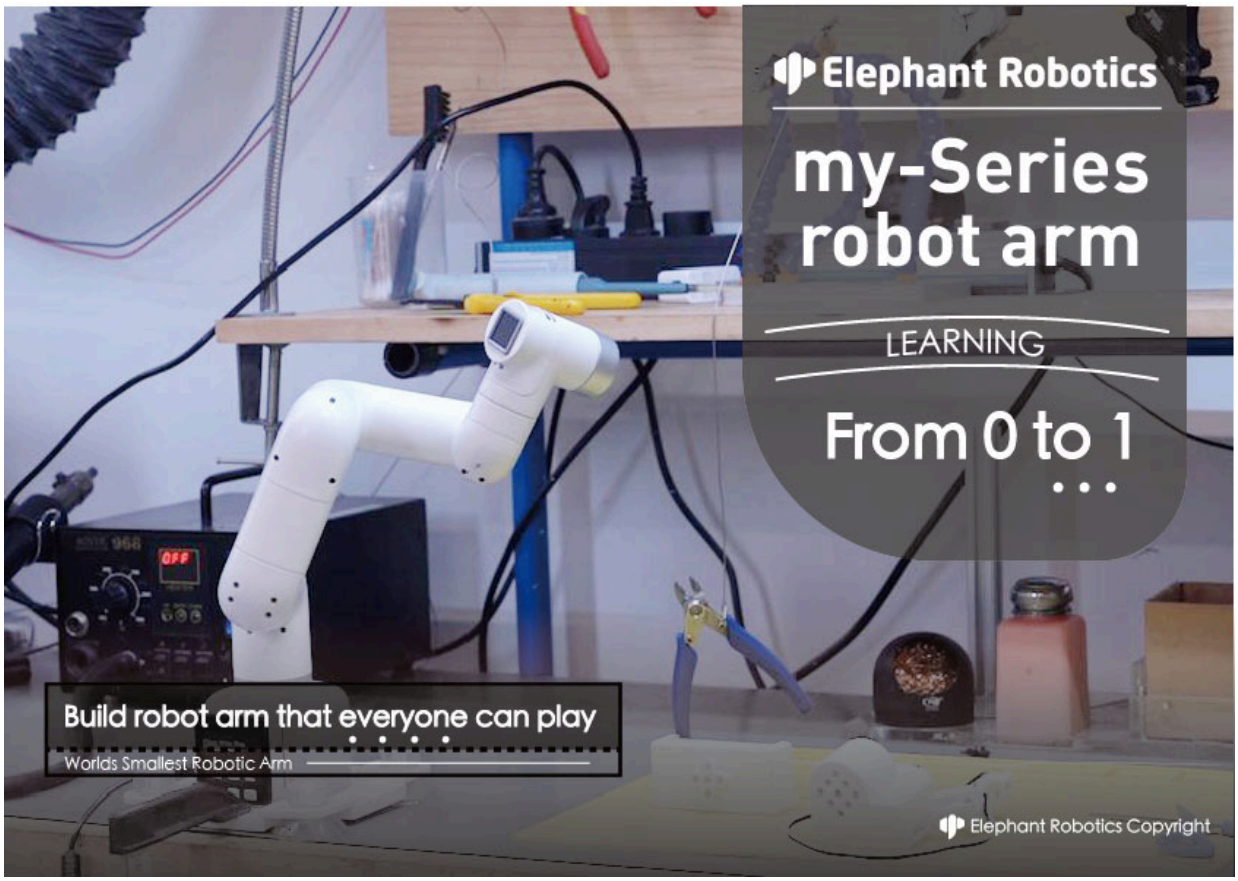
Table of Contents

Introduction

Preparations

Development and API

myCobot: From 0 to 1



1.1 Why do we design myCobot

> **An entry-level collaborative robot arm** that everyone can learn and play

The original design of myCobot is to help friends who are interested in 6-axisseries robot to learn it from entry to master, creating unprecedented experience and teaching value.

What you can learn

> Robotics is based on rigid body kinematics and dynamics, but also an interdisciplinary subject that combines hardware, software, algorithm and control.

With myCobot, you can learn that

- **Hardware**
  - **Embedded Microcontroller Based on ESP32**
  - **Motor and Steering Gear**
  - **M5Stack Basic/ Atom**
- **Software**
  - **Arduino开发环境**
  - **C++**

    - **Python**
    - **ROS，MoveIt**
    - **Communication Data**
    - **Virtual Machines & Linux (visual system)**
- **Algorithm**
    - **Series Manipulator**
    - **Coordinate and Coordinate Transformation坐标与坐标转换**
    - **DH Parameters**
    - **Kinematics**
    - **Manipulator Algorithm (e.g. dynamics)**
- **Machine Vision (Vision Set)**
    - **Color Recognition**
    - **Image Recognition**
    - **Hand-Eye Calibration**
    - **See and Grab**
- **Extended Applications**
    - **End-effector: gripper,suction pump, etc.**
    - **Robot Suit & Industry 4.0 Applications**

Parts of Gitbook

> View the directory on the left to jump

There are four major parts of Gitbook :

- **Introduction & Quick Start**
    - **Introduction** -- introduce what myCobot is and its main features, etc.
    - **How to Read** -- help you read Gitbook efficiently according to your learning level and knowledge background

  - **Use Cases** -- you can know exactly what use cases you can accomplish with
  - **Quick Start** -- learn the unboxing of your myCobot, and its first boot and use

- **Preparation before Development**
  - **Background Knowledge** -- learn about tools, industrial robots, algorithms, software, hardware,etc.
  - **Hardware Learning** -- learn about embedded hardware, structural components, electronic components, etc.
  - **Purpose of Use** -- identify the purpose you want to use it for, and complete the study related to your task

- **Development and Use**
  - **Development Environment** -- learn to use Arduino, ROS, uiFlow, roboFlow, python and others development environment to develop myCobot
  - **Accessories** -- learn to use myCobot with different accessories, such us bases, grippers, suction pumps and so on
  - **Machine Vision** -- learn to control myCobot under the guidance of machine vision
  - **Robot Modification** -- learn how to modificate myCobot into a 4 or 5 axis manipulator

- **myCobot Suit**
  - **Intelligent Warehouse:** learn how to use myCobot to carry different objects
  - **Artificial Intelligence:** learn how to control myCobot to grasp objects intelligently under the guidance of machine vision
  - **Industry 4.0:** learn how to grasp and place objects intelligently by simulating production line

Information Source

- **Official website:** www.elephantrobotics.com
- **Tutorial video:** https://www.youtube.com/channel/UC68l2RaRF2Mp8fzpCTzNBfA
- **Shop website:** https://shop.elephantrobotics.com
- Download PDF

Contact Us

> **If you have any other questions, you can contact us as follows.**
>
> **We will answer you as soon as possible (working day 9:30-18:30)**

- Twitter: myCobot Official\@CobotMy
- Facebook: https://www.facebook.com/MyCobot-116558893805177
- Mail: support@elephantrobotics.com

Elephant Robotics

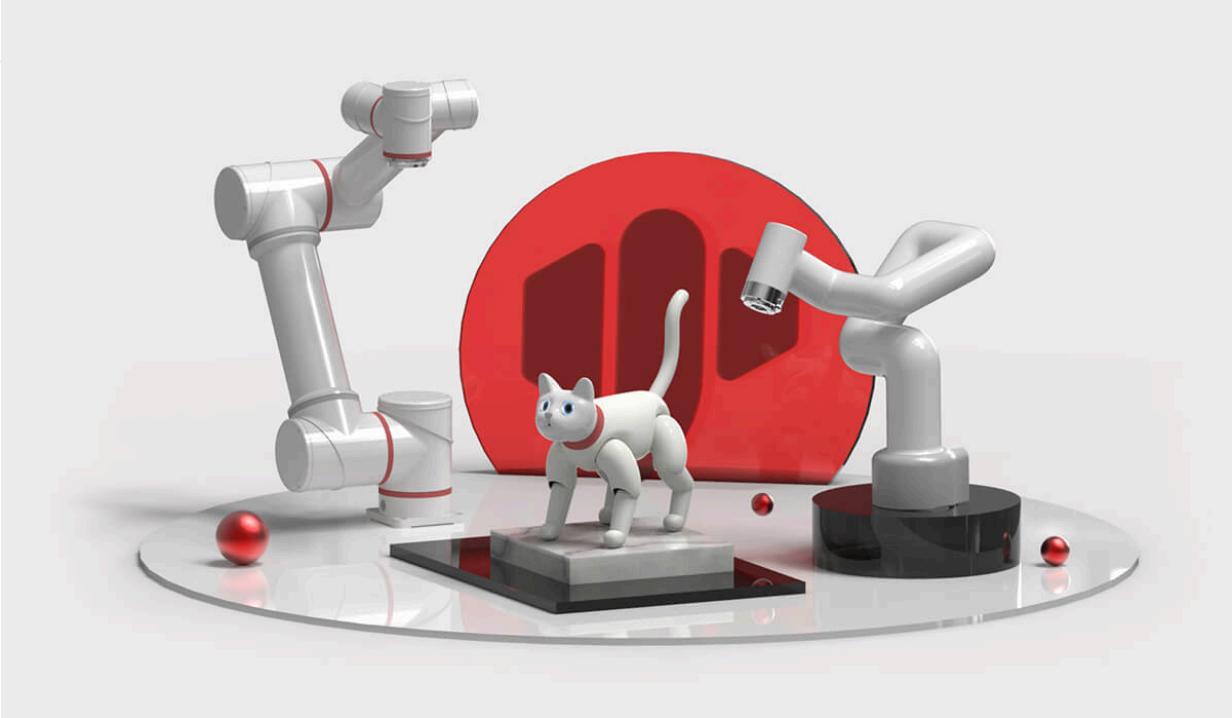
1 Company Introduction

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

4 Contact Us

If you have any other problems, contact us via the ways below. + Email: We will give a reply within 1-2 business days; **Email + WeChat**: We provide one-to-one service only for those users who have purchased myCobot via WeChat.

Development history of my series of products

Development History

In October 2020, we launched the smallest six-axis cobot named **myCobot** 280-M5 in the world.

In December 2020, **myCobot 280** was put on the market.

In April 2021, **myCobot 320** products was put on the market.

In May 2021, we published the Raspberry Pi version of **myCobot 280**.

In June 2021, we published the Raspberry Pi version of **myCobot 320**.

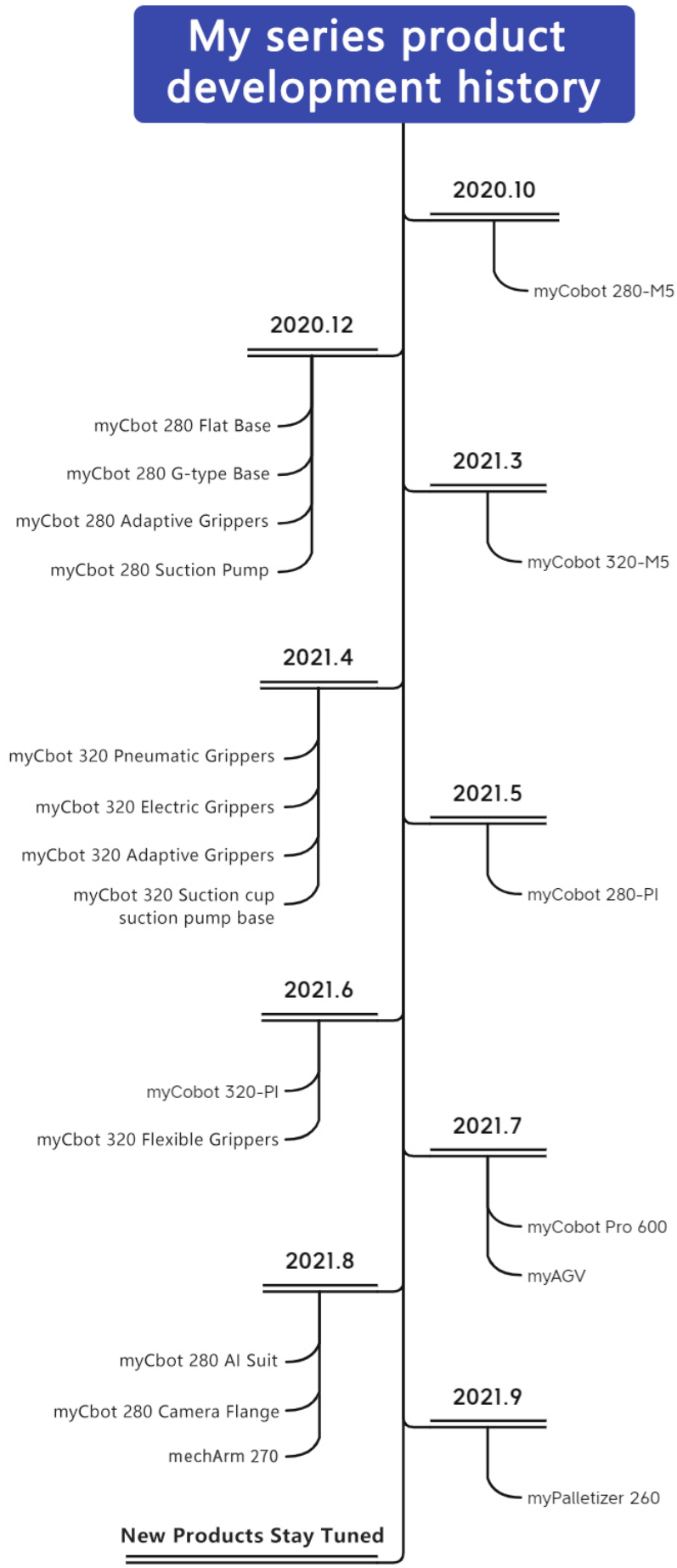
In July 2021, we published the commercial version of a babyelephant coordinative robot arm called **myCobot Pro 600**.

In July 2021, we published the smallest hybrid robot, a baby elephant moving robot called **myAGV**.

In August 2021, **AI kits** and the computer vision enabled for visual machines were put on the market.

In August 2021,The world's most compact and portable small six-axis mechanical **mechArm 270** series came out

In September 2021, launched the world's first type of fully wrapped four-axis robot arm, a tiny palletizing robot arm called **myPalletizer**.


My series product development history
2020.10
myCobot 280-M5
2020.12
myCbot 280 Flat Base
myCbot 280 G-type Base
myCbot 280 Adaptive Grippers
myCbot 280 Suction Pump
2021.3
myCobot 320-M5
2021.4
myCbot 320 Pneumatic Grippers
myCbot 320 Electric Grippers
myCbot 320 Adaptive Grippers
myCbot 320 Suction cup suction pump base
2021.5
myCobot 280-PI
2021.6
myCobot 320-PI
myCbot 320 Flexible Grippers
2021.7
myCobot Pro 600
myAGV
2021.8
myCbot 280 AI Suit
myCbot 280 Camera Flange
mechArm 270
2021.9
myPalletizer 260
New Products Stay Tuned

How to read

1 The target of reading the book

The book is designed to help you reach the following targets.

> Main targets

- Understand the basic use of mechanisms, electronics and software related to robot arms.
- Understand the basic principle, joints, coordinates, terms, control and other information of the robot arm. Able to do simple forward and inverse kinematics calculations.
- Understand the basic operations of API-controlled robot arms and the robot arms controlled by myBlockly graphical programing language.

> Extended targets

- Understand the image recognition algorithms related to machine vision.
- Understand the building of robot vision scenes and the methods and strategies of coordination between vision and robot arm.
- Grasp all skills of artificial intelligence (AI) packages.

2 Division of your background

You should read the book based on your background. We divide the background and related knowledge into three degrees:

Degree	Background	Skills	Estimated time of learning	Recommended platform
Beginner	Specialty related to information, electronics and automation	Understand a kind of programming language and the basic knowledge related to electronics	100 hours	myBlockly
Advanced	Understand Arduino or similar hardwares, servo and programming, IO interface, etc	Able to debug API and interfaces, and understand communication	50 hours	Arduino
Professional	The readers used at least one industrial or consumer robot arms, and have the ability to develop hardware and software	Understand the Cartesian coordinate system, joint control, and basic use of robots	30 hours	Random

3 Learning Steps and Time

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
1	**Quick unpacking**	1 Drag teaching	1. The accessories for myCobot 2. drive the robot to perform drag teaching	1 hour
2	**Background knowledge learning**	1. Use background of industrial robots; 2. coordinate and space learning, and Cartesian 3D coordinate and rotation, xyz; 3. joint and coordinate control of industrial robots	1. Joint control and recurrence of robot joints 2. speed control of robots 3. control and cycle of robot coordinate points	5 hours
3	**Hardware learning**	1. Principles and operations of embedded electronics 2. the principle and knowledge of servo and motor 3. actuator learning	1. Basic/atom control and driving 2. driving and motion of servo 3. robot accessory learning	5 hours
4	**Software and firmware and their updating**	1. Identify different software platforms and their use 2. firmware loading and adaptation principle	1. Select the developing platform suitable for you 2. load and update the corresponding firmware.	2 hours

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
5	**Building of software development environment**	1. Build Arduino platform 2. load and update of Arduino library file 3. understand serial communication	1. Familiar with Arduino platform 2. load a library 3. operate and run the first line of codes	2 hours
6	**Learning and development of robot library**	1. Basic communication and operation types of robots 2. common operating methods of robots 3. control of direction and coordinate modes	1. Communicate with the robot 2. control the robot to move 3. operate the IO interface, gripper, etc. of the robot;	5 hours
7	**myBlockly operation robot arm**	1. Understand the basic architecture and relation of graphical programing language interfaces: sensor, actuator and procedure 2. variable, cycle and judgment 3. control method of robot arm	1. Display different fonts in the basic 2. make the robot arm move to different positions using three buttons of the basic 3. control the robot arm to make it move to several positions circularly	10 hours

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
8	**The use of roboFlow**	1. Learn the industrial operating systems commonly used for robots 2. learn the common modules for roboFlow: point, quick movement, IO control and output 3. learn the advanced modules of roboFlow: cycle, judgment, and pallet program	1. Control the movement of the robot arm 2. basic control of IO input and output 3. cycle control and judgmen	5 hours
9	**Algorithms related to image recognition**	1. Common color recognition methods and strategies 2. common shape recognition methods and strategies 3. common area recognition methods and strategies	1. Building of a ROS environment 2. reading of different colors 3. recognition of different shapes	20 hours
10	**Vision and the joint debugging of the robot**	1. Connect the world with a camera coordinate system 2. QR code image calibration 3. movement and correction	1. Operate the robot arm to the camera coordinate system 2. the robot arm moves in the camera coordinate system 3. recalibrate and set	10 hours

No.	Target knowledge points	Theory	Practice	Estimated hours of learning
11	**Artificial intelligence (AI) package**	1. Flow chart learning and making 2. electrical connection diagram learning and making 3. operation strategies such as image recognition and classification, etc	1. Sensor connection 2. gripper actuator connection and driving 3. robot arm driving and visual joint debugging	20 hours

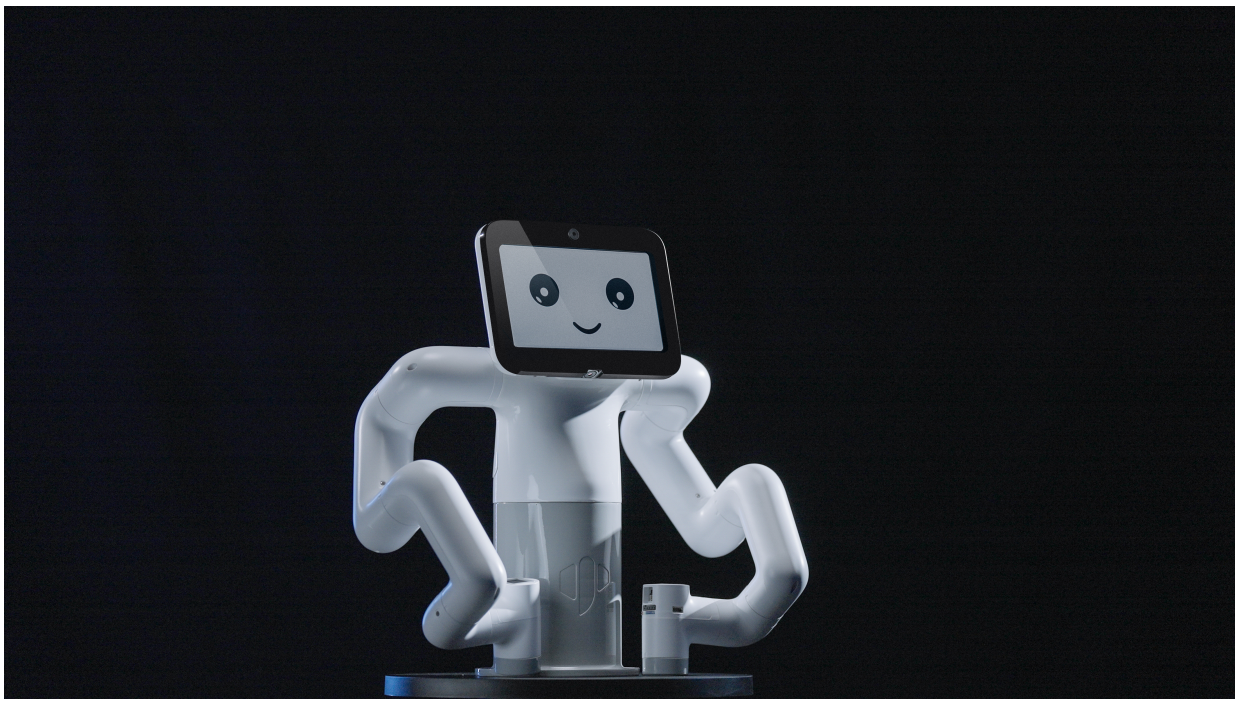
4 Additional problems

If the above learning contents cannot meet your actual use needs, you can contact **Elephant Robotics helper** for further communication. We provide customized services for software and hardware, and the service fee is based on the **actual cost**.

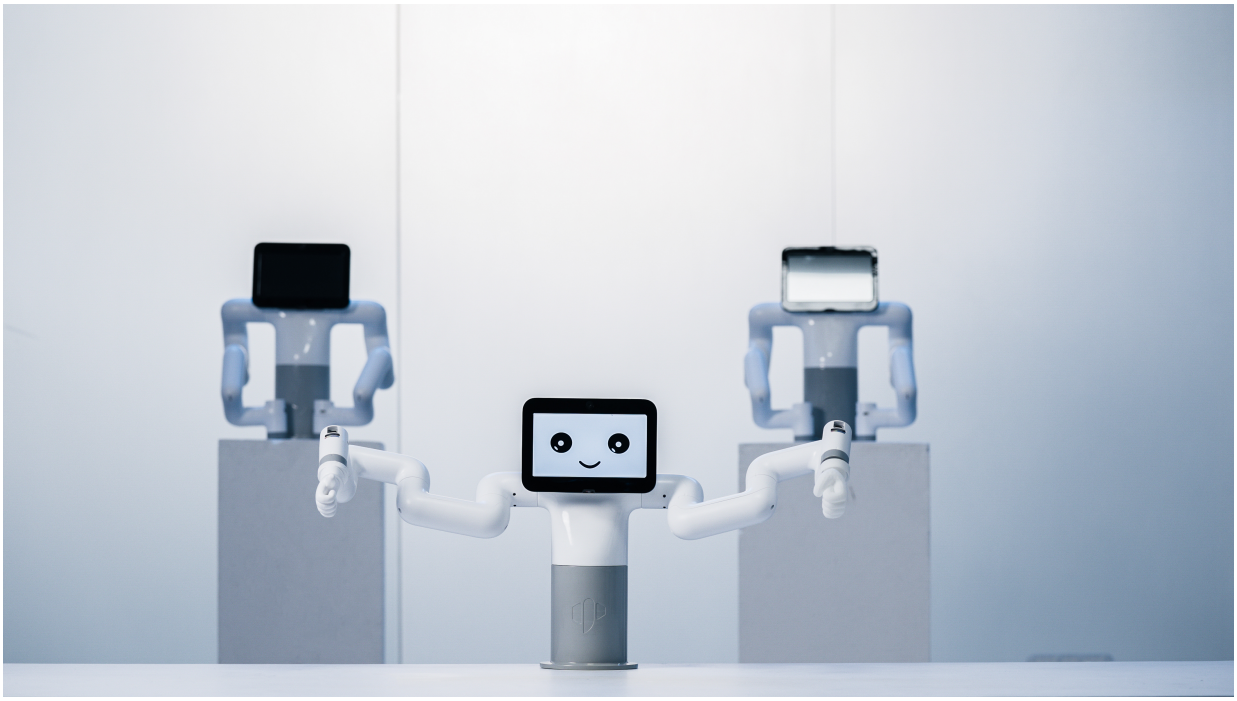
myBuddy 280

1 Product Introduction

The **myBuddy280** dual-arm collaborative robot is a joint product between **Elephant Robotics** and **Raspberry Pi**. It is a **dual six-axis** humanoid collaborative robot with its own **7"** interactive display screen, built-in **20+** dynamic expressions for direct application, provides dual **2million** pixel HD cameras for **image vision** development, provides a standard **3.3V expansion io** interface, provides a **Lego** expansion interface, can be fitted with a variety of adapters such as **suction pumps**, **grippers**, etc., provides **teaching documentation** related to machine vision learning.



myBuddy



1 Product Performance

- **Easy operation and open source**
  - The user can learn the operation of the product in a short time using drag teaching and myblockly simple visual programming.
  - Support ROS/MOVEIT and other development systems and the myBuddy APP operating software independently developed by Elephant Robotics.
- **Affordable and cost-effective**
  - Affordable robotic arm，effectively reducing costs and increasing efficiency for scientific research that requires high performance & low cost.
  - Individual developers with accessories can do creative development to meet a variety of scenarios.
- **Powerful performance, equipped with 13 high-performance servo servos**
  - Using 13 high-performance brushless DC servos, it can achieve a repeatable positioning accuracy of ±0.5mm.
  - Excellent algorithm control in the industry, the fastest command response speed can reach 30ms.
- **Super perfect python control interface**
  - Provide 100+ control interfaces, which can be applied to secondary application development or self-interference algorithm research.
  - Open joint angle and speed control interface, open robot coordinate control interface, make the control more simple and easy to use.
- Supports separate control of left and right arms and waist, making the control freer.

- Provide programming sample programs, which can quickly implement scene applications.

- **7" interactive display**
  - Standard seven-inch interactive display, which can be used for image display, touch control.
  - Built-in 20+ dynamic expressions, ready to use.
  - The use of high-strength wear-resistant surface layer improves the service life of the touch screen.
- **Built-in camera and driver library**
  - Built-in two 2 million pixel cameras, support up to 1080P image capture.
  - Built-in OpenCV development environment for direct visual development
  - Provide visual identity development cases to make development easier.
- **Integrated design, safe collaborative operation**
  - The ingenious structural design makes it possible to make full use of the space and perfectly integrate into the actual environment.
  - Integrated industrial design ID, rounded corner design of the whole machine, safer and more beautiful.
  - It has anti-collision detection function so that it can work with people safely.

2 Product Parameters

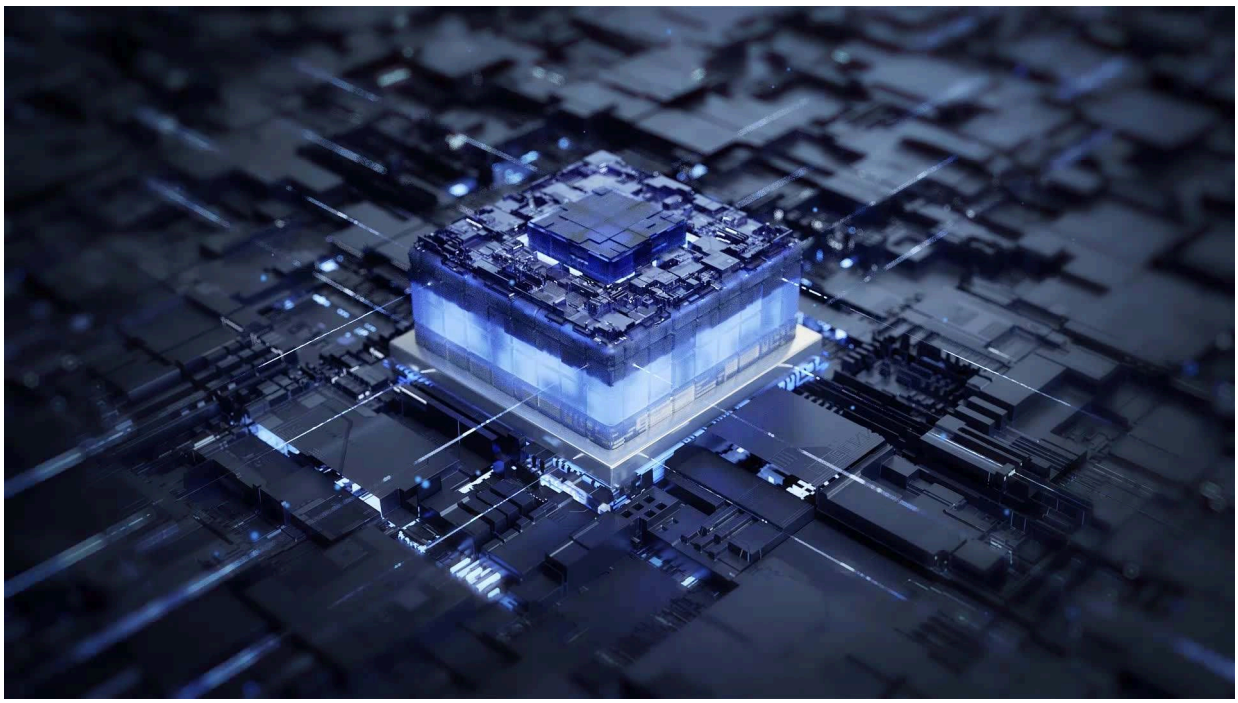
Indicator	Parameter
Name	myBuddy
Model	280
CPU	Broadcom BCM2711，64bit 1.5GHz 4 core
Running Memory	4GB
Memory Card	32GB
Effective working radius	L-ARM 280mm & R-ARM 280mm
Load	L-ARM 250g & R-ARM 250g
Effective arms span	280mm
Repeated positioning precision	±0.5mm
Dead weight	3kg
Power input	24V,9.2A
Operational environment	0°~45°
Interface	Grove & USB & RJ45 & 3.3V IO & HDMI

3 Application Scenarios

myBuddy 280 is a productivity tool as well as a tool to expand the imagination, with a variety of end-effectors adapted to a variety of application scenarios, such as scientific research, educational scenarios, display scenarios, etc., so far the customer feedback is excellent.

Product parameters

| Elephant Robotics provides a wealth of product specifications for our users, including two parts: structure and electronics. Users who need it, please check the sub-documents.

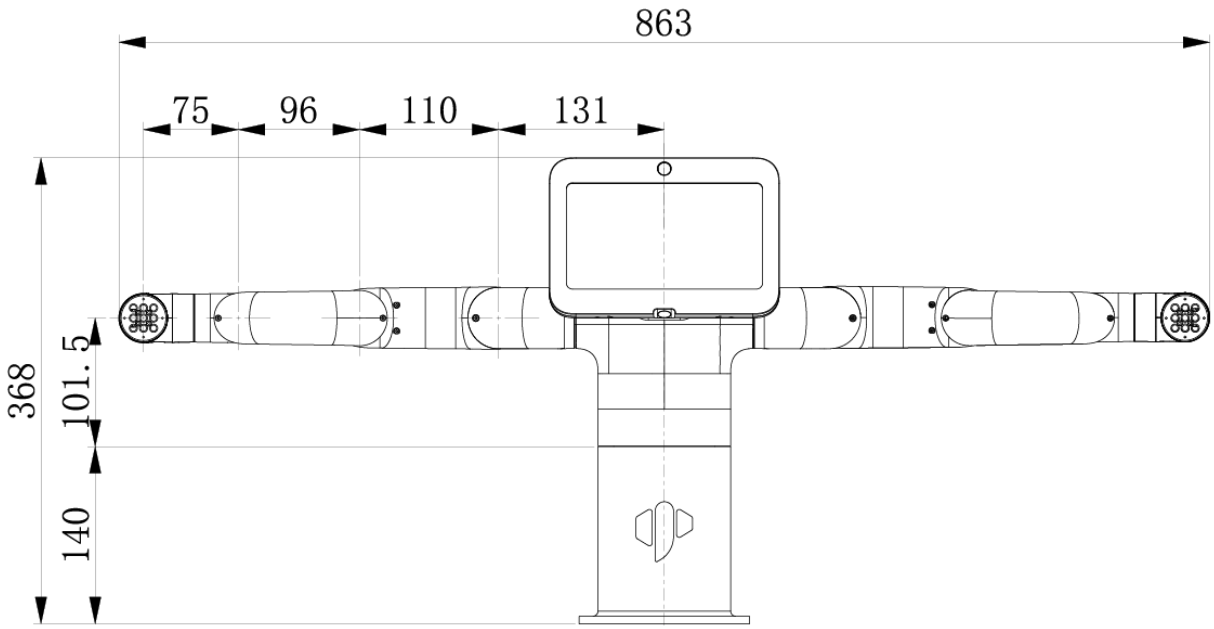


Product Specifications

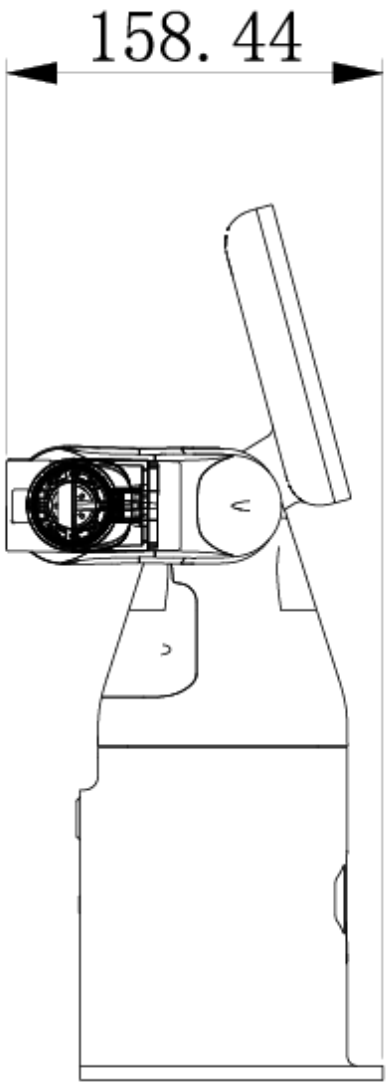
Standard sizes

1、 2D file

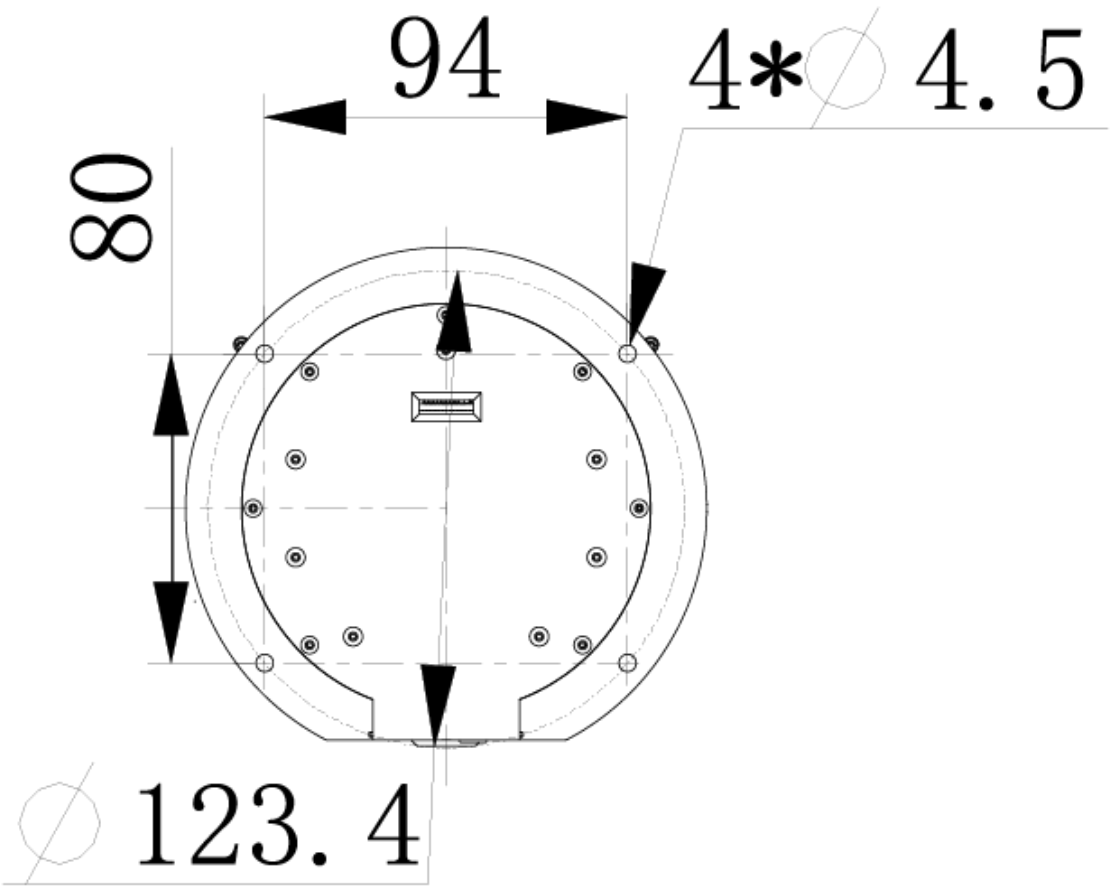
1.1 Front view



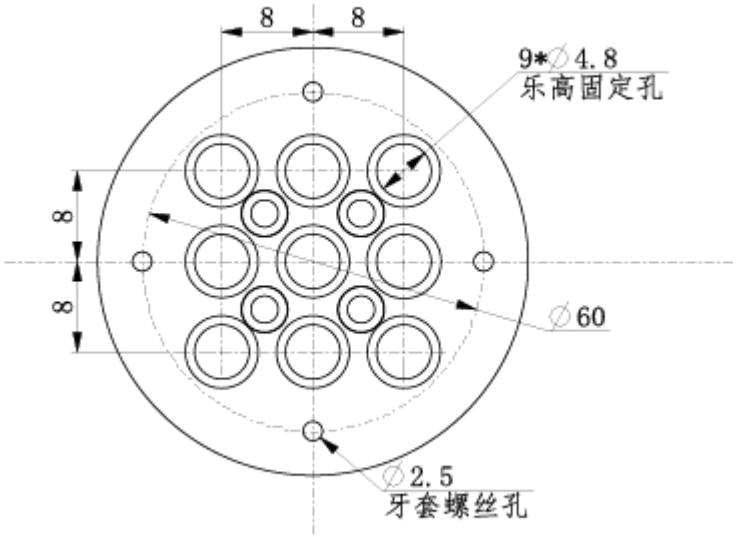
1.2 Side view



1.3 Base mounting holes



1.4 End mounting holes



2、 3D file

3D Download

- Download

Electrical Interface

1 base electrical interface

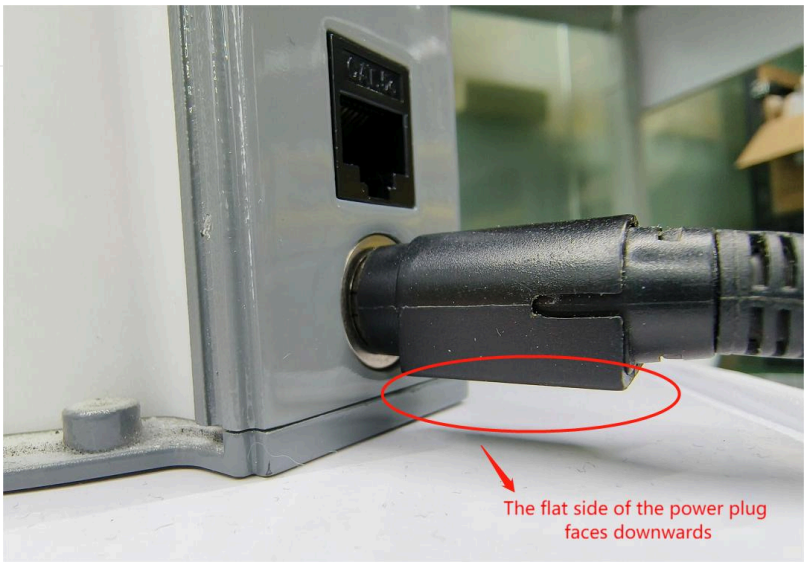
1.1 Base Introduction

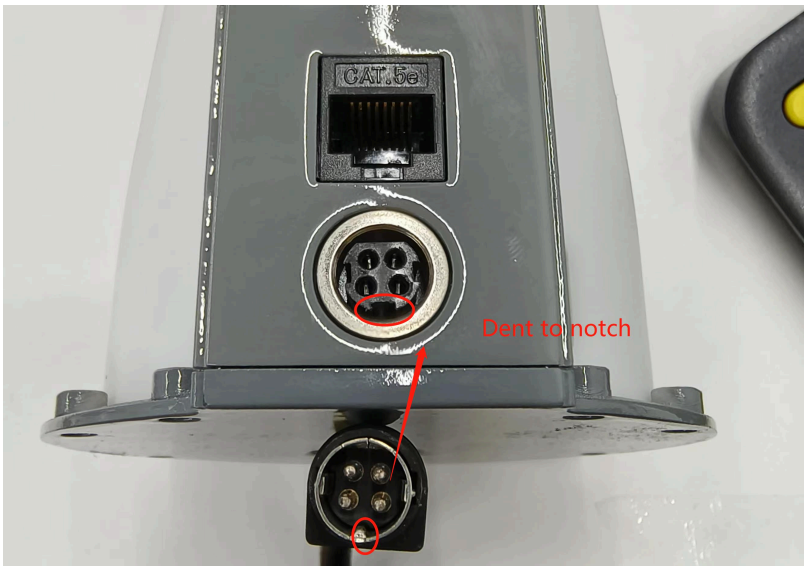
The dock front connector and buttons are shown in Figure 1:

1- Screen toggle light
2- HDMI external interface
3- Screen toggle button
4- External IO interface 1
5- Power connector
6- Power button
7- External USB port
8- External IO interface 2
9- External IO interface 3
10- Network port

Figure 1-1 Front of the base

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





1.2 Chassis interface description

- Note: 14Pin function IO interface 1 is a 2.54mm DuPont interface, external can use 2.54mm DuPont line; function IO interface 2 and function IO interface 3 using Grove interface, as shown in Figure 1-2, 1-3:

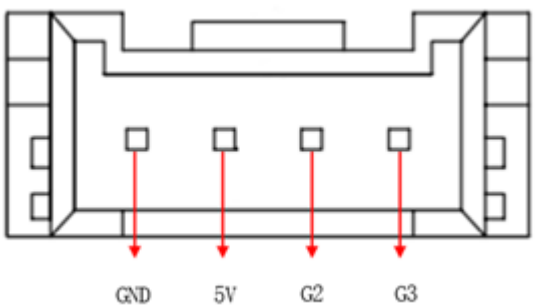


Figure 1-2 Functional IO interface 2

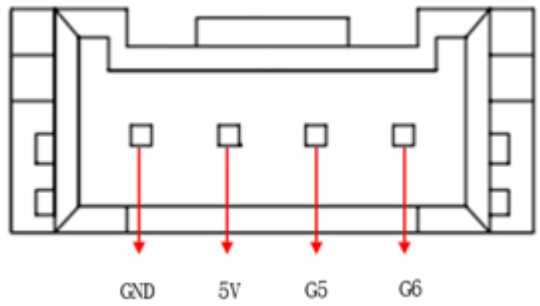


Figure 1-3 Functional IO interface 3

- A. For example, Table 1-1 defines the functions of IO interface 1.

The label name	Signal name	function	remark
3V3	3V3	3.3V power supply	
GND	GND	Motherboard power signal ground	
1	G7	3.3V-OUT-PNP output/3.3V-INT input	
2	G8	3.3V-OUT-PNP output/3.3V-INT input	
3	G25	3.3V-OUT-PNP output/3.3V-INT input	
4	G24	3.3V-OUT-PNP output/3.3V-INT input	
5	G23	3.3V-OUT-PNP output/3.3V-INT input	
6	G18	3.3V-OUT-PNP output/3.3V-INT input	
7	G11	3.3V-OUT-PNP output/3.3V-INT input	
8	G9	3.3V-OUT-PNP output/3.3V-INT input	
9	G10	3.3V-OUT-PNP output/3.3V-INT input	
10	G22	3.3V-OUT-PNP output/3.3V-INT input	
11	G27	3.3V-OUT-PNP output/3.3V-INT input	
12	G17	3.3V-OUT-PNP output/3.3V-INT input	

Table 1-1 IO Interface 1 Feature Table

- B. As Shown in Table 1-2, the definitions of each interface of functional IO interface 2 and functional IO interface 3 are defined.

The label name	Signal name	function	remark
GND	GND	Motherboard power signal ground	
5V	5V	5V power supply	
G2	G2	3.3V-OUT-PNP output/3.3V-INT input	
G3	G3	3.3V-OUT-PNP output/3.3V-INT input	
GND	GND	Motherboard power signal ground	
5V	5V	5V power supply	
G5	G5	3.3V-OUT-PNP output/3.3V-INT input	
G6	G6	3.3V-OUT-PNP output/3.3V-INT input	

Table 1-2 IO interface 2/3 function table

Note: The other function tables about the functional interface are shown in Figure 1-4, and the IO function is not available with other functions:

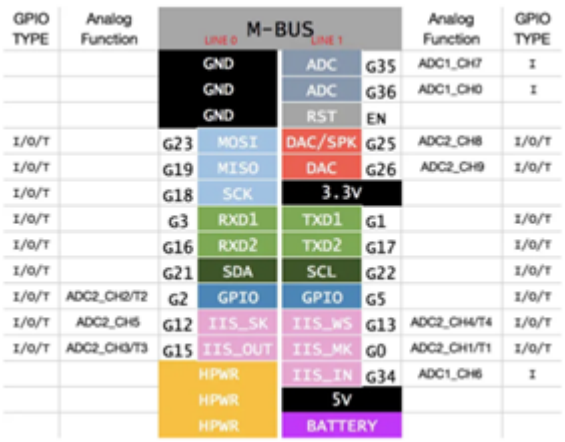
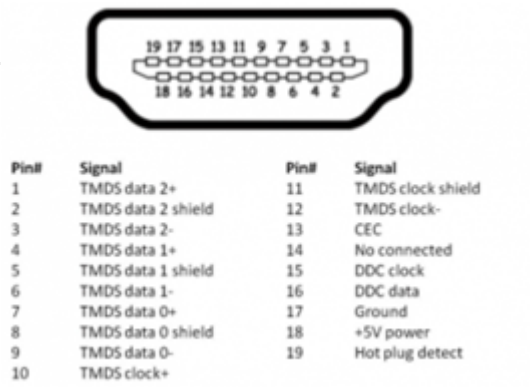


Figure 1-4

- C. Power interface: DC power supply 24V, 9.2A.
- D. Power button: with self-locking button, press on, the whole machine is powered on; Press Disconnect again, the whole machine is powered off.
- E. Screen toggle button: Toggles the internal screen from the external screen display.
- F. Screen switching indicator: When switching to external screen display, the green light is on; When switching to the internal screen display, the light is not lit.
- G. HDMI external interface: （as shown in figure 1-5）This interface is an HDMI type A interface, the user can connect the HDMI display interface to display the operation page to other device terminals, press the screen toggle button can be used.



Pin#	Signal	Pin#	Signal
1	TMDS data 2+	11	TMDS clock shield
2	TMDS data 2 shield	12	TMDS clock-
3	TMDS data 2-	13	CEC
4	TMDS data 1+	14	No connected
5	TMDS data 1 shield	15	DDC clock
6	TMDS data 1-	16	DDC data
7	TMDS data 0+	17	Ground
8	TMDS data 0 shield	18	+5V power
9	TMDS data 0-	19	Hot plug detect
10	TMDS clock+		

Figure 1-5 HDMI interface

- H. External USB interface: （as shown in figure 1-6）serial port bus standard 2.0 interface for data connection; The USB port is used to copy program files and connect peripherals such as mouse and keyboard.

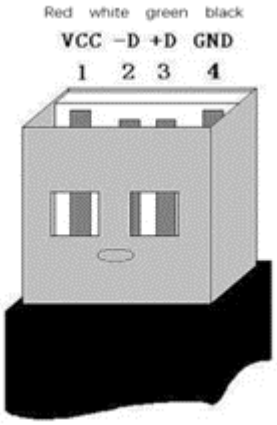


Figure 1-6 Defining USB ports

- I. Network port: （as shown in figure 1-7）ports for network data connection. Ethernet interfaces can be used for communication between a PC and a robot system or for Ethernet communication with other devices.

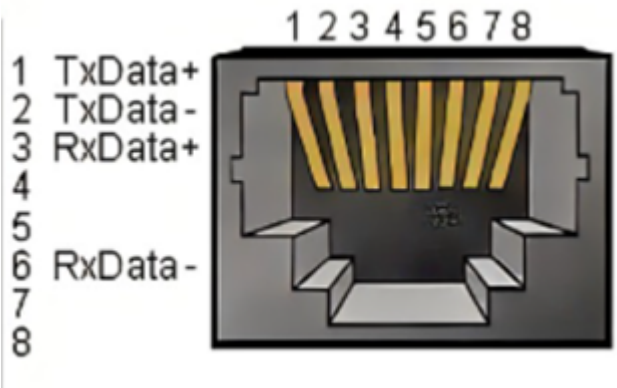


Figure 1-7 Network interface definition

2 robotic arm end electrical interface

Note: The electrical interface at the end of the mybuddy left and right arms is the same.

2.1 Introduction to the end of the robotic arm

- A. The side interface at the end of the picker is illustrated as shown in Figures 2-1 and 2-2:

Figure 2-1 Side view of the end of the manipulator

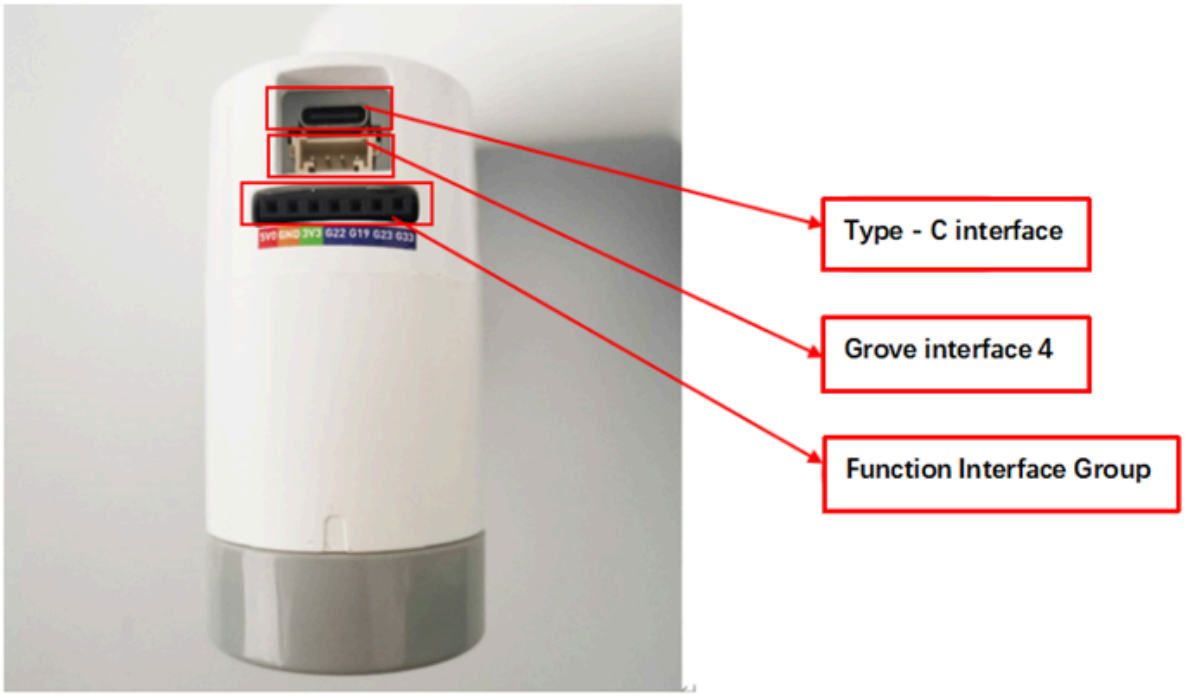


Figure 2-2 Side view of the end of the manipulator

2.2 Description of terminal electrical ports

A. Figure 2-3 shows the definition of each interface in a function interface group.

Tag name	Signal name	Function	Note
5V0	5V0	Power supply, DC5V	
GND	GND	Motherboard power signal ground	
3V3	3V3	DC3.3 V power supply	
22	GPIO22	3.3 V-out-PNP output /3.3 V-int input	
19	GPIO19	3.3 V-out-PNP output /3.3 V-int input	
23	GPIO23	3.3 V-out-PNP output /3.3 V-int input	
33	GPIO33	3.3 V-out-PNP output /3.3 V-int input	

Figure 2-3 Functional interface group

B. Type C interface: used to communicate with PC and update firmware.

C. Grove interface 4: Figure 2-4 shows the definition of Grove interface 4

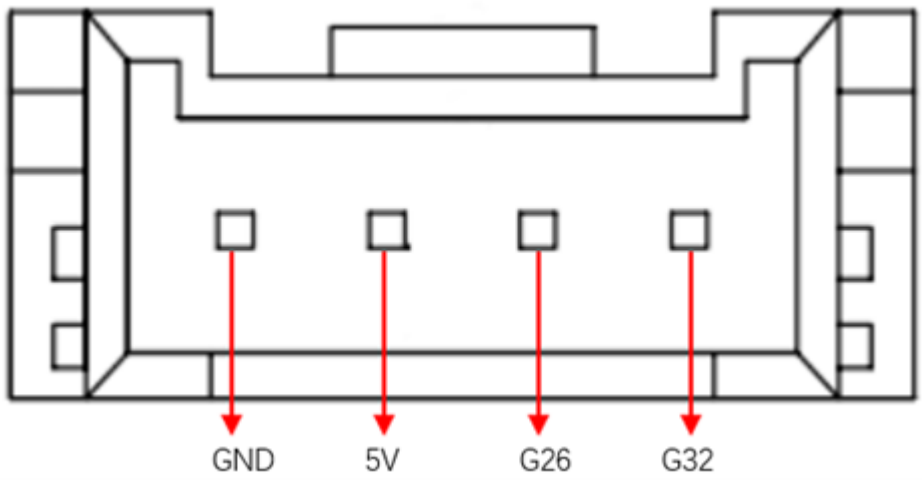


Figure 2-4 Grove port 4

D. Steering gear interface: used for expanding the end of the gripper, currently supporting the use of adaptive gripper.

E. Atom: For 5X5 RGB LED (G27) display and button function (G39)

The First-Time Use

1 Unboxing and the Working Environment

> **Notice:** Please check whether there is any damage or lose at the first place after receiving the package. If any, contact the logistics company and the supplier in time. The picture shown below are for illustration only and actual products may vary due to product enhancement.

Table 2-2 myBuddy Robot Arm [Standard Set]

myBuddy **[Standard Set]**	**-myBuddy 280 Raspberry Pi** **-Product Manual** **-Power Supply** **-Flat Pedestal & G-Shaped Clamp** **-Jumper** **-M4*35, Stick-Shaped Hexagon Socket , Full-Thread Screw, Stainless Steel Screw** **-Hexagon Wrench**

Install the robot system in an environment that meets the conditions described in the table so as to ensure the best performance and safety.

Table 2-3 Working Environment and Conditions

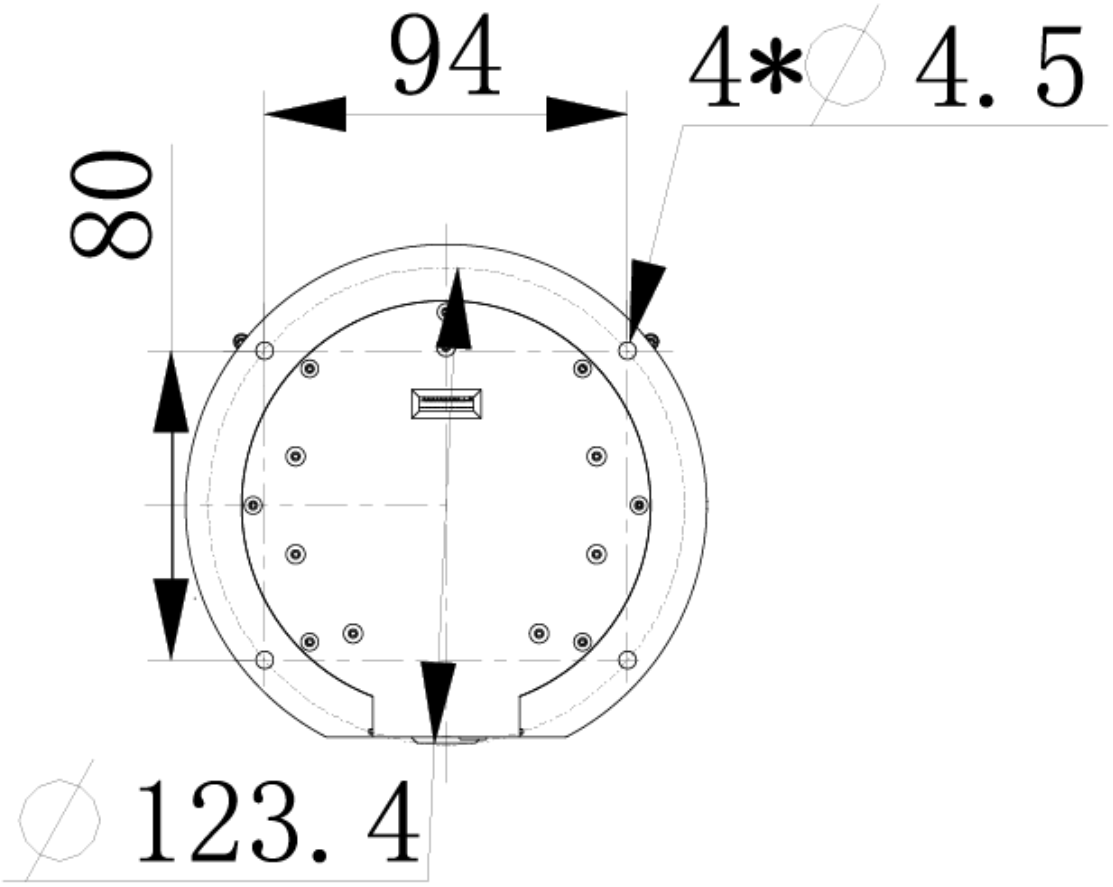
Working Environment	Condition
Temperature	0°C~45°C
Relative Humidity	20%~70%
Space Requirements	Indoor
Other Requirements	-Avoid sunlight exposure. -Avoid dust, oil fume, salt, iron filings, water, etc. -Avoid flammable and corrosive liquids and gases. -Avoid shock and vibration, etc. -Avoid strong electromagnetic interference sources.

2 Installation Requirements

myBuddy weights 2.75kg. Given that the center of gravity changes along with the movement of the robot , it is required that the robot be fixed on a solid ground during utilization. (A fixed pedestal or mobile pedestal are both acceptable.)

Size of Interface

The fixing holes on the pedestal serve to fix robot and connect with other pedestals.



Please make sure that there are corresponding threaded holes on the pedestal.

Before the official installation, please confirm:

- The working environment meets the requirements listed in Section **15.3.1**.
- The installation position is no smaller than the working space of the robot, and there is enough space for installation, use, maintenance and repair.
- Put the robot in a suitable position.
- Installation related tools are ready, such as screws, wrenches, etc.

After confirming the above requeirements, please move the robot to the mounting table, adjust the robot position, and align the fixing holes with the holes on the base mounting table. And then twist the screws tightly.

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

3 Power on the Robot

Before operation, please make sure that you have read **Chapter 1 Safety Instructions**. At the same time, connect the power adapter to the robot, and fix on the table. Refer to Figure 3-1 for connection.

Figure 3-1 Location of the power connector

Figure 3-2 Front of the robot

myBuddy must be powered by an external power for sufficient power:

Rated voltage: 24V

Rated current: 9.2A

Plug: DC 5.5mm x 2.1

4 Basic function

Our company provides various function tutorial videos of myBuddy. Please click here or scan the QR code below for watching.

mybuddy 280 Teaching Series

5 Use of the End Tools

5.1 Sucking Pump

Details: 4 Sucking Pump

Suction pump installation video to be added...

5.2 Adaptive Gripper

Details: 5 Adaptive Gripper

Adaptive gripper installation video to be added...

Unboxing Video

Common Problem

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

How To Ask Questions Gracefully 1 How To Ask Questions Gracefully

Drive Related 2 Drive Related

Software Problem 3 Software Problem

Hardware Problem 4 Hardware Problem

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

How To Ask Questions Gracefully

1 When asking questions in various places, you will find several phenomena:

- No answer after asking question.
- It took a long time for the question to be answered.
- The other party always despise me and do nothing.

2 Before asking questions, make sure you have studied chapters three or four.

Chapters **3 and 4** of this document are the basis for using the **my series**. No matter whether you have development experience or not, be sure to read and operate from front to back.

Many problems will be solved during this process. Do not ask questions in QQ groups, forums, issues, or emails at the very beginning. Many problems explained in the document at the beginning may not be answered in a timely by the community. To save everyone's time, and for a better community environment, everyone can grow together better, please understand each other.

3 When asking questions, try to do the following, which will greatly increase the chances of a quick resolution:

To figure out what's going on and what I did, including:

- What effect and what function do I want to achieve?
- In order to achieve this effect, how do I do it and what is the detailed process?
- In the process of implementation, what error occurred and what is the phenomenon (for example, what is the error reported, what is the **complete** error content?)
- Have I read the error message carefully, and is there any indication of the cause and solution of the error in the error message?
- Based on these error messages, think carefully, can I solve the problem?
- Can I find a solution to the problem by searching the documentation, issues, and using a search engine?

4 If you really can't solve the problem yourself, you need to ask someone for help, and you need to consider:

- Who to ask, where to ask, and who has a better chance of answering my question? And how about real-time?
- What data and phenomena should I give him so that he is willing to help me solve the problem quickly?
    - Provide my purpose (to let the answerer know what you are doing).
    - Provide the complete implementation process and the phenomena that occur in the process (for the answerer to follow your process to do it again, that is, the problem recurs).
    - Give the wrong place, indicate where the phenomenon or result is different from what you expected! (Let the answerer know, where did not meet expectations).
    - For the error information that appears, it needs to be complete, as many screenshots as possible, more logs, don't be stingy to take a small picture, or give a part of the log (because the answerer may not have

done this for a long time, they forgot some details, and they need to rely on screenshots and complete logs to quickly recall.And according to the detailed logs, they can quickly locate where the problem is).

- How to ask questions with a more sincere attitude, even if I don't understand anything, everyone is willing to answer.

5 Question Template

Try to ask questions as elegantly as possible, without adding redundant modal particles, complaining vocabulary, considering every word and punctuation, thinking about the problem from the perspective of the answerer, and how to let the answerer help me solve the problem quickly. Too few words will make it difficult to describe the question, and too many words will make the answerer impatient.

6 Title

No matter where the question is asked (including `QQ group`), prepare a title of about `30` words for your question, clarifying the central idea of the question, including:

- It is necessary to clearly distinguish the type of problem, whether it is a question to ask, a bug submission, or an experience sharing and so on. Let answerer instantly locate what you want to do on a screen full of text.
- One sentence to clarify the core of the problem, such as `run the camera sample program, report an error reset fail, it may be a hardware problem.`

So the title after synthesis can be like this:

- `[Mycobot question] Running the camera sample program, the error reset fail is reported. Could it be a hardware problem?`

Try to **don't** appear in such a title:

- `Why is my board not working again?`
- `Why is my code not working?`
- `Why is my screen black?`
- `[Mycobot question] I received the development board, why is the development board screen red and has a small line of text?`
- `I ran a program and something went wrong.`

You can ask this:

- `[Mycobot question] My board can't start after I connected the power supply in reverse, how can I tell where the board is burnt, and if so, how can I save it?`

7 Content

First stand on the answerer's point of view, if asked a question:

- First of all, I need to know what the other party wants to do and what the goal is to achieve.
- In order to achieve this goal, what steps did he refer to?
- In fact, what specific steps were used, and then at which step the problem occurred, so that I can try to reproduce the phenomenon according to his steps. If this problem seems to be difficult to solve and there are no steps to reproduce it, it may take a lot of time to reproduce, so let's put it aside and solve other problems first.
- What was the specific problem that arose, and if he only stated the problem, how would I know what was wrong with him, maybe a physical discomfort? So this is very important, I need to ask him to explain the phenomenon of the problem and indicate what is different from the expected, otherwise I have to guess what is the difference between the comparison and the expected, and the time to solve the problem has increased.

- If there is a problem, I may need his log file, so that I can analyze the source code according to the log, otherwise it may be difficult to solve the problem, then this problem can be looked at later.

In summary, you can ask the following questions:

- Elaborate on your goals, what you want to do, and what the phenomenon should look like.
- Is there any documentation, code, or tutorial I refer to?
- How to reproduce the error: how to do it in detail, write each step in detail until the problem occurs.
- Elaborate on what happened when the error occurred, and how it was different from what was expected, and needed to prove that the problem did occur.
- Attach log files, as well as screenshots, or even videos, logs and screenshots must be complete, not just a small part, the answerer may find some problems you did not notice from your full log and screenshots, this is very important.
- In addition, pay attention to the format when pasting the code. Do not display it in a mess after pasting, and it cannot be seen. Try to copy it and run it directly.
- Finally, you need to thank the community friends who answered the question.

Drive Related

1 About python

Q: send_coords([x,y,z,rx,ry,rz], speed, 1) What do the parameters in this API mean? What do rx, ry, and rz correspond to Euler angles? What is the rotation order of Euler angles? And what is the value range of each parameter?

- A: The parameters in the previous array are the coordinates of the end of the myCobot, speed is the speed, and the last parameter is the motion mode. rx, ry, and rz should correspond to rpy, that is, corresponding to roll, pitch, and yaw respectively. The order of Euler angles is zyx, and zyx is its own coordinate. The value range of x, y, z is -300~300.00 (the value range is undefined, if the range is exceeded, the inverse kinematics no solution prompt will be returned), and the value range of rx, ry, and rz is -180~180.

Q: Are sample tutorials for the python API provided?

- A: Yes, there is test code in the test folder of github, which can be executed with the terminal. https://github.com/elephantrobotics/pymycobot/tree/main/demo

Q: How does the python drag teaching demo of mycobot280-Pi work?

- A: Run in the terminal, please enter 1000000 baud rate.

Q: Mycobot280-Pi uses python zero to calibrate the demo program, why is there an error?

- A: The atom firmware is not burned, please burn the atom firmware before running the program.

Q: Is the python API of different versions of myCobots the same?

- A: The API is the same.

2 About ROS

Q: How do a microcontroller-based myCobot and a microprocessor-based myCobot run ROS?

- A: The use of ROS for a microcontroller-based myCobot is currently on Ubuntu, and you can also develop your own ROS. A microprocessor-based myCobot have its own ROS environment and can be used directly.

Q: Can a microprocessor-based myCobot connect to a PC to use ROS and moveit?

- A: The current open source data is not controlled by direct communication. It can be implemented by modifying the existing node files through ros + socket.

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

- A: It is available on our github. https://github.com/elephantrobotics/mycobot_ros

Q: Can myAGV create maps remotely?

- A: You can create maps remotely. We are the control interface of open source ROS.

Q: Why is the error permission denied: '/dev/ttyUSB0' reported when using ROS to start the rviz model file?

- A: It is because the serial port permission is not given. You should type sudo chmod 777 port name in the terminal.

Q: Why is the error *init*() takes exactly 2 arguments (3 given) reported when running the slider control and model follow commands of ROS?

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

Q: Q: When using ROS, why is the myCobot angle inconsistent with the model angle after opening the rviz model?

- A: It is very likely that the zero position of the myCobot is not calibrated, and the zero position of the myCobot needs to be calibrated.

Software Problem

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

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

1 About mystudio

Q: What is mystudio?

- A: It is the firmware burner.

Q: Why can't the device run normally 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 the "myCobot firmware burner" to select the ATOM terminal - select the serial port - select the ATOMMAIN firmware for ATOM terminal for burning.

Q: Can the drag teaching in the minirobot firmware control the gripper?

- A: It is achievable.

Q: Why can't drag teaching after the minirobot firmware is burned?

- A: First, check whether the M5Stack-basic firmware and atom firmware have been burned, whether the burned firmware corresponds to the requirements to be realized, and whether the burned firmware is the latest version. The bottom M5Stack-basic burns the minirobot, and the top atom burns the atommain.

Q: What should I do if myCobot's serial port is not recognized on mystudio?

- A: If your computer equipment does not prompt for the connected myCobot, please install the serial port driver first.

Q: Can the track recorded by drag teaching be saved to the card?

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

2 About Roboflow

Q: Can robotstudio software be used for programming?

- A: Our own industrial programming software roboflow can be used. RobotStudio is owned by ABB.

Q: What is the reason for the Quickmove of the roboflow software beyond the limit?

- A: It may be that a joint or multiple joints exceed the limit.

Q: How does roboflow load the already written program?

- A: After logging in, select program robot and click load program. Directly clicking run program cannot be used, only pro600 can.

Q: When the pro600 uses roboflow, the log shows that 456 joints are stopped. Is this normal?

- A: This is normal.

3 About mycobot phone controller

Q: What version of firmware should myCobot phone controller app be programmed with?

- A: You need to burn the atom firmware version atommain 2.5 in mystudio.

4 About myblockly

Q: Why does a pop-up box always appear when myblockly is running?

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

5 About ROS1

Q：When a terminal switches to `~/catkin_ws/src` and uses git to install and update `mycobot_ros`, it appears that the target path "mycobot_ros" already exists. Why?

- A：A `mycobot_ros` package already exists in `~/catkin_ws/src`, so you need to delete it beforehand and run git again.

Q: When rosrun is running, a terminal error is reported saying `counld not open port /dev/ttyUSB0: Permission: '/dev/ttyUSB0'`. Why?

- A: The serial port permission is insufficient. Enter `sudo chmod 777 /dev/ttyUSB0` to grant the permission.

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

- A：Since the vscode terminal cannot be loaded into the ros environment, it needs to run on the system terminal.

Q: `Unable to register with master node [http://localhost:11311]: Unable to register with master node. master may not be running yet. Will he keep trying` ?

- A: Before running the ros program, start the ros node and enter `roscore` on the terminal.

Q: When rosrun runs, the terminal error shows `counld not open port /dev/ttyUSB0: No such file or directory: '/dev/ttyUSB1'`. Why?

- A: The serial port is incorrect. Check the actual serial port of the manipulator. To view the value, run `ls /dev/tty*`.

Q: Failure in Ubuntu18.04 `catkin_make` building code, terminal prompt `Project 'cv_bridge specifies'/usr/include/opencv' as an include dir, which is not found`. Wait for an error message

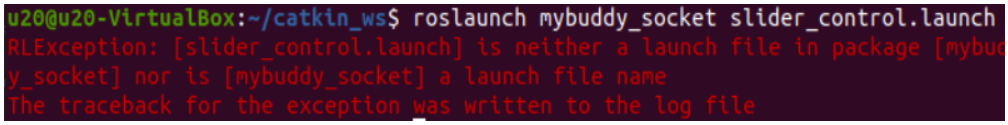
- A: The opencv path in the configuration file is inconsistent with the actual path in the system. Need to use sudo to modify the configuration file (path is `/opt/ros/melodic/share/cv_bridge/cmake/cv_bridgeConfig cmake`), the system actual opencv path in the `/usr/include/` directory.

Q: I cloned the mycobot_ros package and ran the rosrun program directly. package 'mycobot_280' not found or can't find the file?

A: The newly cloned mycobot_ros needs to build code to compile to the ros environment. The input terminal

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

Q: After compiling, why does the following error occur when a new terminal runs launch command?



- A1: The system has not added ros environment variable, so every time a new terminal is started, the source is required:

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

- A2: ros environment variable is added to the system. There is no need to execute source after starting a new terminal:

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

- A3: The file name in the instruction may not be the same as the file name in the mycobot_ros package. Please carefully check whether the instruction is incorrect.

Questions about the Structure

Q1: What are the maximum angle of Joint 1 and Joint 5 of myCobot?

A1: The maximum angle of Joint 1 and Joint 5 are 160° both clockwise and anticlockwise. Remember to rotate the joints gently. Do not force a rotation when the joint reach its maximum angle.

Q2: What controls the six steering engine?

A2: They are controlled by Atom at the top of the robot.

Q3: What functions Atom performs?

A3: It serves to control the robot by algorithm, such as forward, inverse kinematics and coordinate switching. Atom is temporarily not open-sourced.

Questions about the Communication

Q1: Why the screen has no display although the robotic arm has been connected with HDMI cable? Does it need to download a port driver?

A1: Check whether the connection is correct and the power has been turned on. Try to use another interface and plunge into the interface stably. There is no need to download a port driver.

Q2: What are the versions of communication interface supported by different robotic arms?

A2: The robotic arms of micro-processor supports TCP communication, and the robotic arms of micro-controller supports USB-port communication.

Q3: How much is the frequency of the communication of robotic arm?

A3: 10-20Hz.

Questions about Parameters

Q1: What's the unit of measurement of the speed of robotic arm?

A2: 180°/s.

Solutions to Hardware Problems

1. How to deal with the shaking of robotic arms?

   Step 1: Burn the latest version of ATOM via myStudio.

   Step 2: Upgrade pymycobot. Click on Win+R, and type cmd to enter in the terminal. Type pip install pymycobot --upgrade --user and then click on Enter.

   Step 3: Go to GitHub to download pid_read_write.py. Reset each parameter of the steering engine according to the prompt message. And then, operate the system again.

   Notice: The parameters of each version of robot are different. Set the parameters according to the data given in the pictures below.

```
25  # 参数对应地址 corresponding parameter
26  data_id = [21, 22, 23, 24, 26, 27]
27  # 修改后的参数 reset parameters
28  data    = [32, 8, 0, 0, 3, 3]
29
```

Parameter	Connotation	myPalletizer 260			
		J1/J3	J1/J3	J2	J4
21	location loop P ratio coefficient	25	32	32	32
22	location loop D differentia coefficient	25	32	32	32
23	location loop I integral coefficient	0	0	0	0
24	minimun starting force	0	16	3	0
26	clockwise insensitive area	3	1	1	0
27	anticlockwise insensitive area	3	1	1	0

Parameter	Connotation	mechArm 270			
		J1/J3	J2	J4/J6	J5
21	location loop P ratio coefficient	32	32	25	25
22	location loop D differentia coefficient	8	8	25	25
23	location loop I integral coefficient	0	0	1	1
24	minimun starting force	0	0	0	0
26	clockwise insensitive area	3	1	3	3
27	anticlockwise insensitive area	3	1	3	3

Parameter	Connotation	myCobot 280		
		J1-J3	J4	J5-J6
21	location loop P ratio coefficient	32	25	25
22	location loop D differentia coefficient	8	25	25
23	location loop I integral coefficient	0	1	1
24	minimun starting force	0	0	0
26	clockwise insensitive area	3	3	3
27	anticlockwise insensitive area	3	3	3

Parameter	Connotation	myCobot pro 320		
		J1、J2、J3	J2	J5-J6
21	location loop P ratio coefficient	32	32	25
22	location loop D differentia coefficient	8	8	25
23	location loop I integral coefficient	0	0	1
24	minimun starting force	0	0	0
26	clockwise insensitive area	3	3	3
27	anticlockwise insensitive area	3	3	3

Questions about Gripper & End

Q1: Can the gripper close completely?

A1: Due to the thickness of the gripper, it cannot close completely. You resort to a shim to make it completely closed.

Q2: What is the kind of communication of adaptive gripper?

A2: It is TTL communication.

Q3: What is the communication of the end of myCobot 320?

A3: 485 communication interface.

Q4: How to fix a camera at the end of a robotic arm?

A4: A flange is required to fix a camera.

Other Questions

Q1: Why the electric engine powers off during usage?

A1: Because the engine is too hot to be used. Please wait for a few minutes and reuse again.

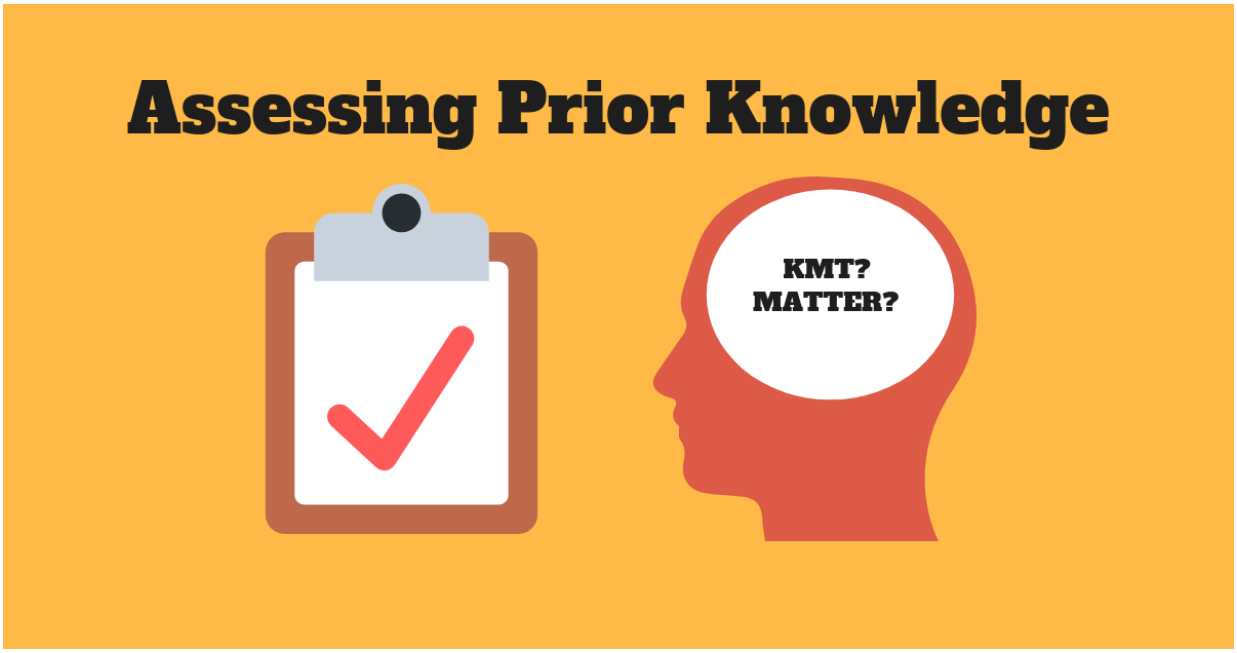
Q2: Does the robotic arms support the development based on Android?

A2: We doesn't support Android development. If you want to develop it by yourself, we can provide you with port protocol.

Q3:What is the function of the USB carried by the PI version?

A3: It serves to charge the robotic arm.

Background knowledge



It is necessary to have an in-depth understanding of myCobot from the aspects of hardware, software and robot algorithm. At the beginning, we can understand the progress of robots by understanding the history of robots.

- **Robot**
- **Electric**
- **Motors**

Robot

This chapter is excerpted from Introduction to robotics mechanics and control by J.Craig. If you want to read more about it, please buy it online.

1 Background

The history of industrial automation is characterized by the rapid renewal of technological means. The renewal of such automation technology is closely related to the world economy, whether as an inducement or a result of the development of the world economy. Industrial robot in the 1960s is undoubtedly a unique equipment, it will be combined with the computer aided design (CAD) system, computer aided manufacturing (CAM) system application, this is the modern manufacturing automation of the latest development trend. These technologies are leading the transition to a new field of industrial automation.

Manipulator is one of the most important types of industrial robots. Whether the manipulator can be called an industrial robot is controversial. The equipment shown here is generally considered to belong to the category of industrial robots, while CNC (NC) grinders are usually outside this category.

Generally speaking, the research of manipulator mechanism and control theory is not a new science, it is only a synthesis of traditional theory.Mechanical engineering theory provides a methodology for the study of manipulator in static and dynamic environments.The mathematical method is used to describe the spatial motion of the manipulator and its characteristics.Control theory provides various design methods and evaluation algorithms for the realization of desired motion or force.Electrical engineering technology can be used in sensor and industrial robot interface design;Computer technology provides the programming platform needed to perform the desired task.

2 Basic Concepts

Mechanical Arm

Mechanical Arm can also be called industrial robot, cooperative robot, manipulator arm, bionic arm, series robot, etc.

Position & Pose

In robot research, we usually study the position of objects in a three-dimensional space. The objects referred to here include not only the lever, parts and grasping tools of the manipulator, but also other objects in the workspace of the manipulator. Usually these objects can be described by two very important properties: position and pose. Naturally, we will first study how to express and calculate these parameters mathematically.

In order to describe the position and posture of a space object, we usually place the object firmly in a space coordinate system, that is, the reference frame, and then we study the position and posture of the space object in this reference coordinate system.

Direct Kinematics

Kinematics is the study of the motion of objects without regard to the forces causing such motion. In kinematics, we study higher-order derivatives of position, velocity, acceleration, and position variables with respect to time or other variables. Thus, the research object of manipulator kinematics is all the geometric and temporal characteristics of motion. Almost all manipulators are composed of rigid links, adjacent links connected by joints that allow for relative motion. If it's a revolute joint, its displacement is called the joint Angle. These joints are usually fitted with position sensors to measure the relative position of adjacent bars. If you have a revolute joint, this displacement is called the joint Angle. Some manipulators have sliding (or moving) joints, so the displacement of two adjacent links is a linear motion, which sometimes called the joint offset.

A typical problem in the study of manipulator kinematics is manipulator forward kinematics. It is a static geometry problem to calculate the position and posture of its end-effector of the manipulator. Specifically, given a set of values for joint angles, the forward kinematics problem is to compute the position and posture of the tool coordinate system relative to the base coordinate system. In general, we refer to this process as the representation of manipulator position from joint space description to Cartesian space description.

Degree of Freedom (DOF)

The number of DOF is the number of manipulator position variables in the coordinate system (reference frame) with figure 1-5 in the manipulator, which determines the position of all components in the mechanism. DOF is universal to all mechanisms. For example, a four-bar mechanism has only one DOF (although it has three movable rods). For a typical industrial robot, the number of joints is equal to the number of DOF because manipulator arms are mostly open motion chains and each joint position is defined by an independent variable.

End-effector

End-effector is installed at the free end of the manipulator. Depending on different applications of robot, it may be a fixture, a welding torch, an electromagnet, or some other devices. We usually describe the position of the manipulator in terms of a tool coordinate system attached to its end-effector, and the corresponding tool coordinate system is the base coordinate system connected to the fixed base of the manipulator.

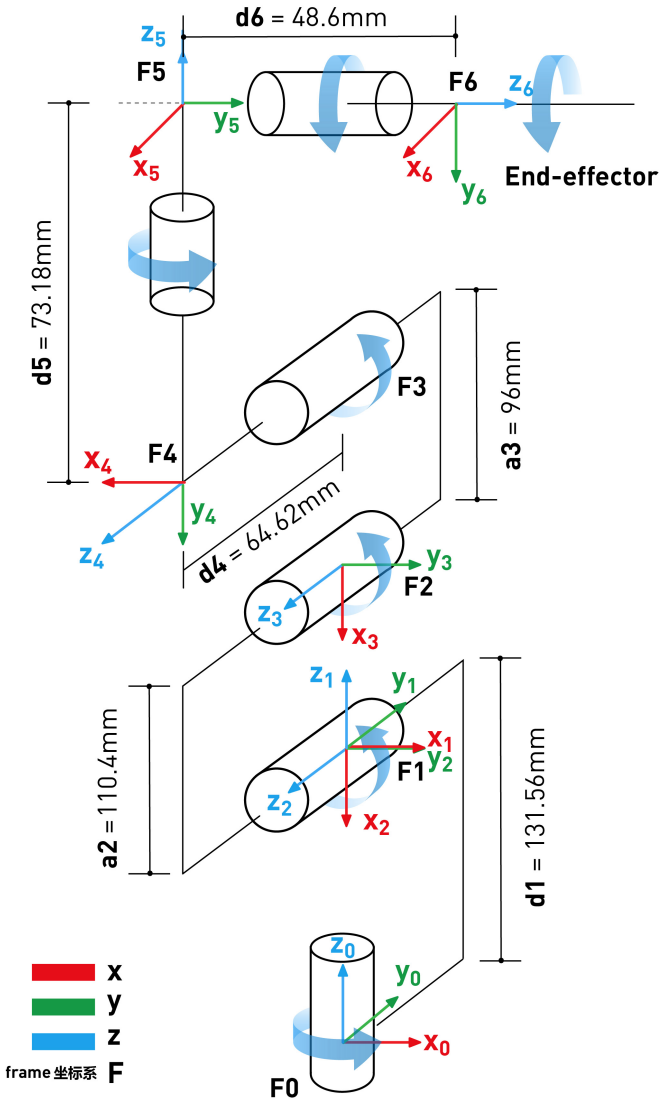
Inverse Kinematic

Given the position and posture of the end-effector of the manipulator, calculating all joint angles that can reach the given position and attitude.

3 Space description

Position

Once the coordinate system is established, we can locate any point in the world coordinate system with a position vector of 3x1. Since many coordinates areoften defined in the world coordinate system, a piece of information must beattached to the position vector indicating which coordinate system is defined.In this book, the position vector has a leading superscript to indicate thecoordinate system to which it refers.



Posture

We find that it is often necessary not only to represent points in space, but also to describe the posture of objects in space. For example, if the vector "P" in Figure 2-2 directly determines a point between the fingers of the manipulator hand, the position of the hand can only be fully determined if the posture of the hand is known. Assuming that the manipulator has a sufficient number of joints, the manipulator can be in any position and the position of the points between the fingers remains constant. To describe the posture of an object, we will fix a coordinate system on the object and give the representation of this coordinate system with respect to the reference system. In Figure 2-2, the coordinate system {B} is known to be fixed to the object in some way. The description in {B} relative to {A} is sufficient to indicate the attitude of object (A).

