Table of Contents

Introduction 1.1

1 Product Introduction 1.2

- 1.1 Design Philosophy 1.2.1
- 1.2 Suitable Users 1.2.2
- 1.3 Application Scenario 1.2.3
- 1.4 Accessories Tools 1.2.4
  - 1.4.1 AdaptiveGripper 1.2.4.1
  - 1.4.2 ElectricGripper 1.2.4.2
  - 1.4.3 PneumaticGripper 1.2.4.3
  - 1.4.4 FlexibleGripper 1.2.4.4
  - 1.4.5 ModulSuctionCup 1.2.4.5
  - 1.4.6 CameraModulePro 1.2.4.6
  - 1.4.7 PenHolderPro 1.2.4.7
  - 1.4.8 PhoneHolderPro 1.2.4.8
  - 1.4.9 Single-ended suction pump 1.2.4.9
  - 1.4.10 Force control gripper 1.2.4.10

2 Product Feature 1.3

- 2.1 Functional Parameters 1.3.1
- 2.2 Controller Parameters 1.3.2
- 2.3 Structural Parameters 1.3.3
- 2.4 Electronic Parameters 1.3.4
- 2.5 Coordinate System 1.3.5

3 User Notes 1.4

- 3.1 Safety Instructions 1.4.1
- 3.2 Transport and Storage 1.4.2
- 3.3 Maintenance and Care 1.4.3
- 3.4 FAQs 1.4.4
  - 3.4.1 First-time self-check 1.4.4.1
  - 3.4.2 software 1.4.4.2
  - 3.4.3 hardware 1.4.4.3
  - 3.4.4 accessory 1.4.4.4
  - 3.4.5 other 1.4.4.5

4 First Install and Use 1.5

- 4.1 Product Standard List 1.5.1
- 4.2 Product Unboxing Guide 1.5.2
- 4.3 Power-on Test Guide 1.5.3

5 Basic Application 1.6

- 5.1 miniRoboflow 1.6.1

1 Drag & Play 1.6.1.1
2 Joint Calibration 1.6.1.2
3 Computer Connect 1.6.1.3
4 Robot Information 1.6.1.4
5.2 Firmware Description 1.6.2
1 Burn Firmware 1.6.2.1
5.3 Application Use 1.6.3
1 myblockly 1.6.3.1
1 FirstUse 1.6.3.1.1
2 install_uninstall 1.6.3.1.2
3 Interface Description 1.6.3.1.3
4 Control RGB 1.6.3.1.4
5 Control Robotic Arm Back 1.6.3.1.5
6 Control Single Joint 1.6.3.1.6
7 Control Singles Joint 1.6.3.1.7
8 Gripper Use 1.6.3.1.8
9 API 1.6.3.1.9
10 Q&A 1.6.3.1.10
2 myStudio 1.6.3.2
1 setup 1.6.3.2.1
2 install 1.6.3.2.2
3 flash 1.6.3.2.3
4 other function 1.6.3.2.4
6 SDK Development 1.7
6.1 Python 1.7.1
1 Environment Building 1.7.1.1
2 Introduction to API 1.7.1.2
3 TCP/IP Control 1.7.1.3
4 Videos and Codes for Display 1.7.1.4
6.2 Robot Operating System 1 (ROS1) 1.7.2
1 Environment Building 1.7.2.1
2 ROS basics 1.7.2.2
3 Rviz use 1.7.2.3
4 Basic function case 1.7.2.4
5 Gazebo Basic function case 1.7.2.5
6.3 Robot Operating System 2 (ROS2) 1.7.3
1 Environment Building 1.7.3.1
2 ROS2 basics 1.7.3.2
3 Rviz2 use 1.7.3.3
4 Basic function case 1.7.3.4

6.4 C Sharp (C#) 1.7.4
1 Environmental construction 1.7.4.1
2 Mechanical control 1.7.4.2
3 myCobot API 1.7.4.3
4 Use Cases 1.7.4.4
6.5 C plus plus (C++) 1.7.5
1 CPlus Download 1.7.5.1
2 build 1.7.5.2
3 API 1.7.5.3
4 example 1.7.5.4
6.6 Arduino 1.7.6
1 arduino Download 1.7.6.1
2 API 1.7.6.2
3 example 1.7.6.3
6.7 Communication Message Protocol 1.7.7
7 Examples of Robots Using 1.8
1 PLC-based robotic arm control 1.8.1
2 Robot gripper carrying wood block example 1.8.2
3 Robot suction pump carrying wood block example 1.8.3
4 Examples of Remote Control Robots 1.8.4
5 320M5 deepseek cases 1.8.5
8 Documents Download 2.1
8.1 Gitbook Download 2.1.1
8.2 Product Brochure 2.1.2
8.3 Software and Source Code 2.1.3
8.4 System Information 2.1.4
8.5 Publicity Material 2.1.5
9 About Us 2.2
9.1 Elephant Robotics 2.2.1
9.2 Contact us 2.2.2

myCobot 320

> Six-axis collaborative robots for user-developed autonomous programming

Product Diagram

- M5:

Product Catalogue



- Pi:

Product Catalogue

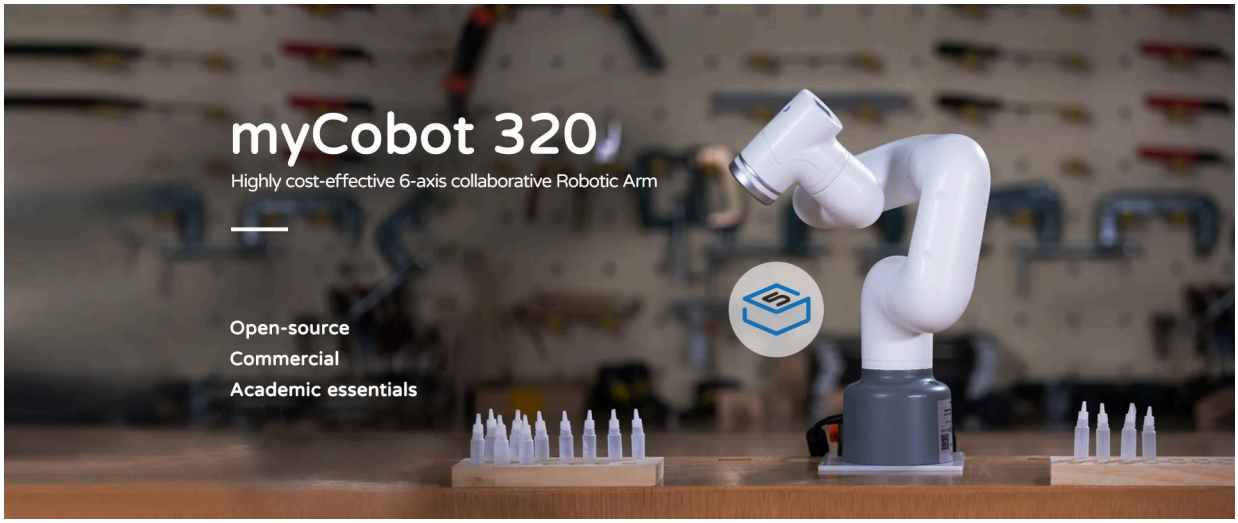


Product Introduction

Developed by Elephant Robotics, the myCobot 320 Robotic Arm is a collaborative robot designed for education, research and light industrial automation. Renowned for its compact design and powerful features, this robot is dedicated to providing endless possibilities for innovation and learning. It has a maximum arm span of 350 mm, a maximum payload of 1 kg, and supports programming languages of varying difficulty, making it suitable for users of all skill levels.

The arm supports Python and has hardware interfaces such as IO and USB, making it easy to connect to a variety of sensors and actuators. It also provides rich open source libraries and APIs to simplify the development process, and is compatible with Windows, Linux, and MacOS, making it suitable for a variety of development environments and encouraging users to participate in extended development.

myCobot 320



> **The effective working radius of the robot arm is 350 mm.**

Product Introduction

The myCobot 320 Robot Arm, developed by Elephant Robotics, is a collaborative robot designed specifically for education, research, and light industrial automation. This robot is renowned for its compact design and powerful capabilities, committed to providing unlimited possibilities for innovation and learning. It features a maximum arm reach of 350mm and a maximum payload of 1kg, supporting programming languages of varying difficulty levels, suitable for users of all skill levels.

Supported Extension Development

The myCobot 320 encourages users to engage in extension development to fully exploit its potential:

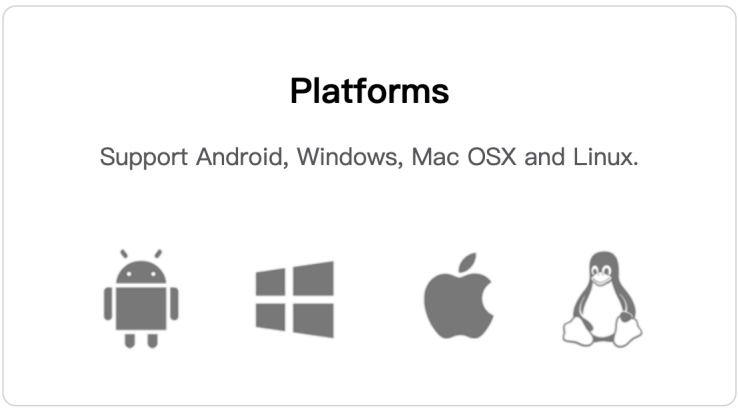
- **Programming Languages**: Supports Python, C++, among others, offering a diverse selection of programming options.



- **Hardware Interfaces**: Includes IO, USB, etc., facilitating the connection to a variety of sensors and actuators.

- **Software Libraries**: Provides a rich set of open-source libraries and APIs to streamline the development process.

- **System Compatibility**: Compatible with Windows, Linux, MacOS, accommodating a wide range of development environments.



Documentation Learning Content

By reading the official documentation, you will gain:

- **Basic Setup**: Guidelines for installation, configuration, and initial operation.
- **Programming Guidance**: How to control and program the myCobot 320 M5 using various languages and platforms.
- **Custom Development**: How to utilize extension interfaces and software libraries for custom application development.
- **Success Stories**: Insights into the application of myCobot 320 M5 across different domains.
- **Maintenance and Troubleshooting**: Tips for maintenance and solutions to common issues.

Address for Purchase

If you are interested in purchasing the device, please click the link below.

- Taobao: https://shop504055678.taobao.com
- shopify: https://shop.elephantrobotics.com/
- AliExpress：https://elephantrobotics.aliexpress.com/store/1101941423

Chapter Summary

Now that you have a grasp of the product's fundamental features and applications, let's delve deeper into obtaining more detailed information together. The next section of this manual will guide you to various sub-chapters, enabling you to gain a more comprehensive understanding of our product's design philosophy, target user demographics, recommended usage scenarios, as well as the supported accessories and tools.

Please feel free to select the following sections based on your interests and requirements:

1.1-Design Philosophy

In this section, we will introduce the product's design principles and philosophy to help you better comprehend why the product possesses specific features and advantages.

1.2-Suitable Users

Understand which user groups are most suitable for using this product and how it caters to the diverse needs of different users.

1.3-Application Scenario

In this part, we will elucidate the product's optimal application methods in various scenarios, enabling you to fully harness its functionality.

1.4-Accessories and Tools

Examine the supported accessories and tools to ensure that you can make the most of the product's full potential.

Please click on the respective links based on your interests to access more detailed information. If you have any questions or require further assistance, please do not hesitate to contact our customer support team. We are committed to providing you with support and guidance. Thank you for choosing our product, and we look forward to delivering an outstanding user experience for you!

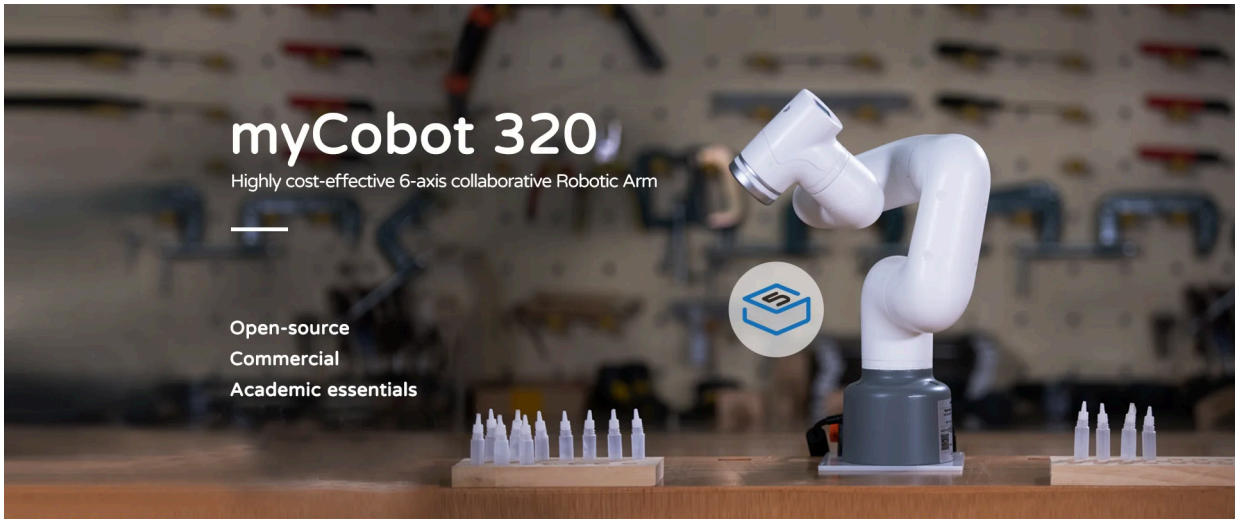
Words of Thanks

We greatly appreciate you taking the time to read the myCobot 320 user manual. We hope this document helps you to better understand and effectively use this robot, thereby inspiring your creativity. Should you have any questions or require further assistance, please do not hesitate to contact our customer support team. We look forward to seeing the innovative projects you accomplish with myCobot 320 and welcome you to our rapidly growing community of developers.

If you have already read all the content in this chapter, please proceed to the next chapter.

Next Chapter →

Design Philosophy



Design Intent

The myCobot 320 is designed for educational research, commercial presentations, and personal learning, aiming to provide a multifunctional robot suitable for both teaching and research, as well as personal interest exploration. In the field of educational research, it aims to be a practical teaching tool, helping students and researchers understand and apply robotics technology. For commercial presentations, the myCobot 320 offers an attractive way to showcase technological products or concepts. For individual learners, it provides an affordable and feature-rich platform for learning programming and robotics technology.

Application Directions

- **Educational Research**: The myCobot 320 provides educational institutions and researchers with an efficient and cost-effective teaching and research tool, allowing students and researchers to gain a deep understanding of robotics technology through hands-on learning and experimentation.
- **Commercial Presentations**: For businesses, the myCobot 320 can serve as an innovative presentation tool, attracting audiences and potential customers with robot demonstrations, showcasing the company's technological strength and innovative concepts.
- **Personal Learning**: For individual enthusiasts, the myCobot 320 offers an accessible platform to begin their journey in robotics technology, whether in programming, mechanical design, or the application of artificial intelligence.

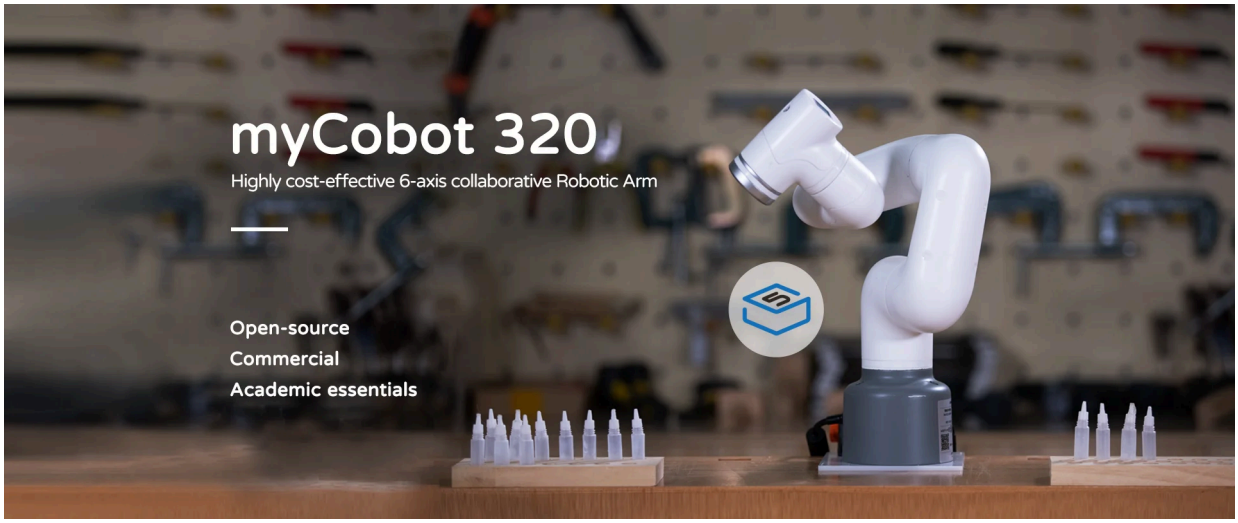
Industry Contribution

- **Promoting STEM Education**: Through its application in educational research, the myCobot 320 encourages the development of STEM (Science, Technology, Engineering, and Mathematics) education, stimulating students' interest in technology and innovation.
- **Facilitating Technology Adoption**: With its affordable price and user-friendly features, the myCobot 320 lowers the barrier to technology, enabling more people to access and utilize advanced robotics technology, thus promoting the widespread adoption of technology and improving public technological literacy.

- **Inspiring Innovation and Personal Development**: By providing an open and flexible learning platform for individual learners and enthusiasts, the myCobot 320 fosters interdisciplinary learning and innovation, cultivating talent for the future development of robotics technology and related fields.

← Previous Page | Next Page →

Suitable Users & Application



Suitable User Groups

Teachers and Students in Education: Given the potential of myCobot 320 in educational research, it is particularly suited for STEM courses and robotics projects in secondary and higher education institutions. Teachers can use myCobot 320 to design practical courses, while students can learn programming, robotics control theory, and the basics of artificial intelligence through it.

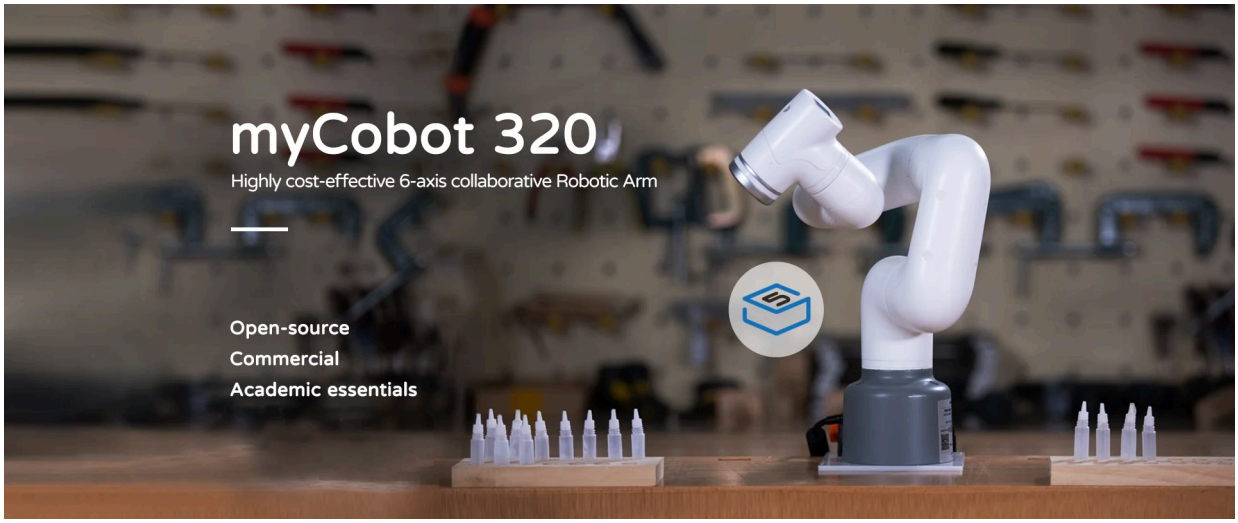
Researchers: Researchers can utilize myCobot 320 for prototype development and testing in experimental studies in robotics technology, artificial intelligence, and automation. Its flexibility and expandability make it an ideal choice for research laboratories.

Professionals in Commercial Presentations and Marketing: Businesses can use myCobot 320 as a presentation tool at exhibitions or events to effectively attract the audience's attention and showcase the company's technological strength and innovative products.

Individual Enthusiasts and Self-Learners: For individuals interested in robotics technology, regardless of their technical background, myCobot 320 provides an accessible and affordably priced platform for learning and exploring programming, robotics technology, and AI applications.

← Previous Page | Next Page →

Application Scenario



Recommended Application Directions

In Education: Use myCobot 320 for programming instruction, robotics design and manufacturing, and interdisciplinary scientific research projects to stimulate students' innovative thinking and problem-solving skills.

In Research: Use myCobot 320 as a research tool for cutting-edge studies in robotics technology, artificial intelligence, and automation, performing algorithm testing, data collection, and model validation.

In Commercial Presentations: Program myCobot 320 to perform eye-catching tasks or simulate actual work processes to showcase technological innovations and enhance brand influence.

For Personal Learning and Entertainment: Individual users can learn basic programming, robotics control, and even advanced concepts in machine learning and artificial intelligence with myCobot 320, also using it as an interesting personal project or hobby.

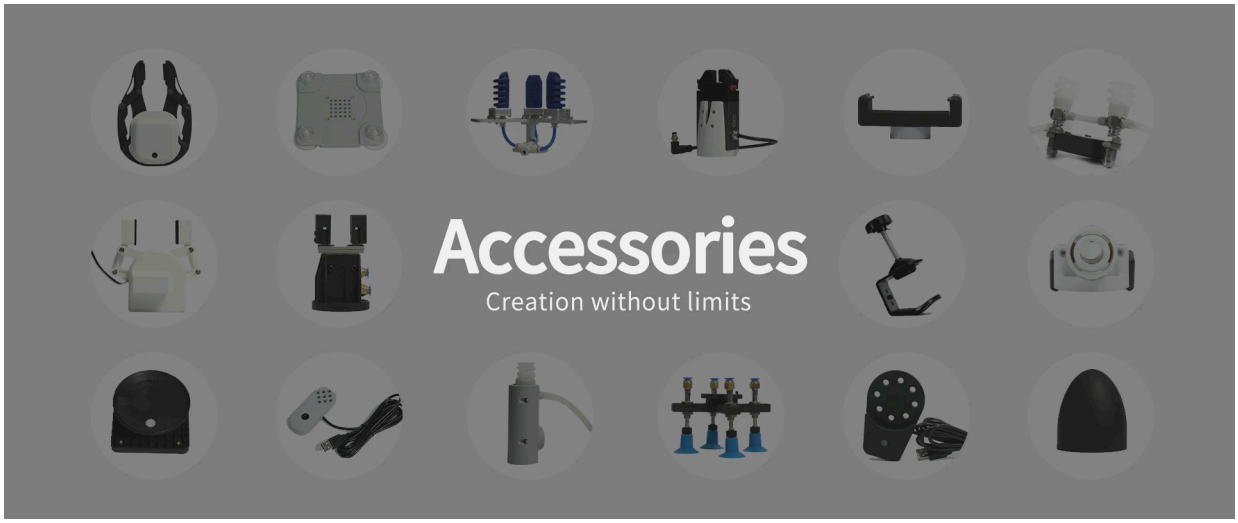
To facilitate customer reference, we have provided the following detailed scenario table, covering some common application scenarios. Please note that this does not mean myCobot 320 is limited to these applications; you can use it for any other applicable scenarios:

User Group	Application Scenarios	Goals and Benefits
Teachers and Students in Education	- STEM Education - Robotics Projects - Interdisciplinary Research Projects	- Increase students' interest in technology - Enhance hands-on and problem-solving skills - Promote innovative thinking and teamwork
Researchers	- Prototype Development - Experimental Research - Algorithm Testing and Validation	- Accelerate research progress - Bridge theory with practice - Advance technological innovation
Professionals in Commercial Presentations and Marketing	- Exhibition Display - Technical Demonstrations - Brand Promotion	- Attract potential customers and investors - Showcase the company's technological strength and innovative products - Enhance brand influence

User Group	Application Scenarios	Goals and Benefits
Individual Enthusiasts and Self-Learners	- Programming Learning - Exploration of Robotics Technology - Basics of Artificial Intelligence	- Improve personal skills and knowledge - Achieve self-education and development - Explore the potential of robotics technology and AI

← Previous Page | Next Page →

Accessories & Tools



Design Concept

In the real world, different accessories can enhance a robot's capabilities in various ways. For instance, attachments like grippers, sensors, and tools can assist robots in performing a wide range of tasks, thereby increasing their versatility and flexibility.

Elephant Robotics is dedicated to enabling everyone to easily use robots alongside these accessories, liberating users from the complexity of selecting the right accessories and enabling a swift start to robot utilization.

Types of Accessories

To meet the demands of customers in various scenarios, we have designed various types of accessories, including grippers, suction cups, camera modules, and other gripping devices, to enable users to directly select the suitable end-effectors.

Gripper

- Adaptive Gripper

  The upgrade adaptive gripper provides greater clamping force, and supports multiple programming environments, suitable for various industrial robotic arms.

- Electric Parallel Gripper

  The compact structure and multiple connecting holes make the gripper meet different installation conditions. It supports IO and serial port control, suitable for various industrial robotic arms.

- Air Parallel Grippers

Driven by gas and working with solenoid valve, it is convenient to be controlled and the fingertip of the gripper can be exchanged for users' secondary development.

- Air Flexible Gripper

The fingertips are made of rubber and rely on air pressure to deform for grasping objects. Pneumatic grippers have a wide range of applications, and they are favored for their softness, adaptability, and efficiency. These advantages make them powerful tools in automation and robotics applications, capable of effectively handling various types of objects and tasks.

Pump Cup

- Module Suction Cup

By operating on the principle of vacuum suction, it can be employed on flat and regular surfaces. Featuring a modular design, users can freely determine the number of suction cups, making it convenient for user development and usage.

Camera

- Camera Module

It provides standard HD camera and USB standard interface. USB interface can be used with a variety of PC equipment. The international standard installation interface is used to fix the camera on robot. It can be used in machine vision, image recognition and other applications.

Holder

- Pro Pen Holder

It supports up and down 15mm large stroke expansion, effectively increase the stability. It can be used in writing, painting and other applications.

- Pro Phone Holder It can hold a variety of mobile phones with simple structure and easy installation and disassembly.

← Previous Page | Next Chapter →

myCobot Pro Adaptive Gripper

> **Compatible models:** myCobot 320, myCobot Pro 630







Specifications

name	myCobotPro Adaptive Gripper Black and White
Material	Photosensitive resin + nylon
process technology	3D printing
clamping rangeclamp size	0-90mm
clamp force	1000 grams
Repeatability precision	0.5 mm
service life lifetime	1 year
drive mode drive	electric
Transmission modetransmission	gear+connecting rod
size	158x105x55mm
weightweight	350 grams
Fixed method fixed	screw fixed
Use environment requirements	Temperature and pressure
control interface control	Serial port/IO control
Applicable equipment	ER myCobot 320 series, ER myCobot Pro 600

Use for Gripping Objects

Introduction

- A gripper is a robotic component that can function like a human hand. It has the advantages of complex structure, firm grasping of objects, not easy to drop, and easy operation.
- The gripper kit includes gripper connecting wires and flanges, and controls the end effector of the robotic arm through a programmable system to realize functions such as object grabbing and multi-point positioning. Gripper can be used in all development environments, such as ROS, Arduino, Roboflow, etc.

working principle

- Driven by a motor, the finger surface of the gripper makes a linear reciprocating motion to realize the opening or closing action. The acceleration and deceleration of the electric gripper is controllable, the impact on the workpiece can be minimized, the positioning point is controllable, and the clamping is controllable .

Applicable object

- small cube
- small ball
- long object

Mall link:

- Taobao
- shopify

How to use

1. Installing



If the video fails to load, please click the link below to view the video. Installing Vidio

Installation and use

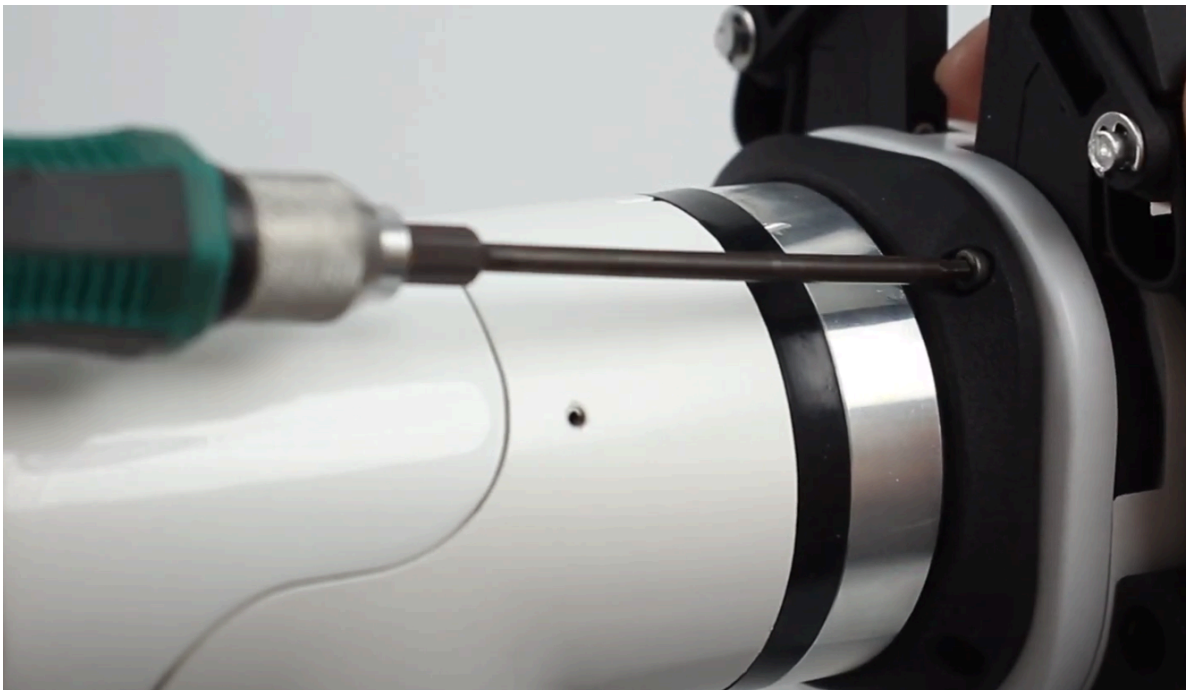
- gripper mounting：

Structural installation：

1. Align the spacer with the hole at the end of the arm and tighten with the screws：



2. Align the screw holes in the gripper with the holes around the gasket and tighten with the fine screws.：



- electrical connection：

> Take care to do this with the robotic arm powered off.

i. Align the m8 cable with the connector of the robot arm, note that the connector has a notch and the connecting cable has a corresponding protrusion, confirm the direction and then insert it and tighten it:



ii. Insert the gripper control connector, again paying attention to the orientation of the notch:



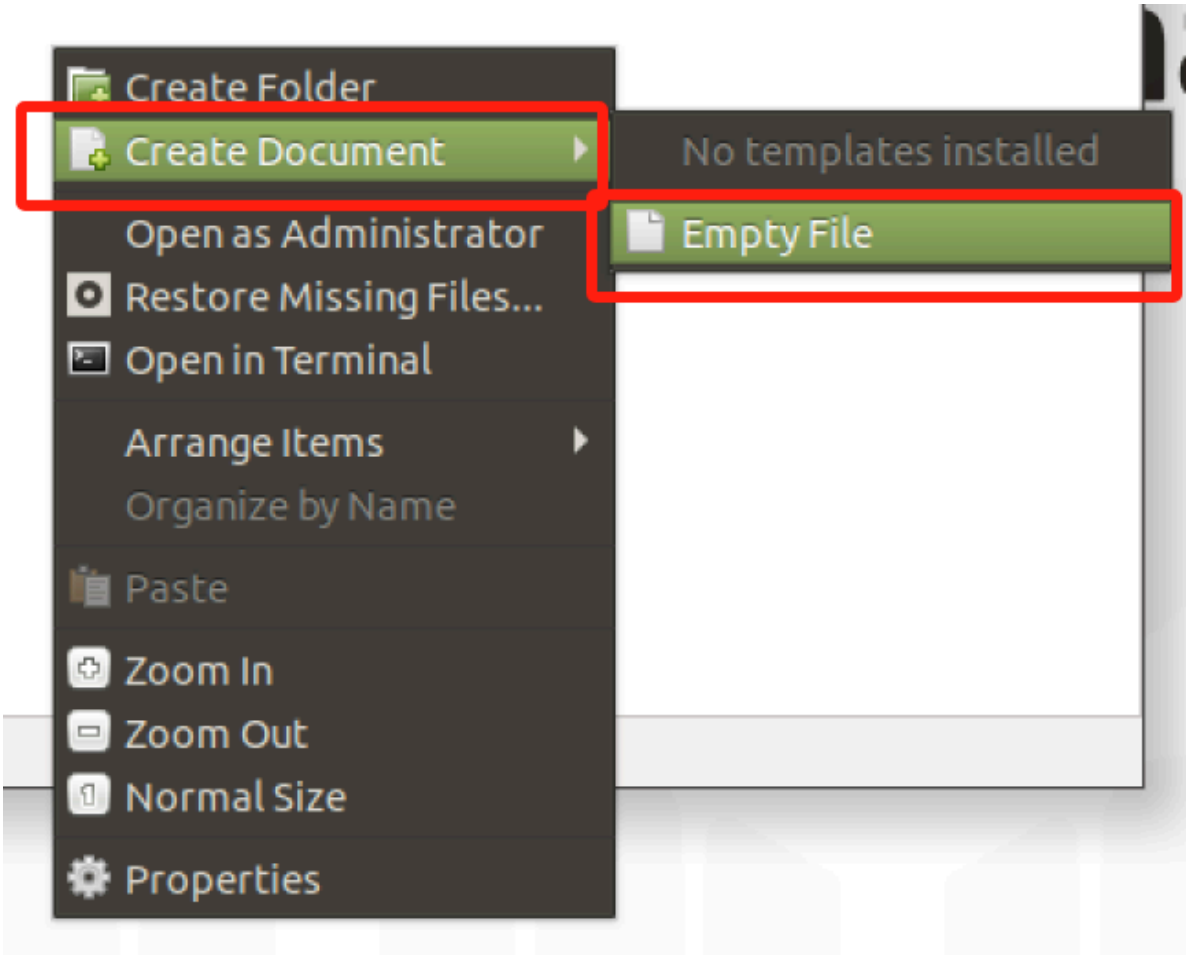
myCobot Pro 320 Instructions for use

Programming development (python)：

> Programming and development of the gripper using python: python environment download

1. Create a new python file:

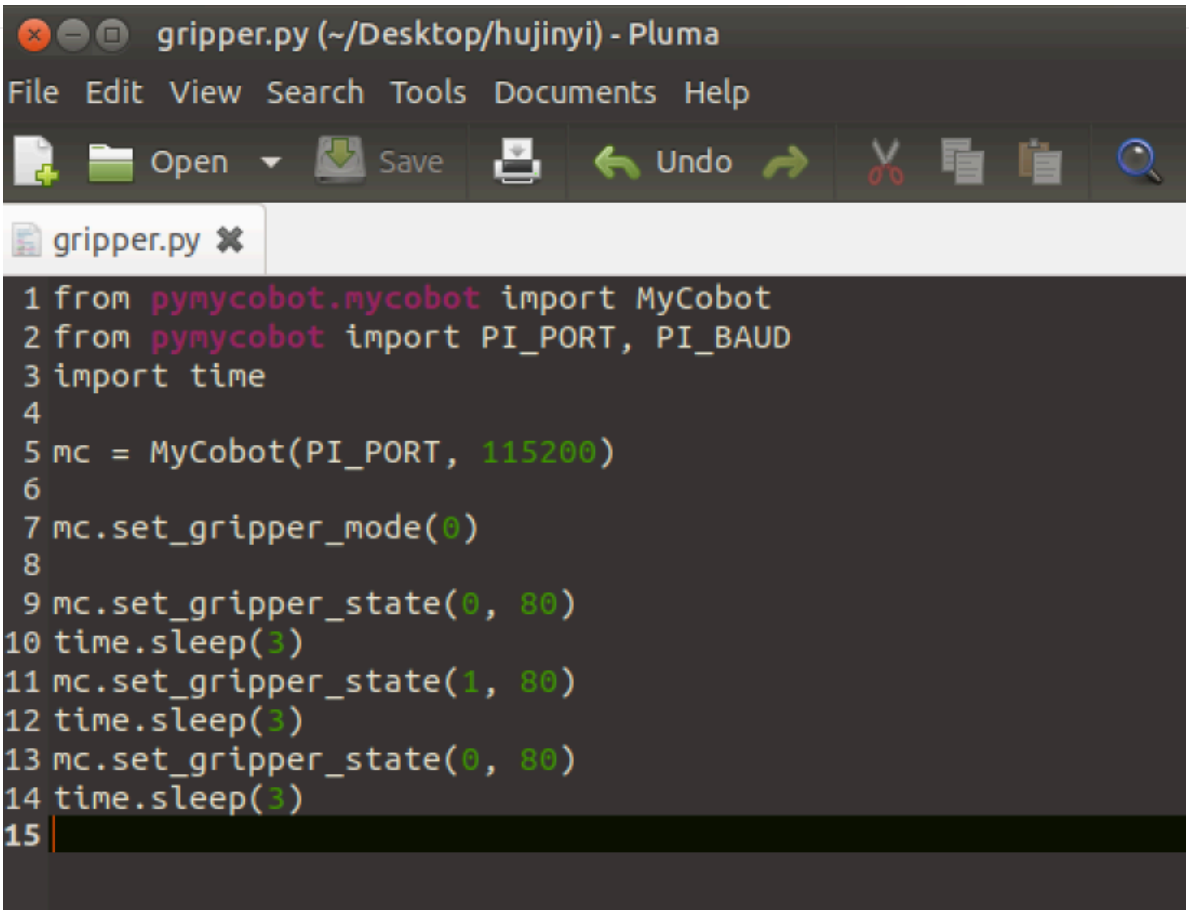
Right click on the desired file path to create a new python file:



> The file name can be changed as needed



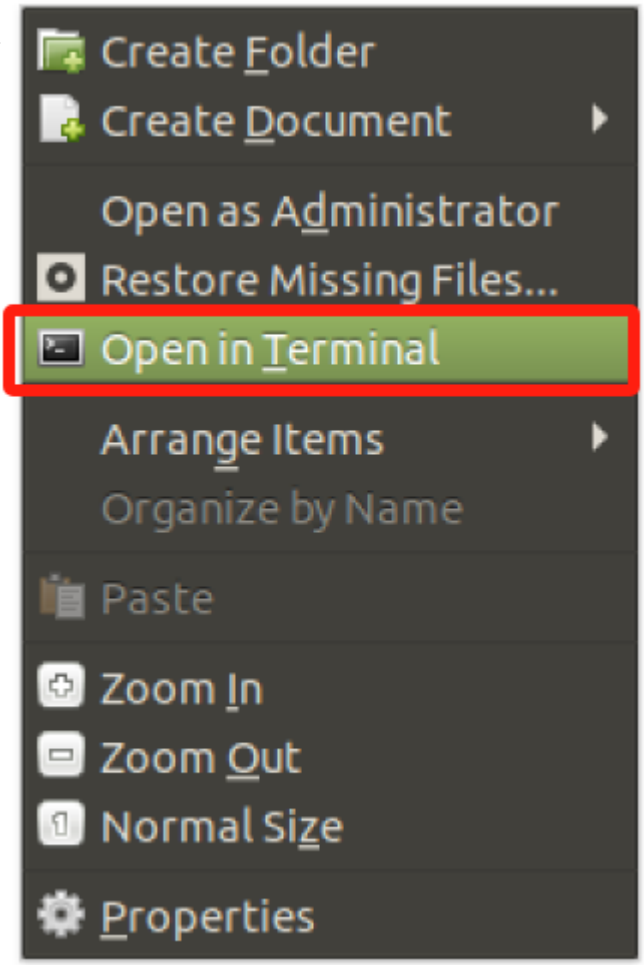
2. Perform function programming:



> The code is as follows:

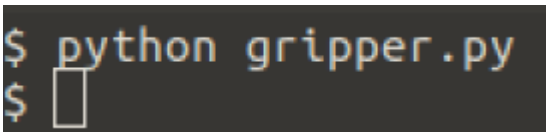
```
from pymycobot.mycobot import MyCobot
import time


# Initialise a MyCobot object
mc = MyCobot("COM3", 115200)


# Setting the gripper to 485 mode
mc.set_gripper_mode(0)
# Controls gripper open-close-open:
# Using the gripper status interface 0 is open, 1 is closed
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)


# For more information on using the interface, see the python API.
```

3. Save the file and close it, right-click on an empty space in the folder to open a command line terminal



Input:

```
python gripper.py
```



> You can see the gripper open-close-open

Programming Development (myblockly):

> Programming and development of the gripper using myblockly: myblockly download

- **Port Passthrough Mode Method 1:**

  1. After confirming that the structural and electrical connections are complete, start the arm and open the myblockly software when the graphical interface appears.

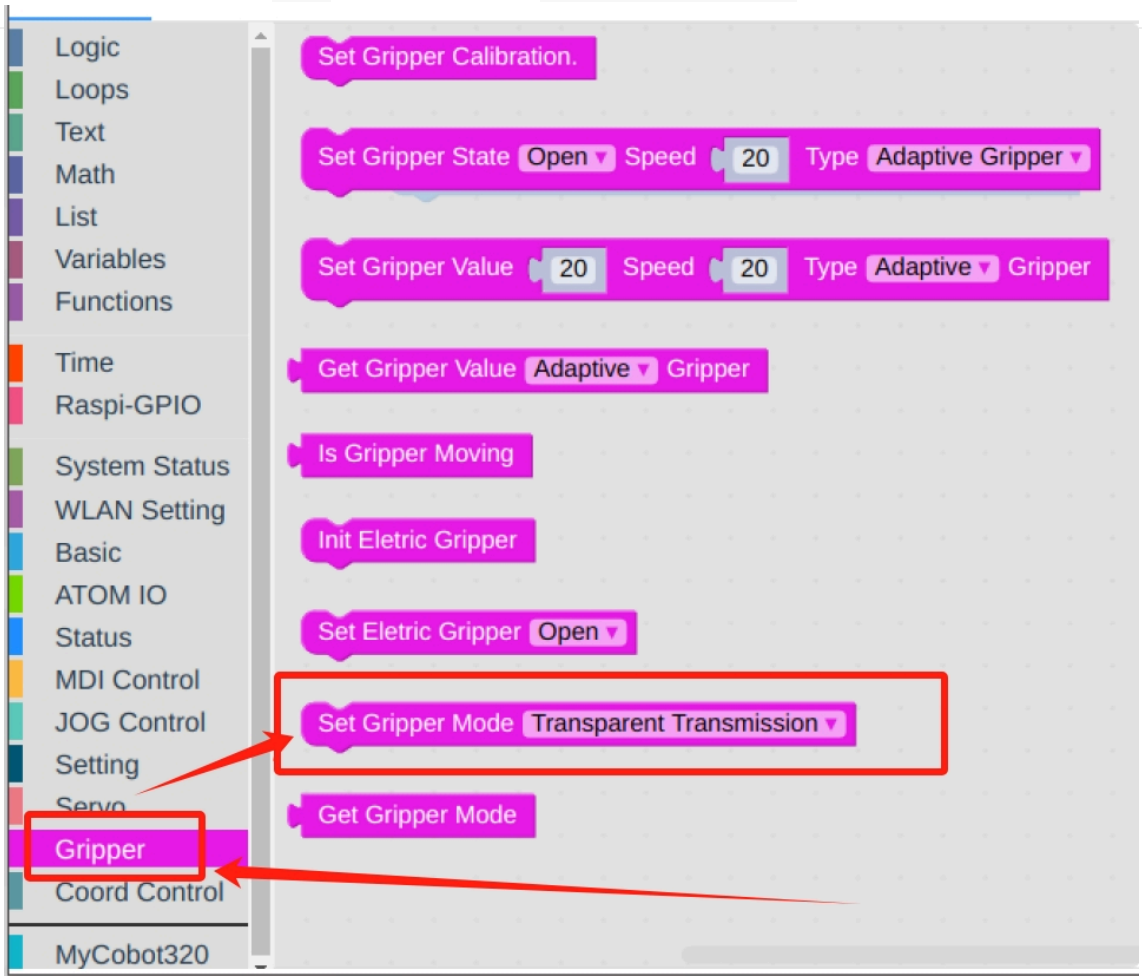


2. Modify the baud rate to 115200.

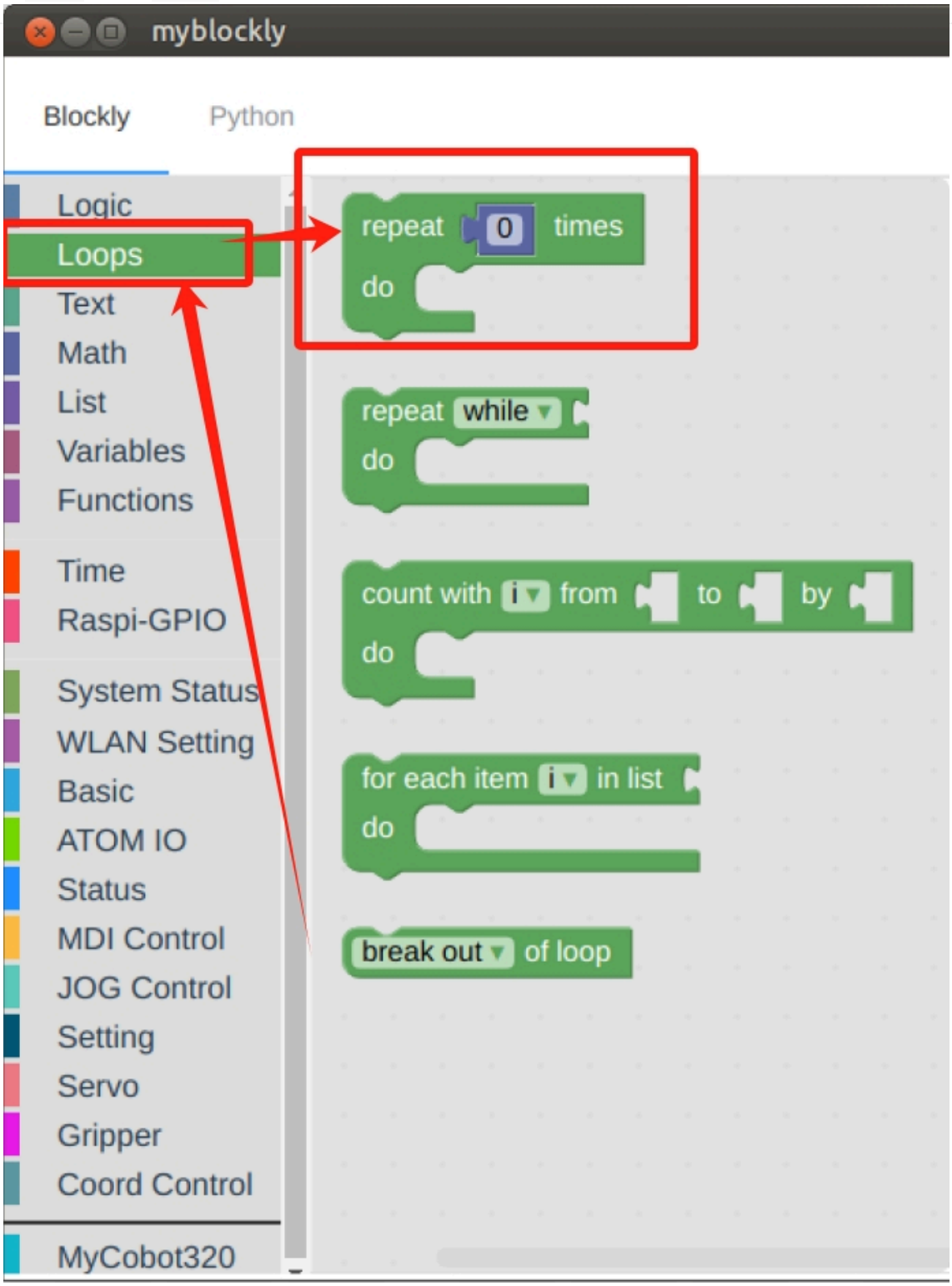


3. Find `Gripper` in the list on the left and select the `Set Gripper Mode` module.

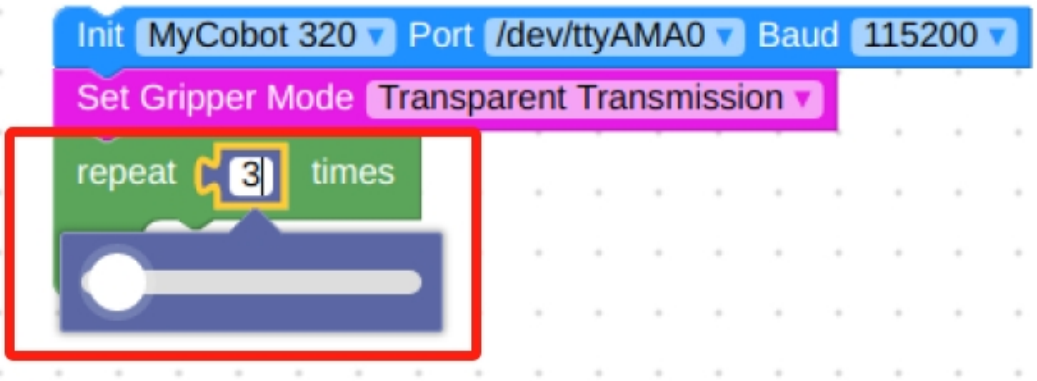
4. Drag and drop under the `init` module and select `Transparent Transmission`.



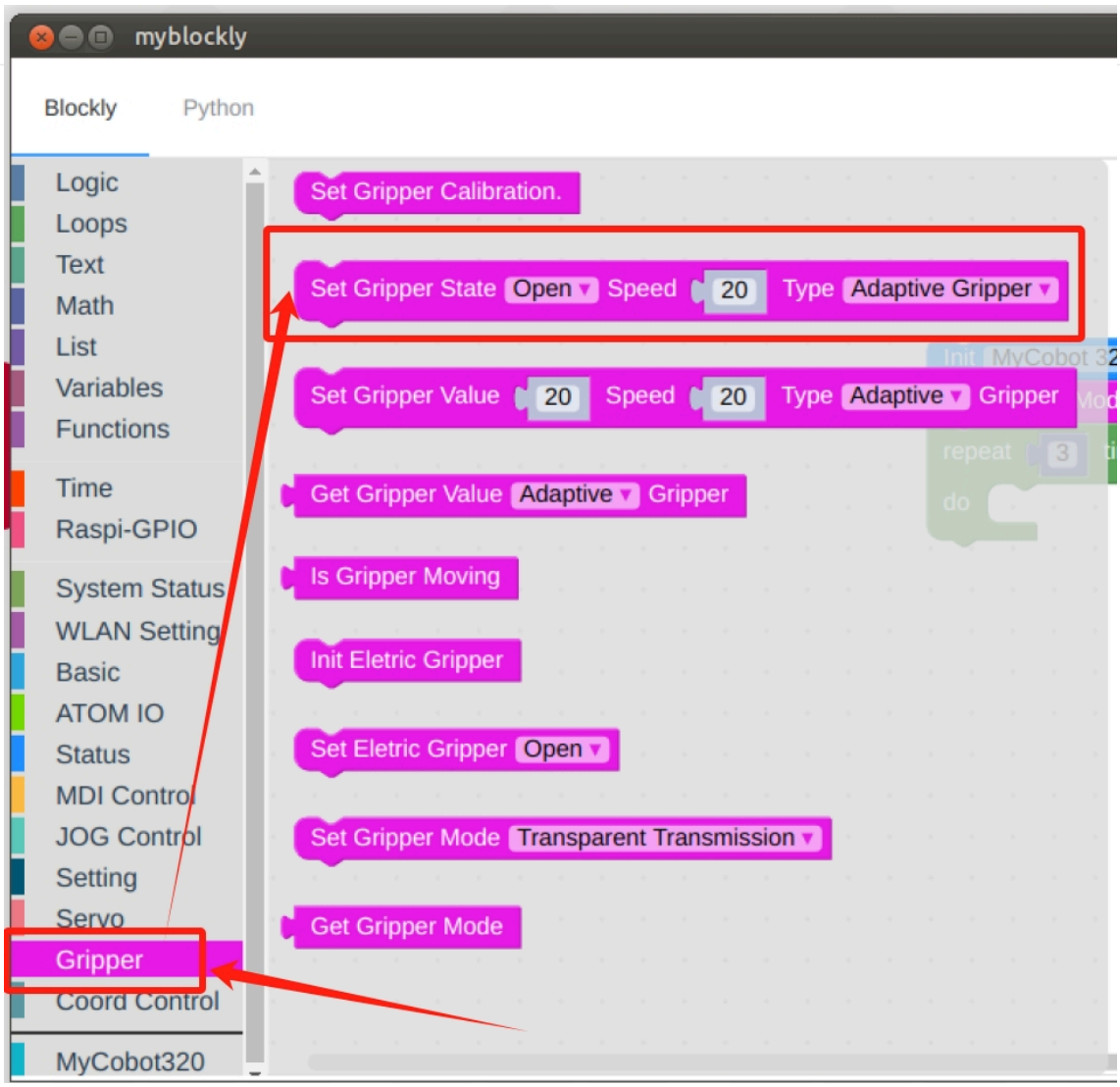
5. In `Loops` , select `repeat` module.

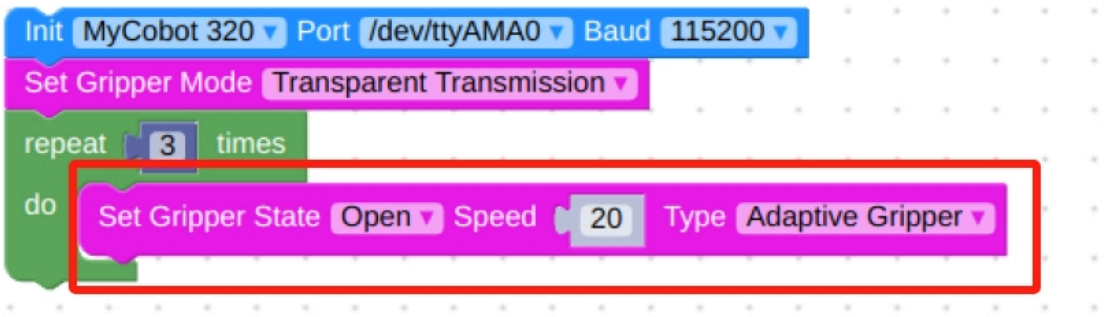


6. Set the times to `3 times`.

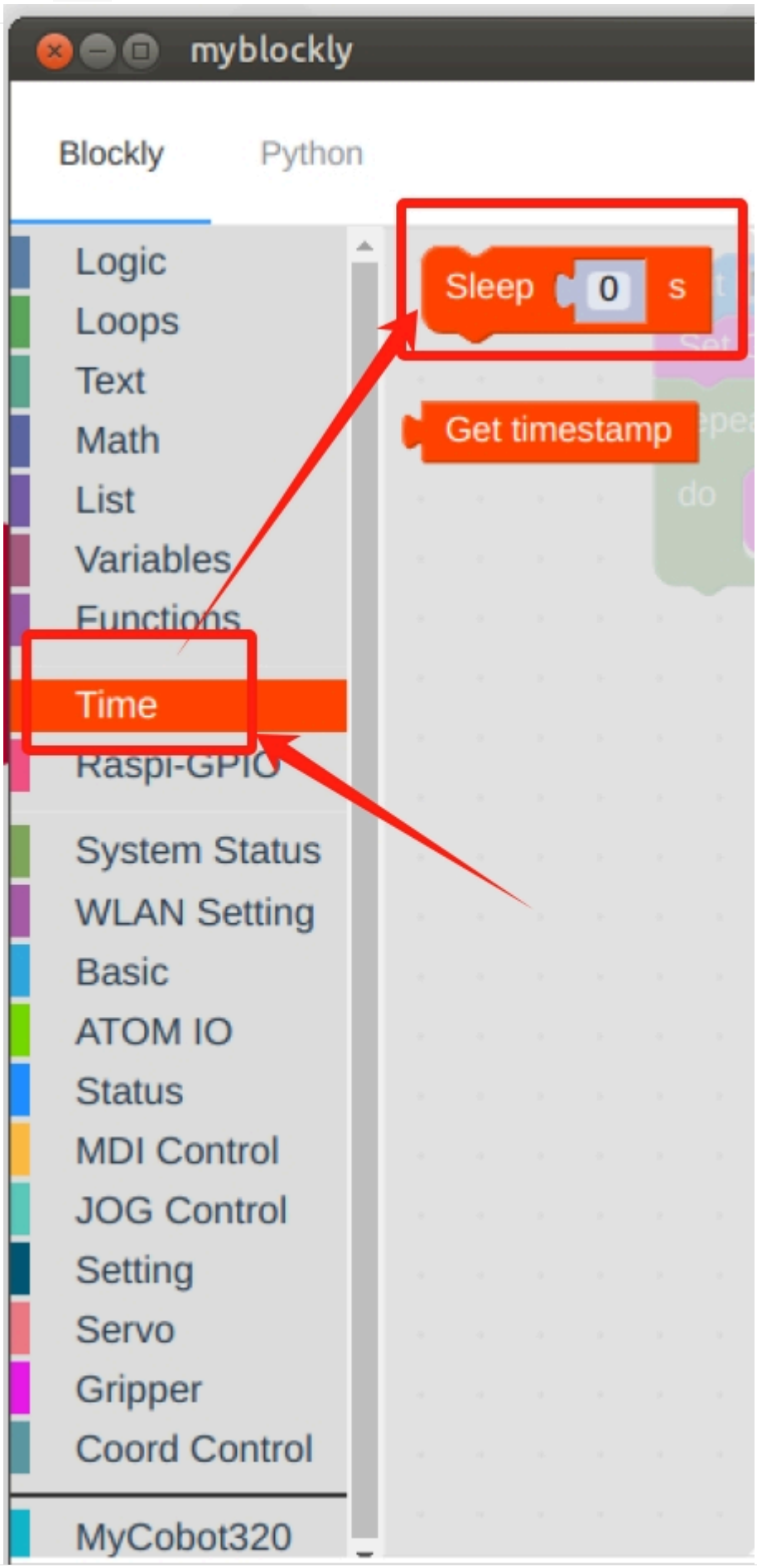


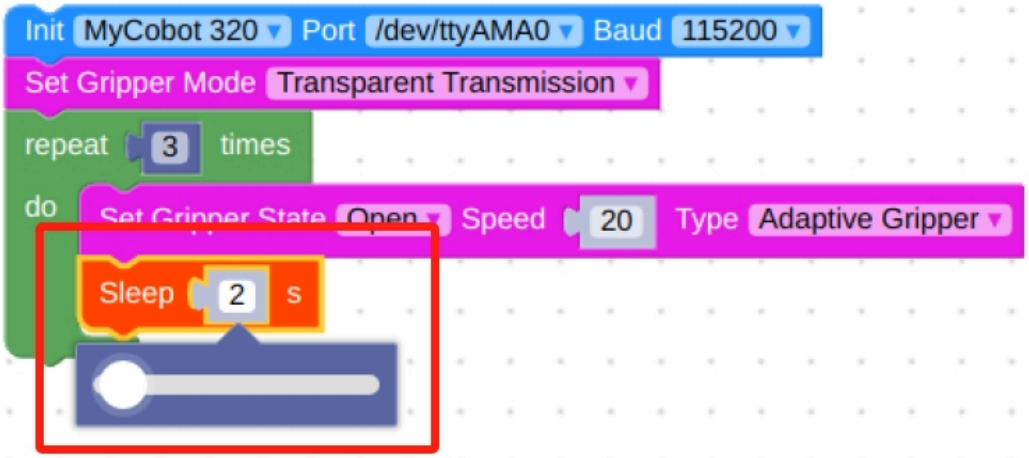
7. In `Gripper`, select the `Set Gripper State` module, put it in the `repeat` module, set it to open at 20 speed, and the gripper type to Adaptive Gripper.


myblockly
Blockly
Python
Logic
Loops
Text
Math
List
Variables
Functions
Time
Raspi-GPIO
System Status
WLAN Setting
Basic
ATOM IO
Status
MDI Control
JOG Control
Setting
Servo
Gripper
Coord Control
MyCobot320
Set Gripper Calibration.
Set Gripper State Open Speed 20 Type Adaptive Gripper
Set Gripper Value 20 Speed 20 Type Adaptive Gripper
Get Gripper Value Adaptive Gripper
Is Gripper Moving
Init Eletric Gripper
Set Eletric Gripper Open
Set Gripper Mode Transparent Transmission
Get Gripper Mode

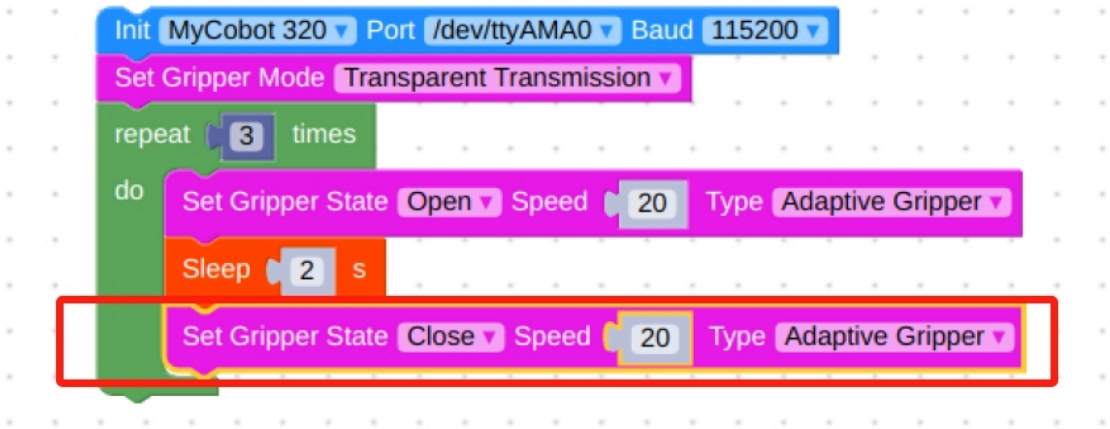

Init MyCobot 320 Port /dev/ttyAMA0 Baud 115200
Set Gripper Mode Transparent Transmission
repeat 3 times
do
Set Gripper State Open Speed 20 Type Adaptive Gripper

8. In `Time`, find the `Sleep` module and set the time to 2s, the purpose is to give the gripper time to make a movement.

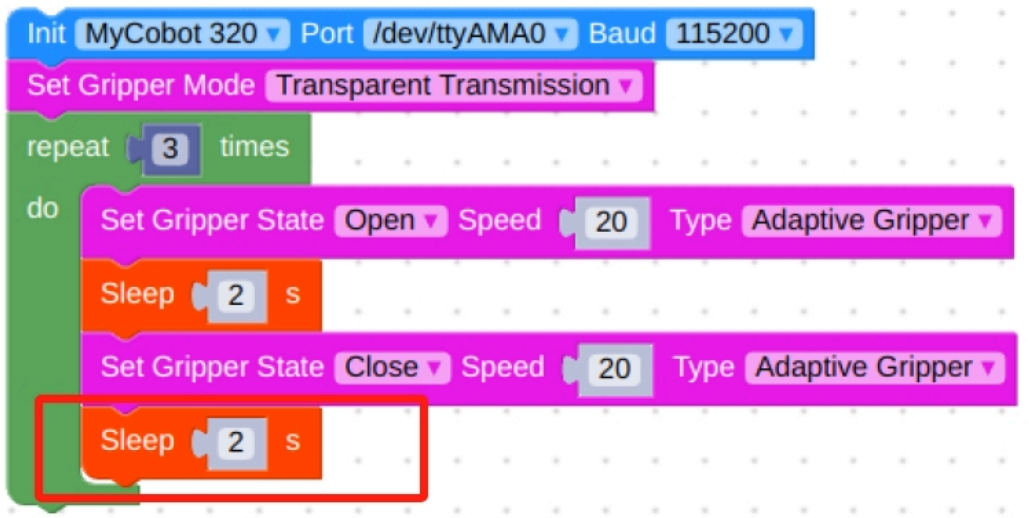




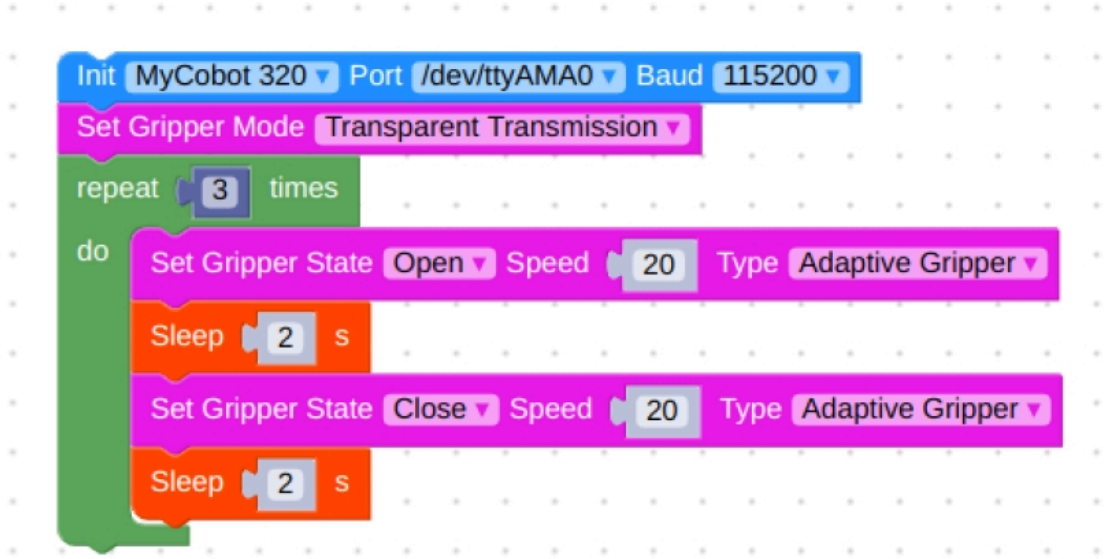
9. In `Gripper`, select the `Set Gripper State` module, put it in the `repeat` module, set it to close at 20 speed, and the gripper type to Adaptive Gripper.



10. In `Time`, find the `Sleep` module and set the time to 2s, the purpose is to give the gripper time to make a movement.



11. Final flowchart and code.

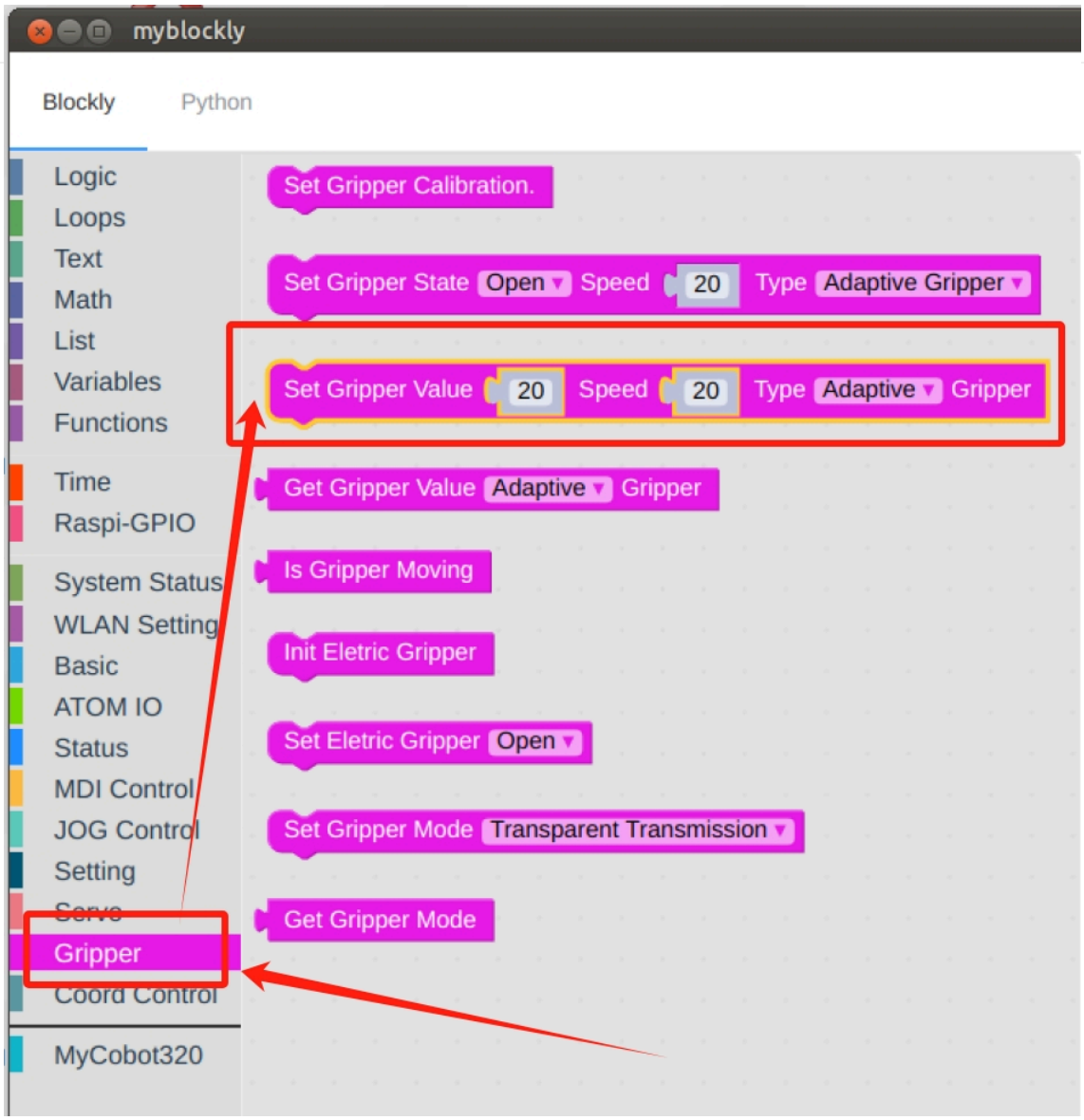



```
from pymycobot.mycobot import MyCobot
import time


mc = MyCobot('/dev/ttyAMA0', 115200)
mc.set_gripper_mode(0)
for count in range(3):
   mc.set_gripper_state(0,20,1)
   time.sleep(2)
   mc.set_gripper_state(1,20,1)
   time.sleep(2)
```

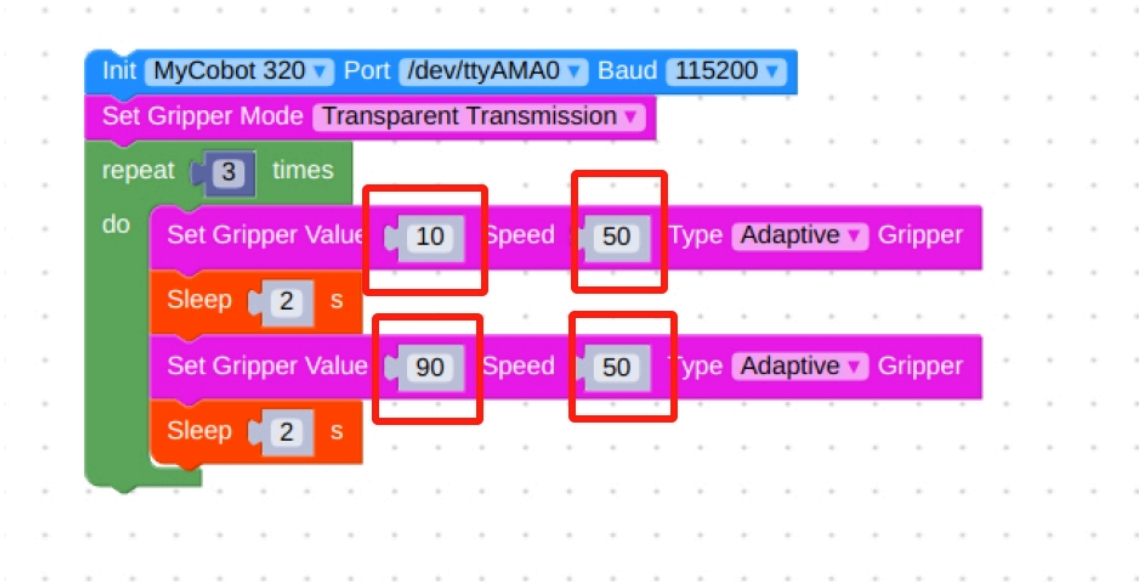
12. Click on the green running icon in the upper right corner to see the gripper `open-close-open` in motion.

- **Port Passthrough Mode Method 2:**

1. The process framework is the same as method 1, except that the `Set Gripper Mode` module is replaced by `Set Gripper Value` .



2. Final flowchart and code.

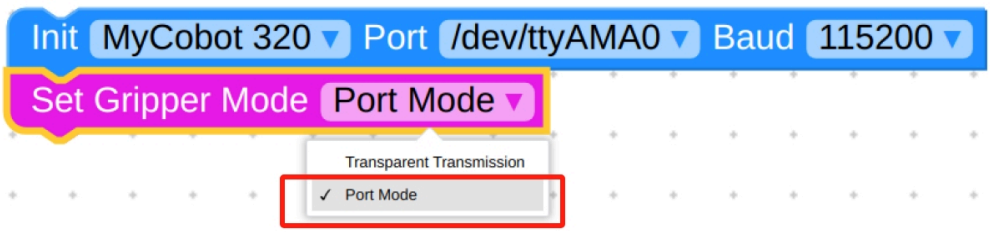


```
from pymycobot.mycobot import MyCobot
import time


mc = MyCobot('/dev/ttyAMA0', 115200)
mc.set_gripper_mode(0)
for count in range(3):
   mc.set_gripper_value(10,50,1)
   time.sleep(2)
   mc.set_gripper_value(90,50,1)
   time.sleep(2)
```

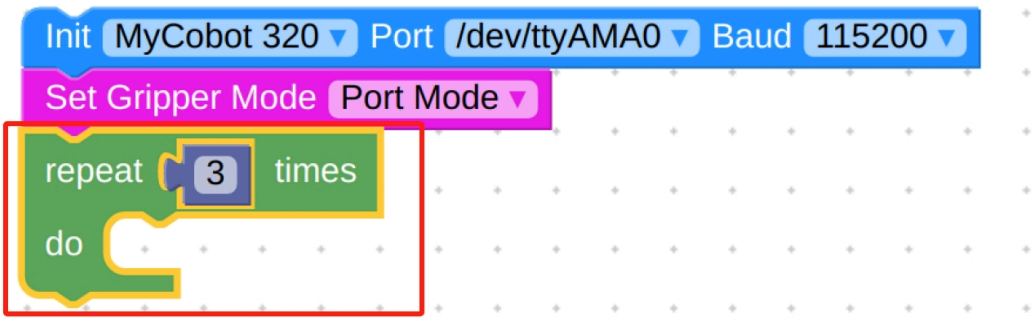
- **IO Control Mode:**

Note: When switching back to IO mode from pass-through mode, you need to power off and restart the machine before you can use IO mode normally.

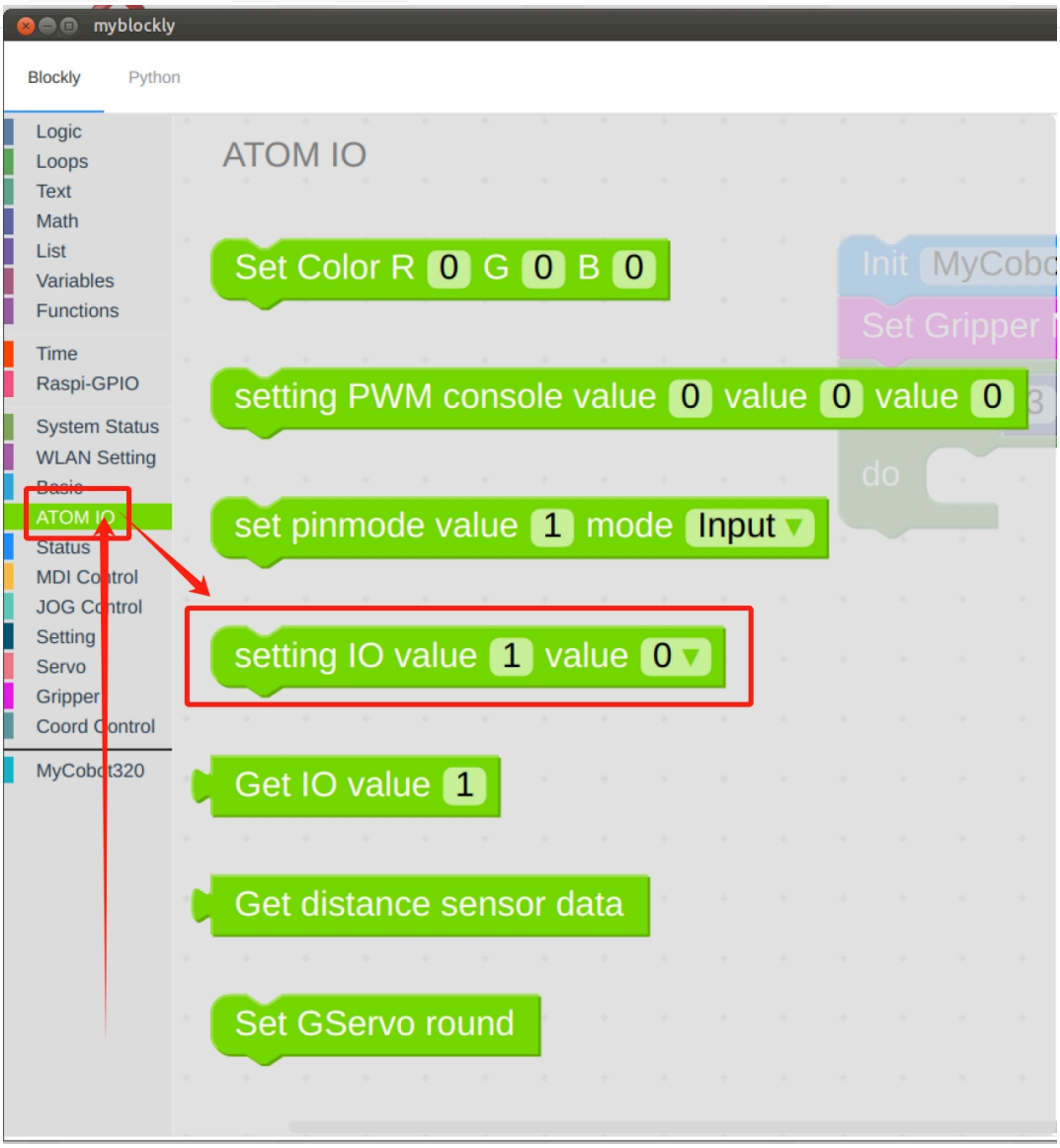
1. Setting the `Set Gripper Mode` module to Port Mode.



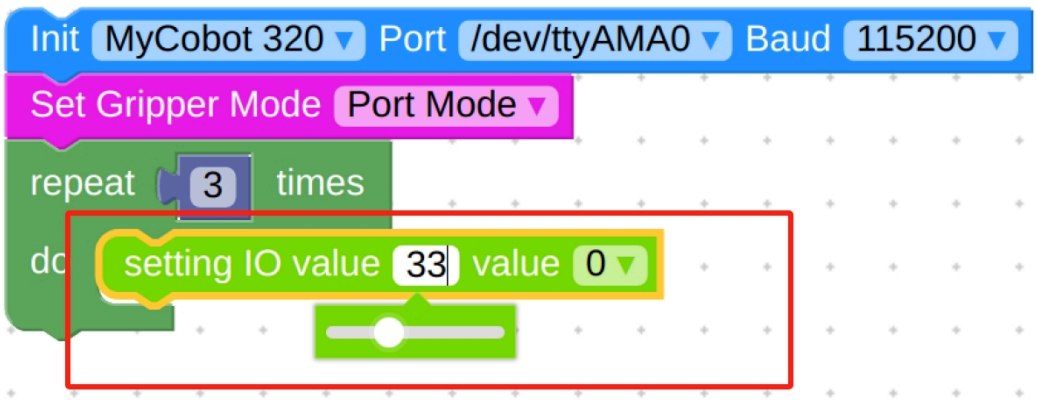
2. Again select the `repeat` module and loop through it three times.



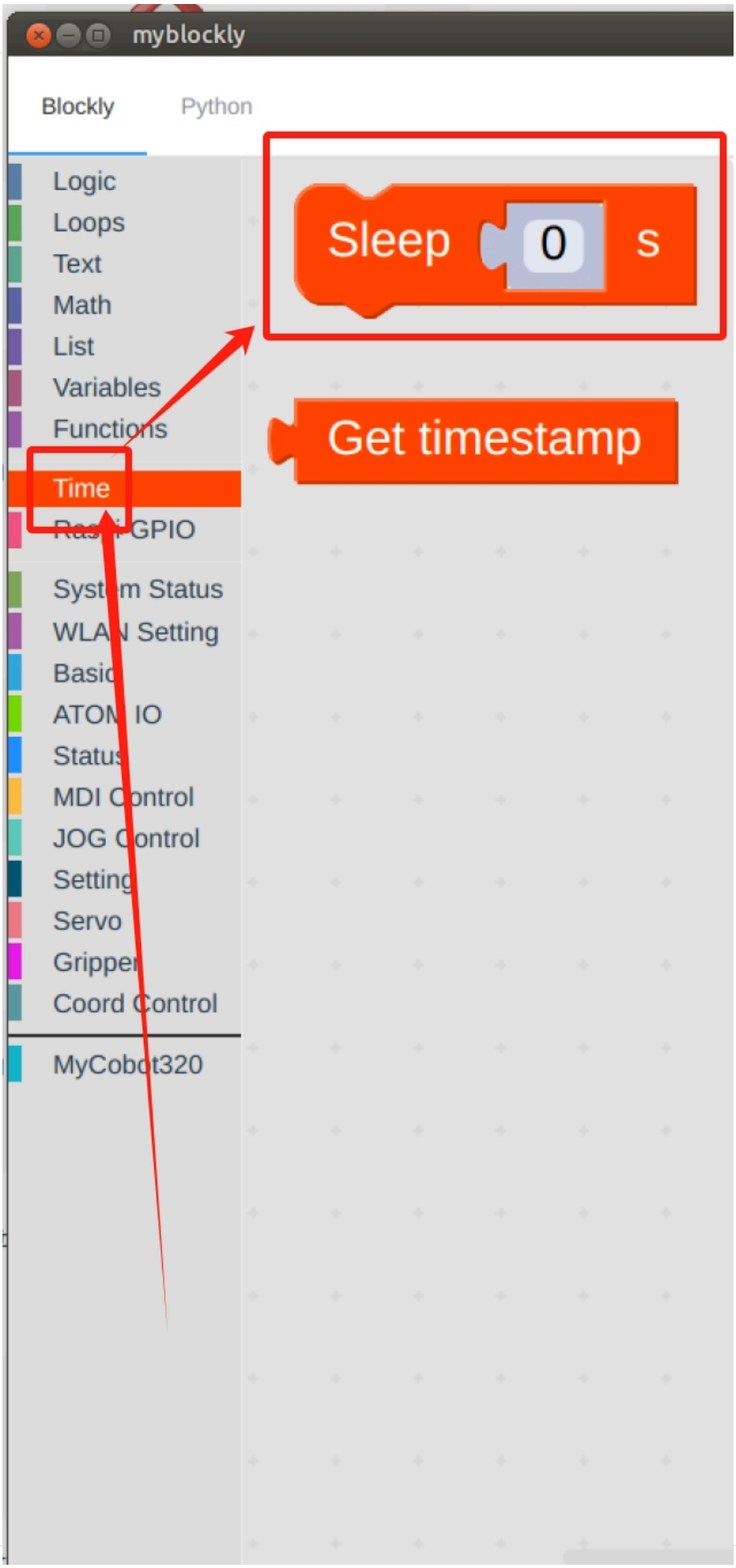
3. In `ATOM IO` select the `Setting IO value` module.



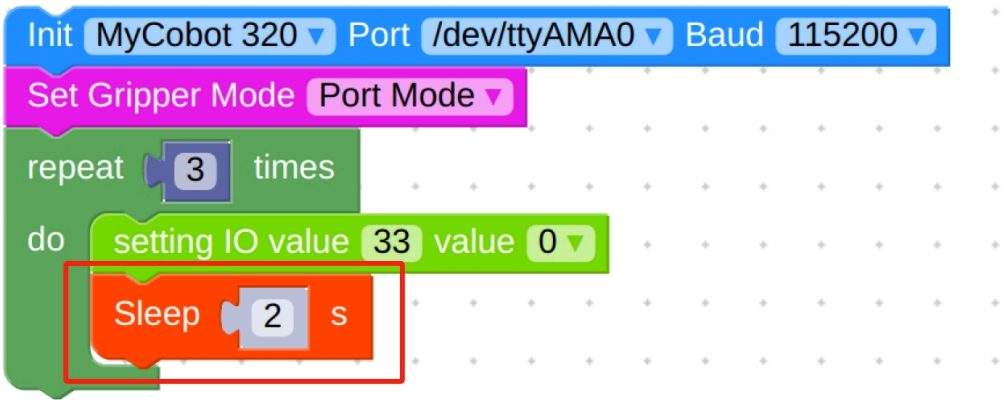
4. Set the IO port to 33 with a value of 0.



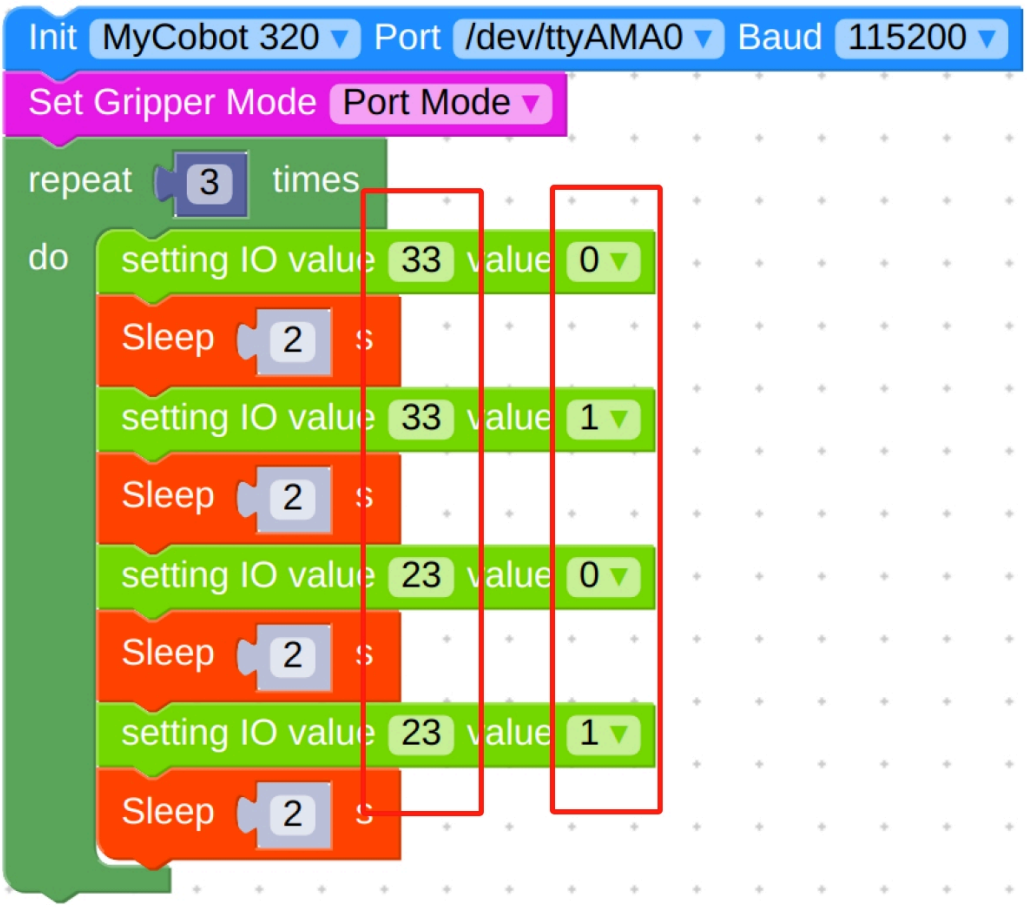
5. In `Time` select the `Sleep` module.


myblockly
Blockly
Python
Logic
Loops
Text
Math
List
Variables
Functions
Time
GPIO
System Status
WLAN Setting
Basic
ATOM IO
Status
MDI Control
JOG Control
Setting
Servo
Gripper
Coord Control
MyCobot320
Sleep 0 s
Get timestamp

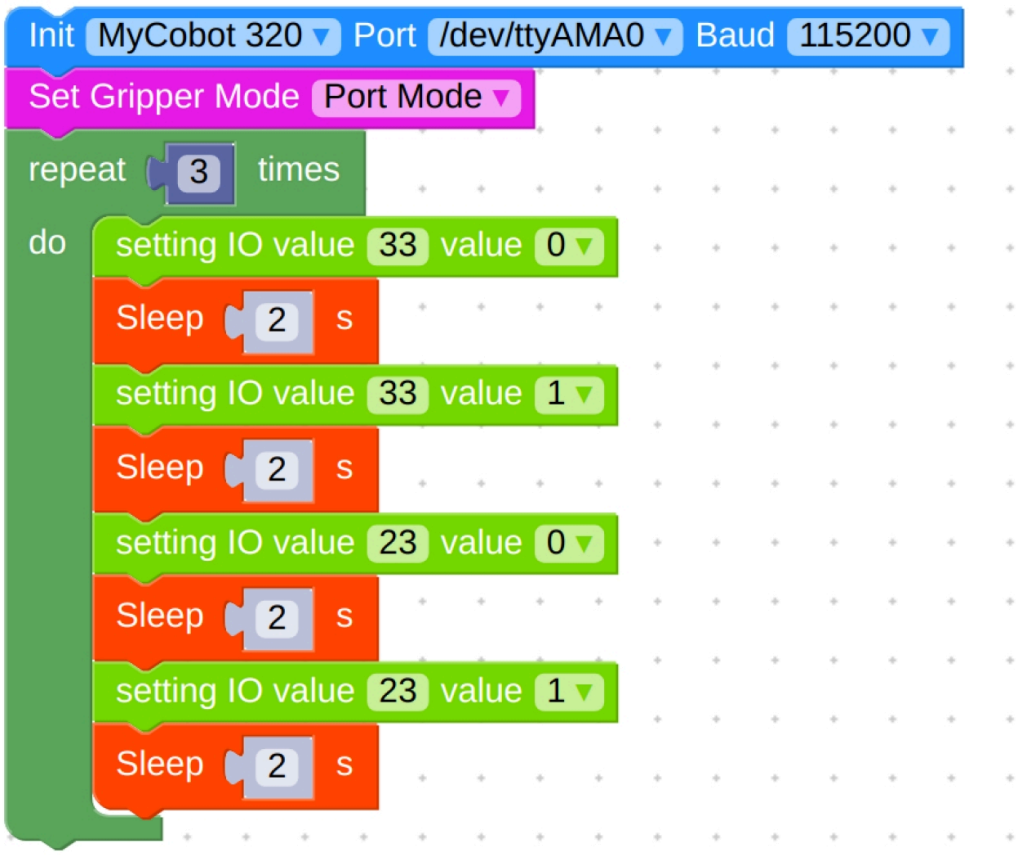
6. Setting time to 2s.



7. Repeat the selection of the `Setting IO value` and `Sleep` modules, noting the change of IO number and the change of value.



8. Final flowchart and code.




```python from pymycobot.mycobot import MyCobot import time

mc = MyCobot('/dev/ttyAMA0', 115200)
mc.set_gripper_mode(1)
for count in range(3):
    mc.set_digital_output(33, 0)
    time.sleep(2)
    mc.set_digital_output(33, 1)
    time.sleep(2)
    mc.set_digital_output(23, 0)
    time.sleep(2)
    mc.set_digital_output(23, 1)
    time.sleep(2)
```

← Accessories Tools Page | Next Page →

Electric Gripper

> **Compatible models:** myCobot 320, myCobot Pro 630

Product icon



Specifications:

name	mycobot Pro Electric Gripper
Model model	myCobotPro_Gripper_PGE_8
craft	Metal + 7500 Nylon
Clamping rangeClamp size	0-14mm
Clamp force	2-5N
Repeatability precision	1mm
Lifetime	one year
drive mode drive	electric
transfer method	Rack and pinion + cross roller guide
size	97×62×31mm
weightweight	460g
Fixed method Fixed	screw fixed
Use environment requirements	Temperature and pressure
control interface control	Serial port/IO control
Applicable equipment	ER myCobot 320 ，ER myCobot Pro 600

Use for Gripping Objects

Introduction

- PGE series are industrial thin parallel electric grippers, and the numbers represent the maximum clamping force of the grippers. The gripper is equipped with a pair of parallel fingertips, which run symmetrically during the movement. The main structure of the gripper is a smooth rectangular structure, which is small in size and saves installation space. It has 5 mounting holes to meet different installation conditions of the equipment.
- Quick response, high grabbing frequency, and equipped with an 8-core communication interface, mainly to achieve clamping or stuck objects, suitable for relatively light objects.

working principle

- The motor drives the rack and pinion and the cross roller guide to realize the opening or closing action of the gripper. The positioning point of the electric gripper is controllable and the clamping is controllable.

Applicable object

- The volume is smaller than the clamping stroke
- The weight is less than the maximum clamping weight
- Custom fingertips can expand more items

Product parameters

PGE-8-14

Slim-type Electric Parallel Gripper

Parameters

Product Parameter	
Gripping force (per jaw)	2~8 N
Stroke	14 mm
Recommended workpiece weight *	0.1 kg
Opening/Closing time	0.2 s/0.2 s
Repeat accuracy (position)	± 0.02 mm
Noise emission	< 40 dB
Weight	0.4 kg
Driving method	Rack and pinion + Liner guide
Size	97 mm x 62 mm x 31 mm
Working Environment	
Communication interface	Standard: Modbus RTU (RS485), Digital I/O Optional: TCP/IP, USB2.0, CAN2.0A, PROFINET, EtherCAT
Rated voltage	24 V DC ± 10%
Rated current	0.4 A
Peak current	0.7 A
IP class	IP 40
Recommended environment	0~40°C, under 85% RH
Certification	CE, FCC, RoHS

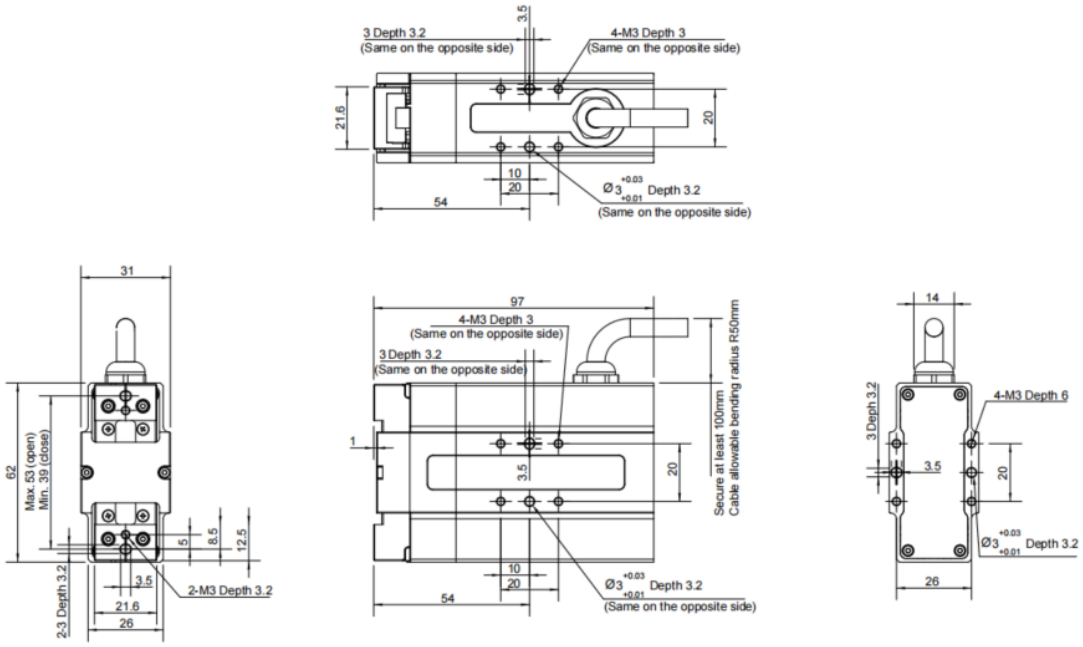
Build-in Controller	Gripping Force Adjustable	Position Adjustable	Speed Adjustable	Drop Detection	Self-locking Mechanism
●	●	●	●	●	○



Vertical Maximum Force	
Fz:	90 N
Allowable Moment	
Mx:	0.55 N·m
My:	0.45 N·m
Mz:	0.55 N·m

*It depends on the shape of the grasping object, the material and friction of the contact surface, and the acceleration of the motion, If you have any questions, please contact us.

技术尺寸图



operating environment

surroundings	illustrate
Protocol	RS-485
Operating Voltage	24V
rated current	0.25A
peak current	0.5A
degree of protection	IP40
Recommended environment	0-40℃

Mall link

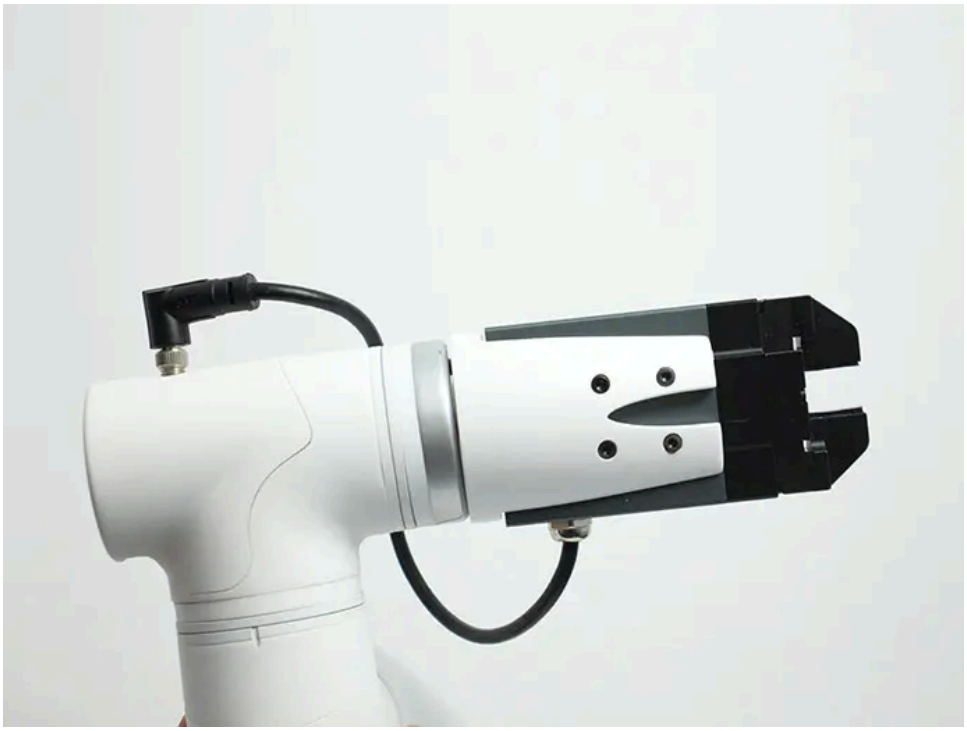
Mall link:

- Taobao
- shopify

How to use

1 Installing the gripper:

- For an electric gripper, insert it into the 485 interface on the top, as shown in the following figure:



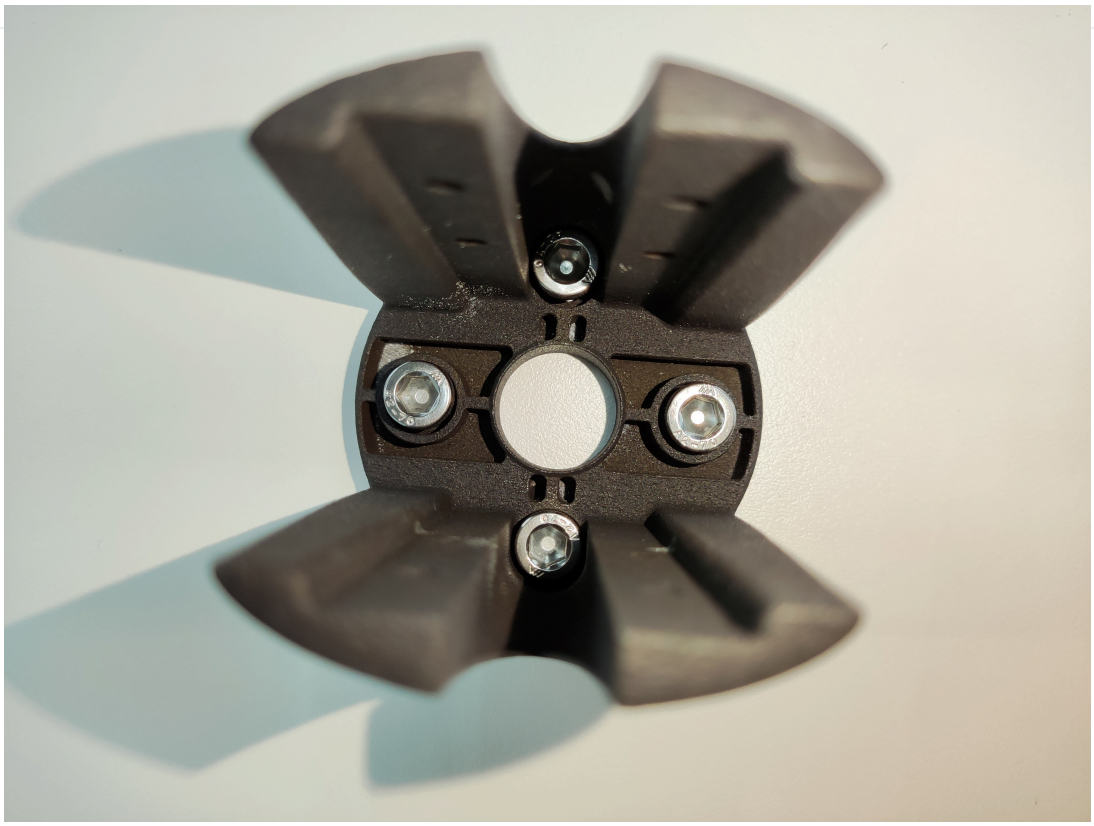
Installation and use

- Check that the kit is complete: screws, hexagonal spanner, gripper with connecting wires, gripper with arm end fixing device



- gripper mounted:
    - Structural installation:

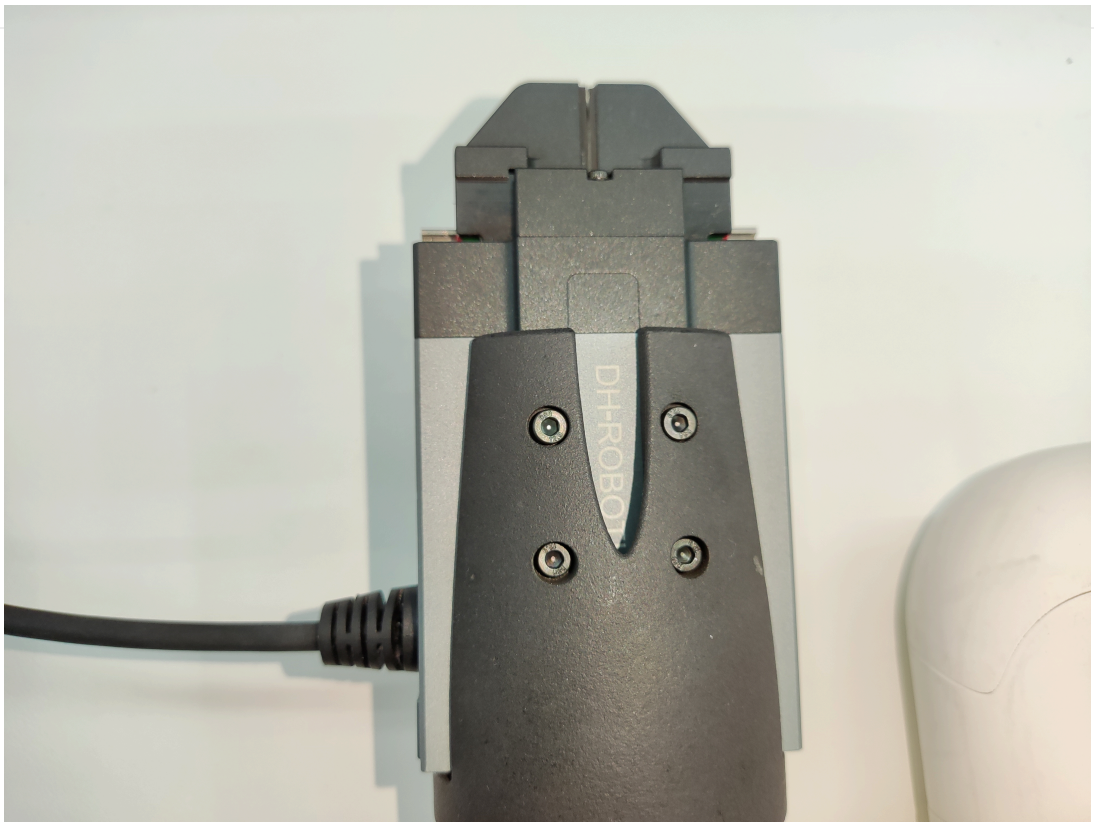
1. Place the short screws into the holes in the fixture:



2. Align the screws with the four screw holes on the end of the robot arm and tighten:

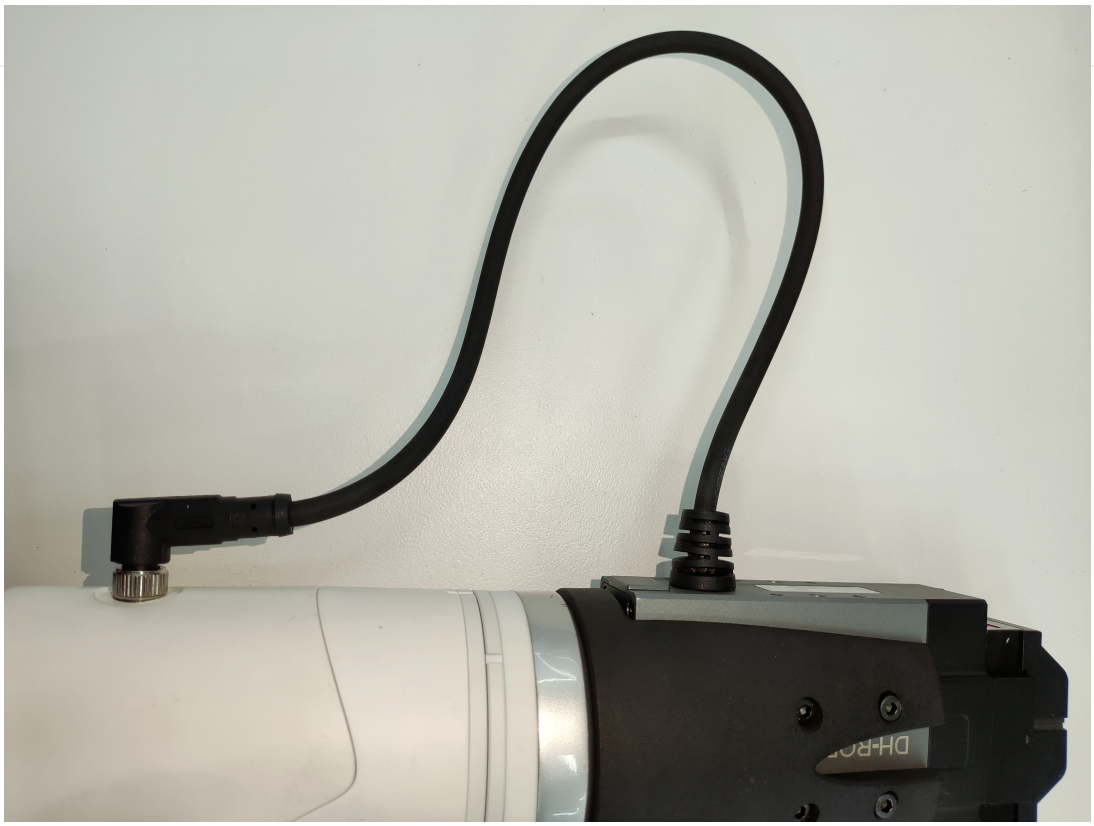


3. Insert the motorised gripper into the fixture and tighten with fine screws on both sides, eight in total:



- Electrical Connections:

1. Plug the 485 cable into the robot arm control connector:



Instructions for use:

```
from pymycobot.mycobot import MyCobot
import time

# Initialise a MyCobot object
mc = MyCobot("COM3", 115200)

# Can control the gripper to open and strike - close and open:
mc.set_eletric_gripper(0)
time.sleep(3)
mc.set_eletric_gripper(1)
time.sleep(3)
mc.set_eletric_gripper(0)
time.sleep(3)
```

← Accessories Tools Page | Next Page →

Pneumatic gripper

Compatible models: myCobot 320, myCobot Pro 630

product icon



Specifications:

name	mycobot320 pneumatic gripper
Model model	myCobotPro_Gripper_Air_10
craft	Metal + 7500 Nylon
Clamping rangeClamp size	0-8mm
Clamp force	Outer diameter 34N Inner diameter 45N
Repeatability precision	±0.01mm
Lifetime	one year
drive mode drive	pneumatic
transfer method	Piston cylinder
size	67.3×38×23.6mm
weightweight	180g
Fixed method Fixed	screw fixed
Use environment requirements	Temperature and pressure
control interface control	I/O control
Applicable equipment	ER myCobot 320 M5 ER myCobot 320 Pi ER myCobot Pro 600

Use for Gripping Objects

Introduction

- Pneumatic grippers, also known as pneumatic fingers or pneumatic grippers, are actuators that use compressed air as power to grip or grab workpieces. It is small in size, light in weight, compact in appearance, capable of single- and two-way grabbing, automatic centering, high repeatability, and automatic control of the magnetic switch.
- Pneumatic gripper set includes gripper flange, air pump, φ8 air pipe, φ6 air pipe, φ8-6 quick connector, solenoid valve and cables. Its main function is to replace human grasping work, which can effectively improve production efficiency and work safety. An external suction pump is required.

working principle

- Single piston: the axis drives the crank, and the air claw is driven by the piston to open and close. A corresponding crank groove is respectively arranged on the two claw pieces. In order to reduce the frictional resistance, the claw piece and the body are connected by a steel ball slide rail structure.
- Double piston: It is operated by two pistons, and each piston is connected with a pneumatic finger by a roller and a double crank to form a special drive unit. Realize that the pneumatic fingers always move axially to the center, and each finger cannot move independently. Parallel gripper cylinder If the finger moves in the opposite direction, the previously compressed piston is in the exhausted state, while the other piston is in the compressed state.

Applicable object

- The volume is smaller than the clamping stroke
- The weight is less than the maximum clamping weight
- Custom fingertips can expand more items

Mall link

- Taobao
- shopify

How to use

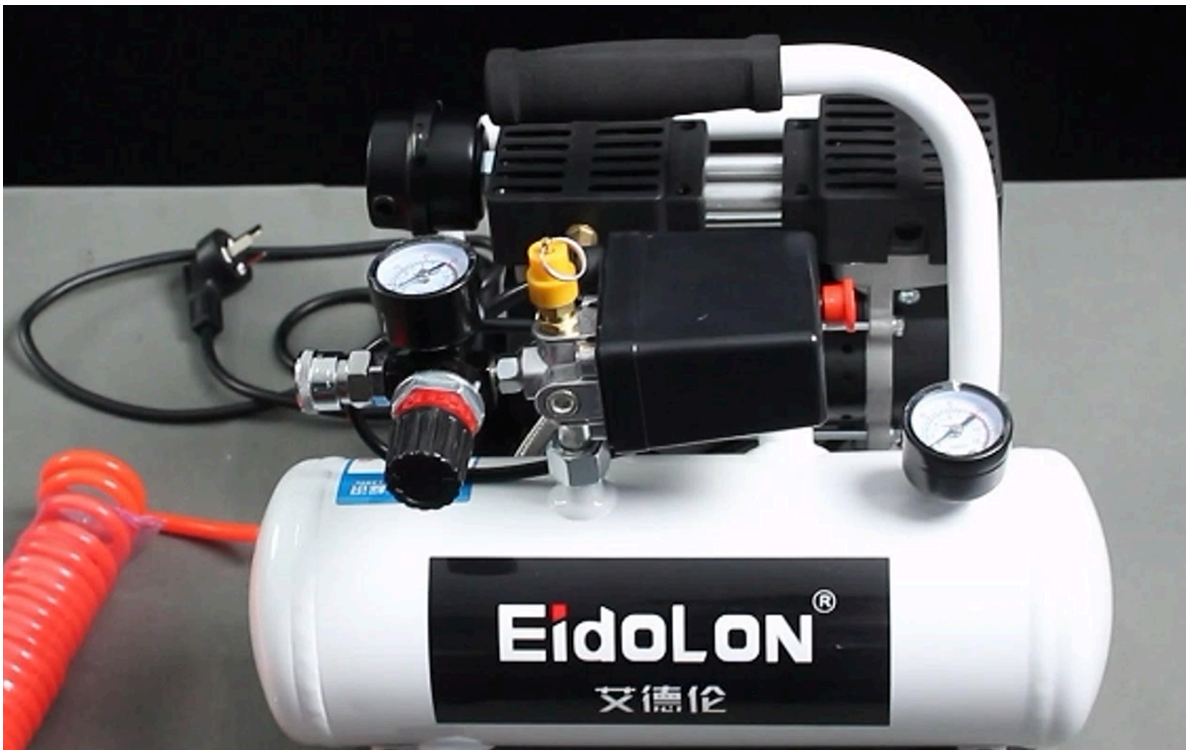
1 Installing the gripper:



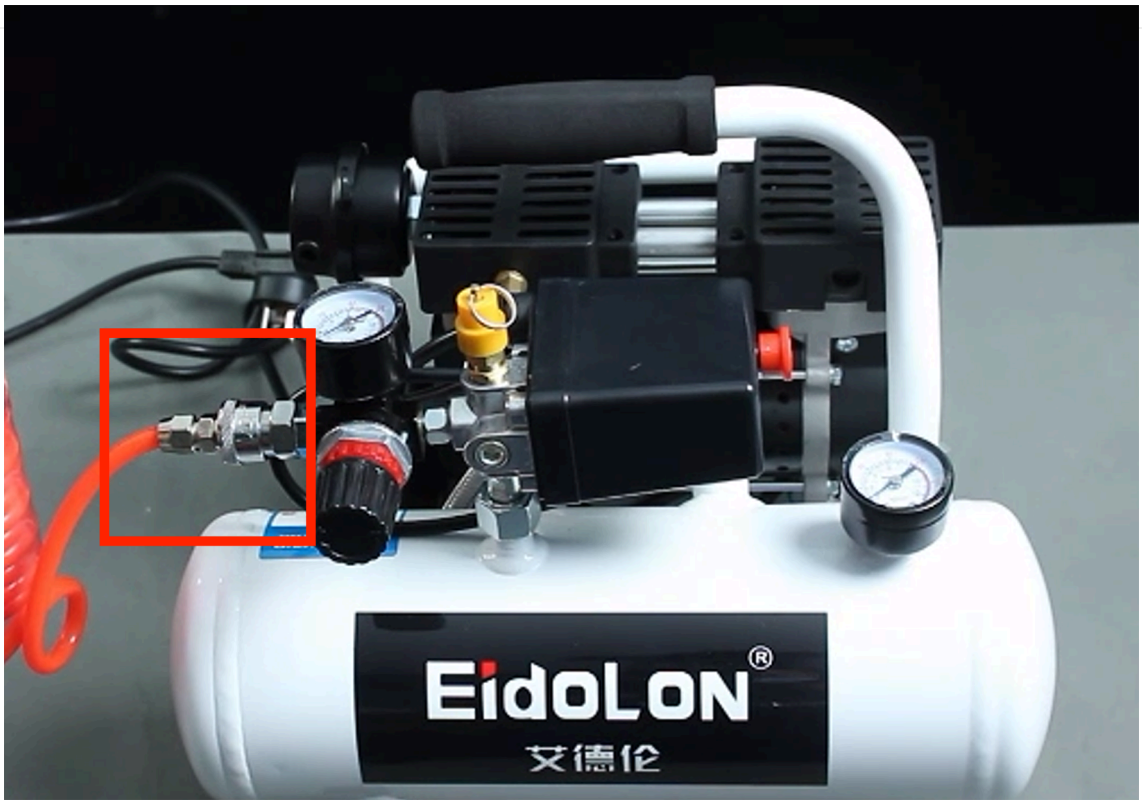


Installation and use

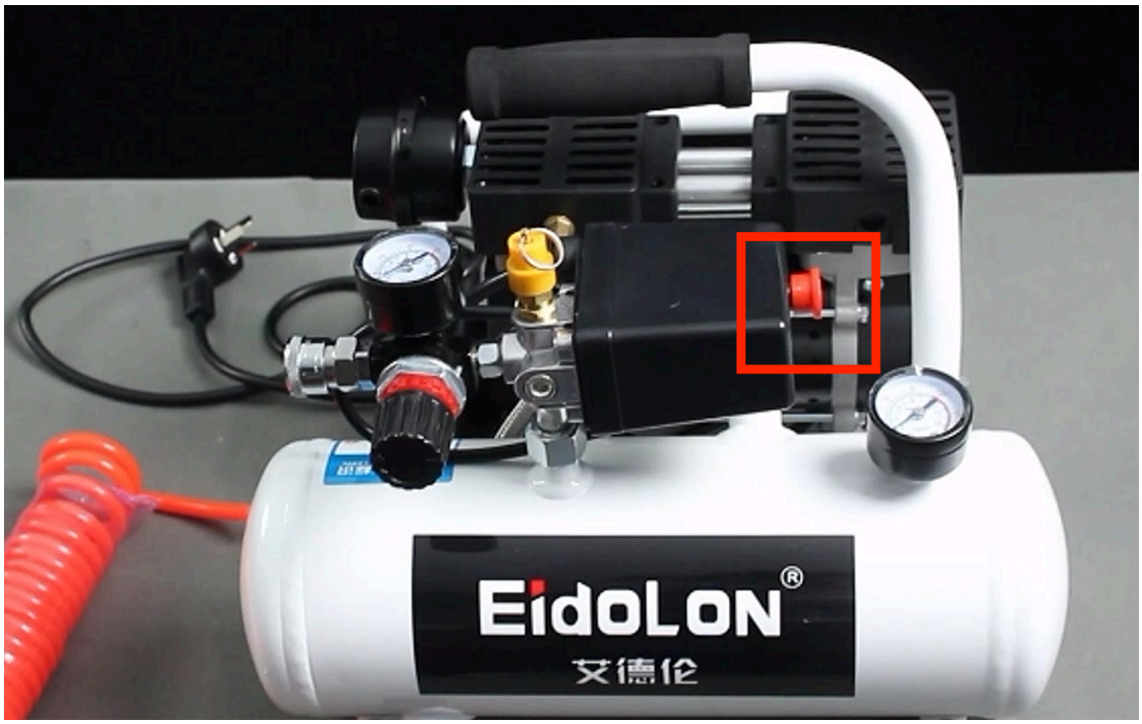
- It needs to be used with an air compressor:



1. Insert the black plug into the row of plugs;
2. Insert the matching red hose into the connector on the machine:

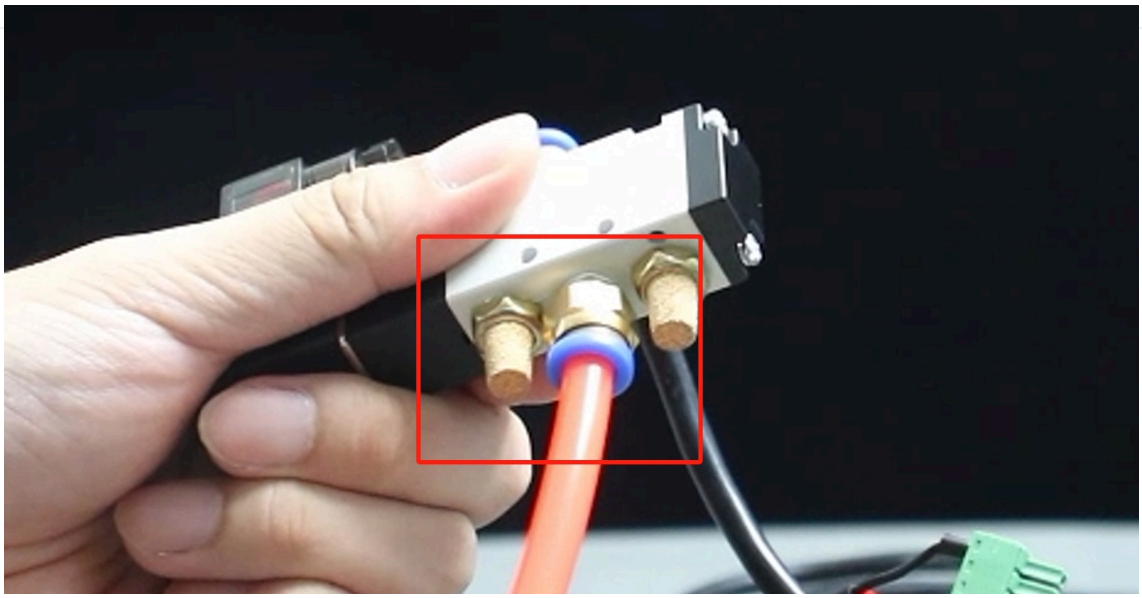


3. The red button is the on/off switch, pulling it outwards turns it on, pressing it back turns the machine off:



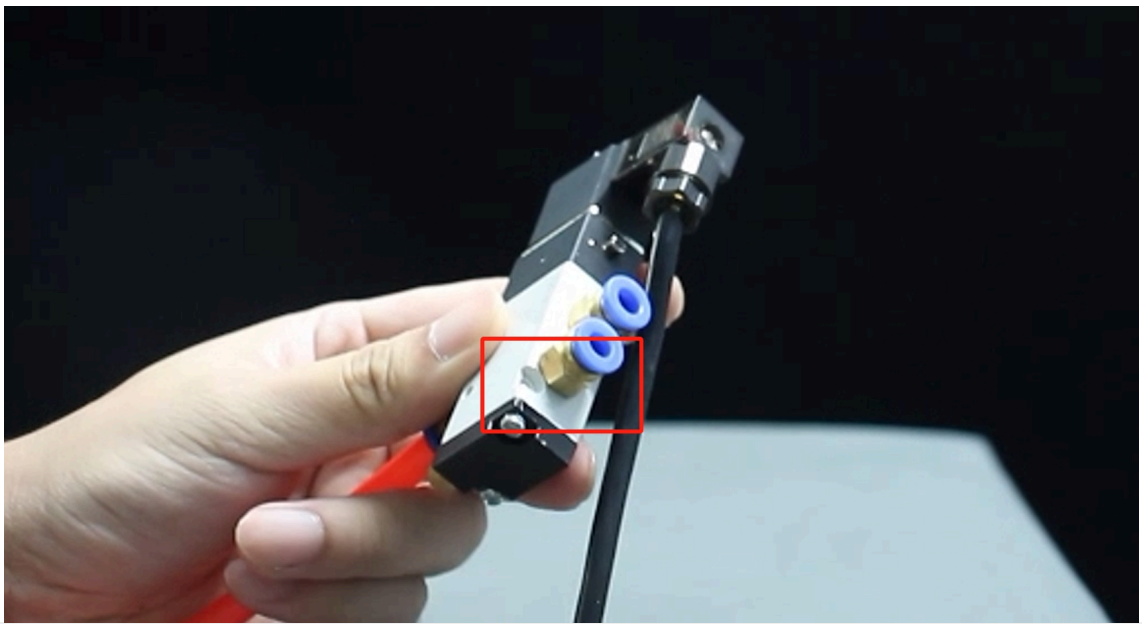
- gripper mounted:

1. Connect the other end of the red hose from the air compressor to the solenoid valve connection:

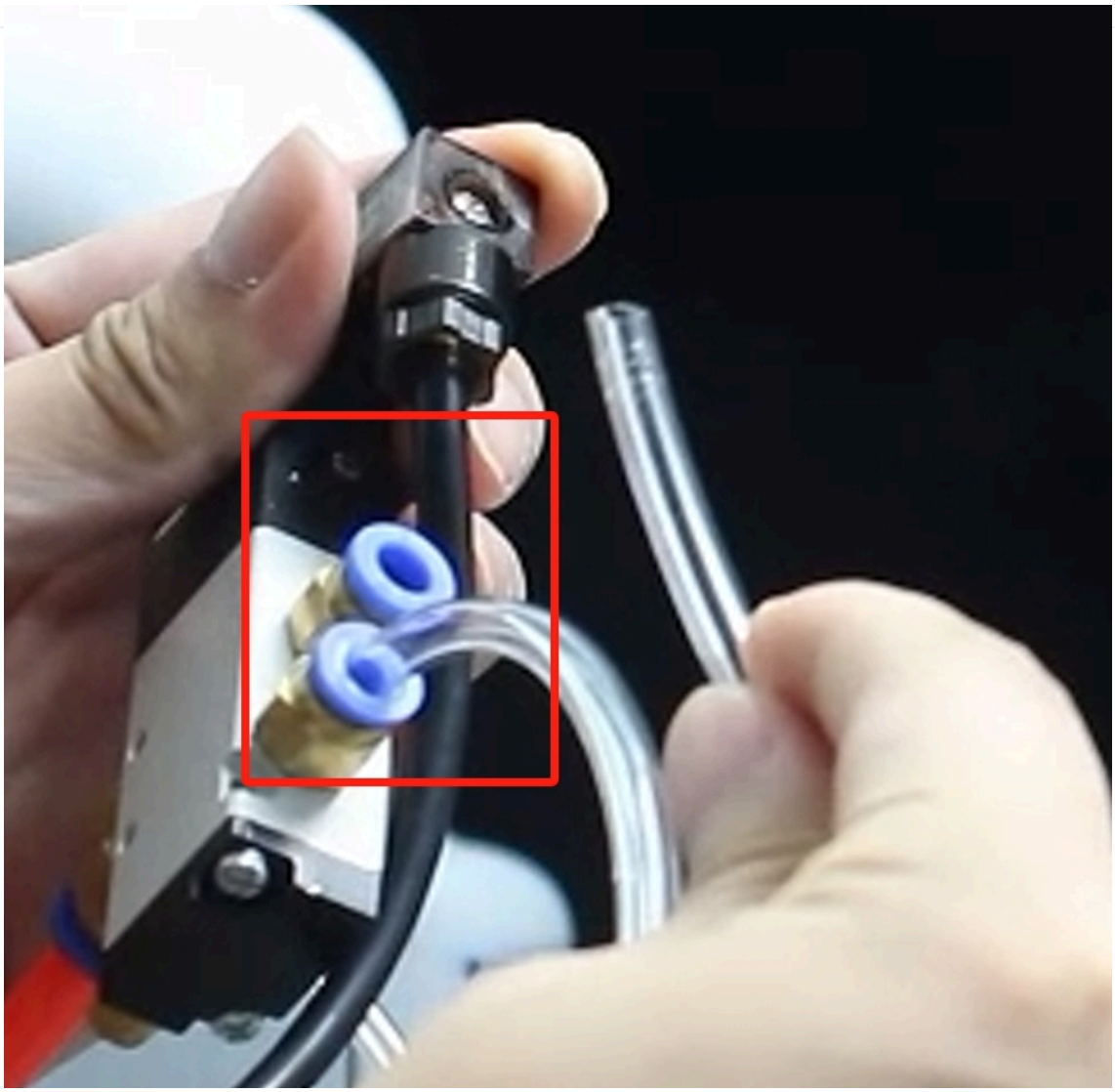


2. The other end of the solenoid valve will be unscrewed another port for the activation of the gripper to control the opening and closing of the use:

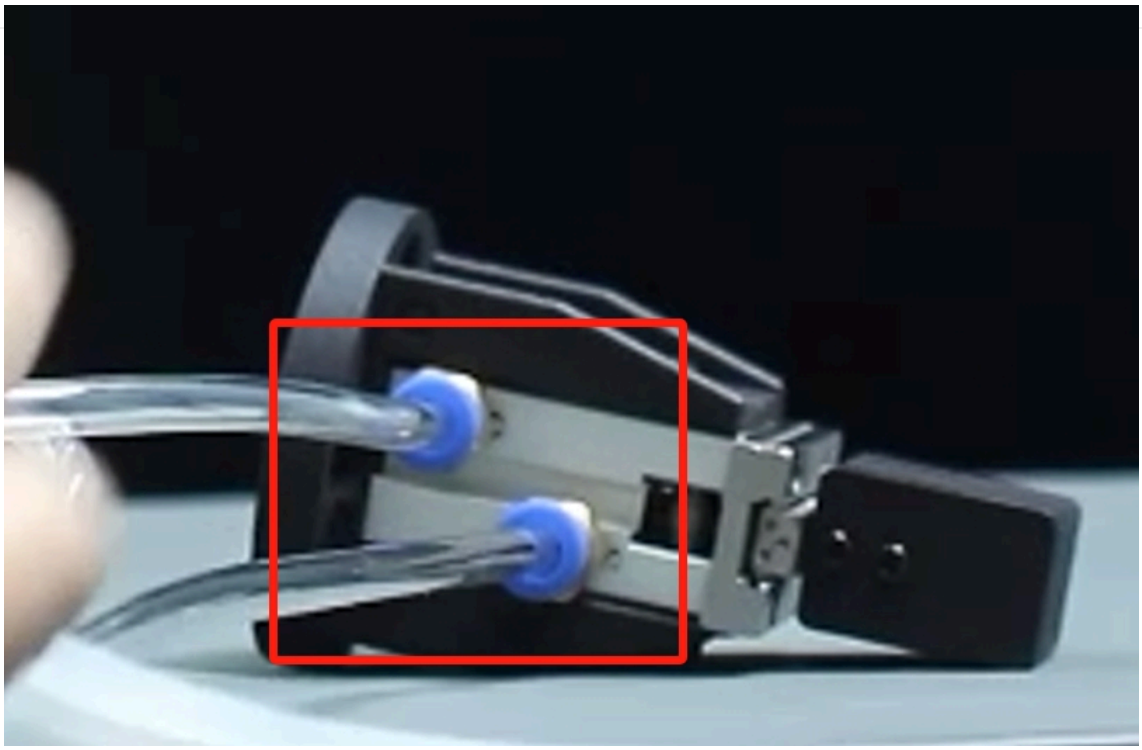




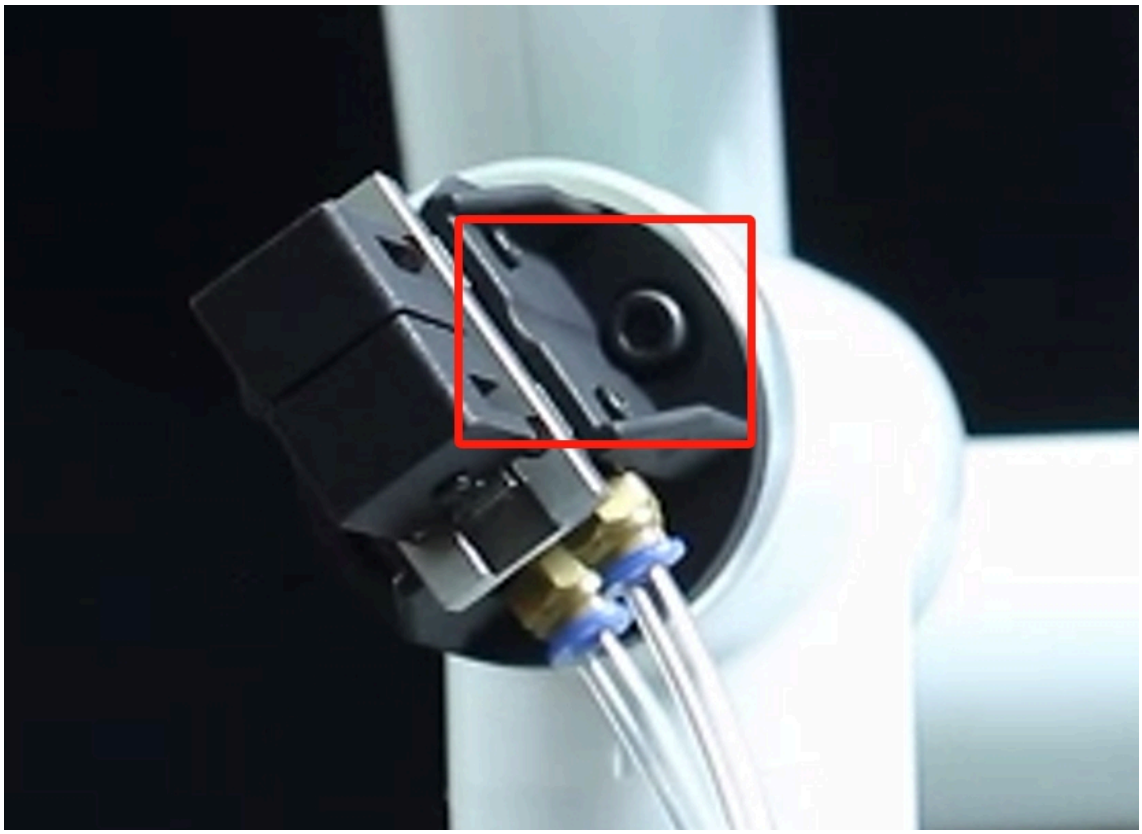
3. Use the two matching clear hoses with one end connected to the two ports of the solenoid valve:



4. The other end of the transparent hose is connected to the two connections of the gripper:

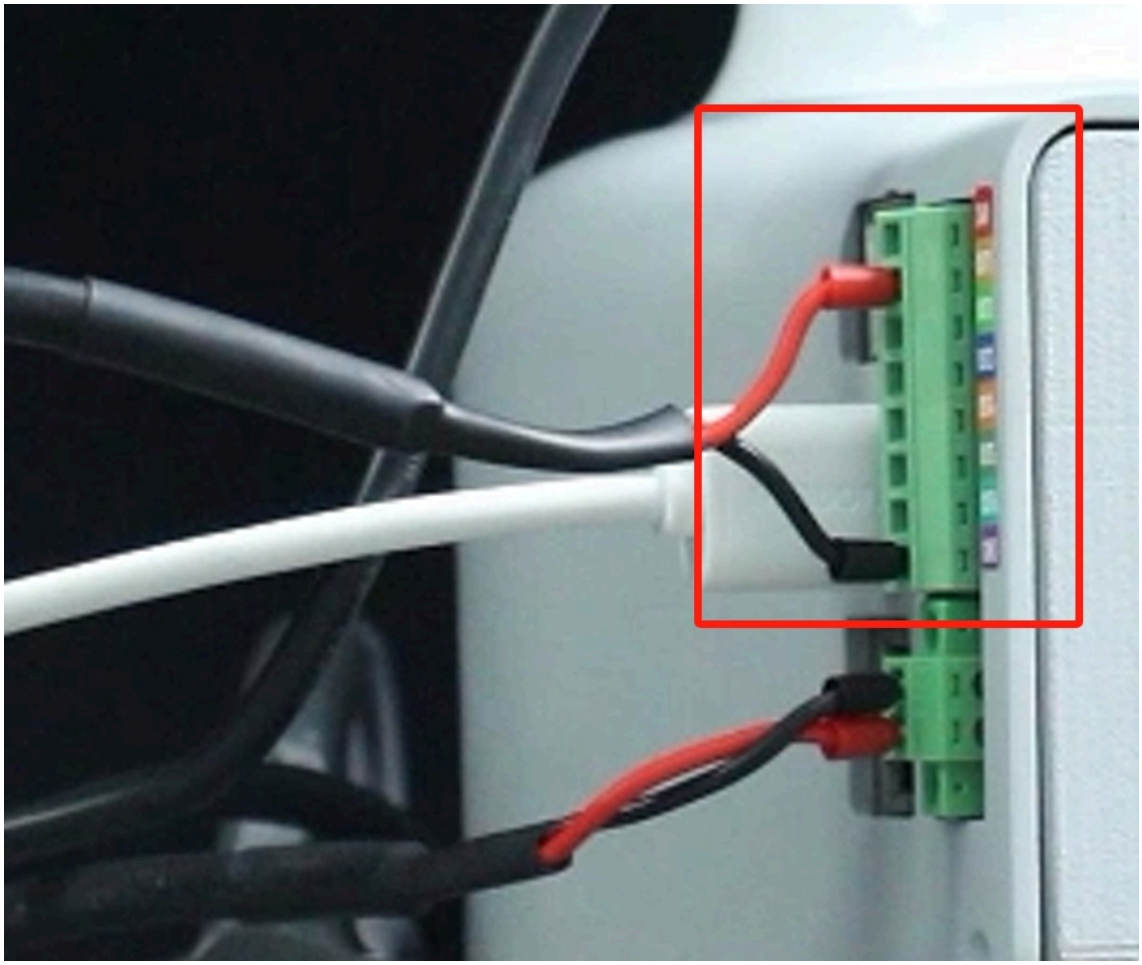


5. Secure the gripper to the end of the arm with the matching screws:



- Electrical Connections:

1. Connect the black wire to the GND of the robot arm base, and the red wire to any one of OUT1~OUT6, and change the pin number of the subsequent programme according to the selected interface, here we use OUT1:



- Software-driven testing:

> To test if the gripper are available after installation, use myBlockly. myblockly 下载

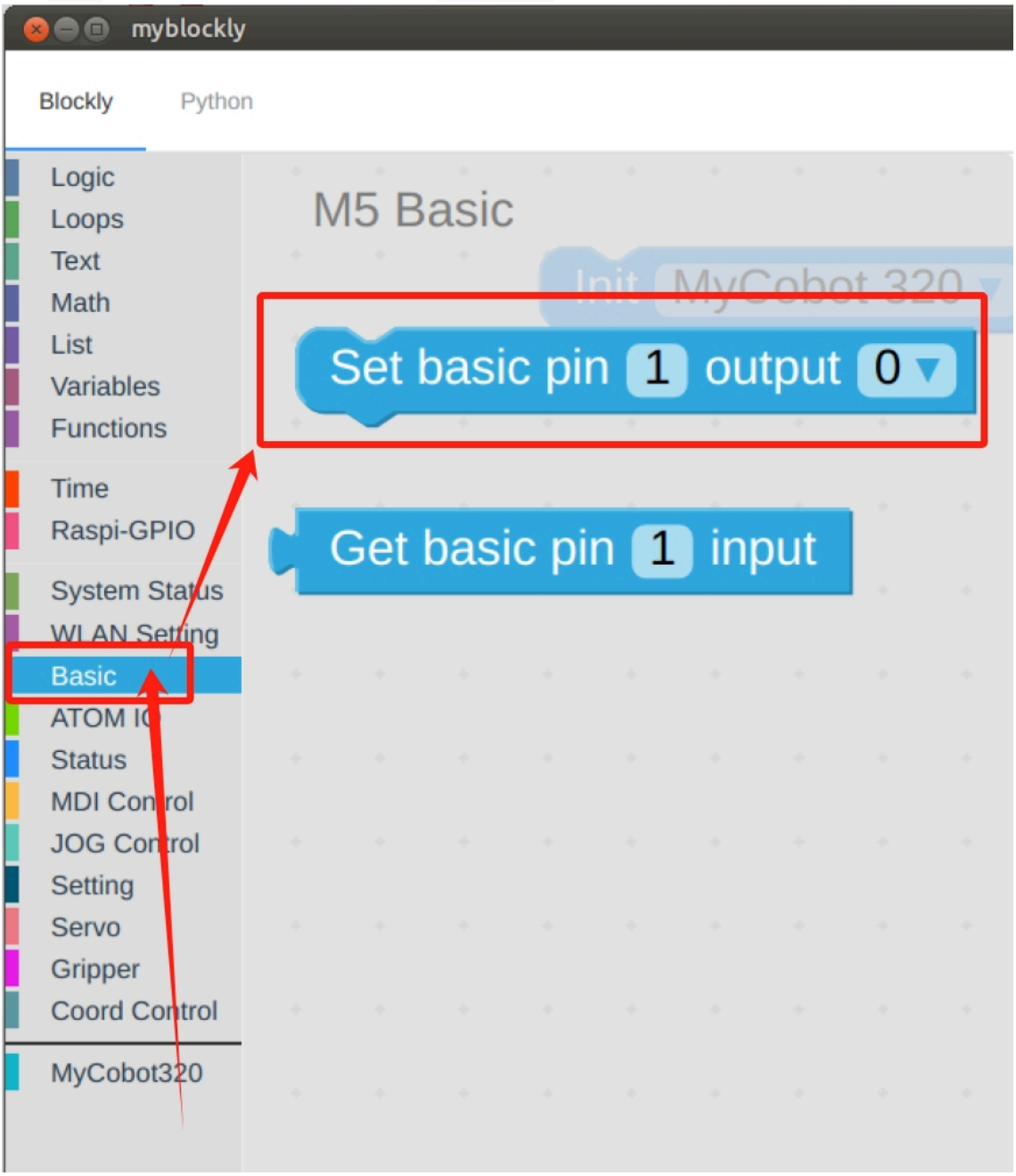
1. After confirming that the structural and electrical connections are complete, start the arm and open the myblockly software when the graphical interface appears:



2. Modify the baud rate to 115200:



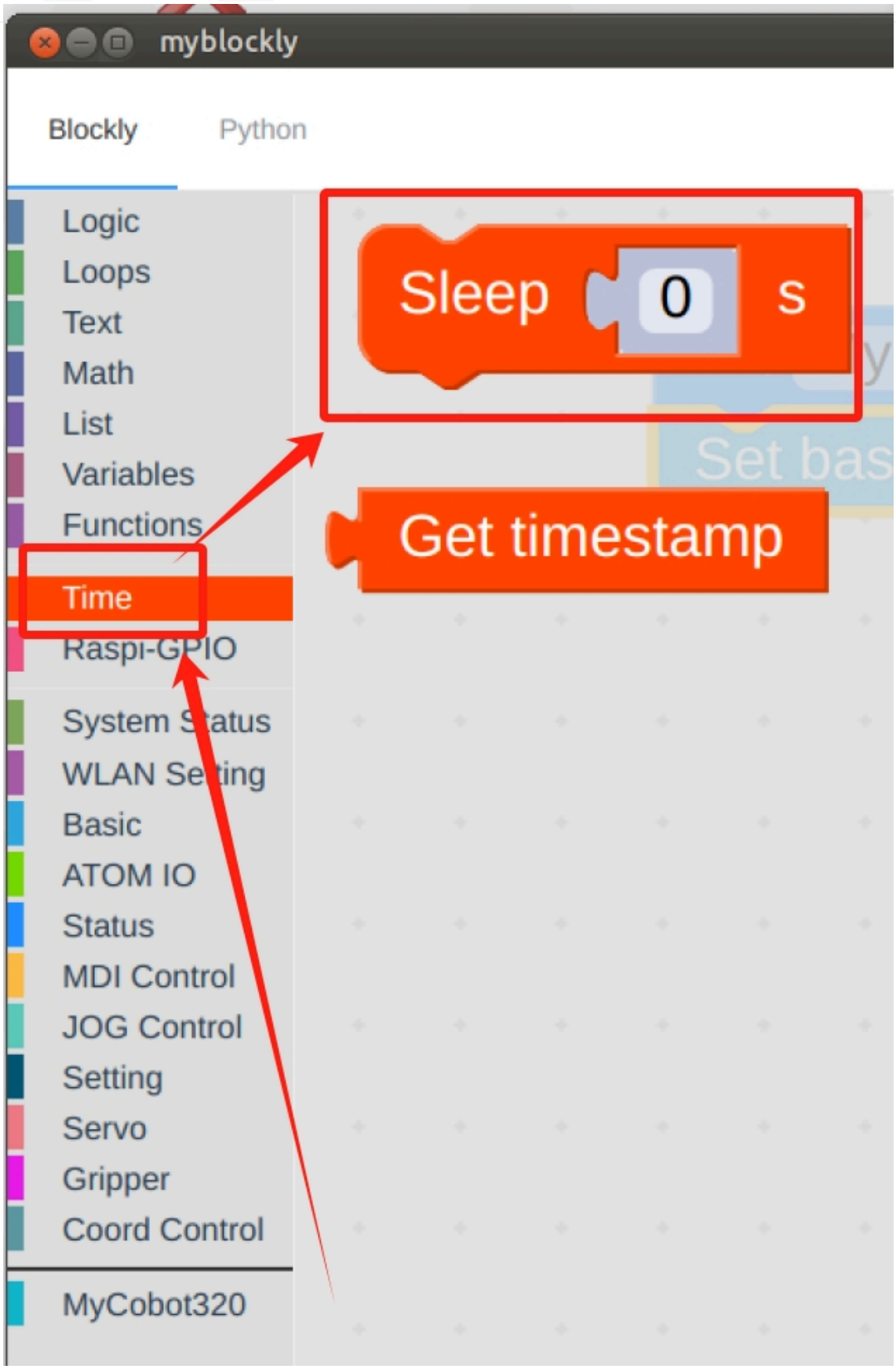
3. Find `Base` in the list on the left and select the `Set Pin Out` module:



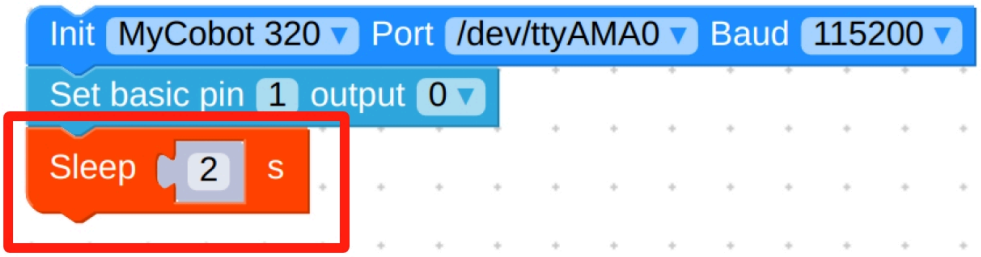
4. Set `pin number` to `1` and `output` to `0` :



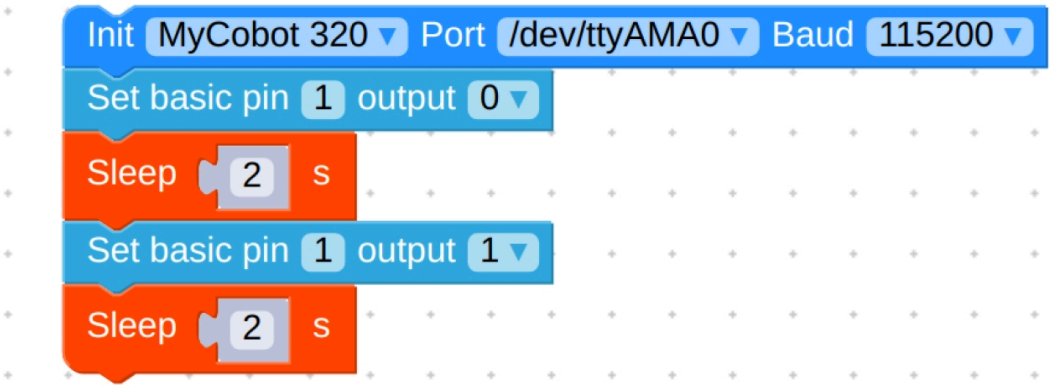
5. Find `Time` and select the `Sleep` module:



6. Set the time as desired, here it is set to `2s` :



7. Repeat the above steps for the final setup as follows:



8. Final code:

```
from pymycobot.mycobot import MyCobot
import time


mc = MyCobot('/dev/ttyAMA0', 115200)
mc.set_basic_output(1, 0)
time.sleep(2)
mc.set_basic_output(1, 1)
time.sleep(2)
```

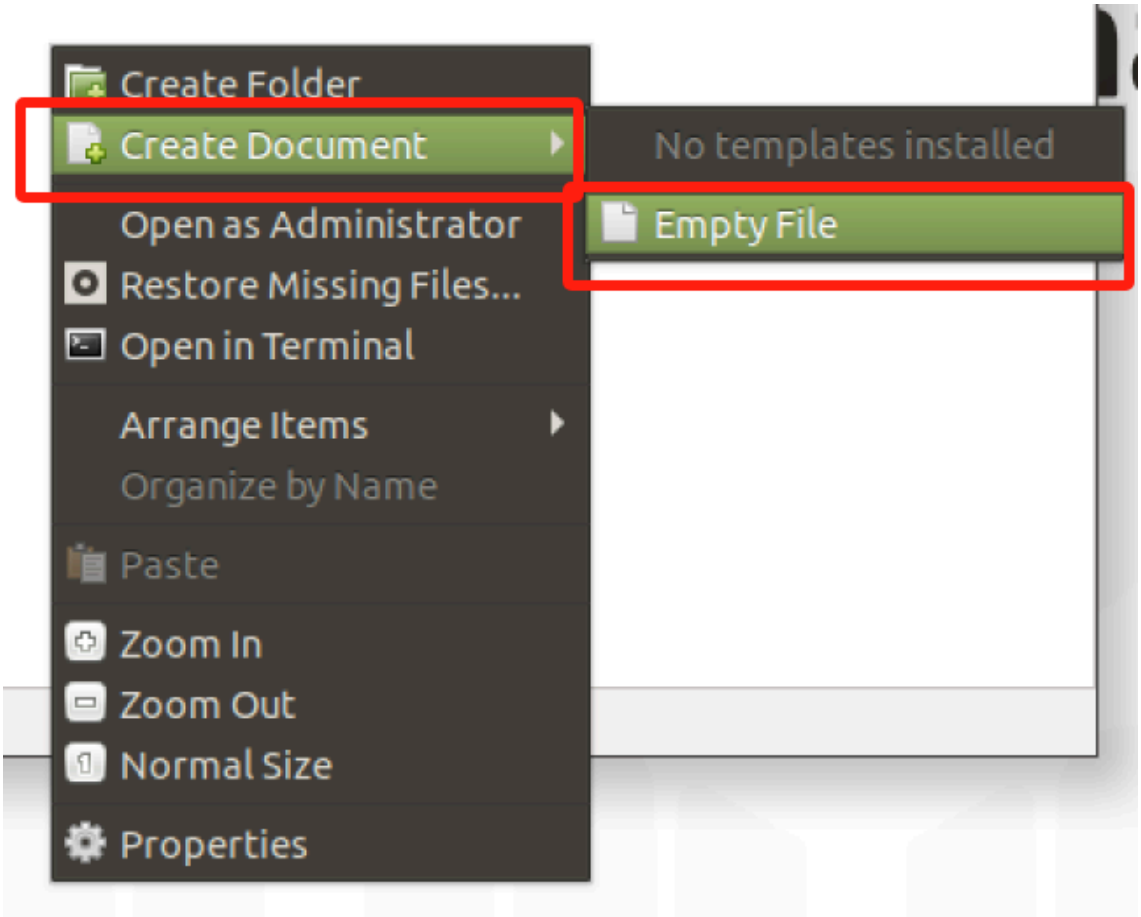
9. Click on the green run button in the top right corner to see the gripper close-open once.

- Programming Development:

> Programming the gripper using python python environment download

1. Create a new python file:

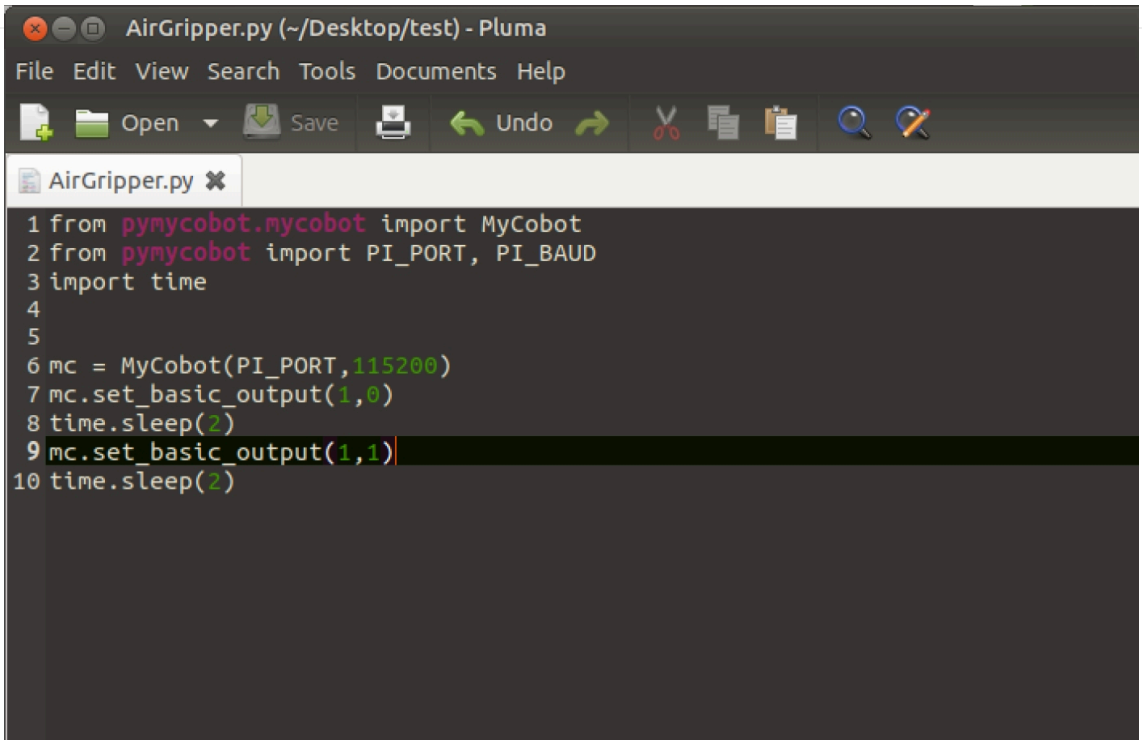
Right click on the desired file path to create a new python file:



> The file name can be changed as needed



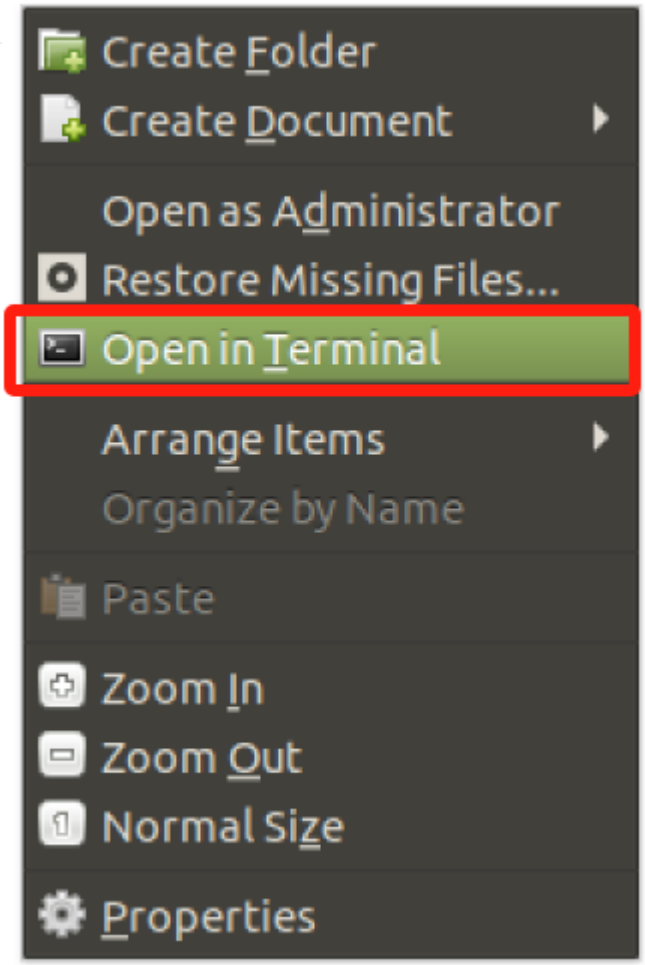
2. Perform function programming:



> The code is as follows:

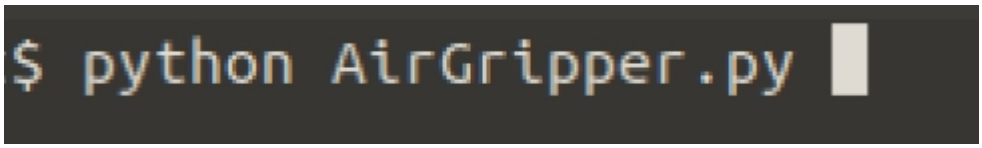
```
from pymycobot.mycobot import MyCobot
import time


# Initialise a MyCobot object
mc = MyCobot("COM3", 115200)


# Controls gripper closed-open:
# Using the gripper status interface 0 is open, 1 is closed
mc.set_basic_output(1, 0)
time.sleep(1)
mc.set_basic_output(2, 1)
time.sleep(1)
mc.set_basic_output(1, 1)
time.sleep(1)
mc.set_basic_output(2, 0)
time.sleep(1)


# For more information on using the interface, see the python API.
```

3. Save the file and close it, right-click on an empty space in the folder to open a command line terminal



Input:

```
python AirGripper.py
```



> gripper can be seen closed-open

← Accessories Tools Page | Next Page →

Flexible Gripper Pro

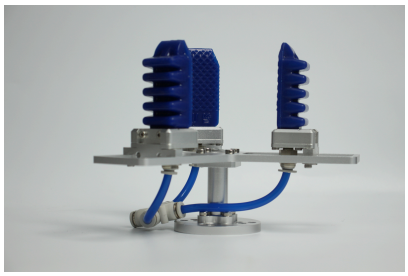
> **Compatible models:** myCobot 320, myCobot Pro 630

Product Icon

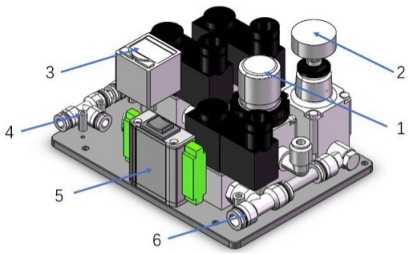


Specifications

Gripper

Picture	
Material	Metal
clamping rangeclamp size	36-136mm
clamp force	Vertical 600g wrap 1080g
Repeatability precision	0.5 mm
service life lifetime	1 year
drive mode drive	pneumatic
Transmission modetransmission	deformation
size	170x128x195mm
weightweight	365 grams
Fixed method fixed	screw fixed
Use environment requirements	Temperature and pressure
control interface control	SAC40A Pneumatic Controller
Applicable equipment	ER myCobot 320 series, ER myCobot Pro 600

Controller

Picture	
Power	24V 1.5A
Controller	IO
Air Source Pressure	> 0.4 MPA
Flow Rate	> 40 L/MIN
IO	
Value State	IN1: HIGH & IN2: LOW = Close Gripper IN1: LOW & IN2: HIGH = Open Gripper Caution ⚠️: When using IO control, ensure the three-position toggle switch above is set to the middle position.
船型开关	The term "船型开关" translates to "toggle switch," specifically referring to a three-position toggle switch.
正压	Gripper Close
负压	Gripper Open

Use for Gripping Objects

Introduction

- Traditional industrial suction cups need to absorb 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 pinches the edge to grab the panel easily and without trace or damage, ensuring that the product surface is flawless. , Improve 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 clamping panel is easy to pinch and warp. The single-finger clamping force of the flexible gripper is precise and controllable, and will not pinch fragile workpieces.

working principle

- The flexible gripper is an innovative bionic flexible gripper developed by researchers to imitate the shape of the starfish's arms and legs. The "fingers" of the soft claw are made of flexible polymer silicone material, which can be bent and deformed by inflation. It can adaptively cover the target object like a starfish, and can complete the flexible and non-destructive grasping of special-shaped and fragile objects. Pick.

