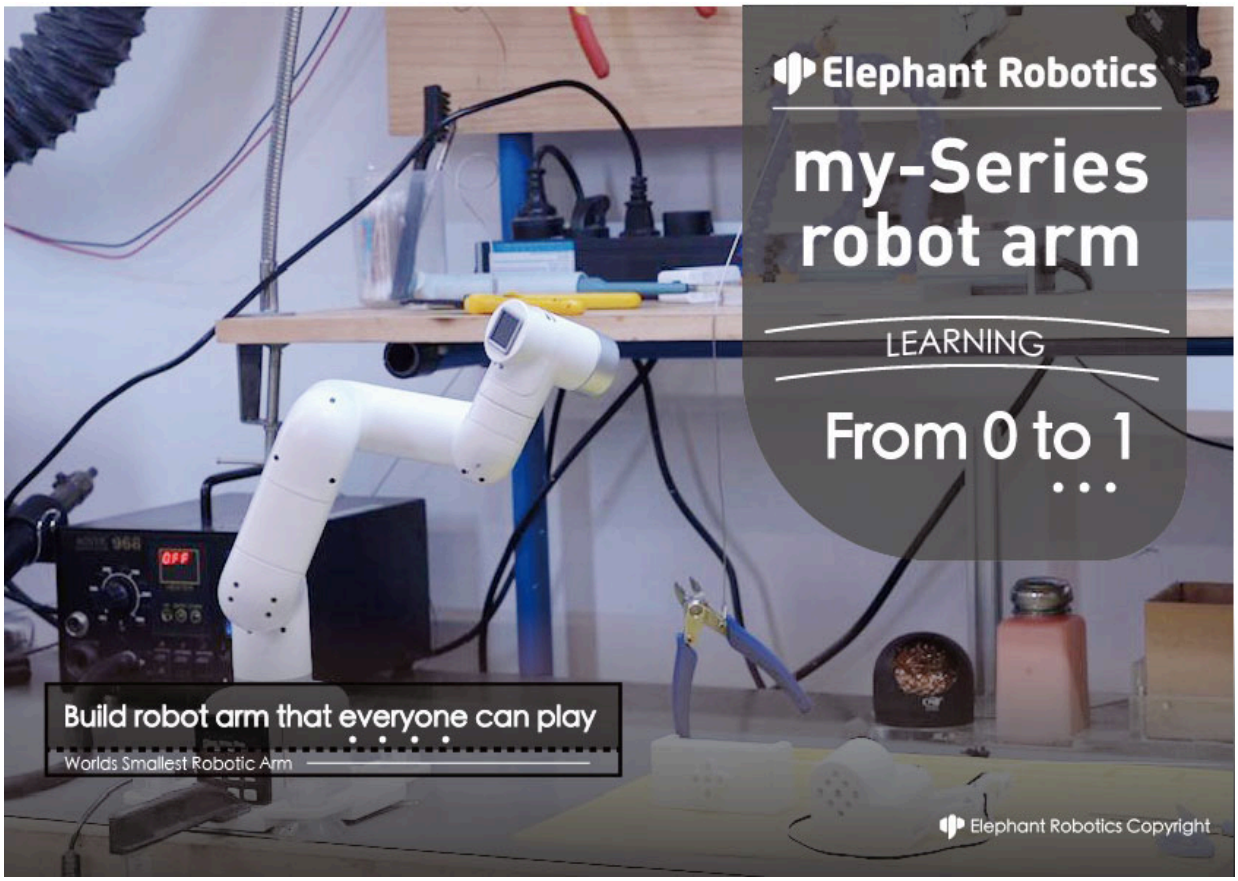
Table of Contents

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

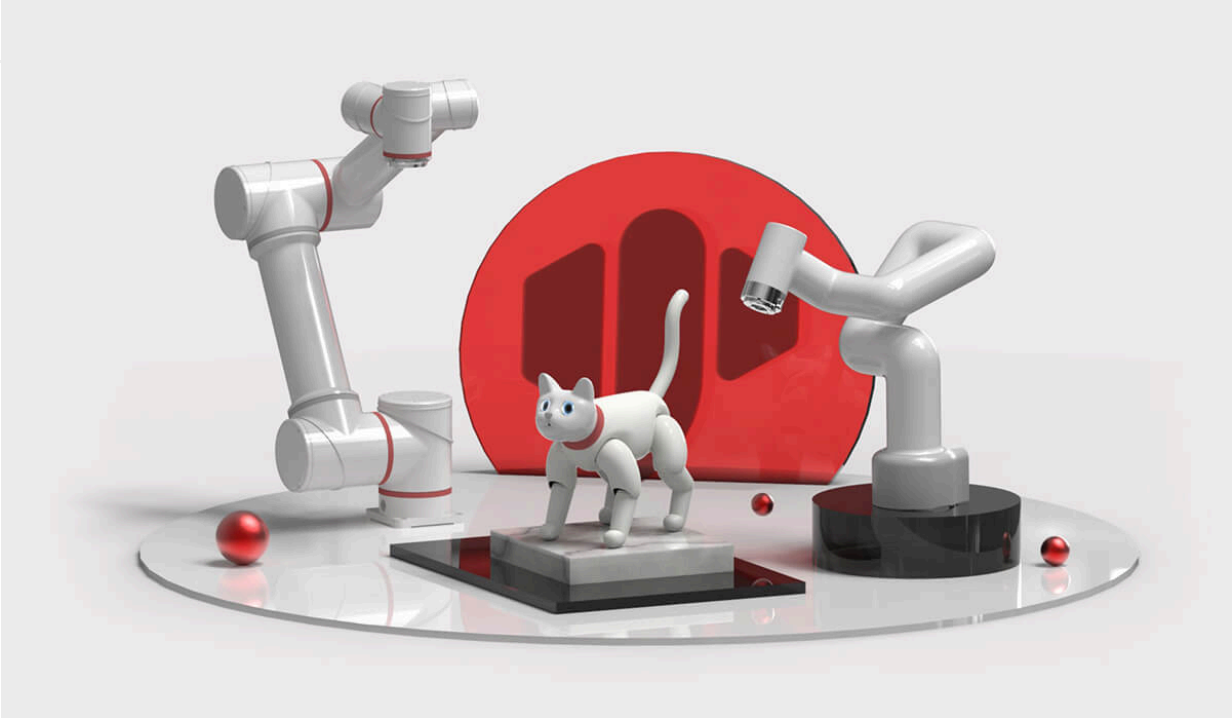
1 Company Introduction

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**.

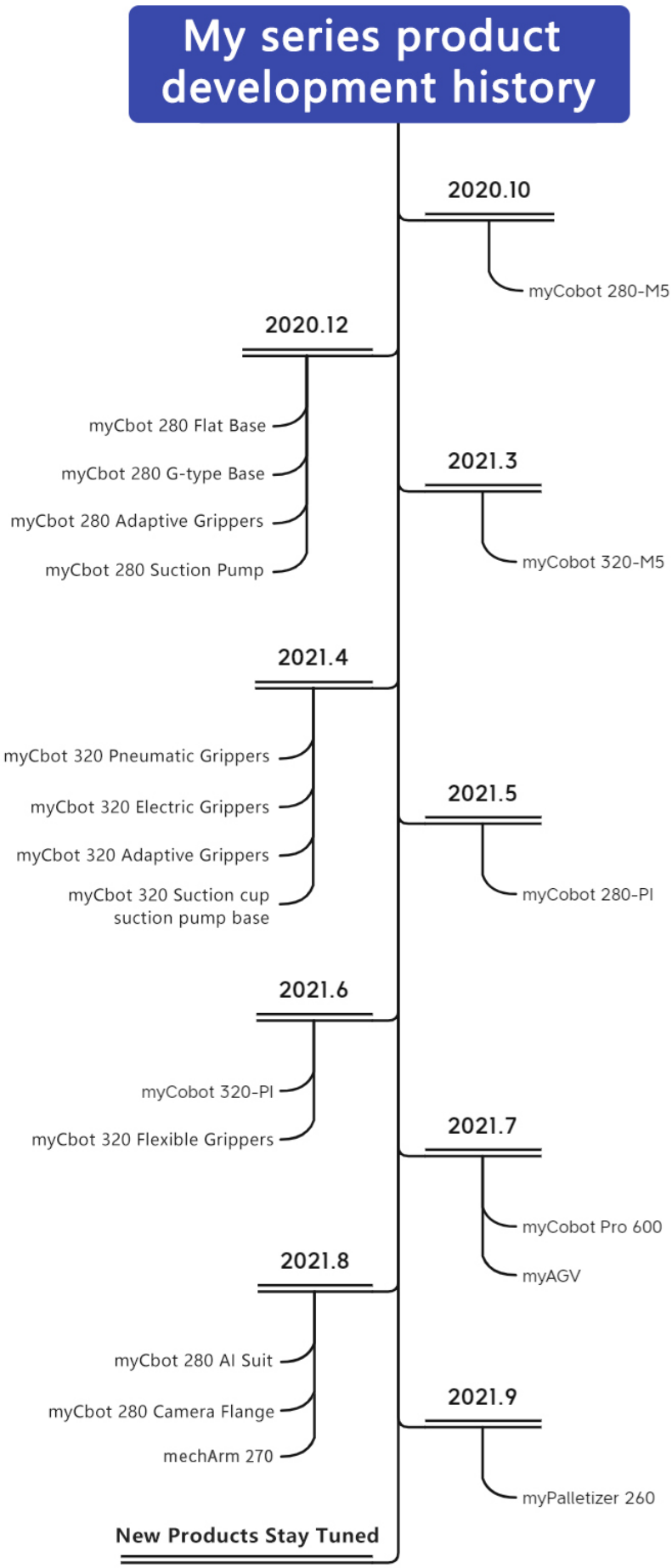
In July 2021, we published the commercial version of a babyelephant coordinative robot arm called **myCobot Pro 600**.

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**.

myPalletizer 260

> 260mm: refers to the effective working radius of the robotic arm

1 Product Introduction

The fully packaged lightweight four-axis palletizing robot arm has an overall finless design, which is compact and easy to carry. Designed for makers and education, it is easy to use, provides rich expansion interfaces, and can be developed twice; a variety of AI suits, machine vision, composite robots, etc. are available.

The 260 series is divided into: M5 version, Pi version, JN version and Arduino version.

- myPalletizer 260 for M5 is a joint product of Elephant Robotics and M5STACK;
- myPalletizer 260 for Pi uses a Raspberry Pi processor and is one of the core products of Elephant Robotics for robotics and artificial intelligence education ecology;
- myPalletizer 260 for JN is an official cooperation product between Elephant Robotics and NVIDIA;
- myPalletizer 260 for Arduino uses the M5STACK-ATOM ESP32 core controller, which is an entry-level product for elephant robots.