Coordinate System

A reference frame can be described in terms of the relation of one coordinate system with respect to another. The reference frame includes the concepts of position and posture, which is considered to be a combination of these two concepts in most cases. The position can be represented by a frame of reference in which the rotation matrix is the identity matrix and the position vector in this frame of reference determines the position of the described point. Similarly, if the position vector in the frame of reference is the zero vector, then it represents the posture.

4 DH Parameters

Definition

For rotational joint n, set 0=0.0, the direction of X axis is the same as that of X, axis, the origin position of coordinate system (n) is selected to satisfy d.=0.0.For prismatic joint n, the direction of axis 8 is set to meet 0.=0.0. When d.=0.0, the origin of the coordinate system {n) is selected to be located at the intersection of axis XN-1 and joint axis n.

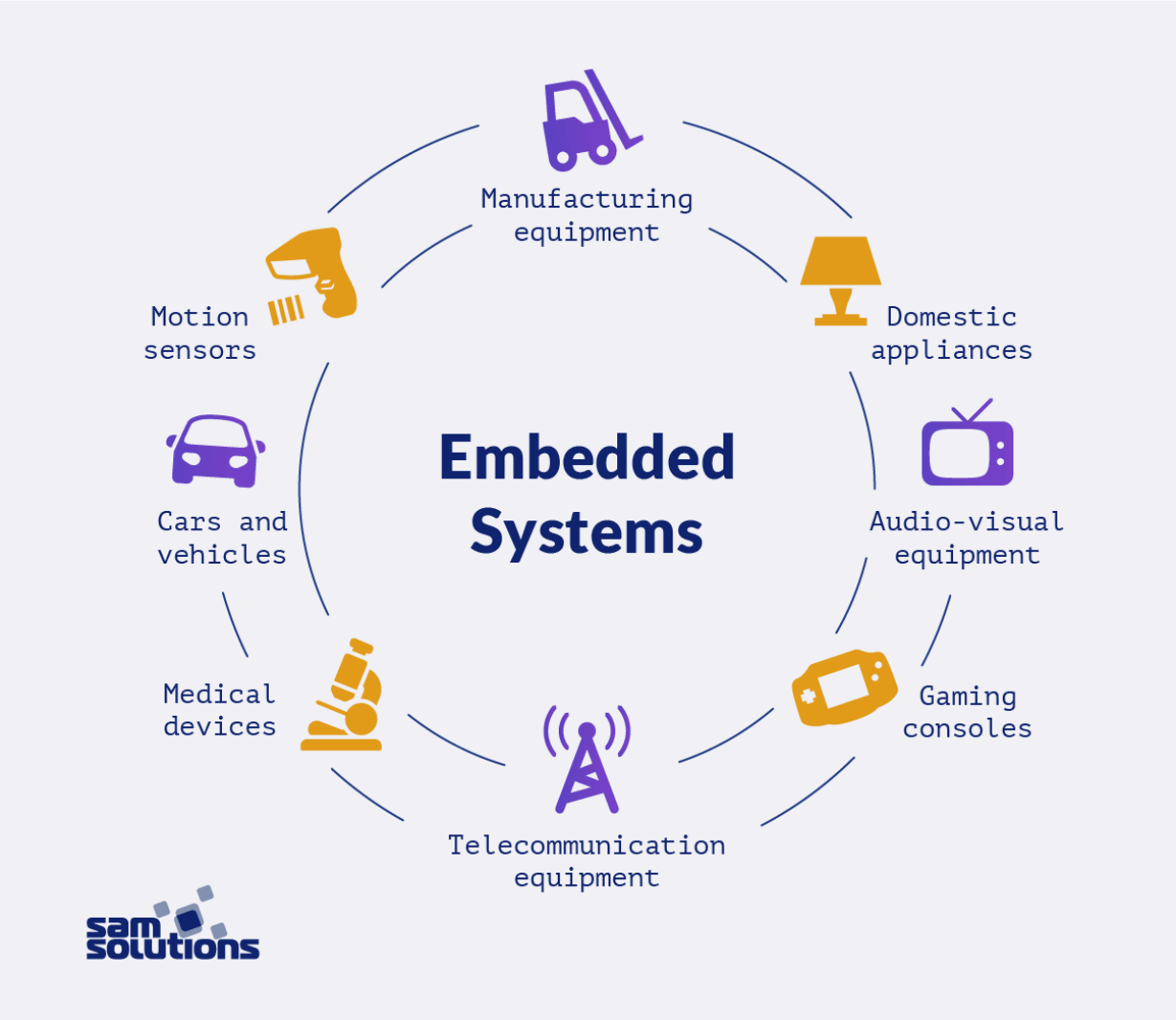
In the link coordinate system, if the link coordinate system is fixedly attached to the link as described above, the link parameters can be defined as follows:

- a_i-1： along x_i-1 : move from the distance of z_i-1 to z_i
- alpha_i-1： aroundx_i-1： rotate from the Angle of z_i-1 to z_i
- d_i： along z_i： move from the distance of x_i-1 to x_i
- theta_i： aroundz_i： rotate from the Angle of x_i-1 tox_i

myCobot DH parameter

Joint	alpha	a	d	theta	offset
1	0	0	131.56	theta_1	0
2	PI/2	0	0	theta_2	-PI/2
3	0	-110.4	0	theta_3	0
4	0	-96	64.62	theta_4	-PI/2
5	PI/2	0	73.18	theta_5	PI/2
6	-PI/2	0	48.6	theta_6	0

Electronics



Introduction

Single-Chip Microcomputer (Microcontrollers/SCM) is a kind of integrated circuit chip, which uses VLSI technology to integrate the central processing unit CPU with data processing capabilities, random access memory RAM, read-only memory ROM, various I/O ports, interrupt systems, and timer /Counter and other functions (may also include display drive circuit, pulse width modulation circuit, analog multiplexer, A/D converter and other circuits) integrated into a silicon chip to form a small and complete microcomputer system, widely used in the field of industrial control. From the 1980s, it developed from the 4-bit, 8-bit microcontroller to the current 300M high-speed microcontroller.

Basic Structure

- **Arithmetic Unit**

Arithmetic Unit is composed of arithmetic logic unit (ALU), accumulator and register. The function of ALU is to perform arithmetic or logical operations on the incoming data. The input comes from two 8-bit data sources, one from the accumulator and the other from the data register. ALU can perform various operations on the data, such as Addition, Subtraction, AND, OR, Comparison, etc., and finally store the results in the accumulator.

The arithmetic machine has two functions:

- Perform various arithmetic operations.
- Perform various logical operations and perform logical tests, such as a zero-value test or a comparison of two values.
- All operations performed by arithmetic unit are directed by the control signal issued by the controller, and an arithmetic operation produces an operation, and a logical operation produces a decision.
- **Controller**

Controller is composed of program counter, instruction register, instruction decoder, timing generator and operation controller, etc. It is the "decision-making body" of issuing commands, namely coordinating and directing the operation of the entire microcomputer system. Its main functions are：

- Extract an instruction from memory and indicates the location of the next instruction in memory.
- Decode and test the instructions,then generate the corresponding operation control signal to facilitate the execution of the specified actions.
- Direct and control the direction of data flow between CPU, memory, and input/output devices.

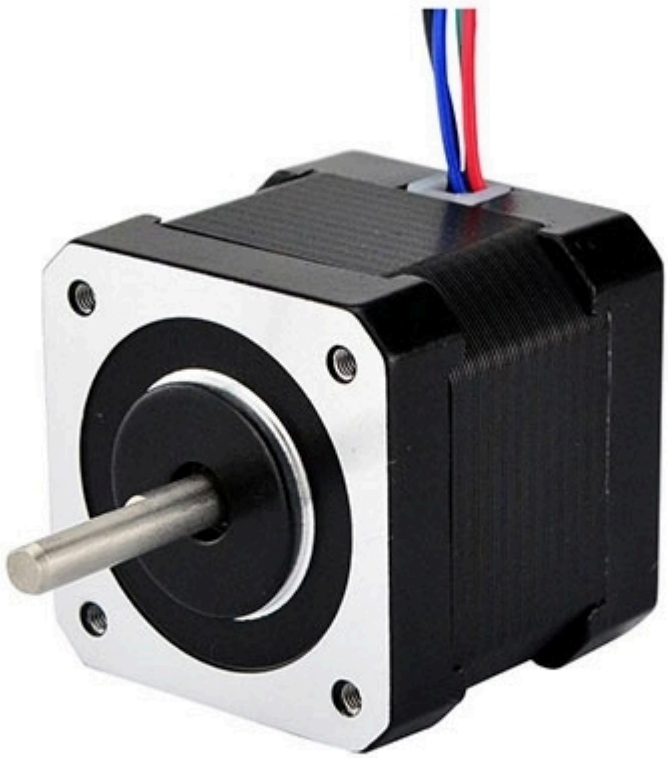
Microprocessor interconnects ALU, counter, register and control parts through an internal bus, and connects the external memory, input and output interface circuit through an external bus. External bus, also known as system bus, is divided into data bus DB, address bus AB and control bus CB. It can be connected with various peripheral devices through the input and output interface circuit.

Motor & Steering Gear

Motor

According to the type of working power supply, it can be divided into:

- 1). DC(Direct Current) motor
- 1.1). Brushless DC motor
    - 1.2). Brushed DC motor
- 1.2.1). Permanent magnet DC motor
- 1.2.1.1). Rare Earth permanent magnet DC motor
- 1.2.1.2). DC Ferrite permanent magnet DC motor
- 1.2.1.3). Aluminium Nickel-cobalt permanent magnet DC motor
- 1.2.2). Electromagnetic DC motor
- 1.2.2.1). DC series motor
- 1.2.2.2). Shunt DC motor
- 1.2.2.3). Separately Excited DC motor
- 2). AC (Alternating Current) motor
- 2.1). Single-phase motor
- 2.2). Three-phase motor



According to the application, it can be divided into:

- **Drive motor:** Electric tools (including drilling, buffing, polishing, grooving, cutting, reaming and other tools) motor, household appliances(including washing machine, electric fan, refrigerator, air conditioner, tape recorder, video recorder, DCD, vacuum cleaner, camera, hair dryer, electric shaver, etc.) motor, and other general small mechanical equipment (including all kinds of small machine tools, small machinery, medical equipment, electronic instruments, etc.) motor
- **Control motor:**
- Stepping motor
- Servo motor

What is Servo Motor

Servo motor (servo motor) refers to the engine that controls the operation of mechanical components in the servo system, and is an auxiliary motor indirect transmission device. The servo motor can control the speed and position accuracy very accurately, and can convert the voltage signal into torque and speed to drive the control object. The rotor speed of the servo motor is controlled by the input signal and can respond quickly. In the automatic control system, it is used as an executive element, and has the characteristics of small electromechanical time constant and high linearity. It can convert the received electrical signal into the motor shaft. angular displacement or angular velocity output. It is divided into two categories: DC and AC servo motors. Its main feature is that when the signal voltage is zero, there is no rotation phenomenon, and the speed decreases uniformly with the increase of torque. Servo Motor refers to the engine that controls the operation of mechanical components in the servo system, it is a kind of indirect speed change device for auxiliary motor.

How Servo Motors Work

1. The servo system is an automatic control system that enables the output controlled quantities such as the position, orientation, and state of the object to follow any changes in the input target (or given value). The servo mainly relies on pulses for positioning. Basically, it can be understood in this way that when the servo motor receives one pulse, it will rotate the angle corresponding to one pulse, thereby realizing the displacement. Because the servo motor itself has the function of sending out pulses, so every time the servo motor Rotating an angle, a corresponding number of pulses will be sent out, which echoes the pulses received by the servo motor, or is called a closed loop. In this way, the system will know how many pulses are sent to the servo motor and how many pulses are received at the same time. , in this way, the rotation of the motor can be precisely controlled, so as to achieve precise positioning, which can reach 0.001mm. DC servo motors are divided into brushed and brushless motors. Brushed motors have low cost, simple structure, large starting torque, wide speed regulation range, easy control, and require maintenance, but maintenance is inconvenient (carbon brush replacement), generates electromagnetic interference, and has environmental requirements. Therefore, it can be used in general industrial and civil occasions that are sensitive to cost. The brushless motor is small in size, light in weight, large in output, fast in response, high in speed, small in inertia, smooth in rotation and stable in torque. The control is complex, and it is easy to realize intelligentization. Its electronic commutation mode is

flexible, and it can be square wave commutation or sine wave commutation. The motor is maintenance-free, has high efficiency, low operating temperature, small electromagnetic radiation, long life, and can be used in various environments.

2. The AC servo motor is also a brushless motor, which is divided into synchronous and asynchronous motors. Synchronous motors are generally used in motion control. It has a large power range and can achieve great power. High inertia, low maximum rotational speed, and decreases rapidly as power increases. Therefore, it is suitable for applications that operate smoothly at low speeds.

3. The rotor inside the servo motor is a permanent magnet. The U/V/W three-phase electricity controlled by the driver forms an electromagnetic field. The rotor rotates under the action of this magnetic field. At the same time, the encoder that comes with the motor feeds back the signal to the driver, and the driver according to the feedback value Compared with the target value, the angle of rotation of the rotor is adjusted. The accuracy of the servo motor is determined by the accuracy (number of lines) of the encoder. The difference between AC servo motor and brushless DC servo motor in function: AC servo is better, because it is controlled by sine wave, and the torque ripple is small. DC servos are trapezoidal waves. But DC servos are simpler and cheaper. ![industrialservo]industrialservo.jpg

Comparison of characteristics of servo motors

DC brushless servo motor features small moment of inertia, low starting voltage and low no-load current; abandoning the contact commutation system, greatly increasing the motor speed, the maximum speed is up to 100 000rpm; the brushless servo motor does not need an encoder when performing servo control It can also realize the control of speed, position, torque, etc.; there is no brush wear, and in addition to high speed, it also has the characteristics of long life, low noise, and no electromagnetic interference. DC brush servo motor features

4. Small size, quick action, quick response, large overload capacity, wide speed regulation range
5. Large torque at low speed, small fluctuation and stable operation
6. Low noise, high efficiency
7. The back-end encoder feedback (optional) constitutes the advantages of DC servo and so on
8. Large transformer range and adjustable frequency

The advantages of servo motors over ordinary motors

1. Accuracy: closed-loop control of position, speed and torque is realized; the problem of stepper motor out-of-step is overcome;
2. Speed: good high-speed performance, generally rated speed can reach 2000 ~ 3000 rpm;
3. Adaptability: strong anti-overload capability, able to withstand loads three times the rated torque, especially suitable for occasions with instantaneous load fluctuations and fast start requirements;
4. Stable: The low-speed operation is stable, and the stepping operation phenomenon similar to the stepping motor will not occur during low-speed operation. Applicable to occasions with high-speed response requirements;
5. Timeliness: The dynamic response time of motor acceleration and deceleration is short, generally within tens of milliseconds;
6. Comfort: heat and noise are significantly reduced. To put it simply: the ordinary motor that is usually seen will rotate for a while due to its own inertia after the power is turned off, and then stop. The servo motor and the stepper motor are to stop when they say they stop, and they can go when they say they are going, and the response is extremely fast. But the stepper motor has out-of-step phenomenon.

Application Scenarios of Servo Motors

There are too many applications for servo motors. As long as there is a power source, and there is a requirement for accuracy, it may generally involve a servo motor. Such as machine tools, printing equipment, packaging equipment, textile equipment, laser processing equipment, robots, automated production lines and other equipment that require relatively high process accuracy, processing efficiency and work reliability.

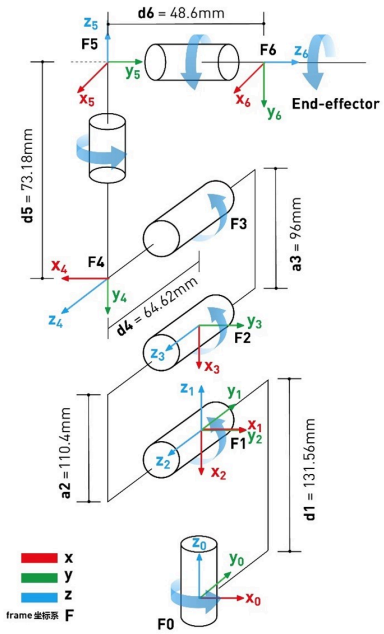
myCobot 280 algorithm

1 Structural Parameters

1.1 DH Parameters of Robotic Arm

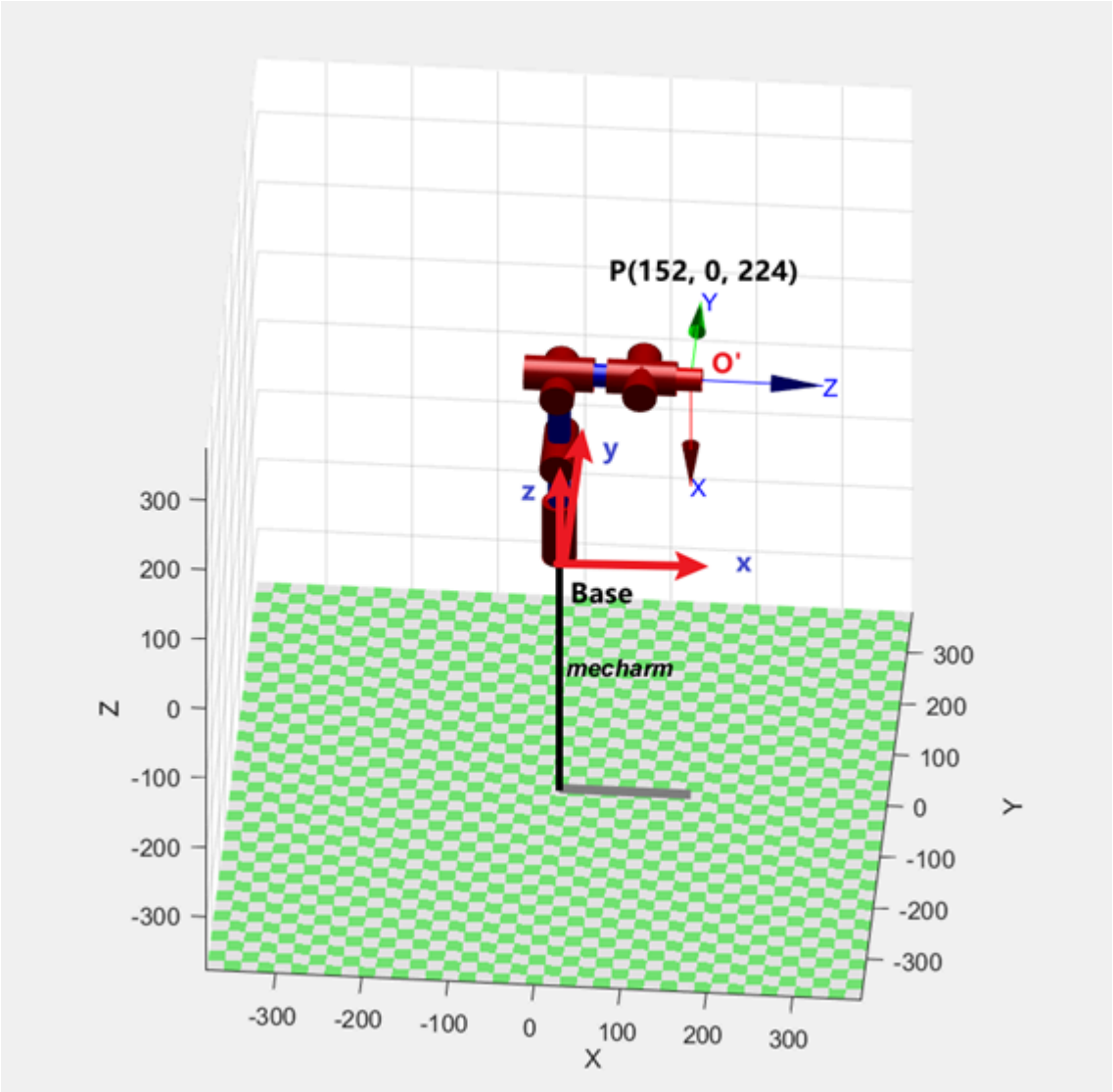
joint	theta	d	a	alpha	offset
1	q1	131.22	0	1.5708	0
2	q2	0	-110.4	0	-1.5708
3	q3	0	-96	0	0
4	q4	63.4	0	1.5708	-1.5708
5	q5	75.05	0	-1.5708	1.5708
6	q6	45.6	0	0	0

1.2 Kinematic Model



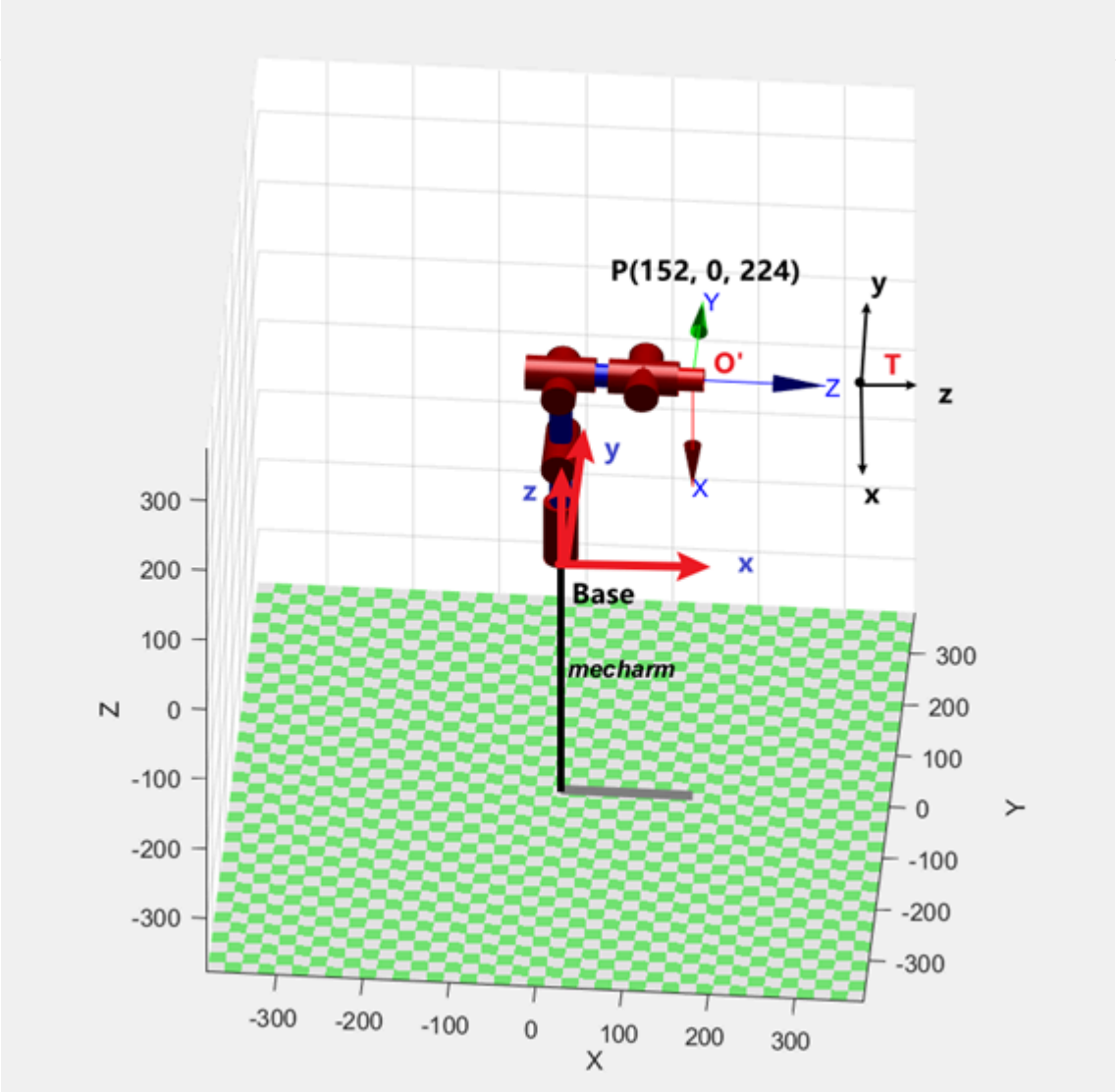
2 Coordinate System Introduction

2.1 Tool Coordinate System



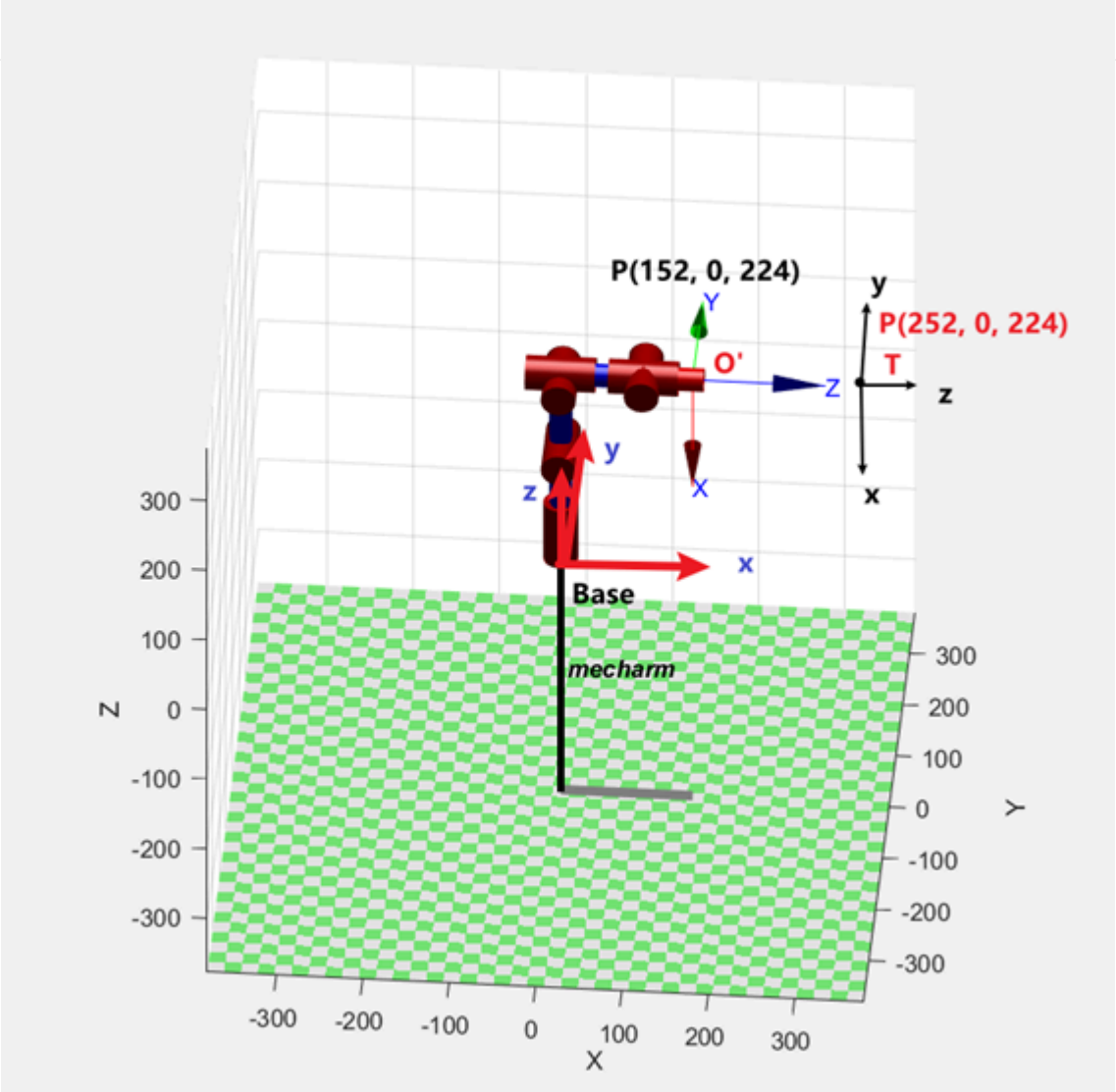
The figure shows the robot model of Mecharm270. Base in the figure represents the base coordinate system of the robot, O' represents the end flange coordinate system, and point P represents the position of the end of the manipulator relative to the base coordinate system (x=152, y=0 , z=224)

Extend a certain pose on the basis of the end flange, and regard the set tool point as the end of the machine:



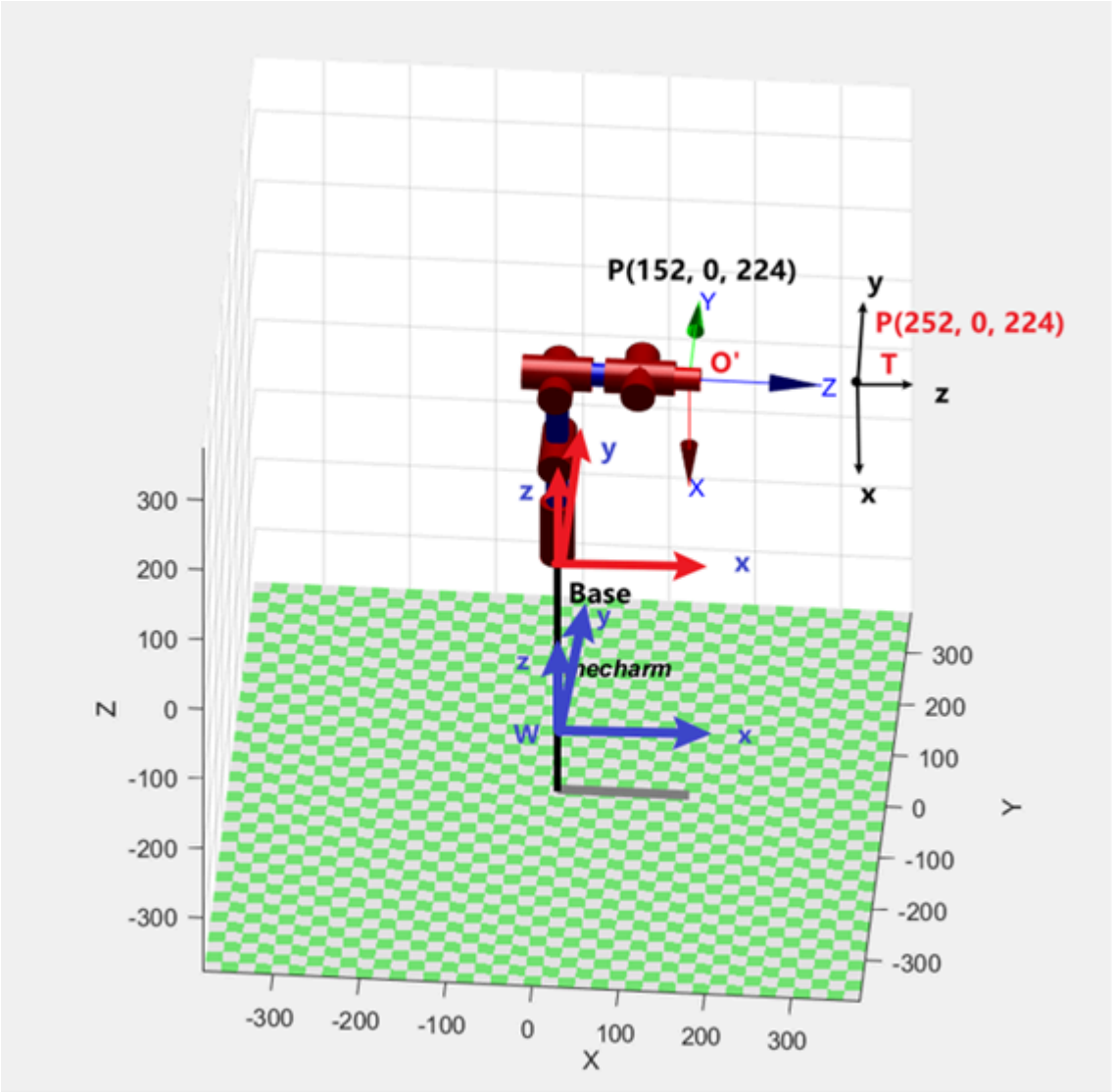
T in the figure is the set tool coordinate system. The posture of this coordinate system is consistent with O', and the relative displacement of the origin has occurred. Use the python function to set the tool coordinate system:

- set_tool_reference([x, y, z, rx, ry, rz]) //Set tool coordinate system
- set_end_type(1) //Set the end coordinate system type as tool
- Assume that the tool coordinate system T is not rotated relative to O' (rx = ry = rz = 0)
- Assume that the origin of the tool coordinate system T is in the coordinate system O' at (x = 0, y = 0, z = 100mm)
- The final tool coordinate system parameter is set_tool_reference(0, 0, 100, 0, 0, 0)



Since the tool coordinate system is set, the end of the robot extends from O' to T at this time, and the coordinates of the end of the machine read at this time become (152+100, 0, 224), and the coordinate posture movement will revolve around the tool point T rotate.

2.2 World Coordinate System



Section 2 introduces that by setting the tool coordinate system, the end coordinate system of the manipulator can be extended to a certain pose. We can also extend a certain pose on the basis of the base coordinate system of the manipulator by setting the world coordinate system. The set world coordinate system will replace the original Base coordinate system and become the new base coordinate system.

W in the figure is the set world coordinate system. The posture of this coordinate system is consistent with Base, and the relative displacement of the origin has occurred. Use the python function to set the world coordinate system:

- set_world_reference([x, y, z, rx, ry, rz]) //Set the world coordinate system
- set_reference_frame(1) //Set the base coordinate system type to the world
- Assuming that the world coordinate system W has not rotated relative to Base(rx = ry = rz = 0)
- Suppose the origin of the world coordinate system W is in the coordinate system Base (x = 0, y = 0, z = -100mm)
- The final world coordinate system parameter is set_world_reference(0, 0, -100, 0, 0, 0)

Since the world coordinate system is set, the origin of the robot extends from Base to W at this time, and the O' coordinate read at this time becomes (152, 0, 224+100).

Application of Basic Functions

1 myStudio

Before using the equipment, you have to **install the driver and update the firmware**.

1.1 Installing the driver

According to the operating system used by him, the user can click the button below to download the zip file of the corresponding **CP210X** or **CP34X** drivers. After decompressing the file, select and install the installation package corresponding to the operating system.

There are two kinds of driver chip versions: **CP210X** (suitable for CP2104 version)/**CP34X** (suitable for CH9102 version) driver zip files. If you are not sure of the USB chip used for your equipment, you can install two drivers at the same time. (During the installation of **CH9102_VCP_SER_MacOS**, an error may be reported; however, the installation has been done, and the error can be ignored.)

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 **Basic** serial port driver **CP210X**

  CP210X

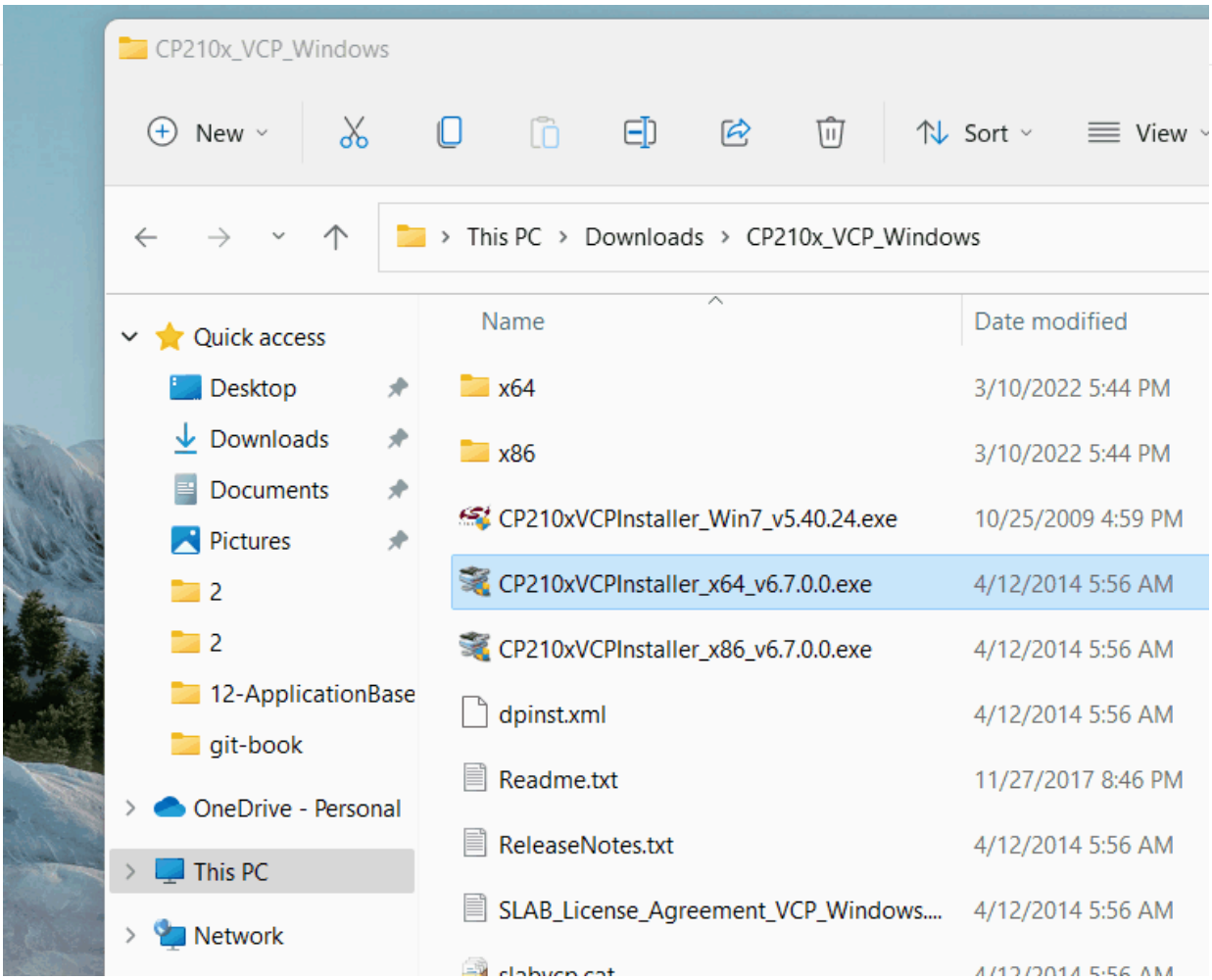
  - **Windows10**
  - **MacOS**
  - **Linux**

  CP34X

  - **Windows10**
  - **MacOS**

- Download the **Atom** serial port driver for the end.

  - **Windows10**



1.2 Updating equipment firmware

Prior to development, the user is required to confirm whether his equipment firmware is the latest version so that he can use the equipment better during subsequent development.

The user can update the firmware through **myStudio**.

2 Factory firmware introduction

2.1 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.

2.2 Robot arm 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.

2.3 Communication forwarding

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, micro-controller devices have three methods of communication: **serial port, Bluetooth, and WIFI**. The user may choose an applicable method of communication for programming.

2.4 Connection detection

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.

3 Use for the first time

After knowing the current function of the firmware, connect and fix the machine by following the steps in this chapter, and start applying the basic functions of the equipment.

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.**

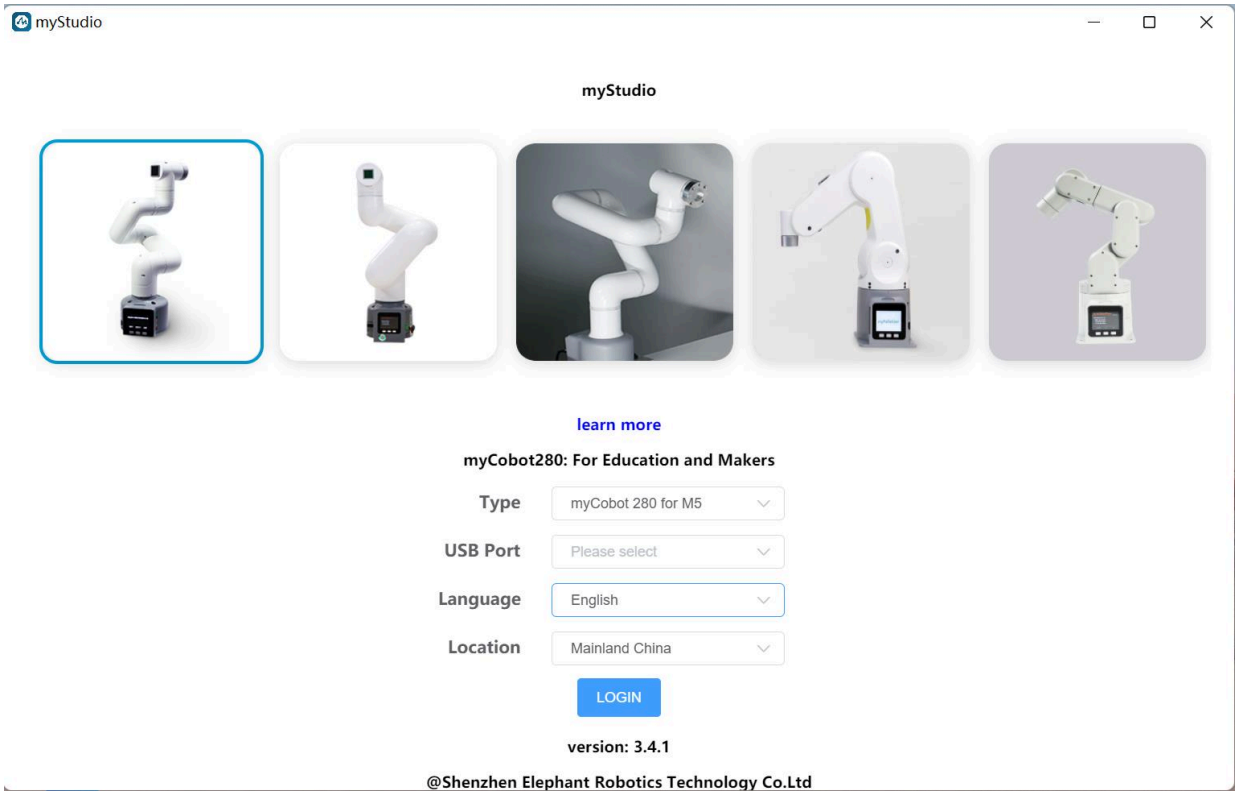
If you have any questions or suggestions about the contents of this manual, please log in the official website of Elephant Robot and submit relevant information：

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.

myStudio



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

4 Matchable Robots

- 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
- myCobot PRO 600
- myBuddy 280

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 280 series

myCobot 280 series have 4 versions: M5、PI、Ardunio and Jetsonnano. 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	v2.1 is recommended for dragging teaching, wifi connection and bluetooth connnection
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before，or not lablled; v5.1 is recommended for robots lablled ER28001202200416 and after
PI	RaspberryPI 4B	ubuntu	v18.04.is recommened
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before，or not lablled; v5.1 is recommended for robots lablled ER28001202200416 and after
Arduino	mega2560	transponder	v1.0 is recommended
	mkrwifi1010	transponder	v1.0 is recommended
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before，or not lablled; v5.1 is recommended for robots lablled ER28001202200416 and after
Jetson nano	JestonNano	ubuntu	v18.04.is recommended
	Atom	atomMain	v4.1 is recommended for robots labelled ER28001202200415 and before，or not lablled; v5.1 is recommended for robots lablled ER28001202200416 and after

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

myPalletizer 260 series

myPalletizer 260 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	Recommended Firmware
M5	M5Stack-Basic	miniRobot	v1.1 is recommended
PI	Atom	atomMain	v1.1 is recommended

1 Environment Building

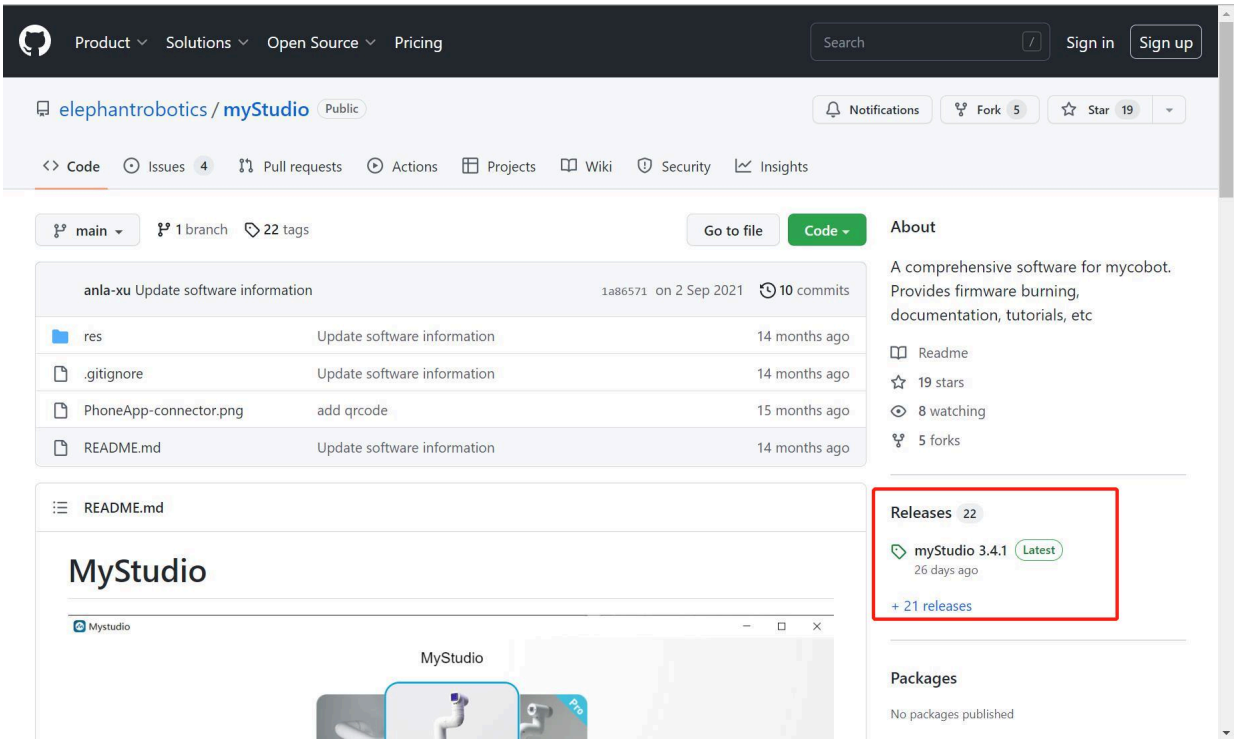
1.1 Download and Installation

> Note：The installation path of myStudio installation cannot have any spaces

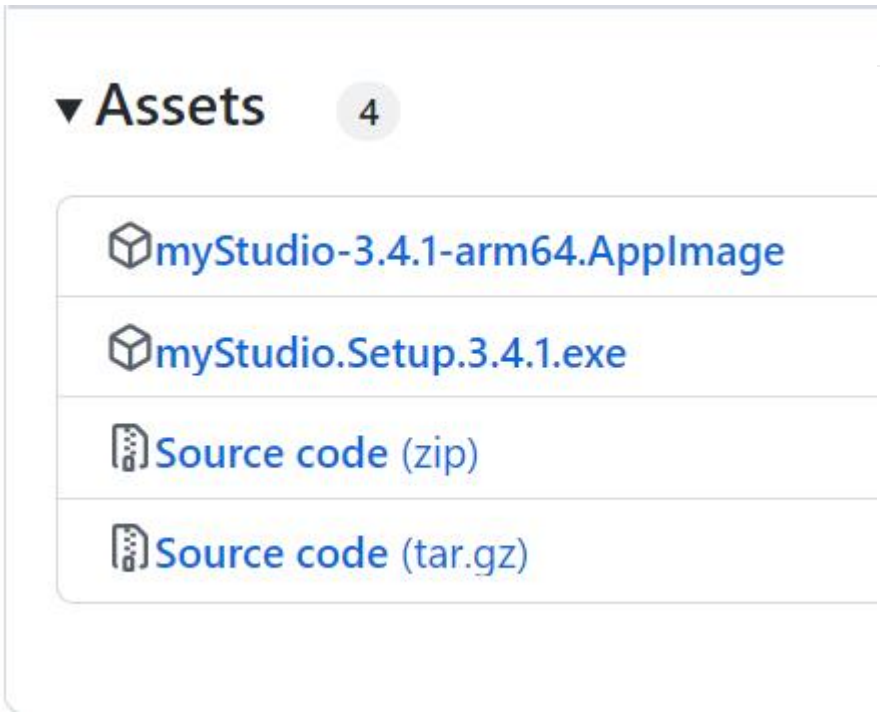
address for downloading:

1. GitHub

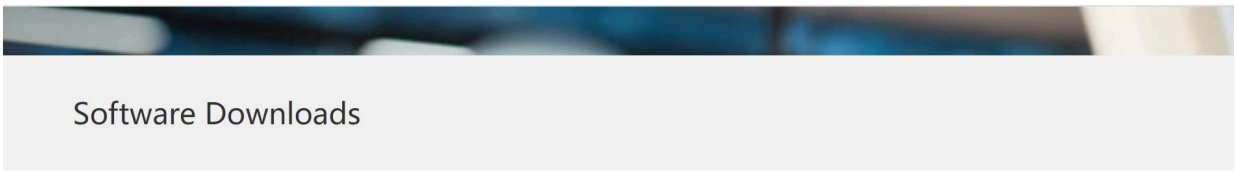
- Click on `myStudio` at the right side and download the version corresponding to your PC. **Do not install myStudio into files with space directory.**



- Different suffixes signifies versions applicable for different systems. -
  - *.tra.xz is applicable to Linux
  - *.dmg is applicable to Mac
  - *.exe is applicable to Window



2. Elephant Robotics Official Website

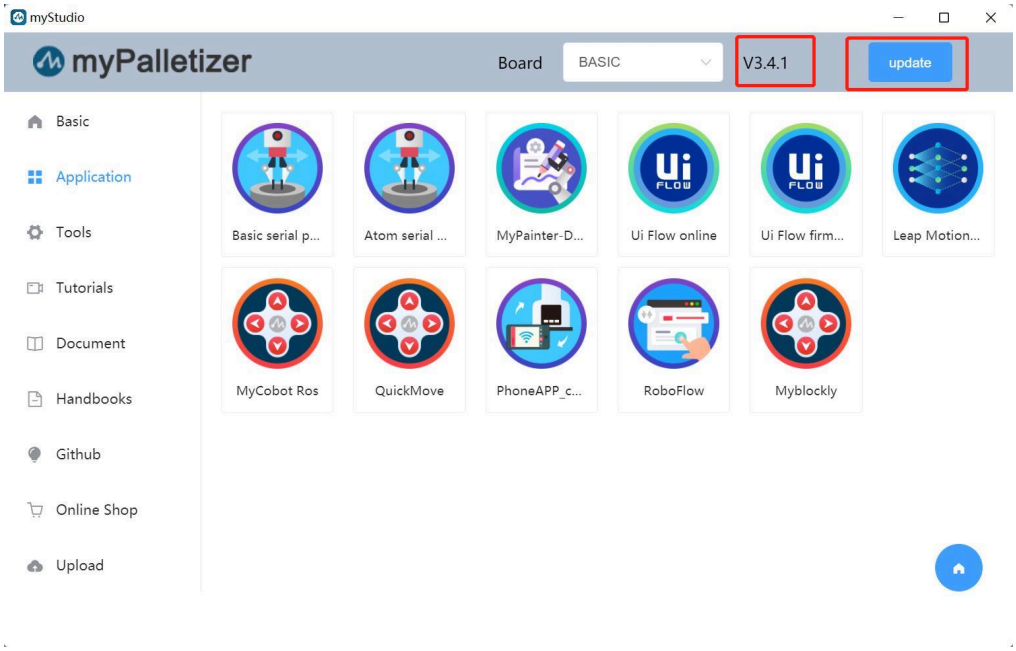


myStudio

myStudio is a one-stop platform for robots of myRobot/myCobot.
The main functions of myStudio are: 1) Update the firmware; 2) Provide video tutorials on how to use the robot; 3) Provide maintenance and repair information (such as video tutorials, Q&A, etc.).

Windows Linux Mac

> **Notice:** Download the latest version. You can log in to check the present version of myStudio and update to the latest.

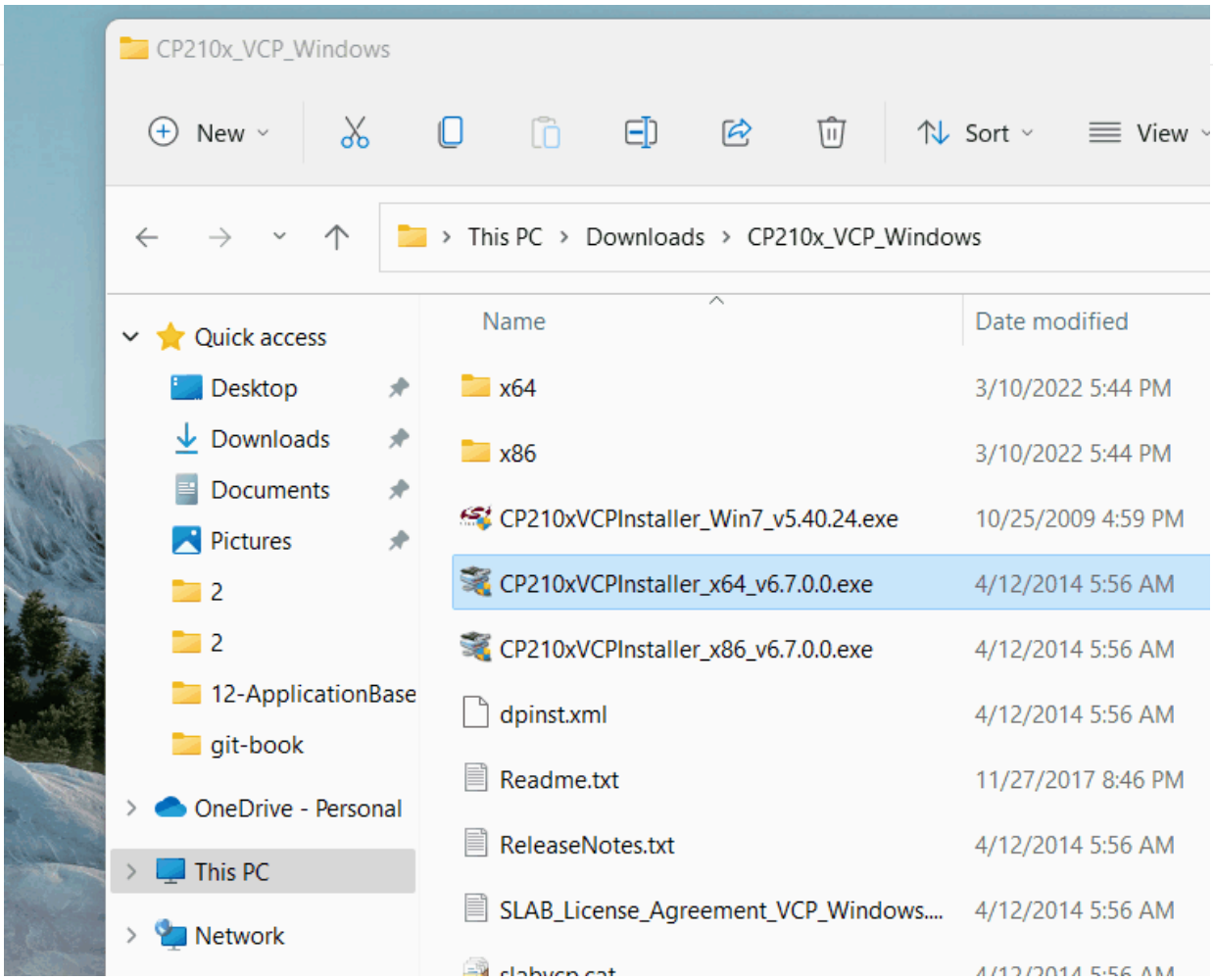


1.2 Serial Port Driver Installation

Download corresponding serial port driver according to the USB chip on your PC. CP210X is suitable for CP2104 version and CP34X is suitable for CH9102 version. You can install both of them if you are unable to confirm the type of USB chip. Go to the address below to download and install serial port driver.

- serial port driver for M5Stack Basic:
  - CP210X: **Windows 10**, **MacOS**, **Linux**
  - CP34X: **Windows 10**, **MacOS**
- serial port driver for Atom:
  - **Windows 10**

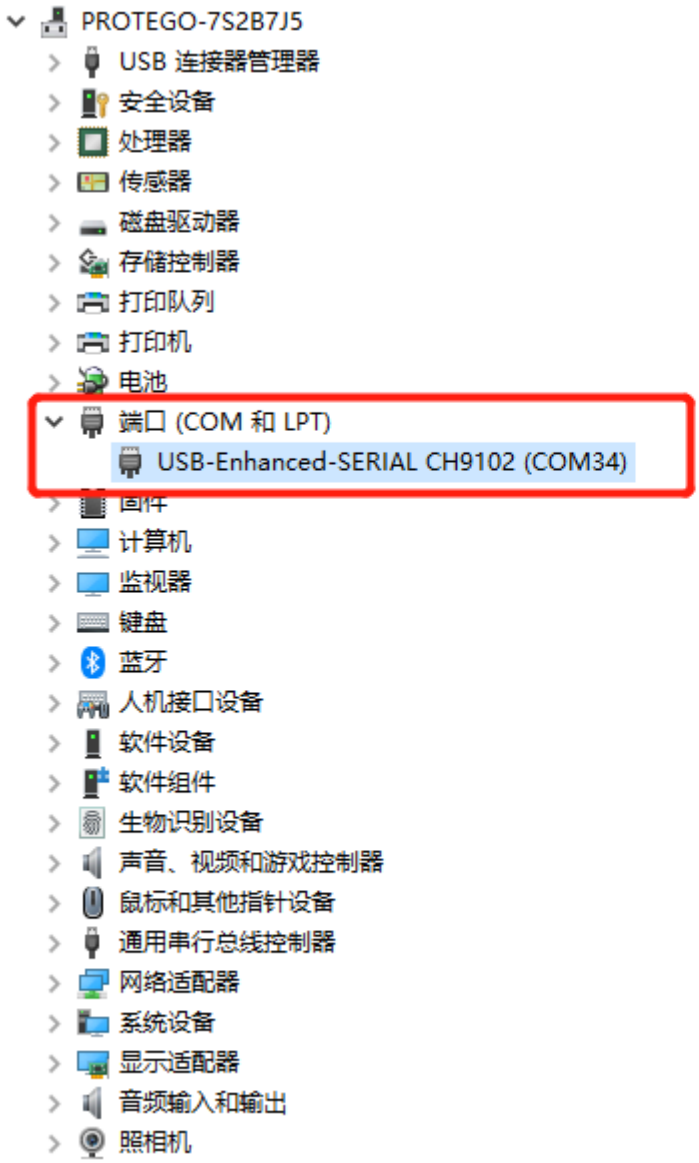
> **Notice:**
>
> - An error may be reported during installation of CH9102_VCP_SER_MacOS. However, the installation already completes. Just ignore the error.
> - Make sure that settings of the system \"Preferred settings -> Security and privacy ->General\" is correct and allow the user to get it from App Store or an approved developer.



1.3 How to distinguish between CP210X chip and CP34X chip

- As shown in the GIF below, go to `equipment manager` and check `ports (COM and LPT)` .

- If the `ports (COM and LPT)` show `USB-Enhanced-SERIAL CH9102`, the chip is CP34X.



- If the `ports (COM and LPT)` show `Silicon Labs CP210x USB to UART Bridge`, the chip is CP210X.


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

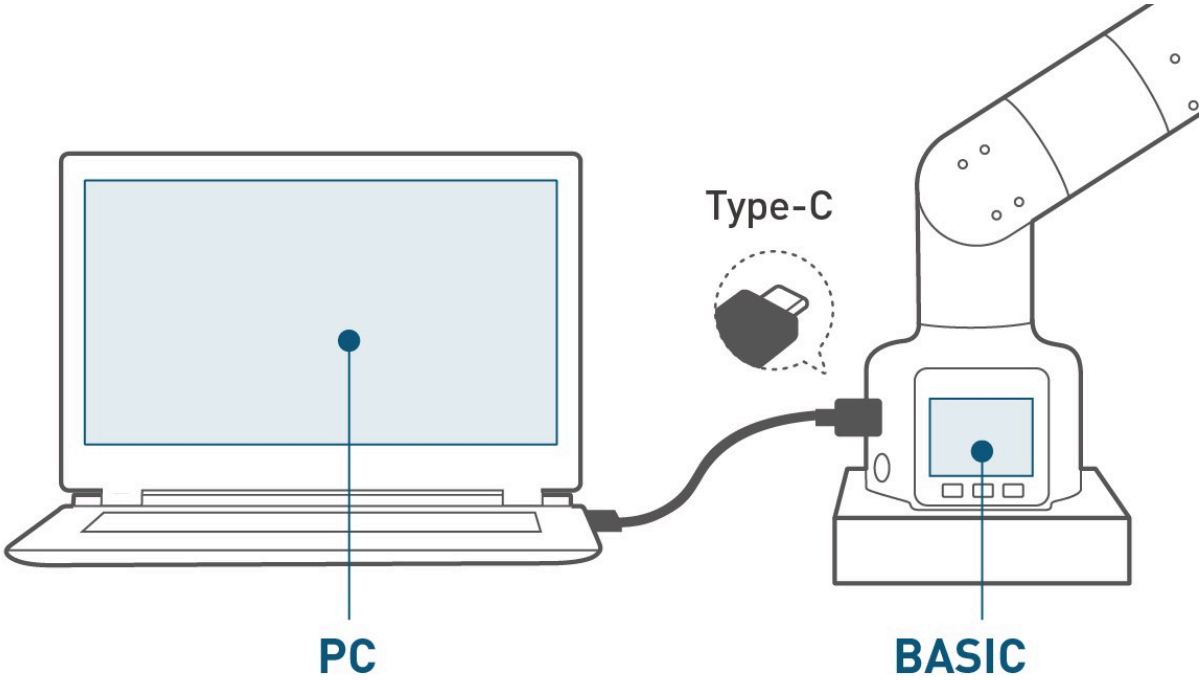
2 Burning and Updating Firmwares

Tutorial Video: https://www.bilibili.com/video/BV1Qr4y1N7B5/

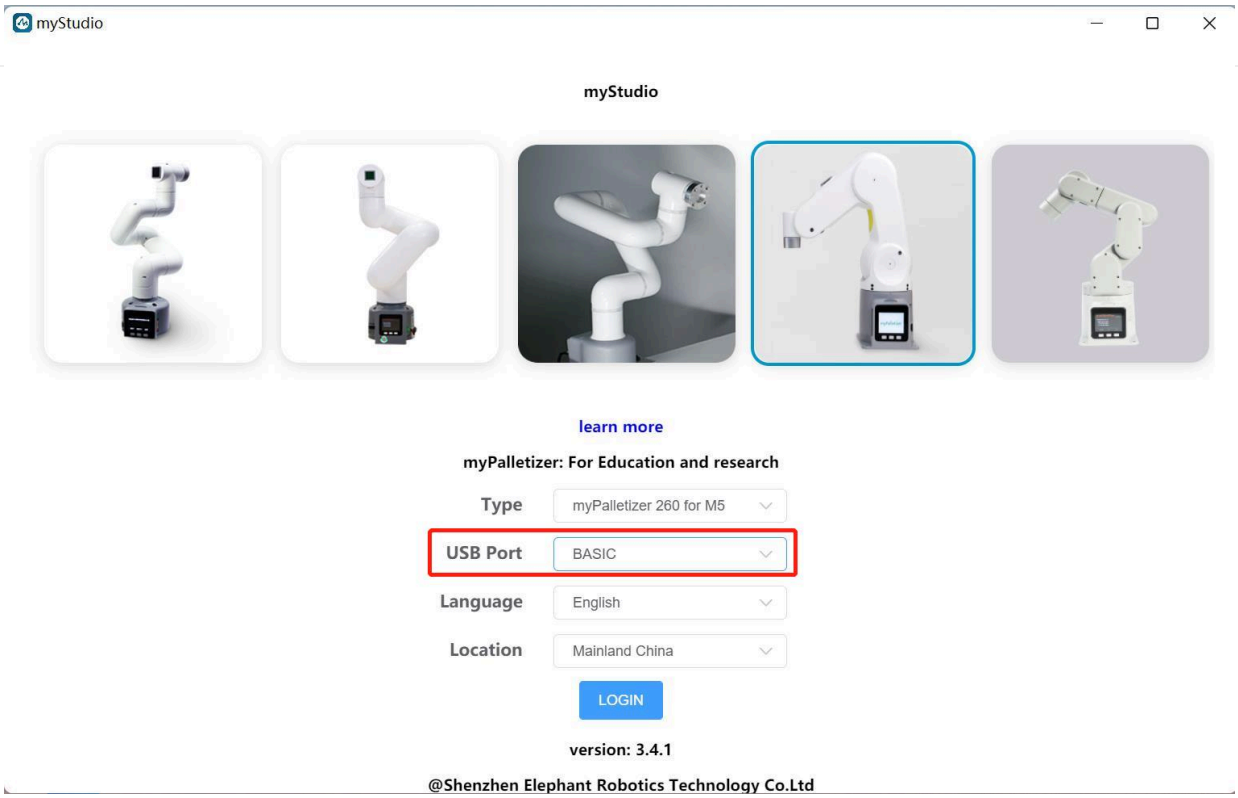
2.1 Burning M5Stack-basic Firmwares

> **Notice:** PI version is not required to burn M5Stack-basic firmwares

Step 1: Connect M5Stack-basic with PC with USB.

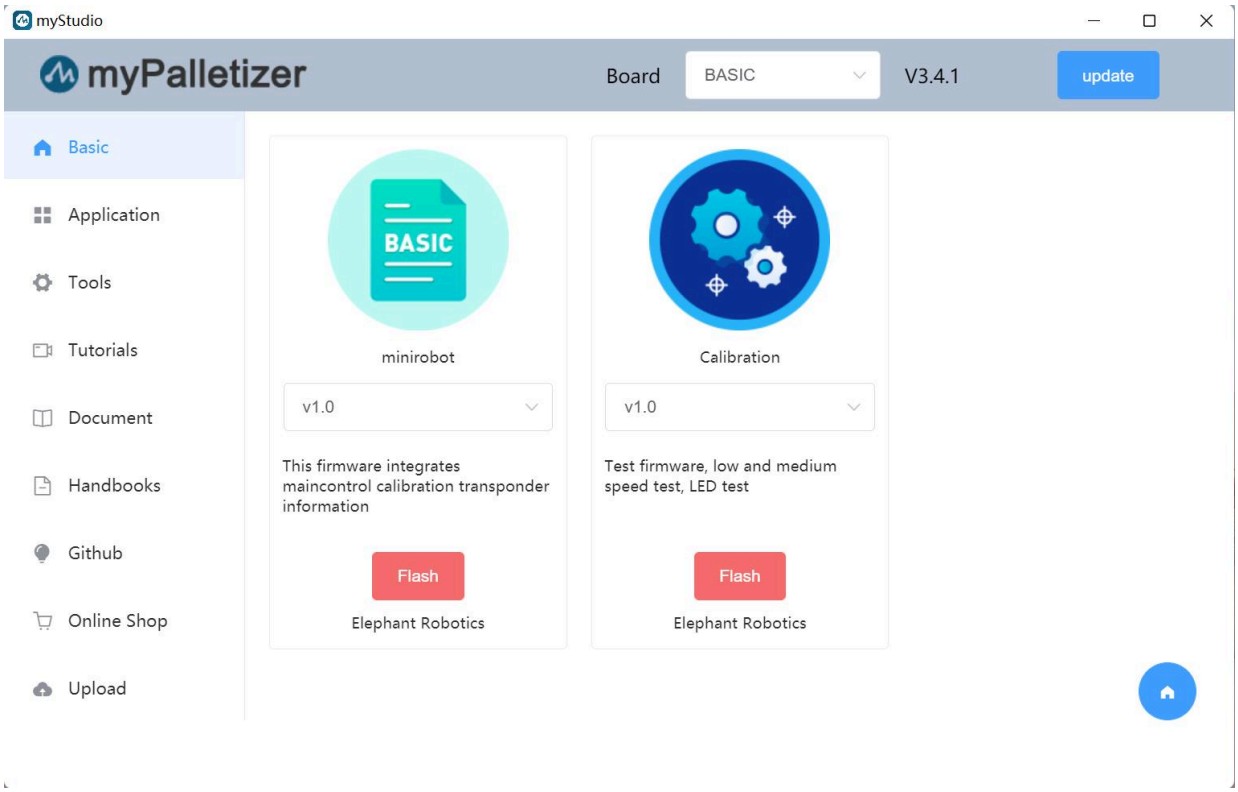


Step 2: Select `USB Port`. After connecting with PC, `USB Port` shows `M5Stack-basic`. The figure below takes myPalletizer 260 M5 as an example.



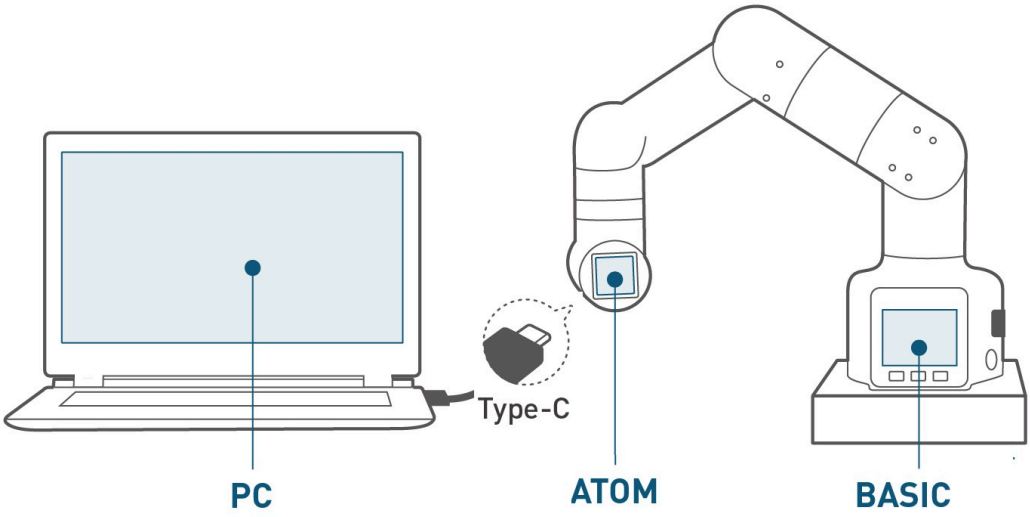
- Click on `LOGIN` -> `M5Stack-basic`, and then you can burn firmwares.

> **Notice:** As 280 PI/Jetson nano/Arduino versions do not have M5Stack-basic, the `USB Port` cannot show anything after connection.

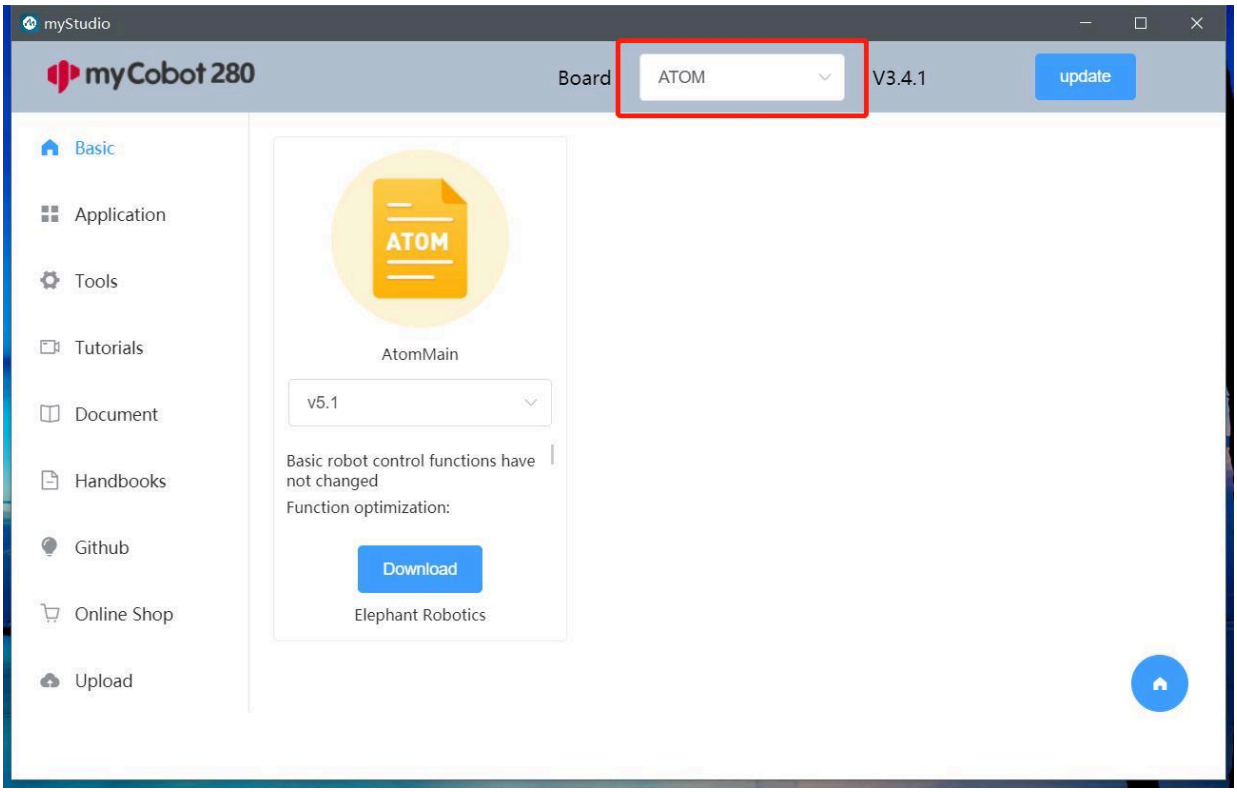


2.2 Burning Atom Firmware

Step 1: Connect Atom with PC with USB.



Step 2: Select `ATOM` at Board, and then burn AtomMain. The picture below takes myCobot 320 as an example.

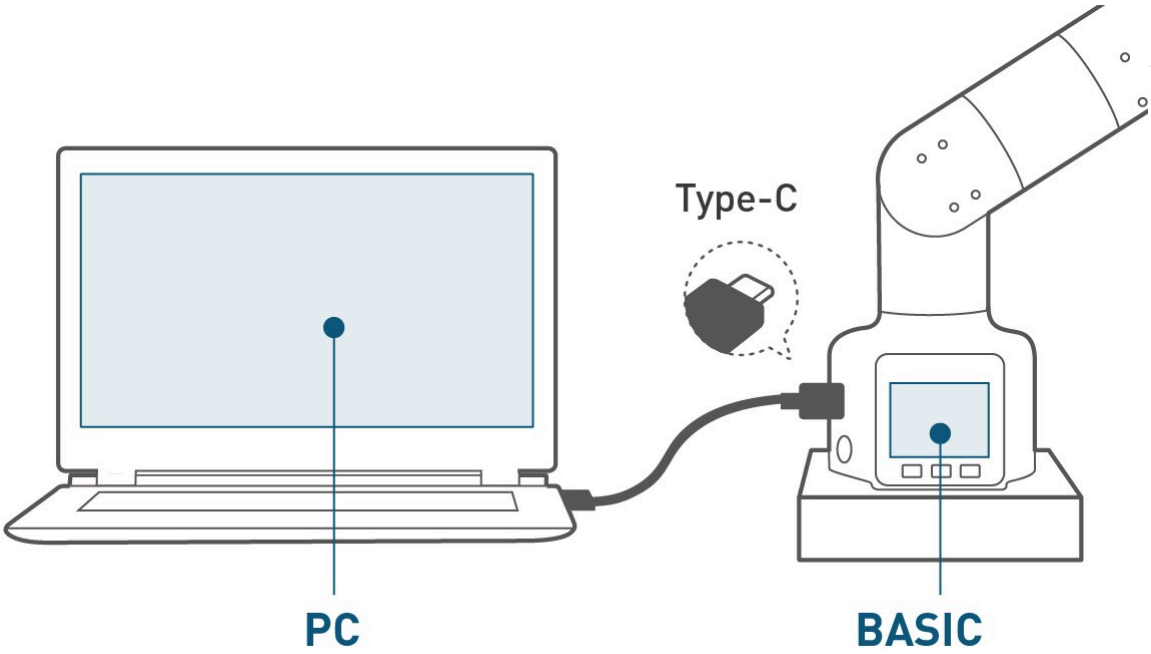


2.3 Firmware Burning for myCobot 320 series

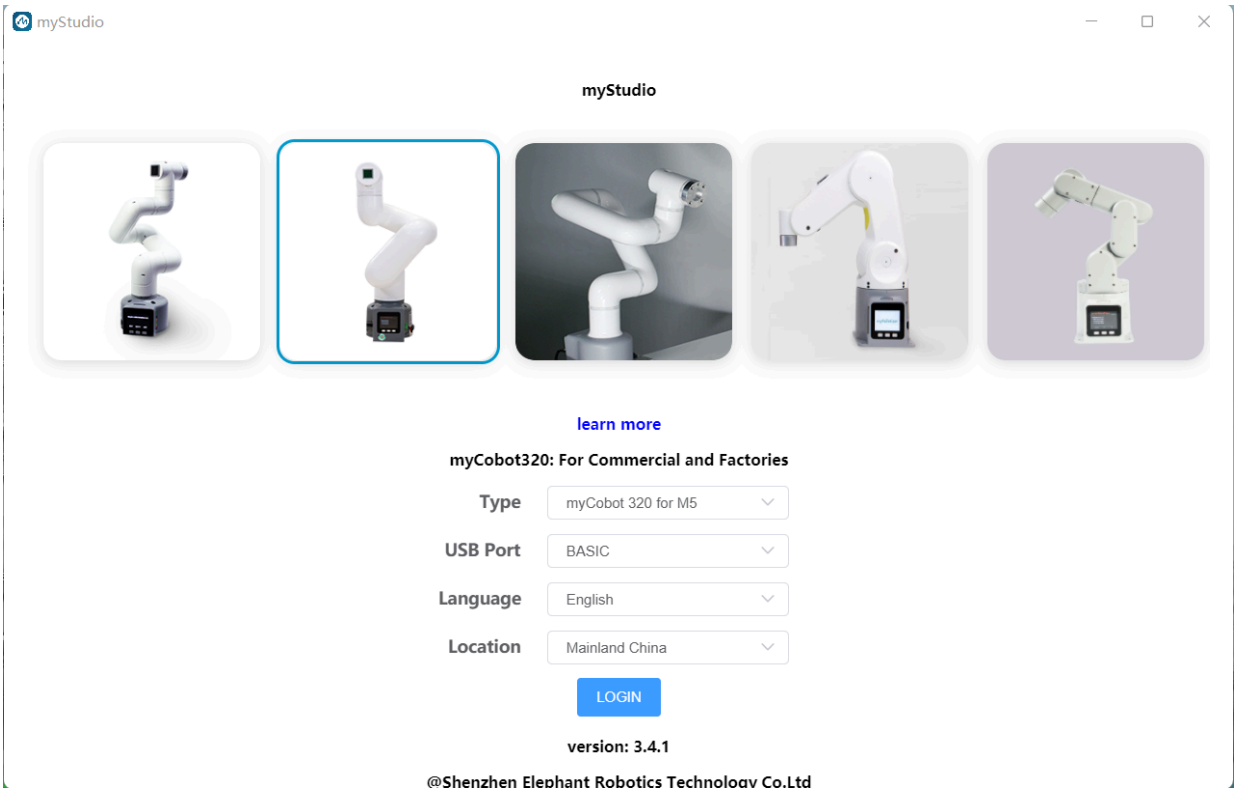
M5Stack-basic and picoMain are required to be burnt on myCobot 320 series. It should be noted that the two firmwares are burnt by different type-c interfaces.

1.Burning M5Stack-basic: miniRobot

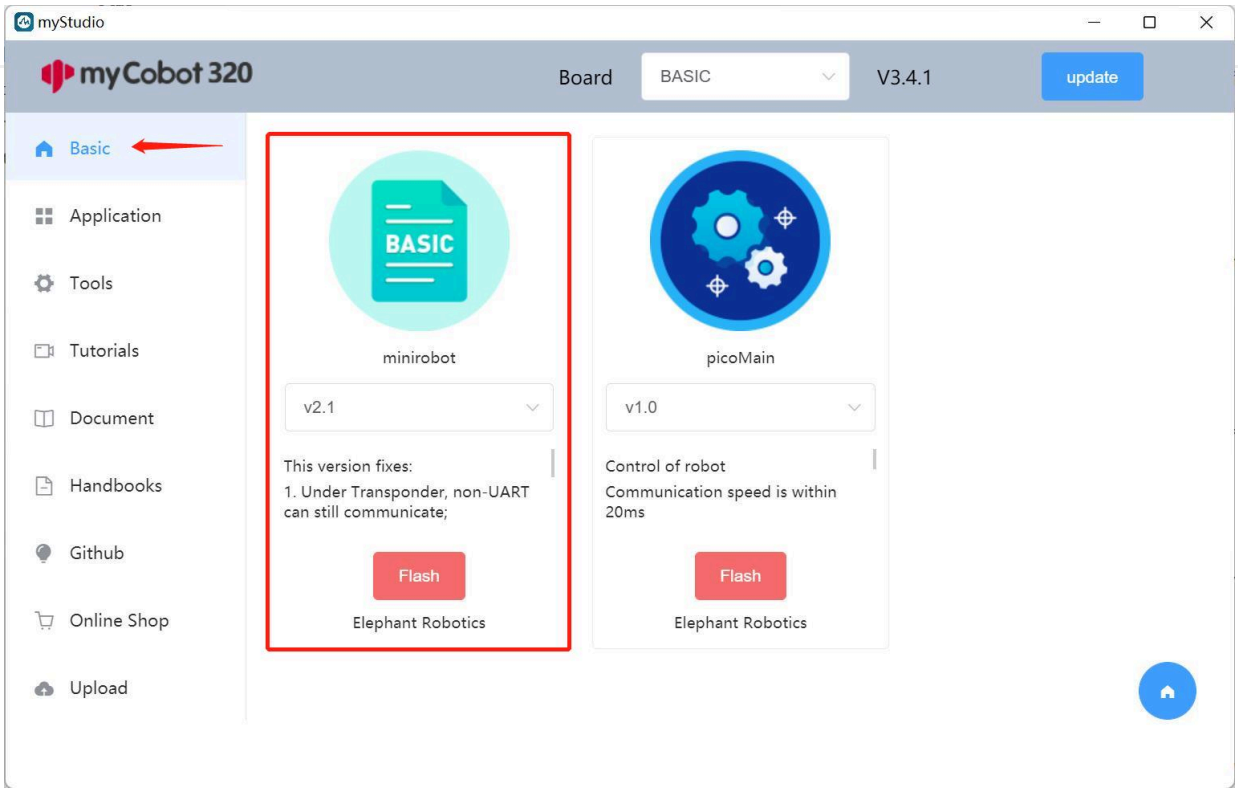
Step 1: Connect M5Stack-basic and PC.



Step 2: Select the 320 series, language and region, and then click on `LOGIN` .



Step 3: Click on `Basic` , and select minirobot to burn.



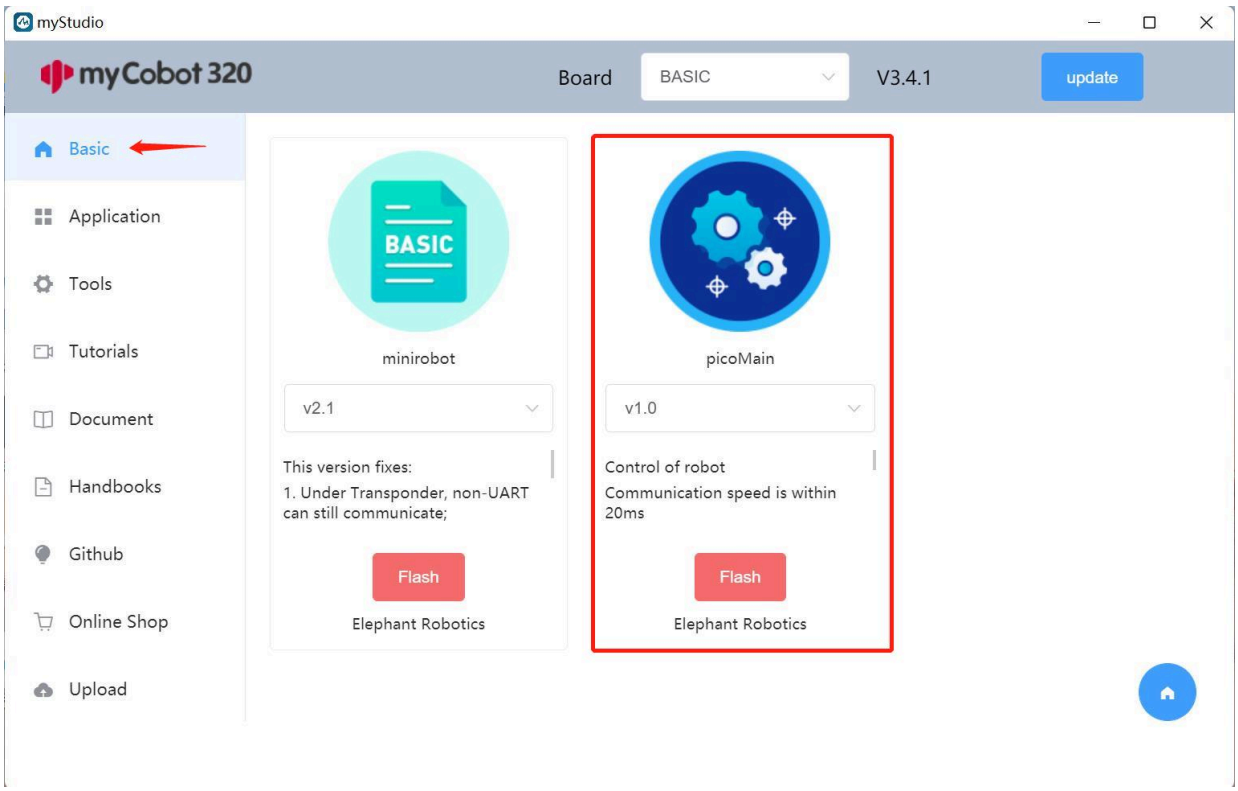
2.Burning Pico Firmware: picoMain

Step 1: Connect M5Stack-basic with PC.