Applicable object

- Objects of any shape within a reasonable size

Mall link:

- Taobao
- shopify

How to use

1 Installing:

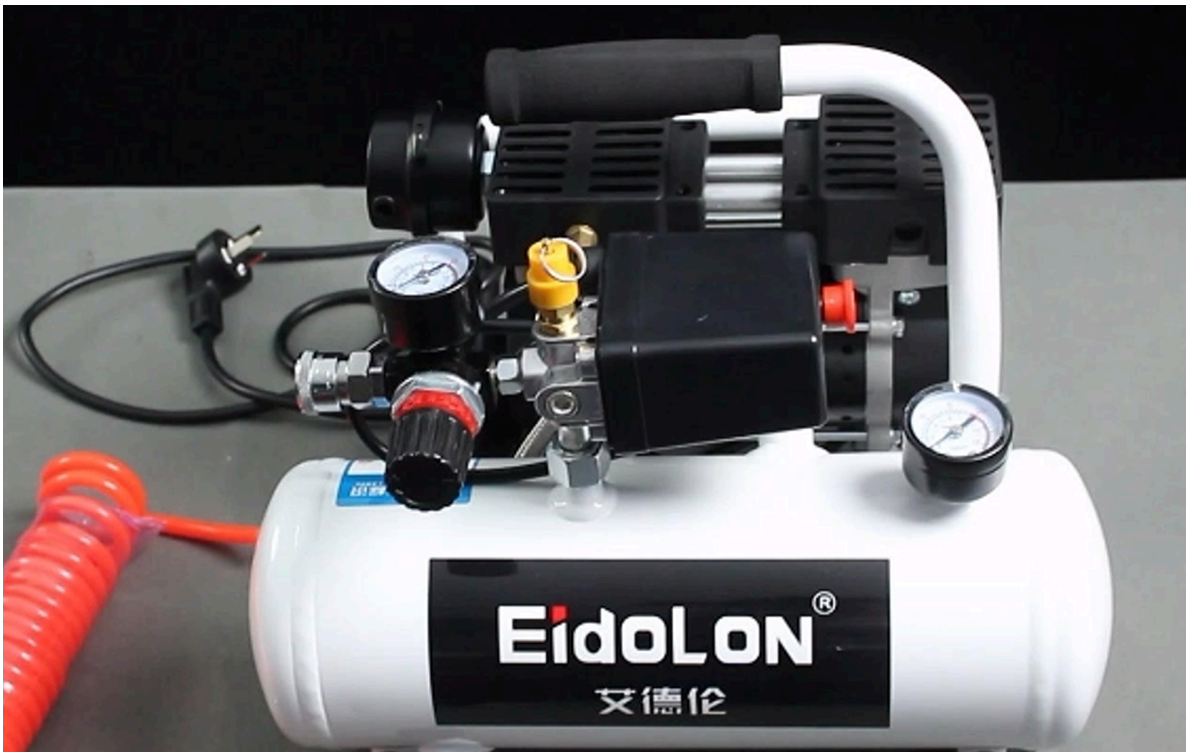
- Pneumatic Circuit Connection and Pressure Adjustment: Start by adjusting the positive pressure before connecting to the soft mechanical grip (the outlet can temporarily be blocked with a fingertip, ensure the positive pressure is strictly less than 100KPA to avoid potential finger injury).
- For connecting and adjusting the pneumatic circuit: First, set the positive pressure before connecting to a soft mechanical grip. Temporarily block the outlet with a fingertip to prevent air escape, ensuring the positive pressure is strictly below 100KPA to avoid potential injury to the finger.
- **Connect to Air Source**: Attach the air source to port 6 (inlet), ensuring the pressure is above 0.4MPA and flow exceeds 40L/MIN.
- **Connect Soft Mechanical Grip**: Link the soft mechanical grip to port 4 (outlet).
- **Power Up**: Supply power to connection 5.
- **Adjust Positive Pressure**: Use the toggle switch to select positive pressure. Adjust the regulator as per the gauge until desired pressure is reached.

- **Switch to Negative Pressure**: For negative pressure adjustment, pull up the regulator cap, turn to set, then push down to lock.

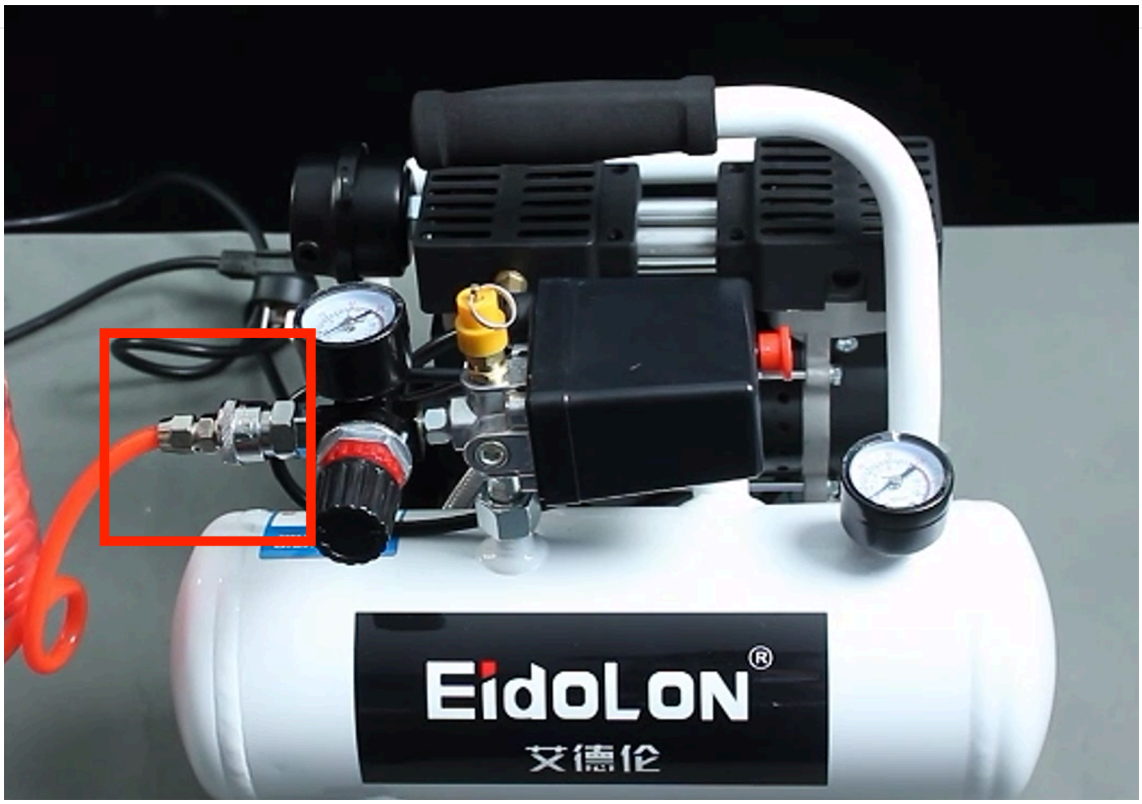
Special Reminder ⚠️: Regarding pressure maintenance, both positive and negative pressures can be maintained after 0.5 seconds of activation, allowing the mechanical grip to retain its preset pressure. This approach is energy-efficient, especially since maintaining negative pressure continuously consumes more air. Additionally, it prevents the loss of grip on objects in case of unexpected power or air supply cut-offs, ensuring operational reliability.

Installation and use

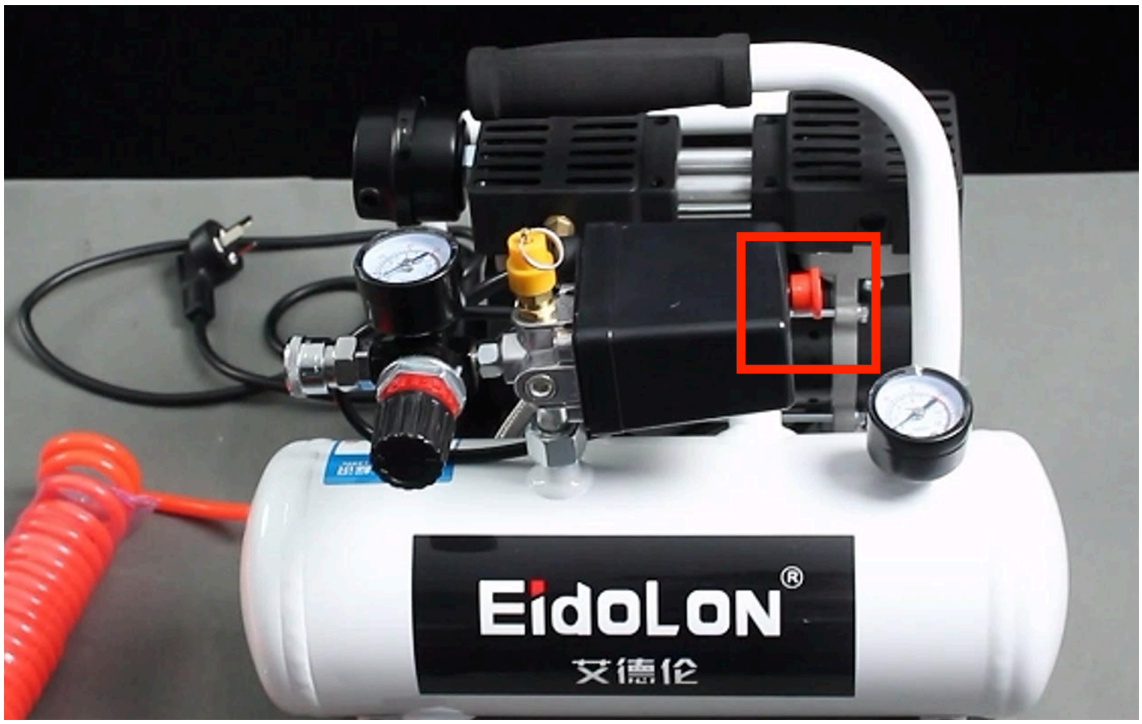
- It needs to be used with an air compressor:



1. Insert the black plug into the row of plugs;
2. Insert the matching red hose into the connector on the machine:



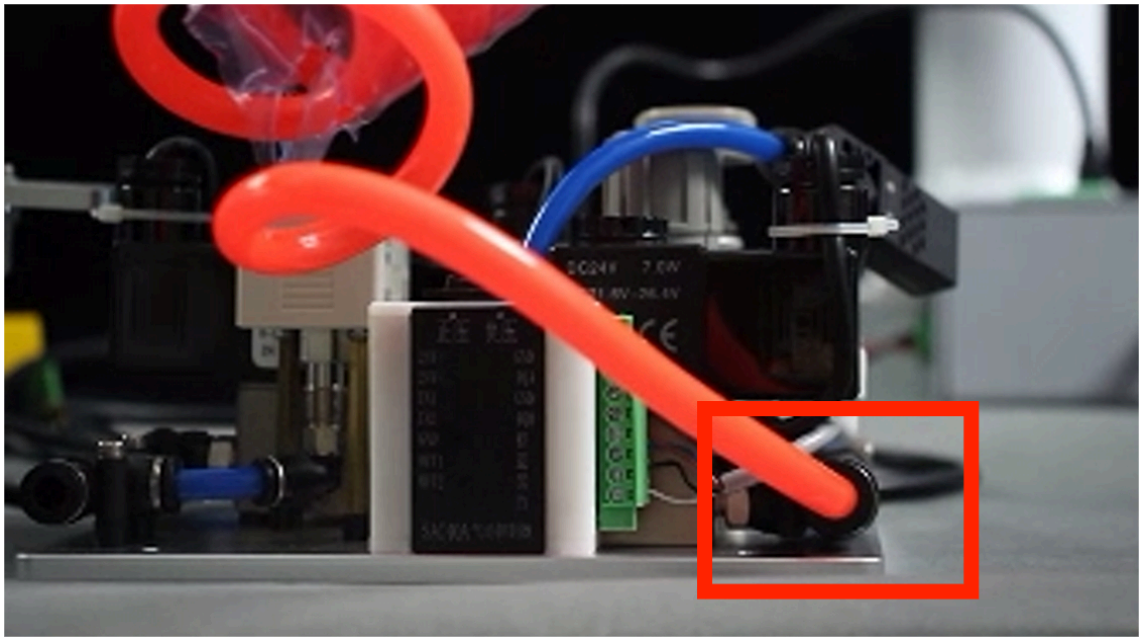
3. The red button is the on/off switch, pulling it outwards turns it on, pressing it back turns the machine off:



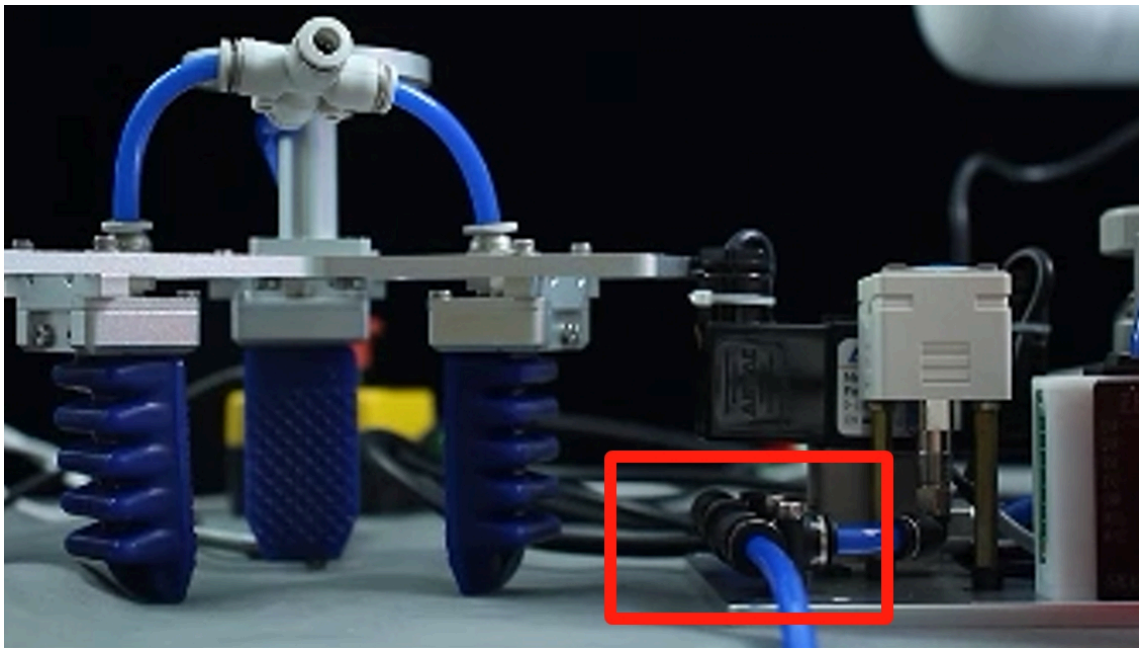
- gripper mounted:

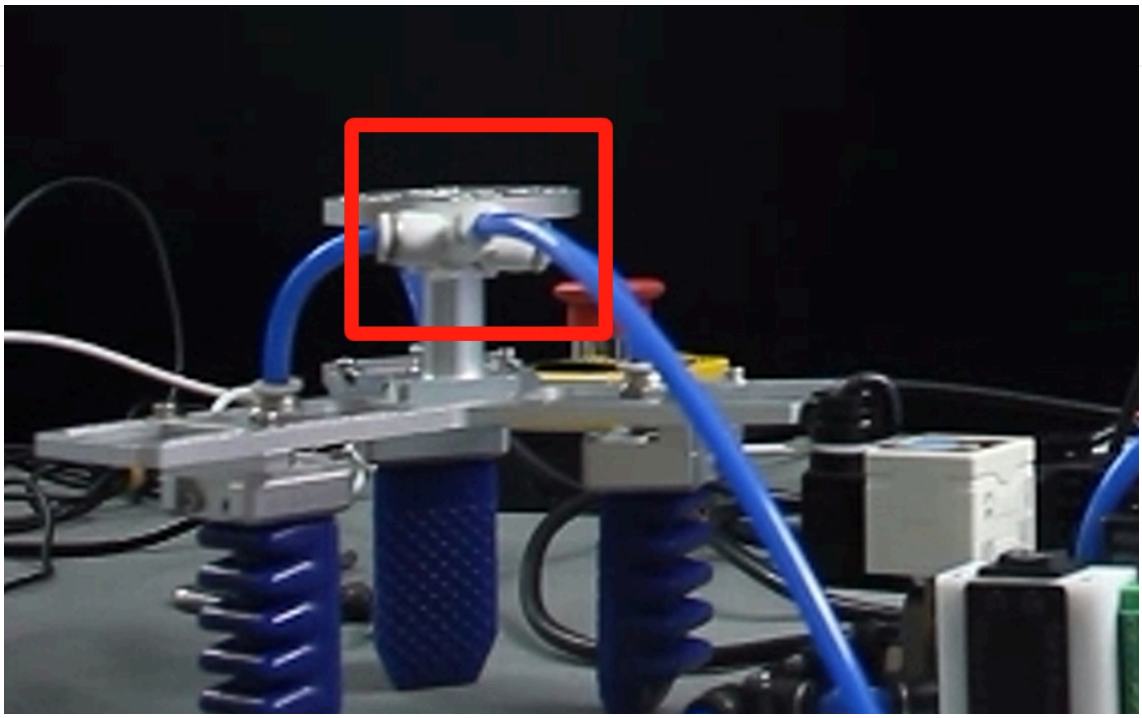
  </video>

1. Connect the other end of the red hose from the air compressor to the connector on the pneumatic controller:

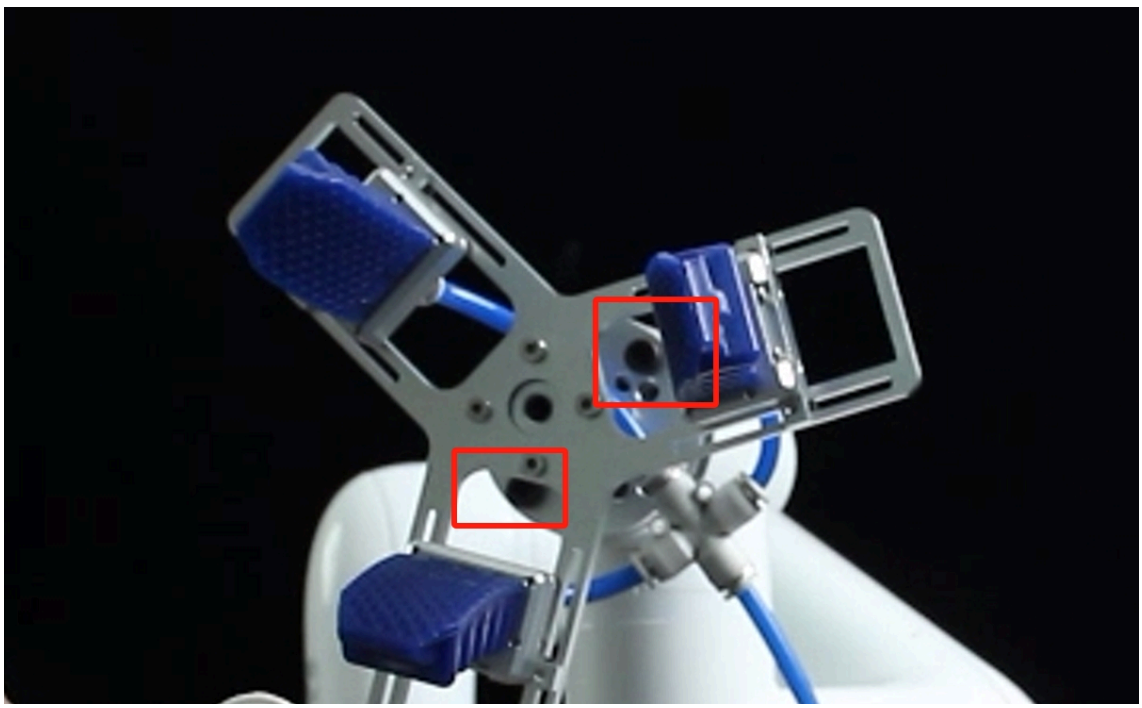


2. Use the blue hose that comes with the flexible gripper to connect the gripper to the pneumatic controller:





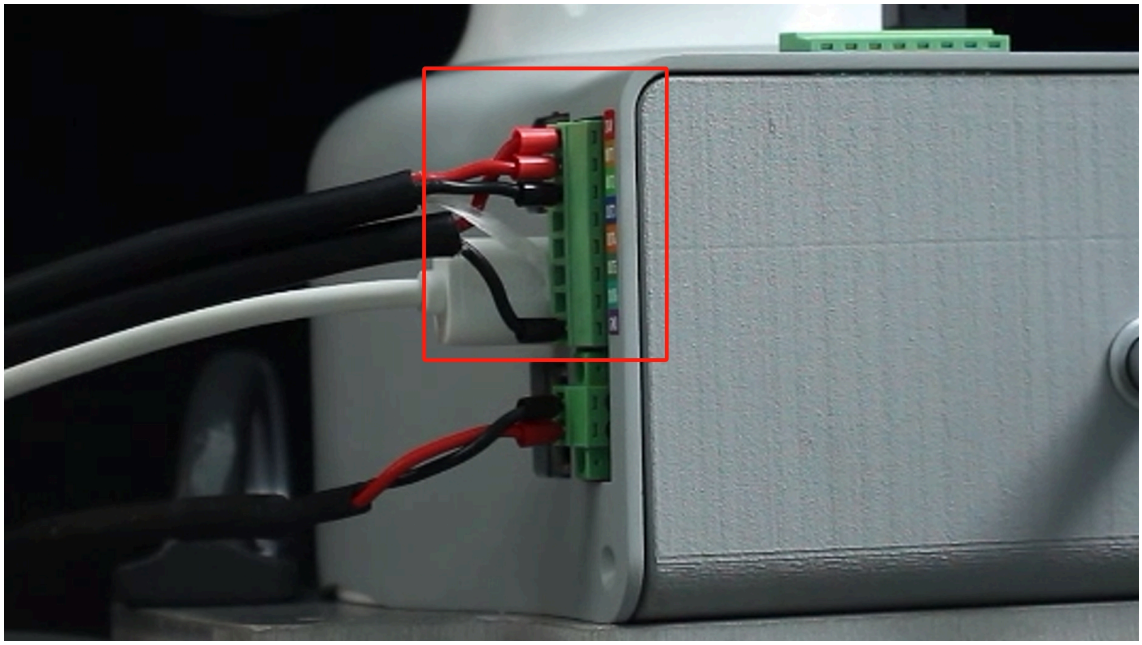
3. Attach the flexible gripper to the end of the arm with the matching screws:



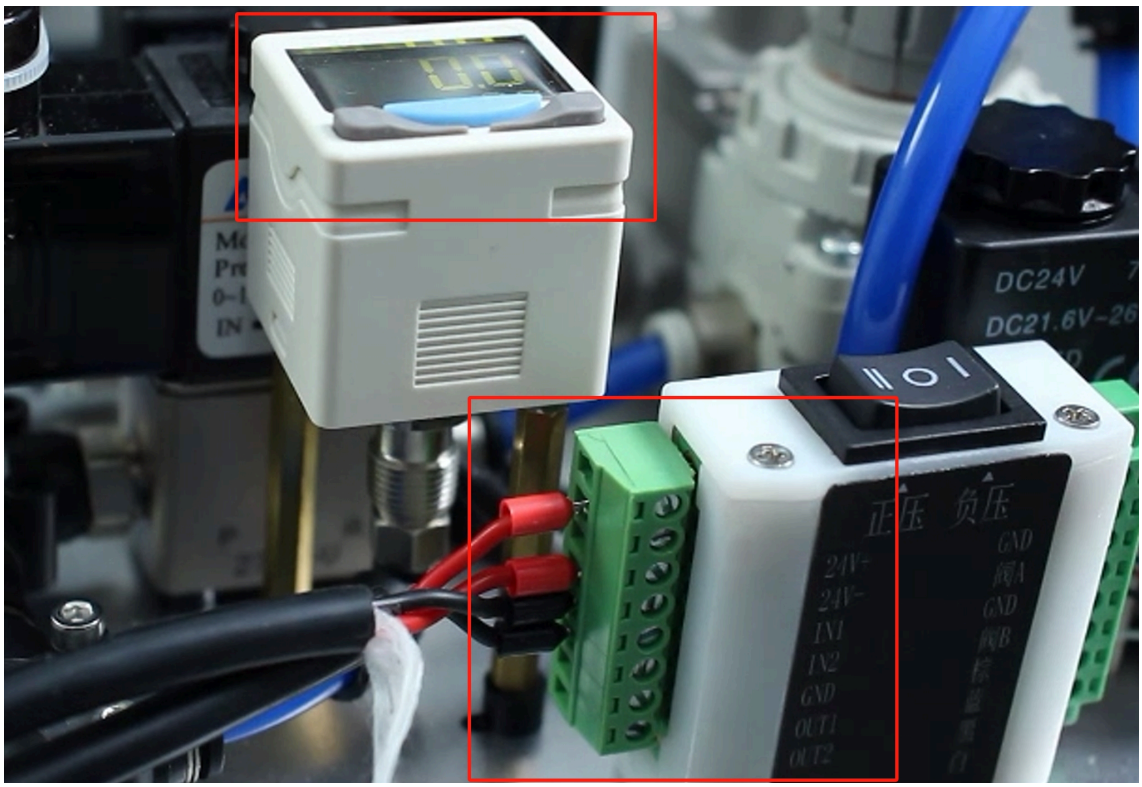
- Electrical Connections:

  1. Two connection cables are issued, one for power and one for control.

2. At the base of the robot arm, for the power supply, connect the red wire to the 24V connector and the black wire to the GND connector. For the control wires, connect red to OUT1 and black to OUT2:



3. Pneumatic controller terminal, power supply wire red connects to pneumatic controller 24V, black connects to pneumatic controller GND, control wire red connects to IN1, black connects to IN2:



> Be careful to connect to the "Positive Voltage" side, if the power supply is successful the display will light up. You can test the connection manually to see if it is working properly by switching the air compressor on and pressing the button on the pneumatic controller, pressing it to the left (positive pressure) will contract the gripper and pressing it to the right (negative pressure) will open the gripper.

- Software-driven testing:

> To test if the gripper are available after installation, use myBlockly. myblockly download

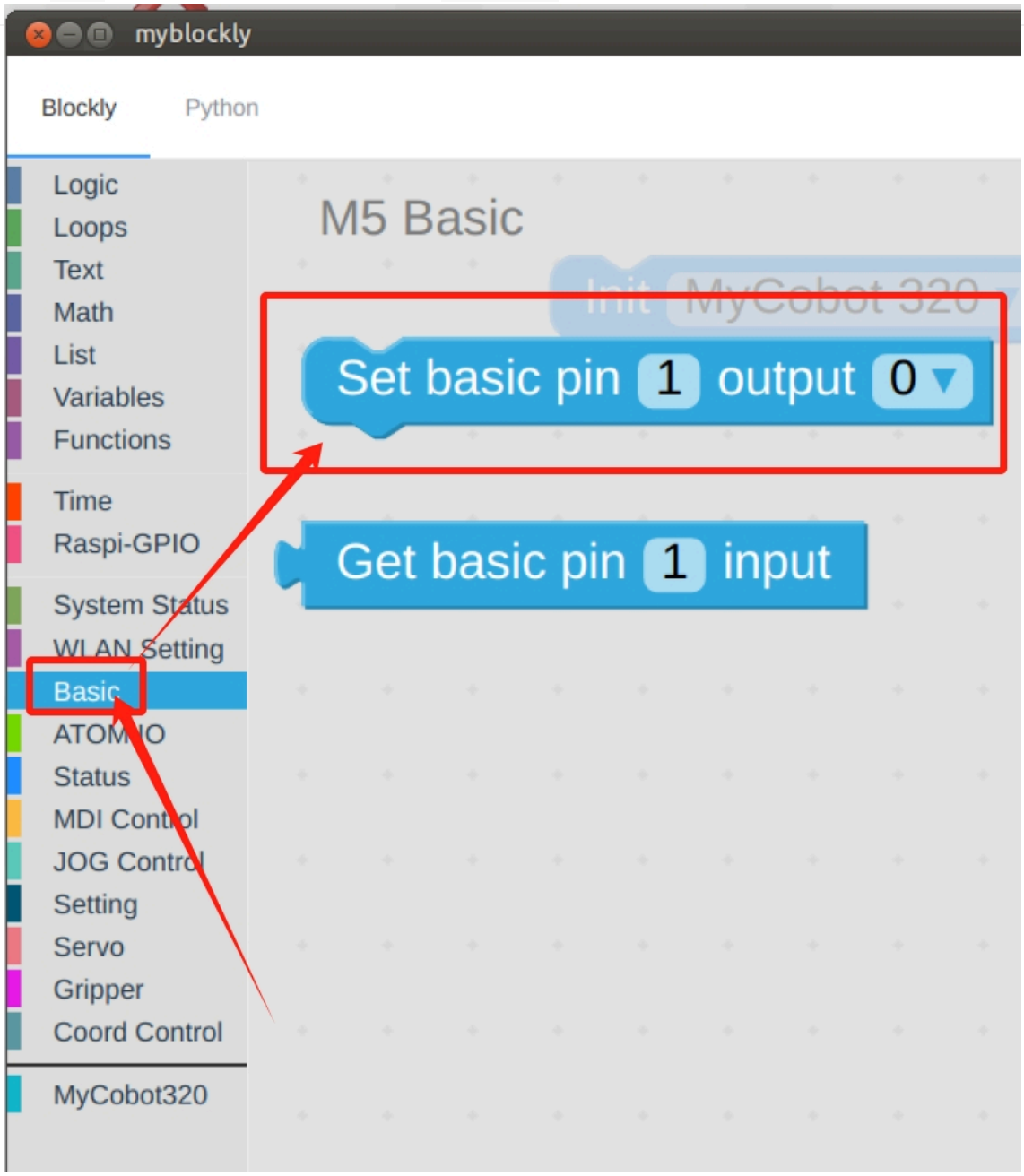
1. After confirming that the structural and electrical connections are complete, start the arm and open the myblockly software when the graphical interface appears.



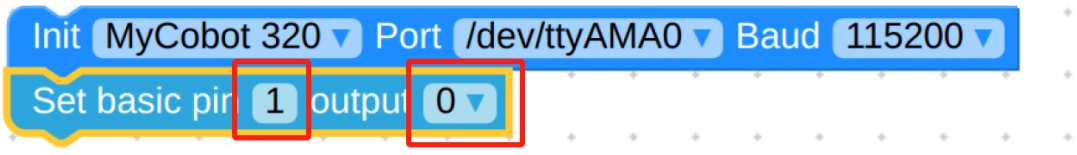
2. Modify the baud rate to 115200:



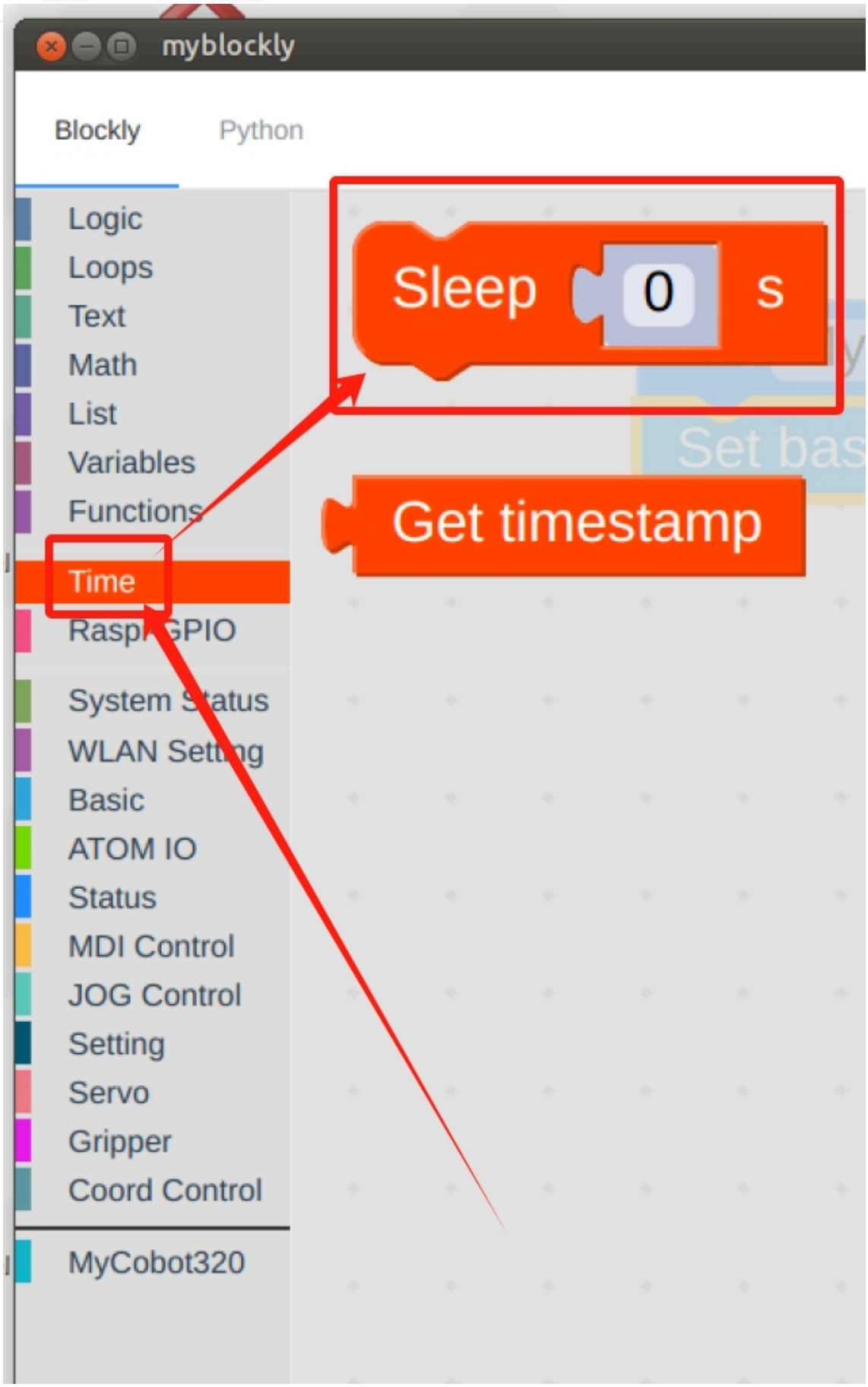
3. Find `Base` in the list on the left and select the `Set Pin Out` module:



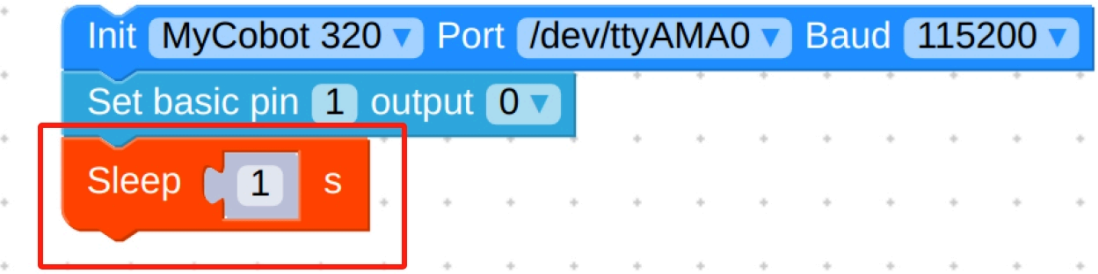
4. Set `pin number` to `1` and `output` to `0` :



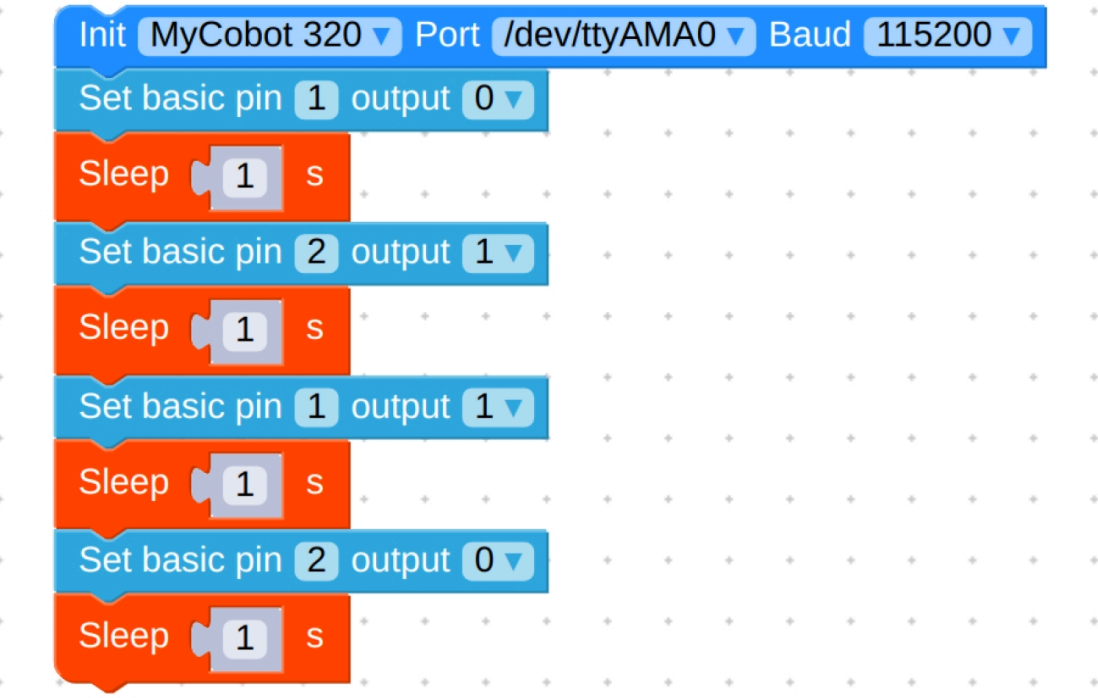
5. Find `Time` and select the `Sleep` module:



6. Set the time as desired, here it is set to `1s` :



7. Repeat the above steps for the final setup as follows:



8. Final code:

```
from pymycobot.mycobot import MyCobot
import time


mc = MyCobot('/dev/ttyAMA0', 115200)
mc.set_basic_output(1, 0)
time.sleep(1)
mc.set_basic_output(2, 1)
time.sleep(1)
mc.set_basic_output(1, 1)
time.sleep(1)
mc.set_basic_output(2, 0)
time.sleep(1)
```

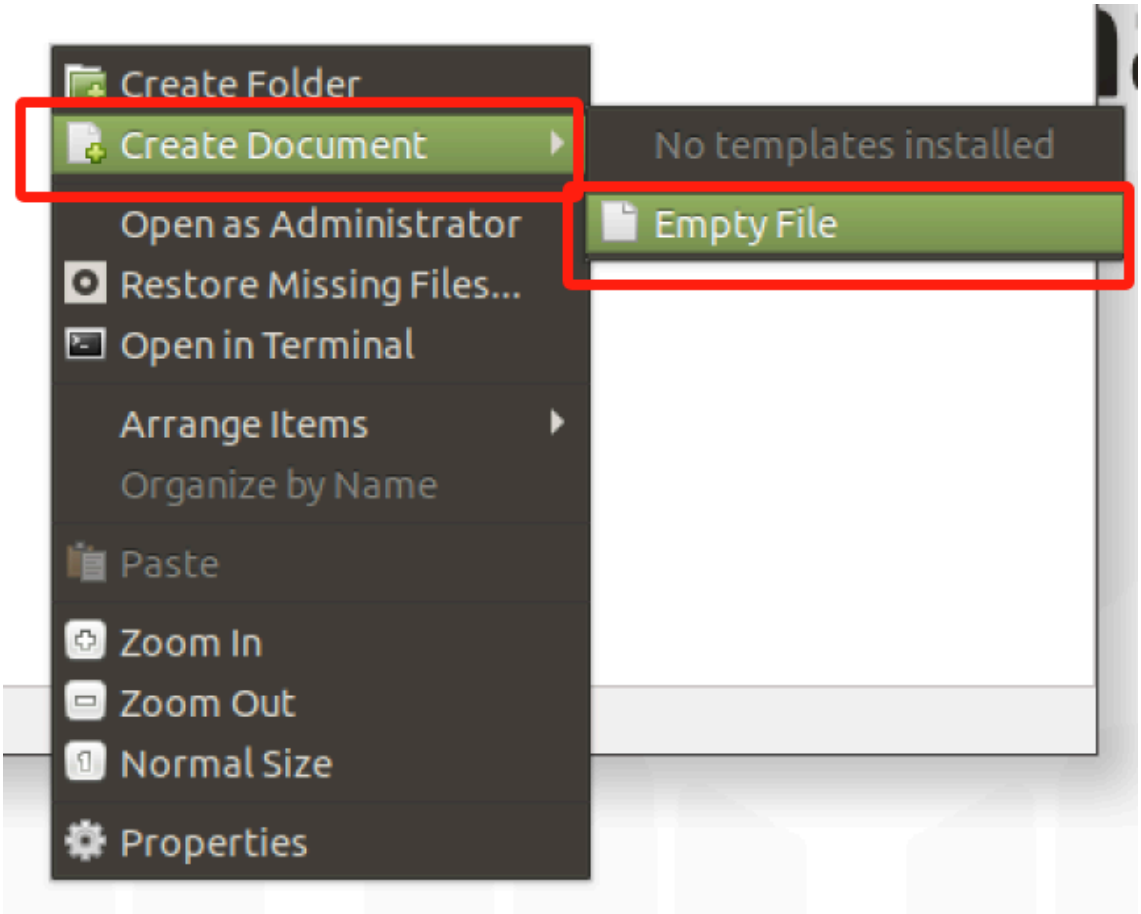
9. Click on the green run button in the top right corner to see the gripper close-open once.

- Programming Development:

> Programming the gripper using python python environment download

1. Create a new python file:

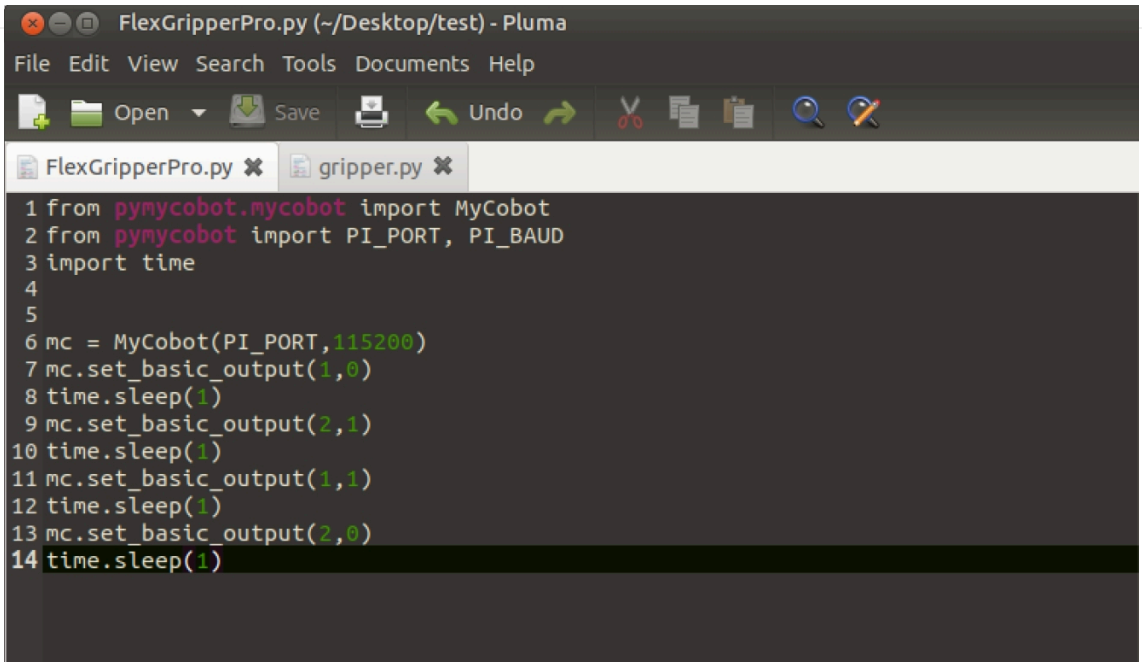
Right click on the desired file path to create a new python file:



> The file name can be changed as needed



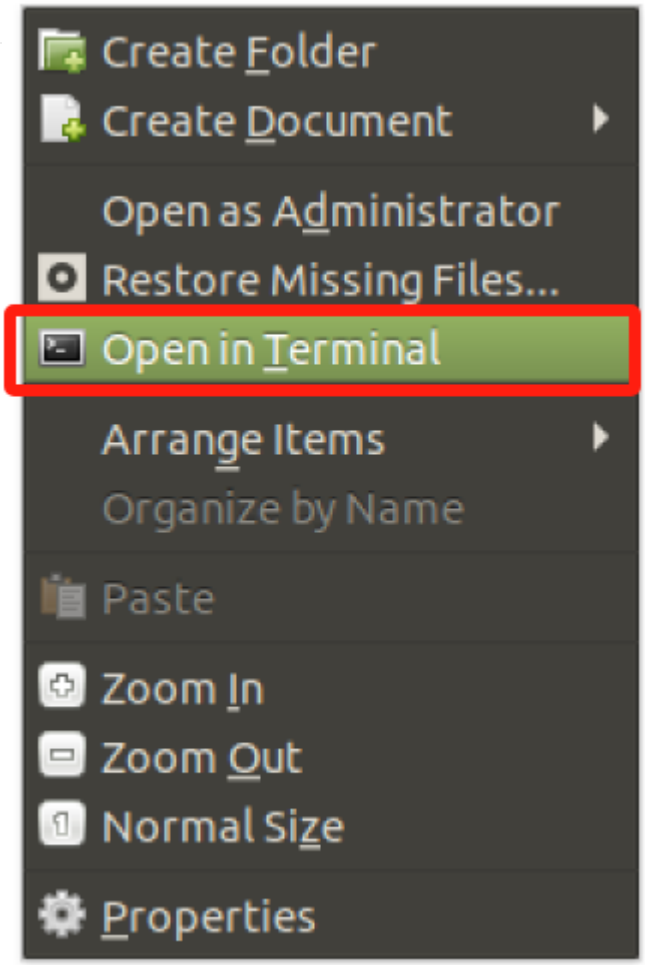
2. Perform function programming:



> The code is as follows:

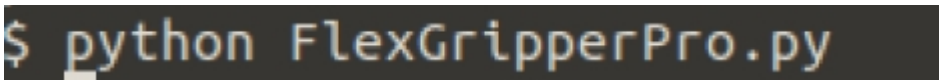
```
from pymycobot.mycobot import MyCobot
import time


# Initialise a MyCobot object
mc = MyCobot("COM3", 115200)


# Controls gripper closed-open:
# Using the gripper status interface 0 is open, 1 is closed
mc.set_basic_output(1, 0)
time.sleep(1)
mc.set_basic_output(2, 1)
time.sleep(1)
mc.set_basic_output(1, 1)
time.sleep(1)
mc.set_basic_output(2, 0)
time.sleep(1)


# For more information on using the interface, see the python API.
```

3. Save the file and close it, right-click on an empty space in the folder to open a command line terminal



Input:

```
python FlexGripperPro.py
```



> gripper can be seen closed-open

← Accessories Tools Page

Module Suction Cup

> **Compatible models:** myCobot 320, myCobot Pro 630

Product Icon



Specifications

name	Module suction cup
model	myCobotPro_suctionPump
Material	Nylon 7100
Number of suction cups	1/2/4
Suction cup size	diameter 33mm
absorb weight	Maximum 1000g
Power source equipment	Suction and blowing integrated air compressor
service life	one year
a fixed way	screw fixed
control interface	I/O control
Use environment requirements	Temperature and pressure
Applicable equipment	ER myCobot 320 series, ER myCobot Pro 600 series

Use for Objects

Introduction

- Suction cup suction pump is that the suction port is connected with the object to be adsorbed through suction cups, tubes and other components, and vacuumizes the suction cup, causing the internal air pressure to change from normal pressure to negative pressure. The pressure difference between the external atmospheric pressure and this negative pressure is used to achieve adsorption. The purpose of living objects.
- The overall structure is a multi-functional expandable suction cup, which can be installed up to 4; high-pressure resistant hose, which can be reused many times; an air compressor with suction and blowing integrated, which can be expanded to other functions.

working principle

- Start the suction of the vacuum equipment to generate negative air pressure in the suction cup, so that the object to be lifted can be sucked firmly, and the object to be lifted can be started to be transported.
- When the object to be lifted is transported to the destination, inflate it smoothly into the vacuum suction cup, so that the negative air pressure in the vacuum suction cup becomes zero or slightly positive air pressure, and the vacuum suction cup is separated from the object to be lifted, thus completing the lifting and conveying of heavy objects task.

Applicable objects : suitable for objects with flat surfaces

Mall link:

- Taobao

- shopify

How to use



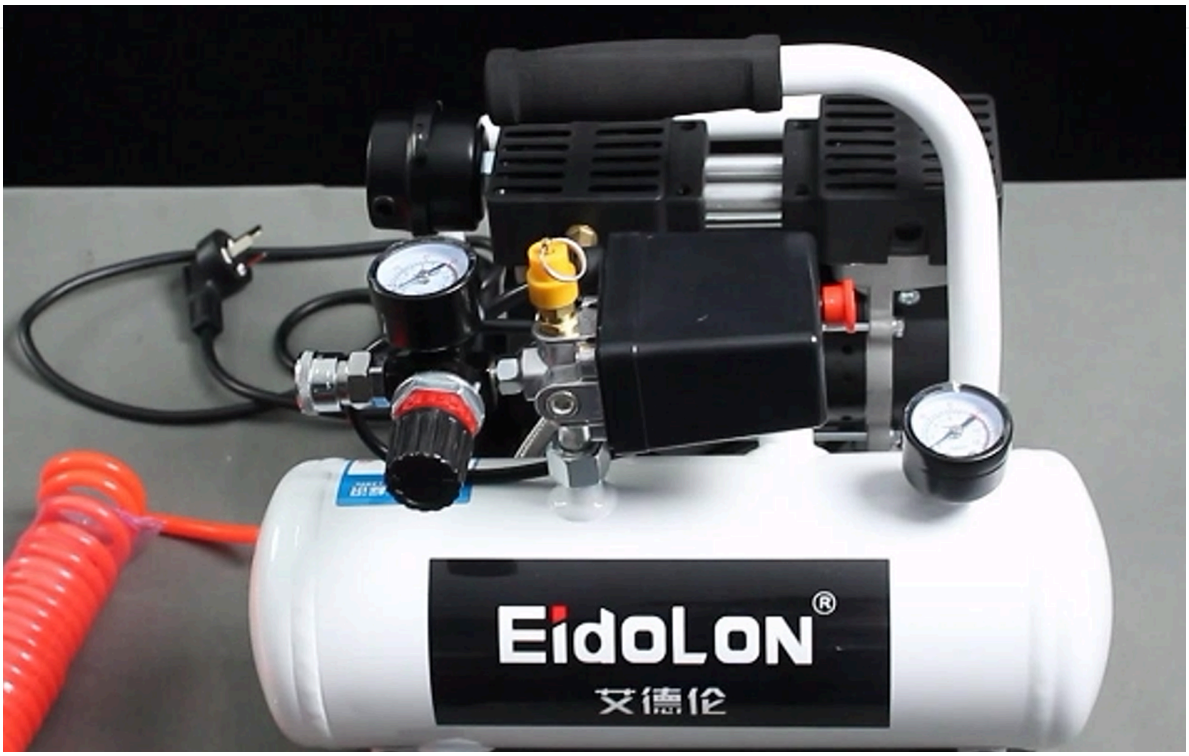
For a system incorporating a suction cup, solenoid valve, and vacuum pump, the connection typically works as follows:

1. **Vacuum Pump**: This is the source of vacuum pressure. It should be connected to the solenoid valve's inlet port.
2. **Solenoid Valve**: Acts as a control mechanism that regulates when and how the vacuum is applied or released to the suction cup. The vacuum pump connects to one side of the solenoid valve, and the suction cup connects to the outlet side.
3. **Suction Cup**: This is the end-effector that directly interacts with the objects to be manipulated. It connects to the outlet side of the solenoid valve.

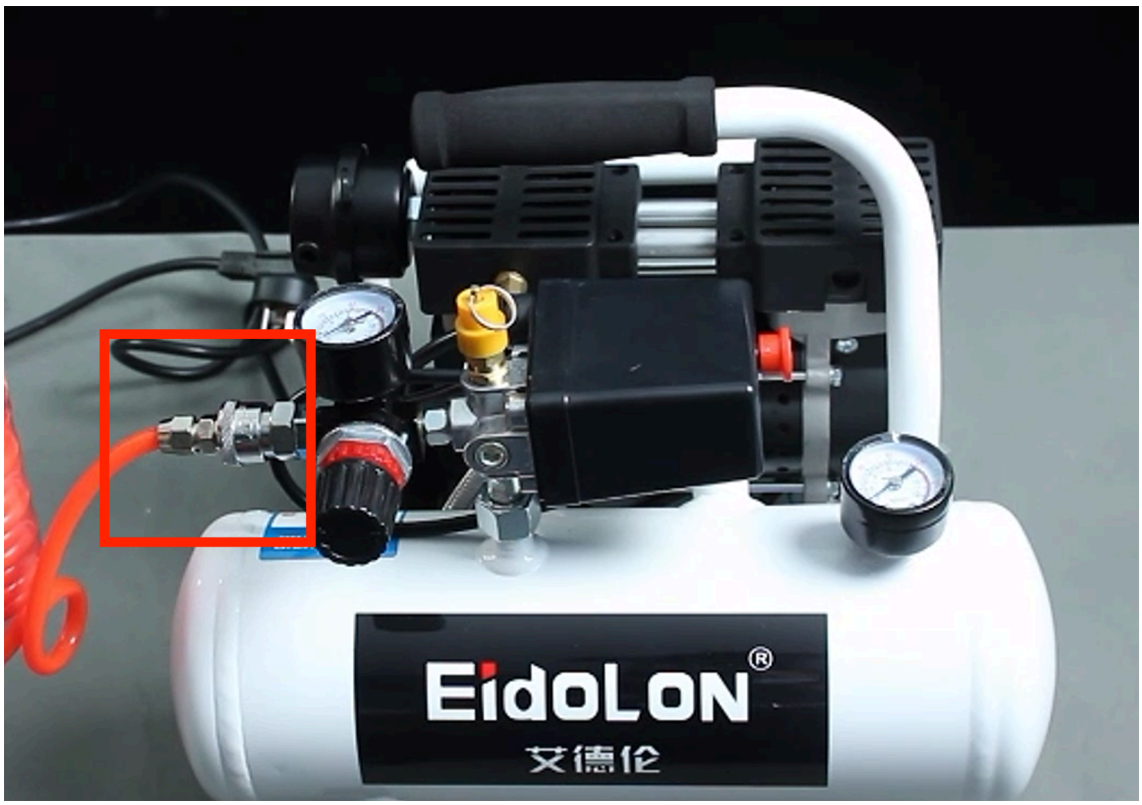
In operation, when the solenoid valve is energized, it opens, allowing vacuum pressure from the pump to reach the suction cup, enabling it to grip objects. De-energizing the valve cuts off the vacuum, allowing air to flow back to the suction cup, releasing the grip on the object. This setup is efficient for automation systems requiring precise control over picking and placing items.

Installation and use

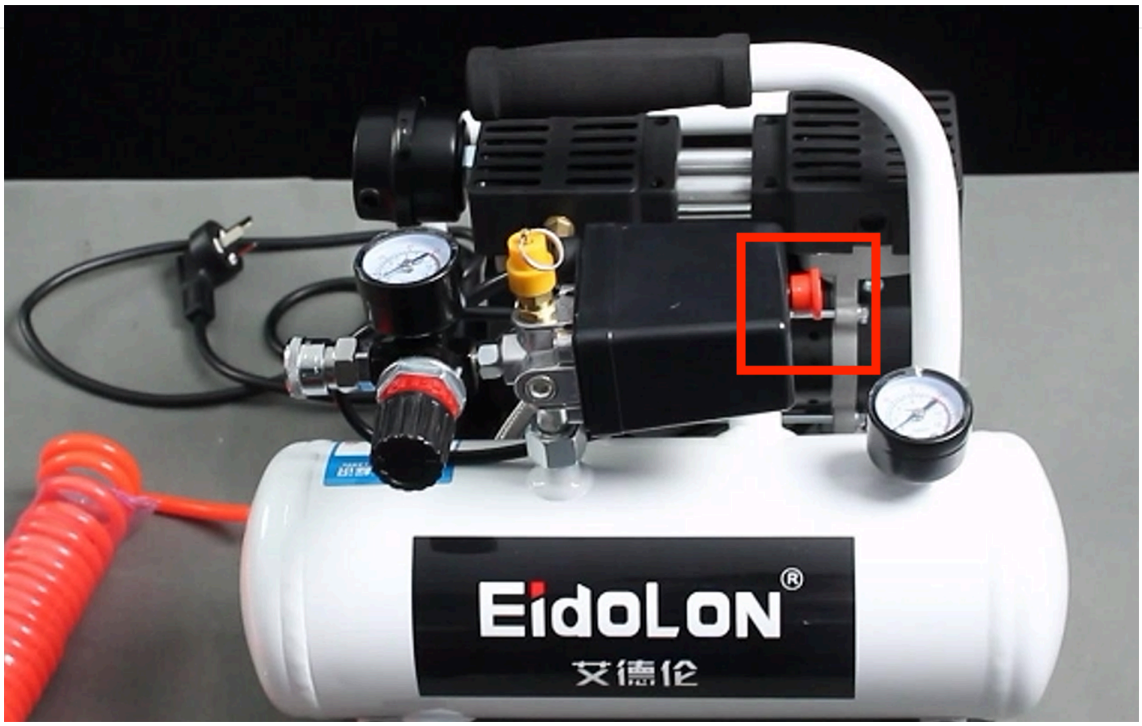
- It needs to be used with an air compressor:



1. Insert the black plug into the row of plugs;
2. Insert the matching red hose into the connector on the machine:

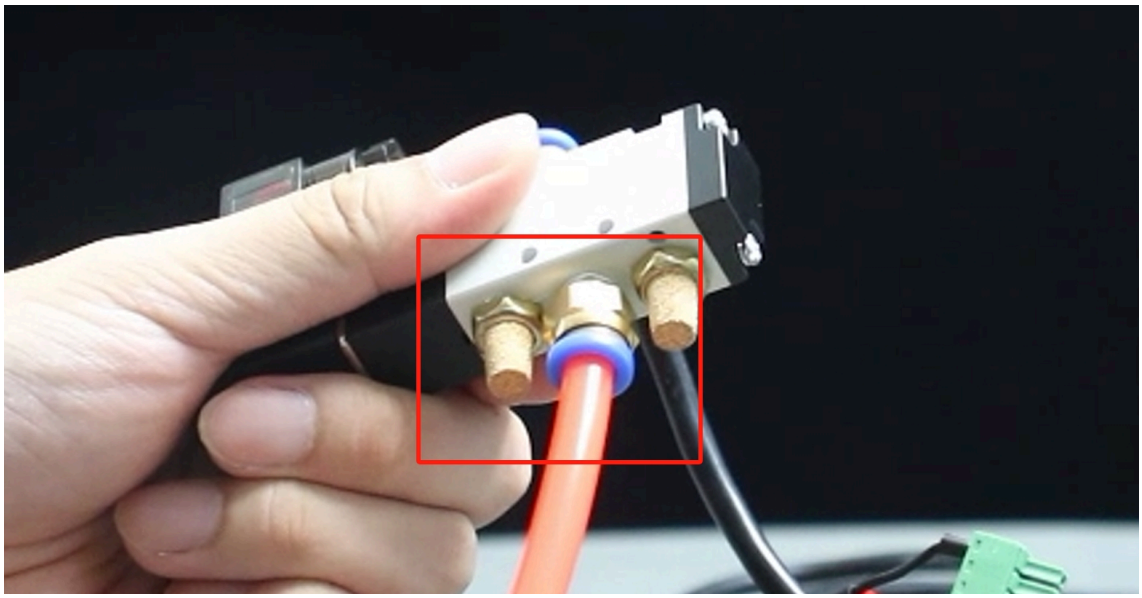


3. The red button is the on/off switch, pulling it outwards turns it on, pressing it back turns the machine off:

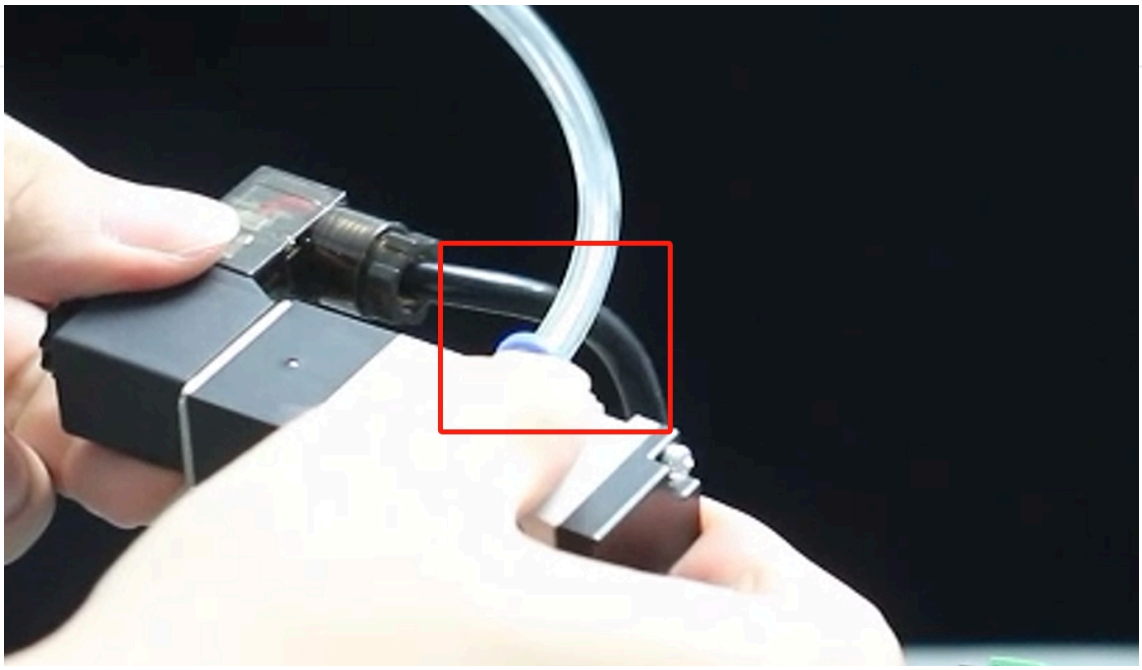


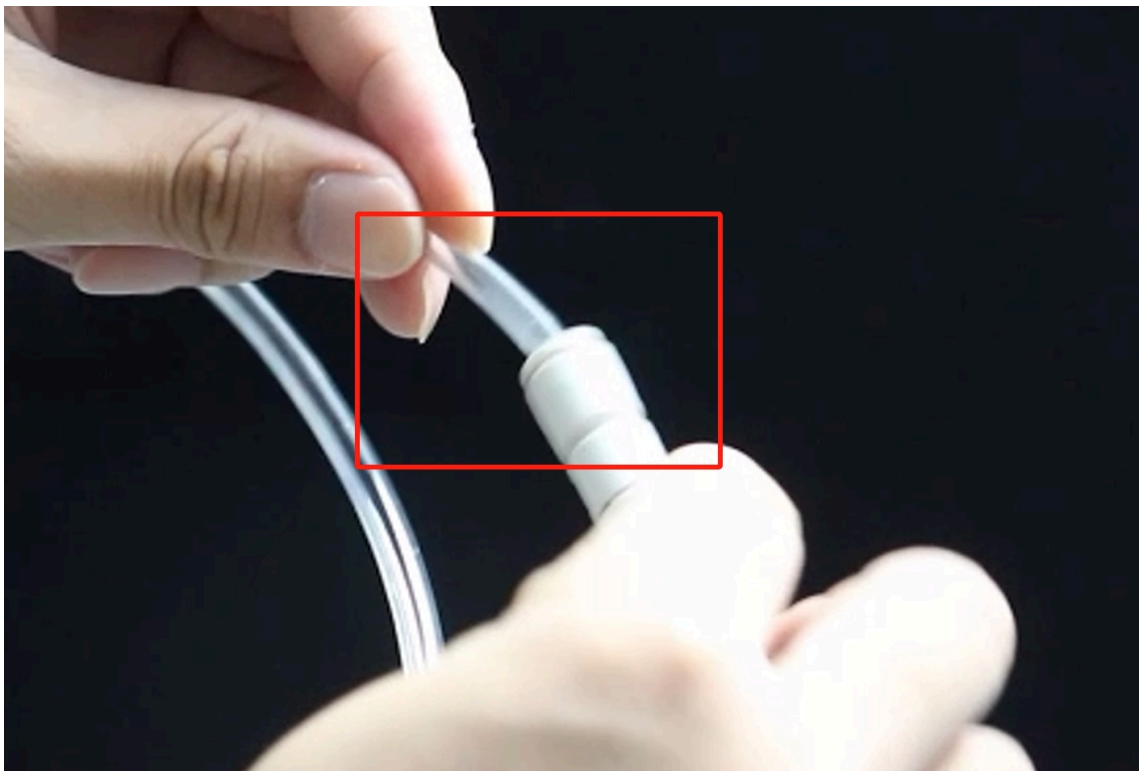
- Suction cup mounting:

1. Connect the other end of the red hose from the air compressor to the solenoid valve connection:

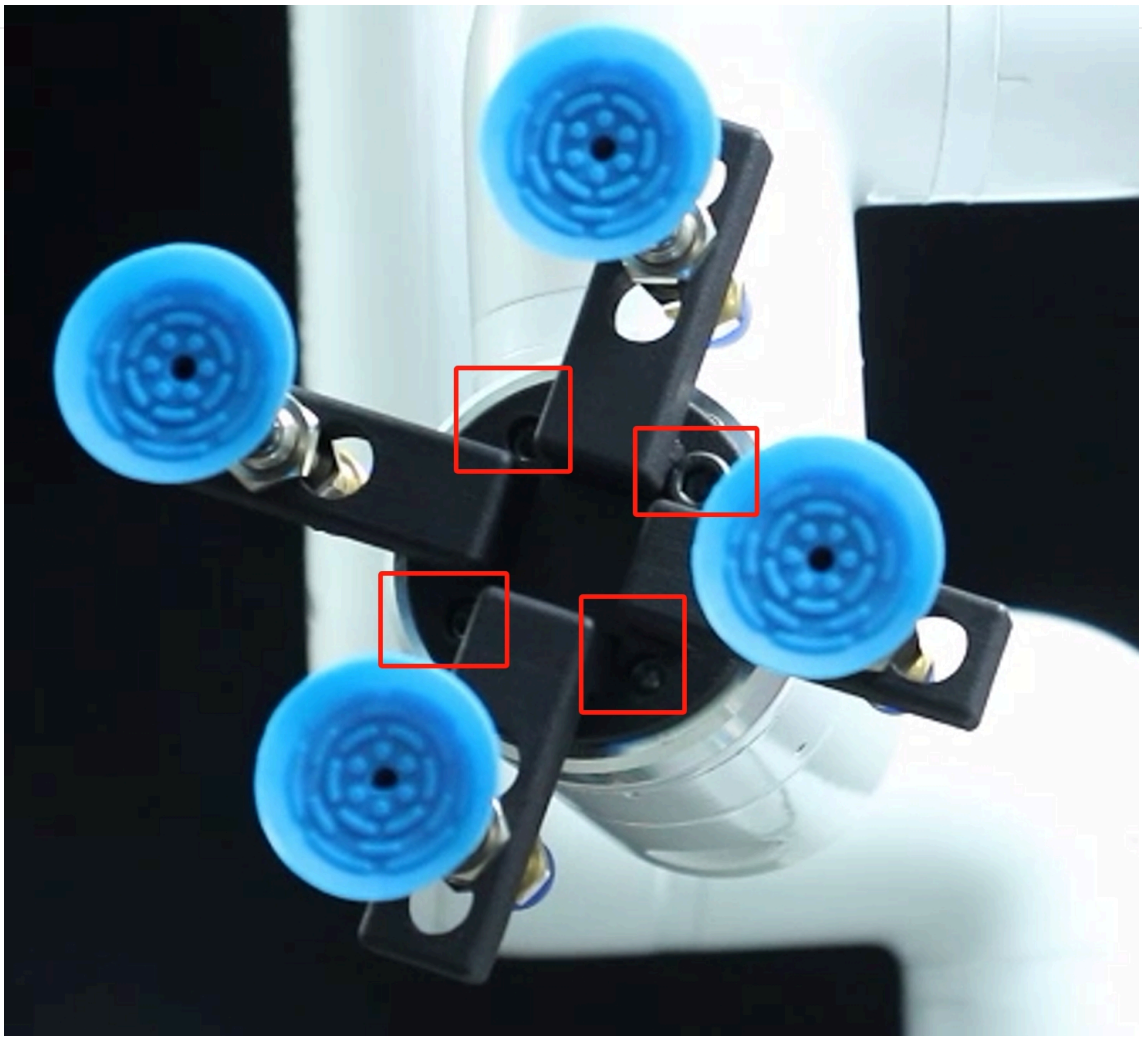


2. Use the clear hose that comes with the modular suction cup to connect to the connector above the solenoid valve and the hose branch of the modular suction cup respectively:

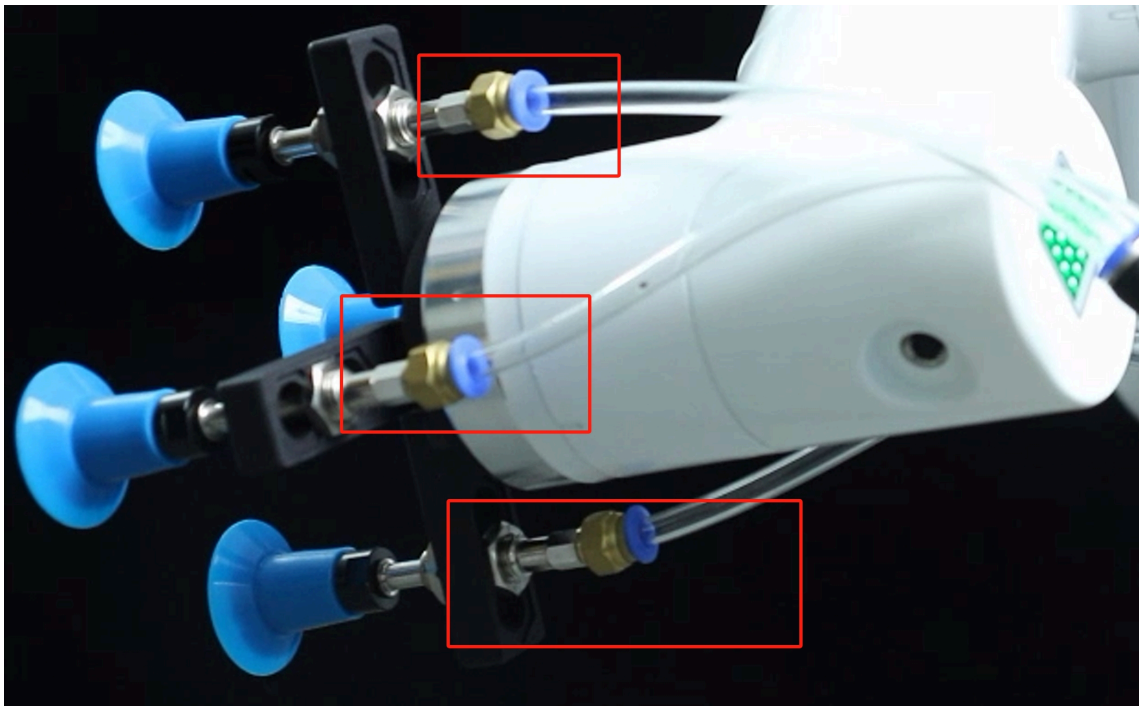




3. Secure the module suction cup to the end of the robot arm with the matching screws:

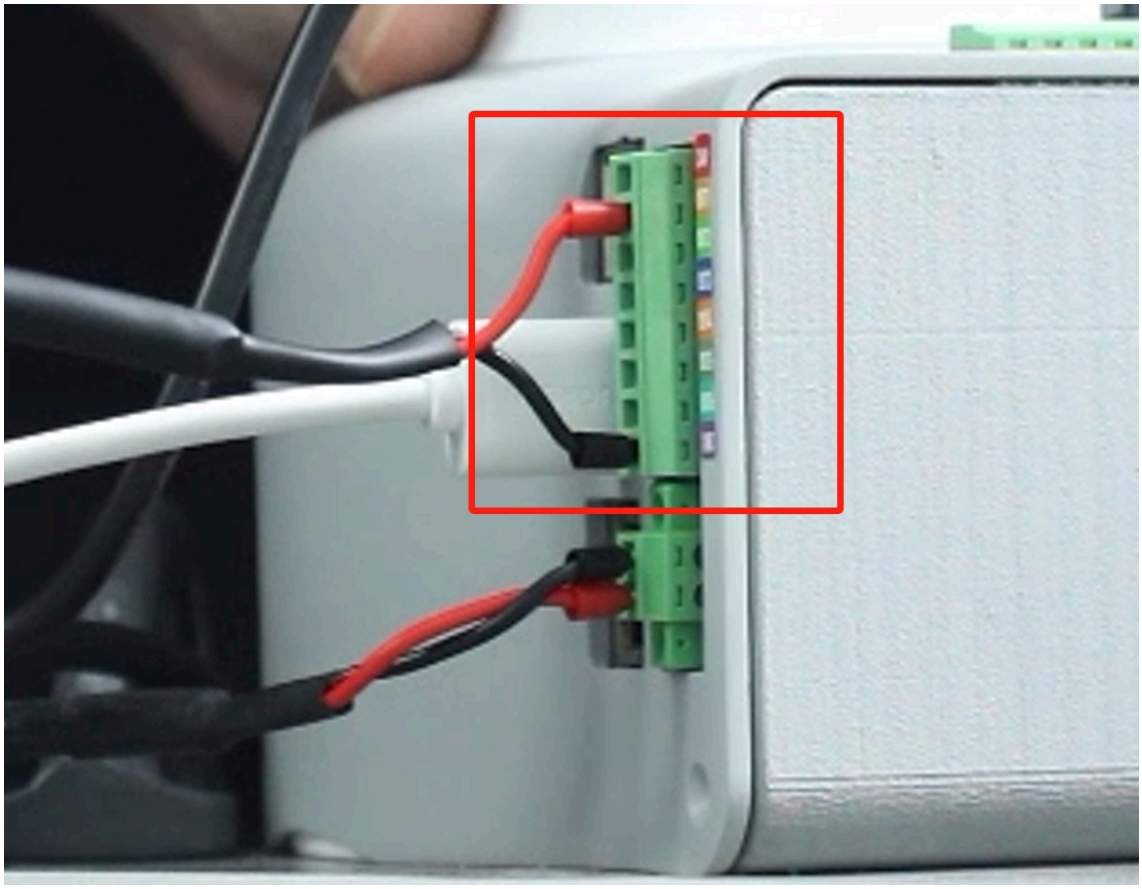


4. Attach the hose branch to the modular suction cup:



- Electrical Connections:

1. Connect the black wire to the GND of the robot arm base, and the red wire to any one of OUT1~OUT6, and change the pin number of the subsequent programme according to the selected interface, here we use OUT1:



- Software-driven testing:

> To test if the gripper are available after installation, use myBlockly. myblockly download

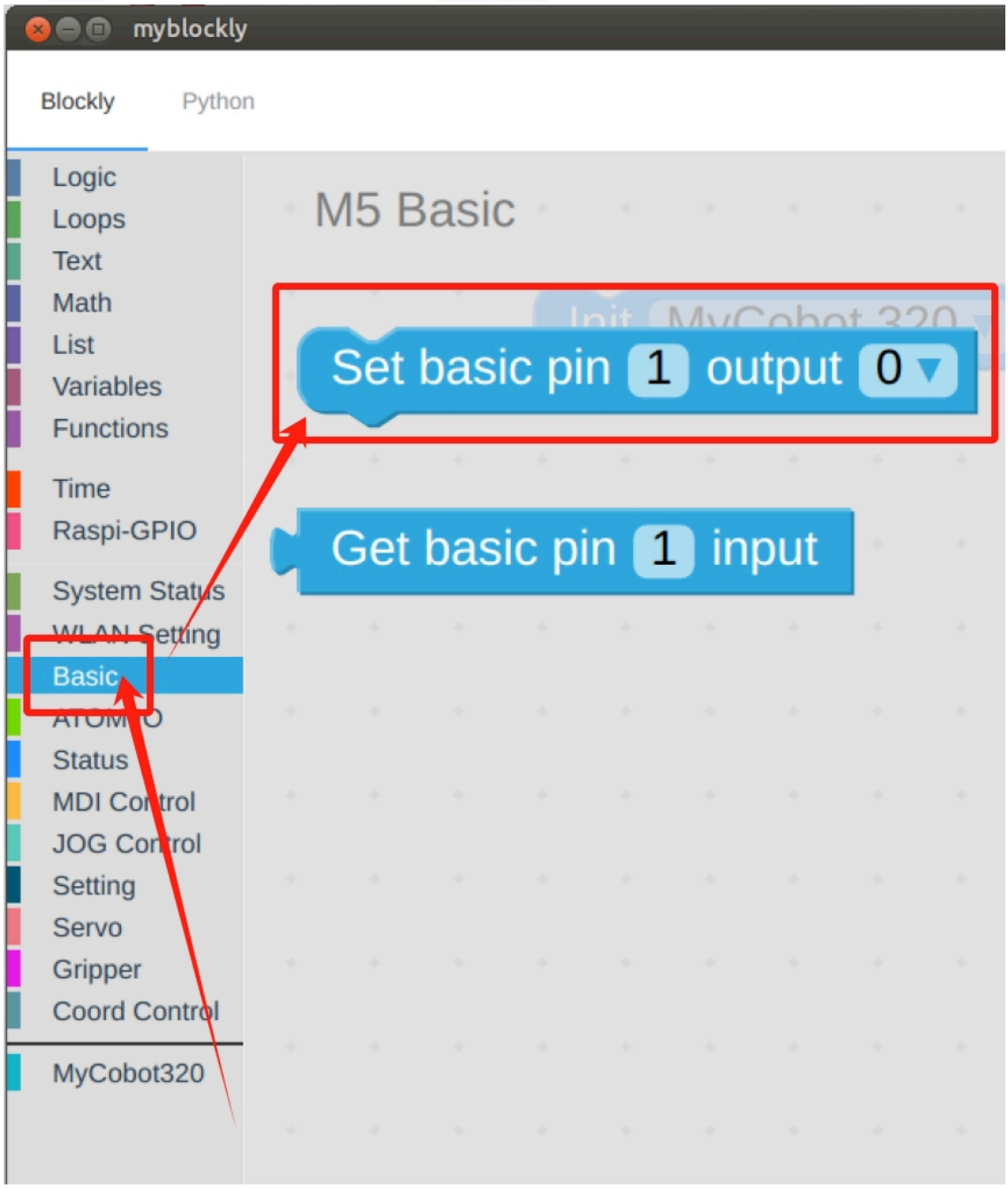
1. After confirming that the structural and electrical connections are complete, start the arm and open the myblockly software when the graphical interface appears.



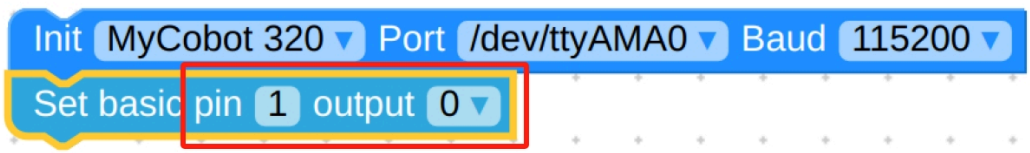
2. Modify the baud rate to 115200:



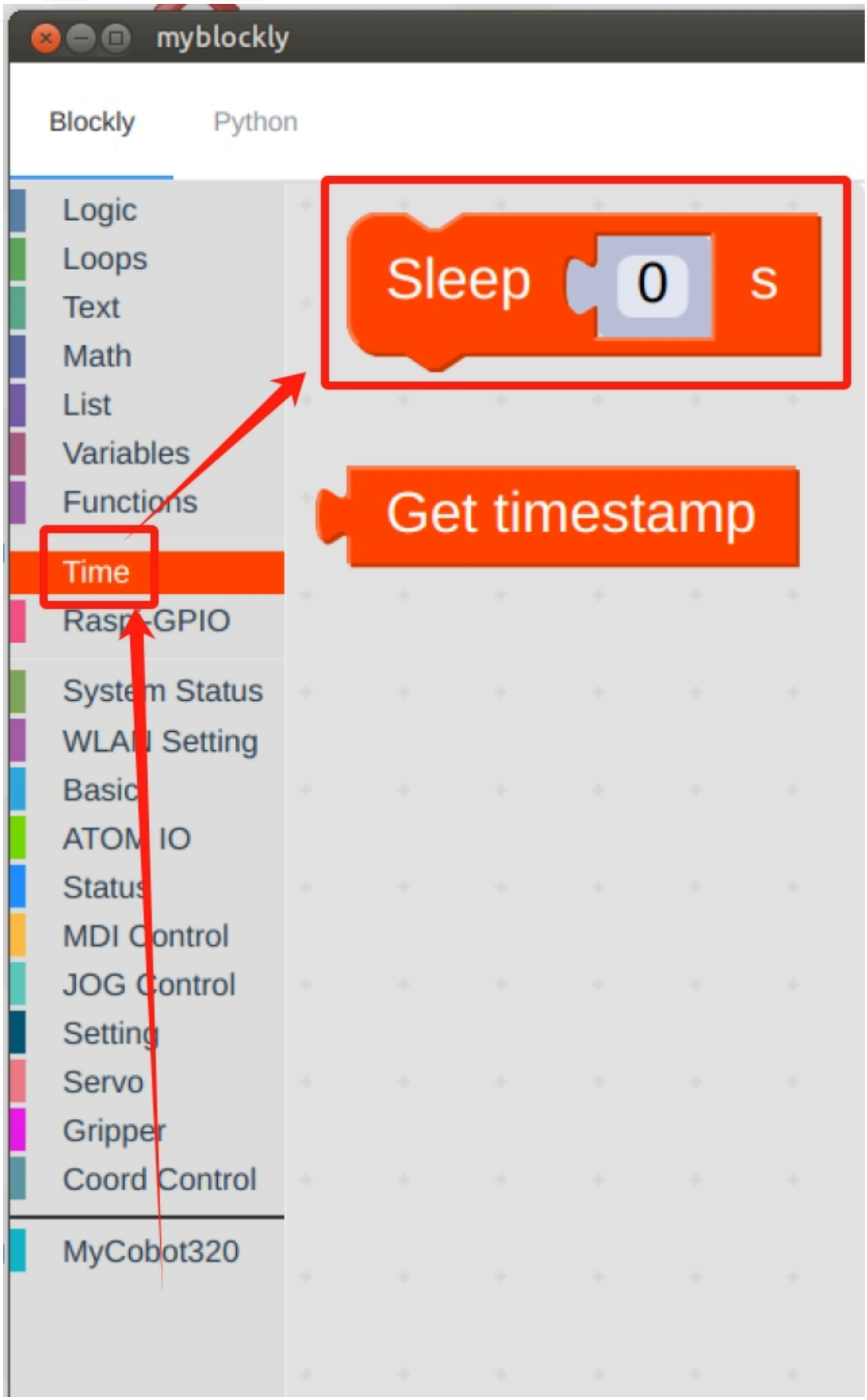
3. Find `Base` in the list on the left and select the `Set Pin Out` module:



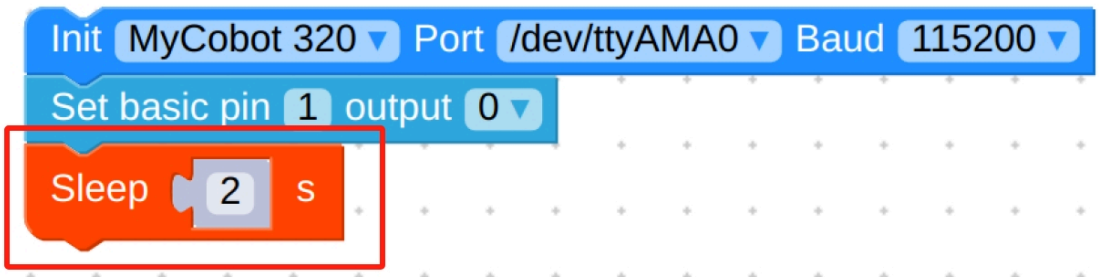
4. Set `pin number` to `1` and `output` to `0` :



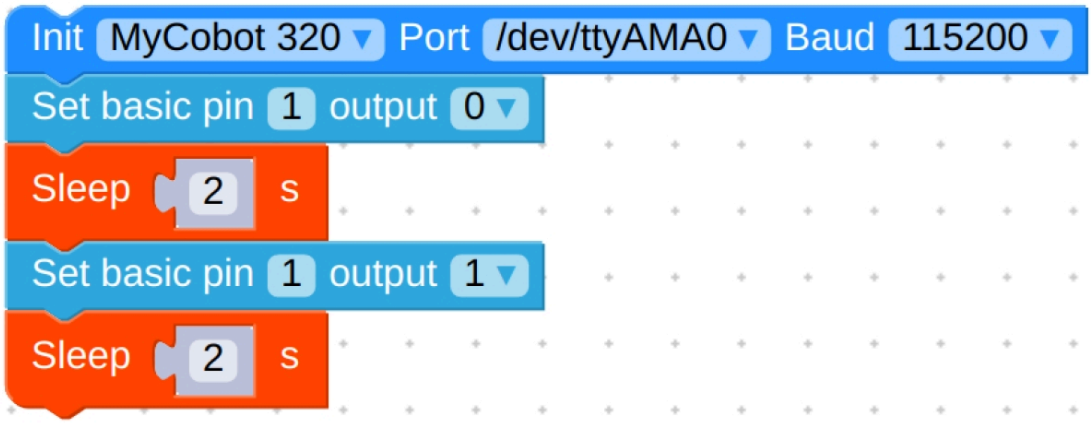
5. Find `Time` and select the `Sleep` module:



6. Set the time as desired, here it is set to `2s` :



7. Repeat the above steps for the final setup as follows:



8. Final code:

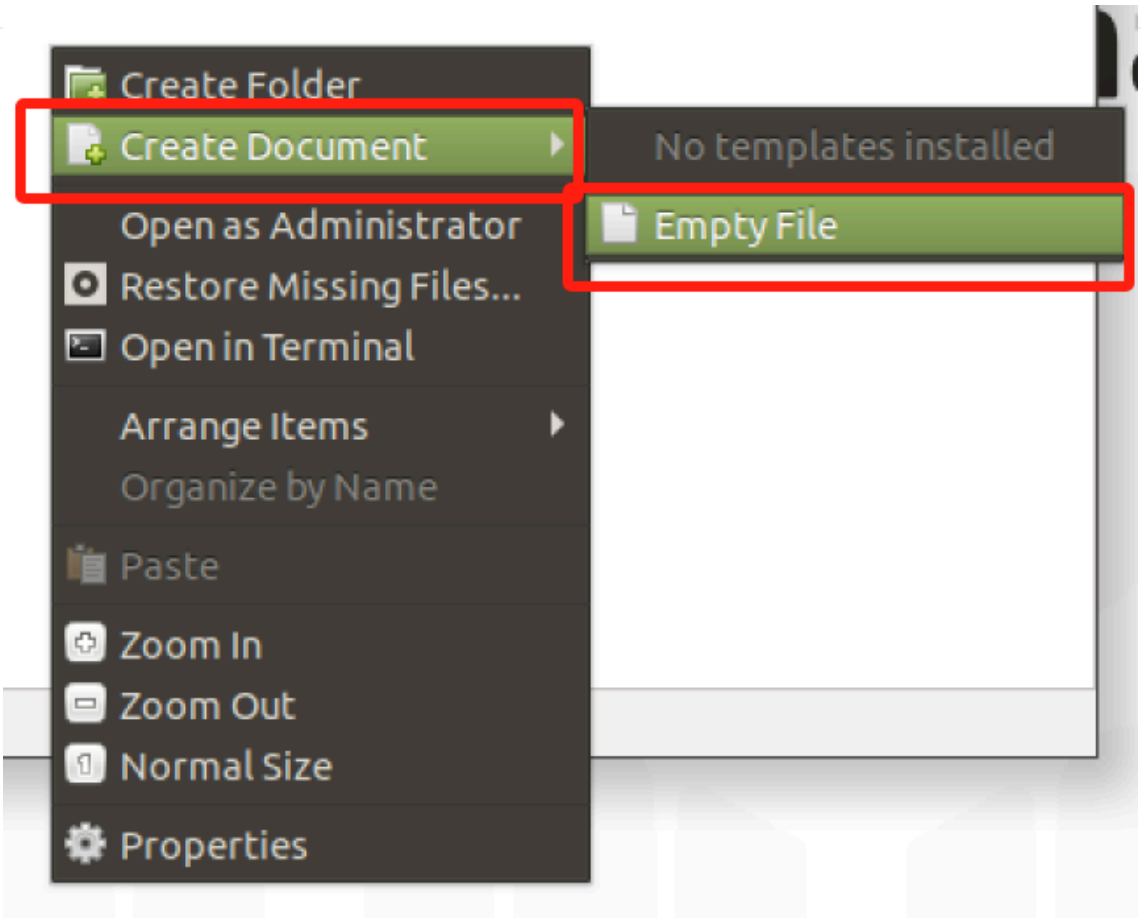
```
from pymycobot.mycobot import MyCobot
import time


mc = MyCobot('/dev/ttyAMA0', 115200)
mc.set_basic_output(1,0)
time.sleep(2)
mc.set_basic_output(1,1)
time.sleep(2)
```

9. Hold the suction cup against the object and click the green run button in the upper right corner to allow the suction cup to hold the object for 2s and then release it.

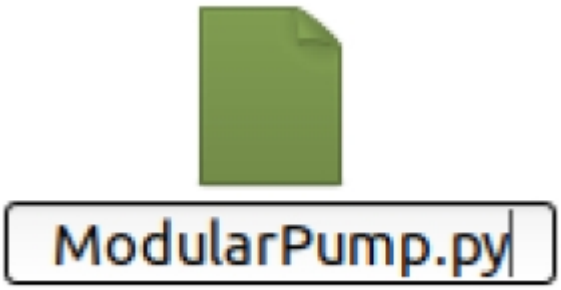
- Programming Development:

> Programming Suction Cups with python python environment download

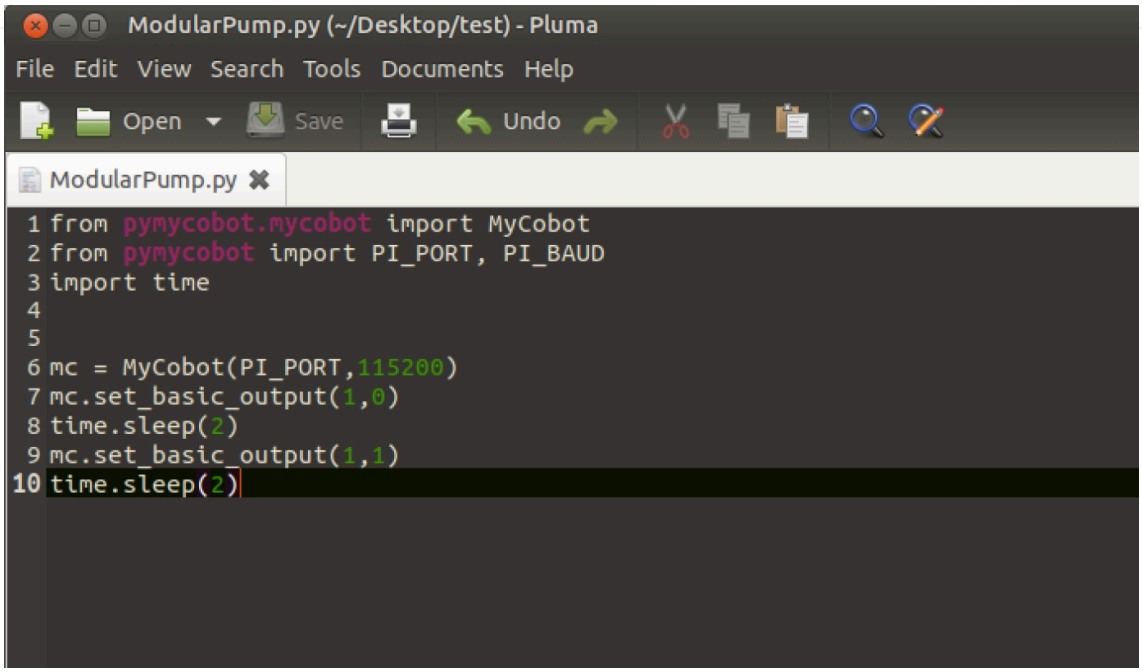
1. Create a new python file: Right click on the desired file path to create a new python file:



> The file name can be changed as needed



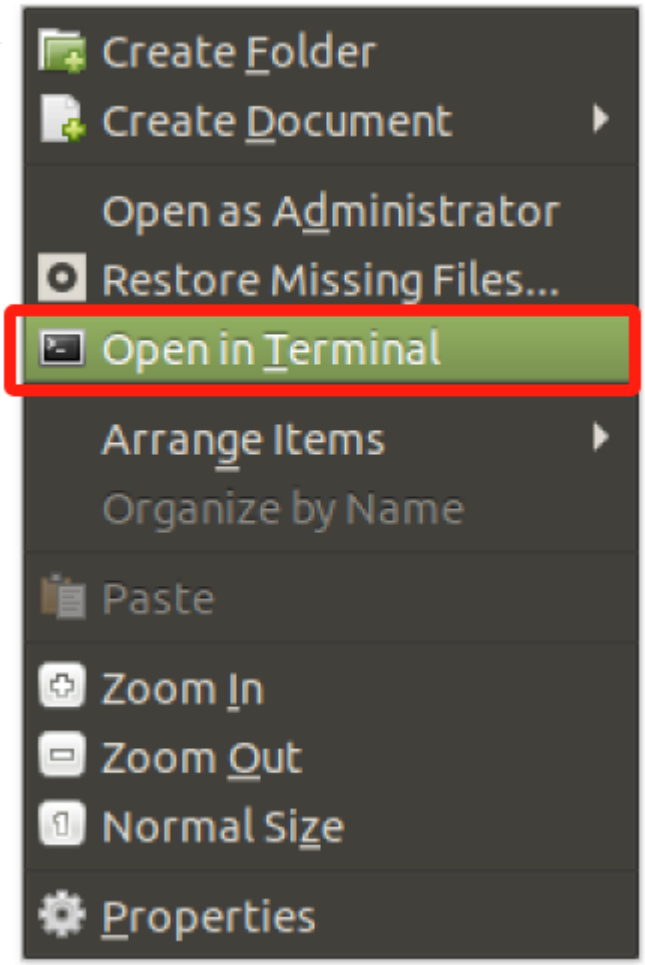
2. Perform function programming:



> The code is as follows:

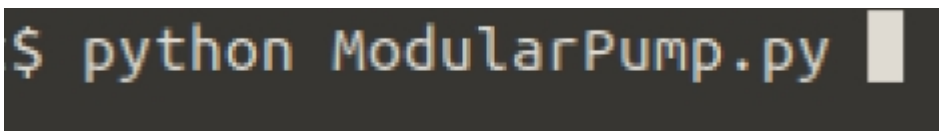
```
from pymycobot.mycobot import MyCobot
import time


# Initialise a MyCobot object
mc = MyCobot("COM3", 115200)


# Controls suction cups closed-open:
# Use suction cup status interface 0 for open, 1 for closed
mc.set_basic_output(1, 0)
time.sleep(2)
mc.set_basic_output(1, 1)
time.sleep(2)


# For more information on using the interface, see the python API.
```

3. Save the file and close it, right-click on an empty space in the folder to open a command line terminal



Input:

```
python ModularPump.py
```



> You can see the suction cups close-open

← Accessories Tools Page

myCobotPro Camera Module

> **Compatible models:** myCobot 320, myCobot 630

Product Icon



Specifications

name	myCobotPro camera module
model	myCobot_Pro_cameraHolder_J6
Material	photosensitive resin
resolution (of a photo)	1080P
thread length	1.5m
USB protocol	USB2.0 HS/FS
lens focal length	Standard 1.7mm
Field of view	about 60°
supported system	Win7/8/10, Linux, MAC
service life	two years
a fixed way	screw fixed
Use environment requirements	Temperature and pressure
Applicable equipment support	ER myCobot 320 Series ER myCobot Pro 600 Series

Use for Objects

Camera Flange: Machine Vision

Introduction

- USB high-definition camera can be used with suction pump, self-adaptive gripper, artificial intelligence kit, etc., eye in hand to achieve precise positioning and calibration.

Installation and use

- Check that the kit has everything: screws and hexagonal spanners, camera module with usb cable



- Camera Installation:
  - Structural installation:

1. Align the camera module to the end of the robot arm according to the desired direction, and tighten the screws with the Allen spanner



- Electrical Connections:

1. 将 USB 线插入底座 USB 接口：

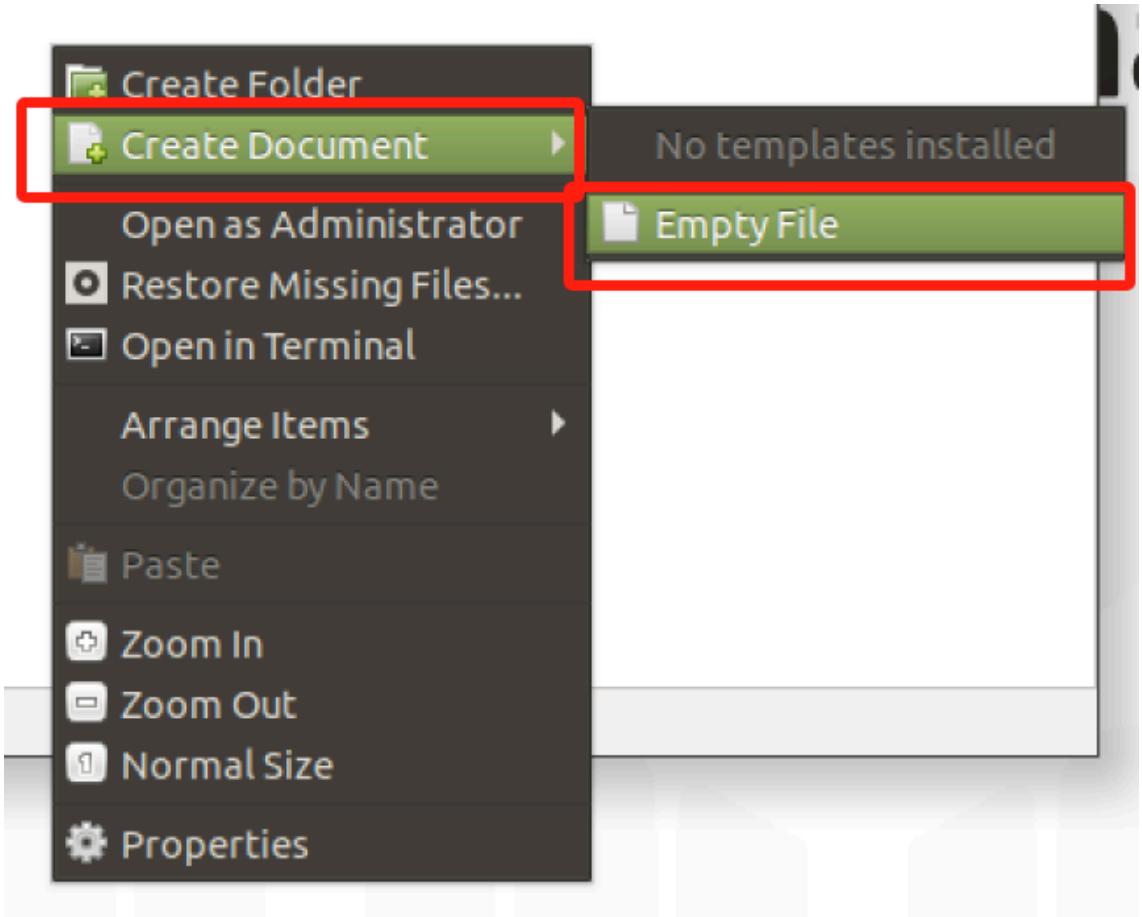


- Programming Development:

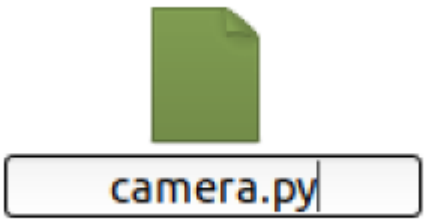
> Programming the camera module using python python environment download

  - Create a new python file:

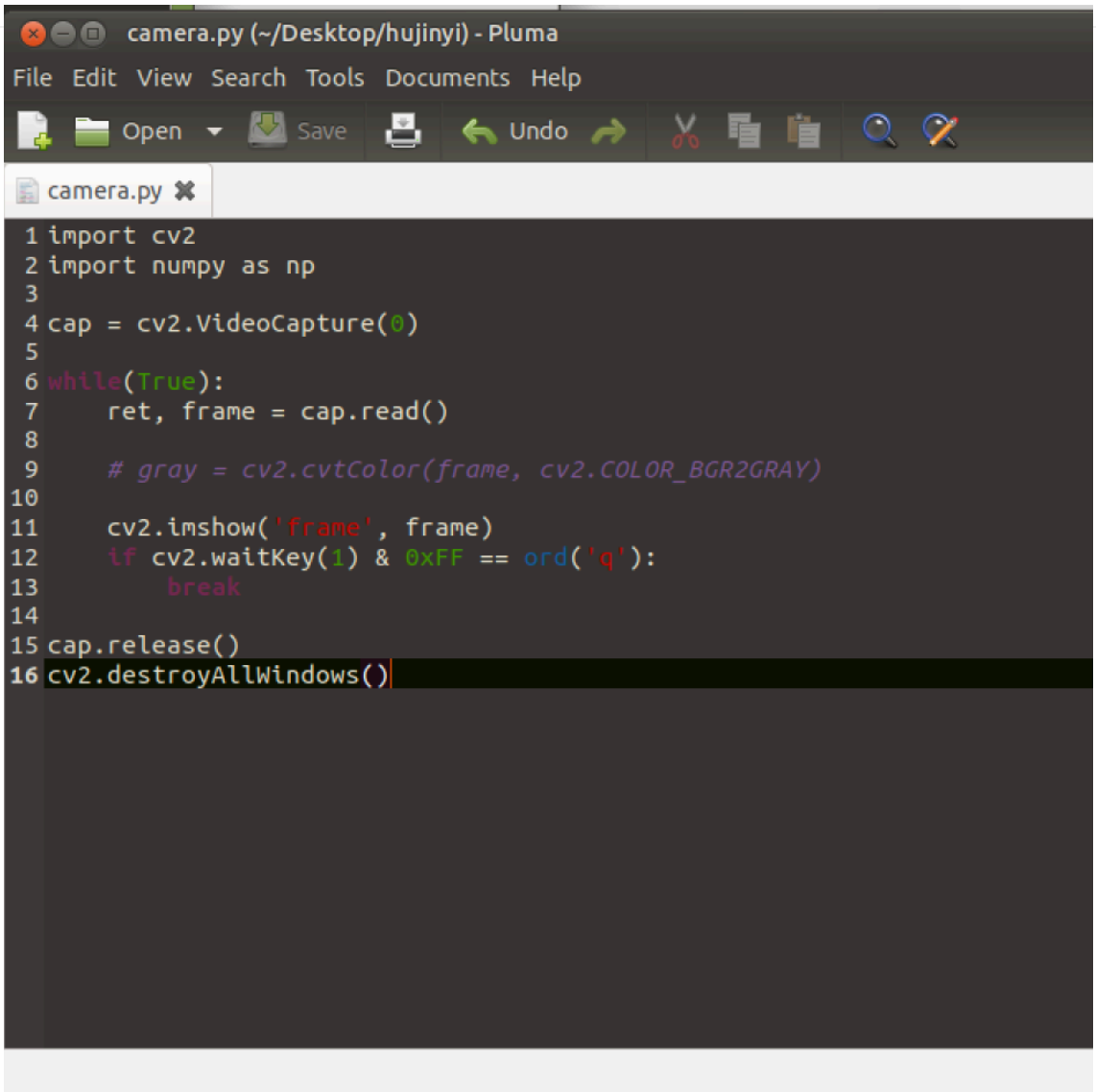
Right click on the desired file path to create a new python file:



> The file name can be changed as needed



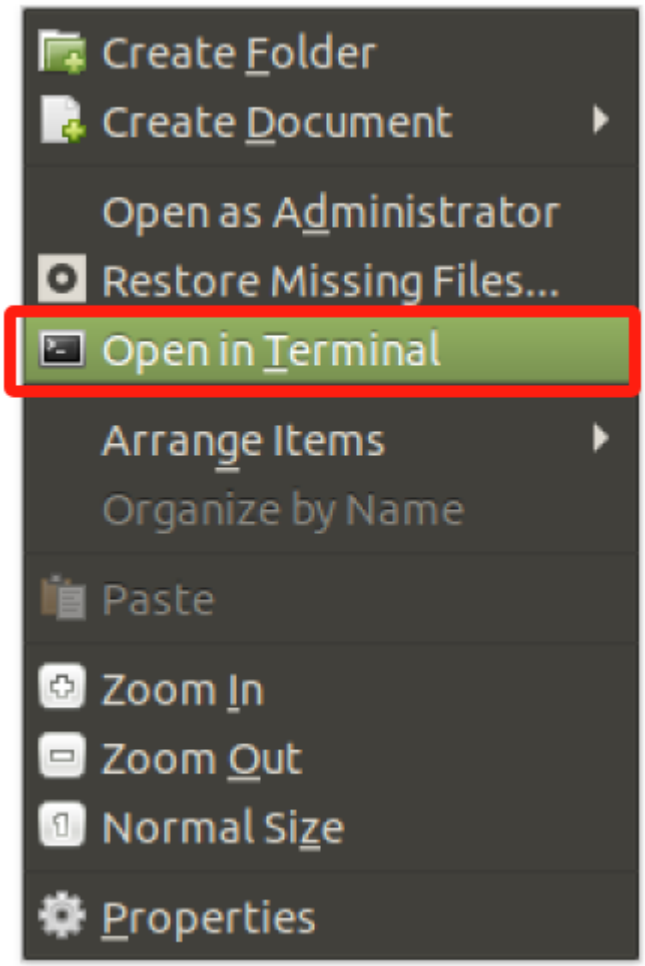
- Perform function programming:



> The code is as follows:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0) # "0", based on the queried camera equipment number

while(True):
    ret, frame = cap.read()

    # gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cv2.imshow('frame', frame)
    # Press 'q' to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

- Save the file and close it, right-click on an empty space in the folder to open a command line terminal



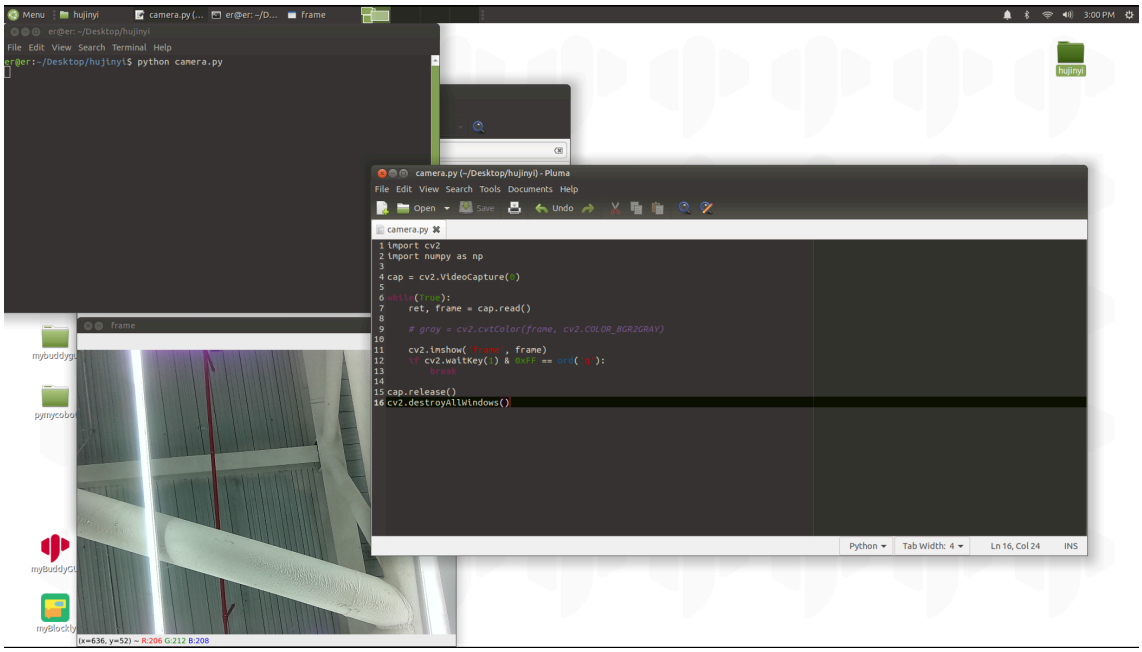
Input:

```
python camera.py
```



> You can see what the camera captured

Run results:

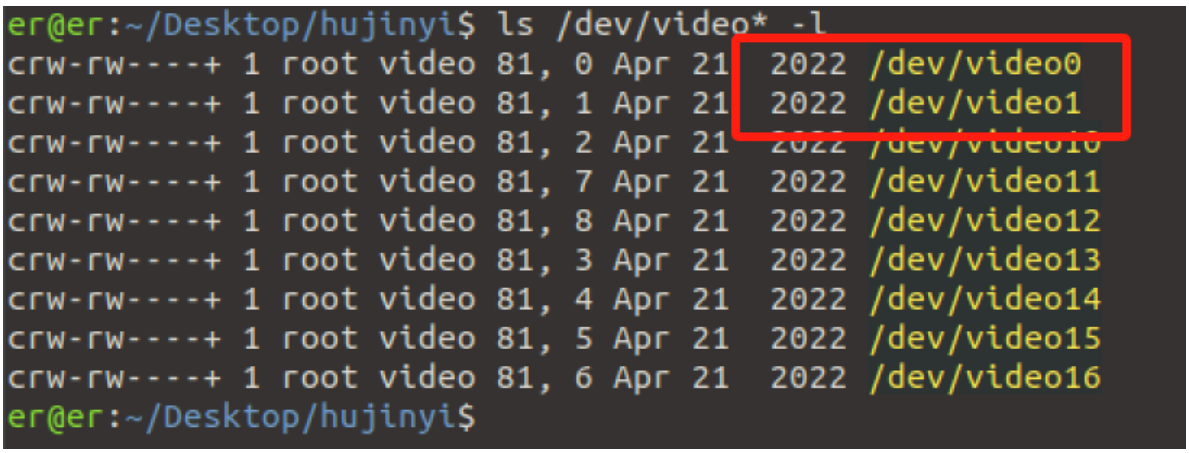


- How to check camera device number

```
ls /dev/video* -l
```

> This can be confirmed by plugging and unplugging the devices and using the command separately to observe the added device number.

Example results:



Mall link:

- Taobao
- shopify

← Accessories Tools Page

myCobot Pro Pen Holder

> **Compatible models:** ER myCobot 320 series, ER myPalletizer 630

Product Icon





Specifications

name	myCobotPro Pen Holder
model	myCobot_Pro_penHolder_J6
Material	photosensitive resin
empty nib	±1 mm
service life	two years
a fixed way	screw fixed
Use environment requirements	Temperature and pressure
Applicable equipment	ER myCobot 320 series, ER myPalletizer 600

Use for Objects

Used when using the robotic arm to write and draw

Introduction

- The overall solid color design supports up and down 15mm ultra-large stroke expansion and contraction, effectively reducing errors, and can be used for writing, drawing and other applications.

Applicable object

- Whiteboard pen

Mall link

- Taobao
- shopify

How to use

Installation and use

- mounting

1. Stick the two short screws into the gripper holes:



2. Use an Allen key to secure the gripper to the end of the arm:





← Accessories Tools Page | Next Page →

myCobot Pro Mobile Phone Holder

> **Compatible models:** ER myCobot 320 series, ER myCobot 630

Product Icon







Specifications

name	myCobotPro Mobile Phone Holder
model	myCobot_Pro_PhoneHolder_J6
Material	photosensitive resin
clamping weight	200g
service life	two years
a fixed way	screw fixed
Use environment requirements	Temperature and pressure
Applicable equipment	ER myCobot 320 series, ER myCobot 600

Use for Objects

Introduction

- It is suitable for shooting and other equipment that requires physical clamping. It can clamp a variety of mobile phones. It has a simple structure and is easy to install and disassemble.

Applicable object

- shooting equipment

Mall link:

- Taobao
- shopify

How to use

1 Installing :



← Accessories Tools Page

Single-ended suction pump

> **Compatible models:** myCobot 320, myCobot Pro 600, myCobot Pro 630

Product images



Specifications

Name	Single-ended suction pump	
Suction pump box size	150mmX150mmX108mm	
Suction pump length	106mm	
Suction cup diameter	25mm	
Tube length	1m	
Working voltage	24V	
Rated current	1A	
Own weight	2kg	
Rated load	1kg	
Flow rate	10-15L/min	
Negative pressure	-85Kpa	
Fixed mode	Screw fixed	
Use environment requirements	Normal temperature and pressure	
Control interface	IO control	
Applicable equipment	myCobot 320, myCobot Pro 600, myCobot Pro 630	

Single-head suction pump: Used for adsorbing objects

Introduction

- The suction cup suction pump is connected to the object to be adsorbed through the suction cup, pipe and other components, and the suction cup is vacuumed, causing the internal air pressure to change from normal pressure to negative pressure, and the pressure difference between the external atmospheric pressure and this negative pressure is used to achieve the purpose of adsorbing the object.

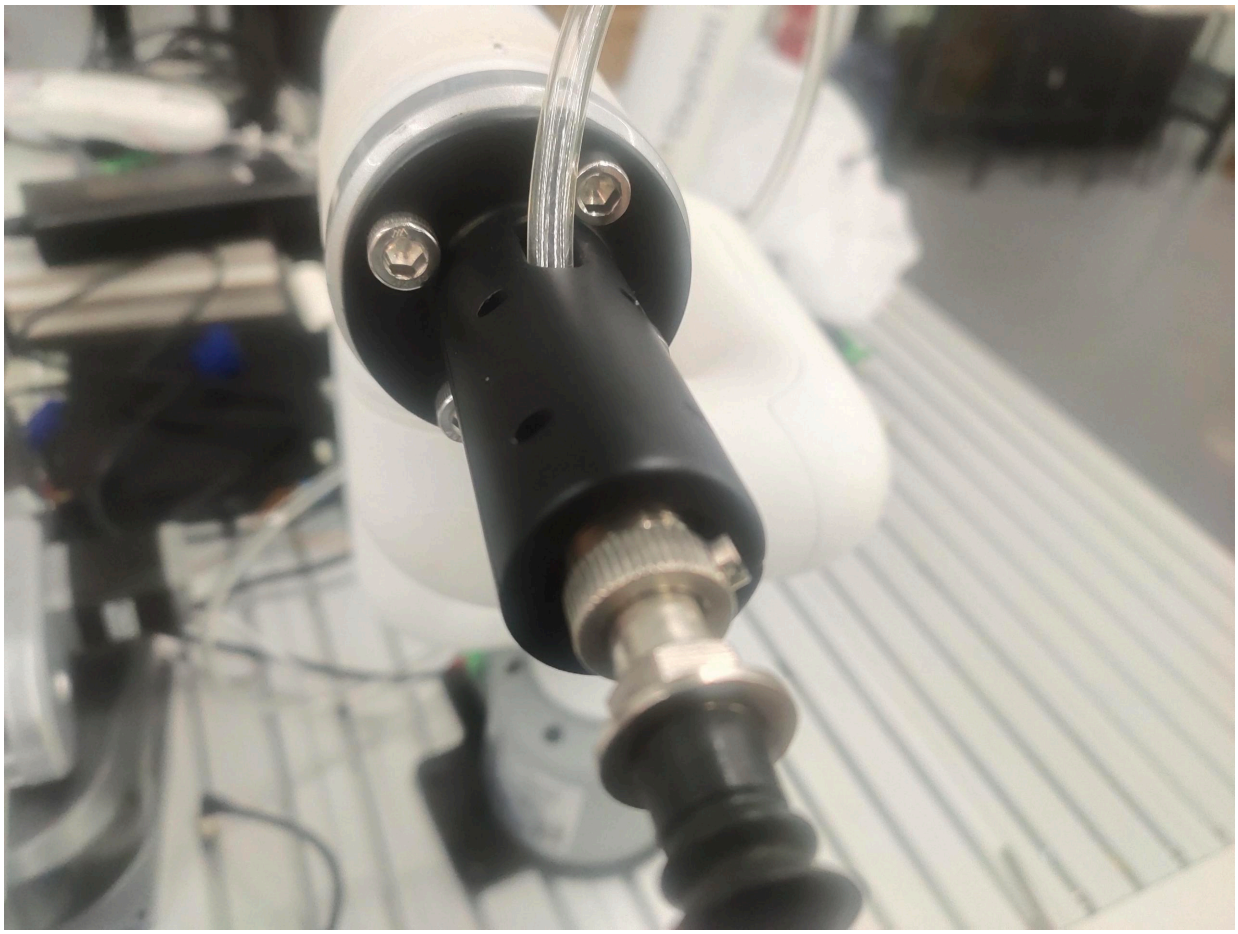
Working principle

- Start the vacuum equipment to suck, so that negative air pressure is generated in the suction cup, so that the object to be lifted is firmly sucked, and the object to be lifted can be transported.
- When the object to be lifted is transported to the destination, the vacuum suction cup is steadily inflated, so that the negative air pressure in the vacuum suction cup changes to zero air pressure or slightly positive air pressure, and the vacuum suction cup is separated from the object to be lifted, thereby completing the task of lifting and transporting heavy objects.

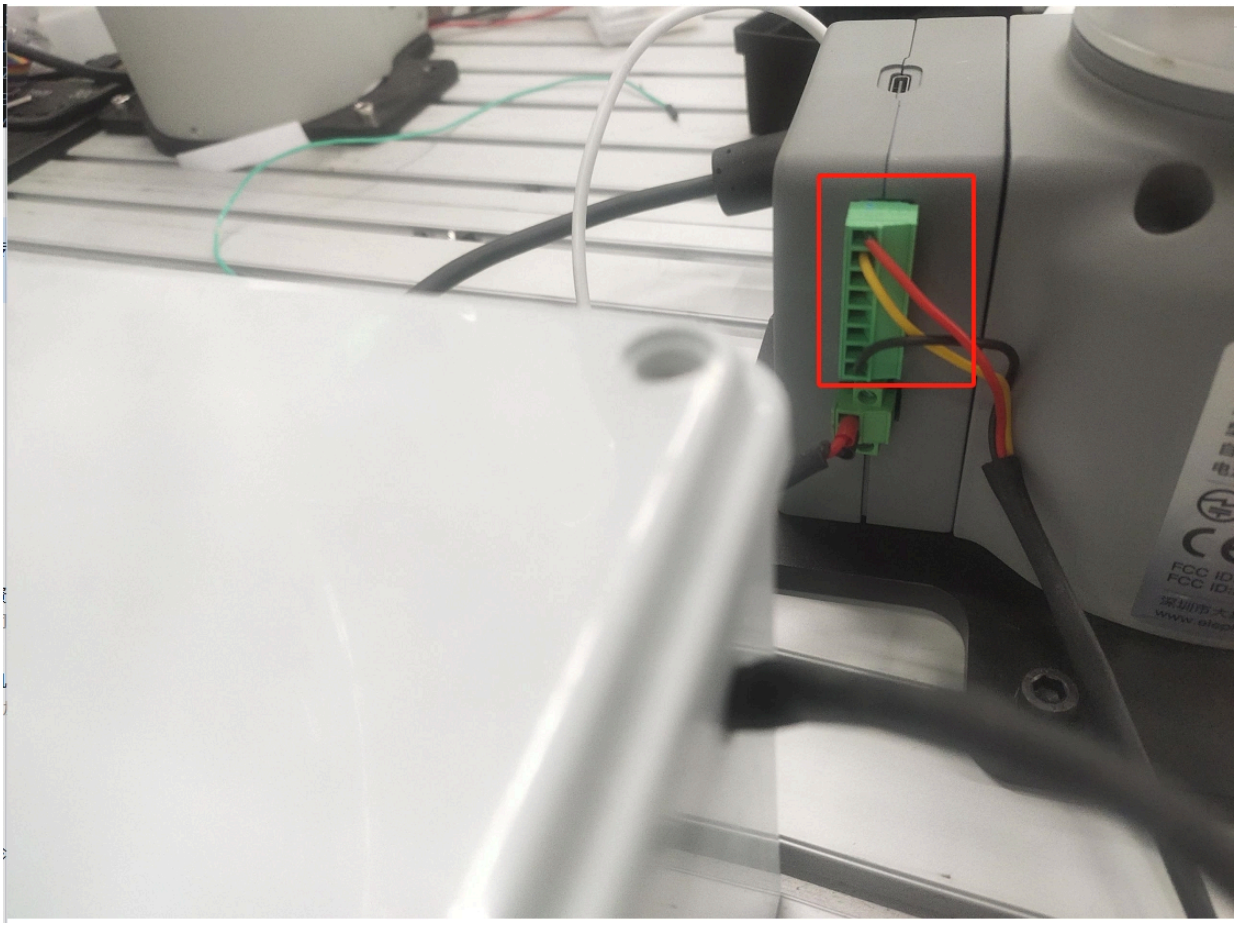
Applicable objects Applicable to flat objects

Hardware installation

First install the suction pump on the end of the robot



Then connect the wire of the suction pump control box to the base IO of the robot



Python control

M5 version

```
from pymycobot import MyCobot,utils
import time
arm=MyCobot(utils.get_port_list()[0])
for i in range(1):
    arm.set_basic_output(1,0)#Turn on the suction pump
    time.sleep(2)
    arm.set_basic_output(1,1)#Turn off the suction pump
    time.sleep(2)
```

PI version

1.4.1 AdaptiveGripper

```
from pymycobot import MyCobot,PI_PORT,PI_BAUD
import time
arm=MyCobot(PI_PORT,PI_BAUD)
for i in range(1):
    arm.set_basic_output(1,0)#Open the suction pump
    time.sleep(2)
    arm.set_basic_output(1,1)#Close the suction pump
    time.sleep(2)
```

myCobot Pro force-controlled gripper

1 Product image



2 Specifications

Name	myCobot Pro force-controlled gripper
Material	PC, PBT
Dimensions	156X106X61mm
Process technology	Injection molding
Gripping range	0-100 mm (default fingertip)
Repeatability	0.5 mm
Service life	300,000 openings and closings
Drive mode	Electric drive
Transmission mode	Gear + connecting rod
Dimensions	158x105x55mm
Weight	340 g
Rated load	500g
Operating voltage	24V
Fixing method	Screw fixing
Environmental requirements	Normal temperature and pressure
Control interface	RS485/IO control/button control
Applicable equipment	ER myCobot 320 series, ER Mercury series, ER myCobot Pro 600, ERmyCobot Pro 630, other general robots

3 Working principle

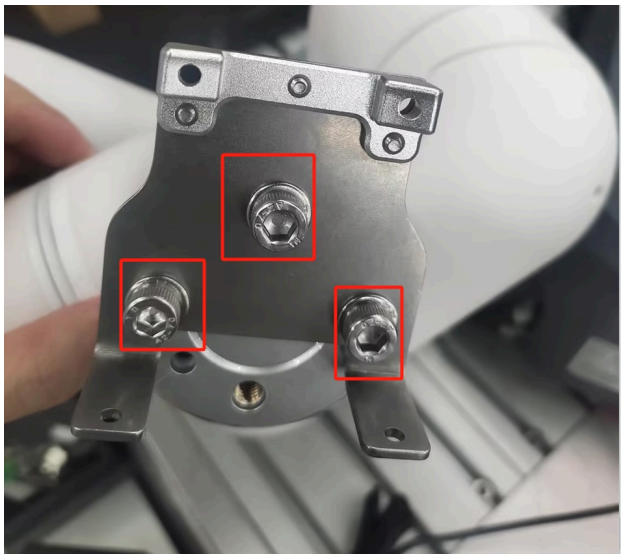
Driven by the motor, the finger surface of the manipulator makes linear reciprocating motion to achieve opening or closing. By setting the clamping torque, the impact of the workpiece is minimized, the positioning point is controllable, and the clamping is controllable.

4 Usage scenario

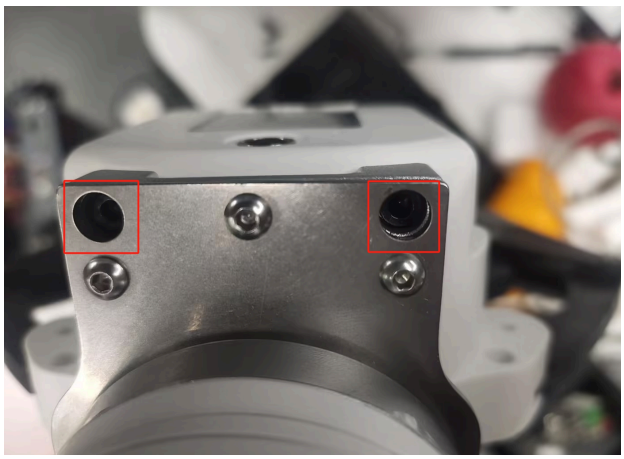
Experimental operation: In scientific research experiments, complete the grasping and moving of test tubes, utensils, etc. to ensure the safety and accuracy of the experiment. Educational demonstration: As a teaching tool, it helps students understand the principle of robot grasping and cultivate practical ability. Material handling: In simulated production lines or warehouses, materials of various specifications are handled to improve work efficiency.

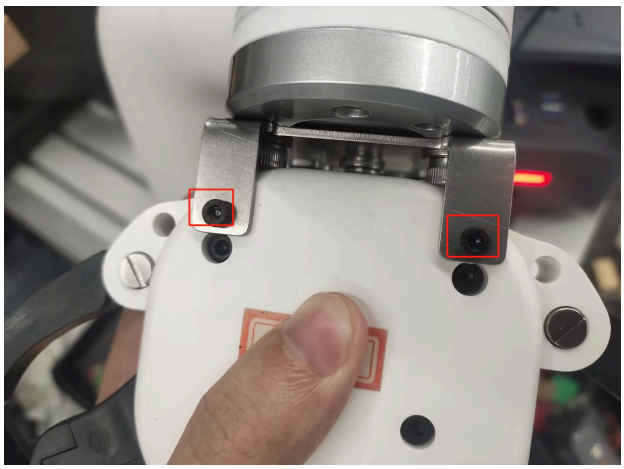
5 Installation method

Use screws and washers to install the gripper connector to the end flange of the robot arm



Then use screws to install the gripper on the connector





Finally, use M8 aviation line to connect the gripper and the robot arm



6 Python control method

The robot arm needs to burn the pico firmware that supports the force-controlled gripper, and install the pymycobot driver library that supports the force-controlled gripper, but since both are still in internal testing and have not been officially released, if necessary, please contact after-sales staff to obtain

6.1 Python control API description

Note: The time interval between calling each instruction of the gripper must be greater than 1.5 seconds.

```
set_pro_gripper(gripper_id, address, value)
```

- **Function**: Set the parameters of the Pro force-controlled gripper. You can set a variety of parameter functions. For details, please see the following table.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- `address` (`int`): The command number of the gripper.
- `value` : The parameter value corresponding to the command number.

Function	gripper_id	address	value
Set gripper ID	14	3	1 ~ 254
Set gripper enable status	14	10	0 or 1, 0 - off enable; 1 - on enable
Set gripper clockwise runnable error	14	21	0 ~ 16
Set the anti-clockwise running error of the gripper	14	23	0 ~ 16
Set the minimum starting force of the gripper	14	25	0 ~ 254
IO output setting	14	29	0, 1, 16, 17
Set IO opening angle	14	30	0 ~ 100
Set IO closing angle	14	31	0 ~ 100
Set servo virtual position value	14	41	0 ~ 100
Set the clamping current	14	43	1 ~ 254

- **Return value**:
- Please check the following table:

Function	Return value
Set the gripper ID	0 - Failure; 1 - Success
Set the gripper enable status	0 - Failure; 1 - Success
Set the clockwise runnable error of the gripper	0 - Failure; 1 - Success
Set the counterclockwise runnable error of the gripper	0 - Failure; 1 - Success
Set the minimum starting force of the gripper	0 - Failure; 1 - Success
IO output setting	0 - Failure; 1 - Success
Set IO opening angle	0 - Failure; 1 - Success
Set IO closing angle	0 - Failure; 1 - Success
Set the servo virtual position value	0 - Failure; 1 - Success
Set the clamping current	0 - Failure; 1 - Success

`get_pro_gripper(gripper_id, address)`

- **Function**: Get the parameters of the Pro force-controlled gripper, which can get a variety of parameter functions. Please refer to the following table for details.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.

- `address` (`int`): The command number of the gripper.

Function	gripper_id	address
Read firmware major version number	14	1
Read firmware minor version number	14	2
Read gripper ID	14	4
Read gripper clockwise runnable error	14	22
Read the anti-clockwise running error of the gripper	14	24
Read the minimum starting force of the gripper	14	26
Read the IO opening angle	14	34
Read the IO closing angle	14	35
Get the amount of data in the current queue	14	40
Read the servo virtual position value	14	42
Read the clamping current	14	44

- **Return value**:
- Check the following table (if the return value is -1, it means that no data can be read):

Function	Return value
Read the firmware major version number	Major version number
Read the firmware minor version number	Minor version number
Read the gripper ID	1 ~ 254
Read the gripper clockwise runnable error	0 ~ 254
Read the gripper counterclockwise runnable error	0 ~ 254
Read the minimum starting force of the gripper	0 ~ 254
Read the IO opening angle	0 ~ 100
Read the IO closing angle	0 ~ 100
Get the amount of data in the current queue	Return the amount of data in the current absolute control queue
Read the servo virtual position value	0 ~ 100
Read the clamping current	1 ~ 254

`set_pro_gripper_angle(gripper_id, gripper_angle)`

- **Function**: Set the force-controlled gripper angle.

- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- `gripper_angle` (`int`): Gripper angle, value range 0 ~ 100.
- **Return value**:
- 0 - Failed
- 1 - Success

`get_pro_gripper_angle(gripper_id)`

- **Function**: Read the angle of the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**: `int` 0 ~ 100

`set_pro_gripper_open(gripper_id)`

- **Function**: Open the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
- 0 - Failed
- 1 - Successful

`set_pro_gripper_close(gripper_id)`

- **Function**: Close the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
- 0 - Failure
- 1 - Success

`set_pro_gripper_calibration(gripper_id)`

- **Function**: Set the zero position of the force-controlled gripper. (The zero position needs to be set first for the first use)
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
- 0 - Failure
- 1 - Success

`get_pro_gripper_status(gripper_id)`

- **Function**: Read the gripping status of the force-controlled gripper.
- **Parameters**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.

- **Return value:**
- `0` - Moving.
- `1` - Stopped moving, no object was detected.
- `2` - Stopped moving, object was detected.
- `3` - After the object was detected, it fell.

`set_pro_gripper_torque(gripper_id, torque_value)`

- **Function**: Set the force-controlled gripper torque.
- **Parameters**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- `torque_value` (`int`): Torque value, value range 1 ~ 100.
- **Return value**:
- 0 - Failed
- 1 - Success

`get_pro_gripper_torque(gripper_id)`

- **Function**: Read the torque of the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value:** (`int`) 100 ~ 300

`set_pro_gripper_speed(gripper_id, speed)`

- **Function**: Set the speed of the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- `speed` (int): Gripper movement speed, value range 1 ~ 100.
- **Return value**:
- 0 - Failed
- 1 - Success

`get_pro_gripper_default_speed(gripper_id, speed)`

- **Function**: Read the default speed of the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**: Gripper default movement speed, range 1 ~ 100.

`set_pro_gripper_abs_angle(gripper_id, gripper_angle)`

- **Function**: Set the absolute angle of the force-controlled gripper.
- **Parameter**:
- `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- `gripper_angle` (`int`): Gripper angle, value range 0 ~ 100.
- **Return value**:
- 0 - Failure
- 1 - Success

`set_pro_gripper_pause(gripper_id)`

- **Function**: Pause motion.
- **Parameter**:
- `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
- 0 - Failure
- 1 - Success

`set_pro_gripper_resume(gripper_id)`

- **Function**: Resume motion.
- **Parameter**:
- `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
- 0 - Failure
- 1 - Success

`set_pro_gripper_stop(gripper_id)`

- **Function**: Stop motion.
- **Parameters**:
- `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
- 0 - Failure
- 1 - Success

6.2 Python zero calibration operation instructions

After connecting the gripper to the end of the robot arm, run the following code, the gripper will first open to the maximum, then completely close, and complete the calibration.

```
from pymycobot import MyCobot320
mc=MyCobot320("COM1")#Fill in the serial port number of the robot
mc.set_pro_gripper_calibration(14)#The default ID of the gripper is 14
```

6.3 Python control scenario example

demo1: Torque value adaptation

The user can observe the deformation degree of the force-controlled gripper after clamping the object under the premise of setting the corresponding torque value Select a torque value with the smallest deformation but can clamp the object normally. This torque value is the most suitable torque value for the current object

```
from pymycobot import MyCobot320
import time
mc=MyCobot320("COM1")#Fill in the serial port number of the robot
mc.set_pro_gripper_torque(14,10)#Set the torque value
print(mc.get_pro_gripper_torque(14))#View the set torque value
mc.set_pro_gripper_open(14)#Gripper fully open
time.sleep(3)
mc.set_pro_gripper_close(14)#Gripper fully closed
```

demo2: Grab and move objects from point A to B

```
from pymycobot import MyCobot320
import time
mc=MyCobot320("COM1")#Fill in the serial port number of the robot
mc.set_pro_gripper_torque(14,10)#Set the torque value of the gripper
print(mc.get_pro_gripper_torque(14))#Print the set gripper torque value
mc.set_pro_gripper_open(14)#First open the gripper completely
start_angles=[19.68, -1.23, -91.4, -0.52, 90.08, 60.29]
target_coords=[[231.3, -61.3, 232.7, 178.35, -2.7, -130.56],[231.3, 65.3, 232.7, 178.35, -2.7, -130.56]]
end_angles=[80,0,-85,0,90,60]
for i in range(len(target_coords)):
  mc.sync_send_angles(start_angles,50)
  mc.send_coords(target_coords[i],100,1)
  time.sleep(2)
  mc.send_coord(3,165,50)
  time.sleep(2)
  mc.set_pro_gripper_close(14)
  time.sleep(2)
  mc.send_coords(target_coords[i],100,1)
  mc.sync_send_angles(end_angles,100)
  mc.set_pro_gripper_open(14)
  time.sleep(2)
```


Elepha
Roboti

myCobot 320 M5



1 Profile

Updated from myCobot 280, **myCobot 320 M5** is mainly developed for to assist users in program-editing. With delicate structure and all-in-one design, myCobot 320 has a maximum arm stretch of 350 mm, maximum load of 1 kg and repeated positioning precision of ±0.5mm.

myCobot 320 for M5 weights 3kg with a load of 1kg and working radius of 320mm. The small-sized product is endowed with powerful functions and is characterized by easy operation, ability to work with human safely.

2 Features

- **Easy to operate and open-source**
  - Users can operate the robot via myBlockly and dragging teaching easily after quick-start learning.
  - It supports the development systems, such as ROS and moveIt.
- **Economical and cost-effective**
  - It adopts a standardized 8-hour working system so as to replace human in doing repetitive and standard jobs.
  - Worth over 10,000 RMB, it reduces costs and synergizes efforts for high-efficiency scientific research.
- **Powerful performance and equipped with two screens**
  - It adopts a brushless DC servo so as to realize repeated positioning precision of ±0.5mm.
  - The body is equipped with two screens and supports M5 ecological applications, effectively expanding coordinative application.
- **On-in-all design and safe collaborative work**
  - With delicate structure, it optimizes space and integrates with application in a coordinated way.
  - It also has kinematics self-interference detection, which can effectively avoid motion collisions.

3 Application

myCobot 320 M5 is not only a tool for production but also a tool to expand imagination boundary. It means it can work with multiple types of end effectors to adapt to a variety of application, such as scientific research, education and function showing, etc. The user experience is excellent.

4 Chapter Summary

The next section of this manual will guide you to the sub-sections that will give you a more complete understanding of our product specifications, control core parameters, mechanical structure parameters, electrical characteristics parameters, and coordinate system definitions.

Please feel free to select the following sections based on your interests and requirements:

2.1-Machine specification

In this section, we will describe the basic attributes of the product's industry consensus, such as robot description load, torque, positioning accuracy, size, functional support, and power parameters.

2.2-Control core parameter

Understand the main control parameters of the product, which is convenient for later custom development and use.

2.3-Mechanical structure parameter

In this part, we will introduce the important parameters of the mechanical structure of the product in detail, and provide customers with the corresponding 3D model download link, so that customers can better understand our products.

2.4-Electrical characteristic parameter

This chapter will provide customers with the electrical characteristic parameters of the product, which is convenient for customers to customize the development and use in the later period.

2.5-Coordinate system

This section describes the Angle and coordinate information of the product and explains the supported coordinate system controls. At the same time, the relevant parameters of the product are provided for the calculation of the corresponding coordinate system, such as DH parameters.

Please click on the respective links based on your interests to access more detailed information. If you have any questions or require further assistance, please do not hesitate to contact our customer support team. We are committed to providing you with support and guidance. Thank you for choosing our product, and we look forward to delivering an outstanding user experience for you!

5 Words of Thanks

We greatly appreciate you taking the time to read the myCobot 320 user manual. We hope this document helps you to better understand and effectively use this robot, thereby inspiring your creativity. Should you have any questions or require further assistance, please do not hesitate to contact our customer support team. We look forward to seeing the innovative projects you accomplish with myCobot 320 and welcome you to our rapidly growing community of developers.

If you have already read all the content in this chapter, please proceed to the next chapter.

← Previous Chapter | Next Chapter →

Product specification parameter



1 Manipulator parameter

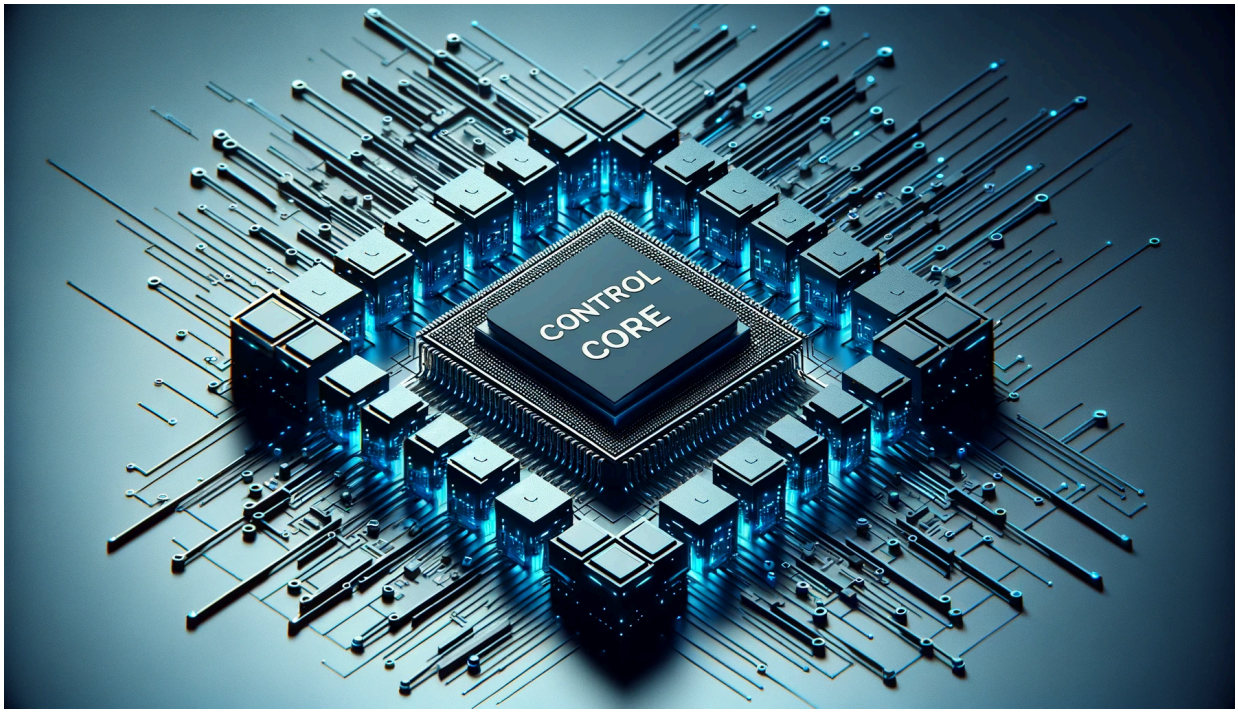
Index	Parameter
Name	Baby Elephant Cooperative Robotic Arm
Model	myCobot 320-M5
Freedom	6
Working Radius	350mm
Efficient Load	1kg
Repeated Positioning Precision	±0.5mm
Weight	3。3kg
Power Input	24V,9.2A
Operating Temperature	0°~45°
Communication Interface	Bluetooth Type-C
Service life	2000h

2 Software basic function support

Function/development environment	Usage situation
Free Mobile	Support
Joint Movement	Support
The Cartesian Movement	Support
Track Recording	Support
Wireless Control	Support
Emergency Stop	Support
Windows	Support
Linux	Support
MAC	Support
ROS 1	Support
Python	Support
C++	Support
C#	Support
JavaScript	Support
myblockly	Support
Arduino	Support
mystudio	Support
Serial Control Protocol	Support
TCP/IP	Support
MODBUS	Support

← Previous Page | Next Page →

Control core parameter



1 Master controller specification table

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

2 Secondary controller 1 Specifications

Index	Parameter
Auxiliary Control	Atom
Auxiliary Control Model	ESP32
Auxiliary Controller Core Parameters	240MHz dual core. 600 DMIPS, 520KB SRAM. Wi-Fi, dual mode Bluetooth
Auxiliary controller Flash	4MB
LCD display	2.0"@320*240 ILI9342C IPS panel, maximum brightness 853nit
TypeC	*1

3 Secondary controller 2 Specifications

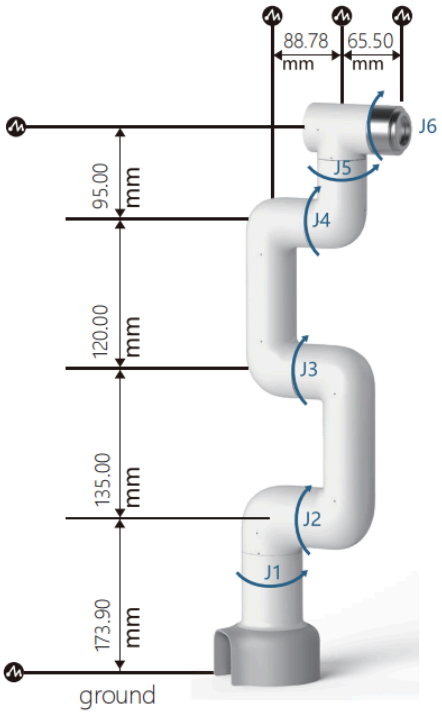
Index	Parameter
uxiliary Control	Pico
Auxiliary Control Model	ESP32
Auxiliary Controller Core Parameters	240MHz dual core. 600 DMIPS, 520KB SRAM. Wi-Fi, dual mode Bluetooth
Auxiliary controller Flash	4MB
TypeC	*1

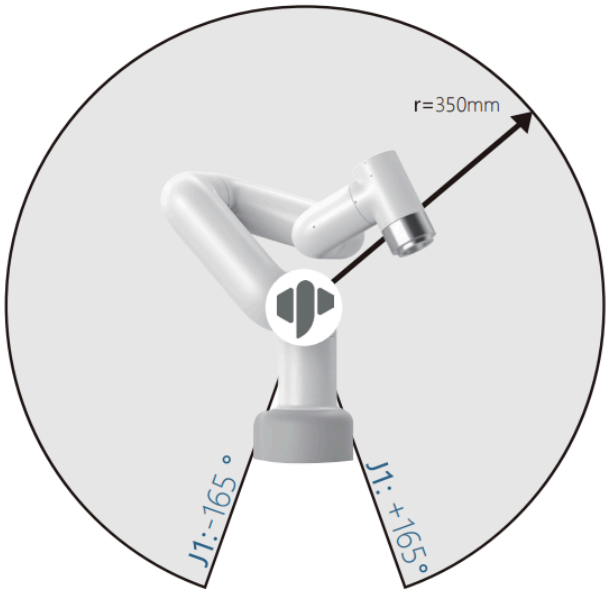
← Previous Page | Next Page →

Structural dimension parameter

> ! This chapter uses millimeters as the unit of distance and degrees as the unit of angle.

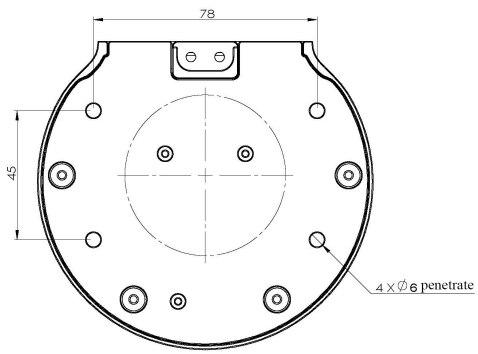
1 Product size and working space





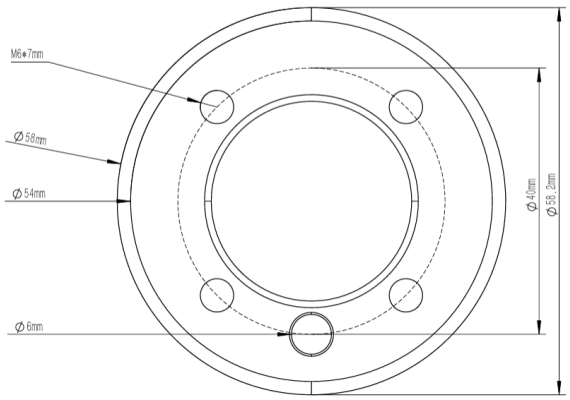
2 Mounting dimension of base

- The base is mounted with flange and is compatible with both installation of LEGO and M6 screw.

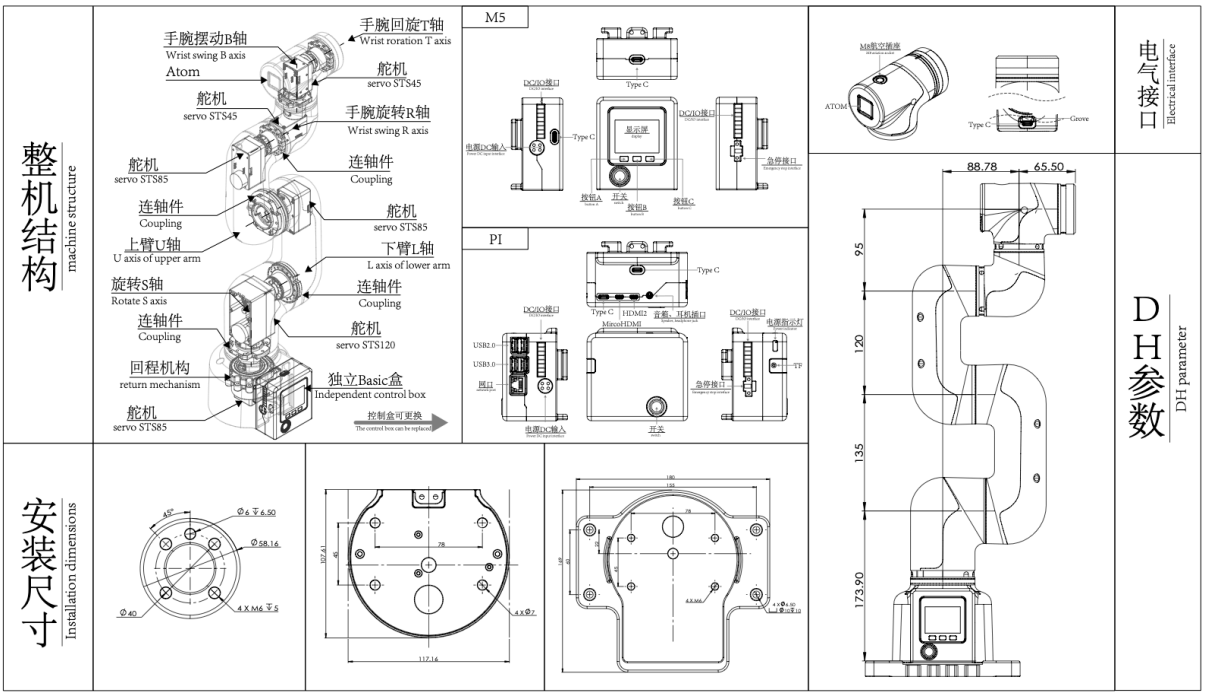


3 The end of the arm

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



4 Product views



5 3D model download

Product 3D model can be provided to provide reference materials for customers.

Download link:
https://download.elephantrobotics.com/Product_3d_files/myCobot_320_M5_2022v1.2_230708.STEP

← Previous Page | Next Page →

Electrical characteristic parameter

1 Base electrical interface overview

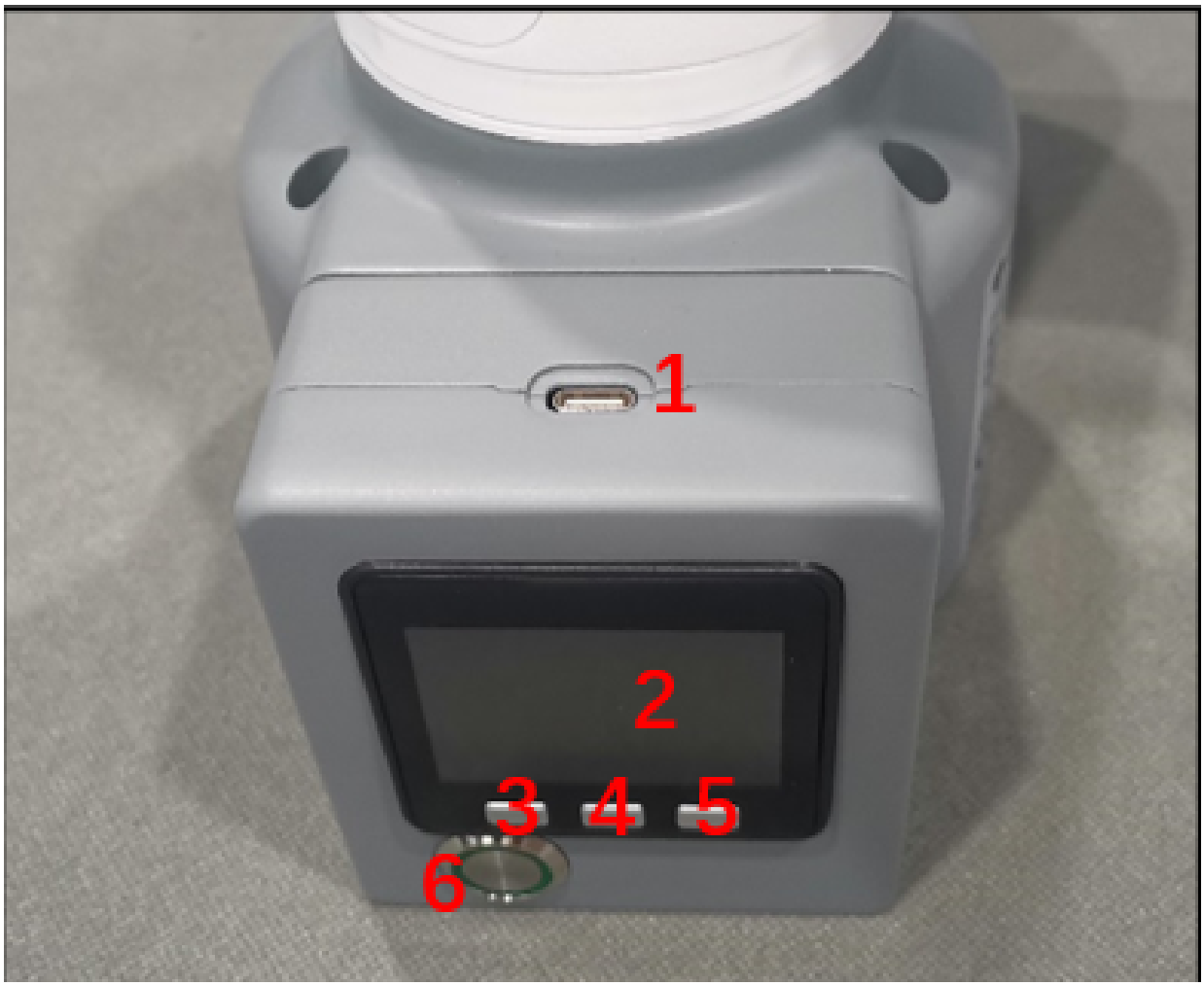


Figure 1 Front view of a base

1.1 Type C： Type C interface is used to connect and communicate with the PC, available for developers.

> This interface is only used for updating pico firmware, not as a general programming communication interface.

1.2 Screen： Screen is used to display the communication status of myCobot and calibrate the robot move to starting point with a 2-inch IPS screen.

1.3 Button A, Button B and Button C is used to work with the screen in a coordinated way.

1.4 Power switch： Power switch is used to control the main power input. If it is switched off, the controller is also powered off.



Figure 2 Left side of the base



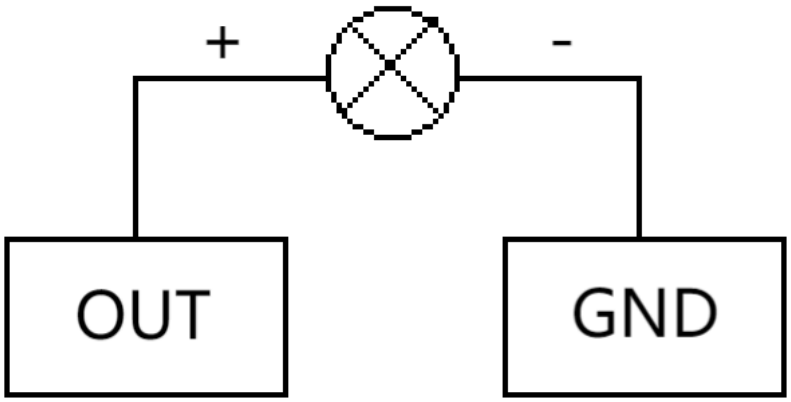
Figure 3 Right side of the base

1.5 24V output： Internal DC24V, available for users.

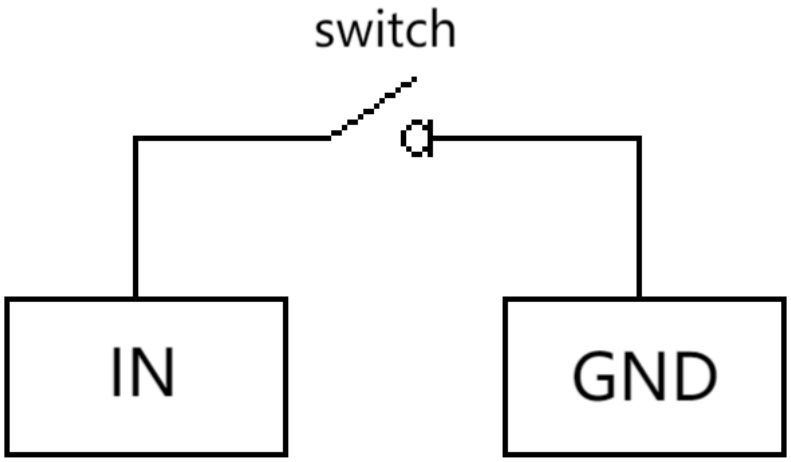
1.6 Digital input/output: includes 6 digital input signals and 6 digital output signals for interaction with other devices, and together with other devices constitute an important part of the automation system.

Built-in I/O power supply, voltage 24V, single output maximum 0.2A. External power supply is not supported.

It should be noted that the output signal is in PNP form, and the input signal is in NPN form. The following is the external wiring diagram:



Example of Output Signal Wiring



Example of Input Signal Wiring

1.7 Power DC input interface : It uses KPPX-4P R7BFDC power socket. The 24V 9.2A DC power adapter provided by the manufacturer can also be used to power myCobot320.

1.8 Emergency stop circuit terminal is connected to the emergency stop button box, which can be used to control the emergency stop of the robot.

> **Notice**: The emergency stop switch must be connected when the robot is in use, and make sure that the emergency stop switch circuit is always connected.