2 Product comparison

- M5--The body is equipped with two displays
  - I/O port: 6
- The base uses M5Stack-basic as the main control, and the end uses M5STACK Atom as the secondary control.
- It supports thousands of application ecosystems of M5, which is convenient for expanding application interaction output.
- Pi--Embedded Raspberry Pi ecology, unlimited development possibilities
  - I/O port: 12
  - Raspberry Pi 4B, 1.5GHz quad-core microprocessor, running Debian/Ubuntu platform.
- Support 4-way USB, 2-way HDMI, standardized GPIO interface, TF card pluggable.
- JN--NVIDIA official cooperation product, using JETSONNANO + ATOM dual-core main control

    - I/O port: 12
    - JETSONNANO, 1.5GHz quad-core microprocessor, running Debian/Ubuntu platform.
    - Support 4-way USB, 2-way HDMI, standardized GPIO interface, TF card pluggable.
- Arduino--M5STACK-ATOM ESP32 core master, infinite possibilities for development
    - I/O port: 6
    - The device can be controlled by a serial port cable, or it can be extended with various types of development versions. There is no need to migrate code, just a simple link to start expansion development.
    - Built-in robot forward and inverse kinematics; open various control interfaces such as angle, coordinates, speed, current, voltage, IO, etc.; support RVIZ simulation and MOVEIT interface development.

3 Purchase link

- Taobao: https://shop504055678.taobao.com
- shopify: https://shop.elephantrobotics.com/

For details, please check: 260 M5 chapter , 260 Pi chapter , 260 JN chapter and 260 Arduino chapter .

myPalletizer 260 for M5

1 Product Introduction

Zero Basic Entry Level-Desktop Robotic Arm

The new series of myPalletizer elephant palletizing robot arm, which is jointly produced with M5STACK, is a fully packaged lightweight four-axis palletizing robot arm. The overall finless design is compact and easy to carry.

myPalletizer body weight 960g, load 250g, working radius 260mm; specially designed for makers and education, easy to use, rich expansion interfaces, unlimited development possibilities; a variety of AI suit options, machine vision, composite robots, etc., learn about the cutting-edge technology of learning robots Knowledge inspires innovative thinking.

2 Product performance

- The fin design, the appearance is fully wrapped
  - The optimal space-removing fin design concept that can be loaded into a backpack subverts the traditional link-type educational four-axis robotic arm.
- LED dual display
  - The robotic arm carries two display screens, which is easy to expand the interactive output of the application, and the gameplay is unlimited.
- LEGO ecology, compatible with all my series accessories
  - The patented Lego hole design is shared globally, and the my series hardware ecological platform concept is implemented, and the end accessories are plug-and-play.
- Zero foundation, easy to get started
  - It is easy to get started and use, and the graphical programming language allows you to easily start the journey of using the robotic arm.
- Support ROS ecology
  - It is developed using the global mainstream robot communication framework ROS, and supports simulation, control and algorithm verification in a virtual environment, which reduces the requirements for the experimental environment and improves the experimental efficiency.
- Super cost-effective

- The first robotic arm that truly realizes one machine for every hand and can be played by everyone.

3 Application scenarios

3.1 Intelligent palletizing

- Simulate an industrial robotic arm, realize a variety of palletizing methods, build your own mini smart warehouse, and achieve more exciting things.

3.2 Object recognition and grasping

- Recognize colors through machine vision, and realize the sorting, sorting, and palletizing of objects by the robotic arm. The creativity of various gameplays is unlimited.

3.3 Writing, drawing, touch

- The perfect combination of robotic arm and art, creating infinite possibilities.

3.4 Composite robo

- The mobile compound robot allows the workspace to expand sufficiently to allow it to complete more tasks.

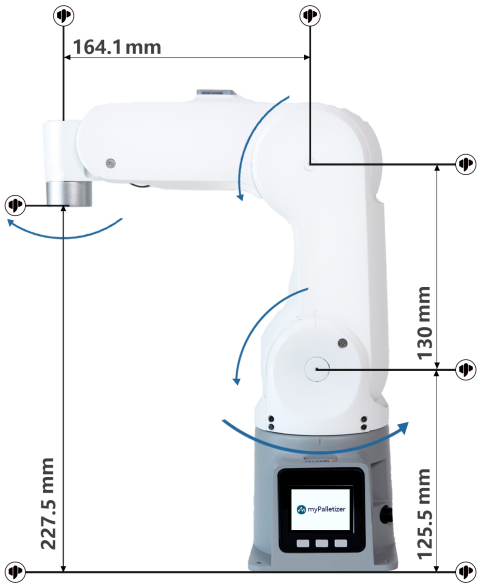
myPalletizer 260 for M5

1 Structural parameters

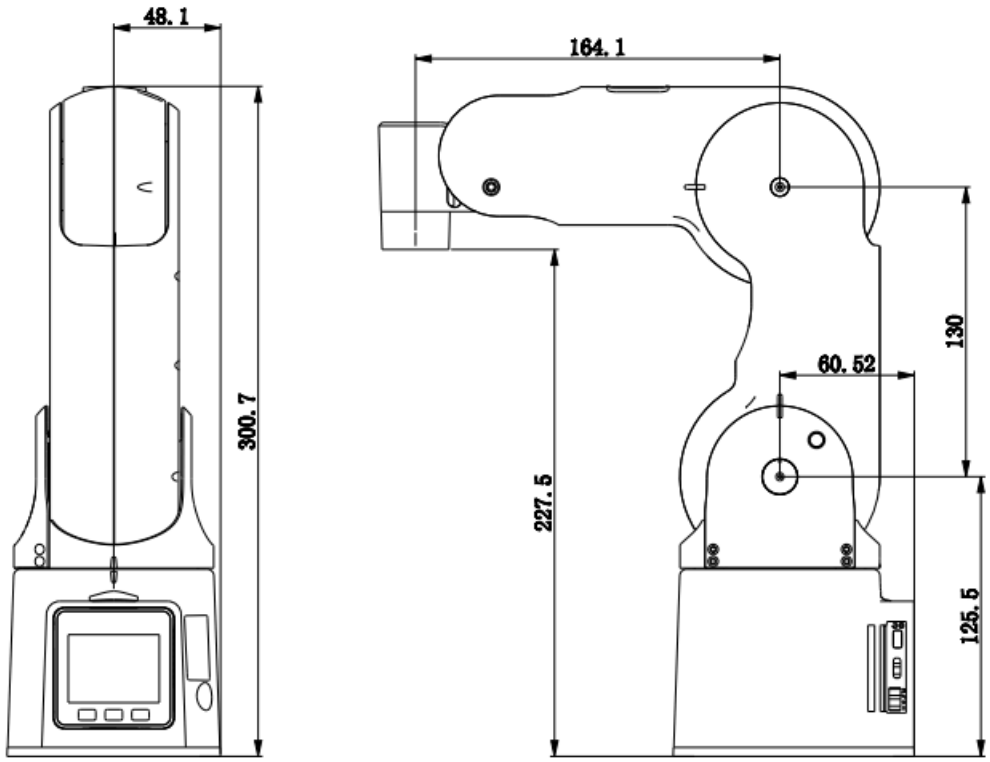
1.1 Robot arm parameters

index	parameter
name	Elephant palletizing robotic arm
model	myPalletizer 260
degrees of freedom	4
Repeatability	±2 mm
load	250g
dead weight	960g
working radius	260mm
Material	Photosensitive resin SLA
Charging voltage	8~12V 5A
Motor type	High Precision Magnetic Encoder Servo Servo
Movement maximum speed	120°/s
control	M5Stack-basic
way of communication	USB/Type-C

1.2 Workspace



1.3 Specifications and dimensions

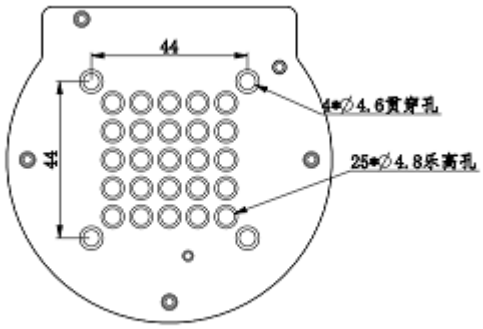


1.4 Range of motion of joints

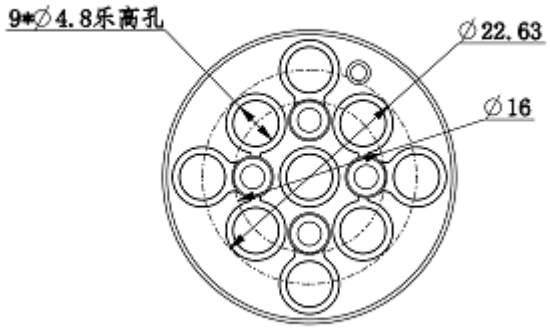
The essential	Scope
J1	-160 ~ +160
J2	0 ~ +90
J3	0 ~ +60
J4	-∞ ~ +∞

1.5 Hole installation

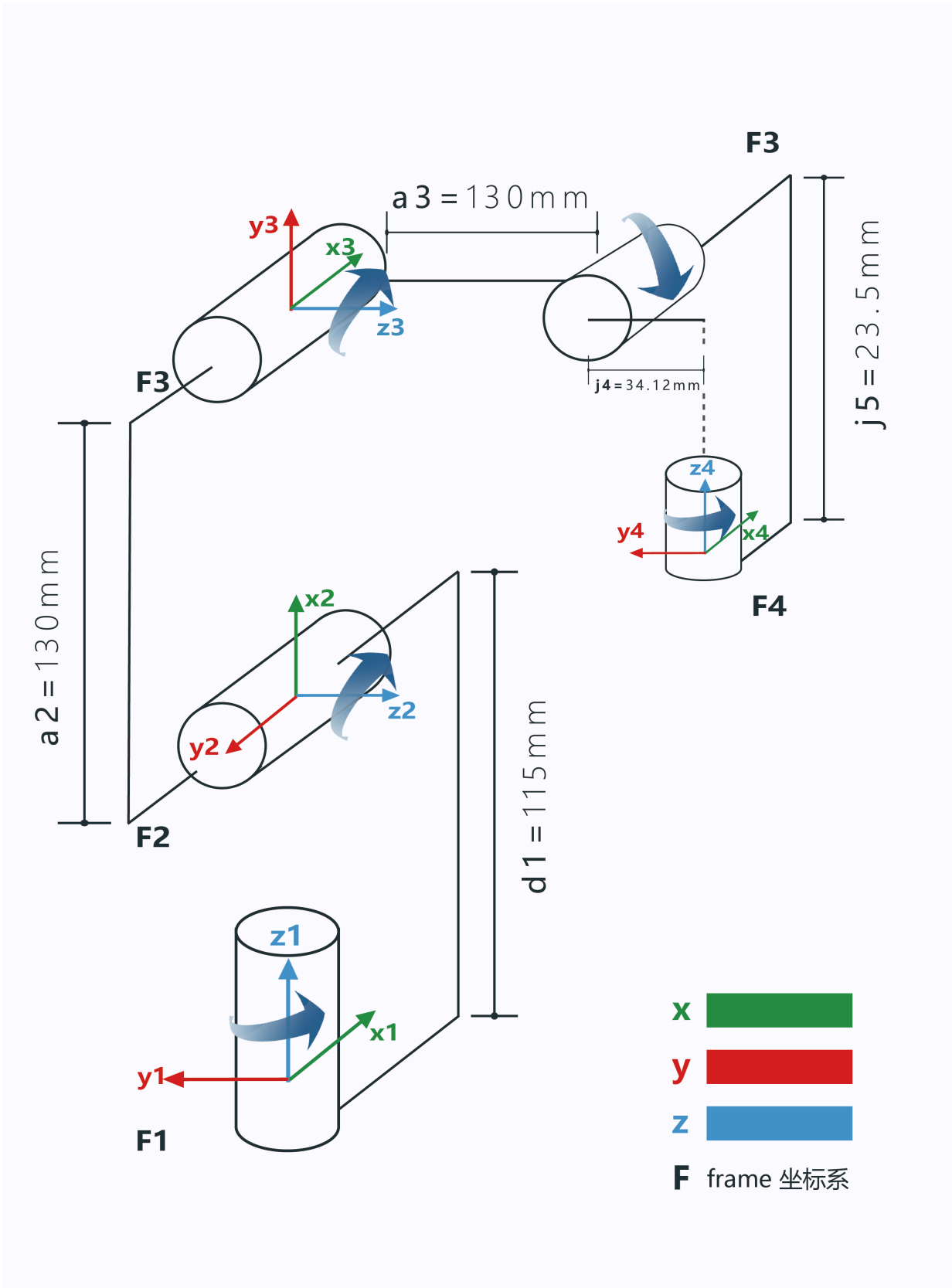
- The Robot Base Mounting Flange Base supports the installation of LEGO components.