Step 2: Select `Flash` (press `UP` once or `DOWN` four times), and then the screen goes dark for approximately 30 seconds.

Step 3: select picoMain to burn via myStudio.

> **Notice:** picoMain is required to be burnt during the sleep period of `Flash` mode. Otherwise it may fail to burn picoMain.



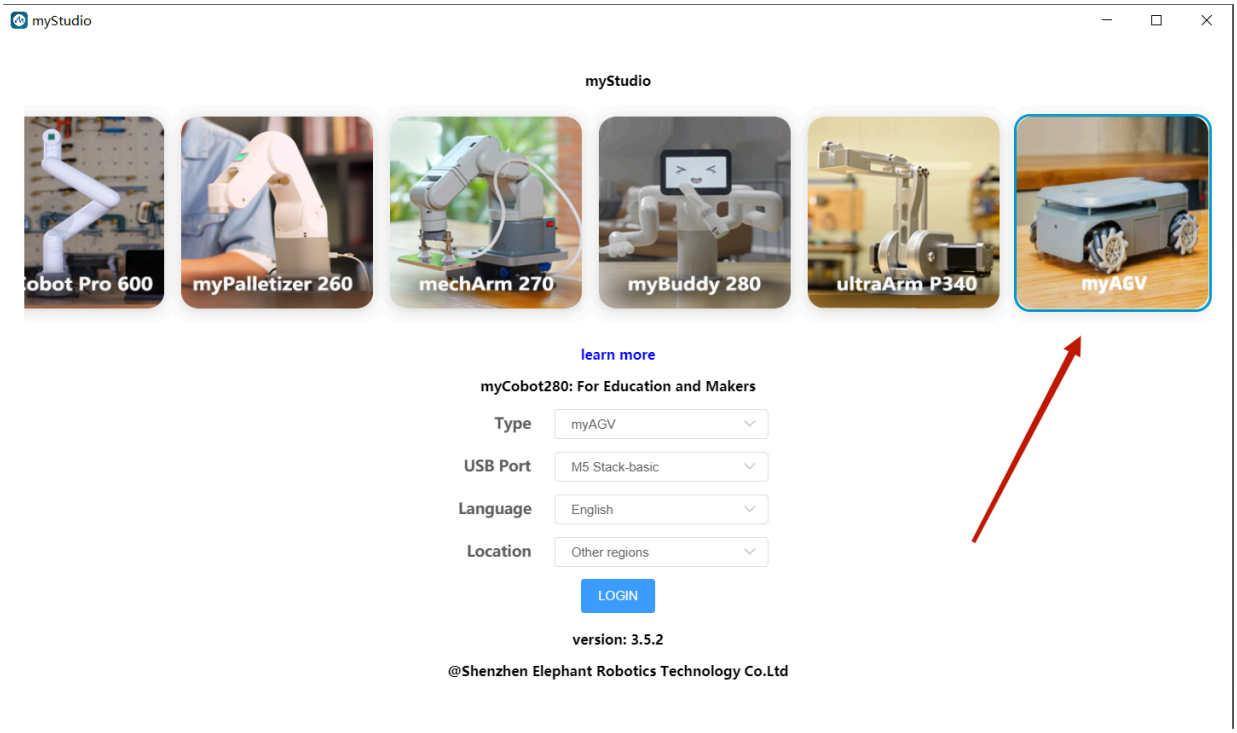
2.4 myAGV Firmware burning

1.Burning Pico Firmware: picoMain

Step 1: Turn on the computer and connect the screen.

For detailed tutorials, please refer to：2.5.1 First-time Use · GitBook (elephantrobotics.com)

Step 2: Select the myAGV image, select the model language and region, and click to log in.

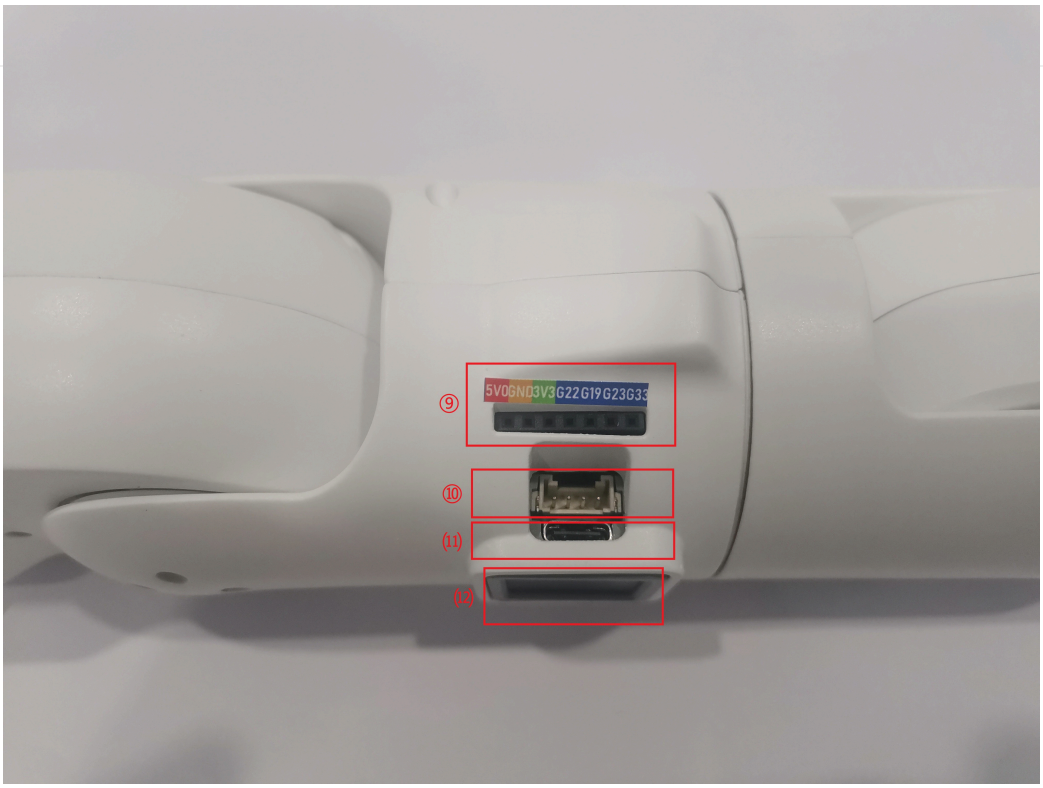


Step 3: After logging in, click 'Basic' and select picoMain to burn.

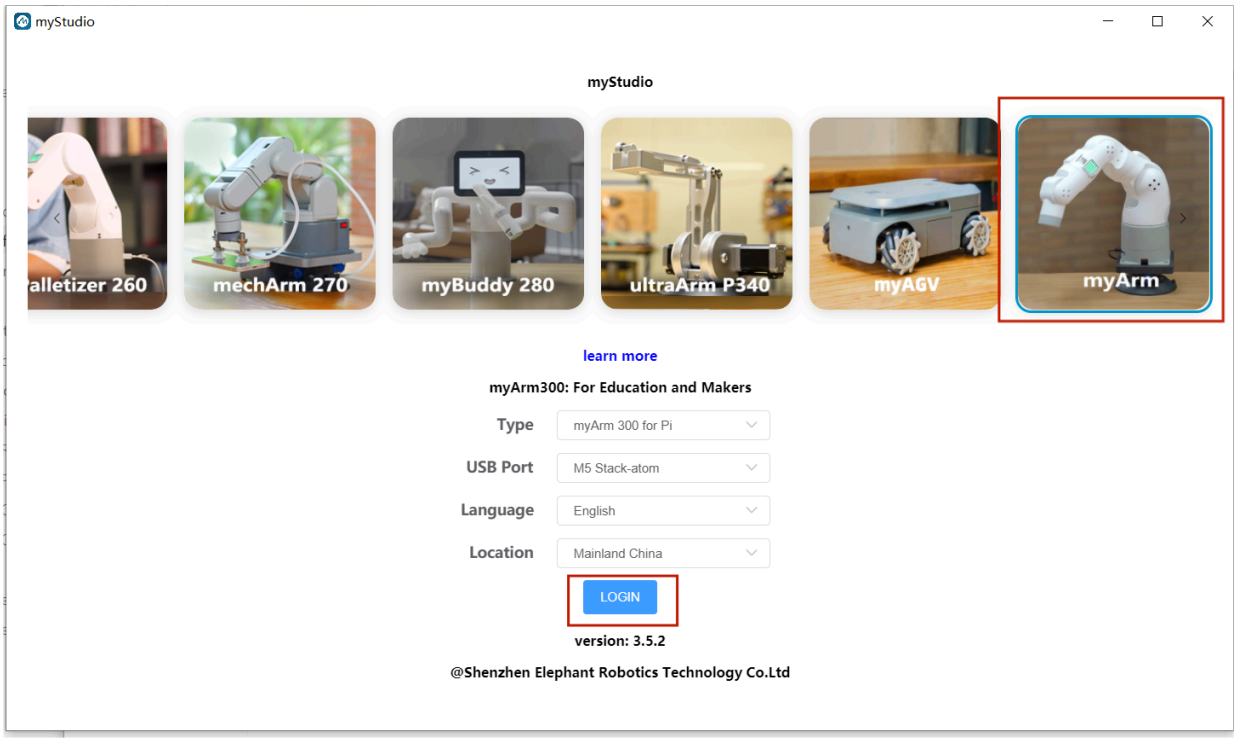
2.5 myArm Firmware burning

1.Burning Atom Firmware

Step 1: Connects to a PC. Use USB to connect the Atom interface (labeled 11 as Atom Type C interface).



Step 2: Select the myArm image, select the model language and region, and click to login.



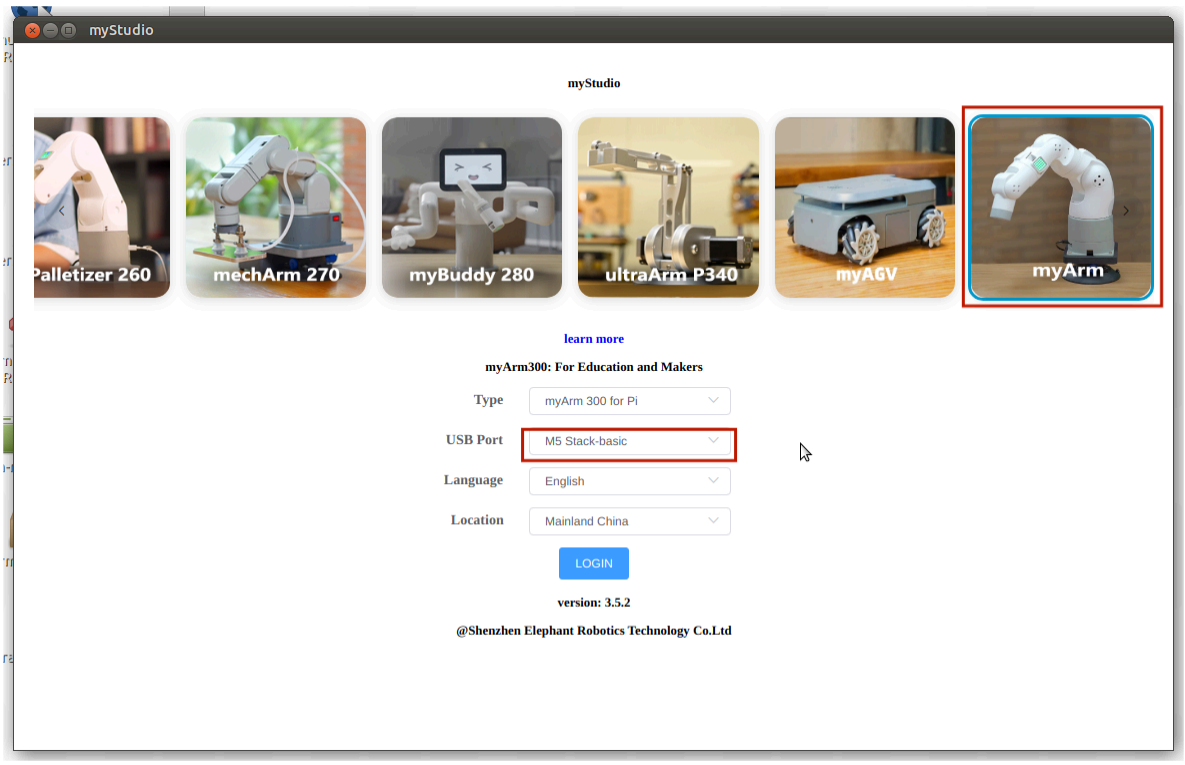
Step 3: After login, click 'Basic', select AtomMain, click download, and then click the burn button to burn.

2.Burning Pico Firmware: picoMain

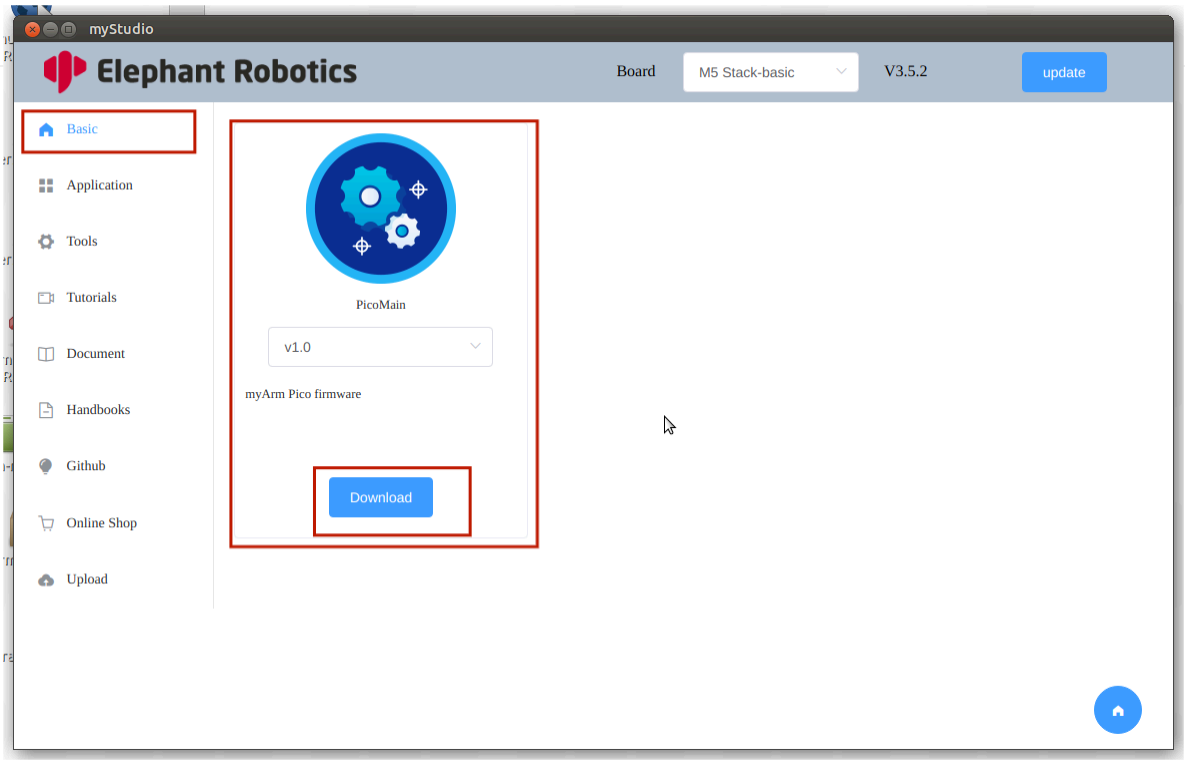
Step 1: Turn on the computer and connect the screen.

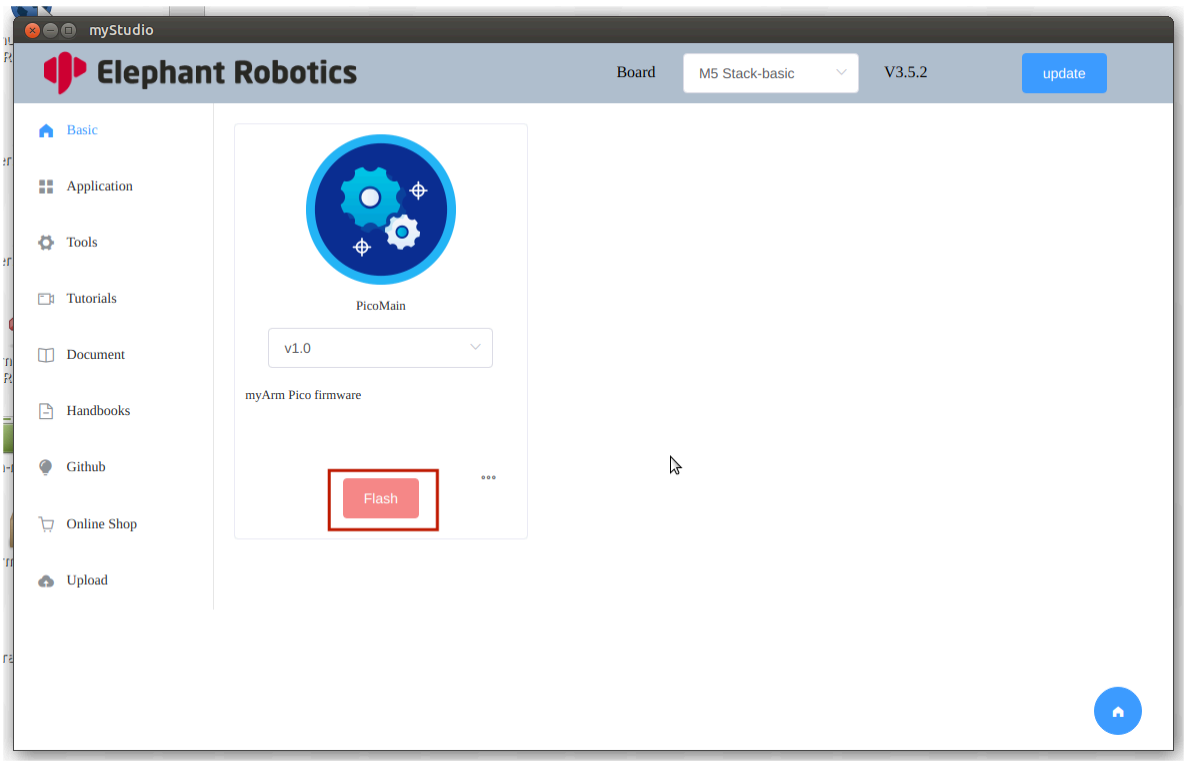
For detailed tutorials, please refer to：initial-setup

Step 2: Select the myArm image, select the model language and region, and click to login.



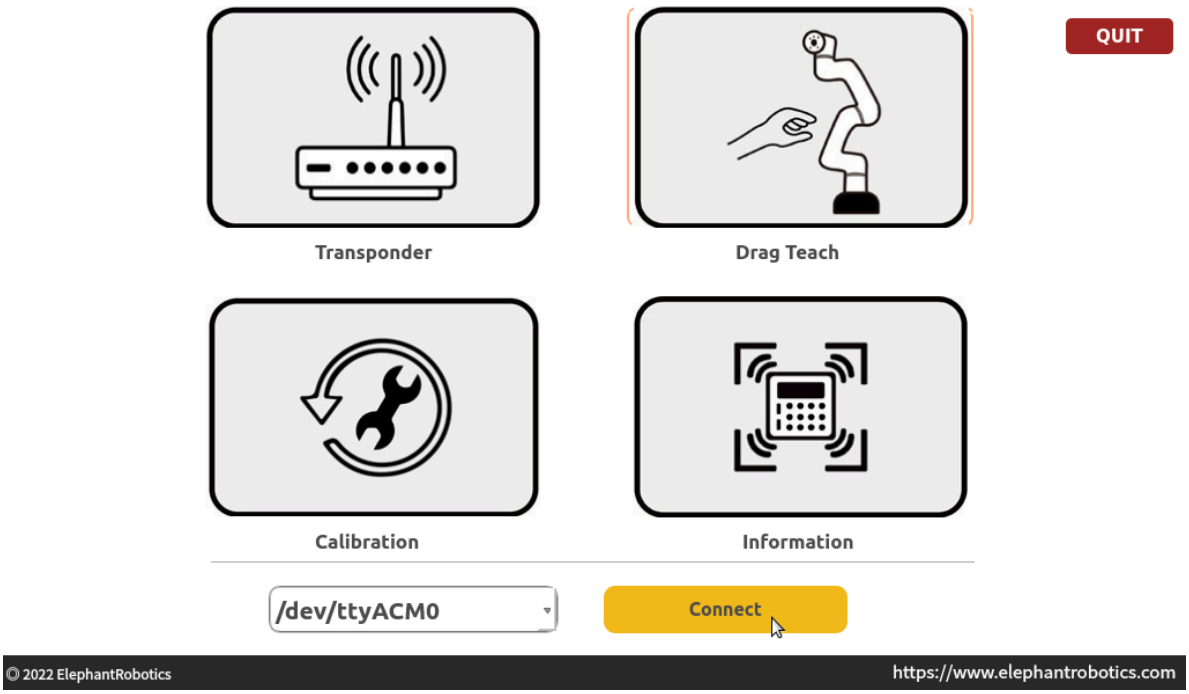
Step 3: After login, click 'Basic', select picoMain, click download, and then click the burn button to burn.


myStudio
Elephant Robotics
Board
M5 Stack-basic
V3.5.2
update
Basic
Application
Tools
Tutorials
Document
Handbooks
Github
Online Shop
Upload
PicoMain
v1.0
myArm Pico firmware
Download

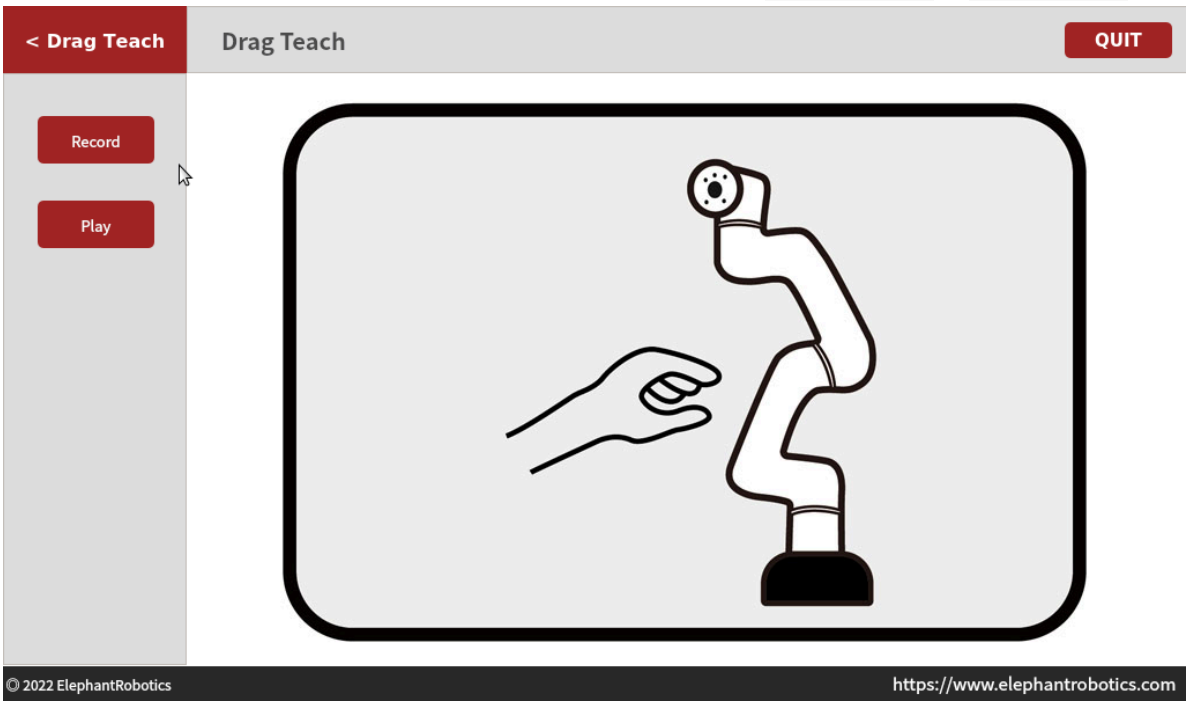

myStudio
Elephant Robotics
Board
M5 Stack-basic
V3.5.2
update
Basic
Application
Tools
Tutorials
Document
Handbooks
Github
Online Shop
Upload
PicoMain
v1.0
myArm Pico firmware
Flash

myBuddy GUI drag teaching use

1. After opening `mybuddy GUI`, select **Drag Teach** and click the **Connect** button to connect.



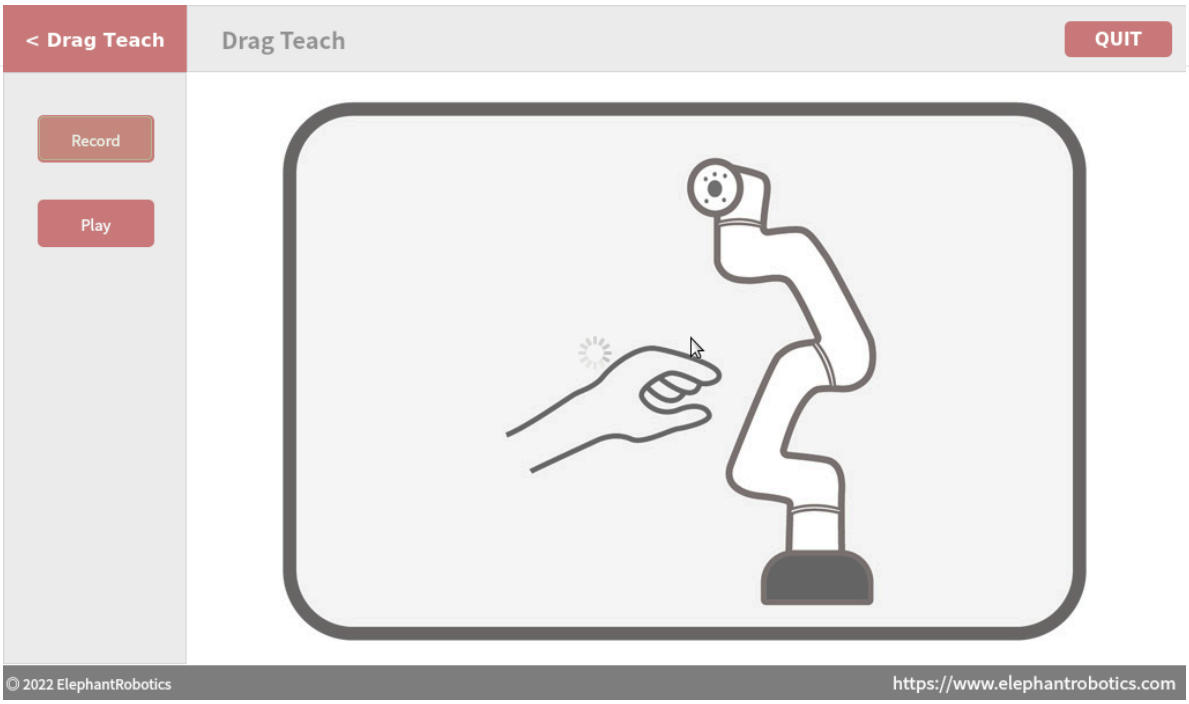
2. After connecting, it will jump to the next page, where you can choose to `track recording` or `track playback`.



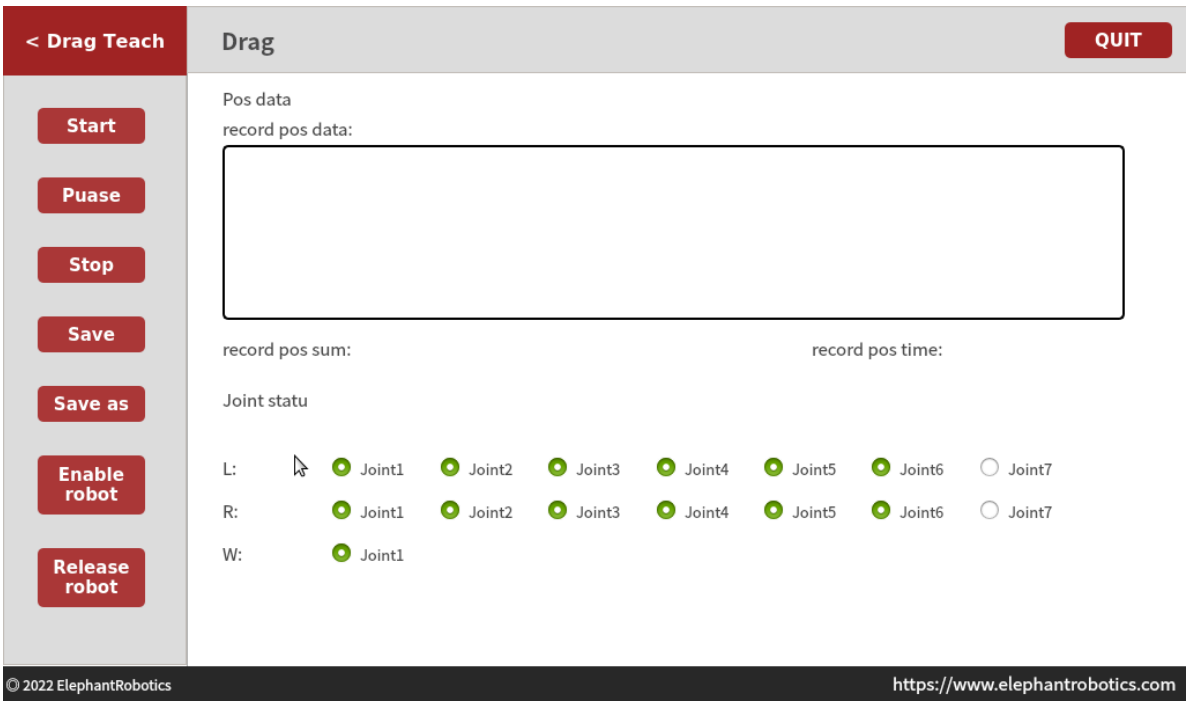
3. **Record**: Track recording
4. **Play**: Track playback

5. **Record**: After clicking this button, a pop-up window will pop up to prompt that it is switching to drag teaching mode. Click **OK**, and after the switch is completed, you will enter the track recording interface.

Switching modes:



Entering the track recording interface:



> This interface only provides the recording function. If you want to execute the track, you need to switch to the track playback interface.

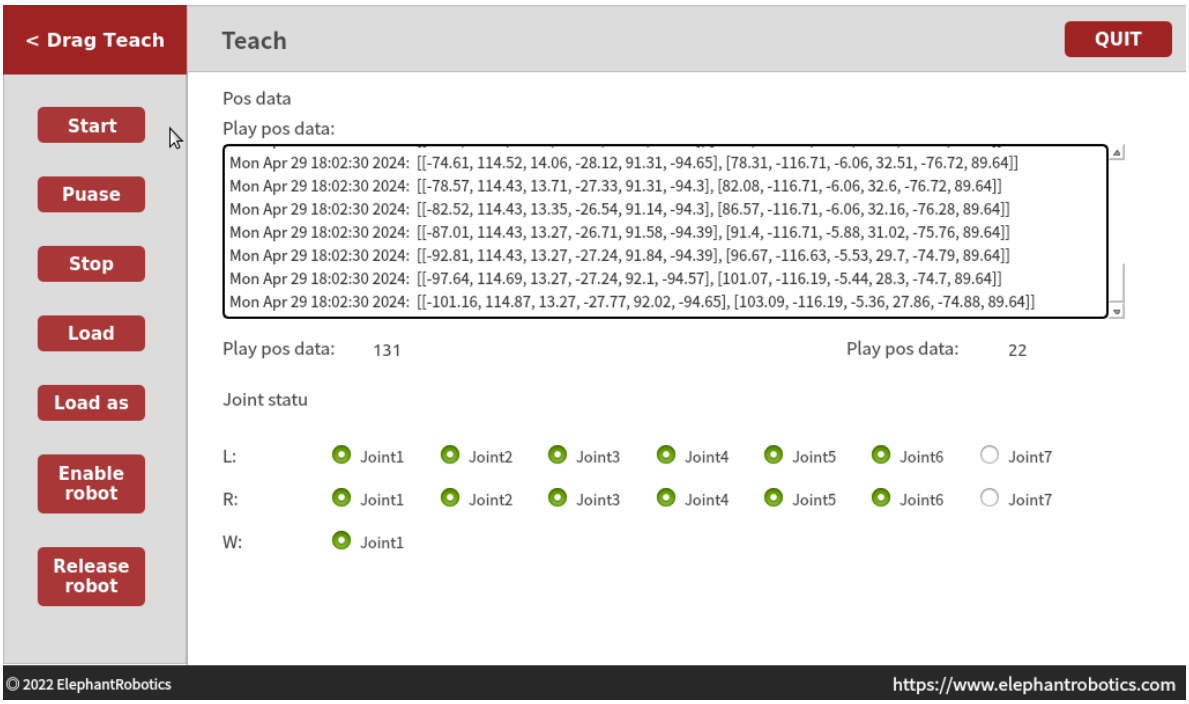
i. **Function introduction**:

- `Start` : Start recording the track.
- `Pause` : Pause the recording track. If you click Pause recording, the last recorded track will be connected when you start recording next time.
- `Stop` : Stop recording the track. If you click Stop recording, the last recorded track will be cleared when you start recording next time.
- `Save` : Save the recorded trajectory information to the /home/er/record.txt file.
- `Save as` : Save the recorded trajectory information, and specify the saved file location.
- `Enable Robot` : Enable the robot joints.
- `Release Robot` : Relax the robot joints.
- `< Drag Teach` : Return to the previous interface.
- `QUIT` : Exit the application

i. **Usage**:

i. Hold the robot's arm with your hand to prevent the arm from falling suddenly after relaxing the joints. Click `Release Robot` to relax the joints.
ii. Click `Start` to start recording the trajectory points. You can drag the arm afterwards.
iii. After recording, you can select `Pause` or `Stop` to pause or stop recording according to specific needs.
iv. Click `Enable Robot` to re-enable the joints
v. Click the upper left corner of the application

6. **Play**: After clicking this button, you will enter the trajectory playback interface.



i. **Function introduction**:

- `Start` : Start looping the recorded track.
- `Pause` : Pause the track playback. If this button is clicked, the next time you start playing, the track before the pause will continue.
- `Stop` : Stop the track playback.
- `Load` : Automatically load the track information of the /home/er/record.txt file.
- `Load as` : You can load the track information of the specified file.
- `Enable Robot` : Enable the robot joint.

- `Release Robot` : Relax the robot joint.
- `< Drag Teach` : Return to the previous interface.
- `QUIT` : Exit the application

ii. **Usage**: After recording the information in the track recording interface or loading the track information, click `Start` to start the execution.

Introduction to PI version robots

- **Supported device models**

Robot	Introduction
myPalletizer 260 PI	Introduction
mechArm 270 PI	Introduction
myCobot 280 PI	Introduction
myCobot 320 PI	Introduction
myBuddy 280	Introduction

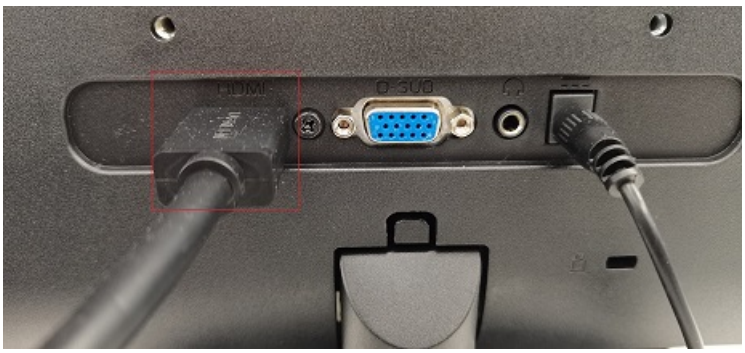
- **PI device function description**
  - PI version robots is a joint product developed by Elephant Robotics and **Raspberry Pi**. Adopting the **Raspberry PI 4B** as its core processor, and reserving the origin hardware interface of RaspBerry PI 4B, the robotic arm meets the application needs of **Linux system** with one-in-all structure for portable robot development. Featuring built-in **Ubuntu 18.04. system** as well as development environments like **Python**, **ROS** and **myBlockly**, it can be developed after connecting with displayer without matching with PC.
  - PI version robotic arms is embedded with Raspberry PI 4B, 1.5GHz 4-core microprocessor, runs Debian/Ubuntu platform, supports 4 USB channels, 2 HDMI channels, standardized GPIO interface, TF card is removable
- **PI robots are computer hosts**
  - The essence of the PI version robotic arms is a development board with an independent system, which can be seen as a miniature computer host. The communication between the host and the host cannot be simply constituted by a wire. It can only be connected to an independent monitor, and equipped with power supply, mouse and keyboard, and can be developed and operated after entering the built-in system of the development board
- **The difference between PI robots and other robots**

  Detailed differences between the M5 and PI versions can be found here

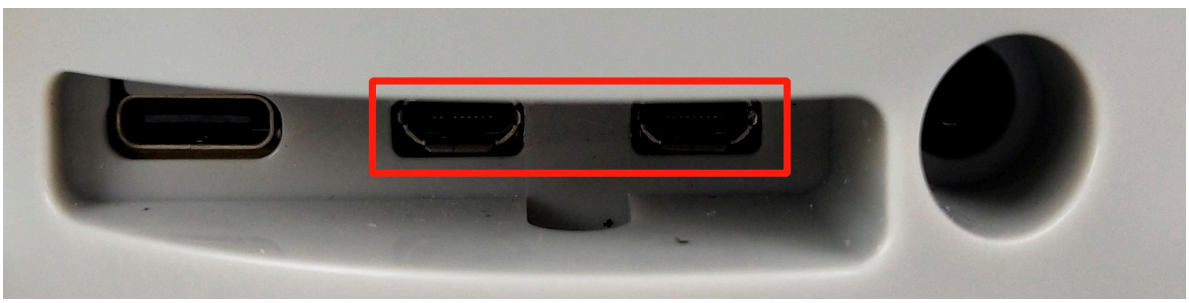
Introduction to PI version robots

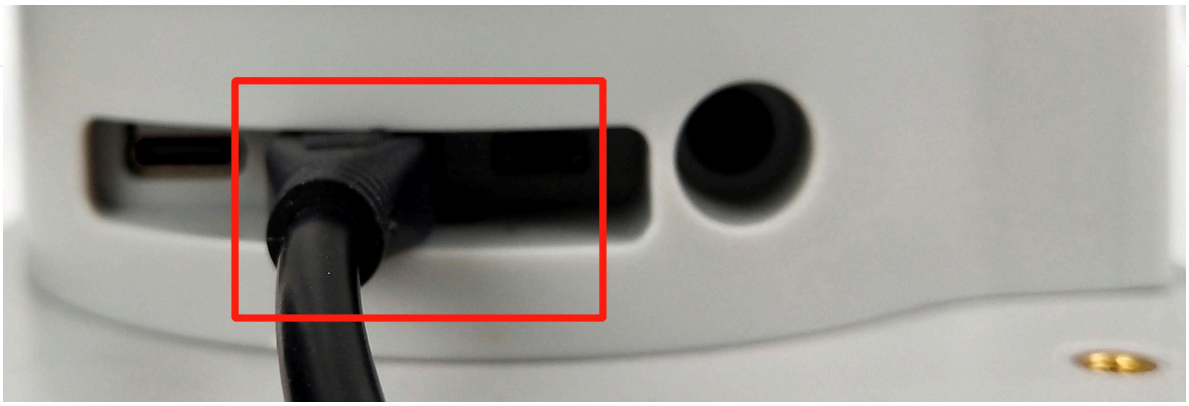
1.1 Start to use

- **Connect devices**
  - PI version robotic arms does not need to be equipped with a PC, laptop and other equipment, and can be connected to the display for application development（**Tips⚠：Use the delivered HDMI cable to connect the monitor and use the built-in system for development**）
  - Plug the HDMI cable into the HDMI port of the monitor.



- Plug the other end into the HDMI port of the robotic arm.



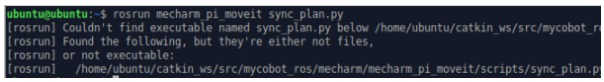
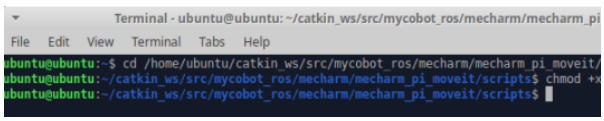
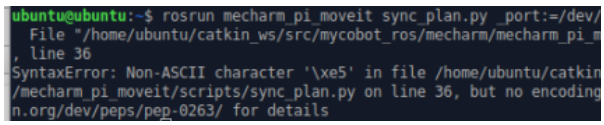
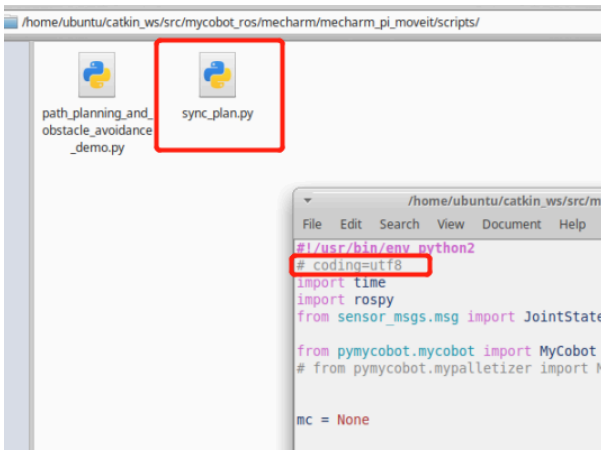


- **SD card description**
  - 32G TF card，built-in Ubuntu20.04 system，had installed**myStudio**，**myBlockly**，adapted **python ROS** develope env.
- **Optional items**
  - A network (Ethernet) cable to connect your Raspberry Pi to your local network and the Internet.
  - If you aren’t using an HDMI monitor with speakers you might also need some form of sound hardware. Audio can be played through speakers or headphones by connecting them to the AV jack (not available on the Raspberry Pi 400). However speakers must have their own amplification since the output from your Raspberry Pi is not powerful enough to drive them directly.
- **Troubleshooting**
  - Make sure you are using a good quality power supply; we recommend using an official power supply.
  - Make sure you are using a good quality power supply; we recommend using an official Raspberry Pi power supply.
  - You can get help with using the robot on our gitbook

1.2 Updates of the system

- **What is Mirroring**
  - Mirroring is a form of file saving. It refers to the fact that data saved in one disc also exists in another disc without any distortion. Mirroring files are often saved as BIN, IMG, TAO, DAO, FCD. It is similar to ZIP packages, which make a series of files into one single file according to certain formats to meet users demands. The most fundamental function of mirroring is that is can be identified by a software immediately and recorded on disc. Generally, mirroring files can be extended to cover more information such as system files. Thus, mirroring files can contain the information of even a hardware. The most typical software for creating mirroring files is Ghost, featuring recording function to save information on a disc.
- **Download system image files**
  - Download links

Product	Version	Link	SHA256 Hash
AI Kit 280	ubuntu 18.04	Download	d44439be351a52decdb4470cb623a032047e223ffce734
myCobot 280 PI	ubuntu 18.04	Download	04e40af5b637ec003a8b23ef9012e353361fd336db4e17
	ubuntu 20.04	Download	ce666e6c1047c512fe6b270336d472e48f231be1280872
myCobot 280 JetsonNano	ubuntu 18.04	Download	2f1e40c1480b077bcc83abd3b79ac175f25d21e9cc344a
myCobot 320 PI	ubuntu 18.04	Download	bc2ed6ef8d51a885f45379392b71e35420638a427d5b4b
	ubuntu 20.04	Download	c95633bfd49246254f2be4783c6a91a152124222191579

mechArm 270	ubuntu 18.04	Download	9af1fcbf9c608eda269dc395a8d68ea0a270008a88ec8ec
	Mirror Known Issues	moveit function abnormal	Solution: Execute moveit and the following prompt will appear, ind permission has been granted:  Enter the path shown in the figure and give the py file ex  Execute moveit and the following prompt will appear, ind encoding format in the code is wrong:  Enter the path shown in the figure, open the py file and e #coding=utf8 and save it 
myPalletizer 260	ubuntu 18.04	Download	f6fe999519146428e4c60960b242f647ae5c73c704852d6
	ubuntu 20.04	-	-

myCobot Pro 600	Raspberry Debian	Download	2e73aaa153bddbf0a49d18669a254b27403f17f8e989c05
myBuddy 280	ubuntu 20.04	Download	2b5452f665bcb999faf1727b2103dc1e5745705f5706728
myAGV	ubuntu 18.04	Download	bedad7d9769cb69380c6a4b9742ba7aefc21db41ab2391
marsCat	-	-	-

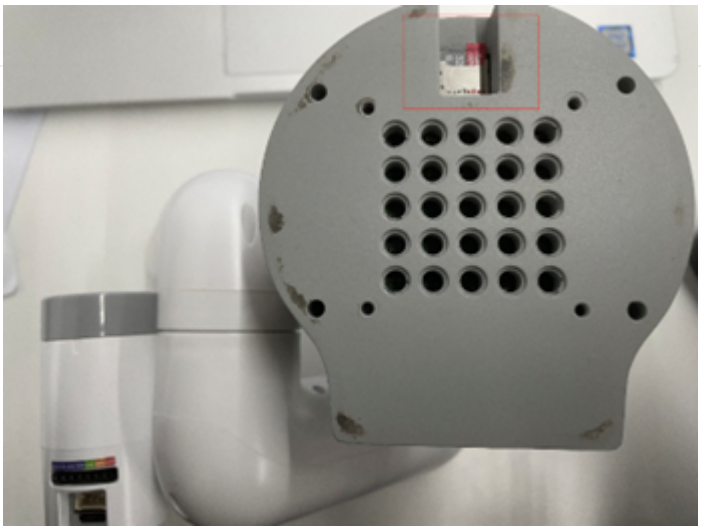
- **How to update the system**

  Step 1: Unzip the package and a file of image style appears.

  Step 2: Download Win32DiskImager.

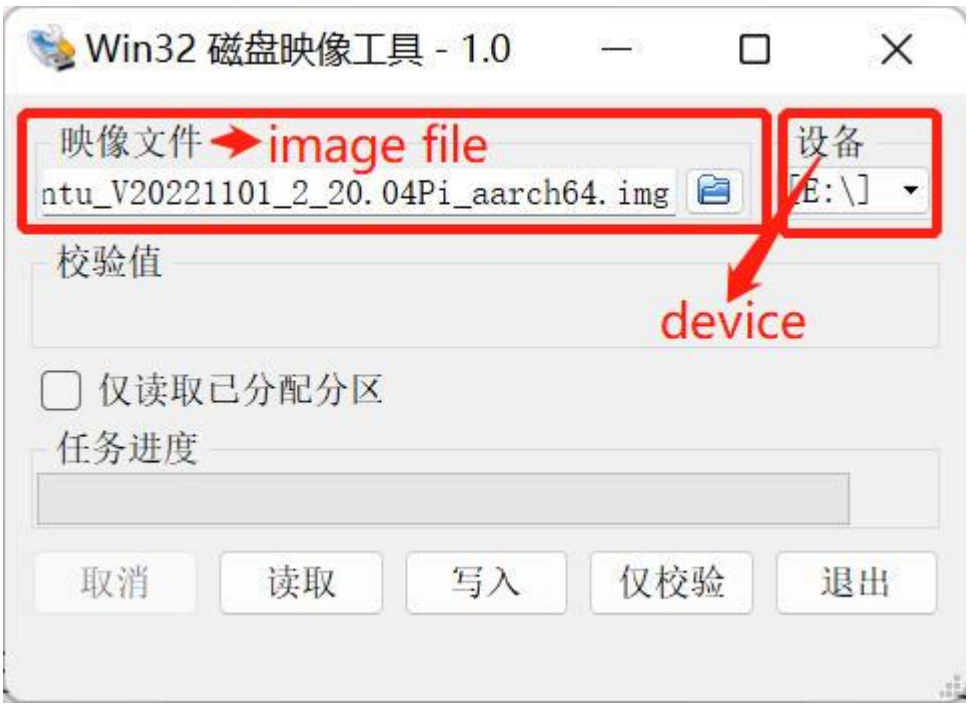
  Go to Win32DiskImager to download.

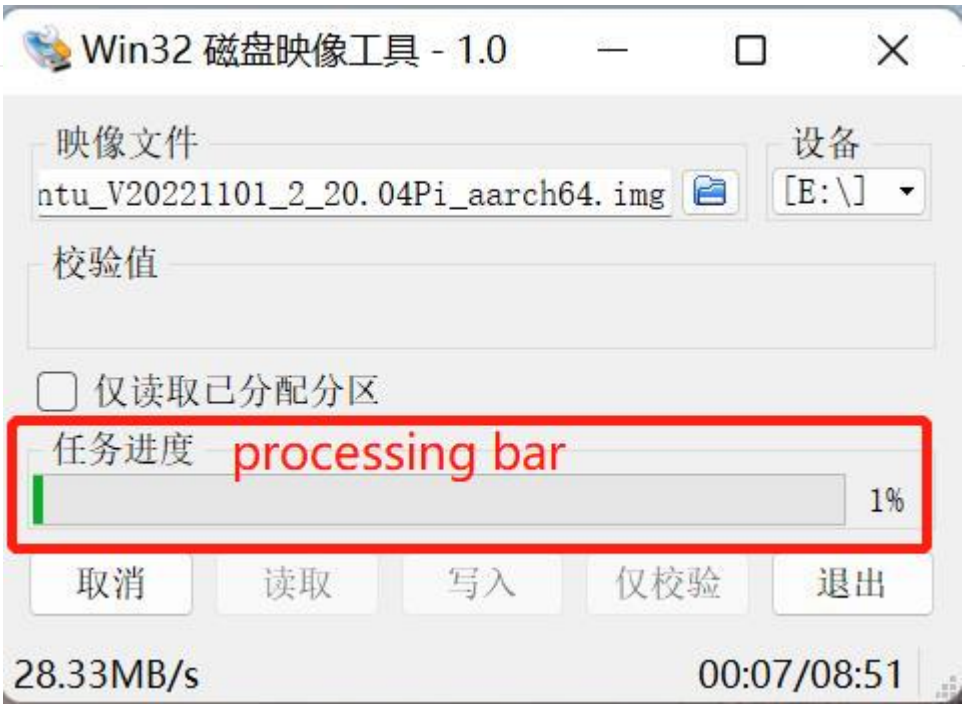
  Step 3: Remove SD card from the pedestal, and then insert the SD card into PC.



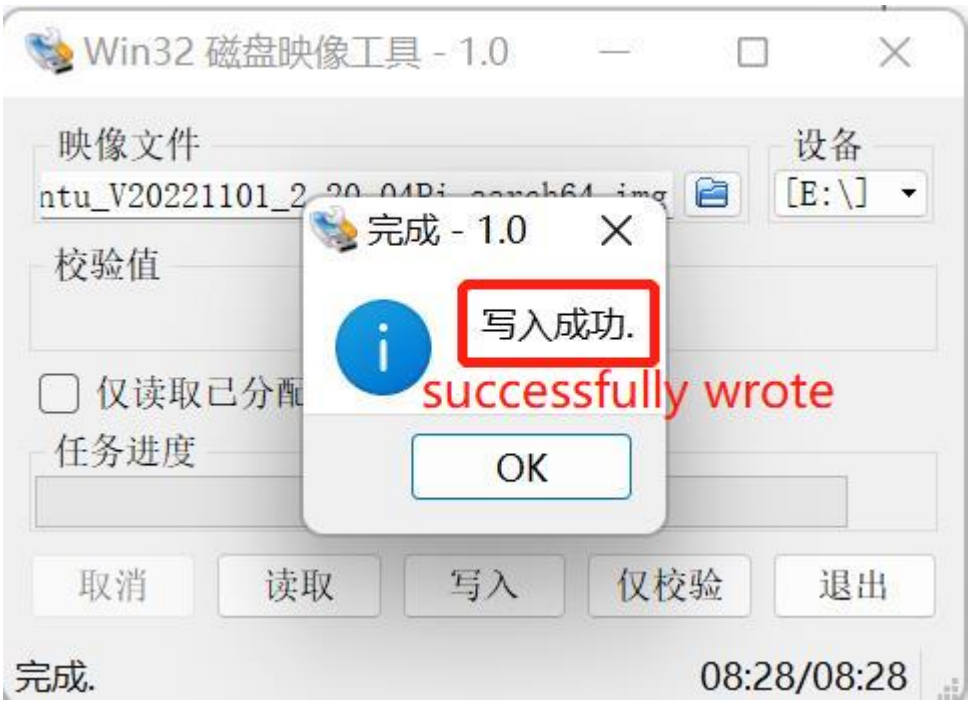
Step 4: Open Win32DiskImager.

Step 5: Select the software and device (E disc) and then write the software into PC.





Step 6: Successfully processed.



Description of basic system functions

2.1 Robot system introduction

- **System**
  - Ubuntu is the most widely used linux operating system for personal desktop operating systems. For beginners, familiarity with the linux environment or some embedded hardware operating system is a good choice. The official ubuntu website has also released a dedicated operating system for the Raspberry PI.

- **Function introduction**
  - **myStudio**：Firmware burning software for updating and burning new firmware.
- **myBlockly**：Graphical programming software, can be directly by dragging building blocks to form running code, control the robot arm
- **ROS1 Shell**：Directly enter the compiled ROS1 environment, you can directly enter the corresponding instructions, run the corresponding ROS1 code
- **ROS2 Shell**：Directly enter the compiled ROS2 environment, you can directly enter the corresponding instructions, run the corresponding ROS2 code
- **Github-ElephantRobotics**：Elephant robotics official open source code repository
- **Home-ElephantRobotics**：Elephant robotics website
- **UserManual - CN/EN**：Robot use manual，contains everything about robot control
- **WiFi_ON/OFF**：WiFi switch, click to turn on/off the WiFi function
- **HotSpot_ON/OFF**：Hotspot switch, click to turn on/off the hotspot function, the hotspot name is**ElephantRobotics_AP_XXXX**
- **Language Support**：System language setting, click to enter the system language setting interface
- **System details documentation link**
  - myPalletizer 260 PI Development environment detailed introduction
  - mechArm270 PI Development environment detailed introduction

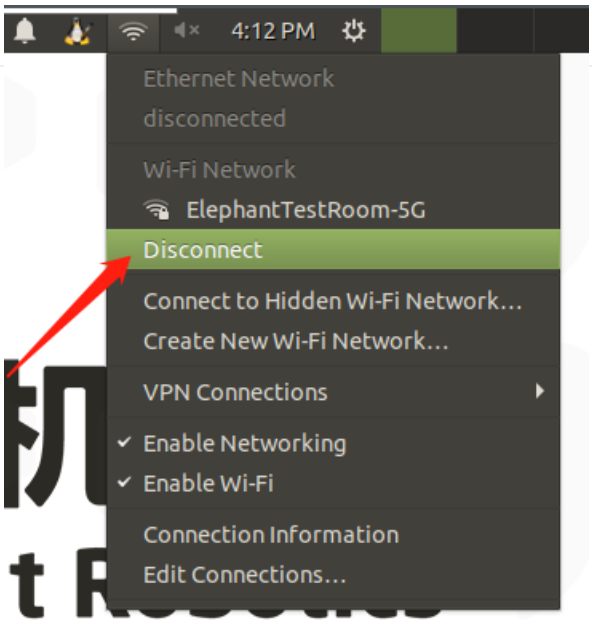
- myCobot280 PI Development environment detailed introduction
- myCobot320 PI Development environment detailed introduction

2.2 System password description

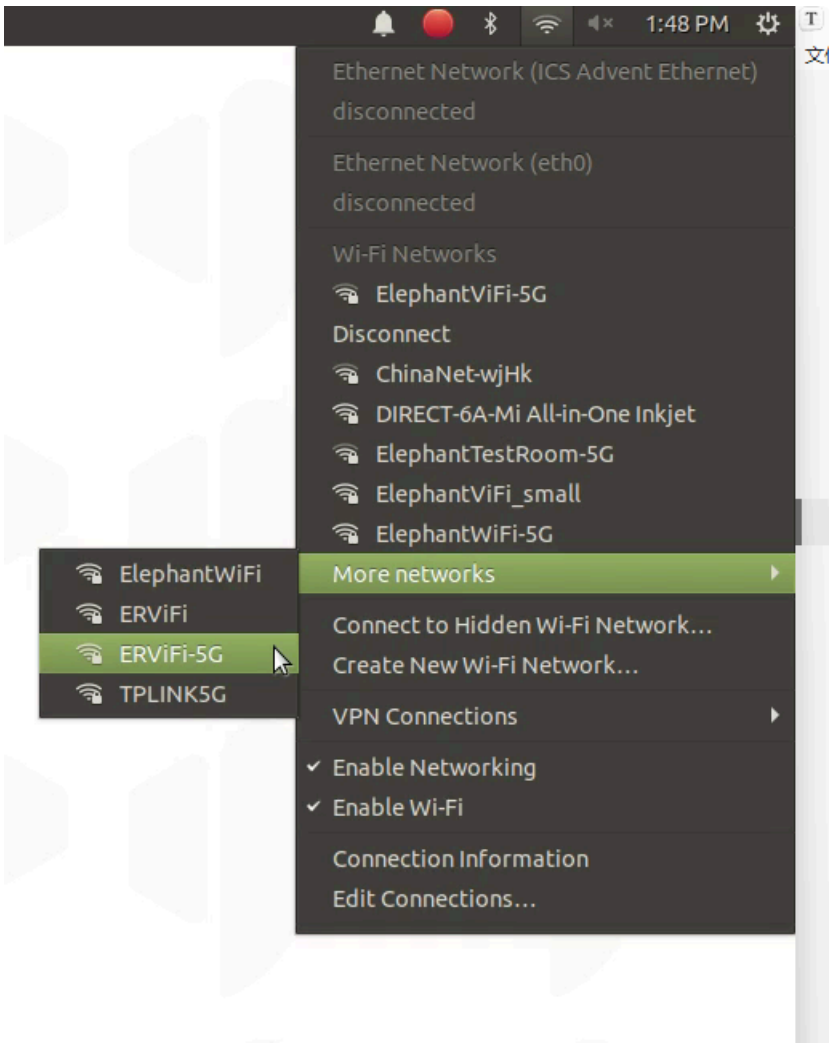
- **account password & VNC password & SSH password & root account password**
  - Password：**Elephant**
- **How to change password**
  - change account password
    - Use the shortcut key `ctrl + alt + T` to open the terminal
    - Enter `passwd`
    - Enter the new password twice
  - change vnc password
    - Use the shortcut key `ctrl + alt + T` to open the terminal
    - Enter `vncpasswd`
    - Enter the new password twice
  - change ssh password
    - For SSH remote connection, the password of the administrator account is entered. You do not need to change the password separately
  - change root account password
    - Use the shortcut key `ctrl + alt + T` to open the terminal
    - Enter `sudo passwd`
    - Enter the new password twice

2.3 VNC

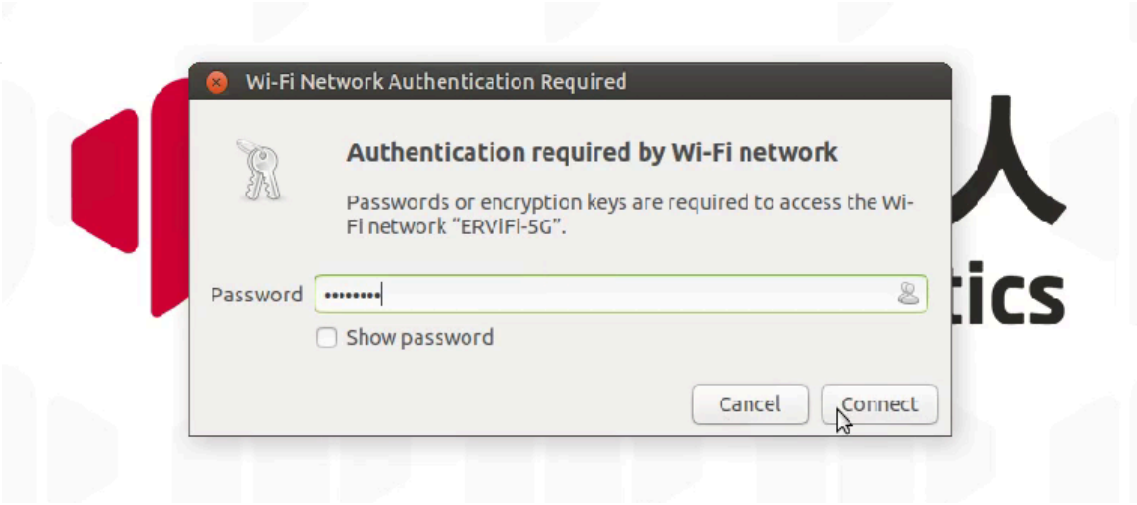
- **VNC Introduction**
  - Is a remote control software, generally used to remotely solve computer problems or software debugging
- **VNC Port**
  - When the robot arm and PC are connected to the same WiFi, the IP address of the robot arm is the port
- **Connect VNC**
  - There are two wireless connection modes. The first mode requires an external monitor to do some operations on the system. The specific steps are as follows:： Click **“Disconnect”，** disconnect the default hotspot connection



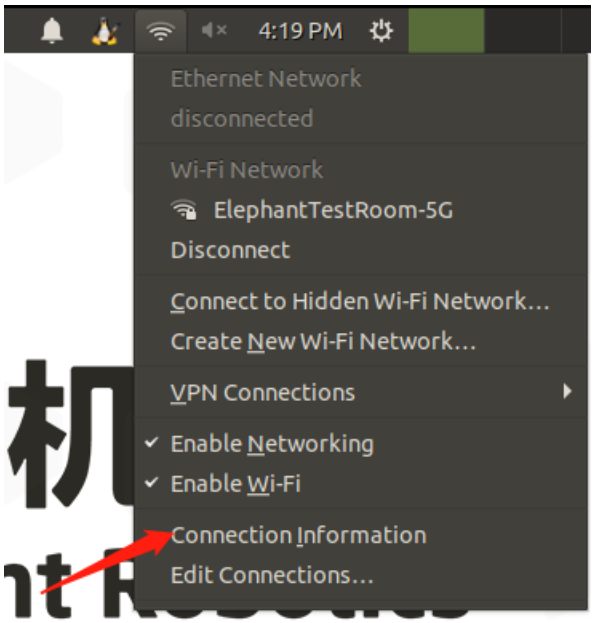
Click **"Enable Wi-Fi"** , and the currently available WiFi will appear



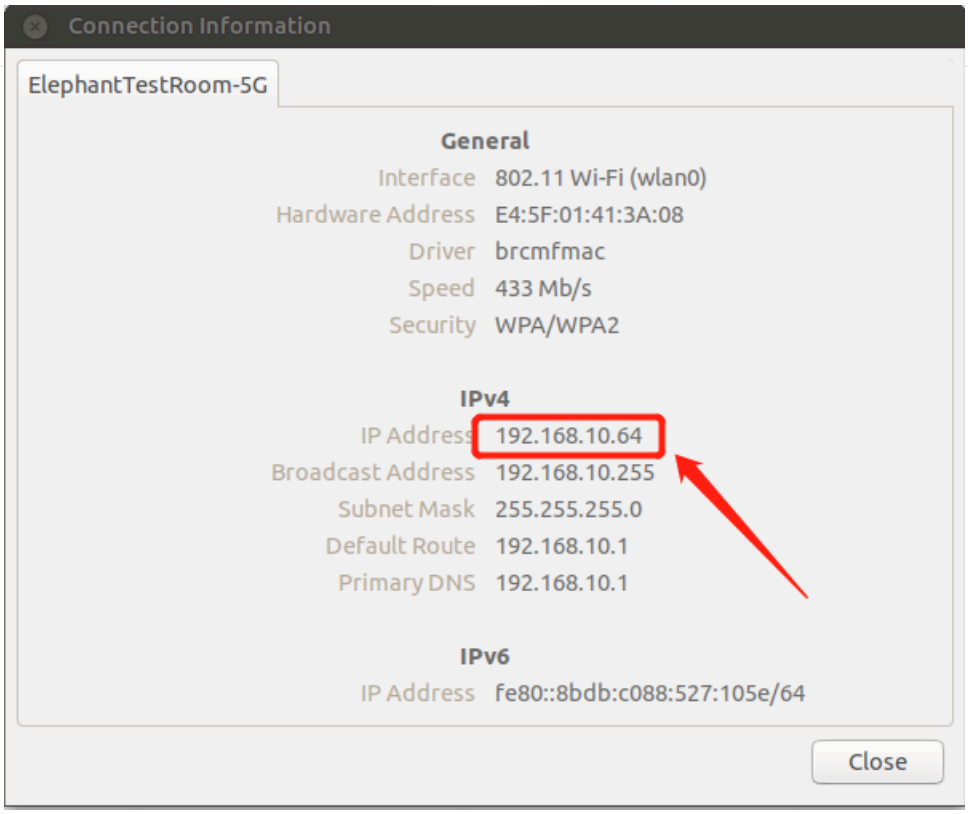
Click on the WiFi you need to connect to, enter the password



After connect successfully，click **"Connection Information"** to check IP address of the robot



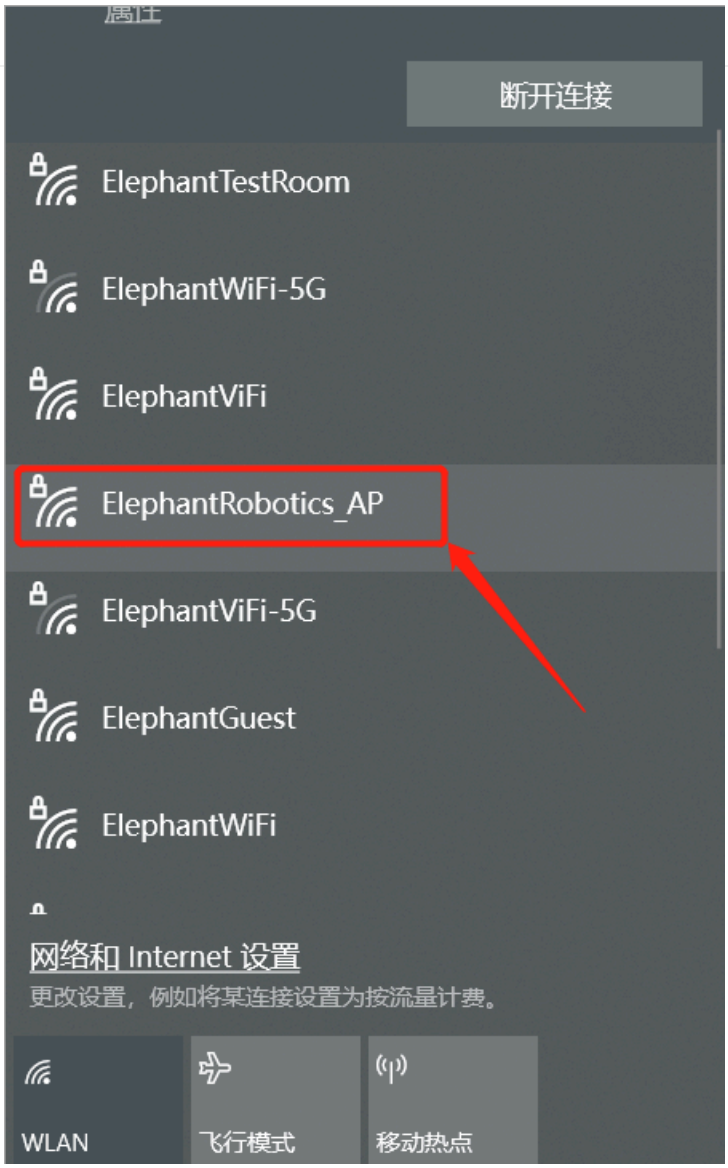
As shown in the example，**“192.168.10.64”** is the current IP address of the robot



Connect your PC and the robot to the same WiFi，open the VNC viewer，enter the IP address（examples : input **192.168.10.64**），enter the password **Elephant**，The user name is not specified by default. The following is an example of a successful connection：

- The second method doesn't need to connect the monitor, directly connect the Ubuntu system hotspot with a PC for remote control, but this connection method does not have the function of Internet surfing, and can only remotely control the robot arm system. The specific steps are as follows:

  Connect tp the hotspot **ElephantRobotics_AP_XXXX**，enter the password **Elephant**

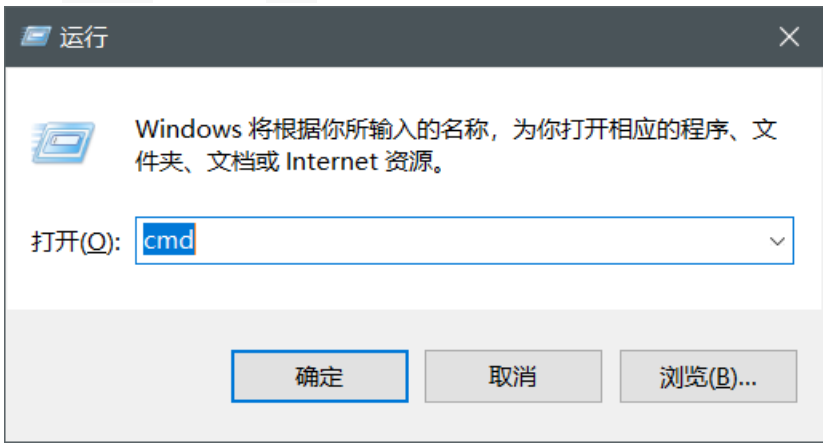


Open VNC viewer，input IP address **10.42.0.1** ，enter，and then enter the password **Elephant**，The user name is not specified by default. The following is an example of a successful connection：

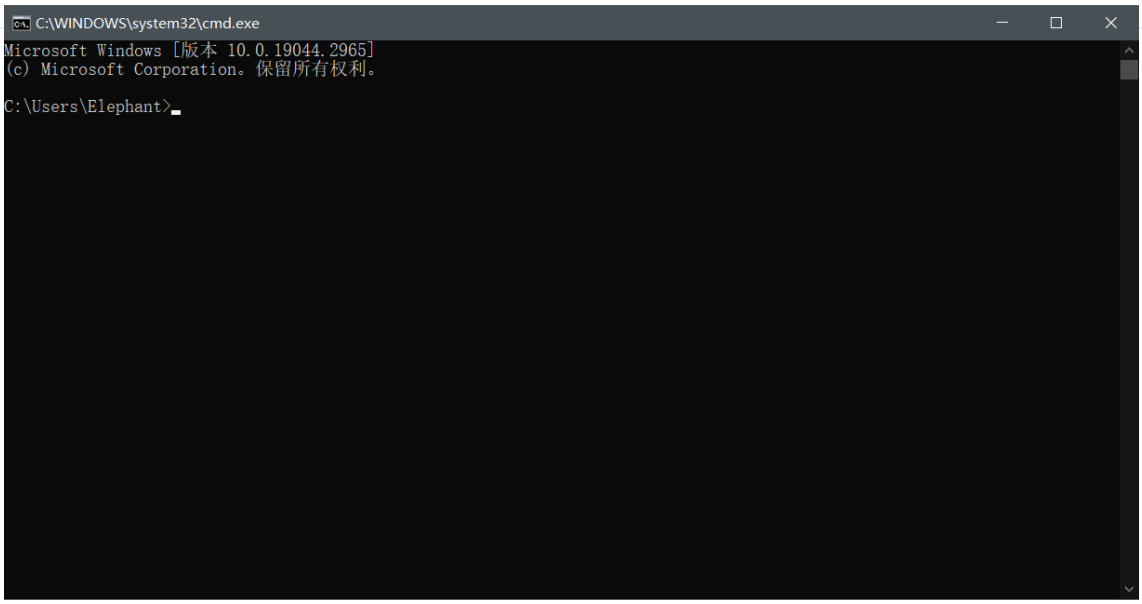
- **How can I improve connection fluency**
  - The fluency of the remote connection depends on the fluency of the connected WiFi. It is recommended to connect to a stable WiFi for remote control

2.4 SSH

- **SSH Introduction**
  - SSH is a network protocol used for encrypted logins between computers. If a user logs in from a local computer to another remote computer using SSH, we can assume that this login is secure, even if it is intercepted in the middle, the password will not be disclosed.
- **SSH Port**
  - The default port number is 22
- **SSH Connect**
  - Confirm the IP address of the robot by following instructions in **2.3 VNC**
  - Click `win + R` and enter `cmd`

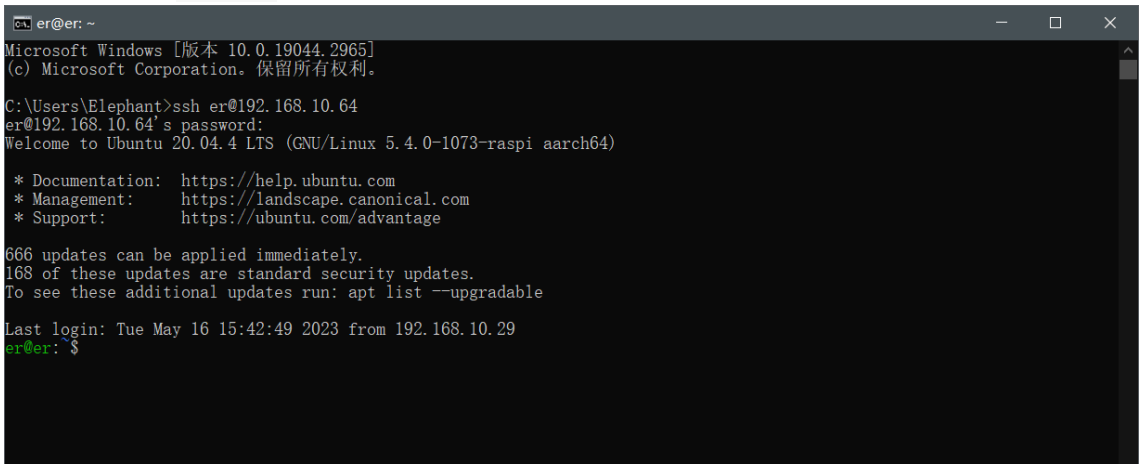


- After input, click OK to open the shell interface



- Enter `ssh er@IP address` ，then press `Enter` (the IP address is mainly displayed by the robot arm, the figure is only an example)

- Enter password `Elephant`



- As shown in the above figure, the robot arm has been successfully connected by ssh

- **How can I improve connection fluency**
  - The fluency of the remote connection depends on the fluency of the connected WiFi. It is recommended to connect to a stable WiFi for remote controll

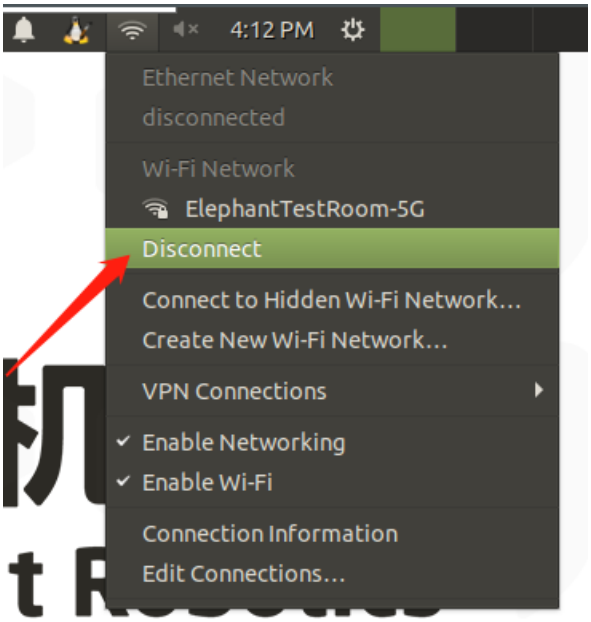
2.5 Network configuration

- **Default AP usage**
  - Power on the robot，by default, the system will connect to the hotspot generated by the PI itself, the hotspot name is**ElephantRobotics_AP_XXXX**，current IP address**10.42.0.1**，this hot spot does not have the function of web surfing, and the transmission rate and information is limited, so there will be some

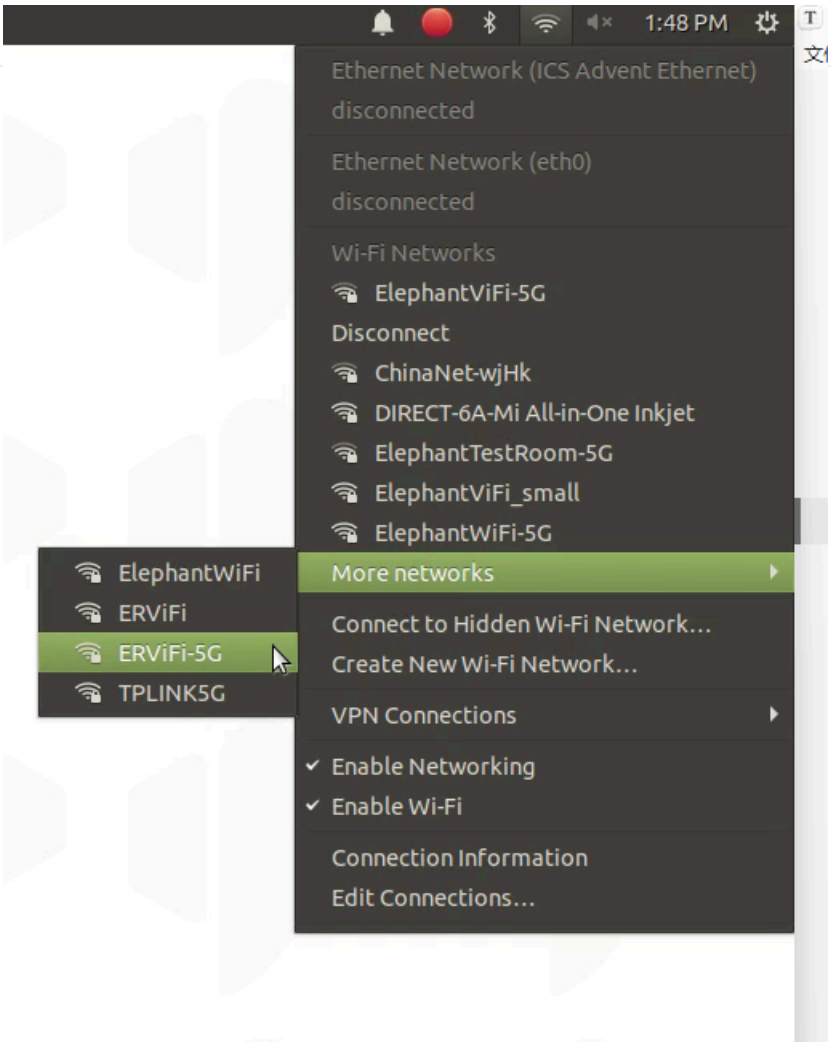
distortion and color difference in the final imaging, and there will be a delay in communication transmission, which is a normal phenomenon

- **Connect to WLAN**

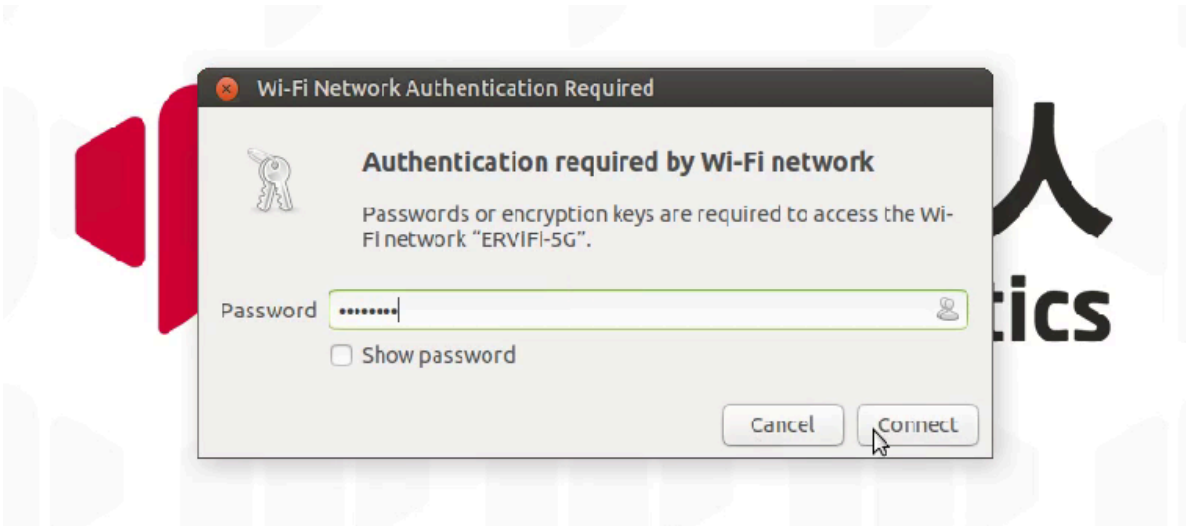
  Click **"Disconnect"** to turn off default hotspot



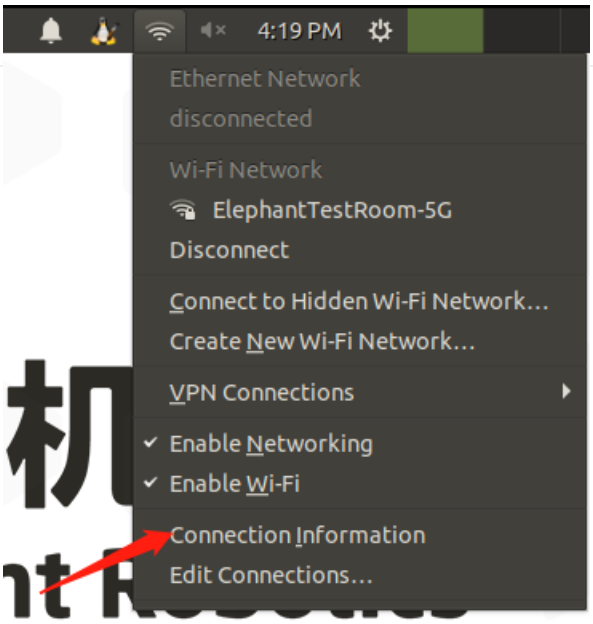
  Click **"Enable Wi-Fi"**， wait for the currently available WiFi to be displayed



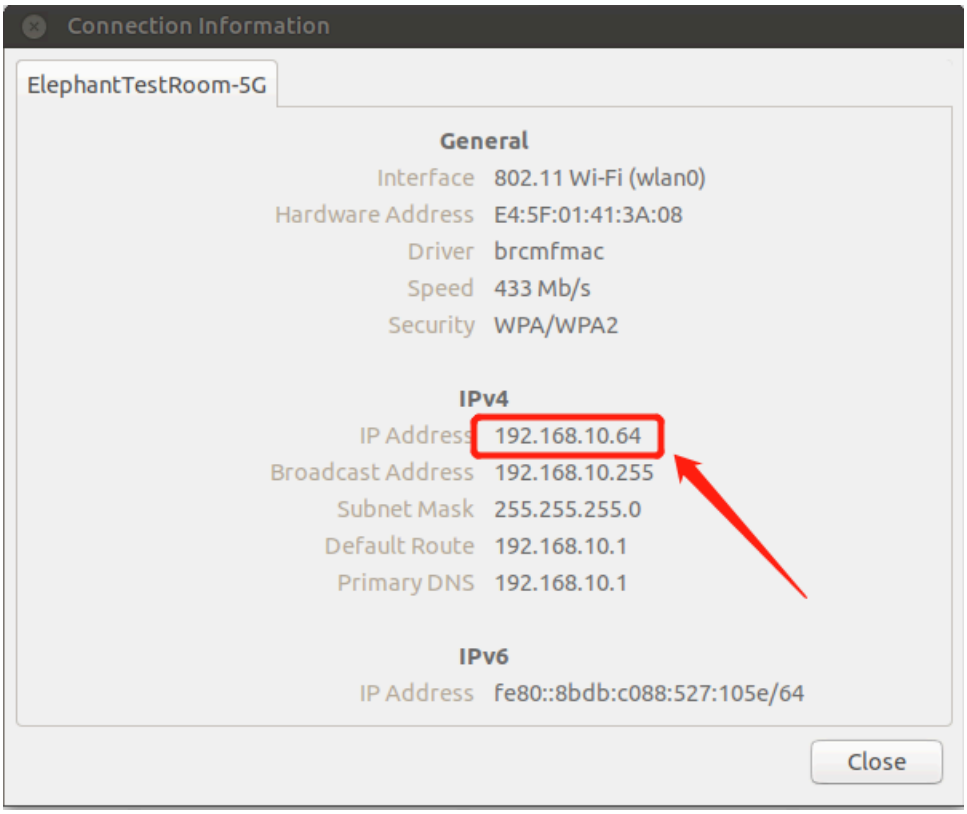
Click the WiFi you want to connect to and enter the WiFi password



After connect successfully，click **"Connection Information"** to check IP address