Number	Interface	Definition	Function	Remarks
1	Type C	Communication Interface	Communicate with PC	development use
2	screen	displaying	Use with buttons	
3	button	button A	Use with the display	
4		button B		
5		button C		
6	switch	power switch	Control input power on/off	With light (power on light)
7	DC/IO interface	GND	GND	
		IN6	Digital input signal 1~6	Input only in NPN mode
		IN5		
		IN4		
		IN3		
		IN2		
		IN1		
		24V	DC24V	
8	Type C	Communication Interface	Communicate with PC	development use
9	Power DC input interface	DC24V input	DC24V input	
10	DC/IO interface	24V	DC24V	
		OUT1	Digital output signal 1~6	output only in PNP mode
		OUT2		
		OUT3		
		OUT4		
		OUT5		
		OUT6		
		GND	GND	
11	Emergency stop interface	STOP	Emergency stop circuit interface	

2 Electrical Interface at the end of the Robot Arm

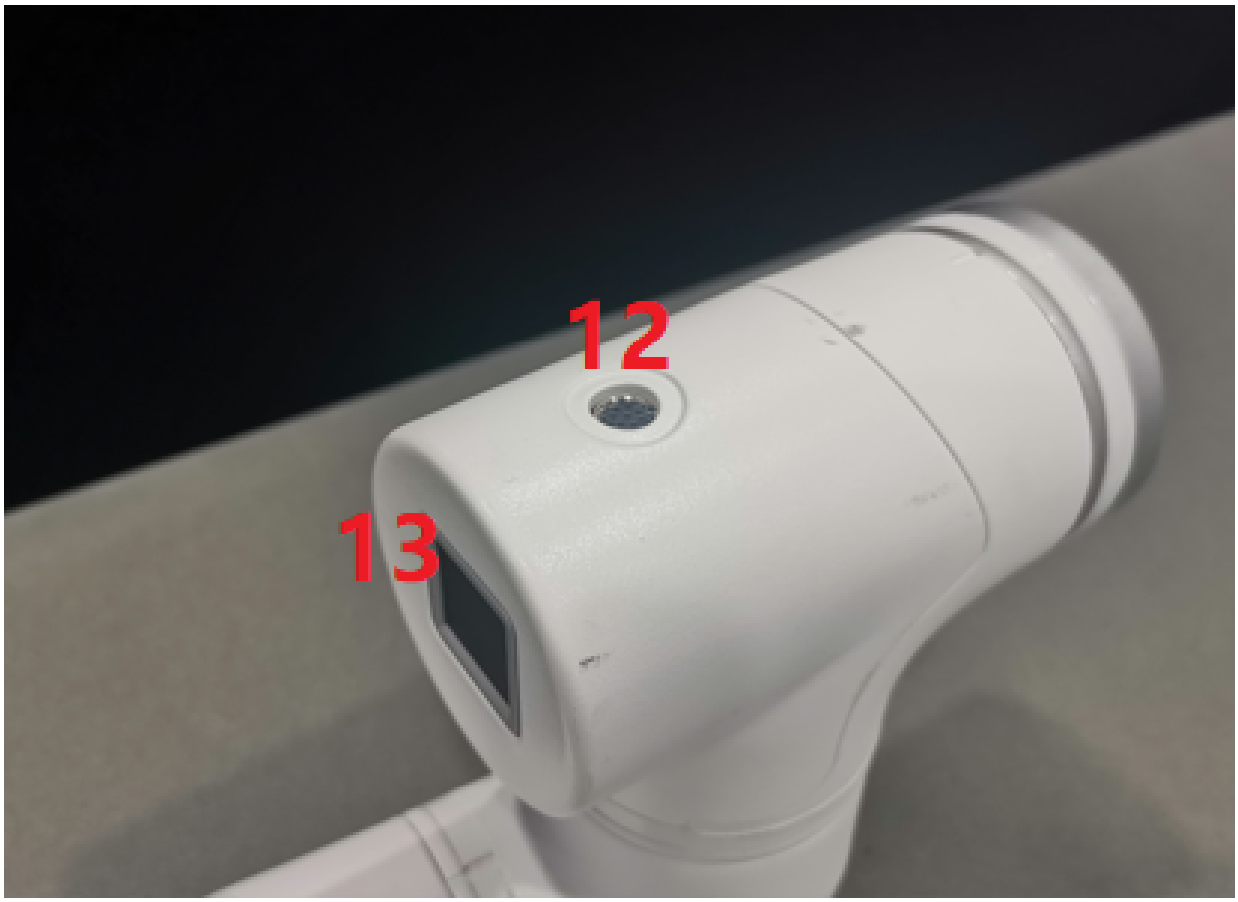


Figure 4 Side view of the end of the robot arm

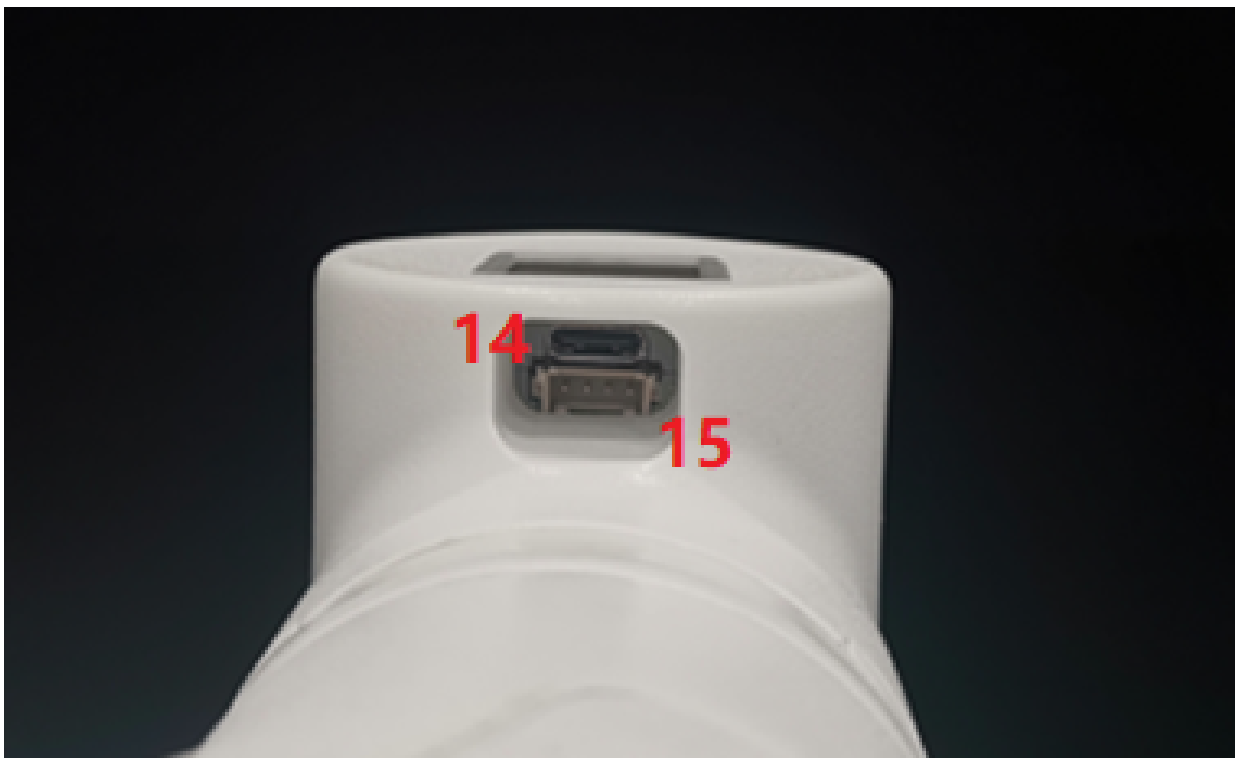
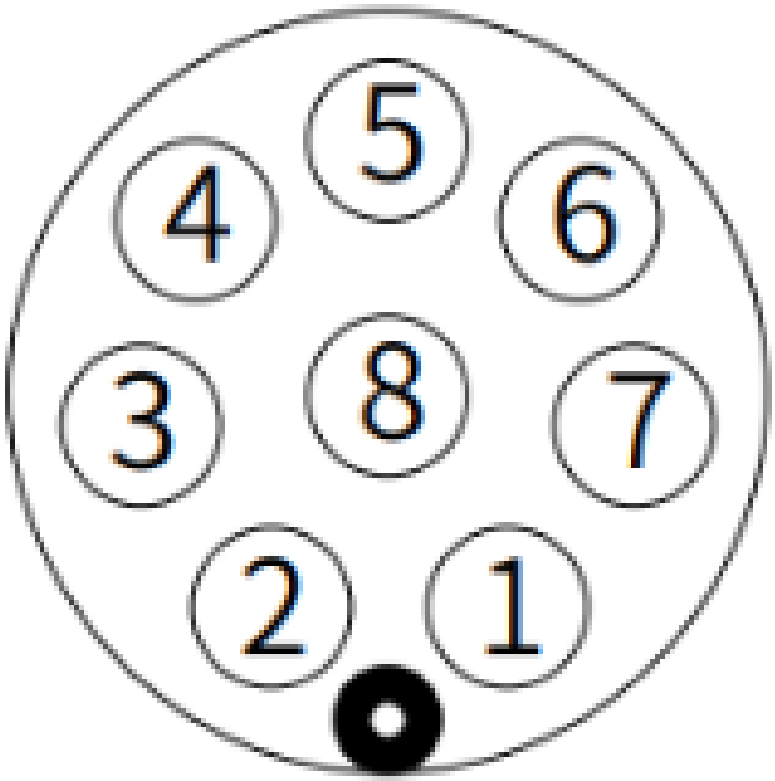


Figure 5 Side view of the end of the robot arm

Number	Interface	Definition	Function	Remarks
12	M8 aviation socket	End tool IO interface	interface Interact with external devices	
13	Atom	Led + button	Status view/drag to teach	
14	Type C	Communication Interface	Communicate with PC	Update Atom firmware using
15	Grove			developers use

2.1 End-tool IO interface: This is a tool I/O diagram, and the myCobot 320 robot provides one input and two outputs.

> Accessory and Atom end tool interface use does not support hot swapping.



Tool I/O Diagram

Built-in I/O power supply, voltage 24V, single output maximum 0.2A. External power supply is not supported.

The definition of each tool I/O port is shown in the following table. It should be noted that both the input and output of the tool I/O are PNP types, and the wiring mode is the same as the bottom output interface.

Number	Signal	Explanation	Matchable Color of M8 Line
1	GND	DC24V negative pole	White
2	OUT1	Tool output interface 1	brown
3	OUT2	Tool output interface 2	green
4	485A	reserved, undeveloped	yellow
5	24V	DC24V positive	Ash
6	IN1	Tool input interface 1	pink
7	IN2	unavailable	blue
8	485B	reserved, undeveloped	purple

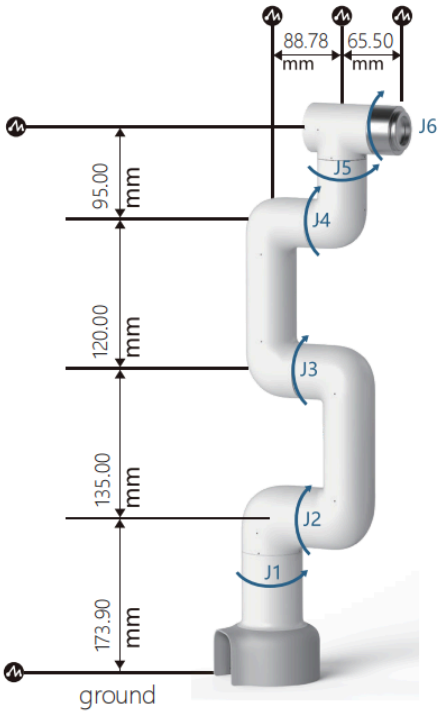
2.2 Atom：Atom is used for 5X5 RGB LED to display the state of the robot arm and key function (used when the robot performs the drag teaching)

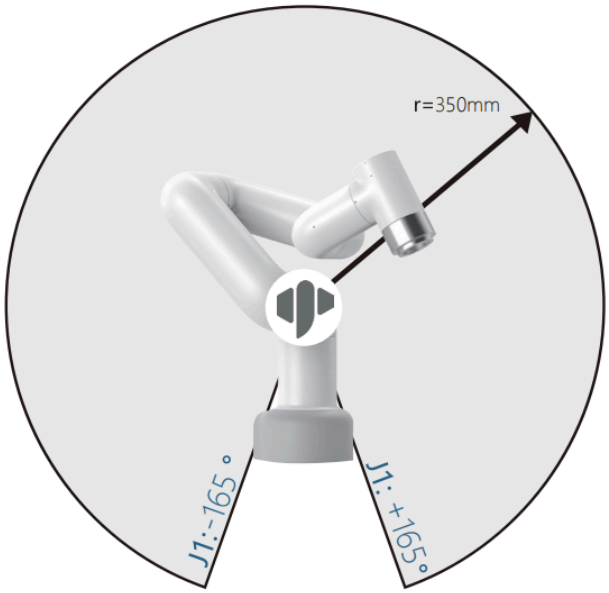
← Previous Page | Next Page →

Coordinate system

1 Joint coordinate system

The joint coordinate system is determined with reference to each moving joint. All joints are rotary joints. Range of motion is as follows:

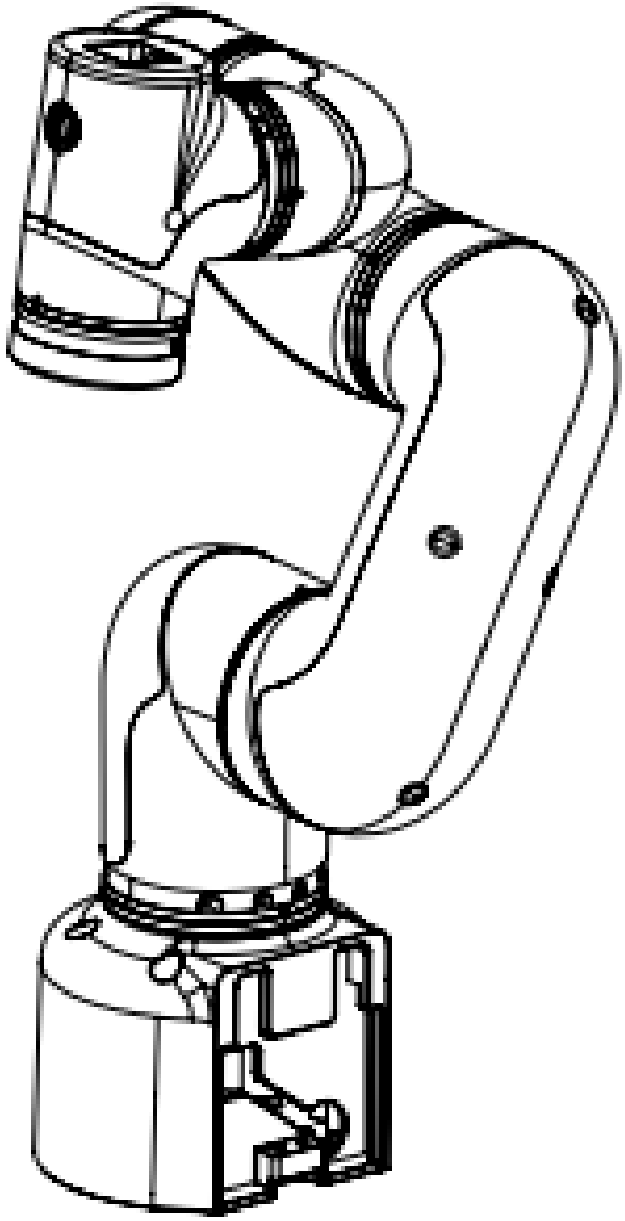




Joint	Angle
J1	-165 ~ +165
J2	-165 ~ +165
J3	-165 ~ +165
J4	-165 ~ +165
J5	-165 ~ +165
J6	-175 ~ +175

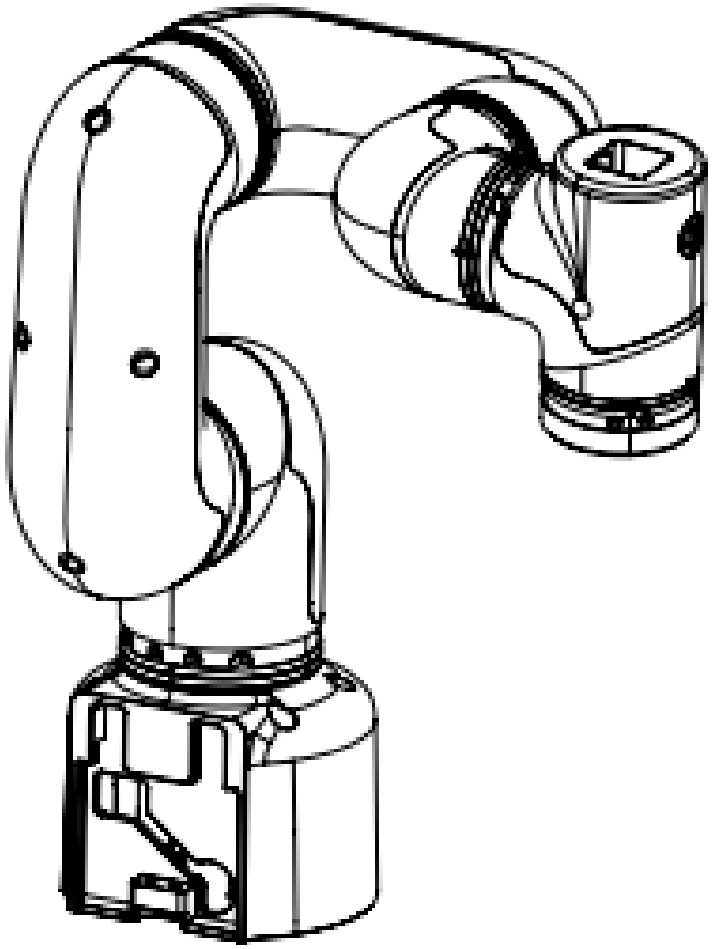
2 User coordinate system

The user coordinate system is a user-defined workbench coordinate system or workpiece coordinate system, whose origin and axis direction can be determined according to actual needs, which can easily measure the position of each point in the working interval and arrange tasks. The default user coordinate system is determined based on the center point of the robot arm base, and the positive direction of the Y axis is the direction of the heavy load line.

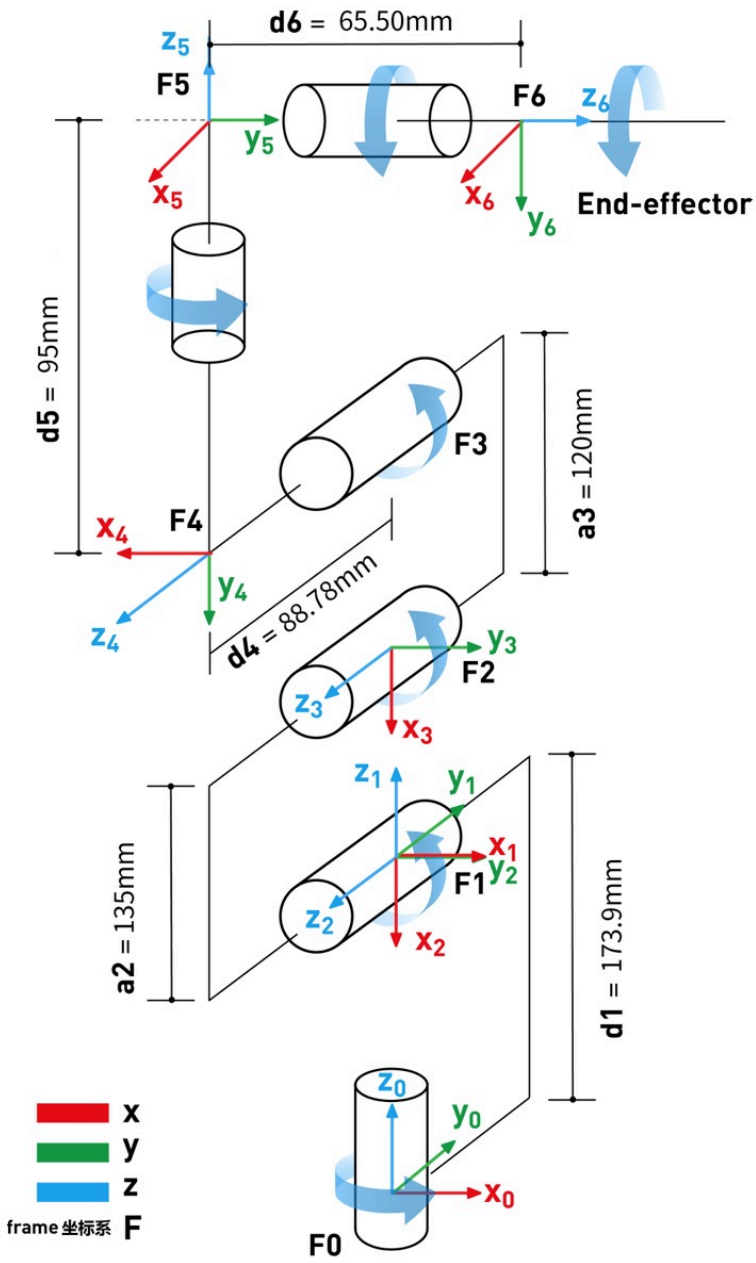


3 Tool coordinate system

The Tool coordinate system is a coordinate system that defines the position of the Tool Center Point (TCP) and the tool attitude, and its origin and direction are constantly changing with the position and Angle of the end workpiece. The default tool coordinate system is determined based on the center point of the tool flange, and the positive direction of the Y-axis is the opposite direction of the aviation socket.。



4 Joint connecting rod parameter specifications



4.1 DH definition

For the rotating joint n, set 0=0.0, where the X axis is in the same direction as the X axis, and select the origin position of the coordinate system (N) to satisfy d.=0.0. For the mobile joint n, set the orientation of the 8 axis so that it meets 0.=0.0. When d.=0.0, select the origin of the coordinate system (N) at the intersection of the XN-1 axis and the joint axis n.

Induction of linkage parameters in the linkage coordinate system If the linkage coordinate system is fixed to the linkage according to the above provisions, the linkage parameters can be defined as follows:

- a_i-1: The distance along x_i-1: from z_i-1 to z_i
- alpha_i-1: goes around x_i-1: goes from z_i-1 to z_i

- d_i: indicates the distance from x_i-1 to x_i along z_i:
- theta_i: around z_i: from the Angle of x_i-1 to x_i

Here is an article to refer to https://blog.csdn.net/hitgavin/article/details/104442034

4.2 SDH Parameter List

关节	theta	d	alpha	a	offset
1	theta_1	173.9	PI/2	0	0
2	theta_2	0	0	-135	-PI/2
3	theta_3	0	0	-120	0
4	theta_4	88.78	PI/2	0	-PI/2
5	theta_5	95	-PI/2	0	0
6	theta_6	65.5	0	0	0

← Previous Page | Next Chapter →

User Notes



This section is crucial for every user of this product and must be read carefully. It includes essential information on product use, transportation, storage, and maintenance to ensure safety and efficiency during operation. Additionally, it outlines liability for product failure or damage resulting from non-compliance with these guidelines. The User Notice is divided into subsections, each providing detailed guidance on different topics:

Chapter Summary

- Safety Information

  Includes liability, safety warning signs, general safety rules, personal safety, and emergency response.

- Transportation and Storage

  Describes packaging, transport, and long-term storage requirements, along with liability.

- Maintenance and Care

  Offers guidance on routine maintenance to extend product lifespan.

- FAQs and Solutions

Provides a navigable guide for quickly resolving common issues.

By thoroughly reading this section, users will better understand how to use the product safely and efficiently, maximizing performance and lifespan.

If you have already read all the content in this chapter, please proceed to the next chapter.

← Previous Chapter | Next Chapter →

Safety Instructions

1 Synopsis

This chapter details general safety information for personnel performing installation, maintenance, and repair work on elephant robots. Read and understand the contents and precautions in this chapter before carrying, installing, and using it.

2 Hazard identification

The safety of cooperative robots is based on the proper configuration and use of robots. Furthermore, injury or damage caused by the operator may occur even if all safety instructions are followed. Therefore, it is very important to understand the safety risks of robot use in order to prevent them.

Table 1-1 to 3 lists the common security risks that may occur when robots are used：

Table 1-1 Risk level Security risks


1 Personal injury or robot damage caused by improper handling of the robot.
2 If the robot is not fixed as required, for example, the screw is missing or the screw is not tight, or the locking capacity of the base is insufficient to support the robot to move at high speed, the robot will fall over, resulting in personal injury or robot damage.
3 The robot's safety function fails to play its role due to the incorrect configuration of safety functions or the lack of safety protection tools.

Table 1-2 Warning security risks


1 Do not stay within the moving range of the robot when debugging the program. Improper safety configurations may not prevent collisions that may cause personal injury.
2 The robot's connection to other devices may lead to new hazards, requiring a thorough re-assessment of the risk.
3 Scratches and punctures are caused by sharp surfaces such as other equipment or robot end-effector in the working environment.
4 The robot is a precision machine. Stepping on the robot may cause damage.
5 If the clamping device is not in place, or the power supply of the robot is turned off, or the object is not removed before the air source (it is not determined whether the end-effector is firmly clamped because the object loses power), it may cause danger, such as damage to the end-effector and injuries to people.
6 The robot is at risk of accidental movement. Do not stand under any axis of the robot under any circumstances!
7 The robot is a precision machine, which may cause vibration and damage to the internal parts of the robot if it cannot be placed smoothly during handling.
8 The robot is a precision machine, which may cause vibration and damage to the internal parts of the robot if it cannot be placed smoothly during handling.

Table 1-3 Potential safety hazards that may lead to electric shock


1 Unknown hazards may arise when using non-original cables.
2 Electrical equipment contact with liquid may cause leakage hazard.
3 Electrical connection error may cause electric shock.
4 Be sure to switch off the power supply of the controller and related devices and remove the power plug before replacement. If the operation is carried out with power on, it may cause electric shock or failure.

3 Safety Precautions

The following safety rules should be followed when using the manipulator:

- The mechanical arm belongs to live equipment. Non-professionals are not allowed to change the circuit at will, otherwise it may cause damage to the equipment or human body.

- When operating mechanical arms, comply with local laws and regulations. The safety precautions and dangers, Warnings, and precautions described in this manual are only supplements to the local safety regulations.
- Please use the mechanical arm within the specified environment. Exceeding the specifications and load conditions of the mechanical arm will shorten the service life of the product and even damage the equipment.
- The person installing, operating and maintaining the myCobot arm, anyway, has to be rigorously trained on safety precautions and the right way to operate and maintain the robot.
- Anyway, don't use the product in humid environments for long periods of time. This product is a precision electronic component, which will damage the equipment in damp environment for a long time.
- Anyway, don't use the product in humid environments for long periods of time. This product is a precision electronic component, which will damage the equipment in damp environment for a long time.
- Highly corrosive cleaning is not suitable for cleaning the mechanical arm, and anodized parts are not suitable for immersion cleaning.
- Unconsciously, do not use the device without installing a base to avoid damaging the device or accidents, instead use the device in a fixed environment without obstacles.
- Do not use other power adapters for power supply. If the equipment is damaged due to the use of non-standard adapters, the after-sales service will not be included.
- Do not disassemble, disassemble, or unscrew the screws or shell of the manipulator. If disassembled, no warranty service can be provided.
- Personnel without professional training are not allowed to repair the faulty products and dismantle the mechanical arm without permission. If the products fail, please contact myCobot technical support engineers in time.
- If the product is discarded, please comply with the relevant laws to properly dispose of industrial waste and protect the environment.
- A child uses a device at some point, forcing someone to monitor the process and switch it off when it's finished.
- When the robot is moving, do not extend your hand into the movement range of the robot arm, for fear of collision.
- It is strictly prohibited to change, remove or modify the nameplate, description, icon and mark of the manipulator and related equipment.
- Please be careful in handling and installation. Put the robot gently according to the instructions on the packing case and place it correctly in the direction of the arrow. Otherwise, the machine may be damaged.
- **Do not burn other product drivers from Atom terminal, or burn firmware using unofficial recommendations. If the equipment is damaged due to the user burning other firmware, it will not be in the after-sales service.**

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

https://www.elephantrobotics.com

Please do not use the mechanical arm for the following purposes：

- Cost of healthcare in life-critical applications.
- Buying a bus can cause an explosion in an environment.
- Lent is used directly without a risk assessment.
- Cost of using a security function at a low level.
- Lo-fi does not conform to the use of robot performance parameters.

Robot Motion Control Guide and Usage Restrictions

Basic Usage Guidelines

myCobot 320 Usage Guidelines

- **Payload Capacity:** Please note that the myCobot 320 has a rated payload of 1 kilogram. To ensure the normal operation of the robotic arm, avoid exceeding the rated payload to prevent potential damage.
- **Power Supply:** For stable power supply to the robotic arm, use only the original matching power adapter. Do not replace or modify the power supply for safety reasons.
- **Disassembly and Maintenance:** When disassembling or performing maintenance on the robotic arm, strictly follow the guidance provided by the official after-sales support. Unauthorized modifications, such as connecting sensors, welding, or short-circuiting IO pins, are not recommended.
- **Operating Environment:** To ensure the normal functioning of the robotic arm, place it in a dry and temperature-appropriate environment. Avoid exposing the robotic arm to damp or high-temperature conditions.
- **Joint Limitations:** The robotic arm is equipped with joint limit functions. Avoid moving the robotic arm beyond its movable range to prevent potential damage or safety issues.
- **Fixation and Prevention of Falls:** Before operating the robotic arm, ensure it is securely fixed. During operation, be cautious to prevent falls or interference from unexpected situations.
- **Firmware Flashing:** Choose firmware carefully and only use firmware provided by the official source. Using unofficial firmware may lead to unnecessary issues.

Note: For your safety and the well-being of the robotic arm, it is recommended to adhere to the above usage guidelines. Any actions, such as overloading or modifying the robotic arm, that result in damage and are not guided by official after-sales support will not be covered by the robotic arm's warranty. If you have further questions, feel free to contact our official after-sales support.

Robot Motion Control Instructions

myCobot 320 **Joint Limitations are as Follows:**

Joints	Range(°)
J1	±165°
J2	±165°
J3	±165°
J4	±165°
J5	±165°
J6	±175°

The range of Cartesian space (coordinates) is as follows:

axis	Cartesian Space Range(mm)
x	-350~350
y	-350~350
z	-41~524
rx\ry\rz	+179°

Note: The above data is for reference only. During the actual operation of the robotic arm, certain positions or orientations may lead to self-interference due to the inherent structure of the robotic arm. For instance, at a specific position, while individual joints may not exceed their limits, it could result in a collision between the J2 and J3 joints of the robotic arm.

Robot Motion Control Tips:

myCobot ##Motion Control Suggestions:

- **Before controlling the robotic arm's motion, ensure you know the specific angles/coordinates of the target position. Exercise caution while controlling the robotic arm's motion.**
- **During the debugging phase, try to avoid running the robotic arm at high speeds to prevent potential harm to yourself and the robotic arm. It is recommended to use lower motion speeds.**
- **To prevent self-interference (collisions between the robotic arm's own joints), ensure that the robotic arm does not collide during its motion.**

Note: We recommend following the above motion control suggestions to minimize potential risks. Please be aware that if damage to the robotic arm is caused by personal use issues, warranty services will no longer apply. If you have any questions or need support, feel free to contact our official after-sales support. We will be happy to assist you in ensuring the safe operation of the robotic arm.

How to Properly Perform Robot Motion Control

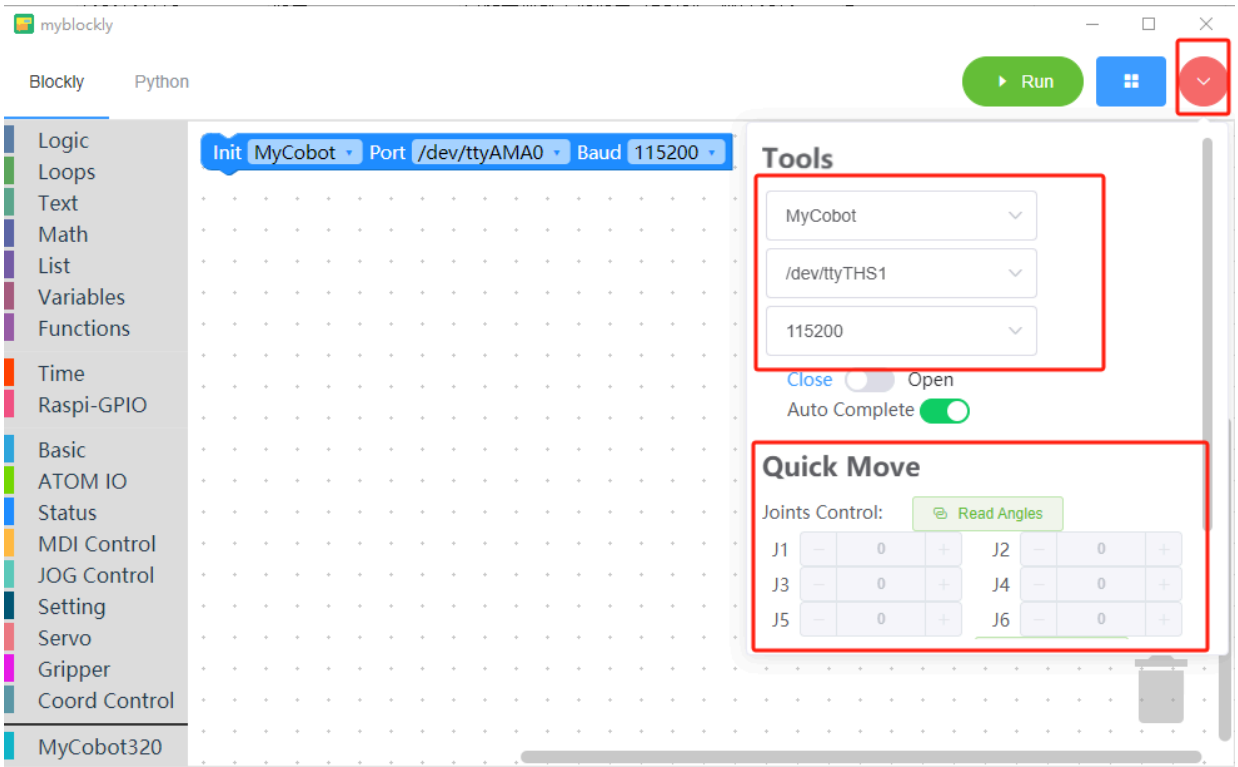
When unsure about the target position's angles/coordinates, you can use the following method:

Use a Python script to obtain:

```python
# Importing the Official Python API
from pymycobot.mycobot import MyCobot
# Importing the Time Module
import time

# Setting up Serial Connection, Serial Port, Baud Rate
# Raspberry Pi Version
mc = MyCobot('/dev/ttyAMA0', 1000000)
# M5 version, specific serial port number needs to be checked in Device Manager
mc = MyCobot('COM0', 1000000)
# Set a slight waiting time, 0.5 seconds
time.sleep(0.5)
# Release all joints of the robotic arm, please support the robotic arm by hand
mc.release_all_servos()
# Set waiting time, can be adjusted as needed; at this point, the robotic arm can be moved to the target position
time.sleep(5)
# Power on the robotic arm and fix it in the target position
mc.power_on()
# Read the coordinate information and angle information of the current position and output to the console
print('Coordinates:', mc.get_coords())
print('Angles:', mc.get_angles())
```

Obtained using myBlockly:

By utilizing the rapid movement feature in myBlockly, you can explore the motion range of the robotic arm, confirming the operational workspace of the robotic arm.



← Previous Page | Next Page →

Transport and Storage

Writing。。。

← Previous Page | Next Page →

Maintenance and Care

As a robotics manufacturer, we emphasize the importance of our customers being able to properly and safely maintain and upgrade their robotic equipment. Therefore, we provide a detailed maintenance and care guide, including common maintenance tasks and sections on repairing or upgrading hardware. Please read carefully for:

1 Recommended cycles.

Maintenance Item	Description	Recommended Frequency
Visual Inspection	Check for obvious damage, foreign object accumulation, or wear.	Daily
Structural Cleaning	Clean robot structural parts with a clean, dry cloth. Avoid moisture and corrosive cleaners.	Daily
Fastener Check	Check and tighten all bolts and connectors.	Daily
Lubrication	Lubricate joints and moving parts with manufacturer-recommended lubricant.	Every 3 months
Cable and Wiring Check	Inspect cables and wiring for damage or wear.	Monthly
Electrical Connection Check	Ensure all electrical connections are secure and undamaged.	Monthly
Software Update	Check and update control software and applications.	As updates are available
Software Data Backup	Regularly backup key software configurations and data.	Quarterly
Firmware Update	Regularly check and update firmware for new features and security patches.	As updates are available
Sensor and Device Check	Inspect sensors and other critical devices for proper operation.	Monthly
Emergency Stop Function Test	Regularly test the emergency stop function to ensure its reliability.	Monthly
Environmental Conditions Monitoring	Monitor temperature, humidity, dust, etc., in the working environment to meet robot operating specifications.	Continuous monitoring
Safety Configuration Review	Regularly check and confirm the robot's safety configurations, such as speed limits and work range settings.	Monthly
Preventive Maintenance Plan Execution	Perform regular checks and maintenance according to the manufacturer's maintenance plan.	According to manufacturer guidelines

2 Repair and Modification

We understand customers may need to upgrade or repair their robot hardware themselves. Before any upgrade, please read product specifications carefully and confirm with our officials if such operations are allowed. Unauthorized operations may lead to product failure and are not covered by the warranty.

Material Requirements

- **Officially Manufactured or Recommended Materials**: All parts and components for repairs and upgrades must be officially made or recommended by us.
- **Material Procurement**: Customers can purchase necessary materials through our official channels to ensure quality and compatibility.

Repair or Upgrade Process

- **Customer Repairs**: Customers are responsible for completing repairs, guided by our detailed manuals.
- **Follow Official Guidance**: Repairs must adhere strictly to our official guidance to prevent damage.

Liability and Warranty Policy

- **Liability Division**: Manufacturer provides official guidance and deals with defects; customers are responsible for following this guidance using official parts.
- **Warranty**: Valid only when repairs follow our guidance using official parts.

Considerations

- **Safety First**: Follow all safety guidelines before any repair or upgrade, including powering off and using proper protective gear.
- **Technical Support**: If issues arise during repairs, contact our technical support team for help.

Customers are strongly advised to follow these guidelines to ensure the safe and effective operation of their robot equipment.

← Previous Page | Next Page →

Common Problem

In this part, some common driver-related problems, software-related problems and hardware-related problems are listed.

1 First Time Self Check

2 Software Problem

3 Hardware Problem

4 Accessory Problem

5 Other Problem

If you have purchase intention or any parameter questions, please send an email to this mailbox.

E-mail : "sales@elephantrobotics.com"

If the listed problems can't help you solve and you have more after-sales questions, please send an email to this mailbox.

E-mail : "support@elephantrobotics.com"

← Previous Page | Next Chapter →

Self-check for the first use - machine joint function test

> **Note:** When starting the robot arm, please be careful not to let the robot arm be in a curled-up or touching posture 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:

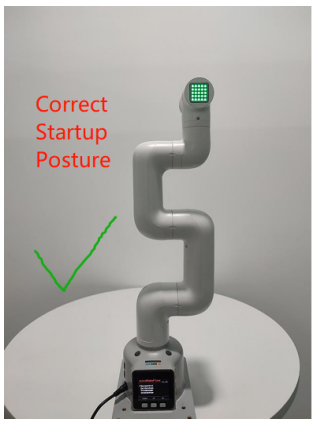
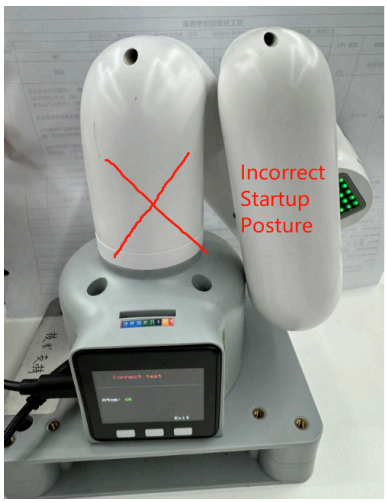
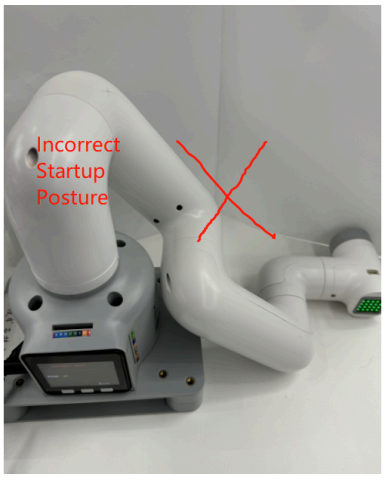
		

Figure 1 (correct posture)	Figure 2 (wrong posture)	Figure 3 (wrong posture)

Joint control method and steps

1. Hardware connection

- Hardware connection of M5 series machines:

For mycobot320M5 machines, you need to make sure that the power adapter, USB data cable and emergency stop switch are connected, and make sure that the emergency stop switch is in the released state. If the emergency stop switch is not used correctly, mycobot320 cannot be used normally. Please refer to the following figure for the emergency stop switch:



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. Choose the correct communication method

Before using each communication method, you need to make sure that the LCD screen of M5 is adjusted to the corresponding mode and maintain this communication state to control the robot arm normally. When using myblockly, python, ros and other development methods for the M5 robotic arm, you need to ensure that the M5 LCD screen stays on the Atom: ok interface, as shown below:



Note: When the screen displays Atom: no, you need to power on the machine again and check according to the self-check steps of the hardware-related "How to solve the problem of the robotic arm not being able to lock when powered on" in this article

4. USB communication example

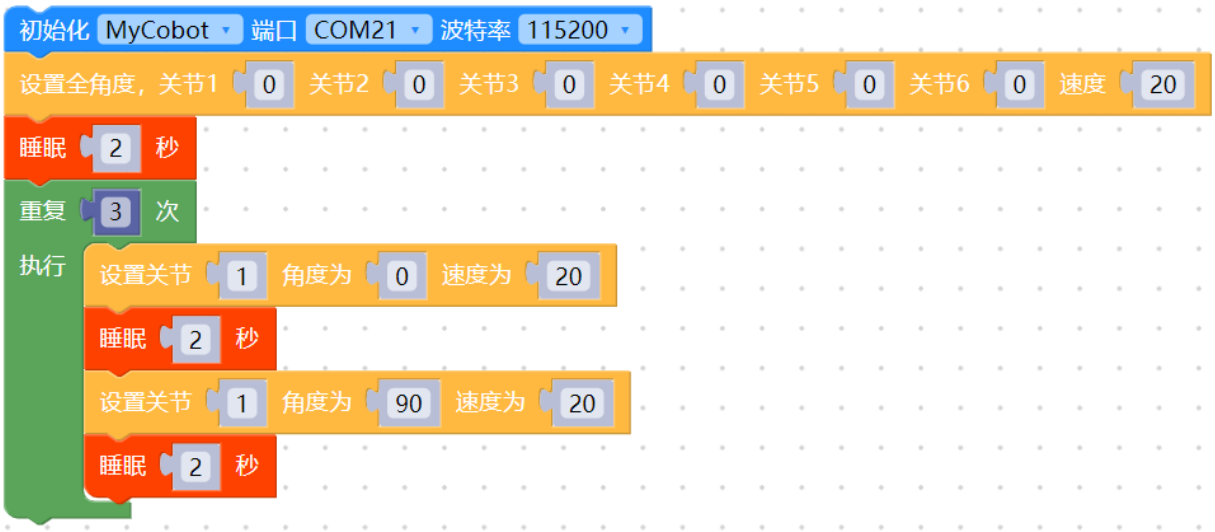
Please use myblockly or python source code examples to verify the joint motion of the robotic arm.

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:

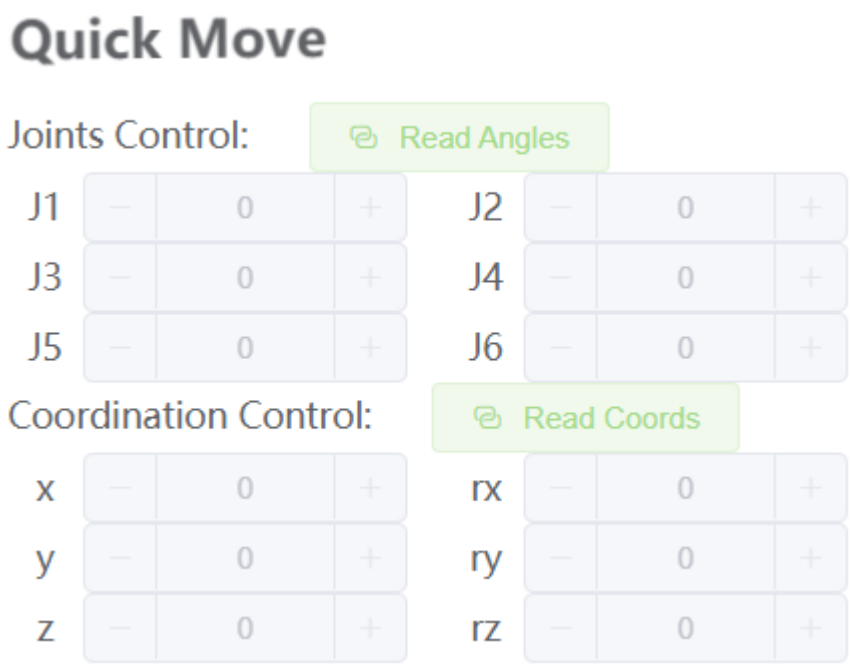
Machine model	Serial port number	Baud rate
320 M5	Win: COM; Linux: /dev/ttyUSB	115200

Note: Regarding the selection of the COM port of the M5 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

4.1 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 movement control. For specific usage, please refer to: Myblockly fast movement tool usage



4.2 Robot arm joint movement joint python source code

```
#The movement effect is that the robot arm moves around the zero position, and the 1-6 joints move one by one by ±20 degre
import time
from pymycobot.mycobot320 import MyCobot320

if __name__ == "__main__":
cobot = MyCobot320('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, 1-6 joints move one by one ±20 degrees, indicating that joints 1-6 respond normally. You can learn to use other cases in gitbook step by step or use the robot arm to do various interesting things!

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

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 robotic arms launched by our company.

Q: What is the method for troubleshooting 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 first to download the firmware.
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 is burning the nimirobot firmware, use a USB cable to connect the side interface of the M5stack to the USB port of the computer. When the 320M5 machine is burning the PICO firmware, you need to use a USB cable to connect the PICO interface of the base to the USB port of the Raspberry Pi.
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: Why does the device not work properly after I burn the firmware to the ATOM terminal?

- A: The firmware of the ATOM terminal needs to use our factory firmware. Other unofficial firmware cannot be changed during use. If the device accidentally burns other firmware, you can use "myCobot firmware burner" to select ATOM terminal-select serial port-select ATOMMAIN firmware to burn the ATOM terminal.

Q: Can the drag teaching in the firmware record the gripper action?

- A: It is temporarily impossible to use drag teaching to record the gripper action, because the gripper belongs to joint number 7, and our drag teaching can only record and play the movement of joints numbered 1-6.

Q: Why can't drag teaching be performed after burning the minirobot firmware?

- A: First check whether the M5Stack-basic firmware and atom firmware are burned, whether the burned firmware corresponds to the requirements to be implemented, and whether the burned firmware is the latest version of the firmware.
- It is recommended to burn the minirobot firmware to version v2.1 and the top atommain firmware to version v4.1 and above (need to support mystudio version v4.3.1 and above).

Q: What should I do if mystudio cannot recognize the serial port of mycobot?

- A: If your computer device does not prompt the connected robot arm, please install the serial port driver first.

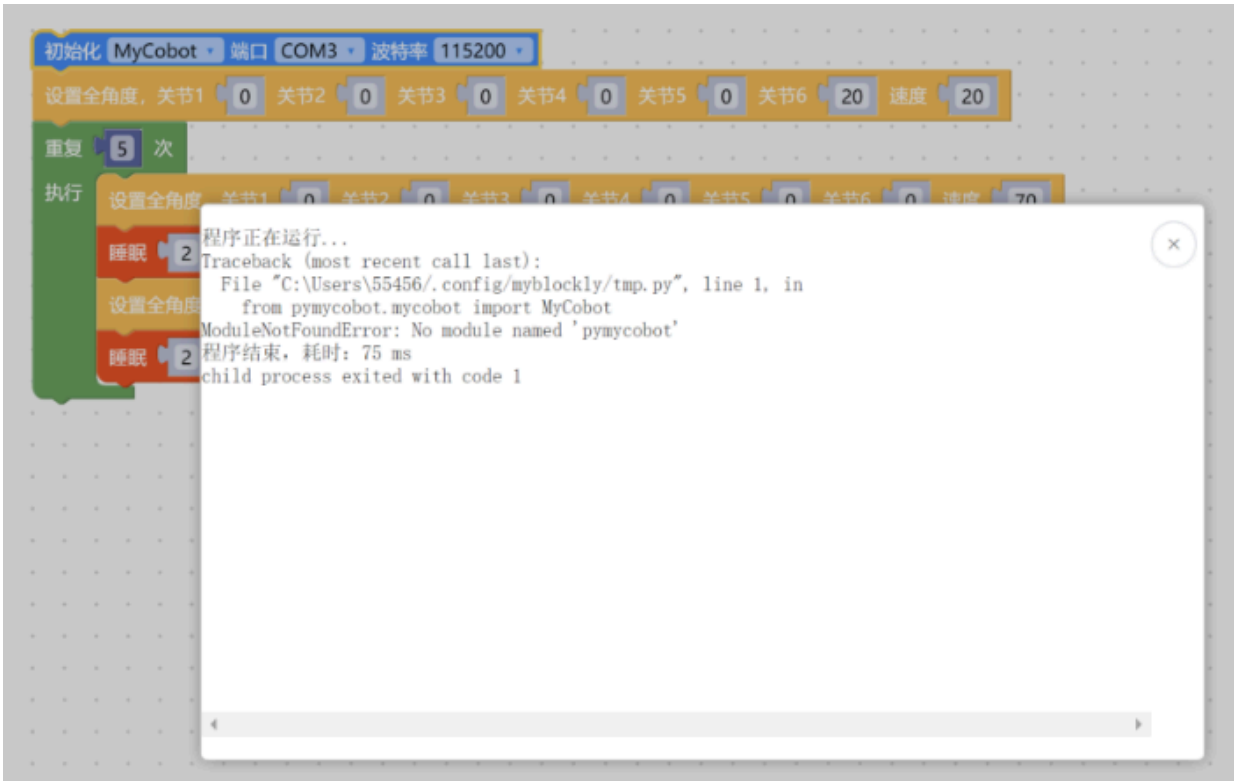
- In addition, it should be noted that the Raspberry Pi, Arduino and Jetson nano series robot arms are **cannot be connected to the laptop using a data cable**, and you need to use mystudio in the built-in system to burn the firmware.

Q: Can I save the trajectory recorded by dragging teaching to the card?

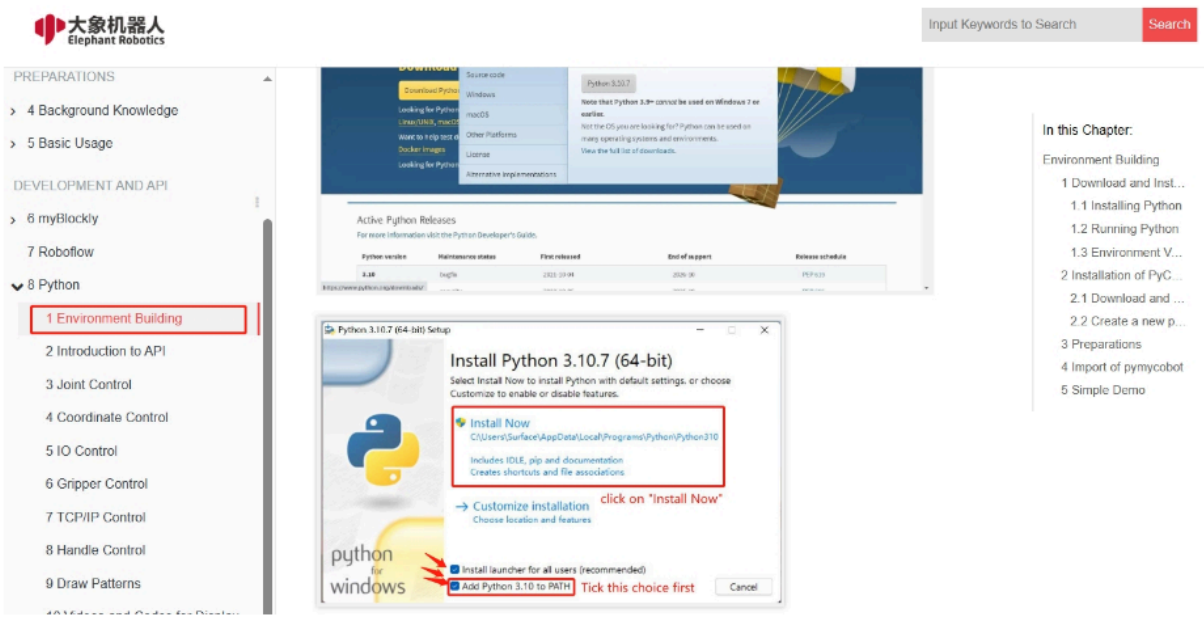
- A: Currently, it cannot be saved to the memory card. And dragging teaching can only save one path at a time, and the next recording will overwrite the previous action.

2 myblockly related

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



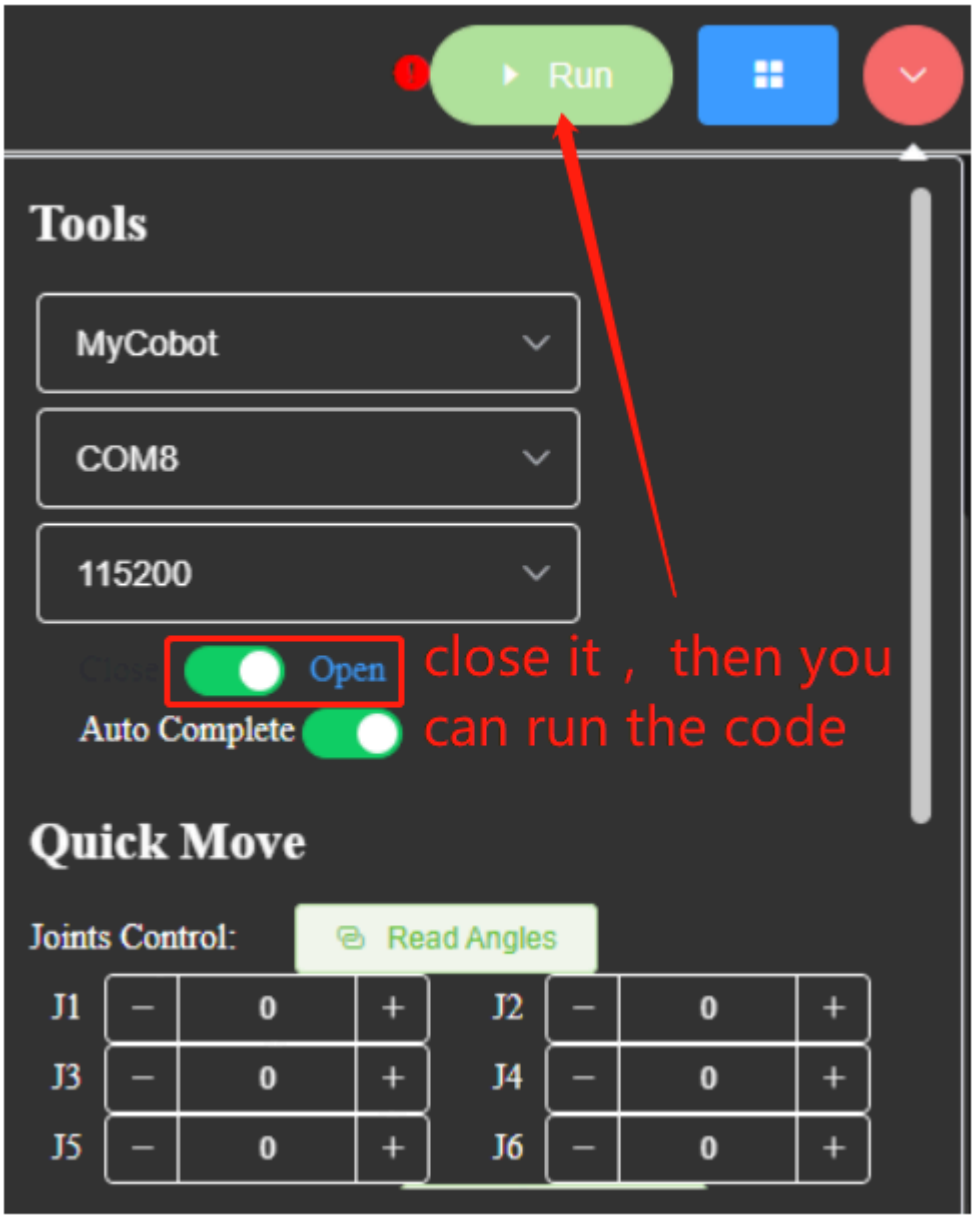
- A: The error message indicates that the pymycobot file is missing. For the reasons and solutions, please refer to the following 3 points: ①If pymycobot is not installed or pymycobot fails, the corresponding solution is to reinstall pymycobot. The command is pip3 install pymycobot --upgrade --user ②For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



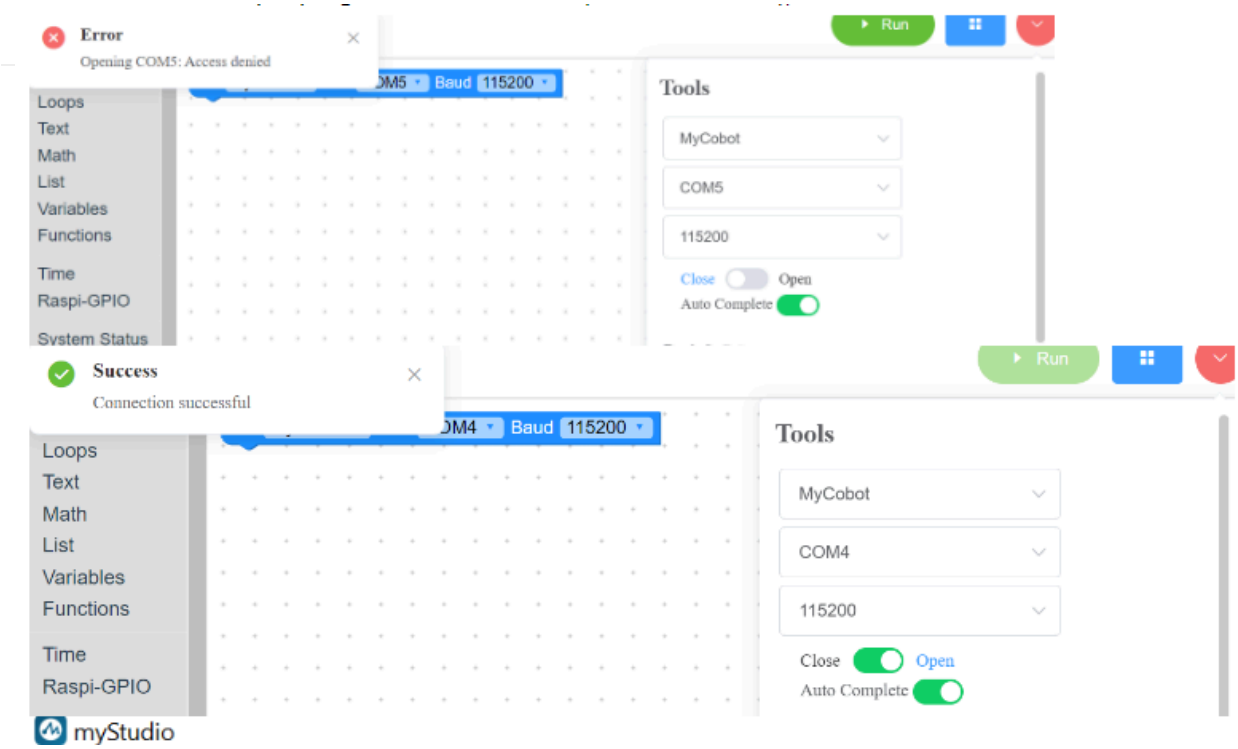
③If it is an M5 or AR series machine, please confirm whether there are multiple python versions in the PC. It is recommended to uninstall all python versions in the PC and reinstall a python version above 3.8. Note that there is only one python version above 3.8 in the PC. If multiple python versions are actually required, please specify the python version used by pymycobot and specify the python version when calling the pymycobot library

Q: Cannot click to run myblockly, how to solve it?

This is because the serial port is occupied and the run button cannot be clicked. You need to check whether the serial port tool of myblockly and other serial ports are occupied. If you want to run the code, you need to close the button in the figure below before clicking run. You can only run after closing the serial port tool, otherwise there will be a serial port conflict.



Q: Why is the connection rejected when selecting a certain com port? Or how to find the corresponding com port?



The reason for the connection rejection is due to the wrong com port selection. When you have multiple devices connected to the computer USB, multiple serial ports will be displayed in myblockly, such as com4 and com5 in the above picture, but only one of them is the robot arm. You need to select the serial port of the robot arm to connect and use the robot arm normally. Obviously, com4 that can be connected normally is the serial port number corresponding to the current robot arm. Regarding how to find the serial port corresponding to the robot arm among multiple serial ports, the corresponding method is: try to unplug and plug the serial port cable connected to the robot arm, and check which serial port number disappears in the serial port number option of myblockly after disconnecting the USB connection between the robot arm and the computer. When the USB is used again to connect the robot arm to the computer, this serial port number appears in the serial port number option of myblockly. The serial port number that disappears and reappears in myblockly synchronously with the disconnection and connection of the robot arm and the computer is the serial port number corresponding to the robot arm. Note that the options for the robot arm's com port number are not always fixed. They may change when connected to different USB ports of different computers or different USB ports of the same computer. It is recommended to view the real-time com port number using the above method.

Q: What should I do if myblockly's express delivery mobile tool cannot display the real-time angle?

- A: This is generally caused by incorrect selection of device serial port information and abnormal pymycobot. It is recommended to troubleshoot 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/1yBWzhbSBUYsZPBl7PBdZKRwk3al71Dc7/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 run successfully.

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 for the time being, but you can make and save an initialization json file yourself. Load this file after entering myblockly next time to get the preset code block. For the method of making and saving json files, please refer to the following: https://drive.google.com/file/d/1g_dd933TK1tptnisUad4PBfwSRsWWFeQ/view?usp=sharing

3 RoboFlow related

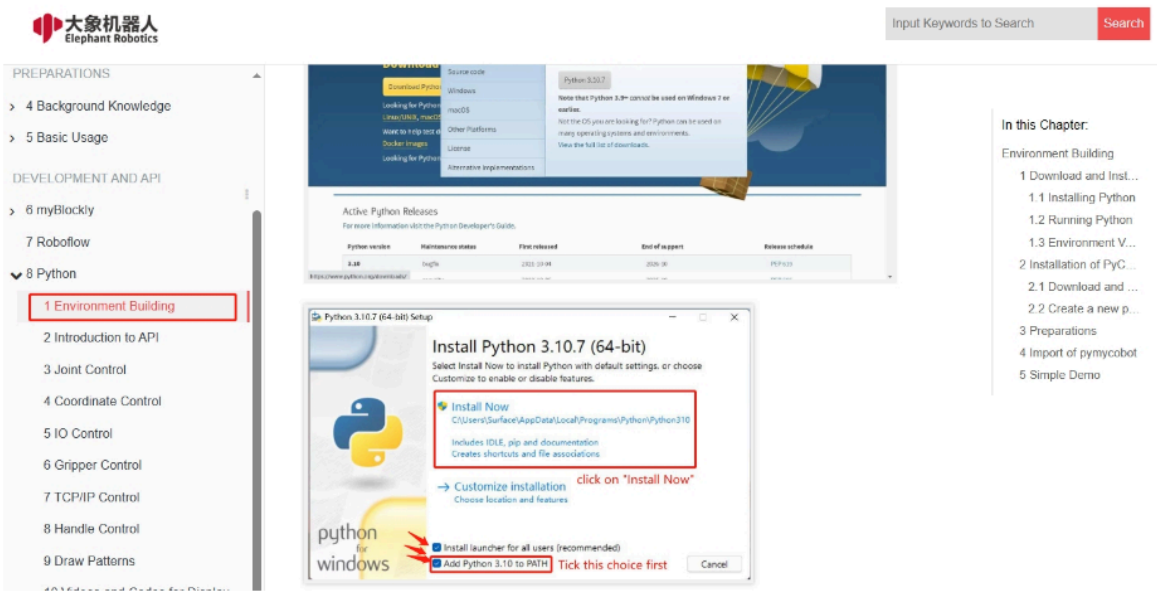
Q: What should I do if I cannot download the Roboflow software and Roboflow cannot control the machine normally?

- A: Currently, Roboflow software only supports the two Pro professional collaboration models 600/630, and no longer supports mycobot collaborative models or other models. The recommended control methods for mycobot series machines are myblockly, python and ros. It is worth mentioning that myblockly is a software with a similar graphical interface to Roboflow. If you need to use visual graphical programming, you can give priority to using myblockly software.

4 Python related

Q: The running prompt shows that the library file is missingQ: 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 Pythonxx 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 needed 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 pyhton, as pyhton12 will be incompatible.

Q: Why does coordinate control sometimes fail to respond after writing coordinates?

1. Delays are required before and after joint movement to ensure that the robot has enough response time
2. Coordinate movement must first ensure that the coordinate can be reached through joint movement. Generally, the coordinate value is read after the joint moves to the specified point for control. Most of the manually written coordinates are invalid. It is recommended not to write the coordinates directly by yourself, but to release the joint and manually rotate the joint to the target position, use get_coords() to record the target position coordinates, and then use send_coords() to set the coordinates

You can refer to the following code:

```
# Import the official python API
from pymycobot import MyCobot320
# Import the time module
import time


# Set the serial port connection, serial port, baud rate
# PI version
mc = MyCobot320('/dev/ttyAMA0', 115200)
# M5 version, the specific serial port number needs to be checked in the device manager
mc = MyCobot320('COM0', 115200)
# Set a short waiting time, 0.5 seconds
time.sleep(0.5)
# Release all joints of the robot arm, please hold the robot arm with your hands
mc.release_all_servos()
# Set the waiting time, which can be changed as needed. At this time, the robot arm can be moved to the target position.
time.sleep(5)
# Re-power the robot arm and fix it at the target position
mc.power_on()
# Read the coordinate information and angle information of the current position and output them to the console
print('coords:', mc.get_coords())
print('angles:', mc.get_angles())
```

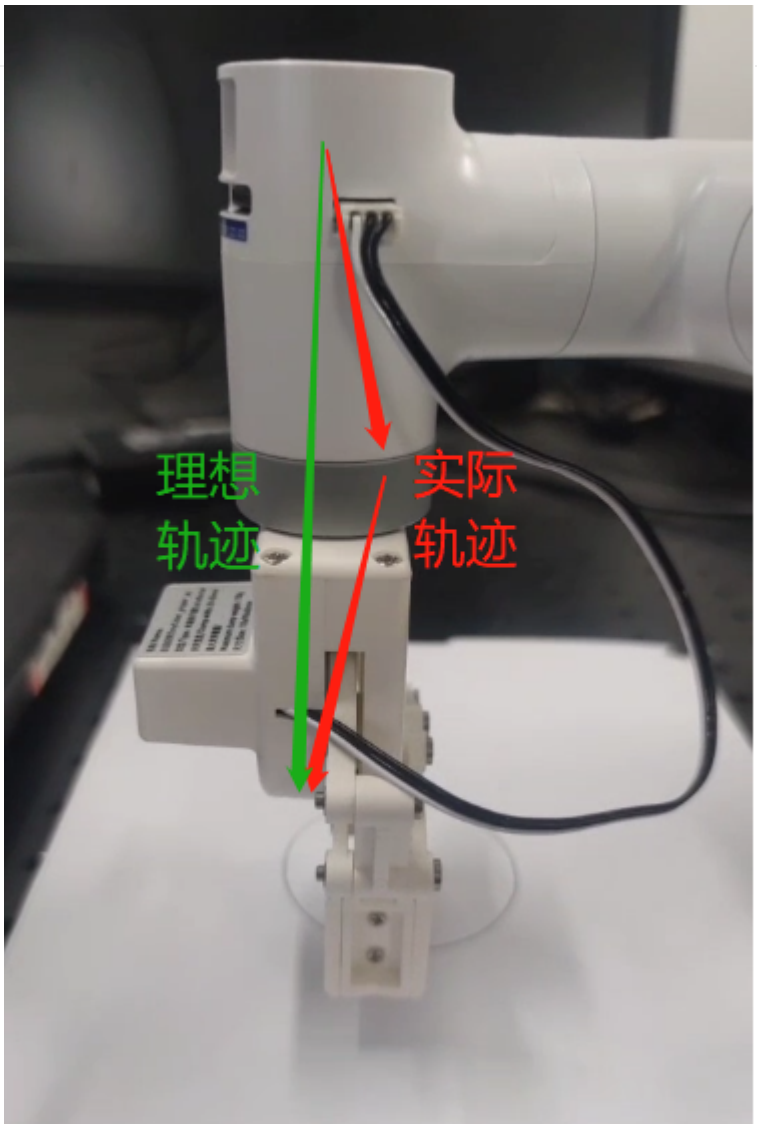
Q: Is there a more popular explanation of the mode in send_coords(coords, speed, mode)?

- 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 limit, structure, etc., the instruction 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 instruction 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 middle segment trajectory. 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: When only the Z axis is changed, the trajectory is not straight up and down, but the final landing point is that only the Z axis is changed. Is this normal? How to ensure that the middle trajectory is also a straight line?



- 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 interpolation is turned on. 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 intermediate segment. Non-interpolation means that there is no planning of the intermediate segment and the trajectory cannot be controlled.

Q: What does the return value of get_error_information() of -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: The drawing case using the 280 machine 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 and hard stationery to use this drawing case. There are two main reasons for this deviation. First, because mycobot uses a servo motor, there is a certain accuracy deviation (if it is a machine that has been used for a long time, due to the aging of the joints, the deviation of its joints will be greater). 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 stuck, so the drawing effect is not ideal. It is currently recommended to use soft stationery for painting, such as brushes and other tools, which will help improve the painting effect.
- A2: In addition, you can change the robot's motion mode 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 in a straight line
```

Note that the intelligent planning route set in send_coords will only be useful after interpolation is enabled. Interpolation means that a lot of dense points are planned between the starting point and the end point to achieve the effect of controlling the middle segment trajectory.

Q: The target position identified cannot be reached at the end. How to determine whether this coordinate can be reached and then processed?

- A: solve inv kinematics(target coords, current_angles) Use this interface 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 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_CUgDE46rVn7zbd0EhraIbgt3qZt/view?usp=sharing

ROS1 virtual machine file download

Download ROS2 virtual machine file

Download virtual machine software VirtualBox

Q: How to deal with the error when importing ROS2 virtual machine file?



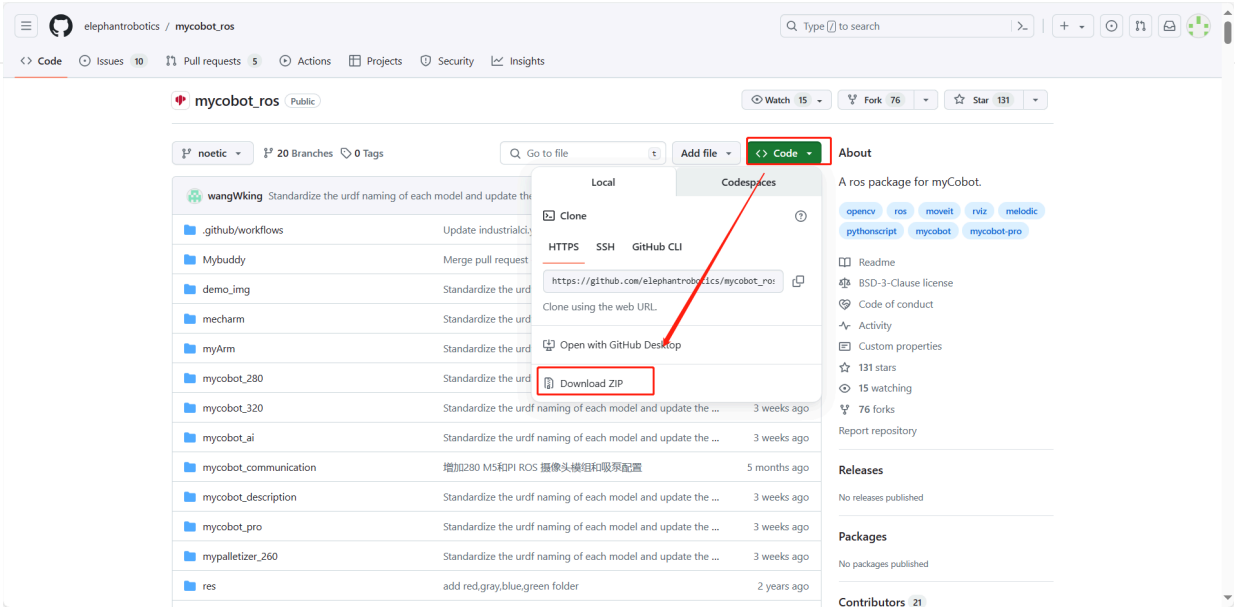
- A: This is because the version of the virtual machine software Oracle VM VirtualBox is too low, and the virtual machine software version needs to be updated.

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

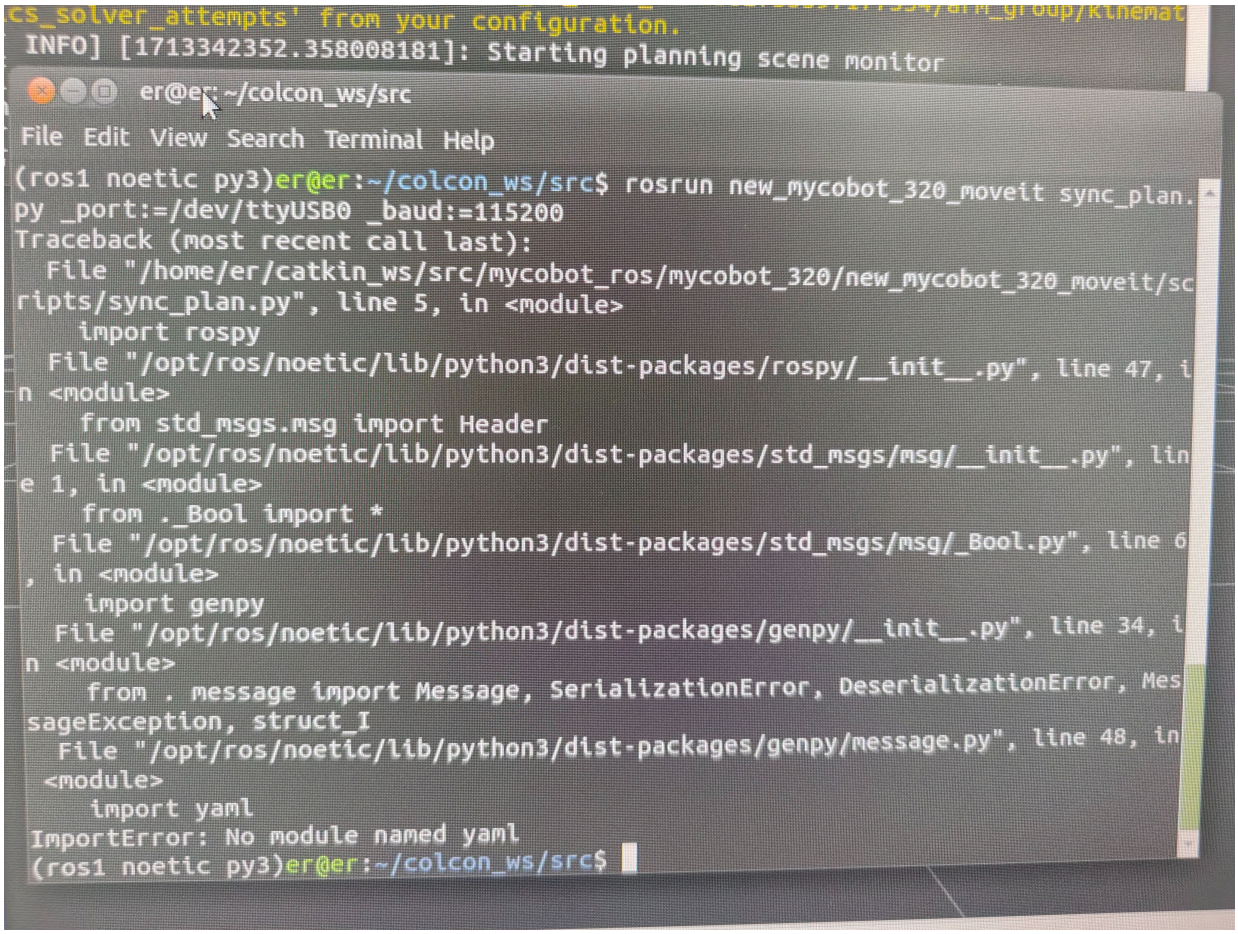
- A: Use the command to pull:

```
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. Source code package address: https://github.com/elephantrobotics/mycobot_ros



Q: What should I do if I get an error ImprotError: No module named yaml when running the ROS moveit case?

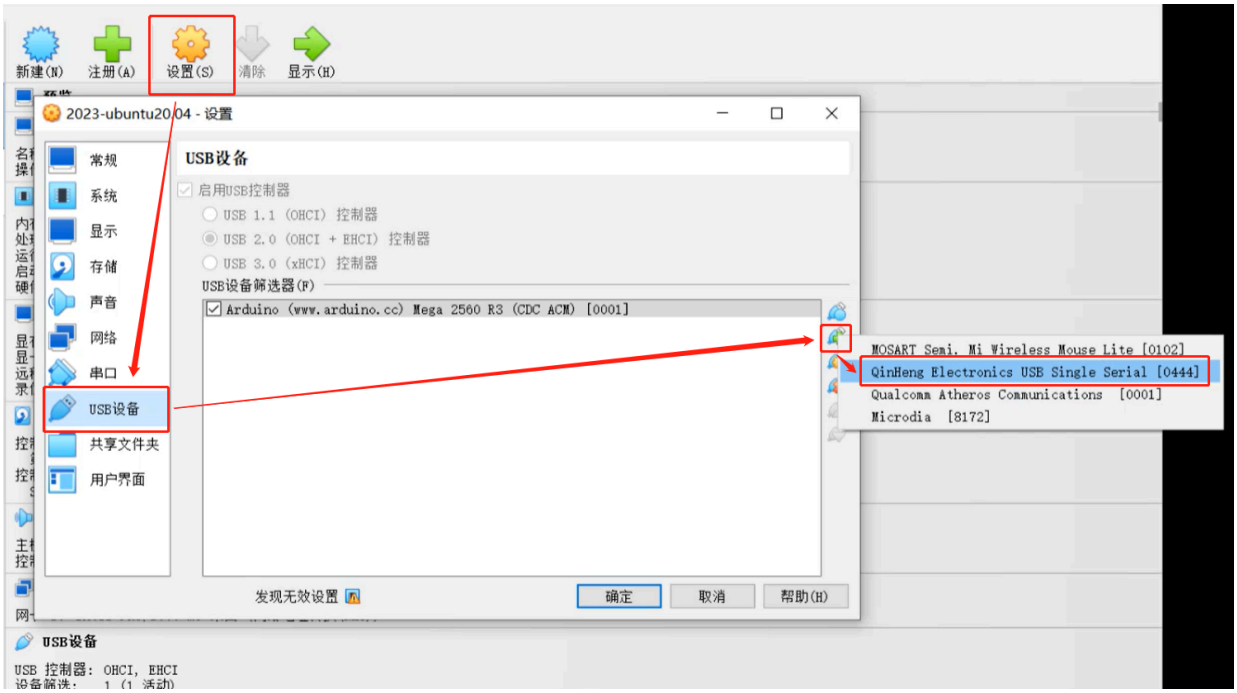


- A: In the first line at the beginning of this script, change the Python interpreter to python3

Q: What should I do if I can't find the serial port 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.

The serial port number that changes after plugging and unplugging is the corresponding machine serial port device number



Q: Use a mujoco-based environment for simulation training, so 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 them use meshlab to convert them 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 rosrun is running, the terminal reports an error message `counld 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 rosrun 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 ros program, you need to open the ros node. Enter `roscore` in the terminal.

Q: When rosrun is running, the terminal reports an error message `counld 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*` .

6 C++ related

Q: How to deal with various dll files that cannot be found?

- A1: If myCobotCpp.dll is missing, put the myCobotCpp.dl previously placed in the lib directory into the directory where mycobotcppexample.exe is located.
- A2: If QT5Core.dll is reported missing, open qt command (search QT in the menu bar), select msvc2017 64-bit, and execute windeployqt--release the directory where myCobotCppExample.exe is located (such as: windeployqt --release D:lvs2019myCobotCpploutlbuildlx64-Releaselbin) After executing the command here, if the vs installation path cannot be found, please check the settings of the vs environment variables.

After executing the above steps, if the qt5serialport.dll file is reported missing, move this file in the gt installation directory (path such as: D:lgt5.12.1015.12.10msvc2017 64bin), copy it to the directory where myCobotCppExample.exe is located

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

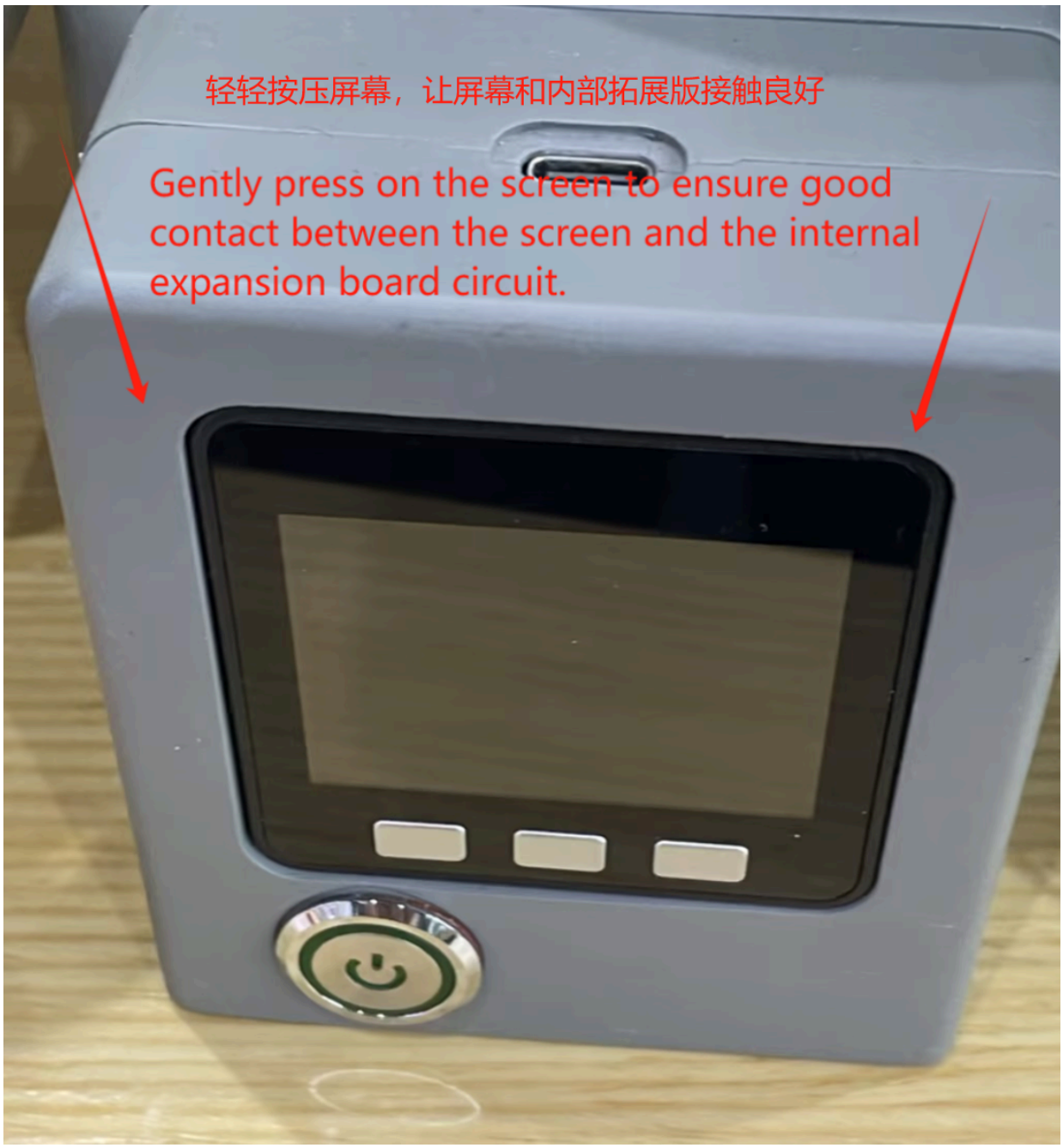
Select Start in the figure below



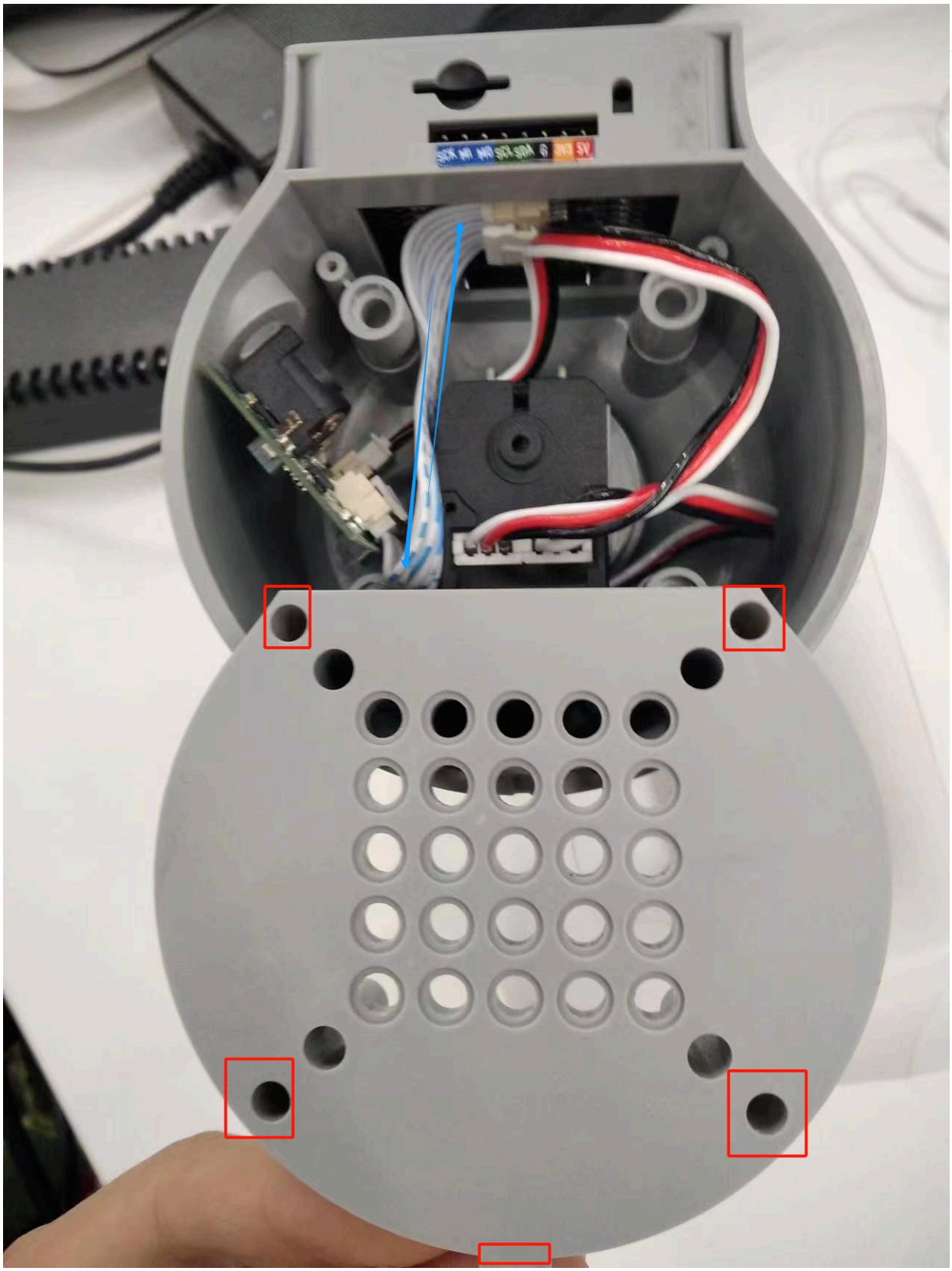
Hardware problem

Q: The M5 screen cannot display the picture, how to solve it?

1. Check whether the power adapter is in good contact with the machine, you can try to re-plug the power adapter
2. Gently press the corners of the screen to make the M5satck in good contact with the internal expansion board

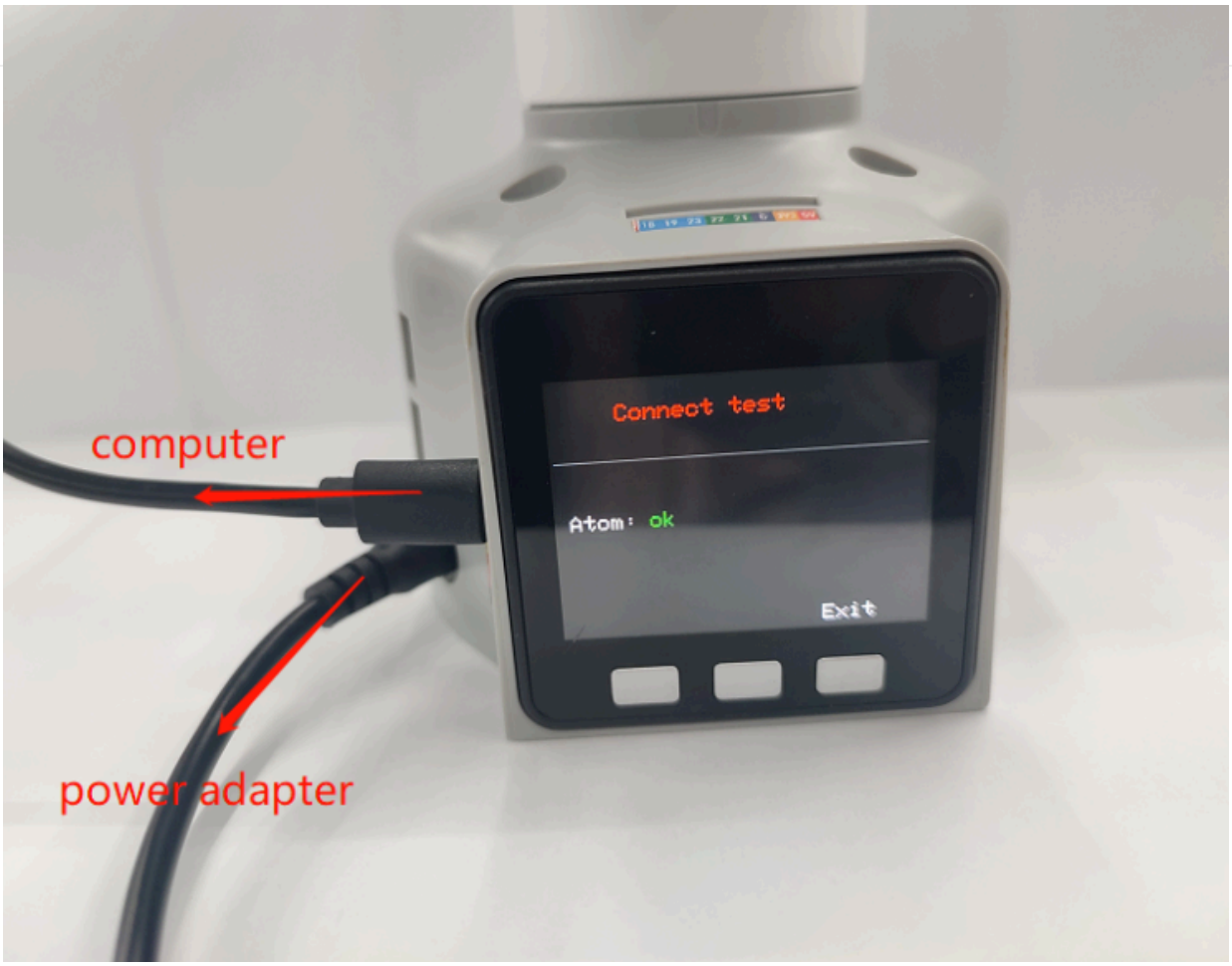


1. Check gitbook and download the corresponding minirobot firmware. Note that if it is 2021 280m5, it is recommended to flash AtomMain2.8, basic1.0
2. You can remove the base screws to check whether the internal cables are detached. If so, please connect them before use



Q: How to solve the problem that the robot arm cannot be locked by power?

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



1. Check whether the joint can rotate normally when it is powered 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. For 320 series products, it is necessary to check whether the emergency stop switch is in the released state. The joint can only be used normally in the released state. When the emergency stop switch is in the stopped state (the switch is pressed), it is impossible to power on and communicate normally. The switch needs to be turned clockwise to keep the emergency stop switch in the released state



1. Check the Atom firmware as follows:

Under normal circumstances, the robot arm will self-lock after power-on, and 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, Atom does not light up green or the joint cannot self-lock. You can follow the following points to troubleshoot: ① Gently press the Atom screen to make Atom in good contact with the internal plate of the robot arm.

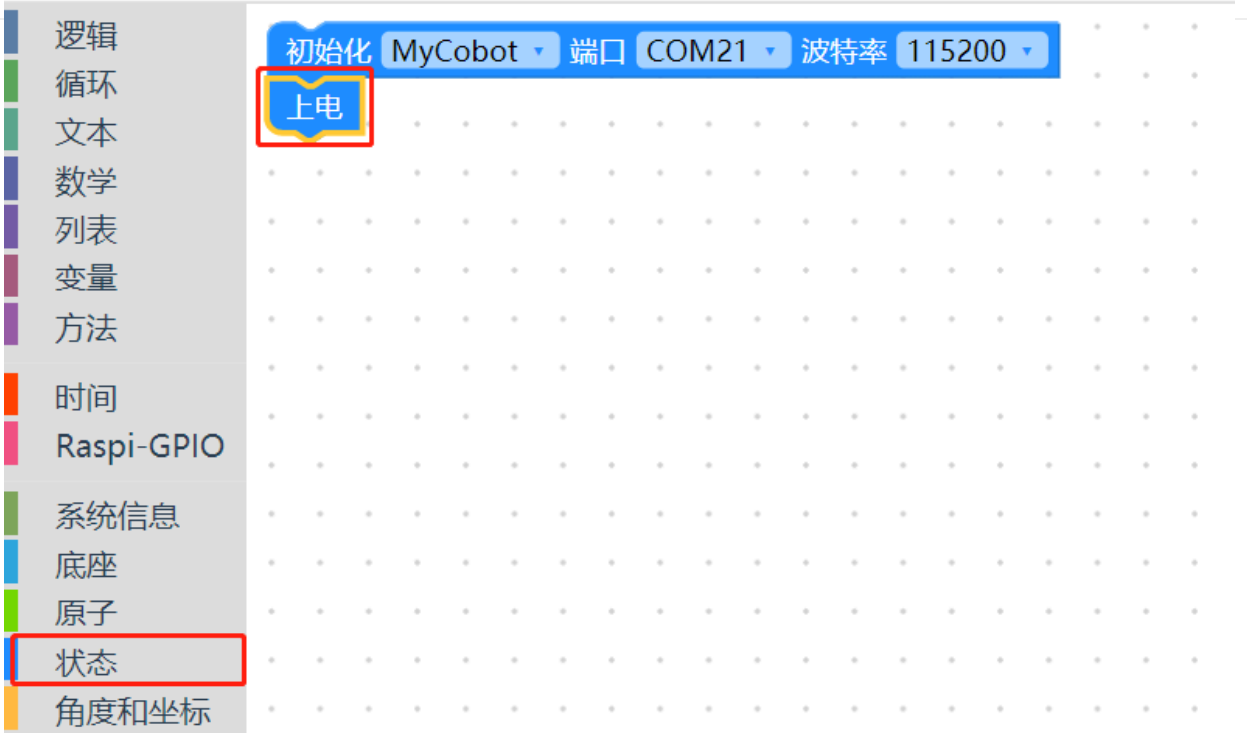
② Check gitbook to get the method of using mystudio. According to the model and version information, use mystudio to download the corresponding Atom firmware. If you encounter any problems during the burning process, please refer to this article "Firmware download exception" related to mystuidio to obtain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

③ After the Atom firmware is successfully burned and the robot 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 goes out after the type-c is unplugged, it is judged that Atom is normal, but there is a problem of line detachment or damage inside the robot arm, and it is necessary to contact the technician to deal with it.

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

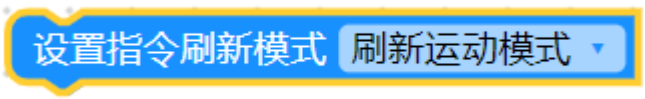
Q: After pressing the emergency stop, the emergency stop cannot be locked. How to lock the robot arm again?

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



Q: How to optimize joint shaking, excessive joint angle deviation or joint weakness and falling?

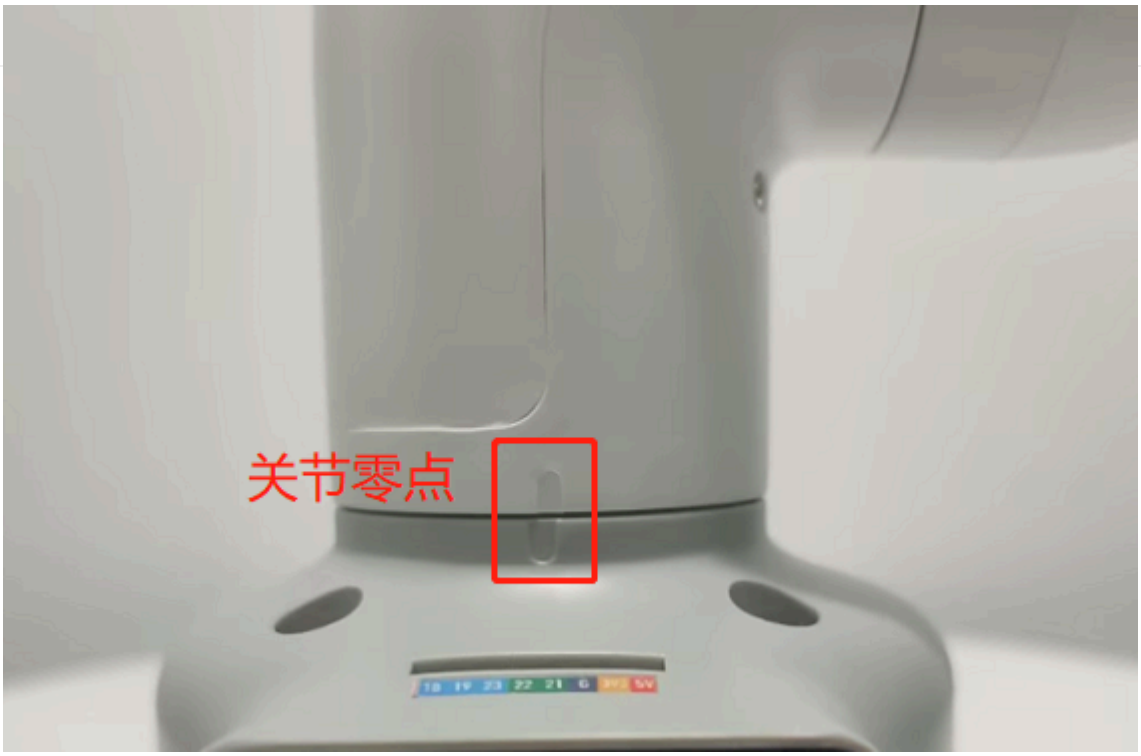
1. Refer to the robot parameter introduction section to check whether the actual load is within the effective load range of the robot arm. Excessive load will cause joint vibration. The load of the actual joint can be appropriately reduced.
2. Change the motion mode to refresh mode, so that the movement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



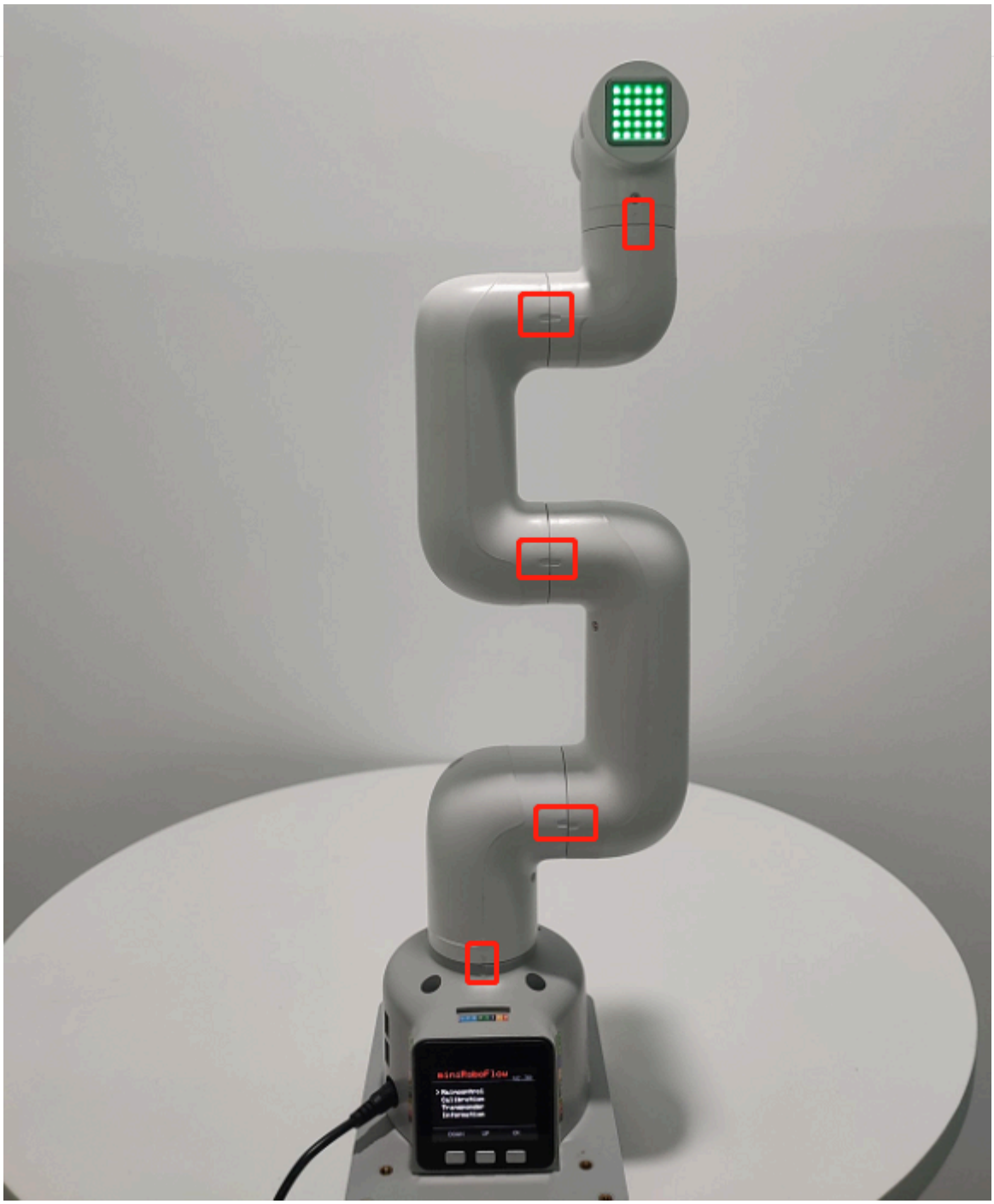
3. Check the following link to adjust pid: https://drive.google.com/file/d/1UWhaaSTuwLFImuEGY1J2tvgxTQDwWxK_/view?usp=sharing
4. Check the gitbook section and use mystudio to download the corresponding version of Atom firmware. It is recommended to download the latest
5. Check Chapter 5 of the gitbook 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-ykKWPH0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing
6. 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/1tXDUALmfw1z0u6lM9uH5hOHivjbRoWxW/view?usp=sharing
7. 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 zero point of the joint



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



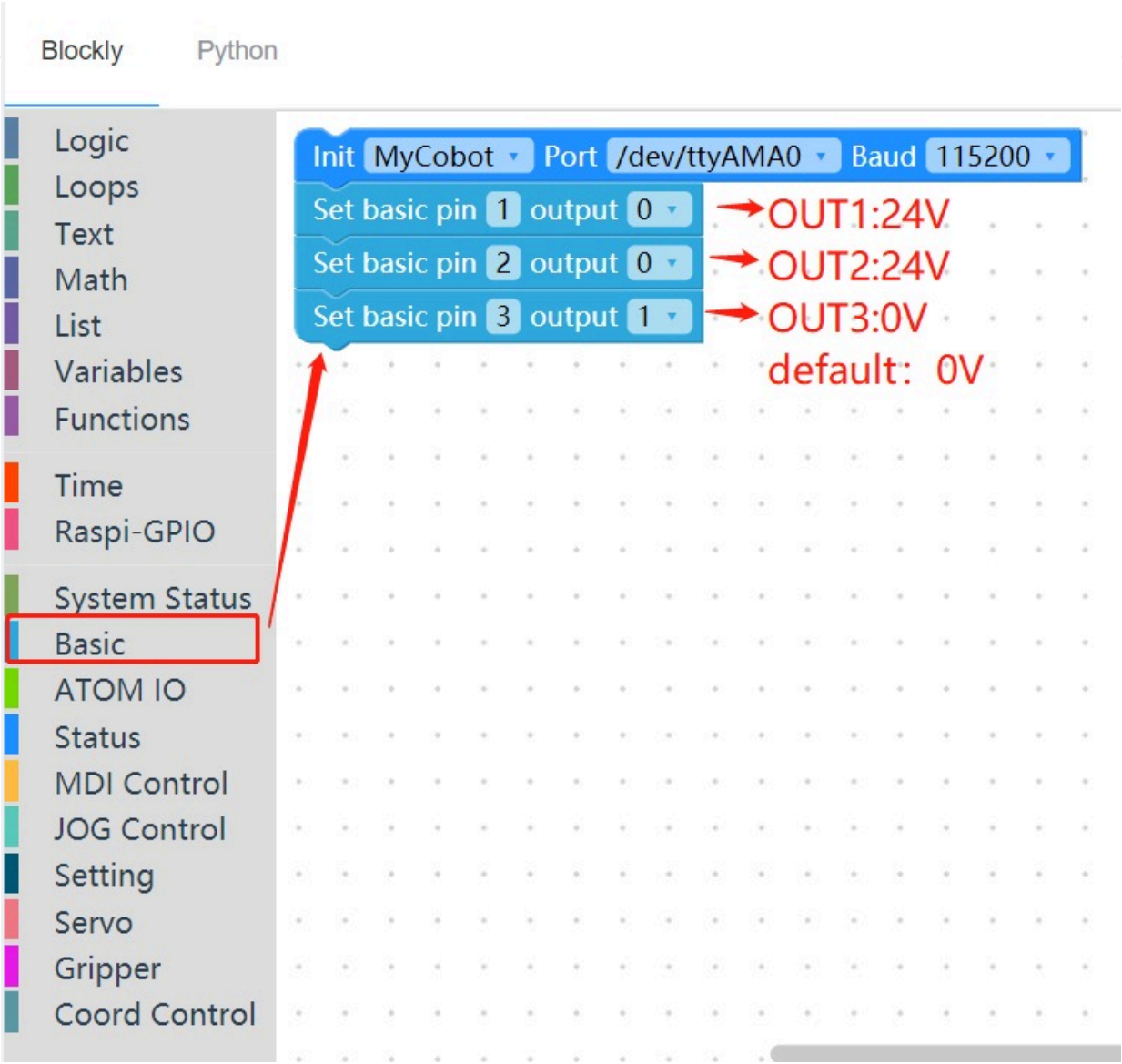
Q: Is there a method for zero point calibration?

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

https://drive.google.com/file/d/1XtKH-ykKWPH0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing

Q: How to use the GPIO at the bottom of the mycobot320 machine?

- A: Please refer to the following usage



Q: Is there any way to control the 5V LED light with the IO at the bottom of the mycobot320 machine?

A: Please refer to the example of the IO pin at the bottom of the Raspberry Pi, where the high level of the output is 24V. If you want to use 5V to control the LED, it is recommended to use it after connecting an external step-down module. At present, we have no case of using the bottom IO to control the LED

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 the required real-time control and multi-threading. The related programs of atom are not open source yet.

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

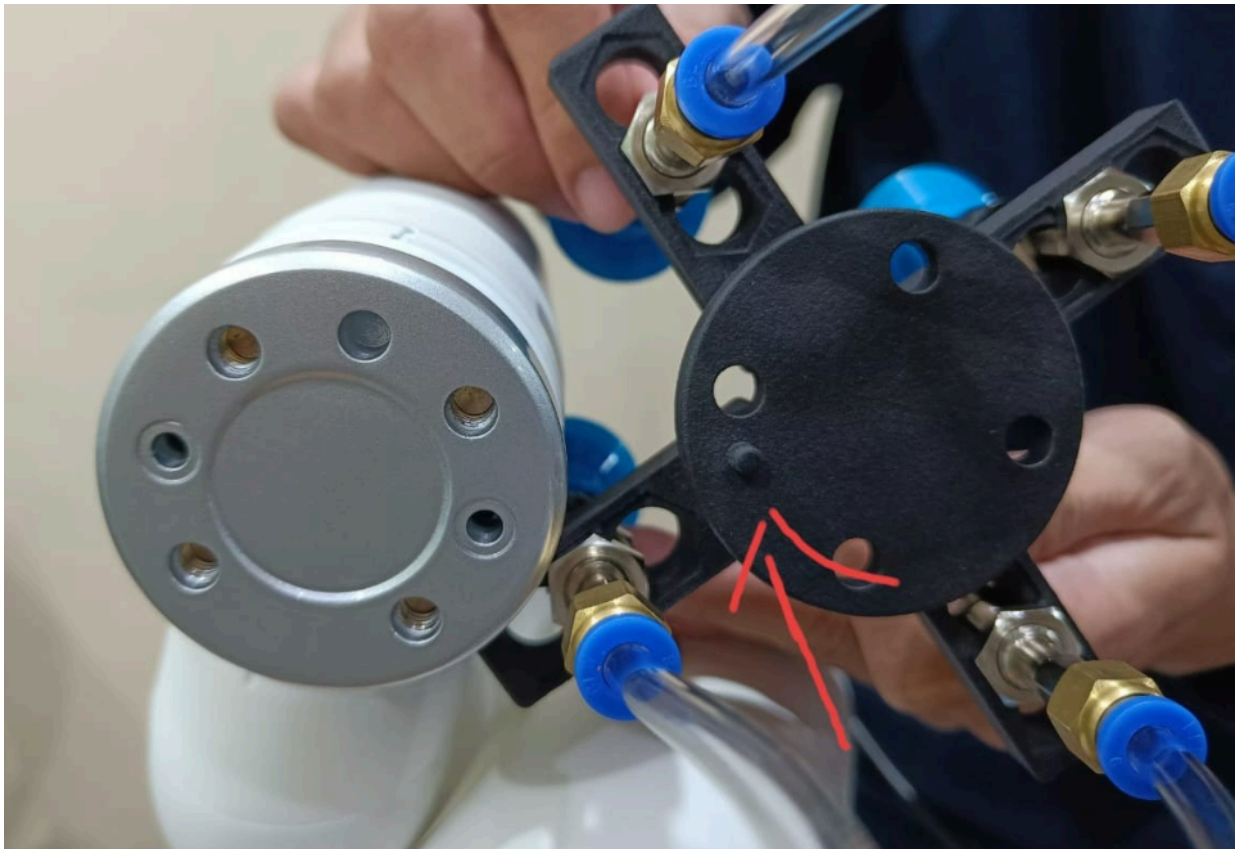
- A: The microprocessor-based robot supports socket communication TCP; the microcontroller-based robot can use USB to serial communication.

Q: Why does the motor automatically cut off during use?

- A: The motor is overheated after a long period of use. This phenomenon is normal and can be continued after waiting for a few minutes.

Accessory related questions

Q: Does the bulge pointed by the arrow on the modular suction cup need to be cut off



- A: It needs to be removed manually

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

Pin introduction of mycobot adaptive gripper refers to the figure below:



Gripper connection method:



Q: The value read by 320pro adaptive gripper using get_gripper_value() fails to read 0-100 correctly, and sometimes it is 255. Is this normal? Does pro adaptive gripper have an interface for reading angles?

A: Normal, currently 320pro adaptive gripper does not have an interface for reading angles, and get_gripper_value() is a dedicated angle reading interface for mycobot280 adaptive gripper.

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%.

Q: Is there a video of using 320 with cylinder and modular suction cup?

Reference link: https://drive.google.com/file/d/1Ei0JRjXn_YWDyYPPZeBVrAW_VzPUlx6e/view?usp=sharing

Q: Is there a video of using 320 with pneumatic gripper?

Reference link: https://drive.google.com/file/d/1nL4mgUf0OYOyCJPf4d5GNkWmbkuWBoup/view?usp=sharing

Q: Is there a video of using 320 with pro adaptive gripper?

Reference link: https://drive.google.com/file/d/1nL4mgUf0OYOyCJPf4d5GNkWmbkuWBoup/view?usp=sharing

Others

Q: How to use mycobot test tool?

Mycobot test tool is factory-installed and is not recommended for users. It may cause zero position or pid abnormalities and damage the machine. Please delete this tool directly. If the tool has been used to cause abnormal movement of the robot arm, please refer to the following to readjust the pid and zero position, and do not continue to use this factory test tool in the subsequent use process. Reference link for resetting pid method: https://drive.google.com/file/d/1UWhaaSTuwLFImuEGY1J2tvgxTQDwWxK_/view?usp=sharing Reference link for zero position calibration method: https://drive.google.com/file/d/1XtKH-ykKWPH0q9Z_YHwzkgwNKRhstHhi/view?usp=sharing

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

Restoring to factory settings mainly involves resetting 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/1UWhaaSTuwLFImuEGY1J2tvgxTQDwWxK_/view?usp=sharing
- **About resetting zero position**: Generally, when the machine has an incorrect zero position and 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 of 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

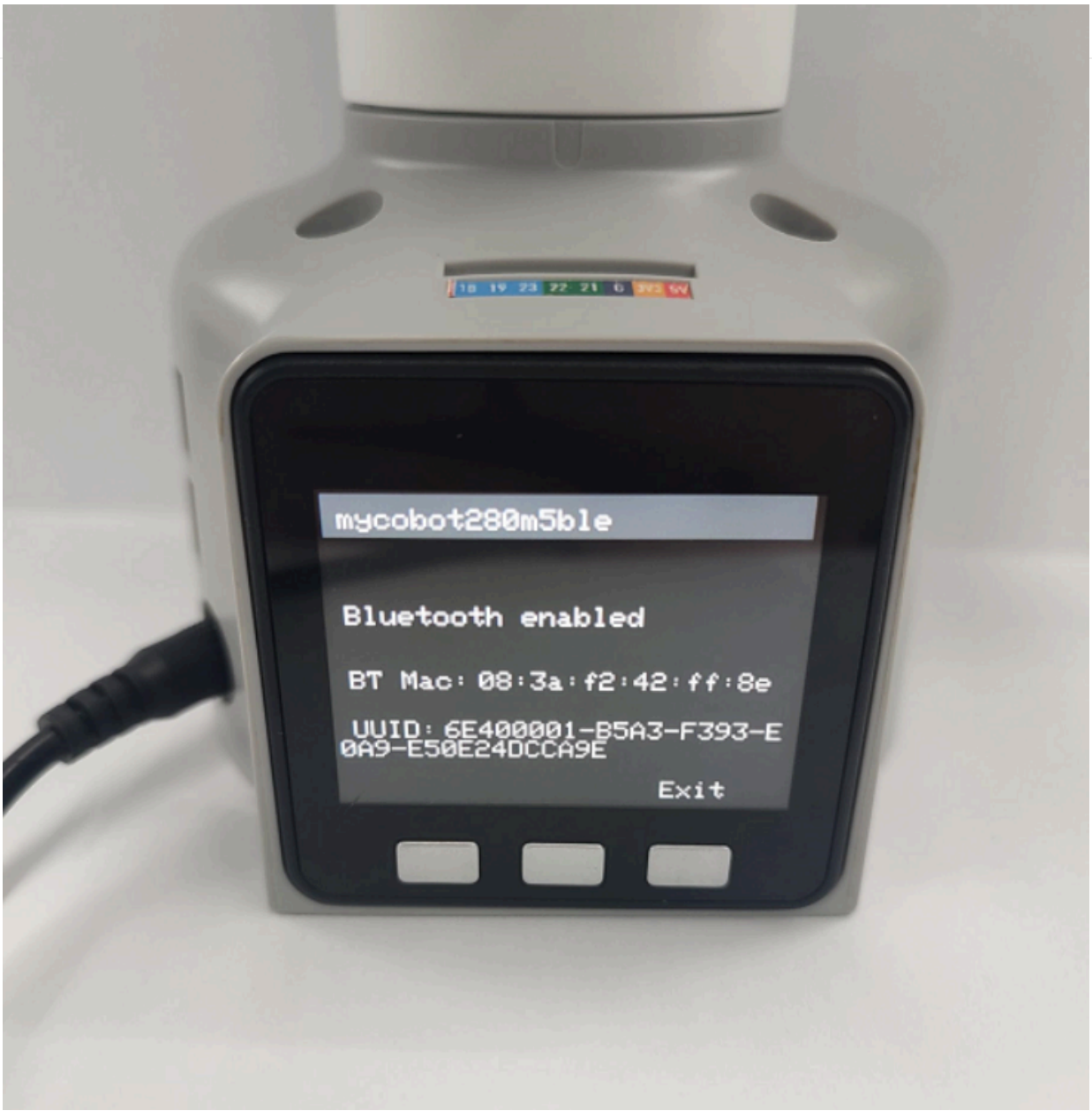
Q: How to troubleshoot when the robot arm cannot be controlled when using Bluetooth or WiFi function?

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. Before using each communication method, you need to make sure that the LCD screen of the M5 is adjusted to the corresponding method 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:

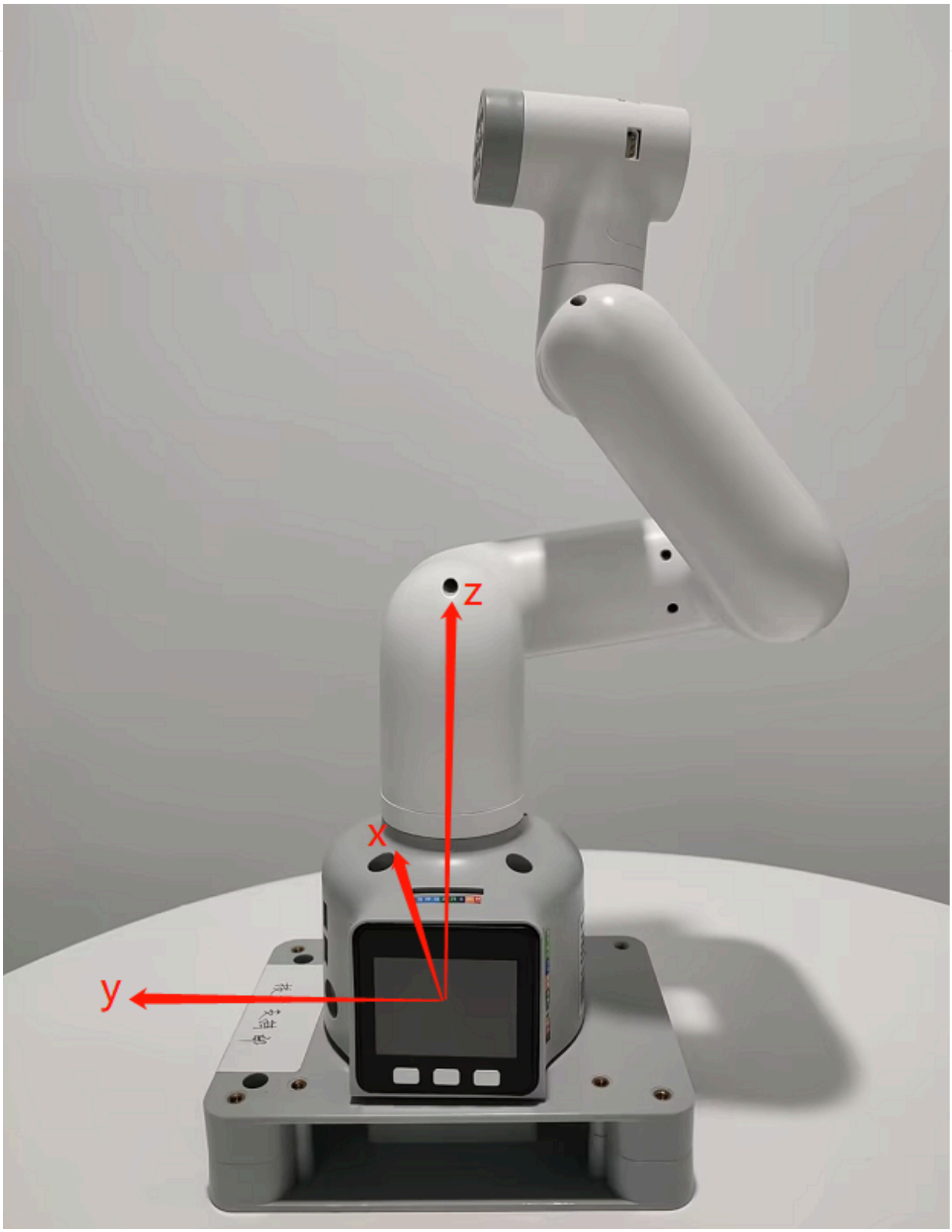


After ensuring that the communication mode status is selected correctly, try to control the robot arm again

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?



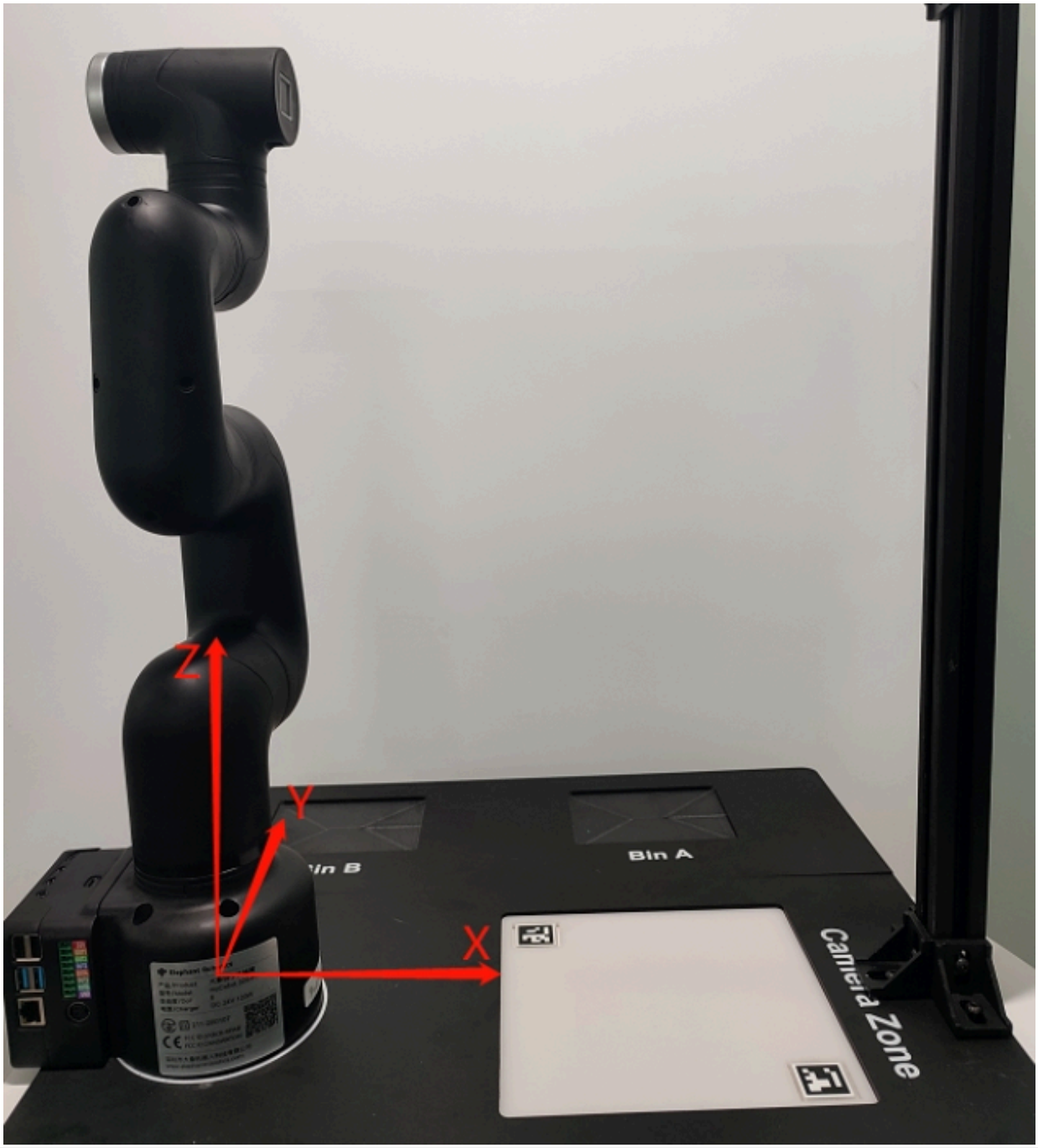
Q: Are the joints of 280 controlled by the serial bus?

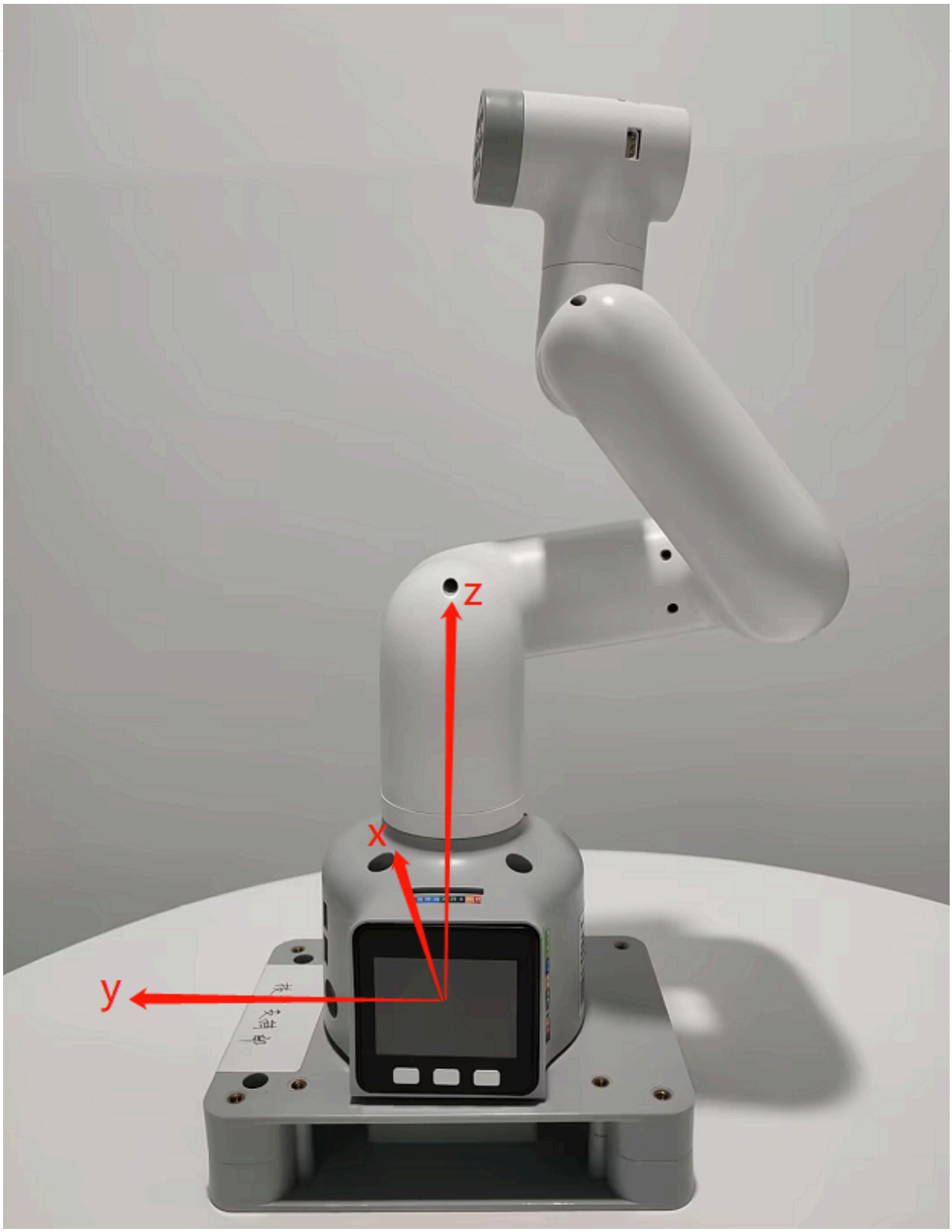
A: Yes

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:**

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.

Q: Which API does trajectory planning?

A: Please refer to the API for setting coordinates and joints: send_coords(), send_angles(), which comes with motion planning

Q: If the robot arm calls the API for controlling coordinates, will there be multiple solutions?

A: Because inverse kinematics has multiple solutions, the same coordinate position may arrive at different joint angles, which will occur.

Q: What is the maximum current output of the 24v output pin on the base of the mycobot 320 M5 2022 robot?

A: Maximum 2A, duration cannot exceed 1S

Q: If you want to control the angle and obtain feedback of each axis servo, 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 algorithm 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: What are the maximum and minimum joint movement speeds and coordinate movement speeds of 320? What is the acceleration limit?

A: Refer to the following:

Mycobot320				
Joint	Joint Minimum Value°	Joint Maximum Value°	Joint Maximum Speed°/s	Joint Maximum Acceleration°/s²
J1	-170	170	150	200
J2	-120	120		
J3	-148	148		
J4	-120	135		
J5	-169	169		
J6	-180	180		

Mycobot320				
Axis	Coordinate Minimum Value mm	Coordinate Maximum Value mm	Coordinate Maximum Speed mm/s	Coordinate Maximum Acceleration mm/s²
x	-350	350	100	400
y	-350	350		
z	-41	523.9		
rx	-180°	180°	40°	66°/s²
ry				
rz				

Q: What is the difference between the 320PI and 320M5 versions?

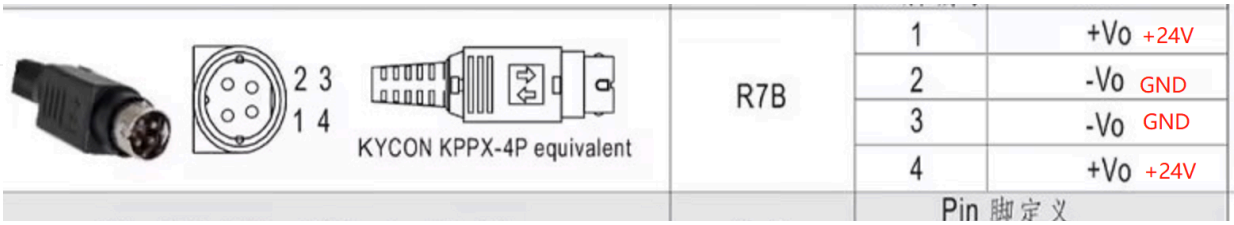
A: M5 is more suitable for beginners and can be directly connected to a computer for programming Pi has stronger computing power than M5 and cannot be directly connected to a computer for development. It is suitable for users who want to use it independently and connect to a display screen for development https://docs.elephantrobotics.com/docs/gitbook/2-serialproduct/2.0-m5-pi.html

Q: I need to repair the machine myself. Is there a repair video?

A: 320 disassembly and assembly video: https://youtu.be/zP2gvLD3kYs

Q: Schematic diagram of voltage parameters of 320 and mybuddy power adapter

A: As shown below



Q: Is the joint torque information provided?

A: Our machines only provide the overall information of the joints, not the internal torque, voltage and current of the servos and motor actuators. 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 figure below?

- **查看两个坐标系之间的关系**

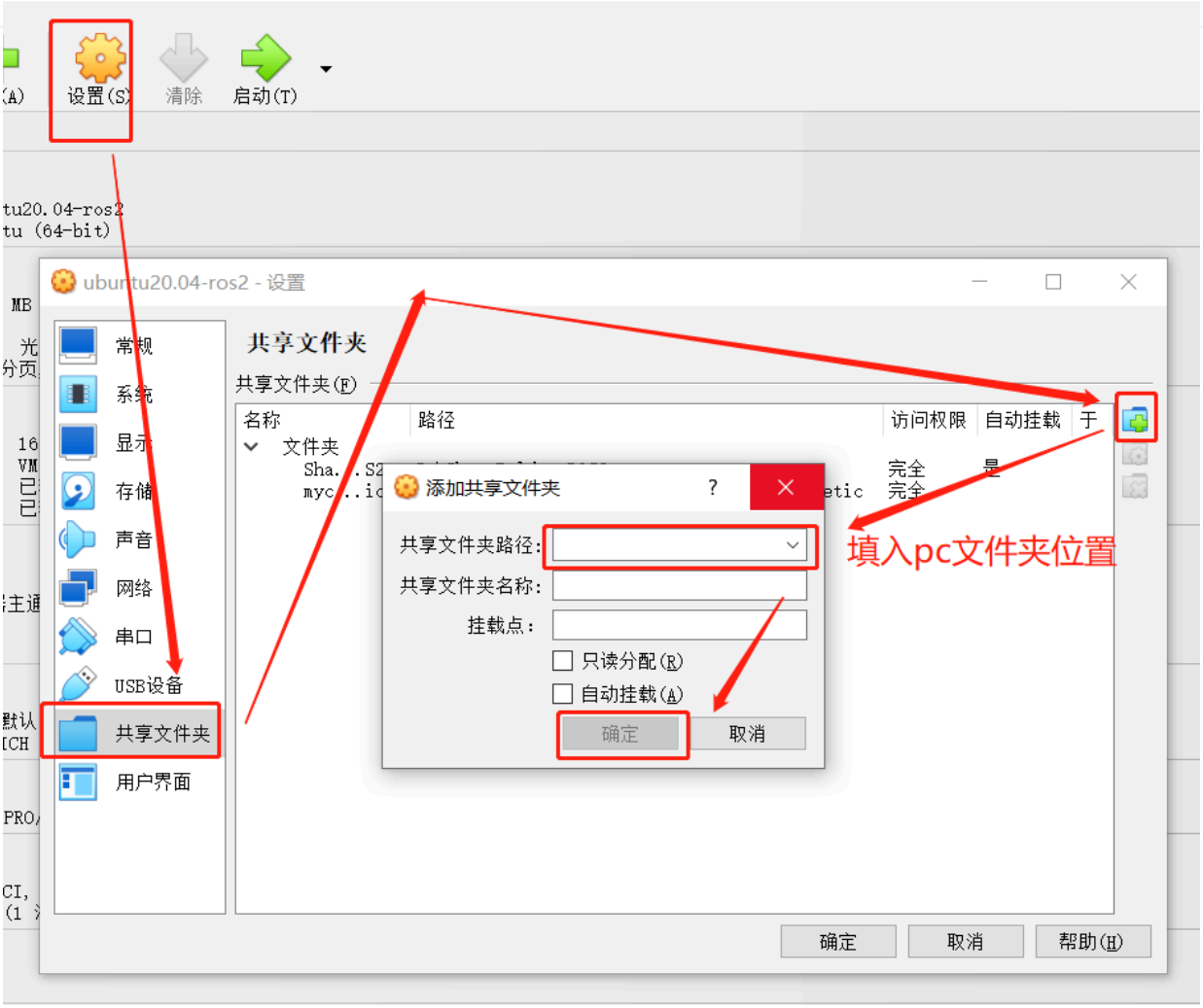
```
ros2 run tf2_ros tf2_echo [reference_frame] [target_frame]
ros2 run tf2_ros tf2_echo turtle2 turtle1
```

A: It means that 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/1UWhaaSTuwLFImuEGY1J2tvgxTQDwWxK_/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 instructions 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 instructions 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 instructions and reports an error



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

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: Overseas maintenance policy form

伺服电机	
保修期保修服务	
≤1个月	大象机器人公司免费提供新的伺服电机，并承担费用。
1—3个月	大象机器人免费提供新的伺服电机，运费由海关承担。
≥3个月	客户需要自己购买。
电气零件（M5五金件）	
≤3个月	客户需要在拆卸后将其退回，大象机器人将免费送回新的，并承担运费出境和返家。
3—6个月	客户需要在拆卸后将其退回并承担运费到家后，大象机器人将免费寄送新的。
≥6个月	客户需要自己购买。
结构零件，包括壳牌零件	
<1年	大象机器人免费提供一次新部件，运费由海关承担。
≥1年	客户需要自己购买。

servo motor	
Warranty Period	Warranty Services
≤1 months	Elephant Robotics offers a free new servo motor and bear the freigh.
1-3 months	Elephant Robotics offers a free new servo motor, customs shall bear the freight.
≥3 months	Customers need to buy it themselves.
Electrical Parts (M5 Hardware)	
≤3 months	Customers need to send it back after disassembly, Elephant Robotics shall send a new one for free and bear the freight out and home.
3-6 months	Customers need to send it back after disassembly and bear the freight out and home, Elephant Robotics shall send a new one for free.
≥6 months	Customers need to buy it themselves.
Structure Parts, including Shell Parts	
≤1 year	Elephant Robotics offers free new components once, customs shall bear the freight.
≥1 year	Customers need to buy it themselves.

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: DH table How to distinguish between standard and improved

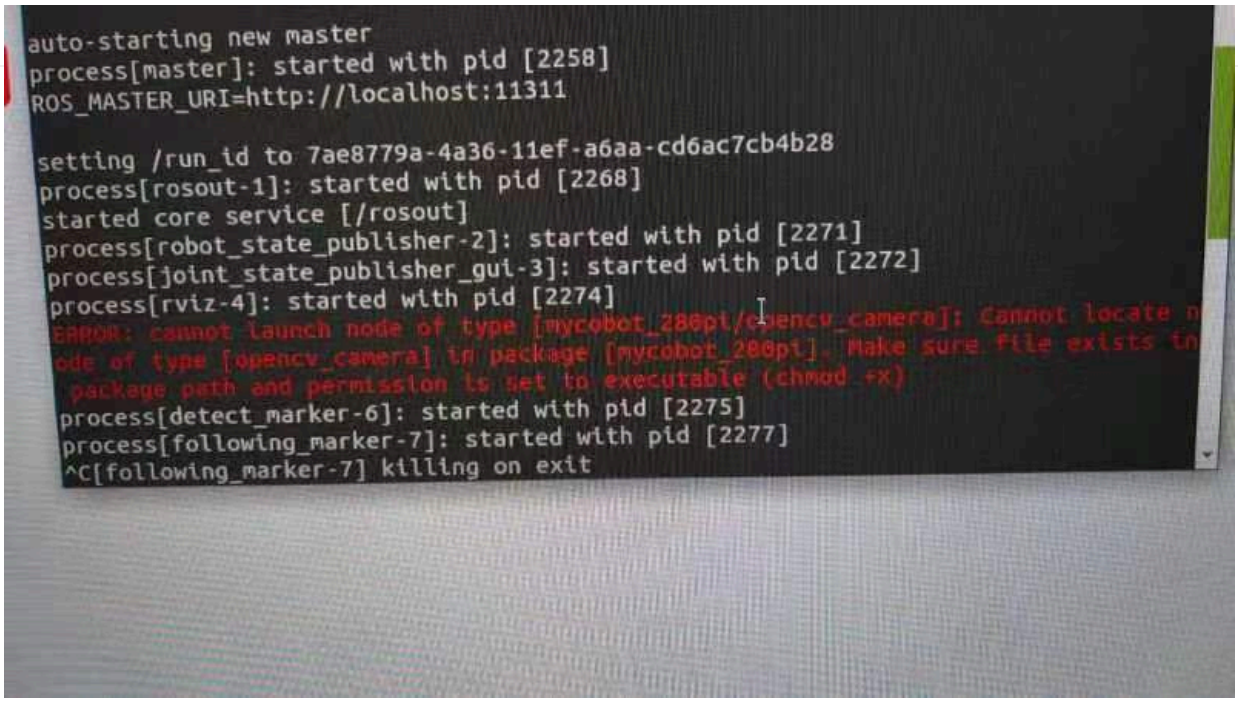
sdh, std, standard mdh, modify, improved We provide the standard DH table. Customers can convert it by themselves if necessary. It is just two different description methods.

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: What should I do if the error message "opencv_camera is missing" appears?



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



Change to using mycobot_280 instead of pi itself, because the file is occupied by m5 and both sides cannot occupy it at the same time, otherwise the subsequent compilation will 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 import MyCobot320
mc = MyCobot320("COM8",115200, debug = True)
while 1:
mc.get_angles()
```

```
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 issued acquisition instruction, read is the returned message, and the left side is the time. Here it shows 518write +611read, indicating that the read of get_angles is completed in about 100ms, and the frequency is 10hz.

Q: The indicator light of the adapter is not on

A: It is possible that the adapter has been 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 for 15 minutes and then 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

代码消息 Python