- Robot end mounting flange The end of the robot arm supports the installation of LEGO components.



1.6 DH parameter



2 Hardware Parameter

Items	Parameter
Core	240MHz dual core. 600 DMIPS, 520KB SRAM. Wi-Fi, dual mode Bluetooth
Flash	4MB
IO	G19, G21, G22, G23, G25, G33
Bluetooth	2.4G/5G
Wireless type	2.4G 3D Antenna
Core type	M5Stack-basic/Atom

Robotic Arm Electrical Interface

1 Base electrical interface

1.1 Introduction to the base

A，Figure 1-1 shows the front ports and buttons on the base：

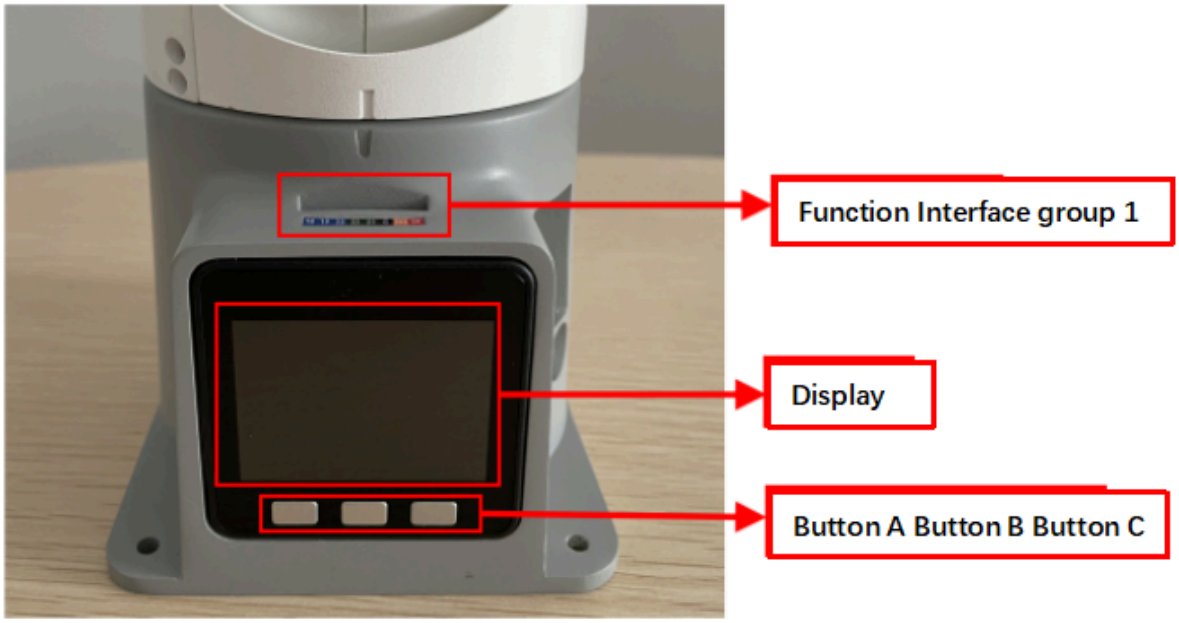


Figure 1-1 Front view of the base

B, Figure 1-2 shows the ports on the left of the base：

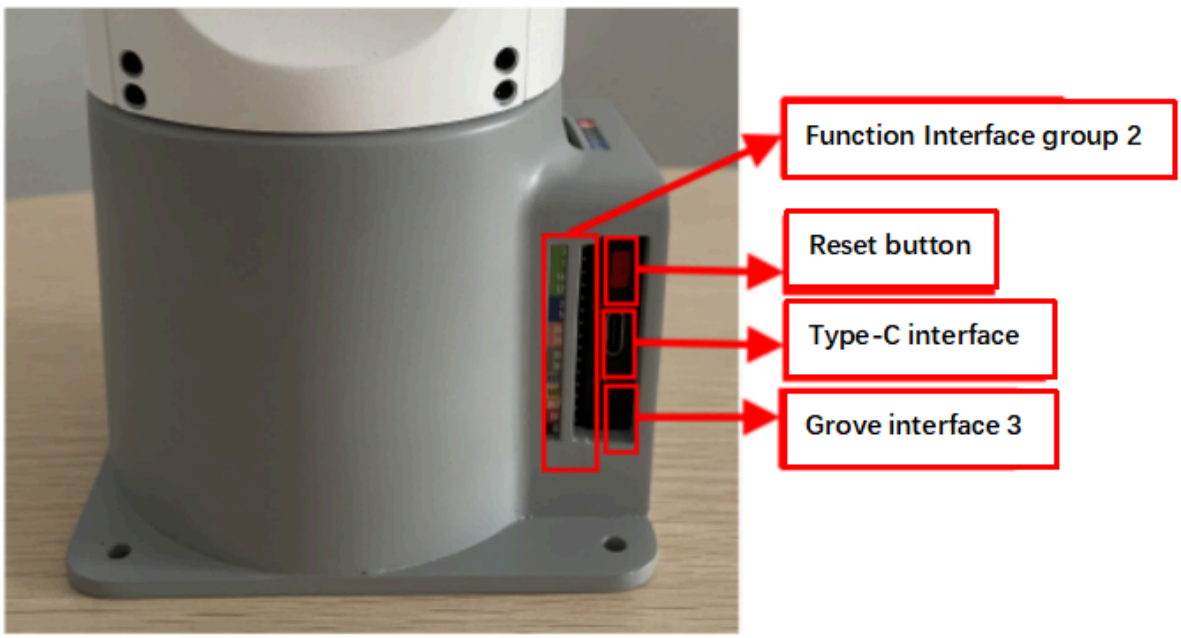


Figure 1-2 Left view of the base

C, Figure 1-3 shows ports on the right of the base：

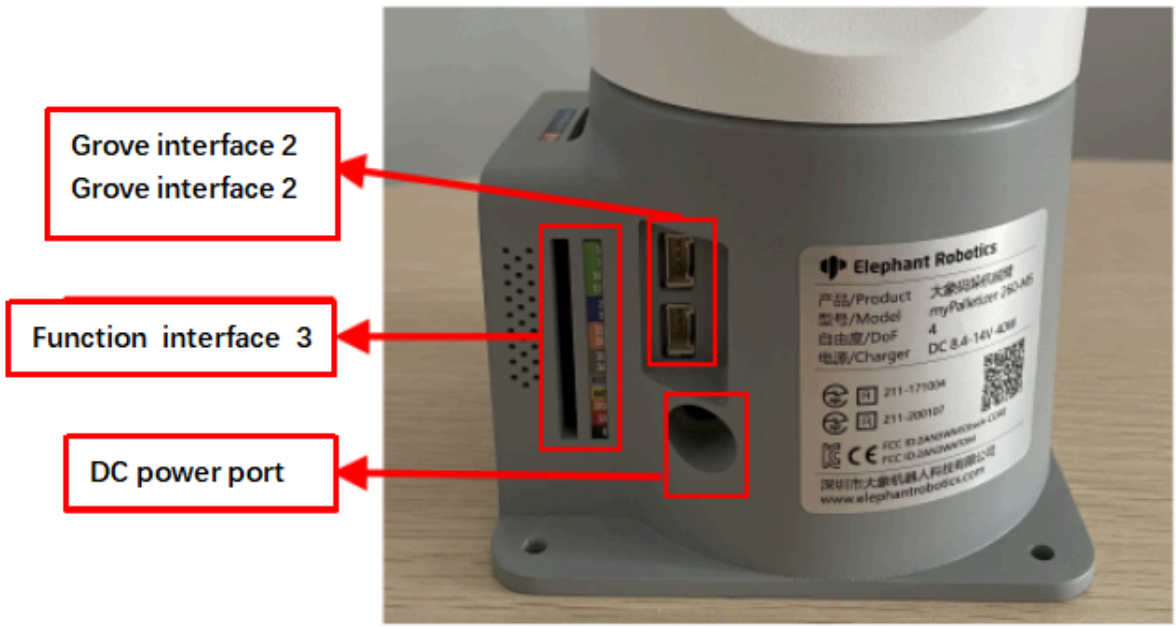


Figure 1-3 Right view of the base

1.2 Description of the bottom electrical interface

Note: Function interface group is 2.54mm Dupont interface, external can use 2.54mm Dupont wire.

A. Table 1-1shows the definition of each interface in a functional interface group 1.

Tag	Signal	Function	Note
18	G18	-	Temporarily not open
19	G19	3.3 V-out-PNP output /3.3 V-int input	
23	G23	-	Temporarily not open
22	G22	3.3 V-out-PNP output /3.3 V-int input	
21	G21	3.3 V-out-PNP output /3.3 V-int input	
G	GND	Motherboard power signal ground	
3V3	3V3	DC3.3 V power supply	
5V	5V	Power supply, DC5V	

Table 1-1 Functional interface group 1

B. The definitions of interfaces in functional interface group 2 are the same as those in functional interface group 3, as shown in Table 1-2.

Tag	Signal	Function	Note
3	G3	3.3 V-out-PNP output /3.3 V-int input	This parameter is unavailable when usB-Type-c is used
1	G1	3.3 V-out-PNP output /3.3 V-int input	This parameter is unavailable when usB-Type-c is used
16	G16	-	Temporarily not open
17	G17	-	Temporarily not open
2	G2	3.3 V-out-PNP output /3.3 V-int input	
5	G5	3.3 V-out-PNP output /3.3 V-int input	
25	G25	3.3 V-out-PNP output /3.3 V-int input	
26	G26	3.3 V-out-PNP output /3.3 V-int input	
35	G35	3.3 V-int input	
34	G34	3.3 V-int input	
RST	RST	Master reset	
BAT	BAT	-	Temporarily not open
3V3	3V3	DC3.3 V power supply	
5V	5V	Power supply, DC5V	
G	GND	Motherboard power signal ground	

Table 1-2 Functional interface groups 2 and 3

Description: Figure 1-5 shows other functions of the interface. If other functions are used, the I/O function is unavailable.