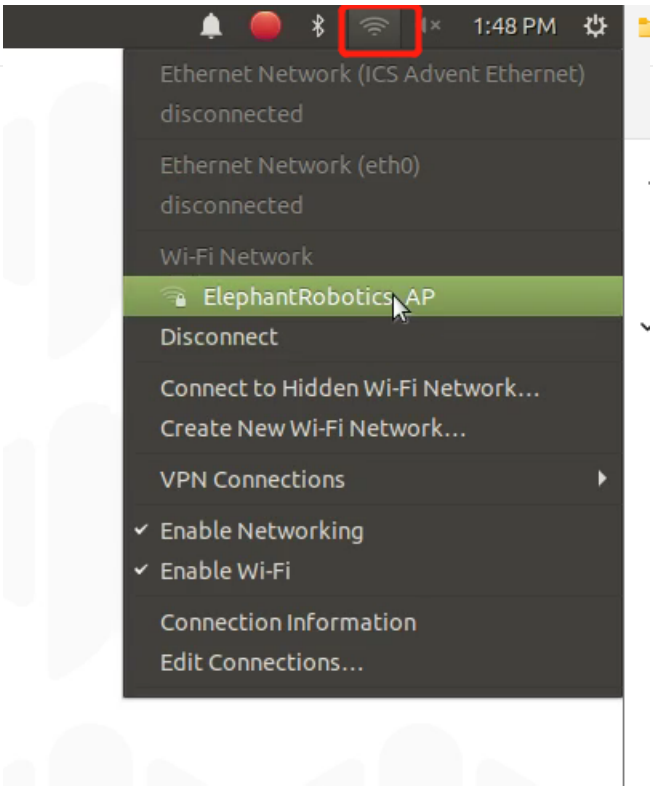


As shown in the example，**“192.168.10.64”**, It is the current IP address of the robot

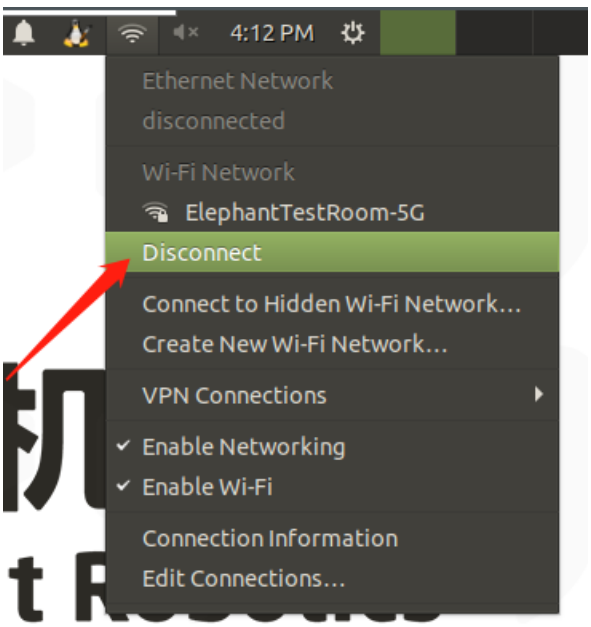


- **Wired network connection**

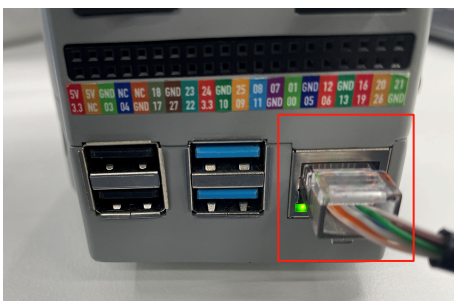
  Power on the robot, it is connected to the hotspot configured by the system by default: **ElephantRobotics_AP_XXXX**



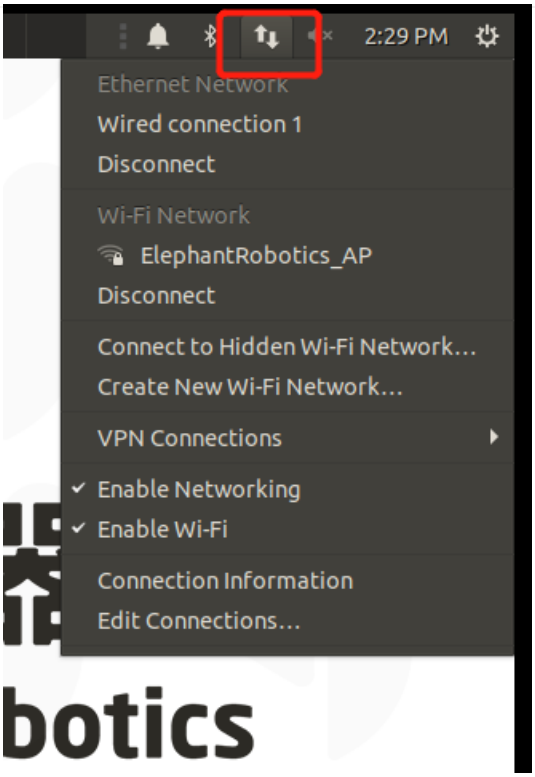
Click **“Disconnect”**，disconnect from the default hotspot



Connect the network cable to the network port of the robot

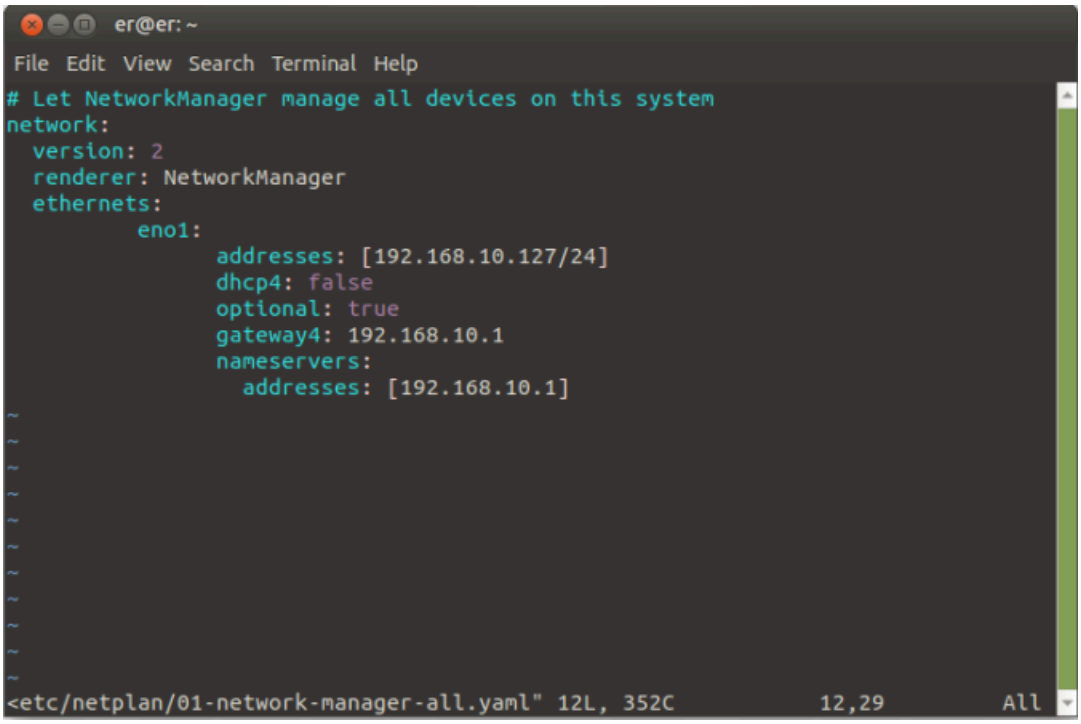


Connect the Internet cable to the network port of the robot



- **How to set default IP address**

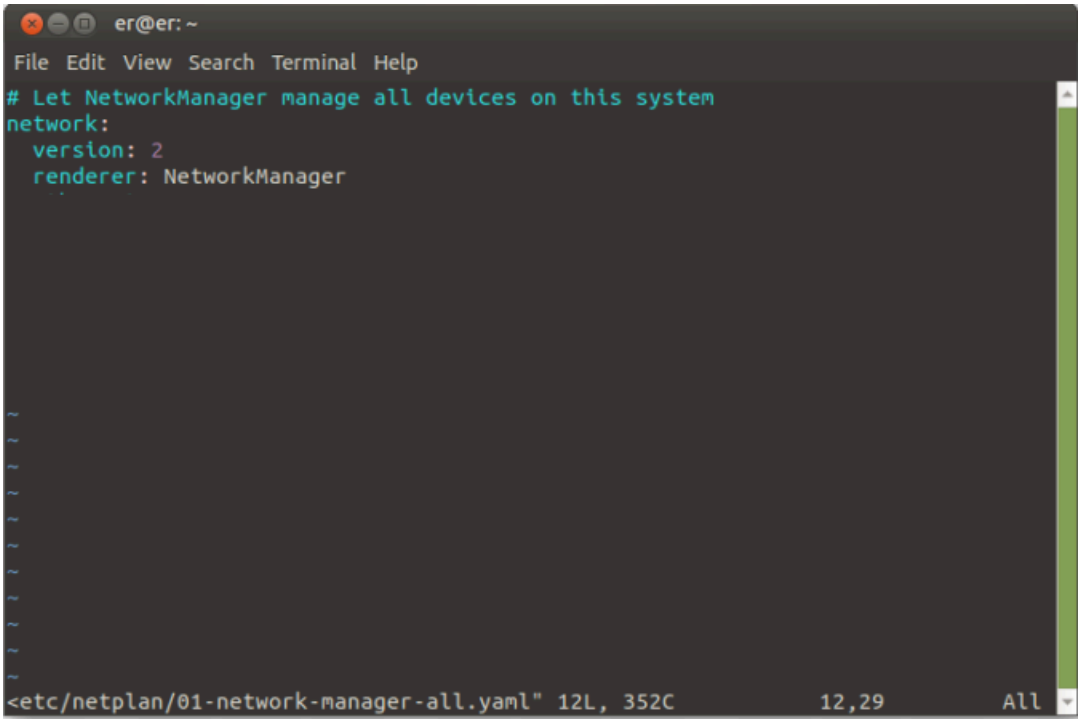
  Use shortcut keys `ctrl + alt + T` to open the terminal，enter `sudo vim /etc/netplan/01-network-manager-all.yaml`，modify 01-network-manager-all.yaml file as follows:



  After modification，enter `sudo netplan apply` to make the configuration take effect

- **How to assign IP addresses automatically**

When the system is connected to WiFi, the IP address is automatically assigned, If you want to change the fixed IP address to automatically assign an IP address, modify /etc/netplan/01-network-manager-all.yaml file as follows:



After modification，enter `sudo netplan apply` to make the configuration take effect

2.6 BT configuration

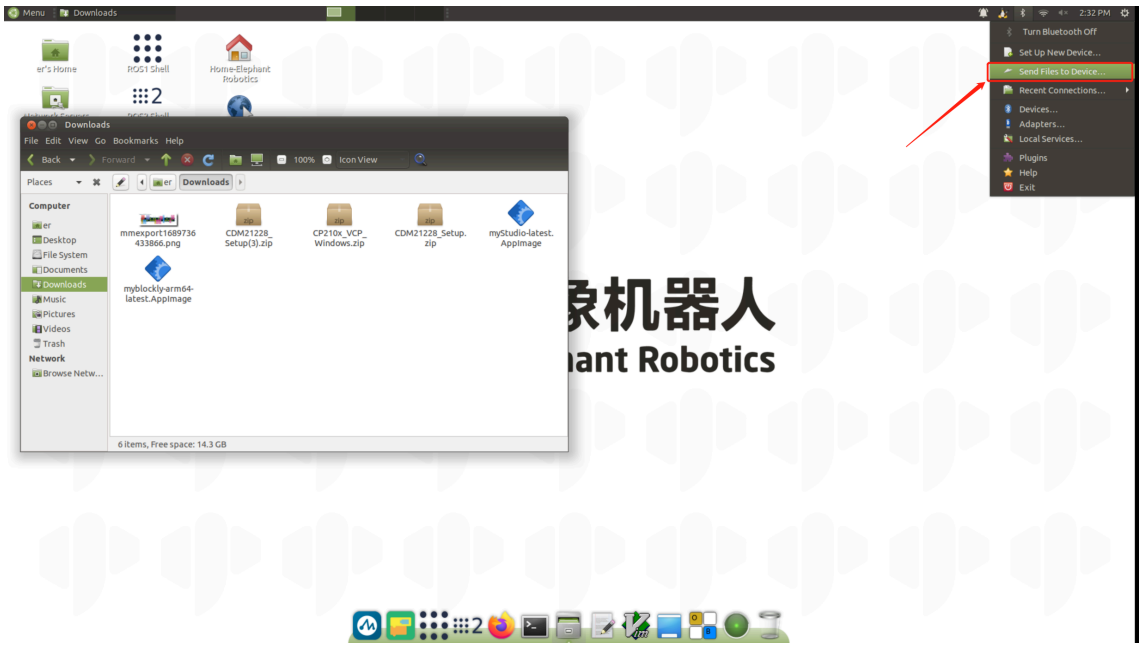
- The Bluetooth of the system is turned on by default. Use the PC/Phone directly , and you can search the robot's BT. The default name of the BT is **MyCobot-Pi**

- Transfers files from PC/Phone to the robot system
  - Select the file you want to transfer
  - Operate in the robot system and click **Accept** to receive file

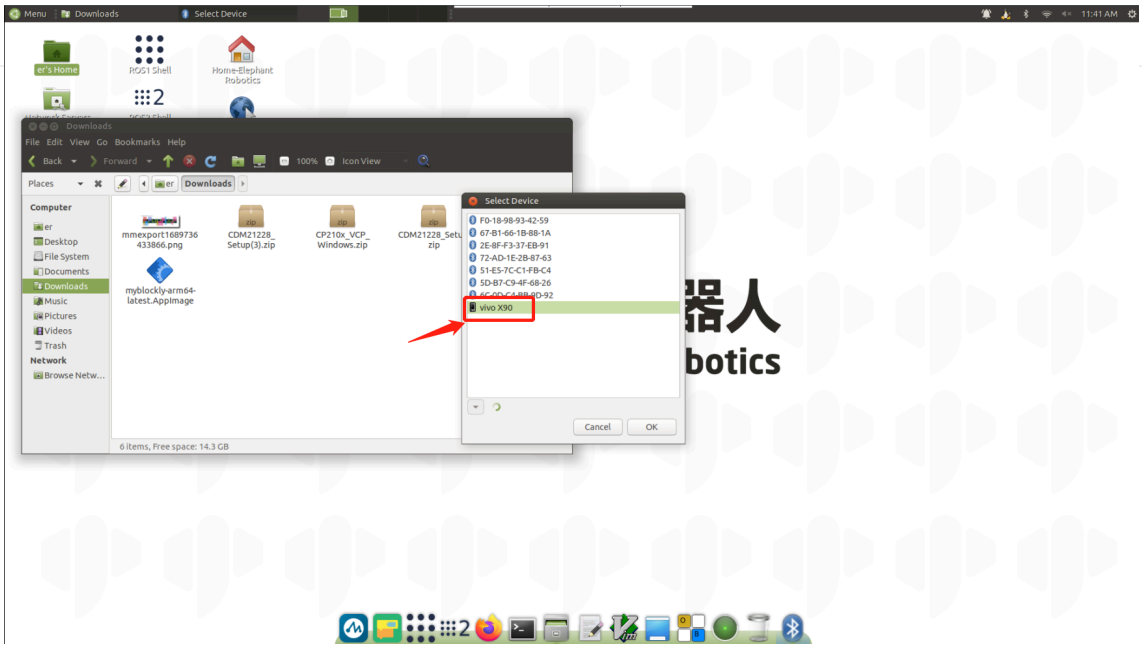
- Wait for the BT transfer to complete

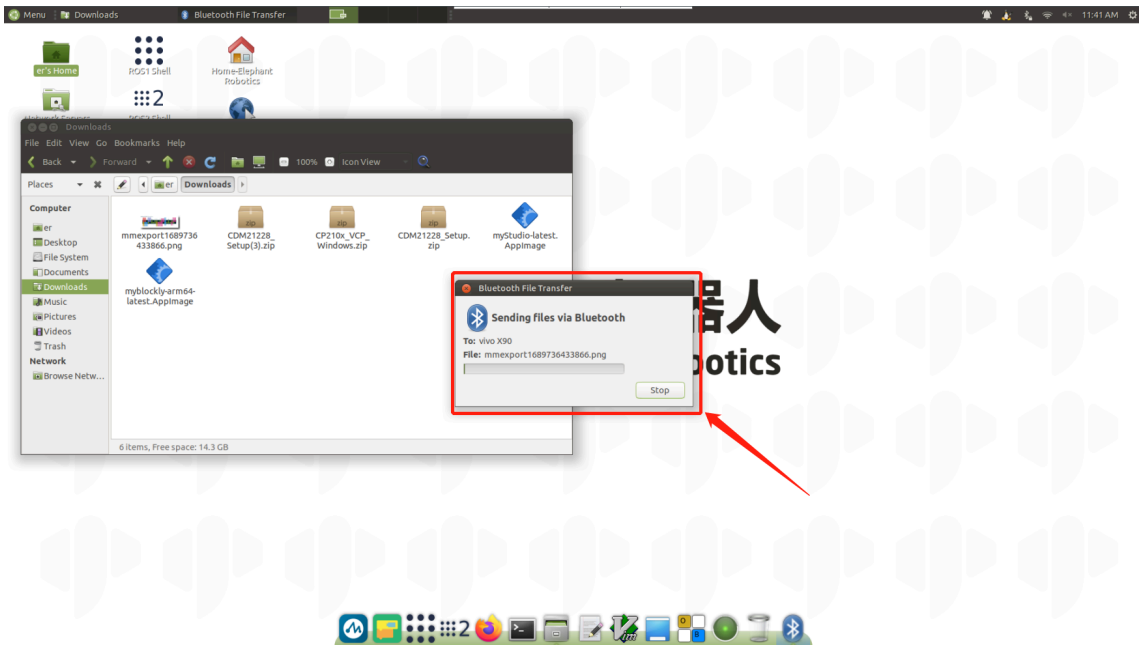
- Can check received files in **/home/er/Downloads** folder

- Transfers files from the robot system to PC/Phone
  - Click BT icon，select **Send Files to Device**



  - Choose PC/Phone





- On PC/ mobile phone, allow the device to receive files

2.7 Language configuration

- **How to change language**

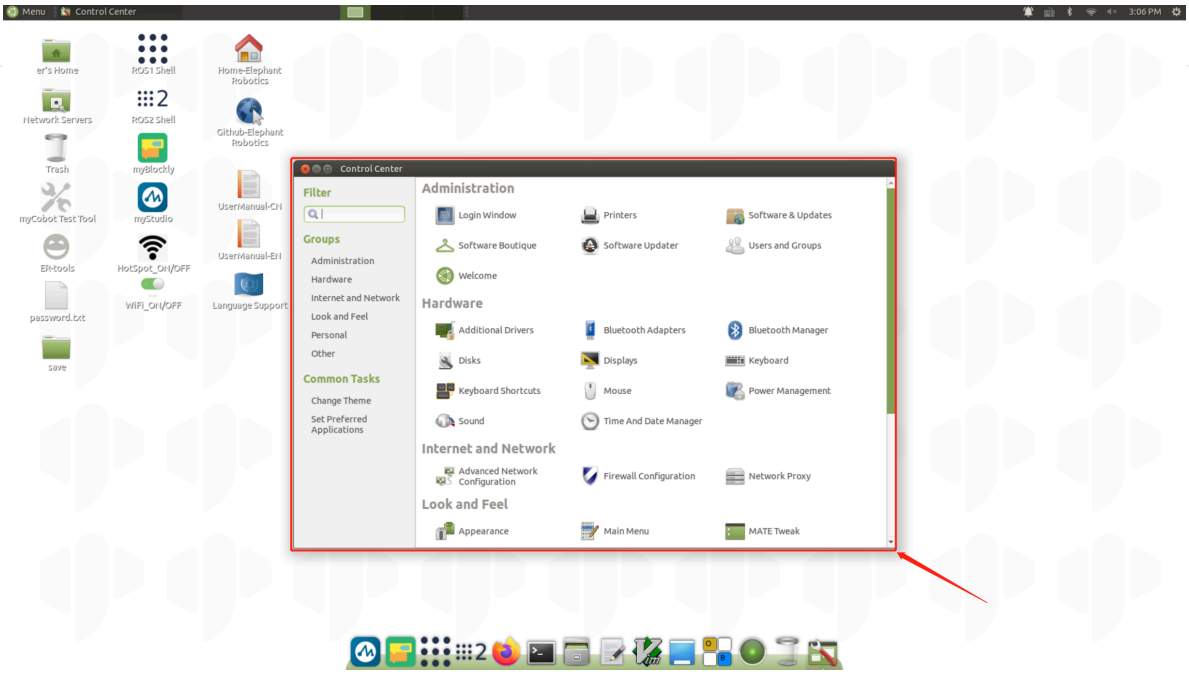
  Click **Language Support**，drag the language you want to the top and restart the system

- **How to download language**

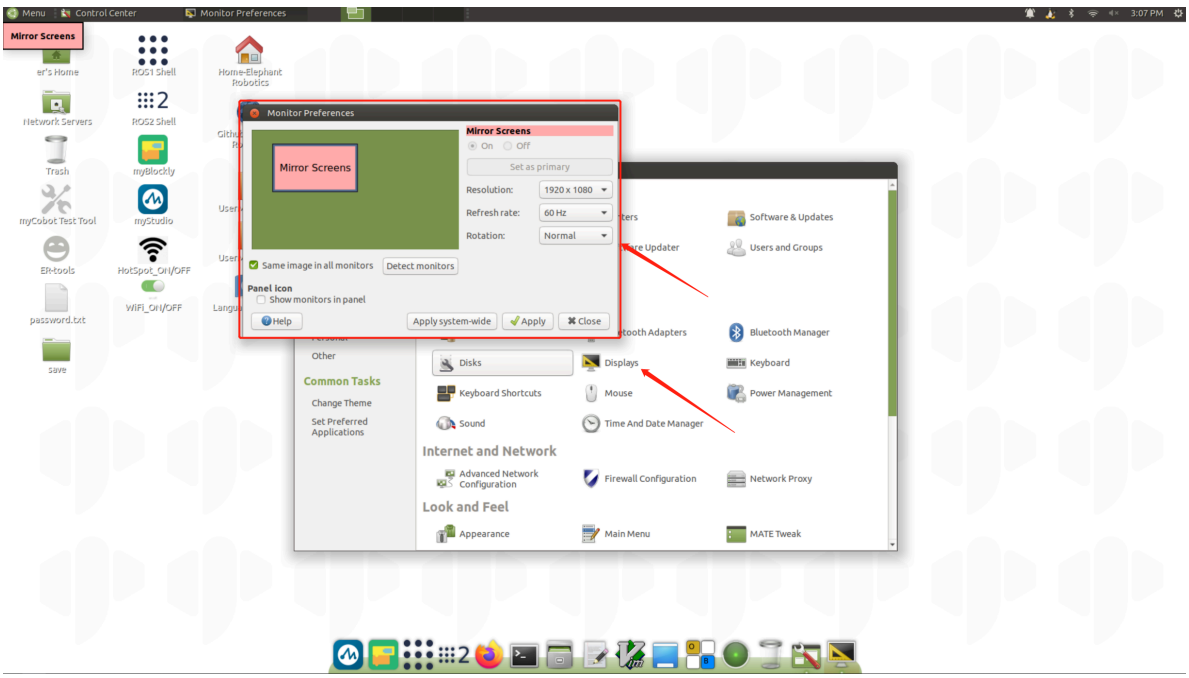
  Click **Language Support**，choose language，click to download，enter password **Elephant**

2.8 System resolution switch

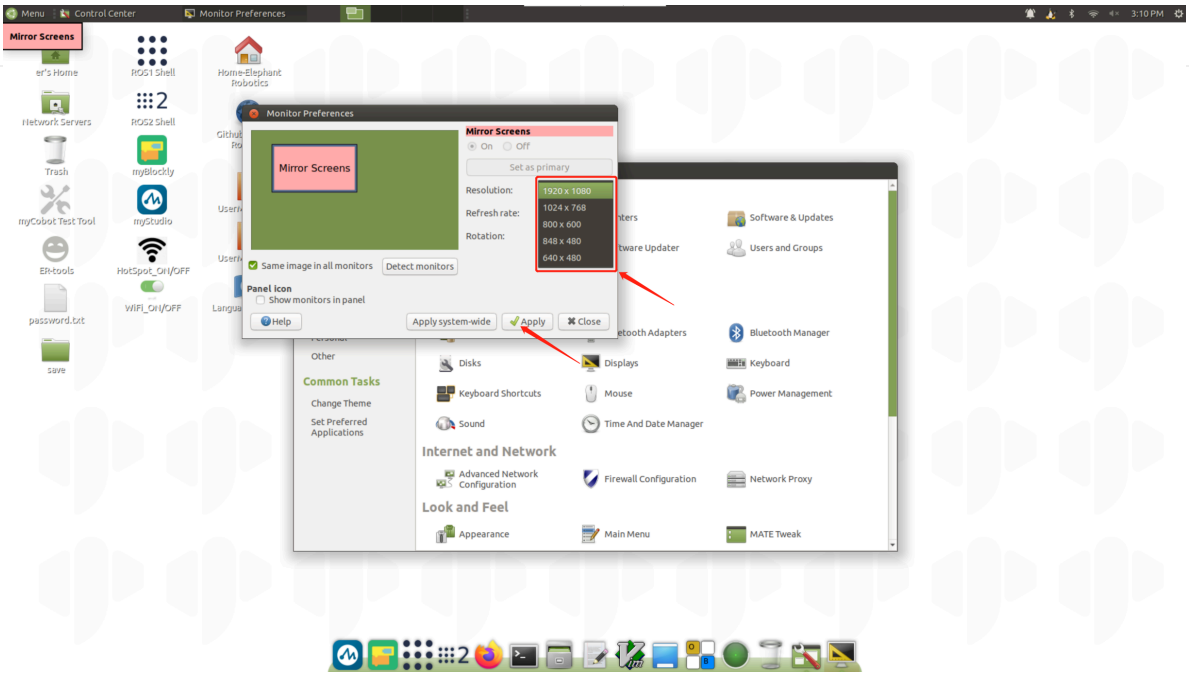
- Click the icon in the upper right corner of the screen and select **System Settings** to enter the control panel

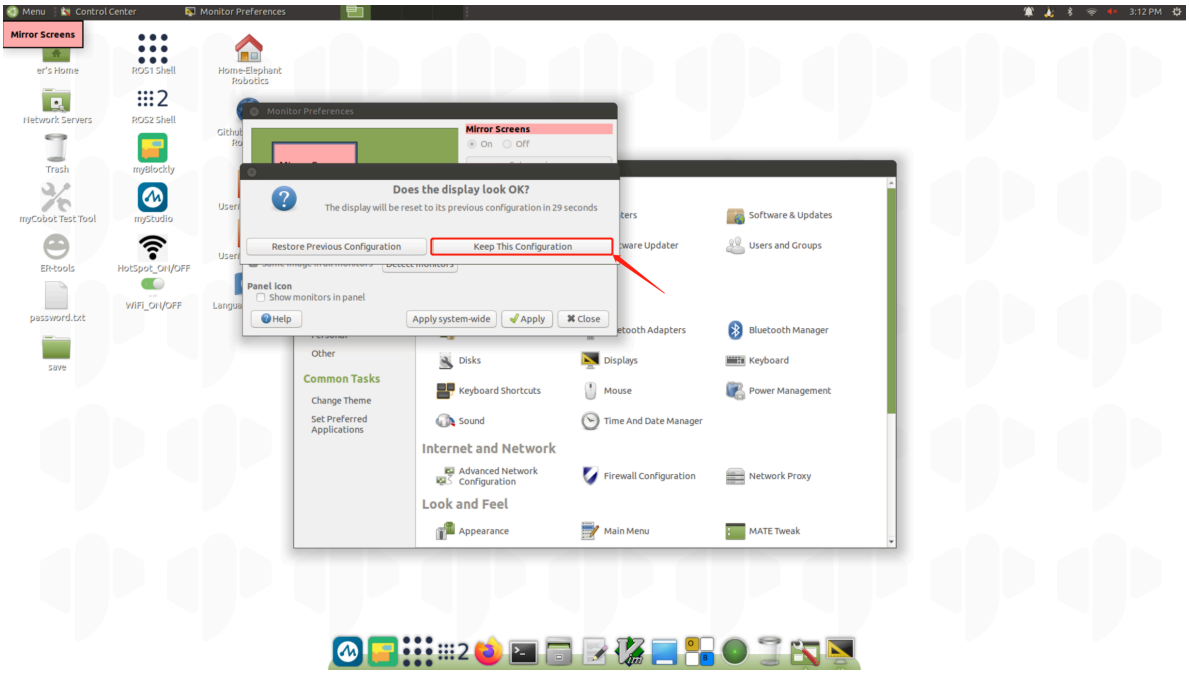


- Choose **Display**



- Toggle the selected resolution and click **Apply** to check and click **Keep this Configuration** to save the configuration





2.9 python

- **Python introduction**

  Built-in installation **Python3.8**, no need to install it yourself

  Installed libraries：

Package	Version
pymycobot	3.1.5
pyserial	3.5
numpy	1.23.5
opencv-contrib-python	4.7.0.72
rospkg	1.4.0
rospkg-modules	1.4.0

- **Try to program**

  If you are new to the python programming language, you can follow the following video to program

You can try this code in the following input fields:

```
print ("Hello World!")
```

- **Run demo**

  Specific case codes can be viewed Python，just copy the code in the cases and use it

- **Corresponding chapter case**

  1 Joint control

  2 Coord control

  3 IO control

  4 Gripper control

  5 TCP/IP control

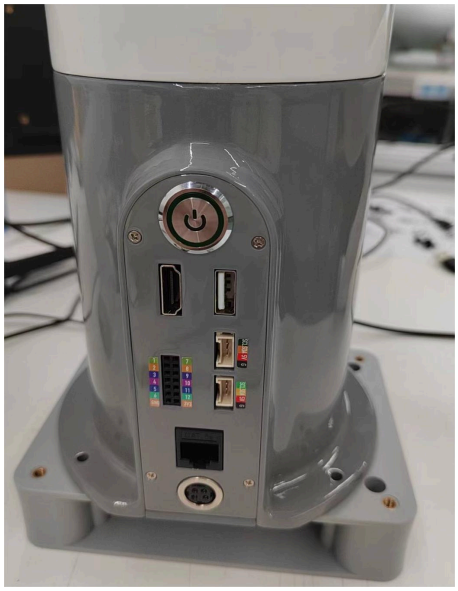
  6 Handle control

  7 Example and videos

myBuddy

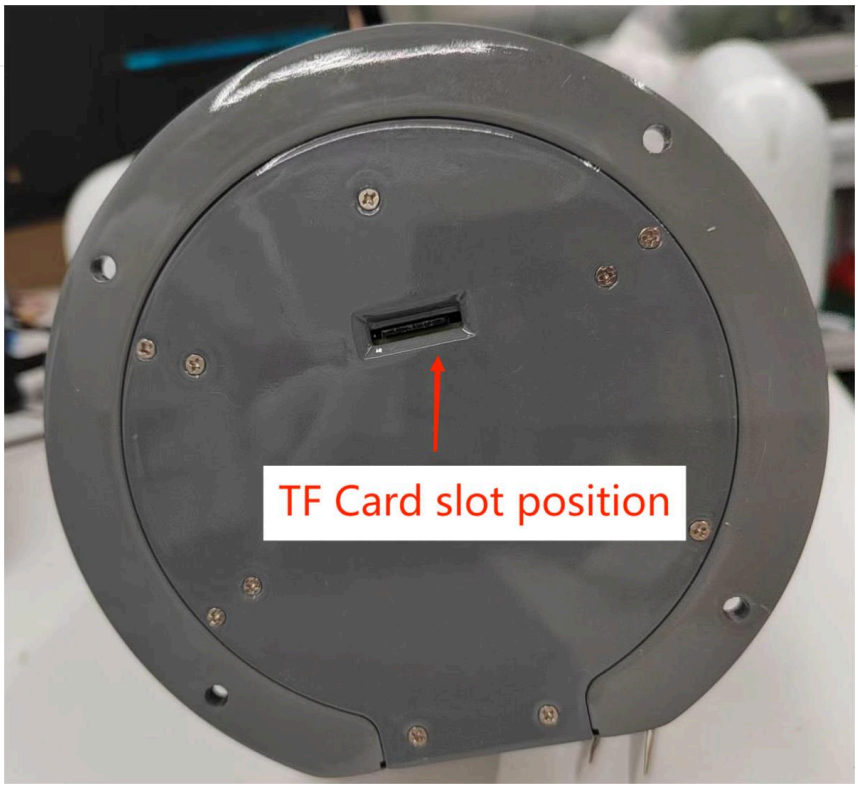
Tutorial on Replacing TF Cards

Front view of equipment base

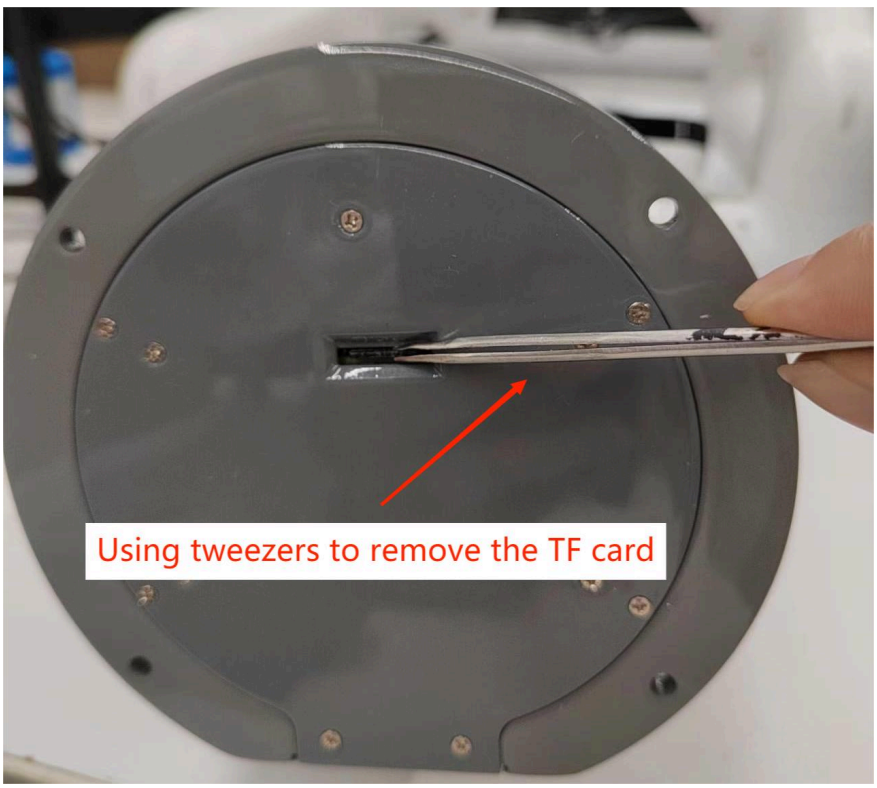


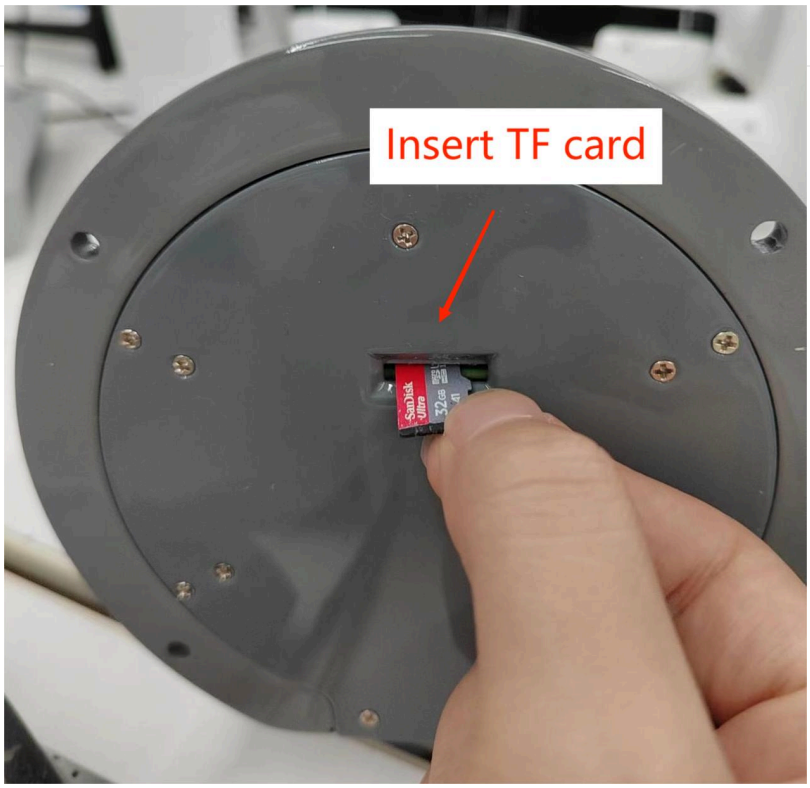
- Step 1: Use an Allen wrench to remove the base(You need to remove the base to see the position of the TF card slot)

- Step 2:Check and confirm the position of the TF card slot



- Step 3: Remove the TF card or replace it. (Note that power off operation is required and tweezers are used to remove the TF card to avoid damaging it)




Insert TF card

Downloads

Brochure Download

Robot	Brochure
myCobot 280 M5	download
myCobot 280 PI	download
myCobot 280 AR	download
myCobot 280 JN	download
myPalletizer 260 M5	download
myPalletizer 260 PI	download
mechArm 270 M5	download
mechArm 270 PI	download
myCobot 320 M5	download
myCobot 320 PI	download
myCobot Pro 600	download
myAGV	download
marsCat	download
metaCat	download
人工智能套装 2023	download
myarm 300PI	download

3D model files

3D model files of robots

Robot	3D model file
myPalletizer 260	download
mechArm 270	download
myCobot 280 M5	download
myCobot 280 PI	download
myCobot 280 Arduino	download
myCobot 280 JN 2023 ver.	download
myArm 300	download
myCobot 320 M5 2020 ver.	download
myCobot 320 M5 2022 ver.	download
myCobot 320 M5 2022 ver. v1.2	download
myCobot 320 PI 2022 ver.	download
myCobot 320 PI 2022 ver. v1.2	download
myCobot Pro 600	download
myBuddy 280	download
myAGV	download
myAGV 2023	download

Robot 2D drawing

Robot	Robot 2D drawing
myPalletizer 260 Atom	download
myPalletizer 260 PI base	download
myPalletizer 260 M5 base	download
mechArm 270 Atom	download
mechArm 270 PI base	download
mechArm 270 M5 base	download
myCobot 280 Atom	download
myCobot 280 X3pi base	download
myCobot 280 M5 base	download
myCobot 280 JN base	download
myArm 300 Atom	download
myArm 300 base	download
myArm 300 joint	download
myCobot 320 Atom	download
myCobot 320 PI base	download
myCobot 320 M5 base	download
myCobot Pro 600 Atom	download
myCobot Pro 600 base	download
myAGV 2023	download

3D models of accessories

- myCobot series

Accessories	3D model file
Adaptive gripper	download
Parallel gripper	download
Flexible gripper	download
G-base 2.0	download
Large suction cup base	download
Flat base	download
Vertical suction pump	download
Double suction pump	download
Camera flange	download
Asparagus flange	download
Dexterous hands	download
Pen holder	download
Phone holder	download

- myCobot Pro series

Accessories	3D model file
Eletric gripper	download
Pneumatic gripper	download
Flexible gripper	download
Module suction cup	download
Adaptive gripper	download

Image Download

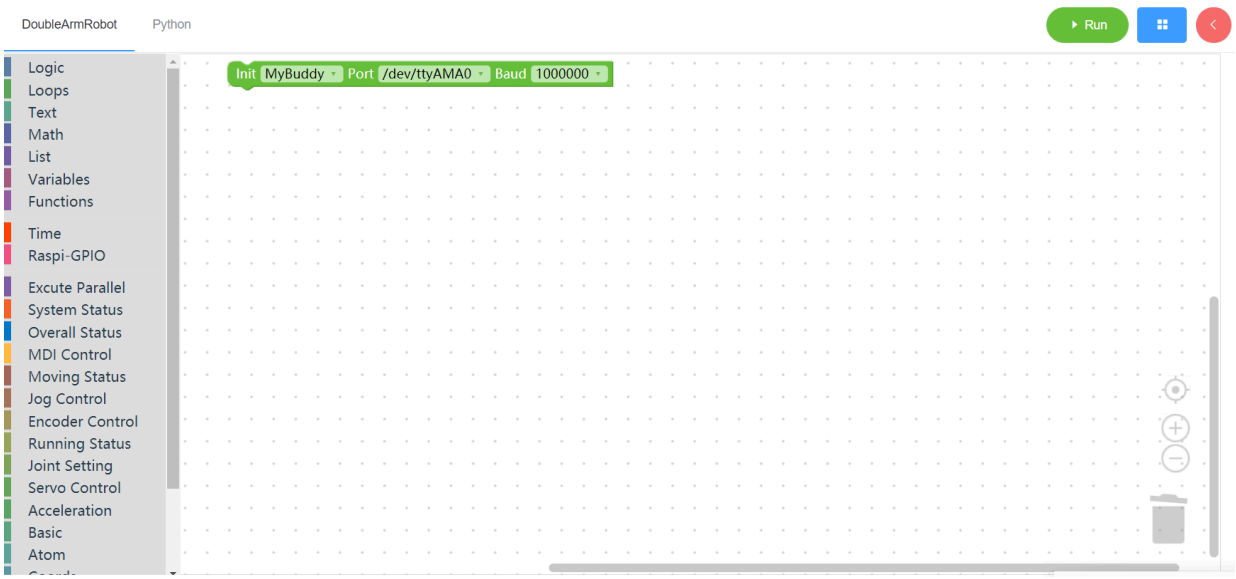
Product	Syetem Version	Link	SHA256 Hash
AI Kit 280	ubuntu 18.04	download	d44439be351a52decdb4470cb623a032047e223ffce73477d29aa9
myCobot 280 PI	ubuntu 18.04	download	04e40af5b637ec003a8b23ef9012e353361fd336db4e17cf9a65feb
	ubuntu 20.04	download	ce666e6c1047c512fe6b270336d472e48f231be12808729ed57f74
myCobot 280 JetsonNano	ubuntu 18.04	download	2f1e40c1480b077bcc83abd3b79ac175f25d21e9cc344a01463616
	ubuntu 20.04	download	8c6f03d2439a2dbdaf1dd689da903ba6b8c5d0665c395954d27bcb
myCobot 320 PI	ubuntu 18.04	download	bc2ed6ef8d51a885f45379392b71e35420638a427d5b4b3a3c9d18
	ubuntu 20.04	download	c95633bfd49246254f2be4783c6a91a15212422219157962c93125
myPalletizer 260	ubuntu 18.04	download	f6fe999519146428e4c60960b242f647ae5c73c704852d686b2858
	ubuntu 20.04	download	04ebbad80926bc7b5d94a4ecb6e64a2205babb1564af07dd45266
mechArm 270	ubuntu 18.04	download	9af1fcbf9c608eda269dc395a8d68ea0a270008a88ec8ec3cf97758
	ubuntu 20.04	download	cbdc05cee595e9955f2d2c8c4a8ed13cdb95db125f6be1be6316bd
myCobot Pro 600	Raspberry Debian	download	2e73aaa153bddbf0a49d18669a254b27403f17f8e989c05d13836d
myArm 300	ubuntu 20.04	download	1325433fec7ec7d028f468315a832210fd9ce004a6dd07eb859afc7
myBuddy 280	ubuntu 20.04	download	2b5452f665bcb999faf1727b2103dc1e5745705f5706728e140d629
myAGV	ubuntu 18.04	download	bedad7d9769cb69380c6a4b9742ba7aefc21db41ab239172b7a5a

1. What is myblockly?

myblockly is a fully visual modular programming software, belongs to the graphical programming language. **myblockly** is similar in function/design to Scratch, MIT's children's programming language. With **myblockly**, users can build code logic by dragging and dropping modules, much like building blocks. From a user perspective, **myblockly** is an easy-to-use visual tool for generating code. From the developer's point of view, **myblockly** is a text box containing the user-typed code. The process of generating the code to the text box is the process of the user dragging and dropping in the **myblockly**.

myblockly installation

myblockly interface



applicable equipment:

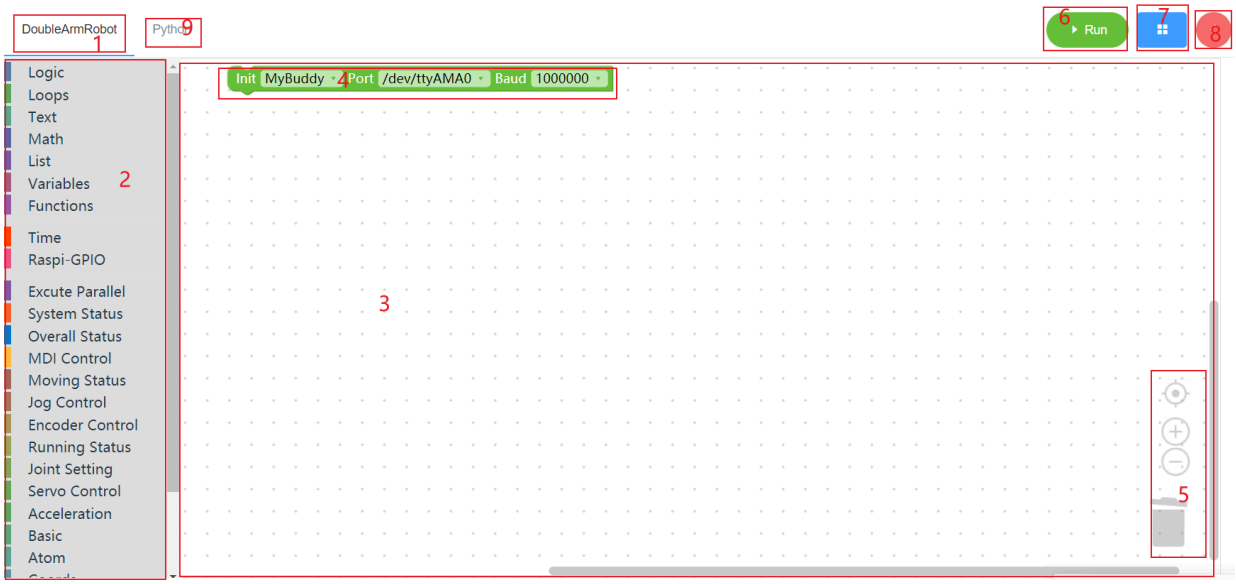
- mybuddy 280

Use the premise

- **Pi** version needs to be added
- Configure the **Python** environment and install the latest **pymycobot** library

2. Basic applicability of myblockly

2.1 myblockly interface introduction

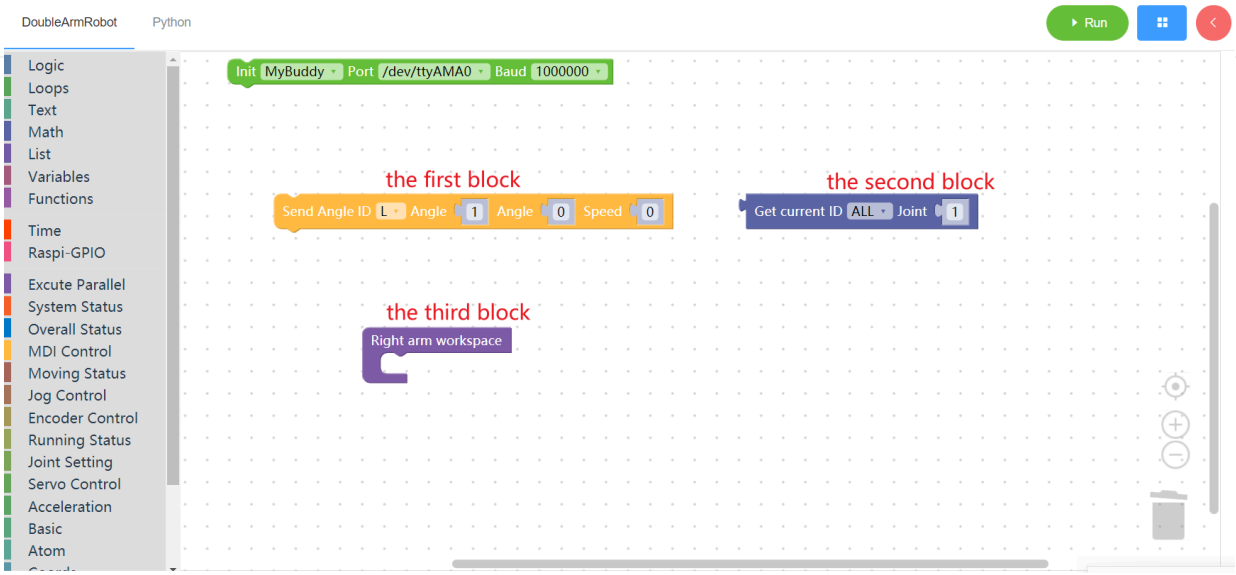


As shown above, myblockly is divided into nine parts:

- **area 1** is the page where the main functions of MyBuddy are located. Users will spend most of their time on this page.
- **area 2** is the toolbar where the building block is located. A corresponding menu can be opened by clicking the label of the toolbar. Users can place the building block in the workspace by dragging the corresponding building block.
- **area 3** is the software workspace, which contains all the building blocks selected by the user.
- **Mark 4** initializes the building blocks for the machines, the default do not delete, click the first drop-down box can switch to other models (temporarily only mybuddy a model), the second drop-down box for a serial port list, click can show on the system after all the serial port, users need to select the initialization of the machine corresponds to a serial port (each system shows the serial port name, The third drop-down box is the baud rate required to initialize the robot. Mybuddy is 115200.

Note: In this software, there are three main shapes of building blocks. The first one is that the joint is in the form of upper and lower buckles, which can be placed arbitrarily according to requirements. The second for the button on the left side of the card, the building blocks in general is the return value type, if you want to put the blocks in the other building blocks, need to make sure that their type is consistent, this within the software already has to deal with, of course, the result of the performance on the user side is if the type is not consistent, cannot be put into the building blocks. The third type is less common. The blocks themselves have no clasp, only internal joints.

Examples of the three blocks are as follows:



- **area 5** is the workspace control module
    - The dustbin is the first one from bottom to top, which stores the blocks deleted by the user. When the user wants to delete a block in the work area, he can drag the block to the dustbin and release the mouse to delete the block, or press del to delete the block. To restore the deleted blocks, click open trash and drag them out.
    - The second button is the shrink button. When the user feels that the workspace block is relatively large, he can click the minus button to shrink the workspace scale, or use the mouse pulley to shrink the workspace
    - The third is the enlarge button, click can enlarge the work area, also can enlarge the operation through the mouse pulley
    - The fourth button is to restore and locate the current workspace. When the user feels that the workspace is too large or too small, he can click this button to restore the workspace to its initial state and locate it to the center of the workspace
    - **Note:** The size of the workspace will affect the blocks in the toolbar, so you need to choose a proper scale to avoid blocks blocking the screen
- **area 6** is the run button. Clicking on the button will convert the user's workspace code into Python code and execute it. At the same time, a pop-up window will print out the running results of the program
- **area 7** For the software setup button, click to save workspace to file, load file to current workspace, set software language, theme, and view current software version information and the version required by PyMycobot
- **area 8** for fast moving toolbar, which contains mybuddy control arm around the waist of the coordinate and Angle, speaking, reading and writing, and the user to select machine, serial port and baud rate (keep pace with the workspace of the initialization data) and then click the open button to connect the robot, and then click add and subtract Numbers coordinates from the Angle of control arm movements. **Note:**The Python button cannot run when Quick Move is enabled because the serial port is occupied. If you need to run Python code, please turn off quick Move first.
- **area 9** is the Python code area. Users can click to view the Python code corresponding to the building blocks in the workspace

3. Building block API introduction

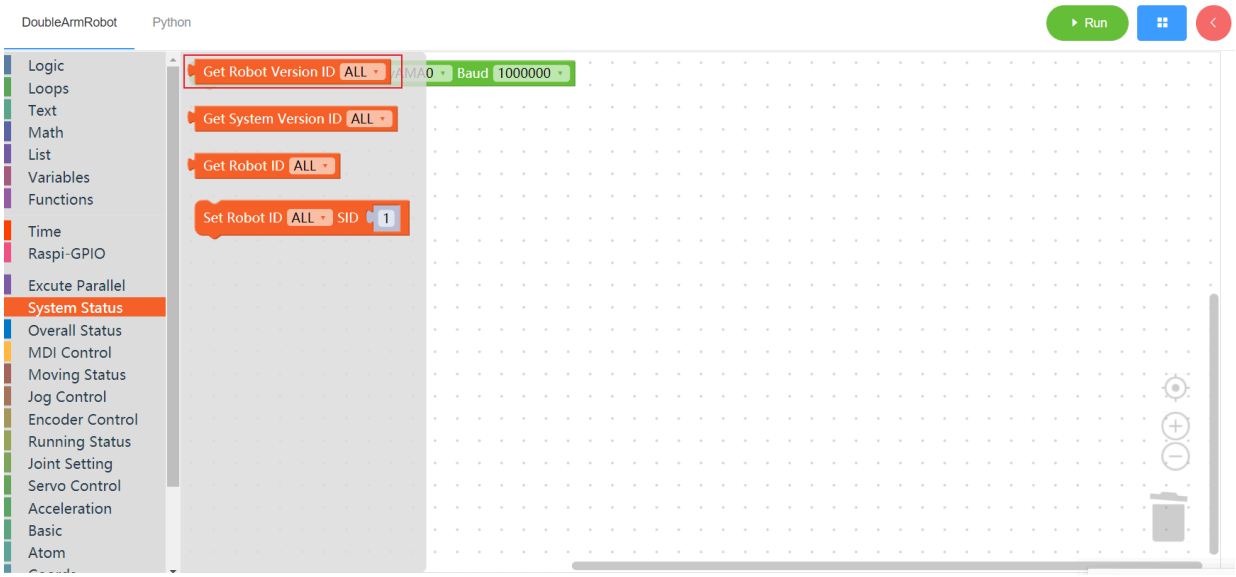
3.1 system information

3.1.1 Get the Robot version

1. Runtime API Reference

- **Function:**Get robot version information
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return** : Robot version information

2. Block display



3.1.2 Checking the Software Version

1. Runtime API Reference

- **Function:**get software version
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**： version information

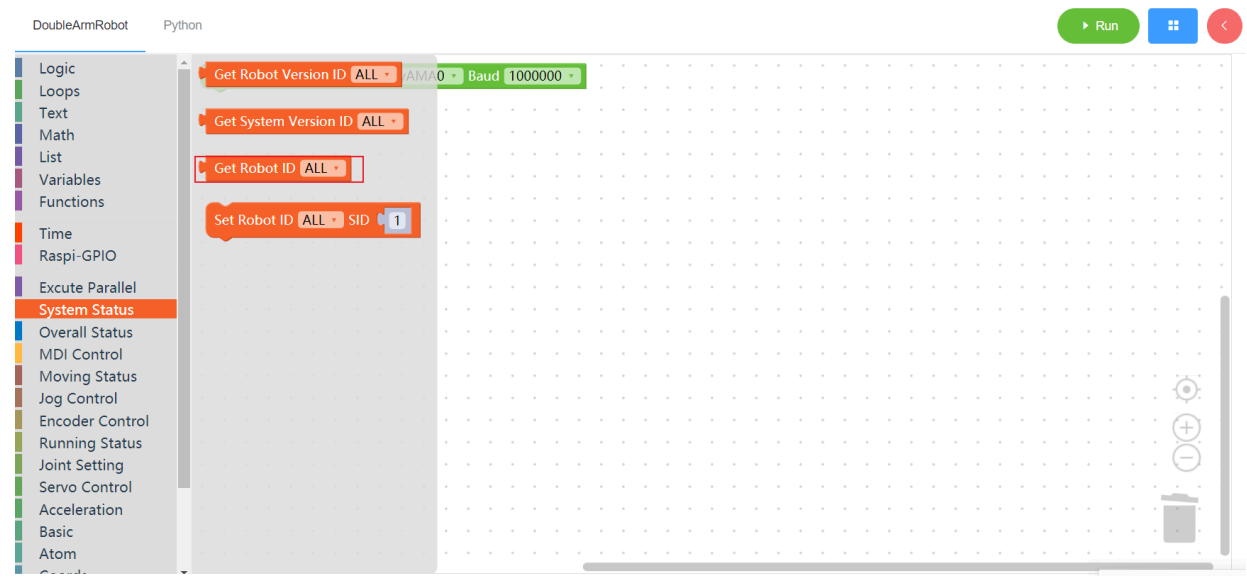
2. Block display

3.1.3 Get Robot ID

1. Runtime API Reference

- **Function:**Get Robot ID
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**： ID

2. Block display

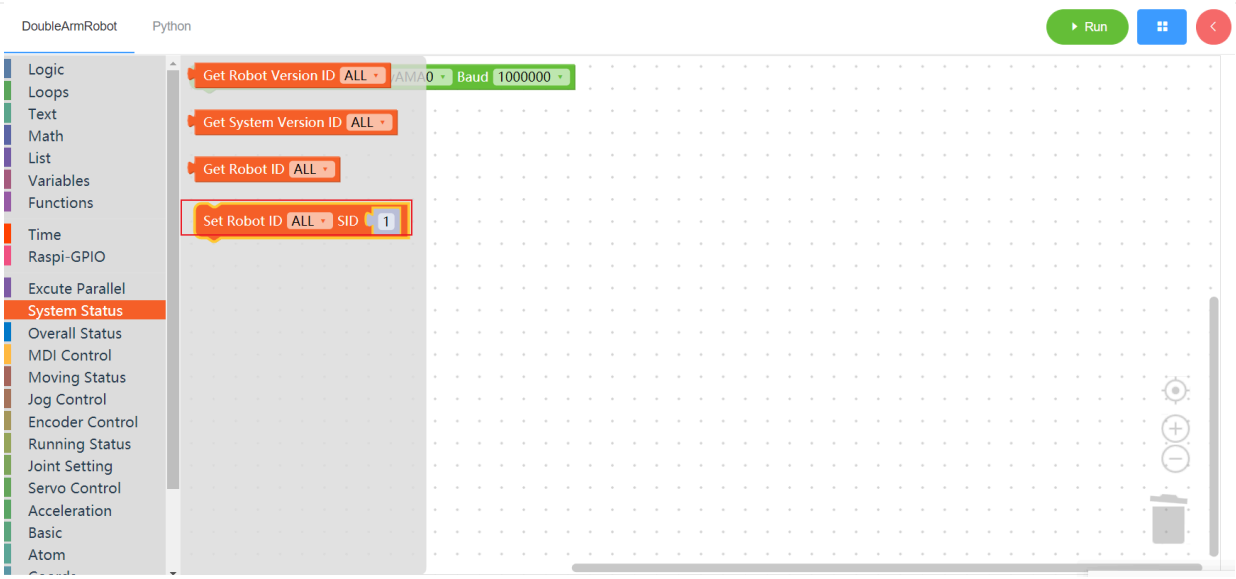


3.1.4 Set Robot ID

1. Runtime API Reference

- **Function:**Set Robot ID
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
  - `NUM` ： integer（1~253）
- **Return**： none

2. Block display



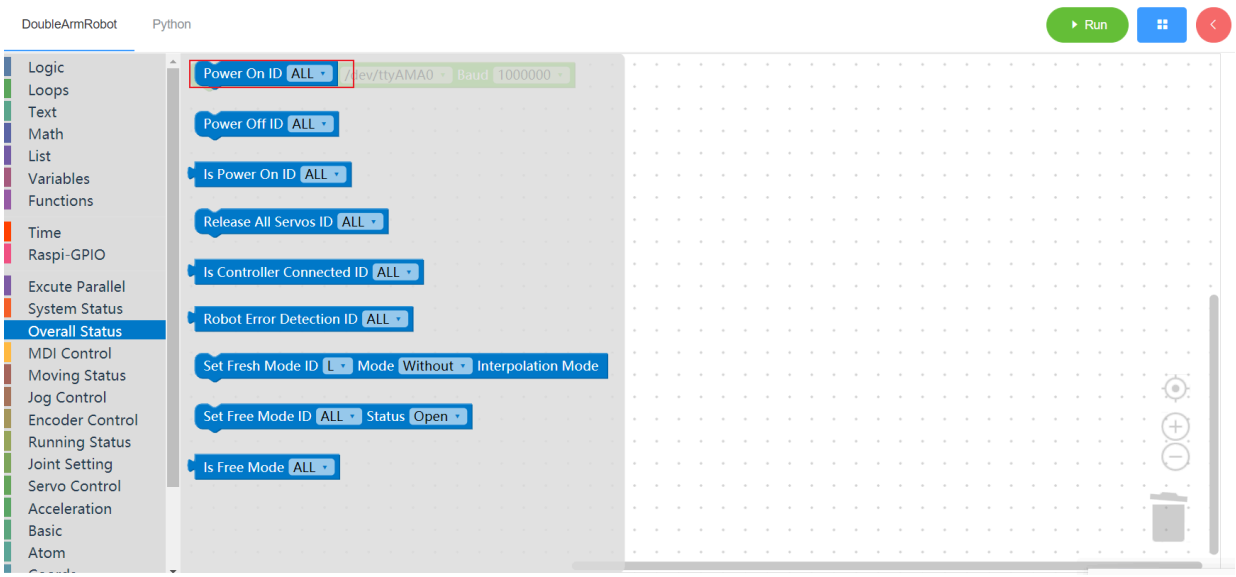
3.2 The overall state of the manipulator

3.2.1 Power on

1. Runtime API Reference

- **Function:**Power on the manipulator
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**： none

2. Block display



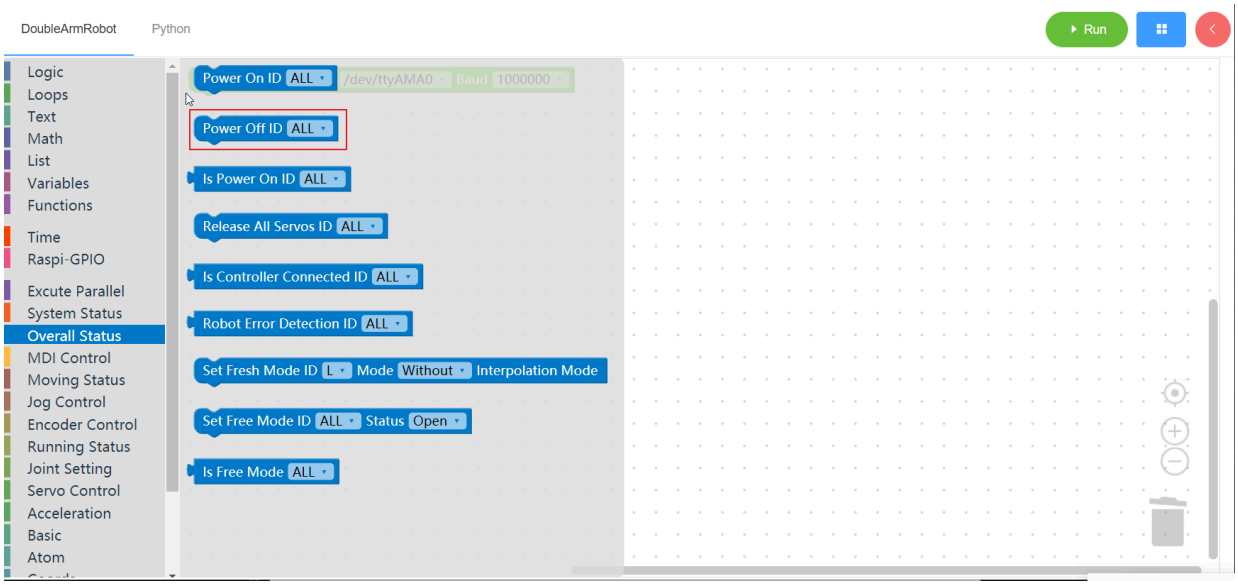
3.2.2 Power iff

1. Runtime API Reference

- **Function:**Power off the manipulator

- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist

2. Block display



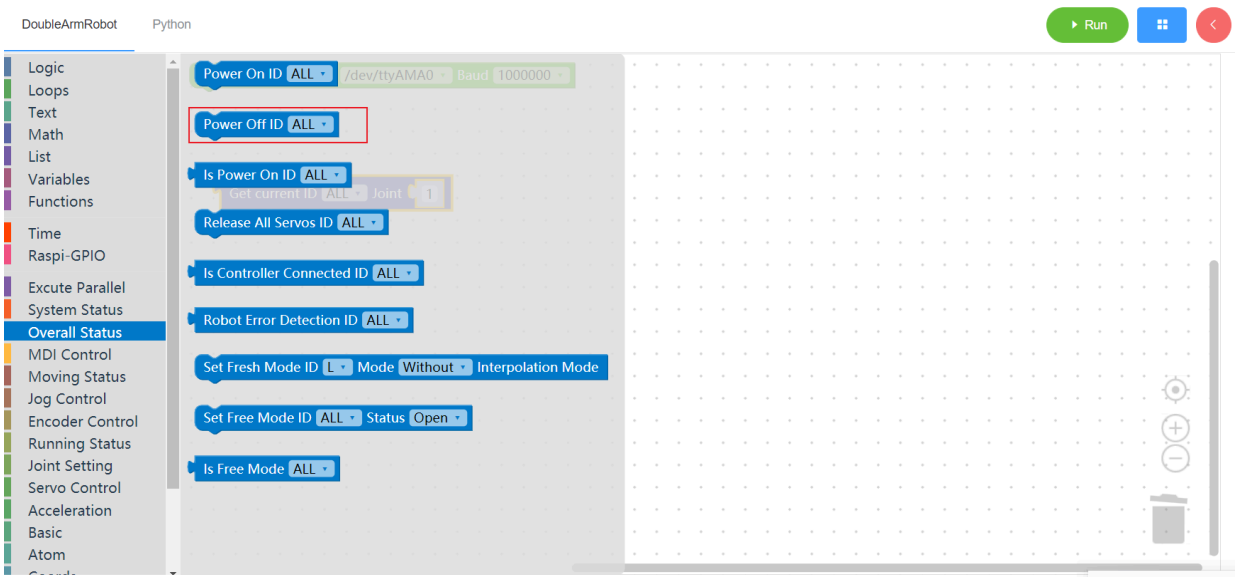
3.2.3 Check whether the mechanical arm is powered on

1. Runtime API Reference

- **Function:**Check whether the mechanical arm is powered on

- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**：
  - 1 power on
  - 0 power down
  - -1 error

2. Block display

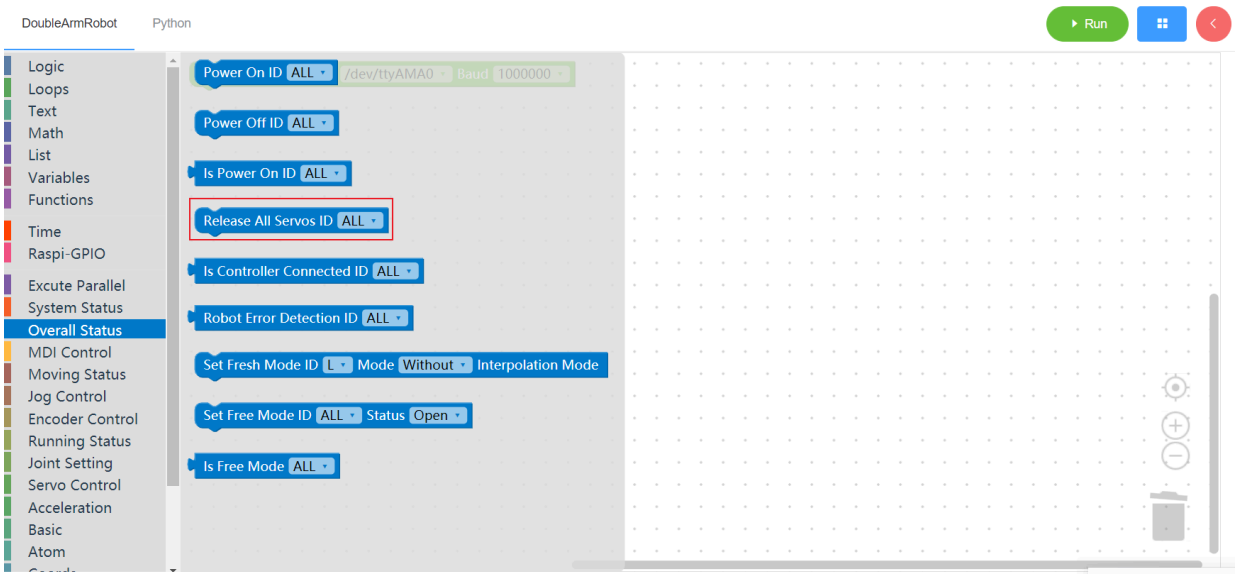


3.2.4 Release all servos

1. Runtime API Reference

- **Function:**Release all servos
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**： none

2. Block display

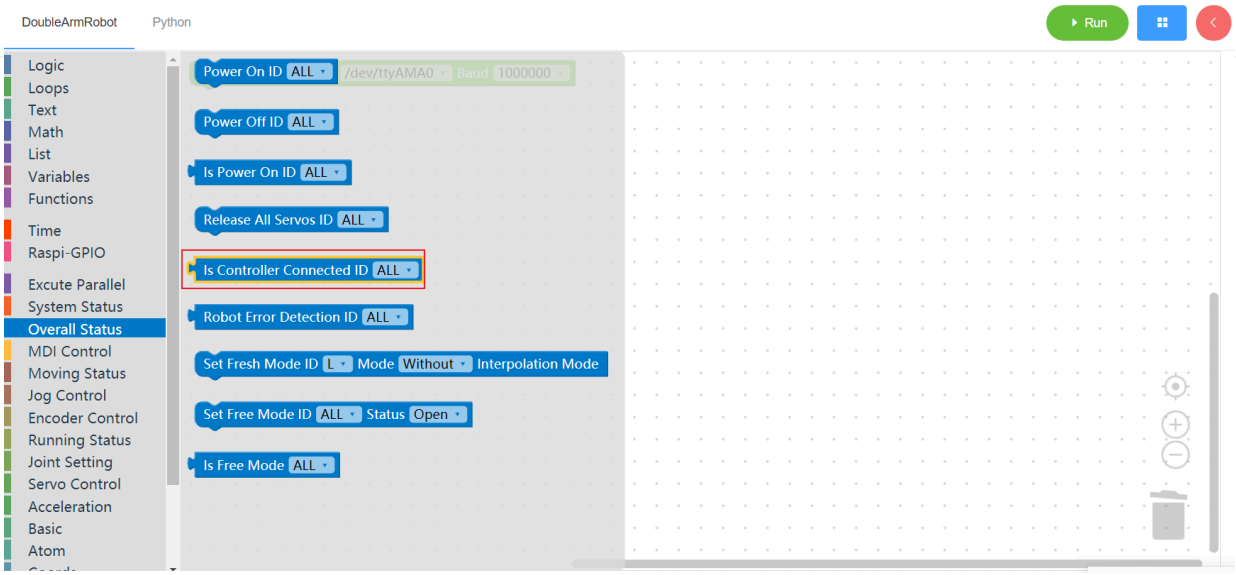


3.2.5 check control connection

1. Runtime API Reference

- **Function:**Check whether the manipulator control is connected
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**： unknown

2. Block display

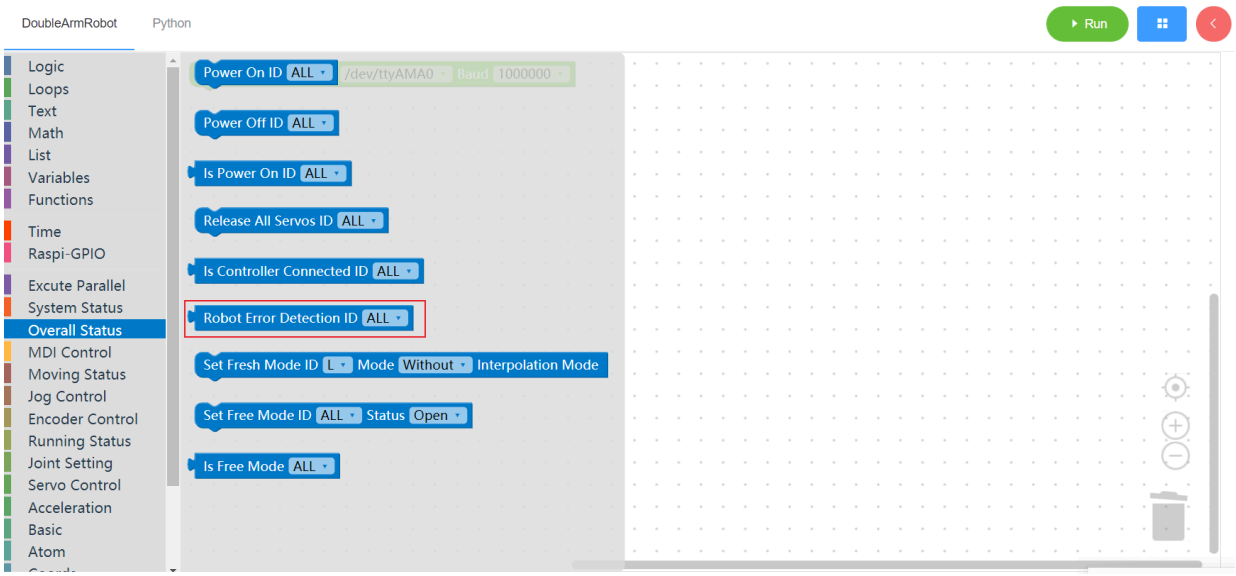


3.2.6 Robot error detection

1. Runtime API Reference

- **Function:**Robot error detection
- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**： none

2. Block display



3.2.7 Set instruction refresh mode

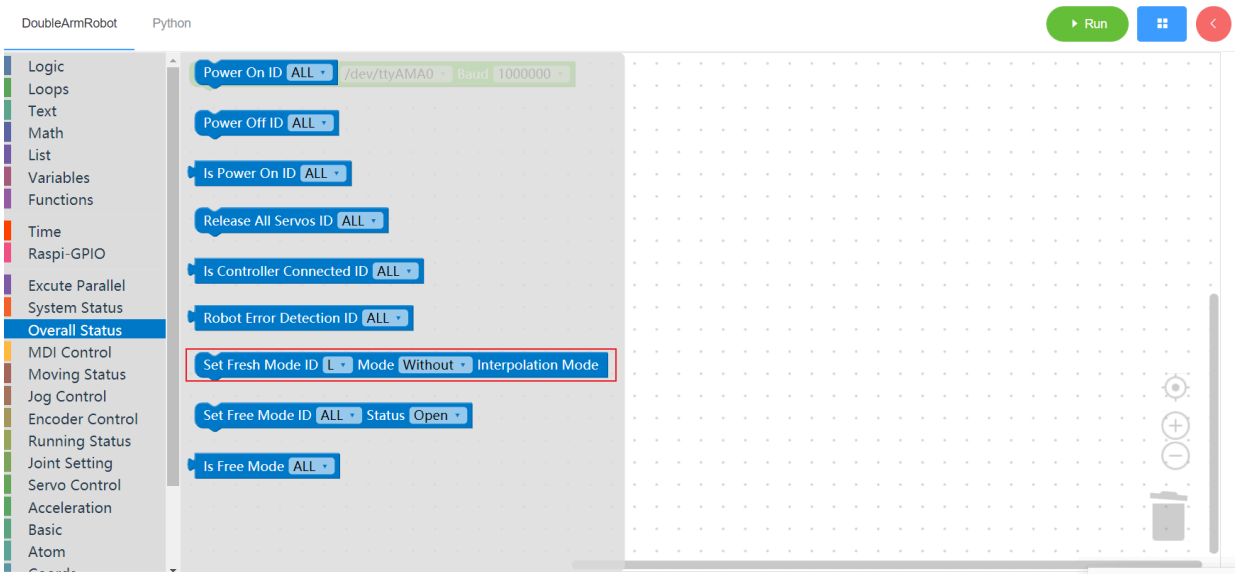
1. Runtime API Reference

- **Function:**Set instruction refresh mode
- **Arguments:**
  - `ID(L/R)` ： L for the left arm, R for the right arm

- `MODE` ： Interpolation mode, non-interpolation mode (default)

- **Return**： none

2. Block display



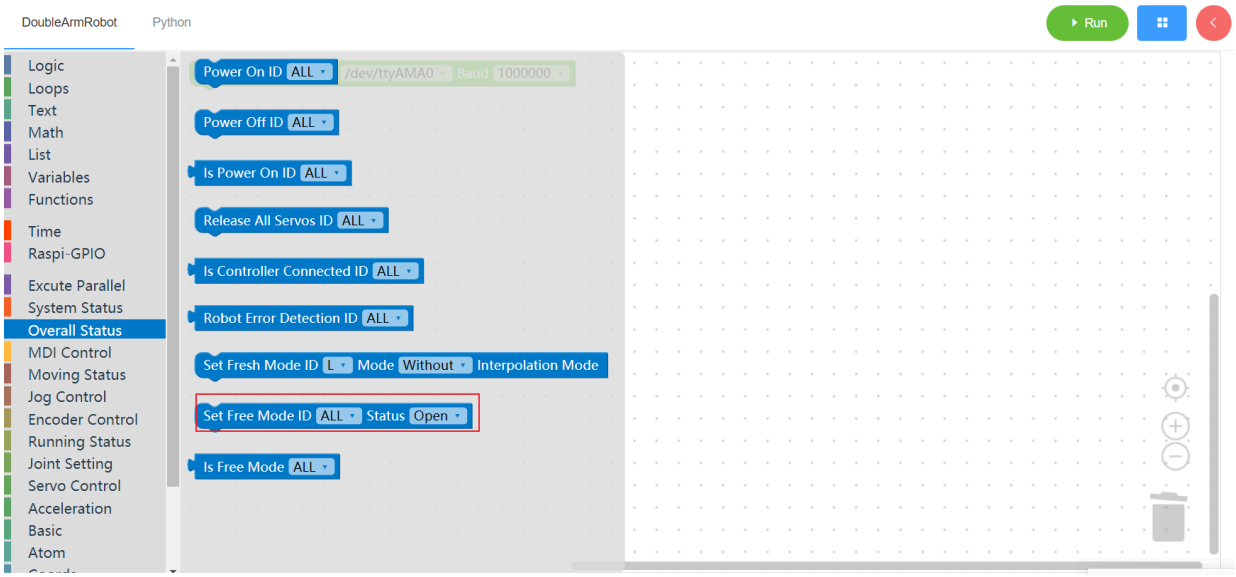
3.2.8 Free mode state

1. Runtime API Reference

- **Function:**Set the free mode state of the manipulator

- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
  - `STATUS` ： status(open/close)
- **Return***： none

2. Block display



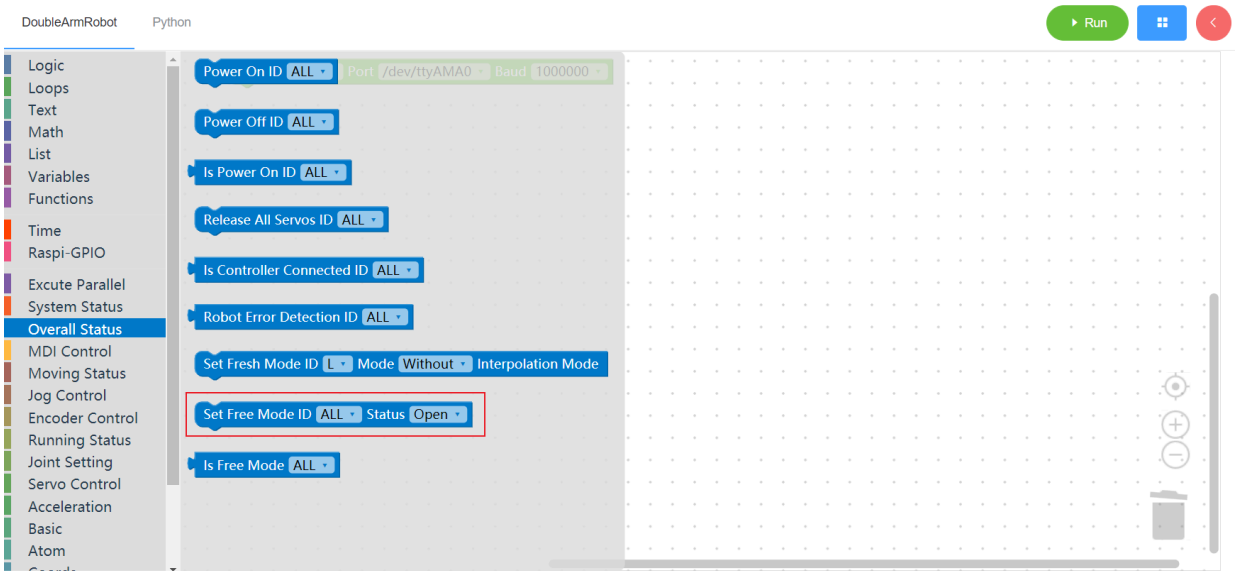
3.2.9 Check whether it is in free mode

1. Runtime API Reference

- **Function:**Check whether the manipulator is in free mode

- **Arguments:**
  - `ID(ALL/L/R/W)` ： ALL for ALL manipulator arms, L for left arm, R for right arm, W for waist
- **Return**：
  - 0 no
  - 1 yes

2. Block display



3.3 MDI control

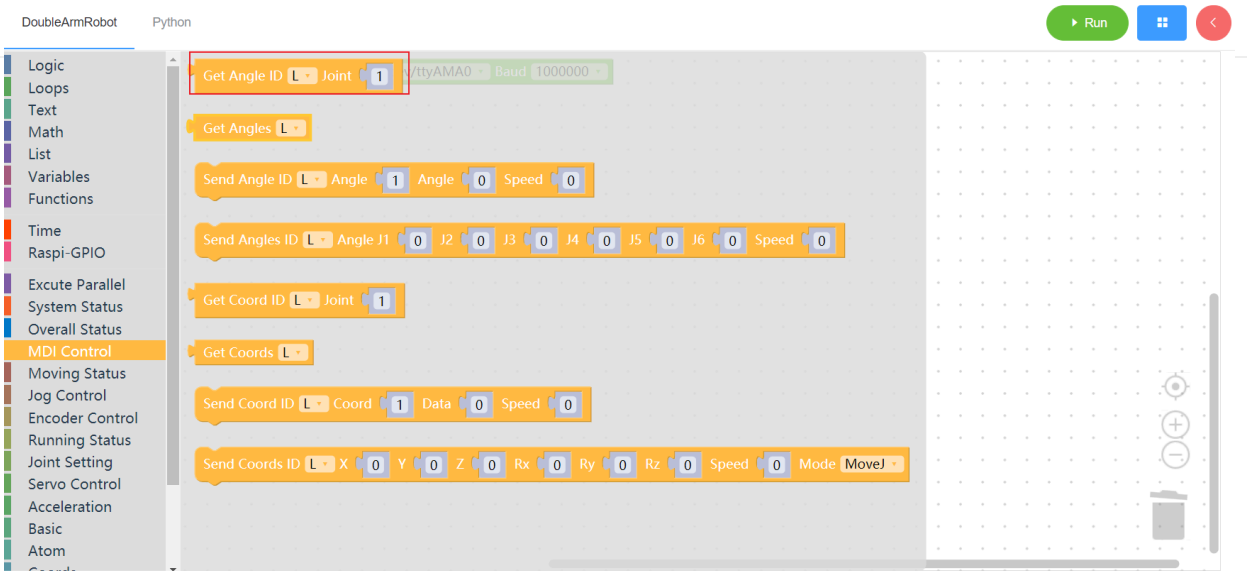
3.3.1 Read single Angle

1. Runtime API Reference

- **Function:**Read single Angle

- **Arguments:**
  - `ID(L/R/W)` ： L for the left arm, R for the right arm，W for waist
  - `JOINT` ： joint id（1~6）
- **Return**： single angle

2. Block display

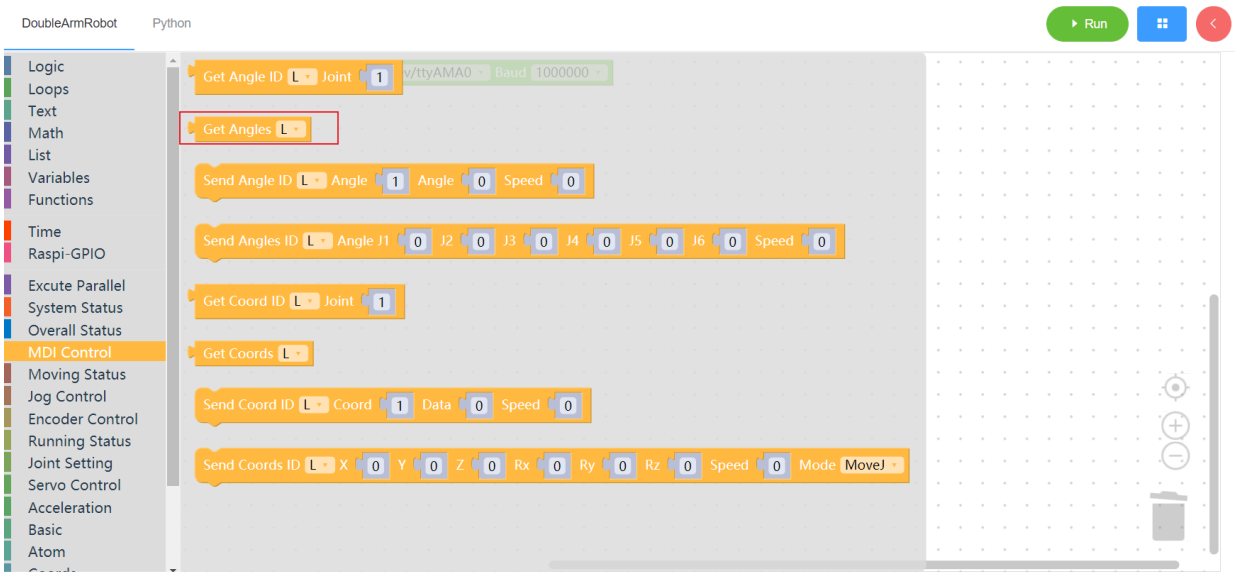


3.3.2 Read all angles

1. Runtime API Reference

- **Function:**Read all angles of the manipulator
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
- **Return**：
  - `ANGLES` ：List of joint angles corresponding to the manipulator (including the angles from joint 1 to joint 6)

