it according to the board

# Turn on the pump
def pump_on():
    arm.set_digital_output(33, 0)
    time.sleep(0.05)

# Stop the pump
def pump_off():
    arm.set_digital_output(33, 1)
    time.sleep(0.05)
    arm.set_digital_output(23, 0)
    time.sleep(1)
    arm.set_digital_output(23, 1)
    time.sleep(0.05)

for i in range(2):
    pump_on()
    time.sleep(2)
    pump_off()
    time.sleep(2)
```

- 280-Pi version:

4.1 First-time self-check

```
from pmycobot import MyCobot280
from pmycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables
import time
import RPi.GPIO as GPIO

# Initialize a MyCobot280 object
mc = MyCobot280(PI_PORT, PI_BAUD)

# Initialization
GPIO.setmode(GPIO.BCM)
# Pins 20/21 control the solenoid valve and the bleed valve respectively
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# Turn on the suction pump
def pump_on():
    # Open the solenoid valve
    GPIO.output(20,0)

# Stop the suction pump
def pump_off():
    # Close the solenoid valve
    GPIO.output(20,1)
    time.sleep(0.05)
    # Open the exhaust valve
    GPIO.output(21,0)
    time.sleep(1)
    GPIO.output(21,1)
    time.sleep(0.05)

pump_off()
time.sleep(3)
pump_on()
time.sleep(3)
pump_off()
time.sleep(3)
GPIO.cleanup() # Release the pin channel
```

myCobot Pen Holder

Applicable models: ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series

Product image



**Specifications:**

| Name | myCobotPro Pen Holder |
|--------------------------|--|
| Model | myCobot_penHolder_J6 |
| Material | Photosensitive resin (painted white) |
| Dimensions | 47.5 x 25.0 x 45.5 mm |
| Weight | Approx. 35g (excluding pen weight) |
| Pen tip clearance | ±1 mm |
| Service life | One year |
| Fixing method | Lego connector |
| Environment requirements | Normal temperature and pressure |
| Applicable equipment | Support ER myCobot 280 series ER myPalletizer 260 series ER mechArm 270 series ER myBuddy 280 series |

myCobot pen holder: Used when writing and drawing with a robotic arm

Introduction

4.1 First-time self-check

- Overall solid color design, supports 15mm large stroke extension and retraction, effectively reduces errors, and can be used for writing, drawing and other applications.
-

Applicable objects

- Whiteboard pen

Installation and use

- Installation

Insert the Lego connector into the holder hole:



4.1 First-time self-check

Insert the holder with the connector installed into the end of the robot arm



- Use Insert the pen into the round hole and tighten the four screws to fix it.

myCobot Phone Holder

Applicable models: ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series

Product image



Specifications:

| Name | myCobot Phone Holder |
|----------------------------|--|
| Model | myCobot_PhoneHolder_J6 |
| Material | ABS injection molding |
| Size | Diameter 34*10 |
| Color | White+Black |
| Clamping weight | 50g |
| Service life | Two years |
| Fixing method | LEGO connector |
| Environmental requirements | Normal temperature and pressure |
| Applicable equipment | ER myCobot 280 series ER mechArm 270 series ER myPalletizer 260 series |

myCobot mobile phone holder: Used to hold mobile phones or objects

Introduction

- Suitable for devices that require physical clamping, such as photography, and can hold a variety of mobile phones. It has a simple structure and is easy to install and disassemble.

Applicable objects

- Camera equipment
- Installation

4.1 First-time self-check

Insert the LEGO connector into the holder hole:



4.1 First-time self-check

Insert the holder with the connector installed into the end of the robot arm



- Use Pull the holder open, put the camera in, and let go. After confirming that the device is fixed, it can be used.

Dexterous Hand

Applicable models: myCobot 280, myPalletizer 260, mechArm 270

Product Image



Specifications:

| Name | mycobot Dexterous Claw |
|--------------------------|--|
| Model | Dexterous Hand |
| Material | 3D Printing |
| Size | 112×94×50mm |
| Color | White |
| Transmission Mode | Gear + Connecting Rod |
| Clamping Range | 20-45mm |
| Maximum Clamping Force | 100g |
| Fixing Mode | Screw Fixing |
| Environment Requirements | Normal Temperature and Pressure |
| Control Interface | Serial Control |
| Applicable Equipment | ER myCobot 280 Series, ER mechArm 270 Series, ER myPalletizer 260 Series |

Dexterous Hand: Used when gripping objects

Introduction

- The gripper is a robot component that can achieve functions similar to human hands. It has the advantages of a complex structure, firm gripping of objects, not easy to drop, and easy operation. The gripper kit includes gripper accessories and LEGO technology parts. The end effector of the robot arm is controlled by a programmable system to achieve functions such as object gripping and multi-point positioning.

Working principle

- Driven by a motor, the gripper's finger surface makes linear reciprocating motion to achieve opening or closing. The acceleration and deceleration of the electric gripper can be controlled, the impact on the workpiece can be minimized, the positioning point can be controlled, and the clamping can be controlled

Applicable objects

- Small cubes
- Small balls
- Long objects

Gripper installation:

Insert the Lego connector into the gripper hole:



4.1 First-time self-check



Electrical connection Insert the gripper with the connector installed into the end of the robot arm



Python programming control

- AR Version