GPIO TYPE	Analog Function	M-BUS LINE 0		LINE 1		Analog Function	GPIO TYPE
		GND		ADC	G35	ADC1_CH7	I
		GND		ADC	G36	ADC1_CH0	I
		GND		RST	EN		
I/O/T		G23	MOSI	DAC/SPK	G25	ADC2_CH8	I/O/T
I/O/T		G19	MISO	DAC	G26	ADC2_CH9	I/O/T
I/O/T		G18	SCK	3.3V			
I/O/T		G3	RXD1	TXD1	G1		I/O/T
I/O/T		G16	RXD2	TXD2	G17		I/O/T
I/O/T		G21	SDA	SCL	G22		I/O/T
I/O/T	ADC2_CH2/T2	G2	GPIO	GPIO	G5		I/O/T
I/O/T	ADC2_CH5	G12	IIS_SK	IIS_WS	G13	ADC2_CH4/T4	I/O/T
I/O/T	ADC2_CH3/T3	G15	IIS_OUT	IIS_MK	G0	ADC2_CH1/T1	I/O/T
		HPWR		IIS_IN	G34	ADC1_CH6	I
		HPWR		5V			
		HPWR		BATTERY			

Figure 1-4

C. Power DC interface: The myPalletizer is powered by a 6.5mm od, 2.0mm OD, and a manufacturer's 8.4V 5A DC power adapter

D. Grove interface: Figure 1-5, Figure 1-6, and Figure 1-7 show the Grove interface definitions

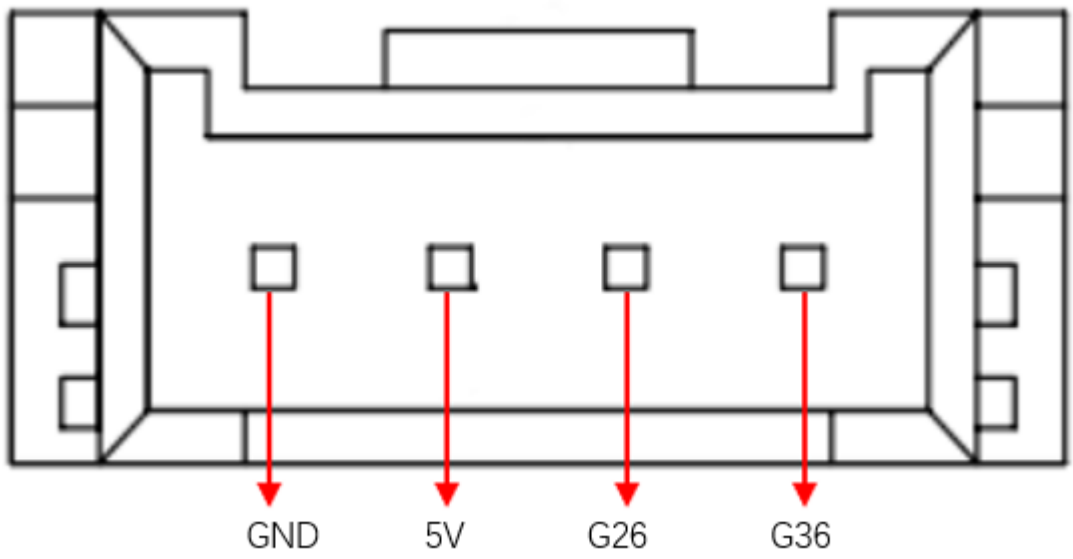


图1-5 Grove port 1

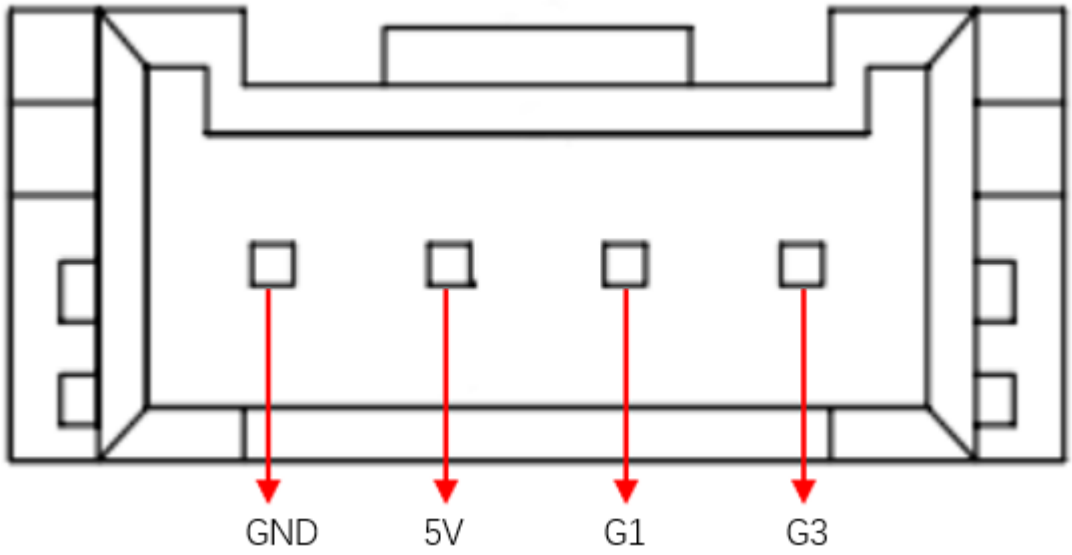


图1-6 Grove pin port 2

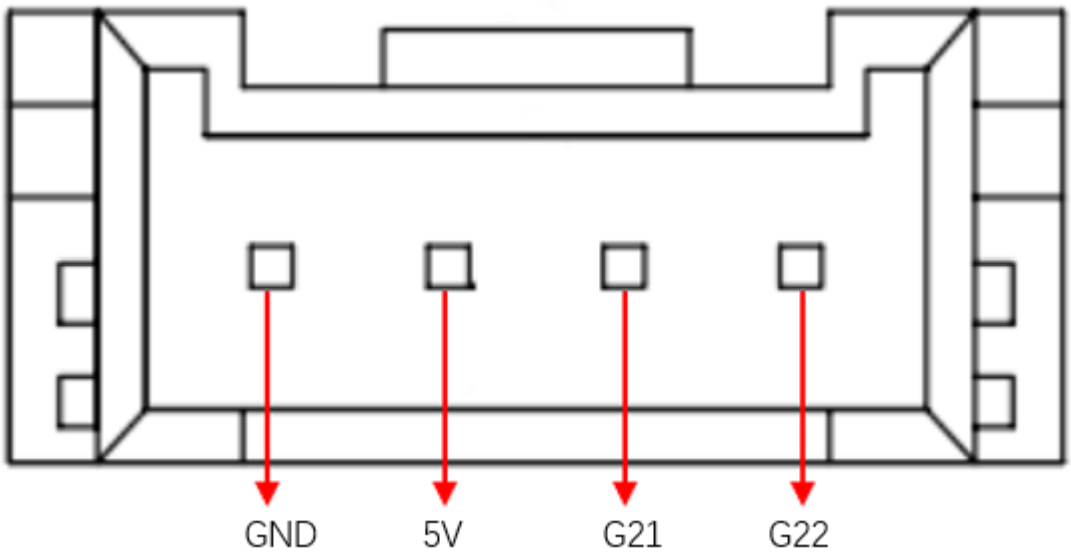


图1-7 Grove port 3

E. type C interface: used to communicate with the PC. G1 and G3 ports are occupied when this interface is used.

F. Reset button: used when the main control system is reset

G. Buttons A, B, and C: Cooperate with the display

H. Display: The 2-inch IPS screen can be used to display mycobot communication status and adjust robot origin with buttons.

2 Mechanical arm end electrical interface

2.1 The end of the manipulator is introduced

A. Figure 2-1 show the side interfaces at the end of the manipulator.

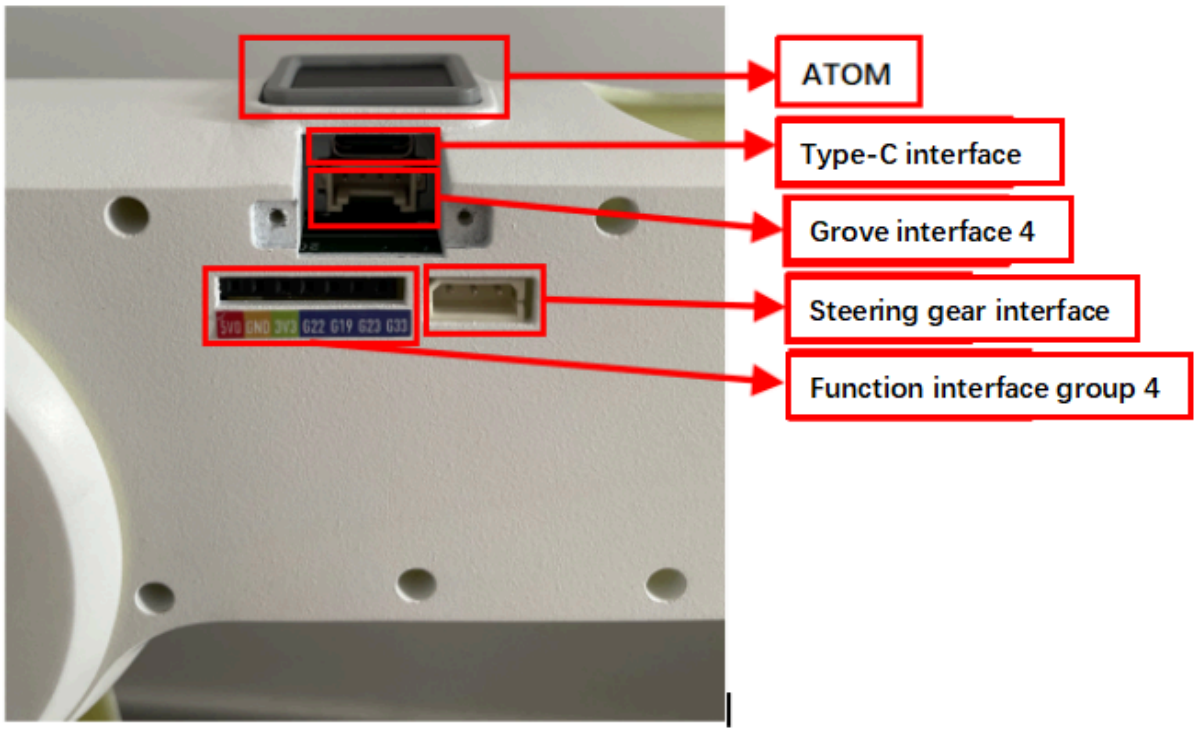


Figure 2-1 End of the manipulator

2.2 Description of terminal electrical ports

A. Table 2-1 shows the definition of each interface in a function interface group 4.

Tag name	Signal name	Function	Note
5V0	5V	Power supply, DC5V	
GND	GND	Motherboard power signal ground	
3V3	3V3	DC3.3 V power supply	
G22	G22	3.3 V-out-PNP output /3.3 V-int input	
G19	G19	3.3 V-out-PNP output /3.3 V-int input	
G23	G23	3.3 V-out-PNP output /3.3 V-int input	
G33	G33	3.3 V-out-PNP output /3.3 V-int input	