2. Block display



3.3.3 Send single angle

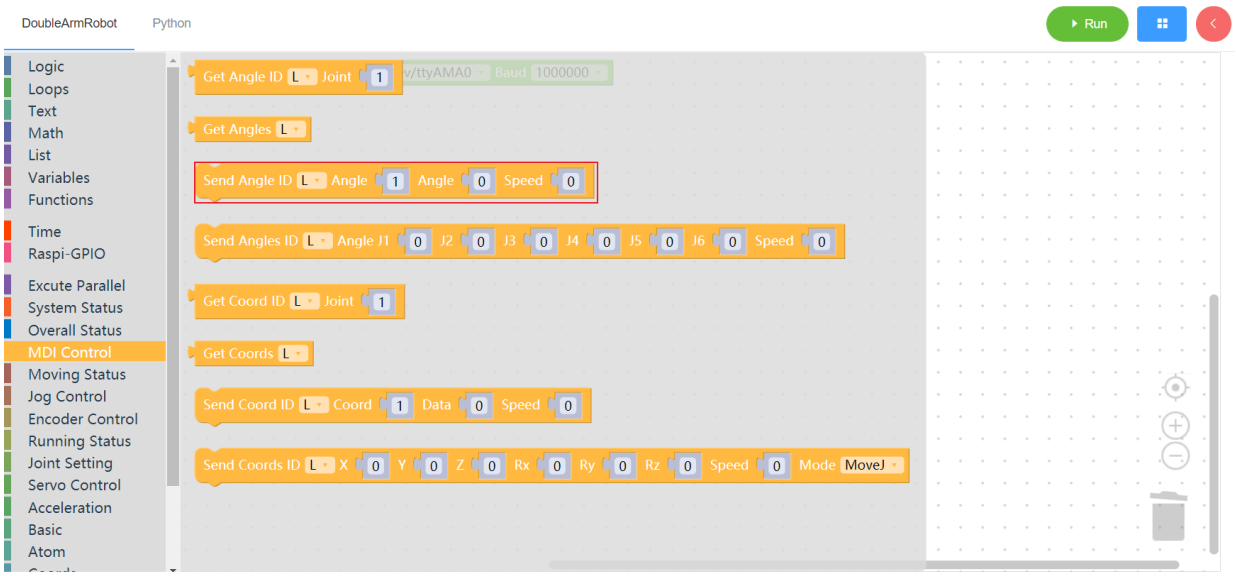
1. Runtime API Reference

- **Function:**Set the single Angle of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist

- `JOINT_ID` ：joint1~6
- `ANGLE` ：angle
- `SPEED` speed(0~100)

- **Return**：none

2. Block display

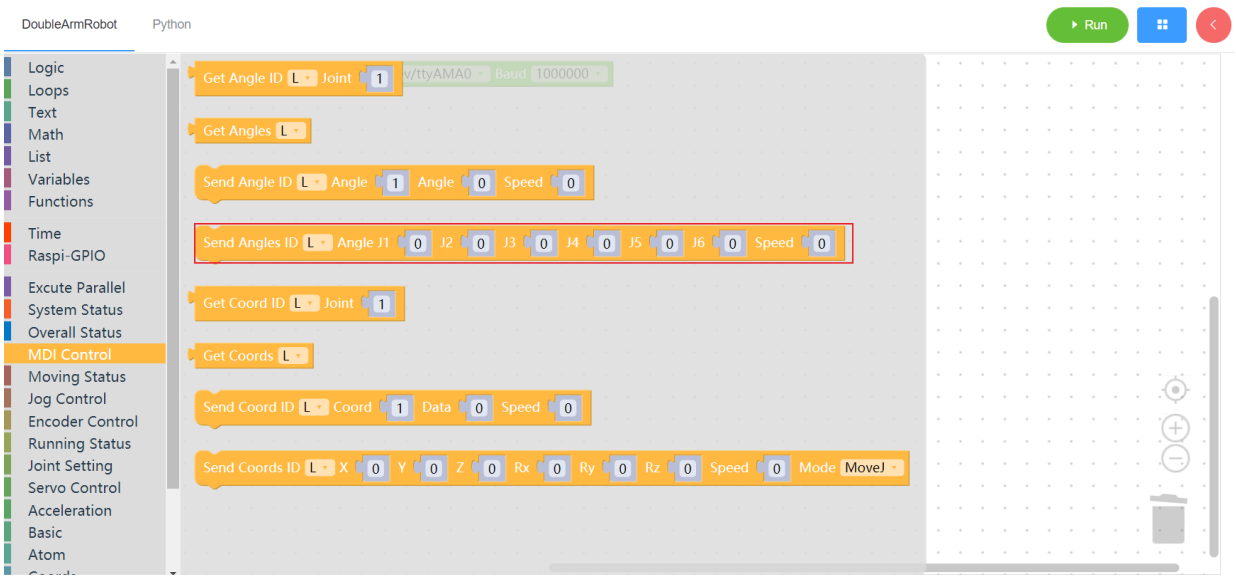


3.3.4 Send all angles

1. Runtime API Reference

- **Function:**Set all the angles of the manipulator
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
  - `ANGLES` : Angle from joint 1 to joint 6
  - `SPEED` ：Moving speed of mechanical arm (0~100)
- **Return**：none

2. Block display

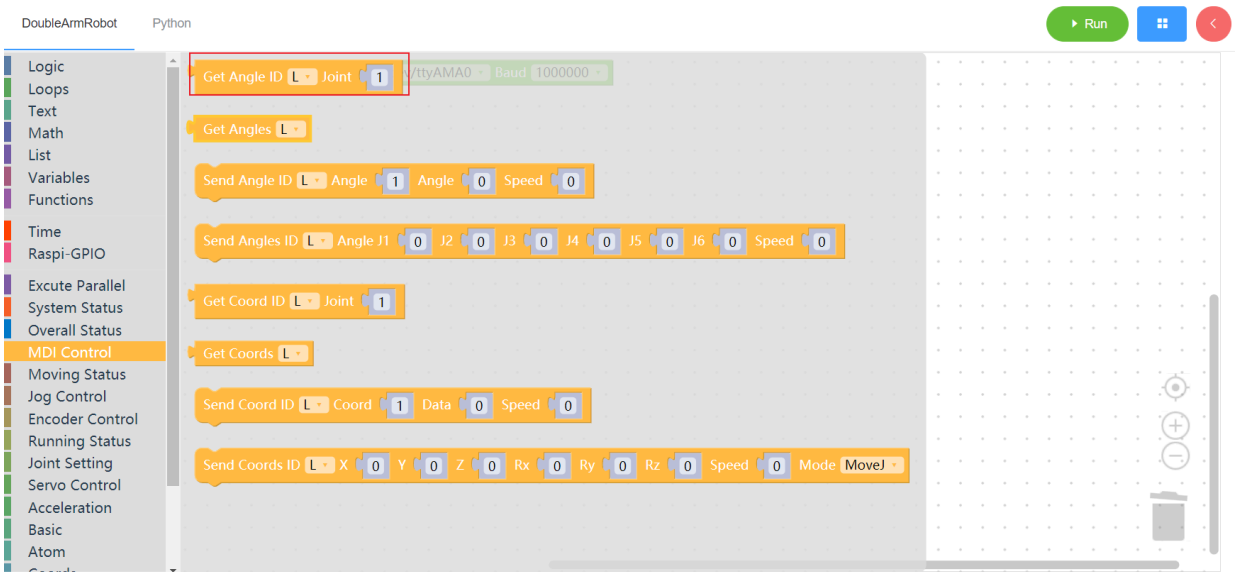


3.3.5 Read single coordinate

1. Runtime API Reference

- **Function:**Read the coordinates of a single joint of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ： L for the left arm, R for the right arm，W for the waist
  - `COORD` ： 1~6（x/y/z/rx/ry/rz）
  - `SPEED` : speed(0~100)
- **Return** : single coordinate

2. Block display

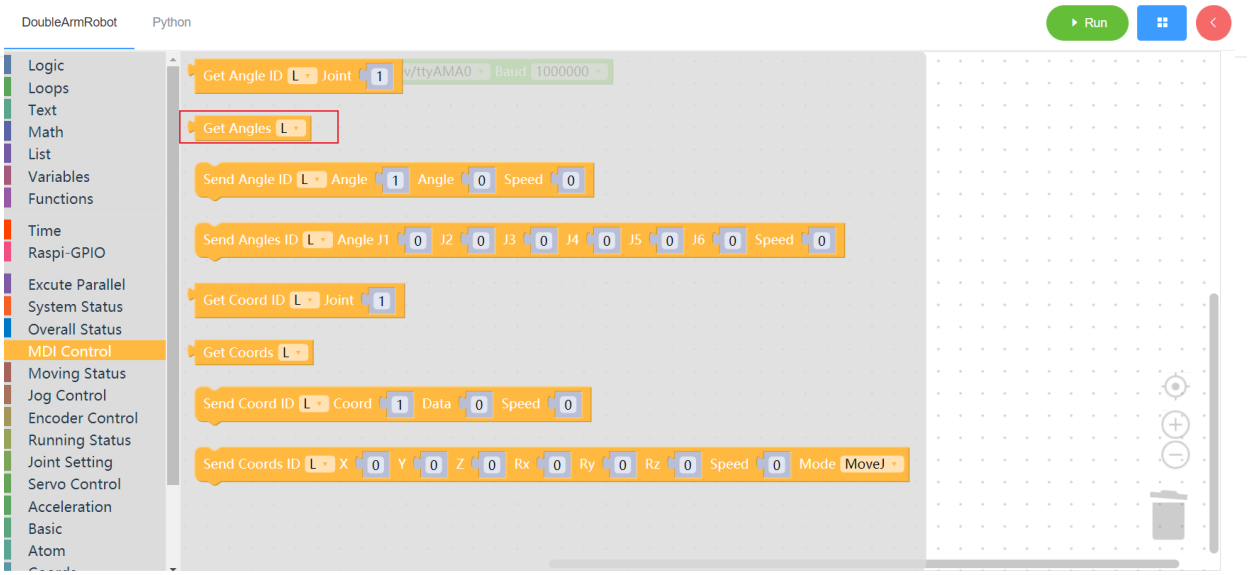


3.3.6 Read all coordinates

1. Runtime API Reference

- **Function:**Read all coordinates of the manipulator
- **Arguments:**
  - `ID(L/R)` ： L for the left arm, R for the right arm
- **Return**：
  - `COORDS` : All joint coordinates of the manipulator

2. Block display

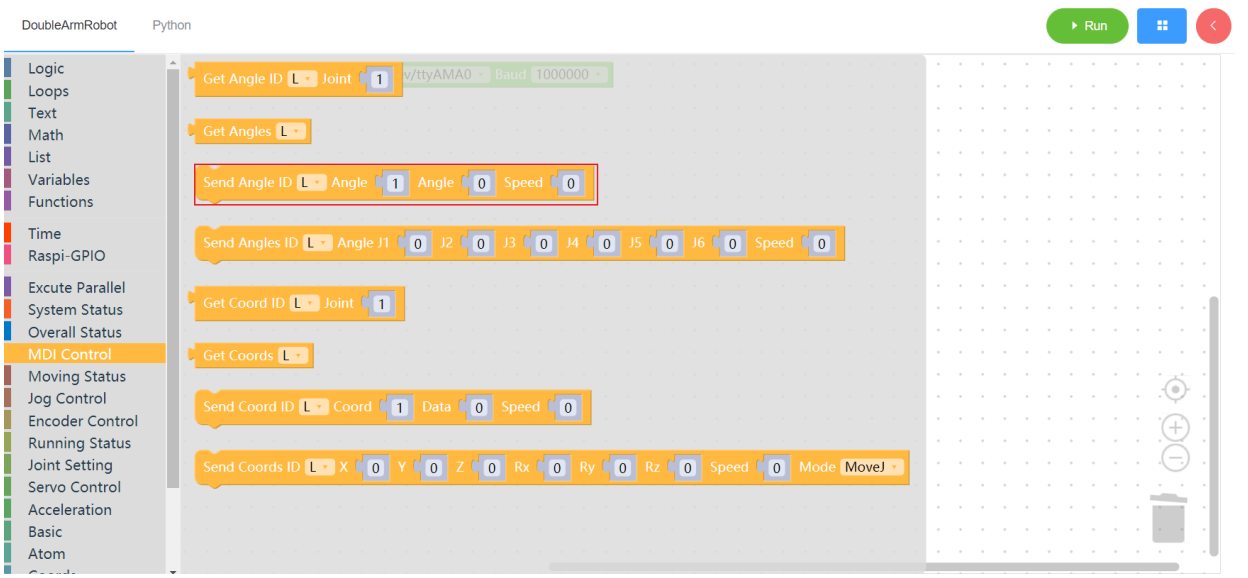


3.3.7 Send single coordinate

1. Runtime API Reference

- **Function:**Set a single coordinate of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `COORD` ：Coordinate ID, range 1~6（x/y/z/rx/ry/rz）
  - `DATA` ：coordinate data
  - `SPEED` : speed(0~100)
- **Return**：none

2. Block display



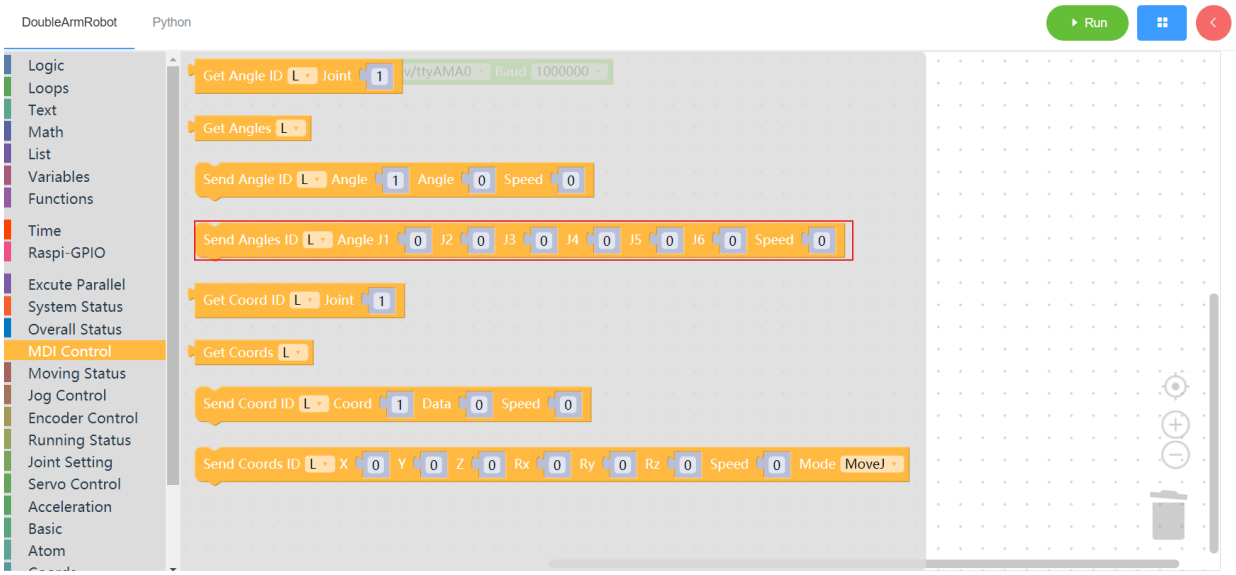
3.3.8 Send all coordinates

1. Runtime API Reference

- **Function:**Set all coordinates of the manipulator

- **Arguments:**
  - `ID(L/R)` ： L for the left arm, R for the right arm
  - `COORDS` : Coordinates of joints 1 to 6
  - `SPEED` : The moving speed of the manipulator(0~100)
  - `MODE` ： 0 - moveJ, 1 - moveL, 2 - moveC
- **Return**： none

2. Block display



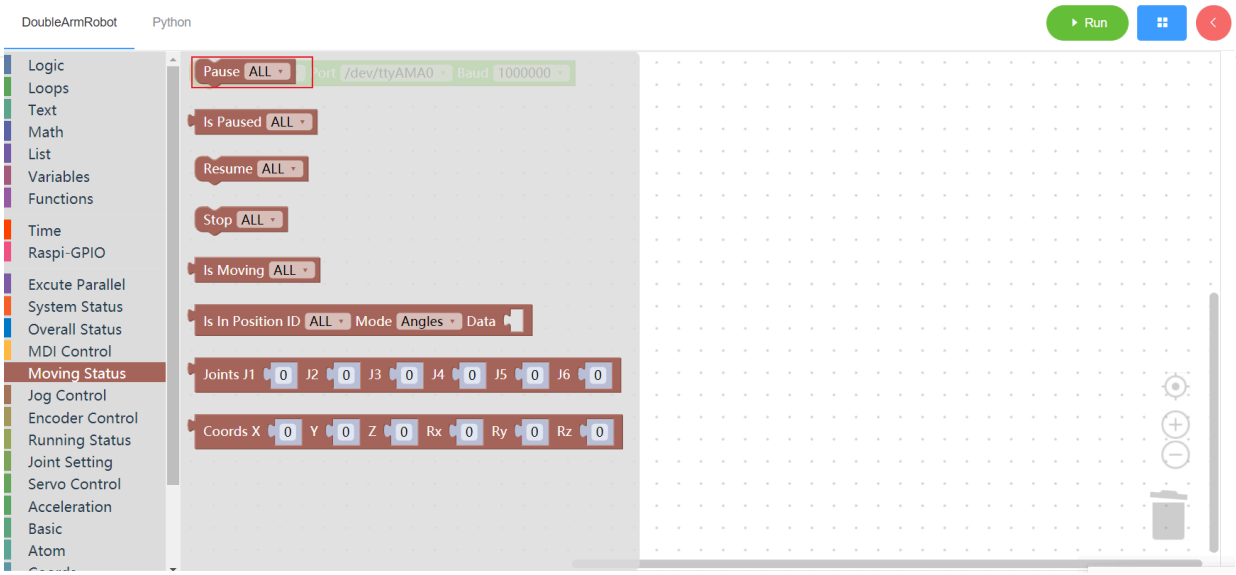
3.4 Motion control

3.4.1 Pause

1. Runtime API Reference

- **Function:**Suspend the movement of the manipulator
- **Arguments:**
  - `ID(ALL/L/R/W)` : All for all arms , for the left arm, R for the right arm，W for waist
- **Return**： none

2. Block display

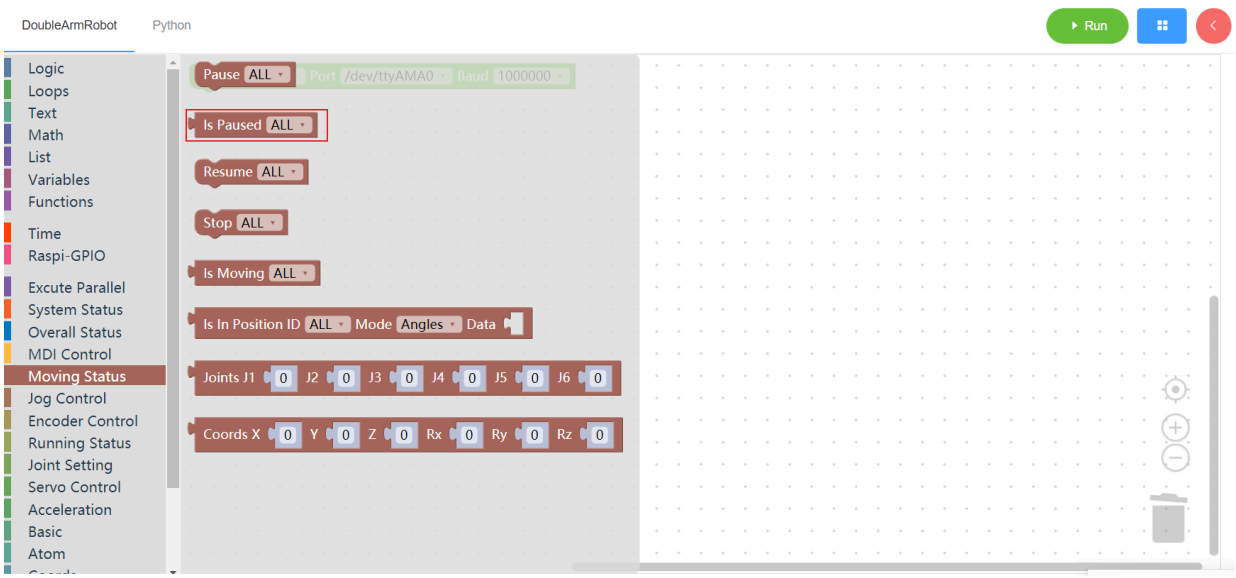


3.4.2 Is paused

1. Runtime API Reference

- **Function:**Check whether the manipulator is suspended
- **Arguments:**
  - `ID(ALL/L/R/W)`：ALL for all arms，L for the left arm, R for the right arm，W for waist
- **Return**：
  - 1 paused
  - 0 did not pause
  - -1 error

2. Block display



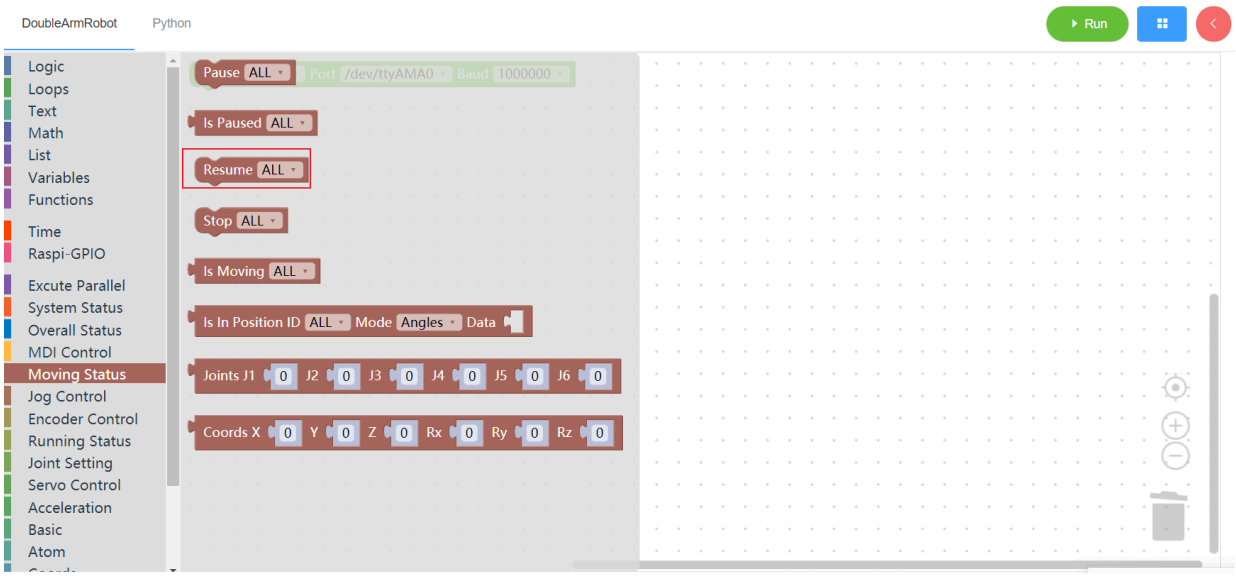
3.4.3 Resume

1. Runtime API Reference

- **Function:**Restore the movement of the manipulator

- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for all arms，L for the left arm, R for the right arm，W for waist
- **Return**：none

2. Block display

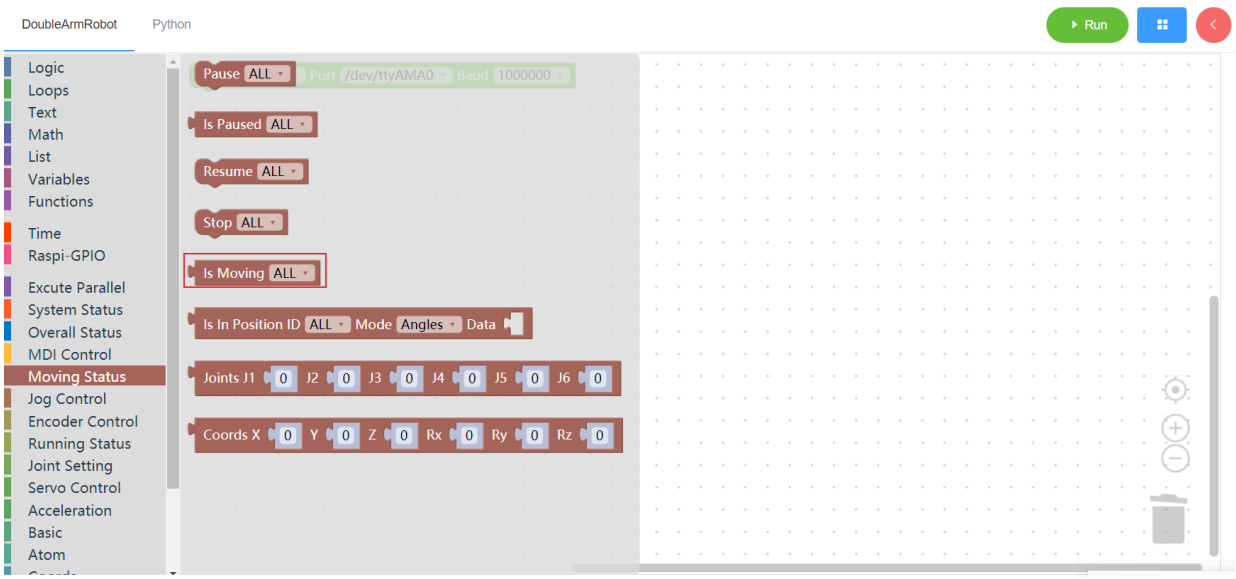


3.4.5 Is moving

1. Runtime API Reference

- **Function:**Check if the arm is moving
- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for all arms，L for the left arm, R for the right arm，W for waist
- **Return**：
  - 1 moving
  - 0 did not move
  - -1 error

2. Block display

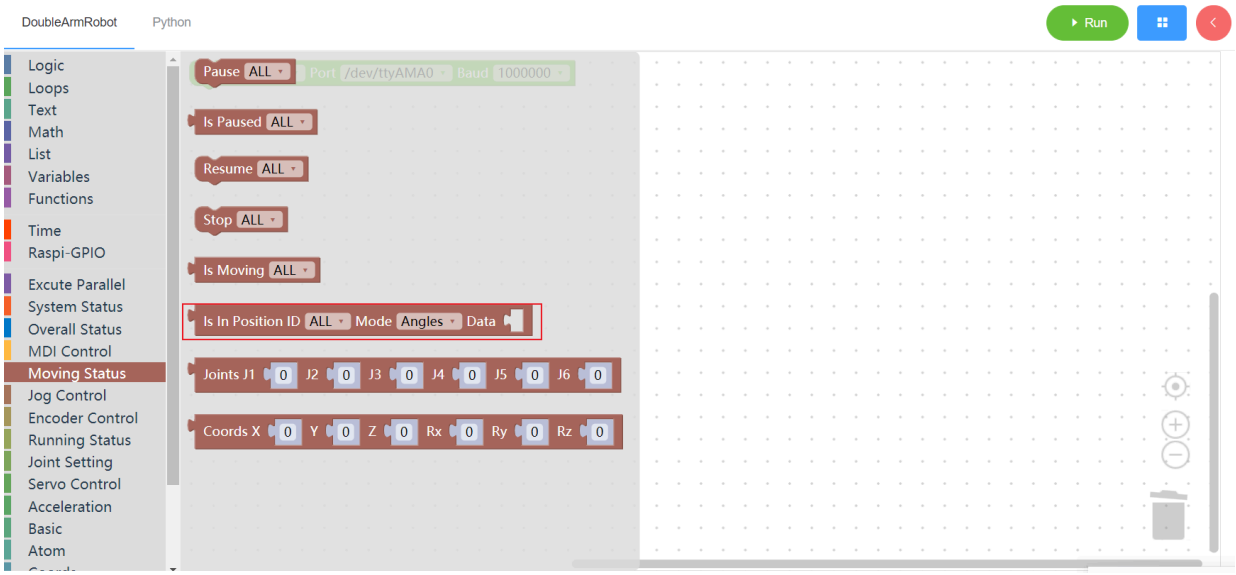


3.4.6 Whether to reach the specified position

1. Runtime API Reference

- **Function:**Check whether the manipulator reaches the specified position
- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for all arms，L for the left arm, R for the right arm，W for waist
  - `DATA` : Data list: 13 data for ALL, 6 data for L/R, and 1 data for W
  - `MODE` : Mode (Angles/Coords)
- **Return**：
  - 1 yes
  - 0 no
  - -1 error

2. Block display

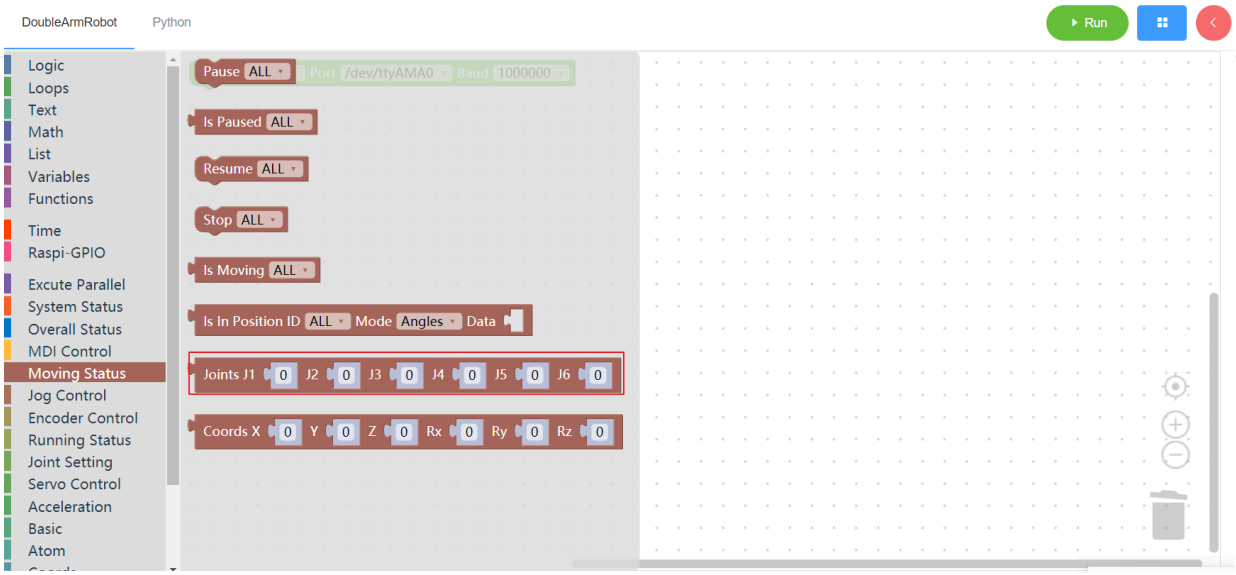


3.4.7 Angles list

1. Runtime API Reference

- **Function:** for **3.4.6 Whether to reach the specified position** separately set the list, can be filled in 6 angles
- **Arguments:**
  - `DATA` ： Data list, fill in 6 angles
- **Return**：
  - `ANGLES` : Angle list (length 6)

2.Block display

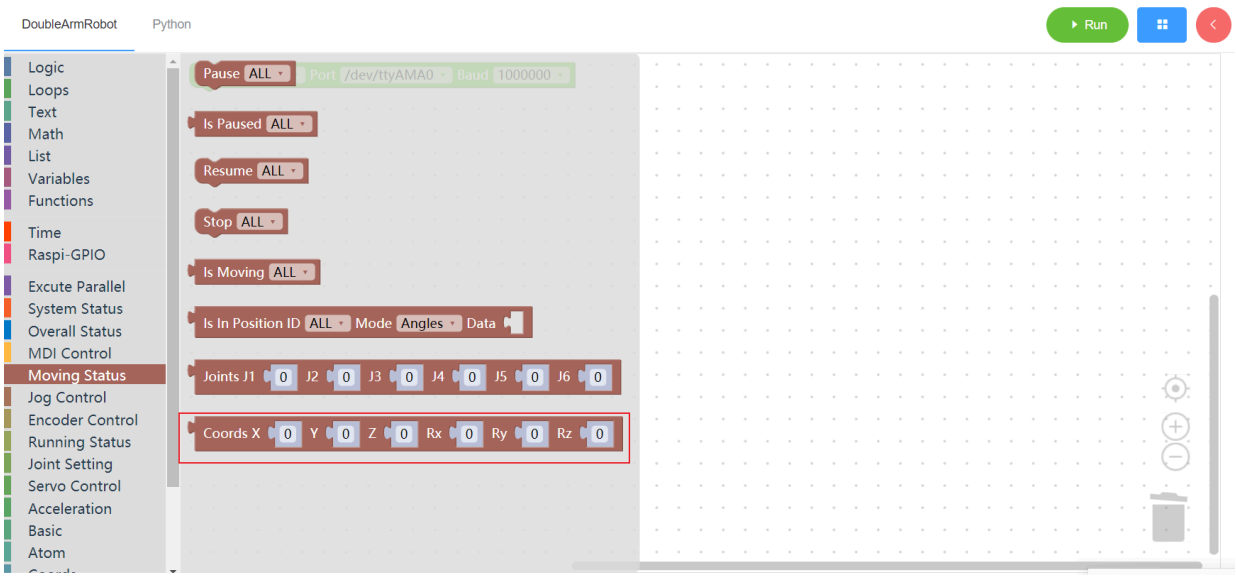


3.4.8 Coordinates list

1. Runtime API Reference

- **Function:** for **3.4.6 Whether to reach the specified location** separately set the list, can be filled in 6 coordinates
- **Arguments:**
  - `DATA` : Data list, fill in 6 coordinates
- **Return**：
  - `COORDS` ： Coordinate list (length 6)

2.Block display



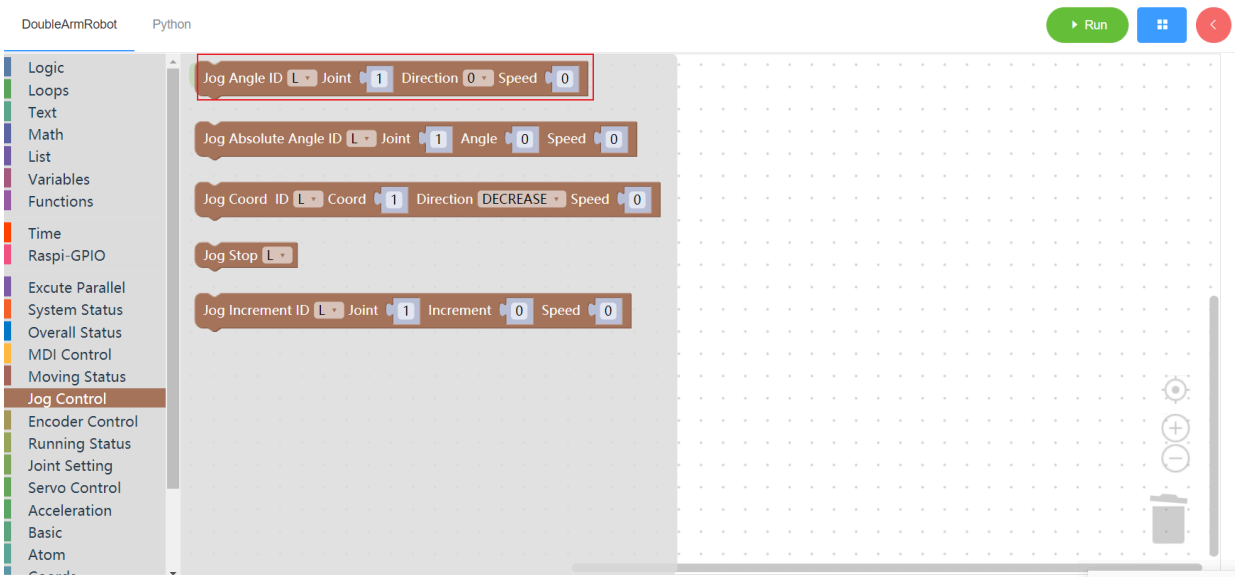
3.5 Jog control

3.5.1 Jog angle

1. Runtime API Reference

- **Function:**Jog angle
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id，1~6
  - `DIRECTION` ：0-decrease 1-increase
  - `SPEED` ：speed(0~100)
- **Return**：none

2.Block display

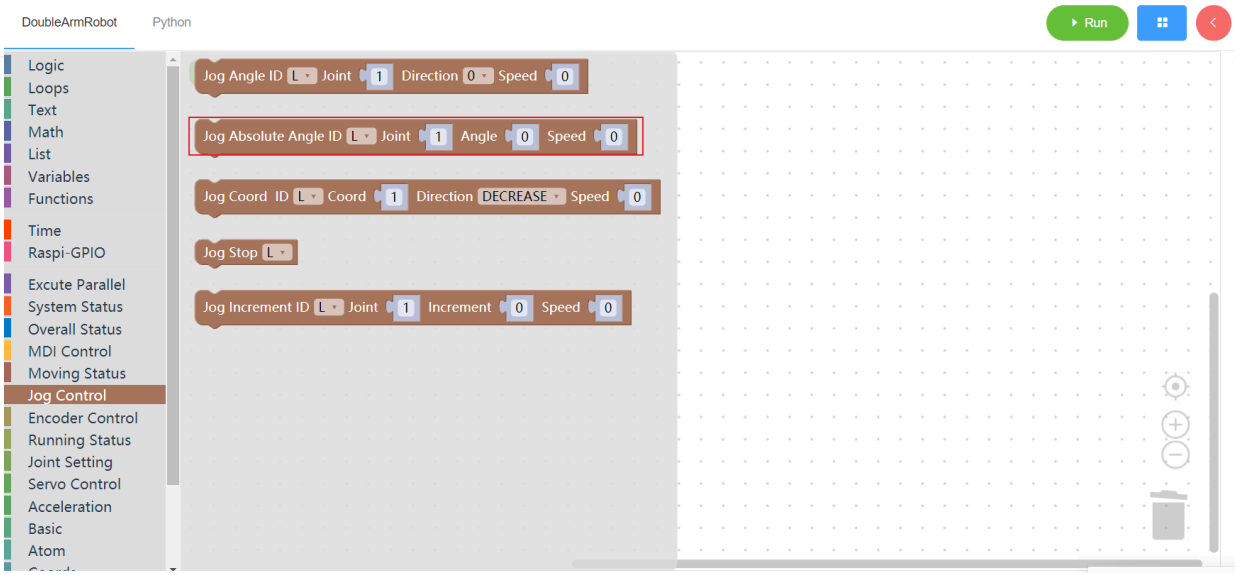


3.5.2 Jog absolute

1. Runtime API Reference

- **Function:**Jog absolute
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id，1~6
  - `AGNLE` ：angle
  - `SPEED` ：speed(0~100)
- **Return**：none

2.Block display

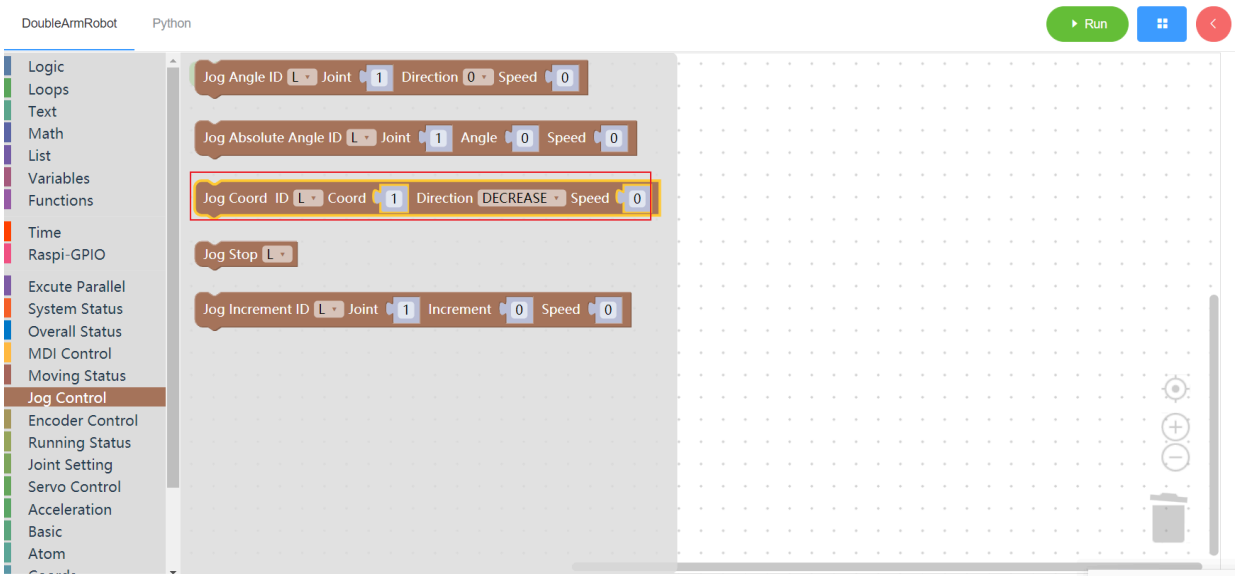


3.5.3 Jog coordinate

1. Runtime API Reference

- **Function:**Jog coordinate
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `COORD` ：coordinate 1~6（x/y/z/rx/ry/rz)
  - `DIRECTION` ：0-decrease 1-increase
  - `SPEED` speed (0~100)
- **Return**：none

2.Block display



3.5.4 Jog stop

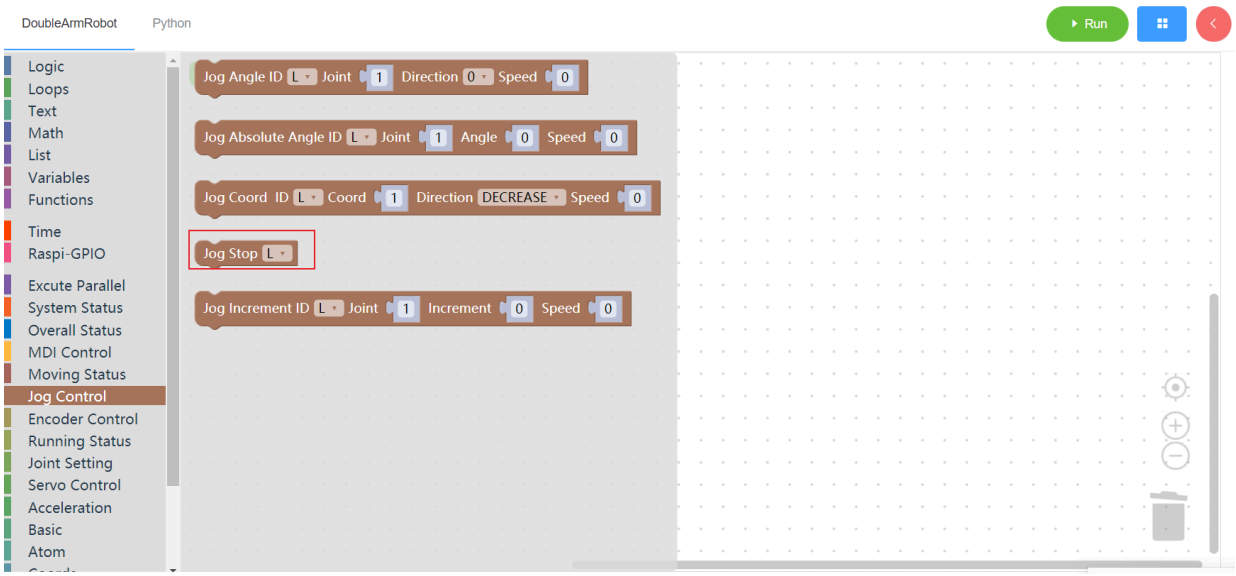
1. Runtime API Reference

- **Function:**Jog stop
- **Arguments:**

- `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist

- **Return**：none

2.Block display

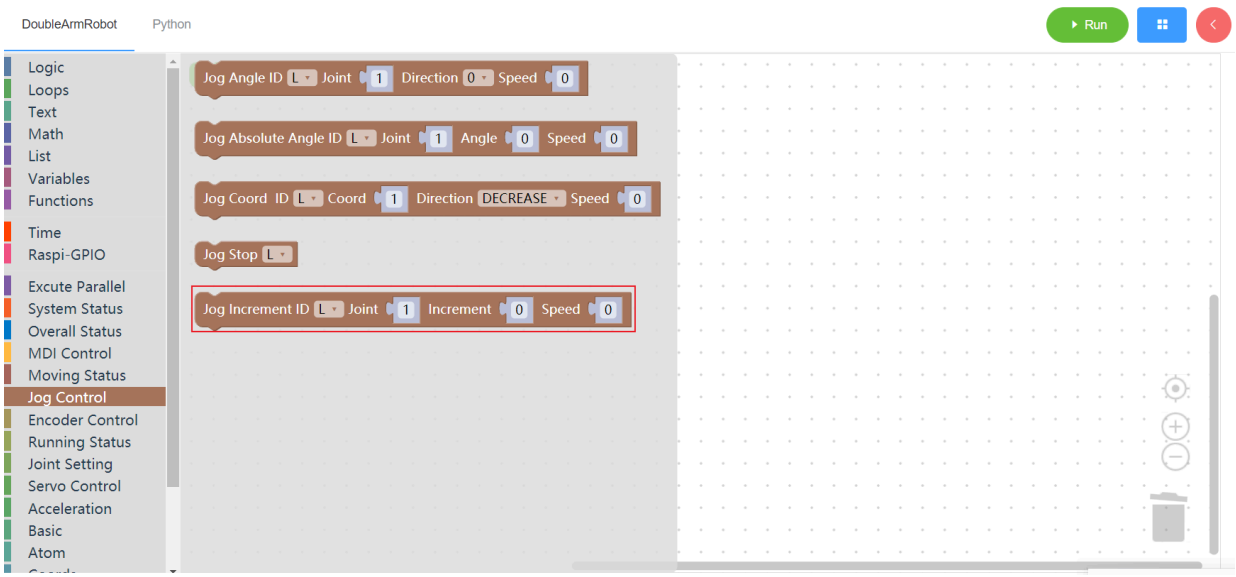


3.5.5 Jog increment

1. Runtime API Reference

- **Function:**Set jog increment
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id（1~6）
  - `INCREMENT` increment (integer)
  - `SPEED` speed (0~100)
- **Return**：none

2.Block display



3.6 Encoder

3.6.1 Set encoder

1. Runtime API Reference

- **Function:**Set the potentiometer code of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ： L for the left arm, R for the right arm，W for waist
  - `JOINT` ： joint 1~6
  - `ENCODER` :value (integer)
- **Return**： none

2.Block display

3.6.2 Get encoder

1. Runtime API Reference

- **Function:**Obtain the potentiometer code of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ： L for the left arm, R for the right arm，W for waist
  - `JOINT` ： joint 1~6
- **Return**：
  - `ENCODER` : value(0 ~ 4096)

2.Block display

3.6.3 Get encoders

1. Runtime API Reference

- **Function:**Obtain all potentiometer codes of the manipulator
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
- **Return**：
  - `ENCODERS` : encoders list

2.Block display

3.6.4 Set encodes

1. Runtime API Reference

- **Function:**Obtain all potentiometer codes of the manipulator
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm

- `ENCODERS`：encodes list(lenth 6)
- `SPEED` speed(0~100)

- **Return**：none

2.Block display

3.7.1 Running status

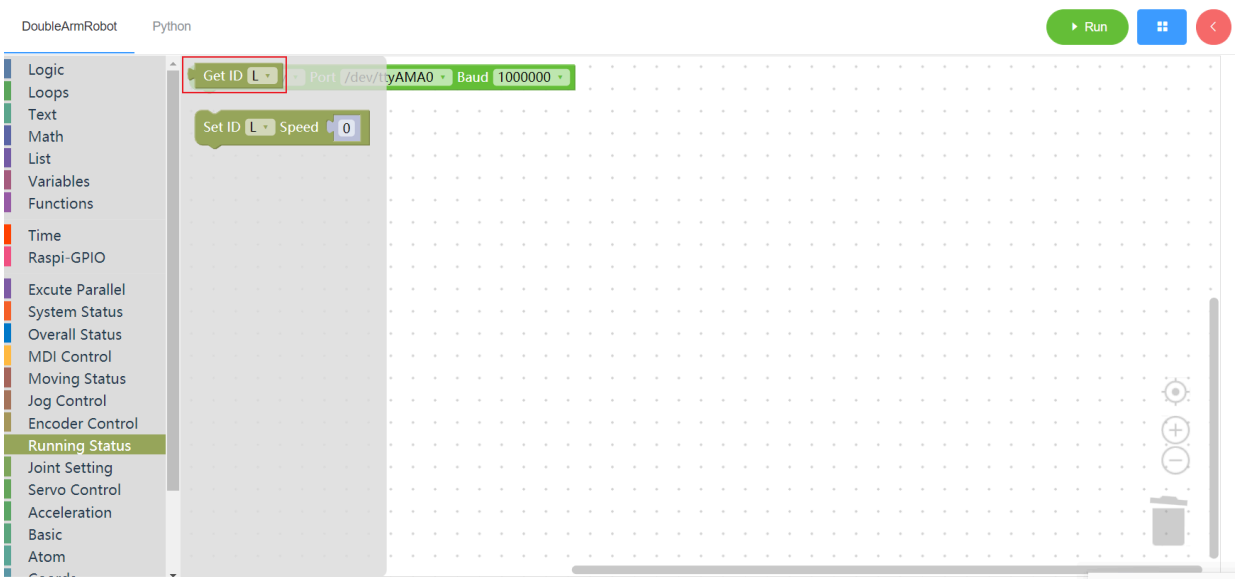
3.7.1 Get speed

1. Runtime API Reference

- **Function:**Get the speed of the manipulator

- **Arguments:**
  - `ID(L/R/W)`：L for the left arm, R for the right arm，W for waist
- **Return**：
  - `SPEED` speed(0~100)

2.Block display

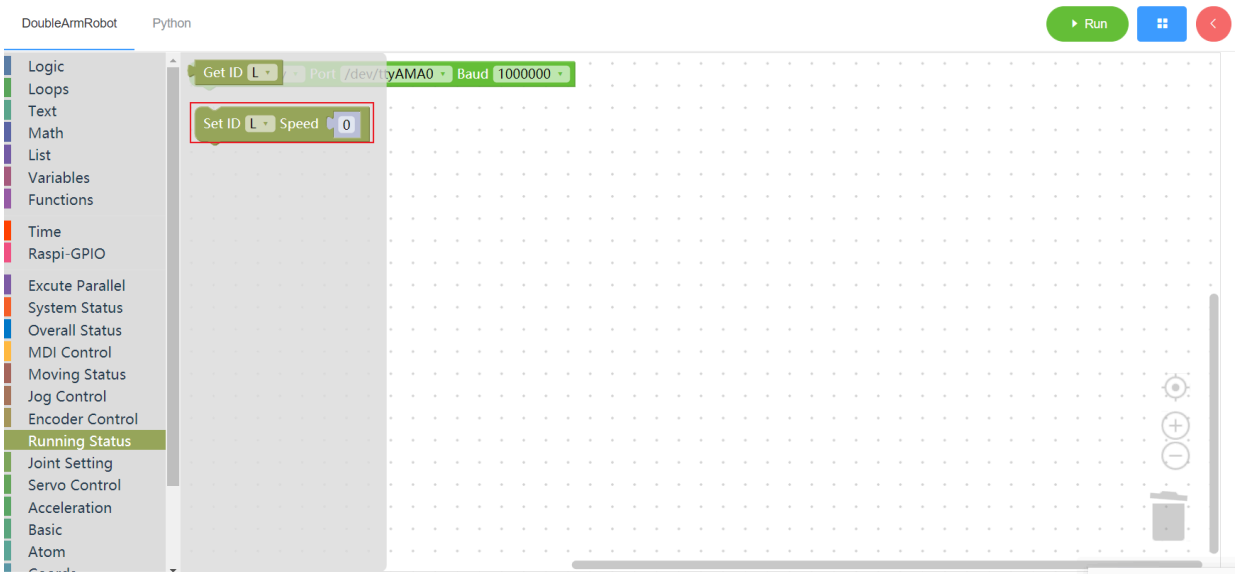


3.7.2 Set speed

1. Runtime API Reference

- **Function:**Set the speed of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `SPEED` ：speed(0~100)
- **Return**：none

2.Block display



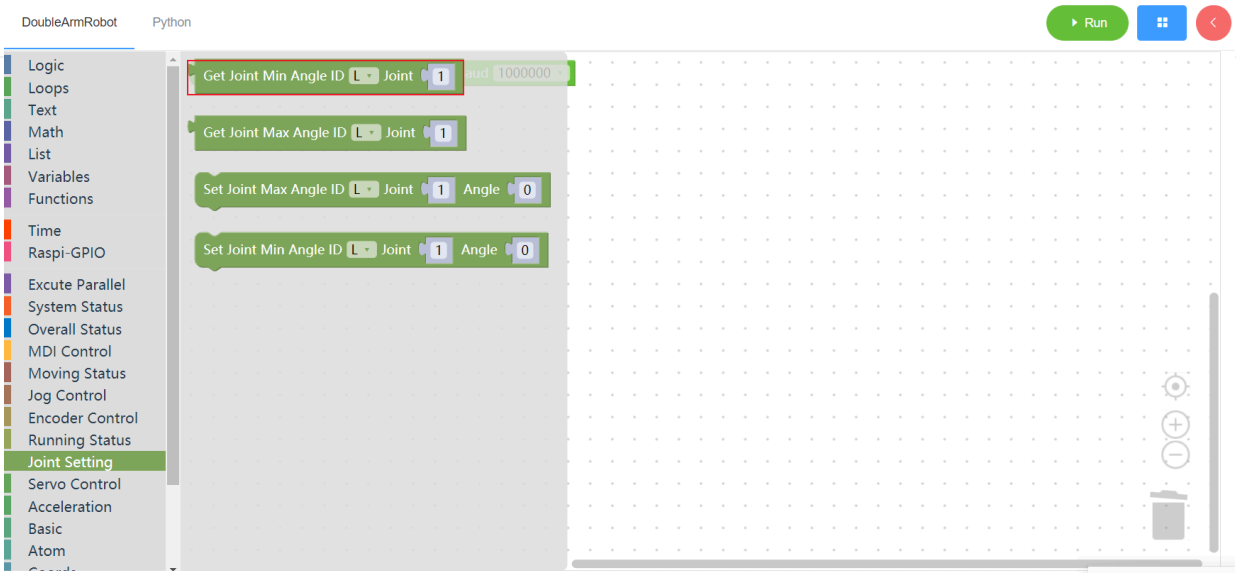
3.8 Joint limit

3.8.1 Read the minimum joint Angle

1. Runtime API Reference

- **Function:**Read the minimum joint Angle of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id(1~6)
- **Return**：
  - `ANGLE` ：angle

2.Block display

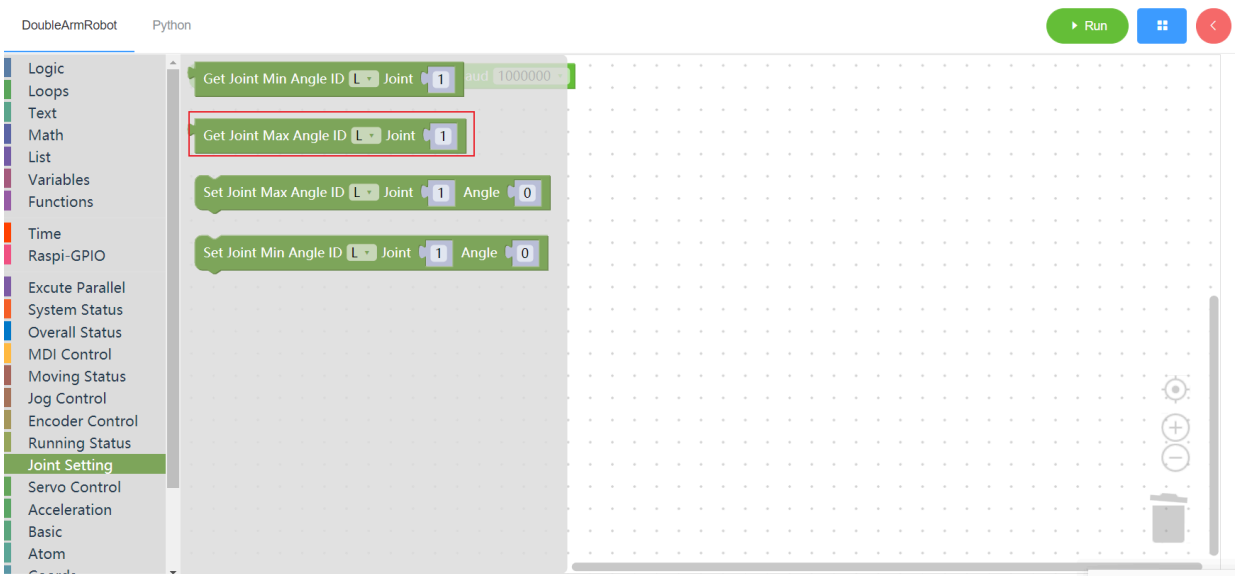


3.8.2 Read the maximum joint Angle

1. Runtime API Reference

- **Function:**Read the maximum joint Angle of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id(1~6)
- **Return**：
  - `ANGLE` : angle

2.Block display



3.8.3 Set the maximum joint Angle

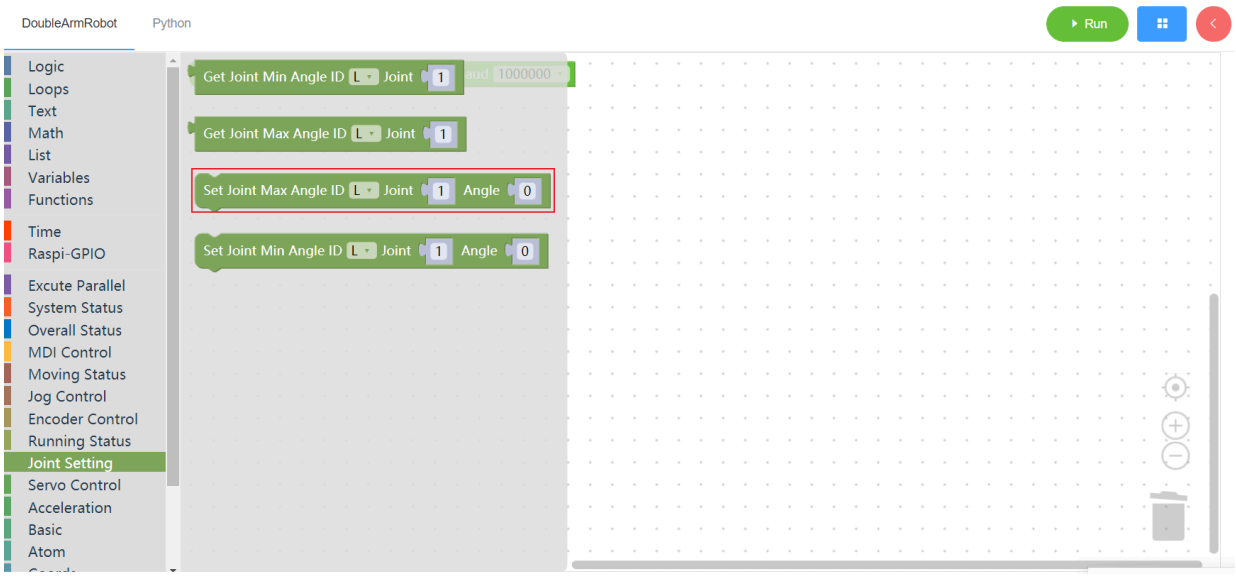
1. Runtime API Reference

- **Function:**Set the maximum joint Angle of the manipulator
- **Arguments:**

- `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
- `JOINT` ：joint id(1~6)
- `ANGLE` : angle

- **Return**： none

2.Block display



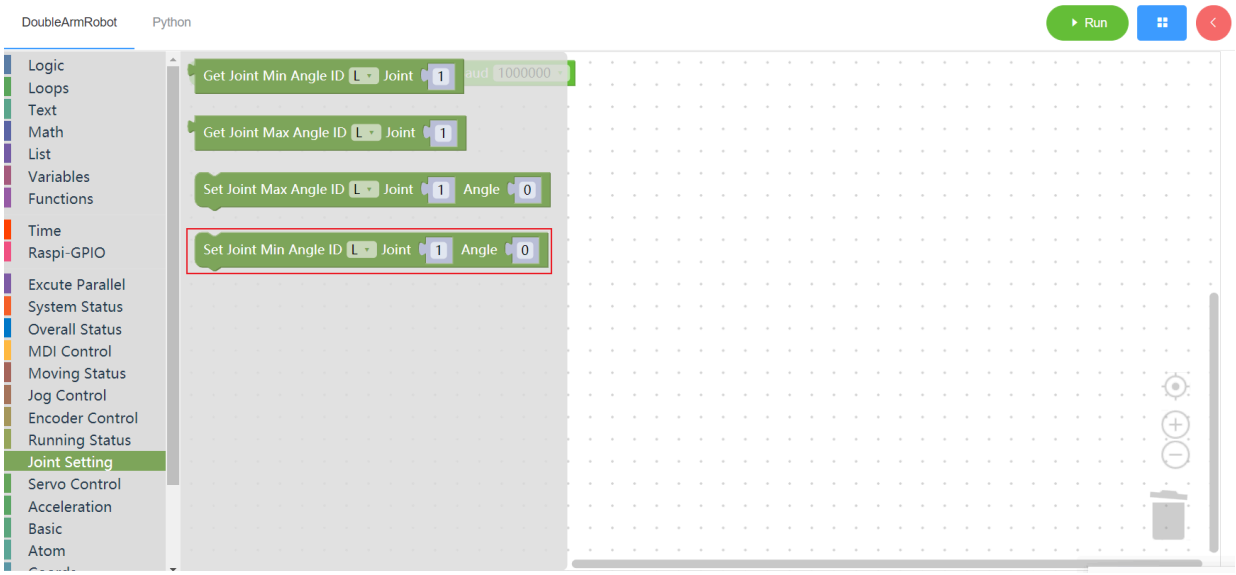
3.8.4 Set the minimum joint Angle

1. Runtime API Reference

- **Function:**Set the minimum joint Angle of the manipulator

- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id(1~6)
  - `ANGLE` ：angle
- **Return**： none

2.Block display



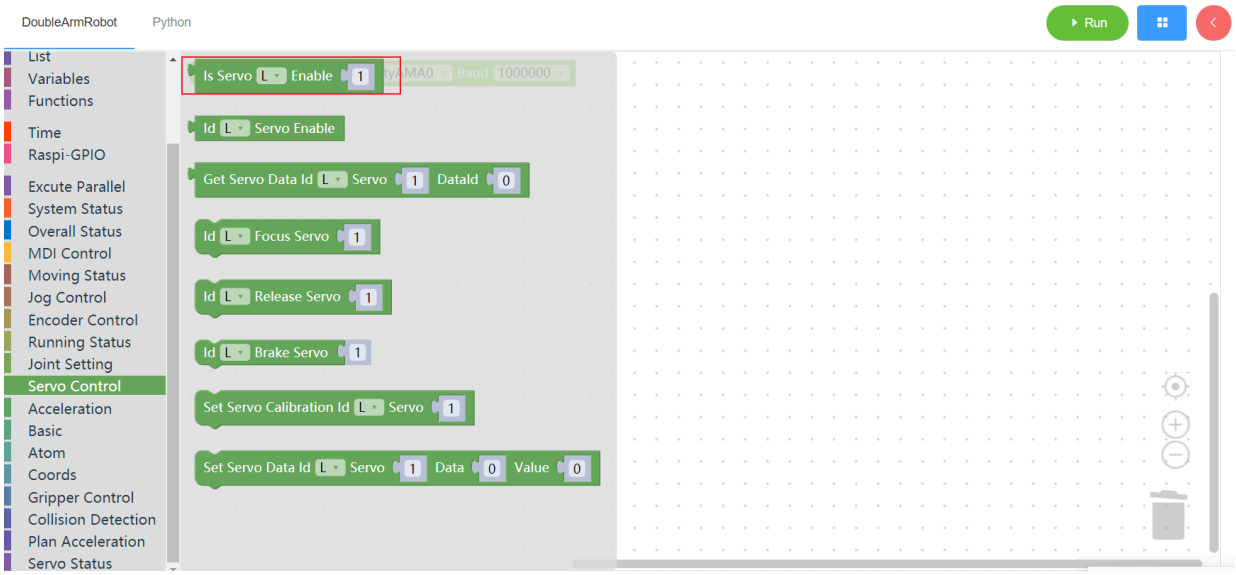
3.9 Servo settings

3.9.1 Joint state

1. Runtime API Reference

- **Function:**Check whether the specified joint of the manipulator is connected
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `SERVOID` ：servo id(1~6)
- **Return**：
  - 0 disenable
  - 1 enable
  - -1 error

2.Block display

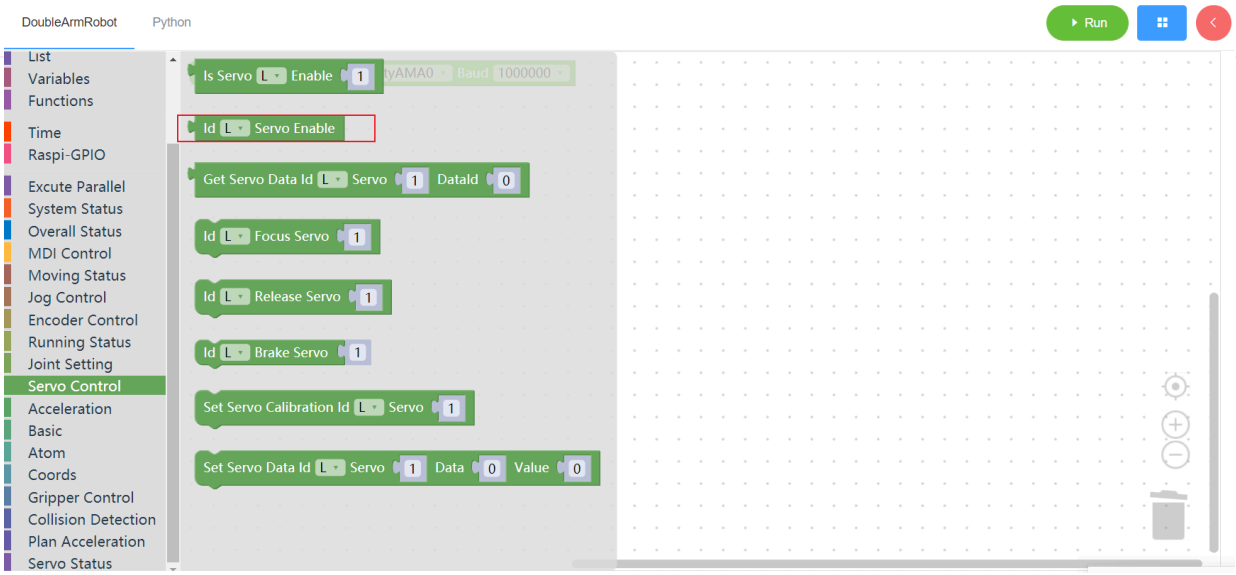


3.9.2 Status of all joint connections

1. Runtime API Reference

- **Function:**Check whether all joints of the manipulator are connected
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
- **Return**：
  - 0 disenable
  - 1 enable
  - -1 error

2.Block display

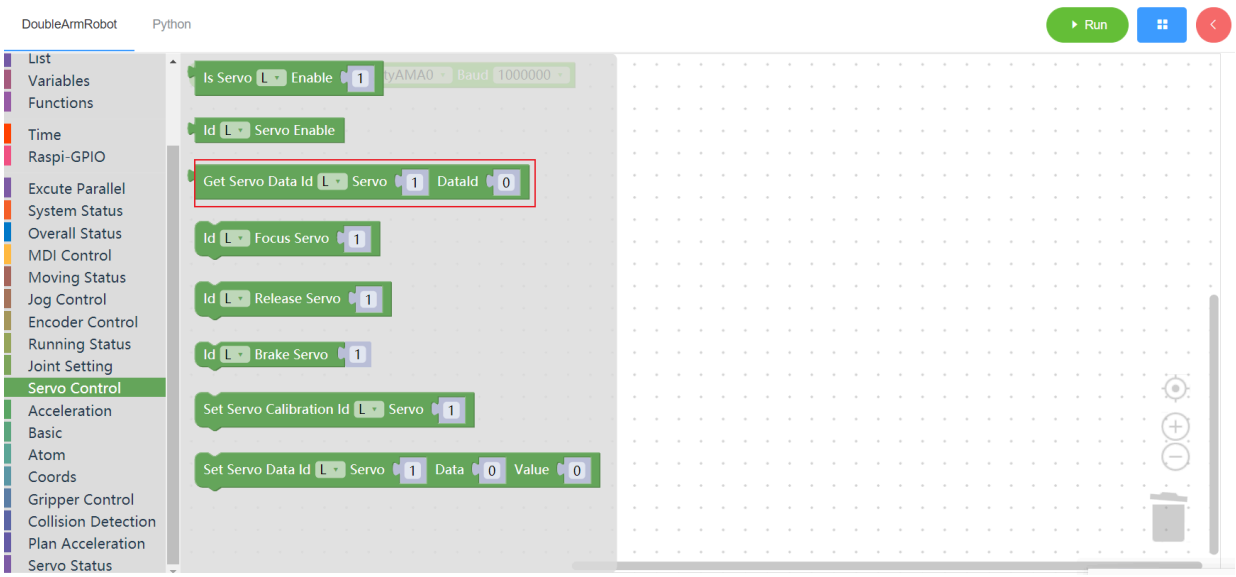


3.9.3 Get servo data

1. Runtime API Reference

- **Function:**Get servo data of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `SERVOID` ： servo id(1~6)
  - `DATA_ID` : address
- **Return**：
  - 0 disenable
  - 1 enable
  - -1 error

2.Block display



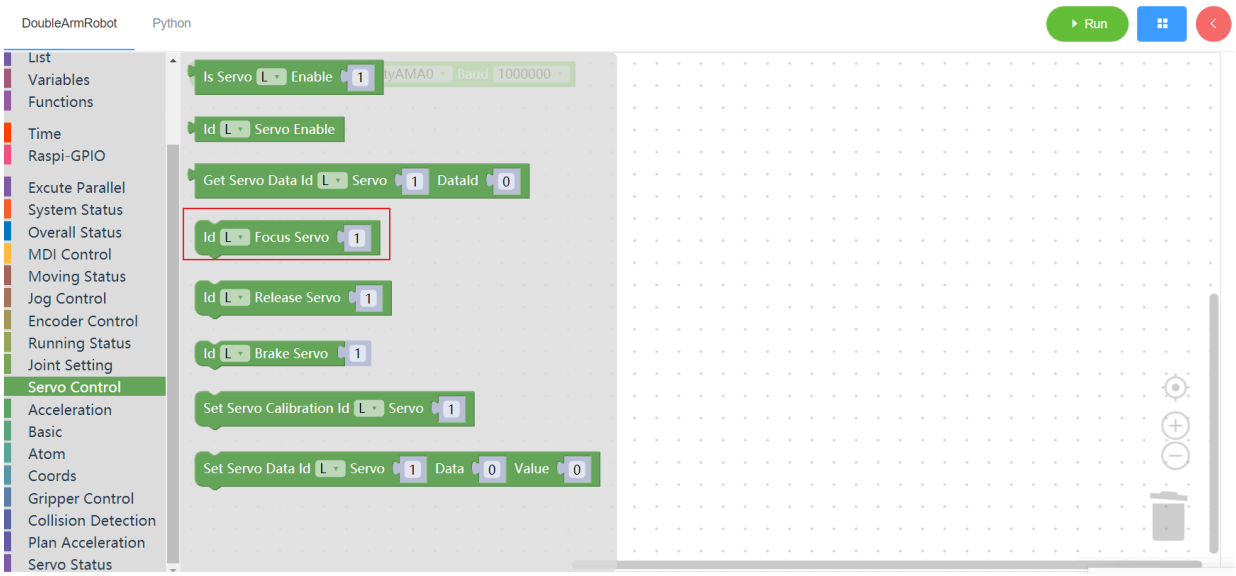
3.9.4 Power on a single motor

1. Runtime API Reference

- **Function:**Power on a single motor of the manipulator

- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `SERVOID` ：servo id(1~6)
- **Return**：none

2.Block display

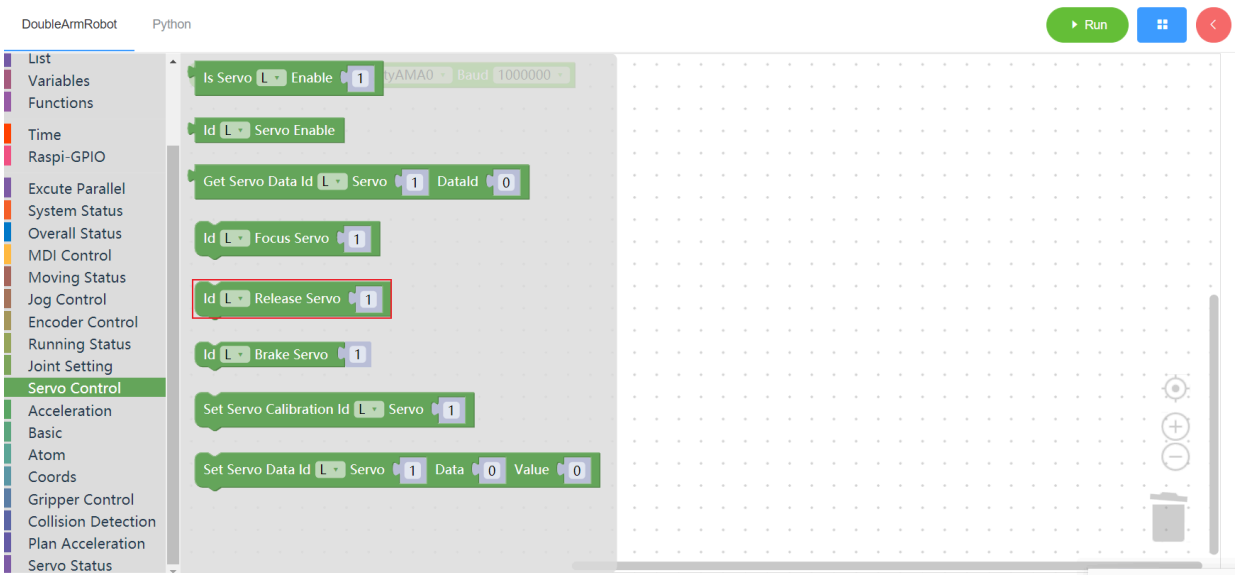


3.9.5 Power off a single motor

1. Runtime API Reference

- **Function:**Power off a single motor of the manipulator
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `SERVOID` ：servo id(1~6)
- **Return**：none

2.Block display

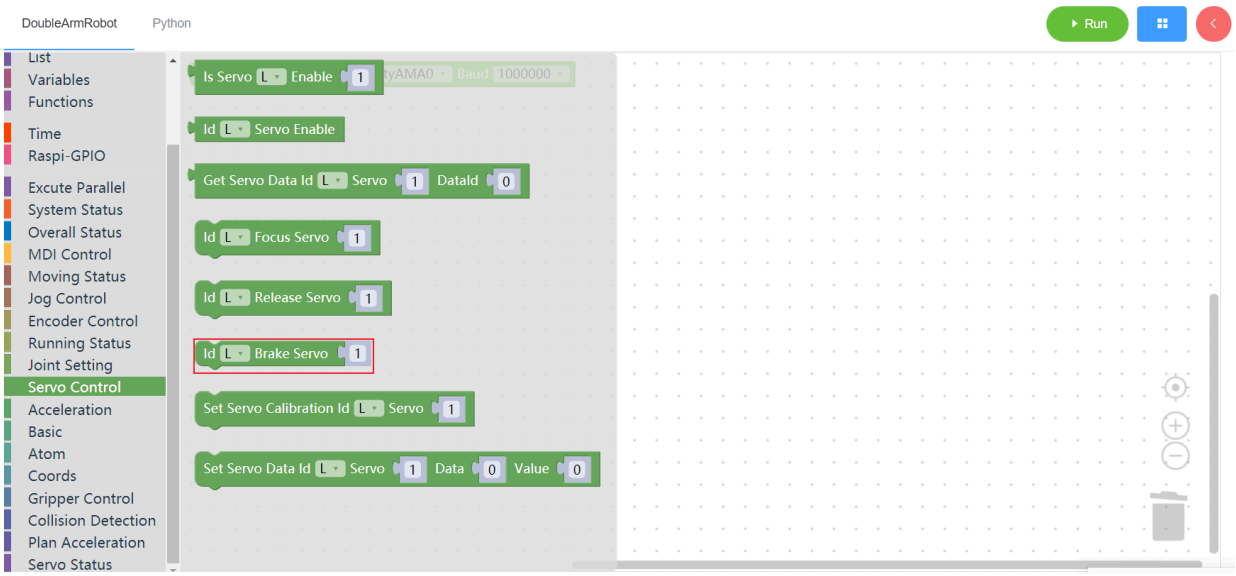


3.9.6 Single joint brake

1. Runtime API Reference

- **Function:**The manipulator brakes on a single joint
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id(1~6)
- **Return**：none

2.Block display

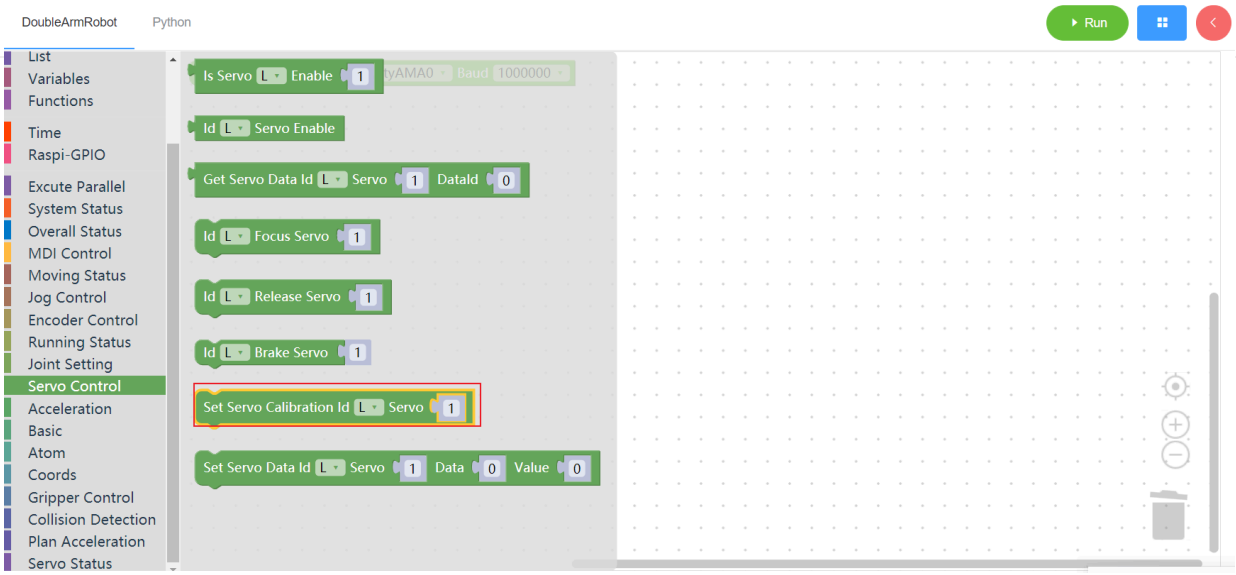


3.9.7 Set motor zero

1. Runtime API Reference

- **Function:**Set motor zero
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `JOINT` ：joint id(1~6)
- **Return**：none

2.Block display

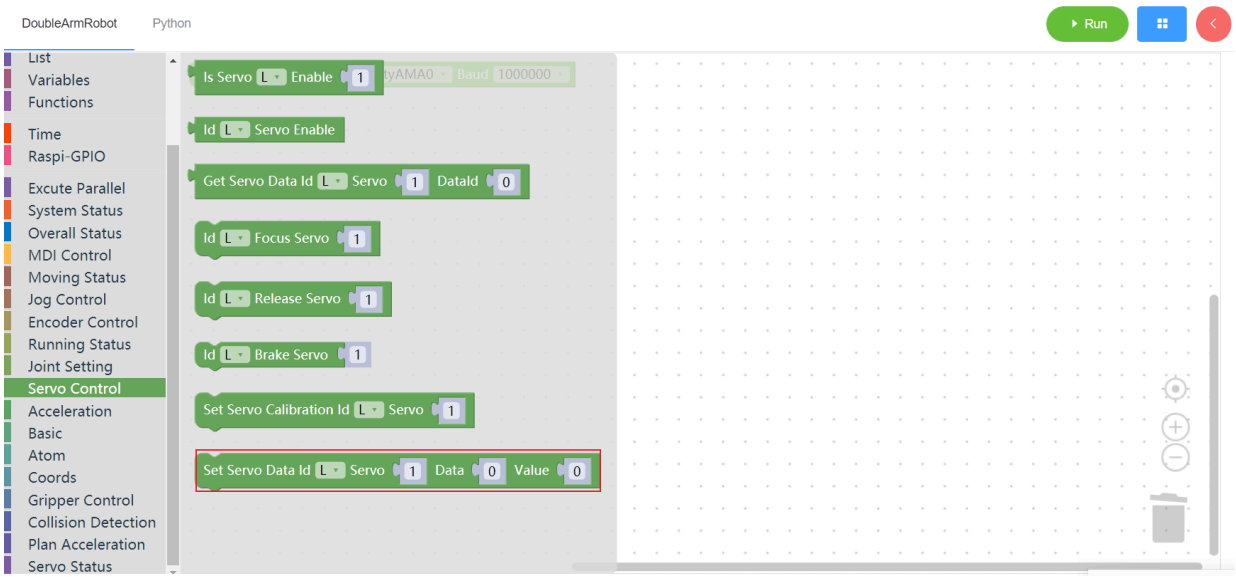


3.9.8 Set servo data

1. Runtime API Reference

- **Function:**Set servo data
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `SERVO_ID` ：servo id(1~6)
  - `DATA` ：data address
  - `VALUE` ：value
- **Return**：none

2.Block display



3.10 Acceleration settings

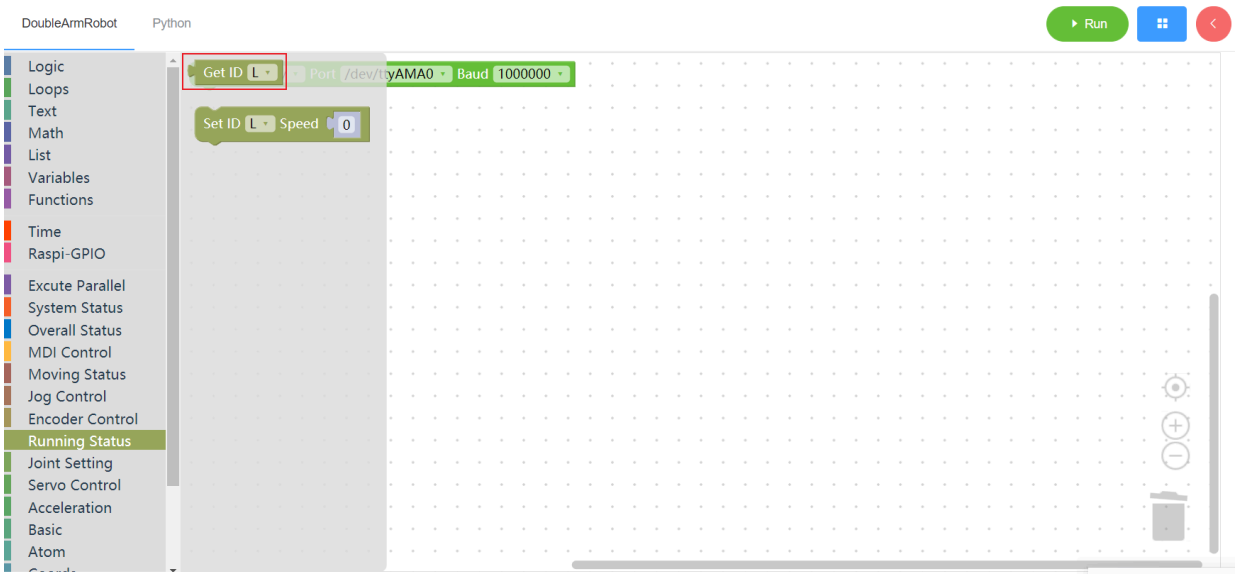
3.10.1 Get acceleration

1. Runtime API Reference

- **Function:**Read the acceleration of all movements

- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
- **Return**：
  - `SPEED` ： speed

2.Block display



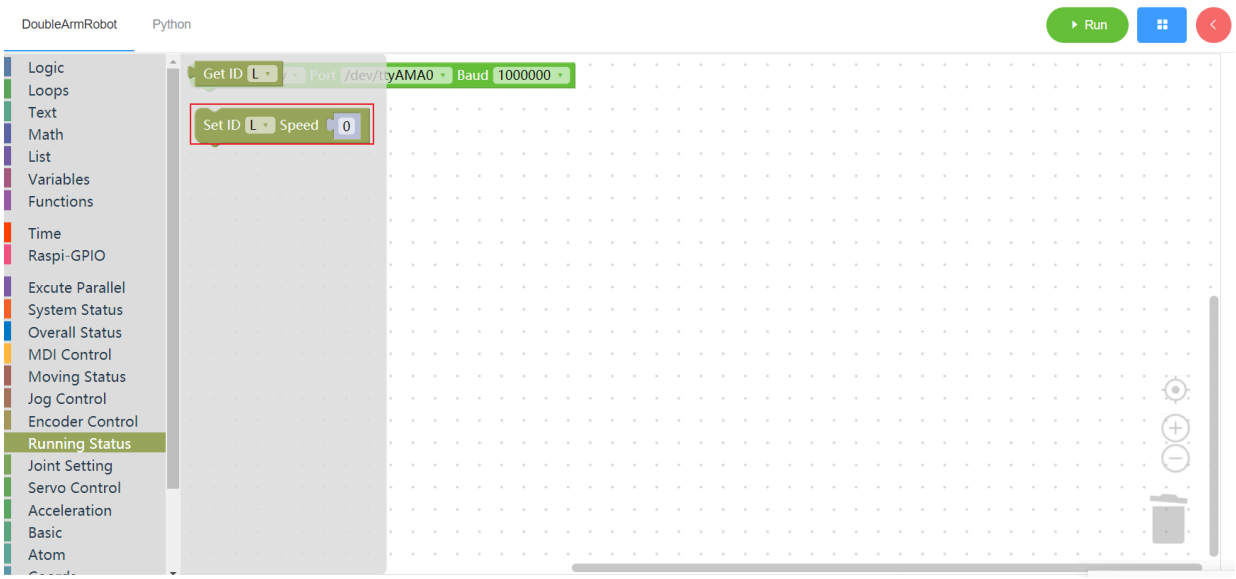
3.10.2 Set movement acceleration

1. Runtime API Reference

- **Function:**Set movement acceleration

- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
  - `ACC` ： integer(1~100)
- **Return**： none