```python
from pymycobot import MyCobot
import time
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

m = MyCobot(port)
id = [1,2, 3,4,5,6]
j = 1
time.sleep(2)

for i in range (len(id)):
    print(id[i], end=": ")
    for j in range (70):
        print (m.get_servo_data(id[i], j), end=" ")
    print("")

```

Atom abnormal LED lights up red

320: Next cutecom, serial assistant sends fefe 02 19 fa

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 appear, and the gripper needs to be stationary.

Q: What are 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 end effector in Cartesian space are known. write_coords(), send_coords()

First installation and use



- **Thank you for choosing our product**

Before we begin, we would like to sincerely thank you for choosing our product. We are committed to providing you with an excellent user experience.

- **First time use and problem handling**

This chapter will introduce in detail the initial use of the product after receiving it, and provide relevant information for solving problems to ensure that you have no worries during use.

- **Jump to each section**
  - 4.1 Product Standard List
  - 4.2 Product Unboxing Guide
  - 4.3 Power-on Test Guide

← Previous Chapter | Next Chapter →

Product Standard List

1 Product List Picture



Each product is numbered and detailed to ensure you can accurately refer to your listing.

2 Product Standard List Comparison Table

Serial number	Product
1	MyCobot 320 M5 Robotic Arm*1
2	Product Brochure*1
3	Product kit, including USB-Type C*1, jumper, M6*35, cup head hexagon, fully threaded, stainless steel screws
4	24V, 9.2A supporting power supply*1
5	Flat base*1
6	Clamp holder*1
7	E-Stop Key*1

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

← Previous Page | Next Page →

Product unboxing guide

1 Product unboxing graphic guide

Why you need to follow the steps to remove the product

In this section we strongly recommend following the specified steps to remove the product. Not only does this help ensure that the product is not damaged during shipping, it also minimizes the risk of unexpected failure. Please read the following graphic guide carefully to ensure that your product is safe during the unboxing process.

- **1** Check whether the box is damaged. If there is any damage, please contact the logistics company and the supplier in your region in time.
- **2** Open the box and take out the product brochure, sponge packaging cover, myCobot robotic arm, supporting power supply, emergency stop switch, flat base, and accessory package.
- **3** Make sure each step is completed before proceeding to the next to prevent unnecessary damage or omissions.

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

2 Product unboxing video guide



← Previous Page | Next Page →

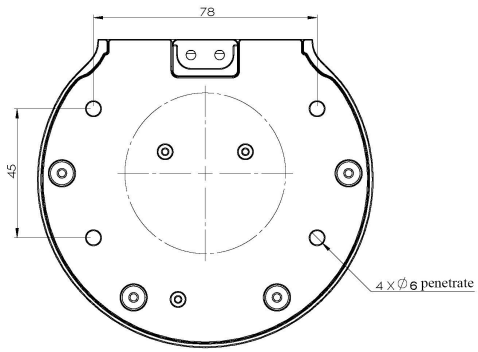
Boot detection guide

1 Structural installation and fixation

myCobot weights 3.3 kg. Due to the fact that the center of gravity will change along with the movement of the robot during utilization, robot is required to be fixed on a solid base at the beginning. A fixed base, or mobile base are both acceptable.

Base Interface Size

- The base fixing holes act as the interface between robot and other bases or planes. The specific hole size is shown in the figure below. There are 4 countersunk holes with a diameter of 4.5 mm, which can be fixed with M6 bolts.



- The end is mounted with flange and is compatible with both LEGO component holes and screw threaded holes. Please make sure that there are corresponding threaded holes on the fixed base before installing.

Before the installation, please confirm:

- The environmental condition meets the requirements listed in Section 2.2.1.3.1 above.
- The installation position is no smaller than the working range of the robot, and there is enough space for installation, use, maintenance and repair.
- Put the base in a suitable position.
- Installation-related tools are prepared, such as screws, wrenches, etc.
- After confirming the above, please move the robot to the base installation table, adjust the robot position, and align the fixing holes of the robot base with the holes on the base mounting table. After aligning the holes, align the screws with the holes and tighten them.

> **Notice:** When adjusting the position of the robot on the base installation table, do not pushing or pulling the robot directly on the base installation table to avoid scratches. When manually moving the robot, do not applying external force to the fragile parts of the robot body, so as to avoid unnecessary damage to the robot.

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

2 External cable connection

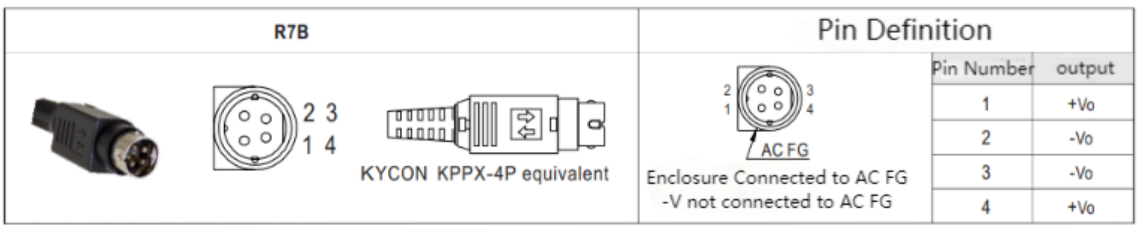
Before operation, confirm that you have read **Chapter Safety Instructions** to ensure safe operation. At the same time, connect the power adapter with the robotic arm, and fix the base of the robotic arm on the table.

myCobot must be powered on with an external power supply to provide sufficient power:

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

  ■ Dc输出插头

  ◎ 标准插头: R7B



Note that you cannot just use the TypeC plugged into the M5Stack-basic for power. Use an officially adapted power supply to avoid damage to the robotic arm.

The use case diagram is shown in the following figure: (Please carefully align it with the use case diagram for connection)

Step 1:


24V
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
GND
STOP
Elephant Robotics
产品/Product
型号/Model
自由度/DoF
6
电源/Charger
211-200107
RoHS

Step 2:


Correct example
24V
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
GND
STOP
Elephant Robotics
产品/Product
型号/Model
自由度/DoF
电源/Charger
211-200107
FCC ID:2AN3WM5Stack-CORE
FCC ID:2AN3WATOM
深圳市大象机器人科技有限公司
www.elephantrobotics.com

Step 3:


24V
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
GND
STOP
Elephant Robotics
产品/Product
大象协作机械臂
型号/Model
myCobot 320-M5
自由度/DoF
6
电源/Charger
DC 24V 120W
211-200107
211-171004
FCC ID:2AN3WM5Stack-CORE
FCC ID:2AN3WATOM
深圳市大象机器人科技有限公司
www.elephantrobotics.com

> ① When using the robot arm, turn the emergency stop knob clockwise to keep the emergency stop switch released;
>
> ② If the movement of the arm is abnormal, please press the emergency stop switch in time, and hold the arm steady to prevent it from falling. ③ When resuming the use of the arm after the emergency situation has ended, turn the emergency stop knob clockwise to release the emergency stop switch and re-power the arm.

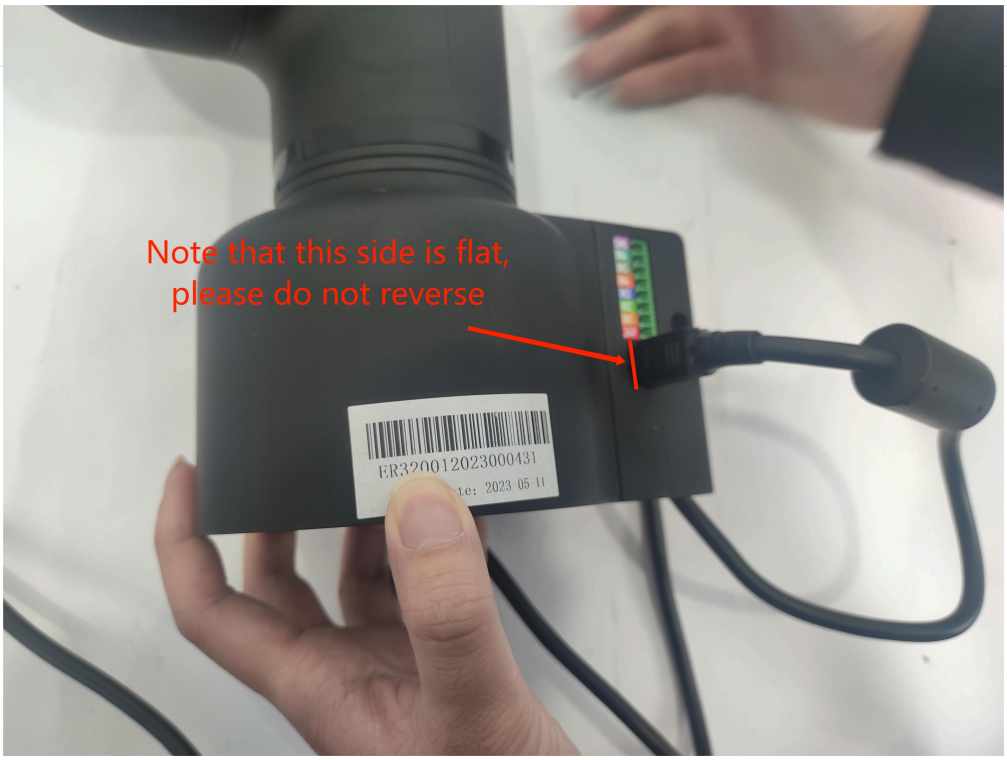
Step 4:



Step 5:


GND
IN6
IN5
IN4
IN3
IN2
IN1
24V
Correct example
ER320012023000431
Date: 2023-05-11

Step 6:



3 Power on status display

Make sure that the power adapter and emergency stop switch are connected, press the power switch **Start button (round)**, then the **LED light panel of ATOM at the end of the robot arm will light**, and the **LED light panel at the bottom of the robot arm will light up**. **BASIC interface** will display **Atom communication status**.



4 Basic function detection

When performing this operation, please refer to section 5.1-4 Robot Information for guidance. Prior to proceeding, ensure that you have followed the electrical connection instructions mentioned above and confirm that your device is securely installed. Failure to properly connect the cables or secure the device may result in accidents. Thank you for your cooperation.

← Previous Page | Next Chapter →

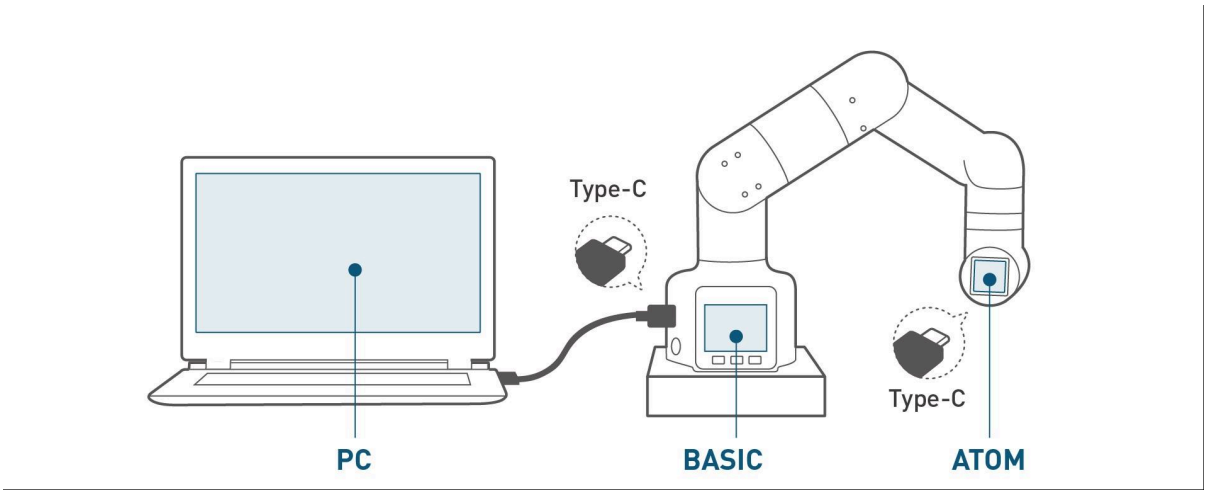
Basic Application



This chapter primarily provides explanations for the basic functionality usage of the product and the use of basic software. This chapter is essential and should be read carefully. Please ensure a correct understanding of the described operations before proceeding with actual robot applications.

For the myCobot series of products, we have introduced software specifically designed for robot applications and maintenance, which is available for our users. Among them, miniRoboflow, myBlockly, and myStudio are essential tools for users working with the robot. This chapter will provide detailed descriptions of the usage of these three software applications.

Function Description



The myCobot 320 robot consists of three controller parts: Basic (a small screen with buttons at the base), Pico (a chip inside the flat Type-C interface on the base), and Atom (the LED light board at the robot's end-effector).

The Basic controller is responsible for handling external interactions, converting data into information that the robot can recognize. So you can use USB Line Connect computer By Basic Type-C Port. (On the left side of the screen.)

The Pico controller manages robot motion algorithms, joint control, and bottom IO control. (No connection is required for normal use.)

The Atom controller is in charge of processing data information from the tool interface and IO control. Together, these three components collaborate to ensure the proper functioning of the robot. (No connection is required for normal use.)

Software Description:

- miniRoboflow

miniRoboflow is a robot control software developed by Elephant Robotics specifically for embedded devices.

This software is developed to simplify user interaction with the robot and runs on the Basic controller located at the bottom. Users can update the corresponding software version through the USB interface.(On the left side of the screen.)

This software provides features such as **path dragging teaching**, joint zero point calibration, **upper computer information processing**, and robot status querying. It is a compact yet feature-rich robot control software.

- myBlockly

myBlockly is a completely visual modular programming software that is a graphical programming language.

myBlockly has a clean interface and comprehensive programming features. Its visual programming approach is suitable for users who are new to robot products.

- myStudio

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

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

For detailed information about the software, please refer to the description within this chapter.

← Previous Chapter | Next Chapter →

Factory firmware introduction

The myCobot 320 M5 version comes with the built-in miniRoboflow application software, developed independently by Elephant Robotics. With simple button interactions, you can achieve functions such as drag teaching, communication forwarding, status checking, and zero-point calibration.

The built-in software has a straightforward user interface and is easy to use. With communication forwarding capability, you can control the robot using various programming languages and development environments, making robot control easily accessible.

miniRoboFlow

miniRoboFlow has four main functions:

- **Maincontrol**
  - Robot drag teaching: The operator can drag robot joints directly to make them do ideal postures, and then save the actions in the robot through **button operation**. The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.
- **Calibration**
  - Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.
- **Transponder**
  - Communication timeliness is vital to the micro-controller robot arm.For such arm, we often send control instructions to **M5Stack-basic** at the bottom. Through communication forwarding, the end effector analyzes the instructions and then implements target actions. At present, **myCobot280** has three methods of communication: **serial port, Bluetooth, and WIFI**.
- **Information**
  - Link test is a detection function that uses the motor in the robot arm and the connection state of **Atom**. The function allows the user to remove equipment faults easily. During the link test, the connection state of the equipment for the robot arm, including the **connection of the servo** and the **communication state of Atom** can be seen. In micro-controller devices, the versions of their current firmwares are shown on M5Stack-basic.

If you have already read all the content in this chapter, please proceed to the next chapter.

Next Section →

Drag teaching



Robot drag teaching is a process during which the operator can drag the joints of the robot directly to make them do ideal postures, and then make records corresponding thereto.

The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.

Different types of equipment have different operating methods. They have the approximate steps below:

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**. Choose the **Maincontrol** function (It is unnecessary to burn **M5Stack-basic** for micro-CPU devices).
- Press the recording button/keyboard key
- select a storage path (micro-CPU devices haven't this step)
- directly drag the joints of the robot arm to move them to your desired positions to complete a set of movements
- press the designated button/keyboard key for saving
- press the play button/keyboard key/keyboard key
- select a corresponding storage path, and then the robot arm starts to move
- and finally press the exit button/keyboard key to exit this function

In this chapter, we will teach you how to get started easily and experience the fun of cobot drag teaching.

Drag-and-Record Demonstration

1 Applicable Robotic Arms

- myCobot 320 M5

2 Steps to Operate the Arms

Step1:

- Please confirm that you are aware of the product safety instructions and ensure that the device has been connected with cables as described in Chapter 4.
- Turn on the power switch and ensure that the emergency stop switch is connected and not pressed.
- Confirm that the Atom LED light board at the robot's end-effector is illuminated and that the robot's joints have torque output, making the joints unable to rotate.

Step 2: Press **OK** Select **Maincontrol** to Maincontrol Main Page.



Step 3: Press **Record** in Maincontrol Main Page.



Step 4: Select path for storage, and press **Ram** or **Flash**.



Step 5: Move the joints of the robot arm to specific positions. After that, press any one of the three buttons so as to stop recording and save the motion.



Step 6: Press **Exit** to Main Page.



Step 6: Press **Play** to let the robot perform the motion recorded just now.


Playing for Ram/Flash?
Ram
Flash
Back



> **Notice:**
>
> **Pause**: pause the movement
>
> **Stop**: stop the movement
>
> **Play**: resume the movement

← Previous Page | Next Page →

Robot arm Calibration

	**This operation has been performed before the default robot is delivered. You do not need to repeat the operation. Incorrect use of this function may cause damage to the robot. If your robot works without abnormal conditions, please do not use it. Thank you for your cooperation.**

> **Before performing a zero calibration, allow the machine to return to zero and check for any deviation in the zero attitude. If there is, then perform zero calibration; if not, zero calibration is not required.**
>
> You can use pythonAPI to make the joints reach the zero position: `send_angles(degrees, speed)`, where degrees is a floating-point list with [0,0,0,0,0,0] representing the zero position of the six joints, and speed is the speed at which the joints are moving, which takes the values of 0 to 100.



Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.

Different types of equipment have different operating methods. They have the approximate steps below:

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**.
- Choose the **Calibration** function (It is unnecessary to burn **Basic** for micro-CPU devices)
- move all joints of the robot arm to their zero positions (align them with the scale line of zero positions)
- Press the calibration button and get started with robot arm calibration
- press the test button to test the zero positions
- press the exit button to exit this function

In this chapter, we will teach you how to calibrate the robot arm and to test and verify the joints.

Calibration Demonstration

1 Applicable Robotic Arms

- myCobot 320 M5

2 Steps to Operate the Arm

Step1:

- Please confirm that you are aware of the product safety instructions and ensure that the device has been connected with cables as described in Chapter 4.
- Turn on the power switch and ensure that the emergency stop switch is connected and not pressed.
- Confirm that the Atom LED light board at the robot's end-effector is illuminated and that the robot's joints have torque output, making the joints unable to rotate.

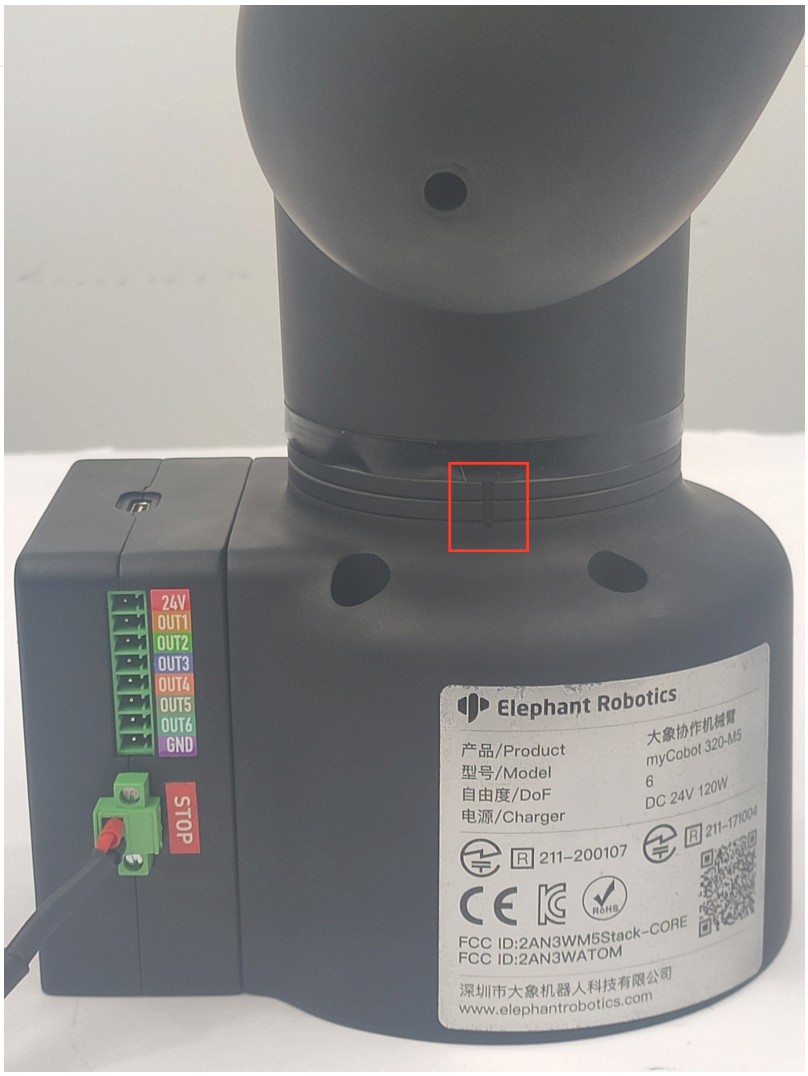
Step 2: Press **OK** Select **Calibration**.



Step 2: Press **A** to start robot arm calibration.



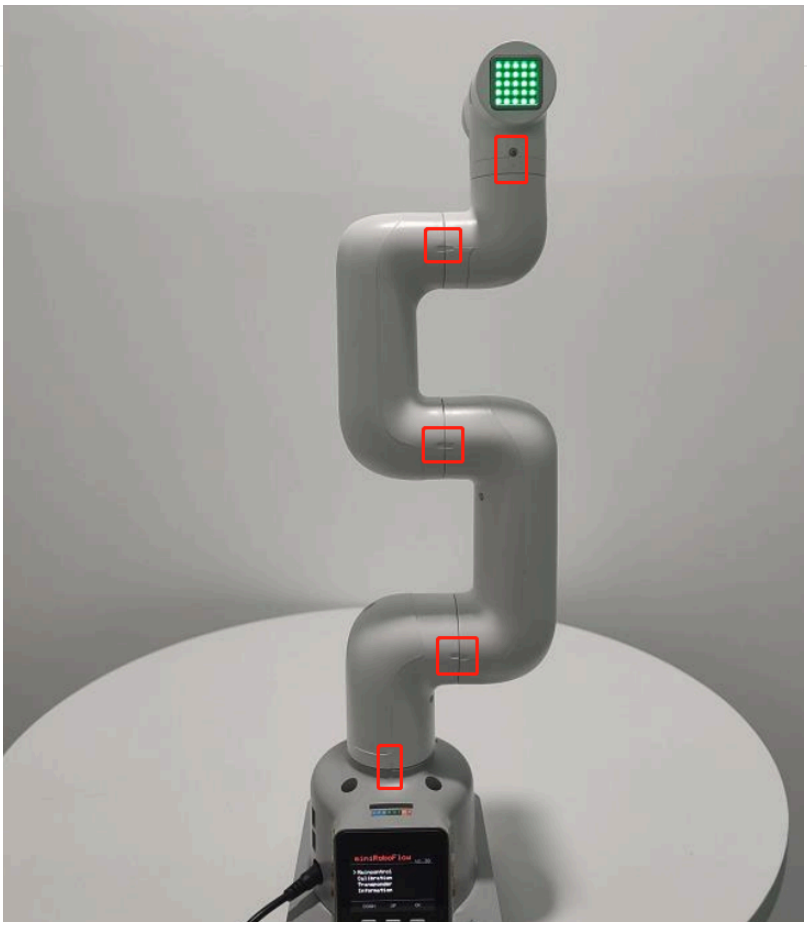
Step 3: Align each joint with zero-position line in sequence.



Step 4: After finishing calibrating all joints, a signal **Already Calibrate all !!** emerges on the screen.



> **Calibration results**



Step 5: Press **EXIT** to stop calibration and then press **B** to test the zero positions of all joints.

> The joints will rotate positively and negatively by 15-30 degrees and return to the zero position.


myCobot
Basic Calibration
Press A - Calibrate Servo
Press B - Test Servos
Press C - Exit



Step 6: Press **C** to exit this function. Calibration completes.



← Previous Page | Next Page →

Communication



This chapter describes the use of the built-in software's extended development capabilities. This feature allows you to expand the device's connectivity options and support different development environments and methods.

At present, communication forwarding supports three connection methods: USB, Wi-Fi, and Bluetooth. They are applied to different functional development scenarios.

- USB and Wi-Fi are used to connect to other master control devices, and through the robot communication protocol, you can control the robot to execute corresponding commands.
- Bluetooth is used to connect to mobile control software, enabling quick mobile control from your phone.

1 Applicable Robotic Arm

- myCobot 320 M5

2 USB Port Control

Step1:

- Please confirm that you are aware of the product safety instructions and ensure that the device has been connected with cables as described in Chapter 4.
- Turn on the power switch and ensure that the emergency stop switch is connected and not pressed.
- Confirm that the Atom LED light board at the robot's end-effector is illuminated and that the robot's joints have torque output, making the joints unable to rotate.

Step 2: Press **OK** Select **Transponder** to Transponder Main Page.



Step 3: Choose the function you want to use, with USB connection as an example below.



Step 3: Check Atom State



`atom: ok` means that Atom is connected normally.

> If all your output statuses are 'no,' please ensure that your device's firmware is correct. Check the power connections and emergency stop switch connections again, and confirm that the emergency stop switch has not been pressed. After confirming, restart the device and repeat the current step.

Step 4: Choose the appropriate development method and try communicating with the robot. For more details, you can refer to the SDK Development chapter.

Step 5: If you want to modify the connection method Press **C** to exit.



← Previous Page | Next Page →

Connection detection

This chapter explains how to use the basic software to check your device's status. Please make sure that your device's software version and firmware version are both unchanged (factory default versions) before use. If you have made any modifications to the software or firmware, ensure that your actions comply with the requirements outlined in the myStudio firmware update section.

Connection Detection

1 Applicable Robotic Arms

- myCobot 320 M5

2 Steps to Operate the Arms

Step1:

- Please confirm that you are aware of the product safety instructions and ensure that the device has been connected with cables as described in Chapter 4.
- Turn on the power switch and ensure that the emergency stop switch is connected and not pressed.
- Confirm that the Atom LED light board at the robot's end-effector is illuminated and that the robot's joints have torque output, making the joints unable to rotate.

Step 2: Press **OK** Select **Information** to Information Main Page.

Step 2: Press the **Information**.



Step 2: Press the **Button A**



Step 3: Check Atom & Servo Motor State

`atom: ok` means that Atom is connected normally.

`servo x: ok` means that six motors are connected normally.



> If all your output statuses are 'no,' please ensure that your device's firmware is correct. Check the power connections and emergency stop switch connections again, and confirm that the emergency stop switch has not been pressed. After confirming, restart the device and repeat the current step.

Notice: Press **B** for the version information.



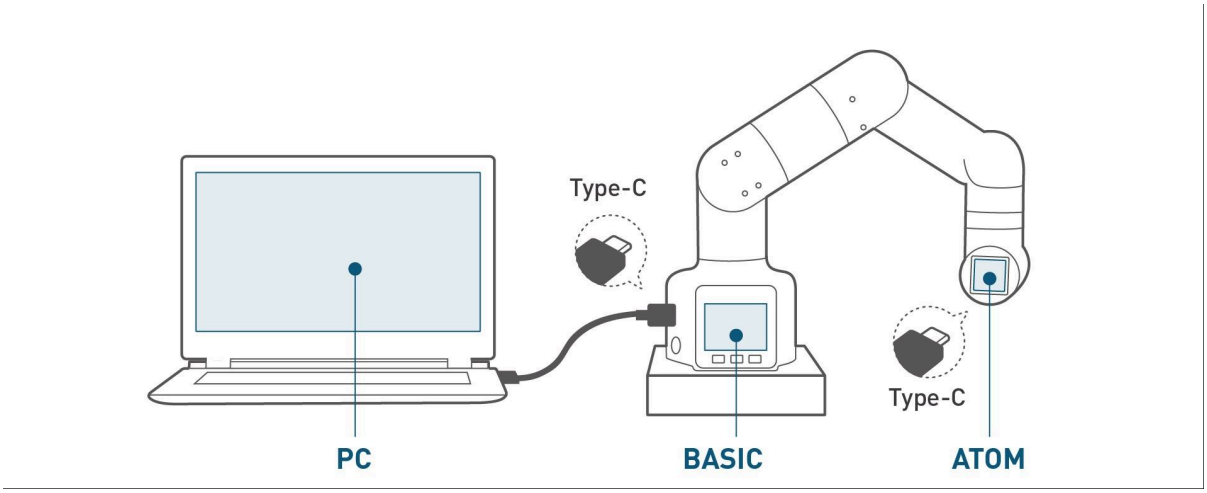
> You can refer to the recommended firmware version in myStudio or the firmware documentation section.

Step 4: Press **C** to exit this function.



← Previous Page | Next Page →

Firmware Description



The myCobot 320 robot consists of three controller parts: Basic (a small screen with buttons at the base), Pico (a chip inside the flat Type-C interface on the base), and Atom (the LED light board at the robot's end-effector).

The Basic controller is responsible for handling external interactions, converting data into information that the robot can recognize. So you can use USB Line Connect computer By Basic Type-C Port. (On the left side of the screen.)

The Pico controller manages robot motion algorithms, joint control, and bottom IO control. (No connection is required for normal use.)

The Atom controller is in charge of processing data information from the tool interface and IO control. Together, these three components collaborate to ensure the proper functioning of the robot. (No connection is required for normal use.)

Atom(atommain) firmware update instructions

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
V1.1	Robot control: 1. Single/multi-angle/coordinate control 2. Obtain angle/coordinates 3. Clamp, pump control			1. This version Requires use of 2020 devices
V2.8	Basic functions remain unchanged	Function repairs: Fixed motion algorithm error Bus repairs: Fixed known bugs	1. In this version, the basic firmware needs to be burned into miniRobotV1.0	1. This The version needs to be used on 2020 devices
V3.1	Basic functions remain unchanged	Function optimization: Communication module optimization Function repair: Repair coordinate control	1. In this version, the basic firmware needs to be burned into miniRobotV1.0	1. This version needs to be used until the 2020 model Equipment

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
V4.1	Basic functions remain unchanged	New features: Added potential value acquisition Function fixes: 1. Fixed single coordinate control 2. Fixed gripper movement problem />3. Repair jog control 4. Repair free movement	1. In this version, the basic firmware needs to be burned into miniRobotV1.0	1. This version needs to be used in 2020 devices
V4.2	Basic functions remain unchanged	New features: New electric gripper control	1. In this version, the basic firmware needs to be burned into miniRobotV1.0	1. This version needs to be used in 2020 devices
V5.0 (the version number cannot be the same as the old version)	1. Control LED lights 2. Atom io control		1. In this version, basic burns the latest firmware 2. Servo firmware is required Supports synchronous reading and writing 3. Pico burns the latest firmware	1. 2022 devices - so far

miniRobot firmware update instructions

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
v1.0	1. Drag teaching 2. Zero calibration 3. Serial communication 4. Information viewing (joint connection status, firmware version)		Suitable for atommainV4. 1Previous Firmware	This version requires up to 2020 devices
v2.0	1. Drag teaching 2. Zero calibration 3. Communication (serial port, wifi, Bluetooth communication) 4. Information viewing (joint connection status, firmware version)	Function optimization: Optimize drag teaching to support recording of speed during drag New features: Add wifi and Bluetooth communication to communication; Bug fix: At the same time, power on the wifi and uart communication interface before shutting down. It will automatically jump to the interface before shutting down	1. This version of atom needs to be burned to v5.0 2. The servo needs to support synchronous reading and writing 3. Pico needs to be burned to V1.0	2022 device - so far

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
v2.1	1. Drag teaching 2. Zero calibration 3. Communication (serial port, wifi, Bluetooth communication) 4. Information viewing (joint connection status, firmware version)	Repair function: 1. Continuous jog control will cause basic to restart; 2. After unplugging and restarting, uart needs to be entered a second time to be effective; 3. Optimize the use of drag teaching Effect.	1. This version of atom needs to be programmed with v5.0 2. The servo needs to support synchronous reading and writing 3. Pico needs to be programmed with V1.1	2022 device - so far
v2.2	1. Drag teaching 2. Zero calibration 3. Communication (serial port, wifi, Bluetooth communication) 4. Information viewing (joint connection status, firmware version)	Optimized functions: bt is upgraded to ble, supports Bluetooth 5.0, and supports the use of Mycobot APP. Repair functions: Drag teaching: 1. Relax the joints before starting recording to make them more sensitive; 2. Fix the jitter problem after hardware changes.	1. This version of atom needs to be burned with v5.0 2. The servo needs to support synchronous reading and writing 3. Pico needs to be burned with V1.2	2022 device--so far

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
v2.3	1. Drag teaching 2. Zero calibration 3. Communication (serial port, wifi, Bluetooth communication) 4. Information viewing (joint connection status, firmware version)	basic all function, the overall use effect is better. Optimization content: 1. Zero calibration, only need to calibrate once, fully relax when entering (with damping), fully relax during single joint calibration 2. Drag teaching: recording is completed It will be powered on automatically, and the LED light at the end will turn yellow during recording and turn green when recording ends; 3. The transponder will not fall when it exits communication.	1. This version of atom needs to be burned to v5.0 2. The servo needs to support synchronous reading and writing 3. Pico is burned to V1.2 or V1.3 (with damping function)	2022 device - so far

pico firmware update instructions

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
V1.0	Robot control communication speed is within 20ms 1. Single/multi-angle/coordinate control; 2. Obtain angle/coordinates 3. Adaptive/electric gripper, system Pump control...		1. This version of atom needs to be burned with v5.0 2. The servo needs to support synchronous reading and writing 3. This version, the basic firmware needs to be burned with miniRobotV2.0	1 , 2022 devices--so far

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
V1.2	Robot control communication speed is within 20ms 1. Single/multi-angle/coordinate control; 2. Obtain angle/coordinates 3. Adaptive/electric gripper, system Pump control...	New functions: 1. Modify the servo pdi: save the last modified value when power off; 2. Adapt to the new adaptive gripper: add io control compared to before Clamp; Optimization content: 1. Control LED lights, terminal IO control, electric clamp control, optimize control reliability, and it will not take effect after several clicks like before. Modification content: 1. releaseServo, releaseallservos, the damping mode can be canceled when releasing, and the default is damping mode (there is resistance when the joint is relaxed, making it difficult to drag).	1. This version of atom needs to be burned with v5.0 2. The servo needs to support synchronous reading and writing 3. This version, the basic firmware needs to be burned with miniRobotV2.2	1. 2022 device-- So far

Version	Firmware function description (function items)	Version iteration record	Firmware usage instructions	Applicable devices
V1.3	Robot control communication speed is within 20ms 1. Single/multi-angle/coordinate control; 2. Obtain angle/coordinates 3. Adaptive/electric gripper, system Pump control...	Repair content: 1. When turning on the machine, the initial sending point will not return to zero first and will move normally; 2. When stopped during motion, the robotic arm will not fall. Optimization content: Motion control optimization.	Applicable to all	1. 2023 devices--so far

← Previous Page | Next Page →

How to burn firmware

we can use myStudio to burn the firmware

← Previous Page

What is myBlockly?



myBlockly is a completely visual modular programming software that is a graphical programming language.

myBlockly is similar in function/design to MIT's children's programming language Scratch.

When using myBlockly, users can build code logic by dragging modules. The process is like building blocks.

From the user's perspective, myBlockly is a simple and easy-to-use visual tool for generating code. From a 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 it in myBlockly.

The operating systems supported by myBlockly are as follows:

- Windows
- macOS
- Linux arm64

← Previous Section | Next Section →

Preparations before using myBlockly

Before downloading myBlockly, you need to configure the Python environment and use myStudio to burn the firmware. Please refer to the following operation methods for details.

1. Environment configuration. Before using myBlockly, please make sure that the Python environment has been configured on your computer (for details on Python download, installation and environment configuration, please refer to install python)

2. Firmware burning.

   How to use **myStudio** to burn the firmware，please refer to myStudio

← Previous Page | Next Page →

myBlockly download and install

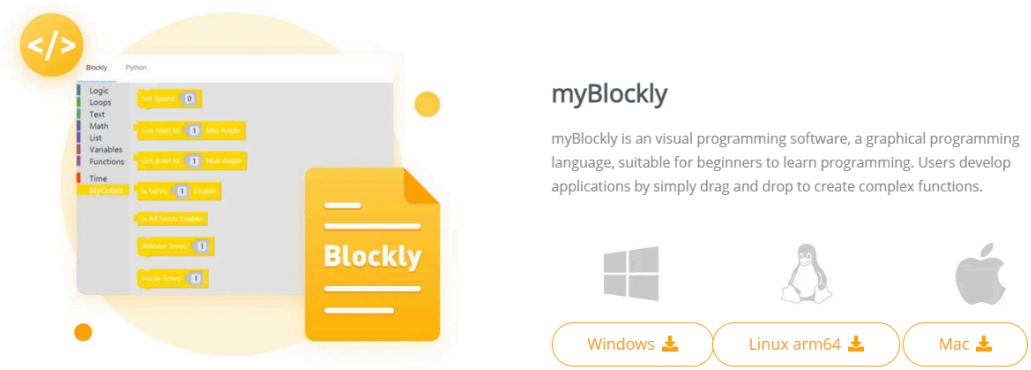
Download

download link **Official website address**

You need to download different installation packages according to your operating system.

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

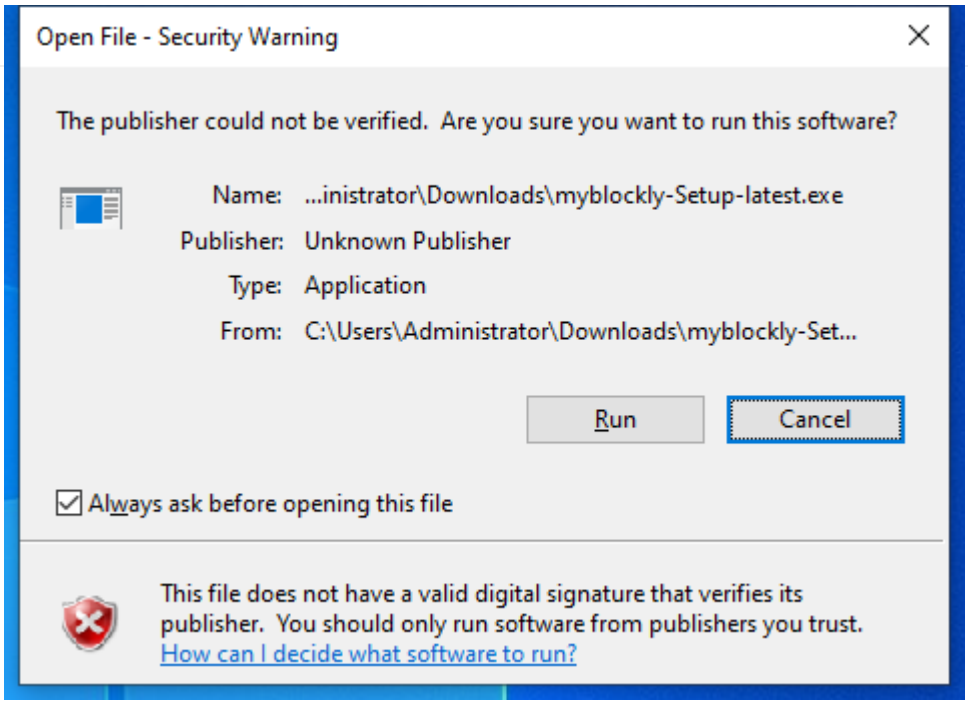
- *.AppImage : Linux system
- *.dmg : Mac system
- *.exe : Window system



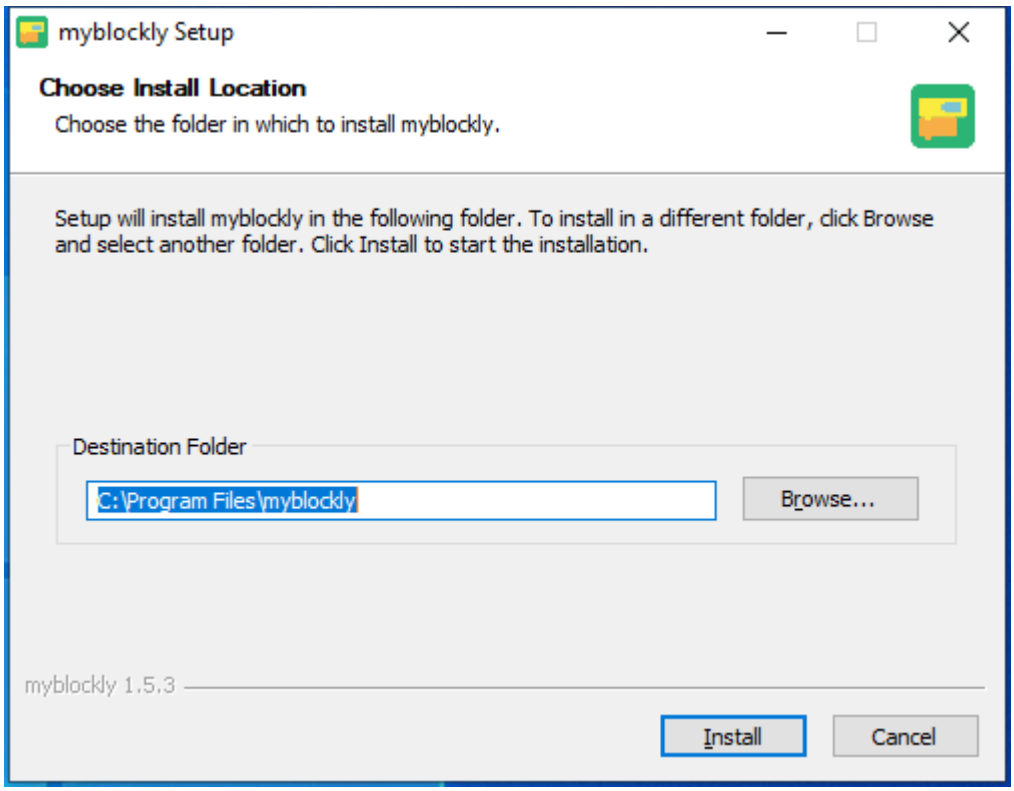
Install

For Windows installation

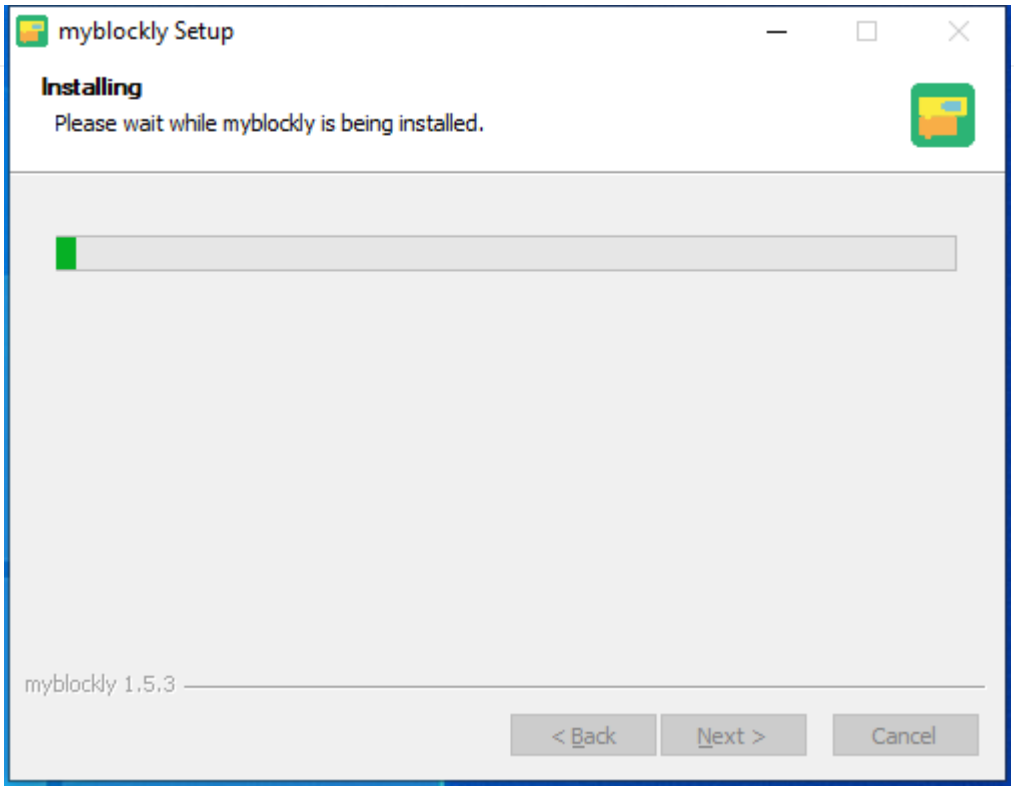
Double-click to open the file named `myblockly-Setup-latest.exe`



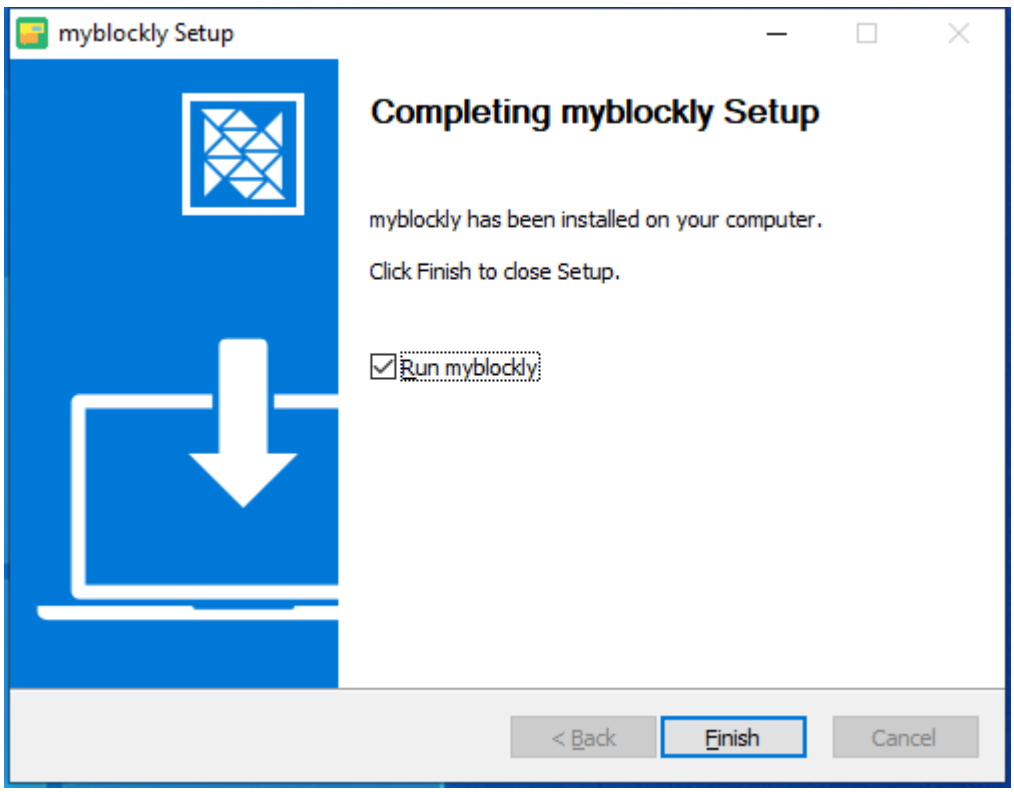
Click to `Run`

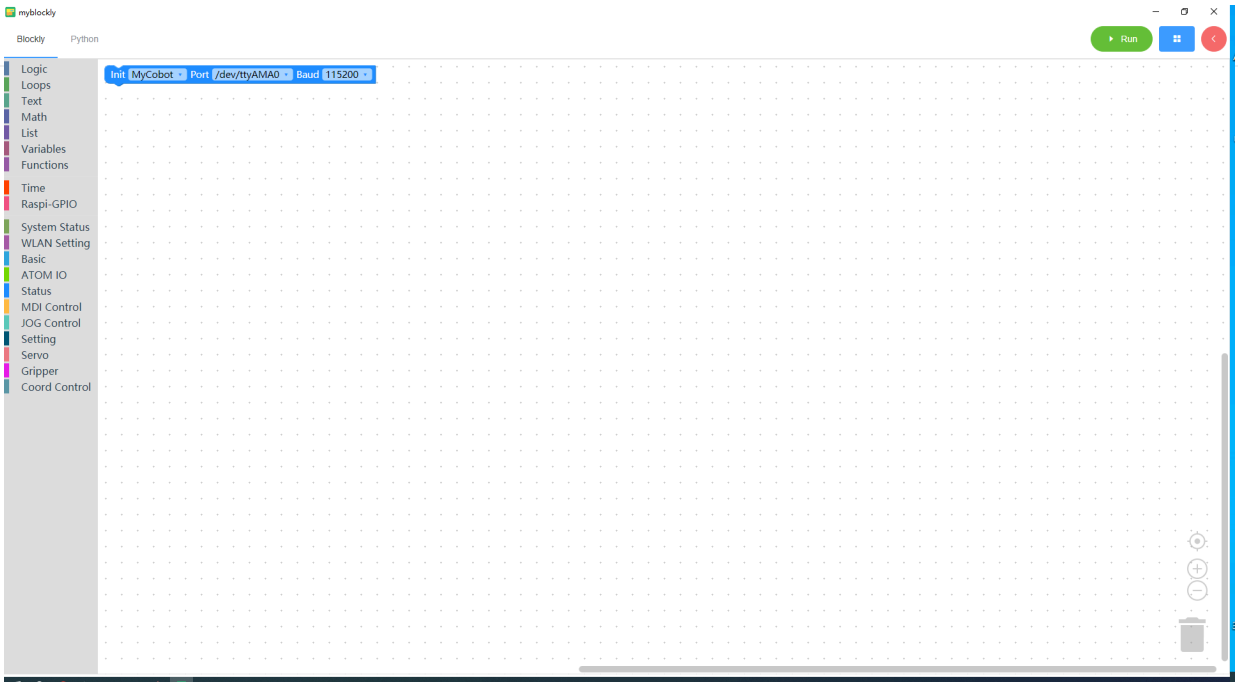


After clicking `Install`, wait for `**myblockly` installation to complete



The installation is complete, click the `Finish` button to open and run myblockly



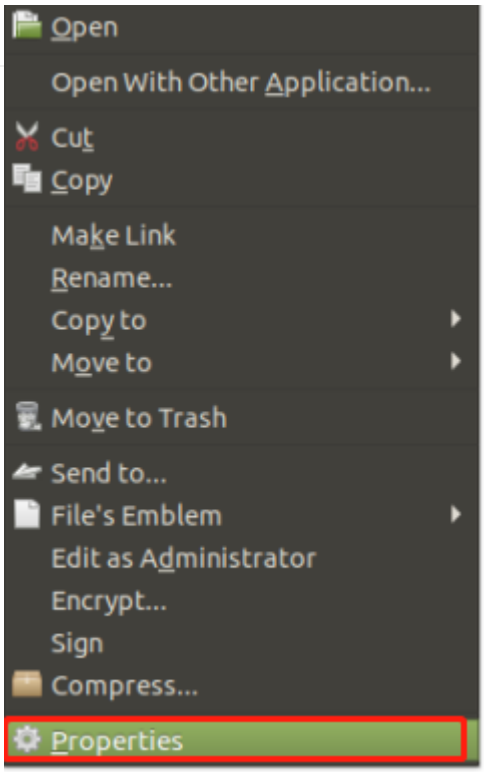


For Linux systems install myblockly

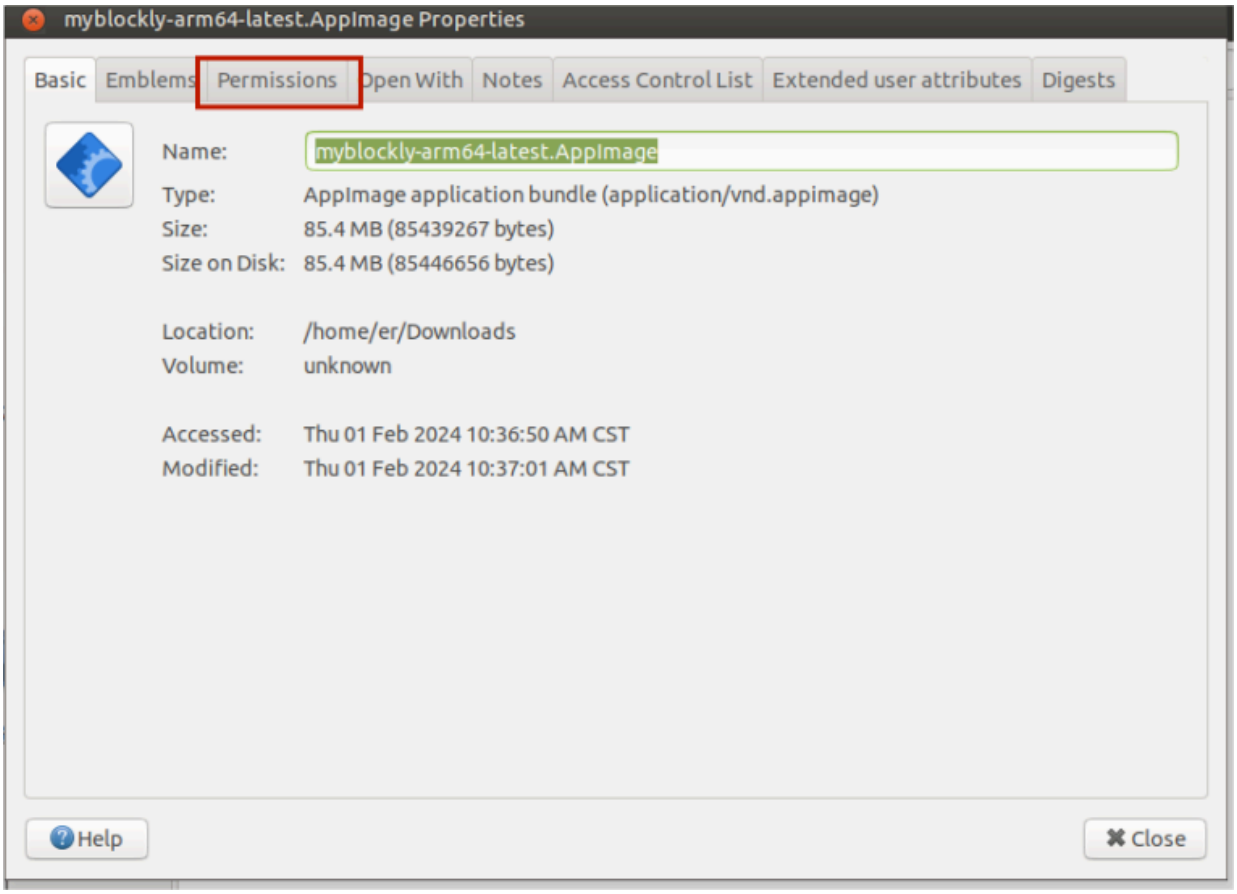
Download the Linux version of `myblockly` from the official website and you will get an installation package as shown below



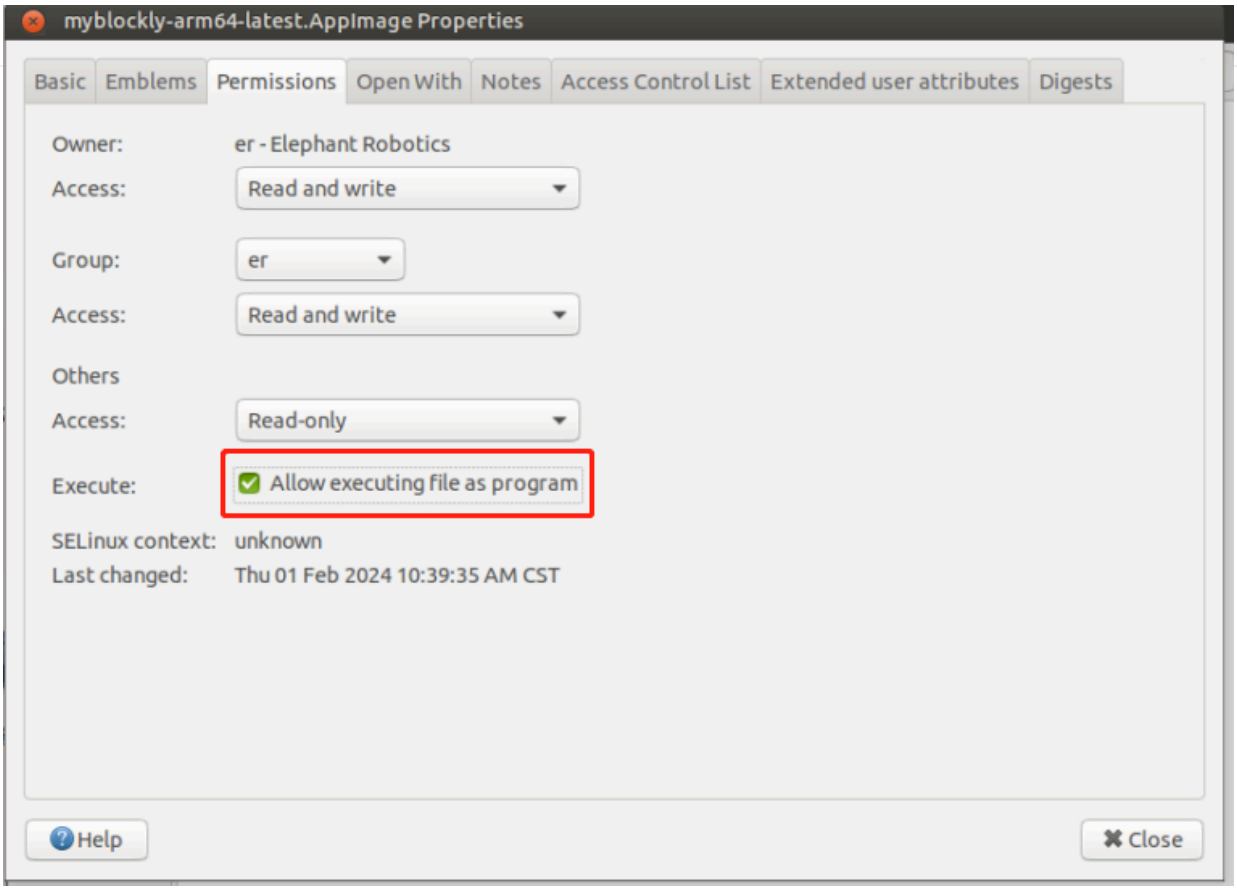
Select `myblockly-arm64-latest.AppImage`, right-click to open it, click `Properties` to open it



Click to enter `Permissions`



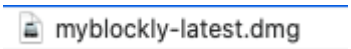
In the `Permissions` page, check `Allow executing file as program`, and then click the `Close` button to close the pop-up window



After closing the pop-up window, double-click the installation package `myblockly-arm64-latest.AppImage` to open myblockly

For MacOS install myblockly

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



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

Uninstall

For Linux uninstall myblockly

Just delete the installation package directly

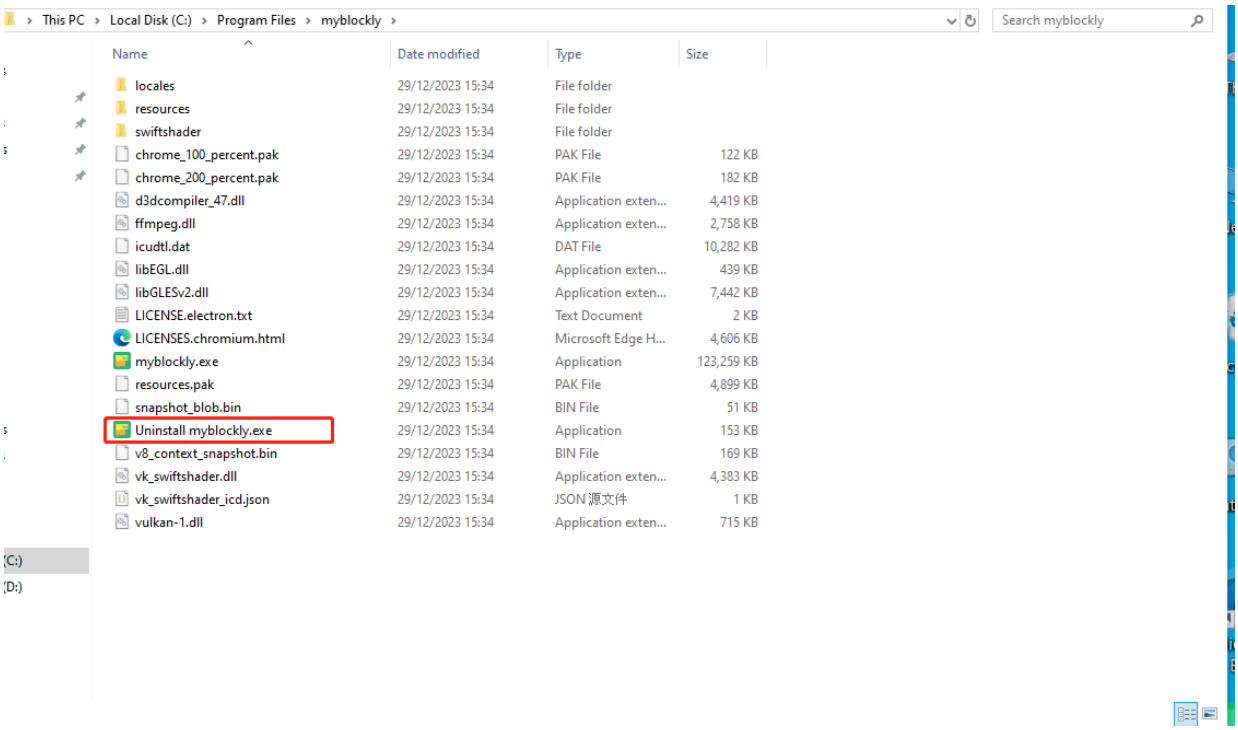
> The default name of the installation package is `myblockly-arm64-latest.AppImage`

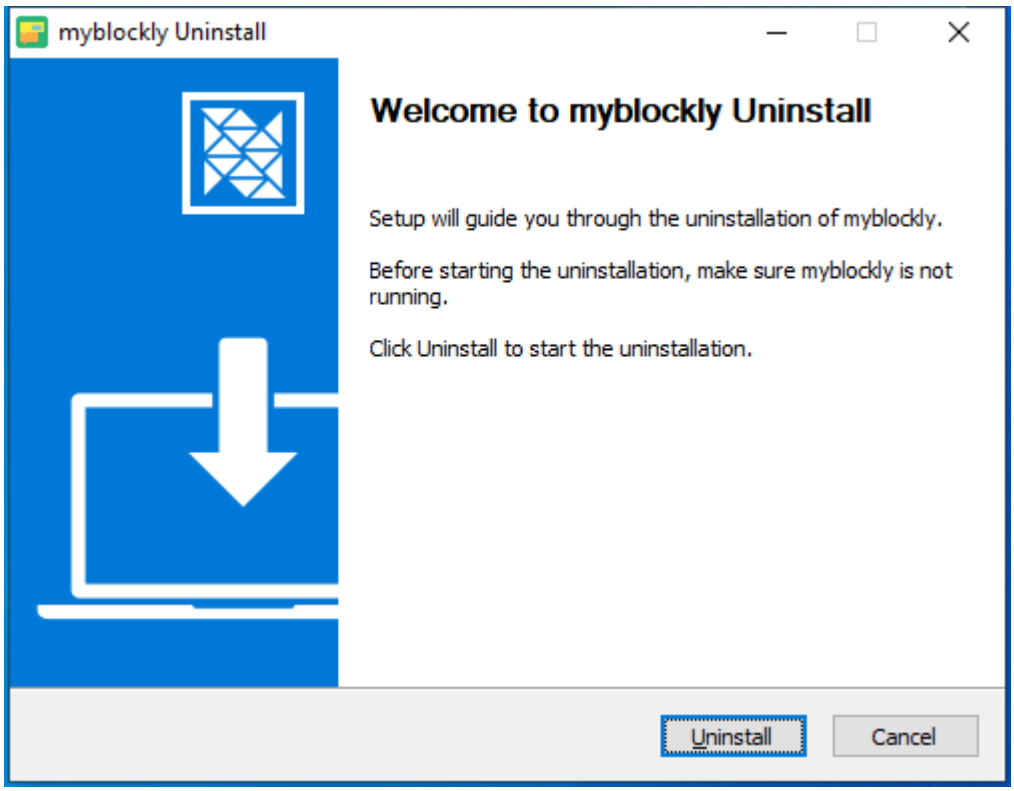
Uninstall myblockly for Mac

Just move myblockly to the Trash in the app

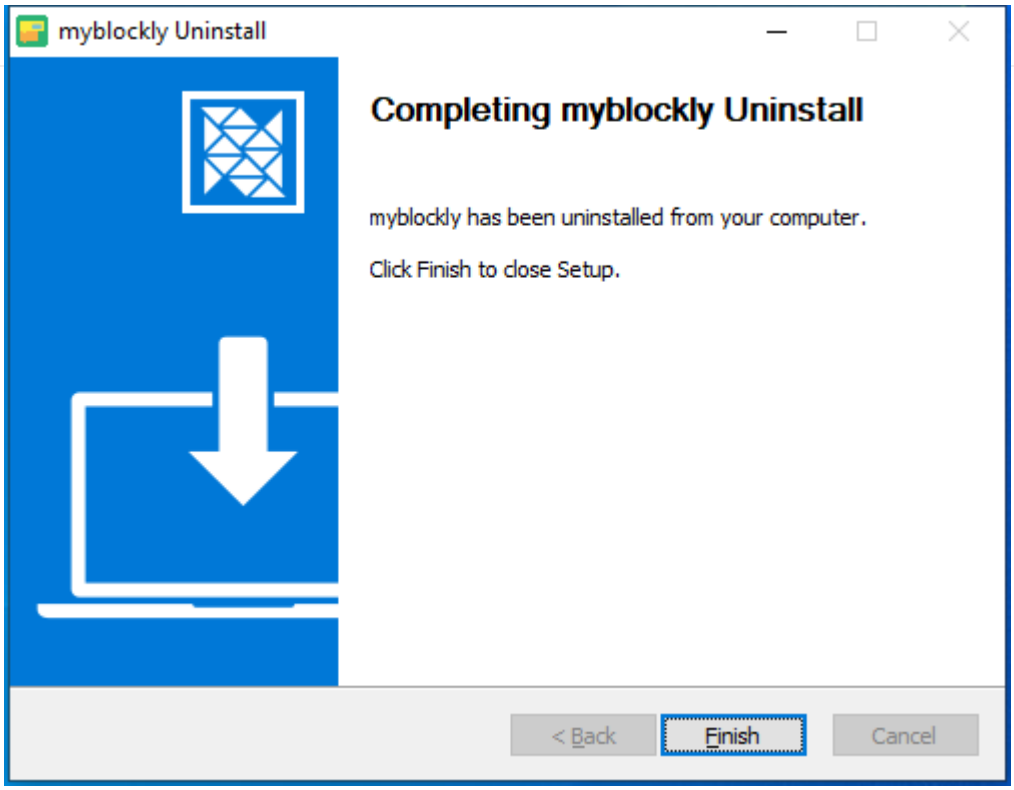
For Windows uninstall myblockly

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





Click `Uninstall`



myblockly has been uninstalled, click `Finish` to exit

Update

If you want to update myblockly, please go to the official website to download the latest version.

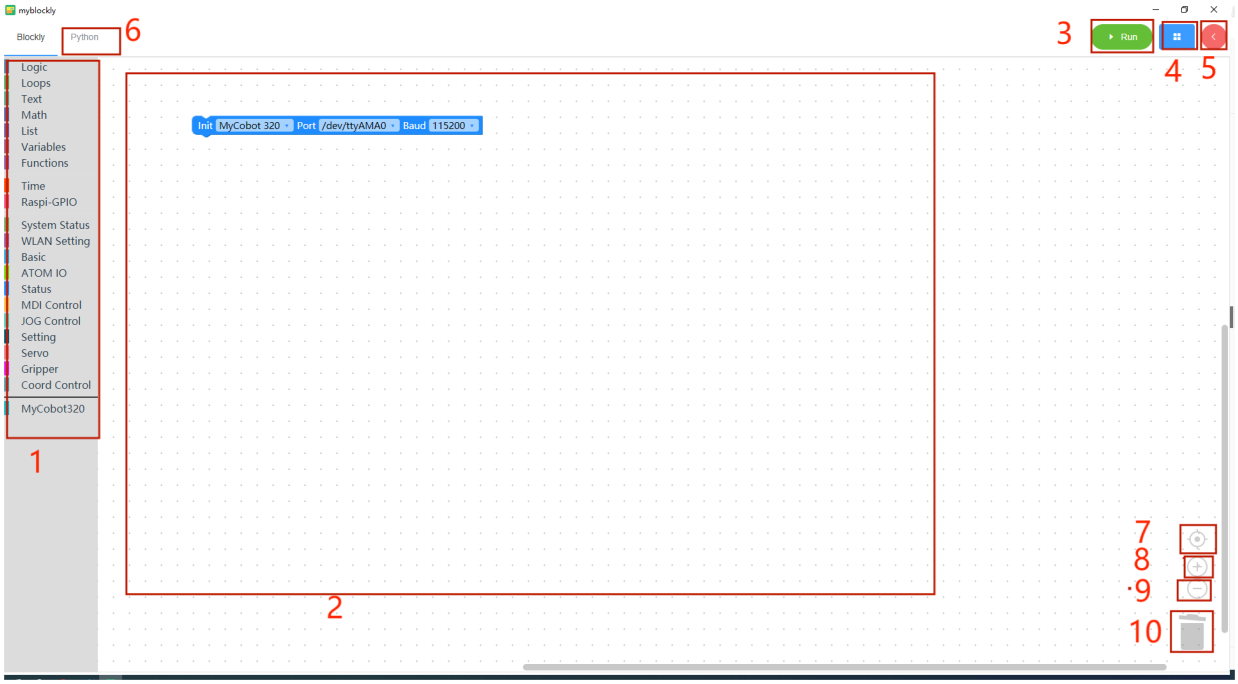
Serial Driver Installation

Before using myStudio to burn, you need to download the serial driver, i.e. CP210X or CP34X driver zip.There are currently two driver chip versions, CP210X (for CP2104 version) / CP34X (for CH9102 version) driver zip. 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 report an error during the installation process, but the installation is actually complete, so just ignore it). Please click the link below to download the driver according to your computer system (please refer to 5.2.2-myStudio driver installation for detailed installation steps).

- Download the **M5Stack-basic** serial port driver at the bottom
  - **CP210X**
    - **Windows10**
    - **MacOS**
    - **Linux**
    - **CP34X**
      - **Windows10**
      - **MacOS**

← Previous Page | Next Page →

myBlockly interface display and use of basic functions

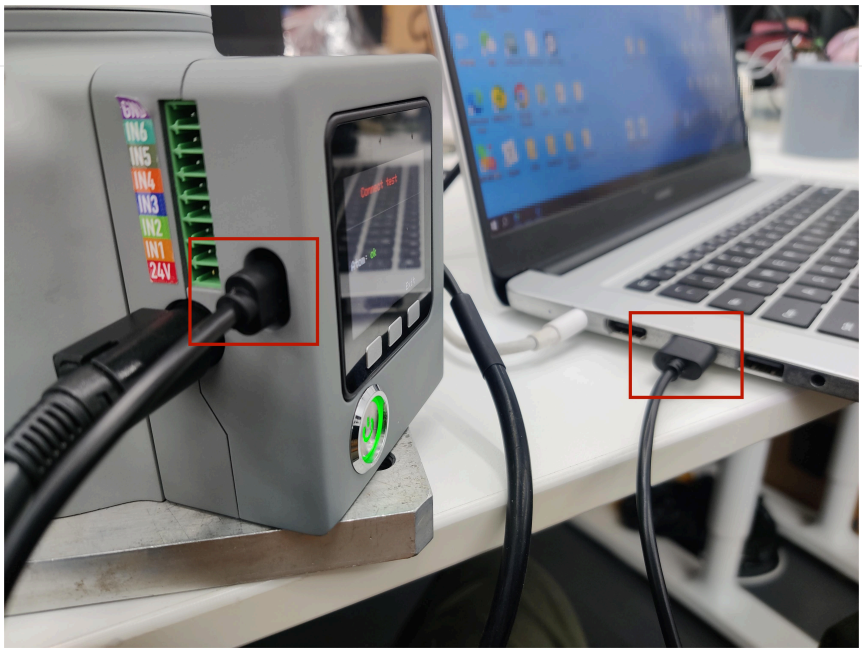


serial number	illustrate
1	Toolbox bar: Click on a specific category to select different building blocks
2	Workspace: You can drag the building blocks of the toolbox to the workspace for use
3	Run button: Run the code in the workspace
4	The following menu will pop up after clicking: - Save: Save the current workspace - Load: Load the saved workspace - Settings: Enter the settings page, where you can set the language and theme
5	After clicking, the control panel will pop up. In the panel, you can quickly control the movement of the robotic arm by clicking "+/-" in the joint control or coordinate control bar. 6 After clicking, you can view the python code generated in the workspace. 7 Center the workspace. 8 Enlarge the workspace. 9 Reduce the workspace.
6	Click to view the python code generated by the workspace
7	Center the workspace
8	Zoom into workspace
9	Reduce work area
10	Trash can: Drag the building blocks in the workspace here to delete the building blocks; click the trash can at the same time to view and restore the deleted building blocks

Program running

Before officially starting programming, be sure to select the corresponding **machine model**, otherwise it will easily cause hardware damage.

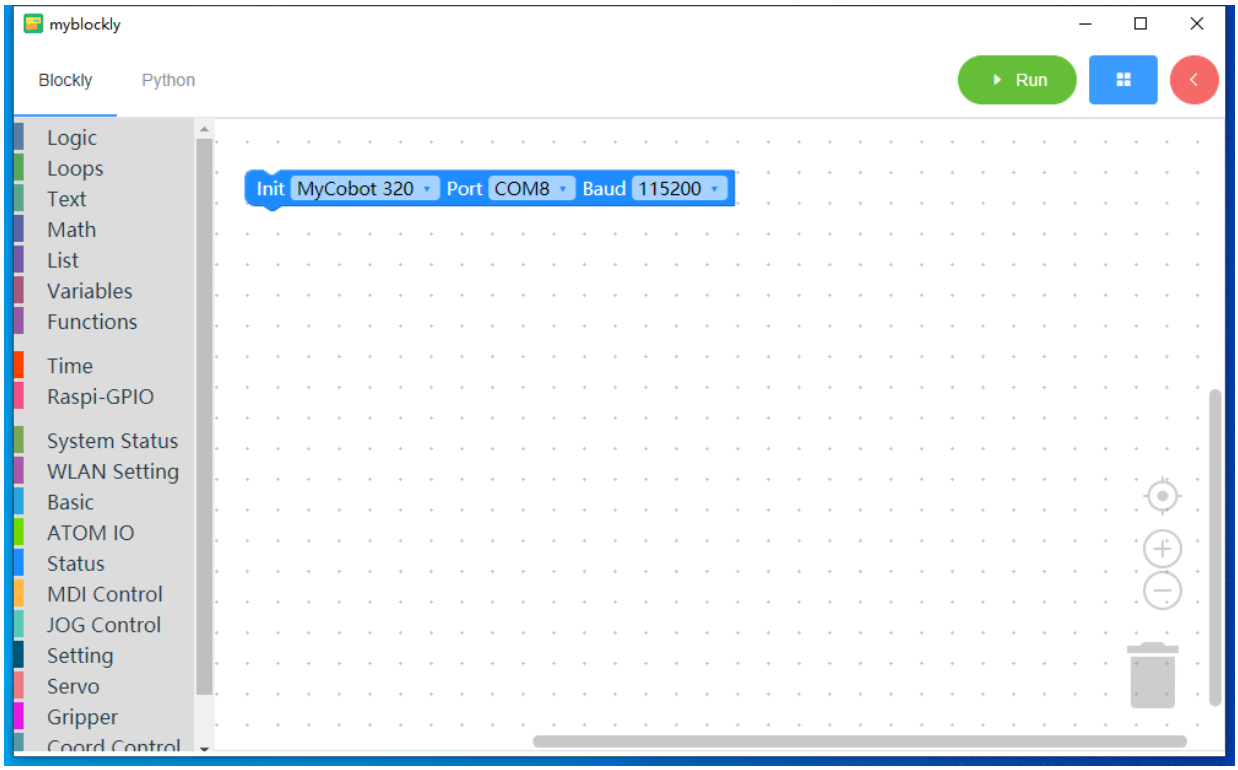
First connect `myCobot 320 M5` to our computer through `USB data cable` ,


IN6
IN5
IN4
IN3
IN2
IN1
24V



And then we select the initialization model as `myCobot 320` , the port connected to my computer is `COM8` , and the baud rate is `115200`

> The com port number is not fixed, you need to choose according to the actual port number of the individual PC connection
>
> How to choose the corresponding port number: by unplugging the usb cable on the side of the M5stack to see if there is any change in the corresponding port number to choose, unplugged in both cases there is a change in the com port is the machine's communication serial port.

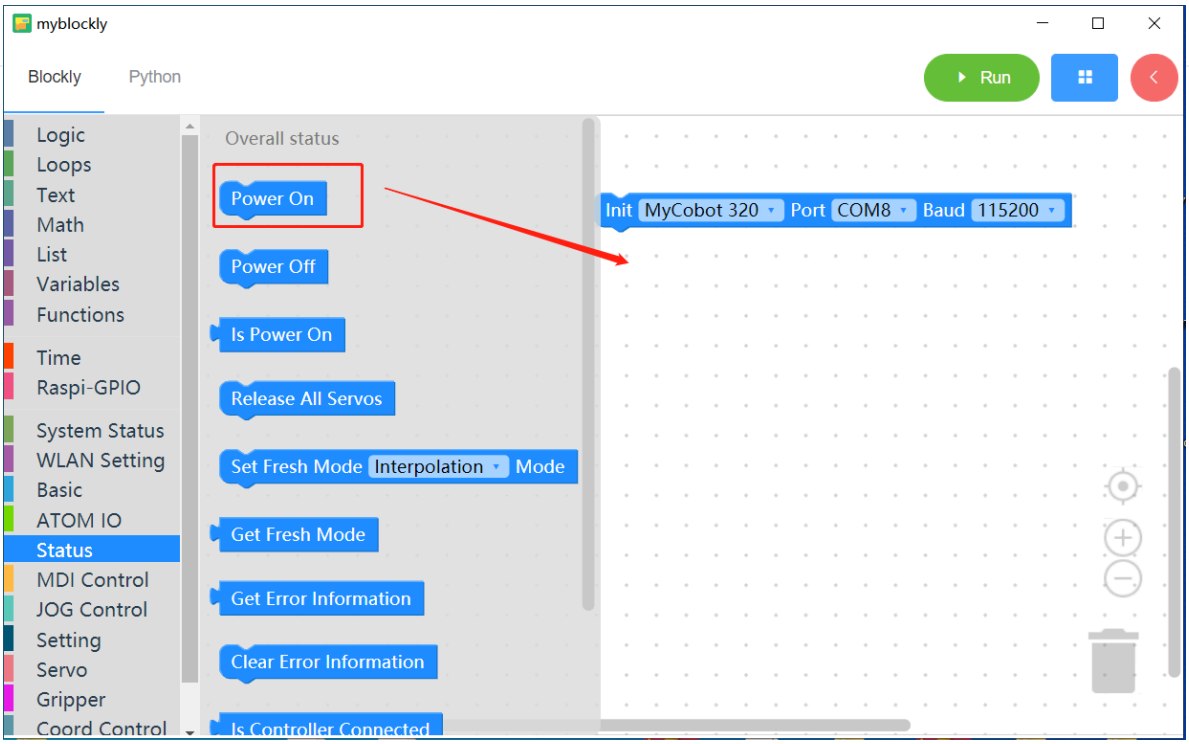


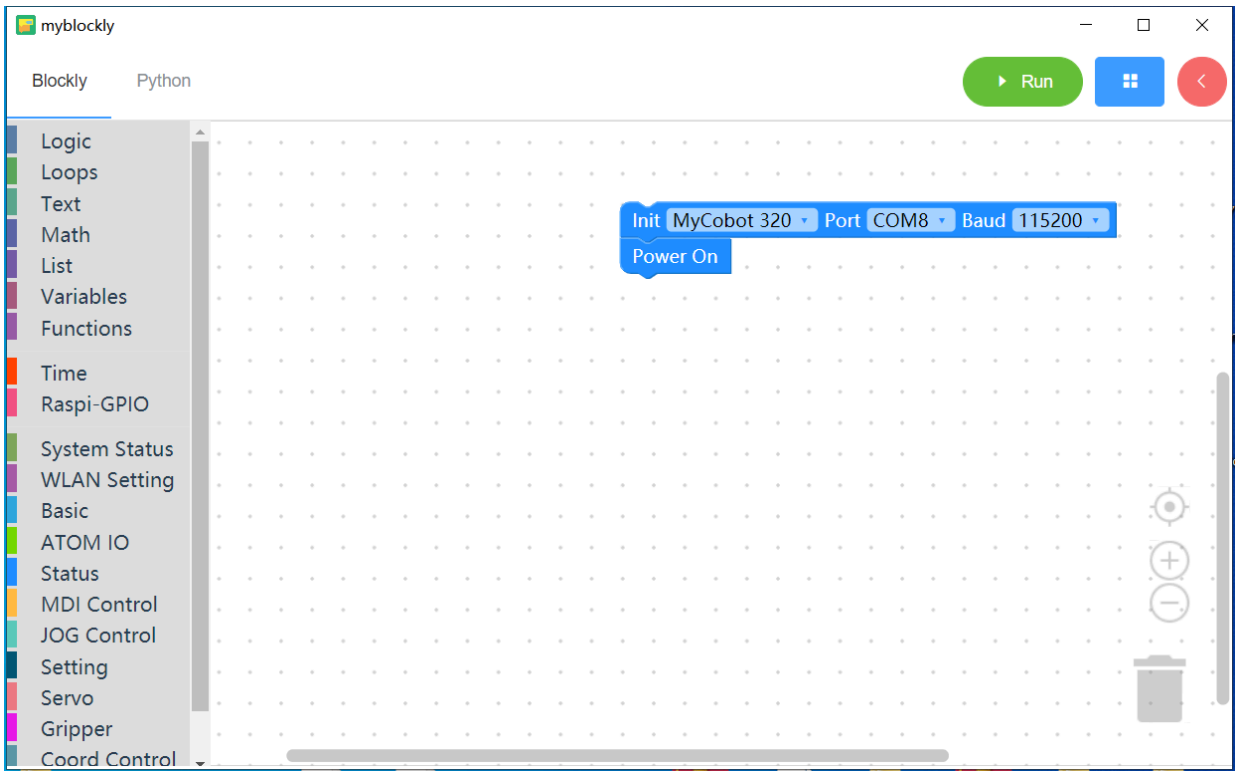
small case

We use building blocks to implement such a small case: let the robotic arm return to the zero point, and then move a joint 30 degrees.

- Make sure robot is power on

  Click the `Status` category in the toolbox, select the `Power On` building block and drag it to the workspace





Click to `Run` button



If the `End of program` text appears in the popup, the program is finish.

Check whether the device is successfully powered on.

If the joint of the machine cannot be broken by hand, it is successfully powered on

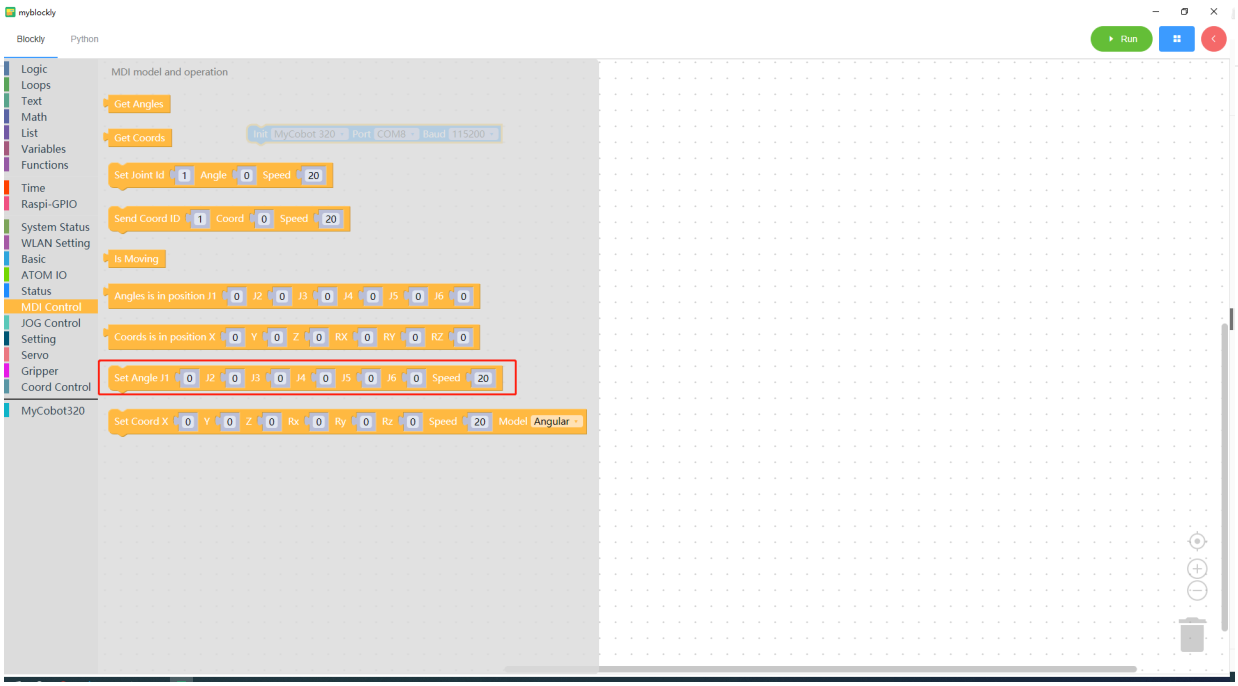
And please click the `X` button to close pop-up window

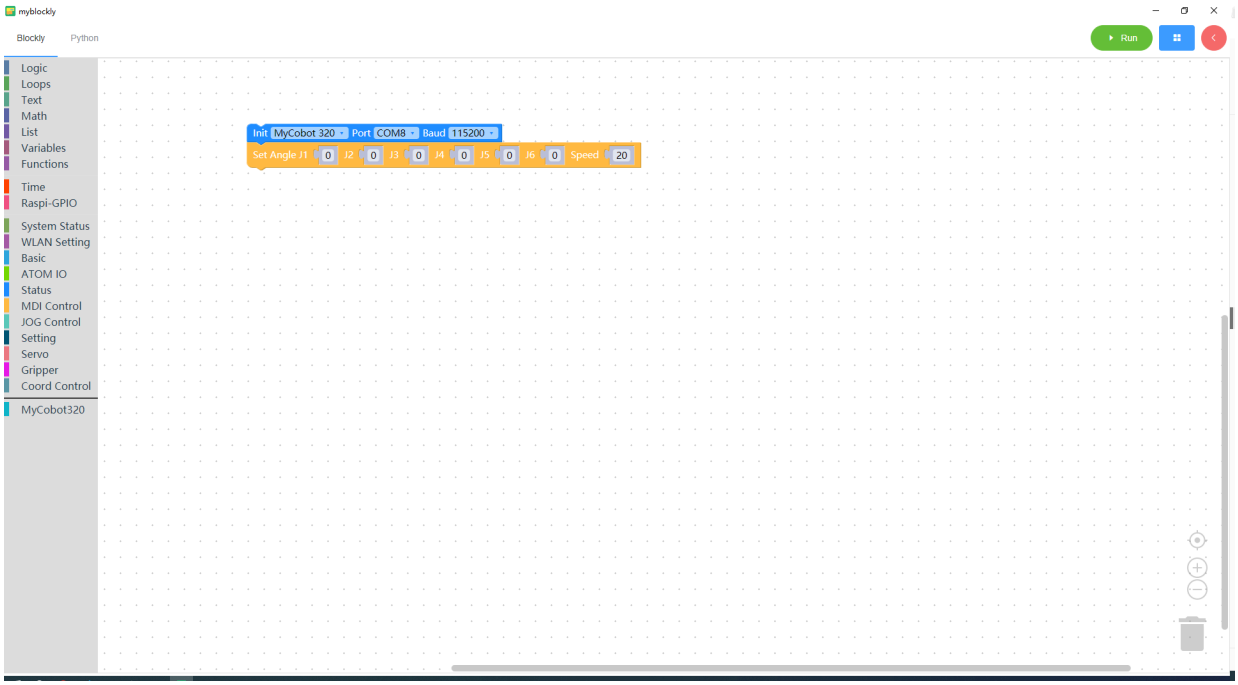


Let's start with our little case study

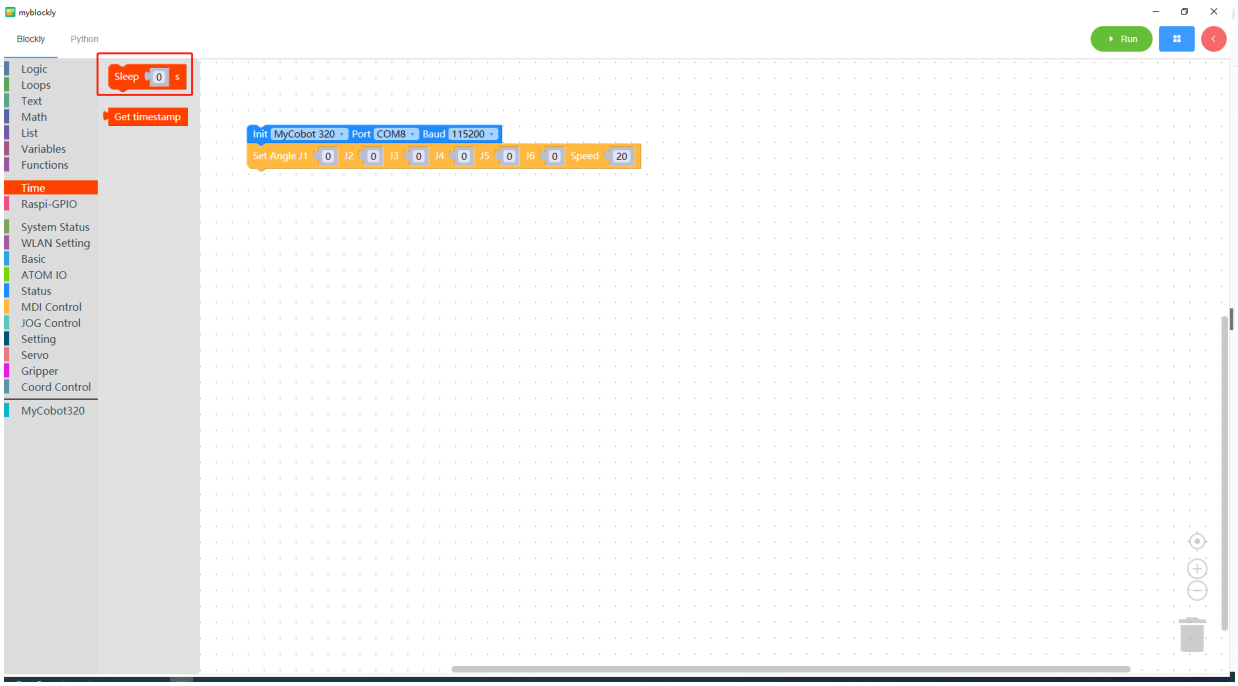
- Click the `MDI Control` category in the toolbox, select the `Set Angle` building block and drag it to the workspace

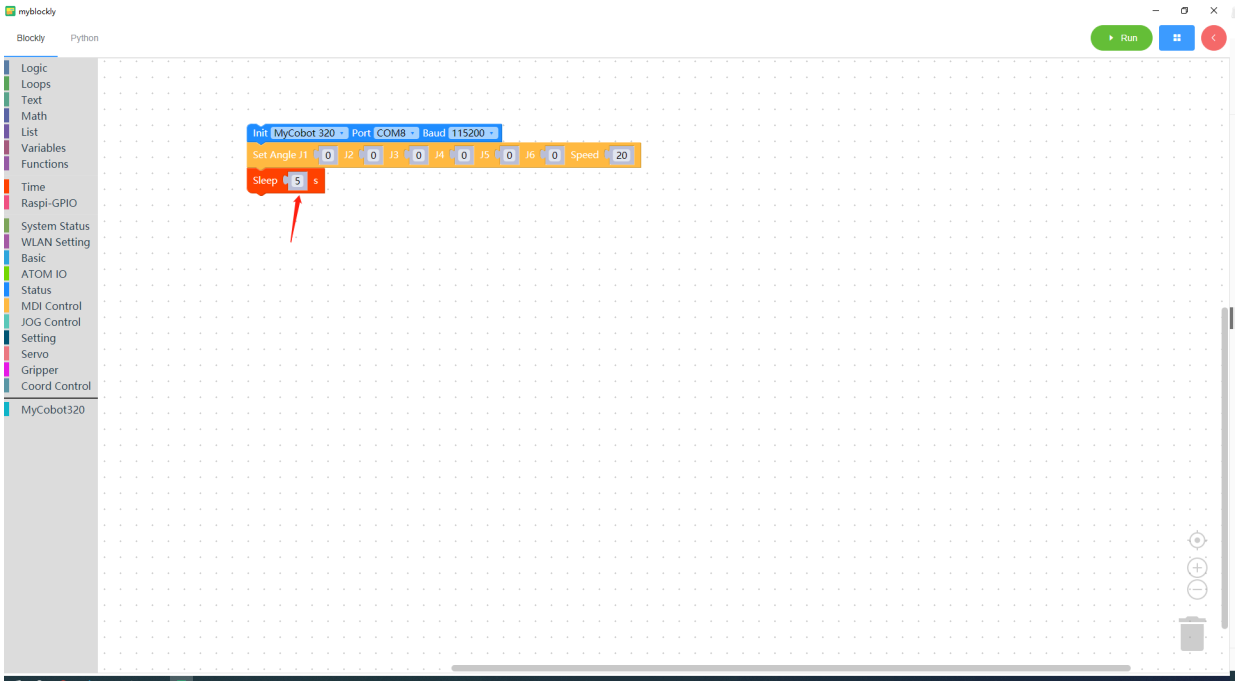
1.4.1 AdaptiveGripper





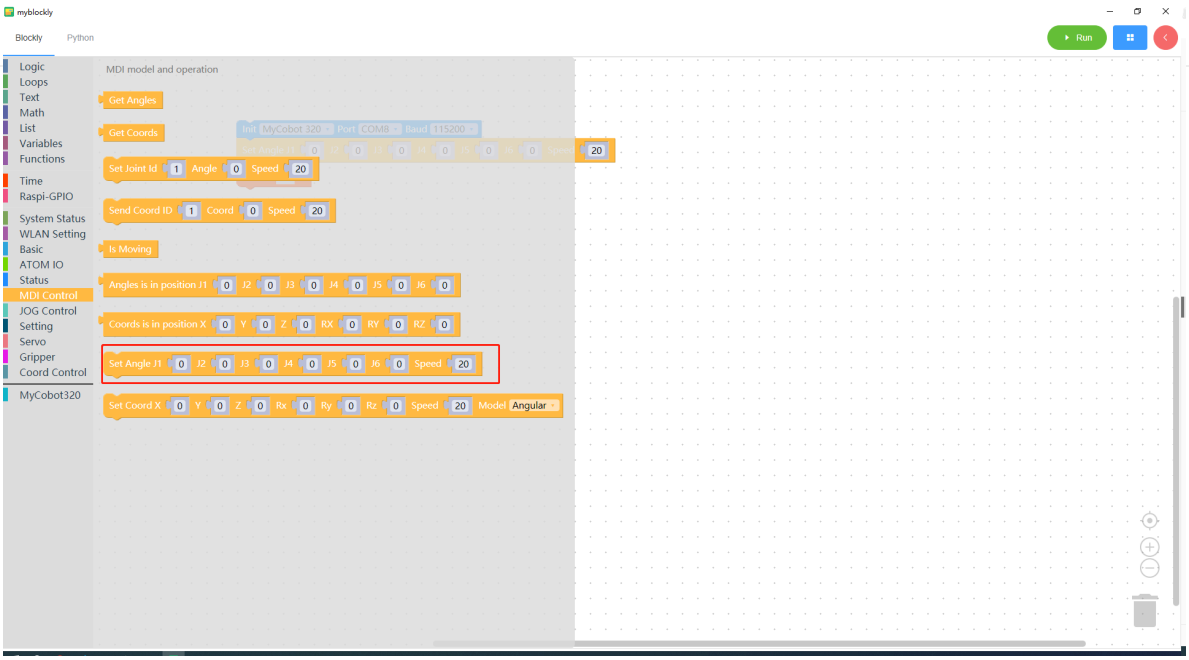
- Click the `Time` category in the toolbox, select the `Sleep` building block and drag it to the workspace; click the mouse on the input box and change the input value of the building block to `5`

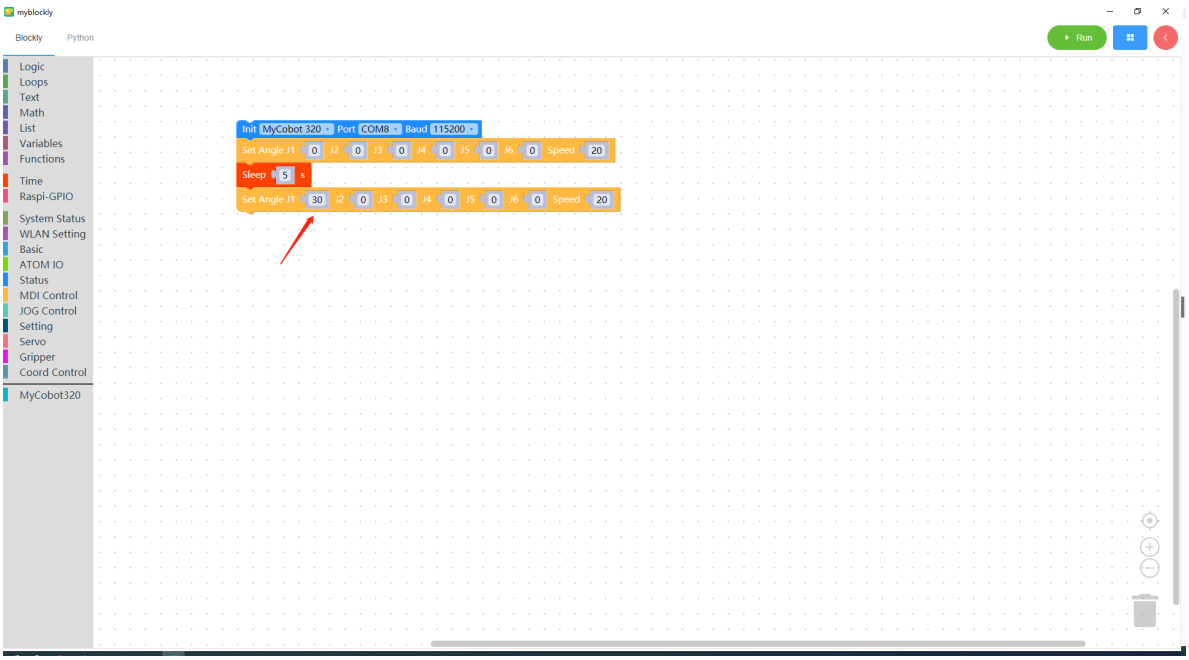




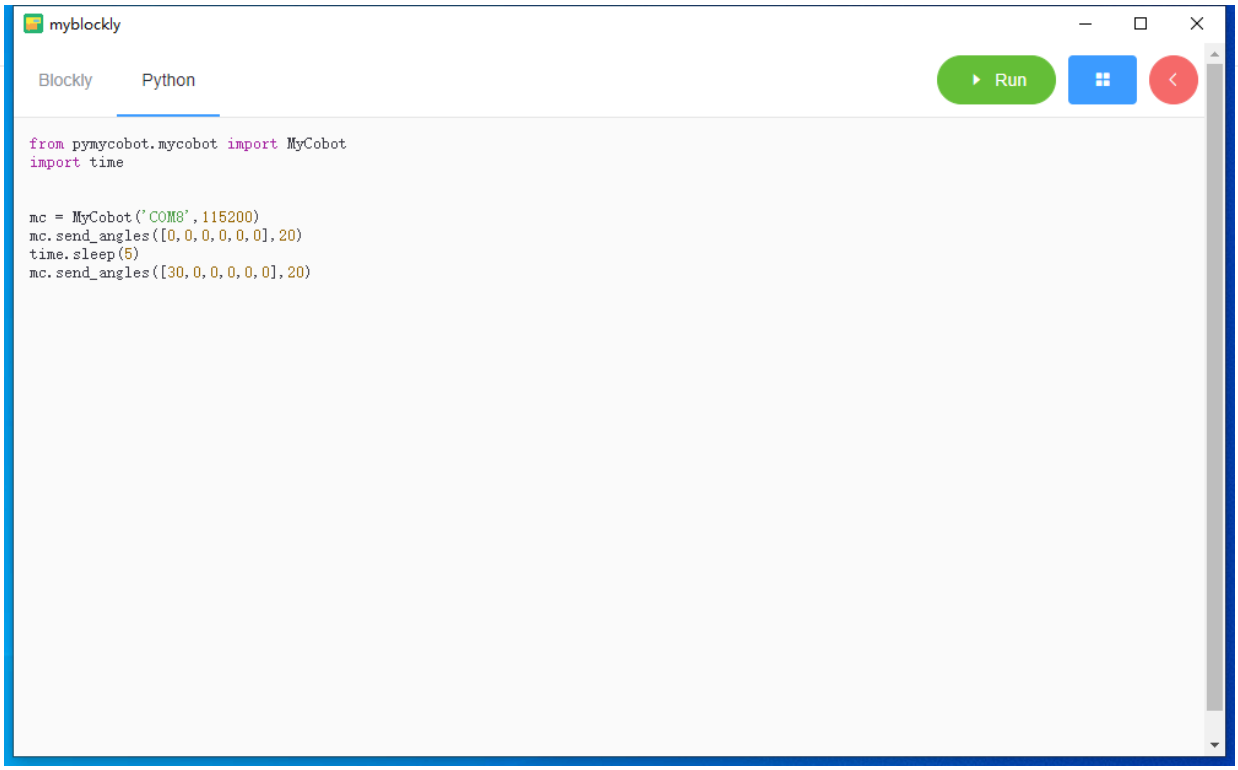
- Click the `MDI Control` category in the toolbox again, select the `Set Angle` building block and drag it to the workspace; at the same time, change the input value of `J1` to 30

1.4.1 AdaptiveGripper

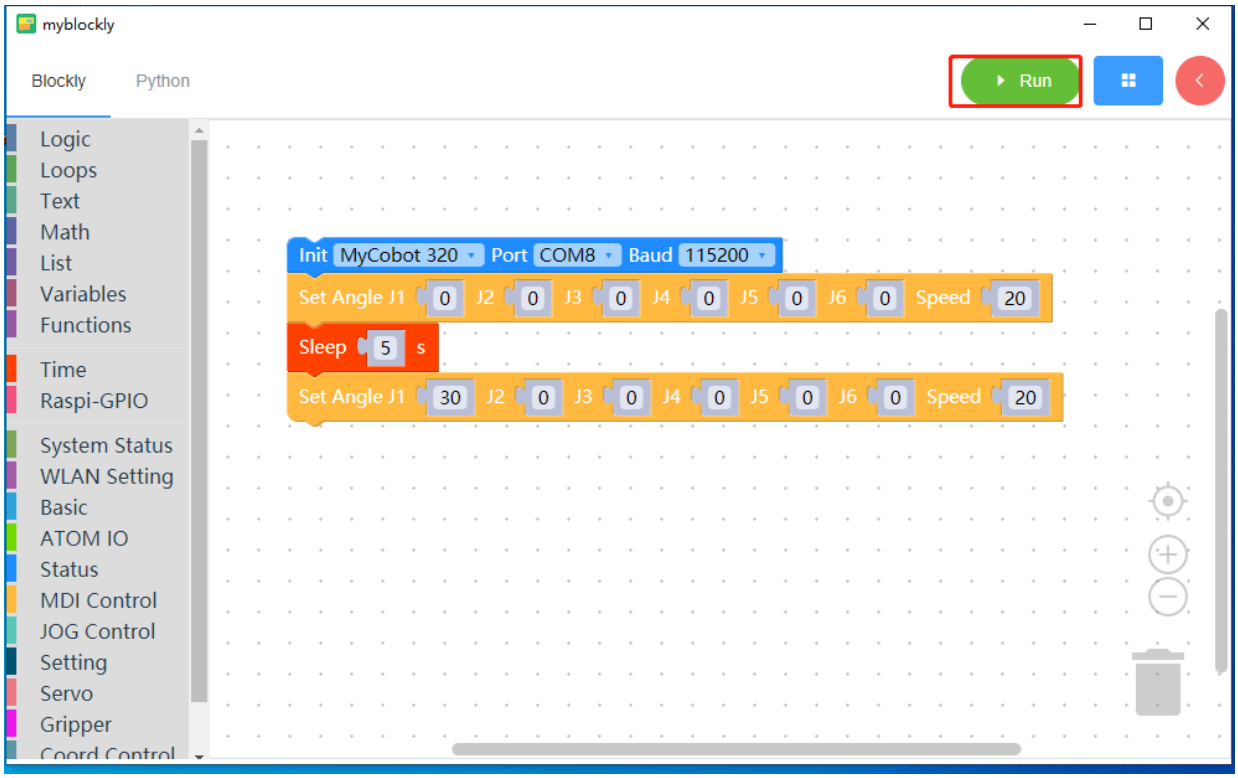




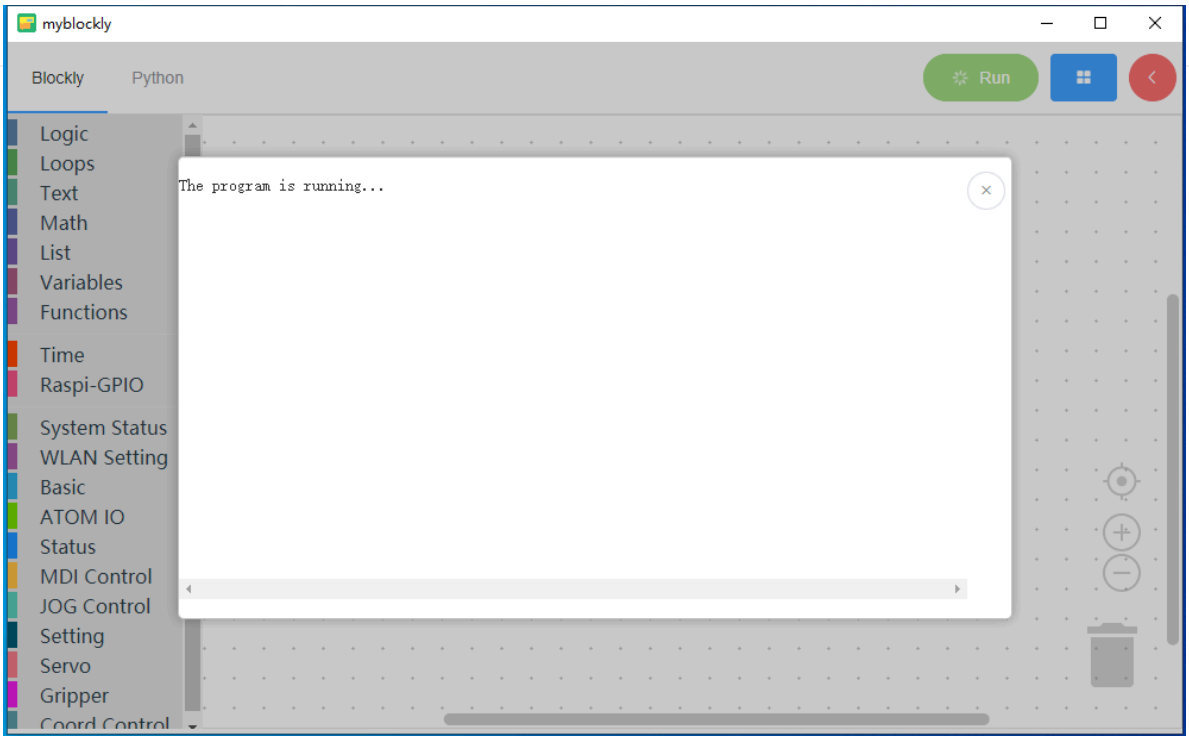
- Click the "Python" option in the upper left corner to view the corresponding Python code, as shown in the figure below.



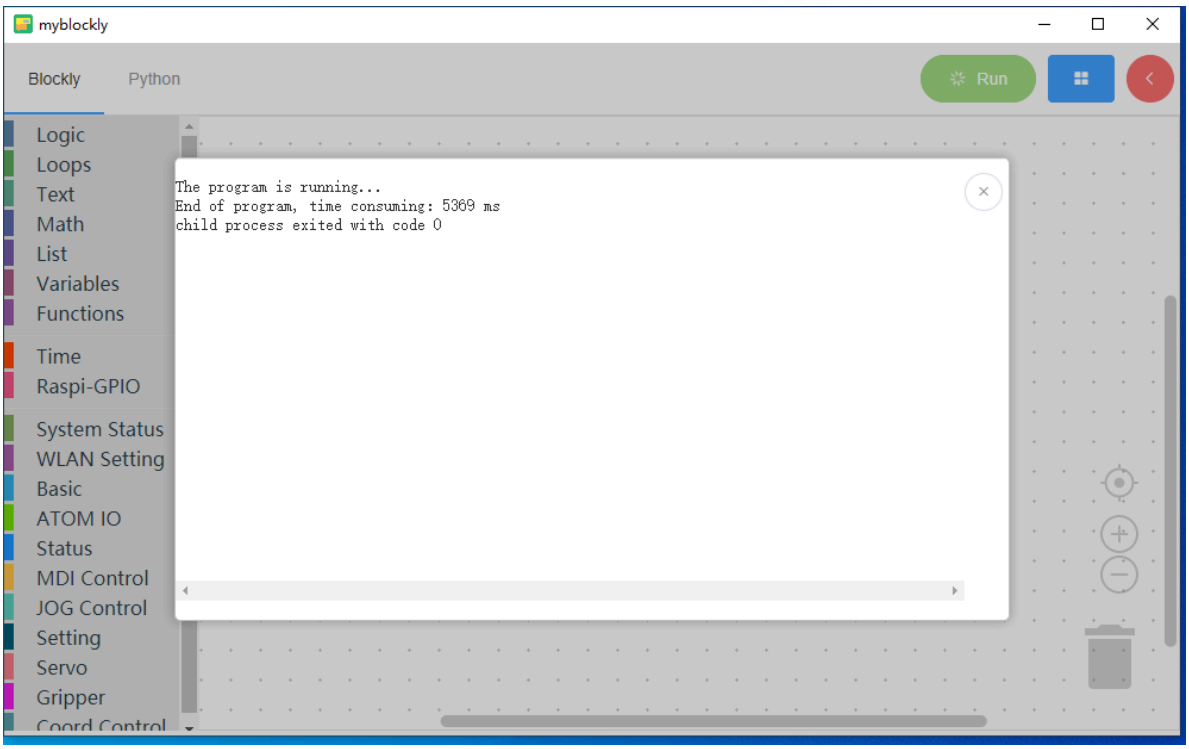
- Click the `Run` button to run the code and observe the movement of the robotic arm.



- Program is running



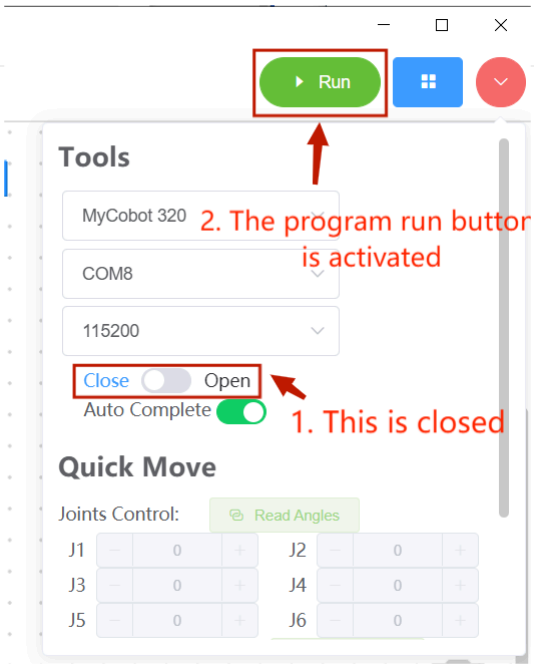
- The program ends

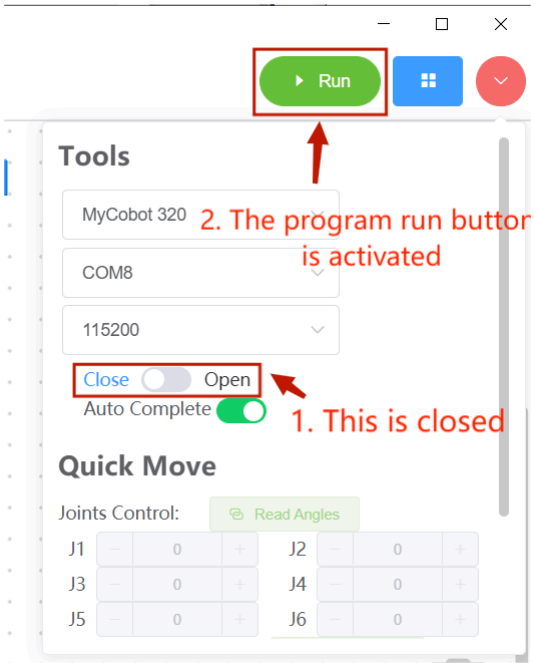


- Click the `x` button in the upper right corner of the run pop-up window to close the pop-up window

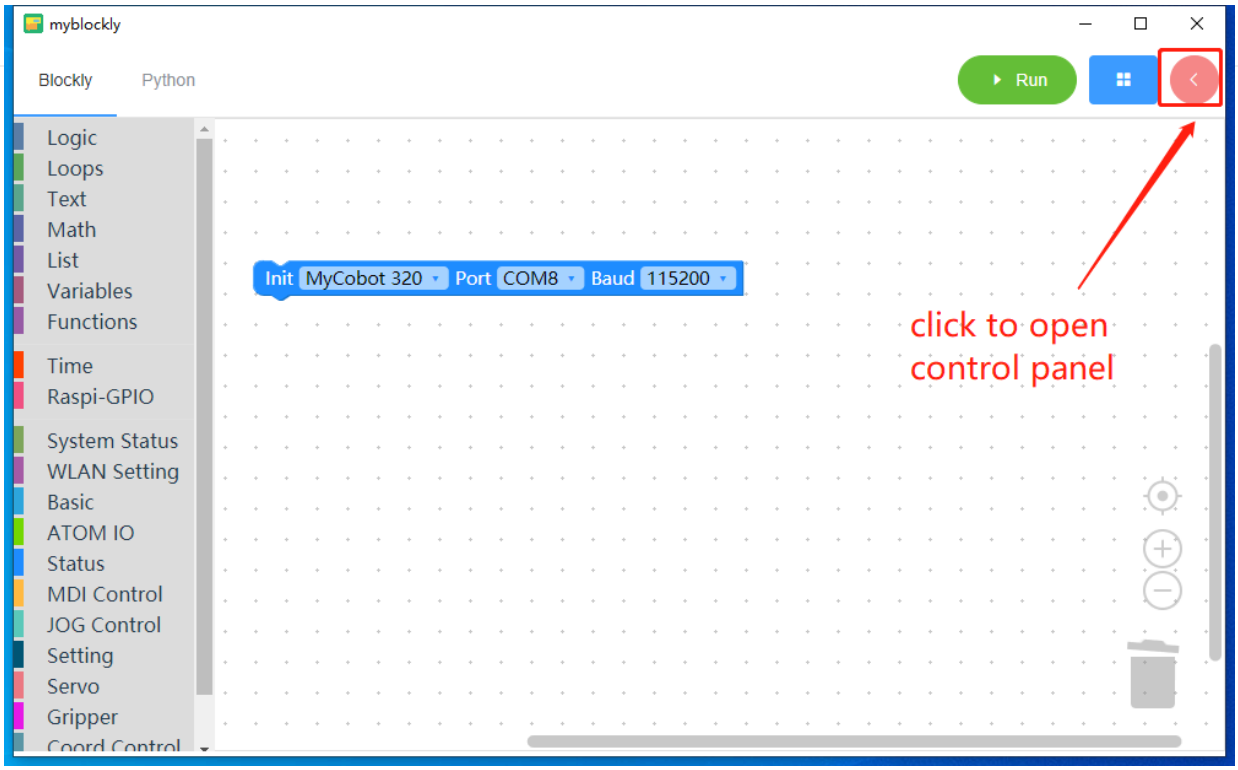
Notice:

- The program to operate the robot arm takes time to complete, so a `sleep' module needs to be connected after one action to give the robot arm time to move before proceeding with the next movement. (You decide the time required based on the situation. By default, the robotic arm is set to run myBlockly for a minimum sleep time of no less than 0.5 seconds.) Otherwise, the robotic arm will not be able to achieve the ideal movement.
- When the program cannot run, please check whether the small toolbar is disconnected (as shown in the figure below).





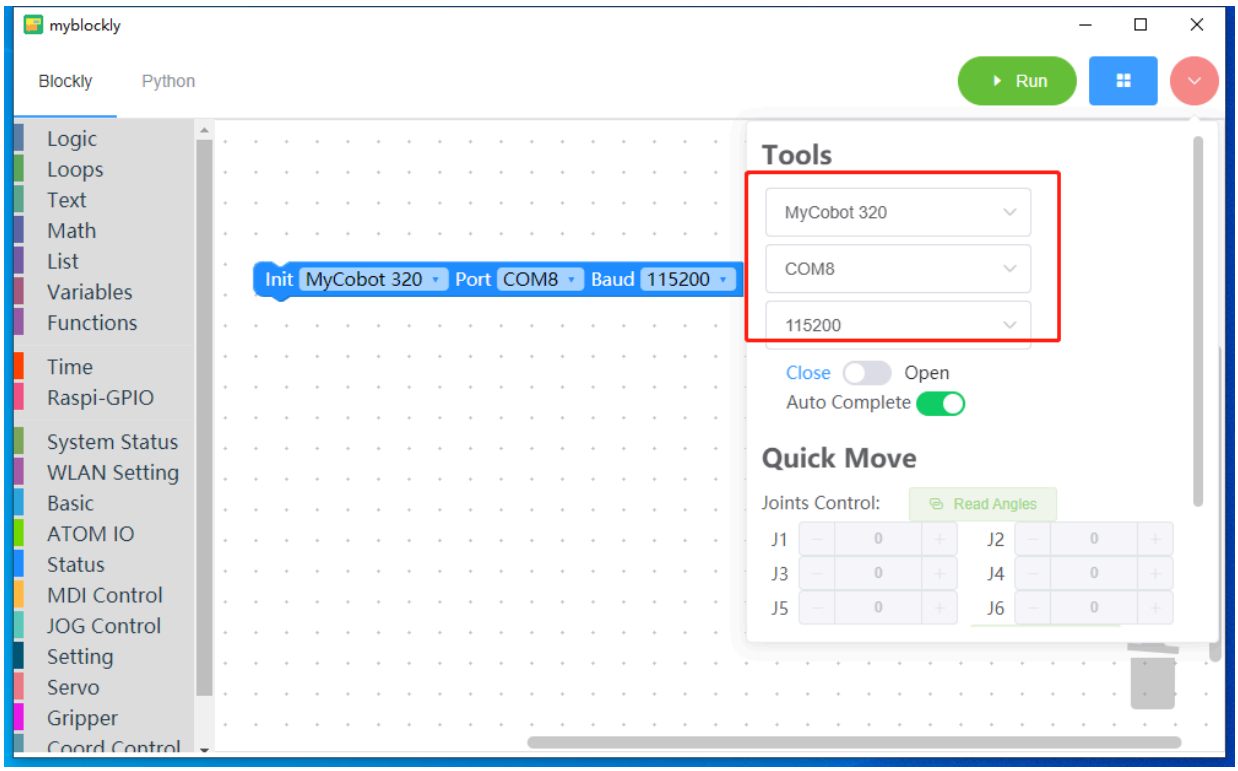
Using the control panel



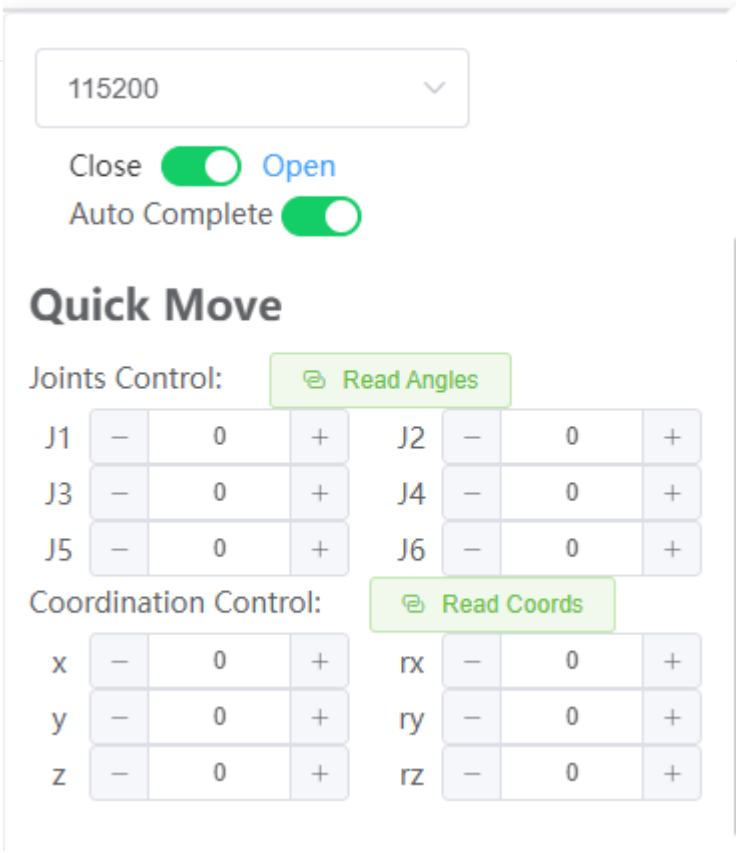
When using the control panel to control the machine, be sure to select the corresponding **machine model**, otherwise it will easily cause hardware damage.

Here we choose the model `myCobot 320` , `Port` is `COM8` ,and the baud rate `115200`

> Note: The port that is actually connected to the computer shall prevail.



After selecting the model and port, click the `Open` button to connect the robotic arm.



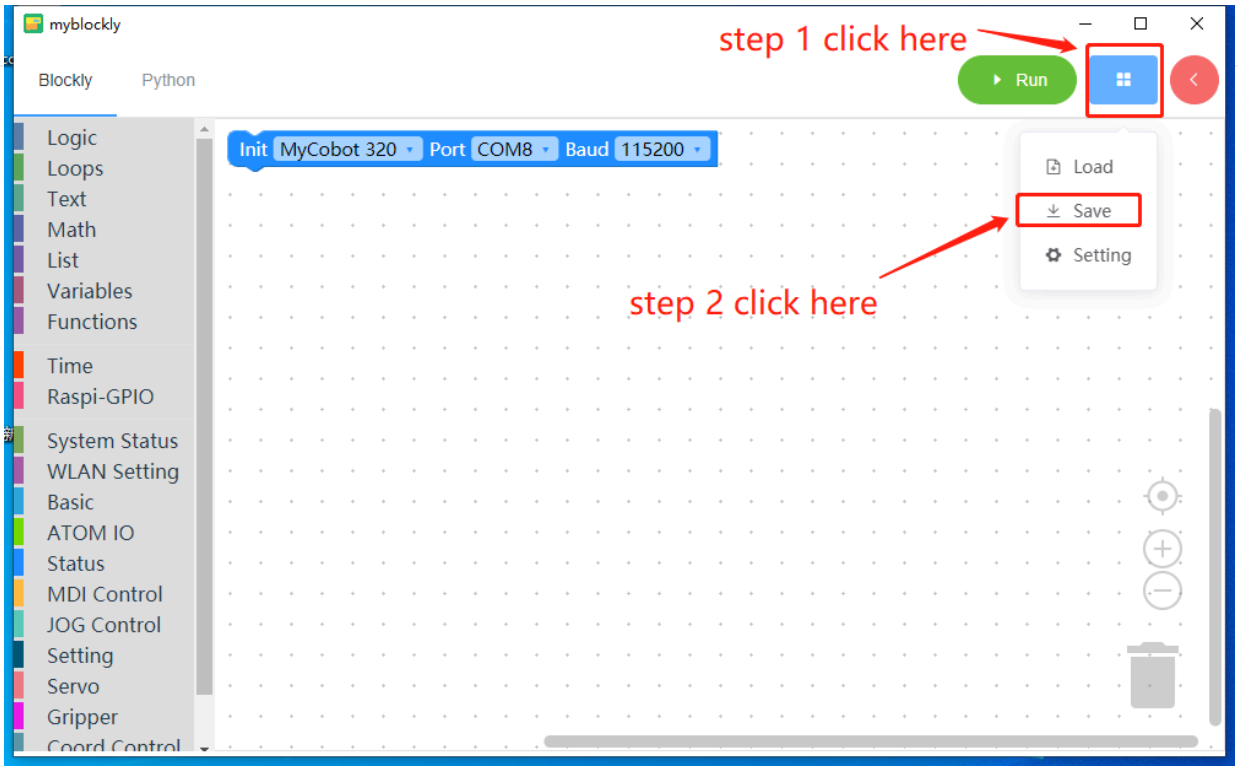
Read angles and coordinates

- Click the read angle button to read the machine angle value
- Click the Read Coordinates button to read the machine coordinates

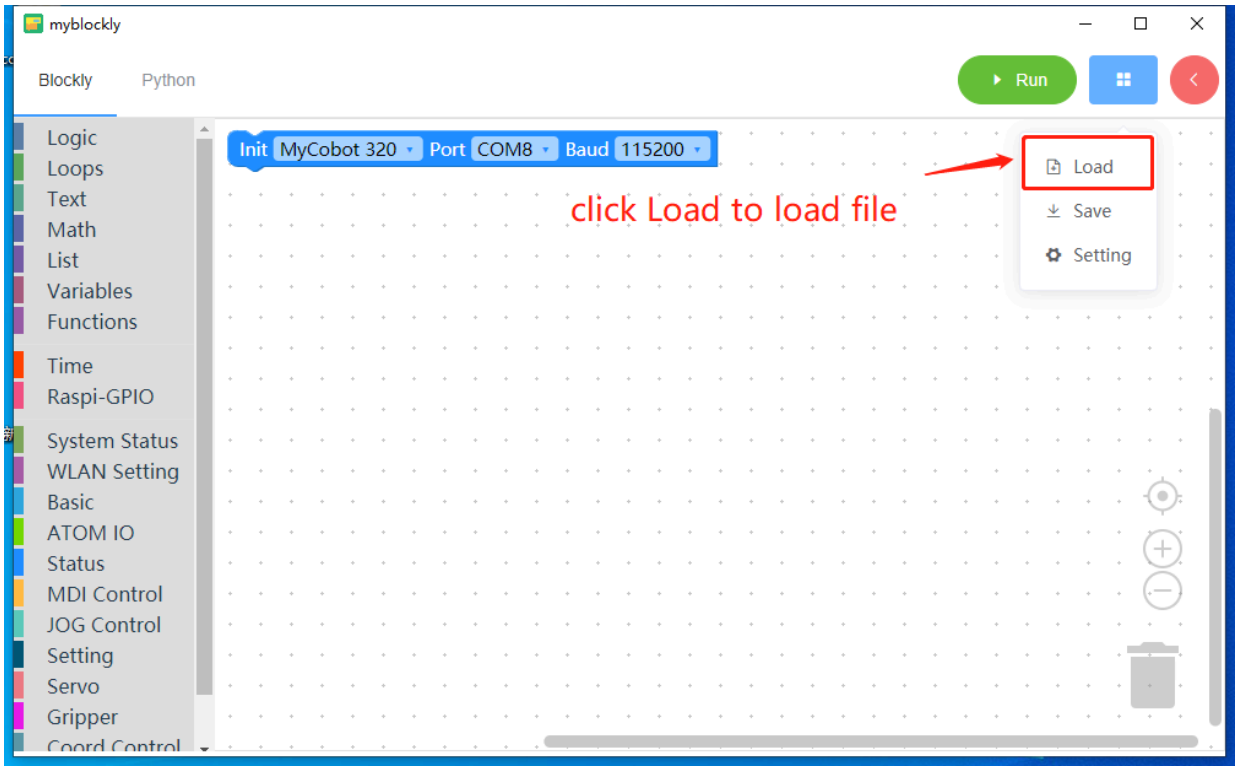
You can control the movement of the robot arm joints by clicking the mouse, long pressing the `-/+` button, and modifying the values.

Program saving and loading

MyBlockly's program is saved in *.json format. Click the blue box in the upper right corner of the interface. When the "Save" option appears, click it to save the program.



Also click the blue box and click the "Load" option to import the saved program.



> **The basic function demonstration is completed, you can check the other information in here**

← Previous Page | Next Page →

Control RGB light panel

Preparation before you begin

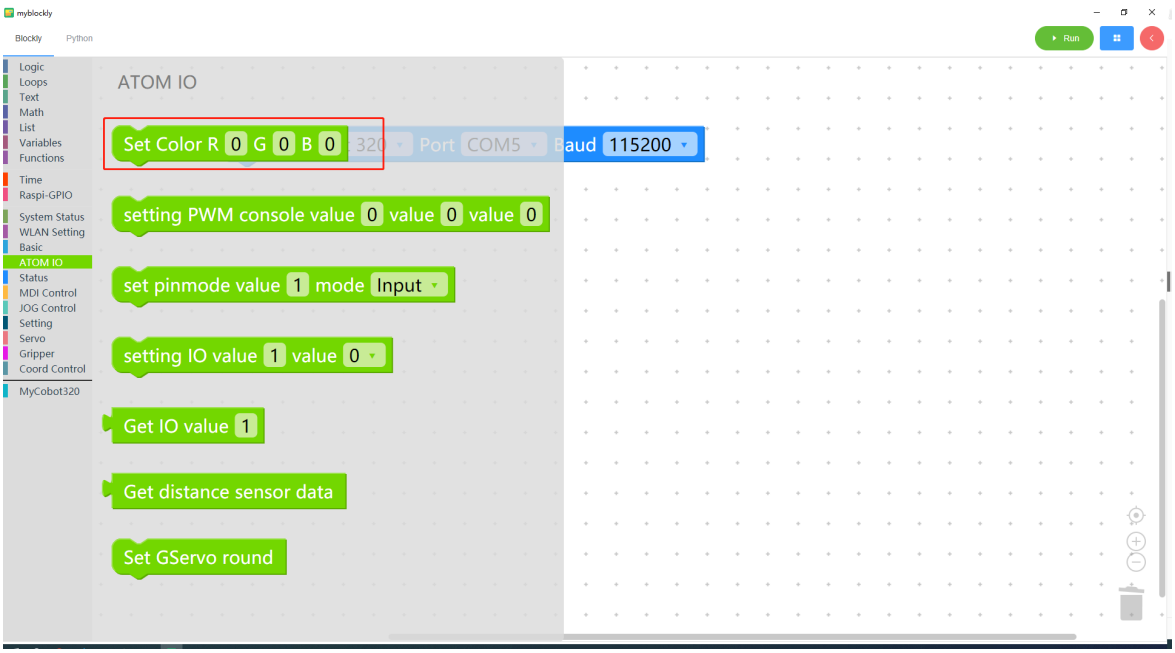
- Make sure the robotic arm is connected to the computer
- Make sure the machine is normal
- Make sure the machine is power on

Learning content of this chapter

How to control RGB light panel using myBlockly

API introduction

- method module： `Set color`





- Parameter introduction:
  - The parameters that need to be set are R (`x`), G (`x`), and B (`x`). Different values represent different colors.
  - Parameter range (for details, please refer to the RGB parameter table):

  *

  R：0~255

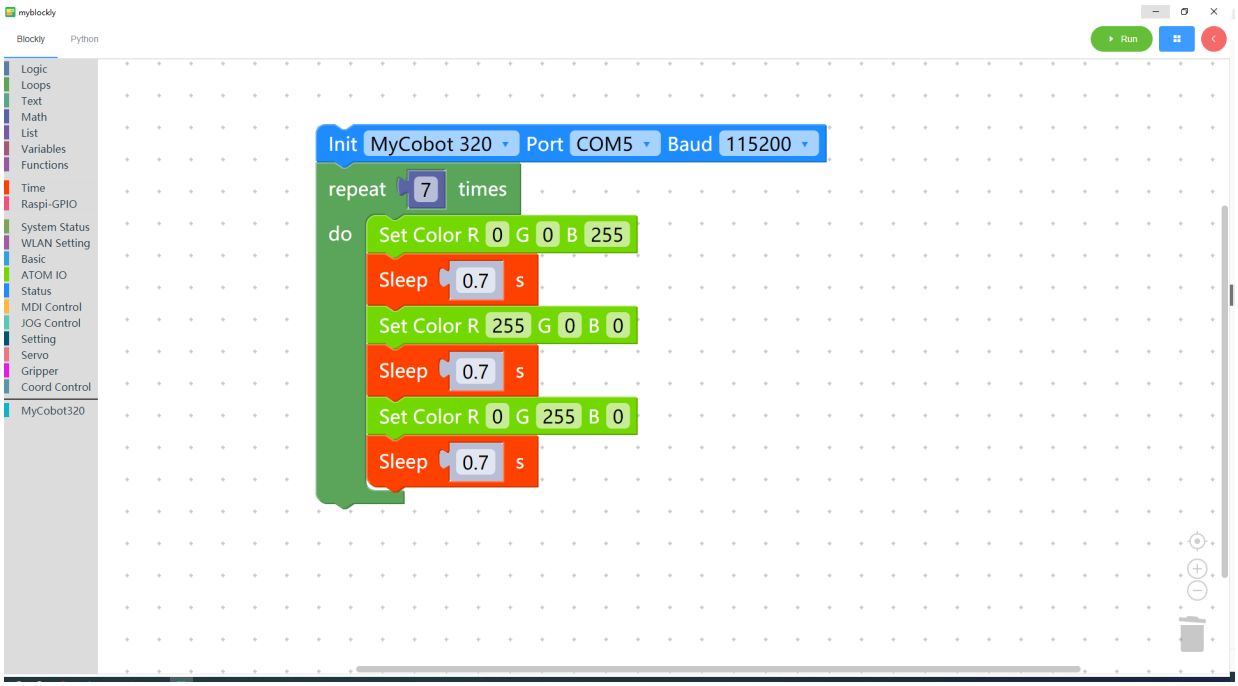
  G：0~255

  B：0~255

- Purpose: Control the color of RGB light panel.

Simple demonstration

The graphics code is as follows:



- What is implemented:

  The color of the robotic arm RGB light panel is controlled to change sequentially from "blue-red-green", and the whole process is cycled seven times.

← Previous Page | Next Page →

Control the robotic arm to return to the origin

Preparation before you begin

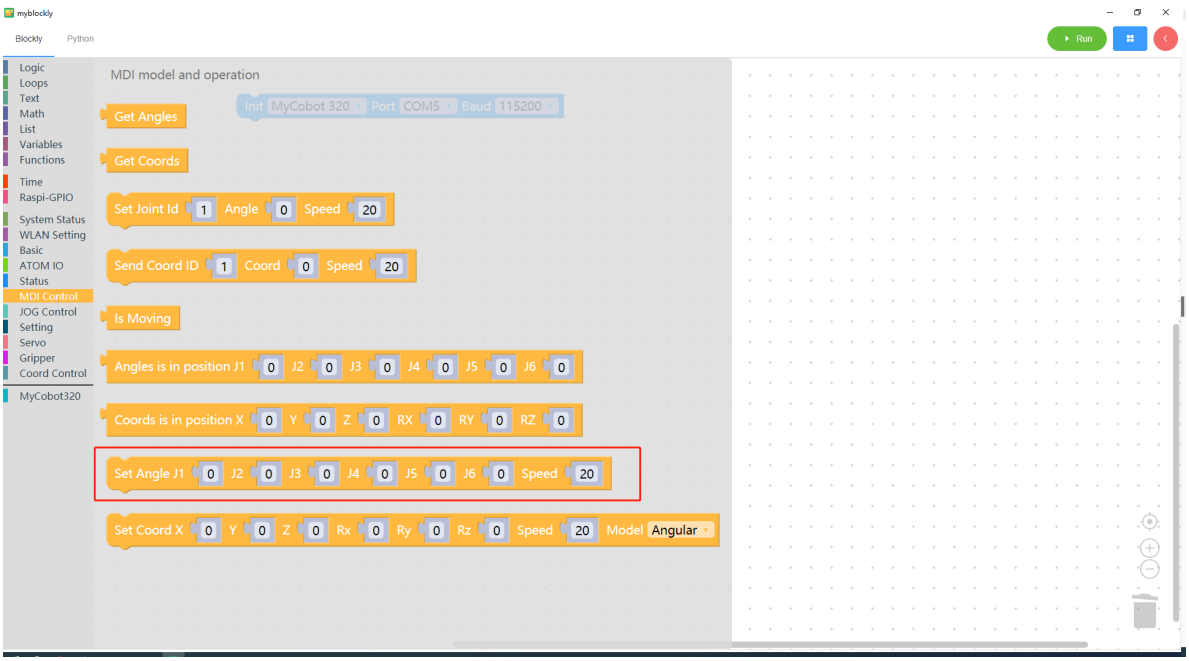
- Make sure the robotic arm is connected to the computer
- Make sure the machine is normal
- Make sure the machine is power on

Learning content of this chapter

How to use myBlockly to control the robot arm to return to the origin

API introduction

- Method module: `Set angle`





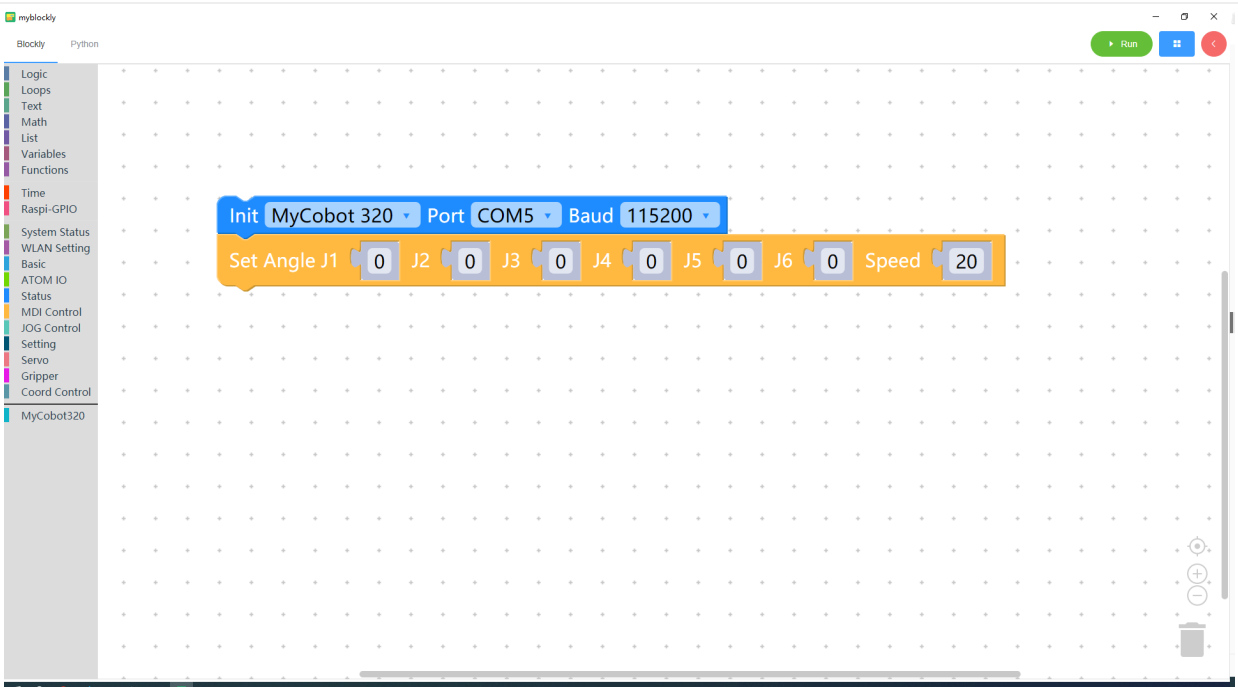
- Parameter introduction:

  This module has two parameters that can be adjusted:

  - Joint angle parameters: If the robot arm returns to the origin, all joint angle parameters need to be set to 0
  - Speed parameter: 0-100

- Purpose: Control the robotic arm and return the angles of all axes of the robotic arm to the origin (angle is 0)

Simple Demonstration



- Implementation content: Control the movement of the robotic arm to return to the origin, so that the angles of all axes of the robotic arm are 0

← Previous Page | Next Page →

Control single joint movement

Preparation before you begin

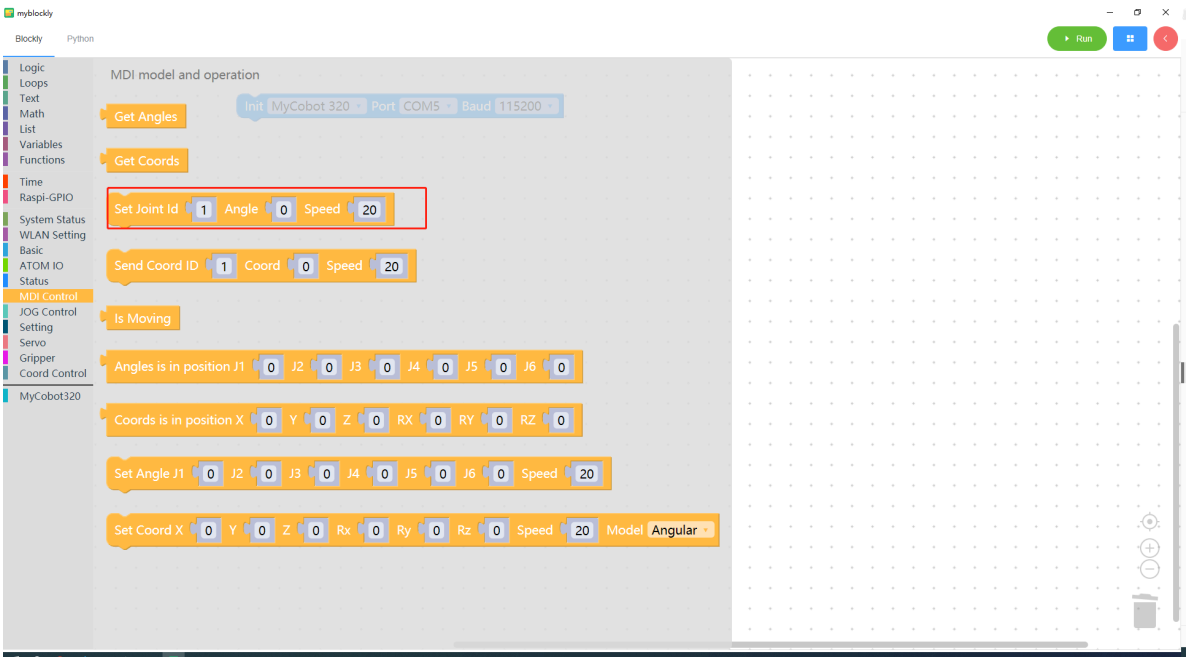
- Make sure the robotic arm is connected to the computer
- Make sure the machine is normal
- Make sure the machine is power on

Learning content of this chapter

How to use myBlockly to control the single joint movement of the robotic arm

API introduction

- method module： `Set Joint`





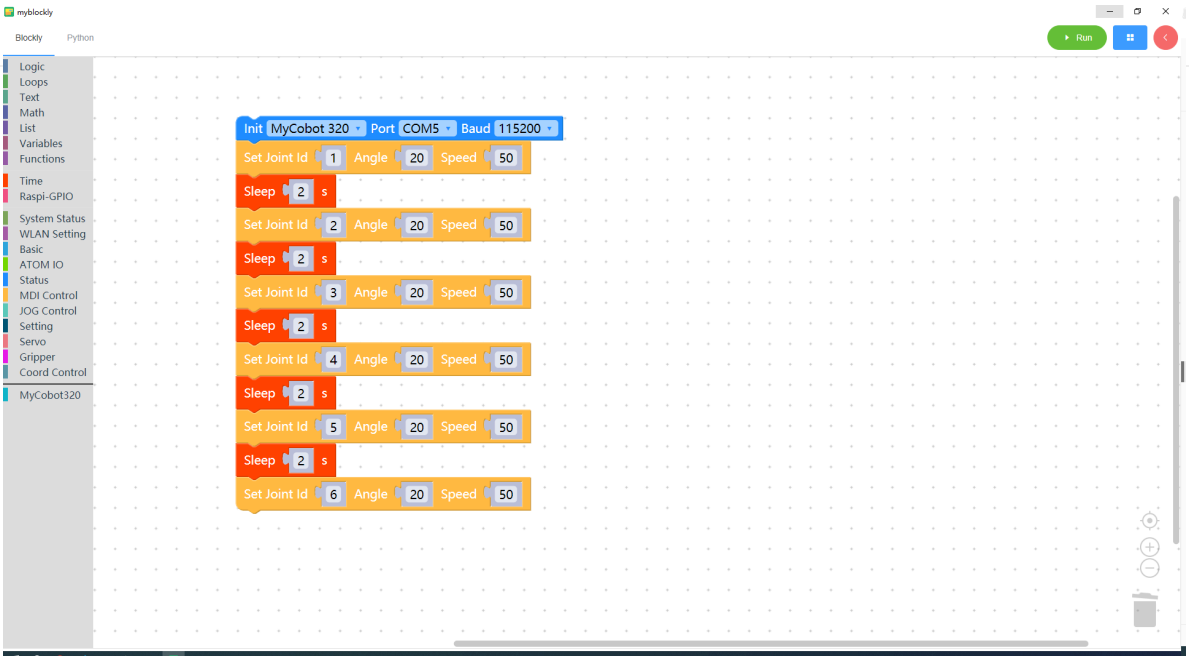
- Parameter introduction:

  This method has three parameters that can be adjusted:

  - Joint parameters: The parameter range is: 1-6 (corresponding to the 6 joints of the robotic arm);
  - Angle parameters: refer to the parameters of the corresponding model
  - Speed: Controls the speed of the robot arm movement. The parameter range is: 0~100
- Purpose: Control the single joint movement of the robotic arm

Simple demonstration

- The graphics code is as follows:



- Implementation content:

  Control joint 1 of the robotic arm to run at a speed of 50 to the position of joint 1 with an angle of 20. After one second,

  Control the 2nd joint of the robotic arm to move at a speed of 50 to the position of the 2nd joint angle of 20. After one second,

  Control the 3 joints of the robotic arm to run at a speed of 50 to the position of the 3 joint angle of 20. After one second,

  Control the 4 joints of the robotic arm to run at a speed of 50 to the position of the 4 joint angle of 20. After one second,

  Control the 5 joints of the robotic arm to run at a speed of 50 to the position of joint 5 with an angle of 20. After one second,

  Control the 6 joints of the robotic arm to run at a speed of 50 to the position of the 6 joint angle of 20

← Previous Page | Next Page →

Control multiple joints

Preparation before you begin

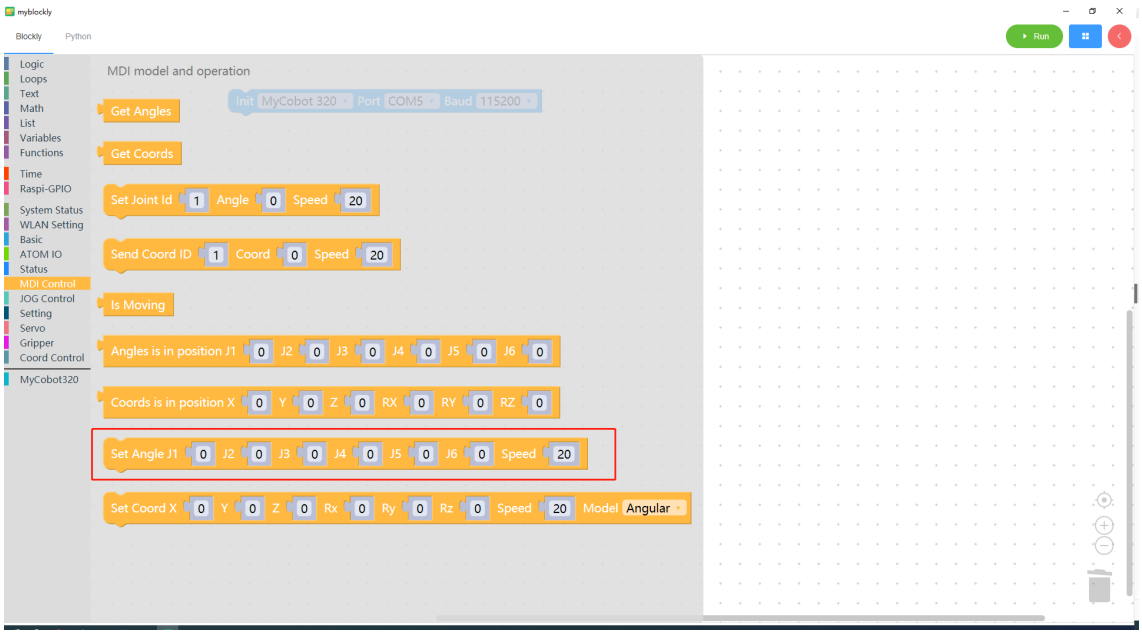
- Make sure the robotic arm is connected to the computer
- Make sure the machine is normal
- Make sure the machine is power on

Learning content of this chapter

How to use myBlockly to control multiple joint movements of a robotic arm

API introduction

- Method 1: Control by joint angle
  - Method module: `Set angles`



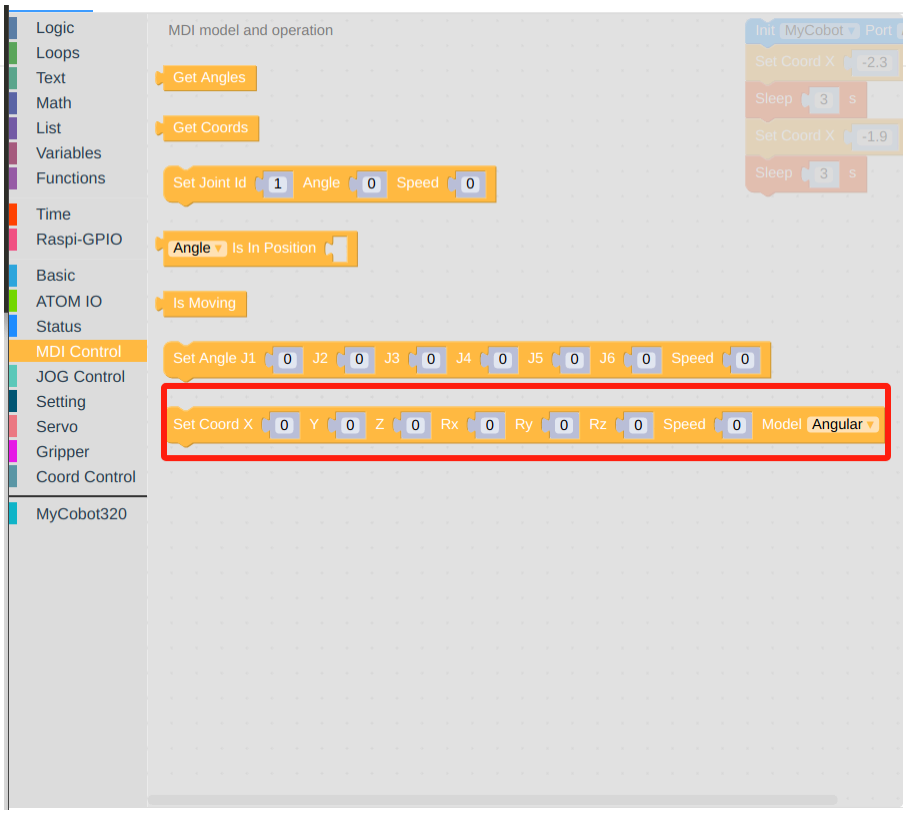


  - Parameter introduction:

    This module has two parameters that can be adjusted:

    - Joint angle parameters
    - Speed parameters

  - Purpose: To control the movement of multiple joints of the robotic arm
- Method 2: Control by coordinates
  - Method module: `Set Coord`





- Parameter introduction:

  This module has two parameters that can be adjusted:

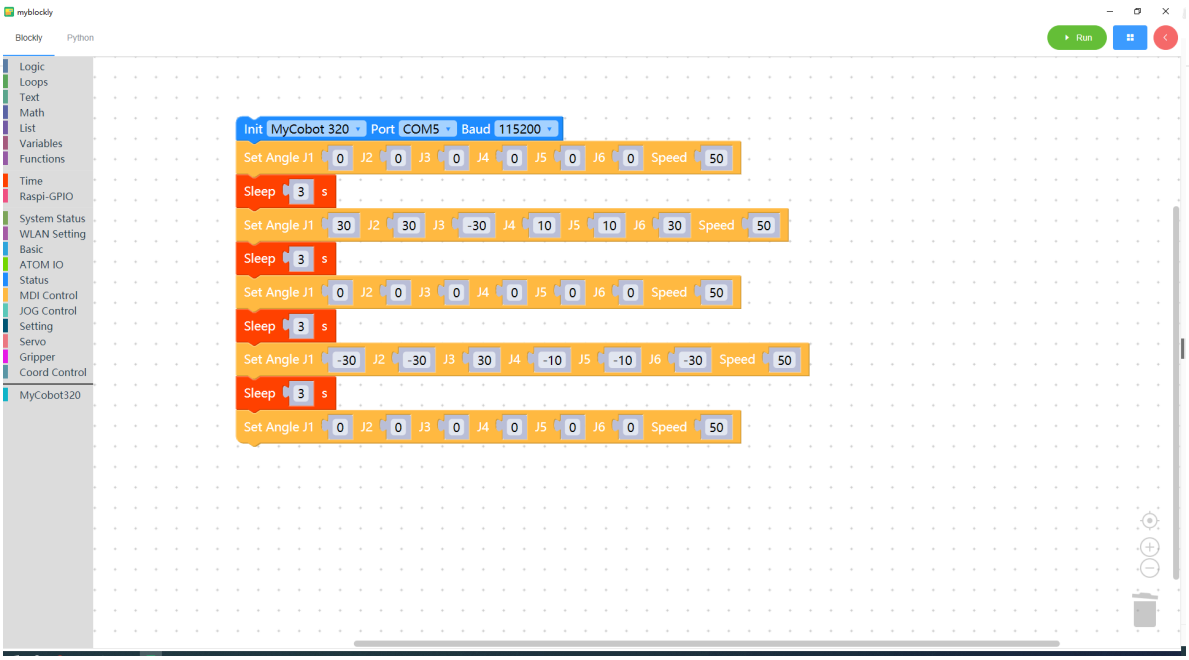
  - coord parameters
  - Speed parameters
  - Route planning approach

- Purpose: Controls the head to arrive at the specified coordinates in a certain attitude.

Simple demonstration

- Method 1: Control by joint angle

  The graphics code is as follows:



- Implementation content:

  Control all joints of the robotic arm to return to the origin. After three seconds,

  At the same time, control the 1 joint, 2 joint, 3 joint, 4 joint, 5 joint, and 6 joint of the robotic arm to move at a speed of 50 to the positions of 30 degrees, 30 degrees, -30 degrees, 10 degrees, 10 degrees, and 30 degrees respectively. After three seconds,

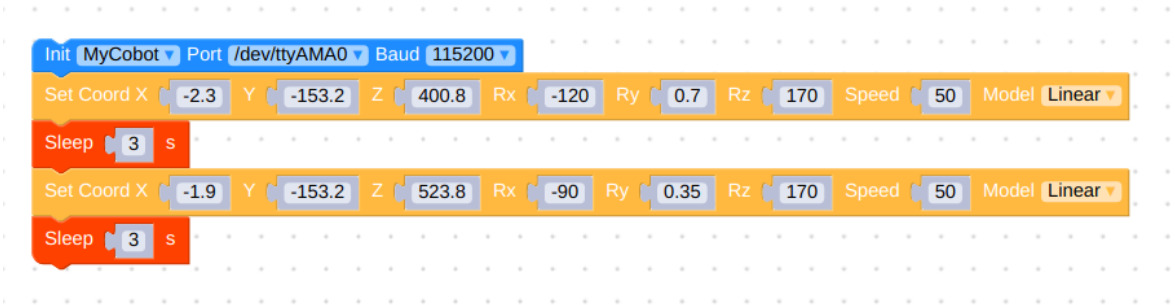
  Return all joints of the robotic arm to the origin at a speed of 50. After three seconds,

  Simultaneously control the 1 joint, 2 joint, 3 joint, 4 joint, 5 joint, and 6 joint of the robotic arm to run at a speed of 50 to -30 degrees, -30 degrees, 30 degrees, -10 degrees, -10 degrees, and -30 degrees respectively. position, after three seconds,

  Control all joints of the robotic arm to return to the origin

- Method 2: Control by coordinates

The graphics code is as follows:



> Note that the parameter at port should be changed to COM8 for the M5 version of the machine.

Intelligent planning of the route, allowing the head to reach the coordinates of [-2.3, -153.2, 400.8] in a linear manner, and maintain the posture of [-120.0, 0.7, 170], with a speed of 50mm/s

Set the waiting time to 3 seconds

Intelligent planning of the route, allowing the head to reach the coordinates of [-1.9, -153.2, 523.8] in a linear manner, and maintain the posture of [-90, 0.35, 170], with a speed of 50mm/s

Set the waiting time to 3 seconds

← Previous Page | Next Page →

Use of gripper

Preparation before you begin

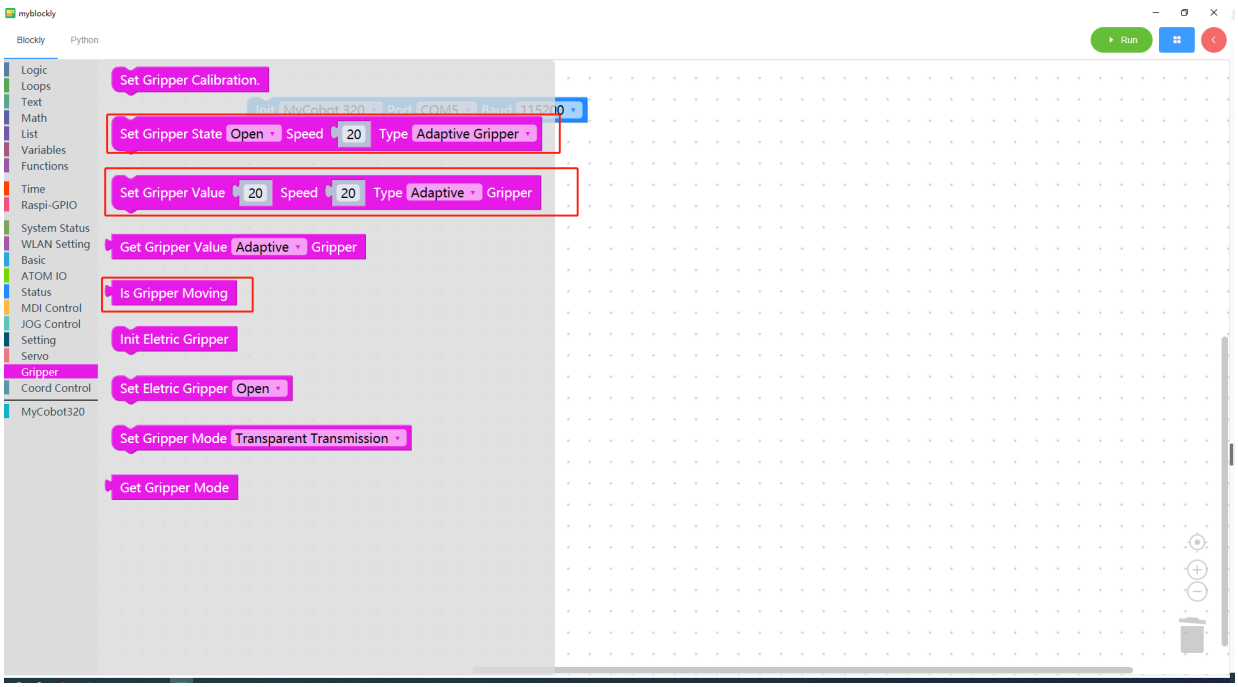
- Make sure the robotic arm is connected to the computer
- Make sure the machine is normal
- Make sure the machine is power on

Grippers include adaptive grippers, electric grippers and pneumatic grippers. Here we take the adaptive gripper as an example to explain how to use myBlockly to control the gripper.

Learning content of this chapter

How to use myBlockly to control the adaptive gripper attached to the myCobot 320 M5Stack robotic arm

API display



- Method module **1**: `Set gripper status`

  Parameter introduction:

  This module has two parameters that can be adjusted:

  - Clamp status parameter: 1 indicates the clamp claws closed state, 0 indicates the clamp claws open state
  - Speed parameter: indicates the speed at which to rotate, the value range is 0~100
  - Clamp type parameter: select adaptive clamp here

Purpose: Make the gripper enter the specified state (open or closed) at a specified speed

- Method module **2**: `Set the value of the gripper`



Parameter introduction:

This module has two parameters that can be adjusted:

- The parameter of the gripper value: indicates the position that the gripper wants to reach, and the value range is 0~100.
- Speed parameter: indicates the speed at which to rotate, the value range is 0~100.
- Clamp type parameter: select adaptive clamp here

Purpose: Make the gripper rotate to a specified position at a specified speed.

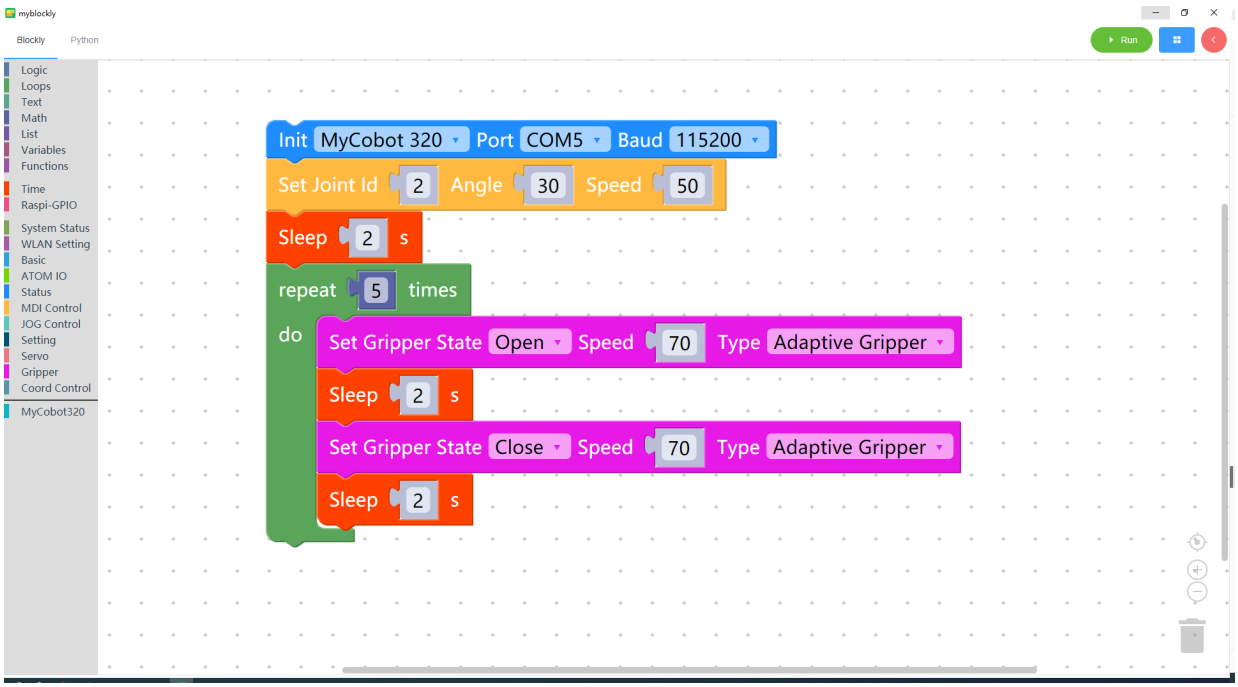
- Method module **3**: `Is the gripper in motion`



Purpose: To determine whether the gripper is running

Simple demonstration

The graphics code is as follows:



- Implementation content:

  Move joint 2 of the robotic arm to 30 degrees at a speed of 50. After two seconds,

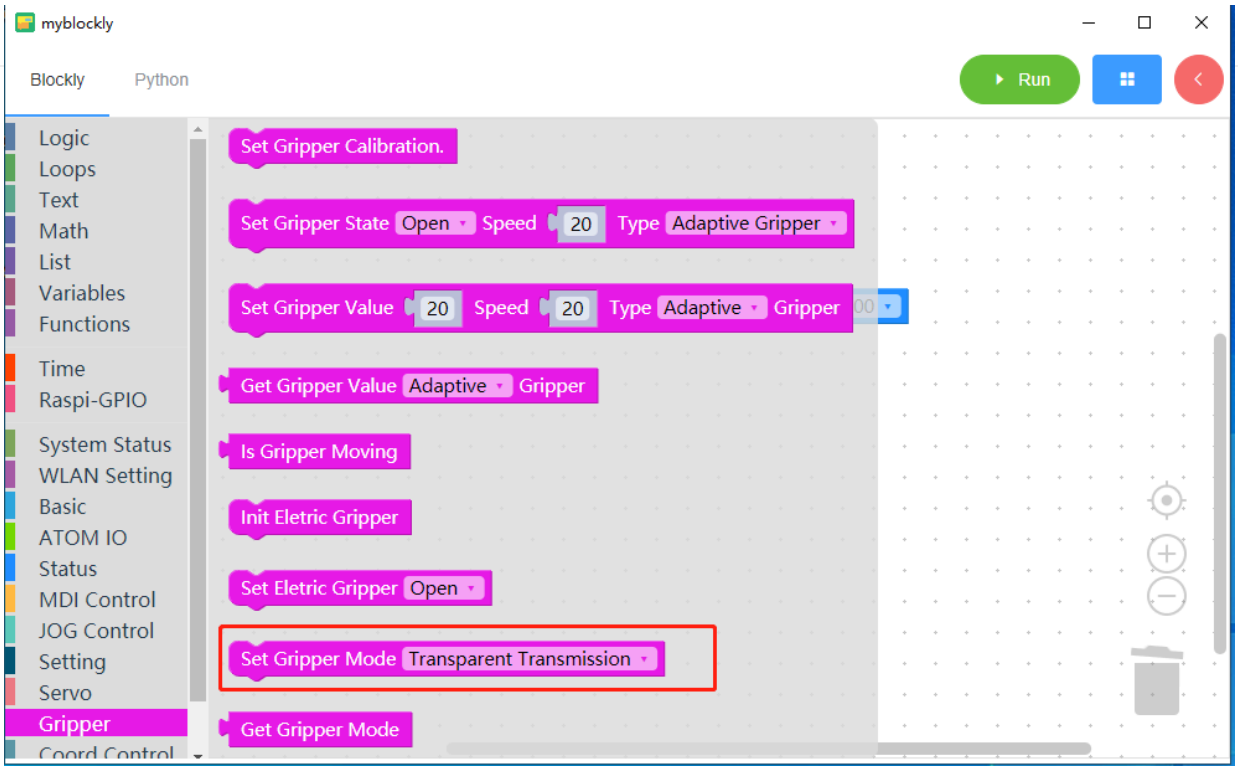
  The gripper opens at a speed of 70. After two seconds,

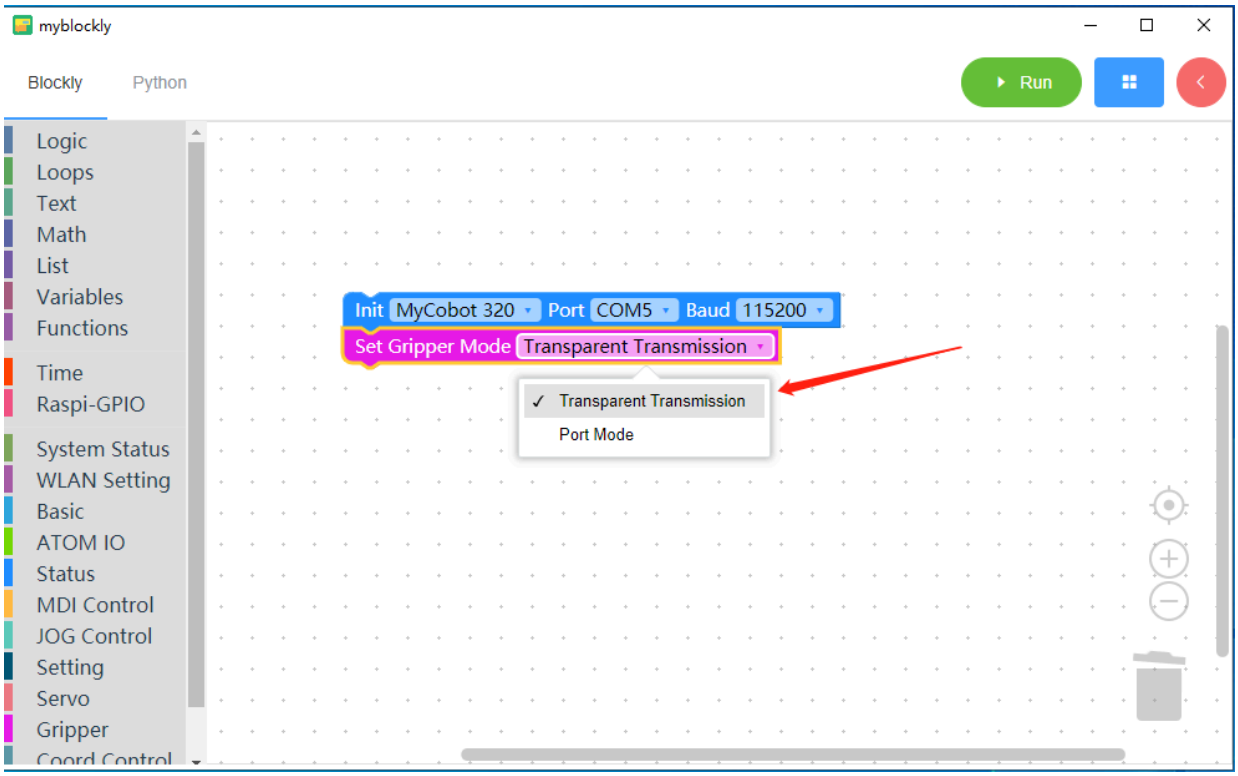
  The gripper close at speed 70,

  After the process of opening and closing the gripper is cycled 5 times, the program ends.

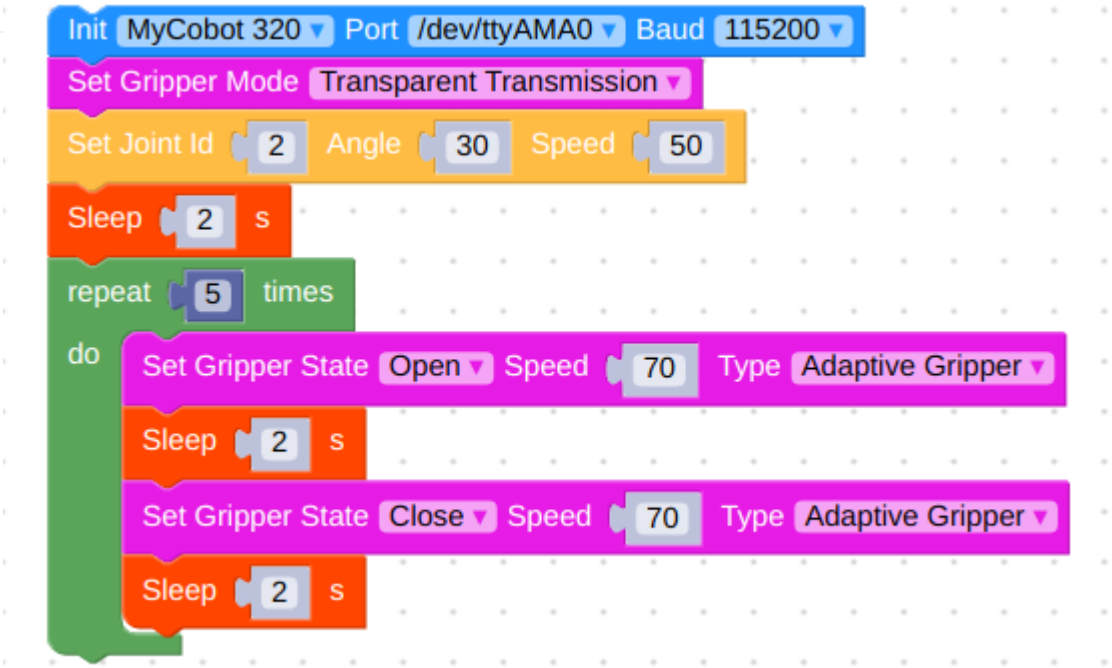
Note:

If you can't control the girpper from the above example, maybe you need to set the gripper mode





And run the program



> Note: If you are using an M5 version of the machine, you need to change the port parameter to COM

← Previous Page | Next Page →

Detailed explanation of myblockly building blocks

System Status

Get Basic firmware version



- **Prototype**: `get_basic_version()`
- **Interface Description**: Get the Basic firmware version.
- **Return**: Return to Basic firmware version.

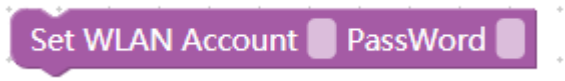
Get Master Control Info



- **Prototype**: `get_system_version()`
- **Interface Description**: Get main control version information.
- **Return**: Return to Basic firmware version.

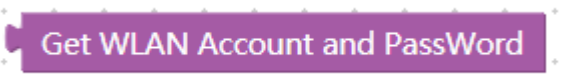
WLAN Setting

Set WLAN account password



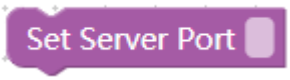
- **Prototype**: `set_ssid_pwd(account,password)`
- **Interface Description**: Set WLAN account password.
- **Parameters**:
  - wlan account
  - wlan password

Get WLAN Account and Password



- **Prototype**: `get_ssid_pwd()`
- **Interface Description**: Get the WLAN account password.
- **Return**:
  - wlan account
  - wlan password

Set Server Port



- **Prototype**: `set_server_port(port)`
- **Interface Description**: Set the port for WLAN connection.
- **Parameters**:
  - The port number

Basic

Set basic pin output



- **Prototype**: `set_basic_output(id,state)`
- **Interface Description**: Set base pin output
- **Parameters**:
  - Pin number
  - Select output status

Get basic pin input



- **Prototype**: `set_basic_output(id)`
- **Interface Description**: Get base pin input
- **Return**:
  - Base pin input status

ATOM Io

Set Color



- **Prototype**: `set_color(r=0,g=0,b=0)`
- **Interface Description**: Set the color of the end LED light.
- **Parameters**
  - red
  - green
  - blue

Set end pin mode



- **Prototype**: `set_pin_mode(id,state)`
- **Interface Description**: Set the end pin mode.
- **Parameters**
  - Pin number
  - Select mode

Set IO value



- **Prototype**: `set_digital_output(id,state)`
- **Interface Description**: Set IO value
- **Parameters**
  - io serial number
  - Select status 0 or 1

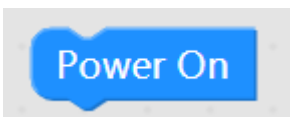
Read IO value



- **Prototype**: `get_digital_input(id)`
- **Interface Description**: Read IO value
- **Parameters**
  - io serial number

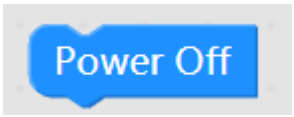
Status

Power On



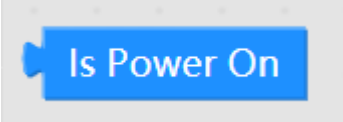
- **Prototype**: `power_on()`
- **Interface Description**: Atom opens communication (open by default).

Power Off



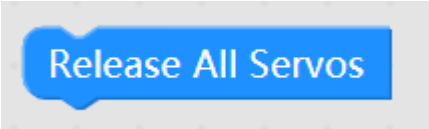
- **Prototype**: `power_off()`
- **Interface Description**: Atom closes communication.

Check whether the robot arm is powered on



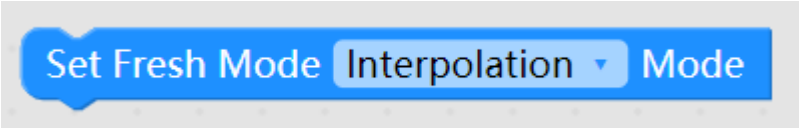
- **Prototype**: `is_power_on()`
- **Interface Description**: Check whether the robot arm is powered on.
- **Return**:
  - `1` : power on
  - `0` : power off
  - `-1` : error

release all Servos



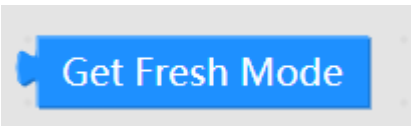
- **Prototype**: `release_all_servos()`
- **Interface Description**: Set the robot arm to free movement mode

Set command refresh mode



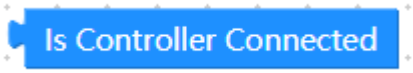
- **Prototype**: `set_fresh_mode(mode)`
- **Interface Description**: Set command refresh mode
- **Parameters**
  - **mode**: `Refresh Sport` - always execute the latest command first. `Interpolation` - Execute instructions sequentially in a queue.

Get command refresh mode



- **Prototype**: `get_fresh_mode()`
- **Interface Description**: Get command refresh mode
- **Return**
  - 1 - Always execute the latest command first.
  - 0 - Execute instructions sequentially in a queue.

Check if Atom is connected



- **Prototype**: `is_controller_connected()`
- **Interface Description**: Check whether Atom is connected
- **Return**:
  - 0: Not connected
  - 1: Connected

MDI Control

Get Angles



- **Prototype**: `get_angles()`
- **Interface Description**: Get the degrees of all joints.
- **Return**:
  - joint angle value

Get Coords



- **Prototype**: `get_coords()`
- **Interface Description**: Get the Cartesian coordinates of the machine.
- **Return**:
  - Cartesian coordinates

Set single joint angle



- **Prototype**: `send_angle(id, degree, speed)`
- **Interface Description**: Robot single joint angle control.
- **Parameters**
  - Joint id: 1-6
  - Angle value
  - speed

Set single coordinates



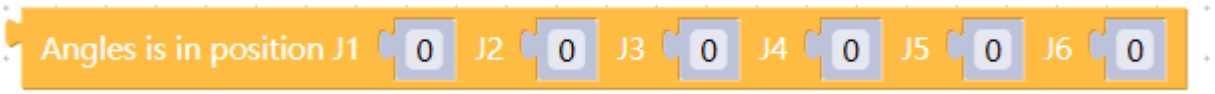
- **Prototype**: `send_coord(id, value, speed)`
- **Interface Description**: Robot single coordinate control.
- **Parameters**
  - Coordinates: 1-6 correspond to x、y、 z、rx、ry、rz
  - coordinate value
  - speed

Check if you are in motion



- **Prototype**: `is_moving(id, value, speed)`
- **Interface Description**: Detect whether the machine is moving.
- **Return**:
  - 0: The machine is not in motion
  - 1: The machine is in motion.

Angles is in position



- **Prototype**: `is_in_position([j1,j2,j3,j4,j5,j6],0)`
- **Interface description**: Check whether the machine angle reaches the specified position
- **Parameters**:
  - j1 angle value
  - j2 angle value
  - j3 angle value
  - j4 angle value
  - j5 angle value
  - j6 angle value
- **Return**:
  - 0: Not reached the specified location
  - 1: Reach the specified location

Coords is in position



- **Prototype**: `is_in_position([x,y,z,rx,ry,rz],1)`
- **Interface description**: Check whether the machine angle reaches the specified position
- **Parameters**:
  - x coordinate value
  - y coordinate value
  - z coordinate value
  - rx coordinate value
  - ry coordinate value
  - rz coordinate value
- **Return**:
  - 0: Not reached the specified location
  - 1: Reach the specified location

Set Angles



- **Prototype**: `send_angles(angles, speed)`
- **Interface Description**: Send all angles to the robot arm.
- **Parameters**
  - `angles` : list of coordinate values (`List[float]`).
  - `speed` : (`int`) 0 ~ 100

Set Coords



- **Prototype**: `send_coords(coords, speed, mode)`
- **Interface Description**: Send all coordinates to the robot arm.
- **Parameters**
  - `coords` : list of coordinate values (`List[float]`).
  - `speed` : (`int`) 0 ~ 100
  - `mode` : (`int`): `0` - angular (default), `1` - linear

-

JOG Control

JOG Angle



- **Prototype**: `jog_angle(joint_id, direction, speed)`
- **Interface description**: Jog control angle
- **Parameters**
  - `joint_id` : (`int`) 1 ~ 6
  - `direction` : `0` - decrease, `1` - increase
  - `speed` : 0 ~ 100

JOG Coord



- **Prototype**: `jog_coord(coord_id, direction, speed)`
- **Interface description**: Jog control coord.
- **Parameters**
  - `coord_id` : (`int`) 1 ~ 6
  - `direction` : `0` - decrease, `1` - increase
  - `speed` : 0 ~ 100

JOG Absolute



- **Prototype**: `jog_absolue(coord_id, direction, speed)`
- **Interface description**: Step mode.
- **Parameters**
  - `coord_id` : (`int`) 1 ~ 6
  - `direction` :
  - `speed` : 0 ~ 100

JOG Increment



- **Prototype**: `jog_increment(coord_id, direction, speed)`
- **Interface description**: Step mode.
- **Parameters**
  - `coord_id` : (`int`) 1 ~ 6
  - `direction` :
  - `speed` : 0 ~ 100

Jog Stop



- **Prototype**: `jog_stop()`
- **Interface Description**: Stop jog moving.

#

Pause



- **Prototype**: `pause()`
- **Interface Description**: Pause movement.

Resume



- **Prototype**: `resume()`
- **Interface description**: Recovery movement.

Stop



- **Prototype**: `stop()`
- **Interface Description**: Stop moving.

is_paused



- **Prototype**: `is_paused()`
- **Interface Description**: Judge whether the manipulator pauses or not.

- **Return**: :
  - `1` - paused
  - `0` - not paused
  - `-1` - error

Set Servo encoder value



- **Prototype**: `set_encoder(joint_id, encoder)`
- **Interface Description**: Set a single joint rotation to the specified potential value.
- **Parameters**
  - `joint_id` : (`int`) 1 ~ 6
  - `encoder` : 0 ~ 4096

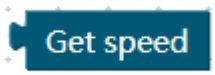
Get Servo encoder value



- **Prototype**: `get_encoder(joint_id)`
- **Interface description**: Obtain the specified joint potential value.
- **Parameters**: `joint_id` : (`int`) 1 ~ 6
- **Return**: `encoder` : 0 ~ 4096

Setting

get_speed



- **Prototype**: `get_speed()`
- **Interface description**: Get speed.
- **Return**: speed: (`int`)

set_speed



- **Prototype**: `set_speed(speed)`

- **Interface description**: Set speed.
- **Parameters**: speed: (`int`) 0 ~ 100

set_joint_min



- **Prototype**: `set_joint_min(id, angle)`
- **Interface description**: Sets the minimum angle for the specified joint.
- **Parameters**:
  - `id` : (`int`) joint id 1-6.
  - `angle` : 0 - 180.

set_joint_max



- **Prototype**: `set_joint_max(id, angle)`
- **Interface description**: Sets the maximum angle of the specified joint.
- **Parameters**:
  - `id` : (`int`) joint id 1-6.
  - `angle` : 0 - 180.

get_joint_min_angle



- **Prototype**: `get_joint_min_angle()`
- **Interface description**: Gets the minimum movement angle of the specified joint
- **Parameters**: `joint_id` : (`int`)
- **Return**: angle value (`float`)

get_joint_max_angle



- **Prototype**: `get_joint_max_angle()`
- **Interface description**: Gets the maximum movement angle of the specified joint
- **Parameters**: `joint_id` : (`int`)
- **Return**: angle value (`float`)

Servo control

is_servo_enable



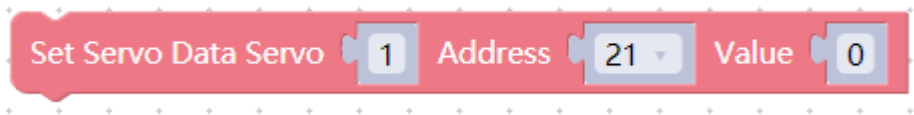
- **Prototype**: `is_servo_enable(servo_id)`
- **Interface description**: Determine whether all steering gears are connected
- **Parameters**: `servo_id` (`int`) 1 ~ 6
- **Return**:
- `0` : disable
  - `1` : enable
  - `-1` : error

is_all_servo_enable



- **Prototype**: `is_all_servo_enable()`
- **Interface description**: Determine whether the specified steering gear is connected
- **Return**:
  - `0` : disable
  - `1` : enable
  - `-1` : error

set_servo_data



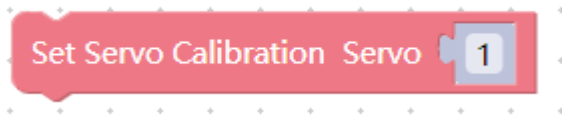
- **Prototype**: `set_servo_data(servo_no, data_id, value)`
- **Interface description**: Set the data parameters of the specified address of the steering gear.
- **Parameters**:
  - `servo_no` : Serial number of articulated steering gear, 1 - 6.
  - `data_id` : Data address.
  - `value` : 0 - 4096

get_servo_data



- **Prototype**: `get_servo_data(servo_no, data_id)`
- **Interface description**: Read the data parameter of the specified address of the steering gear.
- **Parameters**:
  - `servo_no` : Serial number of articulated steering gear, 1 - 6.
  - `data_id` : Data address.
- **Return**: `value` : 0 - 4096
  - `0` : disable
  - `1` : enable
  - `-1` : error

set_servo_calibration



- **Prototype**: `set_servo_calibration(servo_no)`
- **Interface description**: The current position of the calibration joint actuator is the angle zero point, and the corresponding potential value is 2048.
- **Parameters**:
  - `servo_no` : Serial number of articulated steering gear, 1 - 6.

release_servo



- **Prototype**: `release_servo(servo_id)`
- **Interface description**: Power off designated servo
- **Parameters**: `servo_id` : 1 ~ 6

focus_servo



- **Prototype**: `focus_servo(servo_id)`
- **Interface description**: Power on designated servo
- **Parameters**: `servo_id` : 1 ~ 6

Gripper

set_gripper_calibration



- **Prototype**: `set_gripper_calibration()`
- **Interface description**: Set the current position to zero, set current position value is `2048` .

set_gripper_state



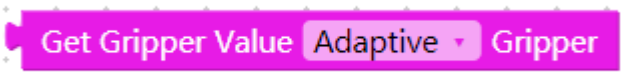
- **Prototype**: `set_gripper_state(flag, speed, mode)`
- **Interface description**: Set gripper switch state
- **Parameters**
  - `flag` (`int`): 0 - open, 1 - close
  - `speed` (`int`): 0 ~ 100
  - `mode` :gripper

set_gripper_value



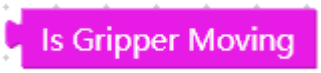
- **Prototype**: `set_gripper_value(value, speed, mode)`
- **Interface description**: Set gripper value
- **Parameters**
  - `value` (int): 0 ~ 100
  - `speed` (int): 0 ~ 100
  - `mode` :gripper

get_gripper_value



- **Prototype**: `get_gripper_value(mode)`
- **Interface description**: Get gripper value
- **Return**: gripper value (int)

is_gripper_moving



- **Prototype**: `is_gripper_moving()`
- **Interface description**: Judge whether the gripper is moving or not
- **Return**:
  - `0` : not moving
  - `1` : is moving
  - `-1` : error data

Init Eletric Gripper



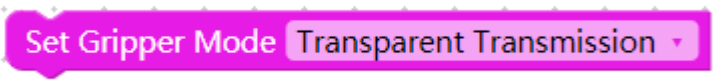
- **Prototype**: `init_eletric_gripper()`
- **Interface description**:Init Eletric Gripper

Set Eletric Gripper



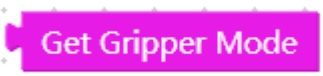
- **Prototype**: `set_eletric_gripper()`
- **Interface description**:Init Eletric Gripper
- **Parameters**
  - 选择打开或关闭

set_gripper_mode



- **Prototype**: `set_gripper_mode(status)`
- **Interface description**: Set gripper mode.
- **Parameters**
- `status` (`int`): 0 - transparent transmission. 1 - Port Mode.

get_gripper_mode



- **Prototype**: `get_gripper_mode()`

- **Interface description**: Get gripper mode.

- **Return**
  - `status` (`int`): 0 - transparent transmission. 1 - Port Mode.

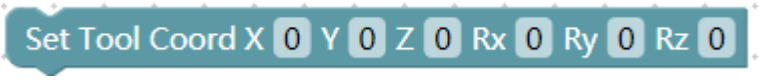
Coord Control

get_tool_reference



- **Prototype**: `get_tool_reference()`
- **Interface description**: Get tool coordinate system.
- **Return**: `list` [x, y, z, rx, ry, rz].

set_tool_reference



- **Prototype**: `set_tool_reference(coords)`
- **Interface description**: Set tool coordinate system.
- **Parameters**:
  - `coords` : (`list`) [x, y, z, rx, ry, rz].

get_world_reference



- **Prototype**: `get_world_reference()`
- **Interface description**: Get world coordinate system.
- **Return**: `list` [x, y, z, rx, ry, rz].

set_world_reference



- **Prototype**: `set_world_reference(coords)`
- **Interface description**: Set world coordinate system.
- **Parameters**:
  - `coords` : (`list`) [x, y, z, rx, ry, rz].

get_reference_frame



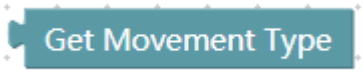
- **Prototype**: `get_reference_frame()`
- **Interface description**: Get base coordinate system.
- **Return**: 0 - base 1 - tool.

set_reference_frame



- **Prototype**: `set_reference_frame(rftype)`
- **Interface description**: Set base coordinate system.
- **Parameters**:
  - `rftype` : 0 - base 1 - tool.

get_movement_type



- **Prototype**: `get_movement_type()`
- **Interface description**: Get movement type.
- **Return**: 1 - movel, 0 - moveJ.

set_movement_type



- **Prototype**: `set_movement_type(move_type)`
- **Interface description**: Set movement type.
- **Parameters**:
  - `move_type` : 1 - movel, 0 - moveJ.

get_end_type



- **Prototype**: `get_end_type()`

- **Interface description**: Get end coordinate system.
- **Return**: 0 - flange, 1 - tool.

set_end_type



- **Prototype**: `set_end_type(end)`
- **Interface description**: Set end coordinate system.
- **Parameters**:
  - `end` : 0 - flange, 1 - tool.

MyCobot320

get_servo_speeds



- **Prototype**: `get_servo_speeds()`
- **Interface description**: Get joint velocity.
- **Return**: `list` Speed of each joint.

get_servo_currents



- **Prototype**: `get_servo_currents()`
- **Interface description**: Get joint current.
- **Return**: `list` Current of each joint.

get_servo_voltages



- **Prototype**: `get_servo_voltages()`
- **Interface description**: Get joint voltage.
- **Return**: `list` Voltage of each joint.

get_servo_status



- **Prototype**: `get_servo_status()`
- **Interface description**: Get the state of each joint.
- **Return**: `list` the state of each joint.

get_servo_temps



- **Prototype**: `get_servo_temps()`
- **Interface description**: Get the temperature of each joint.
- **Return**: `list` temperature of each joint.

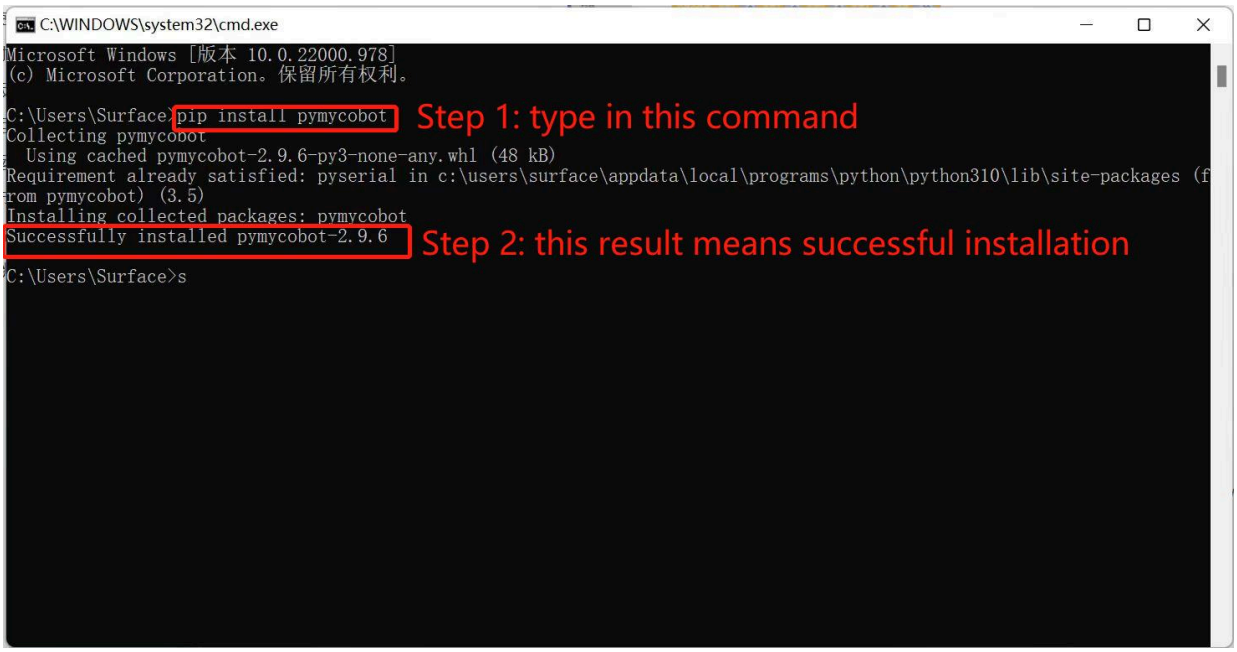
← Previous Page | Next Page →

Q&A

This chapter lists common problems with using myBlockly to control robotic arms for reference.

Q1: When running myBlockly, an error message `ModuleNotFoundError: No module named 'pymycobot'`

A: This is caused by not installing the pymycobot library when setting up the Python environment. To install the pymycobot library, you need to open the terminal (Win key + R key), enter: `pip install pymycobot --upgrade --user`, and click the Enter key to see Successfully installed pymycobot.



Q2: The robot arm is unresponsive because the `sleep` method module is not added

A: The program to operate the robot arm takes time to complete, so after an action, a `sleep' module needs to be connected to give the robot arm time to move before proceeding with the next movement (the time required depends on the situation, the machine The default setting of the arm is to run myBlockly with a minimum sleep time of no less than 0.5s), otherwise the robotic arm will not be able to achieve the ideal movement.

Q3: The `Run` button in the upper right corner cannot be clicked and is gray-green.

A: The new version of myBlockly adds 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) Whether there is a background program occupying the serial port of the robotic arm;

(2) Whether the toolbar under the red arrow on the right is closed. If it is open, it needs to be closed manually.

Q4: Why do I get a bunch of errors after running the program?

A: You need to confirm some information before running the program:

(1) Please confirm that your robot serial port number and baud rate are correct.

How to check the serial port number:

- On Windows systems, find the device manager and check the port. If the port (COM and LPT) displays USB-Enhanced-SERIAL CH9102, it is a CP34X chip. If the port (COM and LPT) shows Silicon Labs CP210x USB to UART Bridge, it is the CP210X chip. The port corresponding to these two names is the serial port of your robot arm.

- Open the terminal on Linux system, enter ls/dev/tty* and press Enter. What is displayed is the serial number of the robot arm. Among them AMA0 or USB0 etc. is the serial number of your robotic arm.
- Open the terminal on Mac system, enter cd/dev/ and press Enter, then run ls -al tty to find it, such as /dev/tty.usbserial-10.

(2) Please confirm that the baud rate is correct. `myCobot 320 M5` baud rate is `115200`.

(3) Please confirm 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 robotic arm.

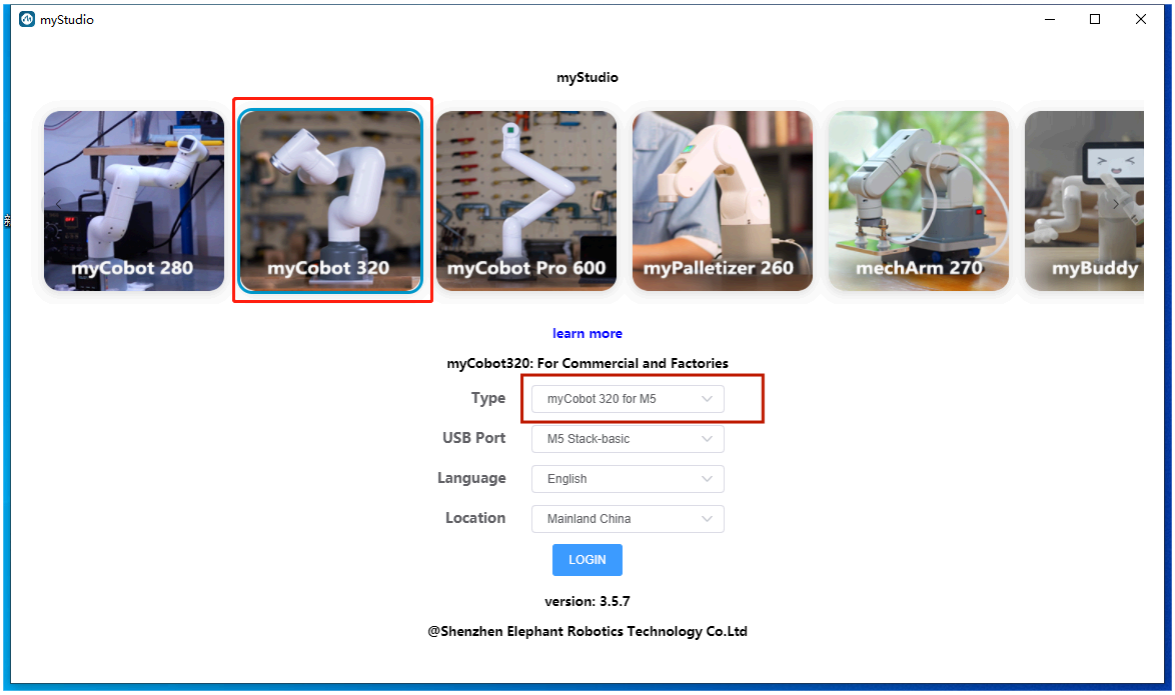
Q5: Error MyCobot.*int*() takes 2 positional arguments but 3 were given.

A: This error will appear in the old version of myBlockly because the versions of myBlockly and pymycobot are not compatible. Just update the versions of myBlockly and pymycobot driver libraries.

Q6: The result of running the program shows child process exited with code 1 A: This is not an error. All programs return the binary number 1 after running. It means everything has been successfully run.

← Previous Page | Next Section →

myStudio



1 myStudio design original intention

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

2 myStudio latest version and supported platforms

- Latest version: V3.5.7
- Available on:
  - Windows
  - Mac
  - Linux arm64

3 myStudio function

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

← Previous Section | Next Page →

Next Chapter →

myStudio environment setup

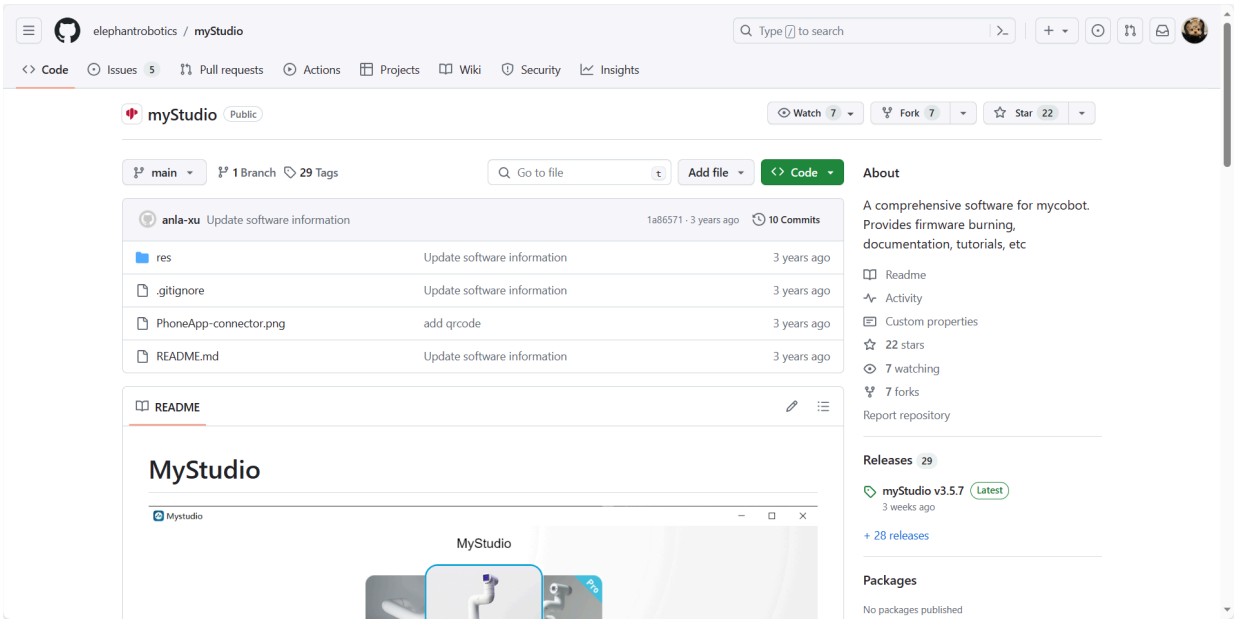
Download

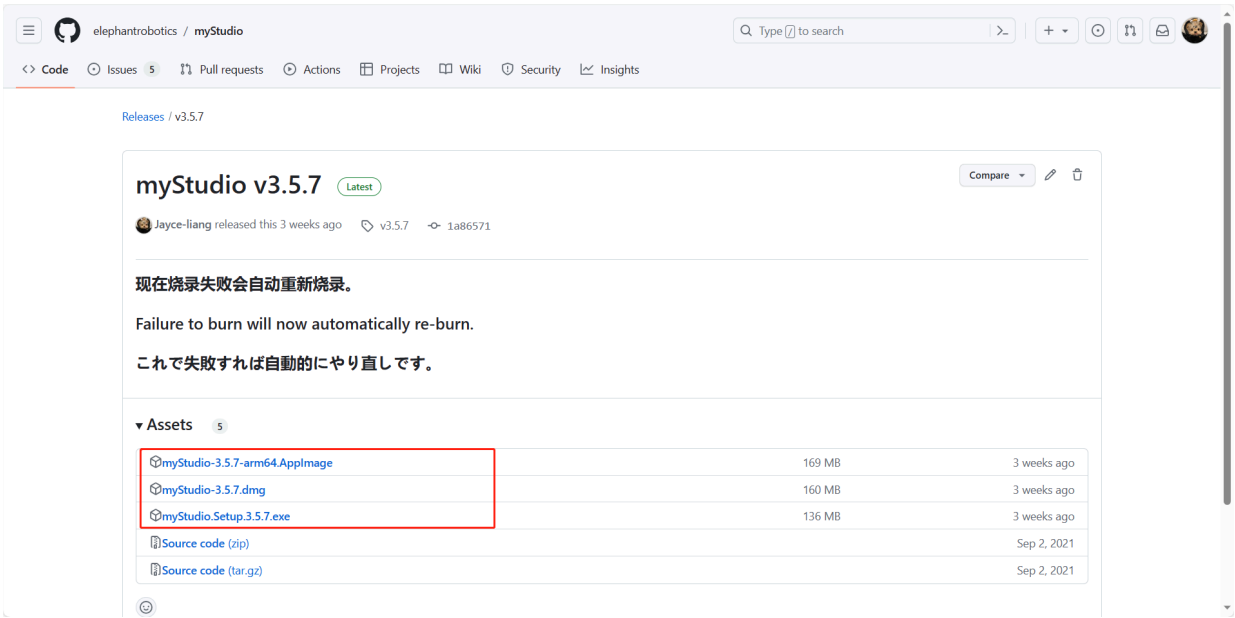
> Note: The installation path when installing myStudio cannot have any spaces.

download link:

1. GitHub

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



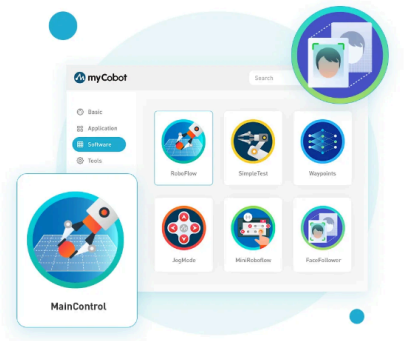


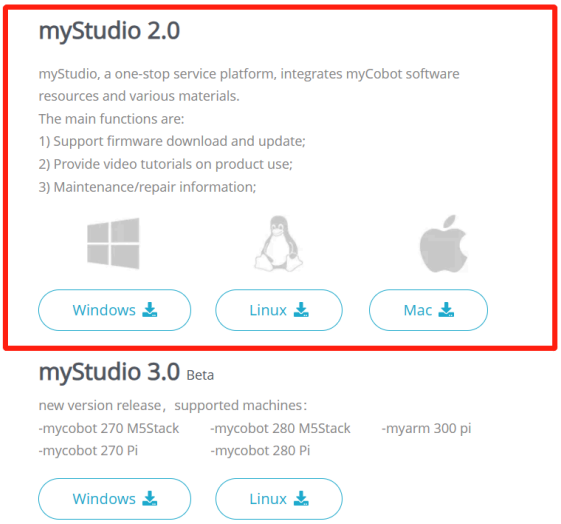
- Different suffixes represent different systems, please download the corresponding version:
  - *.AppImage - Linux system
  - *.dmg - Mac system

- *.exe - Window system

2. Official website address

You can choose to download `myStudio 2.0` according to your computer system.





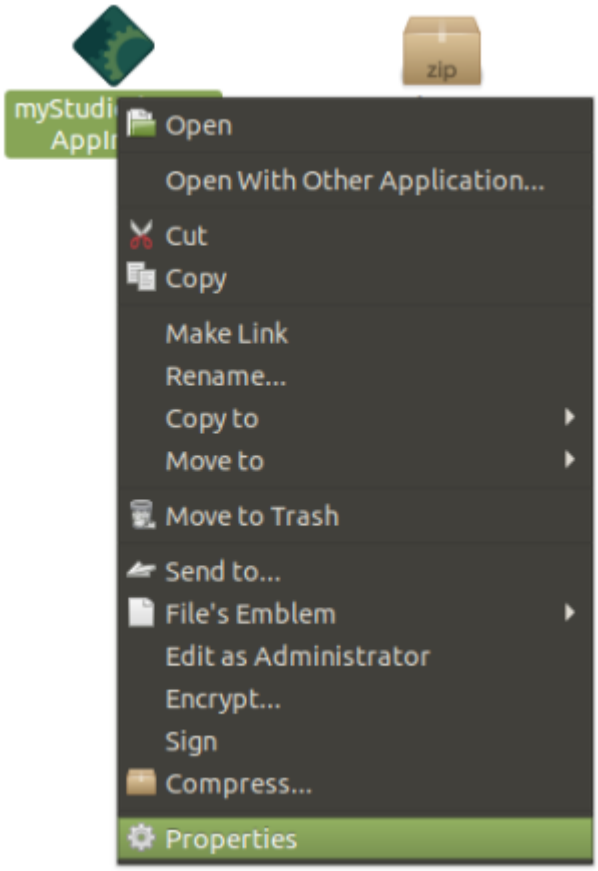
Install

For Linux install myStudio

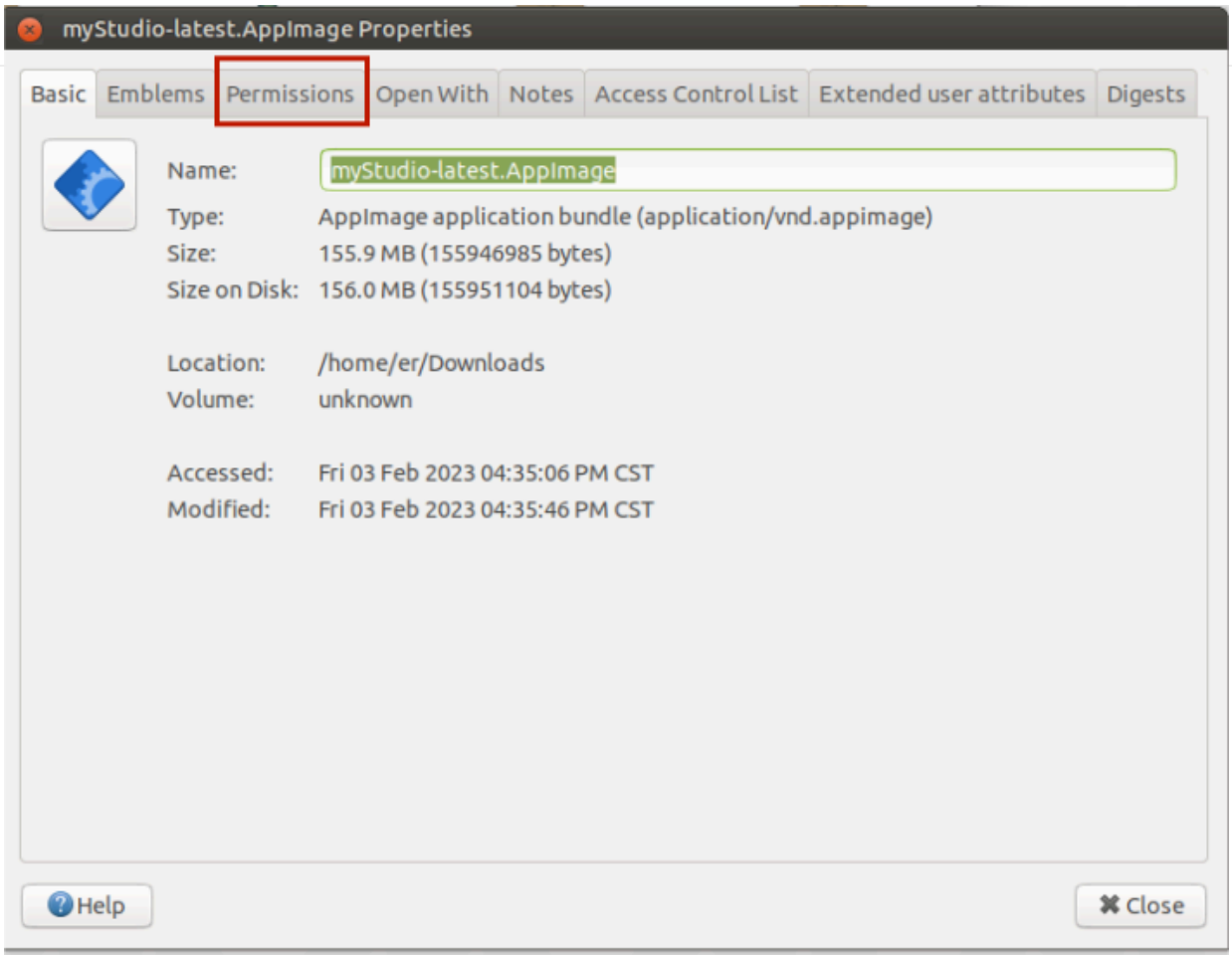
Download the Linux version of myStudio from the official website to get an installation package as shown below



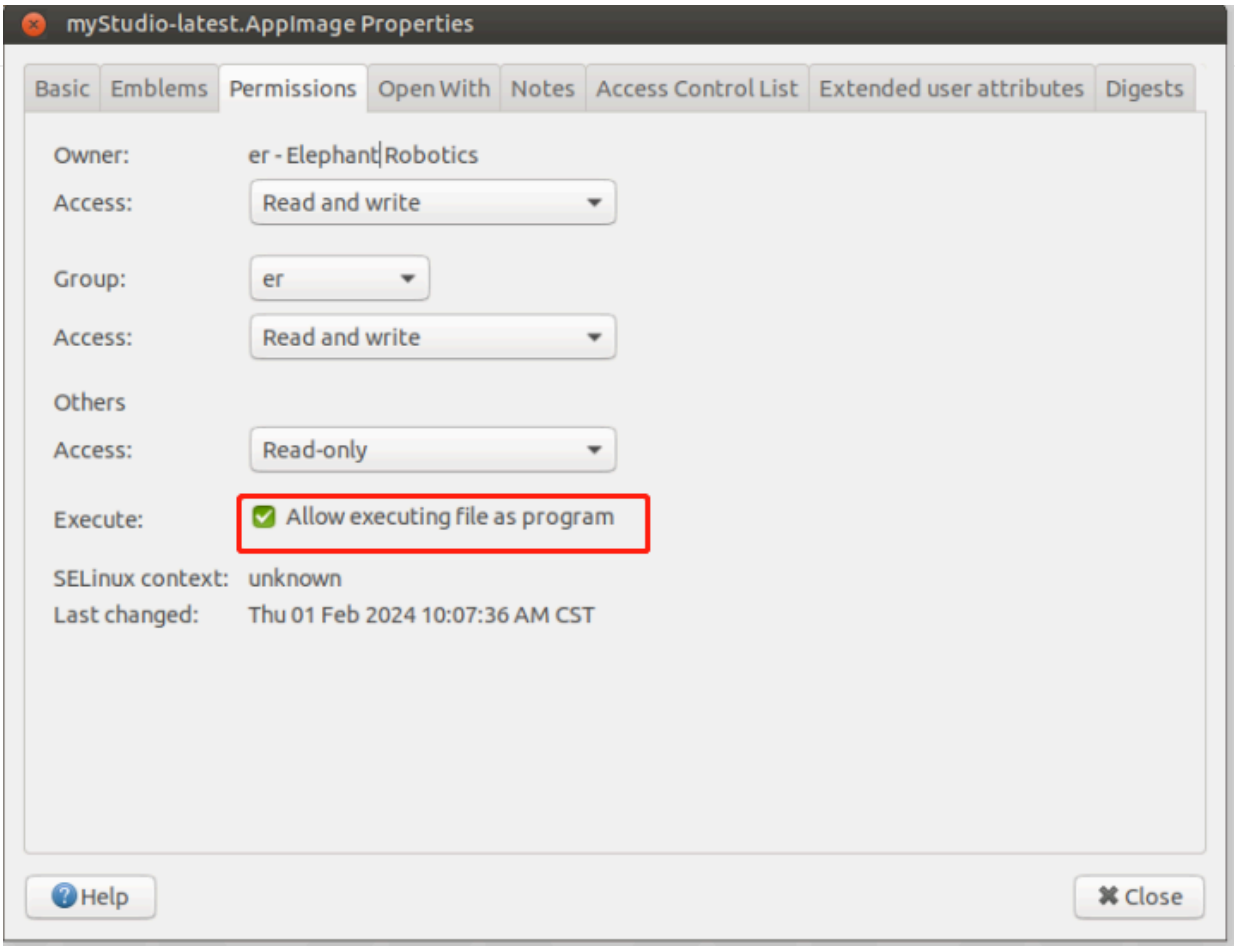
Select `myStudio-latest.AppImage` with the right mouse button to open it, click `Properties` to open it



Click to enter `Permissions`



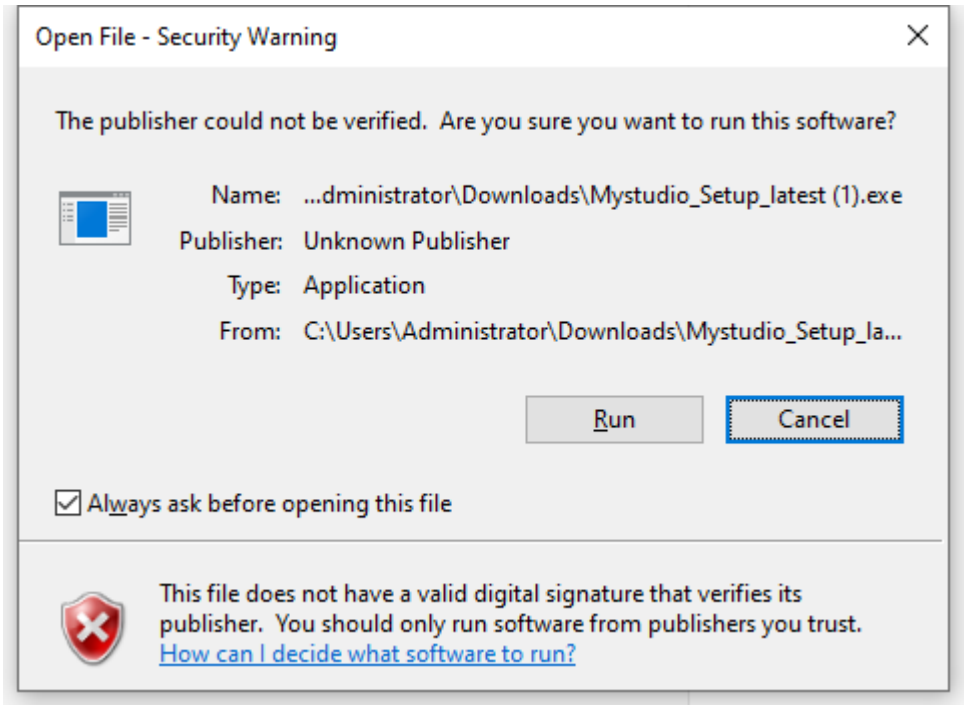
In the `Permissions` page, check `Allow executing file as program`, and then click the `Close` button to close the pop-up window



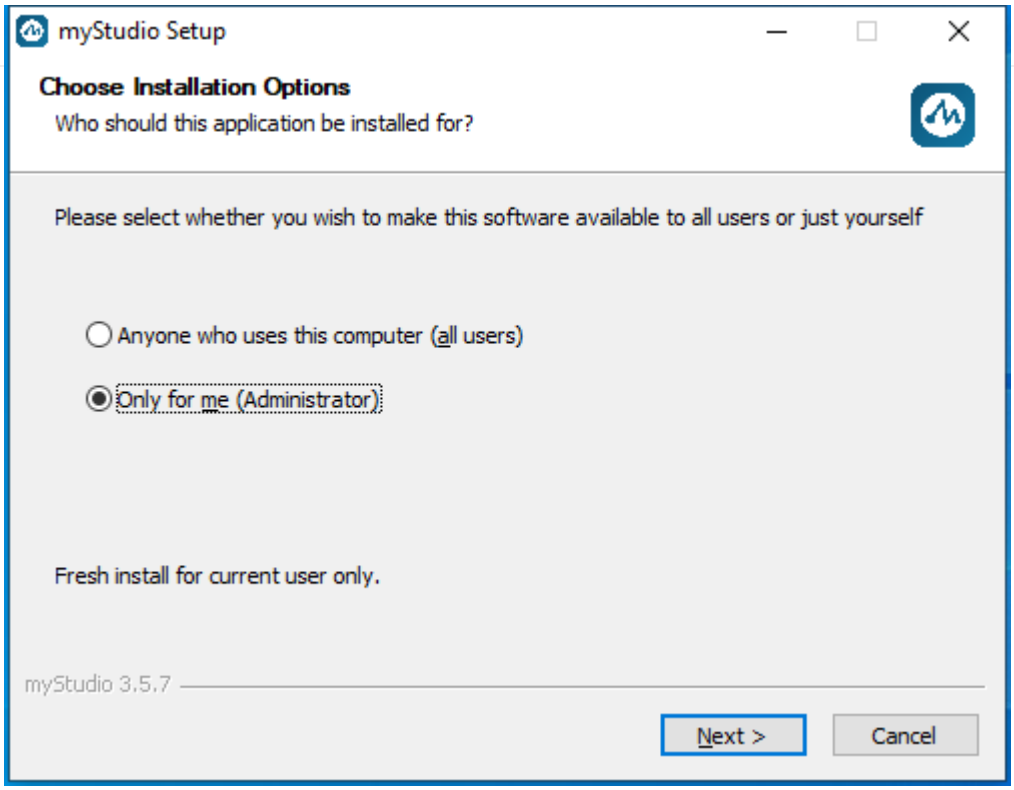
After closing the pop-up window, double-click the installation package `myStudio-latest.AppImage` to open myStudio

For Windows install myStudio

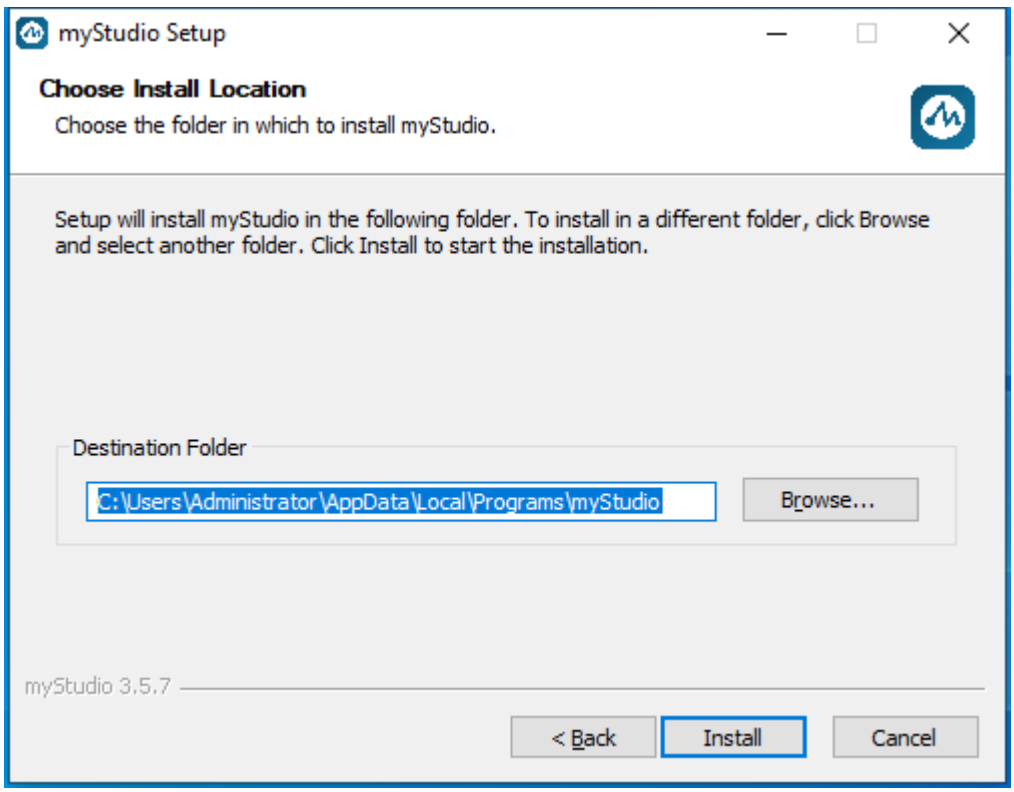
Double-click to open the file named `Mystudio_Setup_latest.exe` ,and click to `Run`



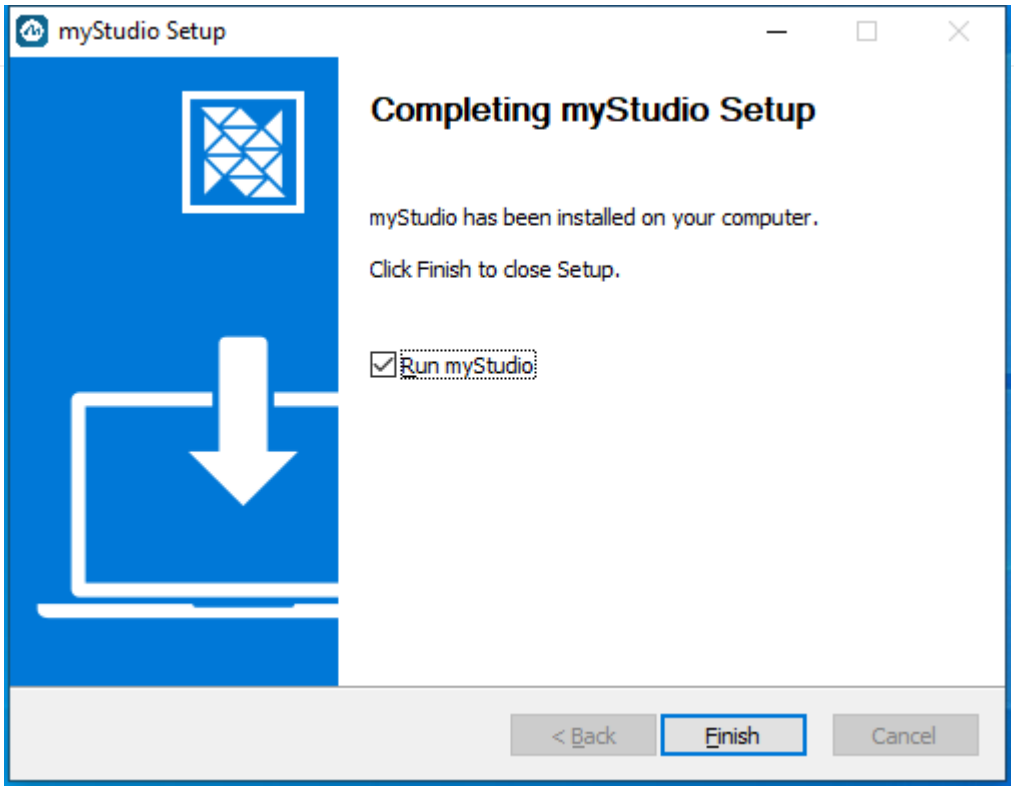
click `Next>`

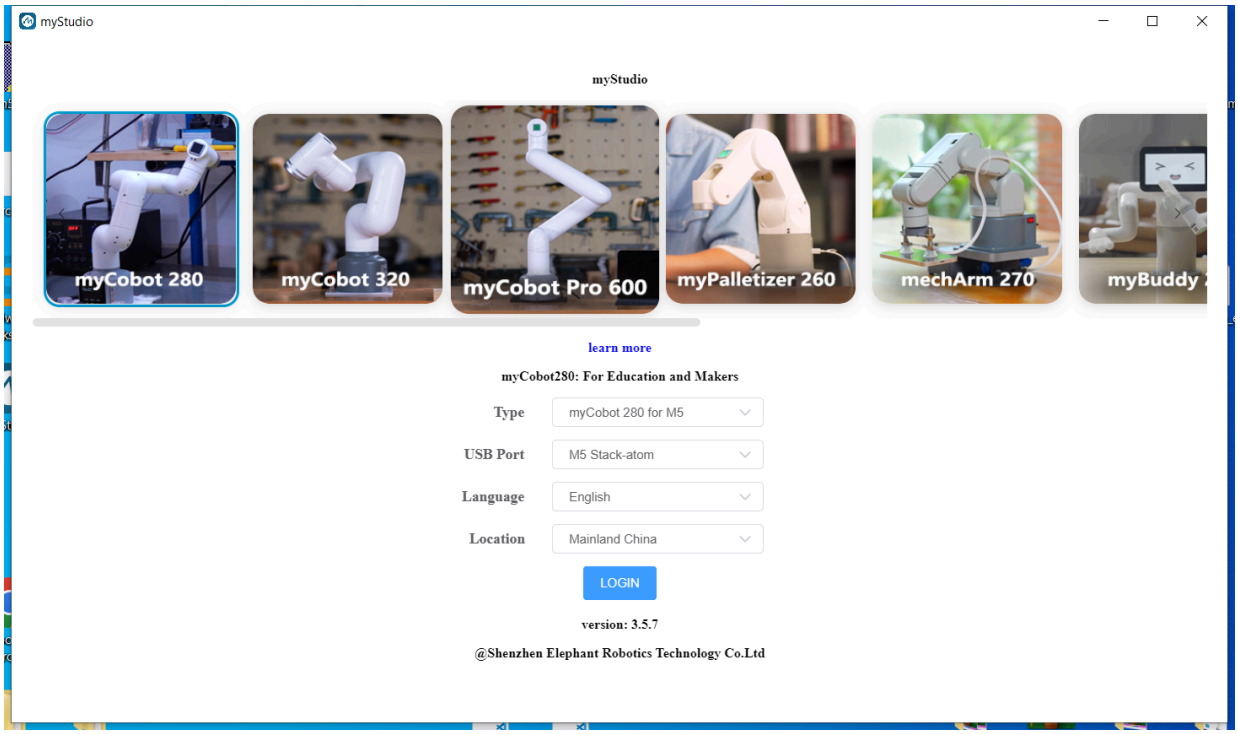


After clicking `Install`, wait for myStudio installation to complete



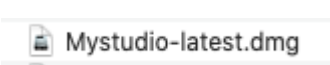
The installation is complete, click the `Finish` button to open and run myStudio





For MacOS install myStudio

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



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

Uninstall

For Linux systems uninstall myStudio

Just delete the installation package directly

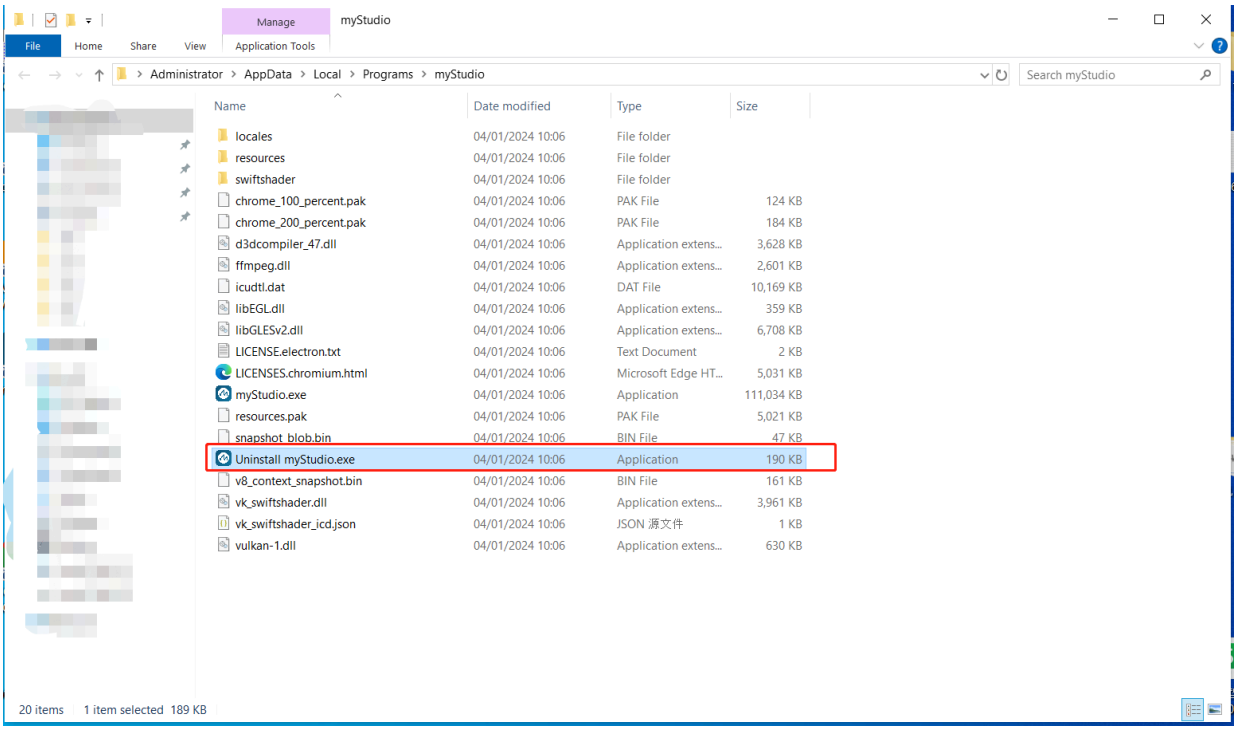
> The default name of the installation package is `myStudio-latest.AppImage`

Uninstall myStudio for Mac

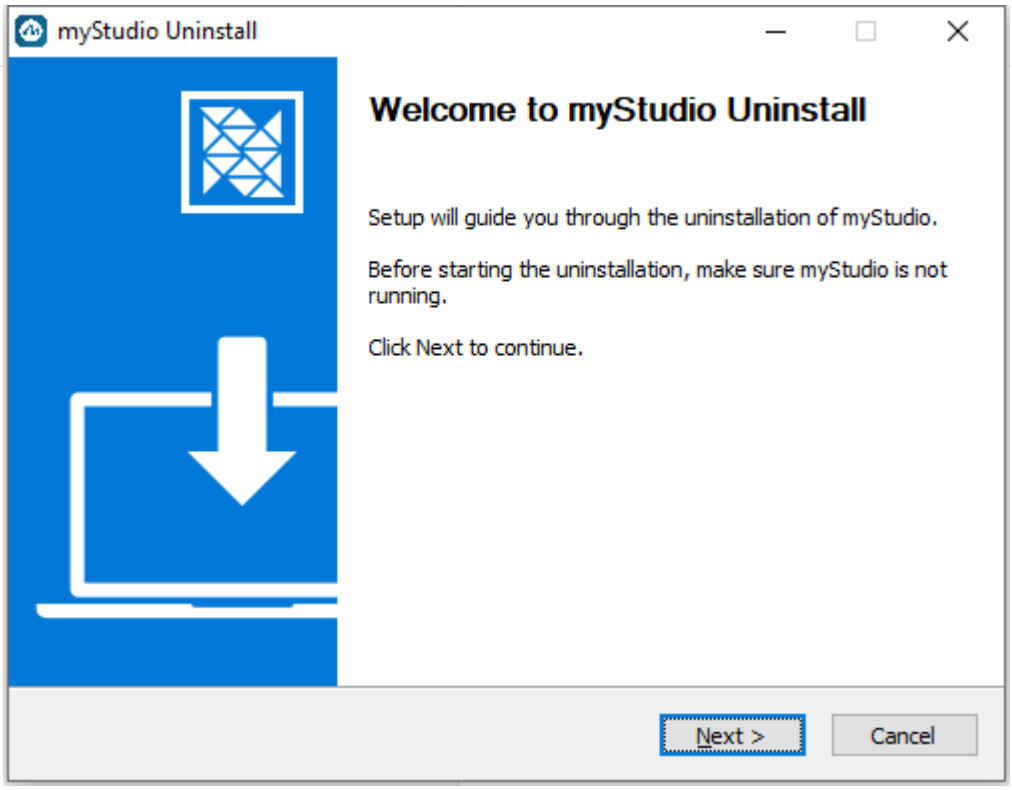
Just move myStudio to the Trash in the app

Uninstall myStudio for Windows systems

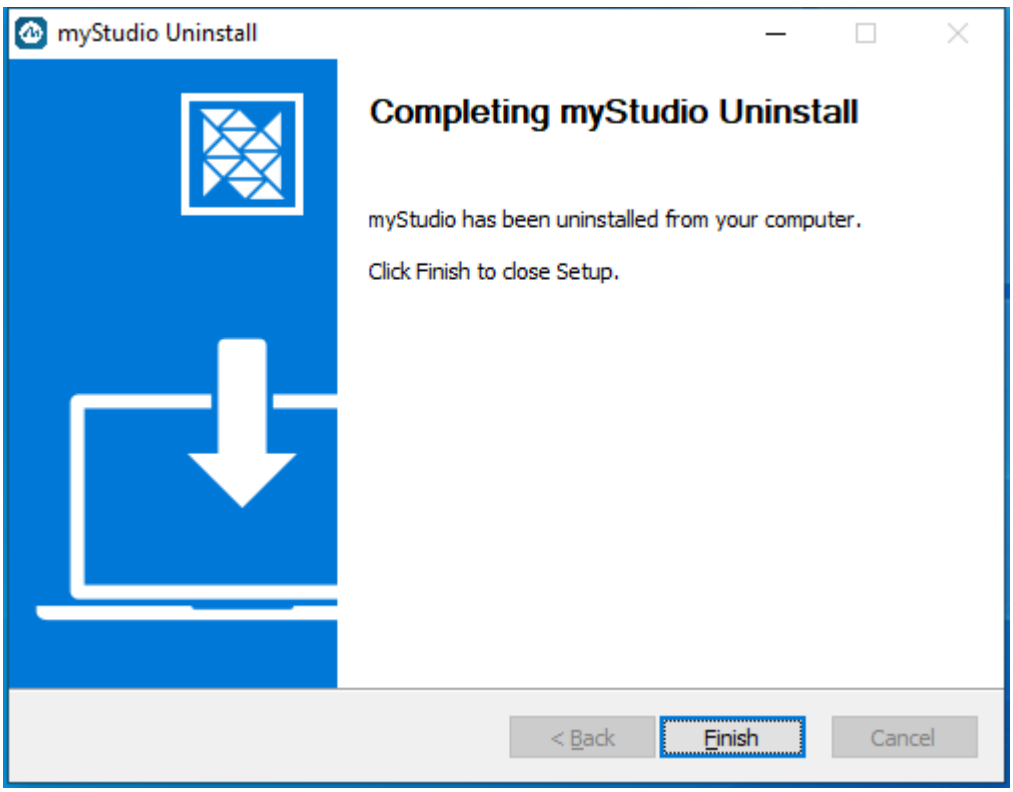
Enter the file directory of myStudio and click to run `Uninstall myStudio.exe`



Click `Next>`

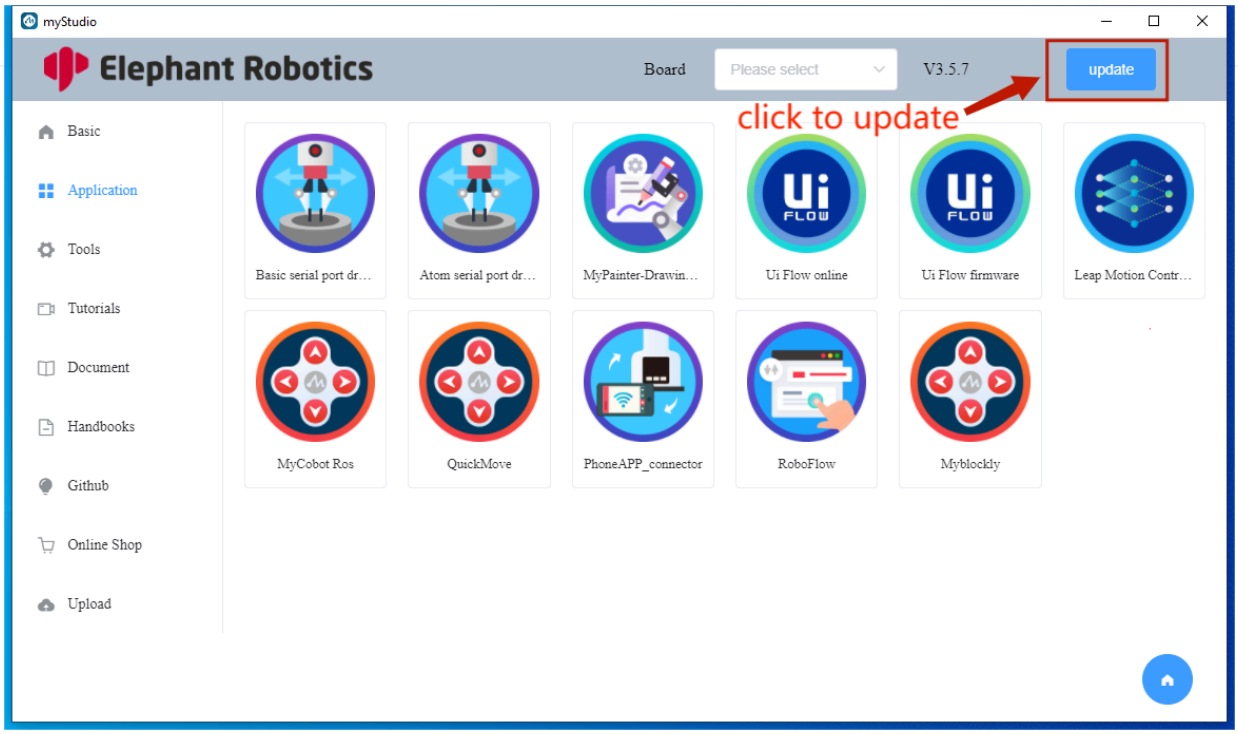


myblockly has been uninstalled, click `Finish` to exit



Update

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



← Previous Page | Next Page →

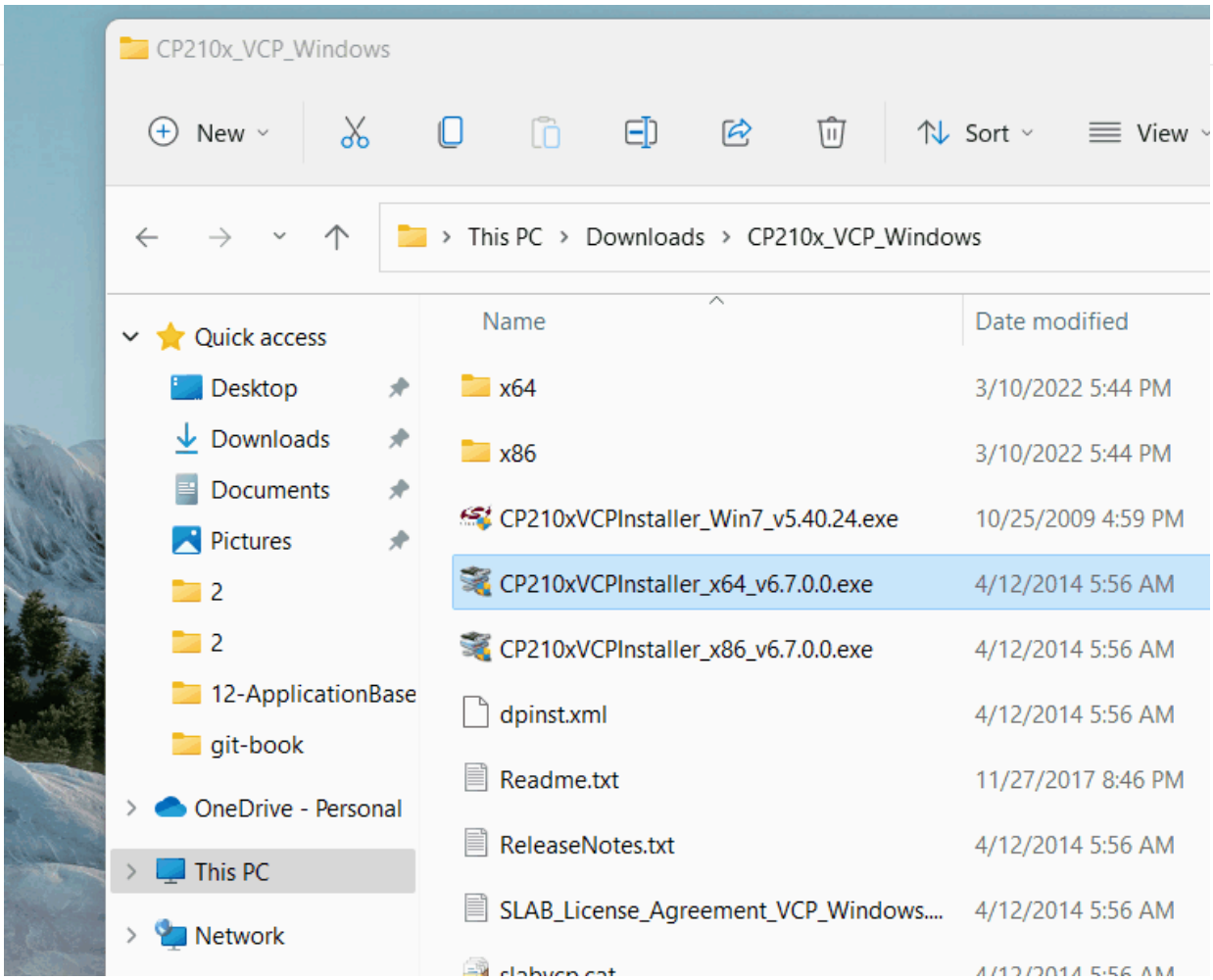
Install driver

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

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

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

- Download the **M5Stack-basic** serial port driver at the bottom
  - **CP210X**
    - **Windows10**
    - **MacOS**
    - **Linux**
    - **CP34X**
      - **Windows10**
      - **MacOS**
- Download the **Atom** serial port driver at the end
  - **Windows10**

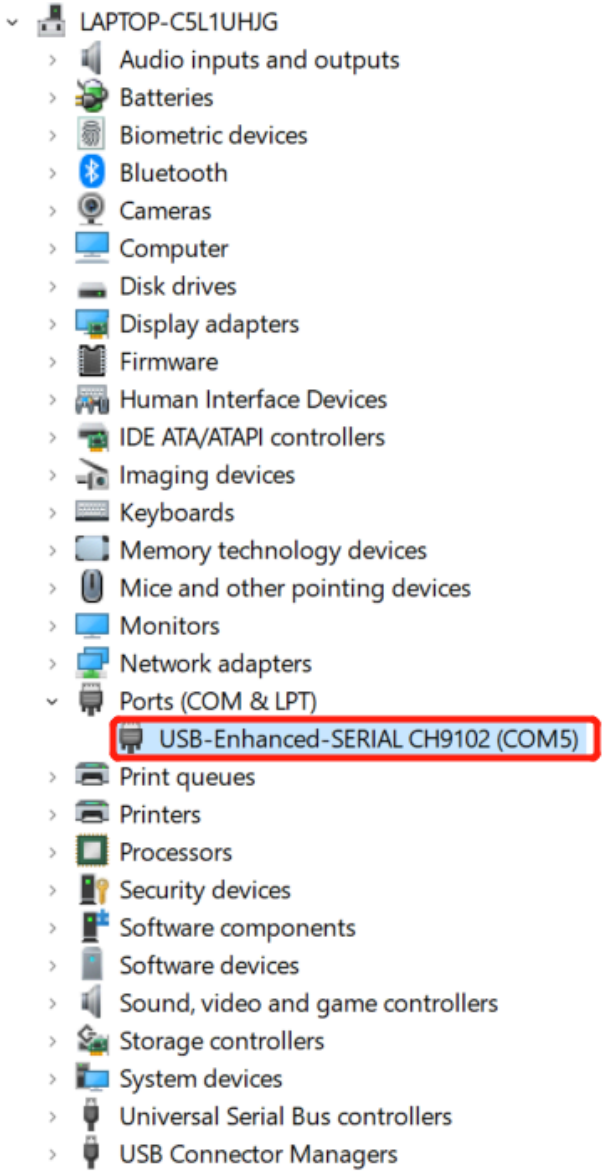


1.3 How to distinguish CP210X and CP34X chips

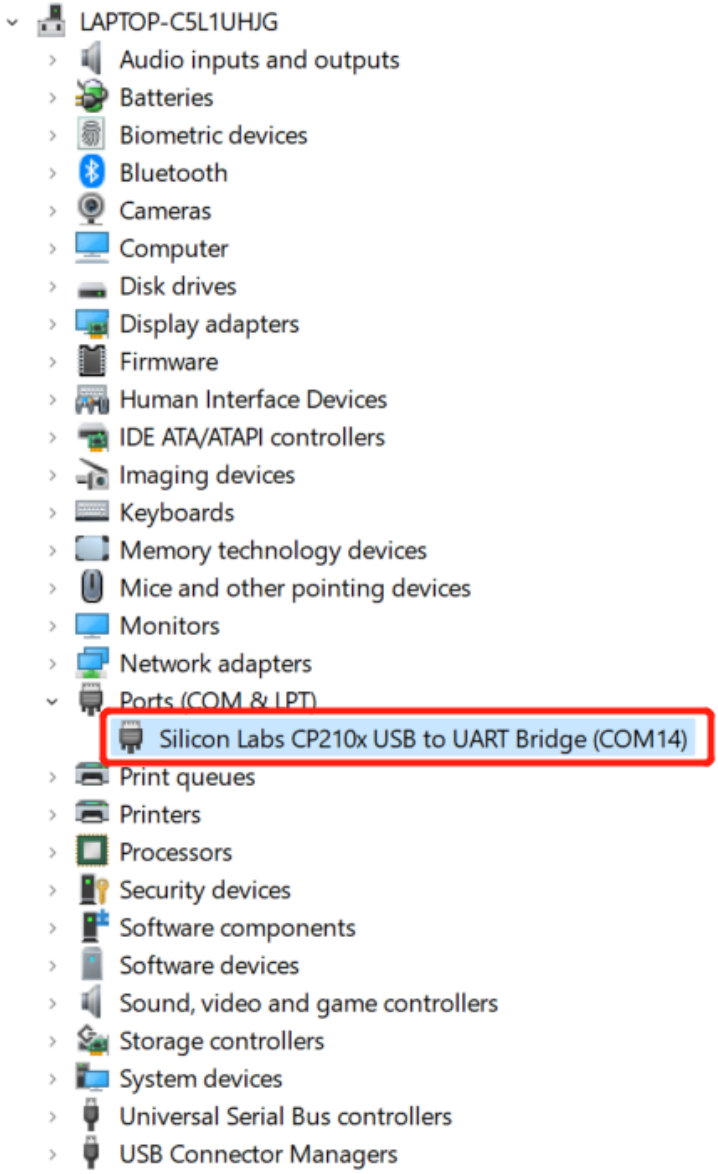
- As shown in the figure below, open **Device Manager** and view **Ports (COM and LPT)**



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



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



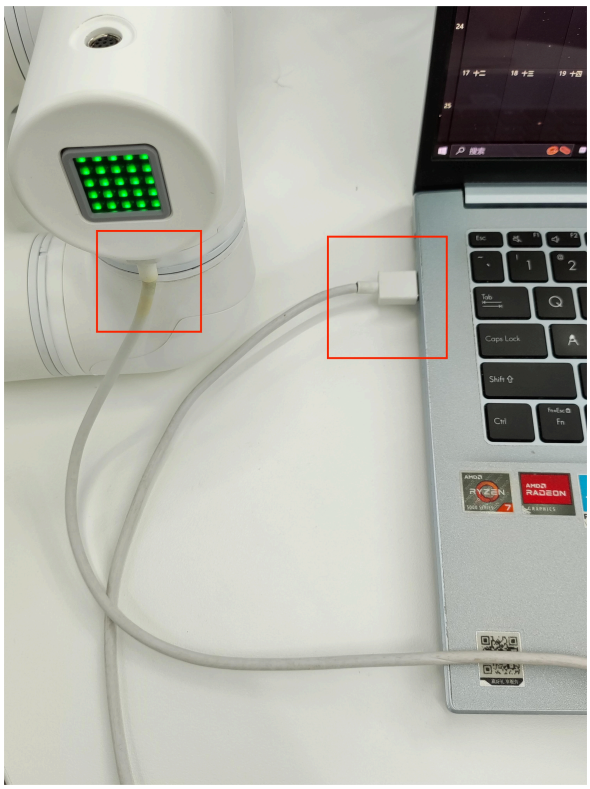
← Previous Page | Next Page →

Burn and update firmware

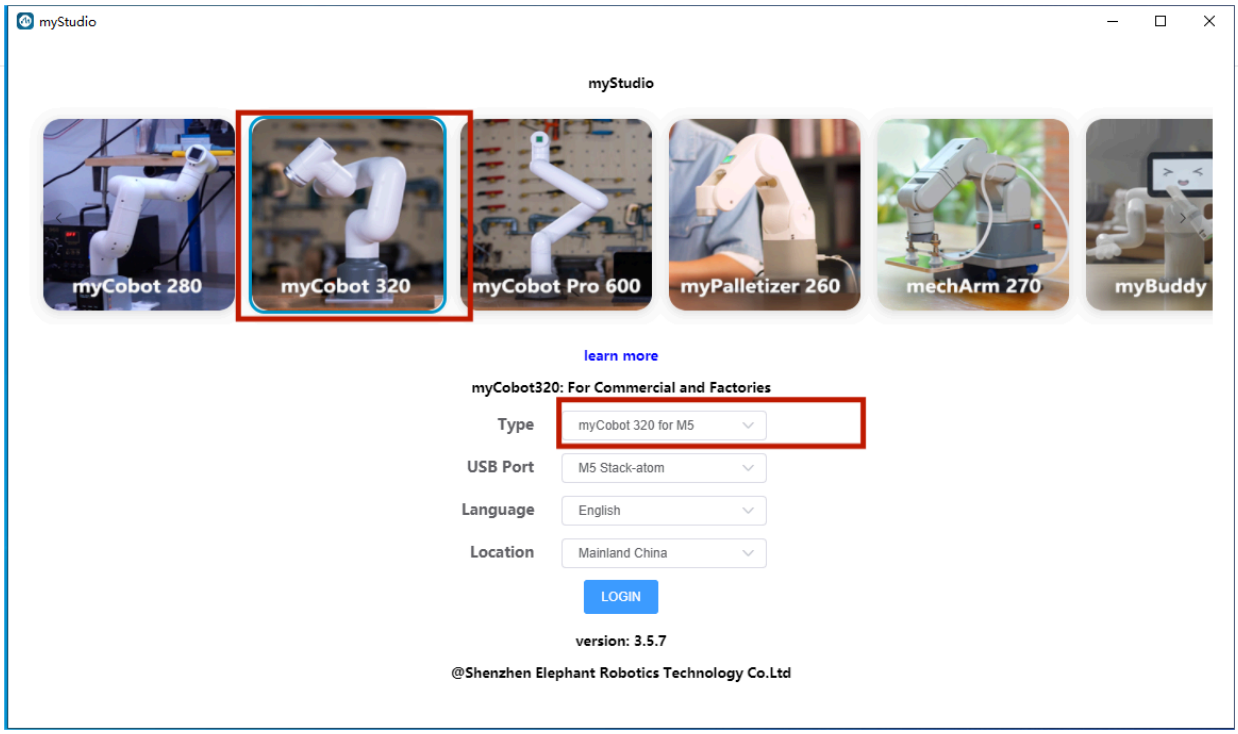
> Customers get the machine does not need to be burned in non-essential cases, the factory has already been burned, only a few cases need to be burned.
>
> For example: minirobot corresponding to the need to burn the scene of the LCD screen does not light up, burn the Arduino program needs to be switched to python program control, Atom firmware corresponding to the need to burn the scene of the joints can not be locked, Atom lights do not light up and so on.

Burn Atom firmware

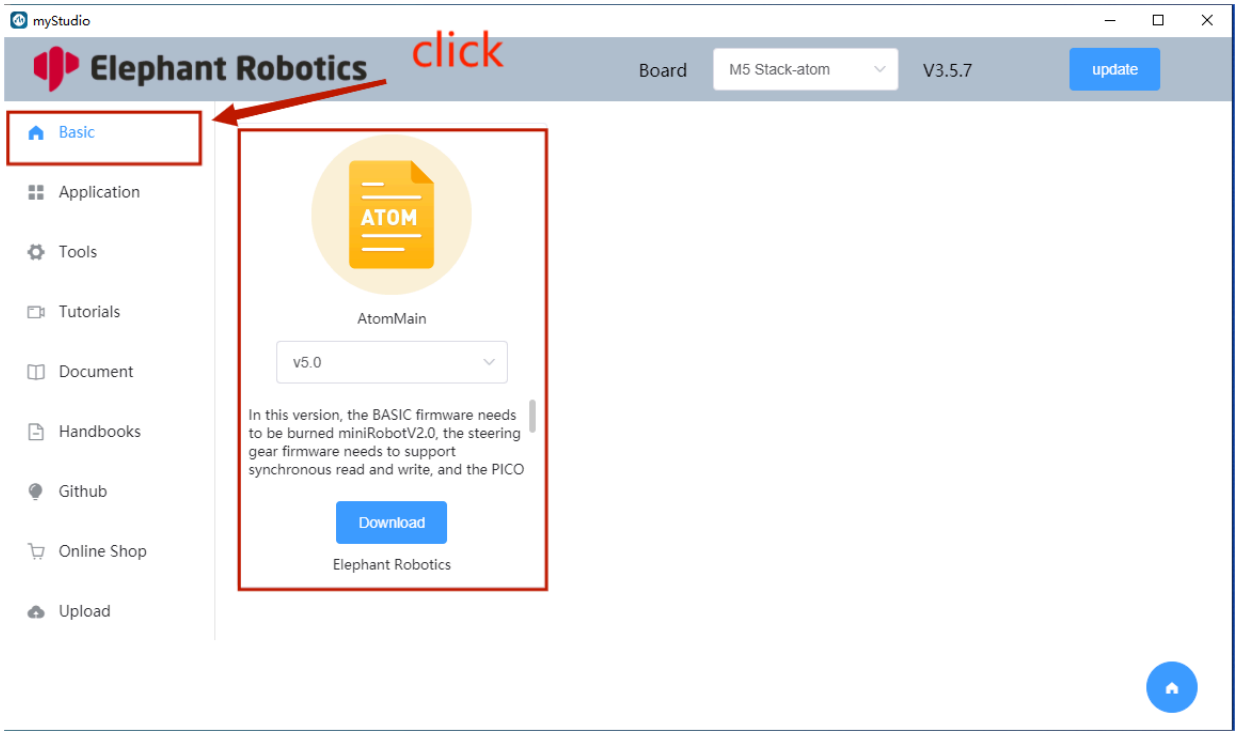
Step 1: Connect to PC. Connect the Atom at the end with USB.



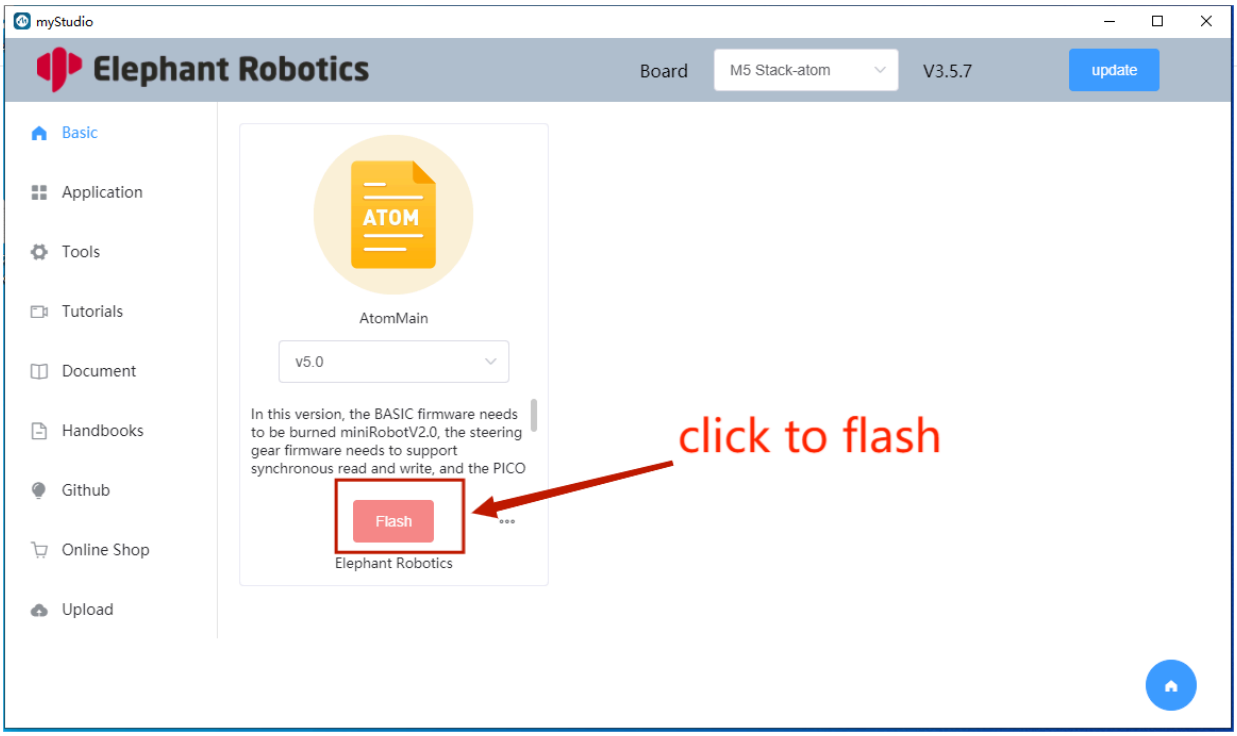
Step 2 :Select Robot `myCobot 320 M5`



Step 2: Select `ATOM` in the `Board` column, and the Atom firmware will appear in the sidebar `Basic`. There is only one firmware for Atom, just click to `Download` it in.

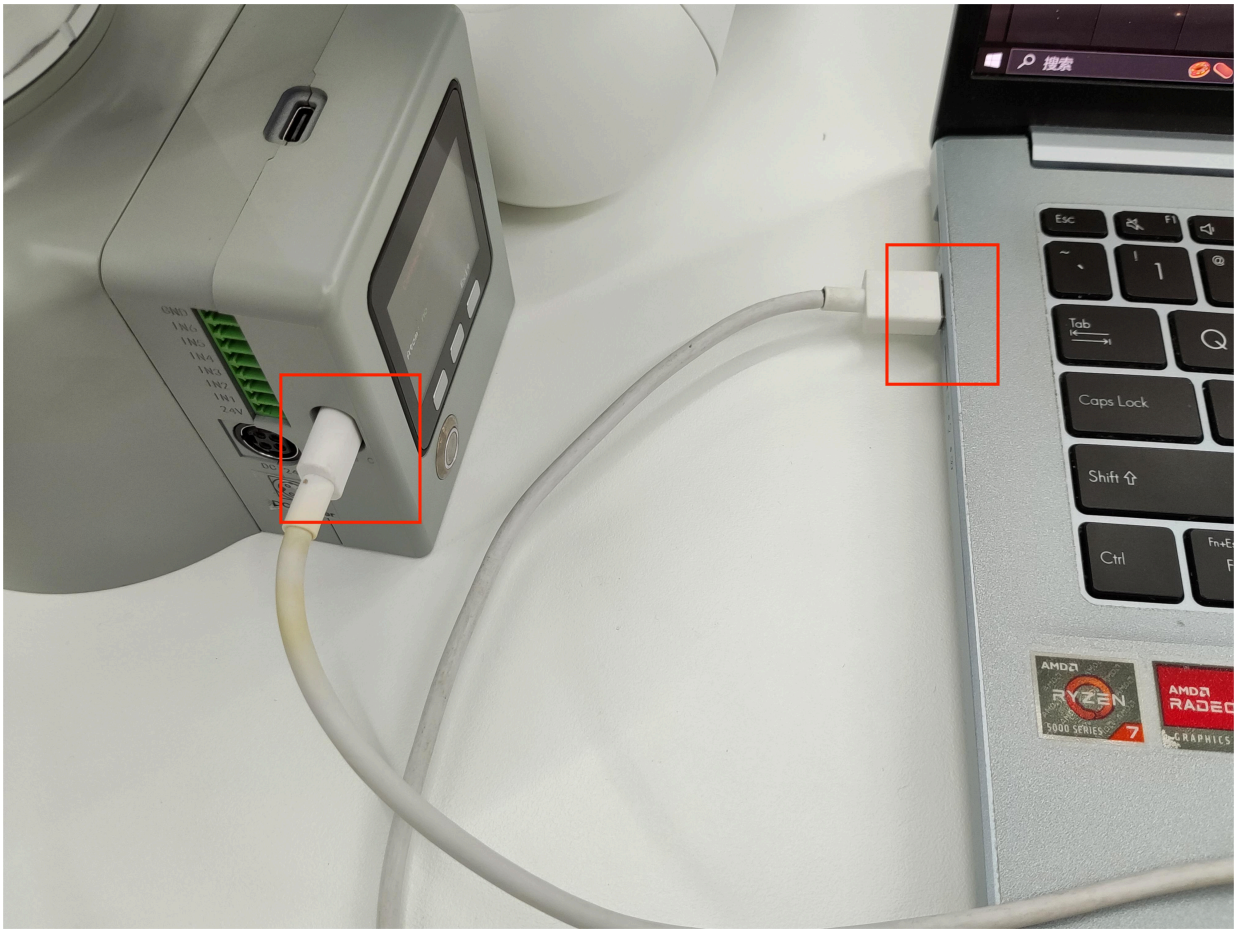


Step 3: After download finish, click the `Flash` button to start burn.

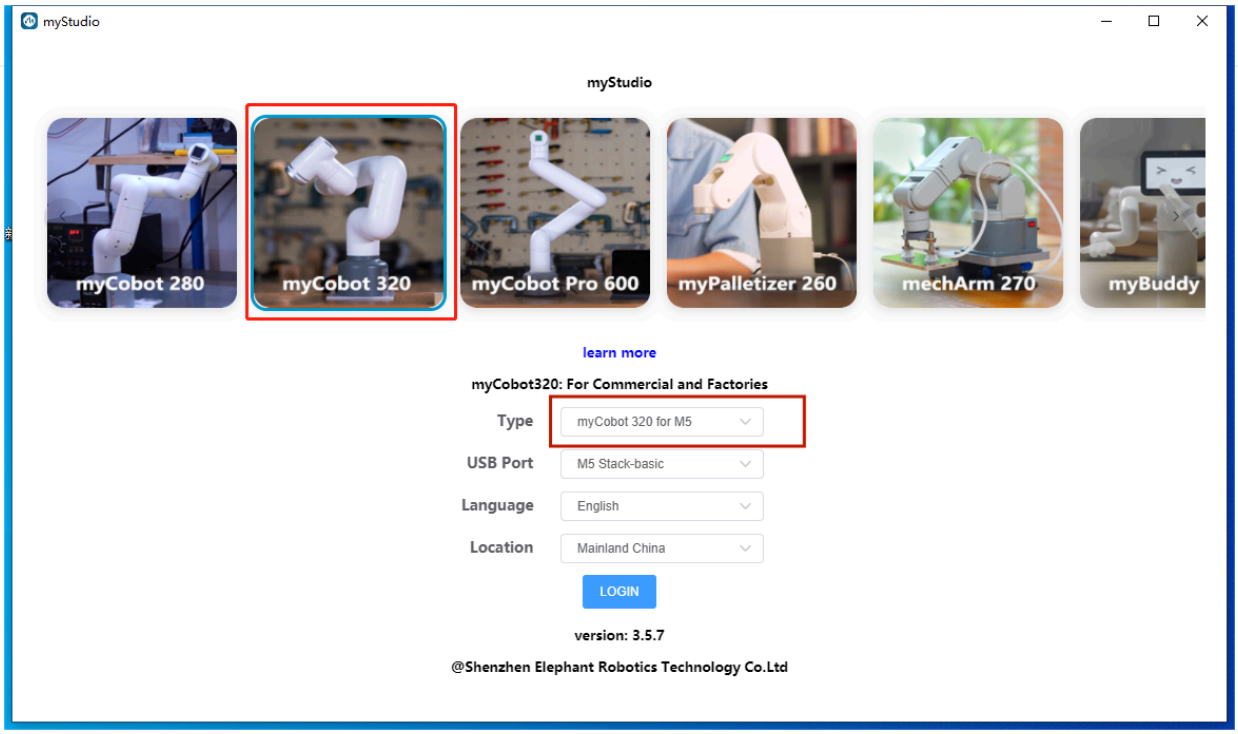


Burning M5Stack-Basic firmware——minirobot

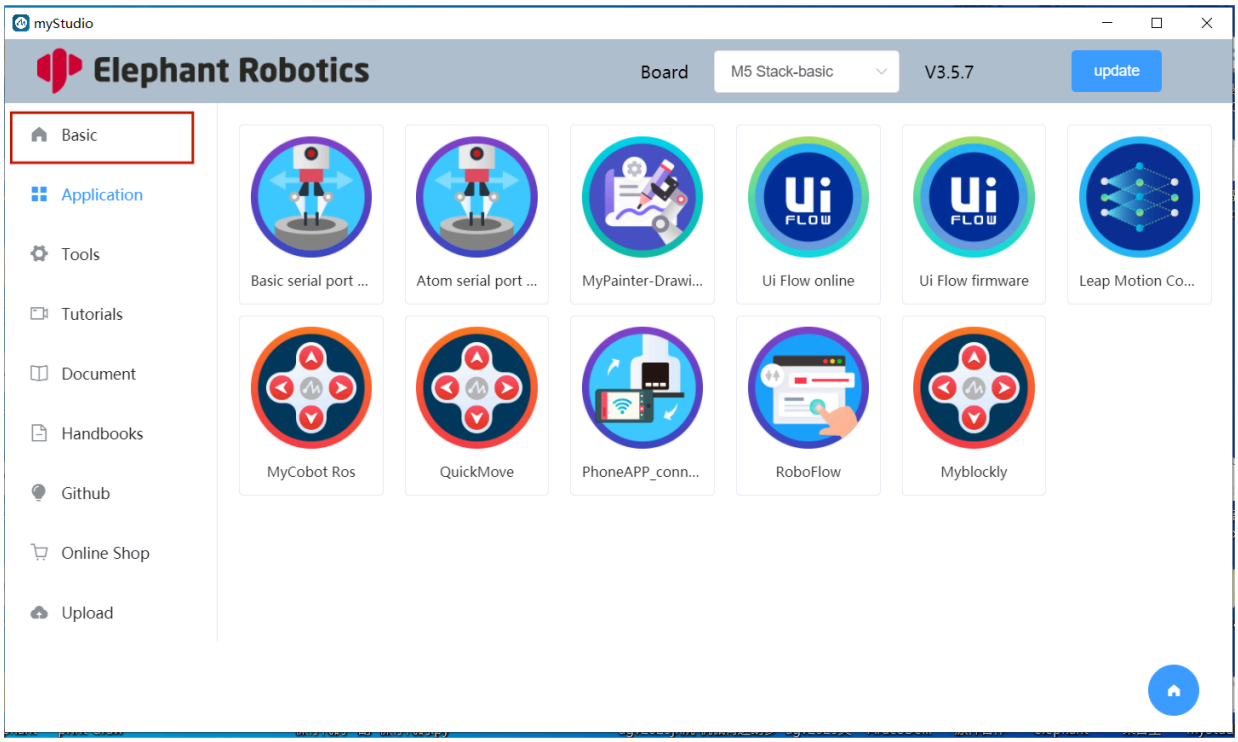
Step 1: Connect M5Stack-Basic to PC.



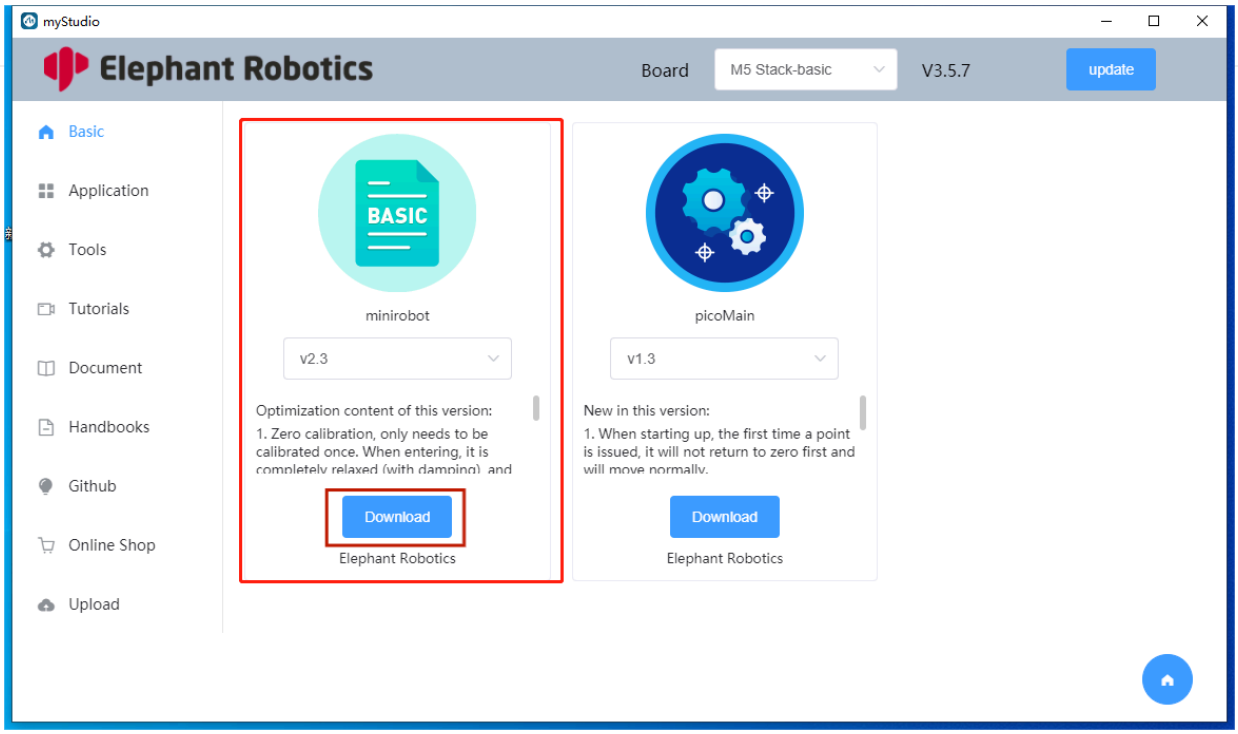
Step 2 :Select Robot `myCobot 320 for M5` ,and click `LOGIN` button.



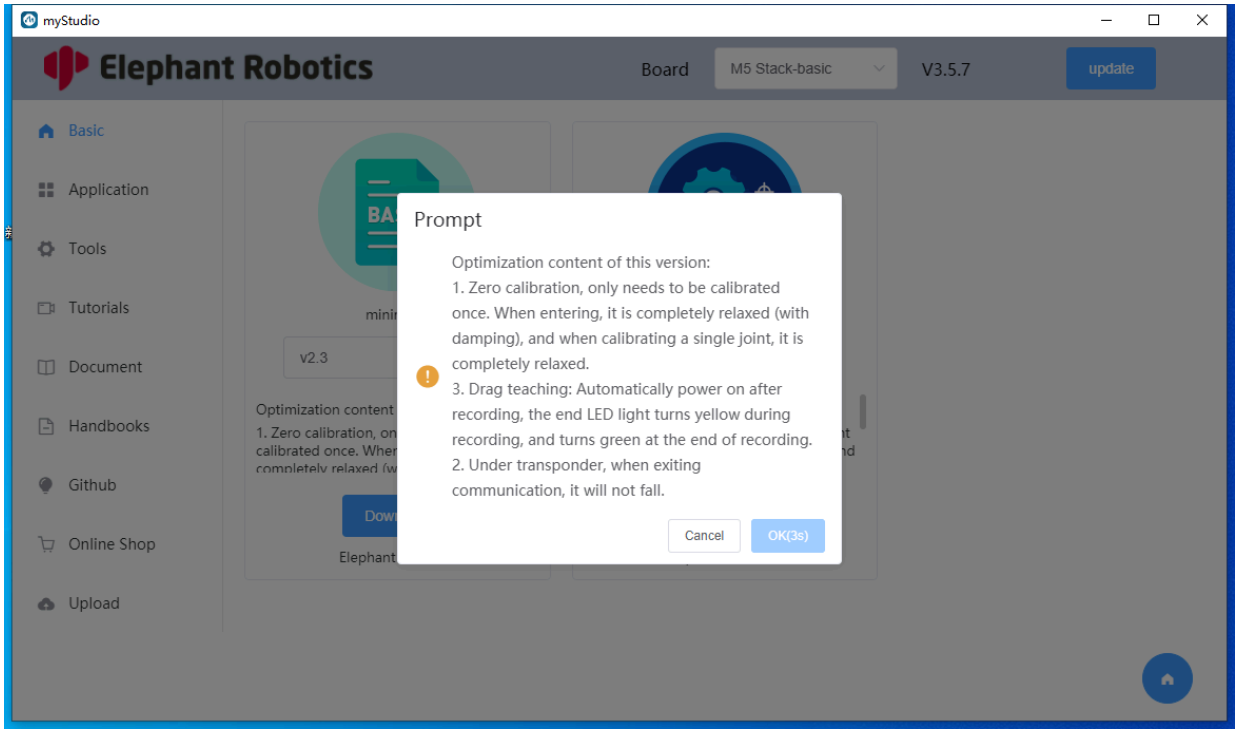
Step 3: After logging in, click `Basic`



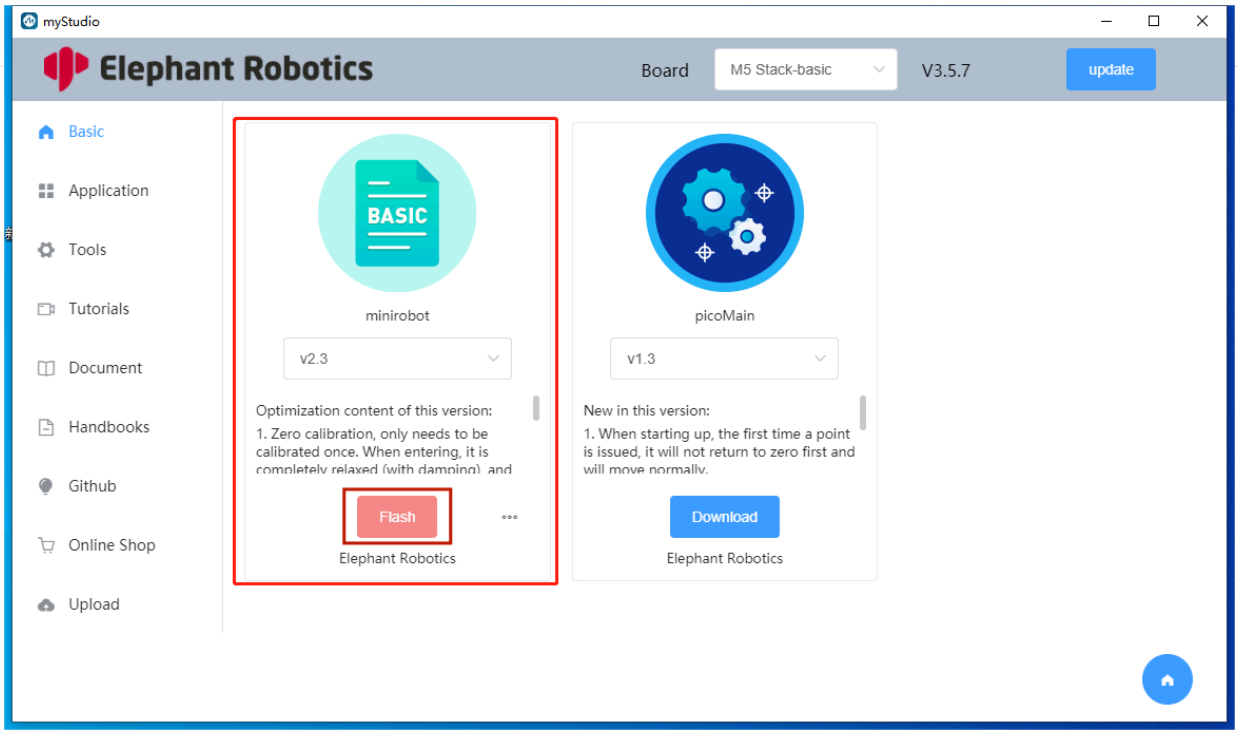
Step 4: click `Download`



Step 5: click `Ok`



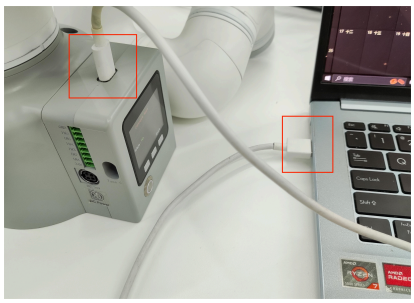
Step 5: click `Flash`



Burn pico firmware——picoMain

Step 1: Connect M5Stack-Basic to PC.





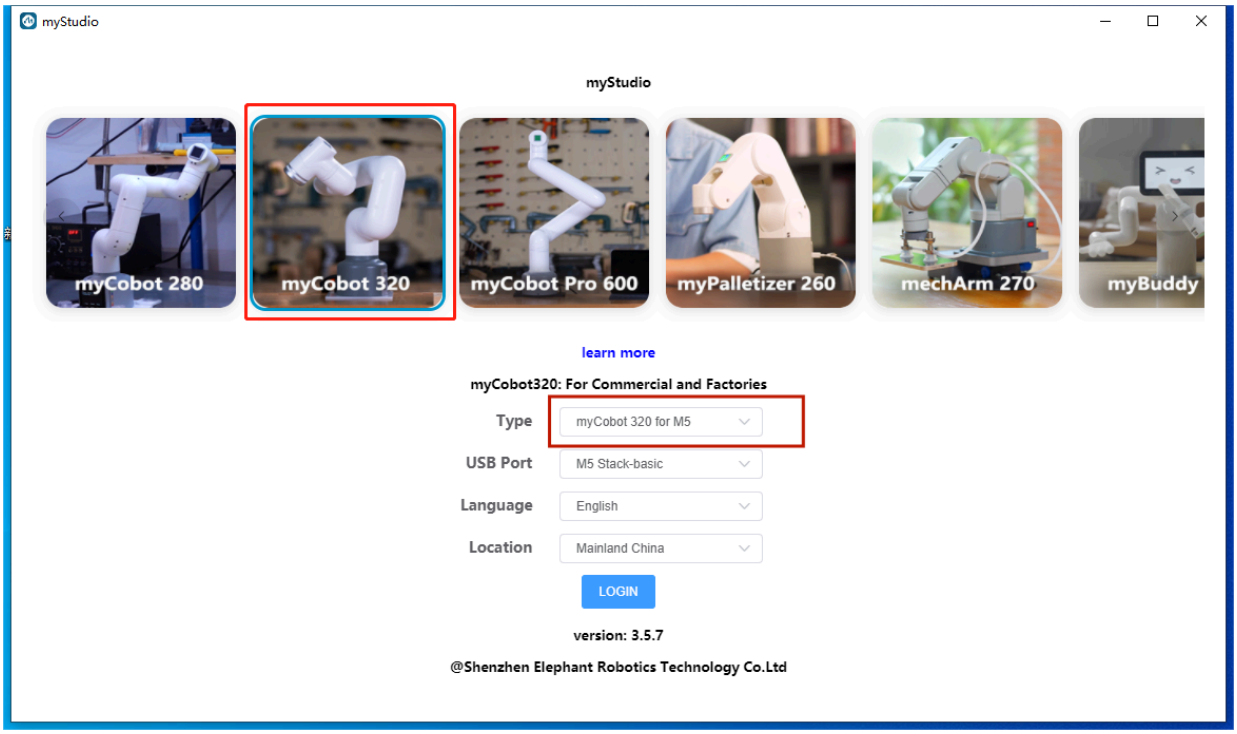
Step 2: Select Flash mode in the robot arm base (you can choose to press the `UP` key once or the `Down` key four times), and then the base will sleep with a black screen for about 30 seconds.



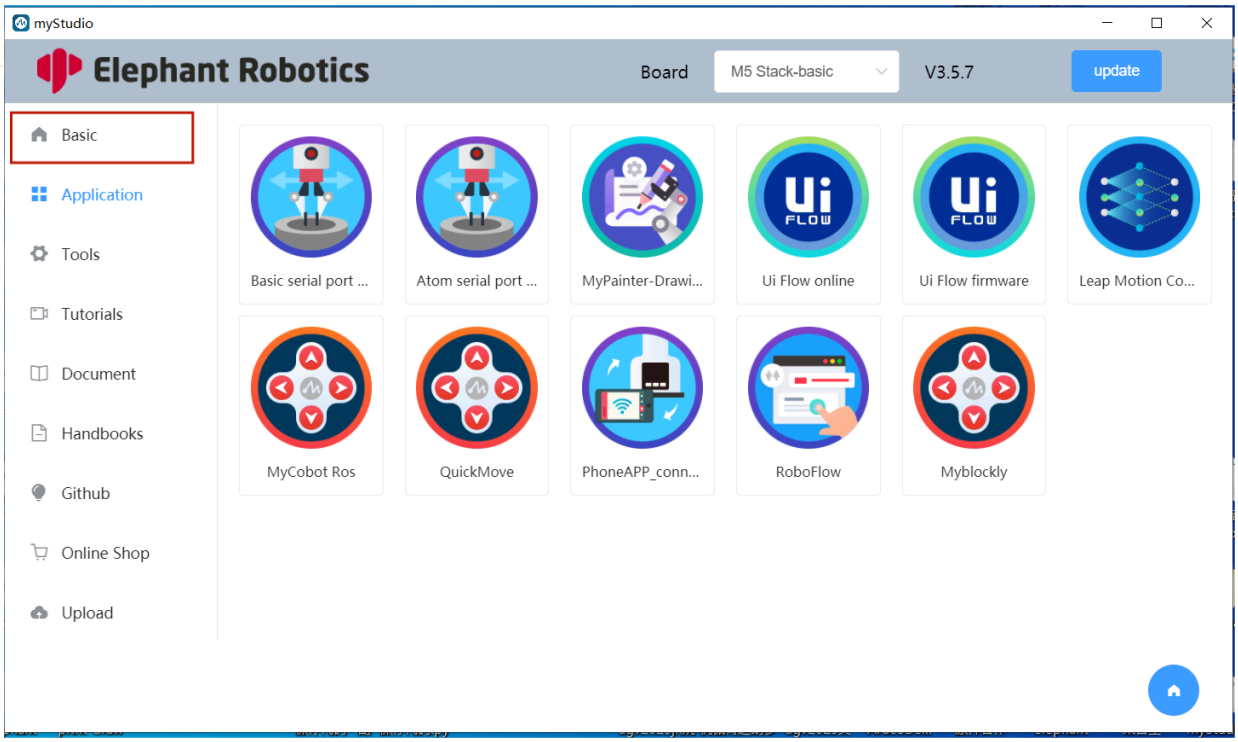
Step 3: Enter the myStudio firmware burning page and select `picoMain` firmware burning.

> **Note**: To burn the pico firmware, you need to select Flash mode, that is, to burn in the sleep state. If the sleep time is exceeded, the base display will restart and then burning the pico firmware will fail. Step 2: Step 2 :Select Robot `myCobot 320 for M5` ,and click `LOGIN` button.

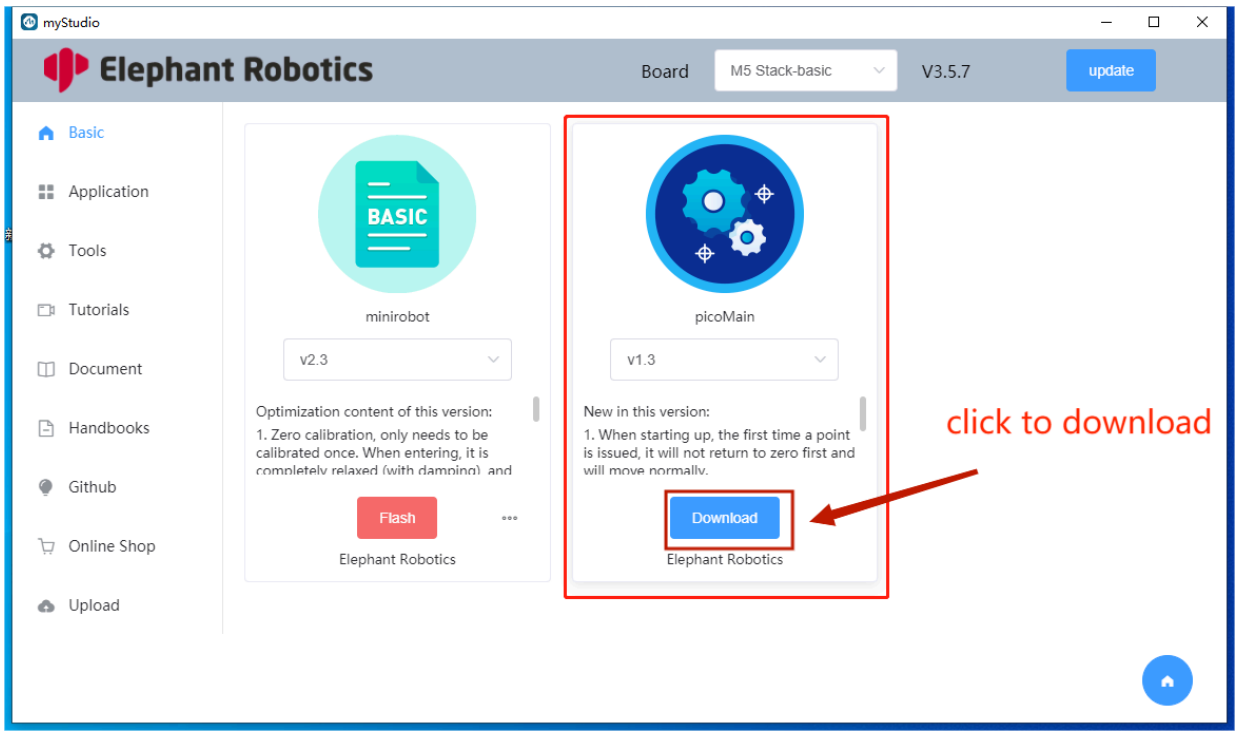
Step 4: Select Robot `myCobot 320 for M5` ,and click `LOGIN` button.



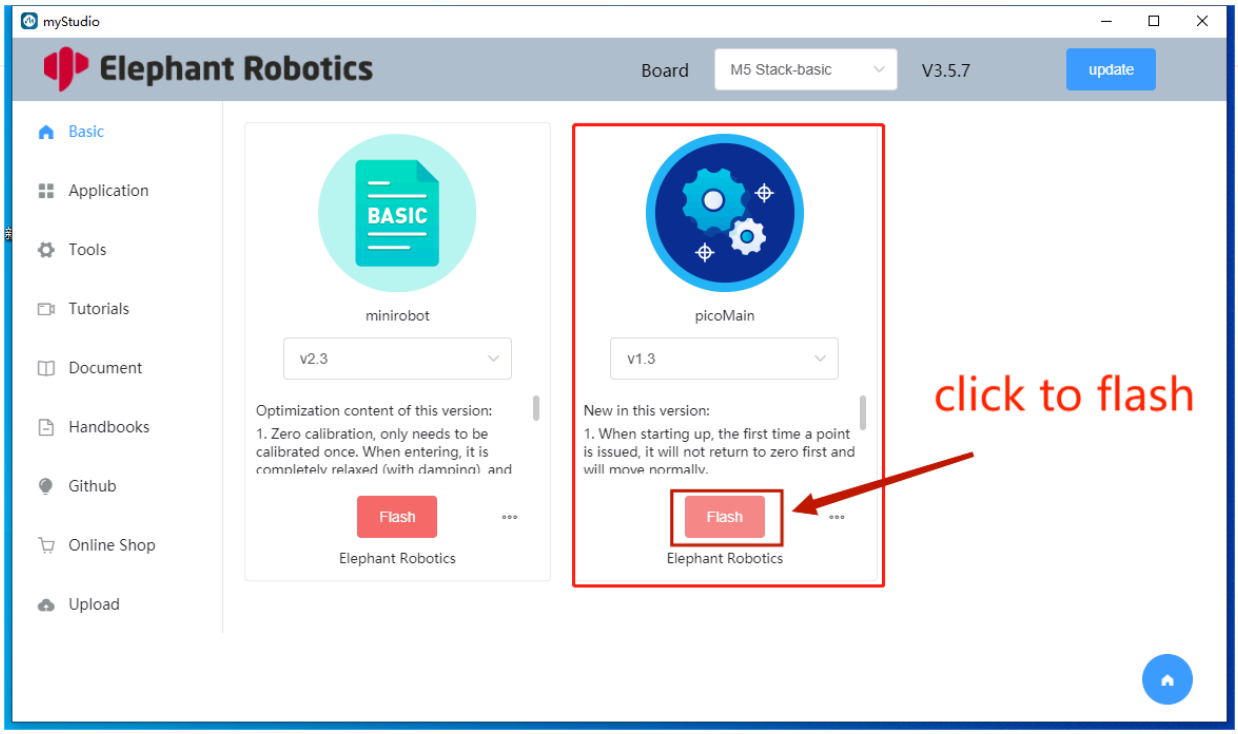
Step 5: After logging in, click `Basic`



Step 6: click `Download` and click `Flash` to burn.



Step 7: click `Flash` to burn.



← Previous Page | Next Page →

Other function

The basic function demonstration is completed, you can check the other information in here

← Previous Page

Development Environment Construction

1 Use Environment

myCobot 320 for M5 is developed and used on the basis of PC. As there is no built-in system inside the robotic arm, the combination of the robotic arm and the PC is required during use. Prepare a PC firstly before use.

2 Development Environment

In order to meet the diverse application needs of robots in different scenarios, we have adapted the robot for multiple programming languages. So far, we have adapted the following mainstream programming languages. If you find any of the following languages challenging, you can return to the **myBlockly** section and use a visual programming language for development. However, we believe that you can use any of the following languages for development. Please make sure to strictly follow the instructions during the operation. Any missed steps may result in the corresponding language not running successfully. We wish you smooth and successful robot usage.

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

- 6.1 Python

  Our robots support Python and the development of the Python API library has become increasingly complete. The joint angle, coordinates, gripper and other aspects of the robot can be controlled via Python.

- 6.2 Robot Operating System 1 (ROS1)

  ROS is open-source and is a post operating system, or secondary operating system, used for robot control. With the use of ROS, the simulation control of the manipulator can be realized in the virtual environment. The robotic arm can be visualized through the rviz platform, and operate the robotic arm in a variety of ways. It can also be used to plan and execute the robotic arm's action path through to freely control the robotic arm.

- 6.3 Robot Operating System 2 (ROS2)

  The predecessor of ROS2 is ROS, and ROS is the Robot Operating System (Robot Operating System). But

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

- 6.4 C Sharp (C#)

C# is an object-oriented programming language derived from C and C++ released by Microsoft, an advanced programming language running on .NET Framework and .NET Core (completely open source and cross-platform) . Using the C#, you can freely develop programs (coordinate control, angle control, io control, gripper control, etc.) through the C# dynamic library provided by our company, and control some of the robots.

- 6.5 C plus plus (C++)

C++ is the inheritance of C Language. It can not only carry out the procedural programming of the C Language, but also the object-based programming characterized by abstract data types, as well as inheritance and polymorphism as the Features of object-oriented programming. With C++ language, you can freely develop programms(coordinate control, angle control, io control, gripper control, etc.) through the C++ dynamic library developed by our company, and control some of the robots.

- 6.6 Arduino

Arduino is an easy-to-use, open-source electronic prototyping platform that includes hardware (various Arduino-compliant development boards) 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-purpose input/output interfaces (GPIO), etc. It can be understood as a microcomputer motherboard. The software part is mainly composed of Arduino IDE on the PC side, related board support packages (BSP) and rich third-party function libraries. Users can use Arduino IDE to easily download the BSP related to the development board and the required function library to write program.

- 6.7 Serial Communication

If you happen to have some understanding of information theory, coding, and robot communication functionality, then you should grasp that all communication originates from data transmission. To make it more convenient for users to operate the robot, we have opened up a communication protocol based on serial communication. You can use a serial assistant or encapsulate it into any programming language you are familiar with to control the robot.

← Previous Chapter | Next Chapter →

What is Python?

Our products are friendly to python and also becomes increasingly perfect for the development of a python API library. Through python, the joint angle, coordinates, gripper and other aspects of the robot can be controlled, and there are many options available. If you want to control our robot arms via Python programming, you are recommended to learn this chapter.



Python was designed in the early 1990s by Guido van Rossum of the Netherlands Society for Mathematics and Computer Science as an alternative to a language called ABC.

Python not only provides efficient, advanced data structures, but also can be used to do simple and effective object-oriented programming.

The Syntax and dynamic typing of **Python** as well as the nature of interpreted languages, make it become a programming language for scripting and rapid application development on most platforms. With the continuous updating of the version and the addition of new features, it is gradually used to develop independent, large-scale projects.

The interpreter of **Python** is easily extensible, and new functions and data types can be extended using the C or C++ language (or other languages that can be called through C language).。

Python can also be used for extending program languages in customizable software. **Python** has rich standard libraries and provides source or machine codes suitable for each major system platform.

Preconditions for use：

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)
- **Pi \ jetsonnano** series, **ATOM** burns the latest version of **atomMain** (factory default already burnt)

← Previous Chapter | Next Page →

Environment Building

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

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

Installing Python



- **Python** Python's official downloading address：https://www.python.org/downloads/
- **Python** downloading and Installation Tutorial for reference only

1 Download and Installation of Python

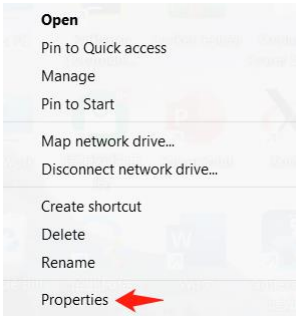
Python is applicable to:

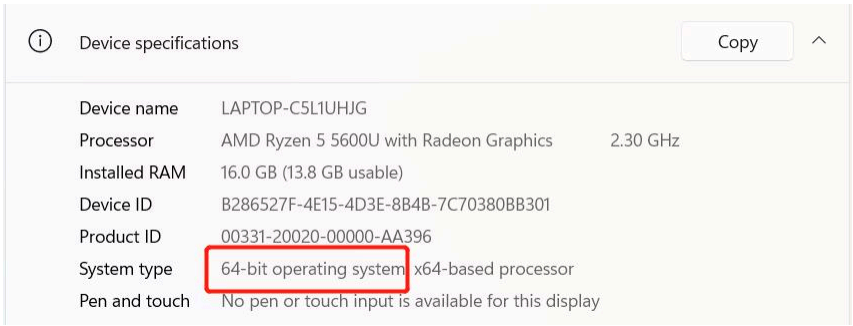
- myCobot 280：
  - myCobot 280 M5
  - myCobot 280 PI
  - myCobot 280 Jetson Nano
  - myCobot 280 for Arduino
- myCobot 320：
  - myCobot 320 M5
  - myCobot 320 PI
- myPalletizer 260：
  - myPalletizer 260 M5
  - myPalletizer 260 PI
- mechArm-270：
  - mechArm-270 M5
  - mechArm-270 PI

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

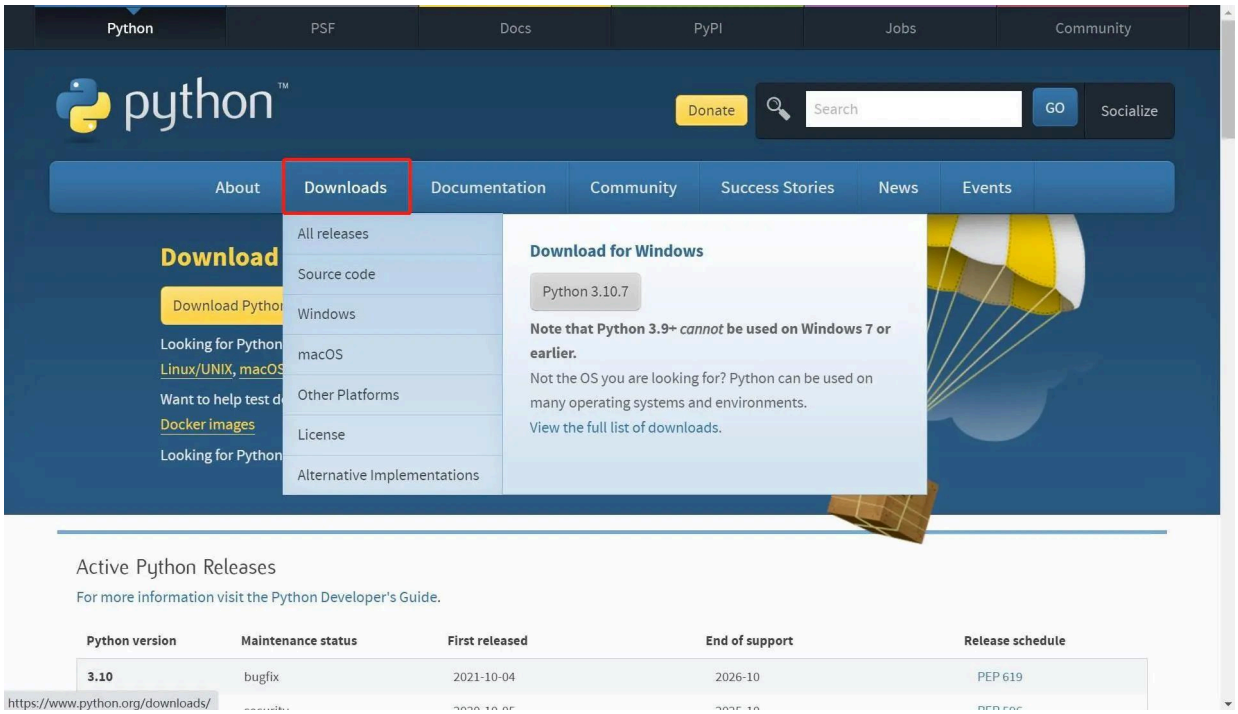
1.1 Installing Python

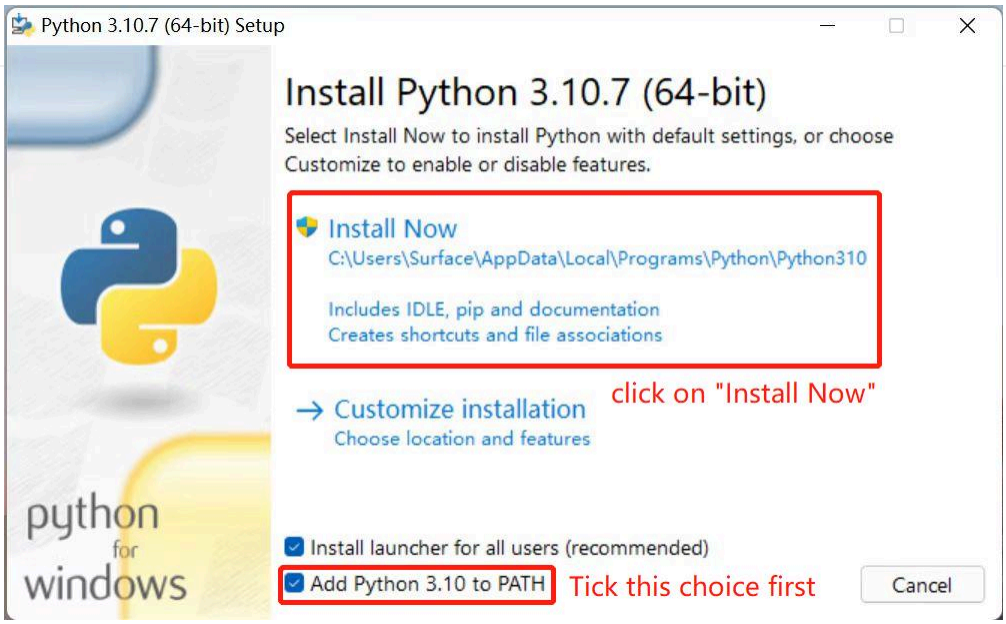
> **Notice:** Before installation, check the operation system of PC. Press right button on the `My Computer` icon and then select `Properties`. Install the corresponding Python.

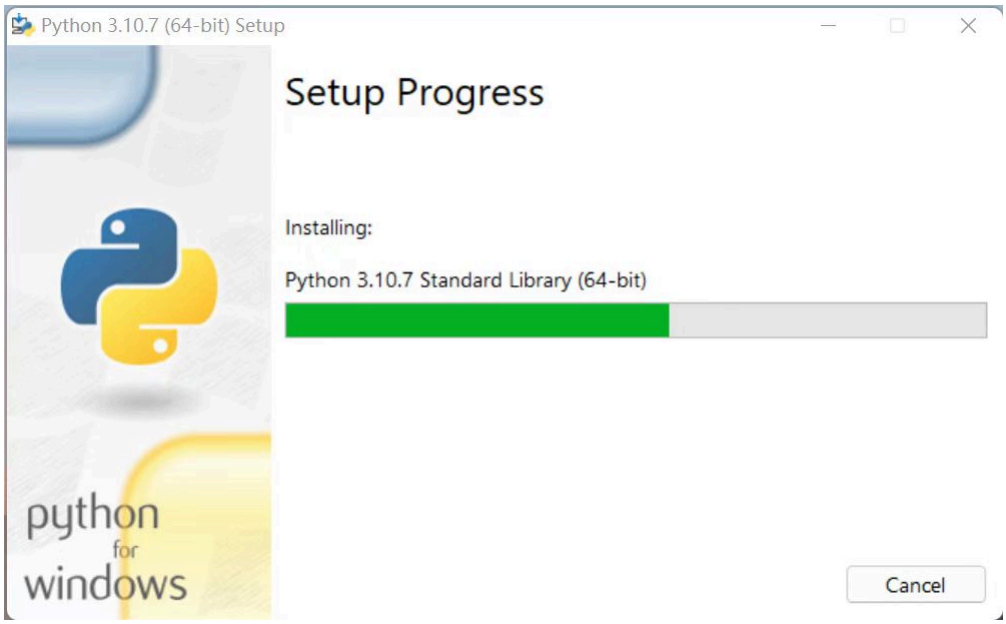




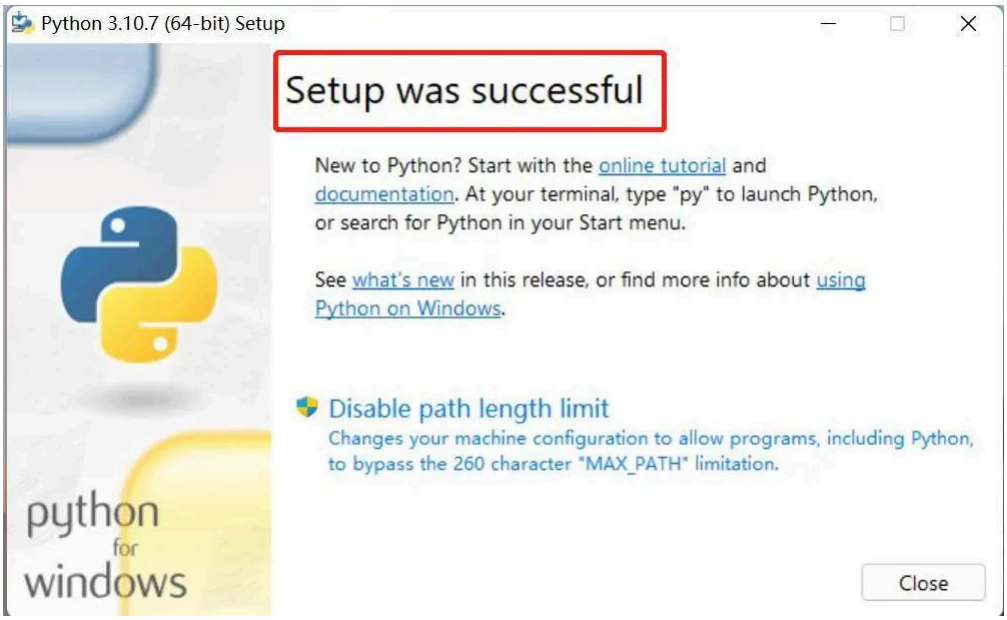
- **Go to http://www.python.org/download/ to download Python.**
- **Click on `Downloads`, and then download begins. Tick `Add Python 3.10 to PATH`. Click on `Install Now`, and then installation begins.**







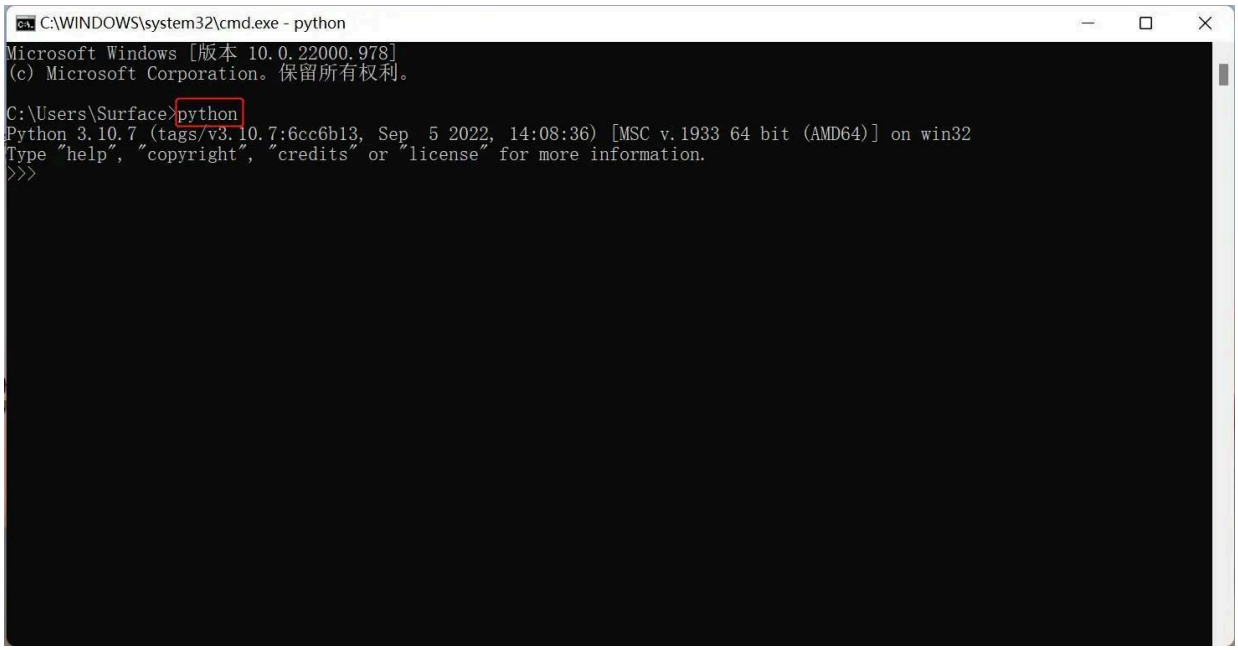
- **Download and installation complete.**



1.2 Running Python

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

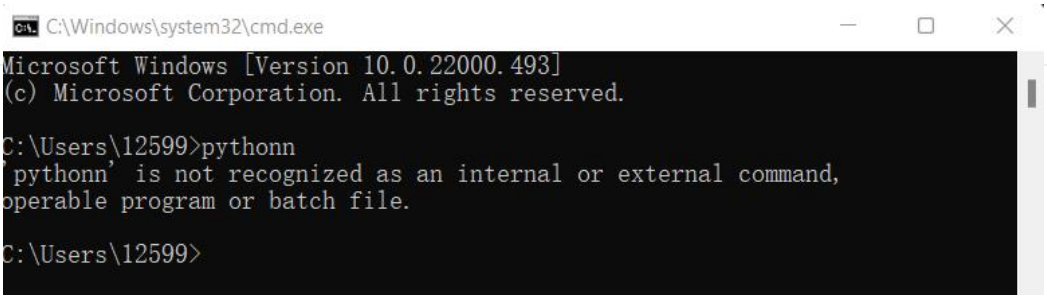
Successful Installation:



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

Error Report:

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



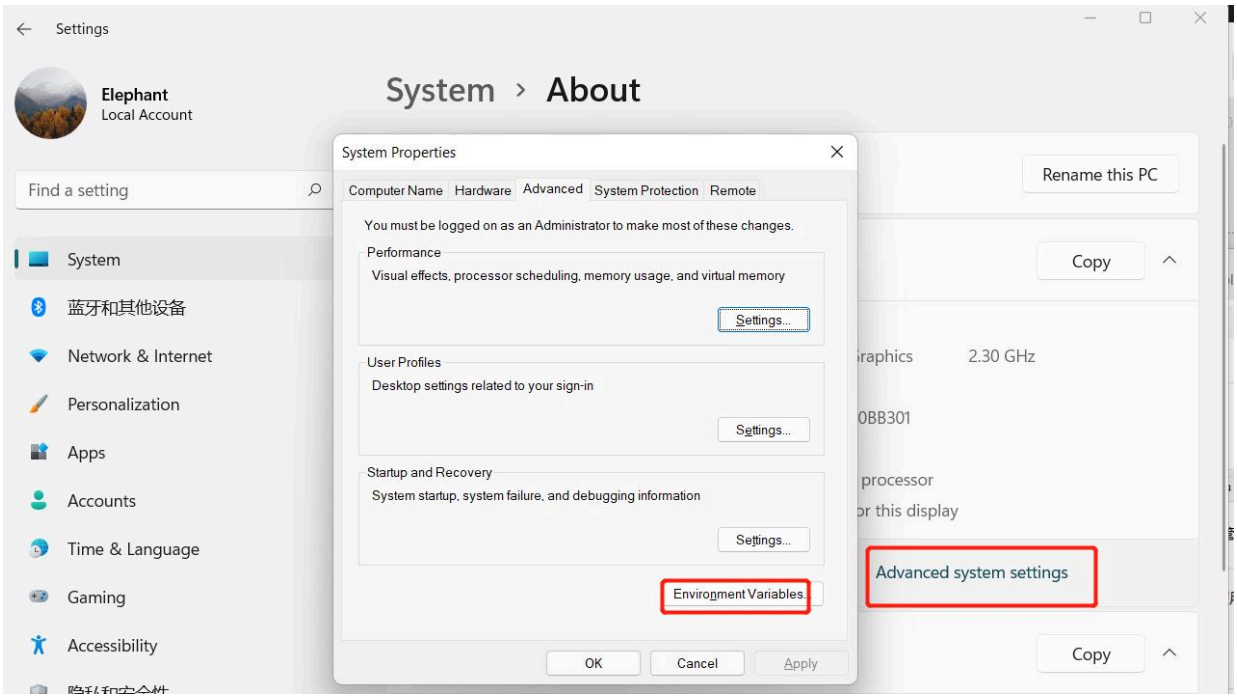
> **Notice:** Generally, the error results from lack of environment configuration. Refer to **1.3 Environment Configuration** to solve problems.

1.3 Environment Variable Configuration

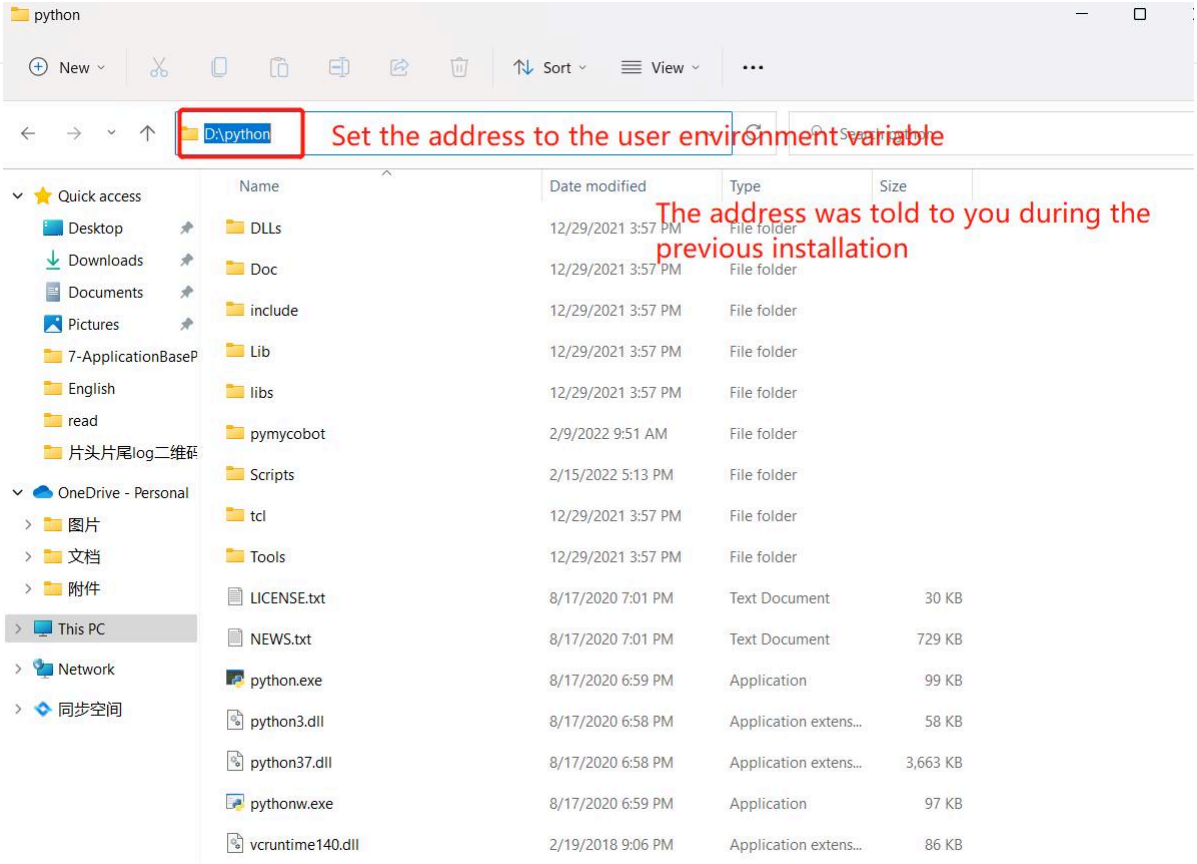
Windows follows the path set by a Path environment variable in search of **python.exe** . Otherwise, an error will be reported. If you fail to tick `Add Python 3.9 to PATH` during installation, you need to manually add the path where python.exe is located into environment variable or download python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

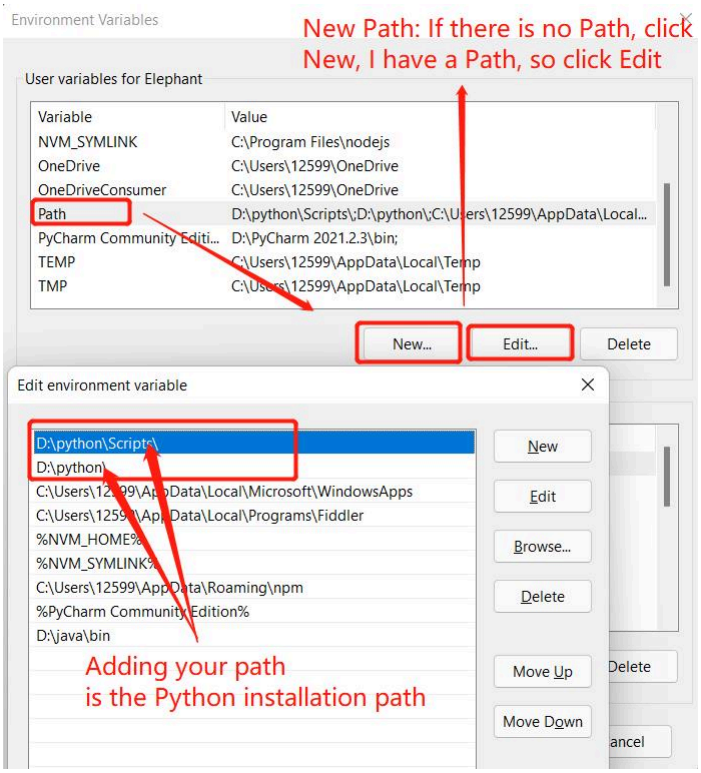
Follow the steps below to add python into environment variable manually.

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

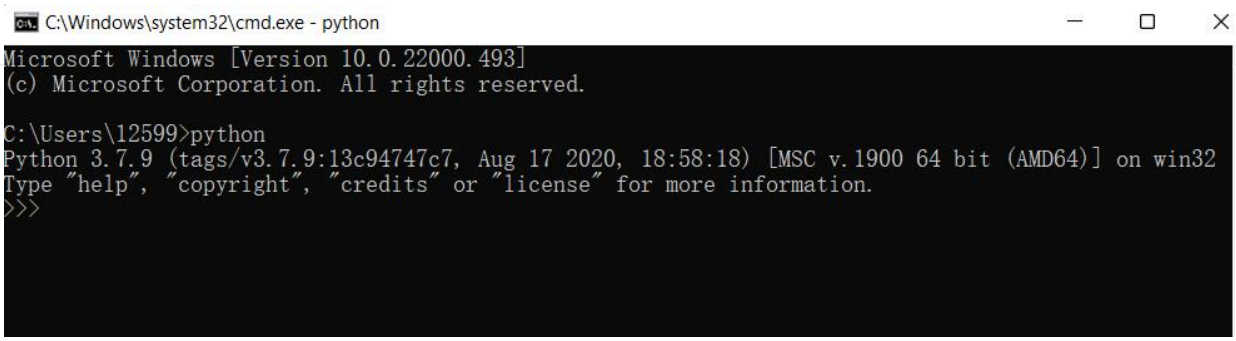


- The environment variables include user variables and system variables. For user variables, users can utilize their own downloaded programs via `cmd` command. Write the absolute path of the target program into the user variables.





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



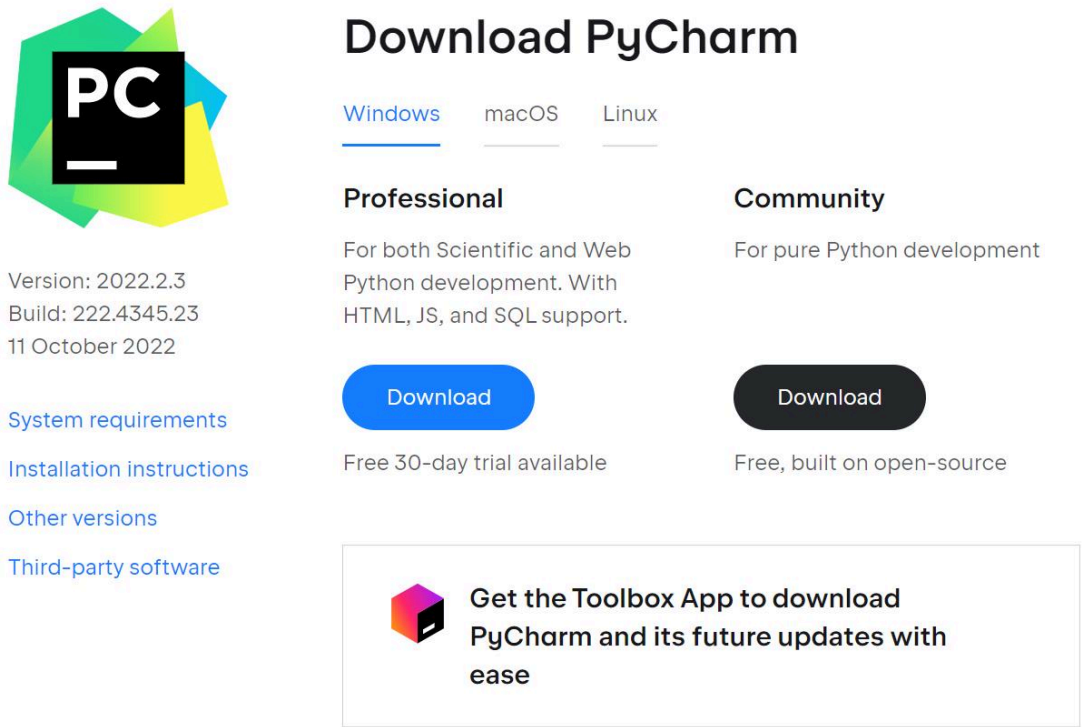
2 Installation of PyCharm

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

Go to **PyCharm** to download PyCharm.

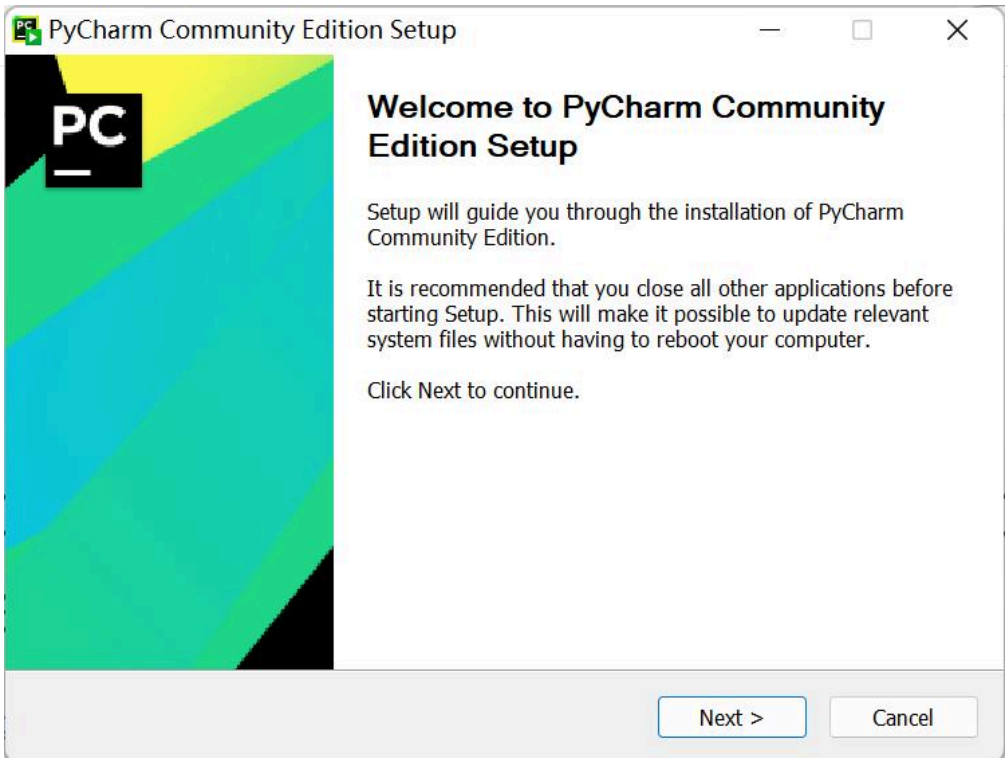
2.1 Download and Installation

Official website view:

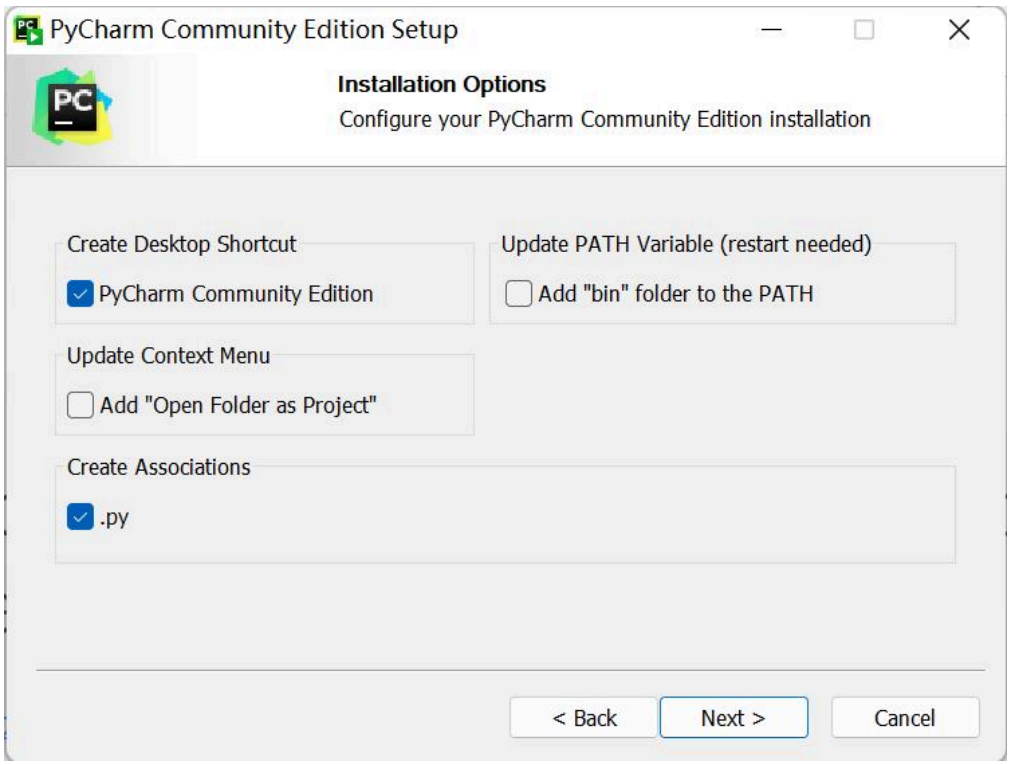


It is recommended to install the free version.

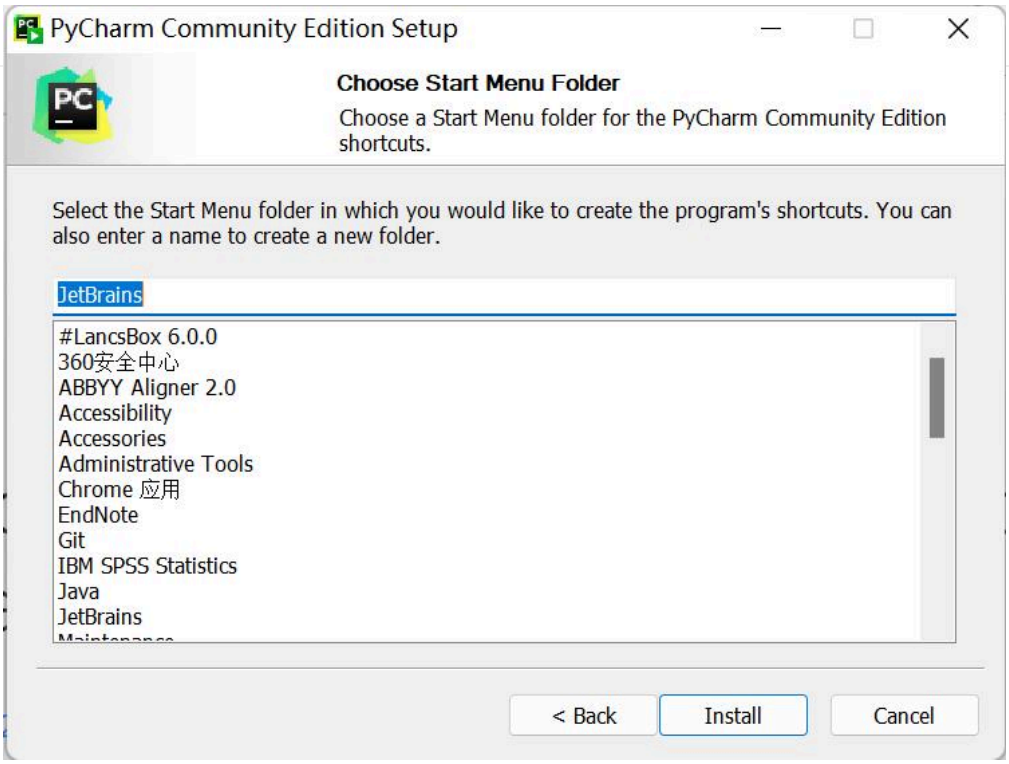
- Click on `Next` :



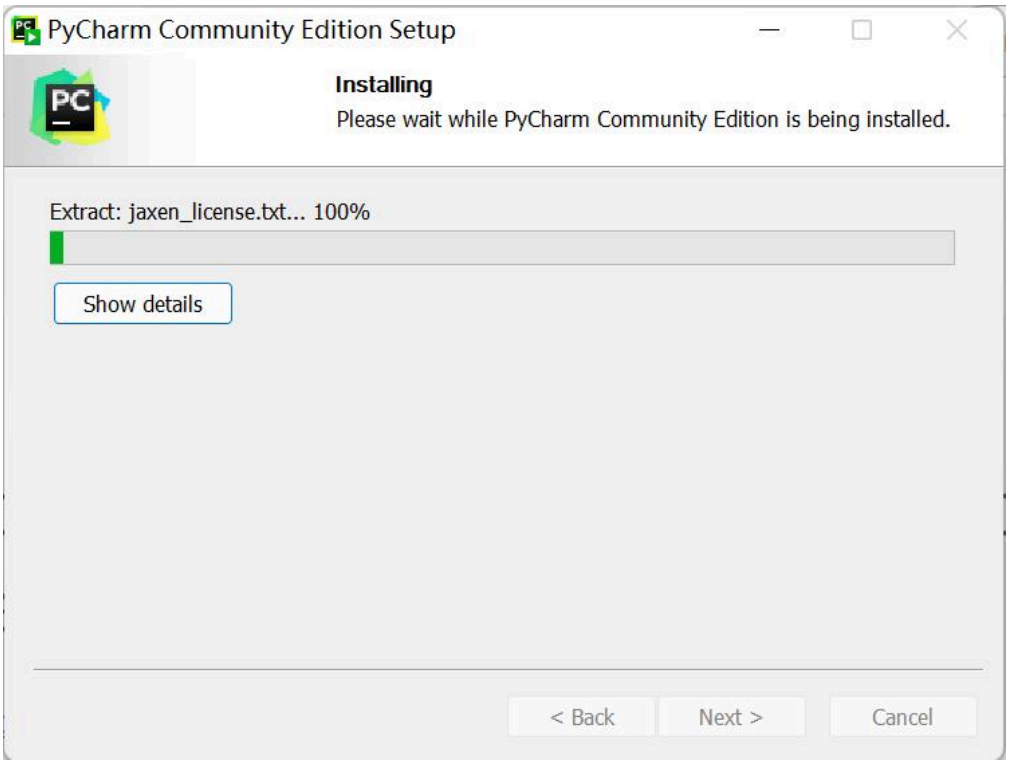
- Select options according to your needs and then select `Next` :



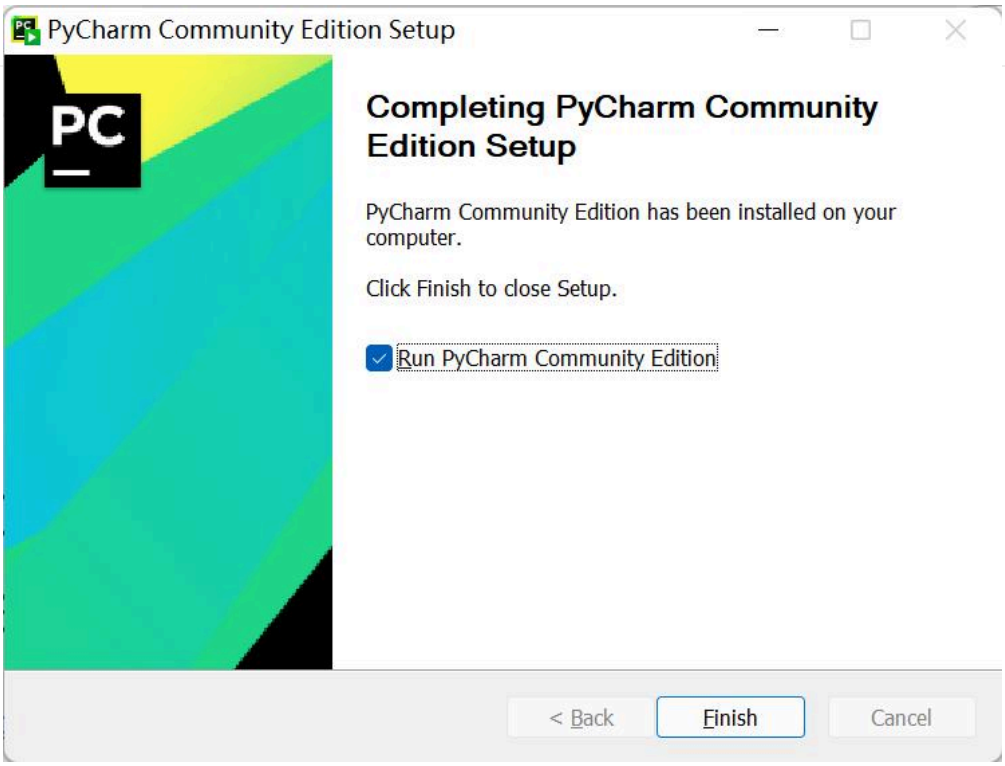
- Tap `Install` :



- Installing:

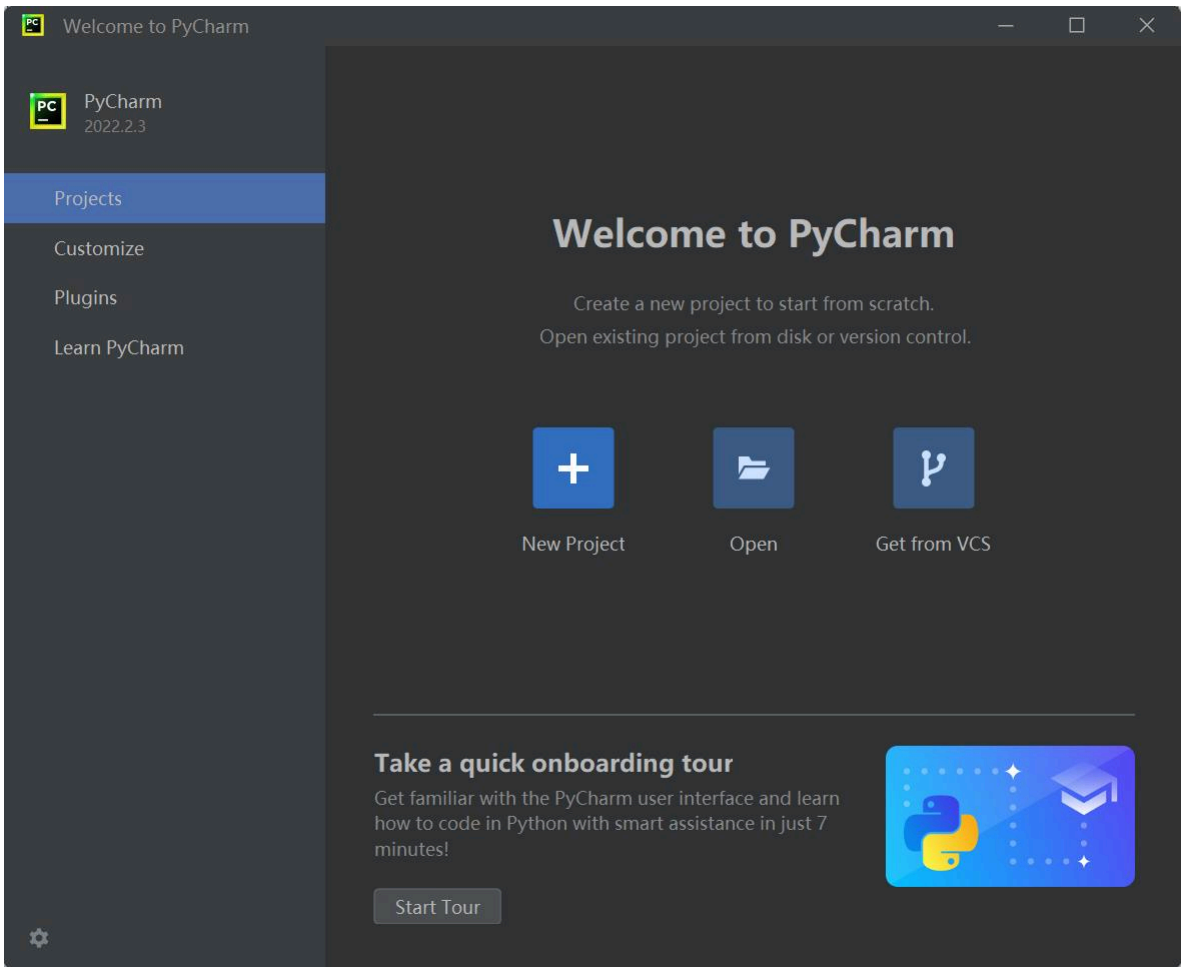


- Tap `Finish`

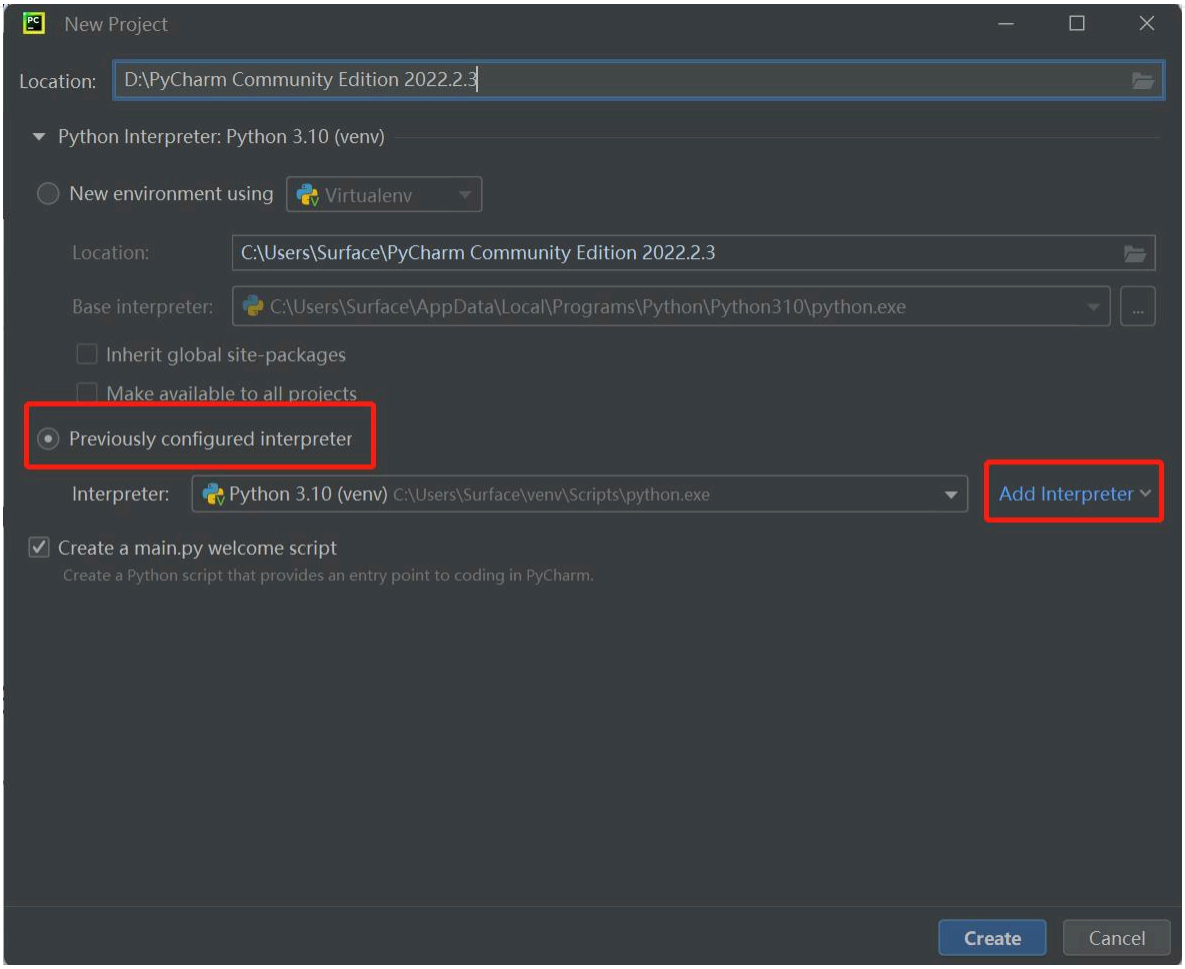


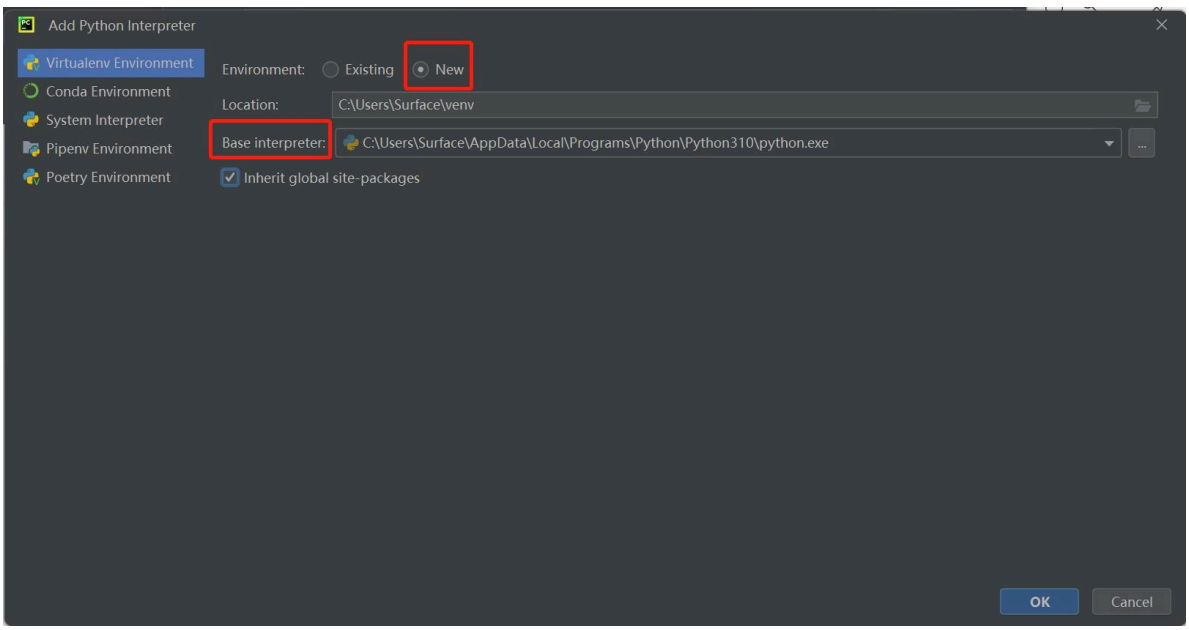
2.2 Create a new project

- Click `+New Project` :

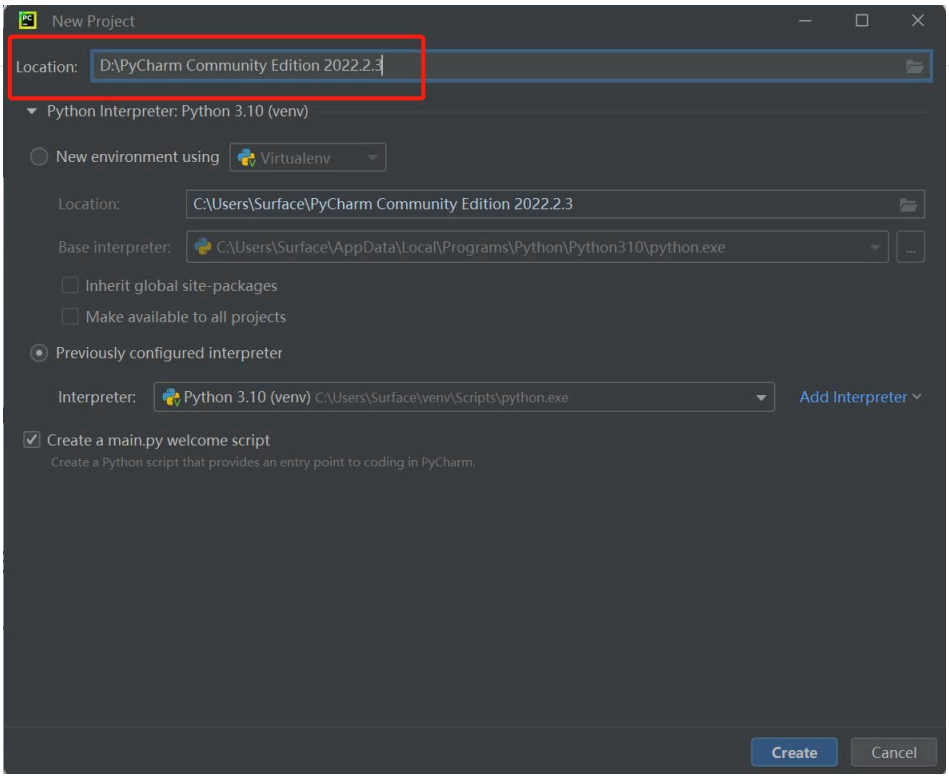


- The `Interpreter` is used to interpret python programs. Select `Add Interpreter` -> `New` to add base interpreter.

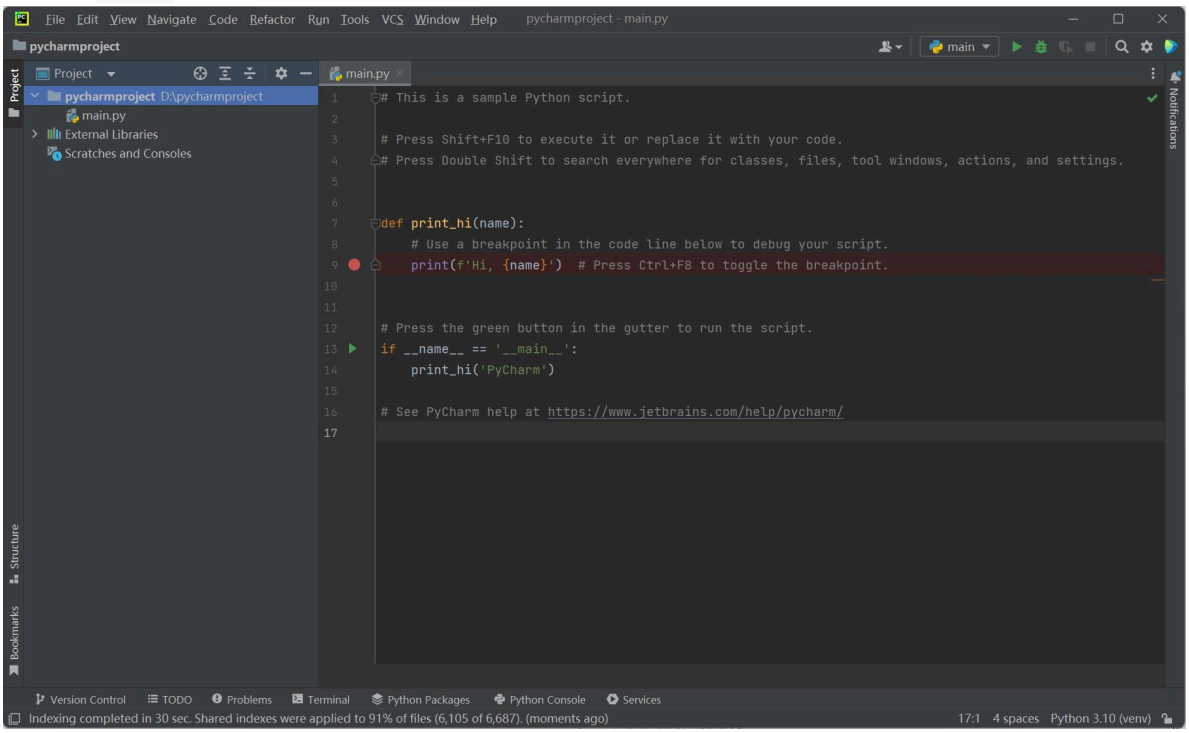




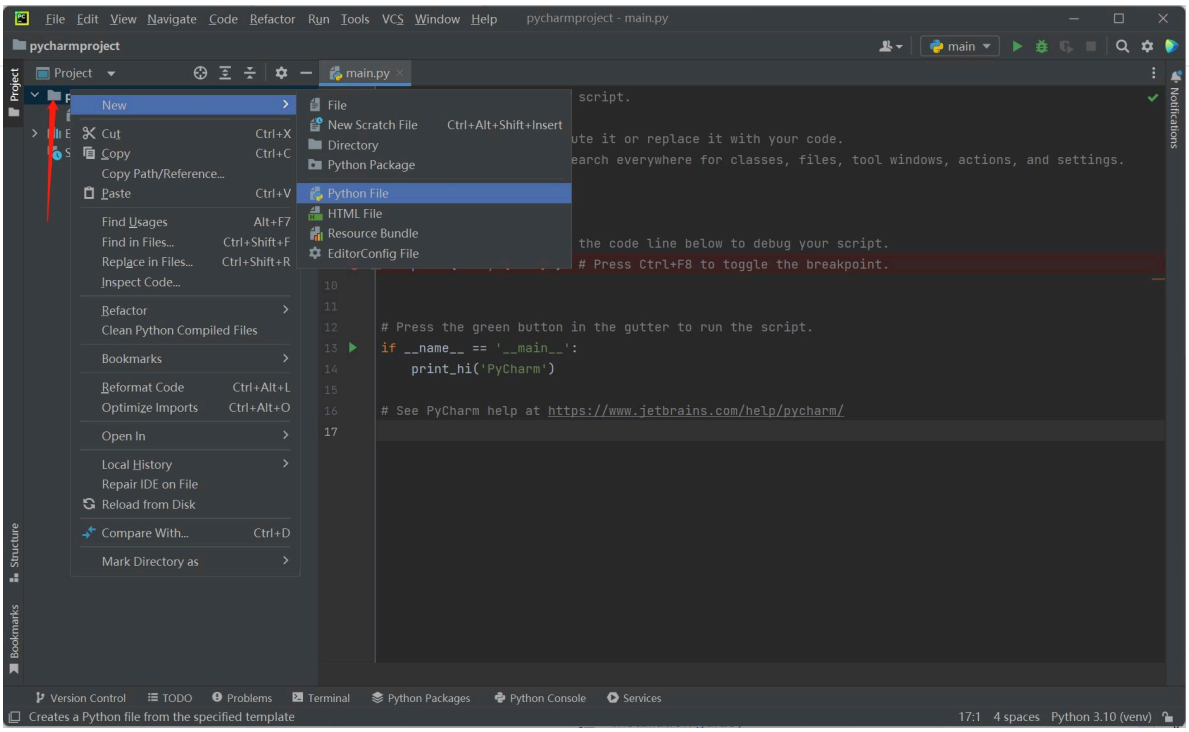
- `Location` refers to the place where to save python file. Choose a file to put your programs.



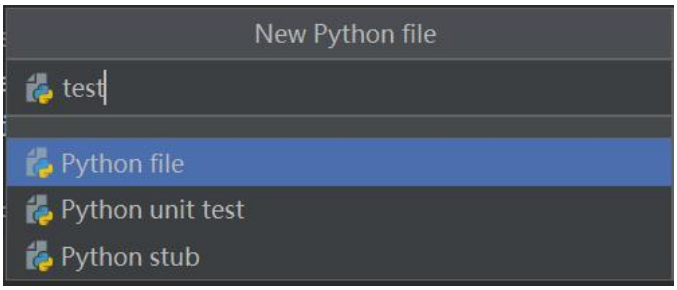
- Click on `Create` and a sample appears:



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



- Type name for the new file.



3 Preparations

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

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



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

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

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

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

- Update pymycobot

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

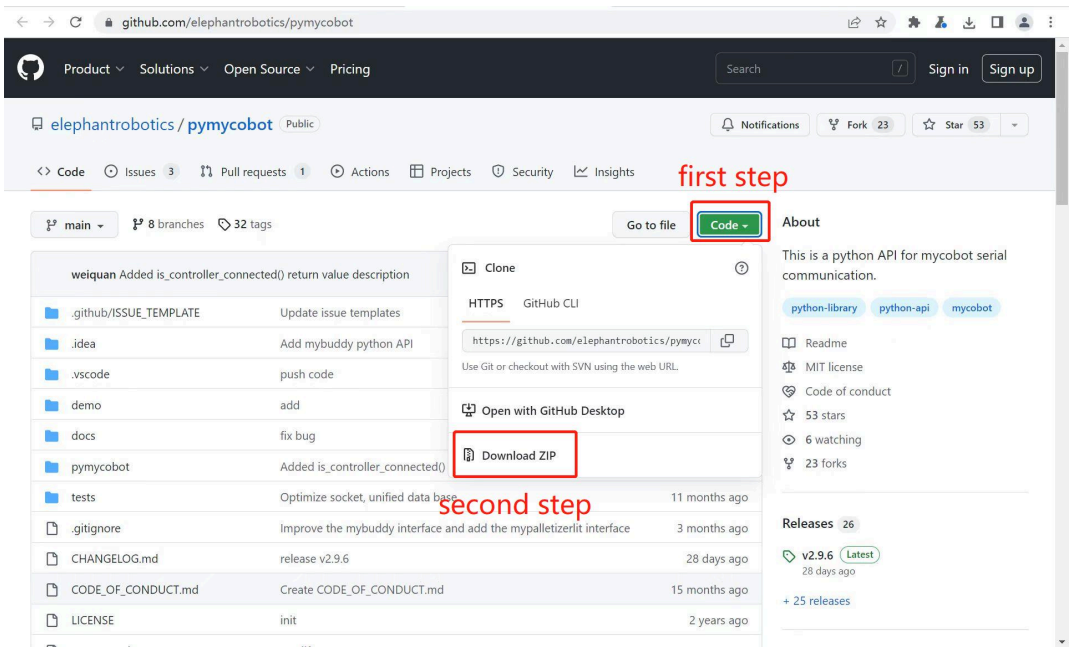
4 Import of pymycobot

This part takes MyCobot 320 M5 as an example to introduce how to control a robot via python.

- Import pymycobot library for MyCobot320 :