Table 2-1 Functional port group 4

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

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

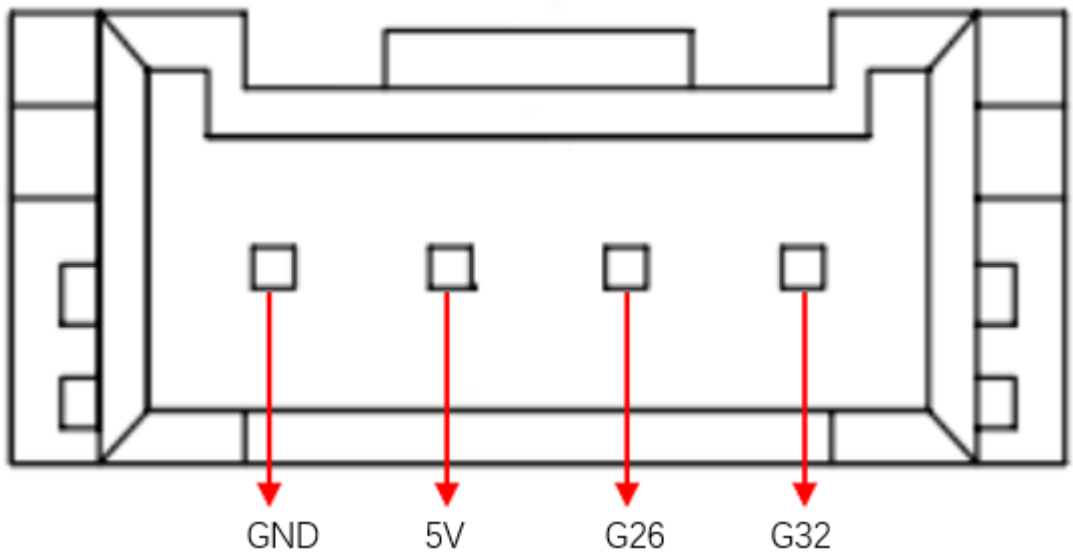


Figure 2-2 Grove port 4

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)

Introduction

1 Unpacking and working environment

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

Table 2-2 myPalletizer Robot Arm [Standard Set]

Product Contents of myPalletizer Robot Arm [Standard Set]	**-myPalletizer robotic arm (model myPalletizer 260 for M5)** **-myPalletizer Robot Arm** **-Product Album** **-myPalletizer Robot Arm** **-Supporting Power Supply** **-USB-Type C** **-Jumper**

Install the robot system in an environment that meets the conditions described in the table in order to develop and maintain the performance of this machine and use it safely.

Table 2-3 Working Environment and Conditions

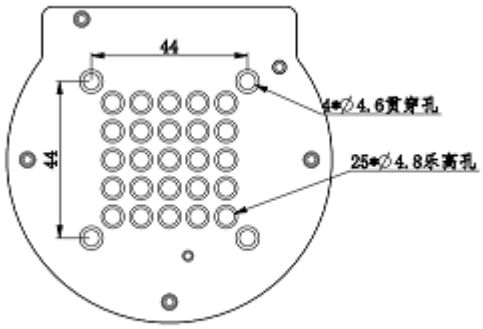
Working Environment	Conditions
temperature	-10°C~45°C
Relative humidity	20%~70%
Indoor and outdoor requirements	indoor
Other environmental requirements	- Avoid sunlight exposure. - Keep away from dust, oil fume, salt, iron filings, etc. - Keep away from flammable and corrosive liquids and gases. - Must not come into contact with water. - Does not transmit shock and vibration etc. - Keep away from strong electromagnetic interference sources.

2 Installation conditions and requirements

The actual weight of the myPalletizer robot is 960g. Considering that during use, as the robot moves, its center of gravity will move, so the robot needs to be fixed on a solid base before it can be used normally. Base weight requirements: fixed base, or mobile base.

Robot base interface size

The base fixing hole is the interface between the fixed robot and other bases or planes. The specific hole size is shown in the figure below, and it is installed with LEGO components.



Please make sure there are corresponding holes on the fixed base before installing.

Before the official installation, please confirm:

- The environment to be installed meets the requirements of Section 2.4.1.3.2 above.
- The installation position is not smaller than the working range of the robot, and there is enough space for installation, use, maintenance and repair.
- Put the stand in a suitable position.
- Installation related tools are ready.

After confirming the above, please move the robot to the base mounting table, adjust the robot position, and align the fixing holes of the robot base with the holes on the base mounting table. After aligning the hole, press it.

Note: 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.

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

3 Power on the robot

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

Figure 3-1 Location of the power connector

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

- Rated voltage: 8-12V
- Rated current: 5A
- Plug Type: DC 5.5mm x 2.5

Note that you can't just use the TypeC plugged into the M5Stack-basic for powering. Use an official power supply to avoid damage to the robotic arm.

4 Driver Installation

Users can download the corresponding CP210X or CP34X driver compressed package according to the operating system they are using, and after decompressing the compressed package, select the installation package with the corresponding operating system digits for installation.

There are currently two driver chip versions, CP210X (applicable to CP2104 version) / CP34X (applicable to CH9102 version) driver package. If you are not sure which USB chip your device is using, you can install both drivers at the same time.

The installation tutorial can be viewed by scanning the QR code:

5 Use USB to connect computer

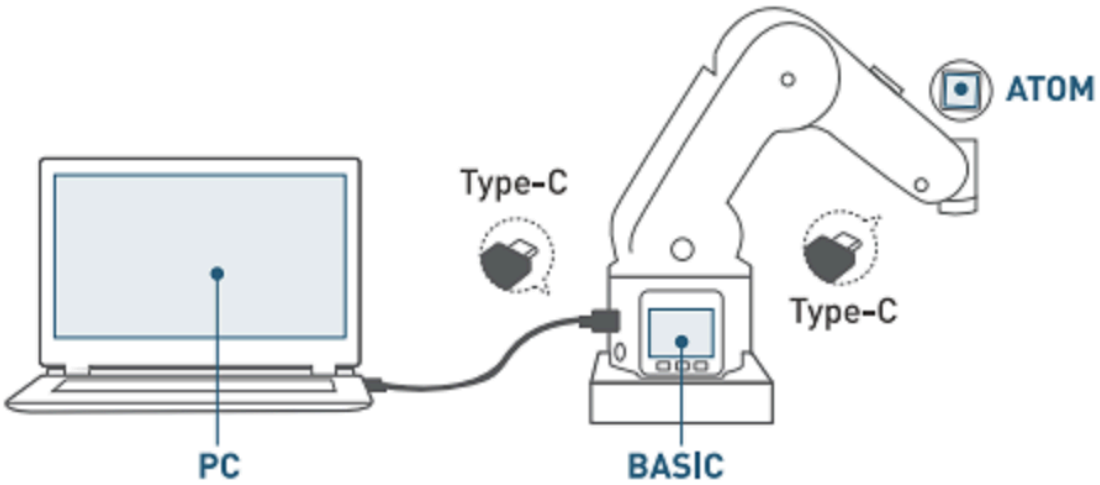


Figure 3-2 Schematic diagram of the USB interface

\1) Use the Type-C cable to connect to the computer and the corresponding USB interface of the robotic arm M5Stack-basic, and check the connection. Connection detection is a detection function for the connection status of the motors and Atoms in the robot arm. This feature makes it easy for customers to troubleshoot equipment.

\2) After passing the connection detection, if the device is not faulty, the communication forwarding detection can be performed. The timeliness of communication is very important for the microcontroller manipulator. For the microcontroller manipulator, we usually send control commands to the M5Stack-basic of the base, and forward them through communication, and the end effector will parse the commands and then execute the target action. .

6 Use of end tools

6.1 Suction pump

Details: 4 Suction pump

Please scan the code to watch the installation tutorial of the suction pump:

6.2 Adaptive gripper

Details: 5 Adaptive gripper

Adaptive gripper installation video to be added...

Development environment and construction

1 Use environment

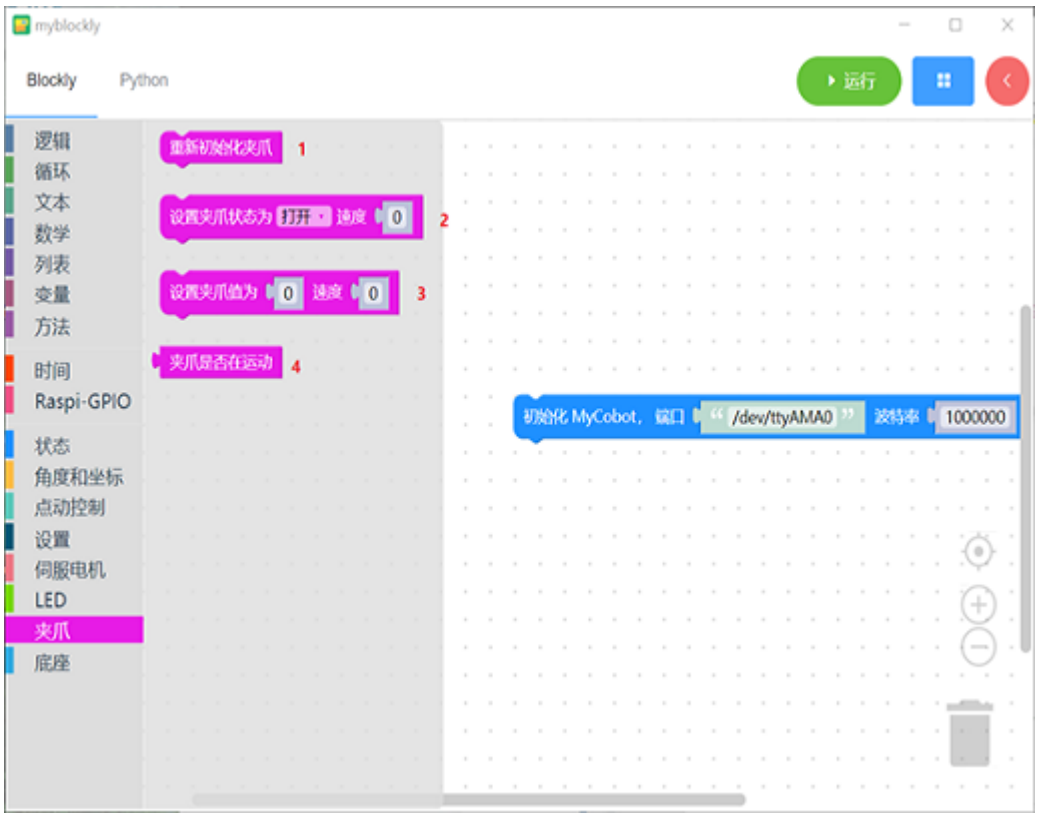
The myPalletizer 260 for M5 version is developed and used based on PC. There is no built-in system inside the robotic arm, so the combination of the robotic arm and the PC is required during use, so please prepare a PC before use.