2.Block display



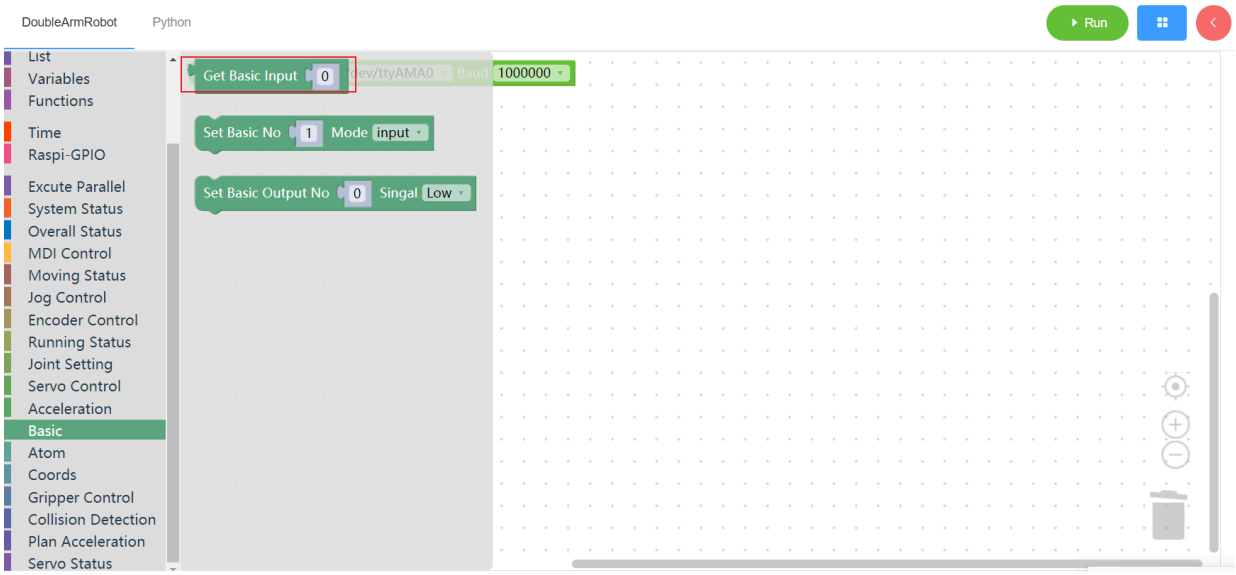
3.11 M5Stack-basic

3.11.1 Get basic input

1. Runtime API Reference

- **Function:**Get basic input
- **Arguments:**
  - `PIN_NO` ：integer(0~20)
- **Return**：unknown

2.Block display

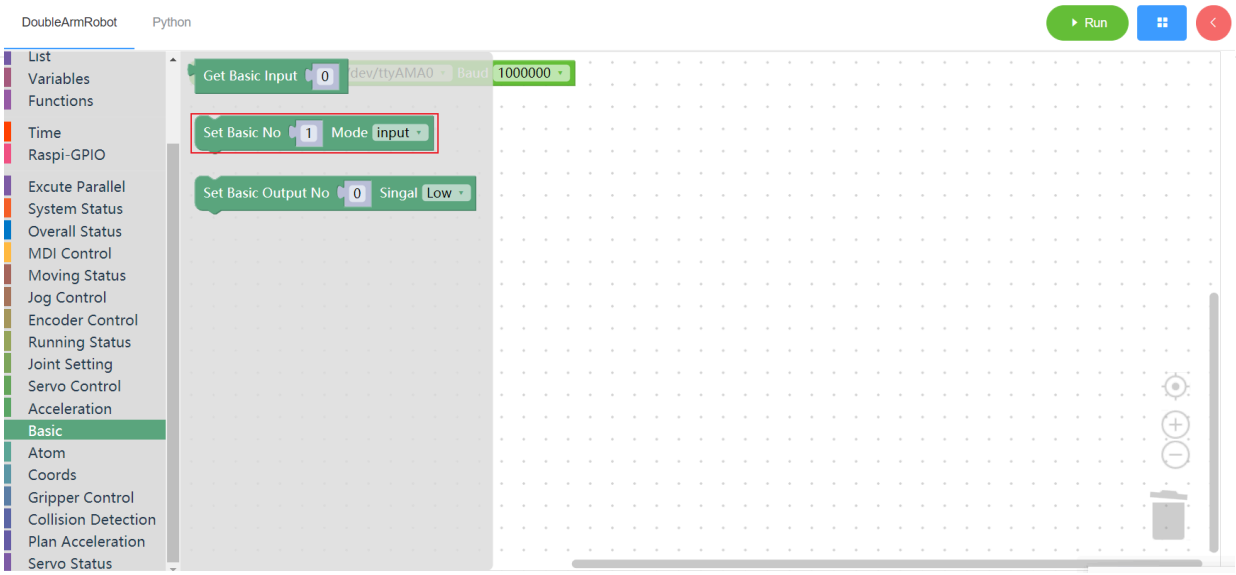


3.11.2 Set basic mode

1. Runtime API Reference

- **Function:**Set the base input/output mode
- **Arguments:**
  - `PIN_NO` ：integer(1~5)
  - `PIN_MODE` ：input/output
- **Return**：none

2.Block display

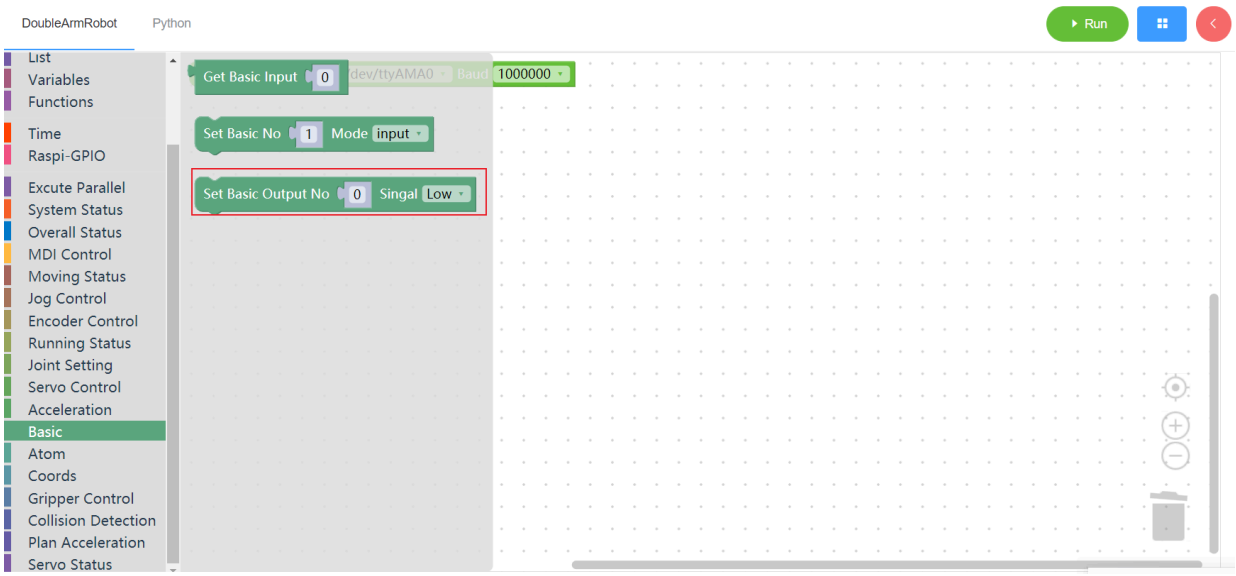


3.11.3 Set basic output

1. Runtime API Reference

- **Function:**Set basic output
- **Arguments:**
  - `PIN_NO` ：integer(0~20)
  - `PIN_SIGNAL` ：low/high
- **Return**：none

2.Block display



3.12 Atom

3.12.1 Get digital input

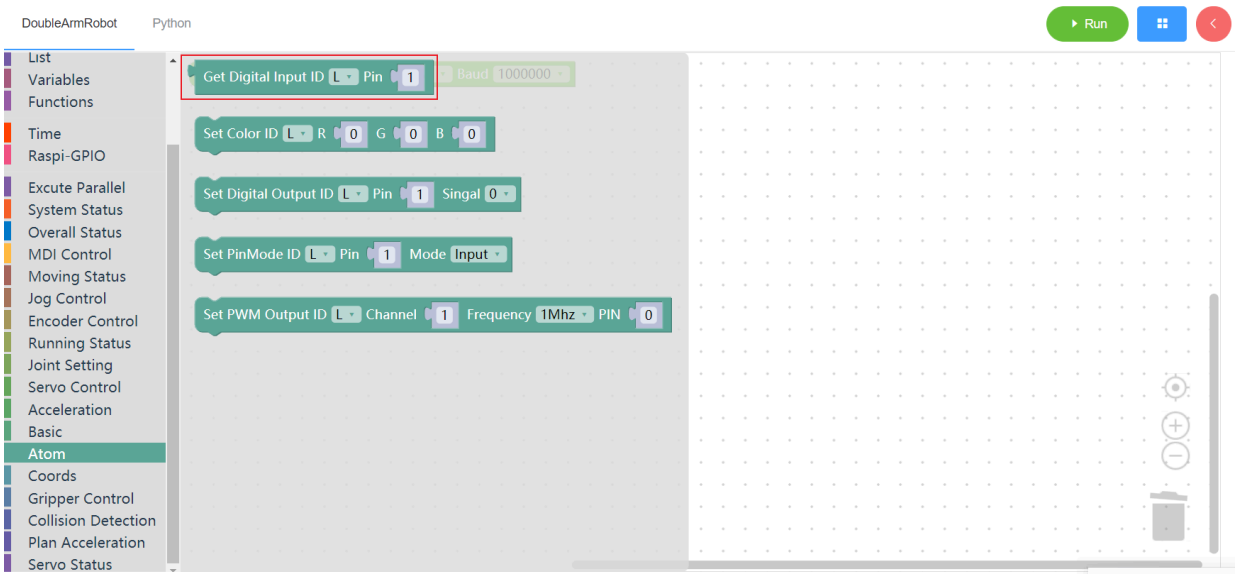
1. Runtime API Reference

- **Function:**Get digital input
- **Arguments:**

- `ID(L/R)` ：L for the left arm, R for the right arm
- `PIN_NO` ：integer(1~5)

- **Return** : singal value

2.Block display

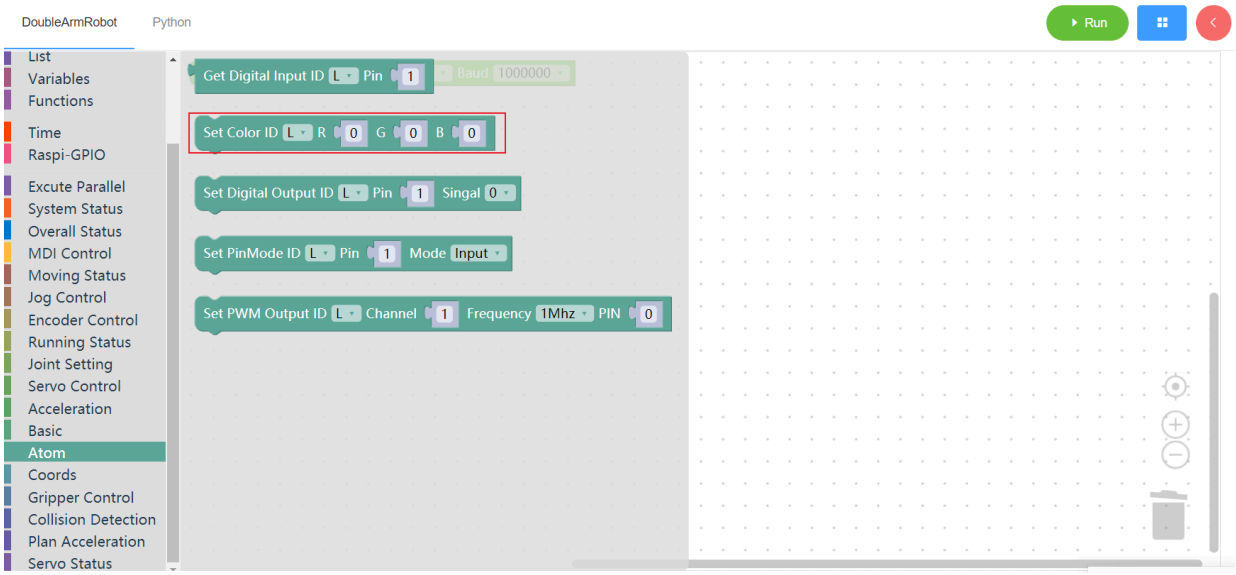


3.12.2 Set color

1. Runtime API Reference

- **Function:**Set the color of the arm at the top of the robot arm
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
  - `RED` ：integer(0~255)
  - `GREEN` : integer(0~255)
  - `BLUE` : integer(0~255)
- **Return**： none

2.Block display

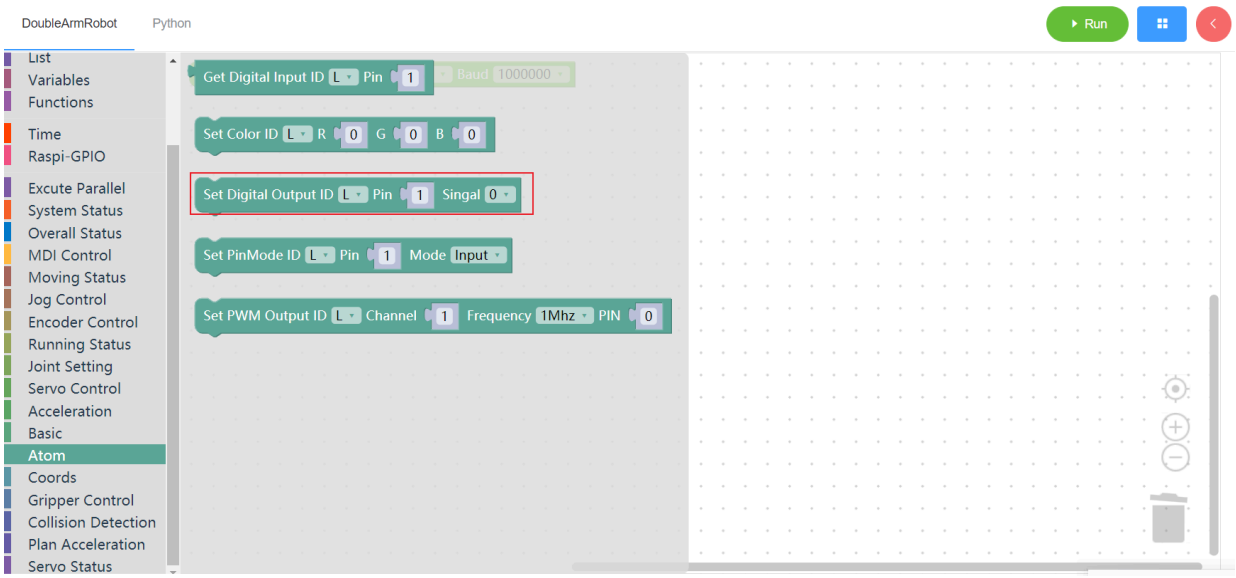


3.12.3 Set digital output

1. Runtime API Reference

- **Function:**Set digital output
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
  - `PIN_NO` ：integer(1~5)
  - `SIGNAL` ：0/1
- **Return**：none

2.Block display

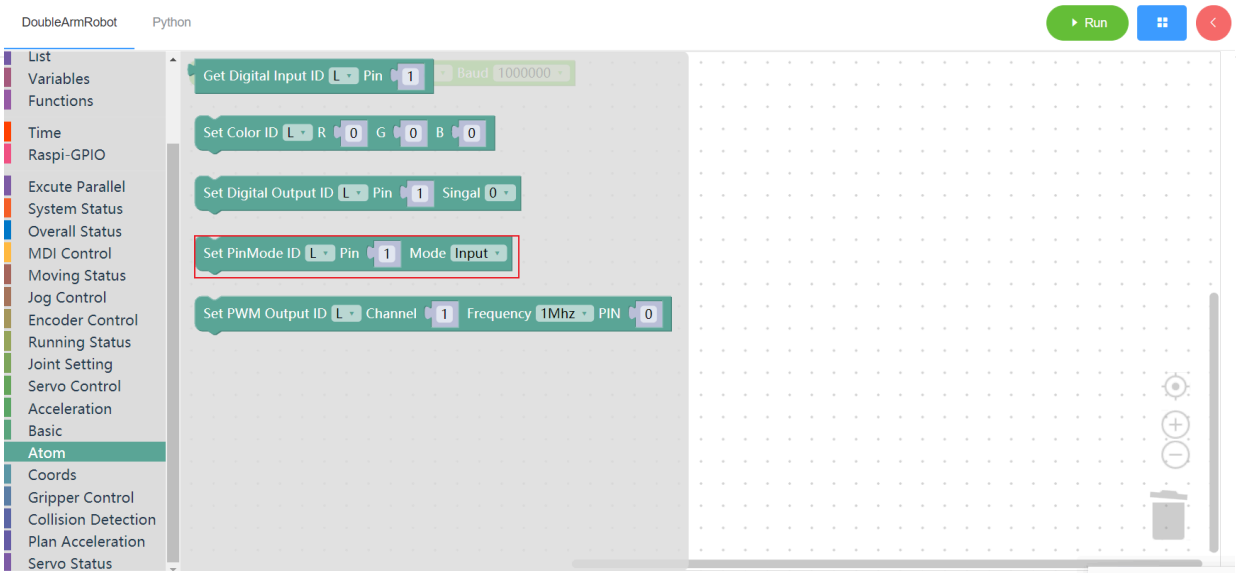


3.12.4 Set PinMode

1. Runtime API Reference

- **Function:**Sets the state mode of the specified pin in the atom.
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
  - `PIN_NO` ：integer(1~5)
  - `PIN_MODE` ：0 - input，1-output
- **Return**：none

2.Block display

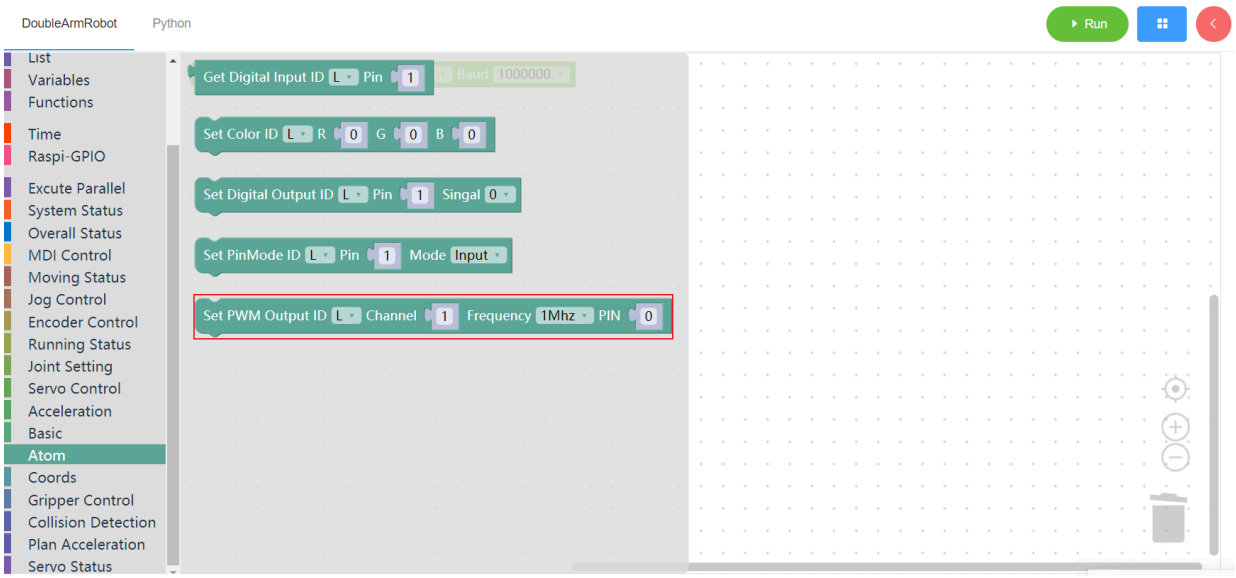


3.12.5 Set PWM output

1. Runtime API Reference

- **Function:**Set PWM output
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
  - `CHANNEL` ：integer(1~5)
  - `FREQUENCY` ：frequency 0 - 1Mhz，1- 10Mhz
  - `PIN_VAL` ：integer(0~100)
- **Return**：none

2.Block display



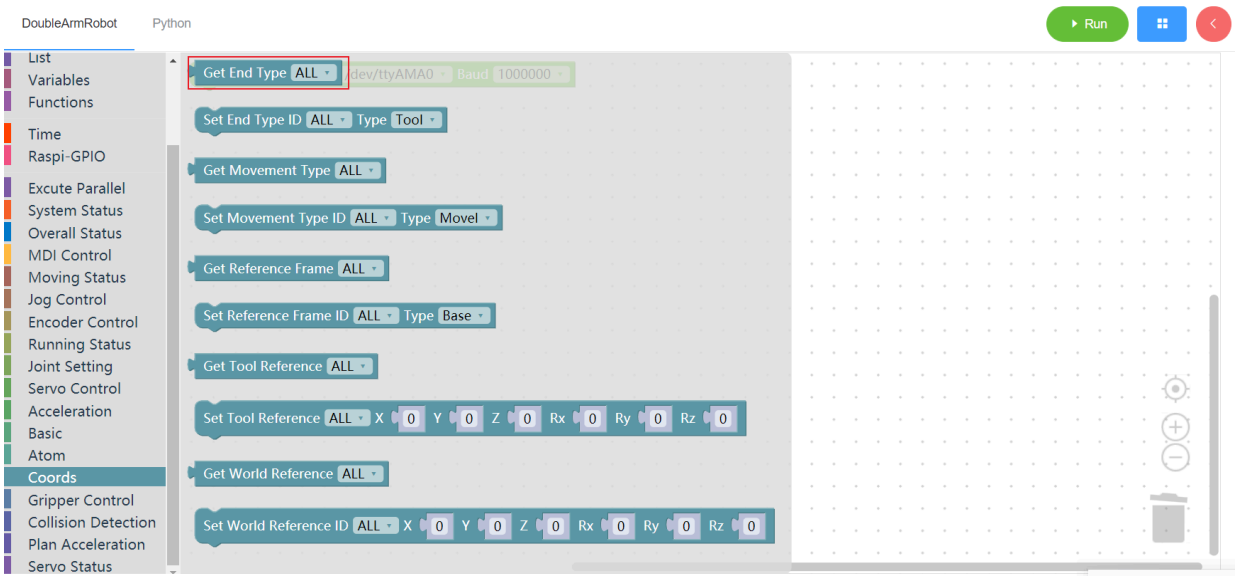
3.13 Coordinates

3.13.1 Get end coordinates system

1. Runtime API Reference

- **Function:**Get end coordinates system
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
- **Return**：
  - 0 - flange，1 - tool

2.Block display

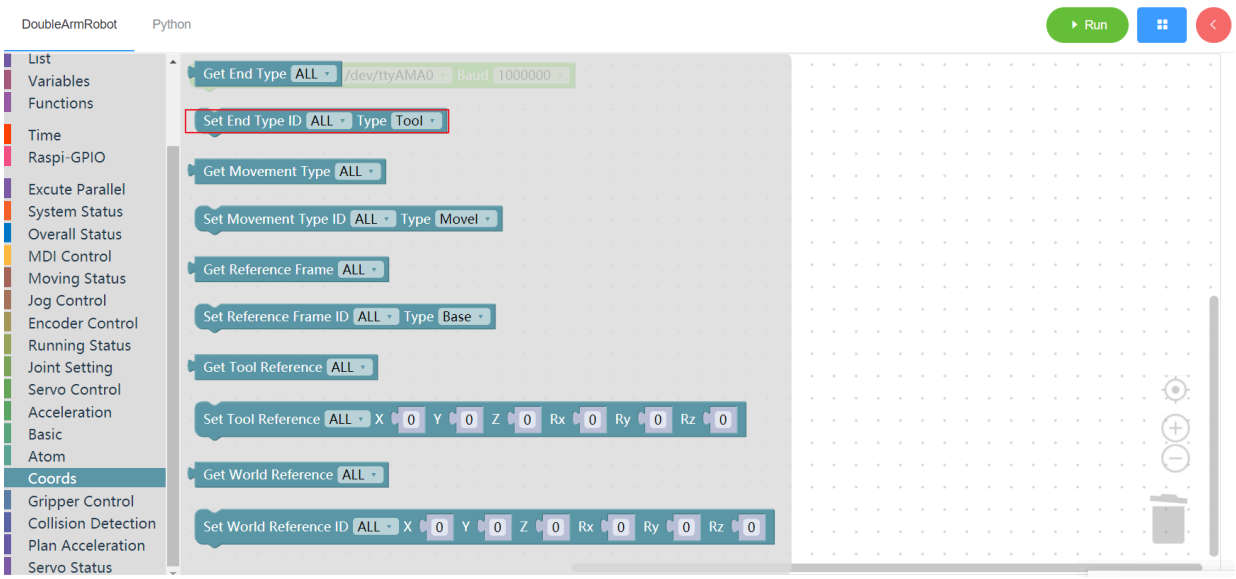


3.13.2 Set end coordinates system

1. Runtime API Reference

- **Function:**Set end coordinates system
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `END` ：0 - flange ， 1 - tool
- **Return**： none

2.Block display

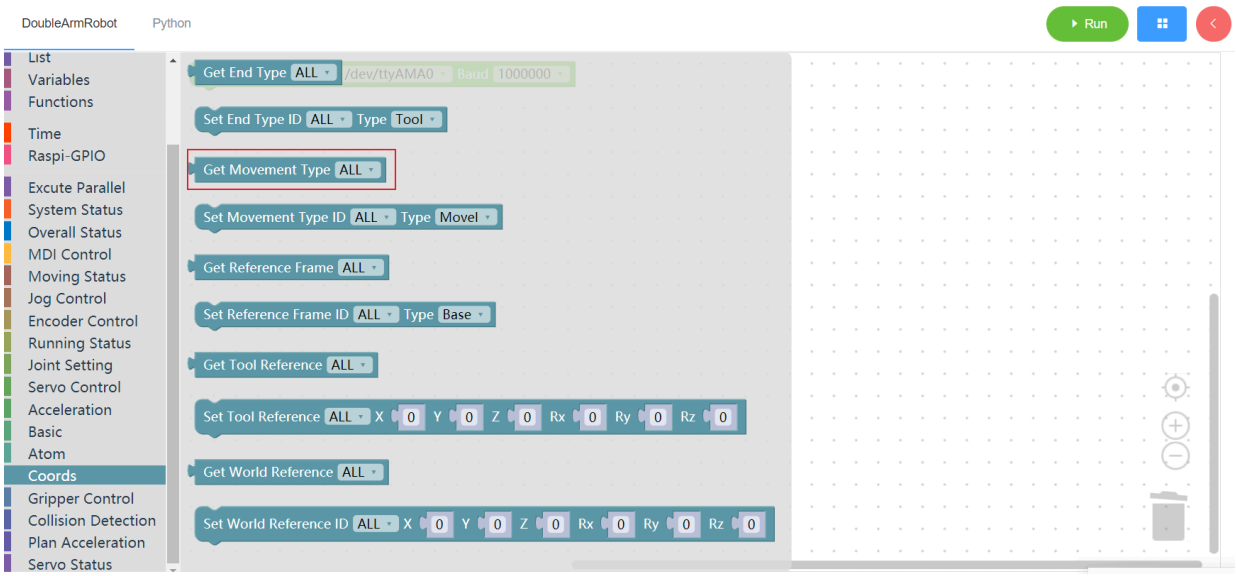


3.13.3 Get movement type

1. Runtime API Reference

- **Function:**Get movement type
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
- **Return**：
  - 1 - movel， 0 - moveJ

2.Block display

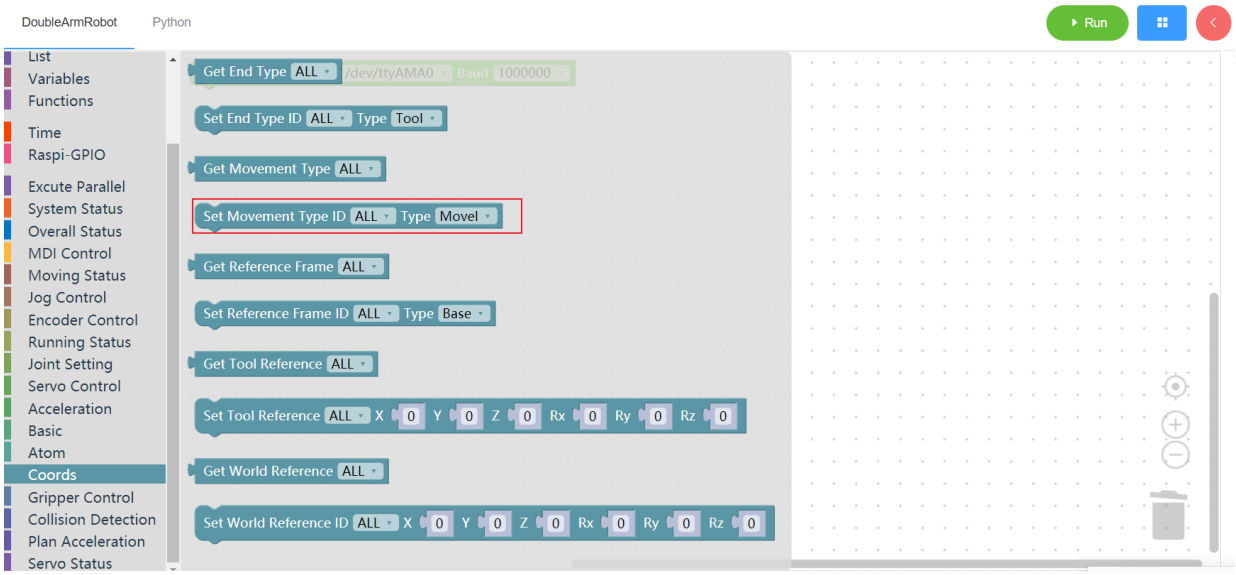


3.13.4 Set movement type

1. Runtime API Reference

- **Function:**Set movement type
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `TYPE` ：1 - movel， 0 - moveJ
- **Return**：none

2.Block display

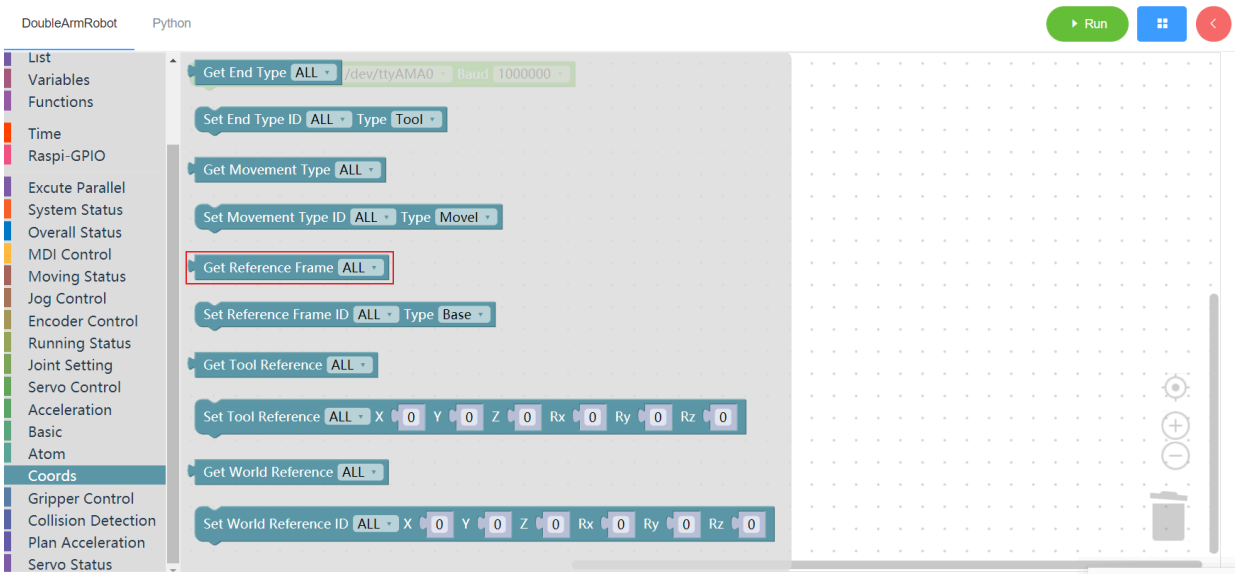


3.13.5 Get basic coordinates system

1. Runtime API Reference

- **Function:**Get basic coordinates system
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
- **Return**：
  - 0 - base ， 1 - tool

2.Block display



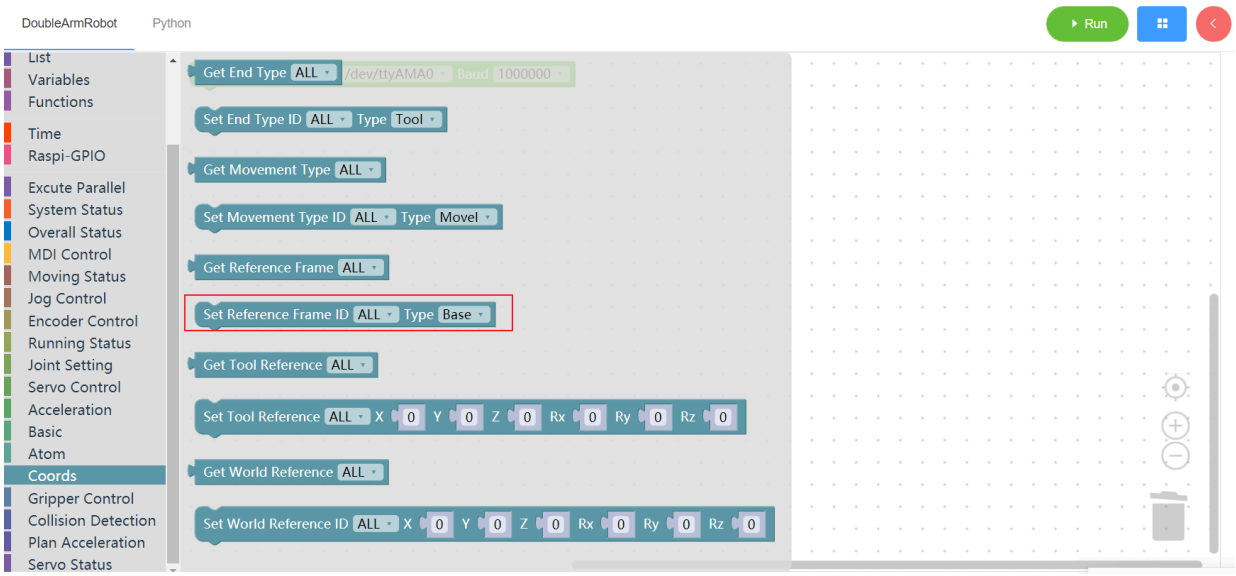
3.13.6 Set basic coordinates system

1. Runtime API Reference

- **Function:**Set basic coordinates system
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `TYPE` ：0 - base ， 1 - tool

- **Return**：none

2.Block display

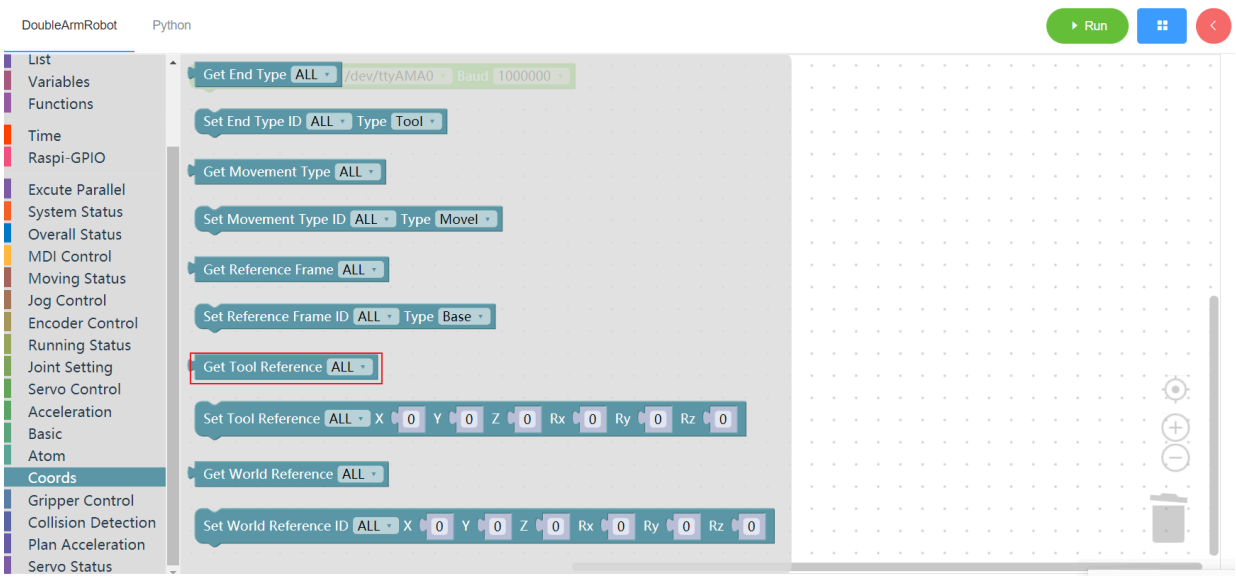


3.13.7 Get tool coordinates system

1. Runtime API Reference

- **Function:**Get tool coordinates system
- **Arguments:**
  - `ID(ALL/L/R)`：ALL for all arms，L for the left arm, R for the right arm
- **Return**：
  - unknown

2.Block display



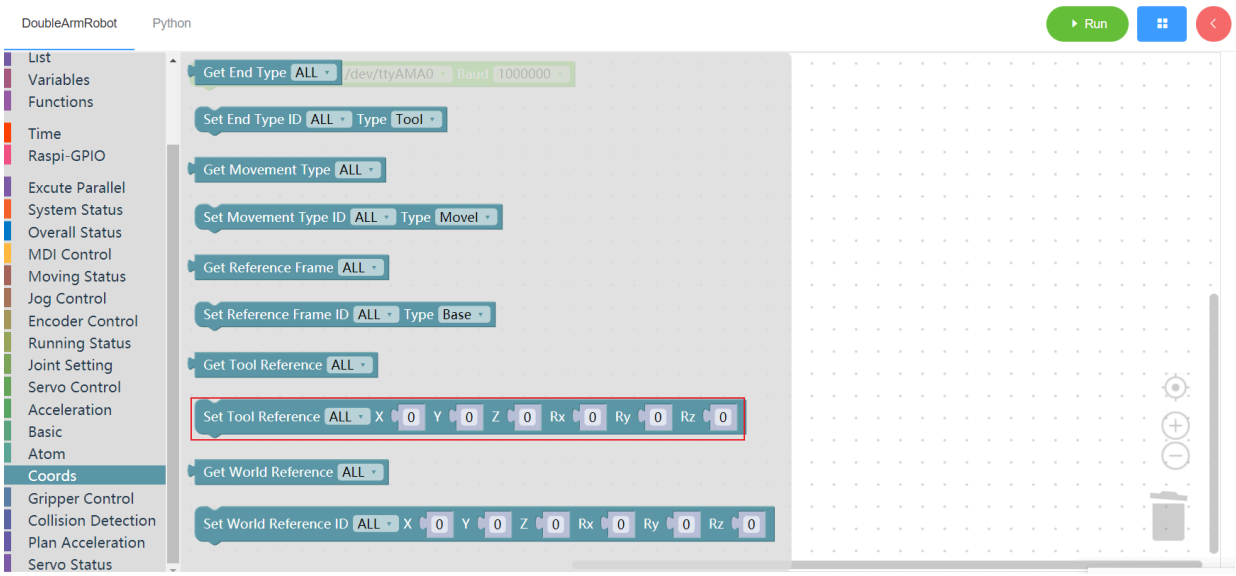
3.13.8 Set tool coordinates system

1. Runtime API Reference

- **Function:**Set tool coordinates system

- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `COORDS` ：coordinates list（lenth 6）
- **Return**：none

2.Block display

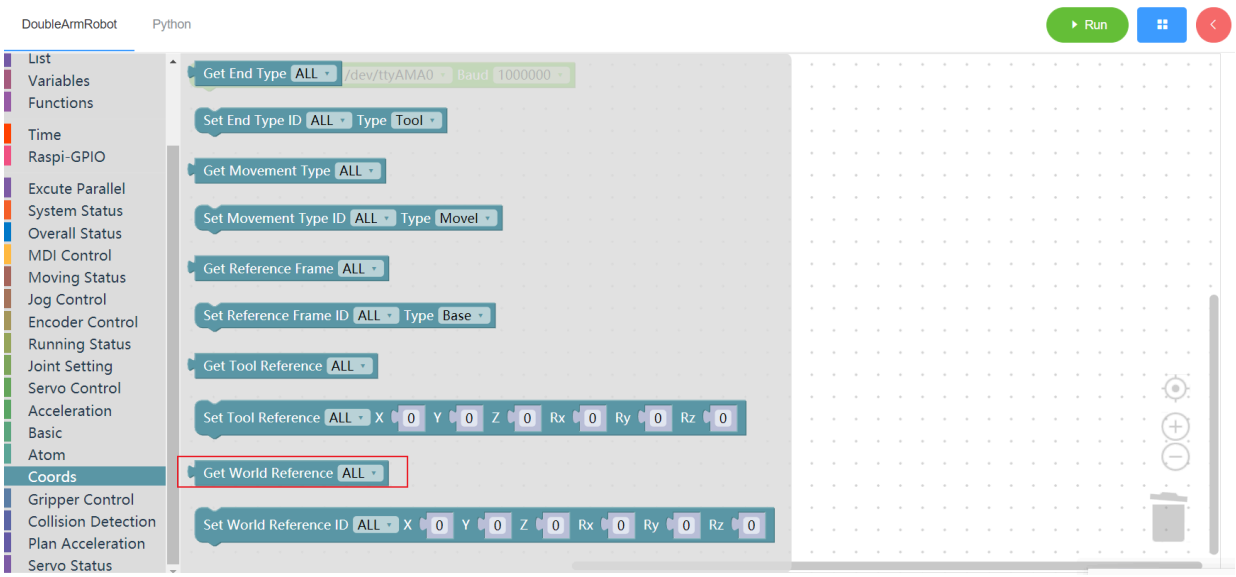


3.13.9 Get world coordinates system

1. Runtime API Reference

- **Function:**Get world coordinates system
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
- **Return**：
  - unknown

2.Block display

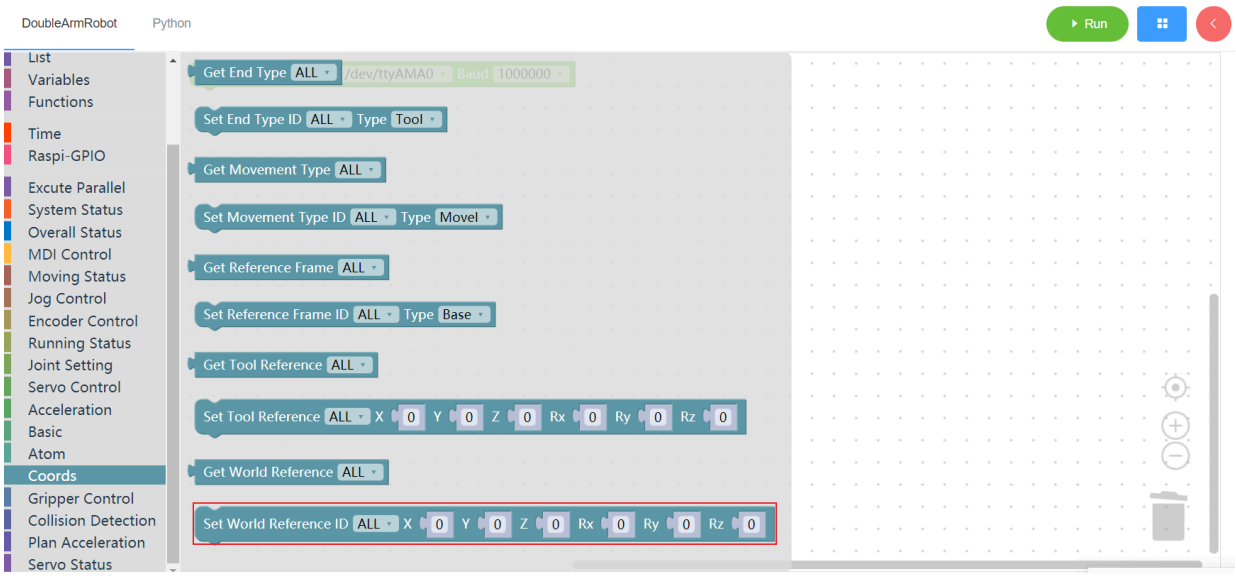


3.13.10 Set world coordinates system

1. Runtime API Reference

- **Function:**Set world coordinates system
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `COORDS` ：coordinates list(length 6)
- **Return**：none

2.Block display



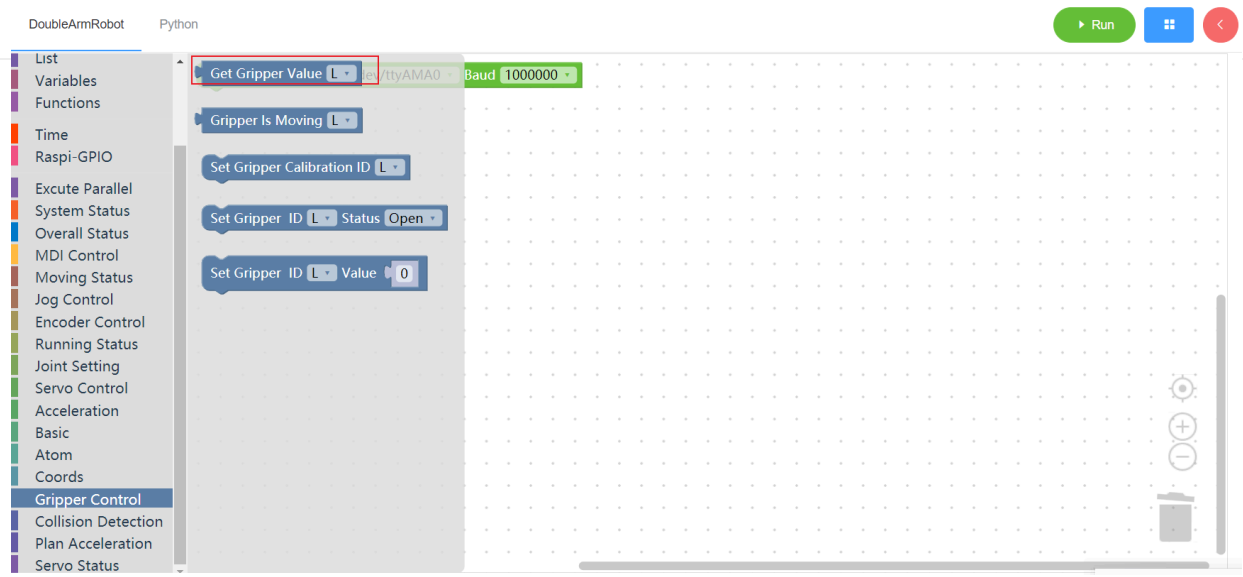
3.14 Gripper

3.14.1 Get gripper angle

1. Runtime API Reference

- **Function:**Get gripper value
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
- **Return**：
  - `GRIPPER_VALUE` ：gripper value

2.Block display

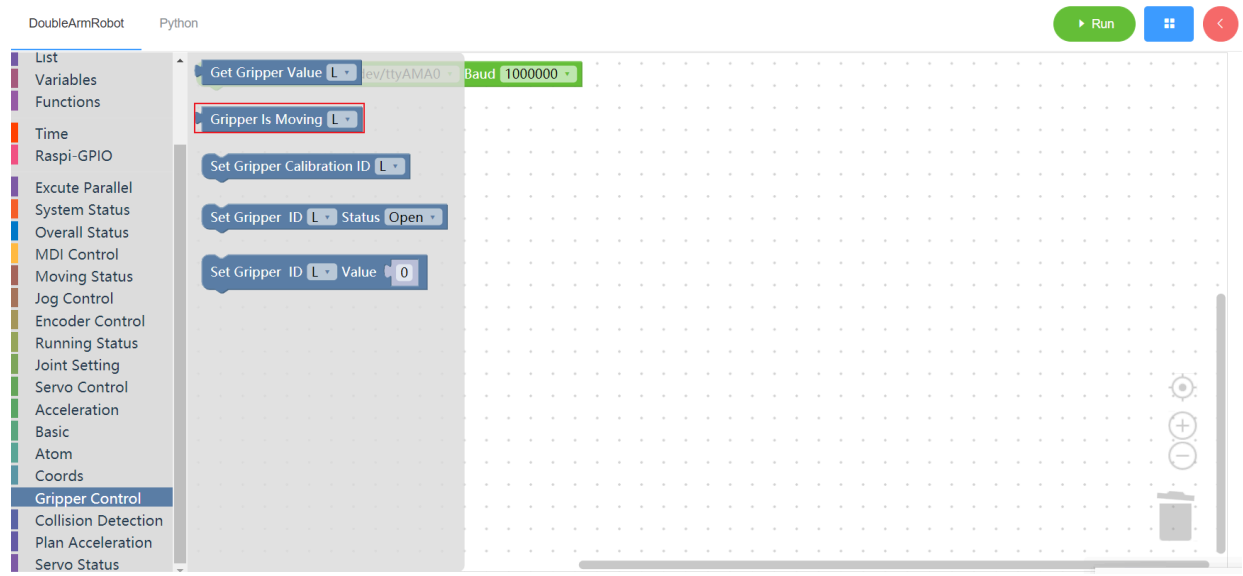


3.14.2 Check that the gripper is in motion

1. Runtime API Reference

- **Function:**Check whether the gripper of the mechanical arm is moving
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
- **Return**：
  - `VALUE` ：0 - did not move，1-moving , -1 error

2.Block display



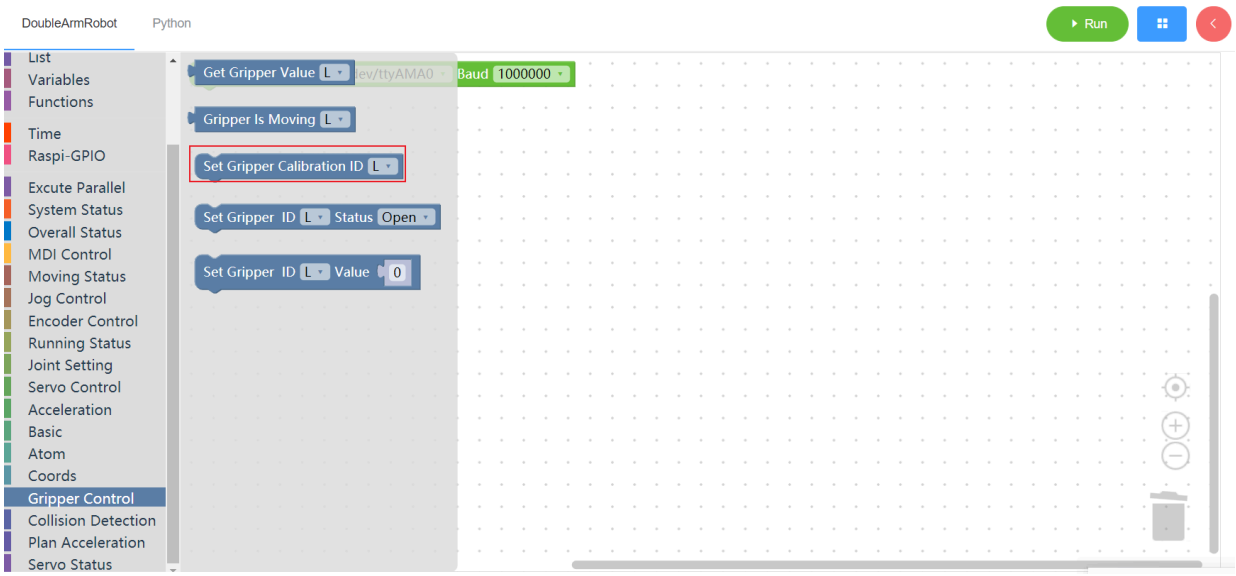
3.14.3 Set end claw zero

1. Runtime API Reference

- **Function:**Set the claw zero of the manipulator
- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm

- **Return**： none

2.Block display

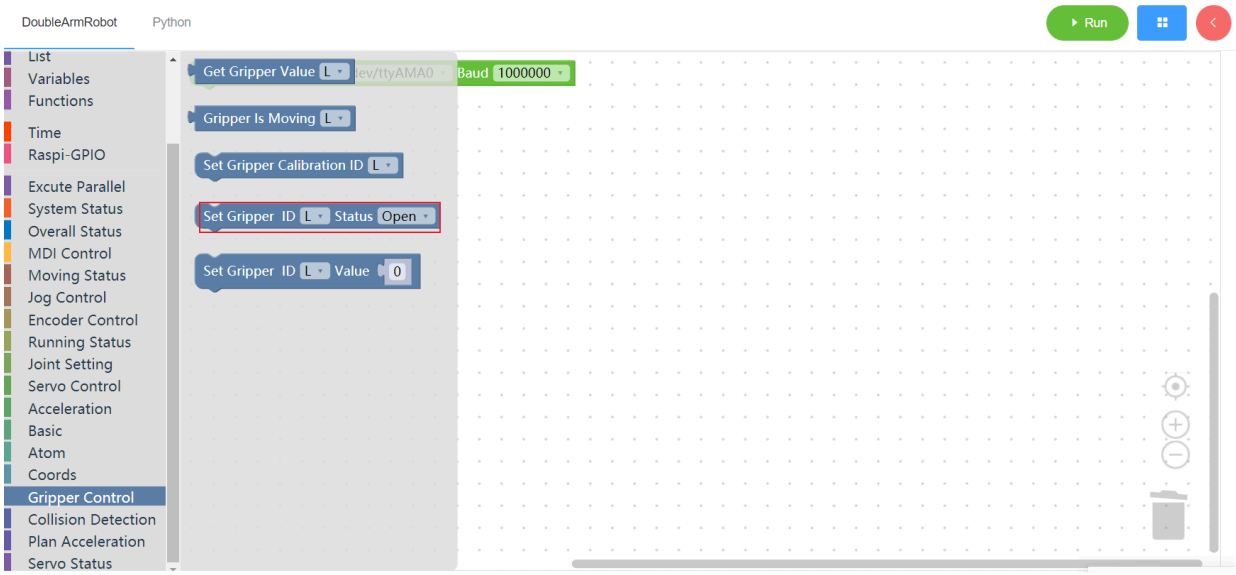


3.14.4 Set the terminal claw state

1. Runtime API Reference

- **Function:**Set the terminal claw state
- **Arguments:**
  - `ID(L/R)` ： L for the left arm, R for the right arm
  - `STATUS` ： 0 - close 1 - open
- **Return**： none

2.Block display



3.14.5 Set the claw range

1. Runtime API Reference

- **Function:**Set the claw value of the manipulator

- **Arguments:**
  - `ID(L/R)` ：L for the left arm, R for the right arm
  - `VALUE` ：integer(0~100)
- **Return**：none

2.Block display

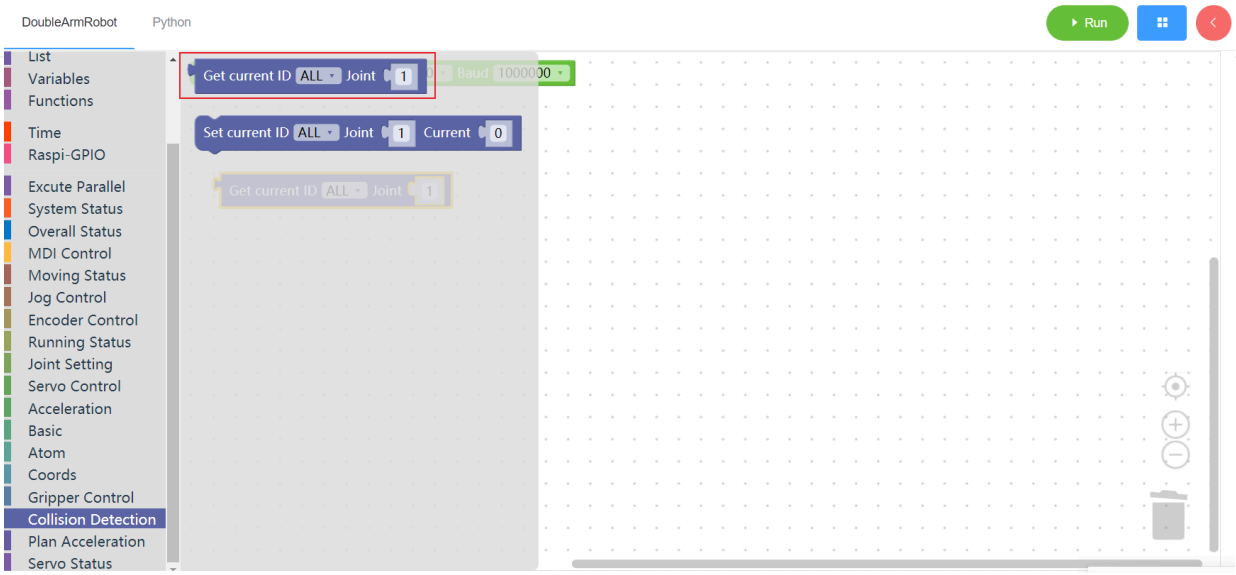
3.15 Collision detection

3.15.1 Get collision current

1. Runtime API Reference

- **Function:**Get collision current
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `JOINT_ID` ：joint id(1~6)
- **Return**：
  - `CURRENT` ：current

2.Block display

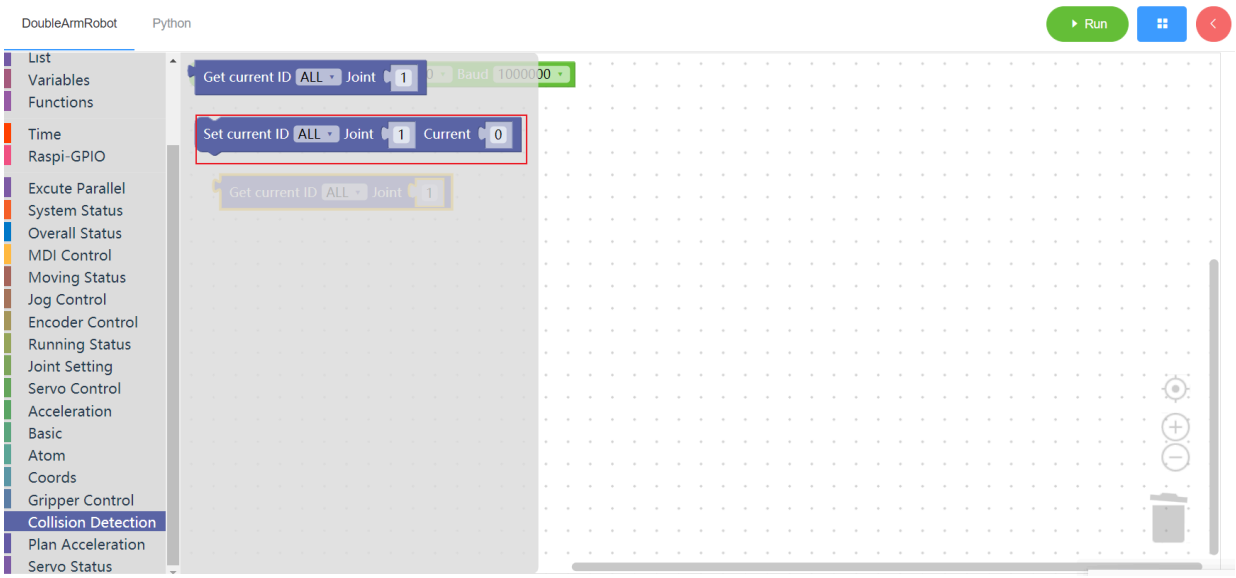


3.15.2 Set joint collision

1. Runtime API Reference

- **Function:**Set joint collision
- **Arguments:**
  - `ID(ALL/L/R)` ：ALL for all arms，L for the left arm, R for the right arm
  - `JOINT_ID` ：joint id(1~6)
  - `CURRENT` ：integer
- **Return**：none

2.Block display



3.16 Plan speed

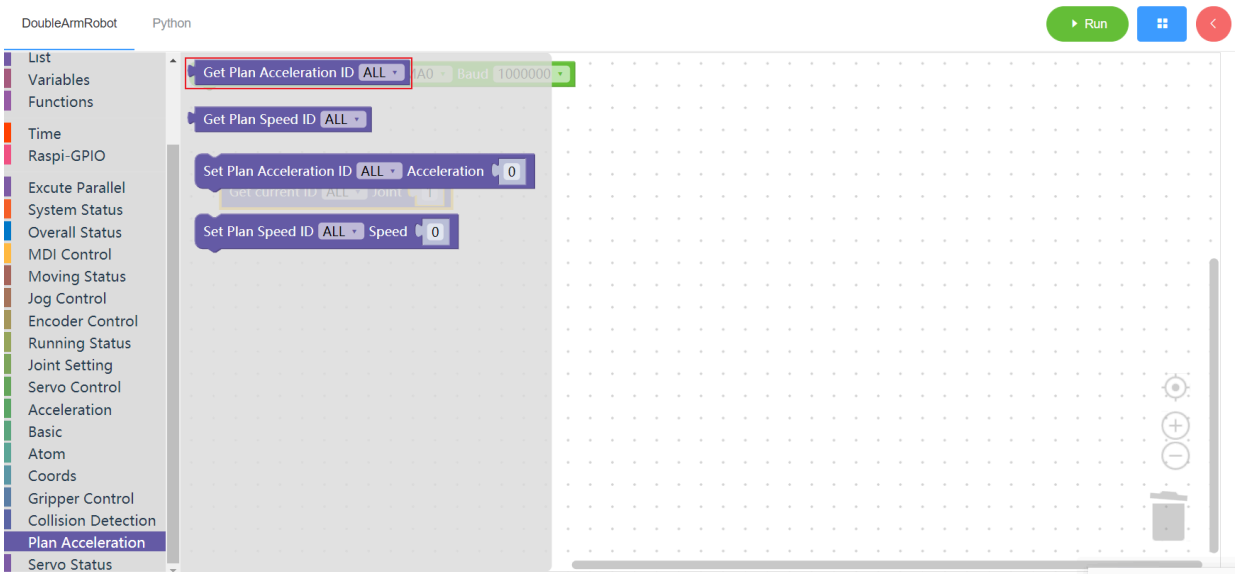
3.16.1 Get planned acceleration

1. Runtime API Reference

- **Function:**Get planned acceleration

- **Arguments:**
  - `ID(ALL/L/R/W)`：ALL for all arms，L for the left arm, R for the right arm，W for waist
- **Return**：
  - `PLAN_ACCELERATION`：planned acceleration list

2.Block display

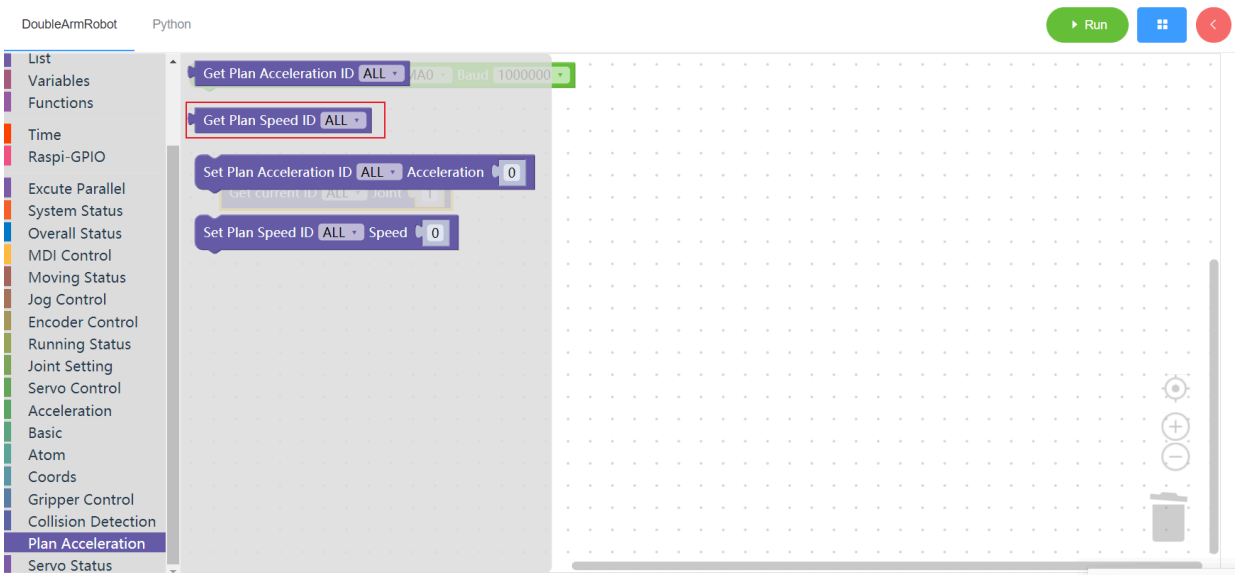


3.16.2 Get planned speed

1. Runtime API Reference

- **Function:**Get planned speed
- **Arguments:**
  - `ID(ALL/L/R/W)`：ALL for all arms，L for the left arm, R for the right arm，W for waist
- **Return**：
  - `PLAN_ACCELERATION`：planned speed list

2.Block display

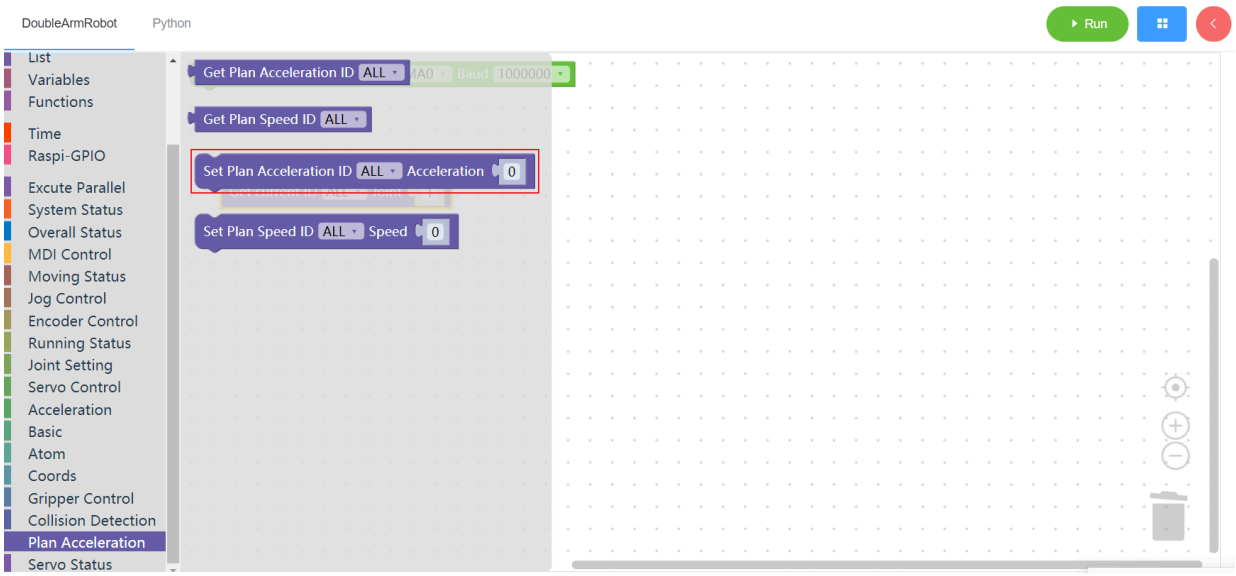


3.16.3 Set planned acceleration

1. Runtime API Reference

- **Function:**Set planned acceleration
- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for all arms，L for the left arm, R for the right arm，W for waist
  - `ACCELERATION` ：acceleration(0~100)
- **Return**：none

2.Block display

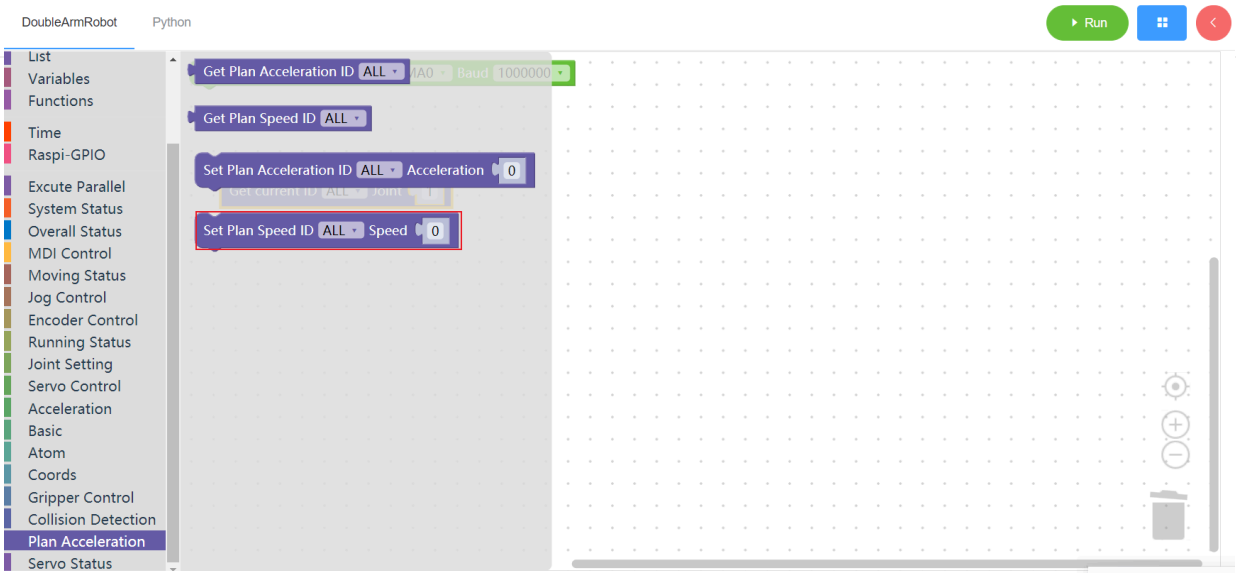


3.16.4 Set planned speed

1. Runtime API Reference

- **Function:**Set planned speed
- **Arguments:**
  - `ID(ALL/L/R/W)` ：ALL for all arms，L for the left arm, R for the right arm，W for waist
  - `SPEED` ：planned speed(0~100)
- **Return**：none

2.Block display



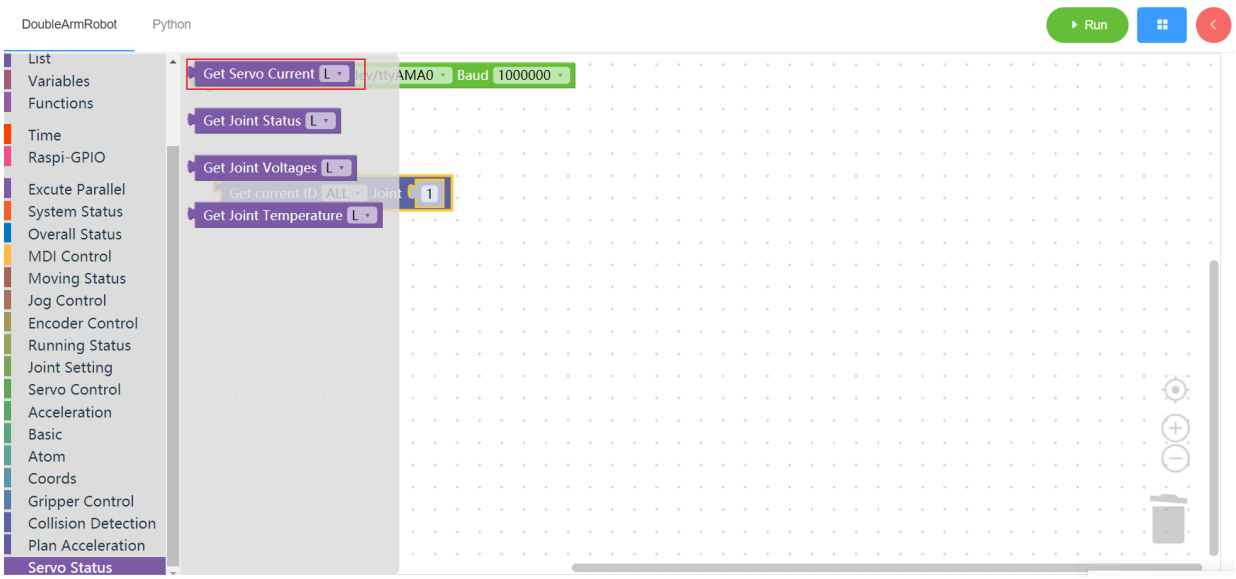
3.17 Servo status

3.17.1 Get joint current

1. Runtime API Reference

- **Function:**Get joint current
- **Arguments:**
  - `ID(L/R/W)`：L for the left arm, R for the right arm，W for waist
- **Return**：
  - `CURRENTS`：joint currents

2.Block display



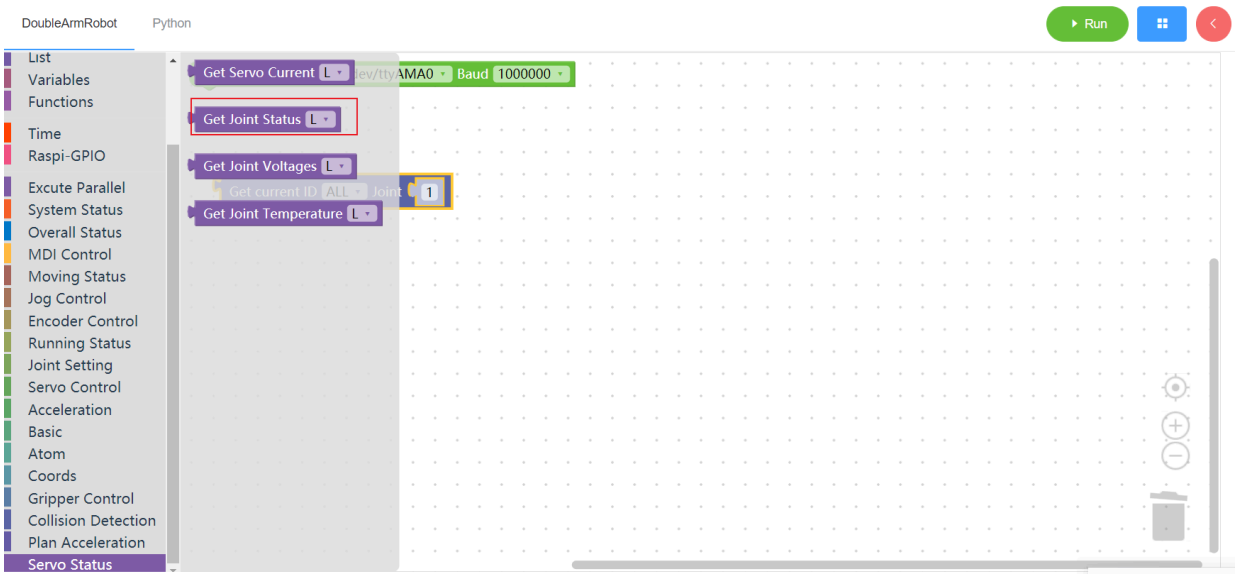
3.17.2 Get joint state

1. Runtime API Reference

- **Function:**Get joint state

- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
- **Return**：
  - `LIST` ：List (voltage, sensor, temperature, current, Angle, overload, 0 indicates none error)

2.Block display

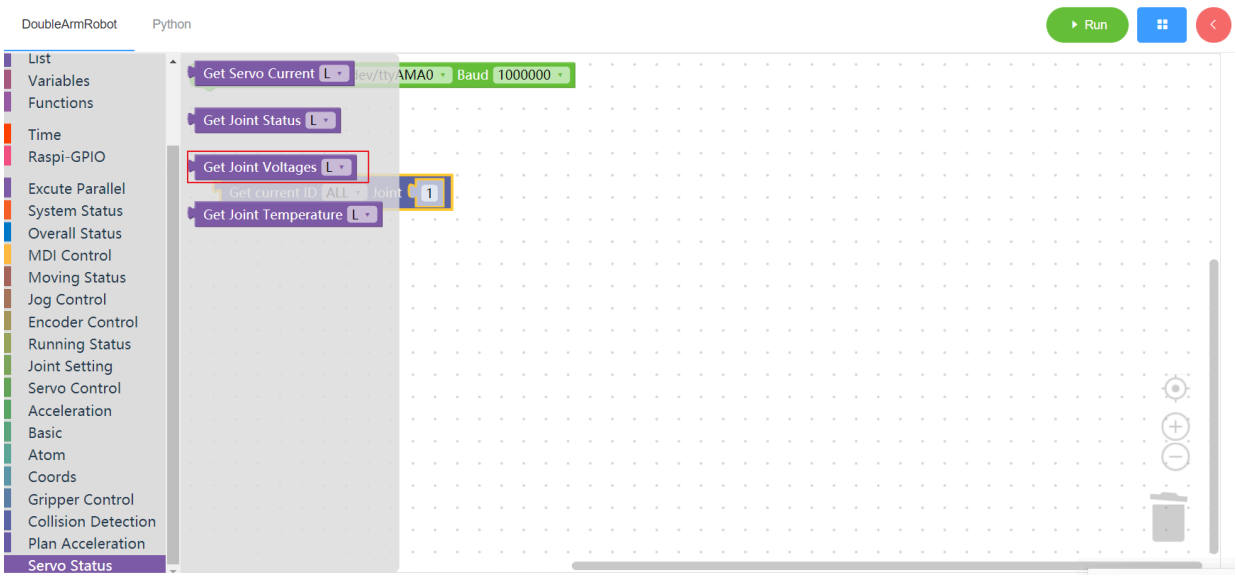


3.17.3 Acquisition of joint voltage

1. Runtime API Reference

- **Function:**Acquisition of joint voltage
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
- **Return**：
  - `VOLTS` ：voltage（less than 24V）

2.Block display

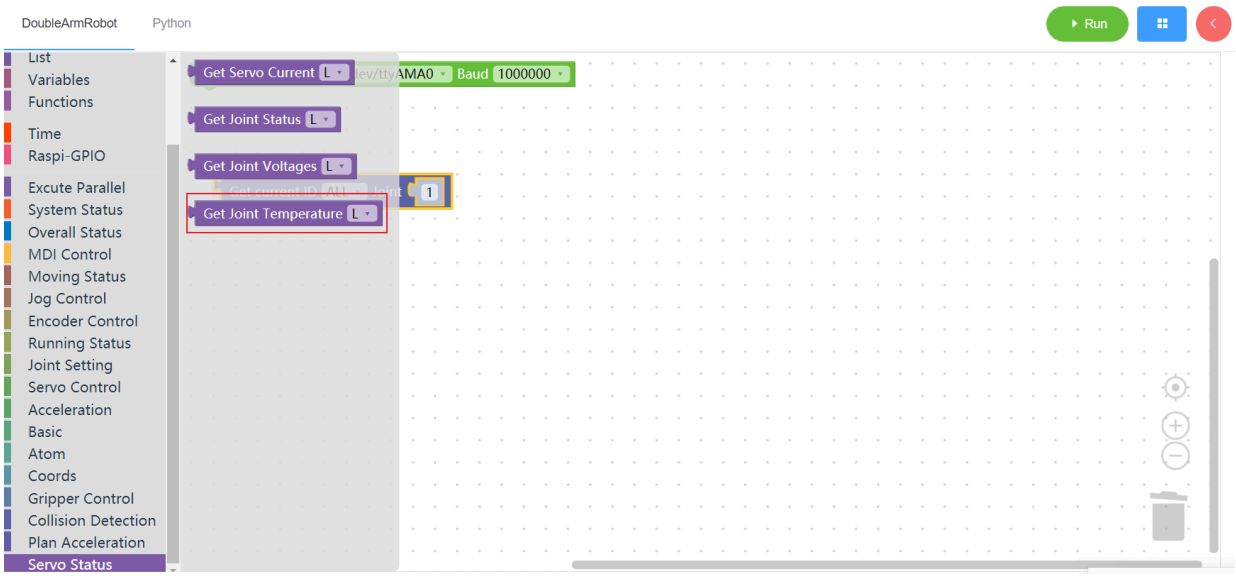


3.17.4 Get joint temperature

1. Runtime API Reference

- **Function:**Get joint temperature
- **Arguments:**
  - `ID(L/R/W)` ：L for the left arm, R for the right arm，W for waist
- **Return**：
  - `TEMPERATURE` ：temperature

2.Block display



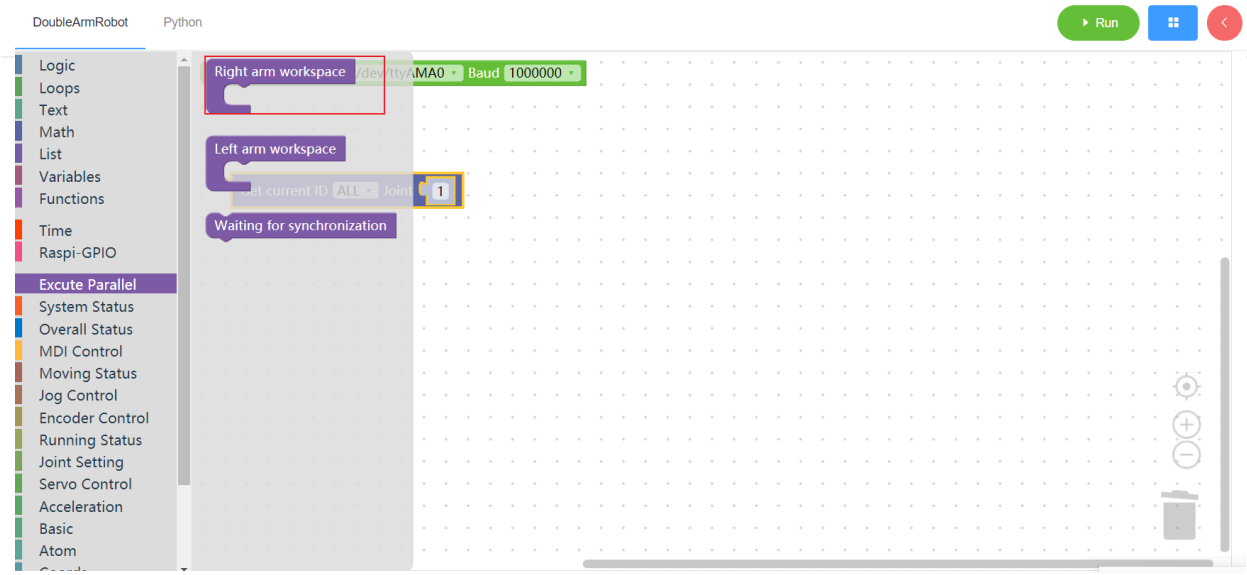
3.18 Execute in parallel

3.18.1 Right arm workspace

1. Runtime API Reference

- **Function:**The mechanical arm is used synchronously. During the synchronous execution, the building blocks of the left and right arms need to be dragged to the working area to ensure the simultaneous movement of the two mechanical arms
- **Arguments:**none
- **Return**： none

2.Block display

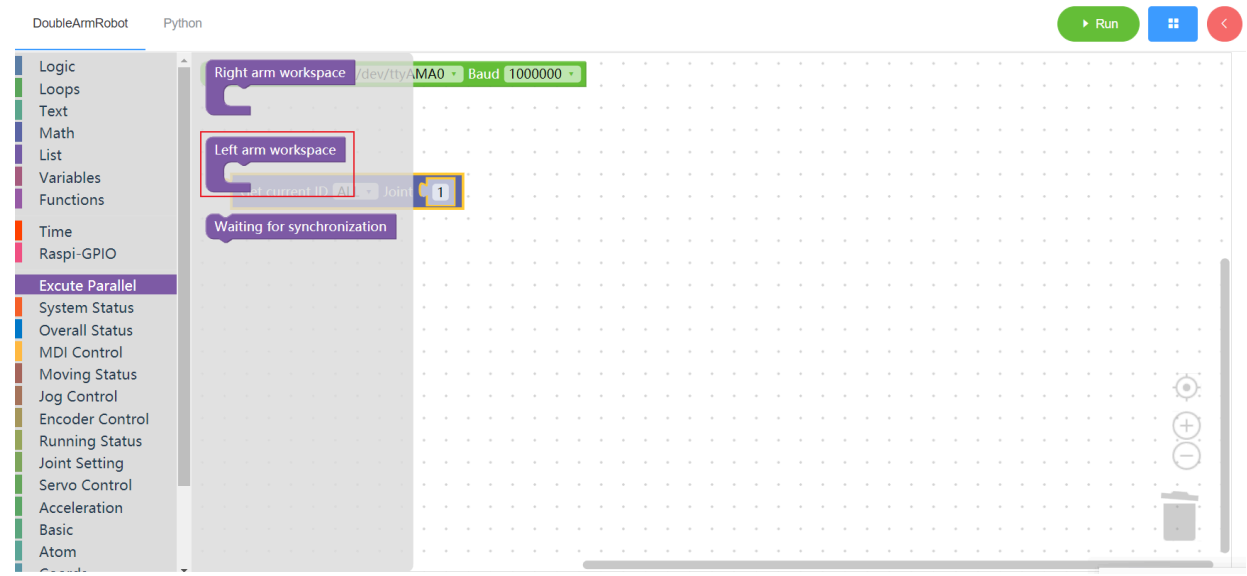


3.18.2 Left arm workspace

1. Runtime API Reference

- **Function:**The mechanical arm is used synchronously. During the synchronous execution, the building blocks of the left and right arms need to be dragged to the working area to ensure the simultaneous movement of the two mechanical arms
- **Arguments:**none
- **Return**： none

2.Block display



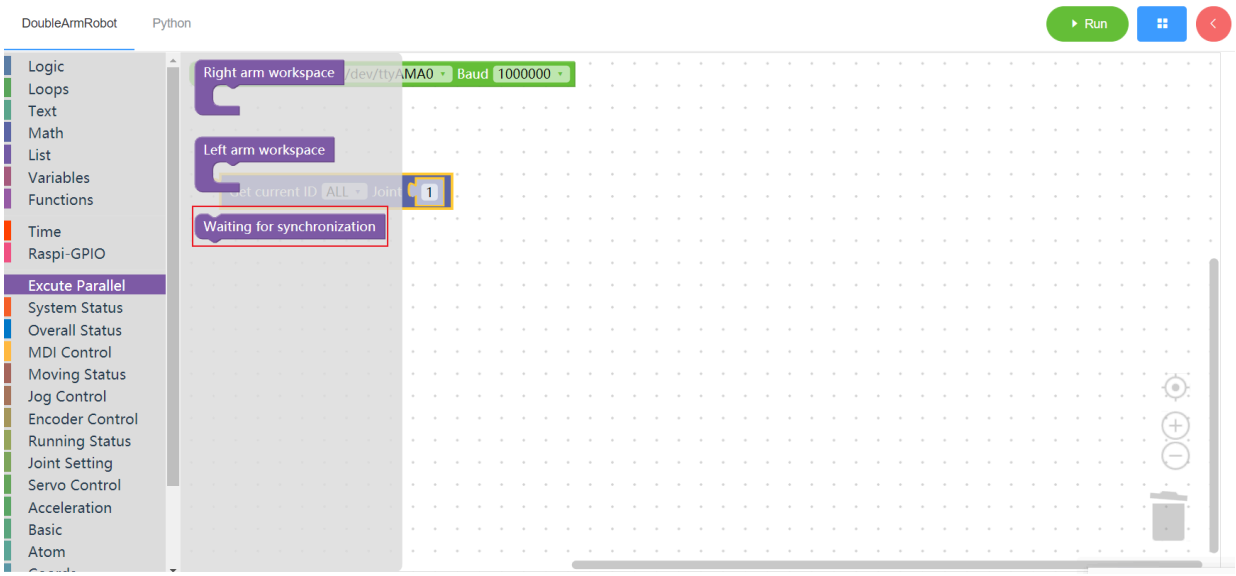
3.18.3 Sync dot

1. Runtime API Reference

- **Function:**It can only be used in the working area of left and right arms, and only takes effect when running both arms synchronously. Using the synchronization point, the mechanical arm can stop and wait for the completion of the execution of the other mechanical arm under specified conditions before performing subsequent operations

- **Arguments:**none
- **Return**： none

2.Block display



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.