```
from pymycobot.mycobot import MyCobot320
```

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



← Previous Page | Next Page →

MyCobot 320

[toc]

Python API usage instaructions

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 MyCobot320

mc = MyCobot320('/dev/ttyAMA0', 115200)

# 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)
```

1. System Status

1.1 `get_system_version()`

- **function：** Get the machine master control version (pico firmware version)
- **Return value：** Firmware version number

1.2 `get_atom_version()`

- **function：** Get the atom version on the machine end
- **Return value：** Firmware version number

1.3 `get_basic_version()`

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

2. Overall Status

2.1 `power_on()`

- **function:** atom open communication (default open)
- **Return value:**
  - `1` : complete

2.2 `power_off()`

- **function:** Power off of the robotic arm
- **Return value:**
  - `1` : complete

2.3 `is_power_on()`

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

2.4 `release_all_servos(data=None)`

- **function:** release all robot arms
  - 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:**
  - `1` : complete

2.5 `focus_servos(servo_id)`

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

2.6 `is_controller_connected()`

- **function:** Wether 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 `get_robot_status()`

- **Function:** Get the robot self-check status to see if important parameters are normal.
- **Abnormal description:**
  - 0: Communication abnormality, please check whether the line, servo firmware version is normal, whether the power is plugged in, whether the firmware is burned correctly, whether the baud rate is correct, etc.
  - 1: The servo motor model is wrong and the motor needs to be replaced
  - 2: The servo motor firmware version is low and needs to be upgraded using FD
  - 3: The servo motor p value is abnormal, the default is 32, this abnormality will be automatically restored
  - 4: The servo motor D value is abnormal, the default is 8, this abnormality will be automatically restored
  - 5: The servo motor I value is abnormal, the default is 0, this abnormality will be automatically restored
  - 6: The servo motor clockwise insensitive zone parameter is abnormal, the default is 3, this abnormality will be automatically restored
  - 7: The servo motor counterclockwise insensitive zone parameter is abnormal, the default is 3, this abnormality will be automatically restored
  - 8: The servo motor phase is abnormal, this abnormality will be automatically restored
  - 9: The servo motor return delay is abnormal, the default is 0, this abnormality will be automatically restored
  - 10: The servo motor minimum starting force is abnormal, the default is 0, this abnormality will be automatically restored
  - 11: The servo motor is abnormal. When the servo is abnormal, the machine cannot be controlled. Please query the servo feedback interface get_servo_status to check the specific error.
  - 255: Unknown error

2.11 `focus_all servos()`

- **Function:** All servos are powered on
- **Return value:**
  - `1` : complete

2.12 `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`)

Joint Id	Range
1	-168 ~ 168
2	-135 ~ 135
3	-145 ~ 145
4	-148 ~ 184
5	-168 ~ 168
6	-180 ~ 180

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

3.3 `send_angles(angles, speed)`

- **function**： Send all angles to all joints of the robotic arm
- **Parameters:**
  - `angles` : a list of degree value(`List[float]`), length 6
  - `speed` : (`int`) 1 ~ 100
- **Return value:**
  - `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:**
  - `id` :send one coord to robot arm, 1-6 corresponds to [x, y, z, rx, ry, rz]
  - `coord` : coord value(`float`)

Coord Id	Range
x	-350 ~ 350
y	-350 ~ 350
z	-41 ~ 523.9
rx	-180 ~ 180
ry	-180 ~ 180
rz	-180 ~ 180

- `speed` : (`int`) 1-100
  - **Return value:**
- `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:**
  - coords: ： a list of coords value `[x,y,z,rx,ry,rz]` ,length6
  - speed `(int)` : 1 ~ 100
  - mode: `(int)` 0 - angluar, 1 - linear
- **Return value:**
  - `1` : complete

3.7 `pause()`

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

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:**
  - `angles` : a list of degree value(`List[float]`), length 6
  - `speed` : (`int`) 1 ~ 100
  - `timeout` : default 15 s
- **Return value:**
  - `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:**
  - `coords` : a list of coord value(`List[float]`), length 6
  - `speed` : (`int`) 1 ~ 100
  - `mode` : (`int`) 0 - angular（default）, 1 - linear
  - `timeout` : default 15 s
- **Return value:**
  - `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:**
  - `1` - paused
  - `0` - not paused
  - `-1` - error

3.12 `resume()`

- **function:** resume the robot movement and complete the previous command

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**:
  - `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**:
  - `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.