2 Development Environment

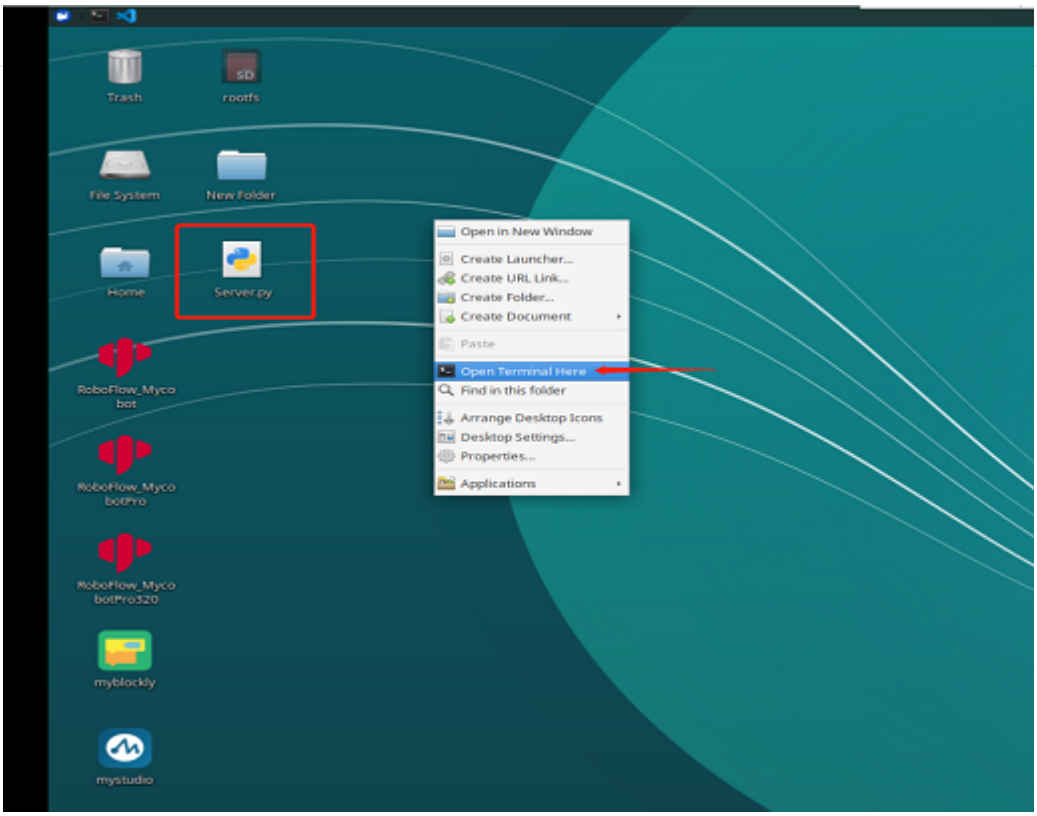
Because myPalletizer 260 for M5 has no built-in development environment, you need to use a PC to install the development environment for the robotic arm.

The following are the development environments supported by myPalletizer 260 for M5 and the installation and usage tutorials:

- 2.1 Development and use based on Blockly Both myBlockly and UIFlow embody the idea of drag-and-drop programming. Both are graphical programming software and visualization tools. Users can build code logic by dragging and dropping modules. The process is very similar to building blocks. After users install myblockly ，can be viewed directly myblockly use case.



- 2.2 Developed and used based on Python Our products are more friendly to python, and the development of python API library is also increasingly perfect. Through python, the joint angle, coordinates, gripper and other aspects of the robot can be controlled, and there are many choices.After users install the python environment ，can be viewed directly use case。



- 2.3 Developed and used based on C++ C++ is the inheritance of C language. It can carry out procedural programming of C language, object-based programming characterized by abstract data types, and object-oriented programming characterized by inheritance and polymorphism. design. Using C++ language, you can freely develop (coordinate control, angle control, io control, gripper control, etc.) through the C++ dynamic library developed by our company, and control some of the robots that our company has developed.After users install C++ environment ， can be viewed directly use case。

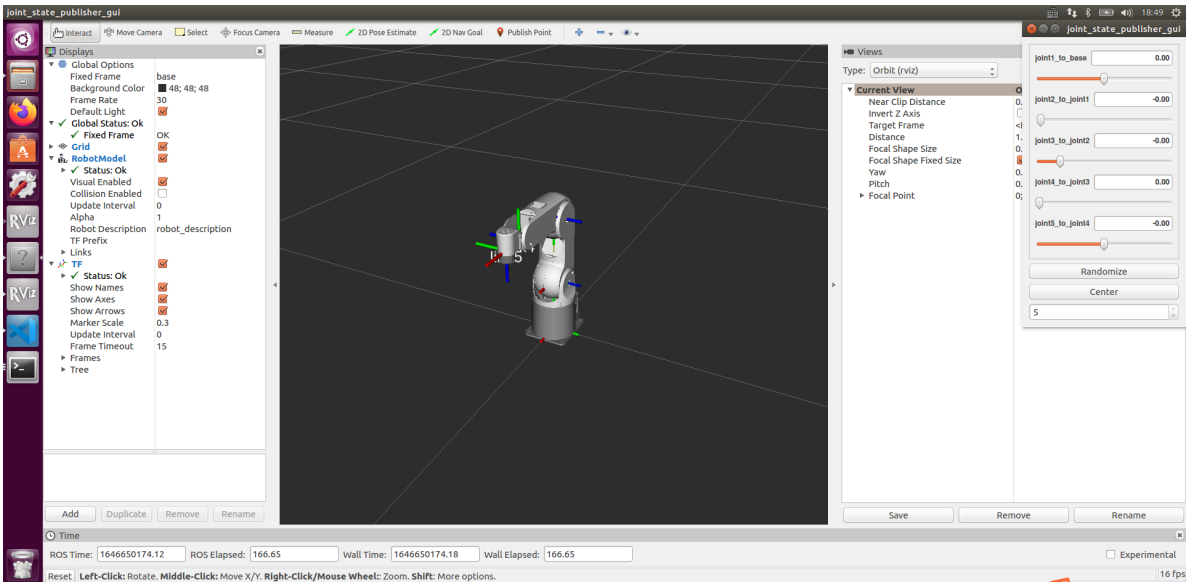
- 2.4 Developed and used based on C# C# is an object-oriented programming language derived from C and C++ released by Microsoft, a high-level programming language running on .NET Framework and .NET Core (completely open source, cross-platform). Using the c# language, you can freely develop (coordinate control, angle control, io control, gripper control, etc.) through the c# dynamic library provided by our company, and control some of the robots that our company has developed.After users install C# environment ， can be viewed directly use case。

- 2.5 Developed and used based on Arduino

  Arduino is an easy-to-use, easy-to-use, open-source electronic prototyping platform that includes hardware (various Arduino-compliant development boards) and software (Arduino IDE and associated development kits). The hardware part (or development board) consists of a microcontroller (MCU), flash memory (Flash), and a set of general-purpose input/output interfaces (GPIO), etc. You can understand it as a microcomputer motherboard. The software part is mainly composed of Arduino IDE on the PC side, related board support packages (BSP) and rich third-party function libraries. Users can easily download the BSP related to the development board you hold and the required function library through the Arduino IDE to write your program.After users install Arduino environment ， can be viewed directly Arduino is easy to use。 aarduino

- 2.6 Developed and used based on JavaScript JavaScript is a scripting language that runs on the client side; it does not need to be compiled, and is interpreted and executed by the js interpreter one by one during the running process. Using the JavaScript language, some of our robots can be controlled through our company's ecological library.After users install JavaScript environment ， can be viewed directly use case。

- 2.7 Developed and used based on ROS ROS is open source and a post-operating system, or secondary operating system, for robot control. Through ROS, we can realize the simulation control of the manipulator in the virtual environment. We will visualize the robotic arm through the rviz platform, and operate our robotic arm in a variety of ways; plan and execute the robotic arm's action path through the moveit platform to achieve the effect of freely controlling the robotic arm.After users Install the ROS development environment ， can be viewed directly use case 和 use of moveit。



3 Firmware update

M5Stack-basic firmware for myPalletizer 260 for M5 and Atom firmware updates need to be updated using myStudio.

Installation and usage tutorial of myStudio:

- myStudio

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

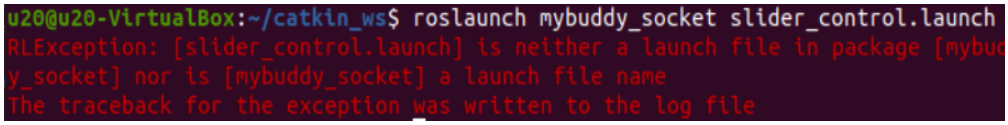
- A: The opencv path in the configuration file is inconsistent with the actual path in the system. Need to use sudo to modify the configuration file (path is `/opt/ros/melodic/share/cv_bridge/cmake/cv_bridgeConfig cmake`), the system actual opencv path in the `/usr/include/` directory.

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.

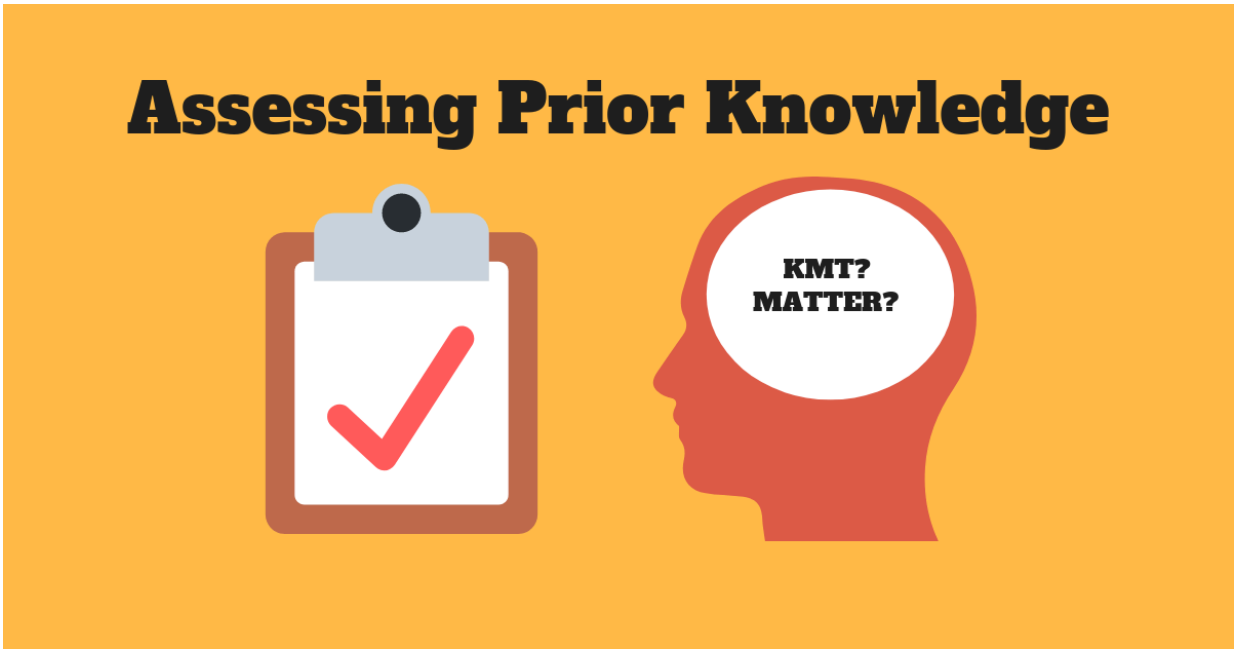
Q2: Does the robotic arms support the development based on Android?

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.