Installing Python

- **Python** Python's official downloading address：https://www.python.org/downloads/
- **Python** downloading and Installation Tutorial for reference only

Applicable equipment:

- 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

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)

Joint Control

For a serial multi-joint robot, the control of the joint space is to control the variables of each joint so as to make each joint reaches a target position at a certain speed.

> **Notice:** When setting the angle, the values corresponding to different manipulators are different. Refer to the parameter introduction section for more information.

myPalletizer 260

Simple Demo

```
from pymycobot.mypalletizer import MyPalletizer260
import time
# import the project package


# Initiate a MyPalletizer260 object, M5 version
mc = MyPalletizer260("COM3",115200)


# PI version
# mc = MyPalletizer260("/dev/ttyAMA0",1000000)


# By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, spe
mc.send_angles([0, 0, 0, 0,], 50)
#  Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)


# Move joint 1 to the 50 position
mc.send_angle(1,20,50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)
# set variable "num"
num = 2
# set the number of loops
while num > 0:
    mc.send_angle(2,20,50)
    time.sleep(2)
    mc.send_angle(2,(-20),50)
    time.sleep(2)
    num -= 1


# make robot arms reach the specified position
mc.send_angles([-0.87, 41.66, -12.13, -0.17], 50)
#  Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

mechArm 270

Single-Joint Control

send_angle(id, degree, speed)

- **Function:** to sends a specified single joint motion to a specified angle.
- **Parameters:**
  - `id` : to stand for the joints of a robot arm. represented by numbers 1-6.
  - `degree` : means the angle of a joint.
  - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** 1

set_encoder(joint_id, encoder, speed)

- **Function:** to sends a specified single joint motion to a specified potential value.
- **Parameters:**
  - `joint_id` : to stand for the joints of a robot arm. represented by numbers 1-6.
  - `encoder` :means the potential value of the robot arm, ranging from 0 - 4096.
  - `speed` : means the movement speed of the robot arm, ranging from 1 to 100.
- **Return value:** 1

Multi-Joint Control

get_angles()

- **Function:** to get the angels of all joints.
- **Return Value:** `List` : a list of floating point values which represent the angles of all joints

send_angles(degrees, speed)

- **Function:** to send all angles to all joints.
- **Parameters:**
  - `degrees` : (List [float]) contains the angles of all joints.the length is 6; The representation method is [20, 20,20, 20, 20, 20]
  - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** 1

set_encoders(encoders, sp)

- **Function:** Send potential values to all joints of the robotic arm.
- **Parameters:**
  - `encoder` : means the potential of the robot arm, ranging from 0 - 4096, length is 6. The way to represent: [2048, 2048, 2048, 2048, 2048, 2048].
  - `sp` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return value:** 1

sync_send_angles(degrees, speed, timeout=15)

- **Function:** to send an angle synchronously; return when reaching a target point.
- **Parameters:**
  - `degrees` : A list of angle values of each joint `List[float]` .
  - `speed` : (`int`) means the movement speed of the robot arm, ranging from 0 to 100.
  - `timeout` : The default time is 15s.

- **Return value:** 1

get_radians()

- **Function:** to get the radian of all joints.
- **Return value:** `list` : a list containing radian values of all joints

send_radians(radians, speed)

- **Function:** to send radian values to all joints.
- **Parameters:**
  - `radians` : `list` : a list containing radian values of all joints, ranging from -5 to 5.
- **Return value:** 1

Simple Demo

```
from pymycobot.mecharm270 import MechArm270

import time

# MechArm270 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:
#       mecharm-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       mecharm-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initiate a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

# PI version
# mc = MechArm270("/dev/ttyAMA0", 1000000)

#By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, 0, 0
mc.send_angles([0, 0, 0, 0, 0, 0], 50)

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

# Move joint 1 to the 90 position
mc.send_angle(1, 90, 50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# The following code can make the robotic arm swing left and right
# set the number of loops
while num > 0:

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

    # Set the waiting time to ensure that the robotic arm has reached the specified position
```

```
    time.sleep(1.5)

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

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

    num -= 1

#Make the robotic arm retract. You can manually swing the robotic arm, and then use the get_angles() function to get the c
# use this function to let the robotic arm reach the position you want.
mc.send_angles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29], 50)

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

# Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

myBuddy

single joint control

send_angle(id, joint, angle, speed)

- **Function** Send one degree of joint to robot arm.
- **Parameters**
  - **id** – 1/2/3 (L/R/W)
  - **joint** – 1 ~ 6
  - **angle** – int
  - **speed** – 1 ~ 100
- **Returns**
  - None

get_angle(id, joint_id)

- **Function** Get the angle of a single joint
- **Parameters**
  - **id** (*int*) – 1/2/3 (L/R/W).
  - **joint_id** (*int*) – 1 - 7 (7 is gripper)

set_encoder(id, joint_id, encoder, speed)

- **Function** Set a single joint rotation to the specified potential value.

- **Parameters**
  - **id** – 1/2/3 (L/R/W).
  - **joint_id** – 1 - 6.
  - **encoder** – The value of the set encoder.
- **Returns**
  - None

multi-joint control

get_angles(id)

- **Function** Get the degree of all joints.
- **Parameters**

  id – 1/2 (L/R)
- **Returns**

  A float list of all degree.
- **Return type**

  list

send_angles(id, degrees, speed)

- **Function** Send all angles to the robotic arm
- **Parameters**
  - **id** – 1/2 (L/R).
  - **degrees** – [angle_list] len 6
  - **speed** – 1 - 100

set_encoders(id, encoders, speed)

- **Function** Set the six joints of the manipulator to execute synchronously to the specified position.
- **Parameters**
  - **id** – 1/2 (L/R).
  - **encoders** – A encoder list, length 6.
  - **speed** – speed 1 ~ 100

get_radians(id)

- **Function** Get the radians of all joints
- **Parameters**

  id – 1/2 (L/R)
- **Returns**

  A list of float radians [radian1, …]

- **Return type**

  list

send_radians(id, radians, speed)

- **Function** Send the radians of all joints to robot arm
- **Parameters**
  - **id** – 1/2 (L/R).
  - **radians** – a list of radian values(List[float]), length 6
  - **speed** – (int)1 ~ 100

Simple Demo

```
from pymycobot.mybuddy import MyBuddy
import time
mc = MyBuddy("/dev/ttyACM0", 115200)

# Send angles to the six joints of the left arm
mc.send_angles(1, [0, 0, 0, 0, 0, 0], 50)
time.sleep(3)

# Send the angle to the first joint of the left arm
mc.send_angle(1, 1, 90, 50)
time.sleep(2)

# Get the joint angle of the left arm
angles = mc.get_angles(1)
print("left angles: ",angles)

# Relax all joints of the left arm. Before running this command, please support the left arm with your hand to prevent it
mc.release_all_servos(1)
```

myArm

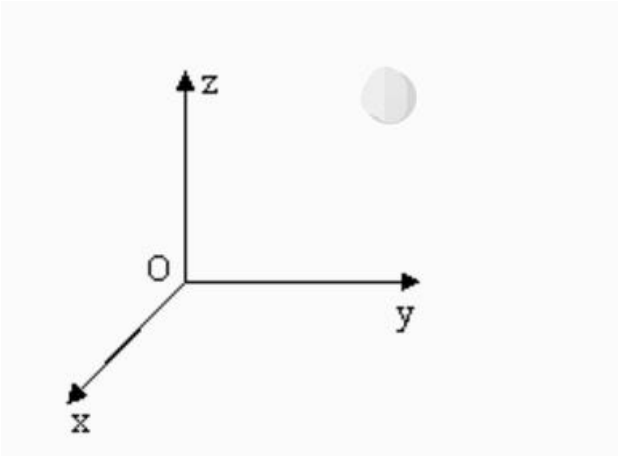
Simple Demo

```
from pymycobot.myarm import MyArm
from pymycobot.genre import Angle
import time
# import the project package

# Initiate a MyArm object
mc = MyArm("/dev/ttyAMA0", 115200)
# By passing the angle parameter, let each joint of the robotic arm move to the position corresponding to [0, 0, 0, 0, 0,
mc.send_angles([0, 0, 0, 0, 0, 0, 0], 50)
#  Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)

# Move joint 1 to the 90 position
mc.send_angle(1,90,50)
# Set the waiting time to ensure that the robotic arm has reached the specified position
time.sleep(2)
# set variable "num"
num = 2
# set the number of loops
while num > 0:
    mc.send_angle(2,20,50)
    time.sleep(2)
    mc.send_angle(2,(-20),50)
    time.sleep(2)
    num -= 1

# make robot arms reach the specified position
mc.send_angles([-5.25,-30,-20,-12,-10,-10,10],50)
#  Let the robotic arm relax, you can manually swing the robotic arm
mc.release_all_servos()
```

Coordinate Control

It is mainly used to make intelligent route planning to move the robot arms from one position to another specified position. The coordinate is `[x, y, z, rx, ry, rz]`. `[x,y,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). The above simple explaination helps you to use functions better.

> **Note:** When setting the coordinates, different series of manipulators have different joint structures. For the same set of coordinates, different series of manipulators will show different postures.



myPalletizer 260

Simple Demo

```
from pymycobot.mypalletizer import MyPalletizer260
import time
# import the project package

# Initiate a MyPalletizer260 object, M5 version
mc = MyPalletizer260("COM3", 115200)

# PI version
  # mc = MyPalletizer260("/dev/ttyAMA0", 1000000)

# # Get the current coordinates and pose of the head
coords = mc.get_coords()
print(coords)

#Plan the route at random, let the head reach the coordinates of [57.0, -107.4, 316.3] in an non-linear manner at the sp
mc.send_coords([187.8, 42.1, 183.3, -159.6], 80, 0)
# wait for 2 seconds
time.sleep(2)

# Plan the route at random, let the head reach the coordinates of [207.9, 47, 49.3,-159.69] in an non-linear manner at t
mc.send_coords([207.9, 47, 49.3,-159.69], 80, 0)
# wait for 2 seconds
time.sleep(2)

#To change only the x-coordinate of the head, set the x-coordinate of the head to 20. Let it plan the route at random an
mc.send_coord(1, 20, 50)
```

mechArm 270

Single-Parameter Coordinate

send_coord(id,coord,speed)

- **Function:** to send a single coordinate value to the robot arm to make it move.
- **Parameter:**
  - `id` : 1-6 represents the coordinates of the robotic arm. For example, you can fill in 1 for X-axis, 2 for Y-axis, and so on.
  - `coord` : Input the coordinate value you want.
  - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
- **Return Value:** 1

Multiple parameter coordinates

get_coords()

- **Function:** to obtain the current coordinate and posture.
- **Return Value:** `list` : a list containing coordinates and postures.
  - Six axes: The length is 6, and they are `[x, y, z, rx, ry, rz]` in order.

send_coords(coords, speed, mode)

- **Function:** to send the overall coordinates and postures to move the robot arm head from the original point to the point you have specified.
- **Parameters:**
  - `coords`：
    - Six axes: The length of the coordinate value of `[x, y, z, rx, ry, rz]` is 6.
    - Four axes: The length of the coordinate value of `[x,y,z,rx]` is 4.
  - `speed` : means the movement speed of the robot arm, ranging from 0 to 100.
  - `mode` : (`int`): The value is limited to 0 and 1.
    - 0 means that the movement path of the robot arm head is non-linear, i.e. the movement route is randomly planned just to make sure that the head moves to a specified point with a specified posture.
    - 1 means that the movement path of the robot arm head is linear, i.e. the movement route is intelligently planned just to make sure that the head moves to a specified point with a specified posture in a linear manner.
- **Return Value:** None

set_tool_reference(coords)

- **Function：** Set Tool coordinate system。
- **Parameters：**
  - `coords` : The coordinate value of [x, y, z, rx, ry, rz] has a length of 6, x, y, z ranging from - 280 to 280, and rx, ry, yz ranging from - 314 to 314
- **Return Value：** None

get_tool_reference()

- **Function：** Get Tool coordinate system。
- **Return Value：** Returns a coordinate list with a length of 6

get_world_reference()

- **Function：** Get World coordinate system。
- **Return Value：** Returns a coordinate list with a length of 6

set_world_reference(coords)

- **Function：** Set World coordinate system。
- **Parameters：**
  - `coords` : The coordinate value of [x, y, z, rx, ry, rz] has a length of 6, x, y, z ranging from - 280 to 280, and rx, ry, yz ranging from - 314 to 314
- **Return Value：** None

set_reference_frame(rftype)

- **Function：** Set Base coordinate system。
- **Parameters：**
  - `rftype` : 0 - Base coordinate system(default)，1 - World coordinate system
- **Return Value：** None

get_reference_frame()

- **Function：** Get Base coordinate system。

- **Return Value：** 0 - Base coordinate system，1 - World coordinate system，-1 - error

set_end_type(end)

- **Function：** Set end coordinate system。
- **Parameters：**
  - `end`: 0 - flange(default)，1 - tool
- **Return Value：** None

get_end_type()

- **Function：** Get end coordinate system
- **Return Value：** 0 - flange(default)，1 - tool, -1 - error

Simple Demo

```
from pymycobot.mecharm270 import MechArm270
import time

# MechArm270 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:
#       MechArm270-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       MechArm270-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initialize a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

# PI version
# mc = MechArm270("/dev/ttyAMA0", 1000000)
# Get the current coordinates and pose of the head
coords = mc.get_coords()
print(coords)

# Intelligently plan the route, let the head reach the coordinates of [152, -9.5, 220.8] in a linear manner, and maintain
mc.send_coords([152, -9.5, 220.8, 143.29, 2, 88], 80)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# Intelligently plan the route, let the head reach the coordinates of [124, -9.5, 232] in a linear way, and maintain the a
mc.send_coords([124, -9.5, 232, 143.29, 2, 88], 80)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# To change only the x-coordinate of the head, set the x-coordinate of the head to -40. Let it plan the route intelligentl
mc.send_coord(1, -40, 70)
```

myBuddy

One-parameter coordinates

send_coord(id, coord, data, speed)

- **Function** Send a single coordinate to the robotic arm
- **Parameters**
  - **id** – 1/2/3 (L/R/W).
  - **coord** – 1 ~ 6 (x/y/z/rx/ry/rz)
  - **data** – Coordinate value
  - **speed** – 0 ~ 100

get_coord(id, joint_id)

- **Function** Read a single coordinate parameter
- **Parameters**
  - **id** (*int*) – 1/2/3 (L/R/W).
  - **joint_id** (*int*) – 1 - 7 (7 is gripper)

Multiparameter Coordinates

send_coords(id, coords, speed, mode)

- **Function** Send all coords to robot arm.
- **Parameters**
  - **id** – 1/2 (L/R).
  - **coords** – a list of coords value(List[float]), length 6, [x(mm), y, z, rx(angle), ry, rz]
  - **speed** – (int) 0 ~ 100
  - **mode** – (int) 0 - moveJ, 1 - moveL, 2 - moveC

Simple Demo

```python
from pymycobot.mybuddy import MyBuddy
import time
mc = MyBuddy("/dev/ttyACM0", 115200)

# Get the coordinates and posture of the current head of the left arm
coords = mc.get_coords(1)
print(coords)

# Intelligently plan the route, let the head reach the coordinates of [57.0, -107.4, 316.3] in a linear manner, and mainta
mc.send_coords(1, [57.0, -107.4, 316.3, -93.81, -12.71, -163.49], 80, 1)

time.sleep(1.5)

# Intelligently plan the route, let the head reach the coordinates of [-13.7, -107.5, 223.9] in a linear manner, and maint
mc.send_coords(1, [-13.7, -107.5, 223.9, 165.52, -75.41, -73.52], 80, 1)

time.sleep(1.5)

# To change only the x-coordinate of the head of the left arm, set the x-coordinate of the head to -40. Let it plan the ro
mc.send_coord(1, 1, -40, 70)
```

myArm

Simple Demo

```
#from pymycobot.myarm import MyArm
#from pymycobot.genre import Coord
#import time

# Initialize a MyArm object
# Create object code here for windows version
mc = MyArm("/dev/ttyAMA0", 115200)

# Get the current coordinates and pose of the head
coords = mc.get_coords()
print(coords)

# Intelligently plan the route, let the head reach the coordinates of [57.0, -107.4, 316.3, -93.81, -12.71, -163.49] in
mc.send_coords([57.0, -107.4, 316.3, -93.81, -12.71, -163.49], 80,0)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# Intelligently plan the route, let the head reach the coordinates of [-13.7, -107.5, 223.9, 165.52, -75.41, -73.52] in
mc.send_coords([-13.7, -107.5, 223.9, 165.52, -75.41, -73.52], 80, 0)

# Set the wait time to 1.5 seconds
time.sleep(1.5)

# To change only the x-coordinate of the head, set the x-coordinate of the head to 20. Let it plan the route intelligent
mc.send_coord(Coord.X.value, 20, 70)
```

IO control

IO is the input and output of data. There are multiple pins on the Basic and Atom of our robot arm. The input and output modes can be set through the following function interface.

myPalletizer 260

Simple Demo

- 260-M5 version:

```
from pymycobot.mypalletizer260 import MyPalletizer260
import time
#Enter the above code to import the packages required by the project

# initiate MyPalletizer260, M5 version
mc = MyPalletizer260("COM3", 115200)

for count in range(5):
# set a loop
    mc.set_basic_output(2,0)
    # Let the basic2 position enter the working state
    mc.set_basic_output(5,0)
    # Let the basic5 position enter the working state
    time.sleep(2)
    #等待两秒
    mc.set_basic_output(2,1)
    #Let the basic2 position stop working
    mc.set_basic_output(5,1)
    #Let the basic5 position stop working
```

- 260-PI version:

```
from pymycobot.mypalletizer260 import MyPalletizer260
import time
#Enter the above code to import the packages required by the project

# initiate MyPalletizer260, PI version
mc = MyPalletizer260("/dev/ttyAMA0", 1000000)

# initialization
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# open suction pump
GPIO.output(20, 0)
GPIO.output(21, 0)
# wait 2 seconds
time.sleep(2)
# Turn off the suction pump
GPIO.output(20, 1)
GPIO.output(21, 1)
```

mechArm 270

Basic IO

get_basic_input(pin_no)

- **Function:** to obtain the working state of the bottom pin number

- **Parameters:** `pin_no` : represents the specific pin number at the bottom of the robot.
- **Return Value:** `pin_signal` (`int`) When the returned value is 0, it means running in the working state; when it is 1, it means the stop state.

set_basic_output(pin_no, pin_signal)

- **Function:** to set the working state of the bottom pin number.
- **Parameters:**
- `pin_no` (`int`). Only the numerical part of the numbers marked at the bottom of the equipment is taken.
- `pin_signal` (`int`): Inputting 0 means setting to the running state, which inputting 1 means setting to the stop state
- **Return Value:** 1

get_tof_distance()

- **Function:** to obtain a detected distance (An external distance detector is required).
- **Return value:** The detected distance value (in mm).

Atom IO

set_pin_mode(pin_no, pin_mode)

- **Function:** to set the state mode of the specified pin in the atom.
- **Parameters:**
  - `pin_no` (int): represents the specific pin number on the top of the robot.
  - `pin_mode` (int): limited to 0-2
    - 0 means setting it to the running state;
    - 1 means setting it to the stop state;
    - 2 means setting to the pull-up mode.
- **Return Value:** 1

set_digital_output(pin_no, pin_signa)

- **Function:** to set the working state of the end pin number.
- **Parameters:**
  - `pin_no` (`int`). Only the numerical part of the number marked at the end of the equipment is taken.
  - `pin_signal` (`int`). Inputting 0 means setting to the running state, while inputting 1 means setting to the stop state.
- **Return Value:** 1

get_digital_input(self, pin_no)

- **Function:** to obtain the working state of the end pin number.
- **Parameters:** `pin_no` : represents the specific pin number at the end of the robot.
- **Return Value:** `pin_signal` (`int`) When the returned value is 0, it means running in the working state; when it is 1, it means the stop state.

Raspberry Pi——GPIO

For Raspberry Pi version, use the following API.

Type the code at the beginning:

```
from pymycobot import MechArm270
import RPi.GPIO as GPIO
```

gpio_init()

- **Function:** Initialize GPIO module, set BCM mode.
- **Return value:** None

set_gpio_mode

- **Function:** Set Raspberry Pi GPIO Pin Mode
- **Parameter**
  - `mode` (`str`) Input: "BCM" or "BOARD" to enter the corresponding mode

set_gpio_output(pin_no, state)

- **Function:** Set the pin to high, low.
- **Parameter:**
  - `pin` (`int`) pin number.
  - `state`： 0 is set to low level 1 is set to high level (low level of suction pump starts working, high level stops working)
  - **Return Value:** None

get_gpio_in(pin_no)

- **Function:** Get the pin level status.
- **Parameter Description:**
  - `pin_no` (`int`) pin number.
- **Return Value:** 0 is low level 1 is high level

Simple Demo

270-M5 version:

```
from pymycobot.mecharm import MechArm270


import time
# Enter the above code to import the packages required by the project

# MechArm270 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:
#       MechArm270-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       MechArm270-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initialize a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)

for count in range(5):
# set a loop
    mc.set_basic_output(2,0)
    # Let the basic2 position enter the working state
    mc.set_basic_output(5,0)
    # Let the basic5 position enter the working state
    time.sleep(2)
    #等待两秒
    mc.set_basic_output(2,1)
    #Let the basic2 position stop working
    mc.set_basic_output(5,1)
    #Let the basic5 position stop working
```

270-Pi version:

```
from pymycobot.mecharm import MechArm270
import RPi.GPIO as GPIO
import time
#Enter the above code to import the packages required by the project

# MechArm270 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:
#       MechArm270-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270("/dev/ttyAMA0", 1000000)
#
# Initialize a MechArm270 object
# Create object code here for Raspberry Pi version
mc = MechArm270("/dev/ttyAMA0", 1000000)

# initialization
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# open suction pump
GPIO.output(20, 0)
GPIO.output(21, 0)
# wait 2 seconds
time.sleep(2)
# Turn off the suction pump
GPIO.output(20, 1)
GPIO.output(21, 1)
```

myBuddy

Atom IO

set_pin_mode(id, pin_no, pin_mode)

- **Function** Set the state mode of the specified pin in atom.
- **Parameters**

- **id** – 1/2 (L/R)
- **pin_no** (*int*) – pin number (1 - 5).
- **pin_mode** (*int*) – 0 - input, 1 - output

set_digital_output(id, pin_no, pin_signal)

- **Function** Set atom IO output level
- **Parameters**
  - **id** – 1/2 (L/R)
  - **pin_no** (*int*) – 1 - 5
  - **pin_signal** (*int*) – 0 / 1

get_digital_input(id, pin_no)

- **Function** singal value
- **Parameters**
  - **id** – 1/2 (L/R)
  - **pin_no** (*int*) – 1 - 5

set_pwm_output(id, channel, frequency, pin_val)

- **Function** PWM control
- **Parameters**
  - **id** – 1/2 (L/R)
  - **channel** (*int*) – IO number (1 - 5).
  - **frequency** (*int*) – clock frequency (0/1: 0 - 1Mhz 1 - 10Mhz)
  - **pin_val** (*int*) – Duty cycle 0 ~ 100: 0 ~ 100%

Raspberry Pi IO

set_gpio_input(pin)

- **Function** Set GPIO input value.
- **Parameters**

  pin – (int)pin number.

set_gpio_mode(pin_no, mode)

- **Function** Init GPIO module, and set BCM mode.
- **Parameters**
  - **pin_no** – (int)pin number.
  - **mode** – 0 - input 1 - output

set_gpio_output(pin, v)

- **Function** Set GPIO output value.

- **Parameters**
  - **pin** – (int)pin number.
  - **v** – (int) 0 / 1

set_gpio_pwm(pin, baud, dc)

- **Function** Set GPIO PWM value.
- **Parameters**
  - **pin** – (int)pin number.
  - **baud** – (int) 10 - 1000000
  - **dc** – (int) 0 - 100

Simple Demo

```
from pymycobot import MyBuddy
import time
mc = MyBuddy("/dev/ttyACM0")

# 设置树莓派IO 20为输出模式
mc.set_gpio_mode(20,  1)

mc.set_gpio_output(20, 1)
time.sleep(2)
mc.set_gpio_output(20, 0)
```

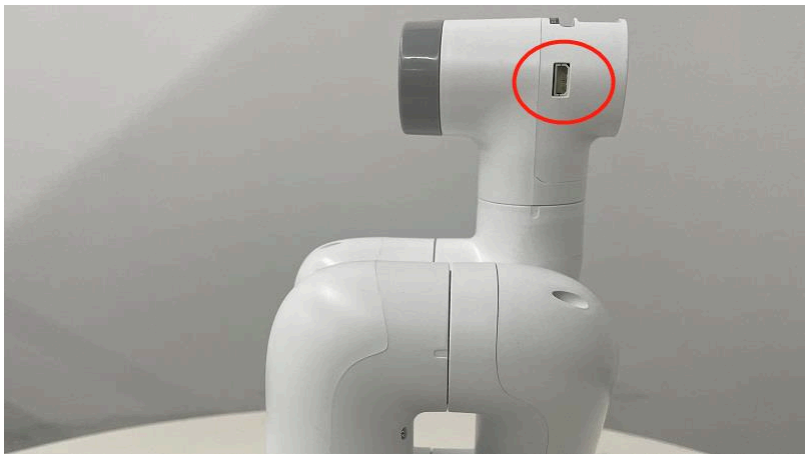
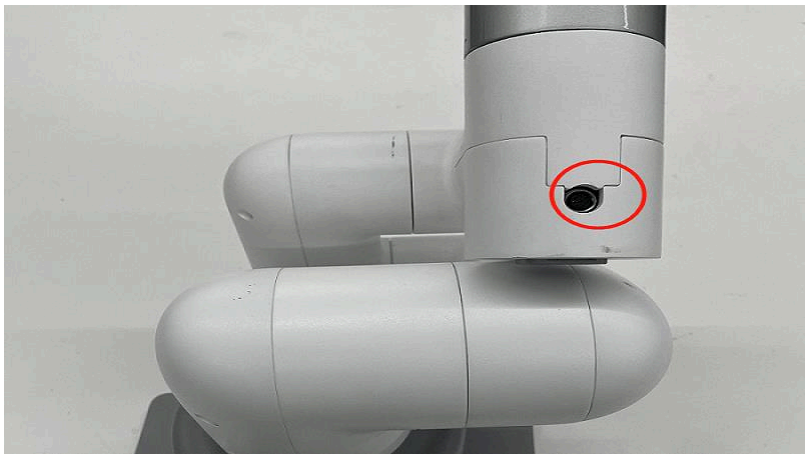
MyArm

Simple Demo

```
from pymycobot.myarm import MyArm
import time
#Enter the above code to import the packages required by the project


# Initialize a MyArm object
mc = MyArm("/dev/ttyAMA0", 115200)


# initialization
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)
# open suction pump
GPIO.output(20, 0)
GPIO.output(21, 0)
# wait 2 seconds
time.sleep(2)
# Turn off the suction pump
GPIO.output(20, 1)
GPIO.output(21, 1)
```

Gripper Control

First install and connect the gripper onto the robot arm. Different types of gripper is compatible with different types of robots. Refer to **2.8 Accessories** for more information.

> **Notice:**
>
> For MyCobot 280, the adaptive gripper is attached to Atom.
>
> 
>
> The electric gripper is attached to 495 port.
>
> 
>
> * MyCobot 280-m5 is not compatible with electric gripper, and MyCobot 320-m5 is only compatible with electric gripper.

myPalletizer 260

Simple Demo

```
from pymycobot.mypalletizer260 import MyPalletizer260
from pymycobot.genre import Angle
import time
#Enter the above code to import the packages required by the project

# initiate MyPalletizer260, M5 version
mc = MyPalletizer260("COM3", 115200)
# PI version
# mc = MyPalletizer260("COM3", 115200)

# let joint2 move to 30 degree at the speed of 50
mc.send_angle(2,30,50)
# waite for 2 seconds
time.sleep(2)

#set a variable num, and then set a loop
num = 5
while num > 0:
    #let gripper open at the speed of 70
    mc.set_gripper_state(0,70)
    # waite for 2 seconds
    time.sleep(2)
    # let gripper close at the speed of 70
    mc.set_gripper_state(1, 70)
    # waite for 2 seconds
    time.sleep(2)
    num -= 1
```

mechArm 270

Controlling Gripper

`is_gripper_moving( )`

- **Function:** Determine whether the gripper is running
- **return value:**
  - `0` : Indicates that the gripper of the robot arm is not running
  - `1` : Indicates that the gripper of the robot arm is running
  - `-1` : indicates an error

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

- **function**: Adaptive gripper enable
- **Parameters**:
  - `flag (int)` : 0 - open 1 - close, 254 - release

- `speed (int)` : 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
- `is_torque` (int): Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-device Atom firmware version ≥ 1.3**)
  - `0` : Non-force-controlled gripper
  - `1` : Force-controlled gripper

- **Return value：**
  - `1` : completed

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

- **function**: Set the gripper value
- **Parameters**:
  - `gripper_value (int)` : 0 ~ 100
  - `speed (int)` : 1 ~ 100
  - `gripper_type (int)` :
    - `1` : Adaptive gripper (default state is 1)
    - `3` : Parallel gripper
    - `4` : Flexible gripper
  - `is_torque` (int): Whether the gripper is force-controlled. This parameter can be omitted if no type parameter is specified. (**Note: This parameter is only supported when the end-device Atom firmware version ≥ 1.3**)
    - `0` : Non-force-controlled gripper
    - `1` : Force-controlled gripper
- **Return value：**
  - `1` : completed

`get_gripper_value(gripper_type=None)`

- **Function:** Get the current position data information of the gripper
- **Parameter Description:**
  - `gripper_type` : Gripper type, the default is adaptive gripper
    - `1` : Adaptive gripper
    - `3` : Parallel jaws
    - `4` : Flexible gripper
- **Return value:** Gripper data information

`set_eletric_gripper(status)`

- **Function:** Set gripper mode (only for 350)
- **Parameter description:** `status` : 1 means the clamping jaw is closed, 0 means the clamping jaw is open.
- **Return value:** 1

`set_gripper_mode(status)`

- **Function:** Set gripper mode
- **Parameter description:** `status` : 1 transparent transmission mode, 0 I/O mode
- **Return value:** 1

`get_gripper_mode()`

- **Function:** Get gripper status
- **Return value:** `status(int)` : 0 - Transparent transmission mode 1 - I/O mode

`set_HTS_gripper_torque(torque)`

- **Function:** Set adaptive gripper torque
- **Parameter Description:**
  - `torque` : 150 ~ 900
- **Return value:** 0 - Setting failed; 1 - Setting successful

`get_HTS_gripper_torque()`

- **Function:** Get adaptive gripper torque
- **Return value:** 150 ~ 900

`get_gripper_protect_current()`

- **Function:** Get gripper protection current
- **Return value:** 1 ~ 500

`init_gripper()`

- **Function:** Initialize gripper
- **Return value:** 0 - initialization failed; 1 - initialization successful

`set_gripper_protect_current(current)`

- **Function:** Set gripper protection current
- **Parameter Description:**
  - `current` : 1 ~ 500
- **Return value:** 0 - initialization failed; 1 - initialization successful

Simple Demo

```python
from pymycobot.mecharm270 import MechArm270
import time
#Enter the above code to import the packages required by the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)

    # Let the gripper reach the state of 100 at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the state of 0 at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to quickly open the gripper at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # Set the state of the gripper so that it quickly closes the gripper at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())


if __name__ == "__main__":
    # MechArm270 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:
#       MechArm270-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyACM0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#      MechArm270-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code here for Raspberry Pi version below
mc = MechArm270("/dev/ttyAMA0", 1000000)

# M5 version
# mc = MechArm270("COM3", 115200)

# make it move to zero position
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```

Controlling Gripper

`is_gripper_moving( )`

- **Function:** Determine whether the gripper is running
- **return value:**
  - `0` : Indicates that the gripper of the robot arm is not running
  - `1` : Indicates that the gripper of the robot arm is running
  - `-1` : indicates an error

`set_gripper_value(value, speed, gripper_type=None)`

- **Function:** Let the gripper rotate to the specified position at the specified speed
- **Parameter Description:**
  - `value` : Indicates the position that the clamping jaw wants to reach, the value range is 0~256
  - `speed` : indicates the speed at which to rotate, the value range is 0~100
  - `gripper_type` : Gripper type, the default is adaptive gripper
    - `1` : Adaptive gripper
    - `3` : Parallel jaws
    - `4` : Flexible gripper
- **Return value:** None

`get_gripper_value(gripper_type=None)`

- **Function:** Get the current position data information of the gripper
- **Parameter Description:**

    - `gripper_type` : Gripper type, the default is adaptive gripper
        - `1` : Adaptive gripper
        - `3` : Parallel jaws
        - `4` : Flexible gripper
- **Return value:** Gripper data information

`set_gripper_state(flag, speed, _type=None)`

- **Function:** Let the gripper enter the specified state at the specified speed
- **Parameter Description:**
    - `flag` : 1 means the clamping jaw is closed, 0 means the clamping jaw is open.
    - `speed` : Indicates how fast to reach the specified state, the value range is 0~100
    - `_type` : Gripper type, the default is adaptive gripper
        - `1` : Adaptive gripper
        - `2` : Five-fingered dexterity
        - `3` : Parallel jaws
        - `4` : Flexible gripper
- **Return value:** None

`set_eletric_gripper(status)`

- **Function:** Set gripper mode (only for 350)
- **Parameter description:** `status` : 1 means the clamping jaw is closed, 0 means the clamping jaw is open.
- **Return value:** None

`set_gripper_mode(status)`

- **Function:** Set gripper mode
- **Parameter description:** `status` : 1 transparent transmission mode, 0 I/O mode
- **Return value:** None

`get_gripper_mode()`

- **Function:** Get gripper status
- **Return value:** `status(int)` : 0 - Transparent transmission mode 1 - I/O mode

`set_HTS_gripper_torque(torque)`

- **Function:** Set adaptive gripper torque
- **Parameter Description:**
    - `torque` : 150 ~ 900
- **Return value:** 0 - Setting failed; 1 - Setting successful

`get_HTS_gripper_torque()`

- **Function:** Get adaptive gripper torque
- **Return value:** 150 ~ 900

`get_gripper_protect_current()`

- **Function:** Get gripper protection current
- **Return value:** 1 ~ 500

`init_gripper()`

- **Function:** Initialize gripper
- **Return value:** 0 - initialization failed; 1 - initialization successful

`set_gripper_protect_current(current)`

- **Function:** Set gripper protection current
- **Parameter Description:**
  - `current` : 1 ~ 500
- **Return value:** 0 - initialization failed; 1 - initialization successful

Simple Demo

```
from pymycobot.mycobot import MyCobot
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time
#Enter the above code to import the packages required by the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024], 20)
    time.sleep(3)

    # Let the gripper reach the state of 100 at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the state of 0 at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to quickly open the gripper at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # Set the state of the gripper so that it quickly closes the gripper at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())


if __name__ == "__main__":
    # MyCobot class initialization requires two parameters:
    #   The first is the serial port string, such as:
    #       linux:  "/dev/ttyAMA0"
    #          or "/dev/ttyAMA0"
```

```
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is:  115200
#
#    Example:
#       mycobot-M5:
#           linux:
#               mc = MyCobot("/dev/ttyAMA0", 1000000)
#          or mc = MyCobot("/dev/ttyAMA0", 115200)
#           windows:
#               mc = MyCobot("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot object
# Create object code here for Raspberry Pi version below
mc = MyCobot(PI_PORT, PI_BAUD)
# make it move to zero position
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```

myBuddy

Controlling Gripper

is_gripper_moving(id)

- **Function** Judge whether the gripper is moving or not
- **Parameters**

  id – 1/2 (L/R)
- **Returns**
  - 0 - not moving
  - 1 - is moving
  - -1 - error data

set_gripper_value(id, value, speed)

- **Function** Set gripper value
- **Parameters**
  - **id** – 1/2 (L/R)
  - **value** (*int*) – 0 ~ 100
  - **speed** (*int*) – 0 ~ 100

get_gripper_value(id)

- **Function** Get the value of gripper.

- **Parameters**

  id – 1/2 (L/R)

- **Returns**

  gripper value (int)

is_gripper_moving(id)

- **Function** Judge whether the gripper is moving or not
- **Parameters**

  id – 1/2 (L/R)

- **Returns**
  - 0 - not moving
  - 1 - is moving
  - -1 - error data

myArm

Simple Demo

```
from pymycobot.myarm import MyArm
import time
#Enter the above code to import the packages required by the project

def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to the position of 2048
    mc.set_encoder(1, 2048)
    time.sleep(2)
    # Set six joint positions and let the robotic arm rotate to this position at a speed of 20
    mc.set_encoders([1024, 1024, 1024, 1024, 1024, 1024,1024], 20)
    time.sleep(3)

    # Let the gripper reach the state of 100 at a speed of 70
    mc.set_gripper_value(100, 70)
    time.sleep(3)
    # Let the gripper reach the state of 0 at a speed of 70
    mc.set_gripper_value(0, 70)
    time.sleep(3)

    # Set the state of the gripper to quickly open the gripper at a speed of 70
    mc.set_gripper_state(0, 70)
    time.sleep(3)
    # Set the state of the gripper so that it quickly closes the gripper at a speed of 70
    mc.set_gripper_state(1, 70)
    time.sleep(3)

    # Get the value of the gripper
    print("")
    print(mc.get_gripper_value())


if __name__ == "__main__":

    # Initialize a MyArm object
```

```
mc = MyArm("/dev/ttyAMA0", 115200)
# make it move to zero position
mc.set_encoders([2048, 2048, 2048, 2048, 2048, 2048, 2048], 20)
time.sleep(3)
gripper_test(mc)
```

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.

MechArm

1.Connection

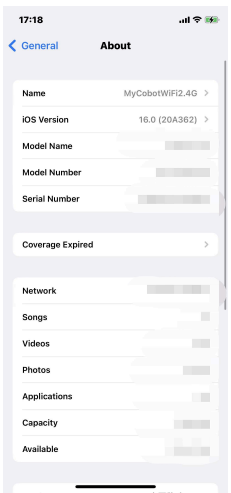
1.1 Create a default network

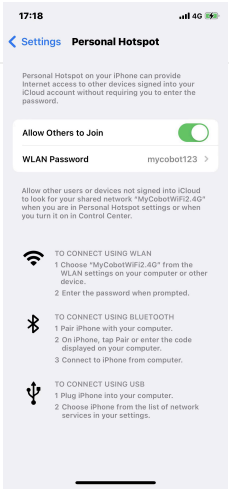
When using TCP/IP with the myCobot 280 m5 , it connects to a "MyCobotWiFi2.4G" network using the default password "mycobot123".

At this point, we can create a hotspot with a mobile phone, rename the hotspot network to "MyCobotWiFi2.4G", and set the password to "mycobot123". After enabling the hotspot, the robotic arm will automatically connect to the mobile phone hotspot using the TCP/IP function. Subsequently, as long as the devices are on the same local area network, they can communicate with each other over the network.

Similarly, you can set up a router by configuring the network name and password. Once the router is set up, and the robotic arm's TCP/IP function is enabled, it will connect to the router.

One important point to note is that the myCobot 280 m5 only supports the 2.4 GHz network band and does not support the 5 GHz network band. The following example uses a mobile phone hotspot.





1.2 Enabled the TCP/IP function

As shown in the figure, the robotic arm connects by pressing the button and selecting Transponder->WLAN Server. If the connection is successful, it will display the IP address and port number. If the connection fails, please check if the network name and password are set correctly.

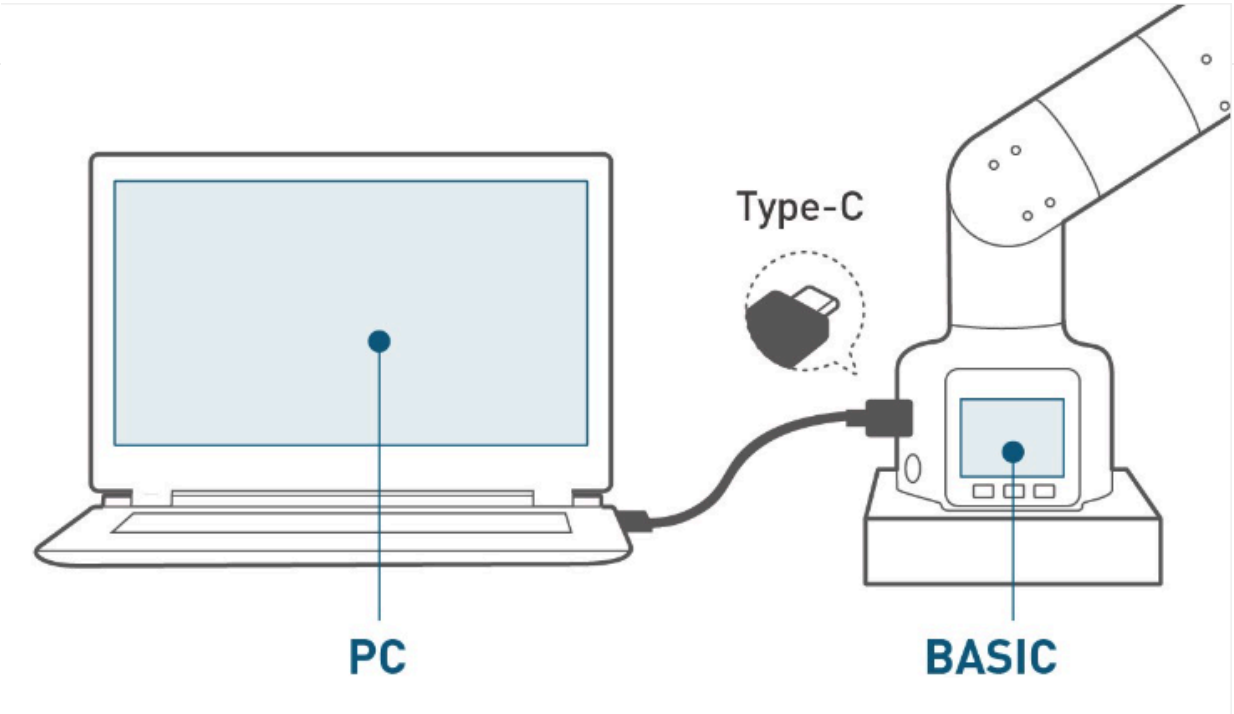
If the screen shows WIFI Connected, IP and Port, it means that robotic arm is successfully connected with WIFI.

1.3 Connect to another network

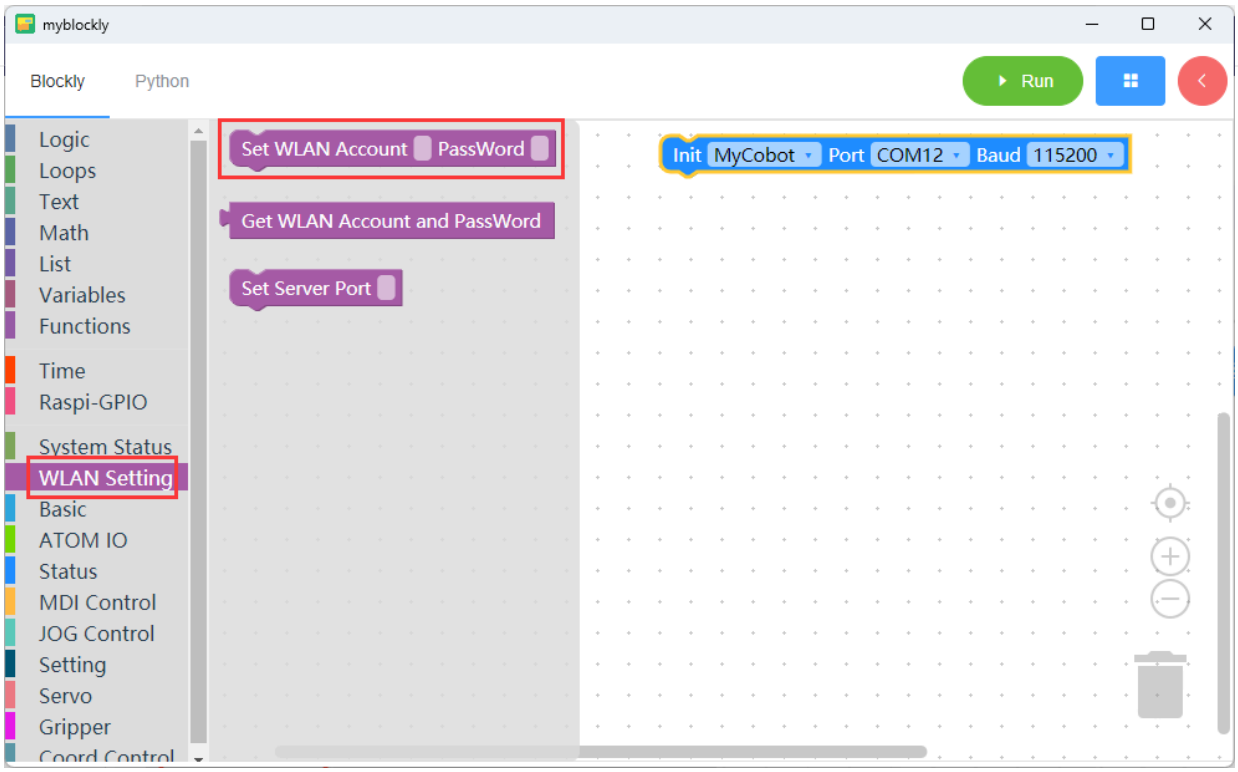
If you need to connect to another network, you can use the myBlockly software to configure the new network settings.

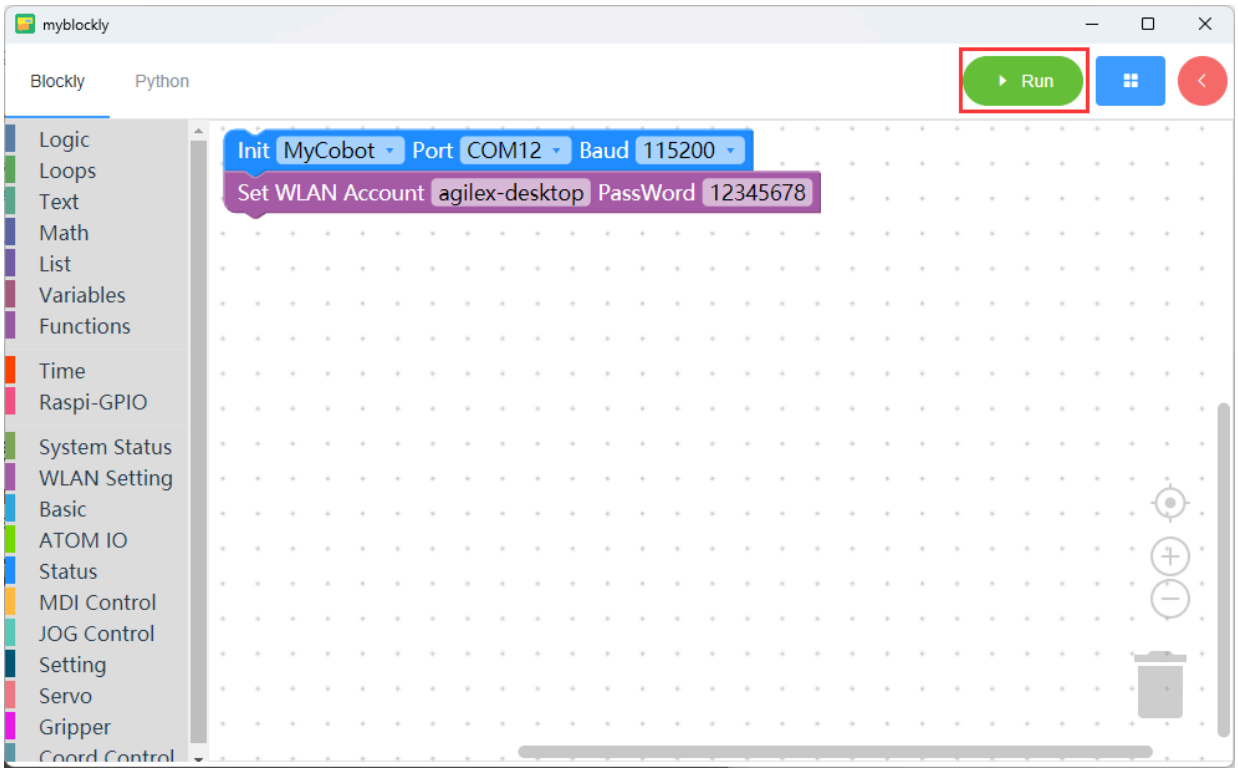
Note: The myCobot 280 m5 cannot save the connected Wi-Fi account and password when powered off. After a power cycle, it will reconnect to the default Wi-Fi account "MyCobotWiFi2.4G" with the password "mycobot123". To connect to a different network, you will need to reconfigure the Wi-Fi account and password.

Step 1: Connect the PC to the myCobot 280 m5



Step 2: Configure the myCobot 280 m5 Wi-Fi Settings Using myBlockly





Step 3: By following these steps, the myCobot 280 m5 will connect to the "agilex-desktop" network.

1.4 myCobot Raspberry Pi Connections

- When using Raspberry Pi for remote connection, you need to pay attention to the following points
  1. The Raspberry Pi and the control end need to be on the same network
  2. The server file needs to be executed first in the Raspberry Pi (For specific operations, see the gif operation diagram)
  3. After the server file is executed, the prompt "Binding succeeded and waiting connect" indicates that the opening is successful, and the control terminal can refer to **2 Simple Demo**

specific operation is:

clone our project library： `git clone https://github.com/elephantrobotics/pymycobot.git`

find the Server.py file in the demo folder and execute it with python

1.4 MechArm Raspberry Pi Connections

- When using Raspberry Pi for remote connection, you need to pay attention to the following points
  1. The Raspberry Pi and the control end need to be on the same network
  2. The server file (**change to Server_270.py**)needs to be executed first in the Raspberry Pi (For specific operations, see the gif operation diagram)
  3. After the server file is executed, the prompt "Binding succeeded and waiting connect" indicates that the opening is successful, and the control terminal can refer to **Simple Demo**

specific operation is:

clone our project library： `git clone https://github.com/elephantrobotics/pymycobot.git`

find the Server_270.py file in the demo folder and execute it with python

Instructions for use:

Please update pymycobot to the latest version before use.

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

- myPalletizer260

Please change the parameters passed in the last line of the Server_260.py file, MycobotServer, based on your model.

The default model is the 260PI.

```
The default parameters are:

    serial_num: /dev/ttyAMA0

    baud: 1000000
```

- MechArm 270

Please change the parameters passed in the last line of the Server_270.py file, MechArmSocket, based on your model.

The default model is the 270PI.

```
The default parameters are:

    serial_num: /dev/ttyAMA0

    baud: 1000000
```

Simple Demo

When the robotic arm successfully activates the TCP/IP function under the mobile hotspot, it will display the IP address and port. Be sure to remember this IP and port.

Connect your PC to the same mobile hotspot as the robotic arm. By using the Python driver library, you can connect to the robotic arm via its IP address, allowing for remote operation without needing to connect through a USB port.

- myPalletizer260

```python
from pymycobot import MyPalletizerSocket
# Use port 9000 by default
# Where "192.168.43.46" is the IP of the robot arm
mc = MyPalletizerSocket("192.168.43.46",9000)

#After the connections is normal, the robot arm can be controlled.
res = mc.get_angles()
print(res)
mc.send_angles([0,0,0,0],20)
...
```

- MechArm 270

```
from pymycobot import MechArmSocket
# Use port 9000 by default
# Where "192.168.43.46" is the IP of the robot arm
mc = MechArmSocket("192.168.43.46",9000)

#After the connections is normal, the robot arm can be controlled.
res = mc.get_angles()
print(res)
mc.send_angles([0,0,0,0,0,0],20)
...
```

myArm

Simple Demo

```
from pymycobot import MyArmSocket
# Use port 9000 by default
# Where "192.168.10.22" is the IP of the robot arm
mc = MyArmSocket("192.168.11.15",9000)

#After the connections is normal, the robot arm can be controlled.
mc.send_angles([0,0,0,0,0,0,0],20)
res = mc.get_angles()
print(res)

...
```

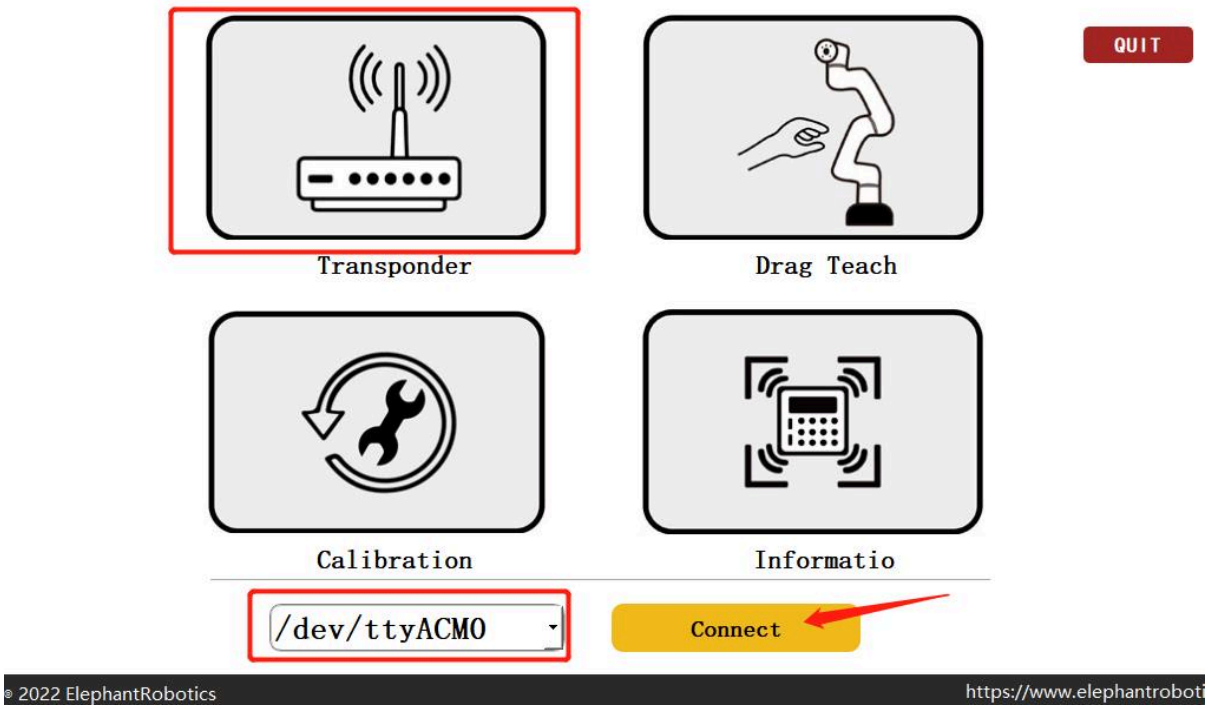
mybuddy

> Note: This function must be used under the same network

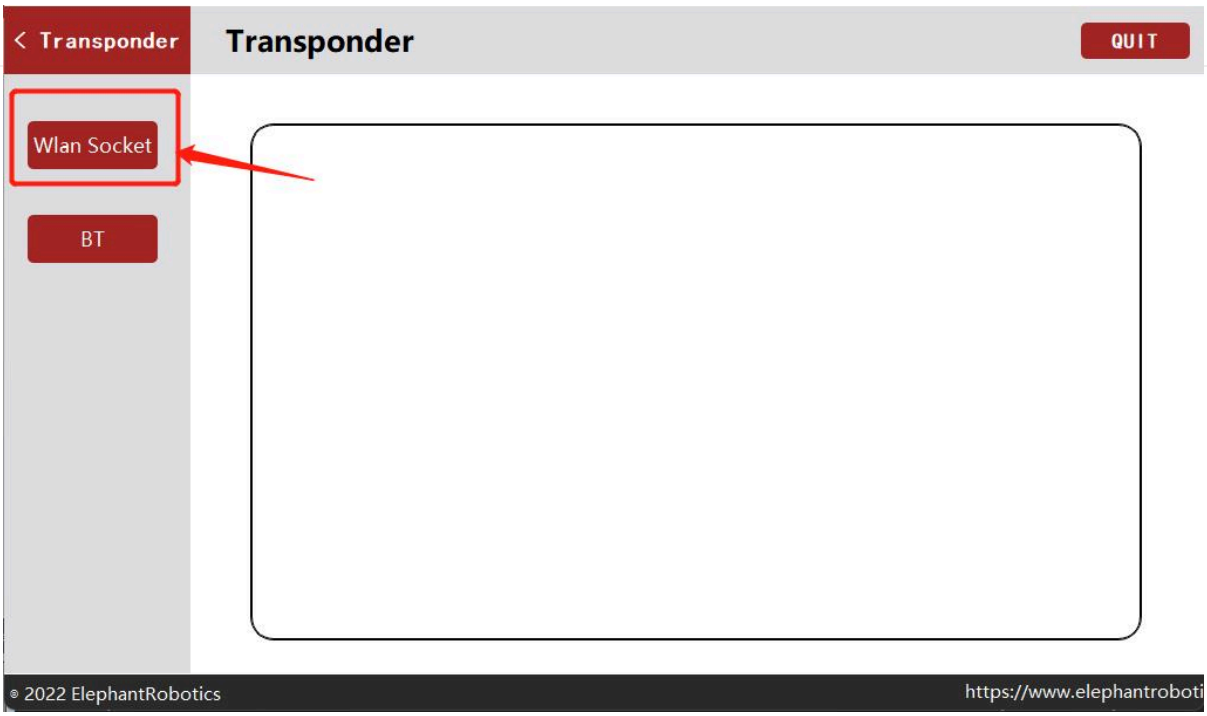
start the server

1.Double click to open this software on the desktop

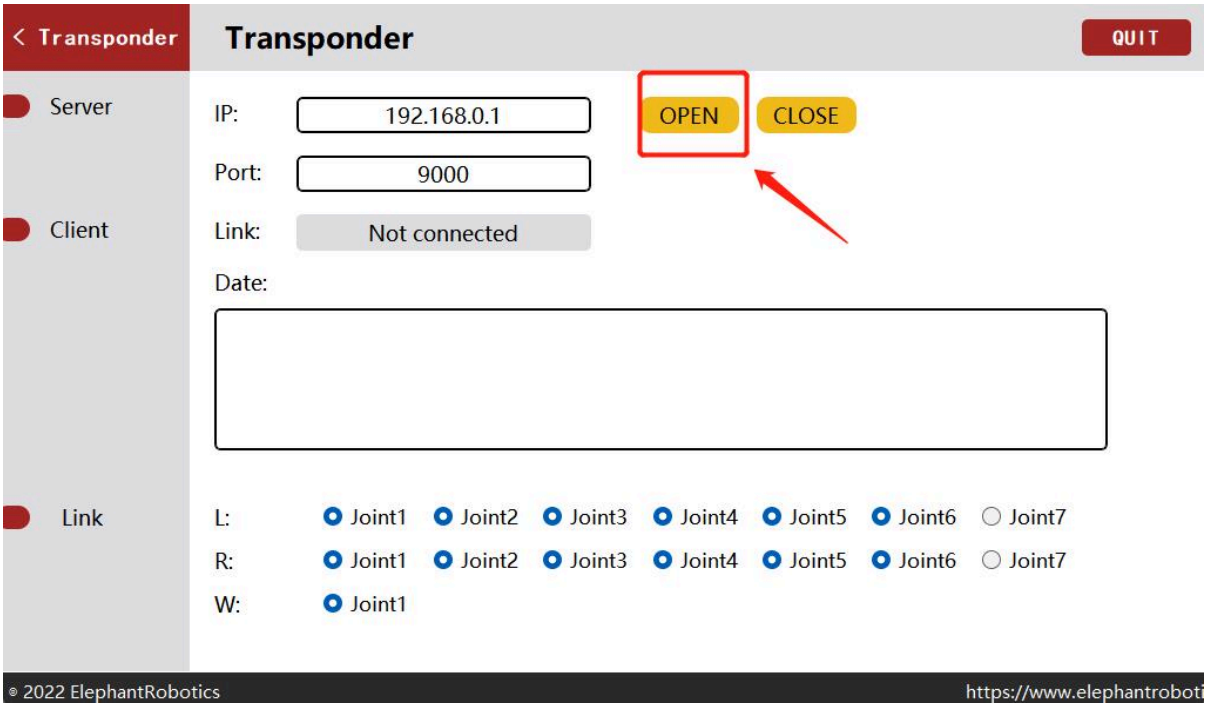
2.Select Transponder, the port is /dev/ttyACM0, click Connect



3.Select "Wlan Socket"



4.Click "OPEN" to open the server



Client connection

```
from pymycobot import MyBuddySocket


mst = MyBuddySocket("192.168.0.1", 9000)
mst.connect("/dev/ttyACM0", "115200")


print(mst.get_angles(1))
```

Bluetooth control

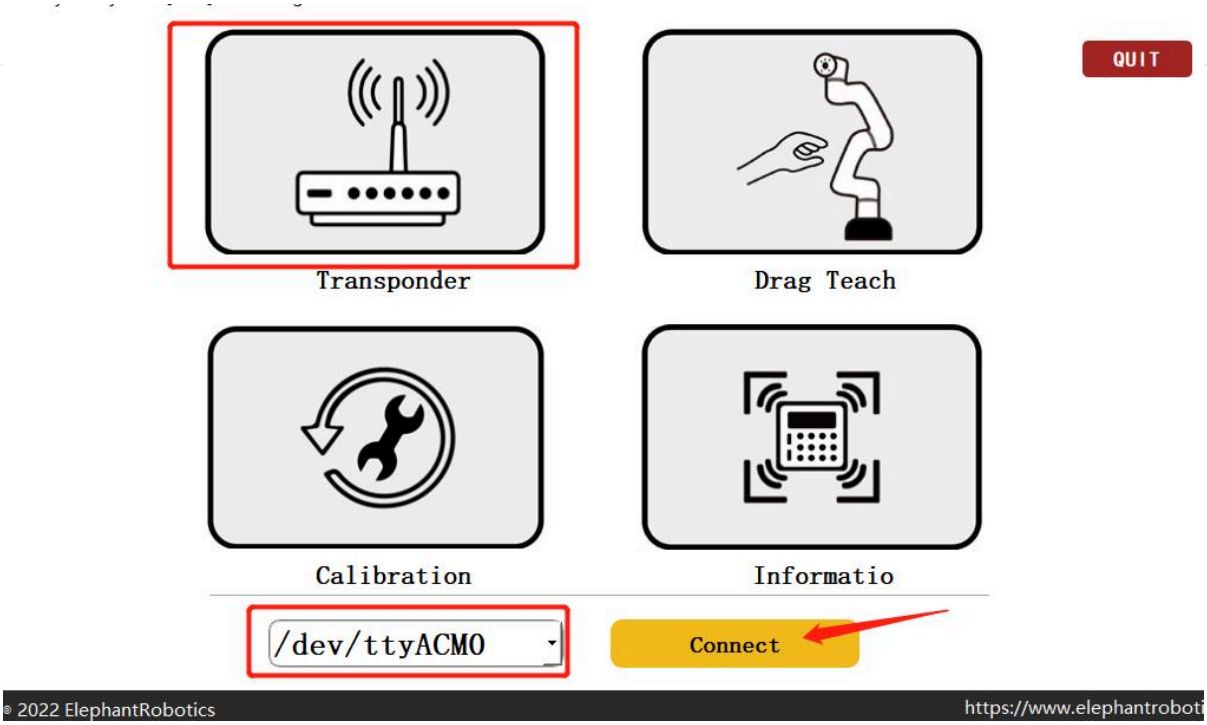
> Note: The Bluetooth server will be automatically closed after the client connection ends, and it needs to be reopened when it is used again (it is best not to actively close the server, which may cause it to fail to open next time)

myBuddy

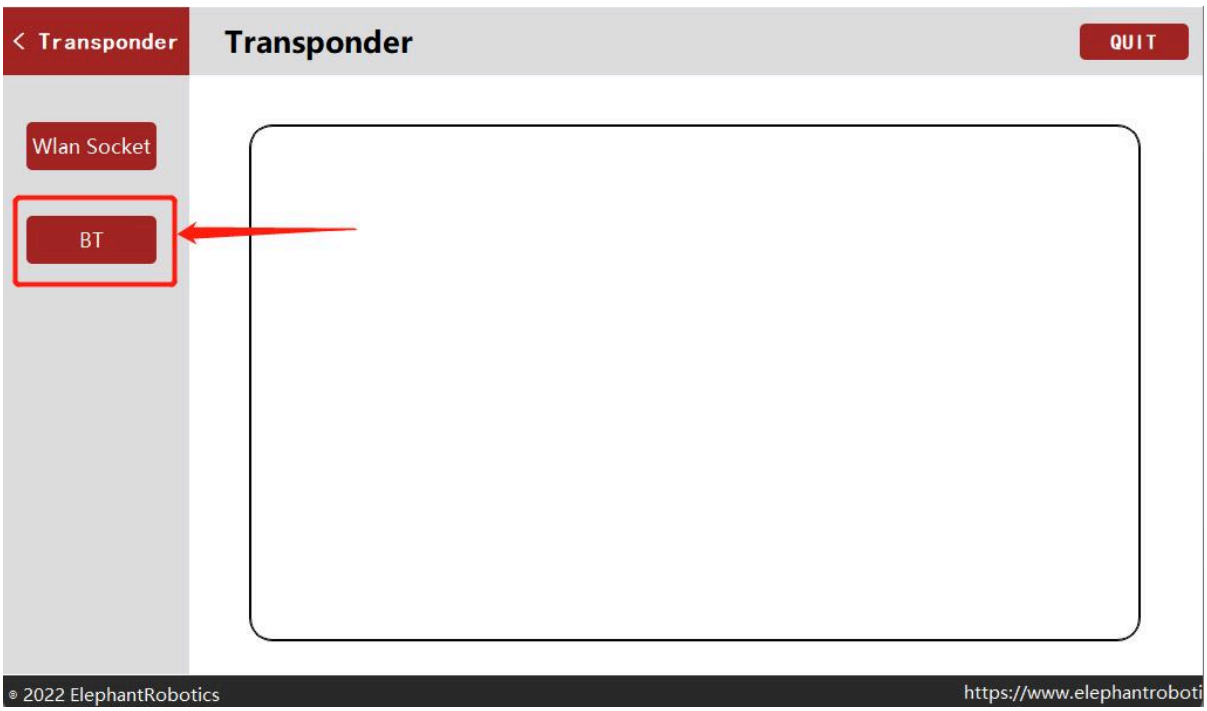
start the server

1.Double click to open this software on the desktop

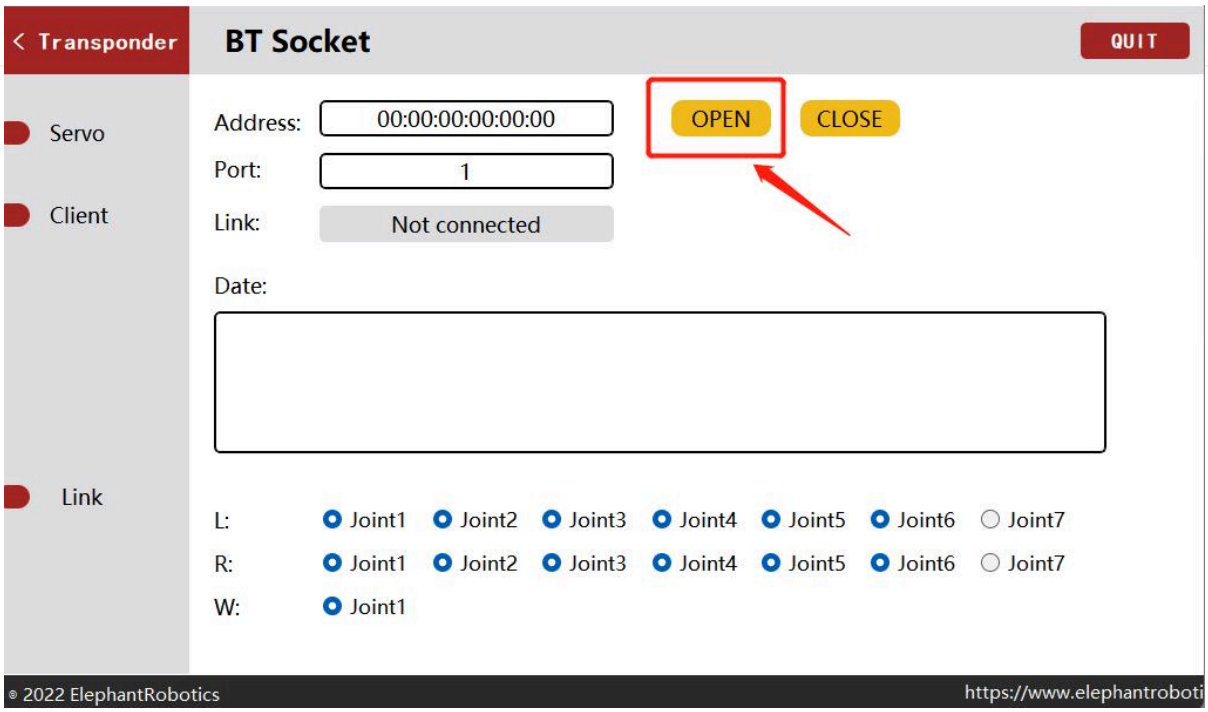
2.Select Transponder, the port is /dev/ttyACM0, click Connect



3.Select "Wlan Socket"



4.Click "OPEN" to open the server



If the server starts successfully, you can start using it

Client

> Note: Please install **pybluez2** before using it, you can install it by command: pip install pybluez2==0.40

```
from pymycobot import MyBuddyBlueTooth

mst = MyBuddyBlueTooth("00:00:00:00:00:00", 1)
mst.connect("/dev/ttyACM0", "115200")

print(mst.get_angles(1))
```

Emotion Using

myBuddy

The screen that comes with myBuddy can display a variety of expressions through the python interface

Instructions：

1. Download and unzip video file and case code into the mybuddy system
2. Change the case file,Replace `/home/er/pymycobot/emo/face_video_3_2.mp4` with the decompressed video file path

```
from pymycobot import MyBuddyEmoticon
import time


# [video_path, Playback_duration(s)]
video1 = ["/home/er/pymycobot/emo/face_video_3_2.mp4", 10]


datas = [video1]


me = MyBuddyEmoticon(datas)
...
```

3. Execute mybuddy_emoticon_demo.py

API description

MyBuddyEmoticon(file_path: list = [], window_size: tuple = (), **Fuction* loop=False) API for playing emoticons

- **Parameters**

  file_path (list): `[[path, time]]` The absolute path of facial expression video and the length of time to play.Time in seconds. **window_size (tuple)**: `(Length, width)` Size of the playback window (default is full screen). **loop (bool)**: Loop playback or not (default False. only once by default).

file_path()

- **Fuction** Get Playfile List
- **Return**
  - video path list

add_file_path(path_time)

- **Fuction** Add Playback File
- **Parameters path_time(list)**: `[path, time]` The video address to be added and the running time

del_file_path(index)

- **Fuction** Delete the element with the specified subscript in the playlist list
- **Parameters**

  index(int):: The subscript of the element in the playlist to be deleted

pause()

- **Fuction** Pause playback

run()

- **Fuction** Continue playing

start()

- **Fuction** start playing video

join()

- **Fuction** Wait for the thread playing the video to finish.

Introduction to API

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

```
# for myBuddy
from pymycobot.mybuddy import MyBuddy
```

> **Notice:** Functions with return value are required to use `print()` to print value. For example, if you want to get the speed value, type `print(get_speed())`, instead of `get_speed()`.

myBuddy

1 Overall Status

1.1 `power_off(id=0)`

- **Fuction:** Close communication with Atom.
- **Parameters:**

  id – 0/1/2/3 (ALL/L/R/W)

1.2 `power_on(id=0)`

- **Function：** Open comminication with Atom.
- **Parameter**

  id – 0/1/2/3 (ALL/L/R/W)

1.3 `read_next_error(id=0)`

- **Function：** Robot Error Detection
- **Parameter**

  id – 0/1/2/3 (ALL/L/R/W)

1.4 `release_all_servos(id=0)`

- **Function：** Robot turns off torque output
- **Parameter**

  id – 0/1/2/3 (ALL/L/R/W)

1.5 `is_power_on(id=0)`

- **Function：** Adjust robot arm status
- **Parameter**

  id – 0/1/2/3 (ALL/L/R/W)

- **Return Value：**

  1 - power on 0 - power off -1 - error data

1.6 `is_controller_connected(id=0)`

- **Function：** Wether connected with Atom.

- **Parameter**

  id – 0/1/2/3 (ALL/L/R/W)

- **Return Value：**

  0 - Not connected 1 - Connected

1.7 `set_free_mode(id, value)`

- **Function：** set free mode
- **Parameter**
  - **id** – 0/1/2/3 (ALL/L/R/W)
  - **value** - 0 - close 1 - open

1.8 `set_fresh_mode(id, mode)`

- **Function：** set command refresh mode
- **Parameter**
  - **id** – 1/2（L/R）.
  - **mode** - int 0 - Always execute the latest command first. 1 - Execute instructions sequentially in the form of a queue.

1.9 `release_servo（id, servo_id）`

- **Function：** Power off designated servo
- **Parameter**
  - **id** – 1/2/3 (L/R/W)
  - **servo_id** - 1 - 6.

1.10 `is_free_mode(id)`

- **Function：** check if it is free mode
- **Parameter**

  id – 0/1/2/3 (ALL/L/R/W)

- **Return Value：**

  0 - No 1 - Yes

2 Operating Mode

2.1 `stop（id）`

- **Function：** Stop moving
- **Parameters：**

  id – 0/1/2/3 (ALL/L/R/W).

2.2 `resume（id）`

- **Function：** Recovery movement
- **Parameters：**

  id – 0/1/2/3 (ALL/L/R/W).

2.3 `is_paused(id)`

- **Function：** Judge whether the manipulator pauses or not.
- **Parameters：**

  id – 0/1/2/3 (ALL/L/R/W).
- **Return Value：**

  1 - paused 0 - not paused -1 - error

2.4 `get_speed(id)`

- **Function：** get speed
- **Parameters：**

  id – 1/2/3（L/R/W）.
- **Return Value：**

  speed
- **Return Value：Type**

  int

2.5 `set_speed（id, speed)`

- **Function：** set speed value
- **Parameters：**
  - **id** – 1/2/3 (L/R/W)
  - **speed** (*int*) - 0 - 100

2.6 `get_joint_min_angle（id, joint_id）`

- **Function：** Gets the minimum movement angle of the specified joint
- **Parameters：**
  - **id** – 1/2/3 (L/R/W)
  - **joint_id** - (int) 1 - 6
- **Return Value：**

  angle value(float)

2.7 `is_servo_enable（id, servo_id）`

- **Function：** Determine whether all steering gears are connected
- **Parameters：**
  - **id** – 1/2/3 (L/R/W)
  - **servo_id** - (int) 1 ~ 6

- **Return Value：**

  0 - disable 1 - enable -1 - error

2.8 `is_all_servo_enable(id)`

- **Function：** Determine whether the specified steering gear is connected
- **Parameters：**

  id – 1/2/3 (L/R/W)

- **Return Value：**

  0 - disable 1 - enable -1 - error

2.9 `set_joint_min（id，joint_id，angle)`

- **Function：** Set the joint minimum angle
- **Parameters：**
  - **id** – 1/2/3 (L/R/W)
  - **joint_id** - int 1-6.
  - **angle** - 0 ~ 180

2.10 `get_system_version(id)`

- **Function：** get system version
- **Parameters：**

  id – 0/1/2/3 (ALL/L/R/W)

2.11 `get_joint_max_angle（id，joint_id`

- **Function:** Gets the maximum movement angle of the specified joint
- **Parameters**
  - **id** – 1/2/3 (L/R/W)
  - **joint_id** - (int) 1 - 6
- **Return Value：**

  angle(float)

2.12 `joint_brake(id, joint_id)`

- **Function：** Make it stop when the joint is in motion, and the buffer distance is positively related to the existing speed
- **Parameters**
  - **id** – 1/2/3 (L/R/W)
  - **joint_id** - 1 - 6

3 MDI Mode

3.1 `get_angles(id)`

- **Function：** Get the degree of all joints.

- **Parameters**

  id – 1/2（L/R）

- **Return Value：**

  A float list of all degree.

- **Return Value：**

  list

3.2 `send_angle(id, joint, angle, speed)`

- **Function：** Send one degree of joint to robot arm.
- **Parameters**
  - **id** – 1/2/3 (L/R/W)
  - **joint** – 1 ~ 6
  - **angle** - int
  - **speed** – 1 ~ 100
- **Return Value：**
  - None

3.3 `send_angles（id，degrees，speed)`

- **Function：** Send all angles to the robotic arm
- **Parameters**
  - **id** – 1/2（L/R）.
  - **degrees** - [angle_list] len 6
  - **speed** - 1 - 100

3.4 `set_joint_max（id，joint_id，angle)`

- **Function：** set the joint maximum angle
- **Parameters**
  - **id** – 1/2/3 (L/R/W)
  - **joint_id** - int 1-6.
  - **角度** - 0 ~ 180

3.5 `send_coord(id, coord, data, speed)`

- **Function：** Send a single coordinate to the robotic arm
- **Parameters**
  - **id** – 1/2/3 (L/R/W).
  - **coord** – 1 ~ 6 (x/y/z/rx/ry/rz)
  - **data** - int

- **speed** - 0 ~ 100

3.6 `send_coords(id, coords, speed, mode)`

- **Function：** Send all coordinates to robotic arm
- **Parameters**
  - **id** – 1/2（L/R）.
  - **coords** – a list of coords（List[float]），length 6，[x(mm), y, z, rx(angle), ry, rz]
  - **speed** - (int) 1 ~ 100
  - **mode** - (int) 0 - moveJ, 1 - moveL, 2 - moveC

3.7 `get_coord（id, joint_id）`

- **Function：** Get the coordinates of the robotic arm
- **Parameters**
  - **id** (*int*) – 1/2/3 (L/R/W).
  - **joint_id** (*int*) – 1 - 7（7 is gripper）

3.8 `get_encoder(id,joint_id)`

- **Function：** Obtain the specified joint potential value.
- **Parameters**
  - **id** - 1/2/3（L/R/W）.
  - **joint_id** - (int) 1 ~ 6
- **Return Value：**

  0 ~ 4096

3.9 `get_encoders(id)`

- **Function：** Get the six joints of the manipulator
- **Parameters**

  id – 1/2（L/R）.
- **Return Value：**

  list

3.10 `get_radians(id)`

- **Function：** Get the radians of all joints
- **Parameters**

  id – 1/2（L/R）
- **Return Value：**

  A list of float radians [radian1, ...]
- **Return Value：** list

3.11 `send_radians(id, radians, speed)`

- **Function：** Send the radians of all joints to robot arm
- **Parameters**
  - **id** – 1/2（L/R）.
  - **radians** – a list of radian values（List[float]），length 6
  - **speed** - (int)0 ~ 100

3.12 `is_in_position(id, data, mode)`

- **Function：** Detect whether in the position.
- **Parameters**
  - **id** – 0/1/2/3 (ALL/L/R/W).
  - **data** – A data list, angles or coords. If id is 1/2. data length is 6. If id is 0. data len 13. if id is 3. data len 1
  - **mode** - 1 - coords, 0 - angles
- **Return Value：**

  1 - True 0 - False -1 - error

3.13 `is_moving(id)`

- **Function：** Detect if the robot is moving
- **Parameters**

  id – 0/1/2/3 (ALL/L/R/W).
- **Return Value：**

  0 - not moving 1 - is moving -1 - error data

3.14 `set_color(id, r=0, g=0, b=0)`

- **Function：** Set the light color on the top of the robot arm.
- **Parameters**
  - **id** - 1/2（L/R）
  - **r** (*int*) – 0 ~ 255
  - **g** (*int*) – 0 ~ 255
  - **b** (*int*) – 0 ~ 255

3.15 `set_encoder（id, joint_id, encoder, speed)`

- **Function：** Set a single joint rotation to the specified potential value.
- **Parameters**
  - **id** – 1/2/3 (L/R/W).
  - **joint_id** - 1 - 6.
  - **encoder** – The value of the set encoder.

3.16 `set_encoders（id, encoder, speed)`

- **Function：** Set the six joints of the manipulator to execute synchronously to the specified position.

- **Parameters**
  - **id** – 1/2（L/R）.
  - **encoders** - A encoder list, length 6.
  - **speed** – speed 1 ~ 100

3.17 `get_angle（id, joint_id）`

- **Function：** Get the angle of a single joint
- **Parameters**
  - **id** (*int*) – 1/2/3 (L/R/W).
  - **joint_id** (*int*) – 1 - 7（7 is gripper）

3.18 `set_servo_calibration（id, servo_no`

- **Function：** The current position of the calibration joint actuator is the angle zero point,

  and the corresponding potential value is 2048.
- **Parameters：**
  - **id** – 1/2/3 (L/R/W)
  - **servo_no** – Serial number of articulated steering gear, 1 - 6.