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. Servo state value

7.1 `get_servo_speeds()`

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

7.2 `get_servo_currents()`

- **function**：Get joint current
- **Return value**： A list 0 ~ 3250 mA

7.3 `get_servo_voltages()`

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

7.4 `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
- **Abnormal description**:

- 0: Servo motor undervoltage/overvoltage, check the voltage, if it is 0, you need to modify the servo parameters; if it is greater than the actual value, the heat sink may be damaged
- 1: Servo motor magnetic encoding abnormality
- 2: Servo motor overtemperature
- 3: Servo motor overcurrent
- 5: Servo motor overload

7.5 `get_servo_temps()`

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

7.6 `set_void_compensate(mode)`

- **function:** Set void compensation mode
- **Parameters**：
  - `mode` : (`int`) 0 - close, 1 - open
- **Return value:**
  - `1` : complete

8. Robotic arm end IO control

8.1 `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

8.2 `set_digital_output(pin_no, pin_signal)`

- **function:** set IO statue
- **Parameters**
  - `pin_no` (int): Pin number
  - `pin_signal` (int): 0 / 1
- **Return value:**
  - `1` : complete

8.3 `get_digital_input(pin_no)`

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

8.4 `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

9. Robotic arm end gripper control

9.1 `set_gripper_state(flag, speed, _type_1=None)`

- **function**: Adaptive gripper enable
- **Parameters**:
  - `flag (int)` : 0 - open 1 - close, 254 - release
  - `speed (int)` : 1 ~ 100
  - `_type_1 (int)` :
    - `1` : Adaptive gripper (default state is 1)
    - `2` : A nimble hand with 5 fingers
    - `3` : Parallel gripper
    - `4` : Flexible gripper
- **Return value:**
  - `1` : complete

9.2 `set_gripper_value(gripper_value, speed, gripper_type=None)`

- **function**: Set the gripper value
- **Parameters**:
  - `gripper_value (int)` : 0 ~ 100
  - `speed (int)` : 1 ~ 100
  - `gripper_type (int)` :
    - `1` : Adaptive gripper (default state is 1)
    - `2` : A nimble hand with 5 fingers
    - `3` : Parallel gripper
    - `4` : Flexible gripper
- **Return value:**
  - `1` : complete

9.3 `set_gripper_calibration()`

- **function**: Set the current position of the gripper to zero

- **Return value:**
  - `1` : complete

9.4 `init_electric_gripper()`

- **function**: Electric gripper initialization (it needs to be initialized once after inserting and removing the gripper)
- **Return value:**
  - `1` : complete

9.5 `set_electric_gripper(status)`

- **function**: Set Electric Gripper Mode
- **Parameters**:
  - `status` : 0 - open, 1 - close.
- **Return value:**
  - `1` : complete

9.6 `set_gripper_mode(mode)`

- **function**: Set gripper mode
- **Parameters**:
  - `mode` : 0 - transparent transmission. 1 - Port Mode.
- **Return value:**
  - `1` : complete

9.7 `get_gripper_mode()`

- **function**: Get gripper mode
- **Return value**:
  - `mode` : 0 - transparent transmission. 1 - Port Mode.

10. Set bottom IO input/output status

10.1 `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
- **Return value:**
  - `1` : complete

10.2 `get_basic_input(pin_no)`

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

11. WLAN Setting

11.1 `set_ssid_pwd(account, password)`

- **function:** Change connected wifi. (Apply to m5)
- **Parameters:**
  - `account` (`str`) new wifi account
  - `password` (`str`) new wifi password
- **Return value:**
  - `1` : complete

11.2 `get_ssid_pwd()`

- **function:** Get connected wifi account and password. (Apply to m5)
- **Return value:** (account, password)

11.3 `set_server_port(port)`

- **function:** Change the connection port of the server
- **Parameters:**
  - `port` (`int`) The new connection port of the server.
- **Return value:**
  - `1` : complete

12. TOF

12.1 `get_tof_distance()`

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

13. Cartesian space coordinate parameter setting

13.1 `set_tool_reference(coords)`

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

13.2 `get_tool_reference(coords)`

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

13.3 `set_world_reference(coords)`

- **function:** Set world coordinate system.
- **Parameters**：

  - `coords` : (`list`) [x, y, z, rx, ry, rz].
- **Return value:**
  - `1` : complete

13.4 `get_world_reference()`

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

13.5 `set_reference_frame(rftype)`

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

13.6 `get_reference_frame()`

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

13.7 `set_movement_type(move_type)`

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

13.8 `get_movement_type()`

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

13.9 `set_end_type(end)`

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

13.10 `get_end_type()`

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

14. 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
```

14.1 `utils.get_port_list()`

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

14.2 `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

15. Pro force-controlled gripper

15.1 `set_pro_gripper(address, value, gripper_id=14)`

- **Function**: Set the parameters of the Pro force-controlled gripper. You can set a variety of parameter functions. For details, please see the table below.
- **Parameter**:
  - `address` (`int`): The command number of the gripper.
  - `value` : The parameter value corresponding to the command number.
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.

Function	address	value	gripper_id
Set gripper ID	3	1 ~ 254	14
Set gripper enable status	10	0 or 1, 0 - off enable; 1 - on enable	14
Set gripper clockwise runnable error	21	0 ~ 16	14
Set gripper counterclockwise runnable error	23	0 ~ 16	14
Set gripper minimum starting force	25	0 ~ 254	14
IO output settings	29	0, 1, 16, 17	14
Set IO opening angle	30	0 ~ 100	14
Set IO closing angle	31	0 ~ 100	14
Set servo virtual position value	41	0 ~ 100	14
Set clamping current	43	100 ~ 300	14

- **Return value**:

- Please refer to the following table:

Function	Return
Set gripper ID	0 - Failure; 1 - Success
Set gripper enable status	0 - Failure; 1 - Success
Set gripper clockwise runnable error	0 - Failure; 1 - Success
Set gripper counterclockwise runnable error	0 - Failure; 1 - Success
Set gripper minimum starting force	0 - Failure; 1 - Success
IO output setting	0 - Failure; 1 - Success
Set IO opening angle	0 - Failure; 1 - Success
Set IO closing angle	0 - Failed; 1 - Success
Set servo virtual position value	0 - Failed; 1 - Success
Set holding current	0 - Failed; 1 - Success

15.2 `get_pro_gripper(address, gripper_id=14)`

- **Function**: Get the parameters of the Pro force-controlled gripper, and you can get a variety of parameter functions. For details, please see the table below.
- **Parameter**:
  - `address` (`int`): The command number of the gripper.
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.

Function	address	gripper_id
Read firmware major version number	1	14
Read firmware minor version number	2	14
Read gripper ID	4	14
Read gripper clockwise runnable error	22	14
Read gripper counterclockwise runnable error	24	14
Read gripper minimum starting force	26	14
Read IO opening angle	34	14
Read IO closing angle	35	14
Get the amount of data in the current queue	40	14
Read servo virtual position value	42	14
Read the clamping current	44	14

- **Return value**:

- See the following table (if the return value is -1, it means that no data can be read):

Function	Return
Read the firmware major version number	Major version number
Read the firmware minor version number	Minor version number
Read the gripper ID	1 ~ 254
Read the gripper clockwise runnable error	0 ~ 254
Read the gripper counterclockwise runnable error	0 ~ 254
Read the gripper minimum starting force	0 ~ 254
Read the IO opening angle	0 ~ 100
Read the IO closing angle	0 ~ 100
Get the amount of data in the current queue	Return the amount of data in the current absolute control queue
Read the servo virtual position value	0 ~ 100
Read the clamping current	100 ~ 300

15.3 `set_pro_gripper_angle(gripper_angle, gripper_id=14)`

- **Function**: Set the force-controlled gripper angle.
- **Parameter**:
  - `gripper_angle` (`int`): Gripper angle, value range 0 ~ 100.
  - `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.4 `get_pro_gripper_angle(gripper_id=14)`

- **Function**: Read the angle of the force-controlled gripper.
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**: `int` 0 ~ 100

15.5 `set_pro_gripper_open(gripper_id=14)`

- **Function**: Open the force-controlled gripper.
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.6 `set_pro_gripper_close(gripper_id=14)`

- **Function**: Close the force-controlled gripper.
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.7 `set_pro_gripper_calibration(gripper_id=14)`

- **Function**: Set the zero position of the force-controlled gripper. (The zero position needs to be set first when using it for the first time)
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.8 `get_pro_gripper_status(gripper_id=14)`

- **Function**: Read the gripping status of the force-controlled gripper.
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value:**
  - `0` - Moving.
  - `1` - Stopped moving, no object was detected.
  - `2` - Stopped moving, object was detected.
  - `3` - After the object was detected, it fell.

15.9 `set_pro_gripper_torque(torque_value, gripper_id=14)`

- **Function**: Set the torque of the force-controlled gripper.
- **Parameter**:
  - `torque_value` (`int`): Torque value, value range 0 ~ 100.
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.10 `get_pro_gripper_torque(gripper_id=14)`

- **Function**: Read the torque of the force-controlled gripper.
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value:** (`int`) 0 ~ 100

15.11 `set_pro_gripper_speed(speed, gripper_id=14)`

- **Function**: Set the force-controlled gripper speed.
- **Parameter**:

- `speed` (int): Gripper movement speed, value range 1 ~ 100.
- `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.

- **Return value**:
  - 0 - Failed
  - 1 - Success

15.12 `get_pro_gripper_default_speed(speed, gripper_id=14)`

- **Function**: Read the default speed of the force-controlled gripper.
- **Parameter**:
  - `gripper_id` (`int`): Gripper ID, default 14, value range 1 ~ 254.
- **Return value**: Gripper default movement speed, range 1 ~ 100.

15.13 `set_pro_gripper_abs_angle(gripper_angle, gripper_id=14)`

- **Function**: Set the absolute angle of the force-controlled gripper.
- **Parameter**:
  - `gripper_angle` (`int`): Gripper angle, value range 0 ~ 100.
  - `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.14 `set_pro_gripper_pause(gripper_id=14)`

- **Function**: Pause motion.
- **Parameter**:
  - `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.15 `set_pro_gripper_resume(gripper_id=14)`

- **Function**: Motion recovery.
- **Parameter**:
  - `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

15.16 `set_pro_gripper_stop(gripper_id=14)`

- **Function**: Stop motion.
- **Parameter**:
  - `gripper_id` (`int`) Gripper ID, default 14, value range 1 ~ 254.
- **Return value**:
  - 0 - Failed
  - 1 - Success

16. myGripper H100 three-finger gripper

16.1 `get_hand_firmware_major_version(gripper_id=14)`

- **Function**: Read the firmware **major** version number.
- **Parameter**:
  - `gripper_id` (`int`) Gripper ID, default 14, range 1 ~ 254.
- **Return value**: (`float`) Major version number.

16.2 `get_hand_firmware_minor_version(gripper_id=14)`

- **Function**: Read the firmware **minor** version number.
- **Parameter**:
  - `gripper_id` (`int`) Gripper ID, default 14, range 1 ~ 254.
- **Return value**: Minor version number.

16.3 `set_hand_gripper_id(id_value, gripper_id=14)`

- **Function**: Set the gripper ID.
- **Parameter**:
  - `id_value` (`int`) New ID, range 1 ~ 254.
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.4 `get_hand_gripper_id(gripper_id=14)`

- **Function**: Get the gripper ID.
- **Parameter**:
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: Integer representing the gripper ID.

16.5 `set_hand_gripper_angle(joint_id, gripper_angle, gripper_id=14)`

- **Function**: Set the angle of a single joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_angle` (`int`) 0 ~ 100
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.6 `get_hand_gripper_angle(joint_id, gripper_id=14)`

- **Function**: Get the angle of a single joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `gripper_angle` (`int`) 0 ~ 100

16.7 `set_hand_gripper_angles(gripper_angles, speed, gripper_id=14)`

- **Function**: Set angles for all joints.
- **Parameter**:
  - `gripper_angles` (`list[int]`) 6 values, 0 ~ 100
  - `speed` (`int`) 0 ~ 100
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.8 `get_hand_gripper_angles(gripper_id=14)`

- **Function**: Get angles of all joints.
- **Parameter**:
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: List of 6 integers (0 ~ 100)

16.9 `set_hand_gripper_torque(joint_id, torque_value, gripper_id=14)`

- **Function**: Set the torque of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `torque_value` (`int`) 0 ~ 100
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.10 `get_hand_gripper_torque(joint_id, gripper_id=14)`

- **Function**: Get the torque of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `torque_value` (`int`) 0 ~ 100

16.11 `set_hand_gripper_calibrate(joint_id, gripper_id=14)`

- **Function**: Calibrate the zero position of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.12 `get_hand_gripper_status(gripper_id=14)`

- **Function**: Get the clamping status of the gripper.
- **Parameter**:

- `gripper_id` (`int`) Gripper ID, default 14, range 1 ~ 254.
- **Return value**:
  - 0 - Moving
  - 1 - Stopped, no clamping
  - 2 - Stopped, clamping detected
  - 3 - Object fell after clamping

16.13 `set_hand_gripper_enabled(flag, gripper_id=14)`

- **Function**: Set the enable state of the gripper
- **Parameter**:
  - `flag (int)` : 0 or 1
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: Operation status

16.14 `set_hand_gripper_speed(joint_id, speed, gripper_id=14)`

- **Function**: Set the speed of the gripper joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `speed` (`int`) 1 ~ 100
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.15 `get_hand_gripper_default_speed(joint_id, gripper_id=14)`

- **Function**: Get the default speed of the gripper joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: Default speed (`int`) 1 ~ 100

16.16 `set_hand_gripper_p(joint_id, value, gripper_id=14)`

- **Function**: Set the P value for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `value` (`int`) 0 ~ 254
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.17 `get_hand_gripper_p(joint_id, gripper_id=14)`

- **Function**: Get the P value of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14

- **Return value**: `value` (`int`) 0 ~ 254

16.18 `set_hand_gripper_d(joint_id, value, gripper_id=14)`

- **Function**: Set the D value for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `value` (`int`) 0 ~ 254
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.19 `get_hand_gripper_d(joint_id, gripper_id=14)`

- **Function**: Get the D value of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `value` (`int`) 0 ~ 254

16.20 `set_hand_gripper_i(joint_id, value, gripper_id=14)`

- **Function**: Set the I value for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `value` (`int`) 0 ~ 254
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.21 `get_hand_gripper_i(joint_id, gripper_id=14)`

- **Function**: Get the I value of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `value` (`int`) 0 ~ 254

16.22 `set_hand_gripper_min_pressure(joint_id, value, gripper_id=14)`

- **Function**: Set the minimum starting force of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `value` (`int`) 0 ~ 254
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.23 `get_hand_gripper_min_pressure(joint_id, gripper_id=14)`

- **Function**: Get the minimum starting force of a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `value` (`int`) 0 ~ 254

16.24 `set_hand_gripper_clockwise(joint_id, value, gripper_id=14)`

- **Function**: Set clockwise error range for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `value` (`int`) 0 ~ 16
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.25 `get_hand_gripper_clockwise(joint_id, gripper_id=14)`

- **Function**: Get clockwise error range for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `value` (`int`) 0 ~ 16

16.26 `set_hand_gripper_counterclockwise(joint_id, value, gripper_id=14)`

- **Function**: Set counterclockwise error range for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `value` (`int`) 0 ~ 16
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**:
  - 0 - Failed
  - 1 - Success

16.27 `get_hand_gripper_counterclockwise(joint_id, gripper_id=14)`

- **Function**: Get counterclockwise error range for a joint.
- **Parameter**:
  - `joint_id` (`int`) 1 ~ 6
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: `value` (`int`) 0 ~ 16

16.28 `set_hand_gripper_pinch_action(pinch_pose, rank_mode, idle_flag=False, gripper_id=14)`

- **Function**: Set the coordinated pinch action and speed.
- **Parameter**:

- `pinch_pose` (`int`) 0 ~ 4
  - 0: All joints return to zero
  - 1: Index finger and thumb pinch
  - 2: Middle finger and thumb pinch
  - 3: Index and middle finger pinch
  - 4: Three fingers pinch, If pinch_pose is 4, rank_mode ranges from 1 to 20
- `rank_mode` (`int`) 0 ~ 5
- `idle_flag` (`bool` , optional): default False
- `gripper_id` (`int`) 1 ~ 254, default 14

- **Return value**:
  - 0 - Failed
  - 1 - Success

16.29 `get_hand_gripper_type(gripper_id=14)`

- **Function**: Get the machine model.
- **Parameter**:
  - `gripper_id` (`int`) 1 ~ 254, default 14
- **Return value**: (`int`)
  - 0 - left hand
  - 1 - right hand

MyCobot 320 Socket

> Note: The robotic arm that uses this class of premise has a server and has been turned on.

Use TCP/IP to control the robotic arm

1. Client

```
# demo
from pymycobot import MyCobot320Socket
# Port 9000 is used by default
mc = MyCobot320Socket("192.168.10.10",9000)


res = mc.get_angles()
print(res)


mc.send_angles([0,0,0,0,0,0],20)
...
```

2. Server

Please see the TCP/IP section for details.

TCP/IP

TCP/IP, or Transmission Control Protocol/Internet Protocol, is one of the most fundemental communicative protocol on Internet, which stipulates the standard and methods of the Internet communication. Users can control robotic arms remotely through connecting with IP address instead of the USB port.

In this chapter, myCobot 280 M5 is used as an example for explanation.

Make sure that M5Stack-basic and Atom are both burnt before using.

Connection

1.1 WIFI

Follow the steps below.

Step 1: Press WLAN Server, and "WIFI Connecting" appears, signaling that the robotic arm is connecting with WIFI.



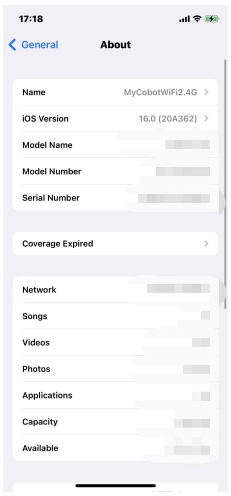


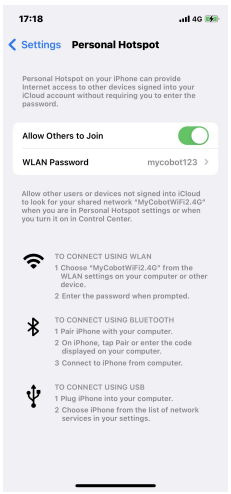
Step 2: If the screen shows WIFI Connected, IP and Port, it means that robotic arm is successfully connected with WIFI.



1.2 Mobile Network

Step 1: It is required to change the name of hotspot on your smartphone into the same as that on the robotic arm, namely "MyCobotWiFi2.4G". Meanwhile, change the password of hotspot as "mycobot123".





Step 2: Press WLAN Server to connect with mobile network.



> **Notice:** If it fails to connect, go back to press USB UART and then try the steps above again.

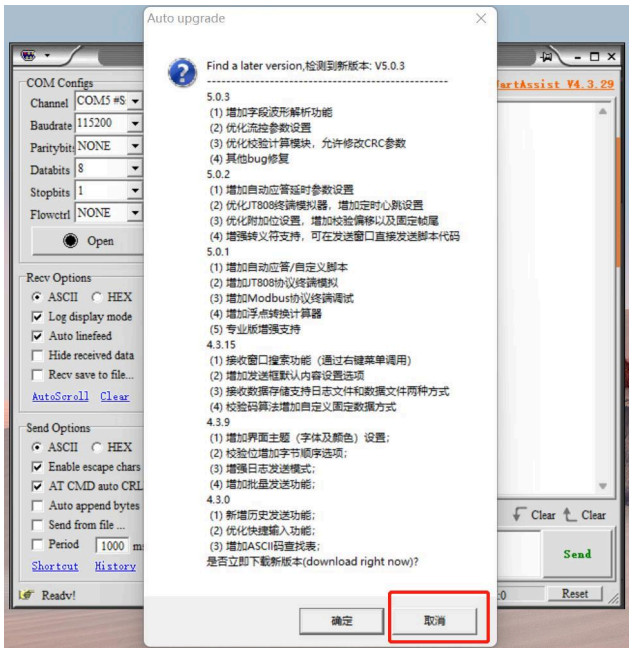
1.3 Solution to Failed Connection

Step 1: Update the version of Atom and M5Stakc-basic to the minirobot v2.0 and atommain v4.1.

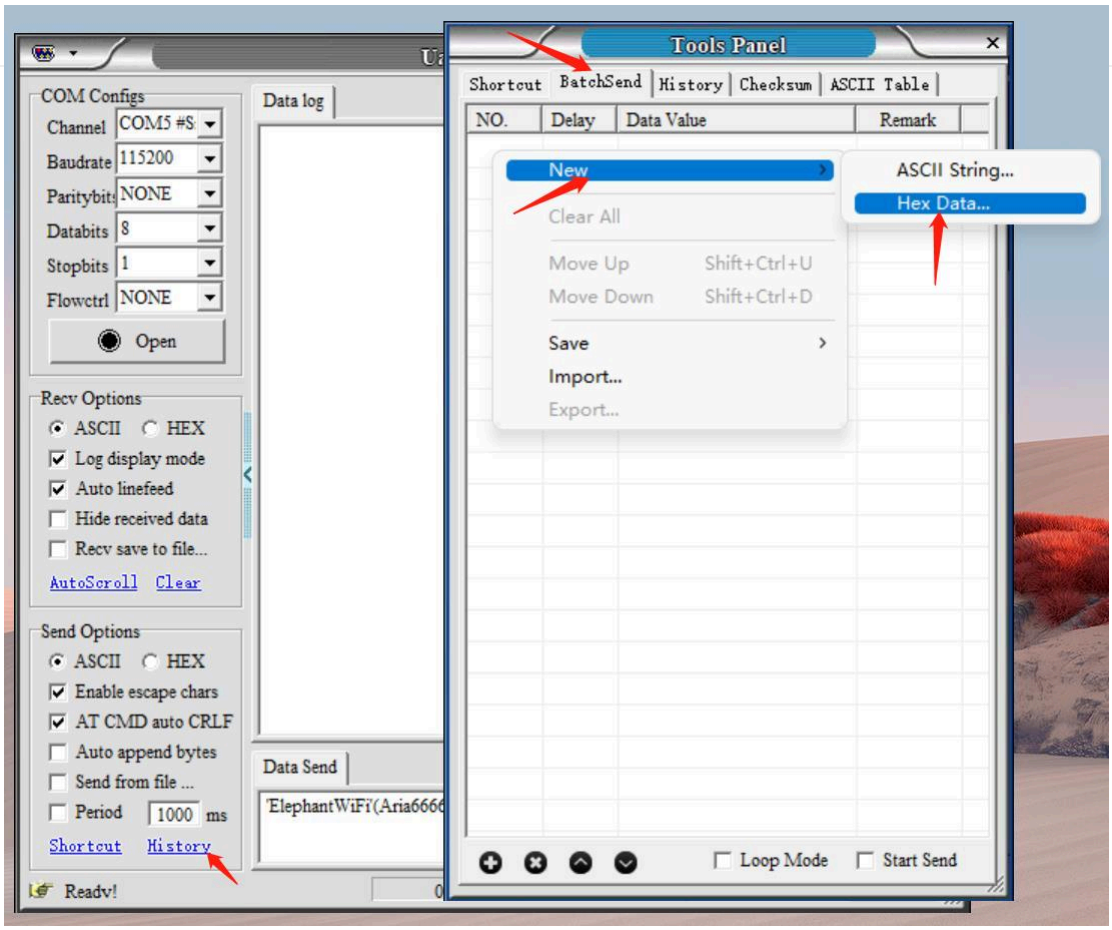
Step 2: Click on the icon to open the software UartAssist.exe.



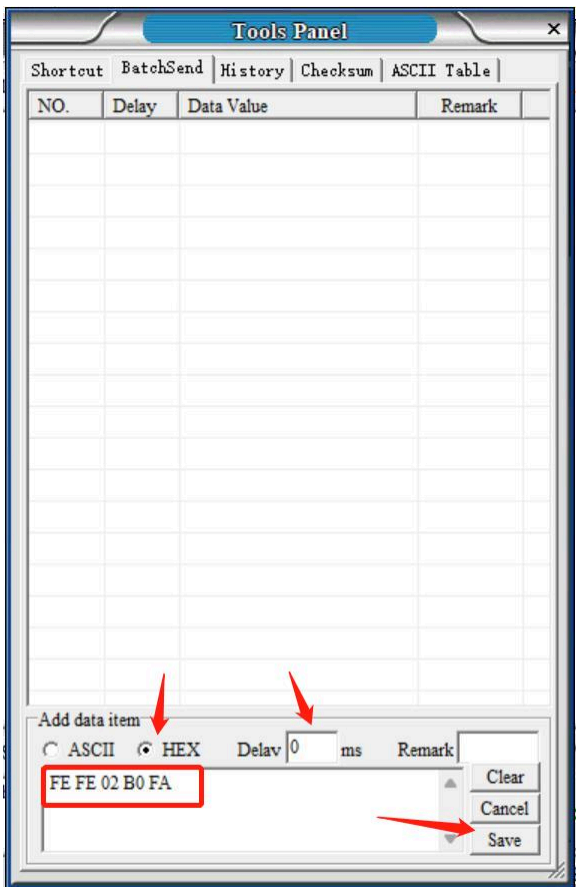
Step 3: Click on "取消".



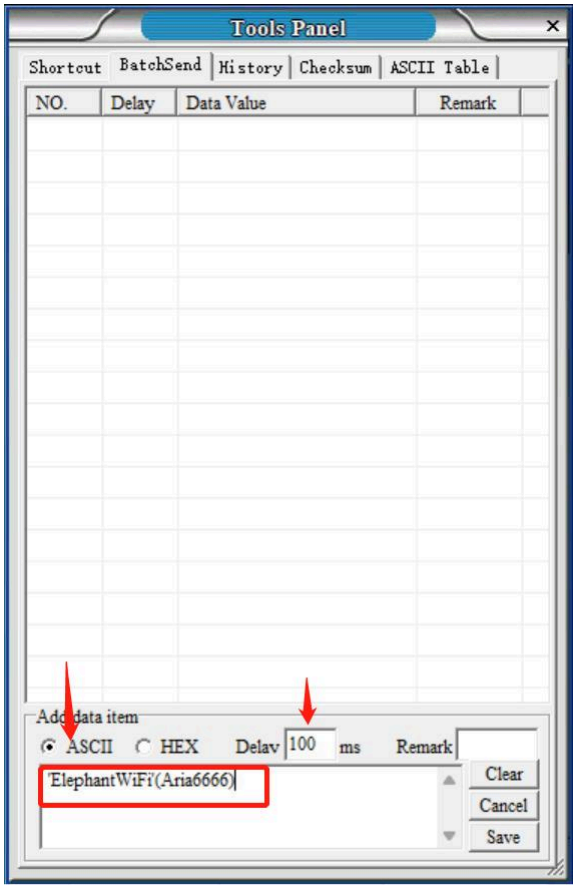
Step 4: Click on History, and then BatchSend. After that, right-click in the white space, and then New to add a data item.



Step 5: Add a data item according to the settings shown in the figure below. And then, save that item.

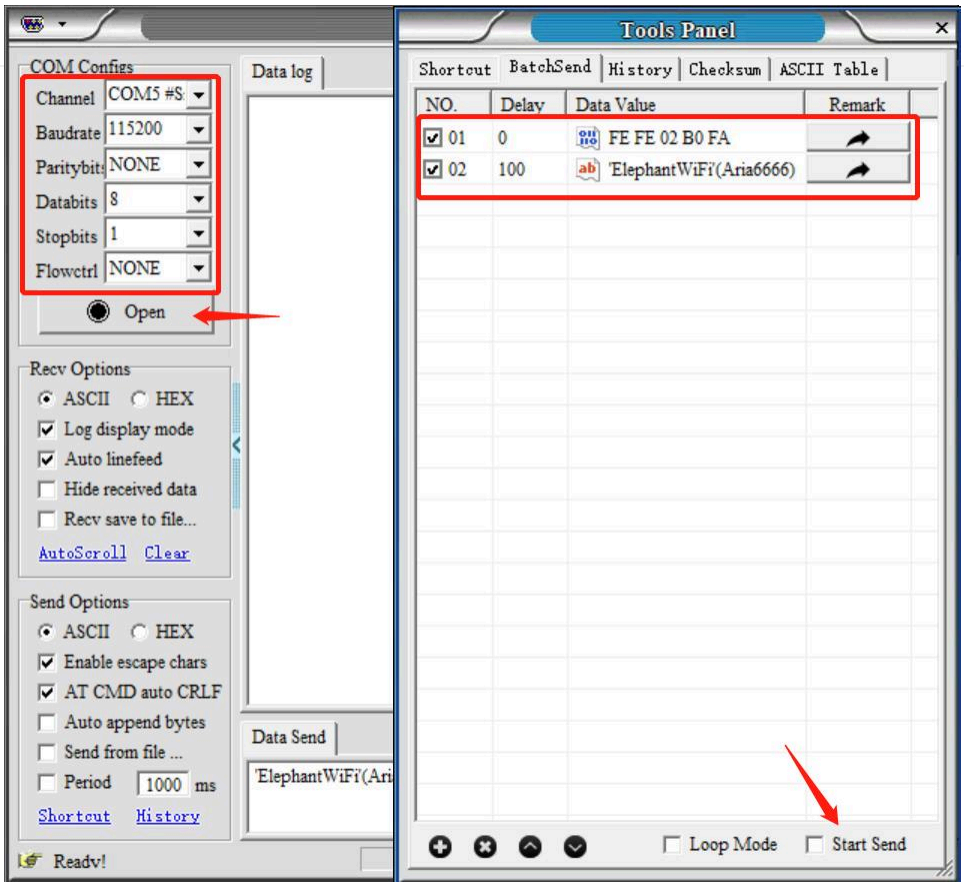


Step 6: Continue to add item in accordance with the settings shown in the figure below. Remember to type your personal WIFI name and password into the blank box. Make sure that WIFI name is surrounded by quotation mark (English) and password is surround by bracket (English). And then save it.

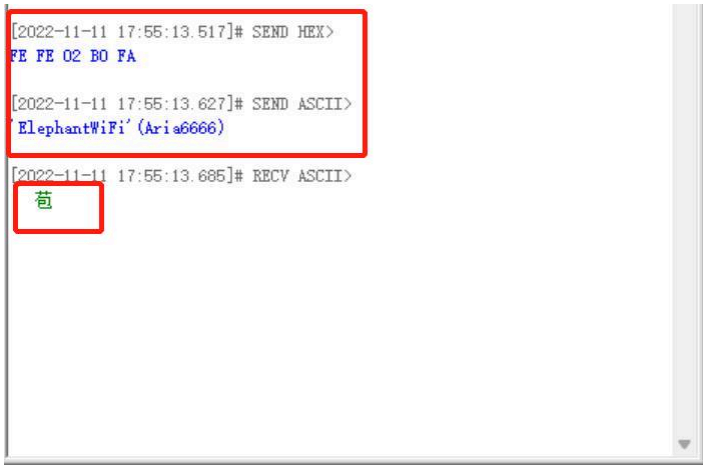


Step 6: Press `Transponder` on the M5Stack-basic and then press USB UART.

Step 7: Set COM configurations, and then click on "Open". After that, click on "Start Send".



Step 8: After sending the data items, there is a response.



Step 9: Press WLAN Server on the M5Stack-basic to connect successfully.



← Previous Page | Next Page →

Videos and Codes for Display

Videos given below are for reference.

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

Please make sure the robot power on.

1 Controlling RGB Light Panel

```
from pymycobot.mycobot320 import MyCobot320
import time


# The above needs to be written at the beginning of the code, which means importing the project package


# MyCobot320 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate:
#       M5 version is: 115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MyCobot320("/dev/ttyUSB0", 115200)
#          or mc = MyCobot320("/dev/ttyACM0", 115200)
#           windows:
#              mc = MyCobot320("COM3", 115200)


# Initialize a MyCobot320 object
# Create object code here for windows version
mc = MyCobot320("COM13", 115200)

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
```



2 Controlling Arms to Move Them to Starting Point

```
from pymycobot.mycobot320 import MyCobot320


mc = MyCobot320("COM3", 115200)


# Check whether the program can be burned into the robot arm
if mc.is_controller_connected() != 1:
    print("Please connect the robot arm correctly for program writing")
    exit(0)


# Fine-tune the robotic arm to ensure that all the bayonets are aligned in the adjusted position
# Subject to the alignment of the mechanical arm bayonet, this is only a case
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
```



3 Single-Joint Motion

```python
from pymycobot.mycobot320 import MyCobot320
import time

mc = MyCobot320('COM3', 115200)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 40)
time.sleep(3)

# Control joint 3 to move 70°
mc.send_angle(3, 70, 40)
time.sleep(3)

# Control joint 4 movement -70°
mc.send_angle(4, -70, 40)
time.sleep(3)

# Control joint 1 to move 90°
mc.send_angle(1, 90, 40)
time.sleep(3)

# Control joint 5 movement -90°
mc.send_angle(5, -90, 40)
time.sleep(3)

# Control joint 5 to move 90°
mc.send_angle(5, 90, 40)
time.sleep(3)
```



4 Multi-Joint Motion

```python
import time
from pymycobot import MyCobot320

# MyCobot320 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate:115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MyCobot320("/dev/ttyUSB0", 115200)
#           windows:
#              mc = MyCobot320("COM3", 115200)
#
#
# Initialize a MyCobot320 object
mc = MyCobot320('COM3', 115200)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
time.sleep(2.5)

# Control different angles of rotation of multiple joints
mc.send_angles([90, 45, -90, 90, -90, 90], 50)
time.sleep(2.5)

# Return the robotic arm to zero
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
time.sleep(2.5)

# Control different angles of rotation of multiple joints
mc.send_angles([-90, -45, 90, -90, 90, -90], 50)
time.sleep(2.5)
```



5 Coordinate control

```
from pymycobot.mycobot320 import MyCobot320
import time

# MyCobot320 class initialization requires two parameters:
# The first one is the serial port string, such as:
      # linux: "/dev/ttyUSB0"
      # windows: "COM3"
      # The second is the baud rate:
      # M5 version is: 115200
        #   like:
        # mycobot-M5:
          # linux:
          # mc = MyCobot320("/dev/ttyUSB0", 115200)
          # windows:
          # mc = MyCobot320("COM3", 115200)

# Initialize a MyCobot320 object
# Create the object code below for the windows version
mc = MyCobot320("COM3", 115200)

# Get the coordinates and posture of the current head
coords = mc.get_coords()
print(coords)

# Send angular motion to a certain attitude for coordinate control, with a speed of 50
mc.send_angles([0, 0, -90, 0, 90, 0], 50)
time.sleep(2.5)

# Intelligent planning of the route, allowing the head to reach the coordinates of [-2.3, -153.2, 400.8] in a linear manne
mc.send_coords([-2.3, -153.2, 400.8, -120.0, 0.7, 179.73], 40, 1)

# Set the waiting time to 1.5 seconds
time.sleep(1.5)

# Intelligent planning of the route, allowing the head to reach the coordinates of [0.0, -120.4, 500.3] in a linear manner
mc.send_coords([0.0, -120.4, 500.3, -70.81, -22.17, -163.49], 50, 1)

# Set the waiting time to 1.5 seconds
time.sleep(1.5)

# Only change the x-coordinate of the head, setting the x-coordinate of the head to 100. Let it intelligently plan the rou
mc.send_coord(Coord.X.value, 100, 70)
```

6 Control the robot arm to swing left and right

```
from pymycobot.mycobot320 import MyCobot320


import time
# Initialize a MyCobot320 object

# M5 version
mc = MyCobot320("COM3", 115200)
# Get the coordinates of the current position
angle_datas = mc.get_angles()
print(angle_datas)

# Use a sequence to pass the coordinate parameters and let the robot move 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 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)

# 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. You can manually swing the robot, and then use the get_angles() function to get the coordinate sequen
# Use this function to make the robot reach the position you want.
mc.send_angles([88.68, -135, 145, -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)
# Let the robot arm relax and you can swing it manually
mc.release_all_servos()
```



7 Controlling the robotic arm to dance

```
from pymycobot.mycobot320 import MyCobot320
import time


if __name__ == '__main__':
    # MyCobot320 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:
        # M5 version: 115200

    # Example:
        # mycobot-M5:
            # Linux:
              # mc = MyCobot320("/dev/ttyUSB0", 115200)
            # Windows:
              # mc = MyCobot320("COM3", 115200)

    # Initialize a MyCobot320 object
    # M5 version
    mc = MyCobot320("COM3",115200)

    # Set the start time
    start = time.time()
    # Let the robot reach the specified position
    mc.send_angles([-1.49, 115, -145, 30, -33.42, 137.9], 80)
    # Determine whether it has reached the specified position
    while not mc.is_in_position([-1.49, 115, -145, 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 exercise last for 30 seconds
    while time.time() - start < 30:
        # Let the robot quickly reach this position
        mc.send_angles([-1.49, 115, -145, 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 quickly reach this position
```

```
        mc.send_angles([-1.49, 55, -145, 80, 33.42, 137.9], 80)
        # Set the color of the light to [0,50,0]
        mc.set_color(0, 50, 0)
        time.sleep(0.7)
```



8 Controlling Gripper

```python
from pymycobot.mycobot320 import MyCobot320
import time


# MyCobot320 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyACM0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#    such as:
#       mycobot-M5:
#           linux:
#              mc = MyCobot320("/dev/ttyUSB0", 115200)
#          or mc = MyCobot320("/dev/ttyACM0", 115200)
#           windows:
#              mc = MyCobot320("COM3", 115200)

# Initialize a MyCobot320 object
mc = MyCobot320("COM13", 115200)
# make it move to zero position
mc.send_angles([0, 0, 0, 0, 0, 0], 40)
time.sleep(3)
# Pro adaptive gripper needs to be set to transparent transmission mode before use
mc.set_gripper_mode(0)
time.sleep(1.5)

num = 5
while num > 0:
    # Set the state of the gripper to quickly open the gripper at a speed of 70, Parameter 1 represents pro adaptive gripp
    mc.set_gripper_state(0, 70, 1)
    time.sleep(3)
    # Set the state of the gripper to quickly close the gripper at a speed of 70, Parameter 1 represents pro adaptive grip
    mc.set_gripper_state(1, 70, 1)
    time.sleep(3)
    num -= 1

num = 3
while num > 0:
    # Set the gripper value to 0 and the speed to 70, Parameter 1 represents pro adaptive gripper type
    mc.set_gripper_value(0, 70, 1)
    time.sleep(3)
    # Set the gripper value to 50 and the speed to 70, Parameter 1 represents pro adaptive gripper type
    mc.set_gripper_value(50, 70, 1)
    time.sleep(3)
    # Set the gripper value to 0 and the speed to 70, Parameter 1 represents pro adaptive gripper type
```

```
    mc.set_gripper_value(0, 70, 1)
    time.sleep(3)
    num -= 1

# Get the value of the gripper
print("gripper value:", mc.get_gripper_value())
```



← Previous Page | Next Section →

ROS

ROS is an open-source meta-operating system used for robots. It provides an operating system with expected services, including hardware abstraction, low-level device control, implementation of common functions, messages transfered between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run codes across computers.

The "graph" of ROS runtime is a loosely coupled peer-to-peer process network based on a ROS communication infrastructure. ROS implements several different communication methods, including a services mechanism based on synchronous RPC-style communication, a topics mechanism based on asynchronous streaming media data, and a parameter server for data storage.

ROS is not a real-time framework, but it 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 may also be seamlessly integrated with Orocos real-time toolkits.

ROS Logo：



1 Design goals and characteristics of ROS

Many people ask "what are differences between ROS and other robot software platforms?" This question is difficult to answer. Because ROS is not a framework that integrates most functions or features. In fact, the main goal of ROS is to provide code reuse support for robot R&D. ROS is a framework (i.e. nodes) for distributed processes, which are encapsulated in program and function packages that can be easily shared and distributed. ROS also supports a federated system similar to a code repository, and this system also enables the collaboration and distribution of a project. This design enables the development and realization of a project to to be decided completely independently from the file system to the user interface (no limit by ROS). At the same time, all projects can be integrated with the basic tools of ROS.

To support the primary goals of sharing and collaboration, the ROS framework has several other features:

- Lean: ROS is designed to be as lean as possible so that codes written for ROS can be used with other robot software frameworks. The inevitable conclusion from this is that ROS can be easily integrated into other robot software platforms: ROS can already be integrated with OpenRAVE, Orocos and Player.
- ROS insensitive libraries: The preferred development model of ROS is written with clean library functions that do not depend on ROS.
- Language independence: The ROS framework can simply be implemented in any modern programming language. ros has implemented Python version, C++ version and Lisp version. It also has experimental libraries for Java and Lua versions.
- Loose coupling: The function modules in ROS are encapsulated in independent function packages or meta-function packages, which are easy to share. The modules in the function package are run in units of nodes. With ROS standard IO used as the interface, developers does not need to pay attention to the internal implementation of the module, as long as they understand the interface rules, they can achieve reuse and point-to-point loose coupling between modules.
- Convenient testing: ROS has a built-in unit/integration testing framework called rostest, which can easily install or uninstall test modules.
- Scalable: ROS is applicable to large runtime systems and large development processes.
- Free and open-source: It has many developers and many function packages.

2 Why ROS is used?

Through ROS, we can realize the simulated control of robot arms in a virtual environment.

we will visualize robot arms through **rviz**, operates our robot arms in a variety of ways, and plan and executes the action path of robot arms through **MoveIt** to achieve the effect of freely controlling the robot arms.

We will learn how to control our products through the platform in ros in the following chapters.

MoveIt

MoveIt is currently the most advanced software for movement operations of robot arms and has been used for more than 100 robots. It integrates the latest achievements in motion planning, control, 3D perception, motion control, control and navigation, provides an easy-to-use platform for developing advanced robot applications, and provides an integrated software platform for design, and integration assessment of new robot products in the fields of industry, commerce, R&D, etc.

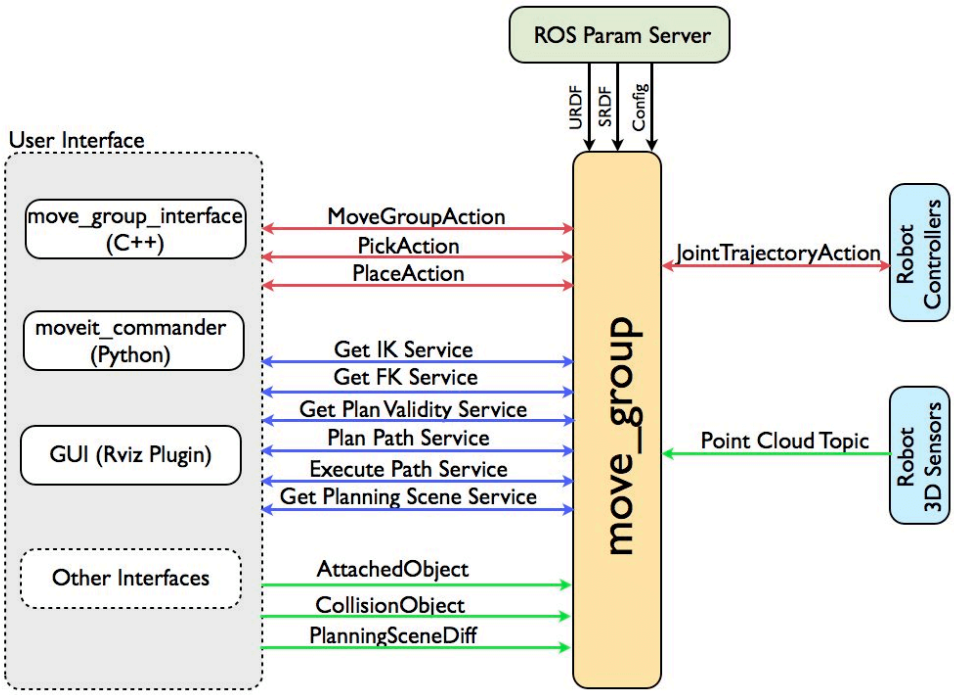
MoveIt Logo :



1 Introduction

MoveIt is an integrated development platform in ROS, which consists of a variety of functional packages for manipulating robot arms, including motion planning, operation, control, inverse kinematics, 3D perception, collision detection, etc.

The following figure shows the high-level structure of the main node **move_group** provided by Moveit. It is like a combiner: all the individual components are integrated together, providing a series of actions and services for users to use.



2 User interface

The user may access the operations and services provided by move_group in three ways:

- In C++, you may use move_group easily by using move_group_interface package.
- In Python, use the moveit_commander package.
- Via GUI: use Rviz (ROS visualization tool) of Motion-commander.

move_group can be configured using the ROS parameter server, from which the robot's URDF and SRDF can also be obtained.

3 Configuration

move_group is a ROS node. It uses the ROS parameter server to obtain three kinds of information:

- URDF - move_group looks for the robot_description parameter in the ROS parameter server to get the robot's URDF.
- SRDF - move_group looks for the robot_description_semantic parameter in the ROS parameter server to get the robot's SRDF. SRDF is typically created by the user using an MoveIt Setup Assistant.
- MoveIt configuration - move_group will look in the ROS parameter server for additional MoveIt-specific configurations, including joint constraint, kinematics, motion planning, perception, and other information. w The configuration files for these components are automatically generated by the MoveIt Setup Assistant and stored in the configuration directory of the robot's corresponding MoveIt configuration package. For the use of configuration assistant, please refer to: MoveIt Setup Assistant

← Previous Section | Next Page →

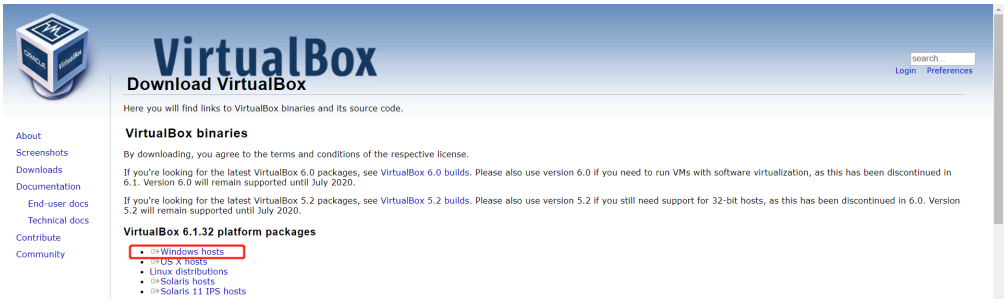
Install different versions of ubuntu system in Linux

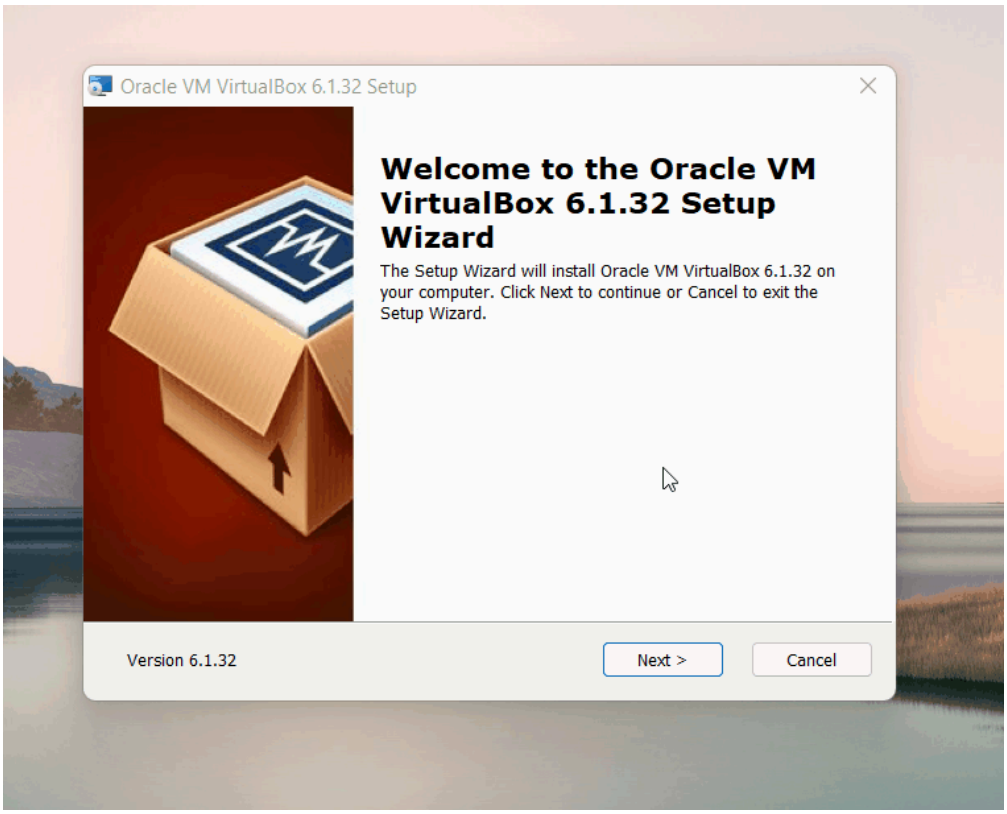
1 Virtual machine installation

Go to Official website to download the virtual machine Virtual Box or go to Official website to download the virtual machine VMware.

Of course, if you already have your virtual machine, you may skip this step.

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





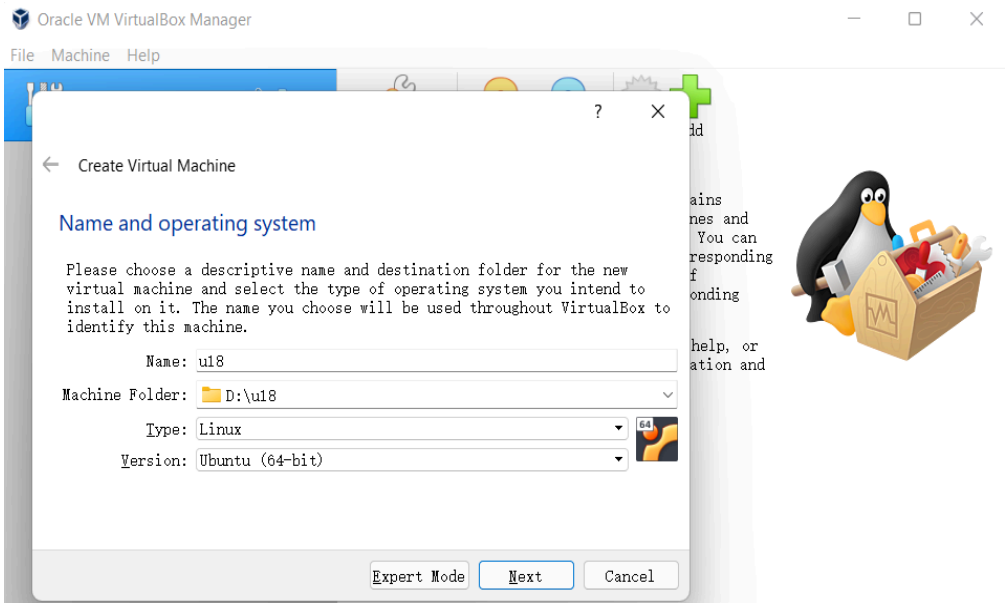
> There are two options for virtual machine creation:
>
> - ① Using our configured virtual machine ROS1 virtual machine files, imported into the virtual machine software, you can save the time to configure the environment, easy to quickly use the ROS case.
>   - ROS1 VM files:
>
>   http://download-elephantrobotics.oss-cn-shenzhen.aliyuncs.com/system_images/ubuntu20.04_ROS1_V20230731.ova.zip
>   - Step-by-step video on how to import VM files into VM software:
>
>   https://drive.google.com/file/d/1KeYk_CUgDE46rVn7zbd0EhraIbgt3qZt/view?usp=sharing
> - ② Follow the steps below to deploy the ROS environment yourself.

2 Creating a virtual machine

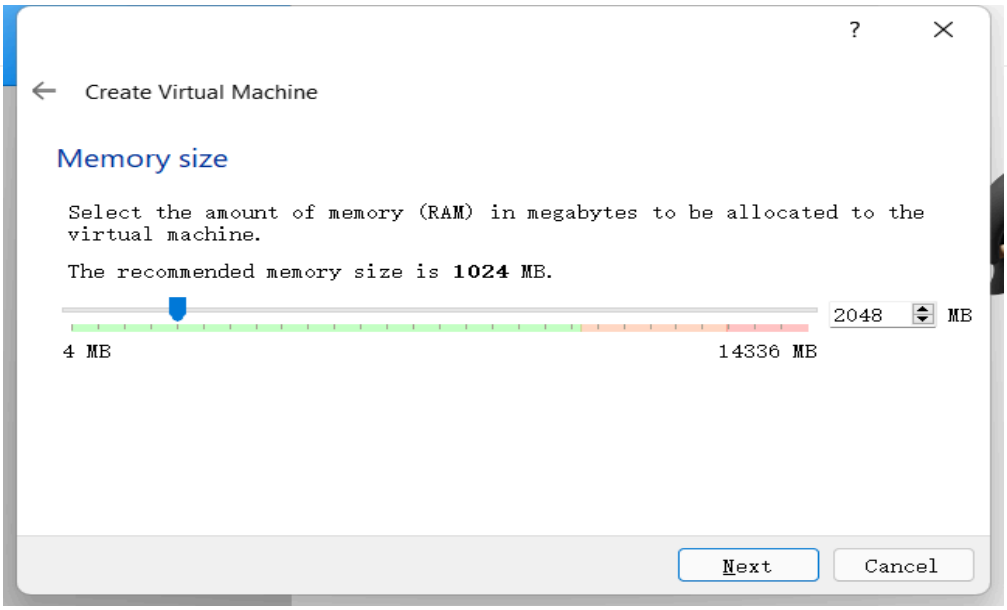
2.1 Creating a virtual machine

Select Create in control

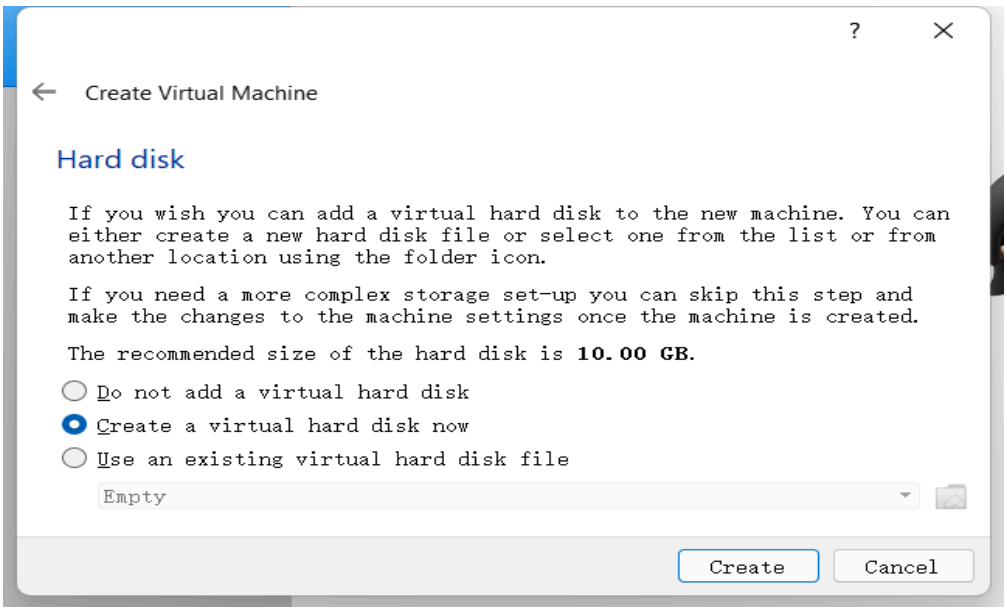
- Input the name of the virtual machine and the location where it is stored, select the type of virtual machine as Linux, select the ubuntu64-bit version, and go to the next step.



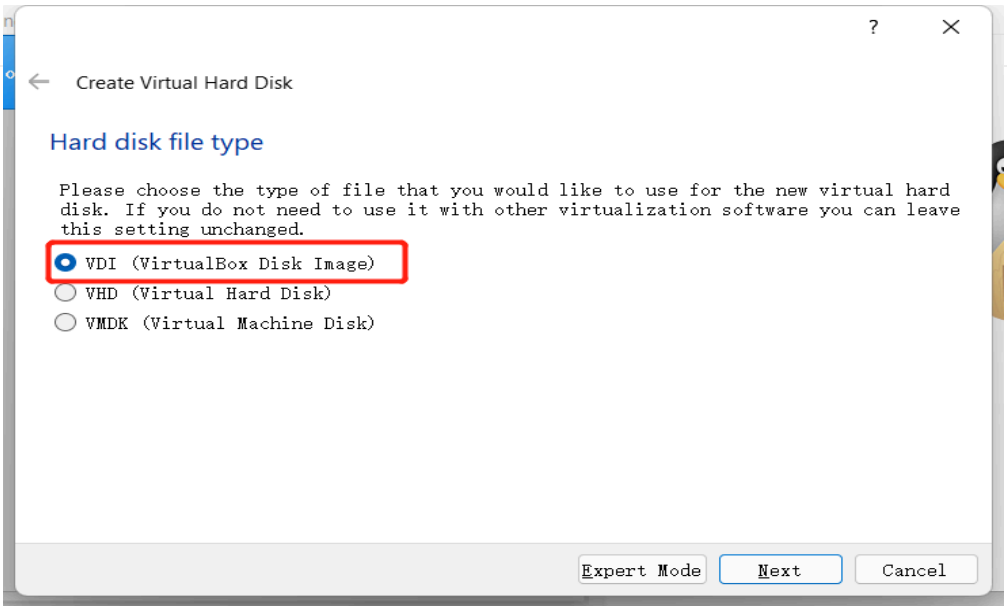
- Configure a memory size according to your own needs, and then go to the next step.



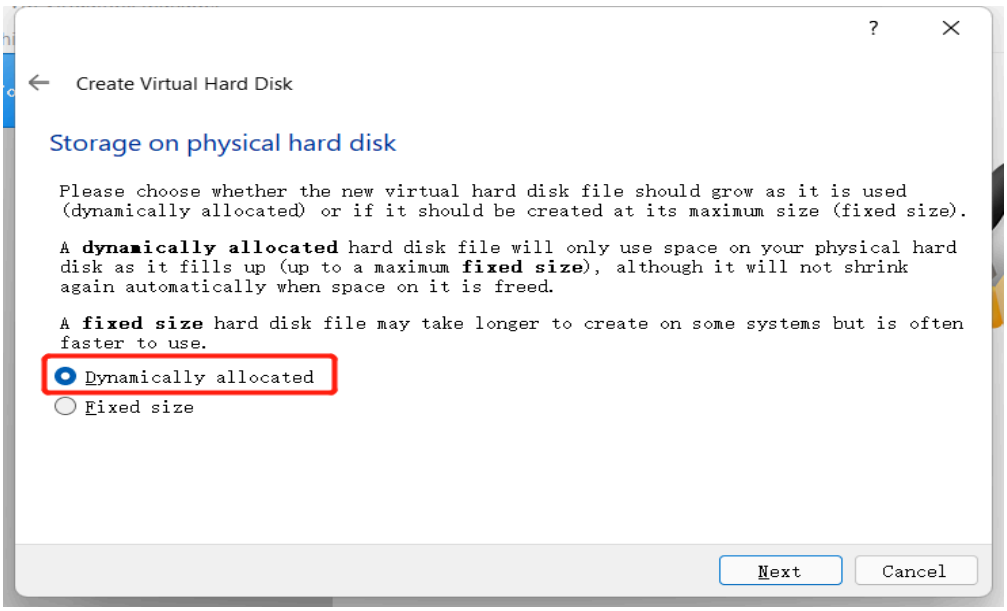
- Choose **Create a virtual hard disk now** to create it.

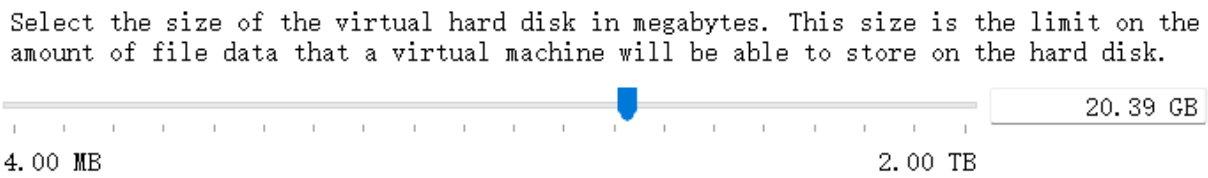


- Select the virtual hard disk type as **VDI** type, and go to the next step.



- Allocate the size of the virtual hard disk. Since an ubuntu system needs to be installed, and the operation will be performed in the system, it is recommended that the size should not be less than 20G.





2.2 Import ubuntu system

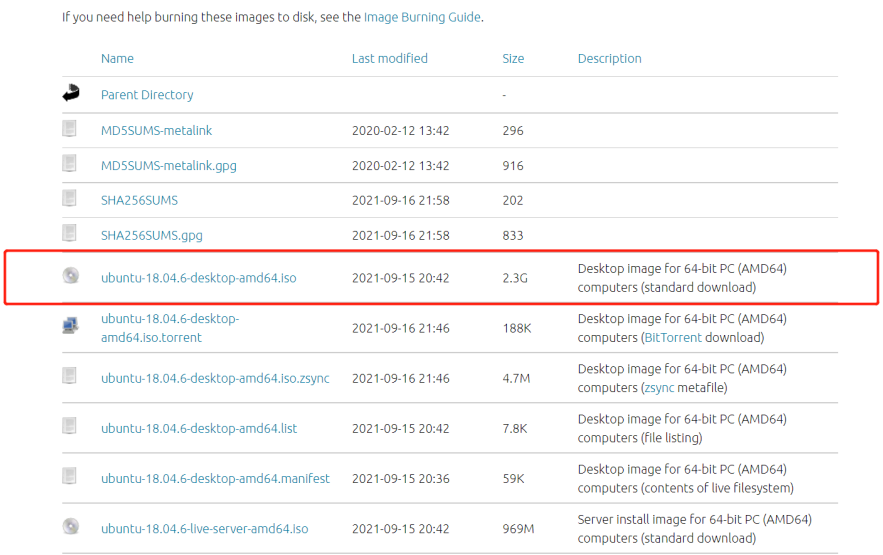
2.2.1 Downloading the ubuntu system

Select the ubuntu version according to your own needs and install it:

- Version 16.04
- Version 18.04

- Version 20.04

The installation method and process of the three versions are the same. Here, we take the version 18.04 version for an example.



If you need help burning these images to disk, see the Image Burning Guide.

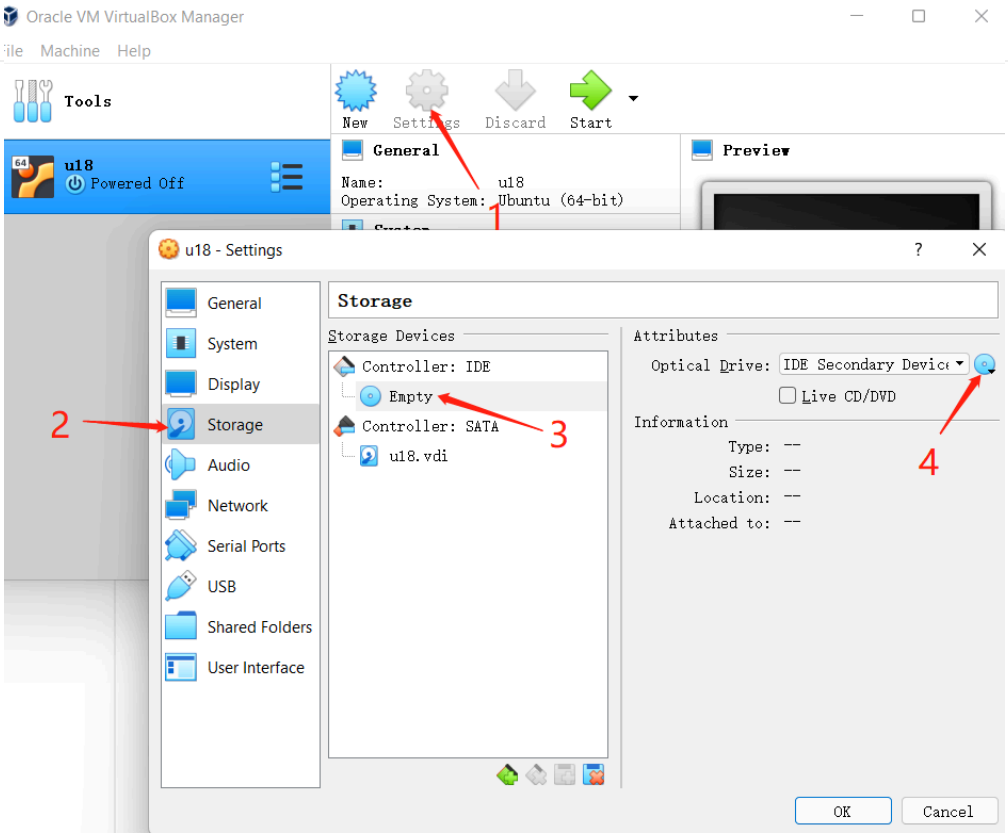
Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers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

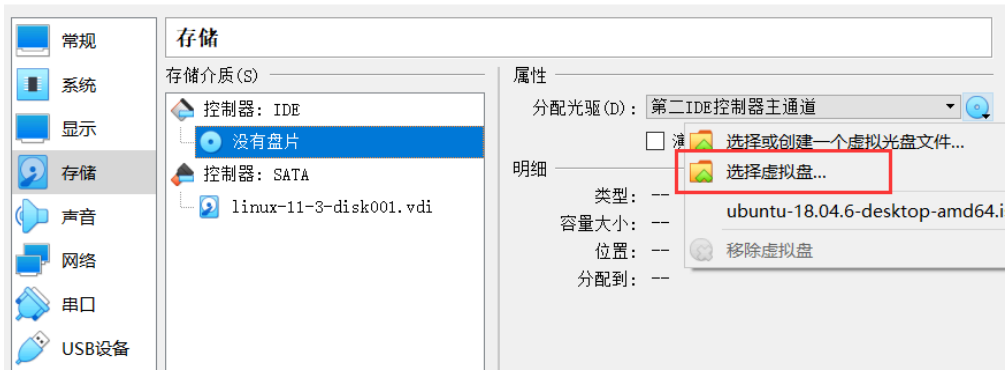
After the downloading is complete, there will be a file shown in the figure:

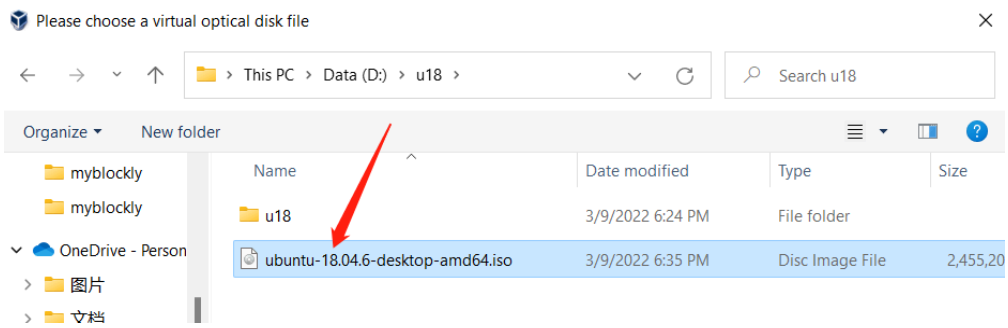


2.2.2 Importing ubuntu into a visual machine

Find the previously installed virtual machine in the Virtual box, enter **Setup**, and assign a CD to the controller in **Storage**:







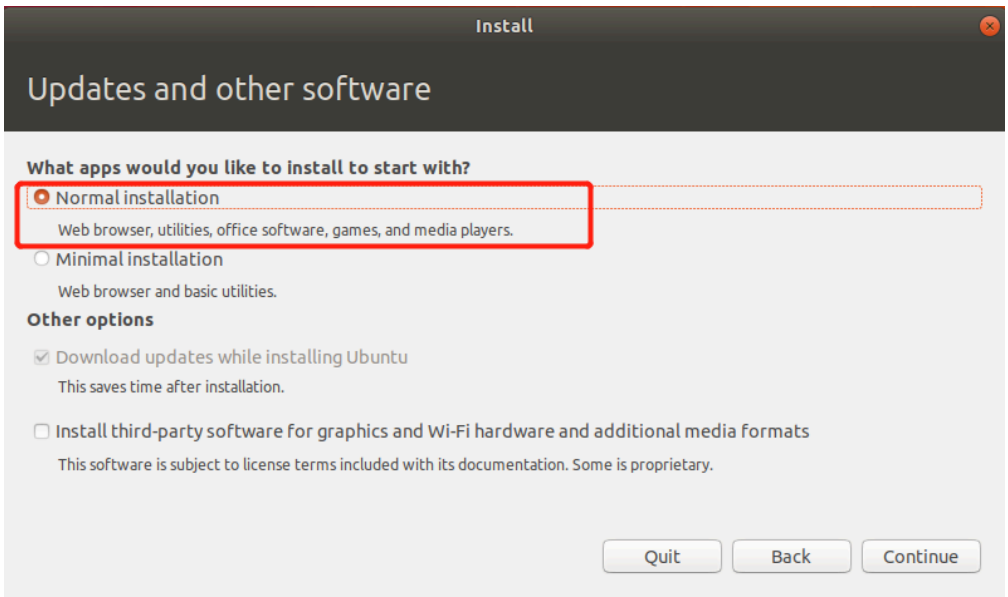
Then open the virtual machine for ubuntu installation, and click start.

2.2.3 Installing ubuntu

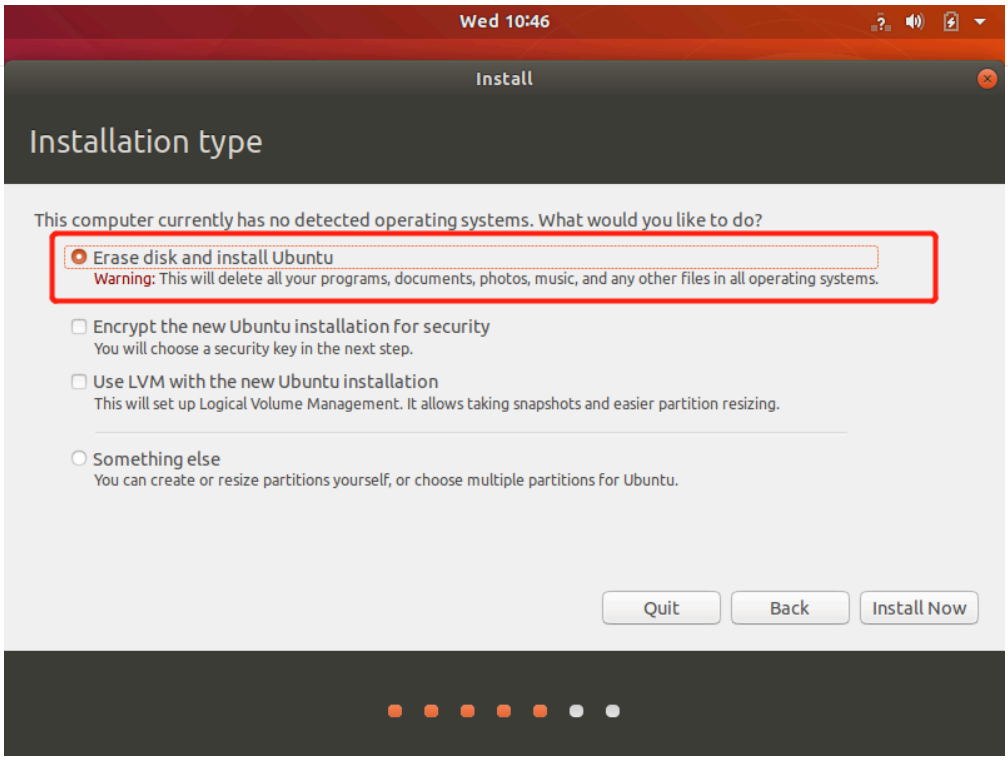
- Wait for the system to start, enter a **welcome** interface, select "Chinese (Simplified)", and click the "Install Ubuntu" button;



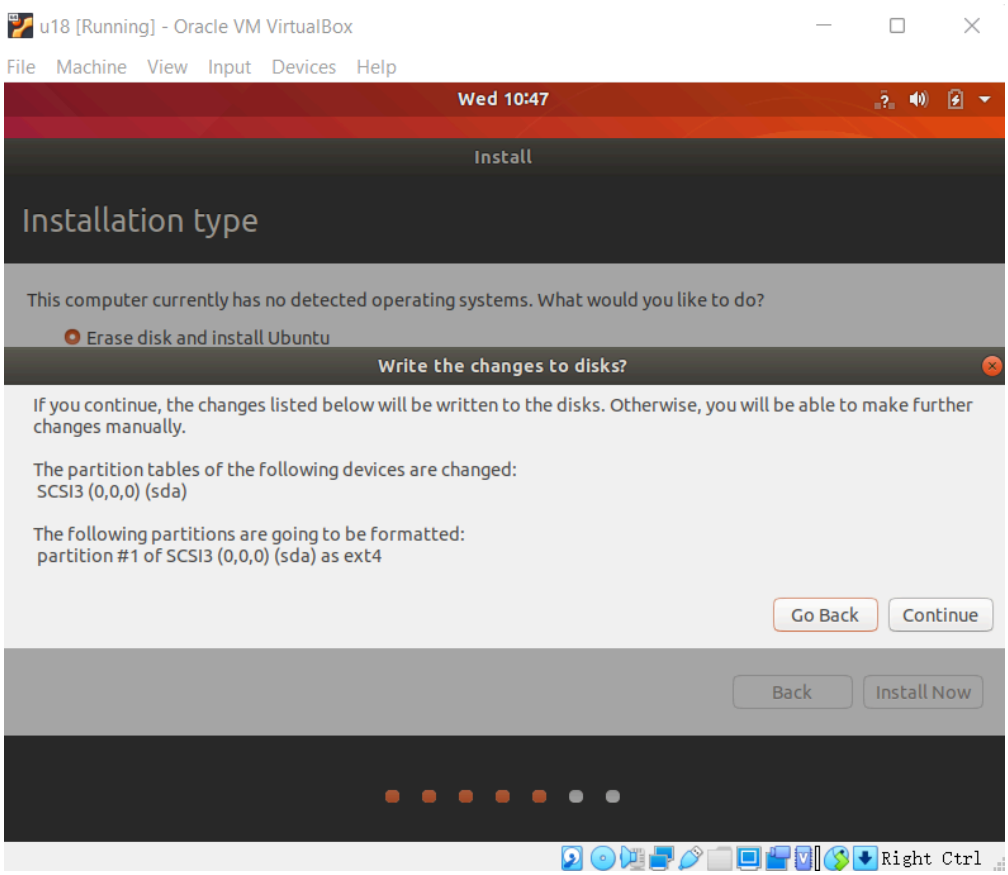
- Click “Continue” button;



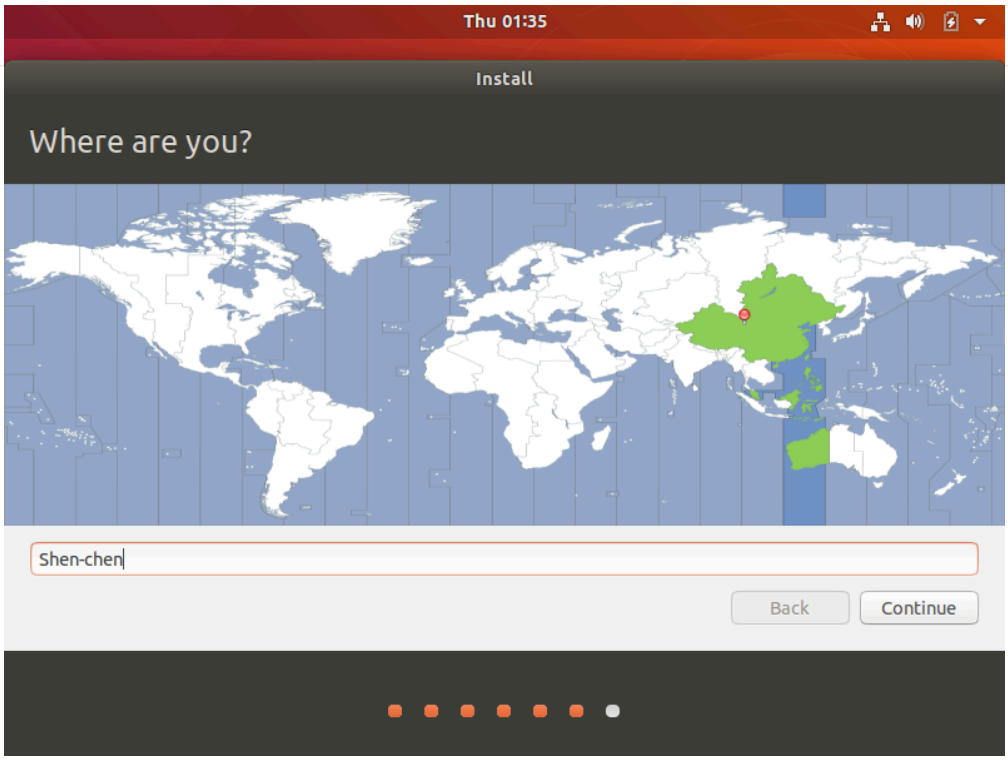
- Select the "Clear entire disk and install Ubuntu" option, and click the "Install Now" button;



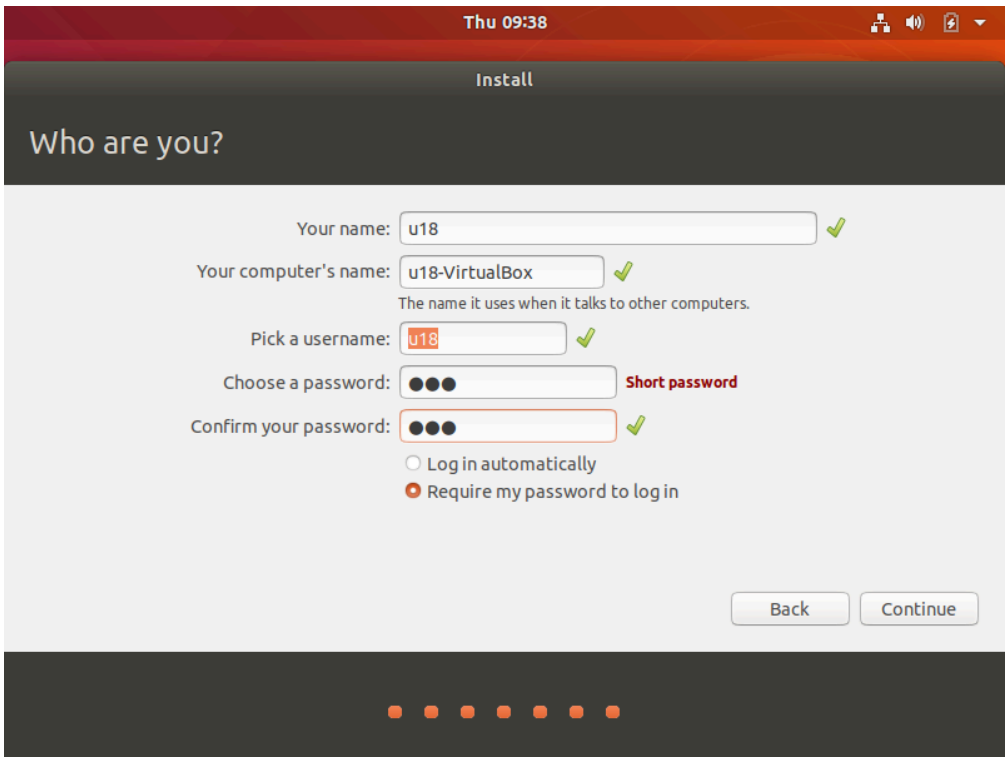
- Click the "Continue" button in the pop-up dialog box;



- Set a geographic position and click the “Continue” button;



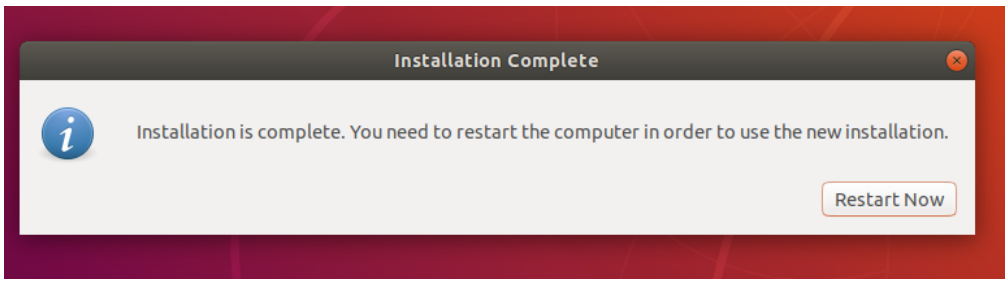
- Set a user name and password, and click the “Continue” button;



- Enter the system installation interface, and wait patiently;



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



3 ROS Environment building

3.1 Installing ROS

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

3.1.1 Selecting a version

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

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

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

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

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

3.1.2 Start installation

1 Adding a software source

There is no ROS software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

- official source:

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

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, Tsinghua University is:

```
sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ `lsb_release -cs` main" > /etc
```

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

2 Setting a key

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

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

The execution result is displayed as follows:

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

3 Installation

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

```
sudo apt-get update
```

Execute **Installing ROS**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command according to your Ubuntu version:

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

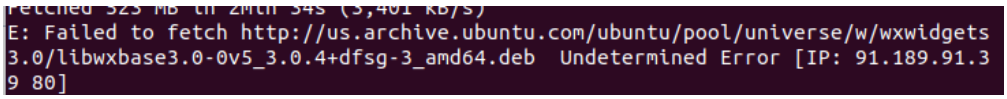
```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

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

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

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



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

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

- 将sources.list中的官方软件源全部替换成下面的阿里云软件源：

Ubuntu 16.04 version：

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

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

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

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

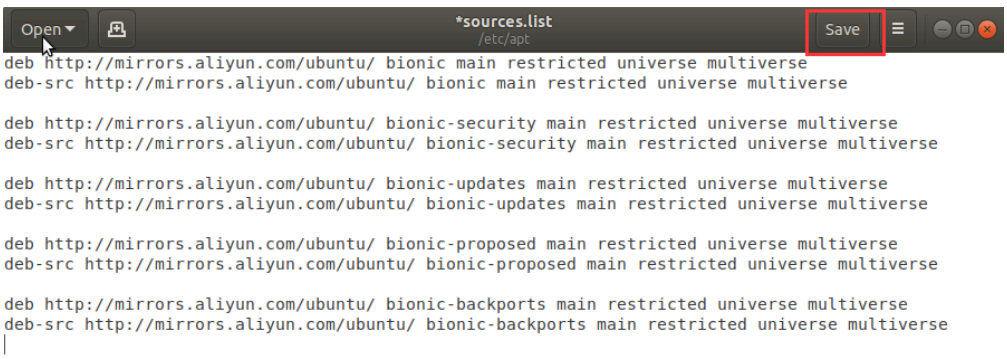
Ubuntu 18.04 version：

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

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

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

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

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

Ubuntu 20.04 version：

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

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

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

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

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

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



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

```
sudo apt-get update
```

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

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

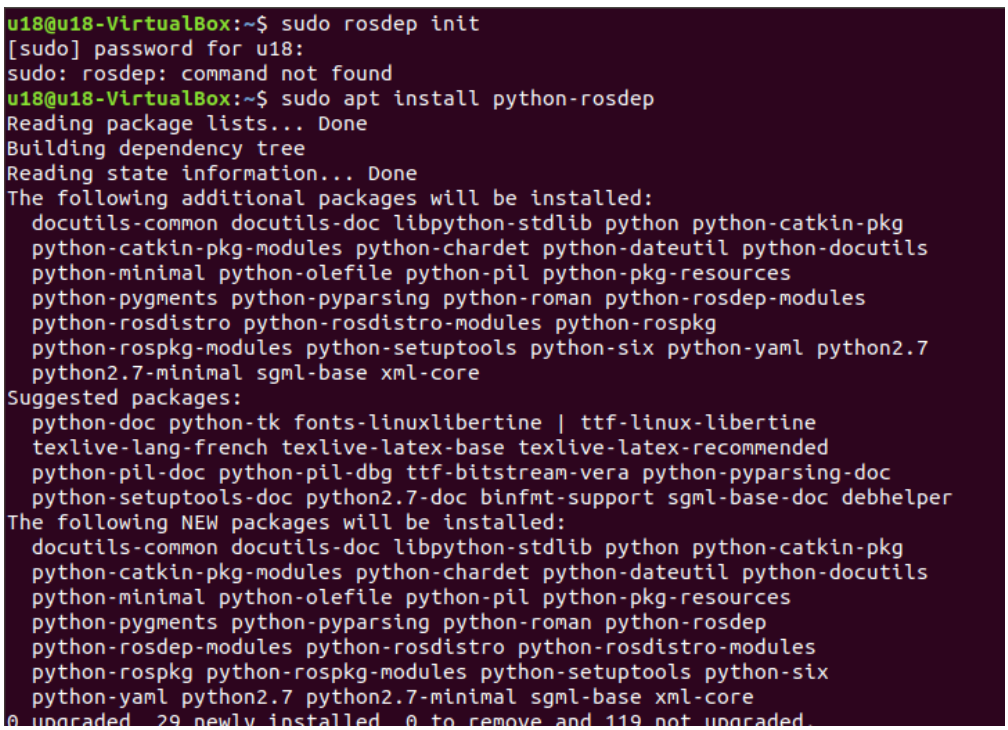
The installation process takes a long time, so please wait with patience.

4 Configuring the ROS environment to the system

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

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

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



initialize rosdep:

```
sudo rosdep init
```

If the following error message appears:

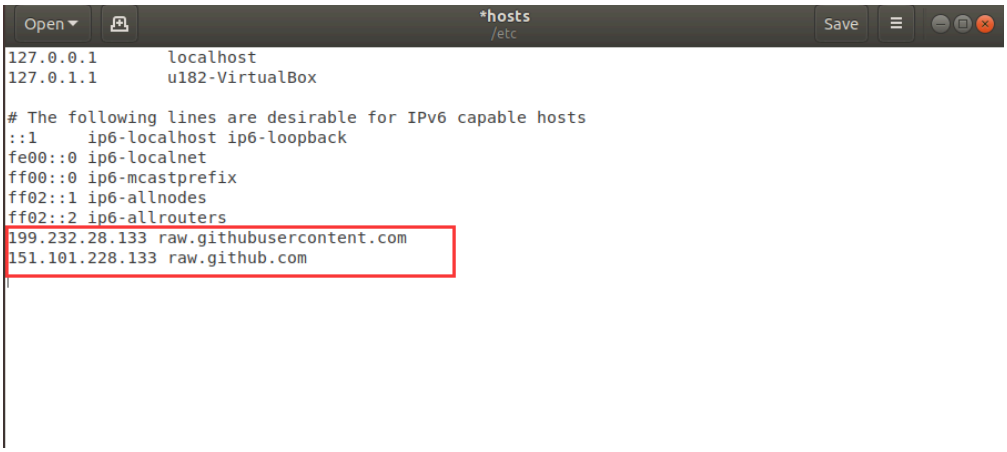
```
u182@u182-VirtualBox:~$ sudo rosdep init
ERROR: cannot download default sources list from:
https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/sources.list.d/20-
default.list
Website may be down.
```

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

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

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

```
199.232.28.133 raw.githubusercontent.com
151.101.228.133 raw.github.com
```



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

```
sudo rosdep init
```

```
rosdep update
```

After the initialization is completed, in order to avoid re-validating the ROS function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

3.1.3 Set up the ros environment

1 Bash

Execute the following commands:

```
# Ubuntu 16.04
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

2 Install dependencies

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

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

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

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

```
# Ubuntu 16.04
sudo apt install ros-kinetic-joint-state-publisher-gui
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-joint-state-publisher-gui
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-joint-state-publisher-gui
```

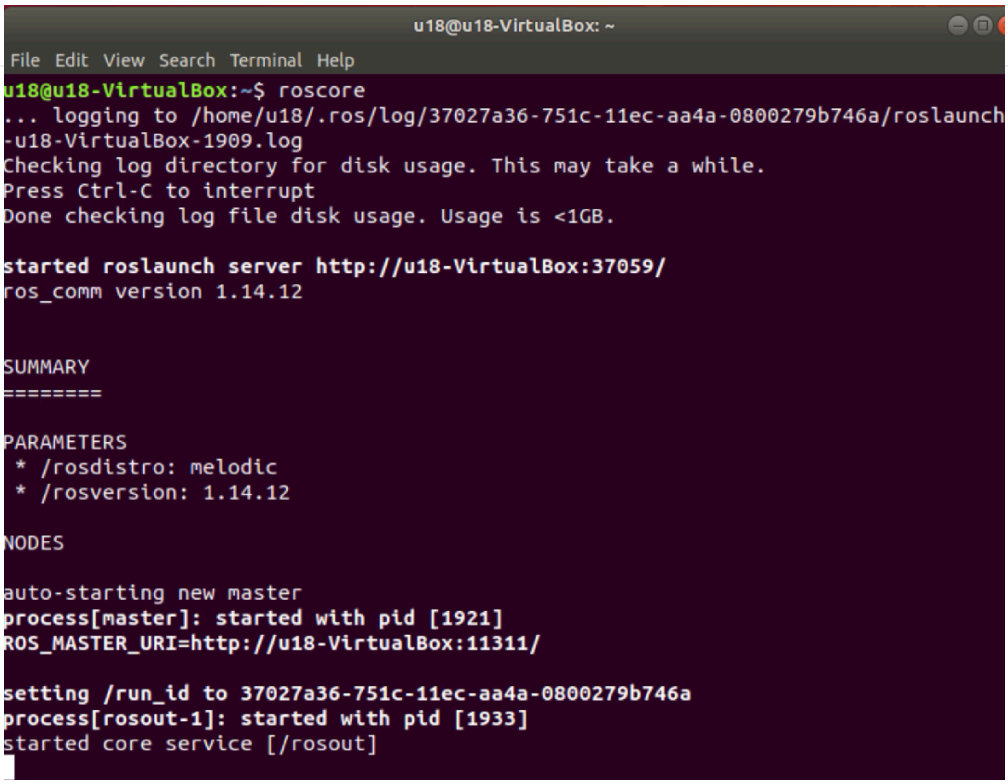
3.1.4 Verify installation

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

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

```
roscore
```

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



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

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

3.2 MoveIt Installation

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

1 Updating the software source list

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

```
sudo apt-get update
```

2 Installing MoveIt

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

```
# Ubuntu16.04
sudo apt-get install ros-kinetic-moveit
```

```
# Ubuntu 18.04
sudo apt-get install ros-melodic-moveit
```

```
# Ubuntu20.04
sudo apt-get install ros-noetic-moveit
```

3.3 Git Installation

1Adding a software source

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

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

2 Updating the software source list

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

```
sudo apt-get update
```

3 Installing git

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

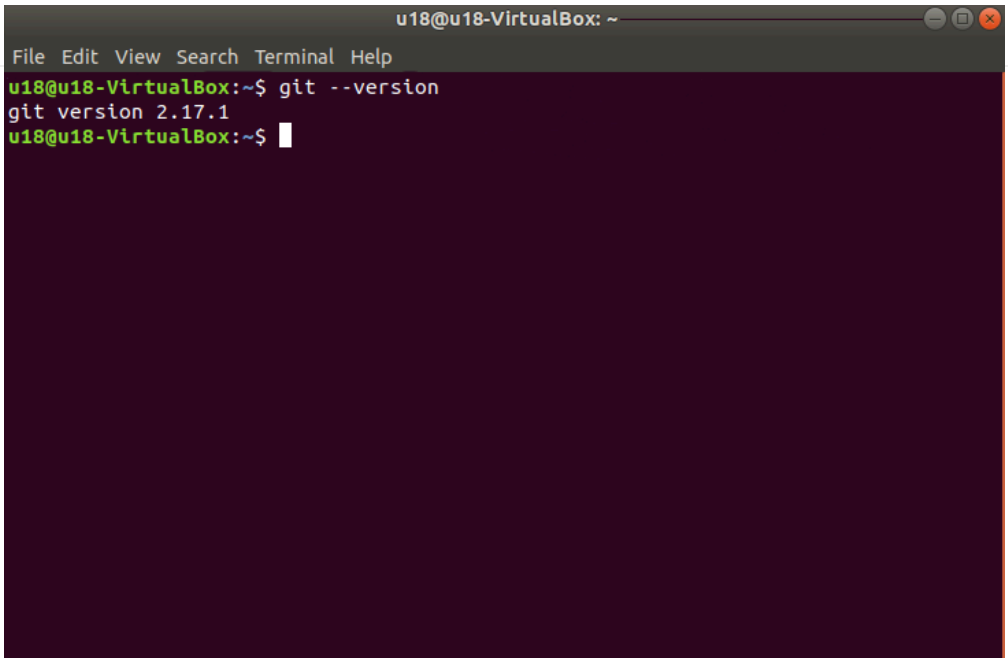
```
sudo apt-get install git
```

4 Verify installation

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

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.



5 Use

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

- https://git-scm.com/book/zh/v2
- https://www.runoob.com/git/git-tutorial.html

4.4 mycobot_ros Installation

`mycobot_ros` is a ROS package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros

1 Precondition

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

Here is an **example command for creating a workspace**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command in the command line:

```
mkdir -p ~/catkin_ws/src  # Create a folder
cd ~/catkin_ws/src        # Enter the folder
catkin_init_workspace     # Initialize the current directory into a ROS workspace
cd ..                     # Return to the parent directory
catkin_make               # Build the code in the workspace
```

Add workspace environment

The official default ROS1 workspace is `catkin_ws` .

```
# Ubuntu 16.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

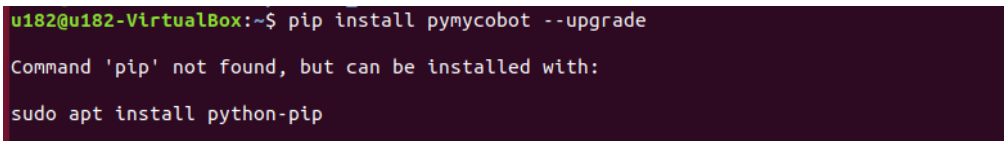
```
source ~/.bashrc
```

2 Installing the package

NOTE:

- This package depends on ROS and MoveIT. Before using it, make sure that you have installed ROS and MoveIT successfully.
- The interaction of this package with a real robot arm depends on PythonApi - `pymycobot`.
- The Api project location 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:

  Enter the following command to install pip at 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 the pip installation is complete, the terminal executes again

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

- The installation method depends on Git, so make sure that Git have been installed in your computer.

The official default ROS1 workspace is `catkin_ws`.

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

So far, the ROS1 environment construction has been completed.You can learn the basics of ROS or ROS use cases

← Previous Page | Next Page →

1 ROS project structure

1.1 catkin workspace

Catkin workspace is the directory where catkin software packages are created, modified, and compiled. Catkin's workspace can be intuitively described as a warehouse, which contains various ROS project projects to facilitate system organization, management and calling.

- **Create workspace**:

```
mkdir -p ~/catkin_ws/src  # Create a folder
cd ~/catkin_ws/src        # Enter the folder
catkin_init_workspace     # Initialize the current directory into a ROS workspace
cd ..                     # Return to the parent directory
catkin_make               # Build the code in the workspace
```

The structure of catkin is very clear. It includes three paths: src, build, and devel. It may also include others under some compilation options. But these three folders are the default for the catkin compilation system. Their specific functions are as follows:

```
src/: ROS catkin software package (source code package)

build/: cache information and intermediate files of catkin (CMake)

devel/: Generated target files (including header files, dynamic link libraries, static link libraries, executable files, e
```

A simple workspace looks like this:

```
workspace_folder/        -- WORKSPACE
  src/                   -- SOURCE SPACE
    CMakeLists.txt       -- 'Toplevel' CMake file, provided by catkin
    package_1/
      CMakeLists.txt     -- CMakeLists.txt file for package_1
      package.xml        -- Package manifest for package_1
    ...
    package_n/
      CMakeLists.txt     -- CMakeLists.txt file for package_n
      package.xml        -- Package manifest for package_n
```

1.2 ROS software package

Package is not only a software package on Linux, but also the basic unit of catkin compilation. The object we use catkin_make to compile is each ROS package.

```
+--PACKAGE
     +-- CMakeLists.txt
     +-- package.xml
     +-- src/
     +-- include/
     +-- scripts/
     +-- msg/
     +-- srv/
     +-- urdf/
     +-- launch/
```

- **CMakeLists.txt:** Defines the package name, dependencies, source files, target files and other compilation rules of the package. It is an essential component of the package.
- **package.xml:** Describes the package name, version number, author, dependencies and other information of the package, which is an indispensable component of the package.
- **src/:** stores ROS source code, including C++ source code (.cpp) and Python module (.py)
- **include/:** stores the header files corresponding to the C++ source code
- **scripts/:** stores executable scripts, such as shell scripts (.sh), Python scripts (.py)
- **msg/:** stores messages in custom format (.msg)
- **srv/:** stores services in custom formats (.srv)
- **urdf/:** stores the robot's model description (.urdf or .xacro) and 3D model files (.sda, .stl, .dae, etc.)
- **launch/:** stores launch files (.launch or .xml)

Create your own package:

- Command format:

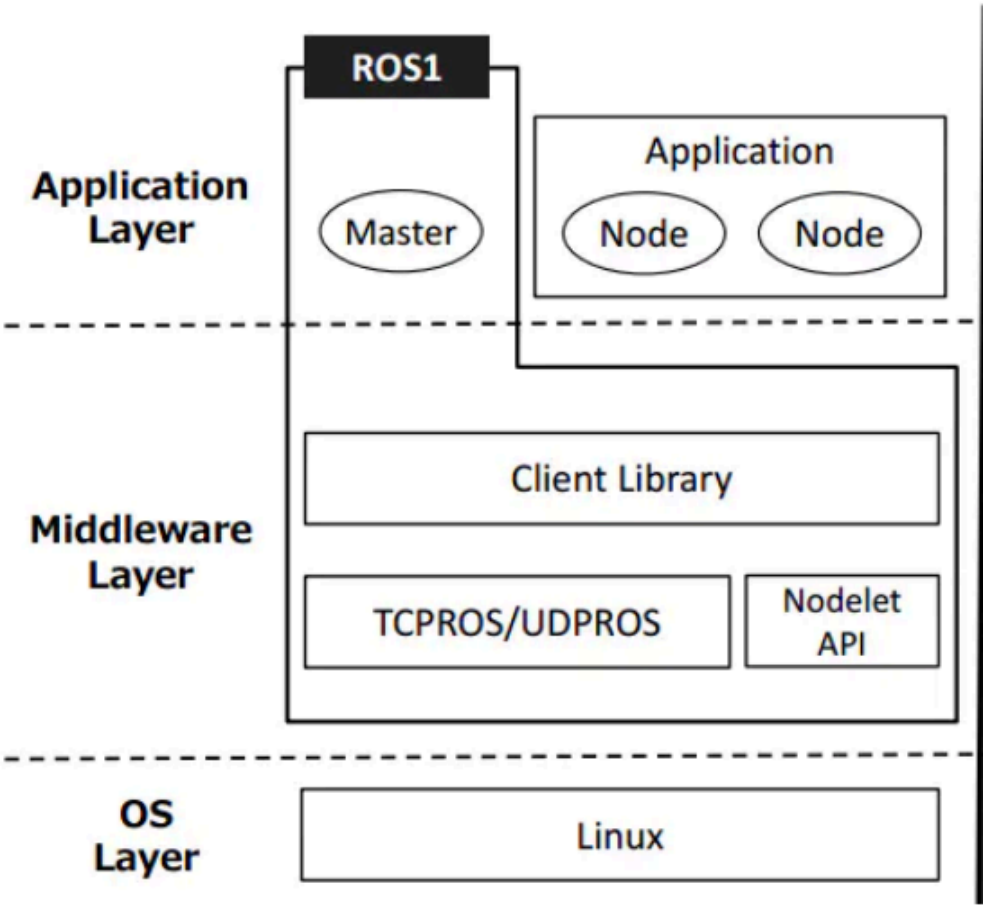
The catkin_create_pkg command will ask you to enter package_name. If necessary, you can also add some other dependent software packages later:

```
catkin_create_pkg <package_name> [depend1] [depend2] [depend3]
```

- For example:

```
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

2 ROS communication architecture



2.1 Master and node

1 Master

Node manager. Each node must register with the master before starting and manage the communication between nodes.

2 roscore

Starting the master will also start rosout (log management) and parameter server (parameter manager)

3 nodes

ROS processes and instances of running executable files in pkg.

```
$rosrun [pkg_name] [node_name] #Start
$rosnode list #List currently running node information
$rosnode info [node_name] #Display detailed information of a node
$rosnode kill [node_name] #End a node
```

4 launch

Start the master and multiple nodes.

```
$roslaunch [pkg_name] [file_name.launch]
```

2.2 Service and Topic

We provide some services and topics for interacting with mycobot.

1 Service

Enter in the command line:

```
source ~/catkin_ws/devel/setup.bash # Add environment variables
roslaunch mycobot_320_communication communication_service.launch
```

Support parameters:

- port：concatenate serial string
- baud：baud rate

Open a new command line:

```
# Display active service information
rosservice list


#/get_joint_angles
#/get_joint_coords
#/set_joint_angles
#/set_joint_coords
#/switch_gripper_status
#/switch_pump_status
```

Related commands and instructions:

command	Detailed description
rosservice list	Display active service information
rosservice info [service name]	Display information about the specified service
rosservice type [service name]	Show service type
rosservice find [service name]	Find a service for a specified service type
rosservice uri [service name]	show ROSRPC URI service
rosservice args [service name]	show service parameters
rosservice call [service name] [parameters]	Request service with input parameters

2 Topic

Enter in the command line:

```
source ~/catkin_ws/devel/setup.bash
roslaunch mycobot_320_communication communication_topic.launch
```

Support parameters:

- port：concatenate serial string
- baud：baud rate

Open a new command line:

```
# Display active service information
rostopic list

#/mycobot/angles_goal
#/mycobot/coords_goal
#/mycobot/angles_real
#/mycobot/coords_real
#/mycobot/pump_status
#/mycobot/gripper_status
```

Related commands and instructions:

Command	Detailed description
rostopic list	Display active topic list
rostopic echo [topic name]	Display the message content of the specified topic in real time
rostopic find [type name]	Display threads with messages of the specified type
rostopic type [topic name]	Displays the message type of the specified topic
rostopic bw [topic name]	Display the message bandwidth of the specified topic（bandwidth）
rostopic hz [topic name]	Display the message data publishing cycle of the specified topic
rostopic info [topic name	Display information about the specified topic
rostopic pub [topic name] [message type] [parameters]	Post a message with the specified topic name

The difference between service and topic:

	service	topic
Synchronization	Asynchronous	Synchronous
communication model	pub/sub	server/client
underlying protocol	ROSTCP/ROSUDP	ROSTCP/ROSUDP
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 learn more about the use of these two features

2.3 Introduction to msg and srv

- msg：The msg file is a simple text file describing the fields of a ROS message. 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: the request (request) and the response (response). msg files are stored in the msg directory of the package, and srv files are stored in the srv directory.

1 rosmsg

rosmsg is a command line tool for displaying information about ROS message types.

rosmsg demo:

```
rosmsg show       # Show message description
rosmsg info       # Display message information
rosmsg list       # list all messages
rosmsg md5        # Display md5 encrypted message
rosmsg package    # Display all messages under a feature pack
rosmsg packages   # List feature packs that contain messages
```

- rosmsg list will list all msgs in the current ROS
- rosmsg packages List all packages containing messages
- rosmsg package List all msgs under a package

```
//rosmsg package  # Package names
rosmsg package turtlesim
```

- rosmsg show Show message description

```
//rosmsg show # message name
rosmsg show turtlesim/Pose
# result:
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

- rosmsg info Works the same as rosmsg show
- rosmsg md5 A check algorithm to ensure the consistency of data transmission

2 rossrv

rossrv is a command-line tool for displaying information about ROS service types, and uses a syntax that is highly similar to rosmsg.

```
rossrv show       # Display service message details
rossrv info       # Display information about service messages
rossrv list       # List all service information
rossrv md5        # Display md5 encrypted service messages
rossrv package    # Display all service messages under a package
rossrv packages   # Show all packages that contain service messages
```

- rossrv list Will list all srv messages in the current ROS
- rossrv packages List all packages that contain service messages
- rossrv package List all msgs under a package

```
//rossrv package # Package names
rossrv package turtlesim
```

- rossrv show Show message description

```
//rossrv show # message name
rossrv show turtlesim/Spawn
# result:
float32 x
float32 y
float32 theta
string name
---
string name
```

- rossrv info The effect is the same as rossrv show
- rossrv md5 Use md5 checksum (encryption) for service data

3 Introduction to URDF

- Unified Robot Description Format，Unified Robot Description Format, abbreviated as URDF. The urdf package in ROS contains a C++ parser for URDF, and URDF files describe robot models in XML format. *URDF cannot be used alone, it 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.

3.1 urdf file description

Code example:

Only part of the code is intercepted here for display:

```
<?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"/>
```

```
    <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, it is mainly composed of two parts, `link` and `joint`, which are repeated continuously.

3.2 link section

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

3.2.1 Attributes

name： The name used to describe the link itself

3.2.2 element

- `<inertial>` (optional)
  - Inertia properties of connecting rods
  - `<origin>` (optional，defaults to identity if not specified)
    - Defines the reference coordinate of the inertial reference system relative to the connecting rod coordinate system. The coordinate must be defined at the center of gravity of the connecting rod, and its coordinate axis may not be parallel to the main axis of inertia.
    - xyz (optional, defaults to zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
    - rpy(optional: defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
  - `<mass>` Mass properties of connecting rods
  - `<inertia>` 3×3 rotational inertia matrix, consisting of six independent quantities: ixx, ixy, ixz, iyy, iyz, izz。
- `<visual>` (optional)
  - Visual properties of the connecting rod. It is used to specify the shape of the link display (rectangle, cylinder, etc.). There can be multiple visual elements in the same link, and the shape of the link is formed by two elements. In general, the model is more complex and can be drawn through soildwork to generate stl calls, and simple shapes such as adding end effectors can be directly written. At the same time, the position of the geometry can be adjusted according to the gap between the theoretical model and the actual model.
  - `<namel>` (optional) The name of the connecting rod geometry.
  - `<origin>` (optional，defaults to identity if not specified)
    - The geometry coordinate system relative to the coordinate system of the connecting rod.

    - xyz (optional: defaults to zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, 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 visualization, which can be one of the following:
  - `<box>` A rectangle with elements including length, width, and height. The origin is in the center.
  - `<cylinder>` Cylinder, elements include radius and length. center of origin.
  - `<sphere>` Sphere, element containing the radius. The origin is in the center.
  - `<mesh>` The grid, as determined by the file, also provides a scale to define its boundaries. Collada .dae files are recommended, .stl files are also supported, but must be a local file.
- `<material>` (optional)
  - Visualize the component's material. It can be defined outside the link tag, but it must be inside the robot tag. When defining outside the link tag, the name of the link must be quoted.
  - `<color>` (optional) Color, consisting of red/green/blue/alpha, in the range [0,1].
  - `<texture>` (optional) Material properties, defined by the file.
- `<collision>` (optional)
  - Collision properties of the link. Collision properties differ from visual properties of connecting rods, and simple collision models are often used to simplify calculations. The same link can have multiple collision attribute labels, and the collision attribute representation of the link is composed of the set of geometric shapes defined by it.
  - `<name>` (optional) Specifies the name of the connecting rod geometry
  - `<origin>` (optional，defaults to identity if not specified)
    - The reference coordinate system of the collision component is relative to the reference coordinate system of the link coordinate system.
    - xyz (optional, default zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
    - rpy (optional, defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
  - `<geometry>` Same as the geometry element description above

Detailed elements and the role of each element can go to official documentation to view

3.3 joint part

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

3.3.1 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 hinged joint that rotates along an axis, the range of which is specified by the upper and lower bounds.
- Continuous - A continuous hinged joint that rotates around an axis with no upper and lower bounds.
- Prismatic - A sliding joint that slides along an axis, the range of which is specified by upper and lower limits. +Fixed - this is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint

does not require axes, calibration, dynamics, limits or safety_controller。
- Floating - This joint allows motion in all 6 degrees of freedom.
- Plane - This joint allows movement in a plane perpendicular to the axis.

3.3.2 elements of joint

- `<origin>` (optional，defaults to identity if not specified) In the transformation from parent link to child link, the joint is located at the origin of the child link. Modifying this parameter can adjust the position of the connecting rod. It can be used to adjust the error between the actual model and the theoretical model, but it is not recommended to modify it greatly, because this parameter affects the connecting rod stl The position of , easily affects the collision detection effect.
  - xyz (optional: default to zero vector) Represents the offset in the x , y , z x,y,zx,y,z axis directions, in meters.
  - rpy (optional: default to zero vector) Represents the angle of rotation around a fixed axis: roll is around the x-axis, pitch is around the y-axis, and yaw is 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 the 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 is the name of the link in the robot structure tree.
- `<axis>` (optional: defaults to (1,0,0))
  - The joint's axis is in the joint's coordinate system. This is the axis of rotation (revolute joint), the axis of movement of the prismatic joint, 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 rotation direction. If the model rotation is opposite to the actual one, just multiply by -1. Fixed and floating joints do not need this element.
  - xyz(required) x , y , z x, y, zx, y, z components representing axis vectors, as normalized vectors.
- `<calibration>` (optional)
  - The reference point of the joint, used to correct the absolute position of the joint.
  - rising (optional) When the joint is moving forward, the reference point triggers a rising edge.
  - falling (optional) When the joint is moving forward, the reference point triggers a falling edge.
- `<dynamics>` (optional)
  - This element is used to specify the physical properties of the joint. Its value is used to describe the modeling performance of the joint, especially during simulation.

`<limit>` (Required when the joint is a rotation or translation joint)

- This element is a joint kinematics constraint.
- lower (optional, default to 0) Specify the attribute of the lower bound of the joint's motion range (the unit of the revolute joint is radians, and the unit of the prismatic joint is meters). This attribute is ignored for continuous joints.
- upper (optional, defaults to 0) Specify the attribute of the upper bound of the joint's motion range (the unit of the revolute joint is radians, and the unit of the prismatic joint is the meter). This attribute is ignored for continuous joints.
- effort (required) This property specifies the maximum force at which the joint will run.
- velocity (required) This property specifies the maximum speed of the joint runtime.

`<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 mimic.
- multiplier(optional) Specify the multiplier factor in the above formula.
- offset(optional) Specify the offset term in the above formula. Default value is 0

`<safety_controller>` (optional)

- This element is a security control limit. The data under this element will be read into move_group, but it is invalid in practice. 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, defaults to 0) This attribute specifies the lower bound of the joint security control boundary, which is the starting limit point of the joint security control. This value needs to be greater than the lower value in the above limit.
- soft_upper_limit (optional, defaults to 0) This attribute specifies the upper bound of the joint security control boundary, which is the starting limit point of the joint security control. This value needs to be less than the upper value in the above limit.
- k_position(optional, defaults to 0) This attribute is used to describe the relationship between position and velocity.
- k_velocity(required) This property is used to describe the relationship between force and velocity.

Detailed elements and the role of each element can go to http://wiki.ros.org/urdf/XML/joint to view.

4 Commonly used command tools

In ROS, there are many commonly used command line tools, which can help you develop, debug, manage ROS nodes, etc. The following are some commonly used ROS command line tools:

4.1 Compile workspace

```
caktin_make
```

4.2 roscore

Start the ROS master node. Before running a ROS node, you usually need to start roscore first

```
roscore
```

4.3 rosrun

Run the specified ROS node.

```
rosrun package_name node_name
```

4.4 roslaunch

Use the Launch file to start one or more ROS nodes.

```
roslaunch package_name launch_file.launch
```

4.5 rosnode

View running ROS node information.

```
rosnode list
rosnode info node_name
```

4.6 rostopic

View information about running ROS topics.

```
rostopic list
rostopic echo topic_name
```

4.7 rosservice

View and call ROS services.

```
rosservice list
rosservice call service_name
```

4.8 rosparam

Get and set ROS parameters.

```
rosparam get parameter_name
rosparam set parameter_name value
```

4.9 rosmsg

View ROS message types.

```
rosmsg show message_type
```

4.10 rosdep

Install dependencies of ROS packages.

```
rosdep install package_name
```

4.11 Environment variables

View the ROS_PACKAGE_PATH environment variable

```
echo $ROS_PACKAGE_PATH
```

← Previous Page | Next Page →

Brief introduction and use of rviz

rviz is a 3D visualization platform in ROS. On one hand, it can realize the graphical display of external information, and on the other hand, it can also release control information to an object through rviz, realizing the monitoring and control of a robot.

1 Installation of rviz and the introduction to its interface

When installing ros, if you perform a complete installation, rviz is already installed, and you may try to run it directly; if it is not fully installed, you may install rviz separately:

```
# Ubuntu16.04
sudo apt-get install ros-kinetic-rviz
```

```
# Ubuntu18.04
sudo apt-get install ros-melodic-rviz
```

```
# Ubuntu20.04
sudo apt-get install ros-noetic-rviz
```

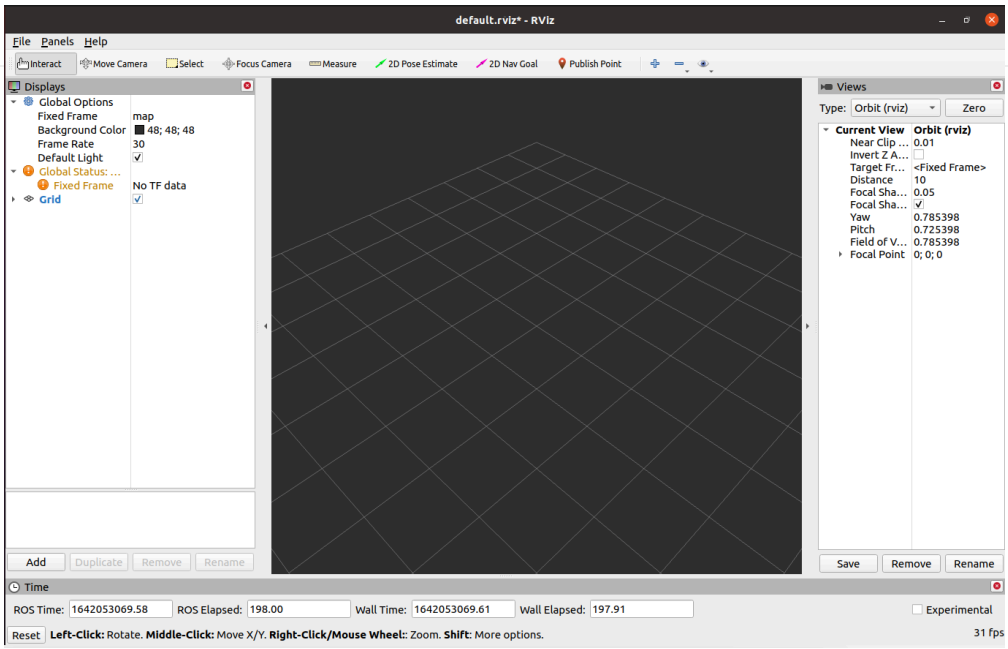
After the installation is complete, open a new terminal (shortcut key: `Ctrl+Alt+T`) and enter the following command:

```
roscore
```

Then open a new terminal (shortcut key: `Ctrl+Alt+T`) and input the following command to open rviz.

```
rosrun rviz rviz
# or
rviz
```

Open rviz, and the following interface will be displayed:



1.1 Introduction of all areas

- There is a list of monitors on the left. The monitor is a device that draws something in a 3D world and may have some options available in the display list.
- On the top is a toolbar, which allows the user to use various function buttons to select tools with multiple functions.
- The middle part is the 3D view: It is a main screen where various data can be viewed 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.
- The right side is the observation angle setting area where different observation angles can be set.

We only give a rough introduction in this part. If you want to know more details, go to User Guide.

2 Prerequisites

- Type Ctrl+Shift+T in the command terminal to open another terminal window in the same directory to view the device name:

```
# View the device name of the robotic arm
ls /dev/ttyUSB* # 2020 version myCobot320 M5


# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # 2022 version myCobot320 M5
```

- Grant the serial port permission to the robotic arm:

```
# 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 # 2020 version myCobot320 M5


sudo chmod 777 /dev/ttyACM0 # 2022 version myCobot320 M5
```

Then enter the user password(**Note:** The password is not displayed, just enter it correctly).

3 Simple use

Start using launch file

This example is built on what you have already done Environment building and you have successfully copied the company's code from GitHub to your virtual machine.

Open a new terminal (shortcut key: `Ctrl+Alt+T`)

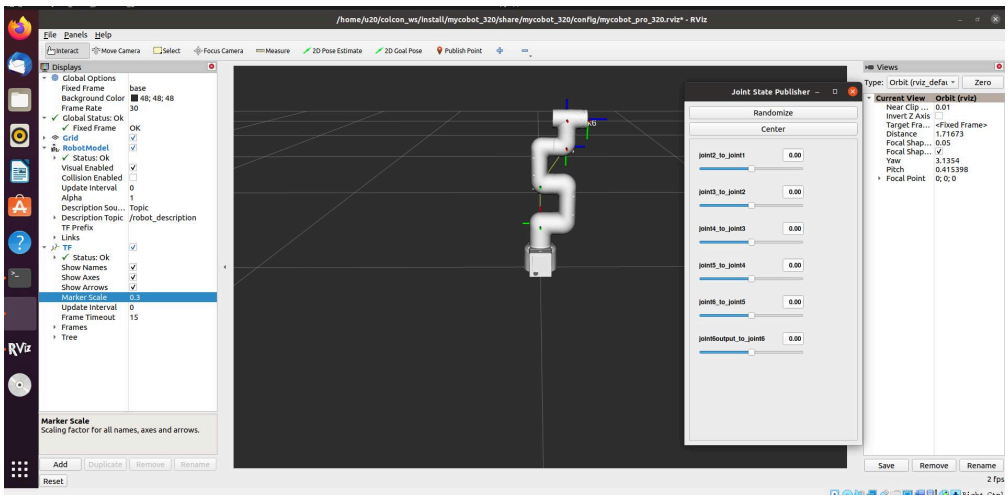
Input the command to **configure the ROS environment**.

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

Input again:

```
roslaunch new_mycobot_320 mycobot_320_test.launch
```

Open rviz, and then you will obtain the following result:



If you want to know more information about rviz, go to Official documents.

← Previous Page | Next Page →

Control and following of the robot arm

1 Slider Control



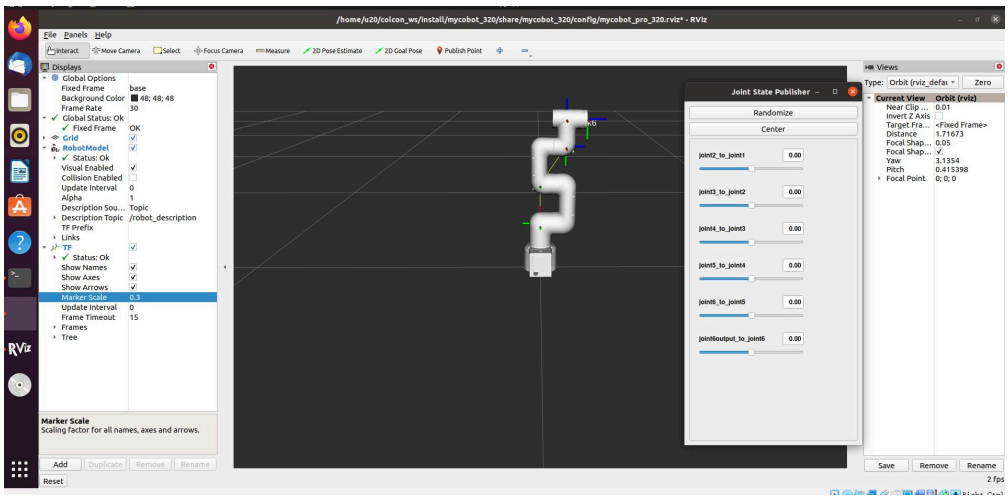


Open a command line and run:

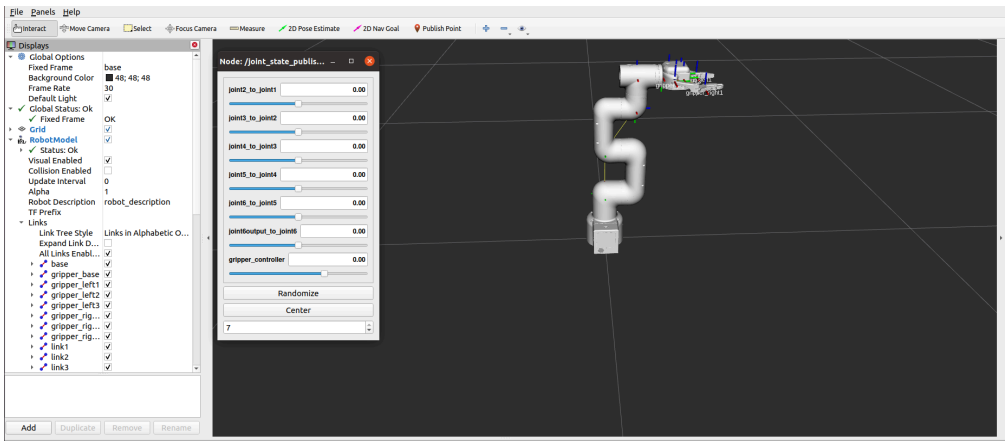
> **Note: If the end effector uses myGripper F100 force-controlled gripper, the version of the pymycobot driver library must be greater than 3.6.4**

```
# The default serial port name of 2022 mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
roslaunch new_mycobot_320 mycobot_320_slider.launch port:=/dev/ttyUSB0 baud:=115200

# If the end is equipped with a Pro adaptive gripper, run (serial port modification is the same as above):
roslaunch new_mycobot_320 mycobot_320_gripper_slider.launch port:=/dev/ttyUSB0 baud:=115200

# If the end is equipped with a myGripper F100 force-controlled gripper, run (serial port modification is the same as abov
roslaunch new_mycobot_320 mycobot_320_force_gripper_slider.launch port:=/dev/ttyACM0 baud:=115200
```

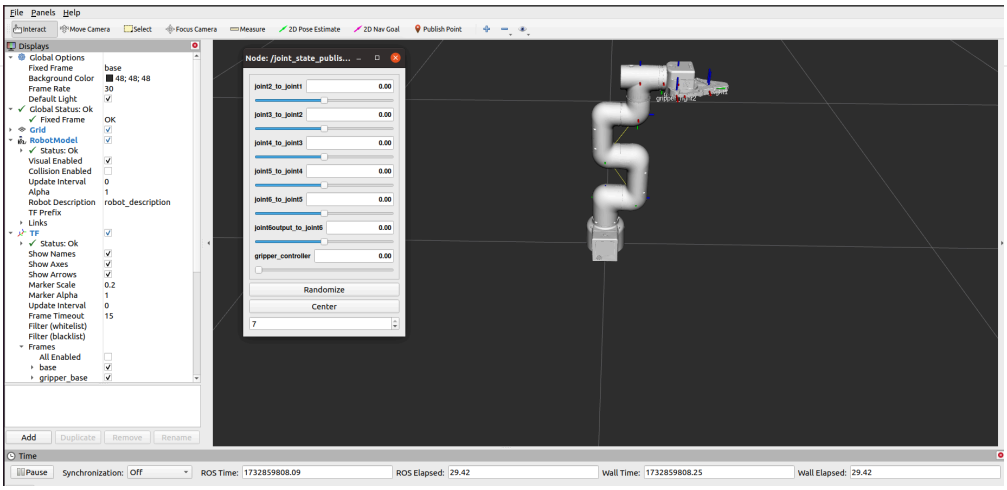
rviz and a slider component will be opened, and you will see the following interface:



If the Pro adaptive gripper is installed at the end, you will see the following interface:



If the myGripper F100 force-controlled gripper is installed at the end, you will see the following interface:



Then you can **control the model in rviz to make it move by dragging the slider**. If you want the real mycobot to move with the model, you need to open another command line and run:

> **Note: If the end effector uses myGripper F100 force-controlled gripper, the version of the pymycobot driver library must be greater than 3.6.4**

```
# The default serial port name of 2022 mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
rosrun new_mycobot_320 mycobot_320_slider.py _port:=/dev/ttyUSB0 _baud:=115200

# If the end is equipped with a Pro adaptive gripper, run (serial port modification is the same as above):
rosrun new_mycobot_320 mycobot_320_gripper_slider.py _port:=/dev/ttyUSB0 _baud:=115200

# If the end is equipped with a myGripper F100 force-controlled gripper, run (serial port modification is the same as abov
rosrun new_mycobot_320 mycobot_320_force_gripper_slider.py _port:=/dev/ttyACM0 _baud:=115200
```

Note: Since the robot arm will move to the current position of the model when the command is input, make sure that the model in rviz does not appear to be worn out 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 controls, we can also let the model move by following the real robot arm.



Open a command line and run:

```
# The default serial port name of 2022 mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
rosrun new_mycobot_320 mycobot_320_follow_display.py _port:=/dev/ttyUSB0 _baud:=115200
```

Then open another command line and run:

```
roslaunch new_mycobot_320 mycobot_320_follow_display.launch
```

It will **open rviz to show the model following effect.**

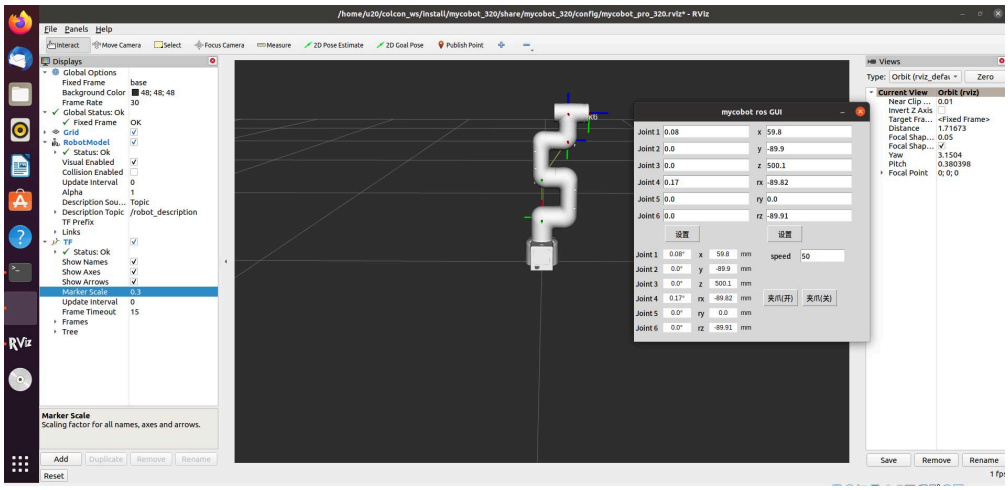
3 GUI control

On the basis of the previous contents, this package also **provides a simple GUI control interface.** This method is used for interaction between real robot arms. Connect to mycobot.



Open a command line:

```
# The default serial port name of 2022 mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
roslaunch new_mycobot_320 mycobot_320_simple_gui.launch port:=/dev/ttyUSB0 baud:=115200
```



Note: Before using the gripper switch button, make sure the adaptive gripper is connected to the end of the robot arm.

4 Keyboard control

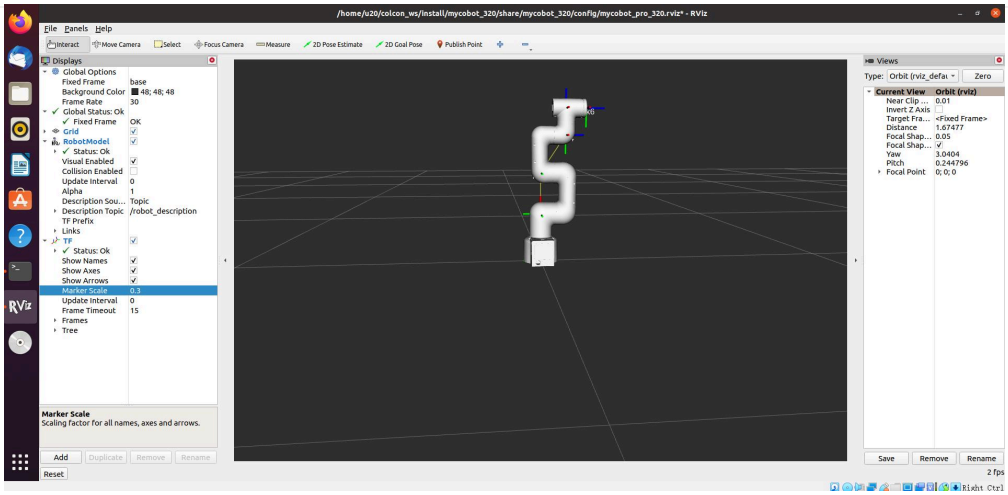
Keyboard control is added in `new_mycobot_320` package, and real-time Synchronization is performed in rviz. This function depends on pythonApi, so be sure to connect with the real robot arm.



Open a command line and run:

```
# The default serial port name of 2022 mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
roslaunch new_mycobot_320 mycobot_320_teleop_keyboard.launch port:=/dev/ttyUSB0 baud:=115200
```

Running effect is as follows:



mycobot information will be output in the command line as follows:

1.4.1 AdaptiveGripper

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

PARAMETERS
 * /mycobot_services/baud: 115200
 * /mycobot_services/port: /dev/ttyUSB0
 * /robot_description: <?xml version="1....
 * /rosdistro: kinetic
 * /rosversion: 1.12.1.17

NODES
  /
    mycobot_services (new_mycobot_320/mycobot_services.py)
    real_listener (new_mycobot_320/listen_real.py)
    robot_state_publisher (robot_state_publisher/state_publisher)
    rviz (rviz/rviz)

auto-starting new master
process[master]: started with pid [1333]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to f977b3f4-b3a9-11eb-b0c8-d0c63728b379
process[rosout-1]: started with pid [1349]
started core service [/rosout]
process[robot_state_publisher-2]: started with pid [1357]
process[rviz-3]: started with pid [1367]
process[mycobot_services-4]: started with pid [1380]
process[real_listener-5]: started with pid [1395]
[INFO] [1620882819.196217]: start ...
[INFO] [1620882819.205050]: /dev/ttyUSB0,115200

MyCobot Status
-------------------------------
Joint Limit:
    joint 1: -165 ~ +165
    joint 2: -165 ~ +165
    joint 3: -165 ~ +165
    joint 4: -165 ~ +165
    joint 5: -165 ~ +165
    joint 6: -175 ~ +175

Connect Status: True

Servo Infomation: all connected

Servo Temperature: unknown

Atom Version: unknown
```

```
[INFO] [1620882819.435778]: ready
```

Then open another command line and run:

```
rosrun new_mycobot_320 mycobot_320_teleop_keyboard.py
```

You will see the following output in the command line:

```
Mycobot Teleop Keyboard Controller
---------------------------
Movimg 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: 50       change percent 5
```

In this terminal, you can control the state of the robot arm and move it using the keys in the command line.

Parameters supported by this script:

- _speed: the movement speed of the robot arm
- _change_percent: movement distance percentage

5 moveit use

`mycobot_ros` has integrated the MoveIt section.

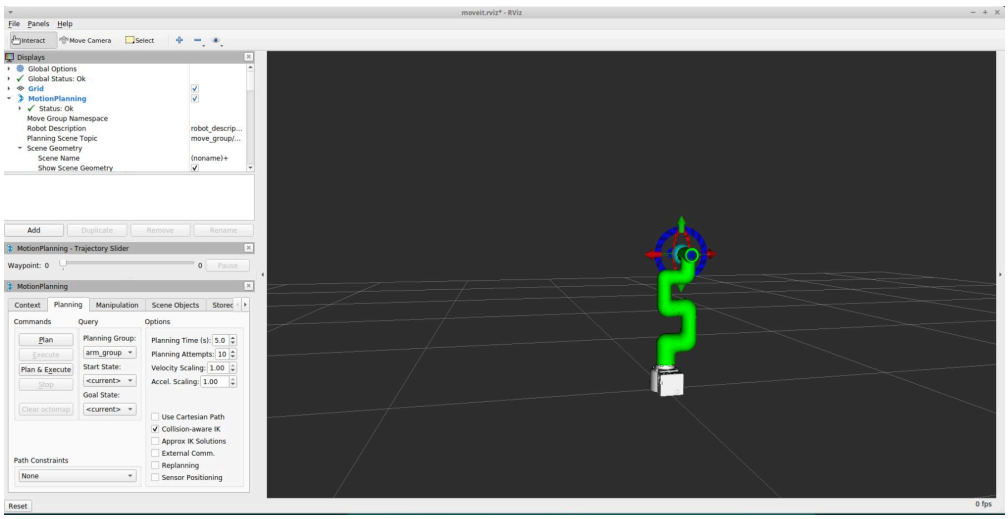
> **Note: If the end effector uses myGripper F100 force-controlled gripper, the version of the pymycobot driver library must be greater than 3.6.4**



Open the command line and run:

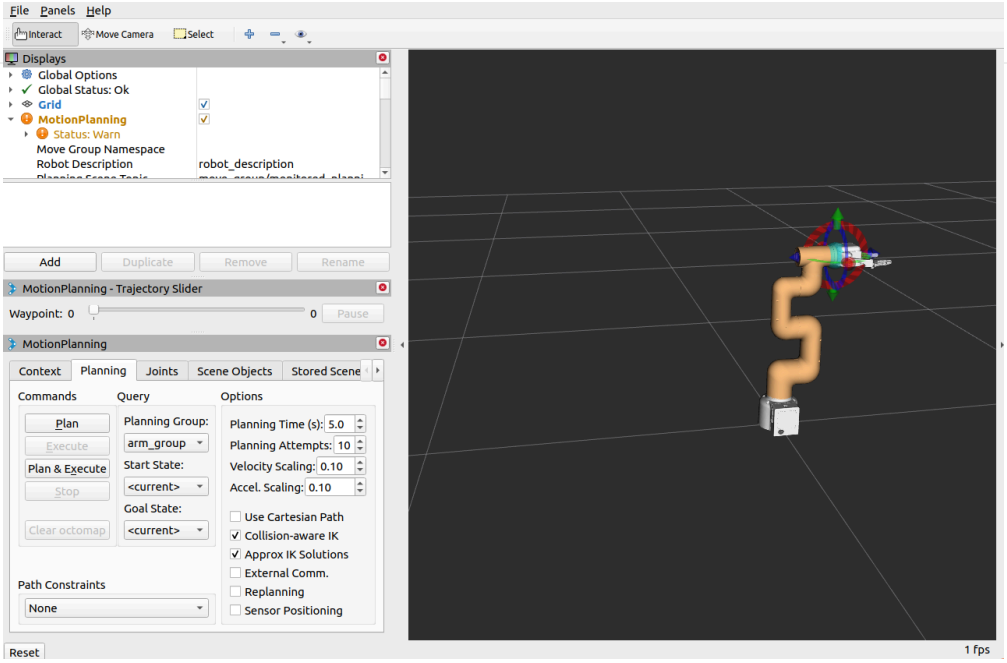
```
roslaunch new_mycobot_320_moveit mycobot320_moveit.launch


# If the end is equipped with a Pro adaptive gripper, run:
roslaunch new_mycobot_320_gripper_moveit mycobot320_gripper_moveit.launch


# If the end is equipped with a myGripper F100 force-controlled gripper, run:
roslaunch new_mycobot_320_force_gripper_moveit mycobot320_force_gripper_moveit.launch
```

The operation effect is as follows:

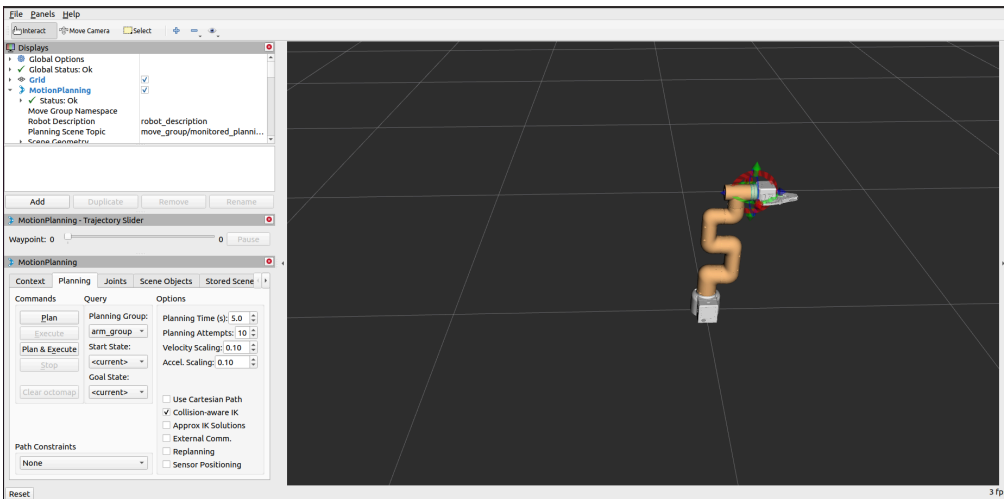


If the end is equipped with a Pro adaptive gripper, the operation effect is as follows:

1.4.1 AdaptiveGripper



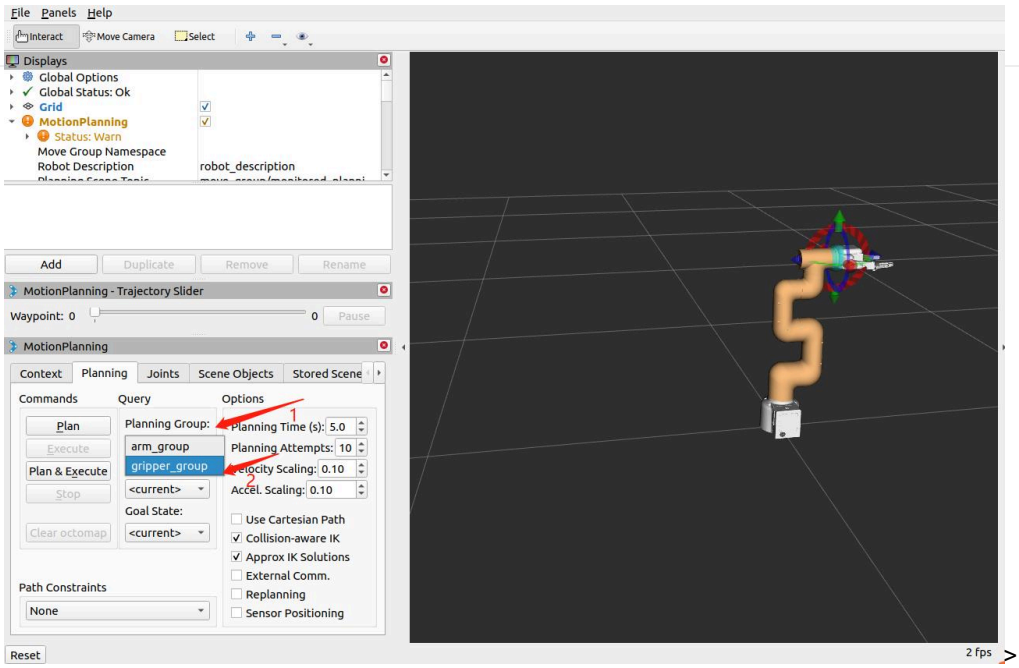
If the end is equipped with a myGripper F100 force-controlled gripper, the operation effect is as follows:



If you want a real robot arm to execute a plan synchronously, you need to open another command line and run:

```
# The default serial port name of 2022 mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
rosrun new_mycobot_320_moveit sync_plan.py _port:=/dev/ttyUSB0 _baud:=115200

# If the end is equipped with a Pro adaptive gripper, run (serial port modification is the same as above):
rosrun new_mycobot_320_gripper_moveit sync_plan.py _port:=/dev/ttyUSB0 _baud:=115200

# If the end is equipped with a myGripper F100 force-controlled gripper, run (serial port modification is the same as abov
rosrun new_mycobot_320_force_gripper_moveit sync_plan.py _port:=/dev/ttyACM0 _baud:=115200
```

Note: If the end point is equipped with Pro adaptive grippers or myGripper F100 force-controlled grippers and the grippers need to be planned, you need to switch the planning group to the planning group of the grippers.



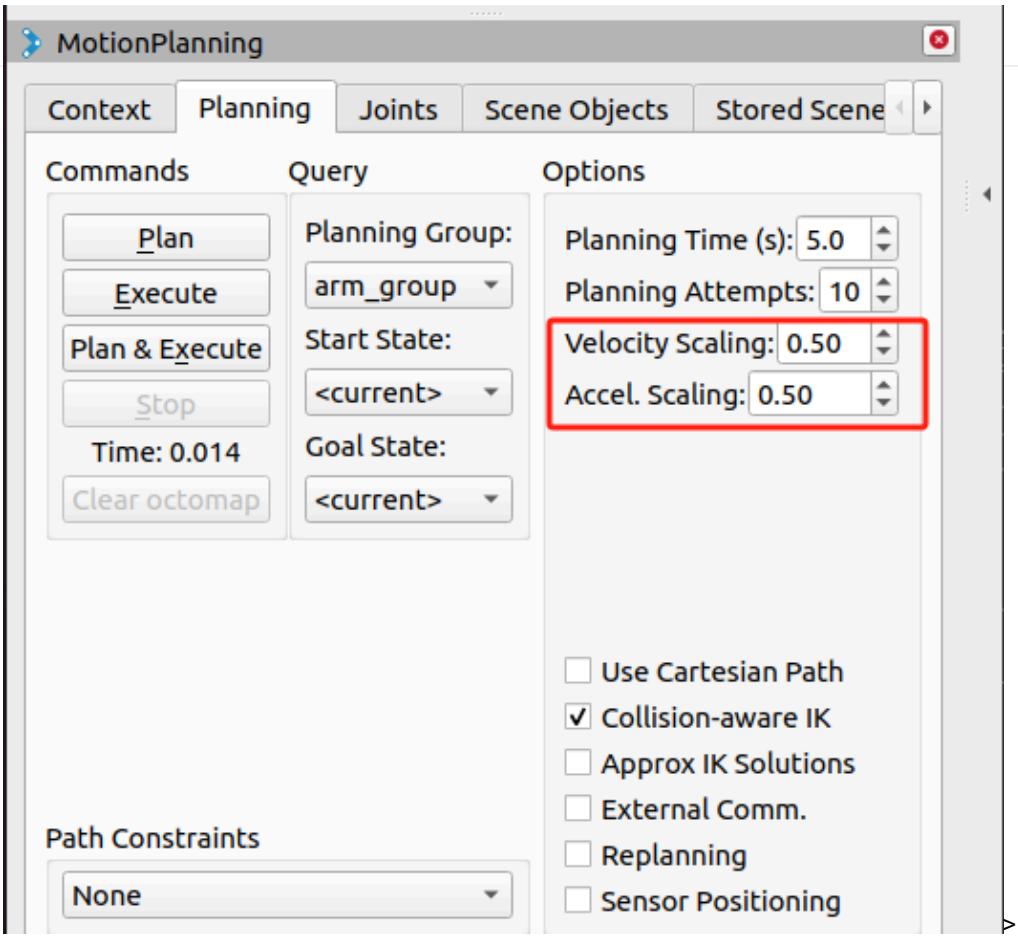
Modify motion speed

In order to prevent the joints from shaking during the movement of the real robotic arm, the movement speed of the joints needs to be reduced.