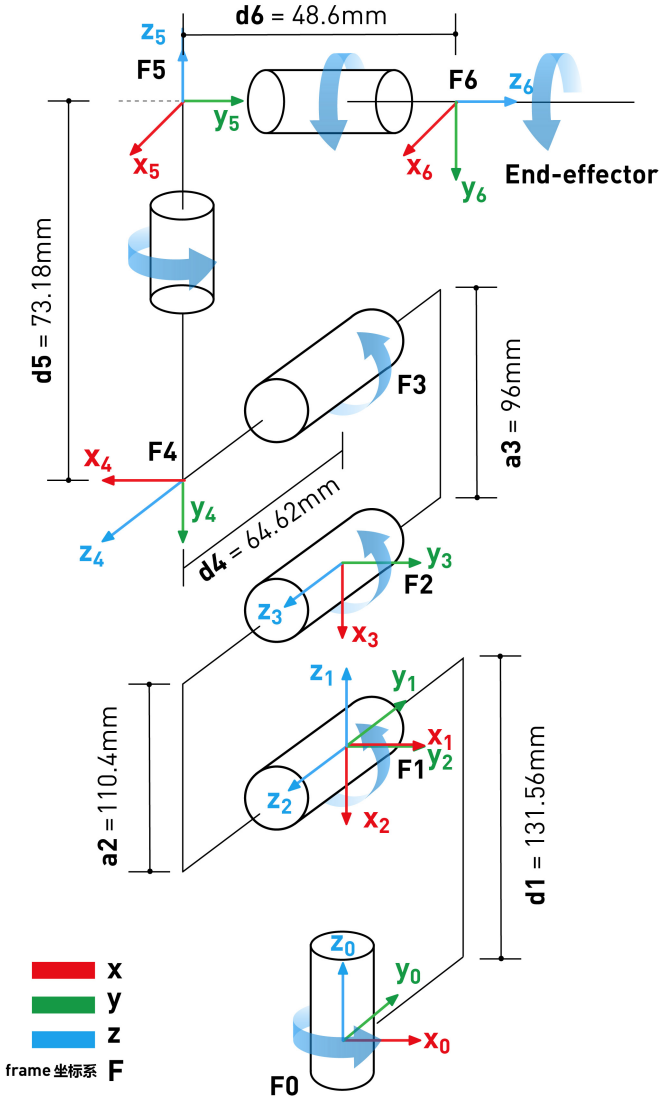
Inverse Kinematic

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.

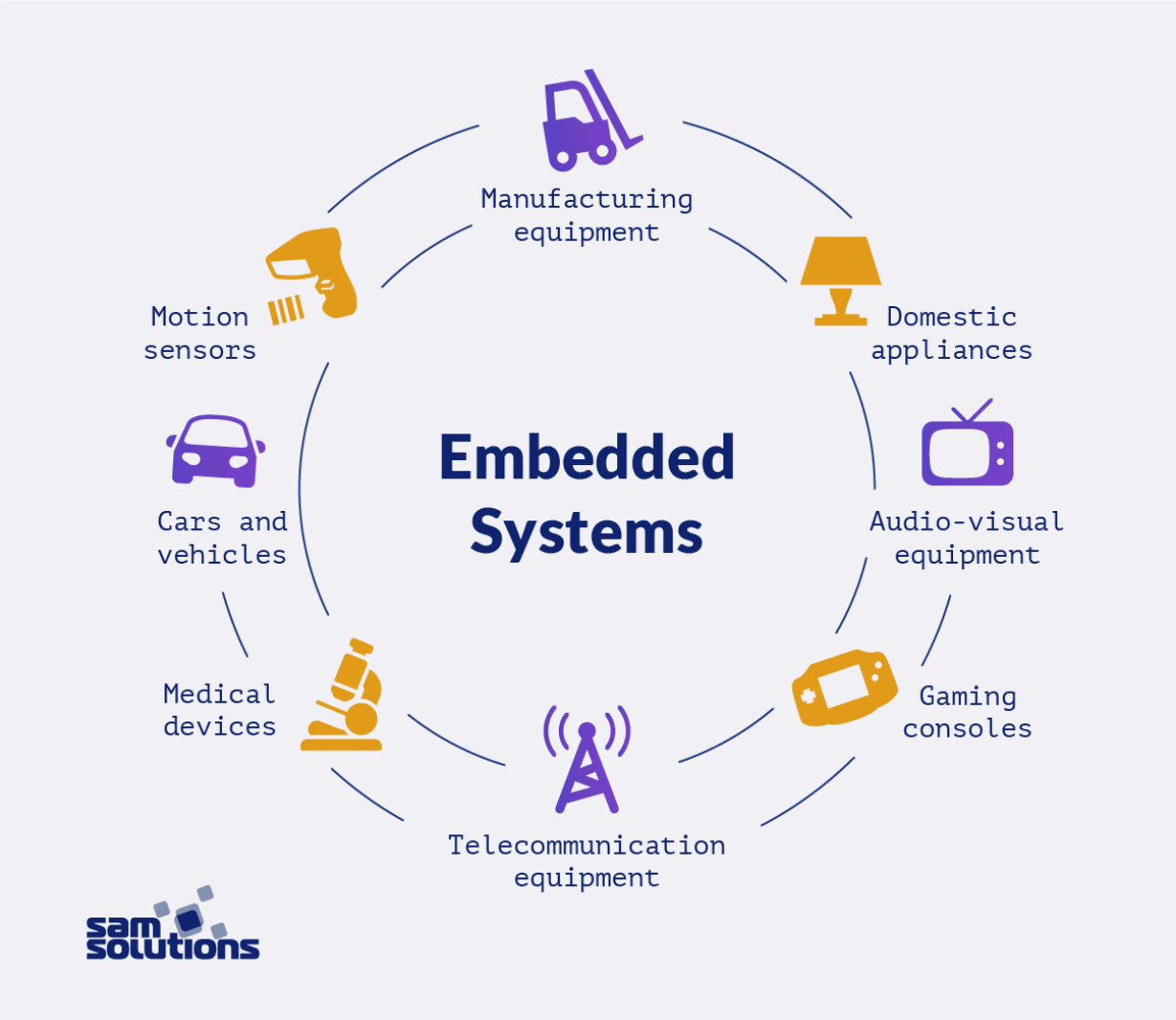
In the link coordinate system, if the link coordinate system is fixedly attached to the link as described above, the link parameters can be defined as follows:

- 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.

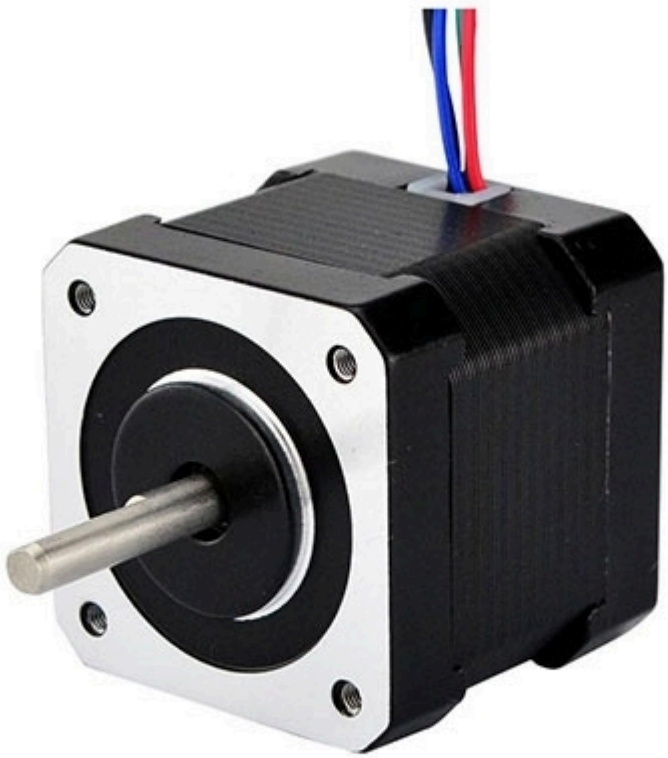
Microprocessor interconnects ALU, counter, register and control parts through an internal bus, and connects the external memory, input and output interface circuit through an external bus. External bus, also known as system bus, is divided into data bus DB, address bus AB and control bus CB. It can be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

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

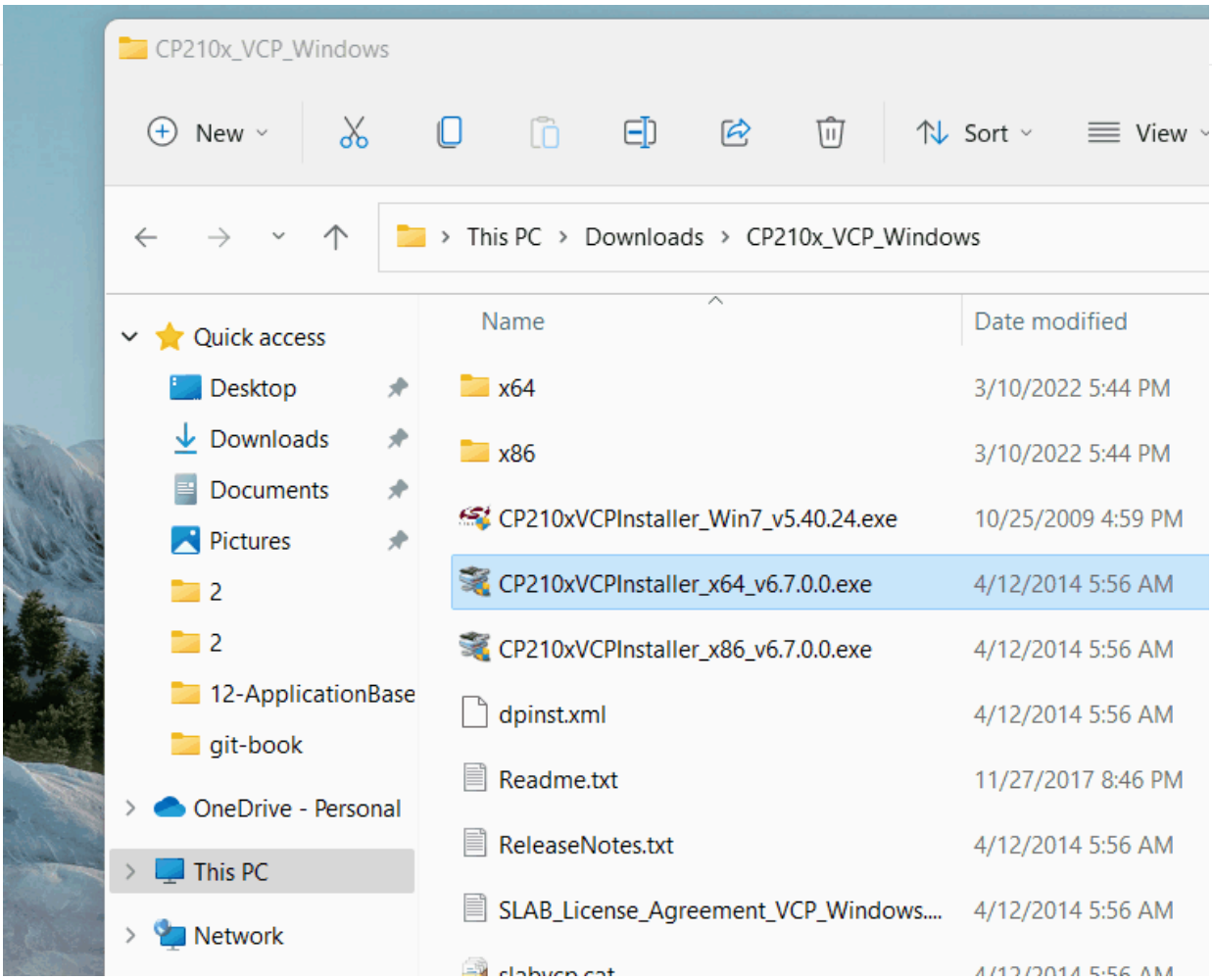
  - **Windows10**
  - **MacOS**
  - **Linux**