```
from pmycobot import MyCobot280
import time

# Initialize a MyCobot280 object
mc = MyCobot280("COM3", 1000000)

mc.set_encoder(7,2048,40)#Open
time.sleep(2)
mc.set_encoder(7,2300,40)#Hold
time.sleep(2)
mc.set_encoder(7,2048,40)#Hold
```

- Pi version ``python from pmycobot import MyCobot280 from pmycobot import PI_PORT, PI_BAUD # When using the Raspberry Pi version of mycobot, you can reference these two variables to initialize MyCobot280 import time

Initialize a MyCobot280 object

```
mc = MyCobot280(PI_PORT, PI_BAUD) mc.set_encoder(7,2048,40)#Open time.sleep(2)
mc.set_encoder(7,2300,40)#Hold tight time.sleep(2) mc.set_encoder(7,2048,40)#Hold tight ``
```

myCobot camera module v2.0

Applicable models: myCobot 280, myPalletizer 260, mechArm 270

Product image



Specifications:

4.1 First-time self-check

| Name | myCobot camera module v2.0 |
|------------------------------|---|
| Model | myCobot_cameraHolder_J6 |
| Color | White (default) |
| Material | ABS injection molding |
| Size | 836416 |
| USB protocol | USB2.0 HS/FS |
| Lens focal length | Standard 1.7mm |
| Field of view | About 60° |
| Supported systems | Win7/8/10, Linux, MAC |
| Supported resolutions | 2592x1944, 2560x1440, 2048x1536, 1920x1080, 1280x72, 1024x768, 800x600, 640x480, 640x360, 352x288, 320x240, 176x144 |
| Service life | Two years |
| Fixing method | Lego connector |
| Environment requirements | Normal temperature and pressure |
| Applicable equipment support | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series, ER myBuddy 280 series |

Camera flange: Machine vision

Introduction

- USB high-definition camera can be used with suction pump, adaptive gripper, artificial intelligence kit, etc., to achieve precise positioning and calibration with eye in hand.

Installation and Use

4.1 First-time self-check

- Check if the accessories package is complete: Lego connector, camera module with USB cable



- Camera installation:

Structural installation:

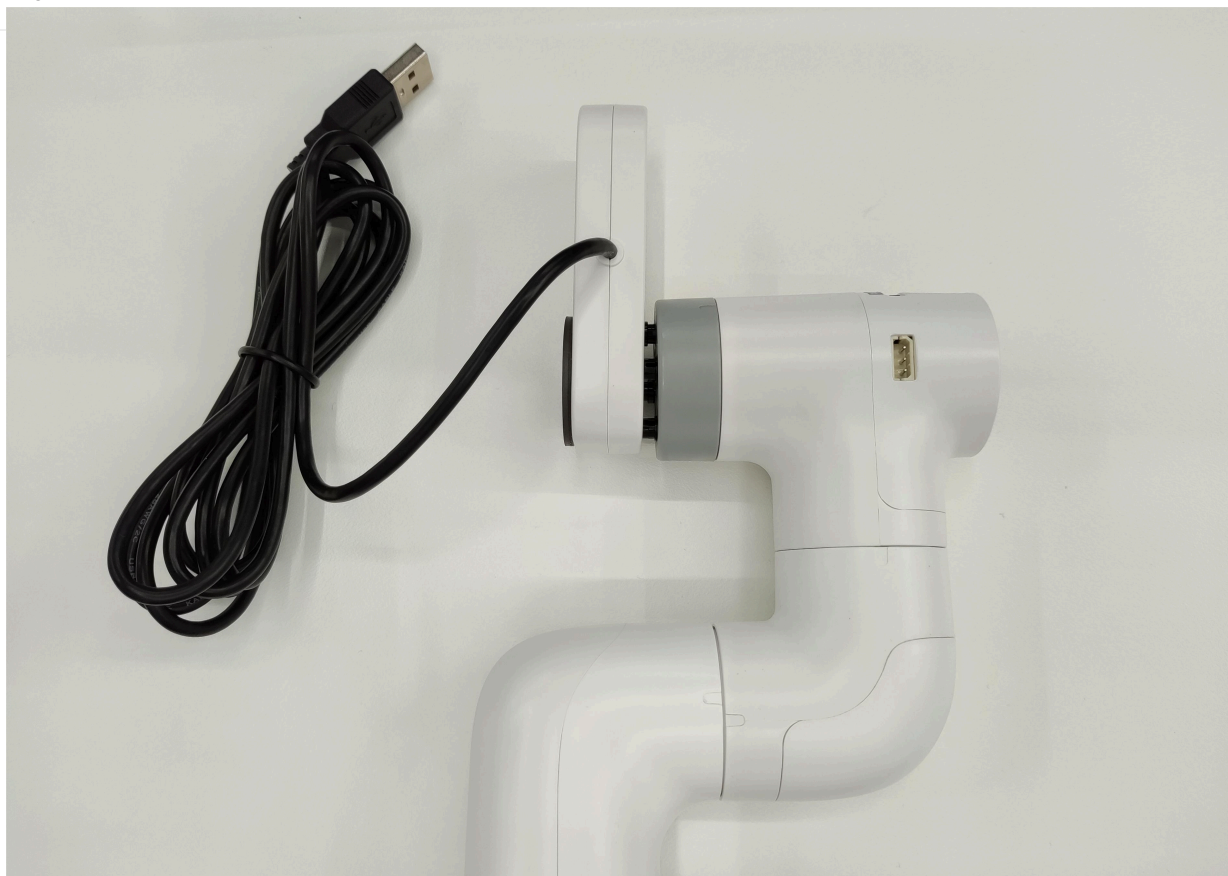
4.1 First-time self-check

Insert the Lego connector into the reserved socket of the camera module:



4.1 First-time self-check

Align the camera module with the connector into the socket at the end of the robot arm:



Electrical connection:

4.1 First-time self-check

Insert the USB cable into the USB port of the base:



Programming development:

The code is as follows:

```
python
import cv2
import numpy as np

cap = cv2.VideoCapture(0) # "0", determined by the camera device number queried

while(True):
    ret, frame = cap.read()

    # gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cv2.show('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    cap.release()
    cv2.destroyAllWindows()
```

Spring Bamboo Shoot Flange

Applicable models: ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series

Product image



Specifications:

| Name | myCobot Spring Bamboo Shoot Flange |
|----------------------------|--|
| Material | Nylon |
| Hardness | Fragile |
| Service life | Two years |
| Fixing method | Lego connector |
| Environmental requirements | Normal temperature and pressure |
| Applicable equipment | ER myCobot 280 series, ER myPalletizer 260 series, ER mechArm 270 series |

myCobot Spring Bamboo Shoot Flange: Used for clicking buttons

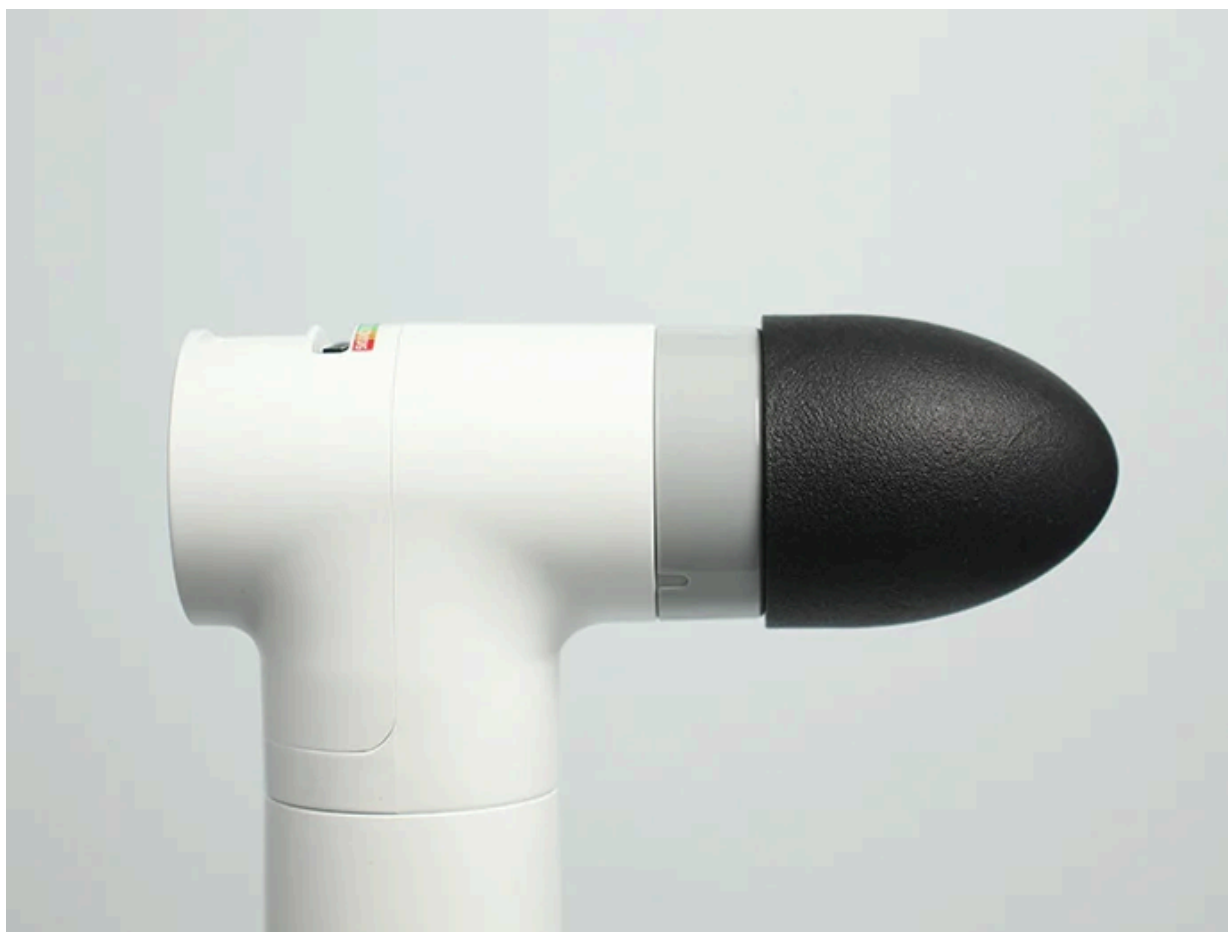
Introduction

- Suitable for devices with physical travel such as click buttons and keyboards. The overall solid color design, simple structure, easy to install and disassemble.

Applicable objects

4.1 First-time self-check

- Keyboard
 - Button
-



Acknowledgements

We would like to express our deep gratitude to all the people who have participated in the development, testing and improvement of the myCobot series of products (including myCobot 280 pi, myCobot 280 M5, myCobot 280 JN, myCobot 280 For Arduino and kits). Every detail polished and every feature innovative is inseparable from the hard work and dedication of the team behind it.

Special thanks:

R&D Team: Thank you for your innovative thinking and countless days and nights of hard work to transform complex technology into user-friendly products. **QA & Testing Team:** Your strict control of every detail ensures the reliability of our products and the ultimate experience of users. **Customer Support Team:** Thank you for providing professional support to our users to help them solve every problem during use. **Partners & Suppliers:** Your support and service are crucial to the success of the product. Thank you for your high-quality raw materials and components, and your attitude of being ready to support. **Investors and Advisors:** Without your trust and financial support, we would not be able to bring these innovations to the market. Your insights and guidance have always been our driving force.

User Thanks:

We are especially grateful to every user who has chosen and trusted the myCobot series of products. Your feedback and suggestions are the driving force for our continuous progress and improvement. We promise to continue to listen to your voice and continuously optimize our products and services.

Future Outlook:

We look forward to continuing to explore and progress on the road of robotics with all stakeholders. Let us work together to create more possibilities and bring greater convenience and innovation to the world.

[← Previous Chapter](#)