- In the `sync_plan.py` file, modify the speed parameter of the robot arm Python API, here it is changed to 25.

```
...

def callback(data):
  # rospy.loginfo(rospy.get_caller_id() + "%s", data)
  data_list = []
  for index, value in enumerate(data.position):
      radians_to_angles = round(math.degrees(value), 2)
      data_list.append(radians_to_angles)

  rospy.loginfo(rospy.get_caller_id() + "%s", data_list)
  mc.send_angles(data_list, 25)  # Change speed to 25

...
```

- In the Moveit RViz interface, modify the scaling ratio of speed and acceleration. Here, change it to 0.5, and then save the current configuration.



← Previous Page | Next Section →

Gazebo Introduction and Usage

Gazebo is a 3D visualization platform in ROS. On one hand, it can display external information graphically; on the other hand, it can publish control information to objects through RViz to monitor and control robots. It supports simulating a real physical world and can be used in conjunction with MoveIt.

Local Setup

1. Operation Process

1.1 Prerequisites

To use this package, you need to install the Python API library first.

```
pip install pymycobot --user


# Environment:
ros1 noetic
gazebo 11
pymycobot 4.0.3
```

1.2 Package Download and Installation

Download the package to your ROS workspace:

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/elephantrobotics/mycobot_ros.git
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
```

MyCobot_320_m5-Gazebo Usage Guide

1. Slider Control

Slider control through joint_state_publisher_gui is now implemented to control the robot arm model pose in Gazebo.

After connecting the real robot arm to the computer, check the port the robot arm is connected to:

```
ls /dev/tty*
# /dev/ttyACM0 or /dev/ttyUSB0
```

You will get output similar to:

```
/dev/tty    /dev/tty26  /dev/tty44  /dev/tty62      /dev/ttyS20
/dev/tty0   /dev/tty27  /dev/tty45  /dev/tty63      /dev/ttyS21
/dev/tty1   /dev/tty28  /dev/tty46  /dev/tty7       /dev/ttyS22
/dev/tty10  /dev/tty29  /dev/tty47  /dev/tty8       /dev/ttyS23
/dev/tty11  /dev/tty3   /dev/tty48  /dev/tty9       /dev/ttyS24
/dev/tty12  /dev/tty30  /dev/tty49  /dev/ttyACM0    /dev/ttyUSB0
/dev/tty13  /dev/tty31  /dev/tty5   /dev/ttyprintk  /dev/ttyS26
/dev/tty14  /dev/tty32  /dev/tty50  /dev/ttyS0      /dev/ttyS27
/dev/tty15  /dev/tty33  /dev/tty51  /dev/ttyS1      /dev/ttyS28
/dev/tty16  /dev/tty34  /dev/tty52  /dev/ttyS10     /dev/ttyS29
/dev/tty17  /dev/tty35  /dev/tty53  /dev/ttyS11     /dev/ttyS3
/dev/tty18  /dev/tty36  /dev/tty54  /dev/ttyS12     /dev/ttyS30
/dev/tty19  /dev/tty37  /dev/tty55  /dev/ttyS13     /dev/ttyS31
/dev/tty2   /dev/tty38  /dev/tty56  /dev/ttyS14     /dev/ttyS4
/dev/tty20  /dev/tty39  /dev/tty57  /dev/ttyS15     /dev/ttyS5
/dev/tty21  /dev/tty4   /dev/tty58  /dev/ttyS16     /dev/ttyS6
/dev/tty22  /dev/tty40  /dev/tty59  /dev/ttyS17     /dev/ttyS7
/dev/tty23  /dev/tty41  /dev/tty6   /dev/ttyS18     /dev/ttyS8
/dev/tty24  /dev/tty42  /dev/tty60  /dev/ttyS19     /dev/ttyS9
/dev/tty25  /dev/tty43  /dev/tty61  /dev/ttyS2
```

Enable communication and add execution permissions to the scripts:

```
sudo chmod -R 777 /dev/ttyACM0  # or sudo chmod -R 777 /dev/ttyUSB0
sudo chmod -R 777 mycobot_320m5_gazebo/mycobot_320m5_gripper_gazebo/scripts/follow_display_gazebo.py
sudo chmod -R 777 mycobot_320m5_gazebo/mycobot_320m5_gripper_gazebo/scripts/slider_control_gazebo.py
sudo chmod -R 777 mycobot_320m5_gazebo/mycobot_320m5_gripper_gazebo/scripts/teleop_keyboard_gazebo.py
roscore
```

After confirming the port, open a terminal in the workspace and enter the following command (remember to change the port to the value found in the previous step):

```
source devel/setup.bash
roslaunch mycobot_320m5_gripper_gazebo slider.launch _port:=/dev/ttyACM0 _baud:=115200
```

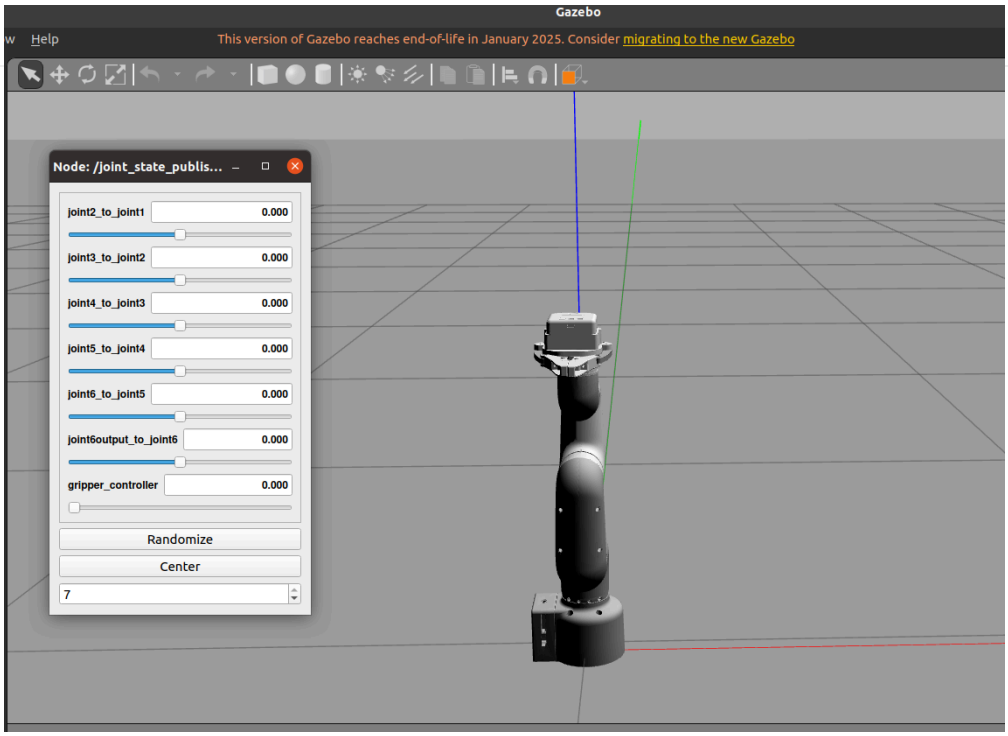
Then open another terminal in the workspace and enter:

```
source devel/setup.bash
rosrun mycobot_320m5_gripper_gazebo slider_control_gazebo.py _port:=/dev/ttyACM0 _baud:=115200
```

Remember to modify the port number to the one found in the previous step. If successful, you will see the following terminal prompt:

```
('/dev/ttyACM0', 115200)
spin ...
```

Now you can control the robot arm model pose in Gazebo by manipulating the joint_state_publisher_gui sliders.



2. Gazebo Model Following

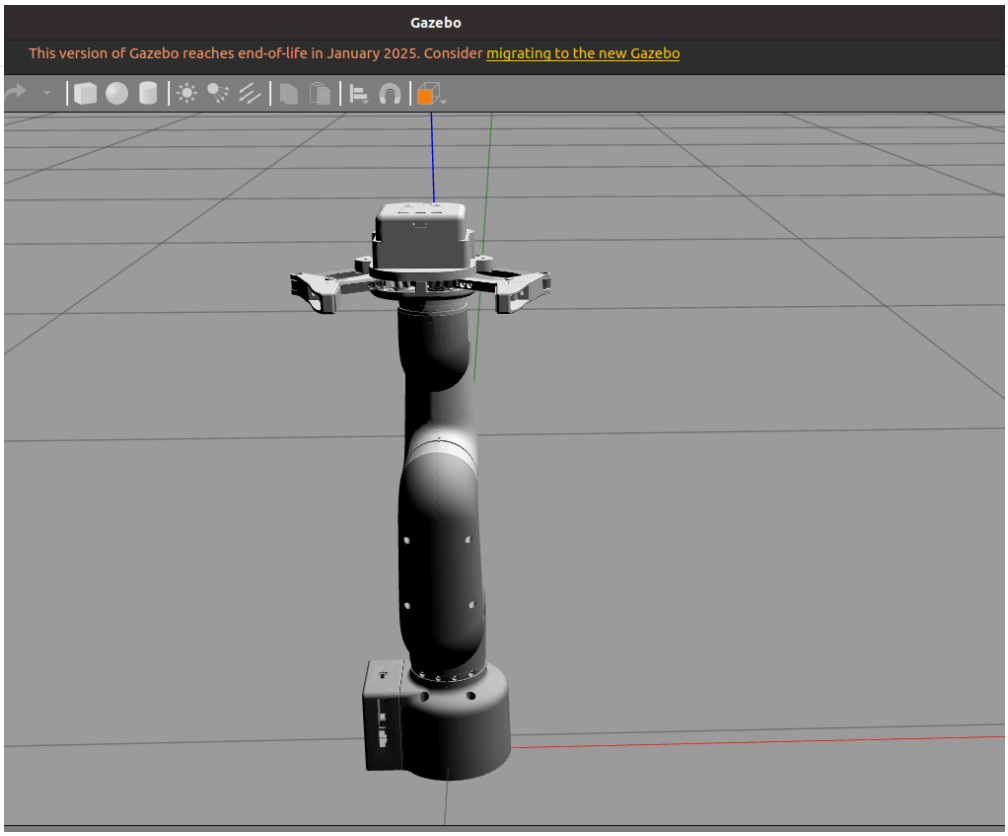
The following commands allow the Gazebo model to follow the actual robot arm's movements. First, run the launch file:

```
source devel/setup.bash
roslaunch mycobot_320m5_gripper_gazebo follower.launch _port:=/dev/ttyACM0
```

If the program runs successfully, the Gazebo interface will load the robot arm model with all joints at their original pose, i.e., [0,0,0,0,0,0]. Then open a second terminal and run:

```
source devel/setup.bash
rosrun mycobot_320m5_gripper_gazebo follow_display_gazebo.py _port:=/dev/ttyACM0 _baud:=115200
```

Now when you manipulate the actual robot arm's pose, you can see the robot arm in Gazebo moving to the same pose.



3. Keyboard Control

You can also use keyboard input to control both the Gazebo robot arm model and the actual robot arm simultaneously. First, open a terminal and enter:

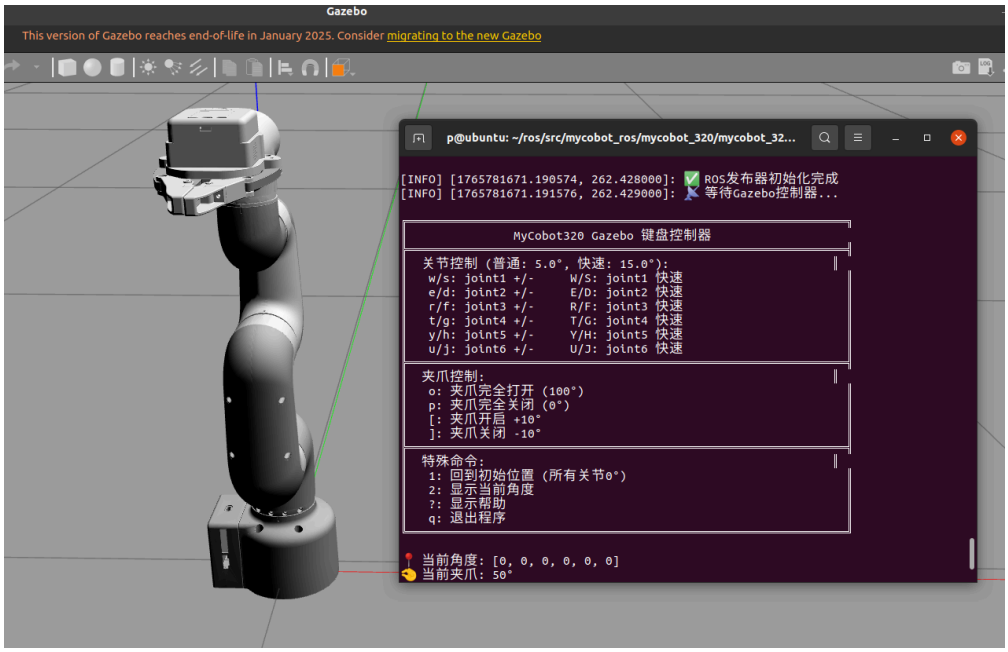
```
source devel/setup.bash
roslaunch mycobot_320m5_gripper_gazebo follower.launch _port:=/dev/ttyACM0 _baud:=115200
```

As in the previous section, you will see the robot arm model loaded in Gazebo with all joints at their initial pose. Then open another terminal and enter:

```
source devel/setup.bash
rosrun mycobot_320m5_gripper_gazebo teleop_keyboard_gazebo.py _port:=/dev/ttyACM0 _baud:=115200
```

If successful, you will see the following output in the terminal:

```
Mycobot_320m5_gripper_gazebo Teleop Keyboard Controller
---------------------------
Moving options (control the angle of each joint):

w: joint2_to_joint1++   s: joint2_to_joint1--
e: joint3_to_joint2++   d: joint3_to_joint2--
r: joint4_to_joint3++   f: joint4_to_joint3--
t: joint5_to_joint4++   g: joint5_to_joint4--
y: joint6_to_joint5++   h: joint6_to_joint5--
u: joint6output_to_joint6++ j: joint6output_to_joint6--

o: open gripper          p: close gripper

Other:
1 - Go to home pose
q - Quit
```



Based on the prompts above, you can learn how to control the robot arm. Each key press moves the robot arm and the Gazebo model by 1 degree. You can try holding down one of the keys to reach a certain pose.

ROS2 introduction

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

1 Design goals and features of ROS2

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

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

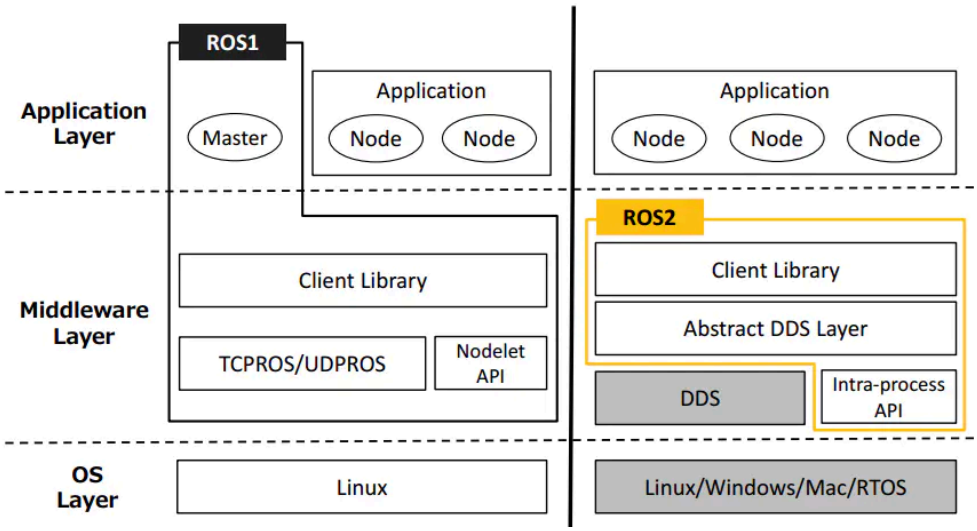
2 Release Version

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

ROS2 version	release date	Maintenance deadline	Ubuntu version
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)

3 Comparison of ROS and ROS2

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



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

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

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

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

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

← Previous Section | Next Page →

Install different versions of ubuntu system in Linux

1 Virtual machine installation

Notice: When installing the virtual machine system, please install the **Ubuntu 20.04 version of the system**, The installation method is consistent with ubuntu 18.04.

To install different versions of Ubuntu systems in Linux, please refer to the 6.1 ROS1 Environment building chapter for specific installation methods.

> There are two options for virtual machine creation:
>
> - ① Using our configured virtual machine ROS1 virtual machine files, imported into the virtual machine software, you can save the time to configure the environment, easy to quickly use the ROS case.
>   - ROS2 VM files:
>
>   https://download-elephantrobotics.oss-cn-shenzhen.aliyuncs.com/system_images/ubuntu20.04_ROS2_V20240228.zip
>   - Step-by-step video on how to import VM files into VM software:
>
>   https://drive.google.com/file/d/1KeYk_CUgDE46rVn7zbd0EhraIbgt3qZt/view?usp=sharing
> - ② Follow the steps below to deploy the ROS environment yourself.

2 ROS2 Environment building

2.1 ROS2 installation

To build a basic development environment, you need to install the robot operating system ROS2 and the git version manager. The installation methods and processes are introduced below.

2.1.1 Selecting a version

There is a one-to-one relationship between ROS2 and ubuntu. Different versions of ubuntu correspond to different versions of ROS2. The reference website is as follows: http://docs.ros.org/en/foxy/Releases.html

Here are the ROS versions supported by Ubuntu:

ROS2 version	release date	Maintenance deadline	Ubuntu version
Foxy	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)

Install the ROS2 version corresponding to the Ubuntu version you have installed.

If the versions are different, the downloading will fail. The system we choose here is Ubuntu 20.04(recommend), so the corresponding ROS2 version is ROS2 Foxy.

> NOTE: At present, we do not provide any reference for installing ROS in windows. If necessary, refer to http://docs.ros.org/en/foxy/Installation/Alternatives/Windows-Development-Setup.html

2.1.2 Start installation

1 Adding a software source

There is no ROS2 software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS2. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command:

- official source:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.or
```

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, huawei cloud is:

```
echo "deb [arch=$(dpkg --print-architecture)] https://repo.huaweicloud.com/ros2/ubuntu/ $(lsb_release -cs) main" | sudo te
```

2 Setting a key

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

```
sudo apt install curl gnupg2 -y
curl -s https://gitee.com/ohhuo/rosdistro/raw/master/ros.asc | sudo apt-key add -
```

3 Installation

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

```
sudo apt-get update
```

Execute **Installing ROS2**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command according to your Ubuntu version:

```
# Ubuntu 20.04 foxy version
sudo apt install ros-foxy-desktop
```

```
# Ubuntu 20.04 galactic version
sudo apt install ros-galactic-desktop
```

```
# Ubuntu 22.04 humble version
sudo apt install ros-humble-desktop
```

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

Refresh environment variables after installation is complete:

```
source /opt/ros/foxy/setup.bash
```

2.1.3 Set up the ros2 environment

In order to avoid re-validating the ROS2 function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS2 function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

Execute the following command according to your Ubuntu version:

```
# Ubuntu 20.04 foxy version
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
```

Then run:

```
source ~/.bashrc
```

Installing a ROS2 extra dependency

Input the following command in the terminal to install a ROS2 extra dependency, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

```
sudo apt install python3-argcomplete -y
```

```
sudo apt install ros-foxy-xacro
```

```
sudo apt-get install python3-colcon-common-extensions
```

```
# Ubuntu 20.04 foxy version
sudo apt install ros-foxy-joint-state-publisher-gui
```

```
# Ubuntu 20.04 galactic version
sudo apt install ros-galactic-joint-state-publisher-gui
```

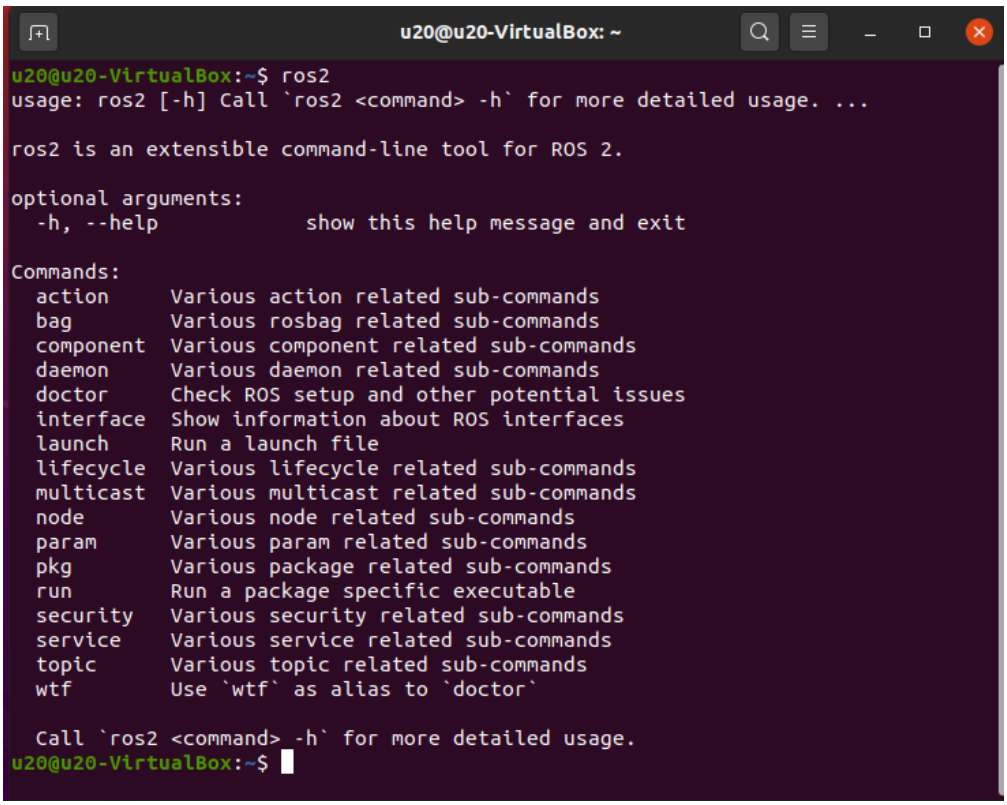
```
# Ubuntu 22.04 humble version
sudo apt install ros-humble-joint-state-publisher-gui
```

2.1.4 Verify installation

To verify whether ROS2 has been installed successfully, open a console terminal (shortcut key: `Ctrl+Alt+T`) and execute the following command in the terminal:

```
ros2
```

When the following interface is displayed, it means that ROS2 has been installed successfully.



2.2 git Installation

1 Updating the software source list

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

```
sudo apt-get update
```

2 Installation git

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

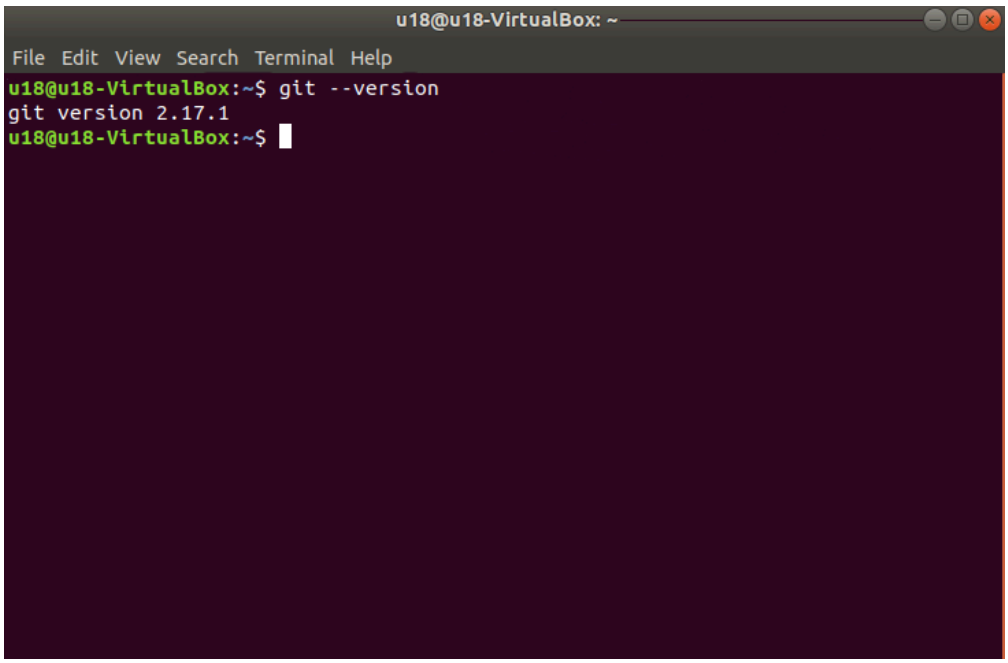
```
sudo apt-get install git
```

3 Verify installation

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

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.



2.3 mycobot_ros2 Installation

`mycobot_ros2` is a ROS2 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros2

1 Precondition

Before installing the package, make sure you have a ros2 workspace.

Here is an **example command for creating a workspace**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command in the command line:

```
mkdir -p ~/colcon_ws/src  # Create a folder
cd ~/colcon_ws
colcon build  # compile workspace
```

Add workspace environment

The official default ROS2 workspace is `colcon_ws` .

```
echo "source ~/colcon_ws/install/setup.bash" >> ~/.bashrc
```

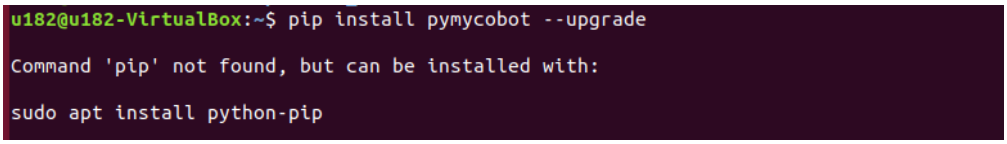
```
source ~/.bashrc
```

2 Installation

NOTE:

- This package depends on ROS2 and MoveIT2. Before using it, make sure that you have installed ROS2 and MoveIT2 successfully.
- The interaction of this package with a real robot arm depends on PythonApi - `pymycobot` .
- The Api project location 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:

  Enter the following command to install pip at the prompt

  ```
  sudo apt install python3-pip
  ```

After the pip installation is complete, the terminal executes again

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

- The installation method depends on Git, so make sure that Git have been installed in your computer.

The official default ROS2 workspace is `colcon_ws` .

```
cd ~/colcon_ws/src  # Enter the src folder of the workspace
# Clone the code on github
git clone --depth 1 https://github.com/elephantrobotics/mycobot_ros2.git
cd ..       # return to the workspace
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile python scri
source install/setup.bash # add environment variables
```

So far, the ROS2 environment construction has been completed.You can learn the basics of ROS2 or ROS2 use cases

← Previous Page | Next Page →

1 ROS2 project structure

1.1 colcon workspace

The colocn workspace is the directory where software packages are created, modified, and compiled. Colcon's workspace can be intuitively described as a warehouse, which contains various ROS project projects to facilitate system organization, management and calling.

- **Create workspace**:

```
mkdir -p ~/colcon_ws/src # Create folder
cd ~/colcon_ws/ # Enter the folder
colcon build # Build the code in the workspace.
```

Note: colcon supports option `--symlink-install`. This allows for faster iteration by changing installed files by changing files in the source space (such as Python files or other uncompiled resources). Avoid the need to recompile every time you modify your python script.

```
colcon build --symlink-install
```

A ROS workspace is a directory with a particular structure. Commonly there is a `src` subdirectory. Inside that subdirectory is where the source code of ROS packages will be located. Typically the directory starts otherwise empty.

colcon does out of source builds. By default it will create the following directories as peers of the `src` directory:

```
src/: colcon package for ROS2 (source code package)

build/: The location where intermediate files are stored. For each package, a subfolder is created in which CMake is calle

install/: The installation location of each package. By default, each package will be installed into a separate subdirecto

log/: Contains various logging information about each colcon call.
```

The directory structure of a ROS2 workspace is as follows:

```
WorkSpace --- Customized workspace.
     |--- build: The directory where intermediate files are stored. A separate subdirectory will be created for each funct
     |--- install: Installation directory, a separate subdirectory will be created for each function package in this direc
     |--- log: Log directory, used to store log files.
     |--- src: Directory used to store function package source code.
         |-- C++ function package
             |-- package.xml: package information, such as: package name, version, author, dependencies.
             |-- CMakeLists.txt: Configure compilation rules, such as source files, dependencies, and target files.
             |-- src: C++ source file directory.
             |-- include: header file directory.
             |-- msg: message interface file directory.
             |-- srv: Service interface file directory.
             |-- action: action interface file directory.
         |-- Python function package
             |-- package.xml: package information, such as: package name, version, author, dependencies.
             |-- setup.py: similar to CMakeLists.txt of C++ function package.
             |-- setup.cfg: Function package basic configuration file.
             |-- resource: resource directory.
             |-- test: stores test-related files.
             |-- Directory with the same name of the function package: Python source file directory.
```

1.2 ROS2 package

Package is not only a software package on Linux, but also the basic unit of colcon compilation. The object we use `colcon build` to compile is each ROS2 package.

Create your own package:

- The command syntax for creating a software package using Python is:

```
ros2 pkg create --build-type ament_python <package_name>
```

- For example:

```
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

2 Basic tool commands

In this chapter, you will learn about the common command tools of ROS2.

2.1 Topics

ROS 2 breaks complex systems down into many modular nodes. Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages. Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

Specific reference: Official Tutorials

- **topics help**

```
ros2 topics -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **Node Relationship Diagram**

```
rqt_graph
```

- **Learn about topic-related commands**

```
ros2 topics -h
```

- **topics list**

```
ros2 topic list
ros2 topic list -t # Display the corresponding message type
```

- **View topic content**

```
ros2 topic echo <topic_name>
ros2 topic echo /turtle1/cmd_vel
```

- **Display topic-related information, type**

```
ros2 topic info <topic_name>
# Output /turtle1/cmd_vel topic related information
ros2 topic info /turtle1/cmd_vel
```

- **Display interface related information**

```
ros2 interface show <msg_type>
# Output geometry_msgs/msg/Twist interface related information
ros2 interface show geometry_msgs/msg/Twist
```

- **Issue an order**

```
ros2 topic pub <topic_name> <msg_type> '<args>'
# Issue speed command
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.
# Issue speed commands at a certain frequency
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y:
```

- **See how often topics are posted**

```
ros2 topic hz <topic_name>
# Output /turtle1/cmd_vel publish 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:
```

2.2 Nodes

Each node in ROS should be responsible for a single, module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters. A full robotic system is comprised of many nodes working in concert. In ROS 2, a single executable (C++ program, Python program, etc.) can contain one or more nodes.

Specific reference: Official Tutorials

- **nodes help**

```
ros2 nodes -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View the node list**

```
ros2 node list
```

- **View Node Relationship Diagram**

```
rqt_graph
```

- **Remapping**

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=my_turtle
ros2 node list
```

- **View node information**

```
ros2 node info <node_name>
ros2 node info /my_turtle
```

2.3 Services

Services are another method of communication for nodes in the ROS graph. Services are based on a call-and-response model, versus topics' publisher-subscriber model. While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.

Specific reference: Official Tutorials

- **services help**

```
ros2 service -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View the service list**

```
ros2 service list
# Display service list and message type
ros2 service list -t
```

- **View the message types received by the service**

```
ros2 service type <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 Definitions**

```
ros2 interface show <type_name>.srv
ros2 interface show std_srvs/srv/Empty.srv
```

- **Call the service command to clear the walking track**

```
ros2 service call <service_name> <service_type>
ros2 service call /clear std_srvs/srv/Empty
```

- **Spawn a new turtle**

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle2'}"
```

2.4 Parameters

A parameter is a configuration value of a node. You can think of parameters as node settings. A node can store parameters as integers, floats, booleans, strings, and lists. In ROS 2, each node maintains its own parameters. For more background on parameters, please see the concept document.

Specific reference: Official Tutorials

- **parameters help**

```
ros2 param -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View service list**

```
ros2 param list
```

- **Get the parameter value**

```
ros2 param get <node_name> <parameter_name>
ros2 param get /turtlesim background_g
```

- **Set parameter values**

```
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**

```
ros2 param load <node_name> <parameter_file>
ros2 param load /turtlesim ./turtlesim.yaml
```

- **Start the node and import parameters at the same time**

```
ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>
ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```

2.5 Actions

Actions are one of the communication types in ROS 2 and are intended for long running tasks. They consist of three parts: a goal, feedback, and a result.

Actions are built on topics and services. Their functionality is similar to services, except actions are preemptable (you can cancel them while executing). They also provide steady feedback, as opposed to services which return a single response.

Actions use a client-server model, similar to the publisher-subscriber model (described in the topics tutorial). An “action client” node sends a goal to an “action server” node that acknowledges the goal and returns a stream of feedback and a result.

Specific reference: Official Tutorials

- **action help**

```
ros2 action -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

Press G|B|V|C|D|E|R|T to achieve rotation, press F to cancel

- **View the server and client of the node action**

```
ros2 node info /turtlesim
```

- **View action list**

```
ros2 action list
ros2 action list -t # show action type
```

- **view action info**

```
ros2 action info <action>
ros2 action info /turtle1/rotate_absolute
```

- **View action message content**

```
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
```

2.6 RQt

RQt is a graphical user interface framework that implements various tools and interfaces in the form of plugins. One can run all the existing GUI tools as dockable windows within RQt! The tools can still run in a traditional standalone method, but RQt makes it easier to manage all the various windows in a single screen layout.

Specific reference: Official Tutorials

You can run any RQt tools/plugins easily by:

```
rqt
```

- **rqt help**

```
rqt -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- Action Type Browser: / Plugins -> Actions ->Action Type Browser
- parameter reconfiguration: / Plugins -> configuration ->Parameter Reconfigure
- Node grap: /Node Graph
- control steering: /Plugins -> Robot Tools -> Robot Steering
- service invocation: /Plugins -> Services -> Service Caller
- Service Type Browser: Plugins -> Services -> Service Type Browser
- message release: Plugins -> Topics -> Message Publisher
- Message Type Browser: Plugins -> Topics -> Message Type Browser
- topic list: Plugins -> Topics -> Topic Monitor
- draw a graph: Plugins -> Visualization -> Plot
- **View logs: rqt_console**

```
ros2 run rqt_console rqt_console
ros2 run turtlesim turtlesim_node
ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z
```

2.7 TF2

tf2 is the transform library, which lets the user keep track of multiple coordinate frames over time. tf2 maintains the relationship between coordinate frames in a tree structure buffered in time and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time.

Specific reference: Official Tutorials

Let's start by installing the demo package and its dependencies.

```
sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools ros-foxy-tf-transformations
```

- **follow**
- launch starts 2 little turtles, the first little turtle automatically follows the second one

```
ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

- Control the movement of the first little turtle through the keyboard

```
ros2 run turtlesim turtle_teleop_key
```

- **View TF tree**

```
ros2 run tf2_tools view_frames.py
evince frames.pdf
```

- **View the relationship between two coordinate systems**

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

- **View TF relationships on rviz**

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz
```

2.8 URDF

URDF is the Unified Robot Description Format for specifying robot geometry and organization in ROS.

Specific reference: Official Tutorials

- **Complete syntax**

1.4.1 AdaptiveGripper

```
<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"
```

```
        <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"
```

- **Install dependent libraries**

```
sudo apt install ros-foxy-joint-state-publisher-gui ros-foxy-joint-state-publisher
sudo apt install ros-foxy-xacro
```

- **Download the source code**

```
cd ~/dev_ws
git clone -b ros2 https://github.com/ros/urdf_tutorial.git src/urdf_tutorial
```

- **Compiling the source code**

```
colcon build --packages-select urdf_tutorial
```

- **Running the example**

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf
```

2.9 Launch

The launch system in ROS 2 is responsible for helping the user describe the configuration of their system and then execute it as described. The configuration of the system includes what programs to run, where to run them, what arguments to pass them, and ROS-specific conventions which make it easy to reuse components throughout the system by giving them each a different configuration. It is also responsible for monitoring the state of the processes launched, and reporting and/or reacting to changes in the state of those processes.

Launch files written in Python, XML, or YAML can start and stop different nodes as well as trigger and act on various events.

Specific reference: Official Tutorials

Setup

Create a new directory to store your launch files:

```
mkdir launch
```

Writer the launch file

Let’s put together a ROS 2 launch file using the turtlesim package and its executables. As mentioned above.

Copy and paste the complete code into the launch/turtlesim_mimic_launch.py file:

```
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, enter into the directory you created earlier and run the following command:

The syntax format is:

```
ros2 launch <package_name> <launch_file_name>
```

```
cd launch
ros2 launch turtlesim_mimic_launch.py
```

- **launch help**

```
ros2 launch -h
```

- **running node**

```
ros2 launch turtlesim multisim.launch.py
```

- **Check the parameters of the launc 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 the 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
```

- **running components**

```
ros2 launch composition composition_demo.launch.py
```

2.10 Run

run is used to run a single node, component program

- **run help**

```
ros2 run -h
```

- **running node**

```
ros2 run turtlesim turtlesim_node
```

- **Run node with parameters**

```
ros2 run turtlesim turtlesim_node --ros-args -r __node:=turtle2 -r __ns:=/ns2
```

- **Run component container**

```
ros2 run rclcpp_components component_container
```

- **running components**

```
ros2 run composition manual_composition
```

2.11 Package

A package can be considered a container for your ROS 2 code. If you want to be able to install your code or share it with others, then you'll need it organized in a package. With packages, you can release your ROS 2 work and allow others to build and use it easily.

Package creation in ROS 2 uses ament as its build system and colcon as its build tool. You can create a package using either CMake or Python, which are officially supported, though other build types do exist.

Specific reference: Official Tutorials

Creating a workspace

Create a new directory for every new workspace. The name doesn't matter, but it is helpful to have it indicate the purpose of the workspace. Let's choose the directory name ros2_ws, for "development workspace":

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
```

- **pkg help**

```
ros2 pkg -h
```

- **List Feature Packs**

```
ros2 pkg executable turtlesim
```

- **Output a function package executable program**

```
ros2 pkg executable turtlesim
```

- **Create a Python package**

Make sure you are in the src folder before running the package creation command.

```
cd ~/ros2_ws/src
```

The command syntax for creating a new package in ROS 2 is:

```
ros2 pkg create --build-type ament_python <package_name>
# you will use the optional argument --node-name which creates a simple Hello World type executable in the package.
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

- **Build a package**

Putting packages in a workspace is 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 root of your workspace:
cd ~/ros2_ws
# Now you can build your packages:
colcon build
```

- **Source the setup file**

To use your new package and executable, first open a new terminal and source your main ROS 2 installation.

Then, from inside the ros2_ws directory, run the following command to source your workspace:

```
source install/setup.bash
```

Now that your workspace has been added to your path, you will be able to use your new package's executables.

- **Use the package**

To run the executable you created using the --node-name argument during package creation, enter the command:

```
ros2 run my_package my_node
```

← Previous Page | Next Page →

Brief introduction and use of rviz2

Rviz2 is a visualization tool for displaying messages in the robot environment, providing a 3D perspective to view the robot's status and activities. It can help developers better understand the current status and activities of the robot, as well as other visual messages. Rviz2 provides a series of visualization tools that can help developers better understand the status and activities of robots, such as visual coordinate systems, laser scanning messages, point cloud messages, robot models, etc. Using Rviz2, robotic systems can be easily viewed and debugged to better achieve robotic goals.

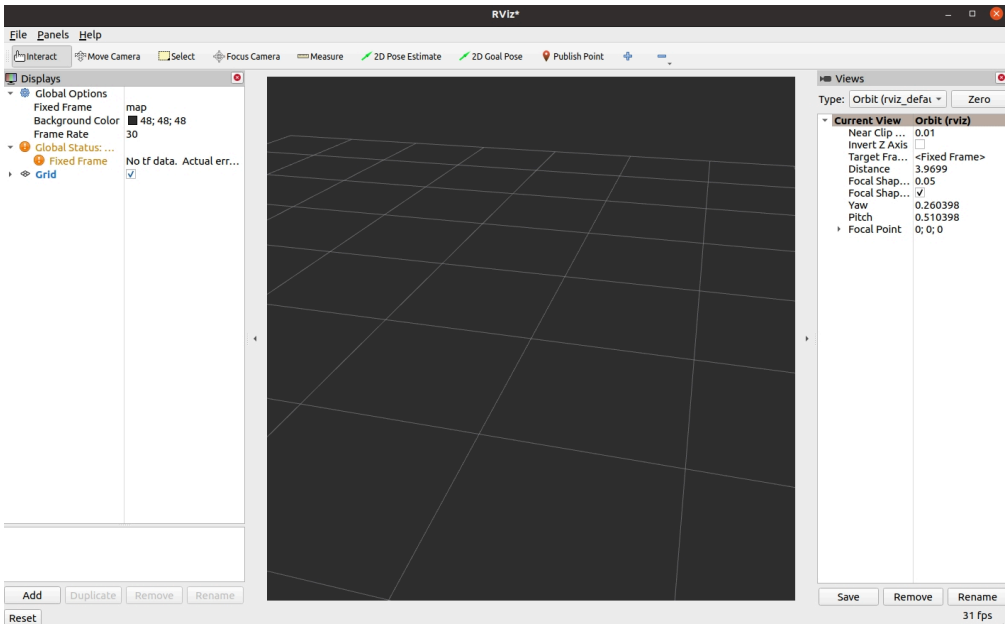
1 Introduction to rviz2

The successful installation of ros2 indicates that rviz2 is also successfully installed together, because the installation of ros2 includes rviz2.

Open a new terminal (shortcut`Ctrl`+`Alt`+`T`)enter the command to open rviz2

```
ros2 run rviz2 rviz2
# 或
rviz2
```

Open rviz2 and display the following interface:



1.1 Introduction of each area

- On the left is the list of monitors, a monitor is something that draws something in the 3D world and may have some options available in the display list. Including functions such as adding, deleting, copying, renaming plug-ins, displaying plug-ins, and setting plug-in properties.
- Above is the toolbar, which allows users to use various function buttons to select tools with multiple functions
- The middle part is the 3D view: it is the main screen where various data can be viewed in 3D. 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.

- The right side is the observation angle setting area, and different observation angles can be set.

We only give a rough introduction in this part. If you want to know more detailed content, you can go to the user guide to view it.

2 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 robotic arm
ls /dev/ttyUSB* # 2020 version myCobot320 M5

# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # 2022 version myCobot320 M5
```

- Grant the serial port permission to the robotic arm:

```
# 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 # 2020 version myCobot320 M5

sudo chmod 777 /dev/ttyACM0 # 2022 version myCobot320 M5
```

Then enter the user password(**Note:** The password is not displayed, just enter it correctly).

3 Simple use

Launch through launch file

This example is based on the fact that you have completed Environment Setup and successfully copied the company's code from GitHub.

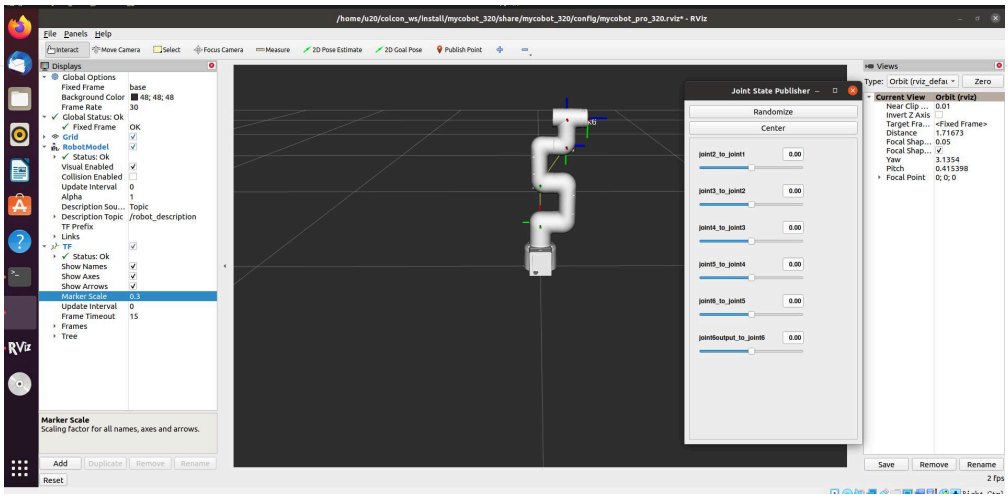
Open a console terminal (shortcut key `Ctrl+Alt+T`) Enter the following command to configure **ROS2 environment**.

```
cd ~/colcon_ws/
colcon build --symlink-install
source install/setup.bash
```

Enter again:

```
ros2 launch mycobot_320 test.launch.py
```

Open rviz2 and get the following result:



If you want to know more information about rviz, you can go to the official documentation to view it.

← Previous Page | Next Page →

Control of the robotic arm

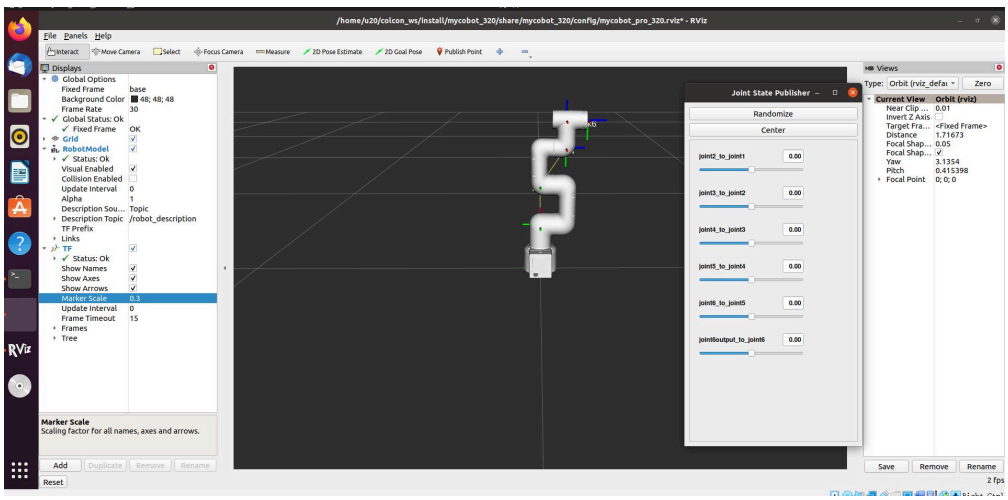
1 Slider Control



Open a command line and run:

```
# The default serial port name of mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port n
ros2 launch mycobot_320 slider_control.launch.py
```

It will **open rviz2 and a slider component** and you will see something like this:



Then you can **control the movement of the model in rviz2 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 input, please make sure that the model in rviz2 does not appear to be worn out before you use the command Do not quickly drag the slider after connecting the robotic arm to prevent damage to the robotic arm

2 Model Follow

In addition to the above controls, we can also **make the model follow the movement of the real robotic arm** .



Open a command line and run:

```
# The default serial port name of mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port n
ros2 launch mycobot_320 mycobot_follow.launch.py
```

It will **open rviz to show the model following effect** .

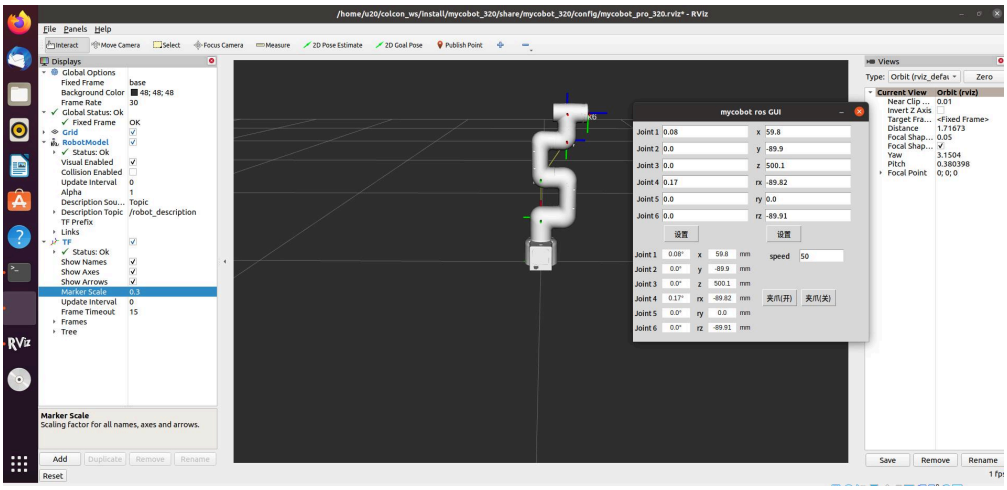
3 GUI Control

On the basis of the previous, this package also **provides a simple GUI control interface** . This method means that the real robotic arms are linked with each other, please connect to mycobot.



Open the command line:

```
# The default serial port name of mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port n
ros2 launch mycobot_320 simple_gui.launch.py
```



4 Keyboard Control

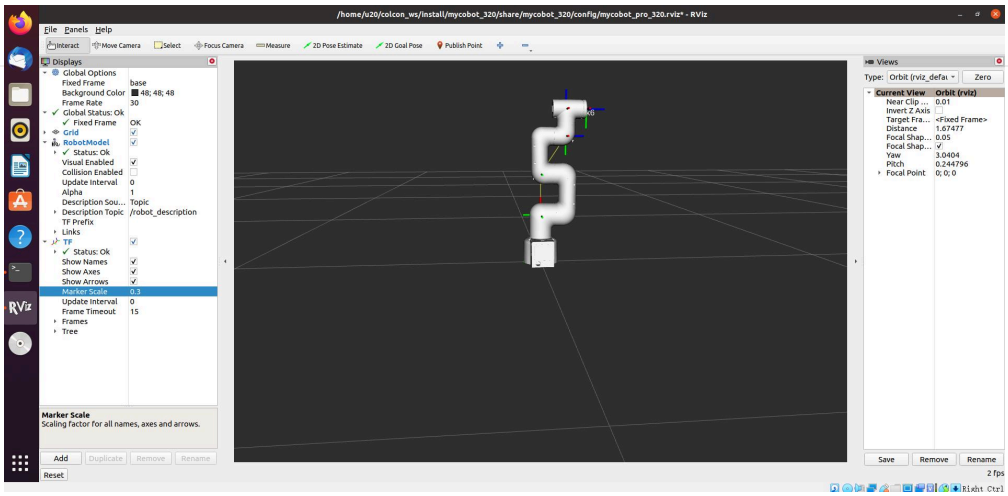
The function of keyboard control is added in the package of `mycobot_320`, and it is synchronized in real time in rviz. This function relies on pythonApi, so make sure to connect with the real robotic arm.



Open a command line and run:

```
# The default serial port name of mycobot 320-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port n
ros2 launch mycobot_320 teleop_keyboard.launch.py
```

The operation effect is as follows:



命令行中将会输出 mycobot 信息，如下：

```
[INFO] [launch]: All log files can be found below /home/u20/.ros/log/2022-08-05-10-43-12-301971-u20-VirtualBox-6290
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [robot_state_publisher-1]: process started with pid [6293]
[INFO] [rviz2-2]: process started with pid [6295]
[INFO] [follow_display-3]: process started with pid [6297]
[robot_state_publisher-1] Parsing robot urdf xml string.
[robot_state_publisher-1] Link link1 had 1 children
[robot_state_publisher-1] Link link2 had 1 children
[robot_state_publisher-1] Link link3 had 1 children
[robot_state_publisher-1] Link link4 had 1 children
[robot_state_publisher-1] Link link5 had 1 children
[robot_state_publisher-1] Link link6 had 0 children
[robot_state_publisher-1] [INFO] [1659667392.703132973] [robot_state_publisher]: got segment base
[robot_state_publisher-1] [INFO] [1659667392.703485410] [robot_state_publisher]: got segment link1
[robot_state_publisher-1] [INFO] [1659667392.703545198] [robot_state_publisher]: got segment link2
[robot_state_publisher-1] [INFO] [1659667392.703571119] [robot_state_publisher]: got segment link3
[robot_state_publisher-1] [INFO] [1659667392.703587512] [robot_state_publisher]: got segment link4
[robot_state_publisher-1] [INFO] [1659667392.703603744] [robot_state_publisher]: got segment link5
[robot_state_publisher-1] [INFO] [1659667392.703619685] [robot_state_publisher]: got segment link6
[rviz2-2] [INFO] [1659667393.588026632] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] [INFO] [1659667393.588472253] [rviz2]: OpenGl version: 3.1 (GLSL 1.4)
[rviz2-2] [INFO] [1659667393.766777360] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] Parsing robot urdf xml string.
[follow_display-3] [INFO] [1659667394.310152595] [follow_display]: port:/dev/ttyUSB0, baud:115200
```

Next, open another command line and run:

```
ros2 run mycobot_320 teleop_keyboard
```

You will see the following output on the command line:

```
Mycobot Teleop Keyboard Controller
---------------------------
Movimg 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: 30    change percent: 2
```

In this terminal, you can control the state of the robot arm and move the robot arm through the keys in the command line.

← Previous Page | Next Section →

C#

Using C# language, you can make developments (coordinate control, angle control, io control, gripper control, etc.) freely through the c# dynamic library provided by our company, and control some of the robots which have been developed by us.

Available for: **myCobot280, 320 and myPalletizer 260**.



What is C#?

C# is an object-oriented programming language derived from C and C++ released by Microsoft, and an advanced programming language running on .NET Framework and .NET Core (completely open source and cross-platform).

C# is obviously different from Java. It draws on a feature of Delphi and is directly integrated with COM (Component Object Model); and it is the protagonist of .NET windows network framework of Microsoft.

C# enables C++ programmers to develop programs efficiently, and because native functions written with C/C++ may be called, the original powerful functions of C/C++ will never be affected. Because of this inheritance

relationship, C# is very similar to C/C++, and the developers familiar with similar languages are able to quickly turn to C#.

Applicable equipment:

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

- myCobot 320
  - myCobot 320 M5

Preconditions for use：

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

Programming development

Some integrated development environments (IDE)

Visual Studio (Visual C#)

MonoDevelop

Next Page →

C# Environment setup

1. Confirm development goals

Mycobot.csharp is a program for serial communication with robots, with simple usage cases. If you want to use C # to freely develop and control the robots that our company has already developed, then it is your choice

Supported robotic arm models: **myCobot320**

Run the recommended software from Mycobot.csharp: vs2019 (developed on Windows) and MonoDevelop (developed on Raspberry Pi robotic arm)

2 Windows Environment Configuration

2.1 Installation vs 2019

Download:

First, download vs2019。

Installation:

After installation is completed, the interface shown below will appear, mainly selecting **.NET desktop development** is sufficient (this is only a suggestion, specific choices can be made according to your own needs, vs2019 installation time is longer)



3. Compile and run the Mycobot.csharp case

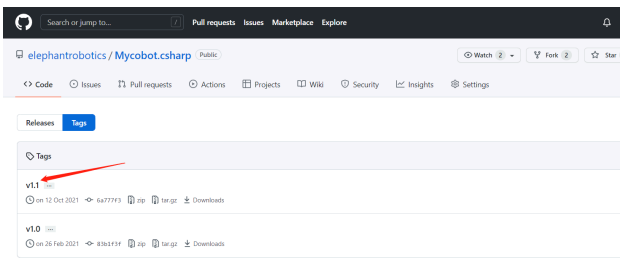
3.1 Source code download

Download from GitHubMycobot.csharp。

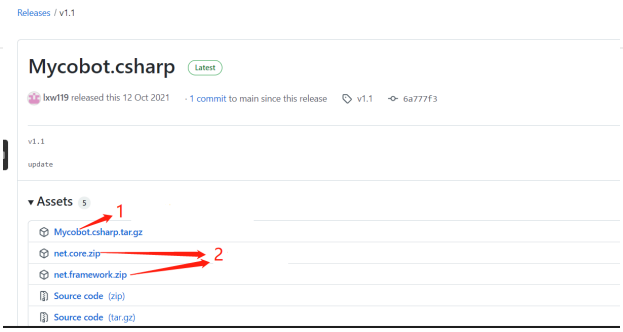
3.2 Dynamic Library Download

Run the case and use it here dynamic library,It encapsulates the API for controlling the robotic arm:

Select the latest version, as shown in the following figure：



Dynamic Library Windows(Windows is divided into. net and. net frameworks. To distinguish them, please refer to the following section for running under Windows)and the Raspberry Pi system version, as shown in the following figure:



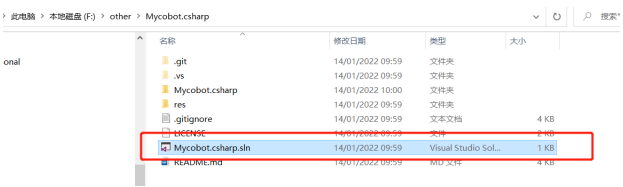
Arrow 1 is applicable to the Raspberry Pi robotic arm system

Arrow 2 applies to Windows systems

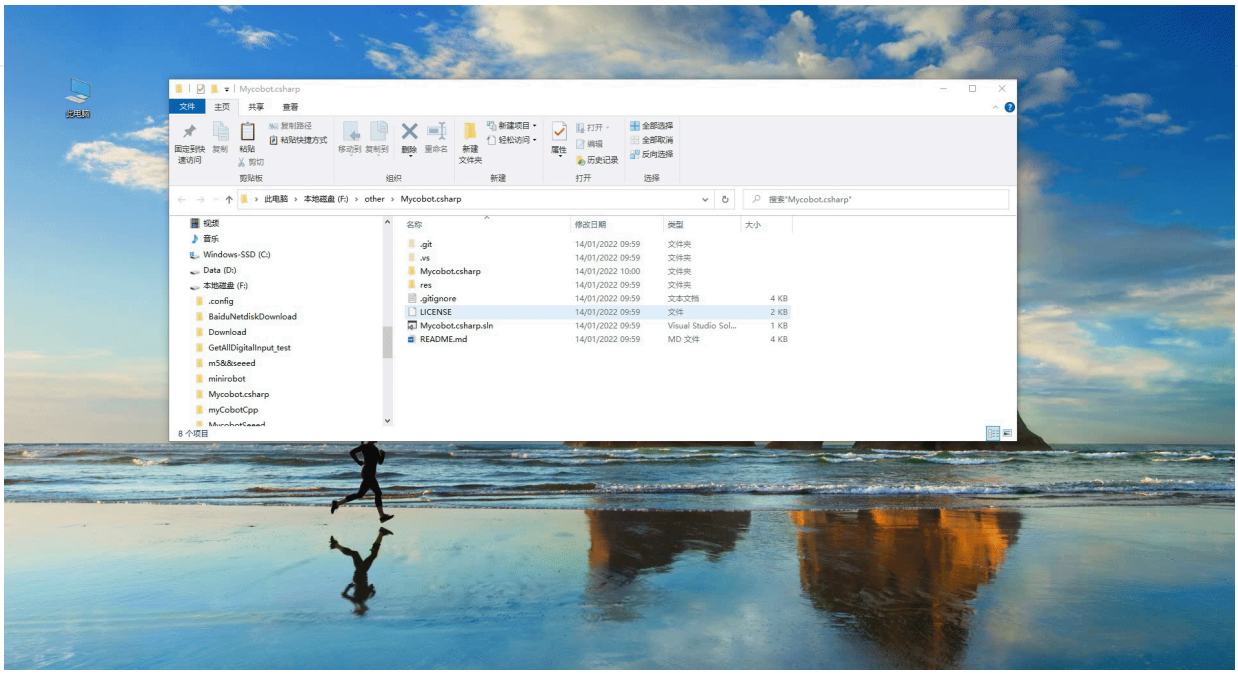
4 Operation

4.1 Example of directly running Mycobot.csharp downloaded from GitHub:

Double click to open Mycobot.csharp.sln (make sure vs2019 is installed on your computer, if not, please refer to 9.1 environment setup)

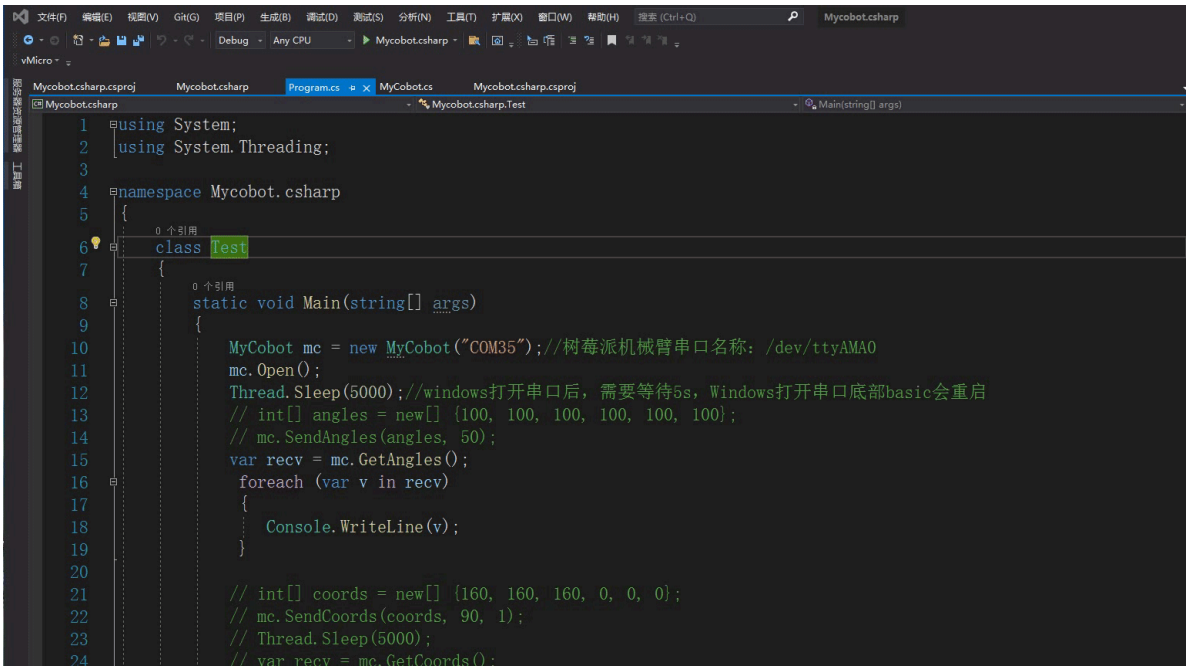


Compile and run the project, check the serial port number of the robotic arm. If it is not consistent with the example, please modify the serial port number, as shown in the following figure:



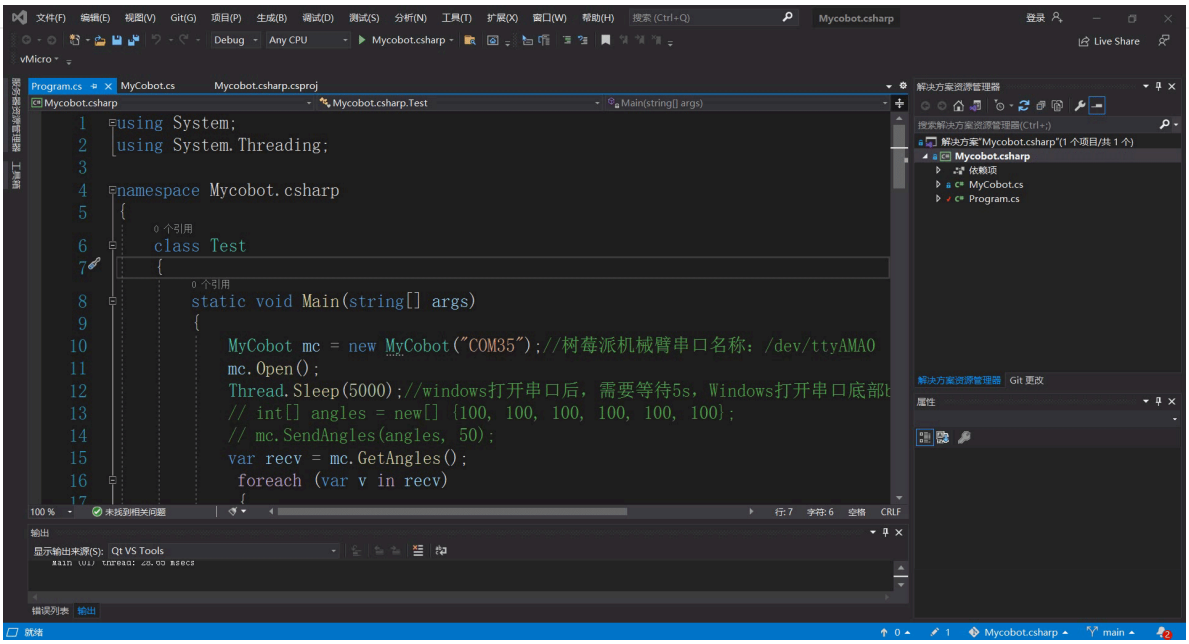
4.2 Call the Mycobot.cshark dynamic library in your own project:

1. Check the target framework of the project and then download the corresponding dynamic library. If the target framework of your project is. net core,download**net core/Mycobot.csharp.dll**If the target framework is. net framework,download**net framework/Mycobot.csharp.dll**)



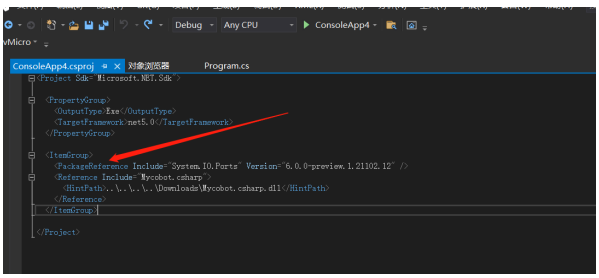
```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    0 个引用
    class Test
    {
        0 个引用
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM35");//树莓派机械臂串口名称: /dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            var recv = mc.GetAngles();
             foreach (var v in recv)
             {
                 Console.WriteLine(v);
             }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
```

Import **Mycobot. csharp. DLL** into the project



2. Add **system. io. ports** to. csproj (project name, which is located in the project directory). Please refer to the image below for details:

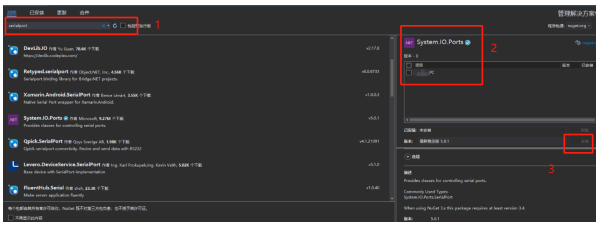
frame: .net core



frame: .net framework

```
  </PropertyGroup>
  <PropertyGroup Condition="'$(Configuration)|$(Platform)' == 'Release|AnyCPU'">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="System.IO.Ports" Version="6.0.0-preview.1.21102.12" />
    <Reference Include="Mycobot.csharp">
      <HintPath>..\..\Mycobot.csharp\Mycobot.csharp\bin\Release\Mycobot.csharp.dll</HintPath>
    </Reference>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data.DataSetExtensions" />
    <Reference Include="Microsoft.CSharp" />
    <Reference Include="System.Data" />
    <Reference Include="System.Net.Http" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Program.cs" />
    <Compile Include="Properties\AssemblyInfo.cs" />
  </ItemGroup>
```

In versions prior to vs2019, as long as using System.IO.Ports You can use SerialPort If an error is displayed: the corresponding type name cannot be found in the namespace, the project needs to be configured with the corresponding DLL, as follows: Tools ->Nuget Package Manager ->Manage Nuget packages for solutions ->Browse, search for the corresponding DLL (such as SerialPort) in the left search bar, check the project you want to add on the right, and click download and install

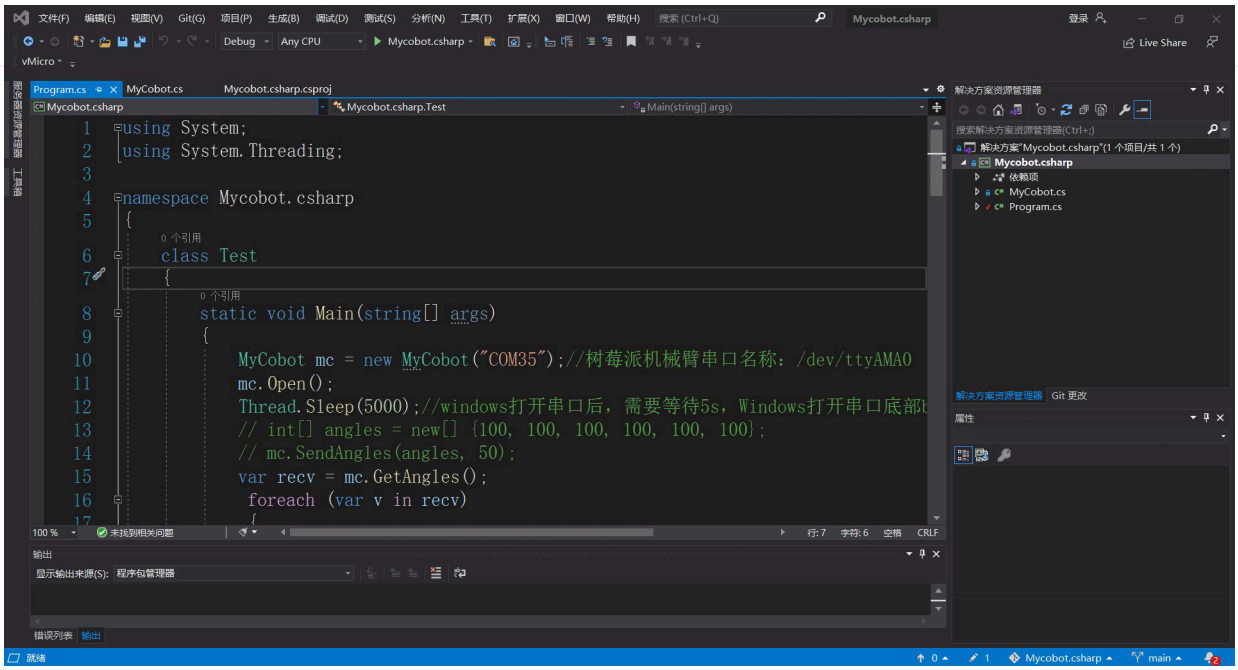


5 Questions

Possible issues encountered during use:

Question 1:System.Runtime, Version=5.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' or one of its dependencies...

Solution: **Update your SDK**(if .net core,update to 5.0 and choose,if .net framework update to 4.0 and choose 4.7.2)Look at the following animation:



Question 2:System.IO.FileNotFoundException:“Could not load file or assembly 'System.IO.Ports, Version=6.0.0.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2ddd51'.

Solution: Check if it has been addedsystem.io.ports to .csproj(Project name, this file is located in the project directory) View the above steps

← Previous Section | Next Page →

Mechanical control

1 Joint control

For serial multi joint robots, joint control is aimed at controlling the variables of each joint of the robotic arm, with the goal of enabling each joint of the robotic arm to reach the target position at a certain speed

1.1 Single joint control

1.1.1 Send single joint angle

SendOneAngle(int jointNo, int angle, int speed)

Return value: None

Parameter Description: Parameter 1: Joint Number (**1-6**), Parameter 2: Angle (Range: **-170 ° -170 °**), Parameter 3: Speed (**0-100**)

Example:

```
mc.SendOneAngle(1, 100,70);
```

1.2 Multi joint control

1.2.1 Obtain all joint angles

GetAngles()

Return value: Returns an int type array, int [], length: 6

Parameter Description: None

Example:

```
var recv = mc.GetAngles();
```

1.2.2 Send all joint angles

SendAngles(int[] angles, int speed)

Return value: None

Parameter Description: Parameter 1: All joint angles (range: **-170 ° -170 °**), Parameter 2: Speed (**0-100**)

Example:

```
int[] angles = new[] {100, 100, 100, 100, 100, 100};
mc.SendAngles(angles ,30);
```

1.3 Complete Use Cases

The program.cs in the project is a complete use case program that can be modified as needed

```
using System;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("/dev/ttyUSB0");
            mc.Open();
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            // Thread.Sleep(5000);
            // var recv = mc.GetAngles();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            mc.SendOneAngle(1, 100,70);

            // byte[] setColor = {0xfe, 0xfe, 0x05, 0x6a, 0xff, 0x00, 0x00, 0xfa};
            mc.Close();
        }
    }
}
```

2 Coordinate control

Coordinate control is to move the robotic arm to a specified point in a specified posture, divided into x, y, z, rx, ry, and rz. X. Y and Z represent the position of the robotic arm head in space (which is a Cartesian coordinate system), while rx, ry, and rz represent the pose of the robotic arm head at that point (which is an Euler coordinate system)

2.1 Single parameter coordinates

2.1.1 Send single parameter coordinates

SendOneCoord(int coord, int value, int speed)

Return value: None

Parameter Description: Parameter 1: Coordinate Number (1-6 (**x, y, z, rx, ry, rz**)), Parameter 2: Coordinate (**X, Y, Z value range -300-300.00 units mm RX, RY, RZ, value range** -180-180), Parameter 3: Speed (0-100)

Example:

```
`mc.SendOneCoord(1, 160,30);`
```

2.2 Multi parameter coordinates

2.2.1 Obtain all coordinates

GetCoords()

Return value: Returns an int type array, int [], length: 6

Parameter Description: None

Example:

```
var recv = mc.GetCoords();
```

2.2.2 Sending multi parameter coordinates

SendCoords (int [] coords, int speed, int mode)

Return value: None

Parameter Description: Parameter 1: All coordinates (**X, Y, Z value range -300-300.00 units mm RX, RY, RZ, value range -180-180**), Parameter 2: Speed (0-100), Parameter 3: Mode (0-angular, 1-linear)

Example:

```
int[] coords = new[] {160, 160, 160, 0, 0, 0};

mc.SendCoords(coords ,30);
```

2.3 Complete Use Cases

The program.cs in the project is a complete use case program that can be modified as needed

```
using System;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("/dev/ttyUSB0");
            mc.Open();
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            // Thread.Sleep(5000);
            // var recv = mc.GetAngles();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            mc.SendOneAngle(1, 100,70);

            // byte[] setColor = {0xfe, 0xfe, 0x05, 0x6a, 0xff, 0x00, 0x00, 0xfa};
            mc.Close();
        }
    }
```

3 IO control

There are pins on both the Basic and Atom pins at the bottom of the robotic arm, which can be controlled by IO to set the high and low levels of the pins and control tools such as pumps (the pin numbers can be viewed from the pin labels attached to the Basic and Atom pins, which are shared for input and output)

3.1 M5Stack basic IO control (m5)

3.1.1 Set the high and low levels of output IO

SetBasicOut(byte pin_number, byte pin_signal)

Return value: None

Parameter Description: Parameter 1: Pin Number (Basic Output Pin Number), Parameter 2: State (0- Low Level, 1- High Level)

Case:

```
mc.SetBasicOut(2, 1);
Thread.Sleep(100);
mc.SetBasicOut(5, 1);
Thread.Sleep(100);
```

3.1.2 Obtaining Input IO Status

GetBasicIn (byte pinnumber)

Return value: Pin status (0- low level, 1- high level)

Parameter Description: Pin Number (Basic Input Pin Number)

Example: Set output pin 2 to high level

```
Console.WriteLine(mc.GetBasicIn(35));
Thread.Sleep(100);
Console.WriteLine(mc.GetBasicIn(36));
Thread.Sleep(100);
```

3.2 Atom io Control

Note: **320m5 does not have atom io, so it is not necessary to use this module API**

3.2.1 Setting the high and low levels of output IO

SetDigitalOut (byte pinnumber, byte pin_signal)

Return value: None

Parameter Description: Parameter 1: Pin Number (Atom Output Pin Number), Parameter 2: State (0- Low Level, 1- High Level)

Case:

```
mc.SetDigitalOut(23, 0);
Thread.Sleep(100);
mc.SetDigitalOut(33, 0);
Thread.Sleep(100);
```

3.2.2 Obtaining Input IO Status

GetDigitalIn (byte pinnumber)

Return value: Pin status (0- low level, 1- high level)

Parameter Description: Pin Number (Atom Input Pin Number)

Case:

```
Console.WriteLine(mc.GetDigitalIn(19));
Thread.Sleep(100);
Console.WriteLine(mc.GetDigitalIn(22));
Thread.Sleep(100);
```

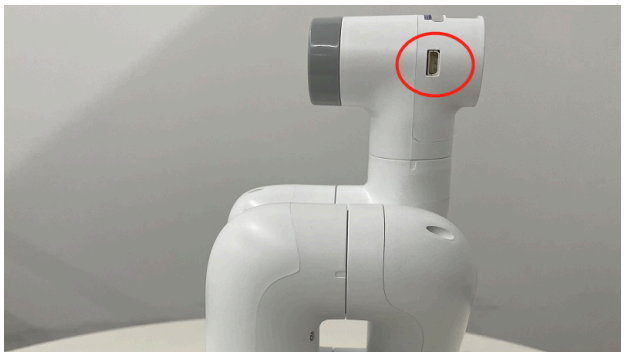
3.3 Complete Use Cases

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启

            //set basic output io
            /*mc.SetBasicOut(2, 1);
            Thread.Sleep(100);
            mc.SetBasicOut(5, 1);
            Thread.Sleep(100);
            mc.SetBasicOut(26, 1);
            Thread.Sleep(100);*/

            //get basic input io
            Console.WriteLine(mc.GetBasicIn(35));
            Thread.Sleep(100);
            Console.WriteLine(mc.GetBasicIn(36));
            Thread.Sleep(100);

            //set atom output io
            /*mc.SetDigitalOut(23, 0);
            Thread.Sleep(100);
            mc.SetDigitalOut(33, 0);
            Thread.Sleep(100);*/

            //get m5 input io
            /*Console.WriteLine(mc.GetDigitalIn(19));
            Thread.Sleep(100);
            Console.WriteLine(mc.GetDigitalIn(22));
            Thread.Sleep(100);*/

            mc.Close();
        }
    }
}
```

4 gripper control

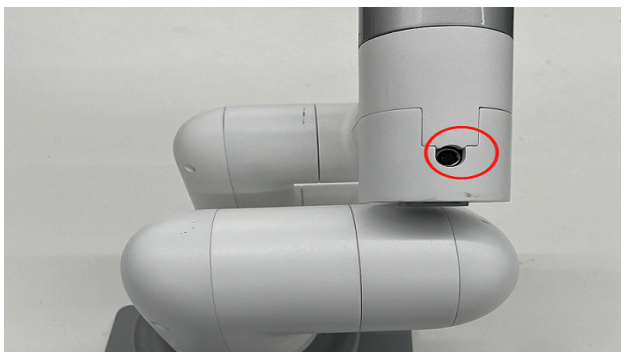
Claw installation:

The adaptive gripper inserts the gripper onto the pins on the atom, as shown in the following figure:



The electric gripper is inserted at the 485 interface on the top, as shown in the following figure:

Attention: **myCobot280 and myPalletizer 260 do not have electric grippers, only myCobot320 has electric grippers**



4.1 Adaptive gripper control

Supported devices: myCobot280, 320&&myPalletizer 260

4.1.1 setGripperValue (byte angle, byte speed)

Return value: None

Call the interface before use Case:

```
mc.SetGripperMode(0);
```

Parameter Description: Parameter 1: Claw opening and closing angle (0-100, 0- closed, 100- maximum opening angle), Parameter 2: Claw opening and closing speed (0-100)

Case:

mc.setGripperValue(0, 10); Thread.Sleep(3000); mc.setGripperValue(50, 100); Thread.Sleep(3000);

4.1.2 getGripperValue ()

Return value: int type, returns the gripper angle (0- closed, 100- maximum open angle)

Parameter Description: None

Case:

```
`Console.WriteLine(mc.getGripperValue());`
```

4.2 Electric gripper control

Supported devices: myCobot320

4.2.1 setEletricGripper (int state)

Return value: None

Parameter description: Claw switch status (0- off, 1- on)

Case:

```
`mc.setEletricGripper(0);`
```

4.3 Complete Use Cases

```
    using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启

            //set gripper open or close 0--close 100-open max 0-100
            mc.setGripperValue(0, 10);
            Thread.Sleep(3000);
            mc.setGripperValue(50, 100);
            Thread.Sleep(3000);

            //set electric gripper
            mc.setEletricGripper(0);
            Thread.Sleep(100);
            mc.setEletricGripper(1);
            Thread.Sleep(100);

            //get gripper state 0--close 1--open
            Console.WriteLine(mc.getGripperValue());
            mc.Close();
        }
    }
}
```

← Previous Section | Next Page →

myCobot API

1 Prerequisite for controlling the robotic arm

1.1 **MyCobot (string port, int baud=115200)**

Function: Instantiate MyCobot

Return value: None

Parameter Description: Parameter 1: Serial Port Number ("COM *" on Windows (such as COM30) Parameter 2: Baud Rate (default is 115200)*

*Note: *If you want to call the following API, you need to first instantiate*

1.2 **Open()**

Function: Open serial port

Return value: None

Parameter Description: None

Attention: **To communicate with the robotic arm, you need to first open the serial port**

1.3 **Close()**

Function: Close serial port

Return value: None

Parameter Description: None

Note: **It is best to close the serial port at the end of the program**

Overall Status of 2 Robots

2.1 **PowerOn();**

Function: Power on the robotic arm

Return value: None

Parameter Description: None

Note: **After powering on the robotic arm, it will not be possible to manually move the arm**

2.2 **PowerOff()**

Function: The robotic arm loses power

Return value: None

Parameter Description: None

Attention: **After powering on the robotic arm, if you want to manually move it, you can use this API to make the robotic arm lose power**

3 Input Program Control Mode MDI Mode and Robot Control (Manual Data Input)

3.1 **SendOneAngle(int jointNo, int angle, int speed)**

Function: Send single joint angle

Return value: None

Parameter Description: Parameter 1: Joint Number (**1-6**), Parameter 2: Angle (Range: **-170 ° -170 °**), Parameter 3: Speed (**0-100**)

Use case: Click to view)。

3.2 **GetAngles()**

Function: Obtain all joint angles

Return value: Returns an int type array, int [], length: 6

Parameter Description: None

Use case: Click to view)。

3.3 **SendAngles (int [] angles, int speed)**

Function description: Send all joint angles

Return value: None

Parameter Description: Parameter 1: All joint angles (range: **-170 ° -170 °**), Parameter 2: Speed (**0-100**)

Use case: Click to view)。

3.4 **GetCoords()**

Function: Obtain all coordinates

Return value: Returns an int type array, int [], length: 6

Parameter Description: None

Use case: Click to view)。

3.5 **SendCoords(int[] coords, int speed, int mode)**

Function: Send multi parameter coordinates

Return value: None

Parameter Description: Parameter 1: All coordinates (X, Y, and Z values range from -300 to 300.00 units in mm RX, RY, RZ, value range **-180**), Parameter 2: Speed (**0 to 100**), Parameter 3: Mode (**0- angular, 1- linear**)

Use case: Click to view)。

3.6 **SendOneCoord(int coord, int value, int speed)**

Function: Send single parameter coordinates

Return value: None

Parameter Description: Parameter 1: Coordinate Number (1-6 (**x, y, z, rx, ry, rz**)), Parameter 2: Coordinate (X, Y, Z value range -300-300.00 units mm RX, RY, RZ, value range **-180-180**), Parameter 3: Speed (**0-100**)

Use case: Click to view)

4 Atom End IO control Atom IO Control

4.1 **SetDigitalOut(byte pin_number, byte pin_signal)**

Function: Set the high and low levels of output IO

Return value: None

Parameter Description: Parameter 1: Pin Number (Atom Output Pin Number), Parameter 2: State (0- Low Level, 1- High Level)

Use case: Click to view)

4.2 **GetDigitalIn(byte pin_number)**

Function: Get input IO status

Return value: Pin status (0- low level, 1- high level)

Parameter Description: Pin Number (Atom Input Pin Number)

Use case: Click to view)

4.3 **SetGripperMode(byte mode)**

Function: Set Gripper Mode

Return value: None

Parameter Description: Parameter 1: mode:1-transparent mode, 0-IO mode

4.4 **setGripperValue(byte angle, byte speed)**

Function: Control adaptive gripper

Return value: None

Parameter Description: Parameter 1: Claw opening and closing angle (0-100, 0- closed, 100- maximum opening angle), Parameter 2: Claw opening and closing speed (0-100)

Use case: Click to view)

4.5 **SetElectricGriper (byte open)**

Function: Control electric gripper

Return value: None

Parameter description: Claw switch status (0- off, 1- on)

4.6 **getGripperValue ()**

Function: Obtain adaptive gripper angle

Return value: int type, returns the gripper angle (0- closed, 100- maximum open angle)

Parameter Description: None

Use case: Click to view)

5 Base M5Stack basicIO control M5Stack-basic IO Control

5.1 **SetBasicOut(byte pin_number, byte pin_signal)**

Function: Set the high and low levels of output IO

Return value: None

Parameter Description: Parameter 1: Pin Number (Basic Output Pin Number), Parameter 2: State (0- Low Level, 1- High Level)

Use case: Click to view)

5.2 **GetBasicIn(byte pin_number)**

Function: Get input IO status

Return value: Pin status (0- low level, 1- high level)

Parameter Description: Pin Number (Basic Input Pin Number)

Use case: Click to view)

← Previous Section | Next Page →

Use Cases

SendOneAngle()

```
    using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            mc.SendOneAngle(1,20,50); //把一关节移动20，速度50
             Thread.Sleep(5000);
            mc.Close();
        }
    }
}
```

GetAngles()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            var recv = mc.GetAngles();//获取角度信息
            foreach (var v in recv)
             {
                Console.WriteLine(v);
             }
            mc.Close();
        }
    }
}
```

SendAngles()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            mc.SendOneAngle(1,0, 70);
            Thread.Sleep(100);
            mc.Close();
        }
    }
}
```

GetCoords()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            var recv = mc.GetCoords();
            foreach (var v in recv)
            {
            Console.WriteLine(v);
           }
            Thread.Sleep(100);
            mc.Close();
        }
    }
}
```

SendCoords()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            int[] coords = new[] {160, 160, 160, 0, 0, 0}; //参数1：坐标，2：速度，3模式
            mc.SendCoords(coords, 90, 1);
            Thread.Sleep(100);
            mc.Close();
        }
    }
}
```

SendOneCoord()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            mc.SendOneAngle(1,0, 70);
            Thread.Sleep(100);
            mc.Close();
        }
    }
}
```

SetDigitalOut()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            mc.SetDigitalOut(23, 0);
            Thread.Sleep(100);
            mc.SetDigitalOut(33, 0);
            Thread.Sleep(100);
            mc.Close();
        }
    }
}
```

GetDigitalIn()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            int num = GetDigitalIn(23);
            mc.Close();
        }
    }
}
```

setGripperValue()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            mc.SetGripperMode(0);//透传
            Thread.Sleep(1000);
            mc.setGripperValue(0,10)
            mc.Close();
        }
    }
}
```

getGripperValue()

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            mc.SetGripperMode(0);//透传
            Thread.Sleep(1000);
            int num = mc.getGripperValue();
            mc.Close();
        }
    }
}
```

SetBasicOut()

```
    using System;
    using System.Threading;

    namespace Mycobot.csharp
    {
        class Test
        {
            static void Main(string[] args)
            {
                MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
                mc.Open();
                Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
                mc.SetBasicOut(2, 1);
                Thread.Sleep(100);
                mc.SetBasicOut(5, 1);
                Thread.Sleep(100);
                mc.SetBasicOut(26, 1);
                Thread.Sleep(100);
                mc.Close();
            }
        }
    }
```

GetBasicIn()

```
using System;
using System.Threading;


namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");//树莓派机械臂串口名称：/dev/ttyAMA0
            mc.Open();
            Thread.Sleep(5000);//windows打开串口后，需要等待5s，Windows打开串口底部basic会重启
            int num = mc.GetBasicIn(35);
            mc.Close();
        }
    }
}
```

← Previous Section

C++

Using C++ language, you can make developments (coordinate control, angle control, io control, gripper control, etc.) freely through the C++ dynamic library developed by our company, and control some of the robots which have been developed by us.



What is C++?

C++ is the inheritance of C language. It may be used to carry out procedural programming of C language, object-based programming characterized by abstract data types, and object-oriented programming characterized by inheritance and polymorphism.

While C++ is good at object-oriented programming, it also can be used to carry out process-based programming, so C++ can adapt to the size of the problem.

C++ not only has the practical characteristics of efficient computer operation, but also strives to improve the programming quality of large-scale programs and the problem description ability of programming languages.

Applicable equipment:

- myCobot 320
  - myCobot 320 M5

Preconditions for use：

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

Programming development

Some integrated development environments (IDE)

Visual Studio (Visual C++)

Dev C++

C++ Builder

Compiler

Ultimate++

Digital Mars

C-Free

MinGW

← Previous Section | Next Page →

C++ Environment building

1 Confirming development goals

MycobotCpp is an interface program used for serial communication with mycobot. It calls the mycobot library developed by our company, which contains simple use cases. If you want to make developments via C++ to control the robot that have been developed by us, it is your choice.

Available for: Robot Arm Models: myCobot 280-M5 and myCobot 320-M5.

The software required to run MycobotCpp: vs2019, qt5.12.10, and vsaddin (qt plugin).

2 Windows Environment Configuration

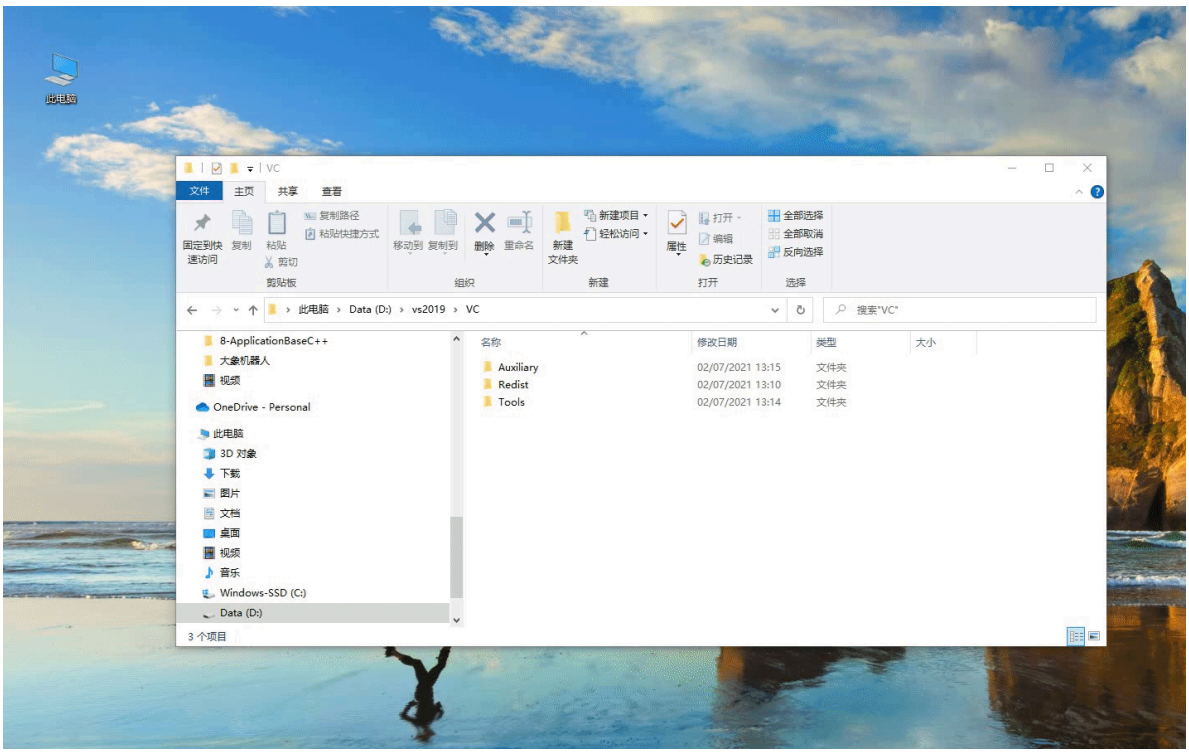
2.1 Installing vs2019

- Downloading: First download vs2019 from the official website.

- Installation: After installation is complete, the interface shown in the figure below will appear. "Universal Windows platform development, desktop development using C++, ASRNET and Web development" are mainly chosen (This is just a suggestion, you can choose according to your own needs. The installation time of vs2019 is long).

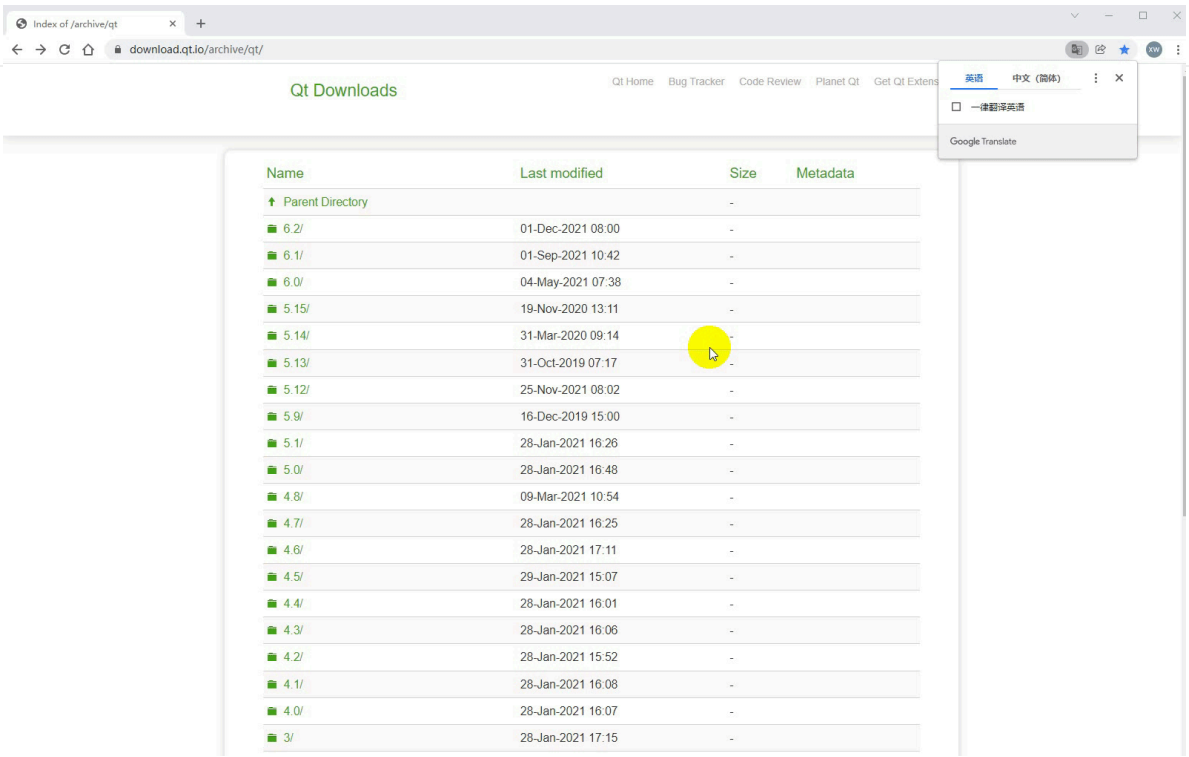


- Environment variable configuration: This Computer -> Right-click Properties -> Advanced System Settings -> Environment Variables -> Look at System Variables, click New-> Variable Name: VCINSTALLDIR variable value: Find the directory where Redist is located (eg: D:2019), as shown in the following figure.

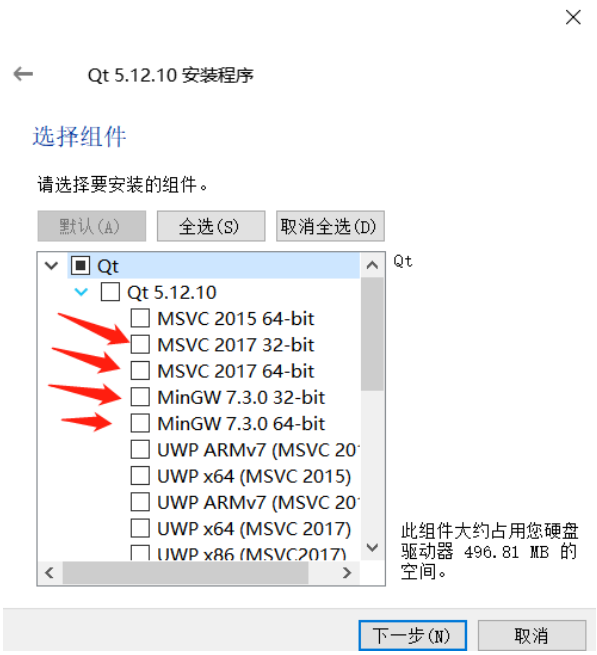


2.2 Installing qt5.12.10

- Downloading: Download qt5.12.10 and above versions. They are all available. The specific operation is shown in the figure below.



- Installation: First log into the qt account. if you haven't an qt account, register first. Next, the interface for selecting components will appear. You may select MinGW and MSVC on Windows, as shown in the following figure.

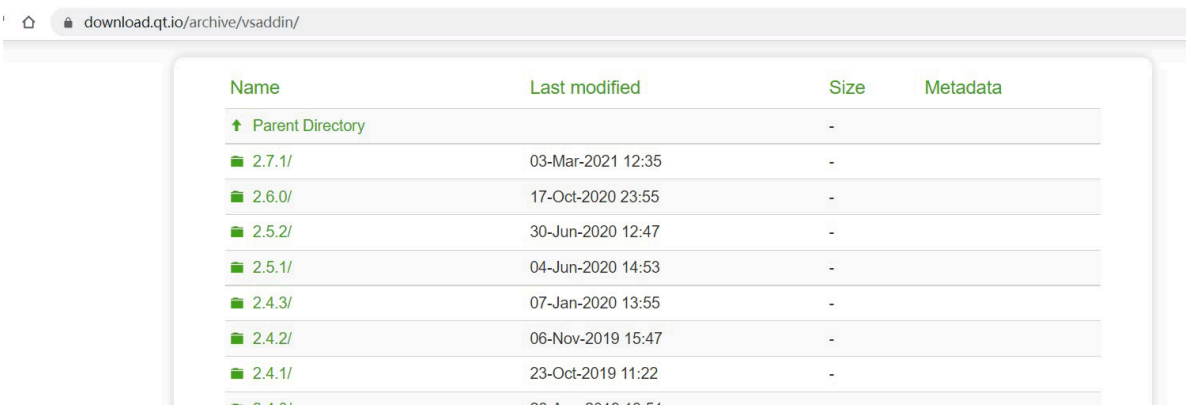


- Environment variable configuration: This Computer -> Right-click Properties -> Advanced System Settings -> Environment Variables -> Look at System Variables, click New-> Variable Name: QTDIR variable value: the directory where msvc2017_64 is located (eg: D:5.12.10\5.12.102017_64. For details, check the installation path in your computer), as shown in the following figure.



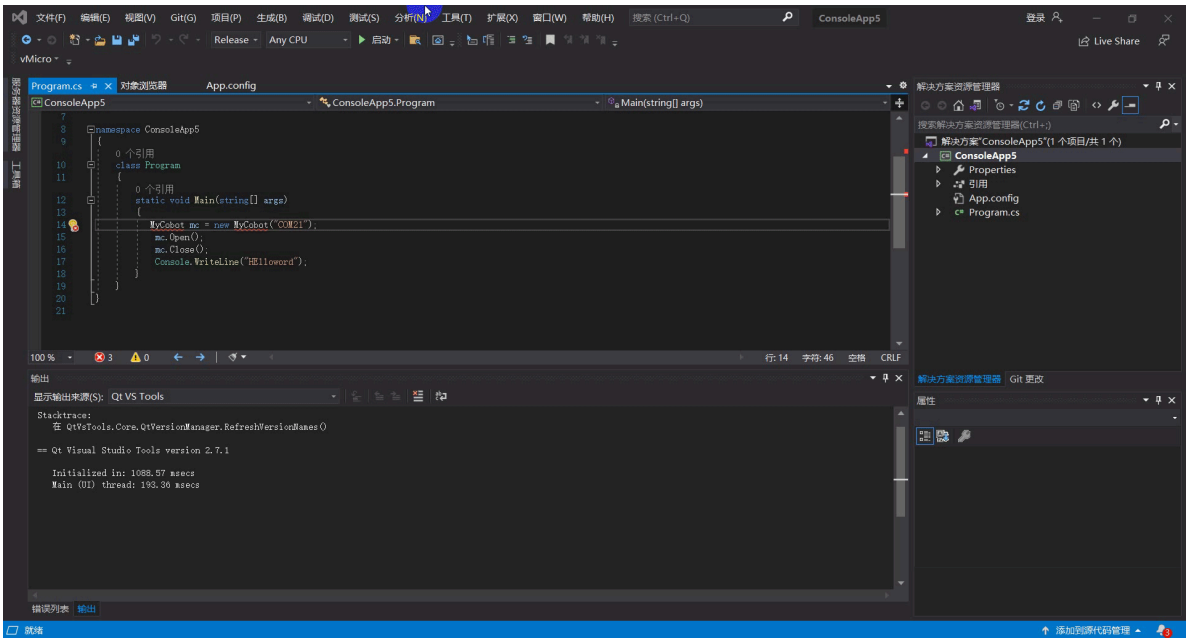
2.3 Installing qt plugin vsaddin

- Downloading: First select the vsaddin version corresponding to vs2019. The specific operation is shown in the figure below.



- installation: direct installation.

- configuration: vs2019 menu bar extension -> QT VS Tools -> QT Versions -> add new qtversion. For Path, select the path where msvc2017_64 is located (eg: D:5.12.10\5.12.102017_64). The specific operation is shown in the following figure:



3 Linux Environment Configuration

3.1 Installing qt5.12.10

- Downloading:

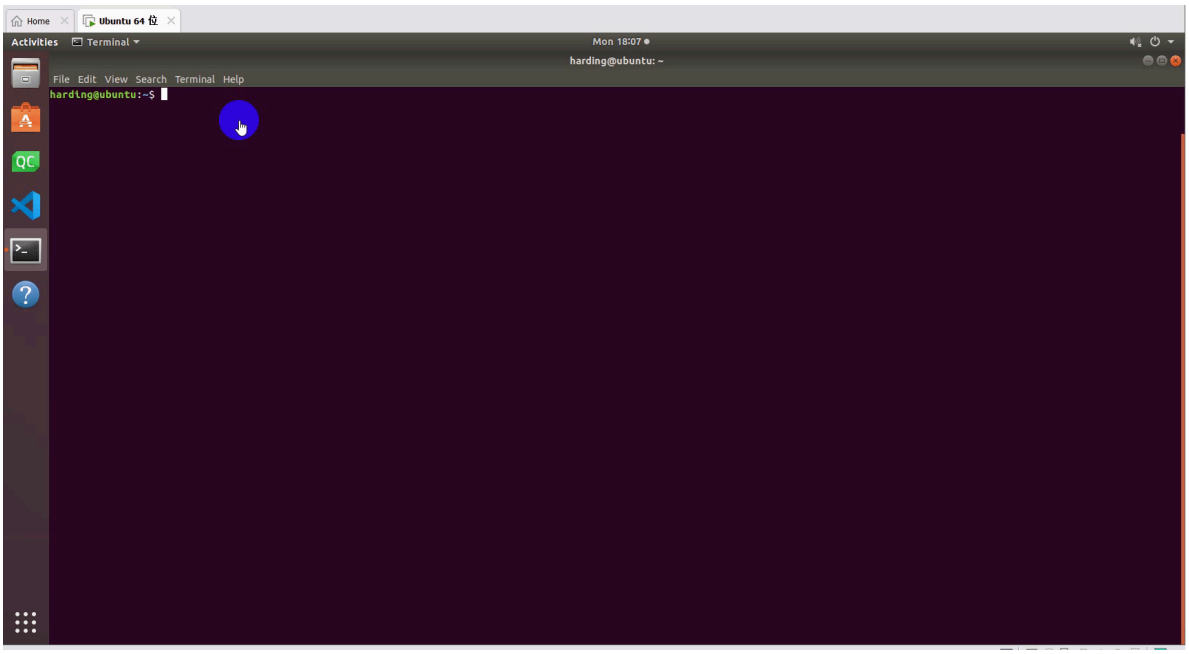
  The download address is the same as Windows. Select the installation package on Linux. For details, see 8.1.2.2 above.

- Installation:

  Command line installation: run ./"installation package name". If there is no execute permission, add execute permission: sudo chmod +x "installation package name", and then enter the graphical interface the same as Windows;

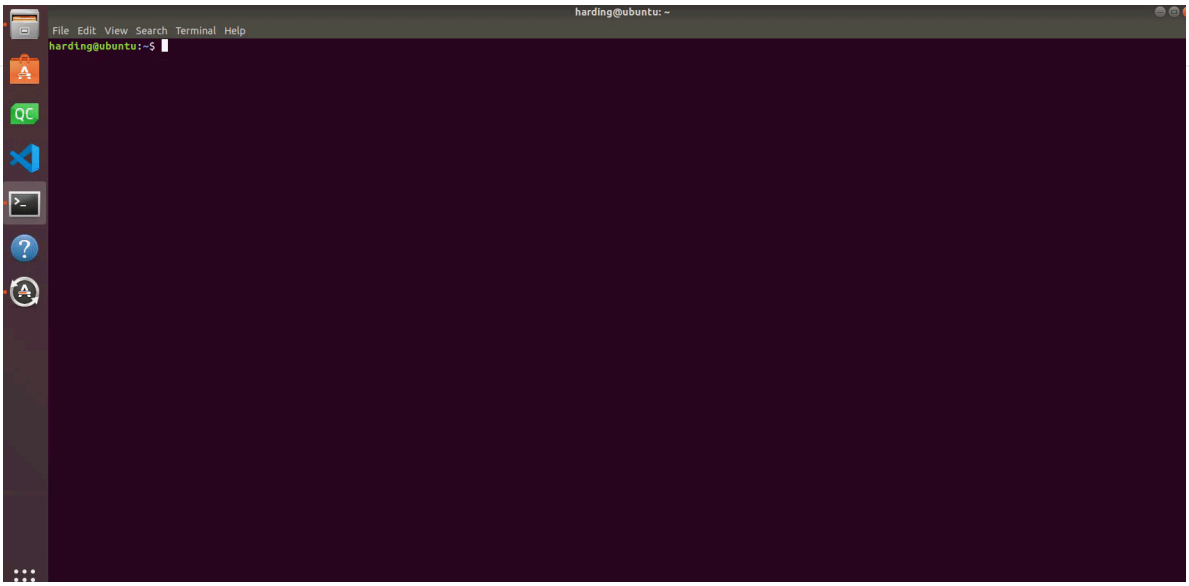
graphical interface installation: the same as Windows.

It is recommended to install qt directly with ordinary user rights. After the installation is successful, you can execute qmake –version, and the following interface appears: installation is successful:



- Configuration:

Open the configuration file, install qt: vi ~/.bashrc for ordinary users, and install qt: vi ~/.profile for root users. Add in the configuration file: export QTDIR=“the directory where qt is located” (eg: export QTDIR=$HOME/Qt/5.12.10/gcc_64), as shown in the following figure:



← Previous Page | Next Page →

Compiling and running of mycobotCpp

1 Downloading

1.1 Downloading source code

Download MycobotCpp on github.

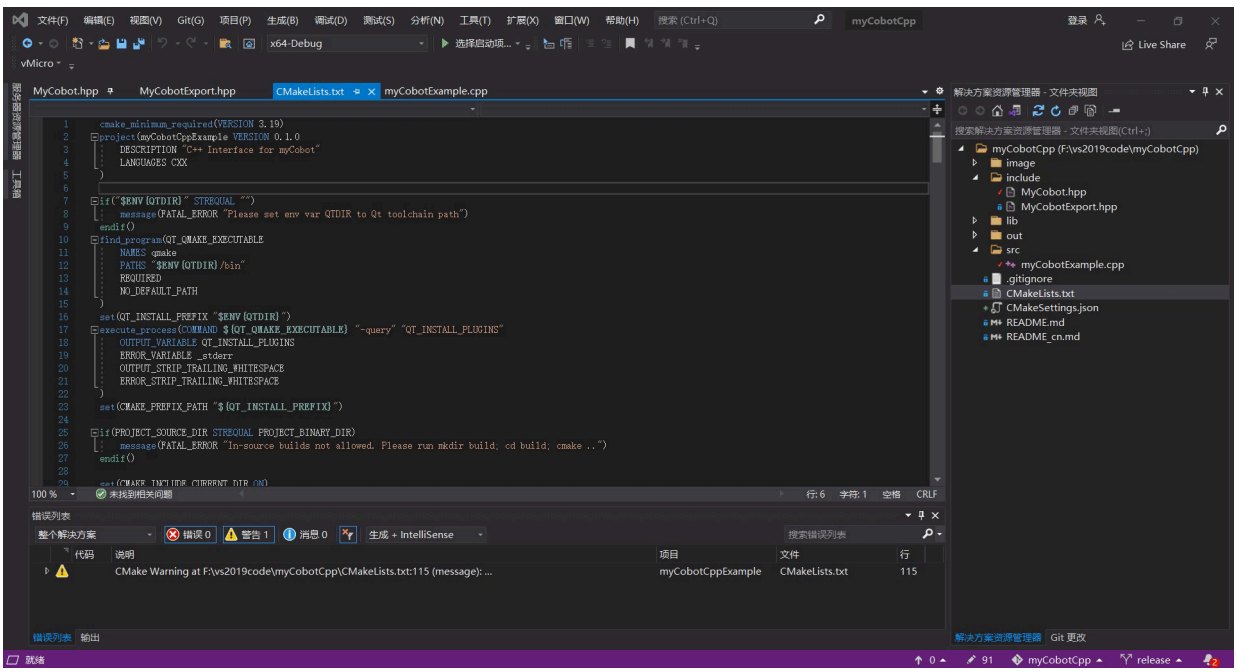
1.2 Dynamic library downloading

Dependency library downloading (Download the latest version, and pay attention to select Windows or Linux; the suffix.zip is the library required by Windows, and .tar.gz is the library required by Linux).

2 Running in Windows

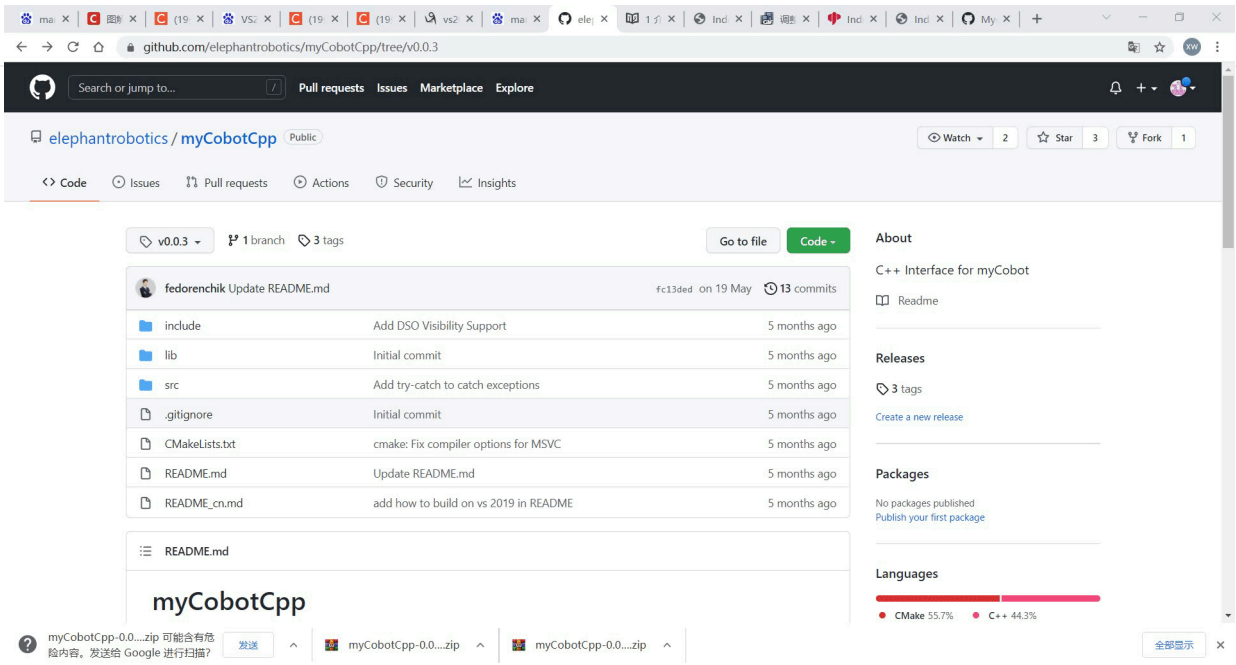
2.1 Compiling

In vs2019, open MycobotCpp, select x64-Release to compile (which is next to the startup item. If there is no x64-Release, click the drop-down box -> Manage configuration to add it), at the same time select release in the configuration of cmake setting, and finally click Generate. Note: Be sure to select x64-Release to compile, as shown in the following figure.



2.2 Running

- Add library files: add .lib to myCobotCpp/lib. Add .lib and .dll to the directory, of myCobotCppExample.exe, at the same level, such as out/build/x64-Release/bin (The two files .lib and .dll are in the myCobotCpp-0.0.3-windows-msvc-x86_64. zipped package).
- Running: Select the startup item (which is next to the green play button and also to the running button), drop down to select myCobotCppEXample.exe (bin.exe), and click Run, as shown in the following figure:



2.3 Common problems and solutions.

Run-time error:

- If myCobotCpp.dll is missing, put myCobotCpp.dll previously saved in the lib directory to the directory where mycobotcppexample.exe is located.
- If QT5Core.dll is reported to be missing, open the qt command (search for QT in the menu bar) and select msvc 2017 64 -bit, execute the directory where windeployqt –release myCobotCppExample.exe is located (for example: windeployqt –release D:201964-Release). If the VS installation path cannot be found after executing the command here, check the settings of VS environment variable.
- After executing the above steps, if the qt5serialport.dll file is reported to be missing, copy the file in the qt installation directory (such as: D:5.12.10\5.12.102017_64) to the directory where myCobotCppExample.exe is located.

3 Running in Linux

3.1 Compiling and building

- mkdir build && cd build
- cmake ..
- cmake –build .

3.2 Run

- copy all .so files to the lib directory;

- command line run: ./bin/myCobotCppExample (Here it is running in the build directory).

4 For example, running on Ubuntu20.04

4.1 Compiling

- mkdir build && cd build

- cmake ..

- cmake –build .

4.2 run

- copy all .so files to the lib directory (Note: after downloading, unzip it. Do not unzip it in Windows, and then copy to Ubuntu. Unzip it directly in Ubuntu, such as: tar -xvf and then drag the file directly to the terminal);

- Soft-link libQt5SerialPort.so.5 (in the QT installation directory, such as: /home/"username"/Qt5.12.10/5.12.10 /gcc_64/lib) to mycobotcpp/build/bin (Do not copy directly). The command is as follows (Be careful to choose your path): ln -s /home/"username"/Qt5.12.10/5.12.10/gcc_64/ lib/libQt5SerialPort.so.5 /home/“username”/myCobotCpp/build/bin/libQt5SerialPort.so.5

4.3 Common problems and solutions

- Error during compilation: QTDIR is not found. Solution: Check whether QTDIR is configured correctly. You may check it in the command line output: echo $QTDIR

- run-time error: Serial port problem: The serial port cannot be opened. Solution: Modify the permission for the serial port of the robot arm. You cannot directly chmod..., because the permission must be modified every time of restart. Modify the file directly:

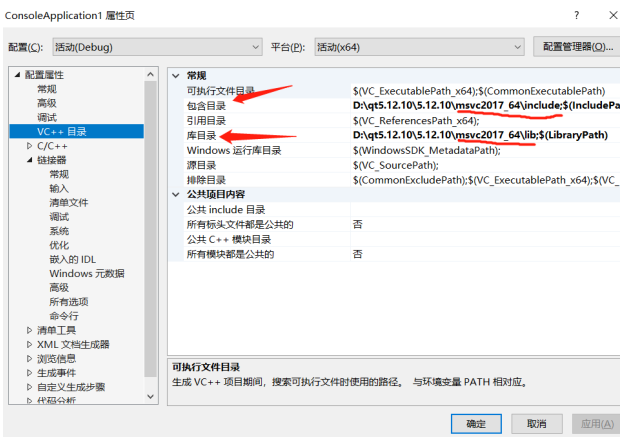
- cd /etc/udev/rules.d
- sudo gedit 20-usb-serial.rules
- Add in the file: KERNEL=="ttyUSB*" MODE="0777"

- The file cannot be found: such as libQt5SerialPort.so.5 could not be opened or found.

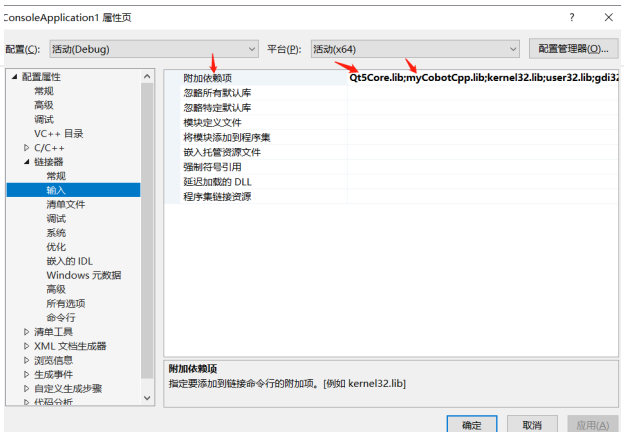
  Solution: Check the running step 2 above.

Notice

If you don't use cmake to compile, and if you use it directly in MFC, do configurations as shown in the figure below.



1.4.1 AdaptiveGripper



← Previous Page | Next Page →

one.Joint control

For a serial multi-joint robot, the joint control is to control the variables of each joint of the robot arm so as to make each joint reaches a target position at a certain speed.

1 Single joint control

1.1 Sending the angle of single joint

WriteAngle(Joint joint, double value, int speed = DefaultSpeed)example

Return value: no parameter

parameter description: Parameter 1: joint number (1-6) Parameter 2: angle (-170°- 170°) Parameter 3: speed (0-100), the example with 30 as a default

example:

`mycobot::MyCobot::I().WriteAngle(mycobot::Joint::J1, 10, 30);

1.2 Obtain the minimum Angle of single joint

GetJointMin(Joint joint) example

Function: to read the minimum angle of a joint

Return value: double type, the minimum angle (the minimum angle that the joint can reach)

Parameter description: joint number (1-6)

example:

```
mycobot::MyCobot::I().GetJointMin(mycobot::Joint::J1);
```

1.3 Get the maximum Angle of single joint

GetJointMax(Joint joint)example

Function: to read the maximum angle of a joint

Return value: double type, the maximum angle (the maximum angle that the joint can reach)

Parameter description: joint number (1-6)

example:

```
mycobot::MyCobot::I().GetJointMax(mycobot::Joint::J1);
```

1.4 Perform a Jog on the specified joint

JogAngle(Joint joint, int direction, int speed = DefaultSpeed)example

Function: to make a joint move until Jogstop or reaching a limited position

Return value: none

Parameter description: Parameter 1: joint number (1-6), Parameter 2: direction (1 - positive direction, and 0 - negative direction), and Parameter 3: speed (Default 30, ranging from 0 to 100). Notice: This API will make the robot arm move in the forward and reverse directions, and it will stop moving after reaching the limited position or when JogStop is called during the movement

example:

```
mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 30);
```

1.5 Jog motion with incremental position on specified joint

JogAngleIncrement(Joint joint, double increment, int speed = DefaultSpeed)example

Function: to make a joint move by a set angle increment

Return value: none

Parameter description: Parameter 1: joint number (1-6), Parameter 2: joint increment value, and Parameter 3: speed (with a default of 30, and ranging from 0 to 100). Notice: The robot arm performs discrete motion; for example, the current coordinate of the joint 1 is -100, and the incremental value is 50. After the movement, the coordinate of the joint 1 will be 50

example:

```
mycobot::MyCobot::I().JogAngleIncrement(mycobot::Joint::J1, 10, 30);
```

2 Multi-joint control

2.1 Get the angles of all joints

GetAngles()example

Return value: Angles type

parameter description: no example:

```
mycobot::Angles angles= mycobot::MyCobot::I().GetAngles();
```

2.2 Send the angles of all joints

WriteAngles(const Angles& angles, int speed = DefaultSpeed)example

Return value: no

parameter description: Parameter 1: all angles (std::array, ranging from -170° to 170°) Parameter 2: speed (0-100), the example with 30 as a default

```
mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteAngles(goal_angles, 30);
mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteAngles(goal_angles, 30);
```

two.Coordinate control

Coordinate control is to make the robot arm move to a specified point with a specified posture, which is divided into x, y, z, rx, ry, rz. X, Y and Z represents the position of the robot arm head in space (The coordinate system is Cartesian coordinate system); [rx,ry,rz] represents the posture of such head at this point (The coordinate system is Euler coordinates).

1 Single parameter control

1.1 Sending single parameter coordinates

WriteCoord(Axis axis, double value, int speed = DefaultSpeed) example

Return value: no

parameter description: Parameter 1: coordinate number (Axis enumeration type, int: 1-6 (X-RZ)); Parameter 2: coordinate (X, Y and Z range from -300 to 300.00 in mm; RX, RY, RZ ranges from -180 to 180); Parameter 3: speed (0-100), the example with 30 as a default

example:

```
mycobot::MyCobot::I().WriteCoord(mycobot::Axis::X, 10, 30);
```

1.2 Jog on the specified axis

JogCoord(Axis axis, int direction, int speed = DefaultSpeed) example

Function: to make the robot arm move in the direction of a coordinate axis

Return value: none

Parameter description: Parameter 1: coordinate number (1-6, xyz rx ry rz), Parameter 2: direction (1 - positive direction, and 0 - negative direction), and Parameter 3: speed (Default is 30, ranging from 0 to 100). Notice: This API will make the robot arm move in the forward and reverse directions of the coordinate axis, and it will stop moving after reaching a limited position or when JogStop is called during the movement

example:

```
mycobot::MyCobot::I().JogCoord(mycobot::Axis::X, 1, 30);
```

1.3 Jog with incremental position on the specified axis

JogCoordIncrement(Axis axis, double increment, int speed = DefaultSpeed)example

Function: to make a coordinate move by a set coordinate increment

Return value: none

Parameter description: Parameter 1: coordinate number (1-6, xyz rx ry rz), Parameter 2: coordinate increment value, and Parameter 3: speed (with a default of 30, and ranging from 0 to 100). Notice: The robot arm performs discrete motion; for example, the current coordinate of X-axis is 100, and the incremental value is 50. After the movement, the coordinate of X-axis will be 150

example:

```
mycobot::MyCobot::I().JogCoordIncrement(mycobot::Axis::X, 1, 30);
```

2 Multi-parameter control

2.1 Get all coordinates

GetCoords() example

Return value: Coords type parameter description: no example: mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();

2.2 Send all coordinates

WriteCoords(const Coords& coords, int speed = DefaultSpeed)example

Return value: no

parameter description: Parameter 1: coordinates (X, Y and Z range from -300 to 300.00 in mm; RX, RY and RZ range from -180 to 180); Parameter 2: speed (0-100), the example with 30 as a default

example:

```
mycobot::Coords goal_coords = {5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteCoords(goal_coords, 30);
mycobot::Coords goal_coords = {5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteCoords(goal_coords, 30);
```

tree.IO control

There are pins on the M5Stack-basic at the bottom of the robot arm and on the Atom at the end. The adsorption pump and other tools can be controlled by setting high and low levels of the pins through io control. For the numbers of input and output pins for each type of robot arm, refer to the following description table.

Table of Description of Input and Output Pins on the M5Stack-basic

Robot arm model	Input pin number	Output pin number
myCobot 280-M5	35、36	2、5、26
myCobot 320-M5	35、36	5、15

Table of Description of Input and Output Pins on the Atom

Robot arm model	Input pin number	Output pin number
myCobot 280-M5	19、22	23、33
myCobot 320-M5	无	无

1 M5Stack-basic io control (m5)

1.1 Setting the high and low levels of output io

SetBasicOut(int pin_number, int pin_signal)example

Return value: no

parameter description: Parameter 1: pin number (basic output pin number); Parameter 2: state (0 - low level; 1- high level)

Case: set the output pin 2 to high level

example:

```
mycobot::MyCobot::I().SetBasicOut(2, 1);
```

1.2 Getting the state of input io

GetBasicIn(int pin_number)example

Return value: pin state (0-low level; 1-high level)

Parameter description: pin number (basic input pin number)

Case:

example:

```
mycobot::MyCobot::I().GetBasicIn(35);
```

2 Atom io Control

Note: 320m5 has no atom io, so the module API is not used.

2.1 Setting the high and low levels of output io

SetDigitalOut(int pin_number, int pin_signal) example

Return value: no

parameter description: Parameter 1: pin number (atom output pin number); Parameter 2: state (0 - low level; 1- high level)

Case:

example:

```
mycobot::MyCobot::I().SetDigitalOut(23, 1);
```

2.2 Getting the state of input io

GetDigitalIn(int pin_number)example

Return value: pin state (0-low level; 1-high level)

Parameter description: pin number (atom input pin number)

Case:

```
mycobot::MyCobot::I().GetDigitalIn(19);
```

four.Other API introductions

1 Instantiate MyCobot

I()

Function: Instantiate MyCobot

return value: MyCobot type, single instance of myCobot object

parameter description: none

Note: When calling the following API, it is not necessary to do separate instantiation, and just call this API

2 Overall running status of Robot

2.1 PowerOn()

Function: to power up the robot arm

Return value: none

parameter description: none

Note: After the robot arm is powered up, it cannot be moved manually

2.2 PowerOff()

Function: to cut off teh power for the robot arm

Return value: none

parameter description: none

Note: After the robot arm is powered up, if you want to move it manually, you may call this API

2.3 SetFreeMoveMode(bool free_move = true)

Function: to set free movement mode

Return value: none

parameter description: enable or disable free movement; true – open free movement, and false – close free automatic

Note: After free movement is enabled, you may move the robot arm manually; and at the same time the light on the atom will turn yellow, and it will turn green when it is turned off

2.4 IsFreeMoveMode()

Function: to check whether the current free movement mode

Return value: bool type, true - free movement is enabled, false - free movement is not enabled

parameter description: none

2.5 IsControllerConnected()

Function: to check whether the system is normal

return value: none

parameter description: bool type, if false is returned, the robot arm cannot be controlled

3 MDI program control mode

3.1 IsInPosition(const Coords& coords, bool is_linear = true)

Function: to check whether the robot arm reaches a specified point (angle or coordinate)

Return value: bool type; if false is returned , it means failing to reach the specified point; and if true is returned, it means having reached the specified point

Parameter description: Parameter 1: all angles or coordinates; Parameter 2: 0 or 1 (Coordinate is 1 (true), and angle is 0 (false))

3.2 IsMoving()

Function: to detect whether the robot arm is moving

Return value: bool type, true - moving, false - not moving

Parameter description: none

3.3 StopRobot()

Function: to stop the movement of the robot arm. When the robot arm is moving, you may stop it by calling this API

Return value: none

Parameter description: none

4 Running Auxiliary Information

4.1 GetSpeed()

Function: to get the speed of the robot arm

Return value: int type, the movement speed of the robot arm (0-100)

Parameter description: none

4.2 SetSpeed(int percentage)

Function: to set the movement speed of the robot arm

Return value: none

Parameter description: the movement speed of the robot arm (0-100)

4.3 GetJointMin(Joint joint)

Function: to read the minimum angle of a joint

Return value: double type, the minimum angle (the minimum angle that the joint can reach)

Parameter description: joint number (1-6)

4.4 GetJointMax(Joint joint)

Function: to read the maximum angle of a joint

Return value: double type, the maximum angle (the maximum angle that the joint can reach)

Parameter description: joint number (1-6)

4.5 SleepSecond(unsigned time)

Function: to wait

Return value: none

Parameter description: time unit is second

← Previous Page | Next Page →

one.Joint control

In this case, the machine will be powered on first, and then the current Angle will be read. After that, the minimum Angle and maximum Angle of joint 1 will be read for single joint Angle control and multi-joint Angle control, and then the specified joint motion, the point motion of absolute position and the point motion of incremental position will be carried out in jog movement. All of the motions mentioned above will print out motion information.

```
int main(int argc, char* argv[]) {
try {
    QCoreApplication a(argc, argv);
    using namespace std::chrono_literals;

    if (!mycobot::MyCobot::I().IsControllerConnected()) {
        std::cerr << "Robot is not connected\n";
        exit(EXIT_FAILURE);
    }

    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn(); // Power on the robot
    mycobot::MyCobot::I().StopRobot(); // Stop the robot
    std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n"; // Check if the robot is moving

    mycobot::Angles angles = mycobot::MyCobot::I().GetAngles(); // Get the current angles of the robot
    std::this_thread::sleep_for(200ms);
    mycobot::Coords coords = mycobot::MyCobot::I().GetCoords(); // Get the current coordinates of the robot
    angles = mycobot::MyCobot::I().GetAngles(); // Get the current angles of the robot
    std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", " << angles[mycobot::J3] << ", "
              << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]";

    mycobot::MyCobot::I().WriteAngle(mycobot::Joint::J1, 10, 30); // Send single joint angle

    while (mycobot::MyCobot::I().IsMoving()) { // Check if the robot has reached the target angle
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "Current joint angles" << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
                  << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
                  << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }


    mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 }; // Define a set of angles
    mycobot::MyCobot::I().WriteAngles(goal_angles); // Send joint angles

    while (mycobot::MyCobot::I().IsMoving()) { // Check if the robot has reached the angles
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "Current angles" << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
                  << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
                  << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }


    mycobot::MyCobot::I().GetJointMin(mycobot::Joint::J1); // Read the minimum joint angle
    std::cout << "Current joint angle" << "[" << angles[mycobot::J1]  << "]" << std::flush;
    std::this_thread::sleep_for(200ms);

    mycobot::MyCobot::I().GetJointMax(mycobot::Joint::J1); // Read the maximum joint angle
```

```
    std::cout << "Current joint angle" << "[" << angles[mycobot::J1]  << "]" << std::flush;
    std::this_thread::sleep_for(200ms);

    // Jog motion on a specified joint
    mycobot::MyCobot::I().JogAngle((mycobot::Joint)0, 100, 5);
    std::this_thread::sleep_for(200ms);
    while (mycobot::MyCobot::I().IsMoving()) { // Check if the motion is completed
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "Performing specified motion, current angle of joint 1" << "[" << angles[mycobot::J1] << "]"
                  << std::flush;
        std::this_thread::sleep_for(200ms);
    }


    // Jog motion with incremental position on a specified joint
    mycobot::MyCobot::I().JogAngleIncrement((mycobot::Joint)0, 5, 180);
    std::this_thread::sleep_for(200ms);

    while (mycobot::MyCobot::I().IsMoving()) { // Check if the motion is completed
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "Performing incremental motion, current angle of joint 1" << "[" << angles[mycobot::J1] << "]"
                  << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    std::this_thread::sleep_for(5000ms); // Delay function
    mycobot::MyCobot::I().StopRobot(); // Stop the robot

    std::cout << "\n";
    exit(EXIT_SUCCESS); // Exit with success
} catch (std::error_code&) {
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
} catch (...) {
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
}

return 0;
}
```

two.Coordinate control

In this case, the machine will be powered on first, and then the current coordinates will be read. After that, the single coordinate control and multiple coordinate control will be carried out, and then the specified joint coordinate movement in jog motion, the coordinate point movement of absolute position and the coordinate point movement of incremental position will be carried out. All of the motions mentioned above will print out motion information.

```
int main(int argc, char* argv[]) {
try {
    QCoreApplication a(argc, argv);
    using namespace std::chrono_literals;

    if (!mycobot::MyCobot::I().IsControllerConnected()) {
        std::cerr << "Robot is not connected\n";
        exit(EXIT_FAILURE);
    }

    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn(); // Power on the robot
    mycobot::MyCobot::I().StopRobot(); // Stop the robot
    std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n";
    mycobot::Angles angles = mycobot::MyCobot::I().GetAngles(); // Get current joint angles
    std::this_thread::sleep_for(200ms);
    mycobot::Coords coords = mycobot::MyCobot::I().GetCoords(); // Get current coordinates
    std::cout << "Coordinates before motion" << "[" << coords[mycobot::Axis::X] << ", " << coords[mycobot::Axis::Y] << ",
        << coords[mycobot::Axis::RX] << ", " << coords[mycobot::Axis::RY] << ", " << coords[mycobot::Axis::RZ] << "]";

    // Send single-parameter coordinates
    mycobot::Coords goal_coords = { 90, 0, 0, 0, 0, 0 }; // Define target coordinates
    mycobot::MyCobot::I().WriteCoord(mycobot::Axis::X, 90, 180); // Send target coordinates
    while (!mycobot::MyCobot::I().IsInPosition(goal_coords, false)) {
        coords = mycobot::MyCobot::I().GetCoords(); // Get current coordinates
        std::cout << "Coordinates after single-parameter motion" << "[" << coords[mycobot::Axis::X] << ", " << coords[myco
            << coords[mycobot::Axis::Z] << ", " << coords[mycobot::Axis::RX] << ", "
            << coords[mycobot::Axis::RY] << ", " << coords[mycobot::Axis::RZ] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    // Send all coordinates
    goal_coords = { 100, 100, 0, 0, 0, 0 }; // Define target coordinates
    mycobot::MyCobot::I().WriteCoords(goal_coords, 30);
    while (!mycobot::MyCobot::I().IsInPosition(goal_coords, false)) {
        coords = mycobot::MyCobot::I().GetCoords(); // Get current coordinates
        std::cout << "Coordinates after all-coordinates motion" << "[" << coords[mycobot::Axis::X] << ", " << coords[mycob
            << coords[mycobot::Axis::Z] << ", " << coords[mycobot::Axis::RX] << ", "
            << coords[mycobot::Axis::RY] << ", " << coords[mycobot::Axis::RZ] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    // Jog motion on a specified coordinate axis
    mycobot::MyCobot::I().JogCoord(mycobot::Axis::X, 1, 30);
    while (mycobot::MyCobot::I().IsMoving()) {
        coords = mycobot::MyCobot::I().GetCoords(); // Get current coordinates
        std::cout << "Coordinates after jogging on specified axis" << "[" << coords[mycobot::Axis::X] << ", " << coords[my
            << coords[mycobot::Axis::Z] << ", " << coords[mycobot::Axis::RX] << ", "
            << coords[mycobot::Axis::RY] << ", " << coords[mycobot::Axis::RZ] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
```

```
    }

    // Jog motion with incremental position on a specified coordinate axis
    mycobot::MyCobot::I().JogCoordIncrement(mycobot::Axis::X, 30, 30);
    while (mycobot::MyCobot::I().IsMoving()) {
        coords = mycobot::MyCobot::I().GetCoords(); // Get current coordinates
        std::cout << "Coordinates after incremental jogging on specified axis" << "[" << coords[mycobot::Axis::X] << ", "
            << coords[mycobot::Axis::Z] << ", " << coords[mycobot::Axis::RX] << ", "
            << coords[mycobot::Axis::RY] << ", " << coords[mycobot::Axis::RZ] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
    std::this_thread::sleep_for(5000ms);
    mycobot::MyCobot::I().StopRobot();

    std::cout << "\n";
    exit(EXIT_SUCCESS);
} catch (std::error_code&) {
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
} catch (...) {
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
}

return 0;
}
```

tree.IO control

In this example, the machine will be powered on first, the io output will be set, 2, 5, 26 are m5 output pins,35, 36 are m5 input pins, and atom output pins 23, 33 output high level, and read the level information of atom input pins 22, 19.

```
int main(int argc, char* argv[])
{
    try {
        QCoreApplication a(argc, argv);
        using namespace std::chrono_literals;
        if (!mycobot::MyCobot::I().IsControllerConnected()) {
            std::cerr << "Robot is not connected\n";
            exit(EXIT_FAILURE);
    }
    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn();
    mycobot::MyCobot::I().SleepSecond(1);

    mycobot::MyCobot::I().SetBasicOut(2, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(5, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(26, 1);
    mycobot::MyCobot::I().SleepSecond(1);

    for (int i = 0; i < 2; i++) {
        std::cout << "35= " << mycobot::MyCobot::I().GetBasicIn(35) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "36= " << mycobot::MyCobot::I().GetBasicIn(36) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }

    /*mycobot::MyCobot::I().SetDigitalOut(23, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetDigitalOut(33, 1);
    mycobot::MyCobot::I().SleepSecond(1);*/

    for (int i = 0; i < 2; i++) {
        std::cout << "22= " << mycobot::MyCobot::I().GetDigitalIn(22) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "19= " << mycobot::MyCobot::I().GetDigitalIn(19) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }

    mycobot::MyCobot::I().StopRobot();
    std::cout << "Robot is moving: " << mycobot::MyCobot::I
    std::this_thread::sleep_for(5000ms);
    mycobot::MyCobot::I().StopRobot();

    std::cout << "\n";
    exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
    } catch (...) {
```

```
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
    }
}
```

← Previous Page | Next Page →

Arduino





What is Arduino?

Arduino is an easy-to-use open source electronic prototype platform that includes hardware (various Arduino-compliant development boards) and software (Arduino IDE and related development pakages).

The hardware (or called development board) consists of a microcontroller (MCU), flash memory (Flash), and a set of general-purpose input/output interfaces (GPIO), etc. You can understand it as a microcomputer motherboard.

The software is mainly composed of Arduino IDE on the PC side, related board support packages (BSP) and rich third-party function libraries. Using Arduino IDE, users may easily download the BSP related to the development board you have and the required function library for writing your program.

What can the MyCobotBasic library do?

MyCobotBasic library is an open source robot control library developed by our company, which can be used only after using the robot developed by our company. Using this library, you can control our robot through Bluetooth, WiFi, serial port, etc. It also supports functions such as external sensors, IIC communication, and LED lights. You can DIY different application scenarios according to your own needs, or you can refer to the MiniRobot sample code or control cases such as angles, coordinates, and grippers we provide. The MiniRobot sample code includes Bluetooth, WiFi, drag teaching, distance sensor and other control-related content.

Applicable equipment:

- myCobot 320
  - myCobot 320 M5

Preconditions for use

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

← Previous Section | Next Page →

Arduino Environment Building

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) / **micro control types of equipment** users according to the operating system you are using, please download the corresponding driver package. After decompressing the package, select the installation package of the corresponding OPERATING system bit.

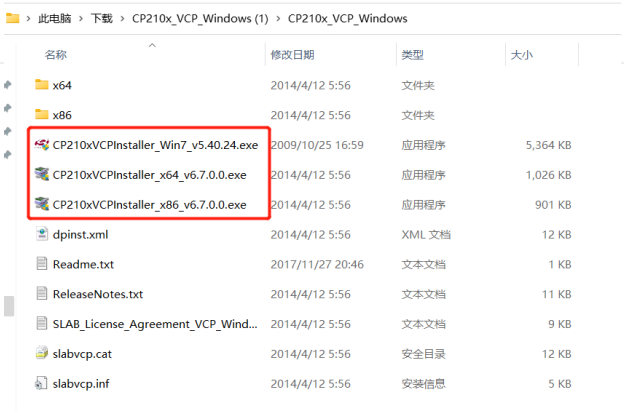
For Mac OS, ensure correct settings of the system **Preferred settings --> Security and privacy --> General** before installation, and allow the user to get it from App Store or an approved developer.

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

CP2104

- **Windows10**
- **MacOS**
- **Linux**

After extracting the zip package, select the corresponding installation package to install according to your computer **operating system** (win10 and win11 select x64 or x86 for installation).

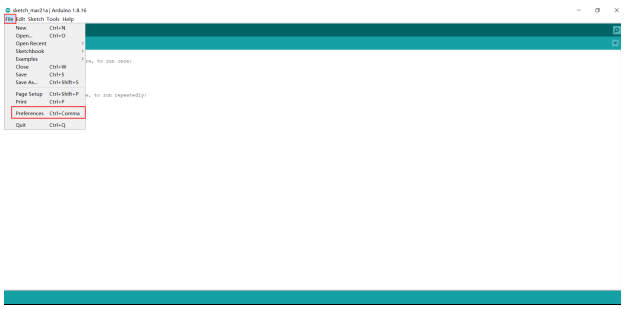


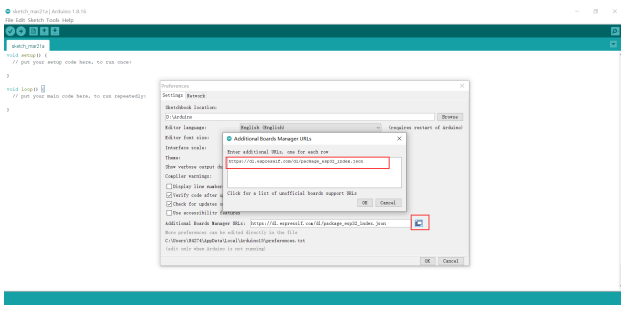
CP34X

- **Windows10**
- **MacOS**

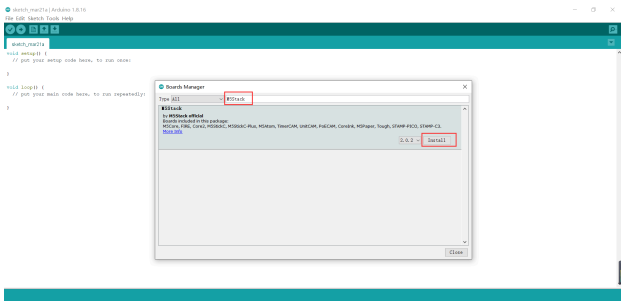
3 Add board

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

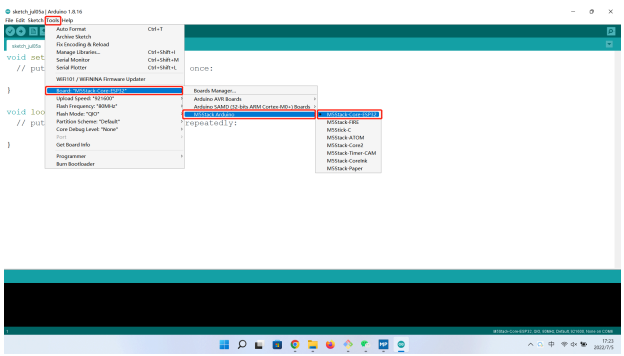




- After adding, select the **Tools --> Board --> Boards Manager**, in the new pop-up dialog, input and search **M5Stack** , click Install (in case of search failure, you can try to restart **Arduino** program), as shown below:



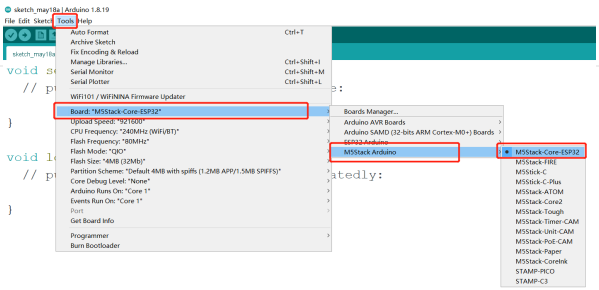
- After adding, select **Tools --> Board**, check whether it is successful, as shown below:



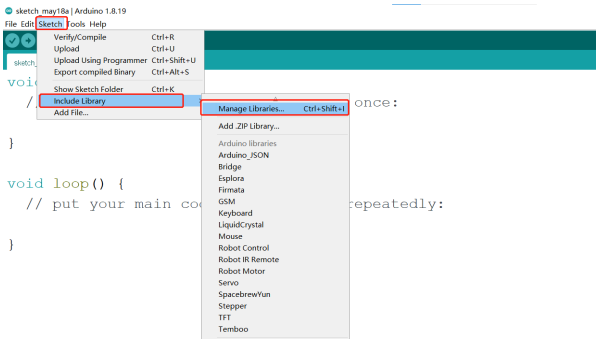
4 Add related libraries

4.1 Install the M5Stack library

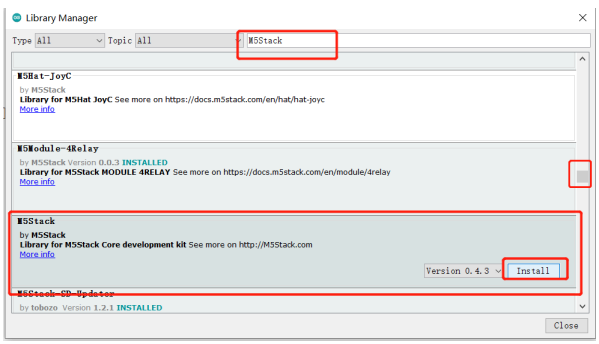
1. Tools --> Development Board --> M5Stack Arduino select **M5Stack-Core-ESP32**, as shown in the following figure：



2. Project --> Load Library --> Manage Library In the search box, enter **M5Stack**, as shown in the figure below：



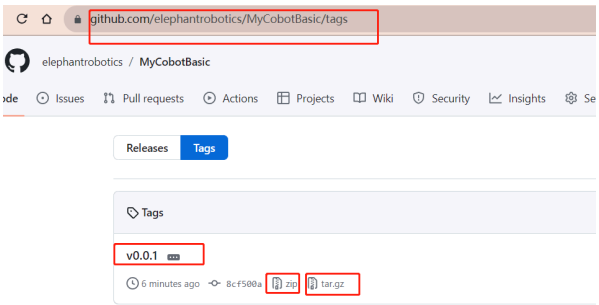
3. After finding it, click Install, scroll down, **M5Stack** is at the back, you can see the location of the drop-down slider in the picture for details, as shown in the figure below：



4.2 Install the MyCobotBasic library

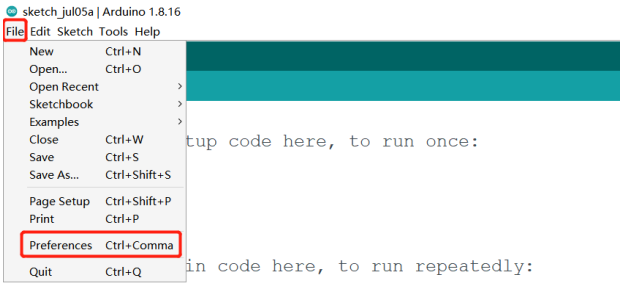
Note: Please download the latest library, the first version is v0.0.1.

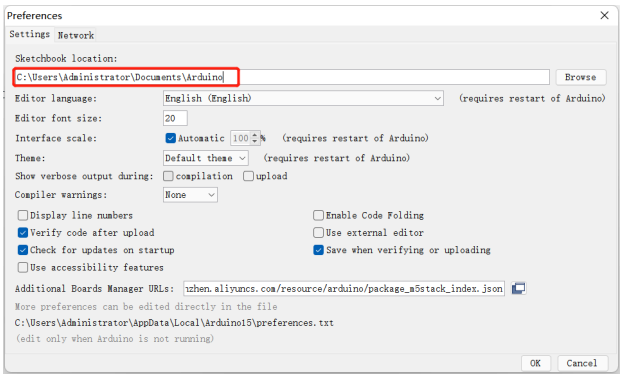
- **MycobotBasic**(After the Mycobot280-Arduino model is imported, you can refer to 10.3-arduinolib_use for use). Please see the figure below for details, .zip is suitable for Windows systems, and .tar.gz is suitable for Linux systems:

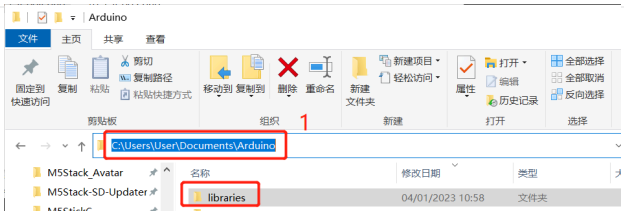


- Library Installation Instructions

  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 libraries folder)

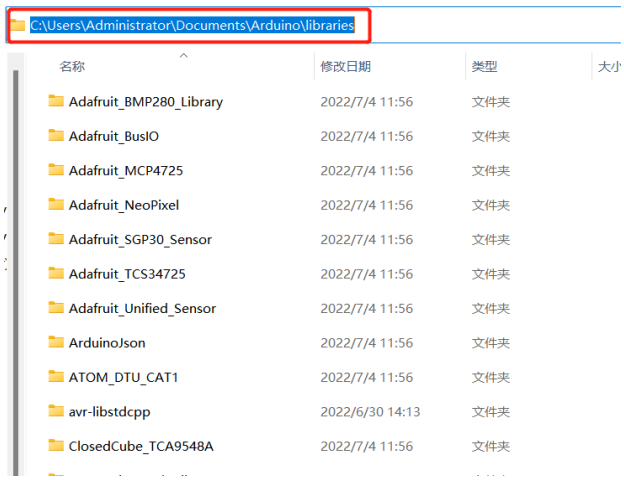






1 Copy the path here and press enter to find the libraries folder

Unzip it into the corresponding folder in the **libraries** directory. If you are using **Arduino**, don't overwrite it, just add to the existing **Library**.



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

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

← Previous Page | Next Page →

Arduino API

1. Overall Status

1.1 `powerOn();`

- Function: The robot is powered on, and the robot can be controlled after the power is turned on（open by default）
- Return Value: None

1.2 `powerOff();`

- Function: Robot power off
- Return Value: None

1.3 `isPoweredOn();`

- Function: Atom status inquiry, return Atom connect status
- Return Value: Power on is TRUE, power off is FALSE

1.4 `getAtomVersion();`

- Function: Get Atom firmware version
- Return Value: Values of type int,the data needs to be /10. For example, if the version number read is 12, it 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 LED at the end will be yellow. Press and hold the atom at the end to move the robot manually
- Parameter Description:
- mode: mode, 0/1, 0--off free movement, 1--open free movement
- Return Value: None

1.6 `isFreeMoveMode();`

- Function: Free movement mode status query, return free movement mode status
- Return Value:Turns TRUE on and FALSE off

2. MDI Mode and Robot Control (Manual Data Input)

2.1 `getAngles();`

- Function: Read all joint angles, when used one Angles should be defined to receive data that was read. Angles are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 angles to store Angle variables, it is used in the same way as arrays.

- Return Value: Arrays of type Angles

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

- Function: Send a single joint Angle
- Parameter Specification：

  Joint Number= joint, range from 1-6; Specified Angle Value= value, range approximately from -170°- + 170°; Specified Speed= speed, range from 1-100

- Return Value: None

2.3 writeAngles(Angles angles, int speed);

- Function: Synchronize joint angles, send joint angles at the same time. Specified Angles is a container with a capacity of 6 data, can be viewed as an array. Use a for loop to assign values, or assign values separately.
- Angles[0] = Specified Angle, Angles[2] = Specify Angle, range from 0-90 (the value range should be the same as writeAngle) unit°

  Movement Speed = speed, range from 0–100 unit %

- Return Value: None

2.4 getCoords();

- Function: Read x,y,z,rx,ry,rz of the end of myCobot, a Coords tempcoords should be defined when used to received angles that was read. Coords are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 tempcoords to store Angle variables, it is used in the same way as arrays.
- Return Value: An array of type Coords. You need to define variables of type Coords.

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

- Function: Send the specific value of the individual coordinate parameters x/y/z, the ends are going to move in a single direction.
- Parameter Specification:

  Value of Moving Path Coordinate = value range from -300–300 (The position coordinates of axis=Axis::X, aixs=Axis::Y and axis=Axis::Z are respectively X,Y,Z, the units would be mm. Position coordinate value range is not uniform, axis=Axis::RX, aixs=Axis::RY and axis=Axis::RZ are respectively RX,RY,RZ ranging from-180°-180°, if the value is beyond the range it will return the clue “inverse kinematics no solution”)

Specified Speed = speed range from 1-100 unit %

- Return Value: None

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

- Function: To send the specified coordinate parameter, which should be of type Coords, declare a variable of type Coords, which is used in the same way as an array
- Parameter Specification

  coords[0] = X, coords[1] = Y, coords[2] = Z,

  X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue ”inverse kinematics no solution” will be given) unit mm

  RX,RY,RZ range from -180-180

  Specified Speed = speed, range from 1-100 unit %

- Return Value: None

2.7 `checkRunning();`

- Function: Check whether the equipment is in motion
- Return Value: In motion is TRUE，on the contrary it's 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 (joint number 7 is generally a gripper, gripper potential value range: 1325-2048); steering gear potential value = encoder value range 0-4096 (the range should be positively related to the range of each joint)
- Return Value: None

2.9 `getEncoder(int joint);`

- Function: Get the specified joint potential value
- Parameter Specification: Servo Motor Number = joint range from 1-7
- Return Value: Int type, range from 0-4096

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

- Function: Set the six joints to run synchronously to the specified position
- Parameter Specification: Need to define a variable of type Angles: angleEncoders, it is used in the same way as arrays. Assign a value to the array angleEncoders, values range from 0-4096 (The range should be positively related to the range of each joint) , the length range of the array is 6. Specified Speed = speed, range from 1-100 unit %
- Return Value: None

2.11 `getEncoders();`

- Function: Get all joint potential values
- Return Value: Arrays of type Angles, range from 0-4096

2.12 `getServoSpeeds();`

- Function: Get all servo speeds
- Return Value: Arrays of type Angles, Velocity is 0 when not in motion

3. JOG Mode

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

- Function: Control the movement of a single joint in one direction
- Parameter Specification:

  Joint/Servo Motor Number = joint, range from 1-6;

  Direction of Joint Motion = Direction, range from -1/1;

  Specified Speed = speed, range from 1-100 unit %
- Return Value: None

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

- Function: Control myCobot moves in one direction in Cartesian space
- Parameter Specification:

  Direction Selection = axis, range from X,Y,Z,RX,RY,RZ;

Direction of Joint Motion = Direction, range from -1/1;

Specified Speed = speed, range from 1-100 unit %

- Return Value: None

3.3 `jogStop();`

- Function: Stops the specified direction of motion that has started
- Return Value: None

3.4 `ProgramPause();`

- Function: Program pause
- Return Value: None

3.5 `ProgramResume();`

- Function: Program continues to run
- Return Value: None

3.6 `TaskStop();`

- Function: Program stop
- Return Value: None

4. Running Status and Settings

4.1 `getSpeed();`

- Function: Read the current running speed
- Return Value: Int tape, range from 1-100, unit %

4.2 `setSpeed(int percentage);`

- Function: Set the running speed
- Parameter Specification: Percentage, range from 1-100, unit %

4.3 `getJointMin(int joint);`

- Function: Read the joint minimal limit Angle
- Parameter Specification: Joint Number = joint, range from 1-6

- Return Value: Float type of the array

4.4 `getJointMax(int joint);`

- Function: Read the joint maximal limit Angle
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: Float type of the array

4.5 `setMovementType(MovementType movement_type);`

- Function: Set the movement method
- Parameter Specification: Movements are non-linear path movement (movej) and linear path movement (movel)
- Return Value: None

4.6 `getMovementType();`

- Function: Read movement method
- Return Value: The non-linear method returns 0; the linear method returns 1

5. Joint Servo Control

5.1 `isServoEnabled(int joint);`

- Function: Check the single joint connection status
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: Connection status, 0/1, 1--connected, 0--not connected

5.2 `isAllServoEnabled();`

- Function: Check whether all joins are properly connected
- Return Value: Connection status, 0/1, 1--connected, 0--not connected

5.3 `getServoData(int joint, byte data_id);`

- Function: Read the system parameters of joint
- Parameter Specification:

  Joint servo serial number = joint, range from 1-6;

  Data Address = data_id, refer to the following Figure for address
- Return Value: Refer to the following Figure for the value range

Address	Function	Range of values	Initial value	Explanation of values
20	LED alarm conditions	0-254	0	Corresponding bit setting 1 to turn on the flashing light alarm Corresponding bit setting 0 to turn off the flashing light alarm
21	Position loop P scale factor	0-254	123 Joints take value 8, 456 take value 5.	Proportionality factor of control motor
22	Position loop D differentiation factor	0-254	123 Joints take value 20, 456 take value 13.	Differential coefficient of control motor
23	Position loop I integration factor	0-254	0	Integral coefficient of the control motor
24	Minimum start-up force	0-1000	0	Set the minimum output starting torque of the servo, set 1000 = 100% * Plugging torque

5.4 `setServoData(int joint, byte data_id, byte data);`

- Function: Set the system parameters of joint
- Parameter Specification:

  Joint servo serial number = joint, range from 1-6;

  Data Address = data_id, refer to the address range in the figure above;

  data = the range of values in the above diagram
- Return Value: None

5.5 `setServoCalibration(int joint);`

- Function:Joint Zero Calibration, the corresponding potential value is 2048
- Parameter Specification: joint number = joint, range from 1-6

5.6 `releaseServo(byte servo_no);`

- Function: Relax/disable a certain joint of the robot
- Parameter Specification: servo_no is 1-6
- Return Value: None

5.7 `releaseAllServos(bool mode);`

- Function:Relax/deactivate all joints of the robot

- Parameter Specification: mode is either true or false
- Return Value: None

5.8 `focusServo(byte servo_no);`

- Function: Enable a certain joint of the robot
- Parameter Specification: servo_no is 1-6
- Return Value: None

5.9 `getServoVoltages();`

- Function: Get all servo voltages
- Return Value: Array of type Angles, reference range from 8.4-12.0

5.10 `getServoStatus();`

- Function: Get the status of all servos
- Return Value: 0 means all states are normal; 1 means voltage overvoltage/undervoltage; 2 means magnetic code state abnormal; 4 means temperature overheating; 8 means current overcurrent; 32 means load overload; when the number that appears is not equal to the above abnormal numbers, for example: 3 means voltage overvoltage/undervoltage and magnetic code state abnormal, 7 means voltage overvoltage/undervoltage, magnetic code state abnormal and temperature overheating

5.11 `getServoTemps();`

- Function: Get all servo temperatures
- Return Value: Array of type Angles, reference range from 0-255

5.12 `getServoCurrents();`

- Function:Get all servo currents
- Return Value:Array of type Angles, reference range from 0-255

6. Atom IO Control

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

- Function: Sets the state mode of the specified pin of atom
- Parameter Specification:

  Pin Serial Number = `pin_no`, reference range from: 19、22、23、26、32、33

  Output Mode = pin_mode, reference range from: 0、1

- Return Value: None

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

- Function: Set the color of the RGB lights of atom
- Parameter Specification:

  Parameter of Red Light = r, range from 0x00 – 0xFF;

  Parameter of Green Light = g, range from 0x00 – 0xFF;

  Parameter of Blue Light = b, range from 0x00 – 0xFF;
- Return Value: None

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

- Function: Set the opening and closing state of the Gripper
- Parameter Description:
- mode, gripper opening and closing mode, range 0/1, 0--the gripper are opened to the maximum, 1--the gripper are closed 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--closed 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 gripper zero point
- Return Value: None

6.6 `getGripperValue();`

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

6.7 `isGripperMoving();`

- Function: Detects if the gripper are in motion
- Return Value: 0 not in motion, 1 in motion

6.8 `void setEletricGripper(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 are opened to the maximum, 1--the gripper are closed to the minimum
- Return value: None

6.9 `void InitEletricGripper();`

- Note: This interface is only available for MyCobot320 robot
- Function: Initialize the opening and closing of the electric gripper. Every time the electric gripper is plugged in, it needs to be initialized before it can be controlled. After the initialization is successful, 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 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. Pins 23 and 33, when turned on or off, both pins The feet need to be set to different states, one must be high and one low)
- Return value: None
- 案例：**GripperControl**

6.11 `bool getGripperMode();`

- Note: This interface is only available for MyCobot320 robot
- Function: Set 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: Setting the operating state of IO pins
- Parameter Specification: 0 input; 1 output; 2 pull_up_input
- Return Value: None

6.13 `getDitialInput(byte pin_no);`

- Function: Read input
- Parameter Specification: Pin Serial Number = pin_no Range of values: 19、22、23、26、32、33
- Return Value: None

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

- Function: Set ATOM end IO to output PWM signal with specified duty cycle
- Parameter Specification:
- pin_no: IO Serial Number

- freq: Clock Frequency

- pin_write: Duty Cycle 0-256; 128 means 50%
- Return Value: None

7. Coordinate Control Mode

7.1 `setToolReference(Coords coords);`

- Function: Set coordinate system of tool
- Parameter Specification:

  X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

  RX,RY,RZ range from -180.00-180.00
- Return Value: None

7.2 `setWorldReference(Coords coords);`

- Function: Set coordinate system of world
- Parameter Specification:

  X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue

"inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

- Return Value: None

7.3 `getToolReference();`

- Function: Get coordinate system of tool
- Return Value:

X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

7.4 `getWorldReference();`

- Function: Get coordinate system of world
- Return Value:

X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

7.5 `setReferenceFrame(RFType rftype);`

- Function: Set coordinate system of frame
- Parameter Specification:

RFType::BASE takes the robot base as the base coordinate, RFType::WORLD takes the world coordinate system as the base coordinate

- Return Value: None

7.6 `getReferenceFrame();`

- Function：Get coordinate system of flange

- Return Value: X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

  RX,RY,RZ range from -180.00-180.00

7.7 `setEndType(EndType end_type)`

- Function: Set coordinate system of the end
- Parameter Specification:

  EndType::FLANGE set the end as flange, EndType::TOOL set the end to the tool end

- Return Value: None

7.8 `getEndType();`

- Function: Get coordinate system of the end
- Return Value: X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

  RX,RY,RZ range from -180.00-180.00

← Previous Page | Next Page →

Partial Case Introduction

Currently, different models support angle, coordinate, and gripper control. MyCobot320 supports adaptive gripper and electric gripper control.

miniRobot:

For the myCobot320m5 use case, it includes zero position calibration, drag teaching, communication, etc. (based on this, control the robotic arm using RoboFlow, python, myblockly, etc.), and information retrieval (get servo atom connection status, as well as basic, atom firmware versions).

Note: For Arduino environment setup and case compilation, refer to our Gitbook documentation (https://docs.elephantrobotics.com/docs/gitbook/10-ArduinoEnv/) and the video on Bilibili (https://www.bilibili.com/video/BV1X3411W7Sn/).

1 Angle Control

This case initializes a MyCobotBasic instance, powers on the robot arm, moves joint 1 at a speed of 30 to 100°, then moves the robot arm to zero at a speed of 50. It waits for 5 seconds for the arm to complete its movement, and the program ends. Let's start by modifying the Chinese characters in the above text.