3.19 `set_joint_current（id, joint_id, current）`

- **Function：** Set Collision Current
- **Parameters:**
  - **id** - 0/1/2 (ALL/L/R)
  - **joint_id** - 1 - 6
  - **current** – current value

3.19 `get_coords(id)`

- **Function：** Read a single coordinate parameter
- **Parameters**

  id – 1/2（L/R）

4 JOG Mode

4.1 `jog_absolute（id, joint_id, angle, speed）`

- **Function：** Absolute joint control
- **Parameters:**
  - **id** – 1/2/3 (L/R/W).
  - **joint_id** - int 1-6.
  - **angle** - int
  - **speed** - int (0 - 100)

4.2 `jog_angle（id，joint_id，direction，speed）`

- **Function：** Jog control joint
- **Parameters:**
  - **id** – 1/2/3 (L/R/W).
  - **joint_id** - int 1-6.
  - **direction** - 0 - decrease，1 - increase
  - **speed** - int (0 - 100)

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

- **Function：** Jog control coordinate
- **Parameters:**
  - **id** – 1/2/3 (L/R/W).
  - **coord_id** – int 1-6 (x/y/z/rx/ry/rz).
  - **direction** - 0 - decrease，1 - increase
  - **speed** - int (0 - 100)

4.4 `jog_inc_coord（axis，increment，speed）`

- **Function：** Double-arm coordinated coordinate stepping
- **Parameters:**
  - **axis** – 1 - 6 (x/y/z/rx/ry/rz)
  - **increment** -
  - **speed** - 1 - 100

4.5 `jog_increment（id，joint_id，increment，speed）`

- **Function：** step mode
- **Parameters:**
  - **id** – 1/2/3 (L/R/W).
  - **joint_id** - int 1-6.
  - **increment** -
  - **speed** - int (1 - 100)

4.6 `jog_stop(id)`

- **Function：** JOG stop
- **Parameters:**

  id – 1/2/3（L/R/W）

5 Servo COntrol

5.1 `focus_servo（id，servo_id）`

- **Function：** Power on designated servo
- **Parameters：**
  - **id** – 1/2/3 (L/R/W)
  - **servo_id** - 1 - 6

5.2 `get_servo_currents(id)`

- **Function：** Get joint current
- **Parameters：**

  id – 1/2/3 (L/R/W)
- **Return Value：**

  value mA

5.3 `get_servo_status(id)`

- **Function：** Get joint status
- **Parameters：**

  id – 1/2/3 (L/R/W)
- **Return Value：**

  [voltage, sensor, temperature, current, angle, overload], a value of 0 means no error

5.4 `get_servo_temps(id)`

- **Function：** Get joint temperature
- **Parameters：**

  id – 1/2/3 (L/R/W)

5.5 `get_servo_voltages(id)`

- **Function：** Get joint voltages
- **Parameters：**

  id – 1/2/3 (L/R/W)
- **Return Value：**

  volts < 24 V

6 Atom IO Control

6.1 `set_pin_mode(id, pin_no, pin_mode)`

- **Function：** Set the state mode of the specified pin in atom.
- **Parameters：**
  - **id** - 1/2（L/R）
  - **pin_no** (*int*) – pin number (1 - 5).
  - **pin_mode** (*int*) – 0 - input, 1 - output

6.2 `set_digital_output(id, pin_no, pin_signal)`

- **Function：** Set atom IO output level
- **Parameters：**
  - **id** - 1/2（L/R）
  - **pin_no** (*int*) - 1 - 5
  - **pin_signal** (*int*) – 0 / 1

6.3 `get_digital_input(id, pin_no)`

- **Function：** signal
- **Parameters：**
  - **id** - 1/2（L/R）
  - **pin_no** (*int*) - 1 - 5

6.4 `set_pwm_output(id, channel, frequency, **6.1pin_val)`

- **Function：** PWM control
- **Parameters：**
  - **id** - 1/2（L/R）
  - **channel** (*int*) – IO number (1 - 5).
  - **frequency** (*int*) – clock frequency (0/1: 0 - 1Mhz 1 - 10Mhz)
  - **pin_val** (*int*) – Duty cycle 0 ~ 100: 0 ~ 100%

7 Gripper Control

7.1 `get_gripper_value(id)`

- **Function：** Get the value of gripper.
- **Parameters**

  id – 1/2（L/R）
- **Return Value：**

  gripper value (int)

7.2 `is_gripper_moving(id)`

- **Function：** Judge whether the gripper is moving or not
- **Parameters**

  id – 1/2（L/R）
- **Return Value：**

  0 - not moving 1 - is moving -1 - error data

7.4 `set_gripper_state(id, flag)`

- **Function：** Set gripper switch state

- **Parameters**
  - **id** - 1/2（L/R）
  - **flag** (*int*) - 0 - close，1 - open

7.5 `set_gripper_value(id, value, speed)`

- **Function：** Set gripper value
- **Parameters**
  - **id** – 1/2 (L/R)
  - **value** (*int*) – 0 ~ 100
  - **speed** (*int*) – 0 ~ 100

8 Socket Control

```
from pymycobot import MyBuddySocket


mst = MyBuddySocket("192.168.0.1", 9000)
mst.connect("/dev/ttyACM0", "115200")


print(mst.get_angles(1))
```

9 Raspberry PI-GPIO

9.1 `set_gpio_input(pin)`

- **Function：** Set GPIO input value.
- **Parameters**

  pin - (int)pin number.

9.2 `set_gpio_mode(pin_no, mode)`

- **Function：** Init GPIO module, and set BCM mode.
- **Parameters**
  - **pin_no** – (int)pin number.
  - **mode** – 0 - input 1 - output

9.3 `set_gpio_output(pin, v)`

- **Function：** Set GPIO output value.
- **Parameters**
  - **pin** - (int)pin number.
  - **v** - (int) 0 / 1

9.4 `set_gpio_pwm（pin, baud, dc）`

- **Function：** Set GPIO PWM value.

- **Parameters**
  - **pin** – (int)pin number.
  - **baud** – (int) 10 - 1000000
  - **dc** – (int) 0 - 100

10 Coordinate Transformation

10.1 `set_tool_reference (id, coords)`

- **Function：** Set tool coordinate system
- **Parameters**
  - **id** - 0/1/2 (ALL/L/R)
  - **coords** – a list of coords value(List[float]), length 6. [x(mm), y, z, rx(angle), ry, rz]

10.2 `set_world_reference (id, coords)`

- **Function：** Set the world coordinate system
- **Parameters**
  - **id** - 0/1/2 (ALL/L/R)
  - **coords** – a list of coords value(List[float]), length 6 [x(mm), y, z, rx(angle), ry, rz]

10.3 `get_reference_frame(id)`

- **Function：** Get the base coordinate system
- **Parameters**

  id – 0/1/2 (ALL/L/R)
- **Return Value：**

  0 - base 1 - tool

10.4 `get_tool_reference(id)`

- **Function：** Get tool coordinate system
- **Parameters**

  id – 0/1/2 (ALL/L/R)

10.5 `get_world_reference(id)`

- **Function：** Get the world coordinate system
- **Parameters**

  id – 0/1/2 (ALL/L/R)

10.6 `set_reference_frame(id, rftype)`

- **Function：** Set the base coordinate system
- **Parameters**
  - **id** - 0/1/2 (ALL/L/R)

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

10.7 `set_movement_type(id, move_type)`

- **Function：** Set movement type
- **Parameters**
  - **id** - 0/1/2 (ALL/L/R)
  - **move_type** - 1 - movel，0 - moveJ

10.8 `get_movement_type(id)`

- **Function：** Get movement type
- **Parameters**

  id – 0/1/2 (ALL/L/R)
- **Return Value：**
  - 1 - moveL: Linear space motion command. Controls the robot's motion by specifying the position or posture of the robot's end effector in space.
  - 0 - moveJ: Joint space motion command. The robot's motion trajectory is not necessarily a straight line, and motion is achieved by changing the angles of each joint.

10.9 `set_end_type(id, end)`

- **Function：** Set end coordinate system
- **Parameters**
  - **id** - 0/1/2 (ALL/L/R)
  - **end** – 0 - flange, 1 - tool

10.10 `get_end_type(id)`

- **Function：** Get end coordinate system
- **Parameters**

  id – 0/1/2 (ALL/L/R)
- **Return Value：**

  0 - flange 1 - tool

10.11 `write_base_coords(id, coords, speed)`

- **Function：** Base coordinate move
- **Parameters**
  - **id** - 1/2（L/R）
  - **coords** – coords: a list of coords value(List[float]), length 6, [x(mm), y, z, rx(angle), ry, rz]
  - **speed** - 1 - 100

10.12 `write_base_coord（id, axis, coord, speed)`

- **Function：** Base single coordinate movement
- **Parameters**

- **id** - 1/2（L/R）
- **axis** – 1 - 6 (x/y/z/rx/ry/rz)
- **coord** - Coordinate value
- **speed** - 1 - 100

10.13 `base_to_single_coords(base_coords, arm)`

- **Function：** Convert base coordinates to coordinates
- **Parameters：**
  - **coords** – a list of base coords value len 6
  - **arm** – 0 - left. 1 - right
- **Return Value：：**

  coords

10.14 `get_base_coord(id)`

- **Function：** Get the base coordinates of the single arm
- **Parameters：**

  id – 1/2（L/R）

10.15 `get_base_coords(\*args: int)`

- **Function：** Convert coordinates to base coordinates. Pass in parameters or no parameters
- **Parameters：**
  - **coords** – a list of coords value(List[float]), length 6 [x(mm), y, z, rx(angle), ry, rz]
  - **arm** - 0 - L. 1 -
- **Return Value：：**

  Base coords

11 Speed Planning

11.1 `get_plan_acceleration(id=0)`

- **Function：** Get planning acceleration
- **Parameters：**

  id – 0/1/2/3 (ALL/L/R/W)
- **Return Value：**

  [movel planning acceleration, movej planning acceleration].

11.2 `get_plan_speed(id=0)`

- **Function：** Get planning speed
- **Parameters：**

  id – 0/1/2/3 (ALL/L/R/W)

- **Return Value：**

  [movel planning speed, movej planning speed].

11.3 `set_acceleration(id, acc)`

- **Function：** Set acceleration during all moves
- **Parameters:**
  - **id** – 1/2/3 (L/R/W)
  - **acc** - 1 - 100

11.4 `get_acceleration(id)`

- **Function：** Read acceleration during all moves
- **Parameters：**

  id – 1/2/3 (L/R/W)

12 Collision Detection

12.1 `get_joint_current（id, joint_id）`

- **Function:** Get Collision Current
- **Parameters:**
  - **id** - 0/1/2 (ALL/L/R)
  - **joint_id** - 1 - 6

12.2 `collision_switch（state）`

- **Function:** Collision Detection Switch
- **Parameters:**

  state (*int*) – 0 - close 1 - open (Off by default)

12.3 `collision（left_angles, right_angles）`

- **Function:** Collision detection main program
- **Parameters:**
  - **left_angles** – left arm angle len 6.
  - **right_angles** – right arm angle len 6.
- **Returns**

  int

12.4 `is_collision_on()`

- **Parameters：** Get collision detection status
- **Return Value：:**

  0 - disbale 1 - enable

ROS Introduction

ROS is the abbreviation of robot operating system. **ROS** is a highly flexible software architecture for writing robot software programs.

ROS is of open source and is a post-operating system or secondary operating system for controlling robots. It provides the functions similar to those provided by an operating system, including hardware abstraction description, low-level driver management, execution of common functions, messages transferred between programs, and program release package management. It also provide some tool programs and libraries used to get, create, write and run multi-machine integration programs.

The primary design goal of **ROS** is to improve the code reuse rate in the R&D field of robots. **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**.

Notice:

- Burn the corresponding firmware by using **mystudio**. Among them, burn minirobot in M5Stack-basic, select the transponder function, and burn the latest version of atommain in atom.
- With Ubuntu 20.04 system, **myBuddy** has built-in development environment, which can be used directly.
- If you use ROS to connect to **myBuddy** through a remote **Python Socket**, you need to build a ROS-related environment **on the remote PC**. For details, please refer to 12.1.2 Environment Building
- ROS1 version support: Ubuntu 20.04 / ROS1 Noetic
- ROS2 version support: Ubuntu 20.04 / ROS2 Foxy / ROS2 Galactic (**myBuddy** already has a ROS2 Galactic version built in)

Application of ROS in myBuddy

In order to drive **myBuddy** through ROS, two tools, rviz and moveit, will be introduced next.

- Rviz is a three-dimensional visualization platform in ROS, which can realize the graphical display of external information. In addition, it can release control information to the manipulator through rviz, so as to realize the monitoring and control of the robot. The **Rviz** section will introduce two control methods of **myBuddy**: Slider Control and Mobile Follow.
- **Moveit** is an integrated development platform in ROS, which is composed of a variety of function packages for manipulating manipulator, including motion planning, operation, control, inverse kinematics, 3D perception and collision detection. The **Moveit** section will introduce four kinds of control groups of **myBuddy**: the control group is used to realize the partial and overall control of the left arm, right arm and waist.

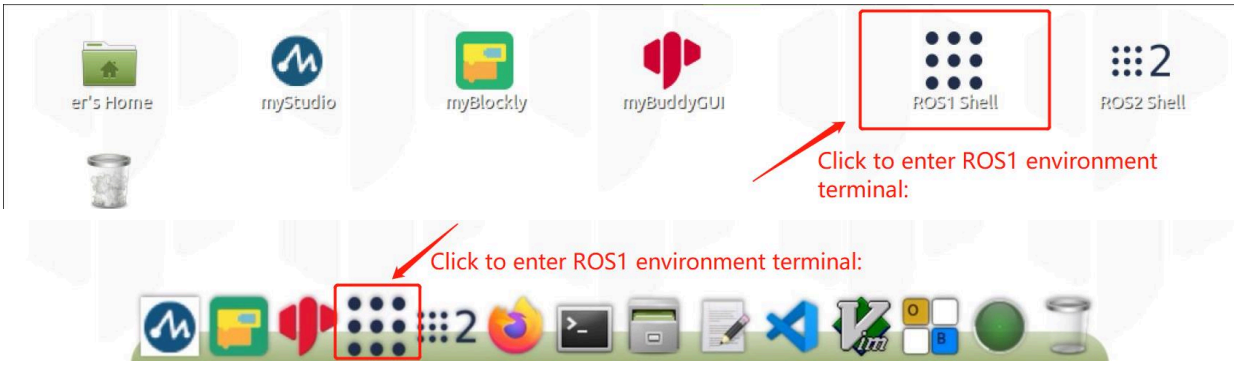
mycobot_ros installation and update

`mycobot_ros` is a ROS package from ElephantRobotics that works with all types of desktop robots.

The address of the project: https://github.com/elephantrobotics/mycobot_ros

`<ros-workspace>` is used to stand for the workspace path of ROS in the computer in the following part.The official default is `catkin_ws` . Be sure to replace `<ros-workspace>` with the real path of your local machine when executing the following command.

Click the `ROS1 Shell` icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS1 environment terminal:



Then enter the following command:

```
cd ~/<ros-workspace>/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
```

Note: If the `/home/ubuntu/catkin_ws/src(equivalent to ~/catkin_ws/src)` already exists in the `mycobot_ros` folder, you need to delete the `mycobot_ros` folder before running the above command. In the directory path, `ubuntu` is the user name of the VM. If they are different, change them.

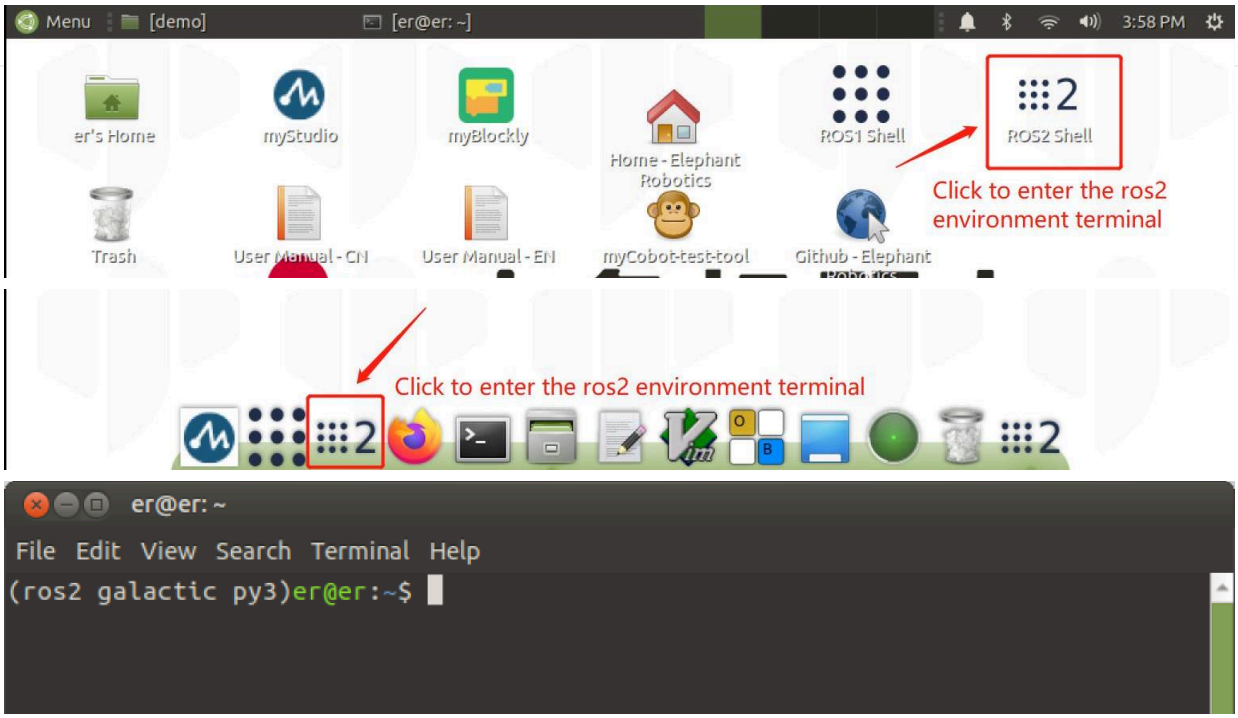
mycobot_ros2 installation and update

`mycobot_ros2` is a ROS package from ElephantRobotics that works with all types of desktop robots.

The address of the project: https://github.com/elephantrobotics/mycobot_ros2

`<ros2-workspace>` is used to stand for the workspace path of ROS in the computer in the following part.The official default is `colcon_ws` . Be sure to replace `<ros2-workspace>` with the real path of your local machine when executing the following command.

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS2 environment terminal:



Then enter the following command:

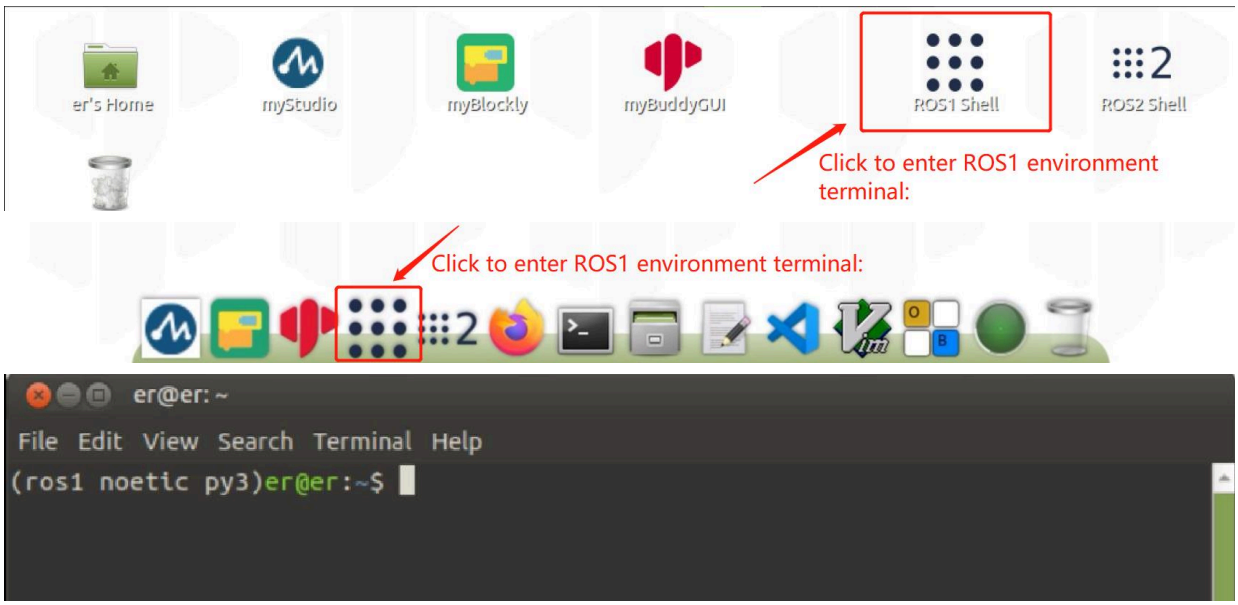
```
cd ~/<ros2-workspace>/src  # Enter the src folder of the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros2.git
cd ..       # return to the workspace
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile python scri
source install/setup.bash # add environment variables
```

Note: If the `/home/er/colcon/src(equivalent to ~/colcon_ws/src)` already exists in the `mycobot_ros2` folder, you need to delete the `mycobot_ros2` folder before running the above command.

myBuddy Moveit

`mycobot_ros` has integrated the MoveIt section.

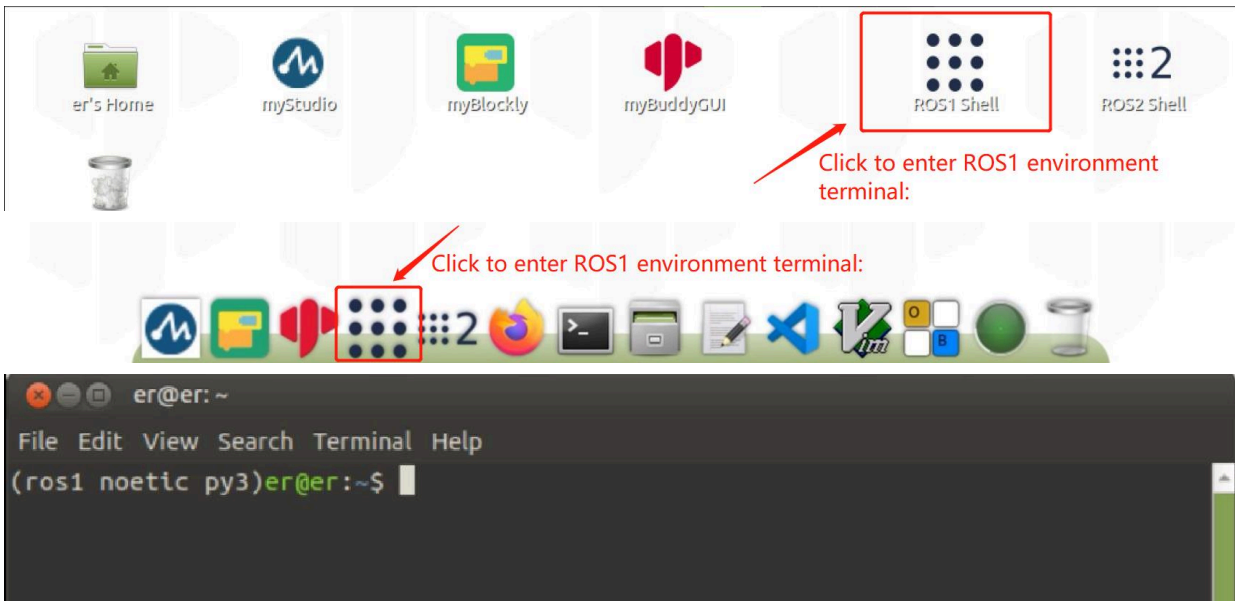
Click the ROS1 Shell icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS1 environment terminal:



Then run the command:

```
roslaunch mybuddy_moveit demo.launch
```

If you want a real robot arm to execute a plan synchronously, you need to open another ROS1 environment terminal:

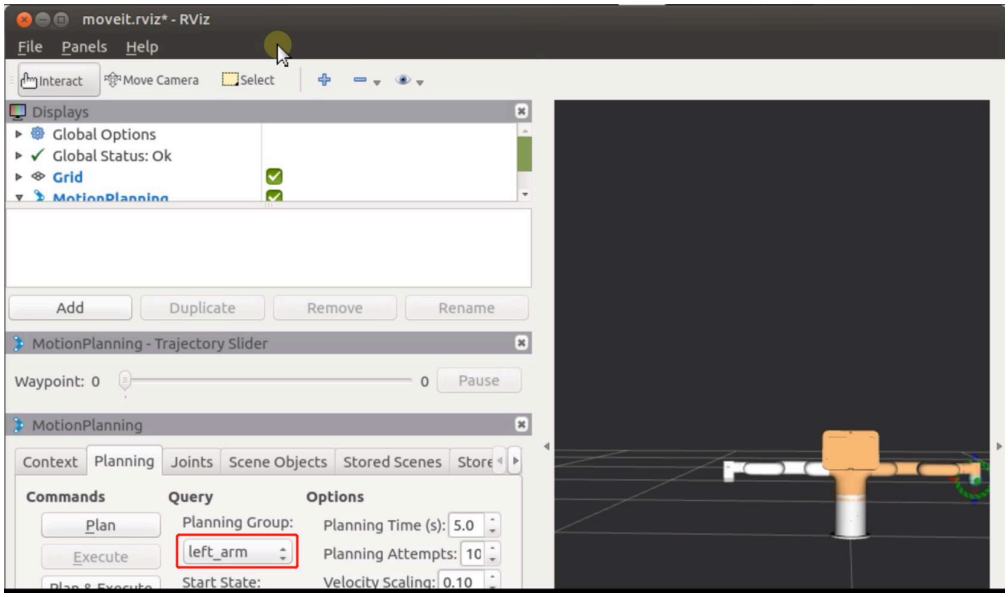


Then run the command:

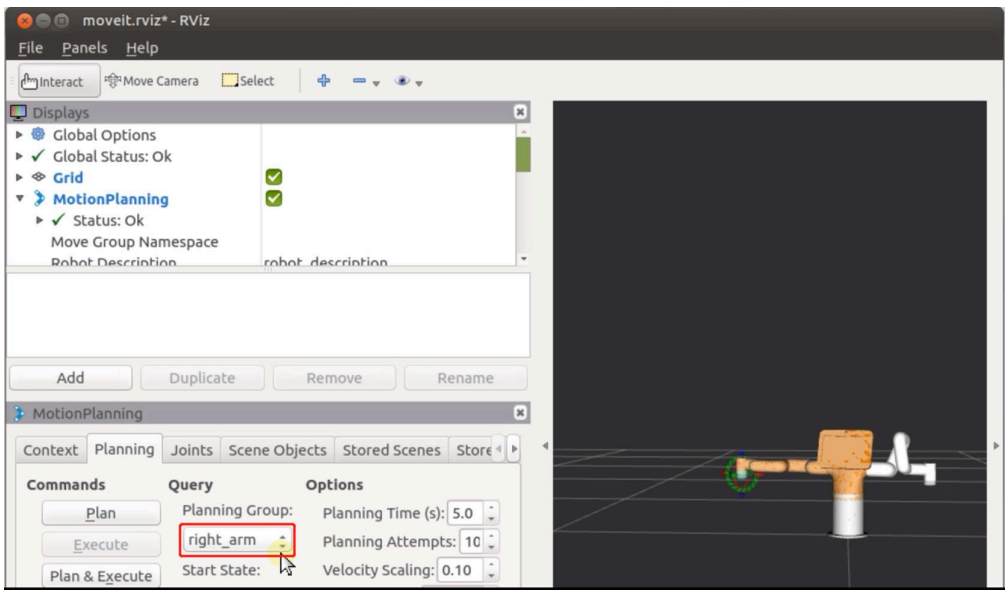
```
# The default serial port name of the mybuddy is "/dev/ttyACM0", and the baud rate is 115200".
rosrun mybuddy_moveit sync_plan.py _port:=/dev/ttyACM0 _baud:=115200
```

Moveit has four control groups, the operation effect is as follows:

1.**Left Arm Control Group**: plan the movement direction of **left arm** by dragging the trackball.



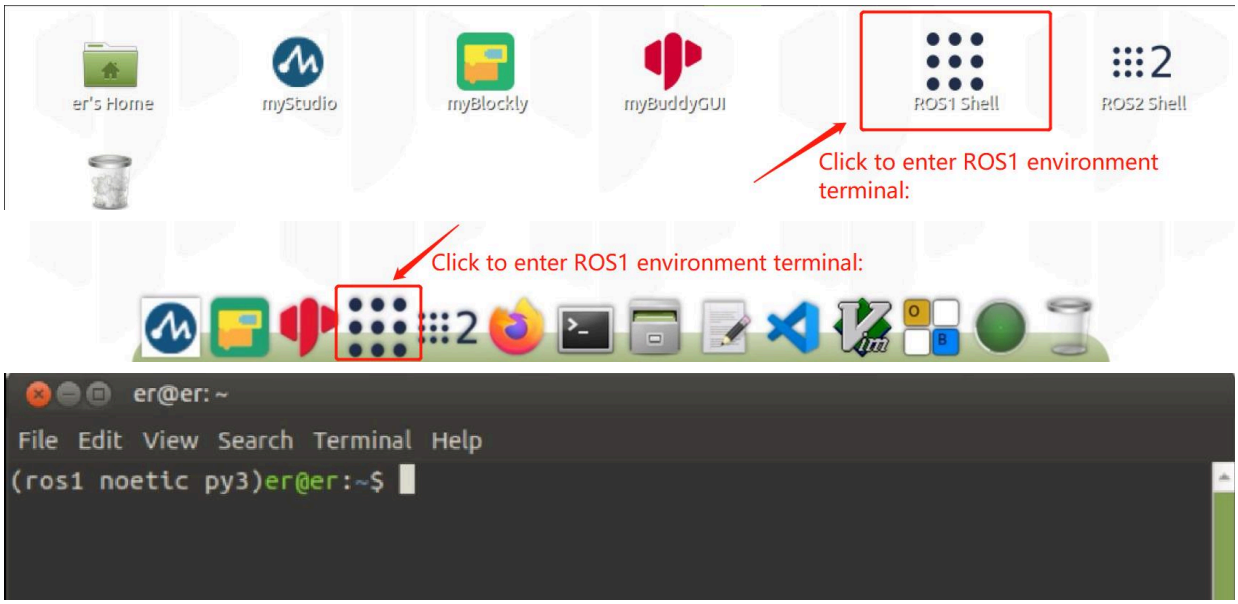
2.**Right Arm Control Group**: plan the movement direction of **right arm** by dragging the trackball.



Control and following of the robot arm

1 Slider Control

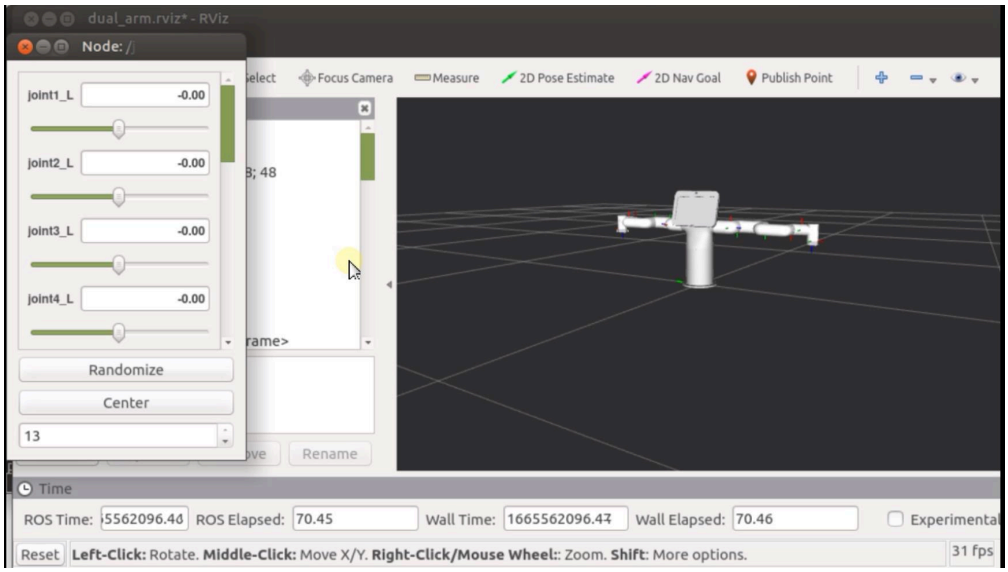
Click the ROS1 Shell icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS1 environment terminal:



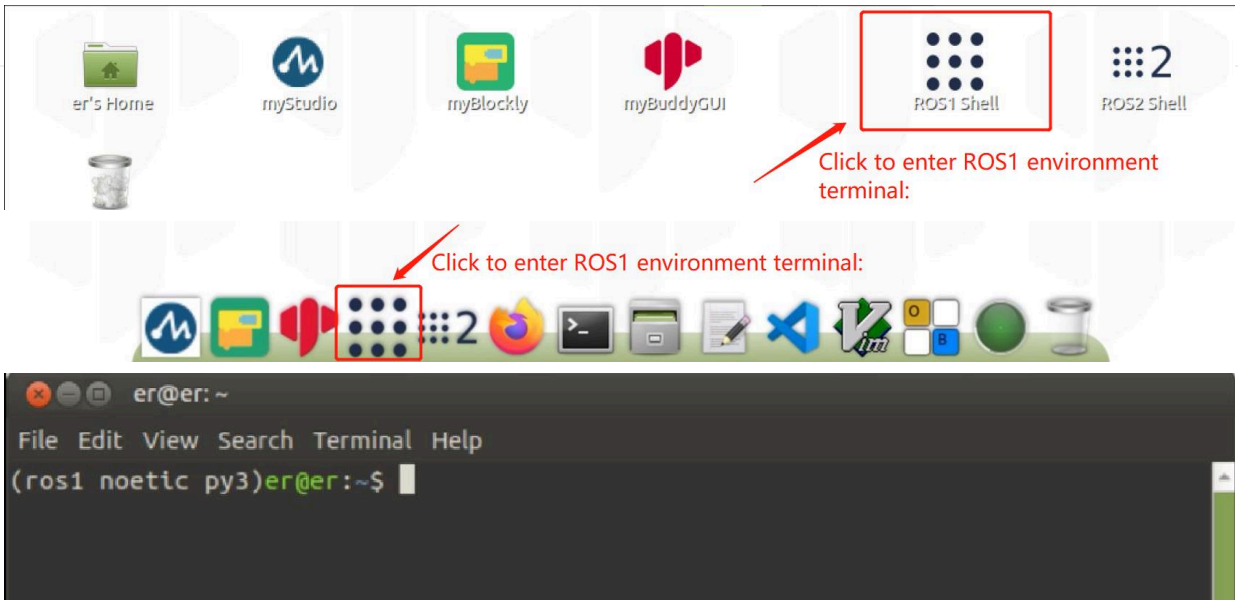
Then run the command:

```
# The default serial port name of the mybuddy is "/dev/ttyACM0", and the baud rate is 115200".
roslaunch mybuddy slider_control.launch _port:=/dev/ttyACM0 _baud:=115200
```

rviz and a slider component will be opened, and 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 mybuddy to move with the model, you need to open another ROS1 environment terminal:



Then run the command:

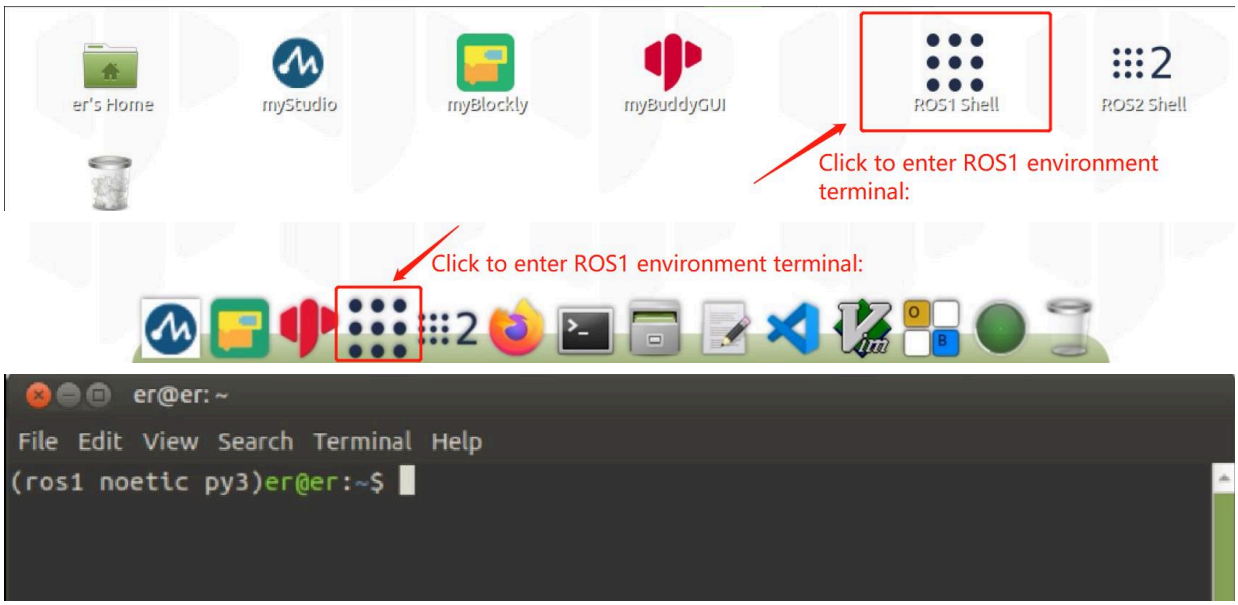
```
# The default serial port name of the mybuddy is "/dev/ttyACM0", and the baud rate is 115200".
rosrun mybuddy slider_control.py _port:=/dev/ttyACM0 _baud:=115200
```

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

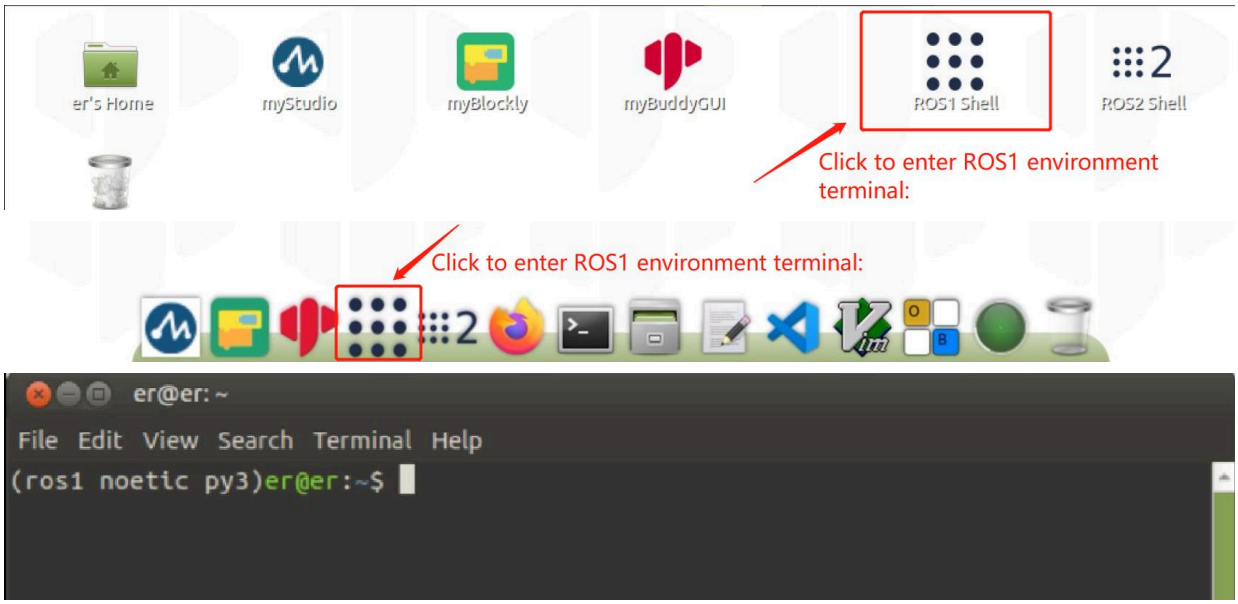
In addition to the above controls, we can also let the model move by following the real robot arm. Open a ROS1 environment terminal:



Then run the command:

```
# The default serial port name of the mybuddy is "/dev/ttyACM0", and the baud rate is 115200".
rosrun mybuddy follow_display.py _port:=/dev/ttyACM0 _baud:=115200
```

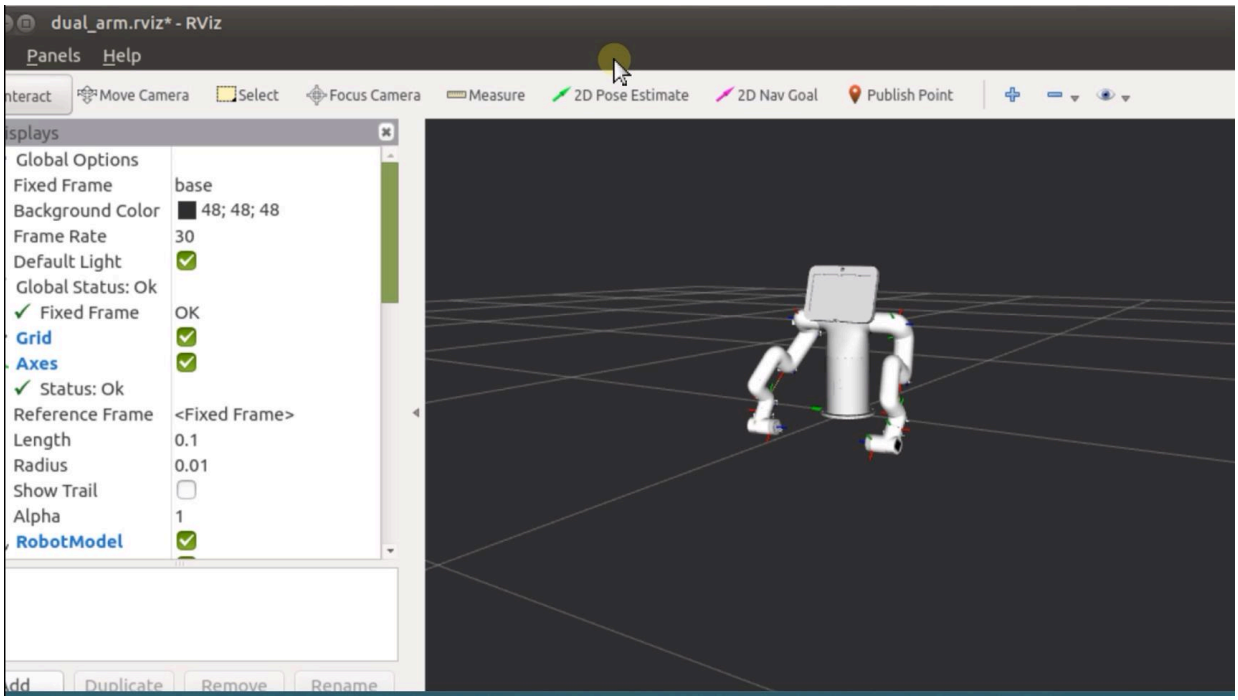
Then open another ROS1 environment terminal:



Then run the command:

```
roslaunch mybuddy mybuddy_follow.launch
```

It will open rviz to show the following effect of the model and the real robot arm, as shown below:

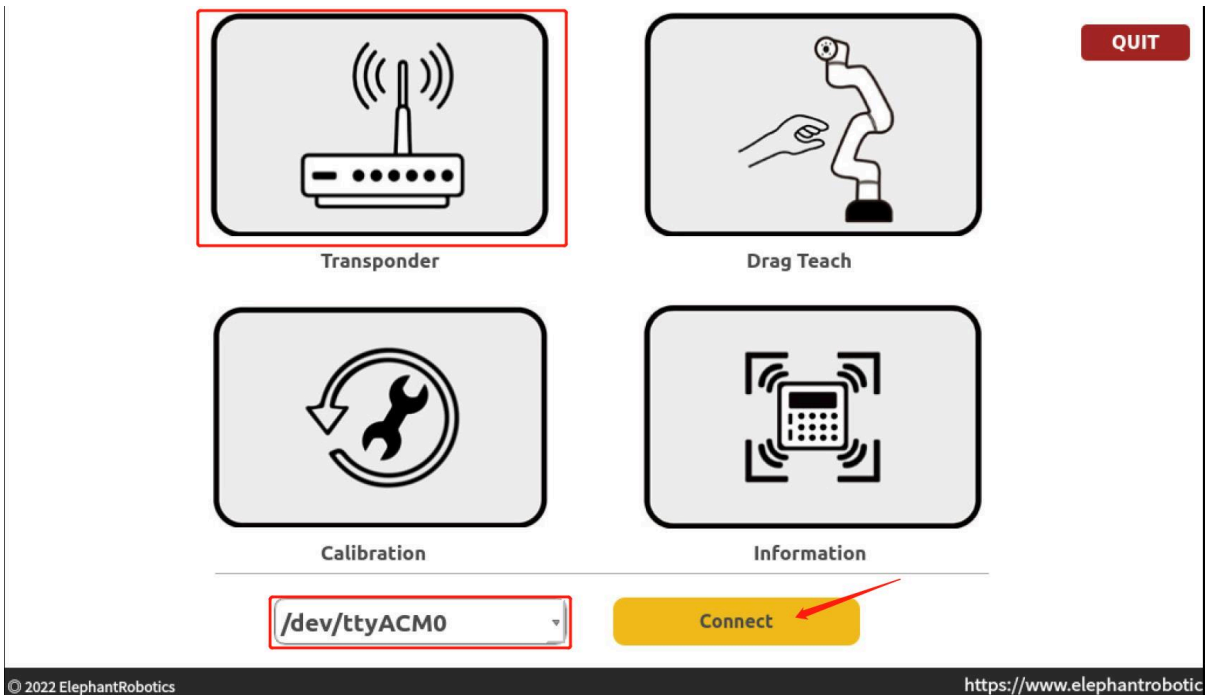


> Note: This function must be used on the same network. For example, both the server and the client use the same mobile hotspot or the same LAN WiFi

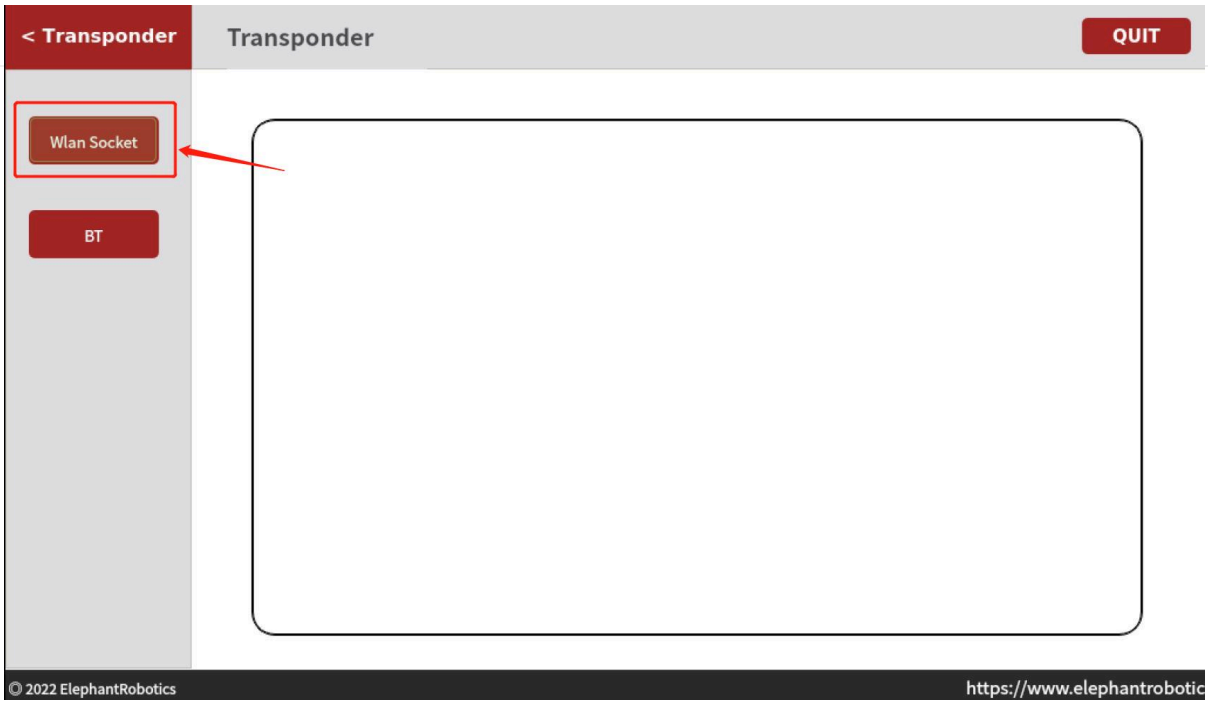
Start the server

1.Double click to open this software on the desktop

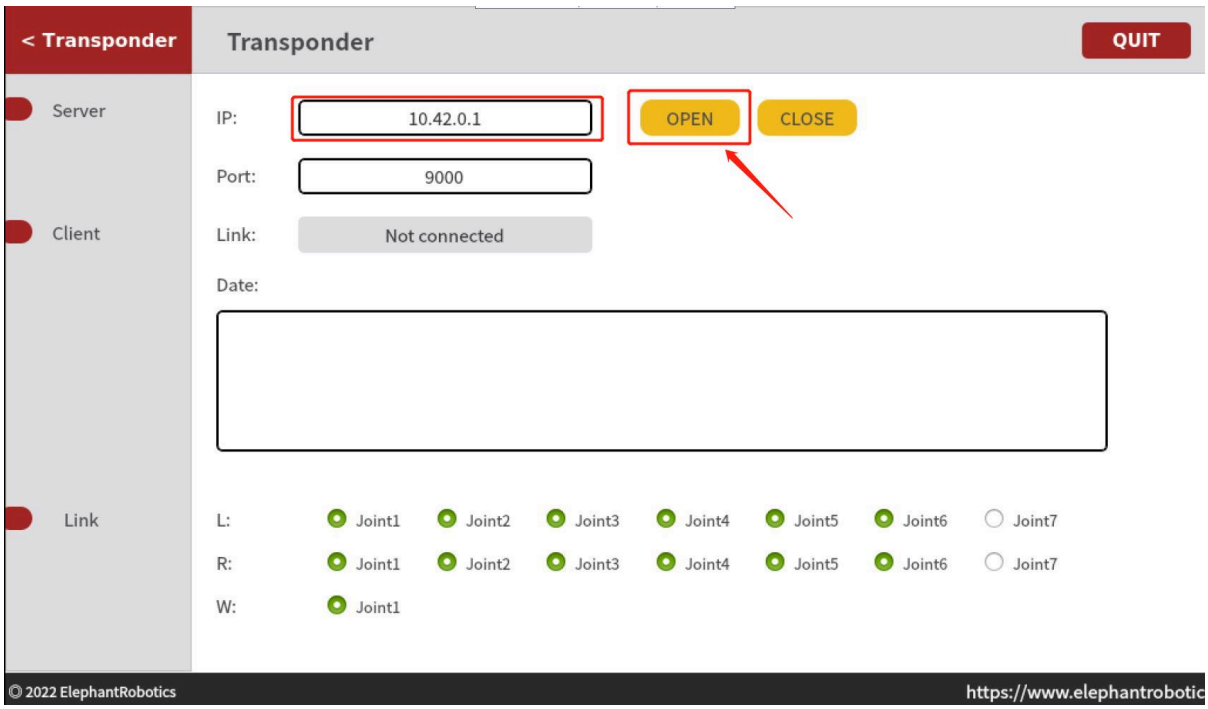
2.Select Transponder, the port is /dev/ttyACM0, click Connect



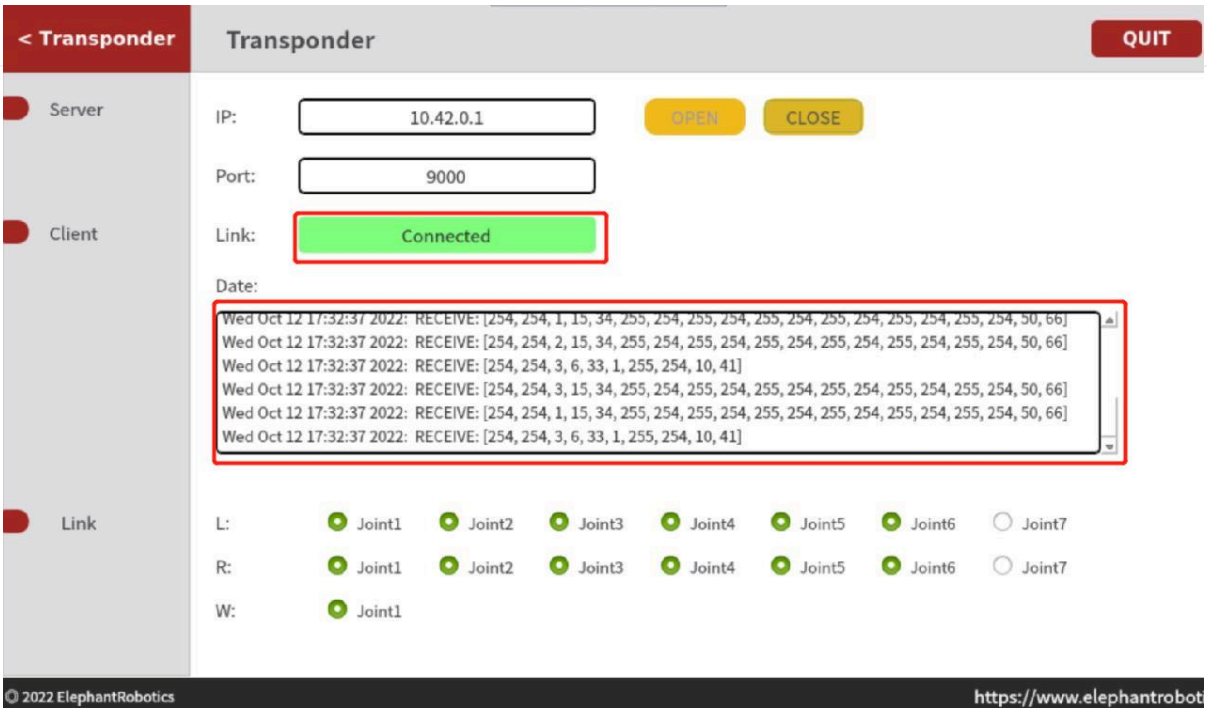
3.Select "Wlan Socket"



4.Click "OPEN" to open the server



If the client connects to the server successfully, "Connected" will be displayed on the page and data will be received:



Note: The IP in the server is not unique and can be determined based on the actual network connected by the robot arm.

Client connection

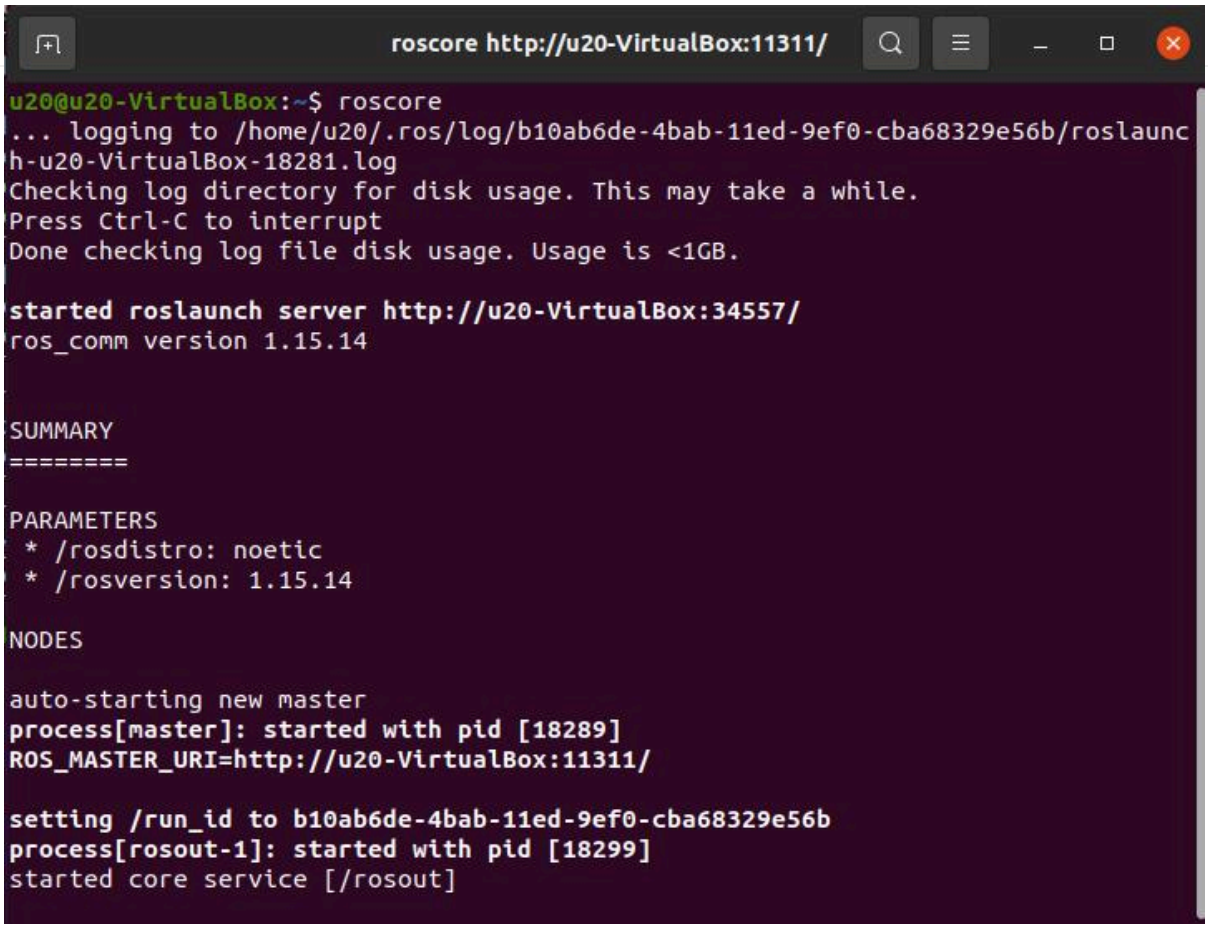
> Note: Ensure that the remote PC has completed the ROS environment building. For details, please refer to 12.1.2 Environment Building

Start ROS node

To start a ROS system, you need a ROS Master, or node manager. You can start the ROS Master by entering the roscore command at the terminal. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and run the following command on the terminal:

```
roscore
```

If the following information is displayed, the ROS node is successfully started



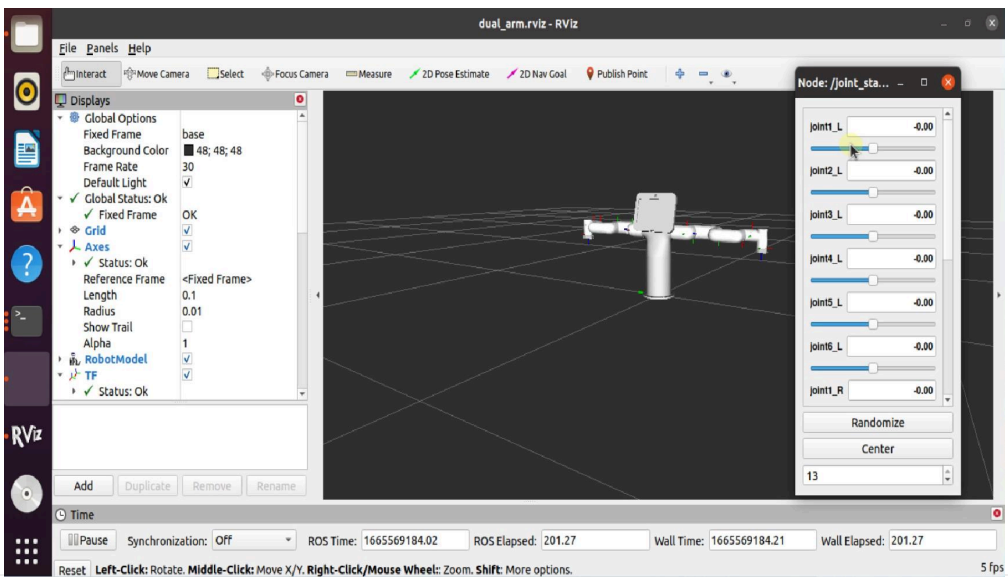
Note: roscore is required as the first step before starting the ros node.

1 Slider Control

Open a console terminal (shortcut key: `Ctrl+Alt+T`) and run:

```
roslaunch mybuddy_socket slider_control.launch
```

rviz and a slider component will be opened, and 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 mybuddy to move with the model, you need to open another console terminal (shortcut key: `Ctrl+Alt+T`), run:

Note: Due to the use of different networks, you can modify the corresponding IP address in `~/catkin_ws/src/mycobot_ros/Mybuddy/mybuddy_socket/scripts/slider_control.py` to ensure that it is consistent with **the server's IP**.

```
rosrun mybuddy_socket slider_control.py
```

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 console terminal (shortcut key: `Ctrl+Alt+T`), run:

Note: Due to the use of different networks, you can modify the corresponding IP address in `~/catkin_ws/src/mycobot_ros/Mybuddy/mybuddy_socket/scripts/follow_display.py` to ensure that it is consistent with **the server's IP**.

```
rosrun mybuddy_socket follow_display.py
```

Then open another console terminal (shortcut key: `Ctrl+Alt+T`), run:

```
roslaunch mybuddy_socket mybuddy_follow.launch
```

It will open rviz to show the model following effect.

3 myBuddy Moveit

`mycobot_ros` has integrated the MoveIt section.

Open a console terminal (shortcut key: `Ctrl+Alt+T`), run:

```
roslaunch mybuddy_socket_moveit demo.launch
```

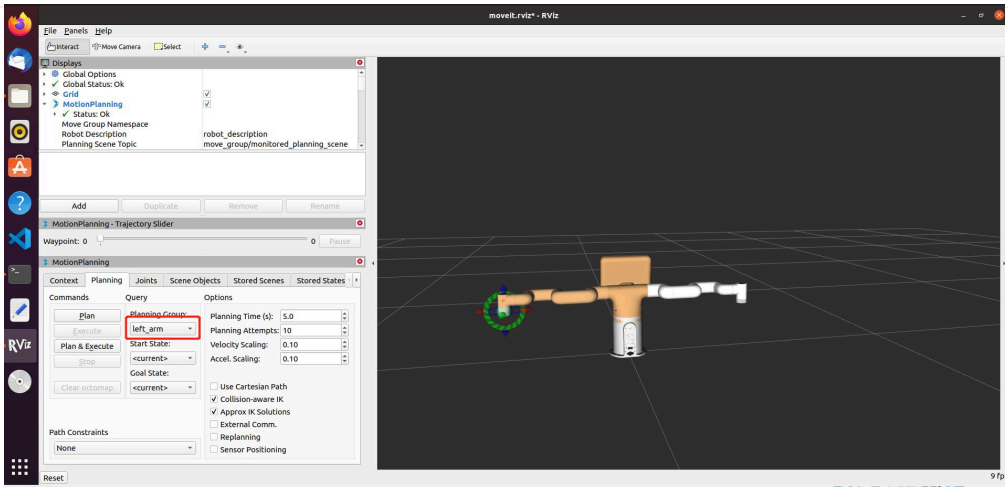
If you want a real robot arm to execute a plan synchronously, you need to open another console terminal (shortcut key: `Ctrl+Alt+T`), run:

Note: Due to the use of different networks, you can modify the corresponding IP address in `~/catkin_ws/src/mycobot_ros/Mybuddy/mybuddy_socket_moveit/scripts/sync_plan.py` to ensure that it is consistent with **the server's IP**.

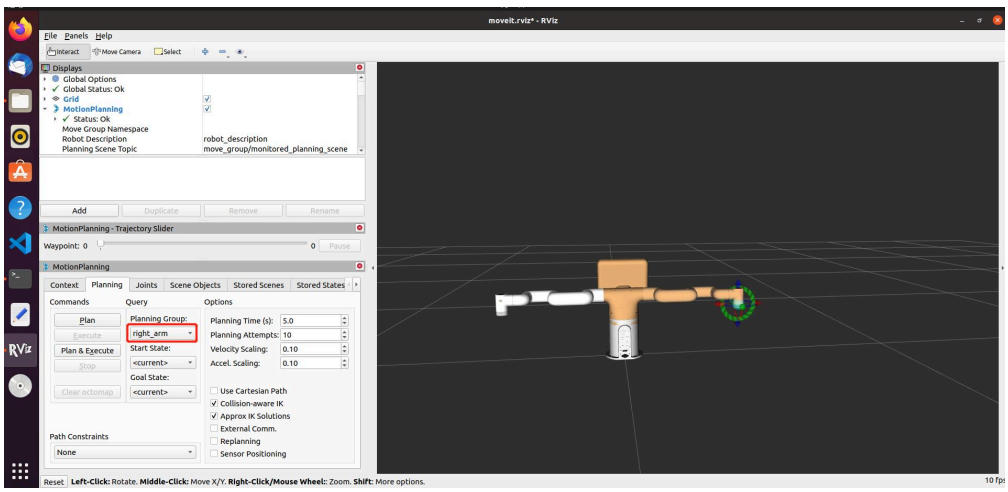
```
rosrun mybuddy_socket_moveit sync_plan.py
```

Moveit has four control groups, the operation effect is as follows:

1.**Left Arm Control Group**: plan the movement direction of **left arm** by dragging the trackball.



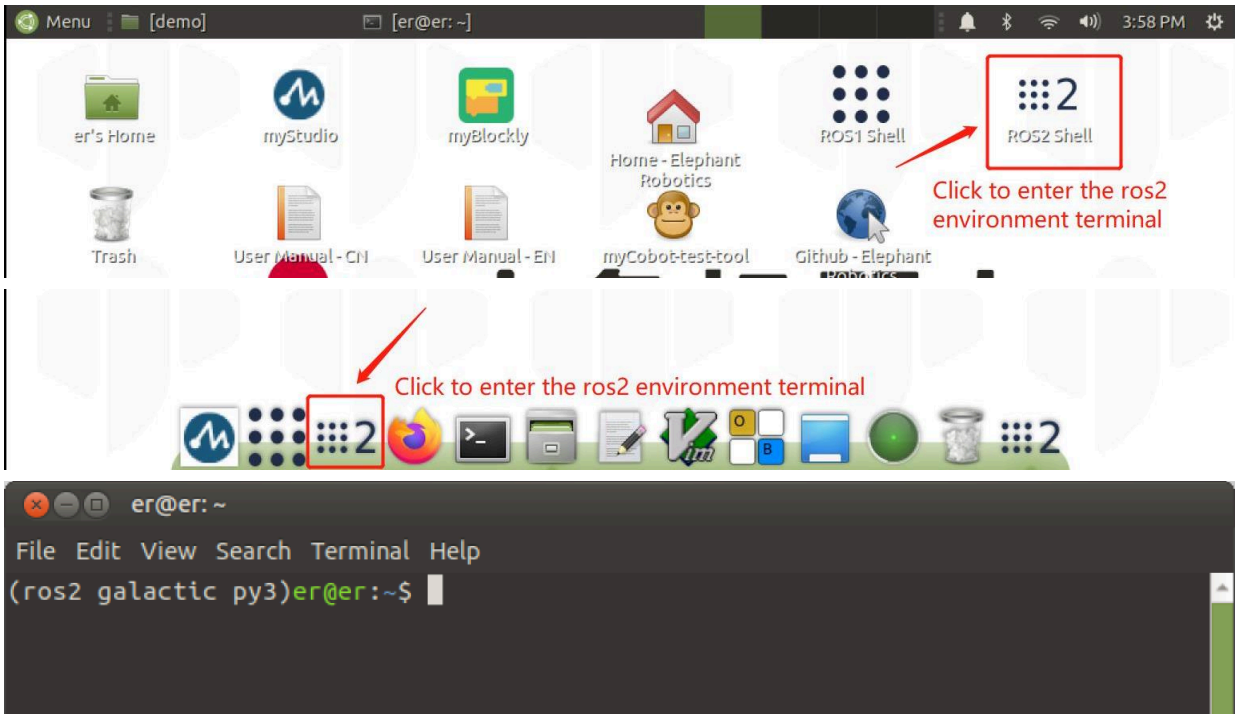
2.**Right Arm Control Group**: plan the movement direction of **right arm** by dragging the trackball.



Control and following of the robot arm

1 Slider Control

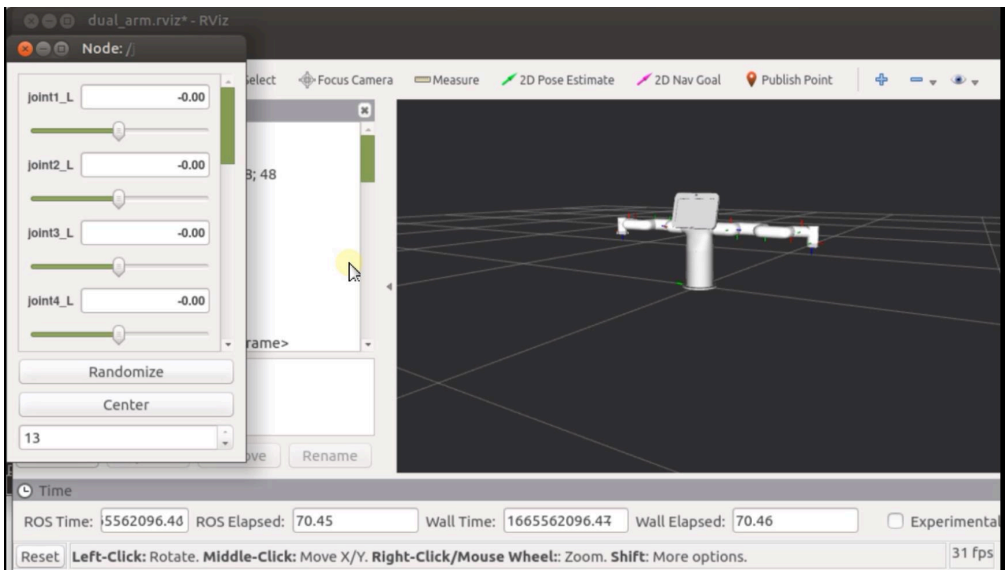
Click the ROS2 Shell icon on the desktop or the corresponding icon in the lower bar of the desktop to open the ROS2 environment terminal:



Then run the command:

```
# The default serial port name of the mybuddy is "/dev/ttyACM0", and the baud rate is 115200".
ros2 launch mybuddy slider_control.launch.py
```

rviz and a slider component will be opened, and you will see the following interface:



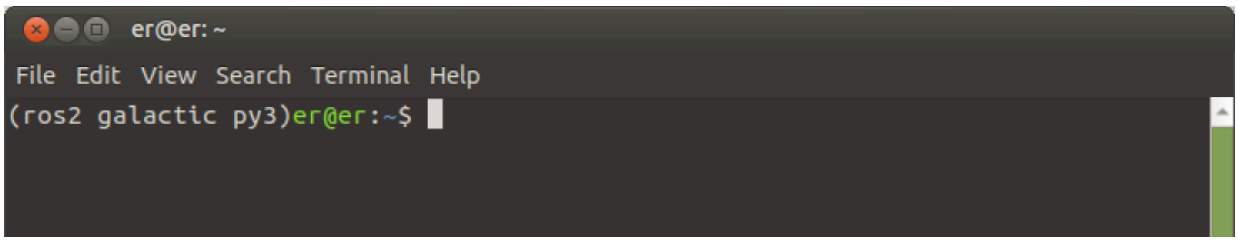
Then you can **control the model in rviz to make it move by dragging the slider**. The real mybuddy will move with it.

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 Follow

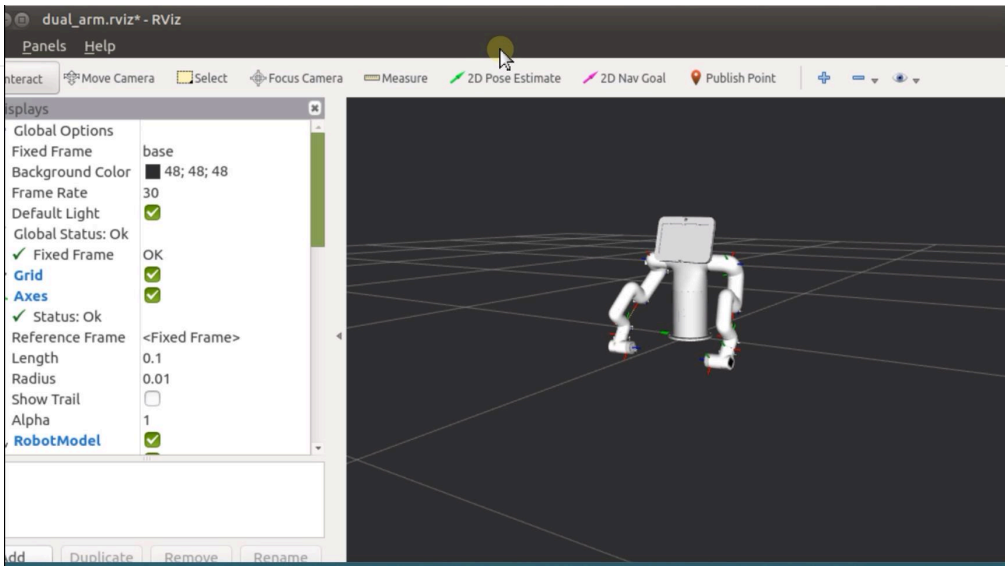
In addition to the above controls, we can also make the model follow the movement of the real robotic arm . Open a ROS2 environment terminal:



Then run the command:

```
# The default serial port name of the mybuddy is "/dev/ttyACM0", and the baud rate is 115200".
ros2 launch mybuddy mybuddy_follow.launch.py
```

It will open rviz to show the model following effect，the following picture：


dual_arm.rviz* - RViz
Panels Help
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Global Options
Fixed Frame base
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Ok
Fixed Frame OK
Grid
Axes
Status: Ok
Reference Frame <Fixed Frame>
Length 0.1
Radius 0.01
Show Trail
Alpha 1
RobotModel
Duplicate
Remove
Rename

What is Mirroring?

Mirroring is a form of file saving. It refers to the fact that data saved in one disc also exists in another disc without any distortion. Mirroring files are often saved as BIN, IMG, TAO, DAO, FCD. It is similar to ZIP packages, which make a series of files into one single file according to certain formats to meet users demands. The most fundamental function of mirroring is that is can be identified by a software immediately and recorded on disc. Generally, mirroring files can be extended to cover more information such as system files. Thus, mirroring files can contain the information of even a hardware. The most typical software for creating mirroring files is Ghost, featuring recording function to save information on a disc.

Mirroring Burning

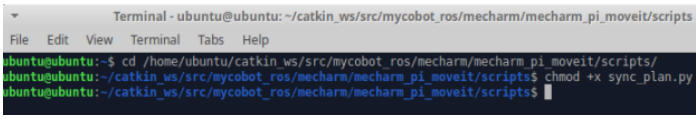
Mirroring is used to install and repair softwares and backup operating systems. If you have downloaded image, can check 1.2 Steps to Flash.

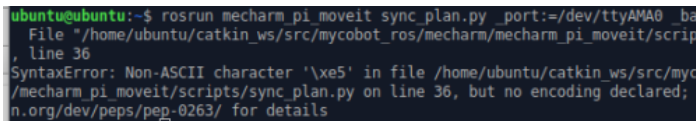
1.1 Links

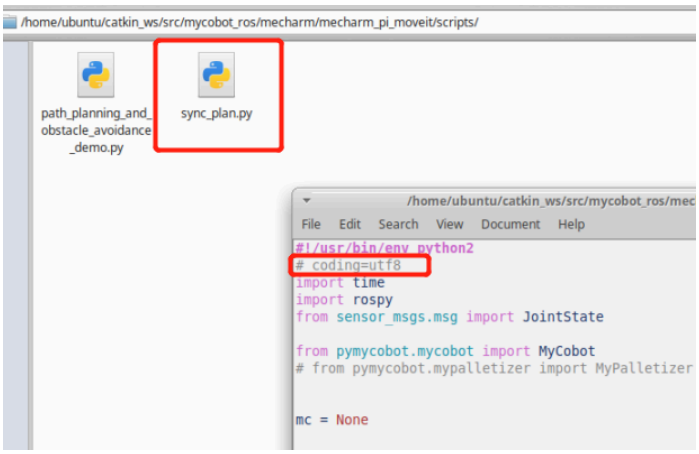
Product	Version	Link	SHA256 Hash
AI Kit 280	ubuntu 18.04	Download	d44439be351a52decdb4470cb623a032047e223ffce73477d29aa
myCobot 280 PI	ubuntu 18.04	Download	04e40af5b637ec003a8b23ef9012e353361fd336db4e17cf9a65feb
	ubuntu 20.04	Download	ce666e6c1047c512fe6b270336d472e48f231be12808729ed57f74
myCobot 280 JetsonNano	ubuntu 18.04	Download	2f1e40c1480b077bcc83abd3b79ac175f25d21e9cc344a0146361
myCobot 320 PI	ubuntu 18.04	Download	bc2ed6ef8d51a885f45379392b71e35420638a427d5b4b3a3c9d1
	ubuntu 20.04	Download	c95633bfd49246254f2be4783c6a91a15212422219157962c9312

<table>
<tr><td rowspan="2">mechArm 270</td><td>ubuntu 18.04</td><td>Download</td><td>9af1fcbf9c608eda269dc395a8d68ea0a270008a88ec8ec3cf97758</td></tr>
<tr><td>Mirror Known Issues</td><td>moveit function abnormal</td><td>Solution:

Execute moveit and the following prompt will appear, indicating th permission has been granted:
 

Enter the path shown in the figure and give the py file executable
 

Execute moveit and the following prompt will appear, indicating th encoding format in the code is wrong:
 

Enter the path shown in the figure, open the py file and enter at th #coding=utf8 and save it
 
</td></tr>
<tr><td rowspan="2">myPalletizer 260</td><td>ubuntu 18.04</td><td>Download</td><td>f6fe999519146428e4c60960b242f647ae5c73c704852d686b2858</td></tr>
<tr><td>ubuntu 20.04</td><td>-</td><td>-</td></tr>
</table>

myCobot Pro 600	Raspberry Debian	Download	6214baba79d9afa1f9036997c31fe2a2f687e7899792b8cb1e2e80
myBuddy 280	ubuntu 20.04	Download	2b5452f665bcb999faf1727b2103dc1e5745705f5706728e140d62
myAGV	ubuntu 18.04	Download	bedad7d9769cb69380c6a4b9742ba7aefc21db41ab239172b7a5a
myAGV 2023 Pi	ubuntu 20.04	点击下载	
myAGV 2023 JN	ubuntu 20.04	点击下载	
marsCat	-	-	-

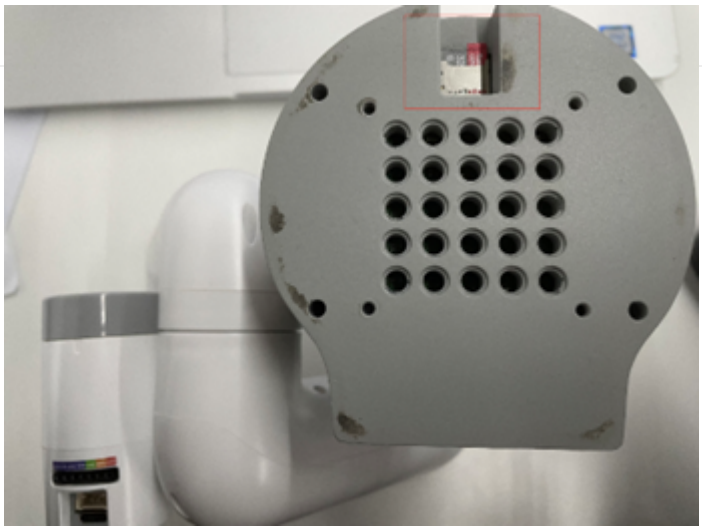
1.2 Steps to Flash

Step 1: Unzip the package and a file of image style appears.

Step 2: Download Win32DiskImager.

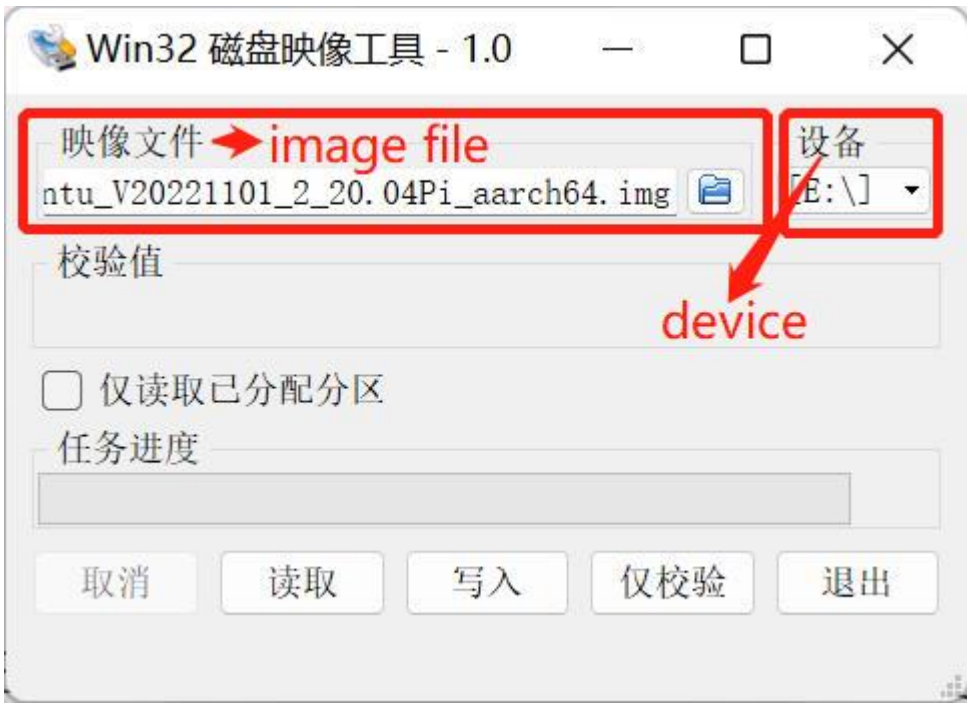
Go to Win32DiskImager to download.

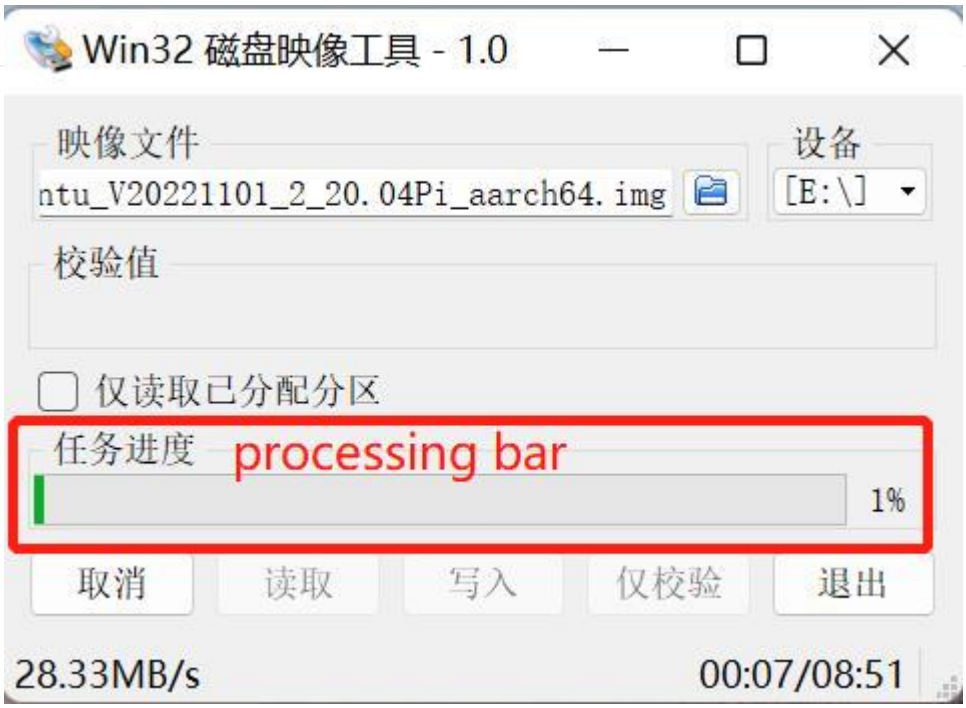
Step 3: Remove SD card from the pedestal, and then insert the SD card into PC.



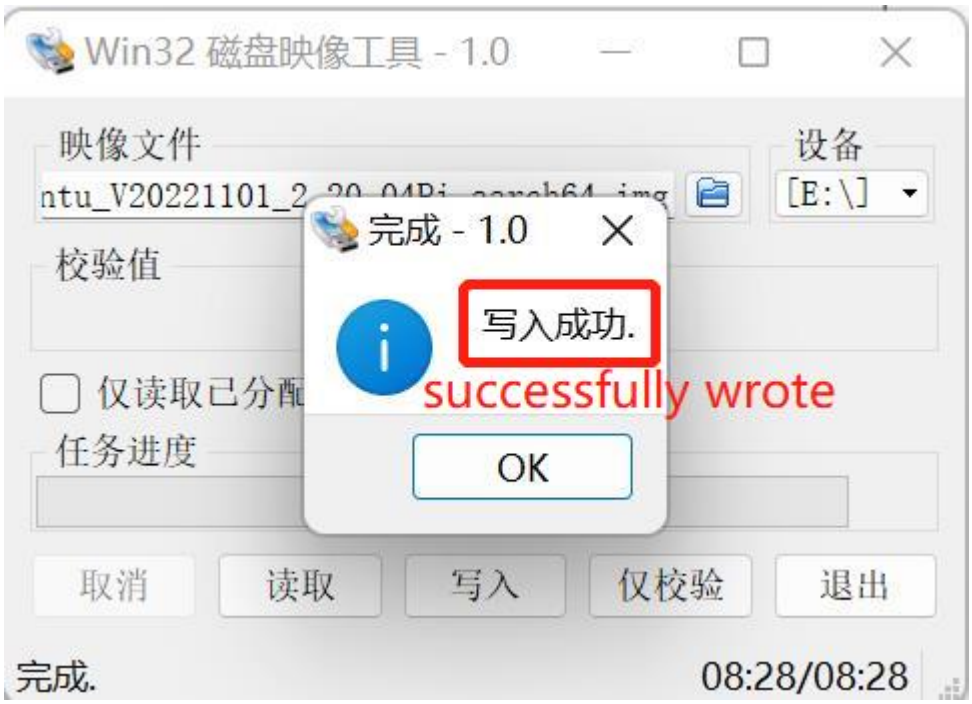
Step 4: Open Win32DiskImager.

Step 5: Select the software and device (E disc) and then write the software into PC.





Step 6: Successfully processed.



When to Download Mirroring

As a kind of copy of file, mirroring can help us solve many problems. If you are confronted with the following questions, please try to solve it through downloading a mirroring file.

1. When the system or the robot is required to be started again;
2. When you need a function package, like ROS;
3. When the original mirroring file is destroyed.

Message Board

💌 **We're delighted to receive your comments, if you have any suggestions or questions about our gitbook, you can write here, we'll reply you as soon as possible, thanks ❤**

⚠ If you can't see the message leaving place, it may be due to 🌐 network loading problems, please refresh the page, thanks！