```
MyCobotBasic myCobot;
Angles angles = {0, 0, 0, 0, 0, 0};

void setup()
{
    myCobot.setup(); // Required API
    delay(100);
    myCobot.powerOn(); // Power on the robot arm
    delay(100);
}

void loop()
{
    myCobot.writeAngle((Joint)1, 100, 30);
    delay(200);
    myCobot.writeAngles(angles, 50);
    delay(5000);
}
```

2 Coordinate Control

This case initializes a MyCobotBasic instance, powers on the robot arm, adjusts all joints to a suitable pose at a speed of 50, then moves the end effector to z=260mm. After the action, it controls the robot arm to reach the pose specified by coords = {215.8,-42.400,209.300,-178.260, 14, -90} at a speed of 30. The parameters are x, y, z, rx, ry, rz, and the program ends.

```
#include <MyCobotBasic.h>
MyCobotBasic myCobot;
Coords coords = {215.8,-42.400,209.300,-178.260, 14, -90};

void setup()
{
    myCobot.setup(); // Required API
    delay(100);
    myCobot.powerOn(); // Power on the robot arm
    delay(100);
    myCobot.writeAngles({2, -7, -89, 15.9, 90, 26}, 50); // Adjust all joints to a suitable pose at a speed of 50
    delay(6000);
}

void loop()
{
    myCobot.writeCoord((Axis)3, 260, 30); // Move the end effector to z=260mm at a speed of 30
    delay(300); // Wait for a while to ensure the robot arm reaches the position
    myCobot.writeCoords(coords, 30); // Move the robot arm to the specified position using coordinate control
    delay(5000);
}
```

3 Electric Gripper Control

This case initializes a MyCobotBasic instance, powers on the robot arm, initializes the electric gripper, opens the gripper, closes the gripper, and the program ends.

```
#include <MyCobotBasic.h>


MyCobotBasic myCobot;


void setup()
{
    myCobot.setup(); // Required API
    delay(100);
    myCobot.powerOn(); // Power on the robot arm
    delay(100);


    myCobot.InitEletricGripper(); // Initialize the electric gripper
    delay(100);
}
void loop()
{
    myCobot.setEletricGripper(0); // Open the gripper
    delay(600);
    myCobot.setEletricGripper(1); // Close the gripper
    delay(600);
}
```

4 Adaptive Gripper Control

This case initializes a MyCobotBasic instance, powers on the robot arm, and initializes the adaptive gripper. Within the initialization, the gripper undergoes one opening and closing operation. Subsequently, the gripper is controlled to reach two specified positions, followed by an opening and closing action, and the program ends.

```
#include <MyCobotBasic.h>


MyCobotBasic myCobot;


void setup()
{
    myCobot.setup(); // Required API
    delay(100);
    myCobot.powerOn(); // Power on the robot arm
    delay(100);
    GipperInit(); // Initialize the adaptive gripper
}
void loop()
{
    myCobot.setGripperValue(80, 50); // Open the gripper to 80° at a speed of 50
    delay(500); // Once in place, proceed to the next step
    myCobot.setGripperValue(20, 50); // Open the gripper to 20° at a speed of 50
    delay(500);
    myCobot.setGripperState(0, 30); // Fully open the gripper at a speed of 30
    delay(600);
    myCobot.setGripperState(1, 30); // Fully close the gripper at a speed of 30
    delay(600);
}


void GipperInit()
{
    myCobot.setGripperMode(0);
    delay(100);
    myCobot.setGripperState(0, 100); // Fully open the gripper at a speed of 100
    delay(200);
    myCobot.setGripperState(1, 100); // Fully close the gripper at a speed of 100
    delay(200);
}
```

← Previous Page | Next Page →

Communication and Message Commands

> **Notice:** It is not necessary to download the ATOM firmware before use, but when switching from Arduino to python programming, it is necessary to re-download the minirobot firmware "Transponder".

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

1 Settings of USB Communication

Make sure the following settings are prepared:

- Interface of mainline: USB Type-C connection PC and Basic
- Port ratio: 115200
- Data bit: 8
- Parity bit: none
- Stop bit: 1

2 Introduction to Command Frame & Sole Instruction

The main PC transmit data via M5Stakc-basic to peripheral PC. The peripheral PC decodes the data like commands with return values and then send the results back to the main PC within 500ms.

3 Formats of Message Commands' Sending and Receiving

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

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

4 Explanation for Commands

The main PC transmit data via M5Stakc-basic to peripheral PC. The peripheral PC decodes the data like commands with return values and then send the results back to the main PC within 500ms.

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

5 Explanation for Sole-Instruction Commands

Powering up

Domain	Explanation	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

Example:

Port transmission: FE FE 02 10 FA

no return value

Power Decreasing and Connection Breaking up

Domain	Explanation	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

Example:

Port transmission: FE FE 02 11 FA

no return value

Checking the Status of Atom

Domain	Explanation	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

Example:

Port transmission: FE FE 02 12 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X12
Data[4]	power on/off	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that Atom is powered on:

port return: FE FE 03 12 01 FA

Power Decreasing

Domain	Explanation	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

Example:

Port transmission: FE FE 02 13 FA

no return value

Robot System Checking: Normal

Domain	Explanation	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

Example:

Port transmission: FE FE 02 14 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X14
Data[4]	connection/connection breaking up	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that Atom is successfully connected:

port return: FE FE 03 14 01 FA

Command Updating Mode (Interpolation Setting/Motion Mode Updating)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X16
Data[4]	connection/connection breaking up	0X01/0X00
Data[5]	end frame	0XFA

Example:

1. Setting updating motion mode:

   Port transmission: FE FE 03 16 01 FA

2. Setting interpolation motion mode:

   Port transmission: FE FE 03 16 00 FA

Free Mode (switch off torsion output)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X1A
Data[4]	turn on/off	01/00
Data[5]	end frame	0XFA

Example:

Setting free motion mode:

Port transmission: FE FE 03 1A 01 FA

Checking whether free mode is set

Domain	Explanation	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

Example:

Port return: FE FE 02 1B FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	command frame	0X1B
Data[4]	turn on/off	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that Atom is in free mode:

port return: FE FE 03 1B 01 FA

Reading Angles (blocking information)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X20
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 20 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X0E
Data[3]	return value: command frame	0X20
Data[4]	high angle of No.1 steering gear	Angle1_high
Data[5]	low angle of No.1 steering gear	Angle1_low
Data[6]	high angle of No.2 steering gear	Angle2_high
Data[7]	low angle of No.2 steering gear	Angle2_low
Data[8]	high angle of No.3 steering gear	Angle3_high
Data[9]	low angle of No.3 steering gear	Angle3_low
Data[10]	high angle of No.4 steering gear	Angle4_high
Data[11]	low angle of No.4 steering gear	Angle4_low
Data[12]	high angle of No.5 steering gear	Angle5_high
Data[13]	low angle of No.5 steering gear	Angle5_low
Data[14]	high angle of No.6 steering gear	Angle6_high
Data[15]	low angle of No.6 steering gear	Angle6_low
Data[16]	end frame	0XFA

Example:

Return value of port: 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 \ 33000 ?(temp – 65536) : temp）/100

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

Other joint angles are counted in a similar way.

Sending Sole Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X21
Data[4]	serial number of steering gear	joint_no
Data[5]	high angle	angle_high
Data[6]	low angle	angle_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Let the No.1 steering gear move to zero position

Port transmission: FE FE 06 21 01 00 00 14 FA

joint number: 1-6

angle_high: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

angle_low: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

no return value

Sending Entire Angles

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0F
Data[3]	command frame	0X22
Data[4]	high angle of No.1 steering gear	Angle1_high
Data[5]	low angle of No.1 steering gear	Angle1_low
Data[6]	high angle of No.2 steering gear	Angle2_high
Data[7]	low angle of No.2 steering gear	Angle2_low
Data[8]	high angle of No.3 steering gear	Angle3_high
Data[9]	low angle of No.3 steering gear	Angle3_low
Data[10]	high angle of No.4 steering gear	Angle4_high
Data[11]	low angle of No.4 steering gear	Angle4_low
Data[12]	high angle of No.5 steering gear	Angle5_high
Data[13]	low angle of No.5 steering gear	Angle5_low
Data[14]	high angle of No.6 steering gear	Angle6_high
Data[15]	low angle of No.6 steering gear	Angle6_low
Data[16]	specified speed	Sp
Data[17]	end frame	0XFA

Example:

Send 0 angle to entire joint, and let the steering gear move to zero position.

Port transmission: FE FE 0F 22 00 00 00 00 00 00 00 00 00 00 00 00 1E FA

angle1_high: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

angle1_low: byte

counting method: angle value is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Other angles are counted in a similar way.

no return value

Reading the Entire Coordinates

Domain	Explanation	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

Example:

Port transmission: FE FE 02 23 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X0E
Data[3]	return value: command frame	0X23
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify high status of z coordinate	z_high
Data[8]	specify low status of z coordinate	z_low
Data[9]	low angle of No.3 steering gear	Angle3_low
Data[10]	high angle of No.4 steering gear	Angle4_high
Data[11]	low angle of No.4 steering gear	Angle4_low
Data[12]	high angle of No.5 steering gear	Angle5_high
Data[13]	low angle of No.5 steering gear	Angle5_low
Data[14]	high angle of No.6 steering gear	Angle6_high
Data[15]	low angle of No.6 steering gear	Angle6_low
Data[16]	end frame	0XFA

Example:

Port return value: FE FE 0E 23 01 BC FD A0 10 15 DC 66 FF 54 DE 21 FA

How to get x coordinate:

temp = x_low + x_high*256

x coordinate =（temp\33000 ?(temp – 65536) : temp)/10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

y coordinate is counted in a similar way.

How to get rx coordinate:

temp = rx_low + rx_high*256

x coordinate =（temp \ 33000 ?(temp – 65536) : temp）/100

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

y coordinate is counted in a similar way.

Sending Parameters of Sole Coordinate

Domain	Explanation	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 high status of the parameters of xyz/rxryrz	xyz/ rxryrz_high
Data[6]	specify low status of the parameters of xyz/rxryrz	xyz/rxryrz_low
Data[7]	specified speed	Sp
Data[8]	end frame	0XFA

Example:

Given that the x coordinate is 200 and target speed is 20,

port transmission: FE FE 06 24 01 07 D0 14 FA

specified axis coordinate: byte

range: 1-6

data type of xyz_high: byte

counting method: value of x/y/z coordinate is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of xyz_low: byte

counting method: value of x/y/z coordinate is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

data type of rxryrz_high: byte

counting method: value of rx/ry/rz coordinate is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of rxryrz_low: byte

counting method: value of rx/ry/rz coordinate is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

no Return Value

Sending Parameters of Entire Coordinate

Domain	Explanation	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 high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	specifyd speed	Sp
Data[17]	mode	0X01
Data[17]	end frame	0XFA

Example:

Given that the coordinate of the end of robotic arm is （150.3，-68.7，101.8，10.18，0，-90）, and speed is 10,

port transmission: FE FE 10 25 05 DF FD 51 03 FA BC 30 00 00 DC D8 0A 01 FA

data type of x_high: byte

counting method: value of x coordinate is multiplied by 10 and get the hexadecimal highbyte

data type of x_low: byte

counting method: value of x coordinate is multiplied by 10 and get the hexadecimal highbyte

y coordinate is counted in a similar way.

data type of rx_high: byte

counting method: value of rx coordinate is multiplied by 10 and get the hexadecimal highbyte

data type of rx_low: byte

counting method: value of rx coordinate is multiplied by 100 and get the hexadecimal highbyte

ry coordinate is counted in a similar way.

no return value

Domain	Explanation	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

Example:

Port transmission: FE FE 02 26 FA

no return value

Judging Whether Program Stops

Domain	Explanation	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

Example:

Port transmission: FE FE 02 27 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X27
Data[4]	suspend/not suspend	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that program suspends,

port return value: FE FE 03 27 01 FA

Program Restoration

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X28
Data[4]	suspend/not suspend	0X01/0X00
Data[5]	end frame	0XFA

Example:

Port transmission: FE FE 02 28 FA

no return value

Program Stops

Domain	Explanation	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

Example:

Port transmission: FE FE 02 29 FA

no return value

Whether Reaching Specified Coordinate

Domain	Explanation	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]	high status of x coordinate/highbyte of angle of No.1 steering gear	x_high/Angle1_high
Data[5]	low status of x coordinate/lowbyte of angle of No.1 steering gear	x_low/Angle1_low
Data[6]	high status of y coordinate/highbyte of angle of No.2 steering gear	y_high/Angle2_high
Data[7]	low status of y coordinate/lowbyte of angle of No.2 steering gear	y_low/Angle2_low
Data[8]	high status of y coordinate/highbyte of angle of No.3 steering gear	z_high/Angle3_high
Data[9]	low status of z coordinate/lowbyte of angle of No.3 steering gear	z_low/Angle3_low
Data[10]	high status of rx coordinate/highbyte of angle of No.4 steering gear	rx_high/Angle4_high
Data[11]	low status of rx coordinate/lowbyte of angle of No.4 steering gear	rx_low/Angle4_low
Data[12]	high status of ry coordinate/highbyte of angle of No.5 steering gear	ry_high/Angle5_high
Data[13]	low status of ry coordinate/lowbyte of angle of No.5 steering gear	ry_low/Angle5_low
Data[14]	high status of rz coordinate/highbyte of angle of No.6 steering gear	rz_high/Angle6_high
Data[15]	low status of rz coordinate/lowbyte of angle of No.6 steering gear	rz_low/Angle6_low
Data[16]	coordinate/angle	0X01/0X00
Data[17]	end frame	0XFA

Example:

Judging whether robotic arm move to zero position:

port transmission: FE FE 0F 2A 00 00 00 00 00 00 00 00 00 00 00 00 00 FA

data type of x_high: byte

counting method: value of x coordinate is multiplied by 10 and converted into integral form to get the hexadecimal highbyte

data type of x_low: byte

counting method: value of x coordinate is multiplied by 10 and converted into integral form to get the hexadecimal lowbyte

y coordinate is counted in a similar way.

data type of rx_high: byte

counting method: value of rx coordinate is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of rx_low: byte

counting method: value of rx coordinate is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

ry coordinate is counted in a similar way.

data type of angle_high: byte

counting method: value angle is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle_low: byte

counting method: value of angle is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Type: byte (temporarily unavailable)

Return Value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X2A
Data[4]	reaching the point/not reaching the point	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that the robotic arm doesn't reach the point:

port return value: FE FE 03 2A 00 FA

Movement Check

Domain	Explanation	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

Example:

port transmission: FE FE 02 2B FA

Return Value: data structure

Domain	Explanation	Data
Data[0]	return value: head frame	0XFE
Data[1]	return value: head frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X2B
Data[4]	in motion/not in motion	0X01/0X00
Data[5]	end frame	0XFA

Example:

Given that the program is running:

port return value: FE FE 03 2B 01 FA

jog-Joint-Oriented Movement

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X30
Data[4]	serial number of steering gear	Joint
Data[5]	direction of steering gear	direction
Data[6]	specified speed	sp
Data[7]	end frame	0XFA

Example:

Given that NO.1 steering gear is revolving clockwise at the speed of 20%:

port return value: FE FE 05 30 01 01 14 FA

range of serial number of joint: 1-6

di: data type of byte, either 0 or 1

sp: data type of byte, ranging from 0 to 100

no return value.

jod-Absolute Control

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X31
Data[4]	serial number of steering gear	Joint
Data[5]	highbyte of angle of steering gear	Angle_high
Data[6]	lowbyte of angle of steering gear	Angle_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Given that No.1 steering gear moves to 45° at the speed of 20%

port transmission: FE FE 06 31 01 11 94 14 FA

range of serial number of joint: 1-6

data type of Angle_high: byte

counting method: value of Angle_high is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of Angle_low: byte

counting method: value of Angle_low is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

sp: data type of byte, ranging from 0 to 100

no return value

jog-Coordinate-Oriented Movement

Domain	Explanation	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 coordinate	axis
Data[5]	direction of steering gear	di
Data[6]	specified speed	sp
Data[7]	end frame	0XFA

Example:

Given that robotic arm moves towards x coordinate at the speed of 20

port transmission: FE FE 05 32 01 01 14 FA

axis ranges from 1 to 6, representing x, y, z,rx, ry, rz

di: data type of byte, either 0 or 1

sp: data type of byte, ranging from 0 to 100

no return value

jog-Stepper Model

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X06
Data[3]	command frame	0X33
Data[4]	serial number of steering gear	Joint
Data[5]	highbyte of angle of steering gear	Angle_high
Data[6]	lowbyte of angle of steering gear	Angle_low
Data[7]	specified speed	sp
Data[8]	end frame	0XFA

Example:

Given that the angle of No.1 steering expand by 45° and revolves at the speed of 20

port transmission: FE FE 06 33 01 11 94 14 FA

serial number of joint ranges from 1 to 6

data type of angle_high: byte

counting method: value angle is multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle_low: byte

counting method: value of angle is multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

sp: data type of byte, ranging from 0 to 100

no return value

jog-Stop

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X34
Data[4]	end frame	0XFA

Example:

Stop moving

port transmission: FE FE 02 34 FA

no return value

Sending Potential

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X3A
Data[4]	serial number of steering gear	Joint
Data[5]	high status of potential	Encoder_high
Data[6]	low status of potential	Encoder_low
Data[7]	end frame	0XFA

Example:

Given that No.5 joint is set to 2048

port transmission: FE FE 05 3A 05 08 00 FA

serial number of joint ranges from 0 to 5

data type of Joint: byte

data type of Encoder_high: byte

counting method: get the high status of potential (in hexadecimal form)

data type of Encoder_low: byte

counting method: get the low status of potential (in hexadecimal form)

no return value

Get the Potential

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X3B
Data[4]	serial number of joint	joint
Data[5]	end frame	0XFA

Example:FE FE 03 3B 02 FA

get the potential of NO.2 steering gear

serial number of joint ranges from 1 to 6

Return Value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0X3B
Data[4]	high status of potential	Encoder_high
Data[5]	low status of potential	Encoders_low
Data[6]	end frame	0XFA

Example:

port transmission: FE FE 04 3B 08 07 FA

How potentials are counted:

potential = low status of potential + high status of potential * 256

Sending Potential of Six Steering Gears

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E/0X0F
Data[3]	command frame	0X3C
Data[4]	highbyte of No.1 steering gear	encoder_1_high
Data[5]	lowbyte of No.1 steering gear	encoder_1_low
Data[6]	highbyte of No.2 steering gear	encoder_2_high
Data[7]	lowbyte of No.2 steering gear	encoder_2_low
Data[8]	highbyte of No.3 steering gear	encoder_3_high
Data[9]	lowbyte of No.3 steering gear	encoder_3_low
Data[10]	highbyte of No.4 steering gear	encoder_4_high
Data[11]	lowbyte of No.4 steering gear	encoder_4_low
Data[12]	highbyte of No.5 steering gear	encoder_5_high
Data[13]	lowbyte of No.5 steering gear	encoder_5_low
Data[14]	highbyte of No.6 steering gear	encoder_6_high
Data[15]	lowbyte of No.6 steering gear	encoder_6_low
Data[16]	specified speed	Sp
Data[17]	end frame	0XFA

Example:

Given that the potential of all is 2048, and the speed is 20

port transmission: FE FE 0F 3C 08 00 08 00 08 00 08 00 08 00 08 00 14 FA

Refer to the chart above for each potential.

data type of encoder_1_high: byte

counting method: convert the potential of No.1 steering gear into integral form and get the highbyte hexadecimal

data type of encoder_1_low: byte

counting method: convert the potential of No.1 steering gear into integral form and get the lowbyte hexadecimal

sp: data type of byte, ranging from 0 to 100

no return value

Domain	Explanation	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

Example:

Port transmission: FE FE 02 3D FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X3D
Data[4]	highbyte of No.1 steering gear	encoder_1_high
Data[5]	lowbyte of No.1 steering gear	encoder_1_low
Data[6]	highbyte of No.2 steering gear	encoder_2_high
Data[7]	lowbyte of No.2 steering gear	encoder_2_low
Data[8]	highbyte of No.3 steering gear	encoder_3_high
Data[9]	lowbyte of No.3 steering gear	encoder_3_low
Data[10]	highbyte of No.4 steering gear	encoder_4_high
Data[11]	lowbyte of No.4 steering gear	encoder_4_low
Data[12]	highbyte of No.5 steering gear	encoder_5_high
Data[13]	lowbyte of No.5 steering gear	encoder_5_low
Data[14]	highbyte of No.6 steering gear	encoder_6_high
Data[15]	lowbyte of No.6 steering gear	encoder_6_low
Data[17]	end frame	0XFA

Example:

Given that all robotic arms are set in zero position,

port return value: FE FE 0E 3D 08 00 08 00 08 00 08 00 08 00 08 00 FA

How to count potential

How potentials are counted:

potential = low status of potential + high status of potential * 256

Reading Potential of Six Steering Gears

Domain	Explanation	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

Example:

Port transmission: FE FE 02 3D FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X3D
Data[4]	highbyte of No.1 steering gear	encoder_1_high
Data[5]	lowbyte of No.1 steering gear	encoder_1_low
Data[6]	highbyte of No.2 steering gear	encoder_2_high
Data[7]	lowbyte of No.2 steering gear	encoder_2_low
Data[8]	highbyte of No.3 steering gear	encoder_3_high
Data[9]	lowbyte of No.3 steering gear	encoder_3_low
Data[10]	highbyte of No.4 steering gear	encoder_4_high
Data[11]	lowbyte of No.4 steering gear	encoder_4_low
Data[12]	highbyte of No.5 steering gear	encoder_5_high
Data[13]	lowbyte of No.5 steering gear	encoder_5_low
Data[14]	highbyte of No.6 steering gear	encoder_6_high
Data[15]	lowbyte of No.6 steering gear	encoder_6_low
Data[17]	end frame	0XFA

Example:

Given that all robotic arms are set in zero position,

port return value: FE FE 0E 3D 00 00 00 00 00 00 00 00 00 00 00 00 FA

How potentials are counted:

potential = low status of potential + high status of potential * 256

Reading Speed

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X40
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 40 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X40
Data[4]	specified speed	Sp
Data[5]	end frame	0XFA

Example:

Given that the speed is 50%

port return value: FE FE 03 40 32 FA

Setting Speed

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X41
Data[4]	specified speed	Sp
Data[5]	end frame	0XFA

Sp: data type of byte, ranging from 0 to 100

Example:

Given that the speed is 50%

port return value: FE FE 03 41 32 FA

Reading the Smallest Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X4A
Data[4]	serial number of steering gear	Joint_number
Data[5]	end frame	0XFA

Example:

Reading the smallest angle of joint 2

Port transmission: FE FE 03 4A 02 FA

serial number of joint ranges from 1 to 6

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X05
Data[3]	return value: command frame	0X4A
Data[4]	serial number of joint	Joint_number
Data[5]	high angle of steering gear	Angle_high
Data[6]	low angle of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

port return value: FE FE 05 4A 02 F9 F2 FA

How smallest angles are counted:

temp = angle1_low+angle1_high*256

Angle1=（temp \ 33000 ?(temp – 65536) : temp）/10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

Reading the Smallest Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X4A
Data[4]	serial number of steering gear	Joint_number
Data[5]	end frame	0XFA

Serial number of joint ranges from 1 to 6

Example:

Reading the largest angle of joint 2

Port transmission: FE FE 03 4B 02 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X05
Data[3]	return value: command frame	0X4B
Data[4]	serial number of joint	Joint_number
Data[5]	high angle of steering gear	Angle_high
Data[6]	low angle of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

port return value: FE FE 05 4B 02 F9 F2 FA

How the largest angles are counted:

temp = angle1_low+angle1_high*256

Angle1=（temp \ 33000 ?(temp – 65536) : temp）/10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

Setting the Smallest Angle

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X05
Data[3]	return value: command frame	0X4C
Data[4]	serial number of joint	Joint_number
Data[5]	high angle of steering gear	Angle_high
Data[6]	low angle of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

Given that the smallest angle of joint 2 is 0

Serial number of joint ranges from 1 to 6

port return value: FE FE 03 4C 32 FA

data type of angle1_high: byte

counting method: angleis multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle1_low: byte

counting method: angleis multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Port transmission: FE FE 05 4C 02 00 00 FA

no return value

Setting the Largest Angle

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X4D
Data[4]	serial number of joint	Joint_number
Data[5]	highbyte of steering gear	Angle_high
Data[6]	lowbyte of steering gear	Angle_low
Data[7]	end frame	0XFA

Example:

Given that the largest angle of joint 2 is 45

Serial number of joint ranges from 1 to 6

port return value: FE FE 03 4D 32 FA

data type of angle1_high: byte

counting method: angleis multiplied by 100 and converted into integral form to get the hexadecimal highbyte

data type of angle1_low: byte

counting method: angleis multiplied by 100 and converted into integral form to get the hexadecimal lowbyte

Port transmission: FE FE 05 4D 02 11 94 FA

no return value

Checking Connection

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X50
Data[4]	serial number of steering gear	Joint_number
Data[5]	end frame	0XFA

Serial number of joint ranges from 1 to 6

Example:

Checking whether No.1 steering gear is connected

Port transmission: FE FE 03 50 01 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	command frame	0X50
Data[4]	serial number of joint	Joint_number
Data[5]	connected/not connected	0X01/0X00
Data[6]	end frame	0XFA

Example:

No.1 steering gear is connected

Port return value: FE FE 04 50 01 01 FA

Checking Whether All Steering Gears Are Powered On

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X51
Data[5]	end frame	0XFA

Example:

Port transmission: FE FE 02 51 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X51
Data[4]	power on/off	0X01/0X00
Data[5]	end frame	0XFA

Example:

Not all steering gears are powered on

Port transmission: FE FE 03 51 01 FA

Reading Servo Parameters

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X53
Data[4]	serial number of joint	Joint_number
Data[5]	data address	data_id
Data[6]	end frame	0XFA

Example:

Reading ratio parameter of P position of No.1 steering gear

Port transmission: FE FE 03 53 01 FA

Serial number of joint ranges from 1 to 6

Data_id: data type byte, referring to the chart below for specific value:

Address	Function	Range	Initial Value	Explanation
20	LED siren	0-254	0	1\0: turn on/off LED siren
21	position loop P	0-254	10	control the ratio coefficient
22	position loop I	0-254	0	control the differential coefficient
23	position loop D	0-254	1	control the integral-action coefficient
24	minimum starting force	0-1000	0	set the smallest torque output capability 1000 = 100%

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X53
Data[4]	return value: data	data
Data[5]	end frame	0XFA

Setting Servo Parameters

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X05
Data[3]	command frame	0X52
Data[4]	serial number of joint	Joint_no
Data[5]	data address	data_id
Data[6]	parameter	view the table below
Data[7]	end frame	0XFA

Example:

Reading ratio parameter of P position of No.1 steering gear

Port transmission: FE FE 04 52 01 15 01 FA

Serial number of joint ranges from 1 to 6

no return value

Data_id: data type byte, referring to the chart below for specific value:

Address	Function	Range	Initial Value	Explanation
20	LED siren	0-254	0	1\0: turn on/off LED siren
21	position loop P	0-254	10	control the ratio coefficient
22	position loop I	0-254	0	control the differential coefficient
23	position loop D	0-254	1	control the integral-action coefficient
24	minimum starting force	0-1000	0	set the smallest torque output capability 1000 = 100%

Setting Zero Position of Steering Gear

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X54
Data[4]	serial number of joint	Joint_no
Data[6]	end frame	0XFA

Example:

Reading zero position of No.1 steering gear

Port transmission: FE FE 03 54 01 FA

Serial number of joint ranges from 1 to 6

no return value

Braking Single Motor

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X55
Data[4]	serial number of joint	Joint_no
Data[5]	end frame	0XFA

Example:

Serial number of joint ranges from 1 to 6

Braking No.1 steering gear,

Port transmission: FE FE 03 55 01 FA

no return value

Power Failure of Single Motor

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X56
Data[4]	serial number of steering gear	Servo_no
Data[5]	end frame	0XFA

Example:

Let No.3 steering gear reducing power

Serial number of servo ranges from 1 to 6

no return value

Powering On a Single Motor

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X57
Data[4]	serial number of steering gear	Servo_no
Data[5]	end frame	0XFA

Example:

power on No.1 steering gear

Port transmission: FE FE 03 57 01 FA

Serial number of servo ranges from 1 to 6

no return value

Setting Mode of Atom

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X60
Data[4]	serial number of pin	pin_no
Data[5]	input/output	00X00/00X01
Data[6]	end frame	0XFA

Example:

setting atom pin22 as input mode

Port transmission: FE FE 04 60 16 00 FA

Pin_no: data type byte

Pin_mode：0/1

no return value

Setting Atom IO (setDigitalOutput)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X61
Data[4]	serial number of pin	pin_no
Data[5]	signal of electrical level	00X00/00X01
Data[6]	end frame	0XFA

Example:

setting atom pin23 as high electrical level

Port transmission: FE FE 04 61 17 01 FA

no return value

Reading Atom IO(getDigitalInput)

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X62
Data[4]	serial number of pin	pin_no
Data[5]	end frame	0XFA

Example:

reading signal of electrical level of pin23

Port transmission: FE FE 03 62 16 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0X62
Data[4]	pin number	pin_no
Data[5]	signal of electrical level	0X00/0X01
Data[6]	end frame	0XFA

Example:

Given that pin 22 is of high electrical level

Port return value: FE FE 04 62 16 01 FA

Reading the Angle of Gripper

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X65
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 65 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X65
Data[4]	span of gripper's opening	value
Data[5]	end frame	0XFA

value：0-100%

Example:

Given that gripper is completely opening

Port return value: FE FE 03 65 64 FA

value: 6 * 16 + 4 = 100

Setting the Mode of Gripper

Domain	Explanation	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 opening/closing	0X00/0X01
Data[5]	speed	Sp
Data[6]	end frame	0XFA

Example:

making gripper open at the speed of 50

Port transmission: FE FE 04 66 00 32 FA

no return value

Setting Angles of Gripper

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0X67
Data[4]	span of gripper's opening/closing	value
Data[5]	speed	Sp
Data[6]	end frame	0XFA

Example:

Given that gripper opens by 50% at the speed of 20

Port transmission: FE FE 04 67 32 14 FA

The value can be converted into hexadecimal

no return value

Setting the Gripper to Zero Position

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X68
Data[4]	end frame	0XFA

Example:

Setting the Gripper to Zero Position

Port transmission: FE FE 02 68 FA

Checking Whether the Gripper is Moving

Domain	Explanation	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

Port transmission: FE FE 02 69 FA

Return value: data structure

Domain	Explanation	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[6]	end frame	0XFA

Given that gripper stops working

Port transmission: FE FE 03 69 00 FA

Setting RGB of Atom

Domain	Explanation	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

Example:

Setting RGB color as blue

Port transmission: FE FE 05 6A 00 00 FF FA

no return value

Setting IO Output

Domain	Explanation	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]	signal of electrical level	0X00/0X01
Data[4]	end frame	0XFA

Example:

Setting high electrical level of pin 2

Port transmission: FE FE 04 A0 02 01 FA

Reading IO Output

Domain	Explanation	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[4]	end frame	0XFA

Port transmission: FE FE 04 a1 02 01 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X04
Data[3]	return value: command frame	0Xa1
Data[4]	pin number	Pin_no
Data[5]	signal of electrical level	0X00/0X01
Data[6]	end frame	0XFA

Example:

Given that pin 2 has high electrical level

Port return value: FE FE 04 a1 02 01 FA

Acquirng WiFi Account & Password

Domain	Explanation	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

Example:

Port transmission: FE FE 02 b1 FA

port return value: ssid: MyCobotWiFi2.4G password: mycobot123

ssid: WiFi account

password: WiFi password

Setting Port Number

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X04
Data[3]	command frame	0Xb2
Data[4]	highbyte of port	port_high
Data[5]	lowbyte of port	port_low
Data[6]	end frame	0XFA

Example:

Given that port number is 7000

Port transmission: FE FE 04 b2 1b 58 FA

port_high: highbyte of hexadecimal port number

port_low: lowbyte of hexadecimal port number

No return value

Setting Tool Coordinate

Domain	Explanation	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 high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	end frame	0XFA

Example:

Given that （0，0，50，0，0，0）is tool coordinate,

Port transmission: FE FE 0E 81 00 00 00 00 13 88 00 00 00 00 00 00 FA

no return value

Acquiring Tool Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X82
Data[4]	end frame	0XFA

Example:

Port transmission: FE FE 02 82 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X0E
Data[3]	command frame	0X82
Data[4]	specify high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	end frame	0XFA

Port return value: FE FE 0E 82 00 00 00 00 13 88 00 00 00 00 00 00 FA

How to get x coordinate:

temp = x_low + x_high*256

x coordinate =（temp\33000 ?(temp – 65536) : temp)/10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

y coordinate is counted in a similar way.

How to get rx coordinate:

temp = rx_low + rx_high*256

x coordinate =（temp \ 33000 ?(temp – 65536) : temp）/100

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

ry coordinate is counted in a similar way.

Setting World Coordinate

Domain	Explanation	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 high status of x coordinate	x_high
Data[5]	specify low status of x coordinate	x_low
Data[6]	specify high status of y coordinate	y_high
Data[7]	specify low status of y coordinate	y_low
Data[8]	specify high status of z coordinate	z_high
Data[9]	specify low status of z coordinate	z_low
Data[10]	specify high status of rx coordinate	rx_high
Data[11]	specify low status of rx coordinate	rx_low
Data[12]	specify high status of ry coordinate	ry_high
Data[13]	specify low status of ry coordinate	ry_low
Data[14]	specify high status of rz coordinate	rz_high
Data[15]	specify low status of rz coordinate	rz_low
Data[16]	end frame	0XFA

Port return value: FE FE 0E 84 00 00 00 00 13 88 00 00 00 00 00 00 FA

How to get x coordinate:

temp = x_low + x_high*256

x coordinate =（temp\33000 ?(temp – 65536) : temp)/10

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 10; if temp is less than 33000, then temp is divided by 10 directly.

y coordinate is counted in a similar way.

How to get rx coordinate:

temp = rx_low + rx_high*256

x coordinate =（temp \ 33000 ?(temp – 65536) : temp）/100

Explanation: if temp is greater than 33000, temp minus 65536, and then is divided by 100; if temp is less than 33000, then temp is divided by 100 directly.

ry coordinate is counted in a similar way.

Setting Base Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X85
Data[4]	base coordinate/world coordiante	00/01
Data[5]	end frame	0XFA

Example:

Given that the coordinate should be set as world coordinate

port transmission: FE FE 03 85 01 FA

no return value

Acquiring Base Coordinate

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X02
Data[3]	command frame	0X86
Data[4]	end frame	0XFA

Example:

port transmission: FE FE 02 86 FA

Return value: data structure

Domain	Explanation	Data
Data[0]	return value: identification frame	0XFE
Data[1]	return value: identification frame	0XFE
Data[2]	return value: data-length frame	0X03
Data[3]	return value: command frame	0X86
Data[4]	base coordinate/world coordiante	00/01
Data[5]	end frame	0XFA

Example:

port transmission: FE FE 03 86 01 FA

Setting End Coordinate

Domain	Explanation	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	00/01
Data[5]	end frame	0XFA

Example:

Given that end coordinate are set as tool

port transmission: FE FE 03 89 01 FA

no return value

Acquiring End Coordinate

Domain	Explanation	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

Example:

port transmission: FE FE 02 8a FA

Return value: data structure

Domain	Explanation	Data
Data[0]	identification frame	0XFE
Data[1]	identification frame	0XFE
Data[2]	data-length frame	0X03
Data[3]	command frame	0X8a
Data[4]	flange	00/01
Data[5]	end frame	0XFA

Example:

port return value: FE FE 03 8A 01 FA

Appendix

The specific method of adding coordinate-exchanging programs from Atom repository and sports repository are listed as following.

1. Change end coordinate.
2. Set end coordinate by setEndType and getEndType.

   FLANGE: Setting EndType as FLANG; TOOL: Setting EndType as TOOL End.
3. Read tool coordinate via setToolReference and getToolReference. (FLANGE serves as relative coordinate and information about tool end is relevant to FLANGE coordinate.)
4. After setting EndType as FLANGE, GetCoords and WriteCoords are counted depending on FLANG's position.

5. After setting EndType as TOOL, GetCoords and WriteCoords are counted depending on end's position.
6. Change base coordinate.
7. Set base coordinate by setReferenceFrame.

   RFType: Base means setting robotic pedestal as base coordinate; RFType:WORLD means make world coordinate as base coordinate; getReferenceFrame serves to read the type of base coordinate.
8. setWorldReference and getWorldReference works to read information about base coordinate. World coordinate serves as relative coordinate to type pedestal information of robots that is relevant to world coordinate.
9. If base coordinate acts as pedestal, GetCoords and WriteCoords take base coordinate as reference coordinate.
10. If world coordinate acts as pedestal, GetCoords and WriteCoords take world coordinate as reference coordinate.

Information Updating About Communication

These are newly-added functions: setting and reading of end coordinate, world coordinate, present coordinate, type of end, as well as moving method, and sending and receiving of information on robotic arms.

The communication is temporarily set as from 0x80 to 0x8A.

The roboticMessages space is specially added in ParameterList.h for more communication information. Now only `No Analytical Solution` signal has been added, while more will be added later.

The outline of MOVEL are listed as follows:

First, count the Euclidean Distance between starting point and target point. Second, interpolation points are inserted every 10mm, based on the Euclidean Distance. If interpolation points do not have analytical solution, then search for plus or minus PI/30 from three unchangeable direction for analytical solution. Avoid singular value and some distinctive position that cannot get analytical solution.

Change the interval between MOVEL and JOG into dynamic time. And then, count the time spend moving through the maximum joint. And then, minus the moving time by specific period to take it as interval.

Mycobot320 and PLC IO interactive control case

1 Functional effect description

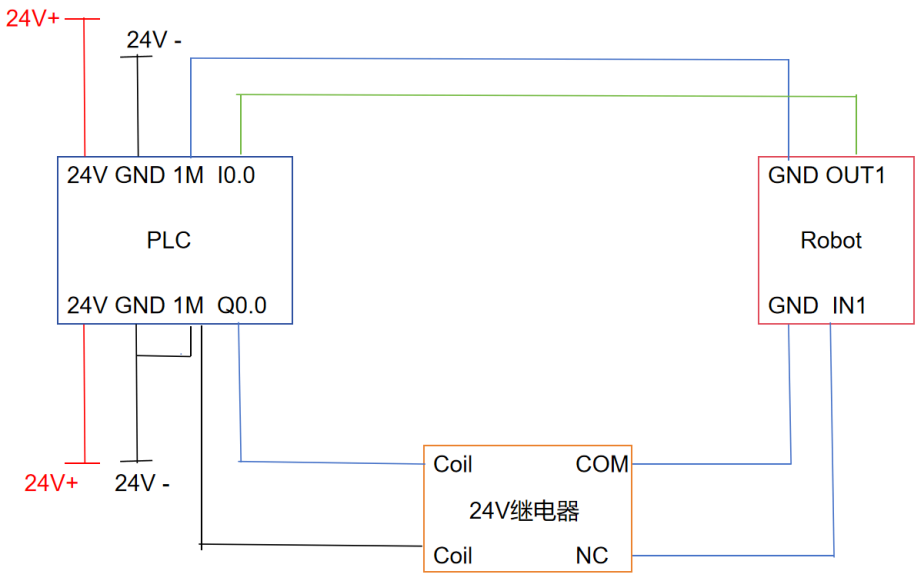
After the robot receives the IO signal from the PLC, it will perform an action of returning each joint to zero position

2 Principle description

The output end of the robot will first output a signal. After the PLC collects the input signal, the PLC output end will output a signal, so that the 24v relay coil is energized, the normally open contact is connected, and the low-level signal is transmitted to the input end of the robot. After the robot 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 and the output of the PLC

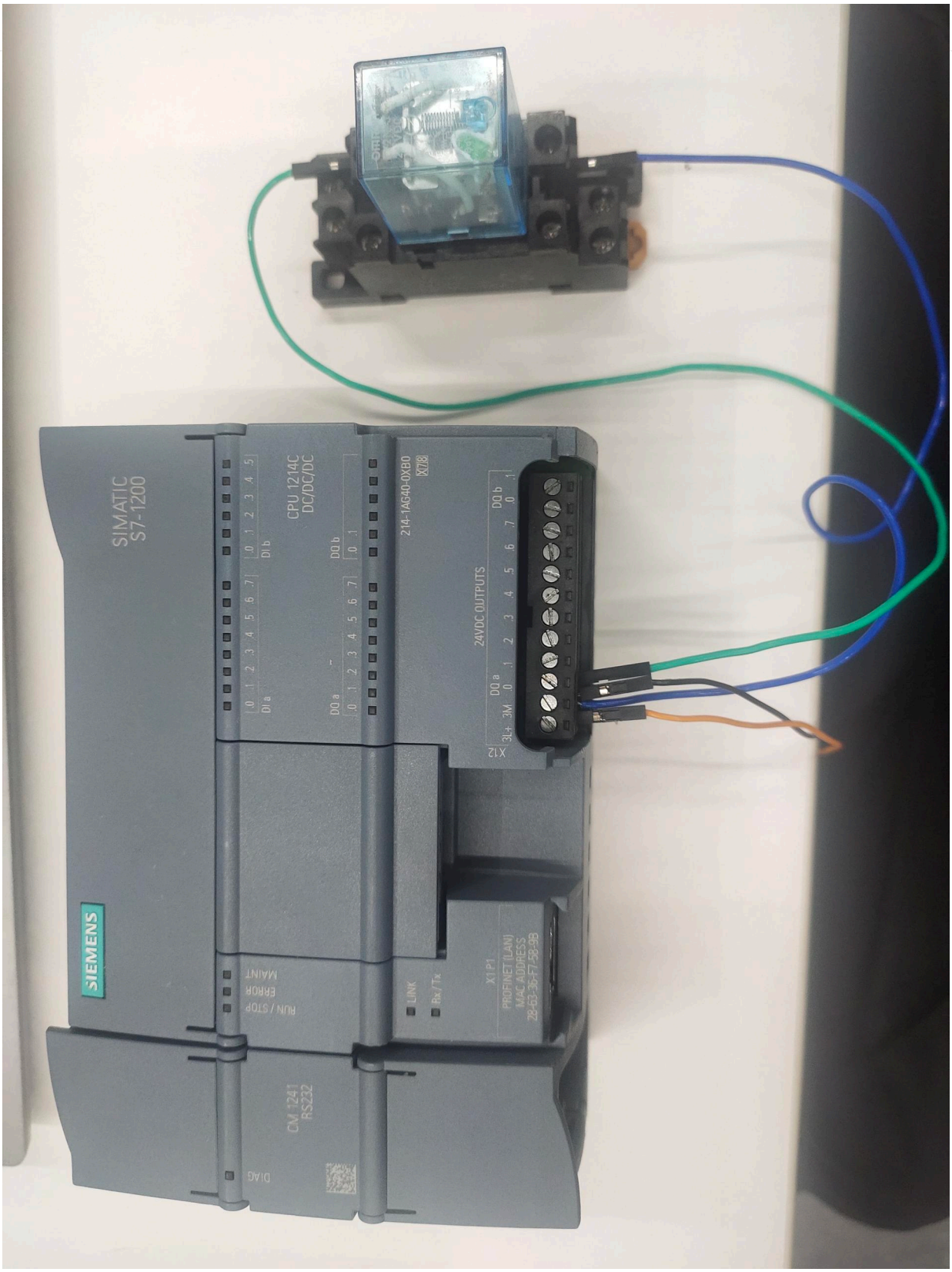
The PLC is Siemens 1200, the output type of the PLC is PNP, and the input type of the robot is NPN, so an external intermediate relay is required to convert the signal.

First connect 24V to the PLC output



Then connect the relay coil to the common terminal and Q0.0 of the PLC




SIMATIC
S7-1200
CPU 1214C
DC/DC/DC
214-1AG40-0XB0
24VDC OUTPUTS
SIEMENS
CM 1241
RS232
X1 P1
PROFINET (LAN)
MAC ADDRESS
28-63-36-F7-58-9B
LINK
Rx / Tx
RUN / STOP
ERROR
MAINT
DIAG

Then connect the normally open contact wire of the relay to the terminal



Then connect the terminal to IN1 of the robot

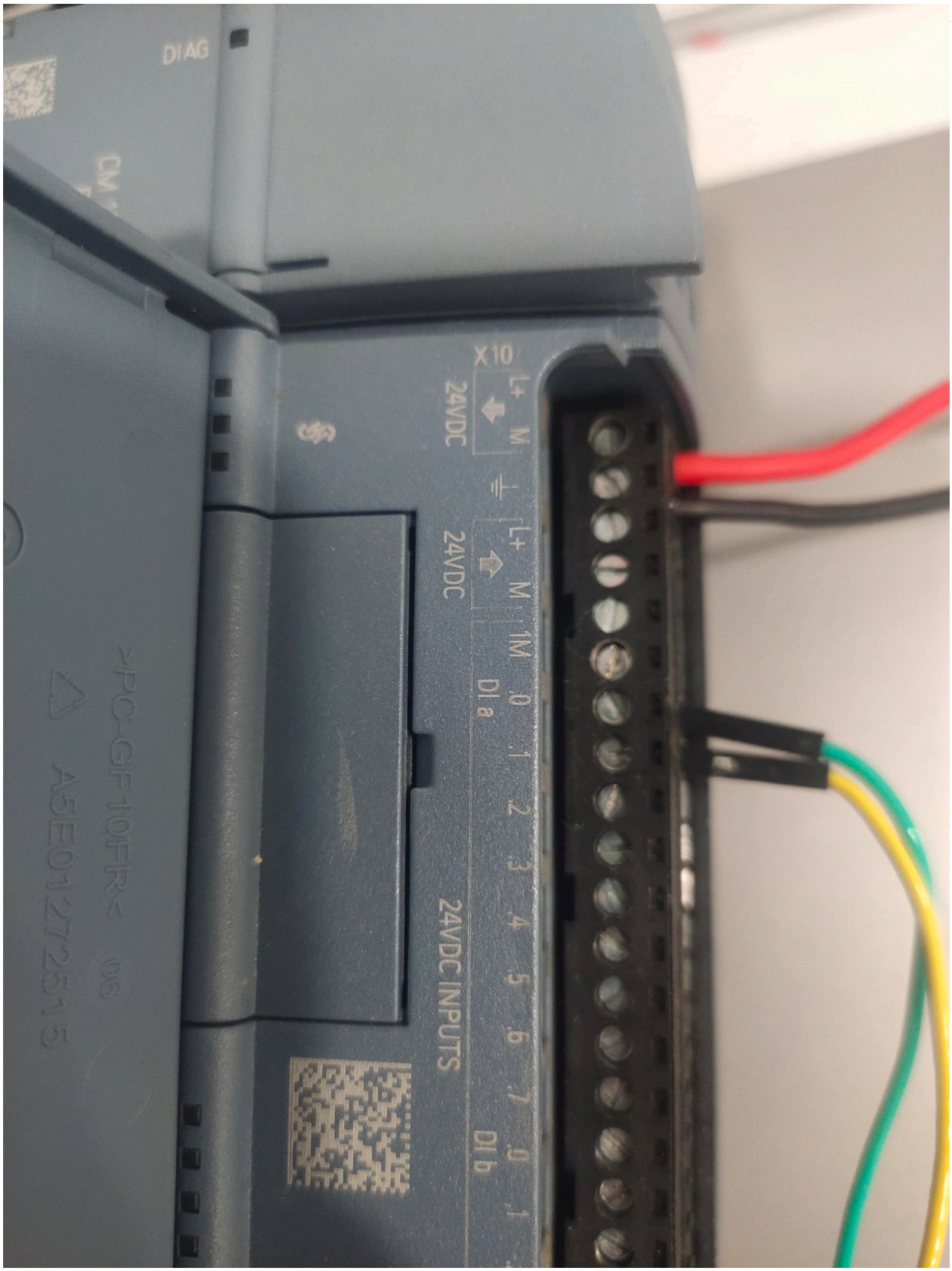
The output of the robot arm is connected to the PLC Input wiring

PLC is Siemens 1200, the input type of PLC supports PNP or NPN, the output type of the robot is PNP, so the input of PLC adopts PNP type connection

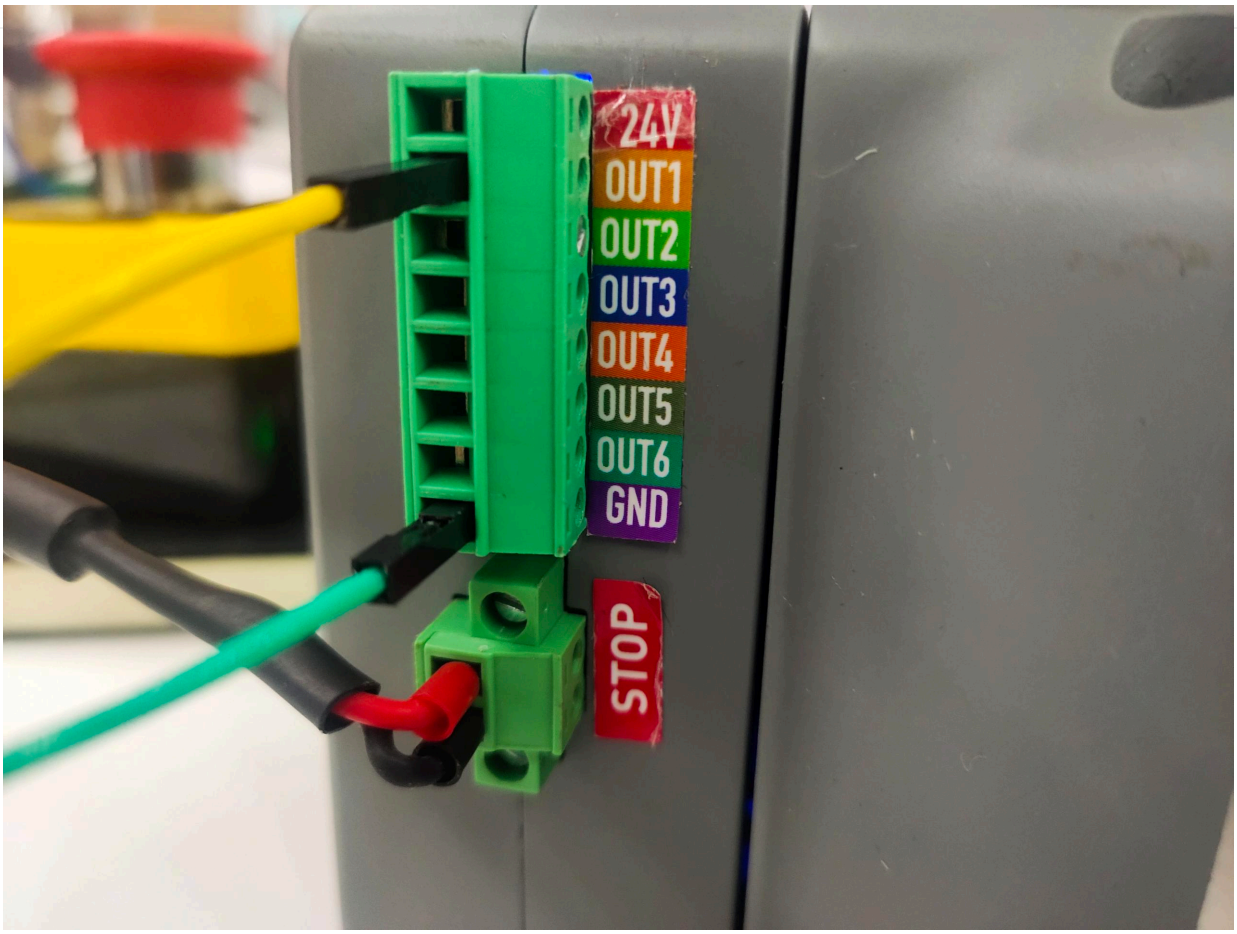
First connect 24V to the input terminal of PLC



Then connect the GND and OUT1 of the robot to the common terminal and I0.0 of PLC



Then insert the terminal into the output of the robot

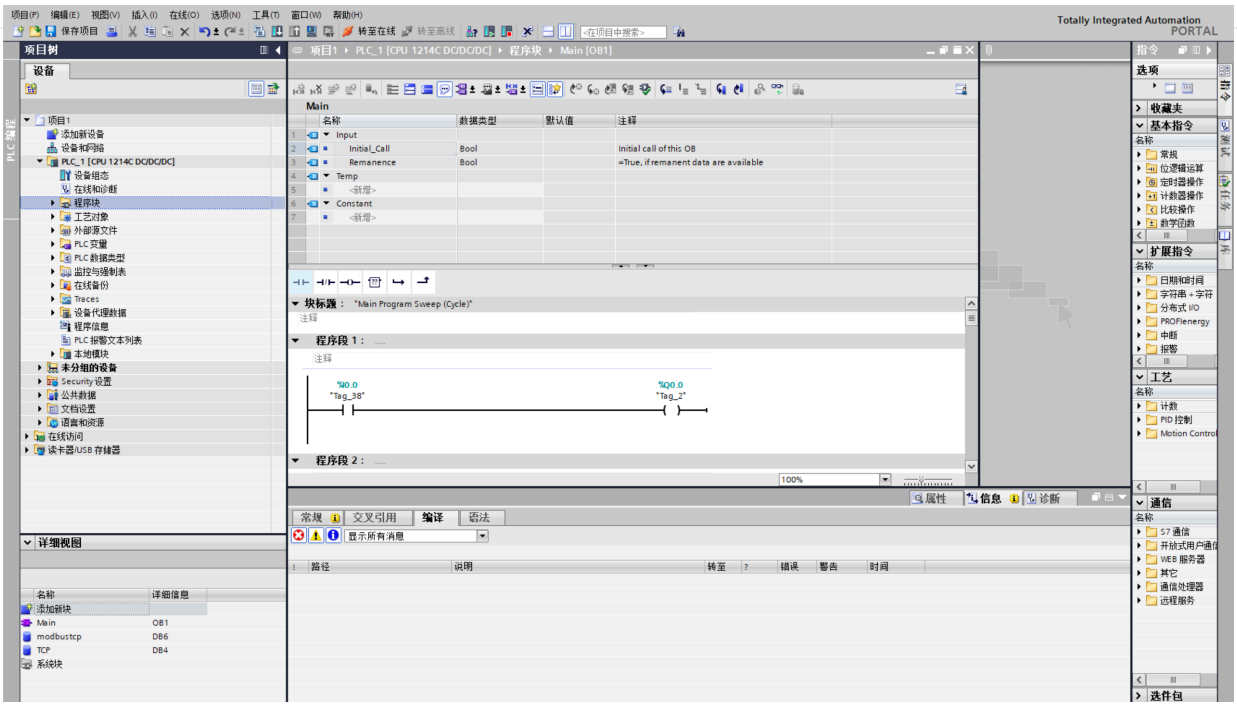


4 Software Programming

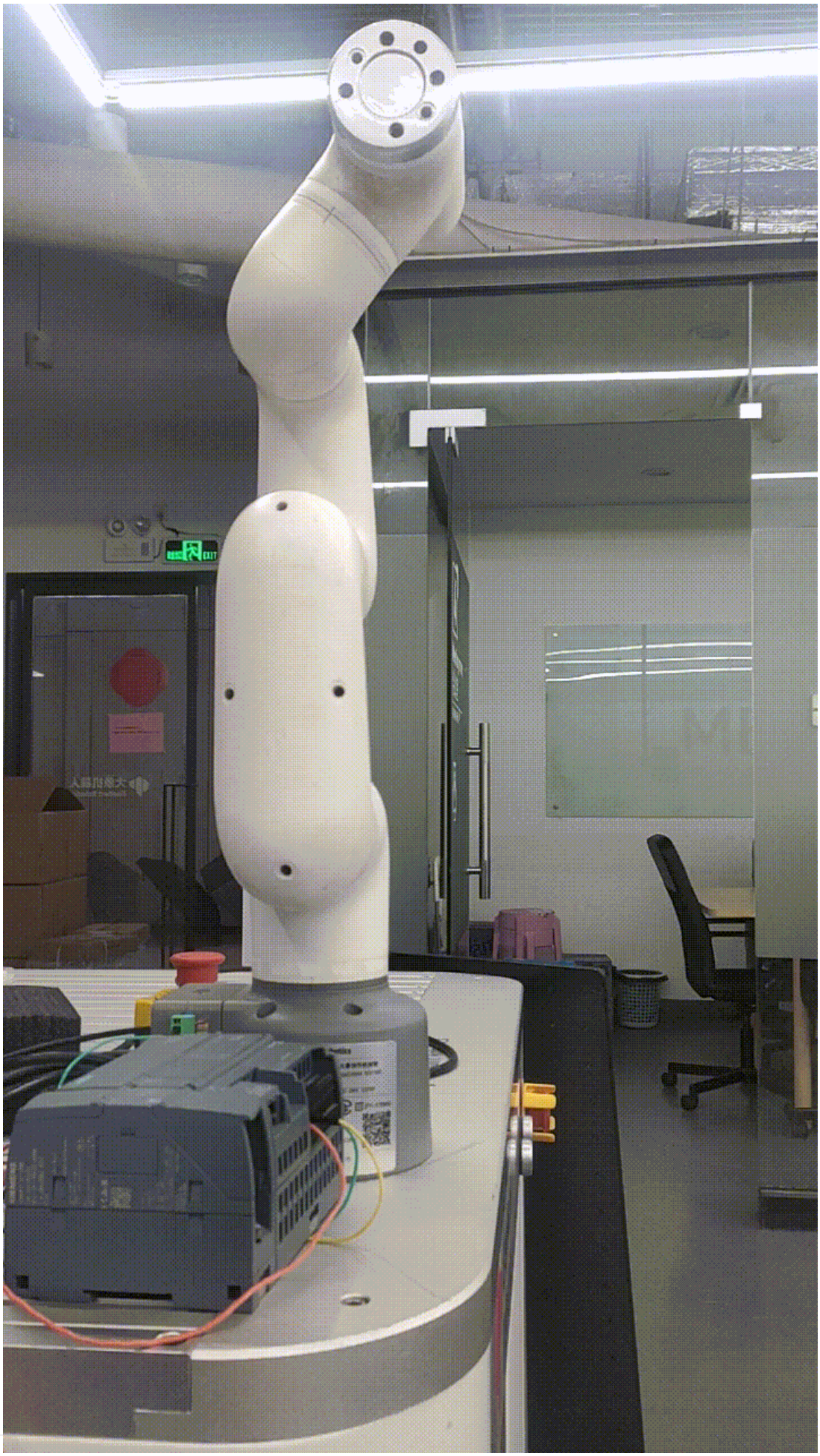
Robotic Arm Program

```
from pymycobot import MyCobot
import time
mc=MyCobot("COM8")
mc.set_basic_output(1,0)
while 1:
if mc.get_basic_input(1)==0:
mc.send_angles([0,0,0,0,0,0],50)
break
else:
pass
mc.set_basic_output(1,1)
```

PLC Program


项目树
设备
项目1
添加新设备
设备和网络
PLC_1 [CPU 1214C DC/DC/DC]
设备组态
在线和诊断
程序块
工艺对象
外部源文件
PLC 变量
PLC 数据类型
监控与强制表
在线备份
Traces
设备代理数据
程序信息
PLC 报警文本列表
本地模块
未分组的设备
Security 设置
公共数据
文档设置
语言和资源
在线访问
读卡器/USB 存储器
详细视图
名称
详细信息
添加新块
Main
OB1
modbustcp
DB6
TCP
DB4
系统块
项目1 ▸ PLC_1 [CPU 1214C DC/DC/DC] ▸ 程序块 ▸ Main [OB1]
Main
名称
数据类型
默认值
注释
Input
Initial_Call
Bool
Initial call of this OB
Remanence
Bool
=True, if remanent data are available
Temp
Constant
<新增>
块标题： "Main Program Sweep (Cycle)"
注释
程序段 1：
注释
%I0.0
"Tag_38"
%Q0.0
"Tag_2"
程序段 2：
100%
属性
信息
诊断
常规
交叉引用
编译
语法
显示所有消息
路径
说明
转至
错误
警告
时间
指令
选项
收藏夹
基本指令
常规
位逻辑运算
定时器操作
计数器操作
比较操作
数学函数
扩展指令
日期和时间
字符串 + 字符
分布式 I/O
PROFIenergy
中断
报警
工艺
计数
PID 控制
Motion Control
通信
S7 通信
WEB 服务器
其它
通信处理器
远程服务
选件包

5 Effect Display


EXIT

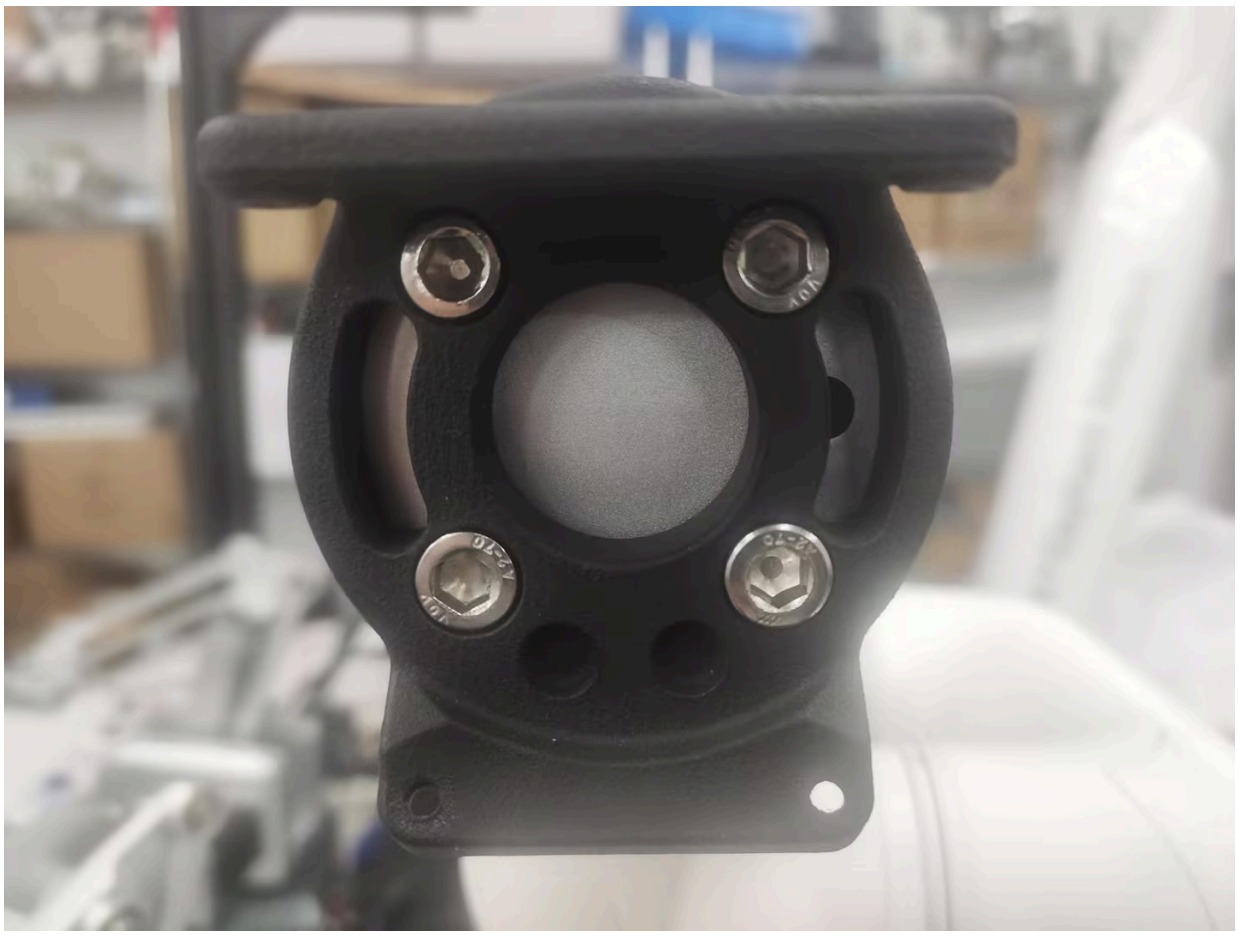
Robot gripper carrying wood block example

1 Functional description

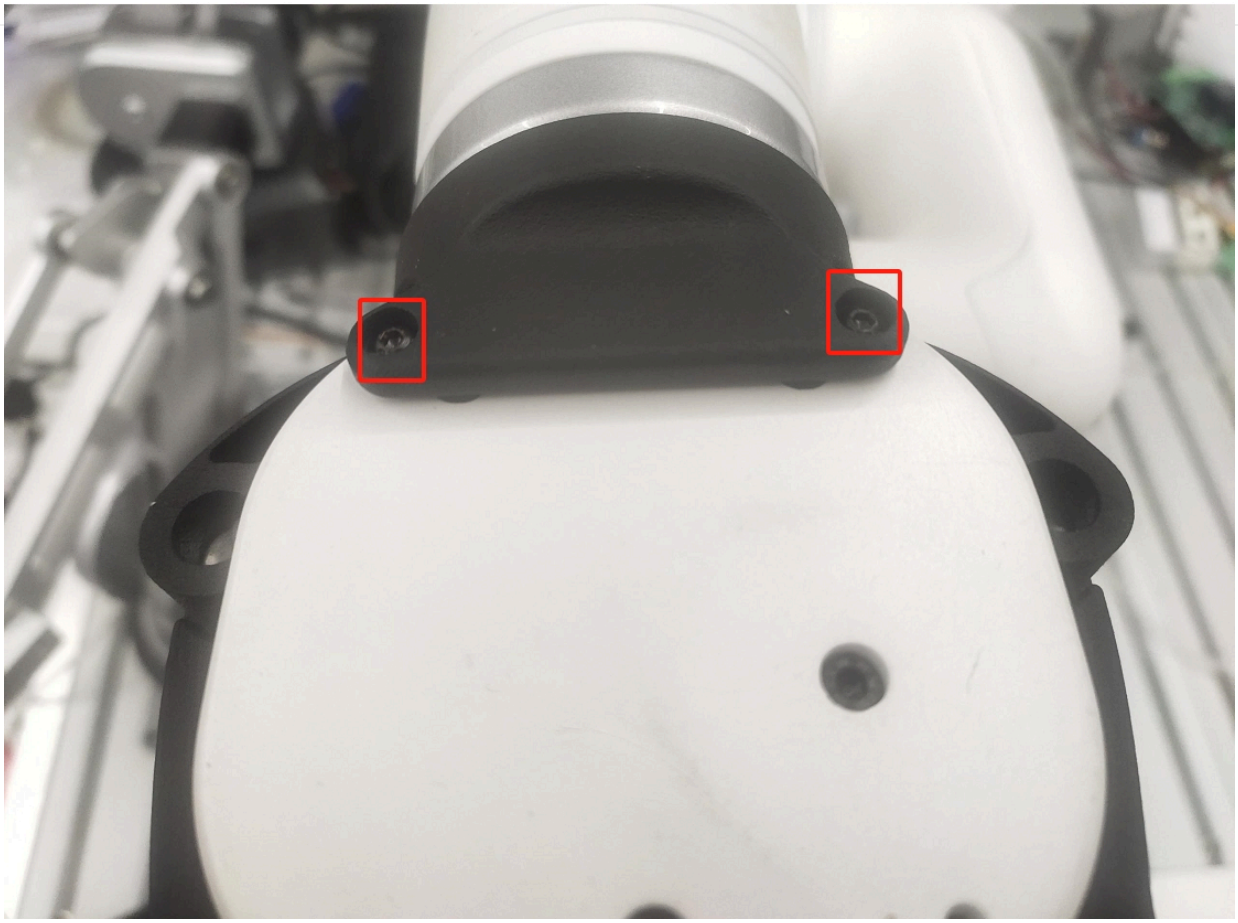
The robot will use the gripper to carry the wood block from point A to point B

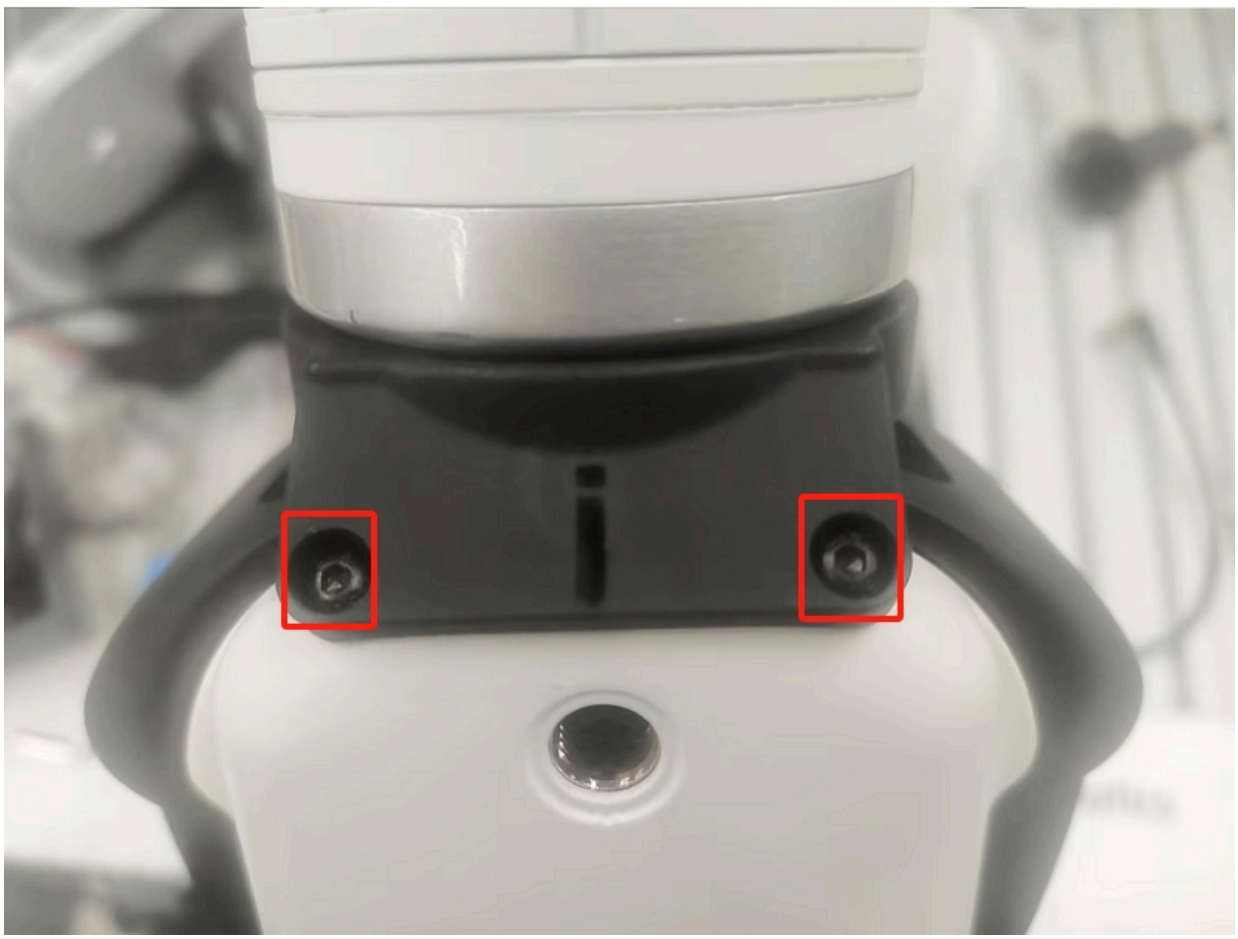
2 Hardware installation

First install the gripper flange to the end of the robot arm



Then install the gripper on the gripper flange





Then use the gripper cable to connect the gripper box to the end IO of the robot arm. When connecting, be sure to turn off the power of the robot arm first to avoid hot plugging and damage to the gripper.



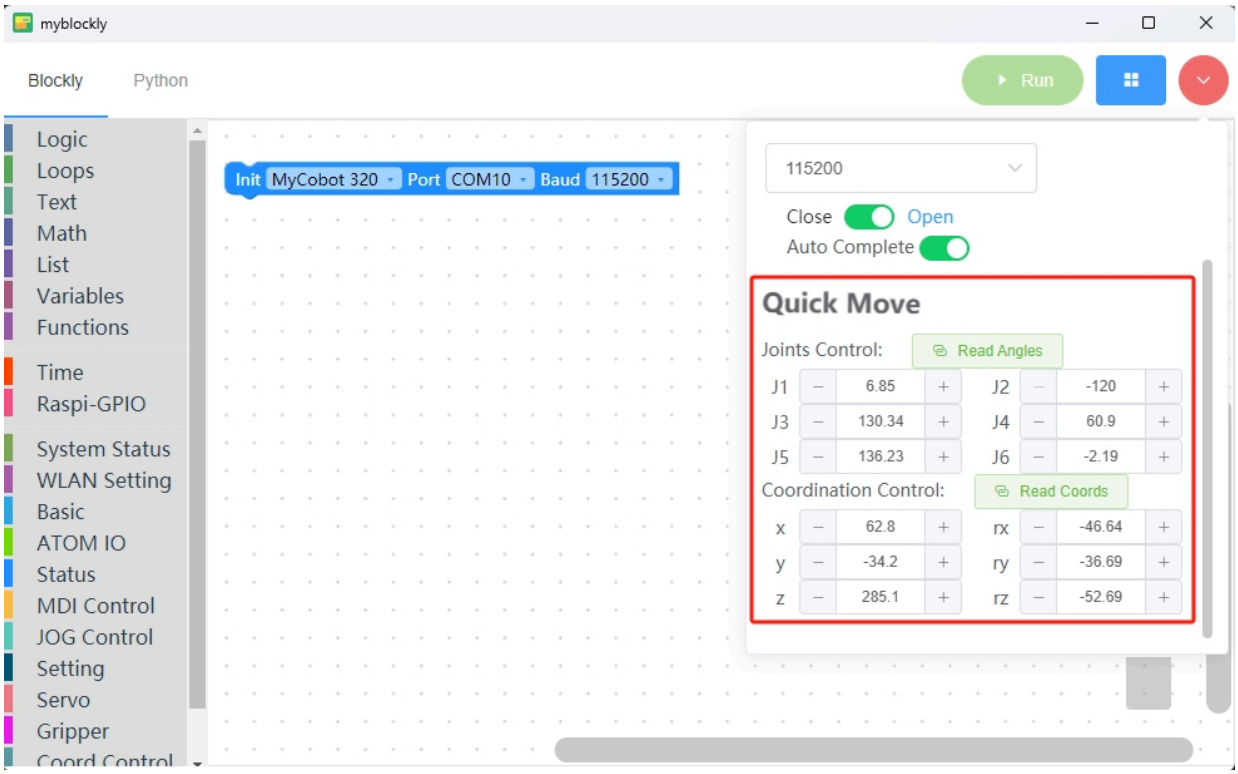
3 Gripper test

Run the following program, the gripper will repeat the action of closing and opening twice

```
from pymycobot import MyCobot320,utils
import time
arm=MyCobot320(utils.get_port_list()[0])

if __name__=="__main__":
    arm.set_gripper_mode(0)# Set the gripper to transparent mode
    for i in range(2):
        arm.set_gripper_state(1,100)# Gripper closed
        time.sleep(1)
        arm.set_gripper_state(0,100)# Gripper open
        time.sleep(1)
```

4 Software Usage

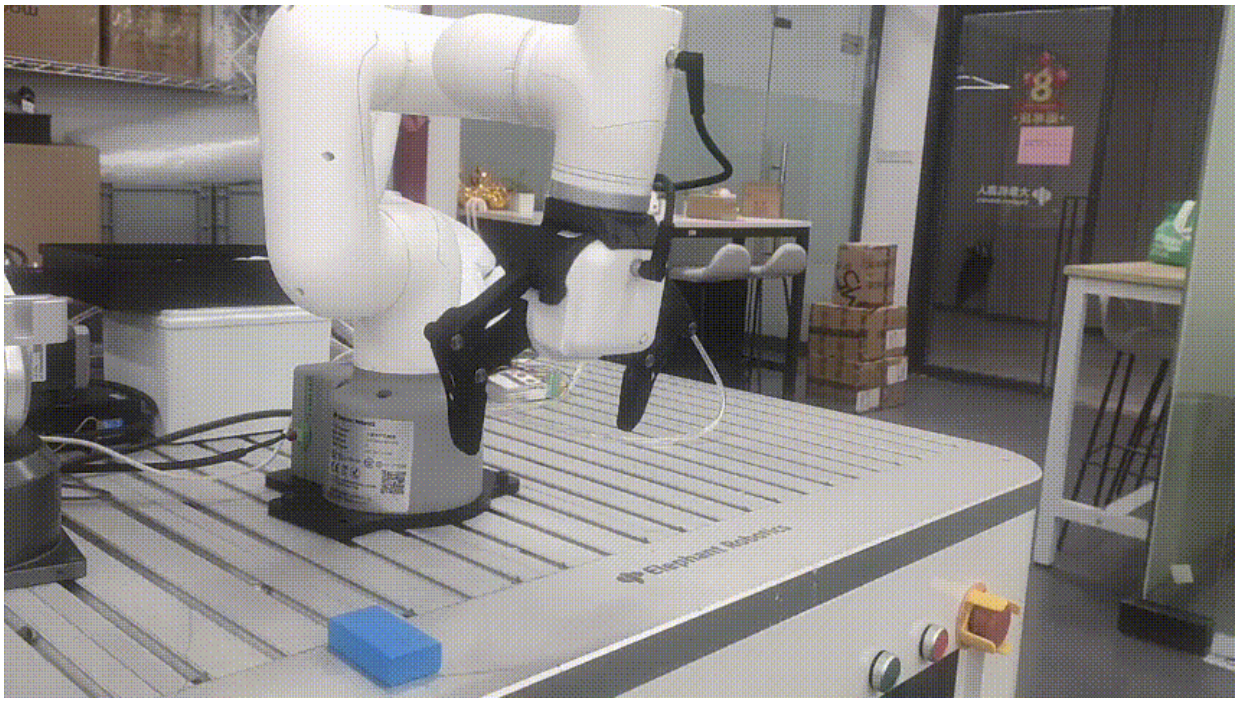
Use the fast movement function of myblockly to teach the grabbing point and placement point of the wooden block, and record the position information. After teaching, you need to disconnect the serial port connection, otherwise the serial port will be reported when running the python script.



5 Composite Applications

```
from pymycobot import MyCobot320,utils
import time

init_angles=[-3.25, -2.46, -95.09, 9.22, 86.39, 93.33]#6 joint angles at the initial position
grab_point=[214.5, -189.9, 185.5, -177.5, 1.91, 173.49]#Coordinates of the grab point
place_point=[214.5, -50.9, 185.5, -177.5, 1.91, 173.49]#Coordinates of the placement point

arm=MyCobot320(utils.get_port_list()[0])

if __name__=="__main__":
    arm.set_gripper_state(0,100)#Open the gripper 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 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)#Gripper closed
    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)#Gripper open
    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



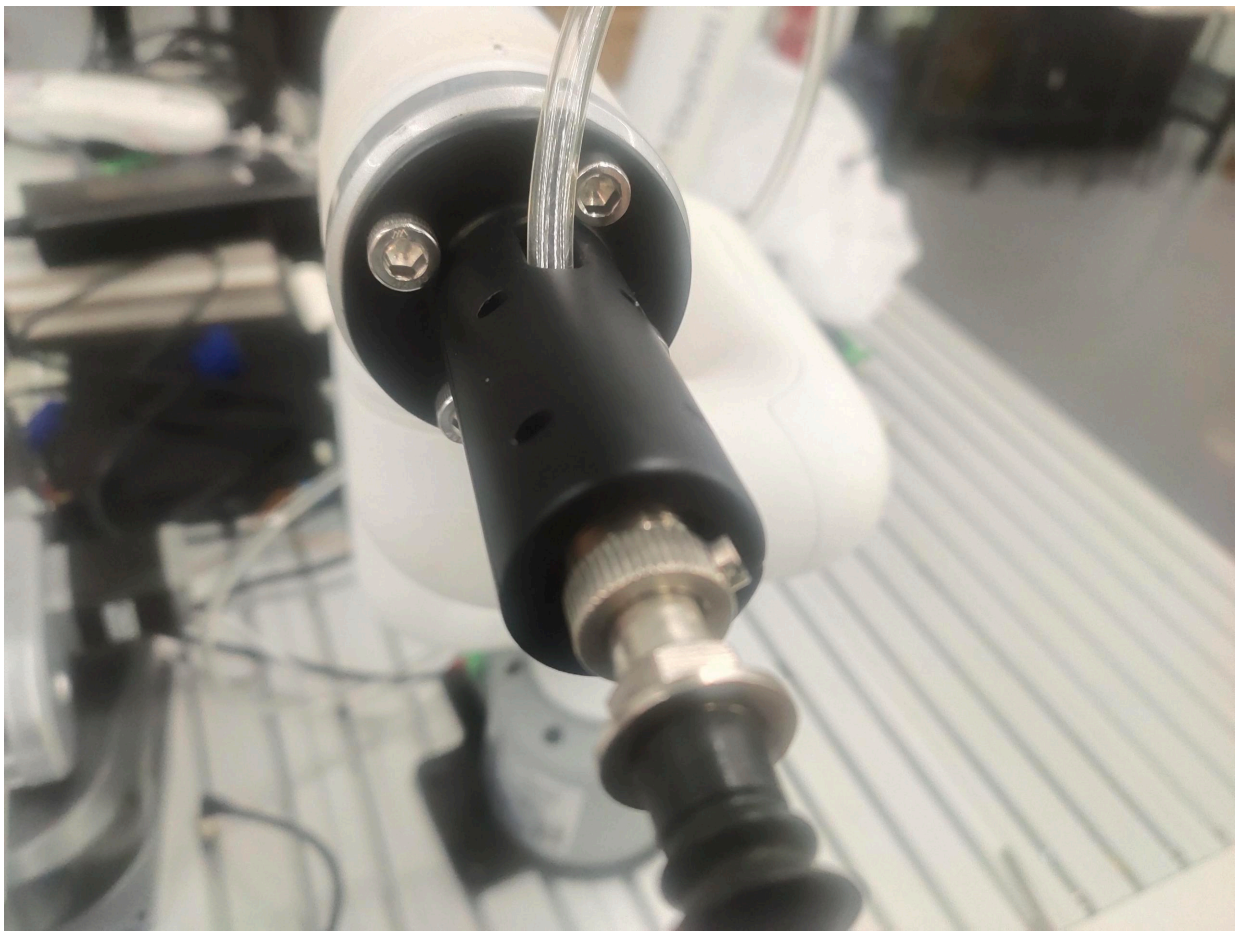
Robot suction pump to move wood blocks

1 Functional description

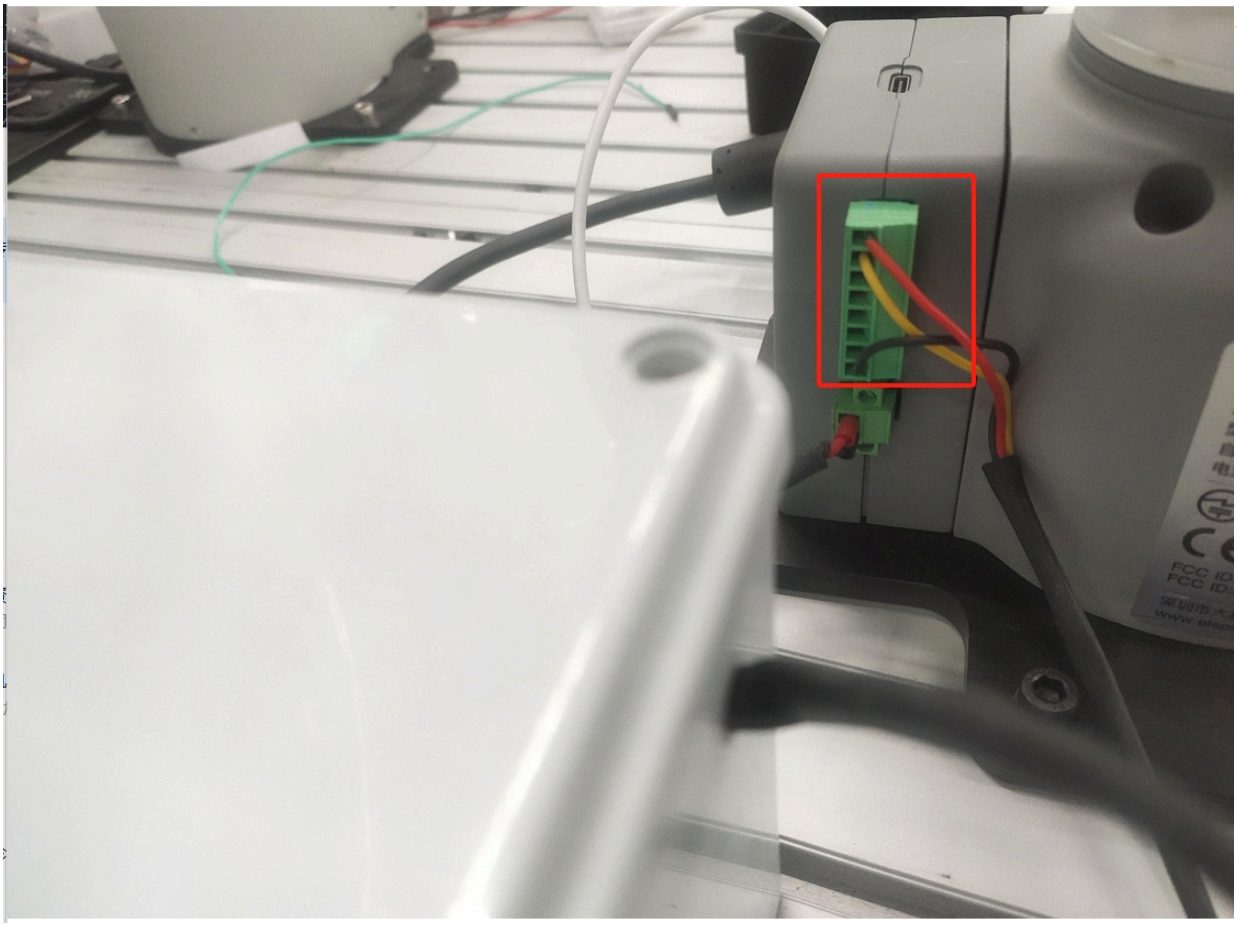
The robot will use the suction pump to move the wood blocks from point A to point B

2 Hardware installation

First install the suction pump on the end of the robot arm



Then connect the wire of the suction pump control box to the base IO of the robot arm



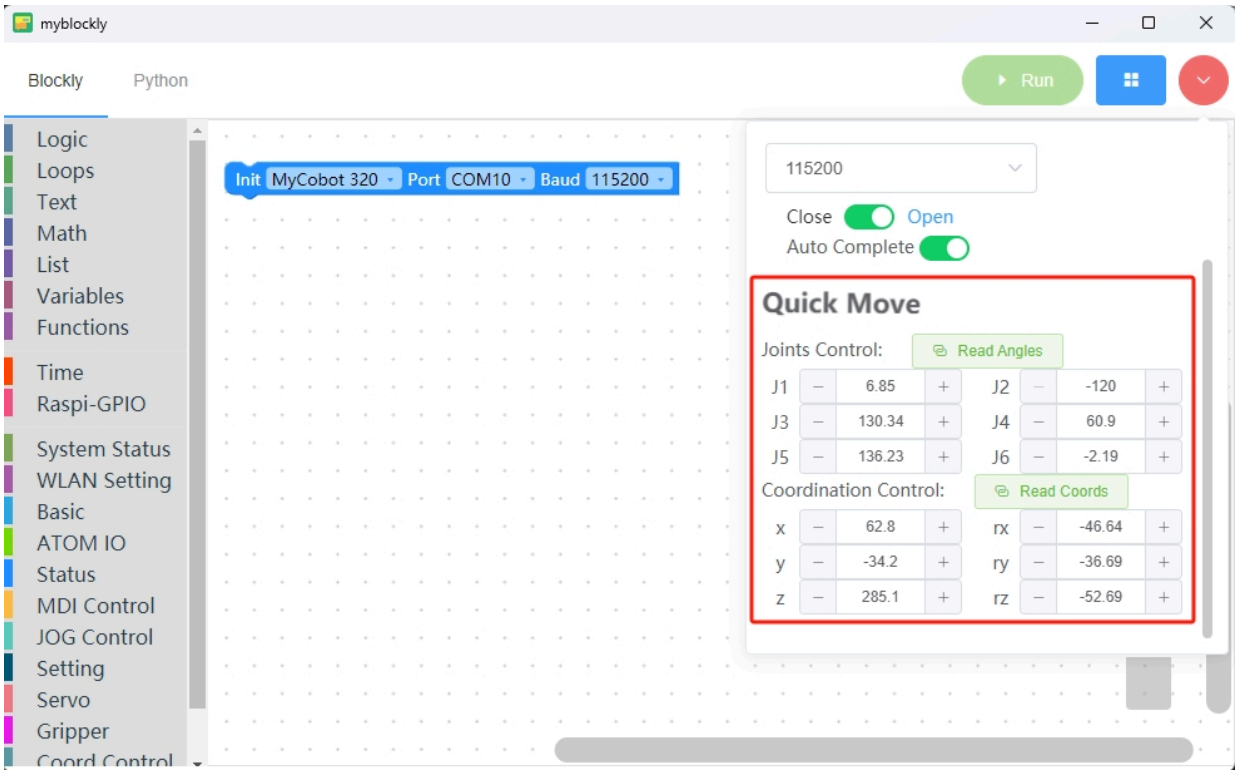
3 Suction pump test

Run the following program, the suction pump will repeat the opening and closing action twice

```
from pymycobot import MyCobot320,utils
import time
arm=MyCobot320(utils.get_port_list()[0])
for i in range(2):
    arm.set_basic_output(1,0)#OUT1 output open
    time.sleep(2)
    arm.set_basic_output(1,1)#OUT1 output closed
    time.sleep(2)
```

4 Software Usage

Use the fast movement function of myblockly to teach the grabbing point and placement point 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.



5 Composite Applications

```
from pymycobot import MyCobot320,utils
import time

init_angles=[-3.25, -2.46, -95.09, 9.22, 86.39, 93.33]#6 joint angles at the initial position
grab_point=[196.9, -197.1, 124.5, -178.8, 1.25, 173.32]#Coordinates of the grab point
place_point=[196.9, -97.1, 124.5, -178.8, 1.25, 173.32]#Coordinates of the placement point

arm=MyCobot320(utils.get_port_list()[0])

if __name__=="__main__":
    arm.set_basic_output(1,1)#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 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_basic_output(1,0) #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)
    arm.set_basic_output(1,1) #Turn off the suction 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

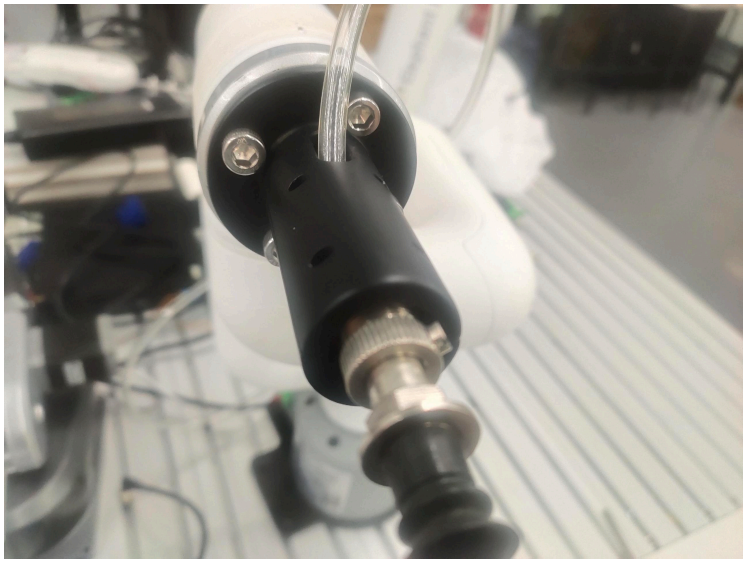


320M5 handle remote control case

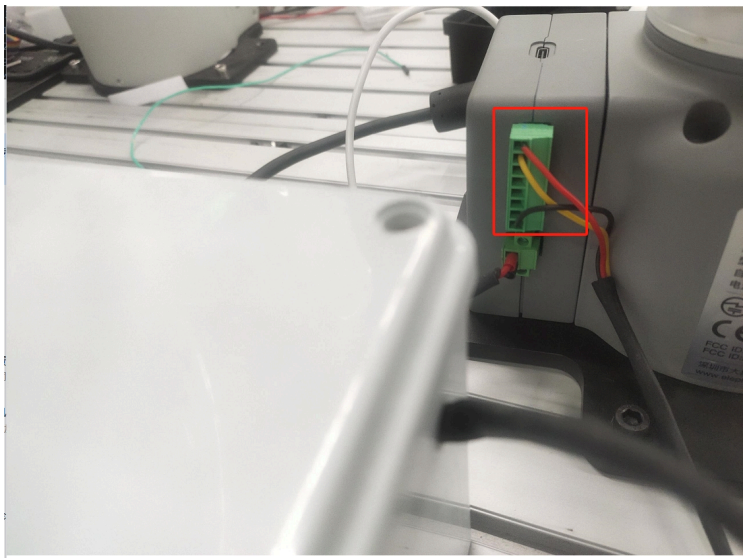
Function description: Use the game controller to control the robot to perform coordinate movement and suction pump switch

1 Suction pump installation

First install the suction pump to the end of the robot arm



Then connect the suction pump control box line to the base IO of the robot arm



2 Suction pump test

```
from pymycobot import MyCobot320,utils
import time
arm=MyCobot320(utils.get_port_list()[0])
for i in range(1):
    arm.set_basic_output(1,0)#Turn on the suction pump
    time.sleep(2)
    arm.set_basic_output(1,1)#Turn off the suction pump
    time.sleep(2)
```

3 Controller function description

Note: The controller 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 the Y key	Turn on the suction pump
Press the A key	Turn off the suction pump

Note: Some handle buttons are not used, so they will not have any effect on the robot arm

4 Installation of handle dependency library

Open the terminal and enter the following command to install the handle driver library

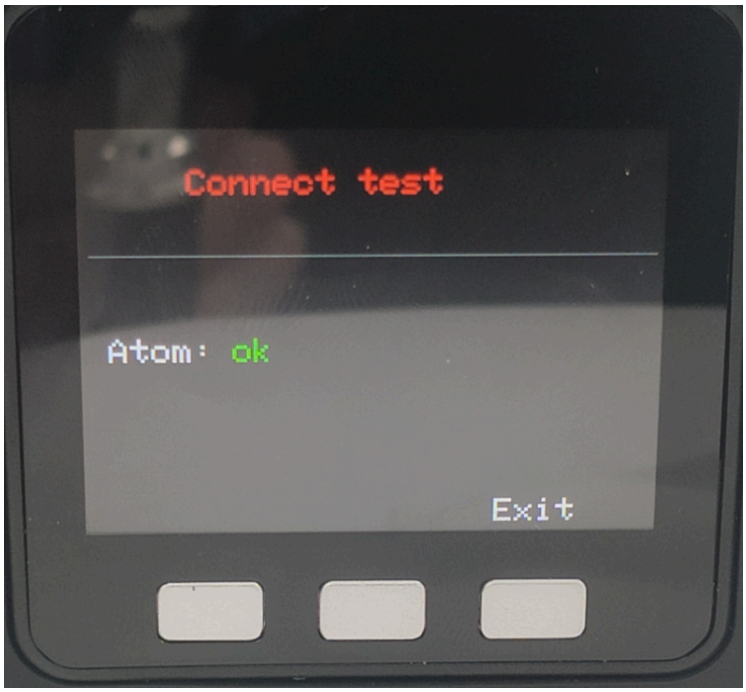
```
pip install pygame
```

5 Preparation

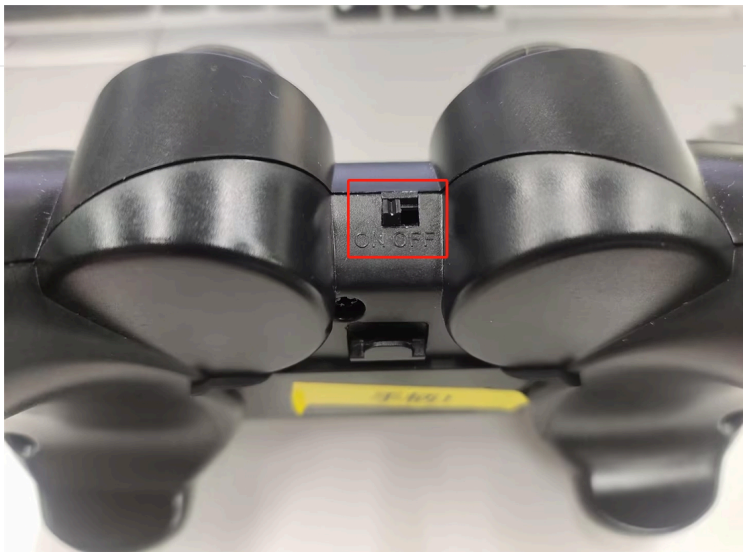
Adjust the robot arm to the posture shown in the figure below. There should be no debris around the robot arm to avoid collision



Note that the screen of the robot arm seat should display ok



Turn on the switch of the handle



Note whether the MODE LED of the handle is on



Note: Only MODE The LED lights up before you can control the robot arm. If the handle is not used for a long time, it will enter the standby state. You can press the START button of the handle to activate it.

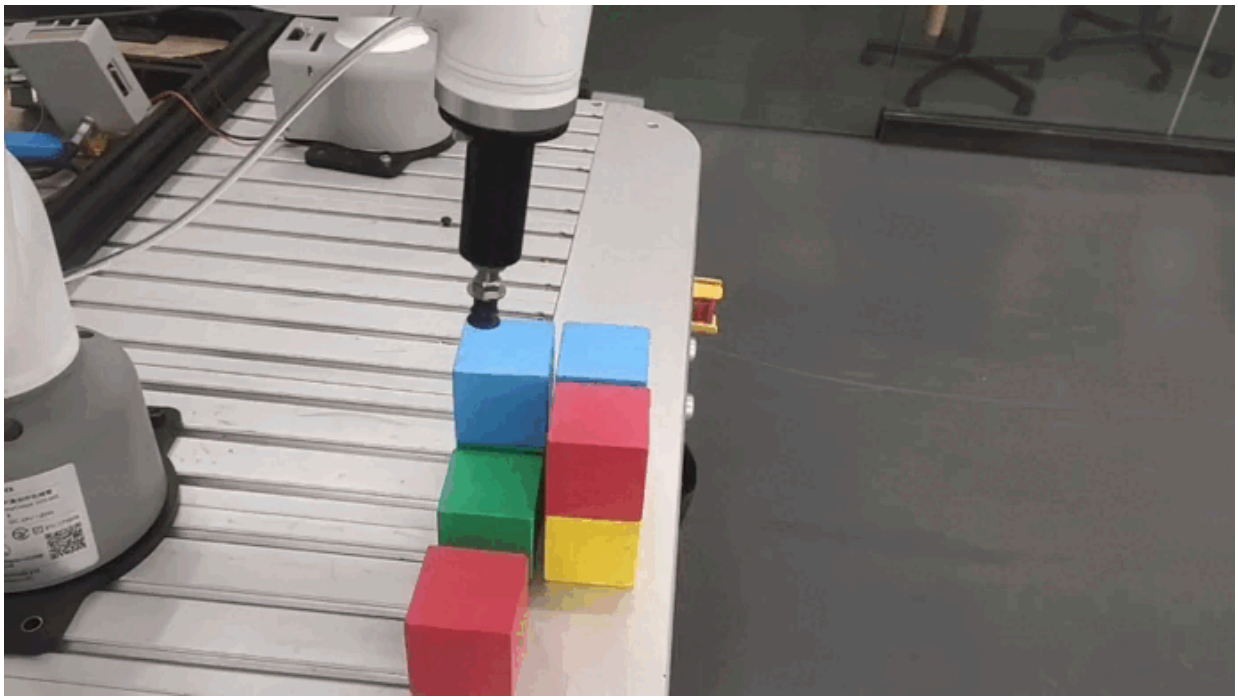


6 Case program

```
import pygame
import sys
from pymycobot import MyCobot320,utils
mc=MyCobot320(utils.get_port_list()[0])
init_angles=[0, 0, -90, 0, 90, 0]
mc.sync_send_angles(init_angles,50)
pygame.init()
pygame.joystick.init()
button_pressed = False
hat_pressed=False
previous_state = [0,0,0,0,0,0]

def joy_handler():
        global button_pressed
        global hat_pressed
        global previous_state
        if event.type == pygame.JOYAXISMOTION:
            axis = event.axis
            value = round(event.value, 2)
            if abs(value) > 0.1:
                flag = True
                previous_state[axis] = value
                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==2 and value==1.00:
                    mc.jog_coord(6,1,50)
                elif axis==2 and value==-1.00:
                    mc.jog_coord(6,0,50)
                if axis==3 and value==1.00:
                    mc.jog_coord(3,0,50)
                elif axis==3 and value==-1.00:
                    mc.jog_coord(3,1,50)
            else:
                if previous_state[axis] != 0:
                    mc.stop()
                    previous_state[axis] = 0

        if event.type == pygame.JOYBUTTONDOWN:
            if joystick.get_button(3)==1:
                mc.set_basic_output(1,0)
            if joystick.get_button(0)==1:
```

```
                mc.set_basic_output(1,1)


        if event.type == pygame.JOYHATMOTION:
            hat_value = joystick.get_hat(0)
            if hat_value ==(0,-1):
                mc.jog_coord(5,1,50)
            elif hat_value ==(0,1):
                mc.jog_coord(5,0,50)
            elif hat_value ==(-1,0):
                mc.jog_coord(4,0,50)
            elif hat_value ==(1,0):
                mc.jog_coord(4,1,50)
            if hat_value != (0, 0):
                hat_pressed = True
            else:
                if hat_pressed:
                    mc.stop()
                    hat_pressed = False

if pygame.joystick.get_count() > 0:
    joystick = pygame.joystick.Joystick(0)
    joystick.init()
else:
    print("No handle detected")
    pygame.quit()
    sys.exit()
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        joy_handler()
pygame.quit()
```

7 Effect display



320M5 Deepseek case

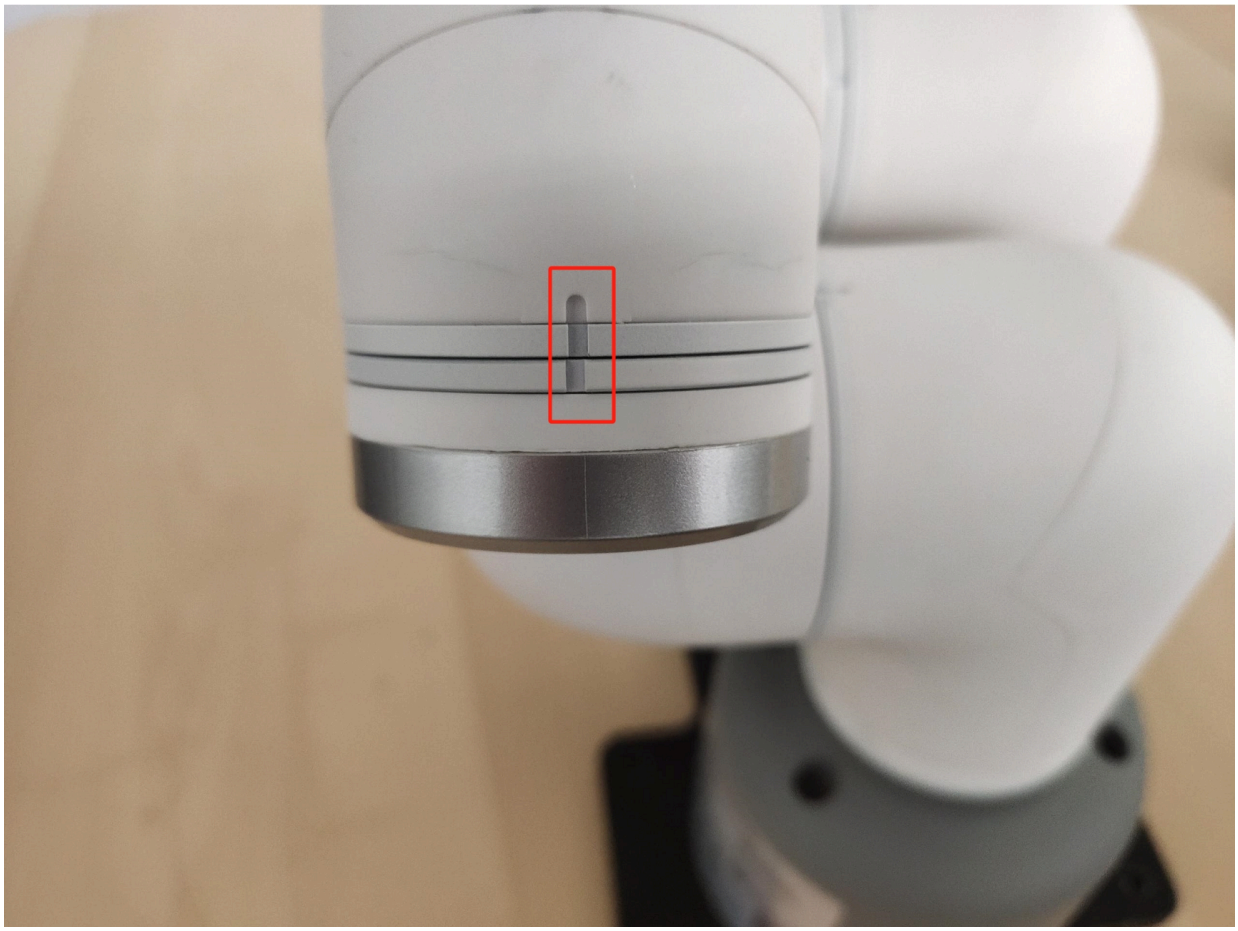
Function introduction: This case uses Baidu online speech recognition API to convert user speech into text, and passes the text command to the locally deployed DeepSeek-R1 large language model. DeepSeek-R1 arranges predefined function functions according to the command content and returns a structured JSON dictionary. The subsequent script parses the JSON data, combines visual technologies such as OpenCV, YOLO target detection and OCR text recognition, and guides the robotic arm to complete the grasping operation

1 Case operation configuration requirements

Requirements	Configuration
Computer type	Laptop
CPU	12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
GPU	NVIDIA GeForce RTX 3060 or above
Memory	16.0 GB
Hard disk	128G available space
System	Windows 10 and above
Network requirements	Able to access the Internet normally
Robotic arm	320M5
Python	Version 3.9 or above

2 Hardware installation

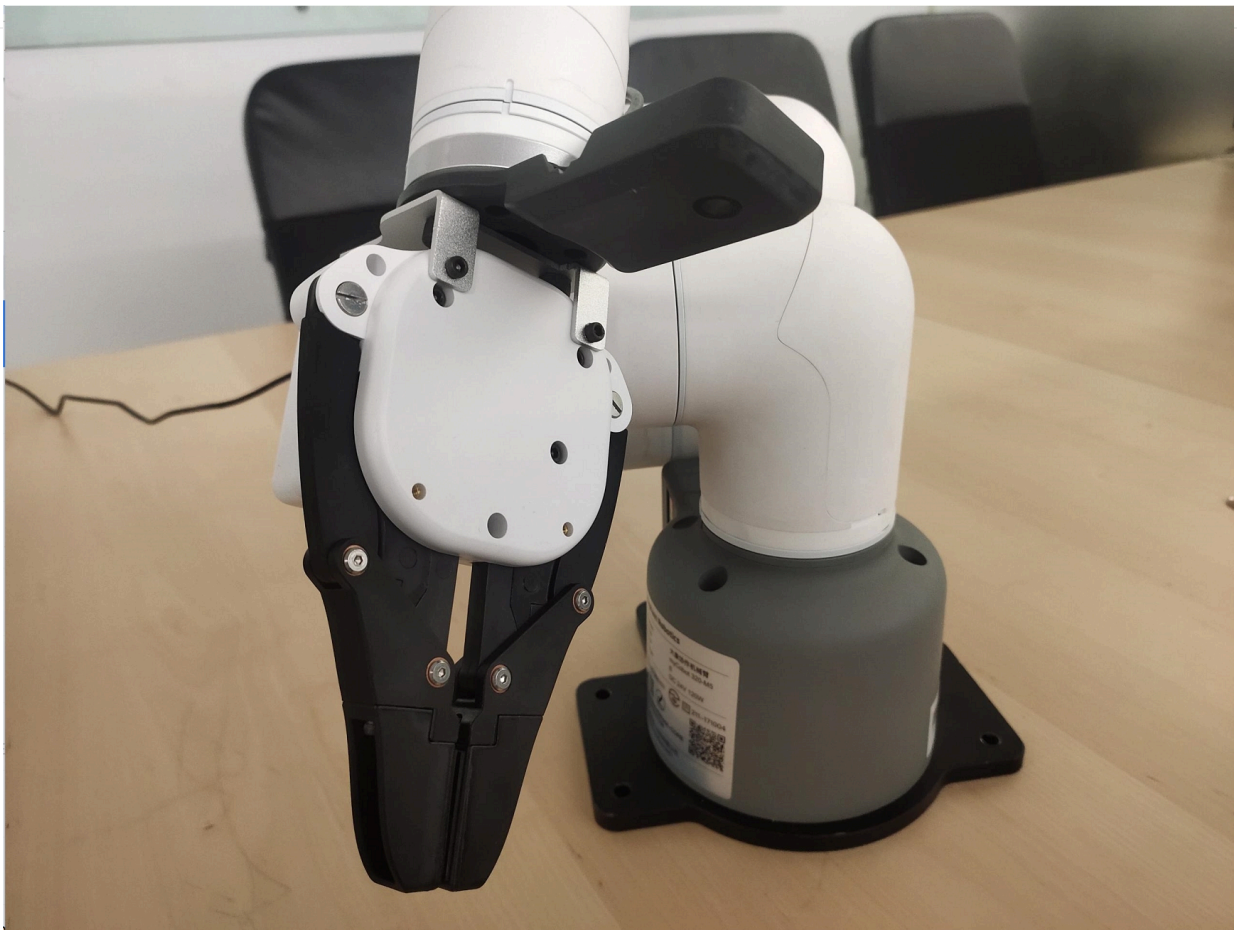
Manually turn the J6 joint to the zero scale line



Be sure to install the camera and force control gripper flange to the end of the robot arm according to the figure below. During installation, be careful not to move the J6 zero scale line, otherwise the robot arm will not be able to grasp normally



Install the force control gripper on the flange



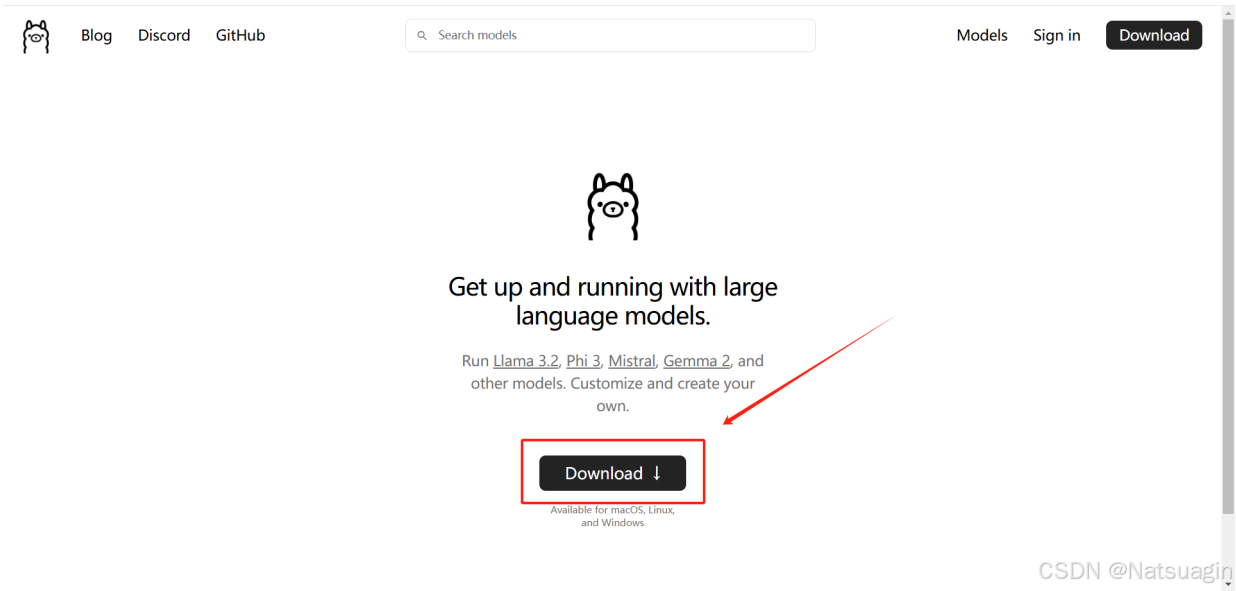
Connect the M8 aviation plug cable to the end interface of the robot arm and the force control gripper interface respectively

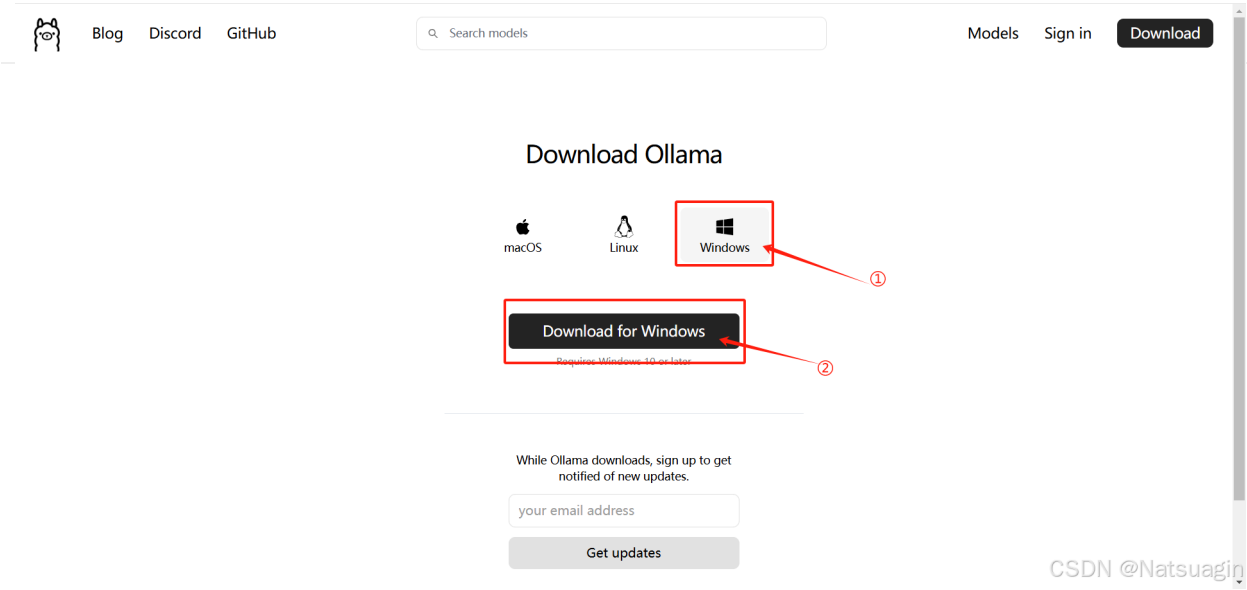


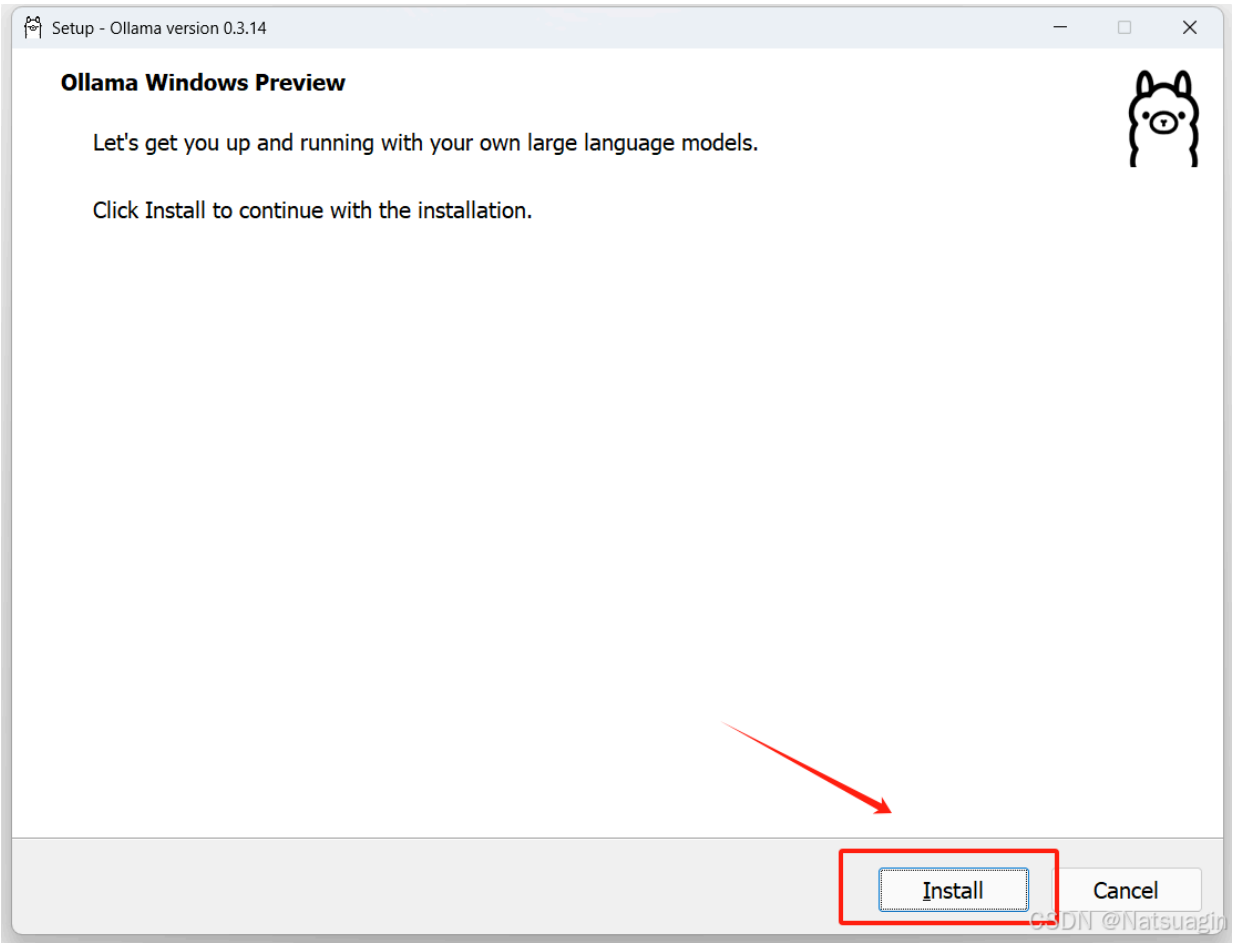
Finally, connect the camera cable to the computer USB port

3 ollama installation

Download official website: https://ollama.com/







4 deepseek Model installation

Open the terminal, enter the following command and press Enter to install

```
ollama run deepseek-r1:7b
```

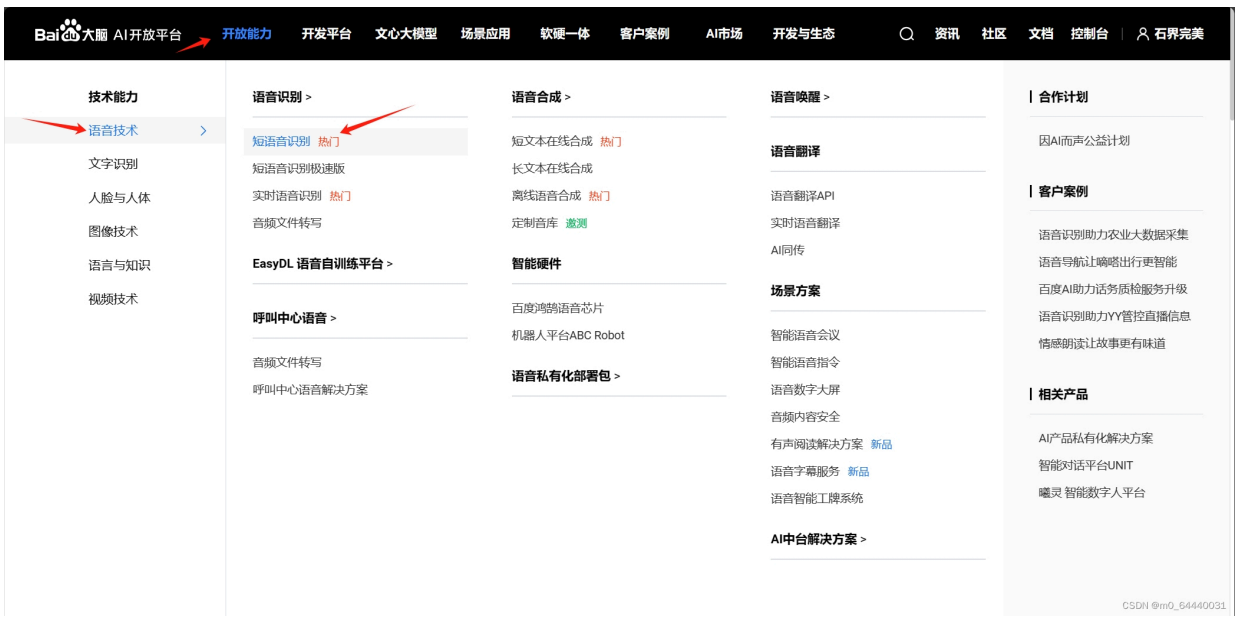
After installation, restart the computer

5 Baidu API creation

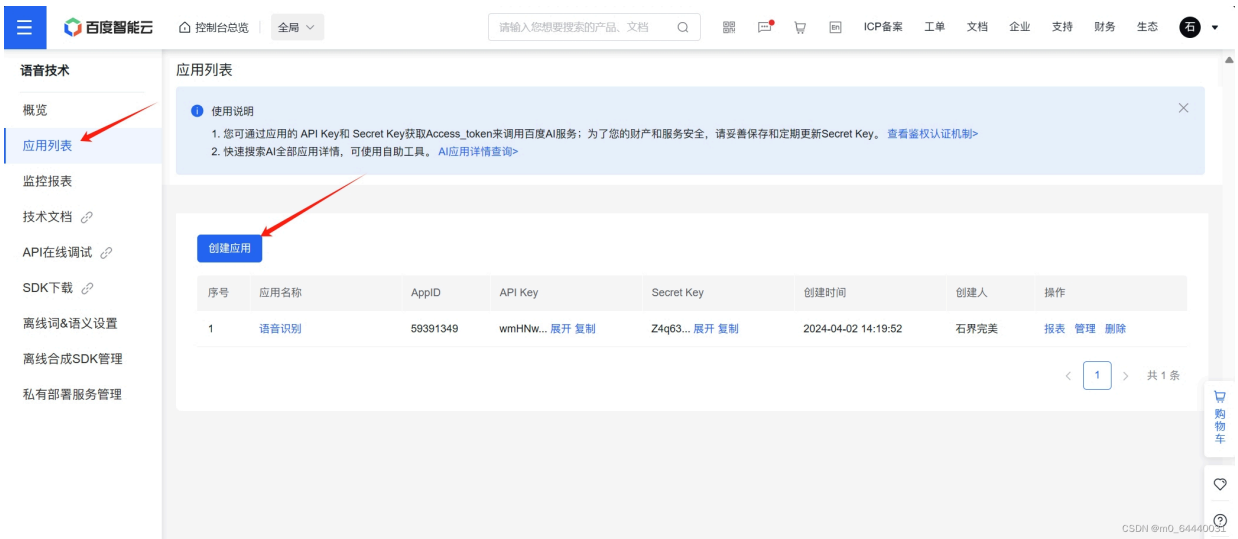
Register Baidu AI Open Platform account First, visit Baidu AI Open Platform, register and log in to your account.

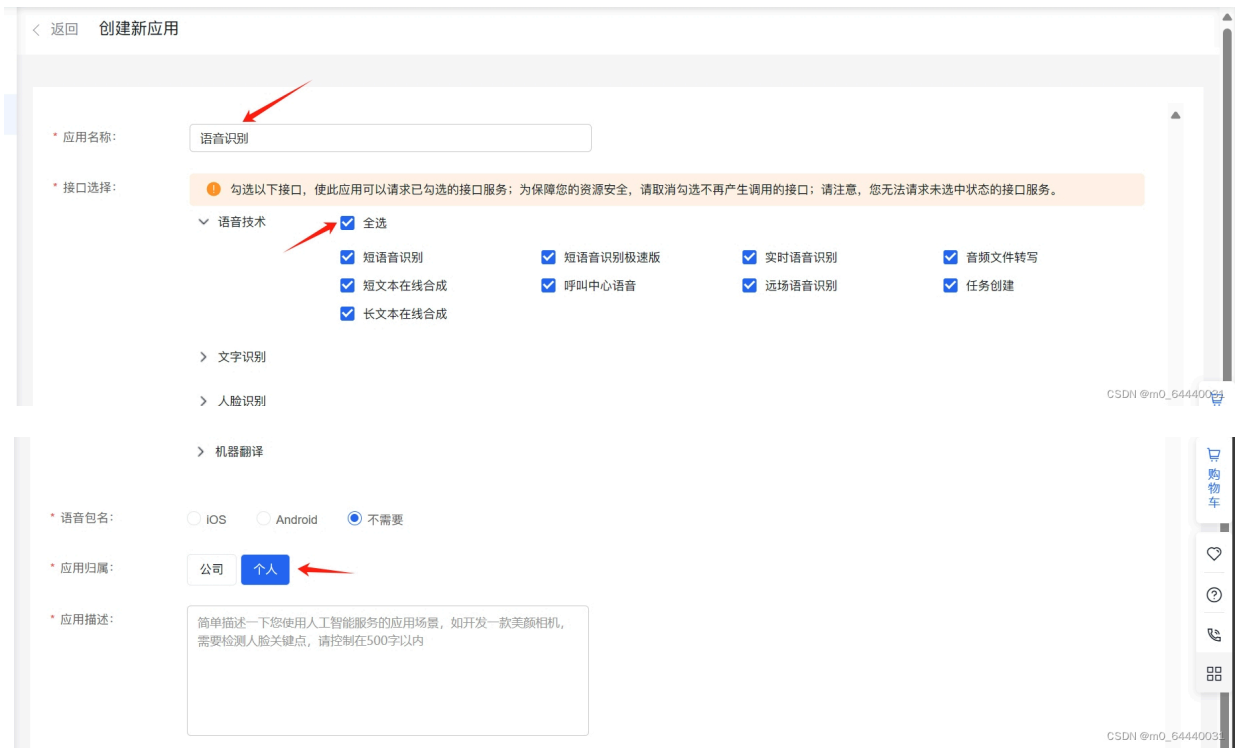


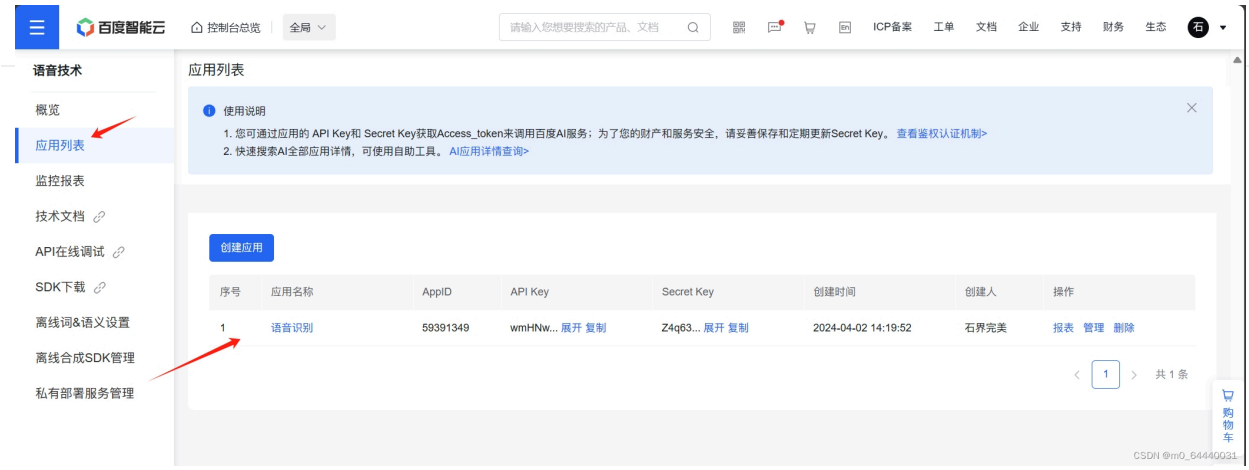
Create an application and obtain APP_ID, API Key and Secret Key In the Baidu AI Open Platform console, create a new application and select the "Speech Recognition" service. After completing the application creation, you will get a pair of API Key and Secret Key as well as APP_ID for subsequent API call authentication.




Bai大脑 AI开放平台
开放能力
开发平台
文心大模型
场景应用
软硬一体
客户案例
AI市场
开发与生态
资讯
社区
文档
控制台
石界完美
短语音识别标准版
将60秒以内的语音精准识别为文字，可适用于手机语音输入、智能语音交互、语音指令、语音搜索等短语音交互场景
新客
短语音识别限时低至1折，售完即止！>
立即使用
技术文档
产品价格
功能介绍
应用场景
特色优势
产品定价
支持交流
相关推荐
合作咨询
CSDN @m0_64440031


百度智能云
控制台总览
全局
请输入您想要搜索的产品、文档
ICP备案
工单
文档
企业
支持
财务
生态
语音技术
概览
应用列表
监控报表
技术文档
API在线调试
SDK下载
离线词&语义设置
离线合成SDK管理
私有部署服务管理
应用列表
使用说明
1. 您可通过应用的 API Key和 Secret Key获取Access_token来调用百度AI服务；为了您的财产和服务安全，请妥善保存和定期更新Secret Key。查看鉴权认证机制>
2. 快速搜索AI全部应用详情，可使用自助工具。AI应用详情查询>
创建应用
序号
应用名称
AppID
API Key
Secret Key
创建时间
创建人
操作
1
语音识别
59391349
wmHNw... 展开 复制
Z4q63... 展开 复制
2024-04-02 14:19:52
石界完美
报表 管理 删除
共 1 条
购物车
CSDN @m0_64440031

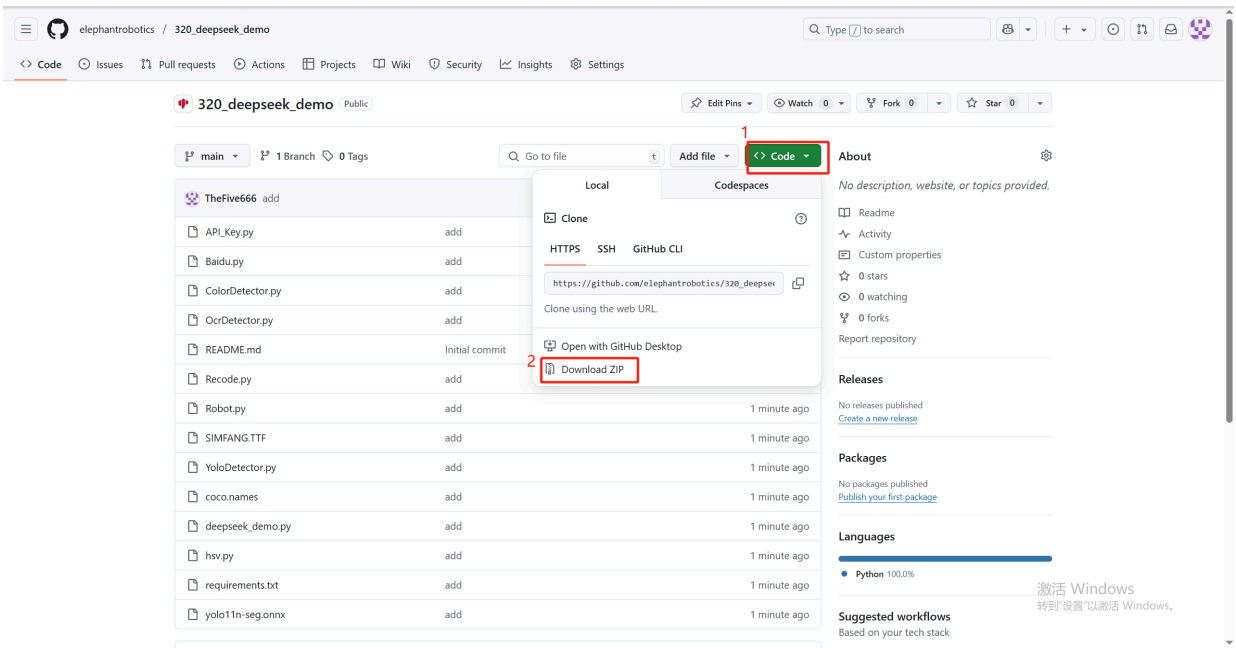

返回
创建新应用
* 应用名称:
语音识别
* 接口选择:
勾选以下接口，使此应用可以请求已勾选的接口服务；为保障您的资源安全，请取消勾选不再产生调用的接口；请注意，您无法请求未选中状态的接口服务。
语音技术
全选
短语音识别
短语音识别极速版
实时语音识别
音频文件转写
短文本在线合成
呼叫中心语音
远场语音识别
任务创建
长文本在线合成
文字识别
人脸识别
机器翻译
* 语音包名:
IOS
Android
不需要
* 应用归属:
公司
个人
* 应用描述:
简单描述一下您使用人工智能服务的应用场景，如开发一款美颜相机，需要检测人脸关键点，请控制在500字以内
购物车
CSDN @m0_64440031



5 Source code acquisition

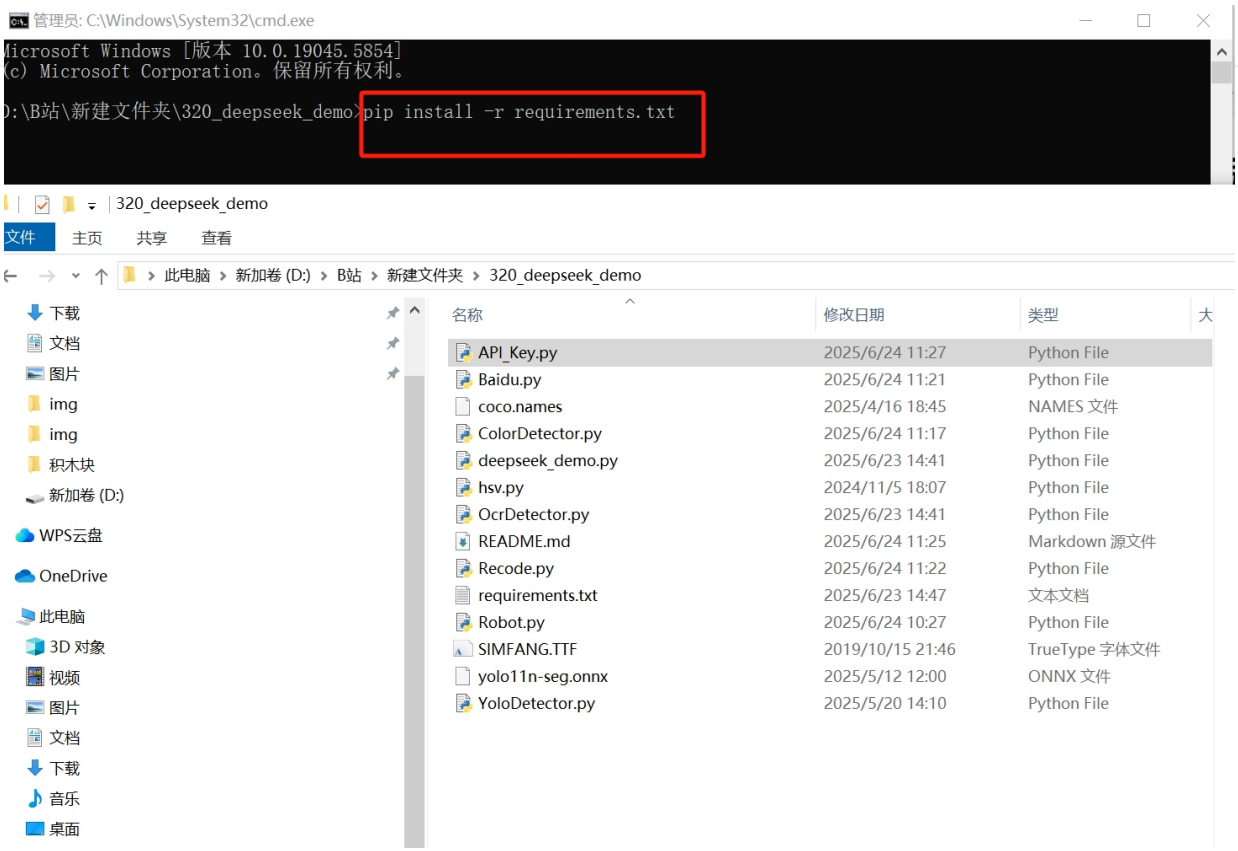
Download address: https://github.com/elephantrobotics/320_deepseek_demo

Download the compressed package and solve it before deploying the environment



Dependency library installation

```
pip install -r requirements.txt
```



API_Key fill in

In the API_Key.py script file, fill in your Baidu APP_ID, API Key, Secret Key

7 Case reproduction

Run deepseek_demo.py, press the Enter key, and the recording will start

Notes When recording, it is necessary to do it in a quiet environment. When converting voice to text, the conversion may be incorrect. You can wait until the robot arm completes the action, and then press the Enter key again to speak the command

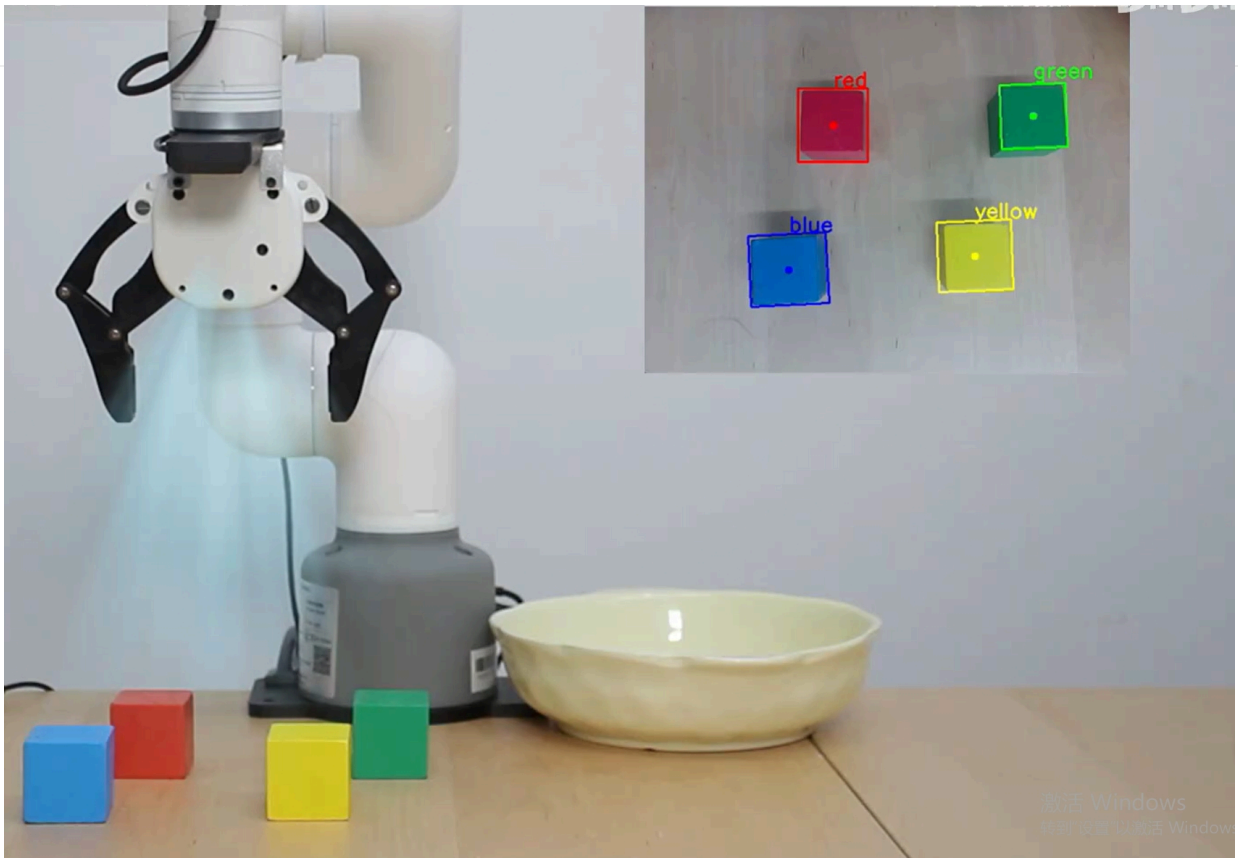
7.1 Mobile display

Voice command words:

```
向前移动50毫米，向后移动40毫米，向左移动30毫米，向右移动60毫米。
```

7.2 Color recognition

The colored wooden blocks should be placed neatly, and a certain distance should be kept between the wooden blocks, otherwise the robot arm cannot complete the grasping normally , please refer to the following figure for placement

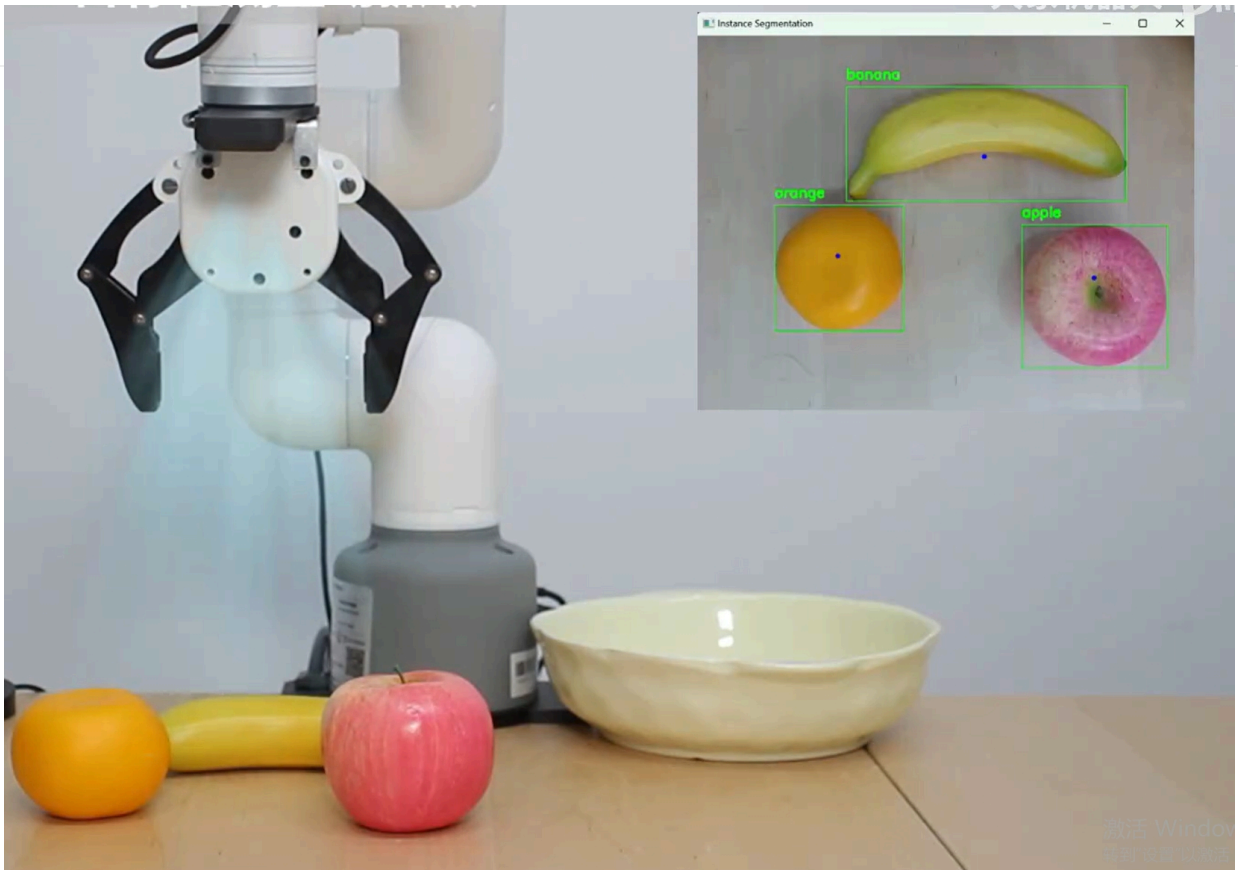


Voice command words:

```
先把番茄一样颜色的木块拿走,再把和香蕉一样颜色的木块拿走
```

7.3 Fruit recognition

The fruit models should be placed neatly and a certain distance should be kept between them, otherwise the robot arm cannot grasp them normally. Please refer to the following figure for placement. If you want to grasp a banana, place it vertically



Voice command:

```
我现在有点口渴，帮我拿个橙子。
```

```
我现在有点饿，帮我拿个苹果。
```

Note: Currently, fruit recognition only supports recognition of apples, oranges, and bananas

7.4 Drug recognition

The medicine boxes to be grasped must be placed neatly and vertically, without too much angle deviation, and a certain distance should be kept between the medicine boxes, otherwise the robot arm cannot grasp them normally. Please refer to the following figure for placement.



Notes: Currently, the medicine box recognition only supports the recognition of 999 Ganmao Ling Granules and Yunnan Baiyao Band-Aids. If you want to reproduce, please prepare this medicine box in advance

Voice command words:

```
我有点流鼻涕和鼻塞，帮我取个治疗这个症状的药。
```

7.5 Case video

Video link: https://www.bilibili.com/video/BV1coMczkEGd/?spm_id_from=333.337.search-card.all.click&vd_source=672e3f7240eaaca210b45e7c033dc45f

This chapter provides comprehensive downloads of relevant information, including product information, product drawings (2D and 3D), software information and source code (internal software download, commonly used software download, source code download area), system information (system image, image burning) , promotional materials (product brochures, product videos, product pictures). Each section provides clear instructions and version information so users can easily select and download the materials they need. At the same time, the burning tools and tutorials are also introduced in detail, allowing users to smoothly burn system images. Overall, this chapter provides a full range of data download services, providing users with a convenient way to obtain and use product data.

← Previous Chapter | Next Chapter →

2D Drawing

Robot	Robot 2D drawing
myCobot 320 Atom	download
myCobot 320 M5 base	download

3D Drawing

Robot	3D model file
myCobot 320 M5 2020 ver.	download
myCobot 320 M5 2022 ver.	download
myCobot 320 M5 2022 ver. v1.2	download

Software Download

- **myStudio**

  myStudio is our own software. It is a tool for firmware burning or modification of the existing robotic arm launched by our company.

  github download：GitHub - elephantrobotics/myStudio: A comprehensive software for mycobot. Provides firmware burning, documentation, tutorials, etc

  Website：Software Download - Elephant Robotics

- **python**

  Python's official downloading address

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

  Go to **PyCharm** to download PyCharm.

- **VMware**

  https://www.vmware.com/cn/products/workstation-pro.html

- **vituralbox**

  https://www.virtualbox.org/wiki/Downloads

- **VisualStudio**

  https://visualstudio.microsoft.com/zh-hans/vs/

- **Arduino**
  - Arduino Official Website
  - Windows X64
  - Mac OS X
  - Linux ARM 64

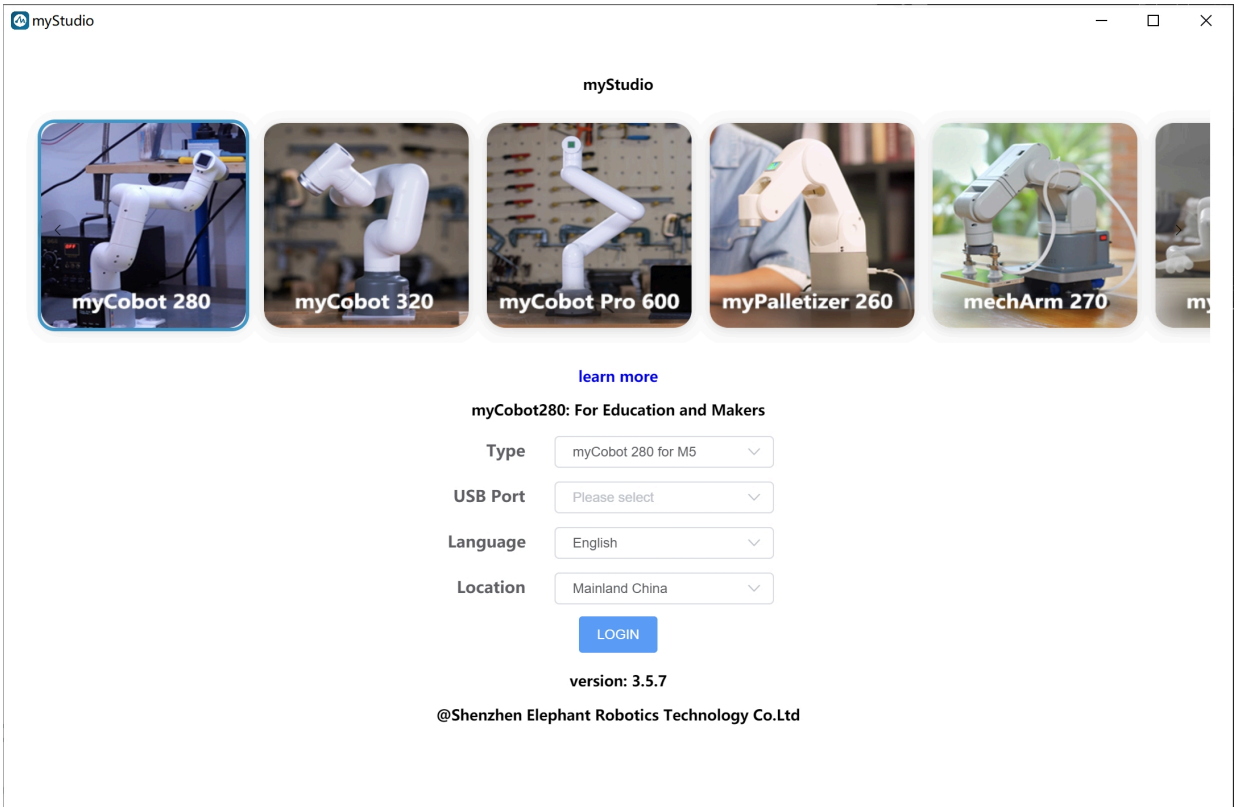
Souce Code Download

ROS1 Source Code：GitHub - elephantrobotics/mycobot_ros: A ros package for myCobot.

ROS2 Source Code：GitHub - elephantrobotics/mycobot_ros2: myCobot ROS2 package

pymyCobot： **GitHub - elephantrobotics/pymycobot: This is a python API for mycobot serial communication.**

System Firmware



1 Reasons for Designing myStudio

- Because myStudio is a one-step application platform for a variety of robots.
- Because it is easy for users to select corresponding firmwares to meet demands and acquire related tutorial data online.

2 Latest Version and Supported Systems

- Version 3.4.1(updated on Oct 1, 2022)
- Supported Systems: Windows, Mac and Linux

3 Functions

- Burning and updating firmwares
- Providing tutorial data, such as user manuals and tutorial video
- Providing information about maintenance and repair

5 Recommended Firmwares

The optimal firmwares varies with the type of robots in use. Recommended firmwares for different types of robots are listed as follows.

myCobot 320（2022） series

myCobot 320（2022） series have two versions: M5 and PI. With different core for signal connection, different firmwares are required to be burnt.

Robot Version	Core	Firmware to be Burnt	Recommened Firmware
M5	M5Stack-basic	miniRobot	v2.0 is recommended
	Atom	atomMain	v5.0 is recommended
	Pico	picoMain	v1.0 is recommended
PI	RaspberryPI 4B	ubuntu	v18.04. is recommended
	Atom	atomMain	v5.0 is recommended
	Pico	picoMain	v1.0 is recommended

myCobot 320（2020）series

Robot Version	Core	Firmware to be Burnt	Recommened Firmware
M5	M5Stack-basic	miniRobot	v1.0 is recommended
	Atom	atomMain	v4.2 is recommended
PI	RaspberryPI 4B	ubuntu	v18.04. is recommended
	Atom	atomMain	v4.2 is recommended

Product Brochure

Robot	Brochure
myCobot 320 M5	download

Product Video



Product Photos







About Us

- 9.1 Elephant Robotics
- 9.2 Contact us

← Previous Chapter

Elephant Robotics



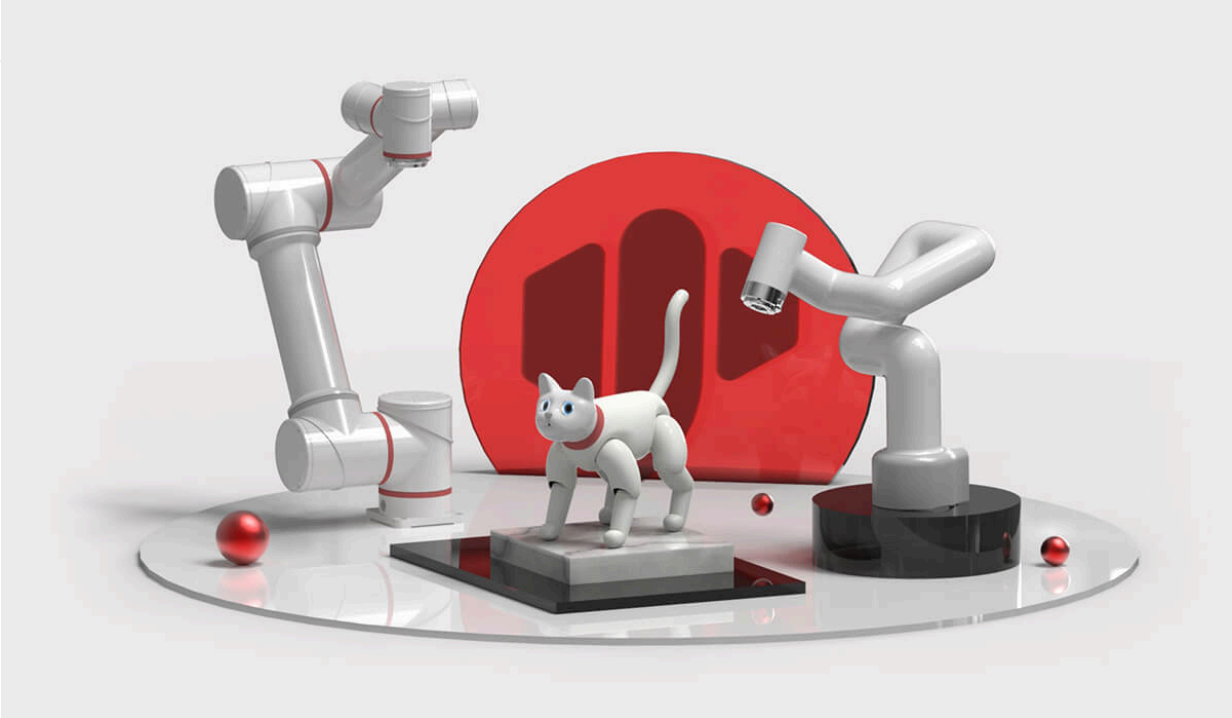
1 Company Introduction

Elephant Robotics is based in Shenzhen, China, a high-tech company which focuses on the design, research and development of robots and automation solutions.

We are devoted to providing highly flexible robots, easy-to-learn operating systems and intelligent automation solutions for robot education, scientific research institutions, business situations and industrial production. Our product quality and intelligent solutions have obtained unanimous acceptance and favorable comments from a number of world top 500 enterprises & factories in South Korea, Japan, America,Germany, Italy, Greece, etc.

Abiding by the vision of "enjoy robots world, Elephant Robotics initiates collaborative work between human and robots to make robots be good life and work helpers for human so as to free people from simple,repeated and dull jobs and give full play to the advantages of human-robot coordination, thus improving work efficiency and helping human to create a nice, 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 starts a new era of automation and intelligence with its customers hand in hand.



2 Development History

In August 2016, Elephant Robotics was established.

In August 2016, entered HAX Incubator and obtained SOSV seed round investment.

In July 2017, the two founders were included in Forbes Asia\'s "30 Business Elites under Age 30".

In October 2017, published the fifth generation of single-arm industrial cobot called Elephant S.

In April 2018, obtained angel round investment from Cloud Angel Fund.

In June 2018, was awarded "Intelligent Manufacture Entrepreneurship MBA Award" by CKGSB.

In June 2018, was awarded "Startup Accelerator X-elerator Award" operated by Tsinghua University.

In November 2018, won the second place in the Asian Smart Hardware Competition in Shenzhen Division

In November 2018, obtained the \"Most Invested Company Award\" in GaogongGold Globe Award.

In March 2019, obtained the \"Leading Person Award \" in Gaogong Gold Globe Award.

In April 2019, obtained Catbot "Industrial Robot Innovation Award".

In September 2019, attended Huawei European Eco-Conference (HCE) and became a member of Huawei eco-partners.

In November 2019, Elephant Robotics attended the IROS International Conference on Intelligent Robots and Systems jointly with Harbin Institute of Technology.

In December 2019, obtained "Gaogong 2019 Innovation Technology Award".

In December 2019, was awarded as one of the Gaogong 2019 Top 10 Fast Growing Enterprises.

In December 2019, was awarded the \"Emerging Enterprise Award" in the industrial robotics segment field of Shenzhen equipment industry.

In December 2019, launched the fist type of bionic robotic cat called MarsCat in the world.

In May 2020, the founders obtained \"Shenzhen Robot Emerging Talent Award" in 2019.

In October 2020, launched the smallest six-axis cobot named myCobot in the world.

In March 2020, launched the smallest cobot named myCobotPro 320 for scientific research in the world.

In May 2021, the Mars bionic cat named MarsCat was reported by several media such as Xinhua Finance, China Daily, Nanjing Daily, Harbin Daily, etc.

In July 2021, published the seat for the smallest hybrid robot, a baby elephant moving robot called myAGV.

In September 2021, launched the world's first type of fully wrapped four-axis robot arm, a tiny elephant palletizing robot arm called myPalletizer.

3 Related Links

- Official website: https://www.elephantrobotics.com
- Purchase link
  - Shopify : https://shop.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

← Previous Page| Next Page →

Contact Us

> Our working hours are on Chinese working days, from 10 AM to 6 PM Beijing time.

If you have any other problems, contact us via the ways below.

Email :

If you have purchase intention or any parameter questions, please send an email to this mailbox.

E-mail :

```
sales@elephantrobotics.com
```

If the listed problems can't help you solve and you have more after-sales questions, please send an email to this mailbox.

E-mail :

```
support@elephantrobotics.com
```

We will give a reply within 1-2 business days;

WeChat:

We provide one-to-one service only for those users who have purchased

myCobot via WeChat.



← Previous Page