  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.**

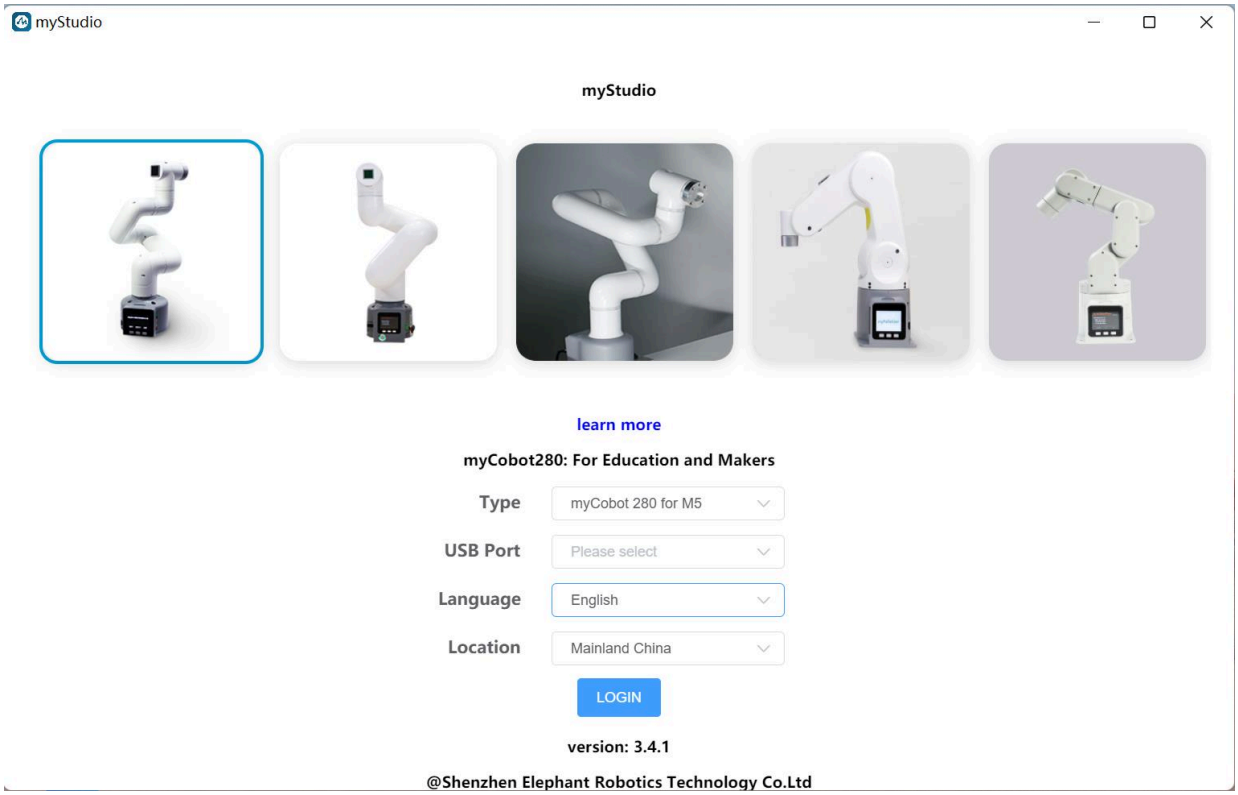
If you have any questions or suggestions about the contents of this manual, please log in the official website of Elephant Robot and submit relevant information：

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

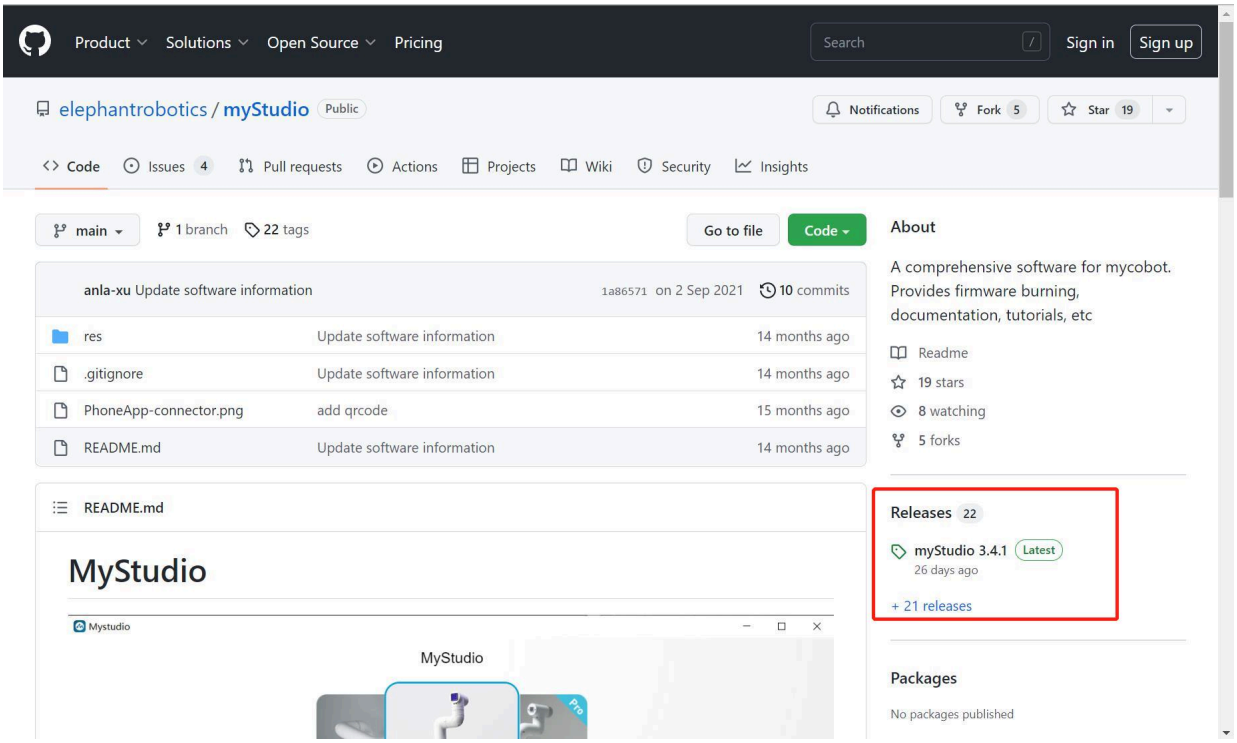
1.1 Download and Installation

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

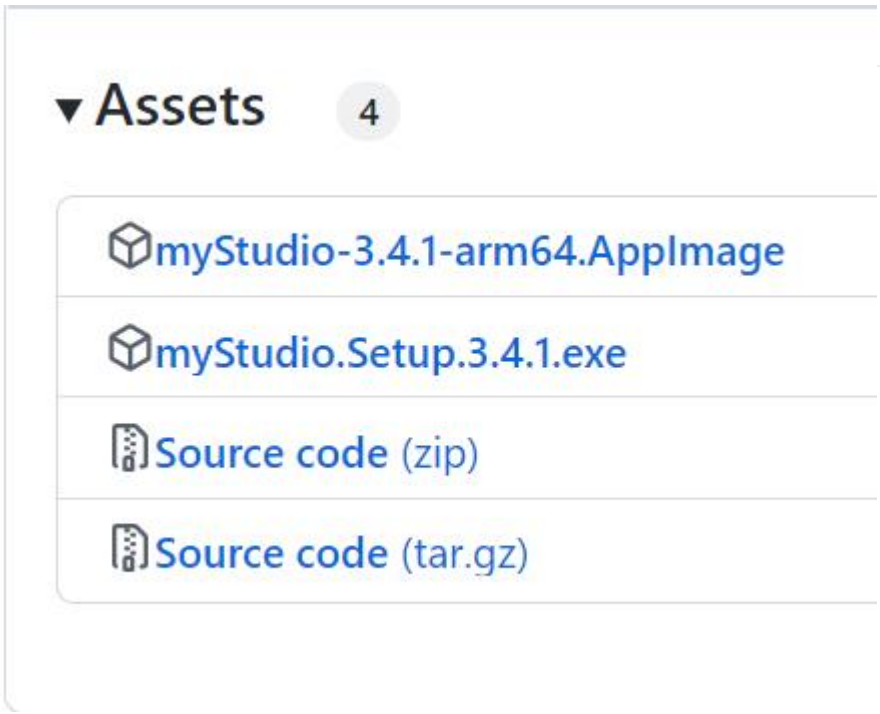
address for downloading:

1. GitHub

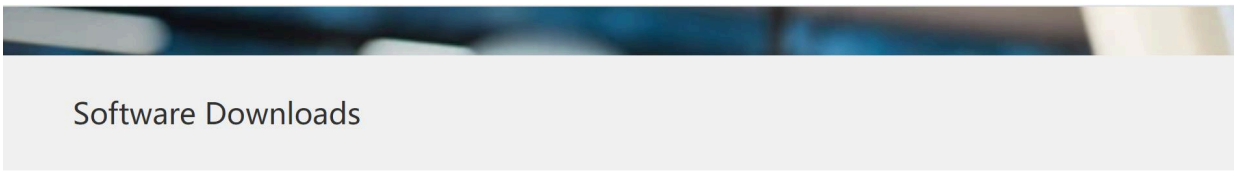
- Click on `myStudio` at the right side and download the version corresponding to your PC. **Do not install myStudio into files with space directory.**



- 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

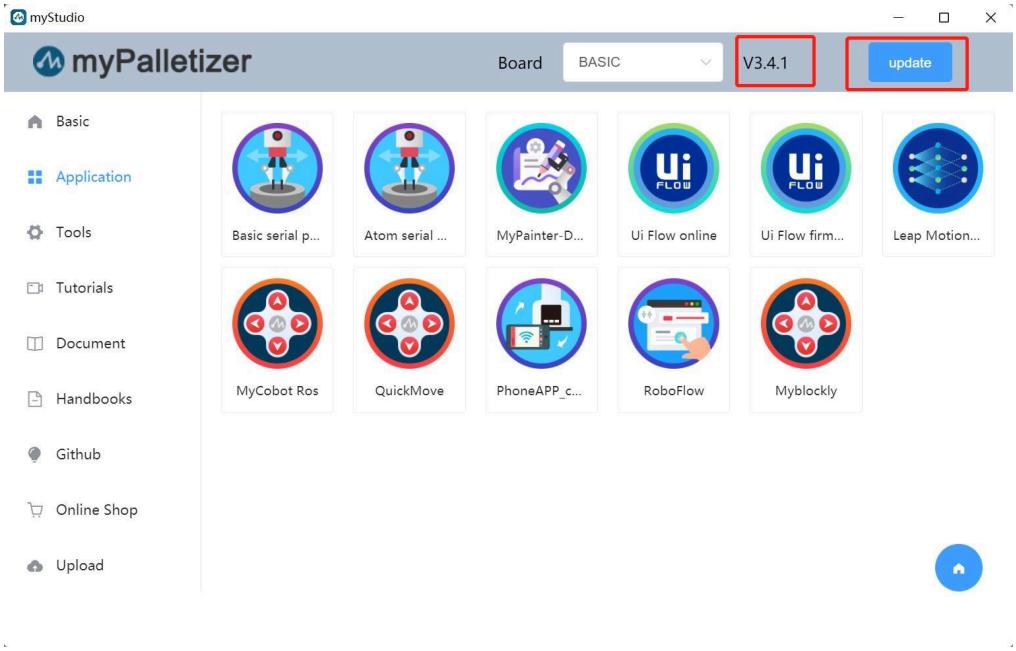


myStudio

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.

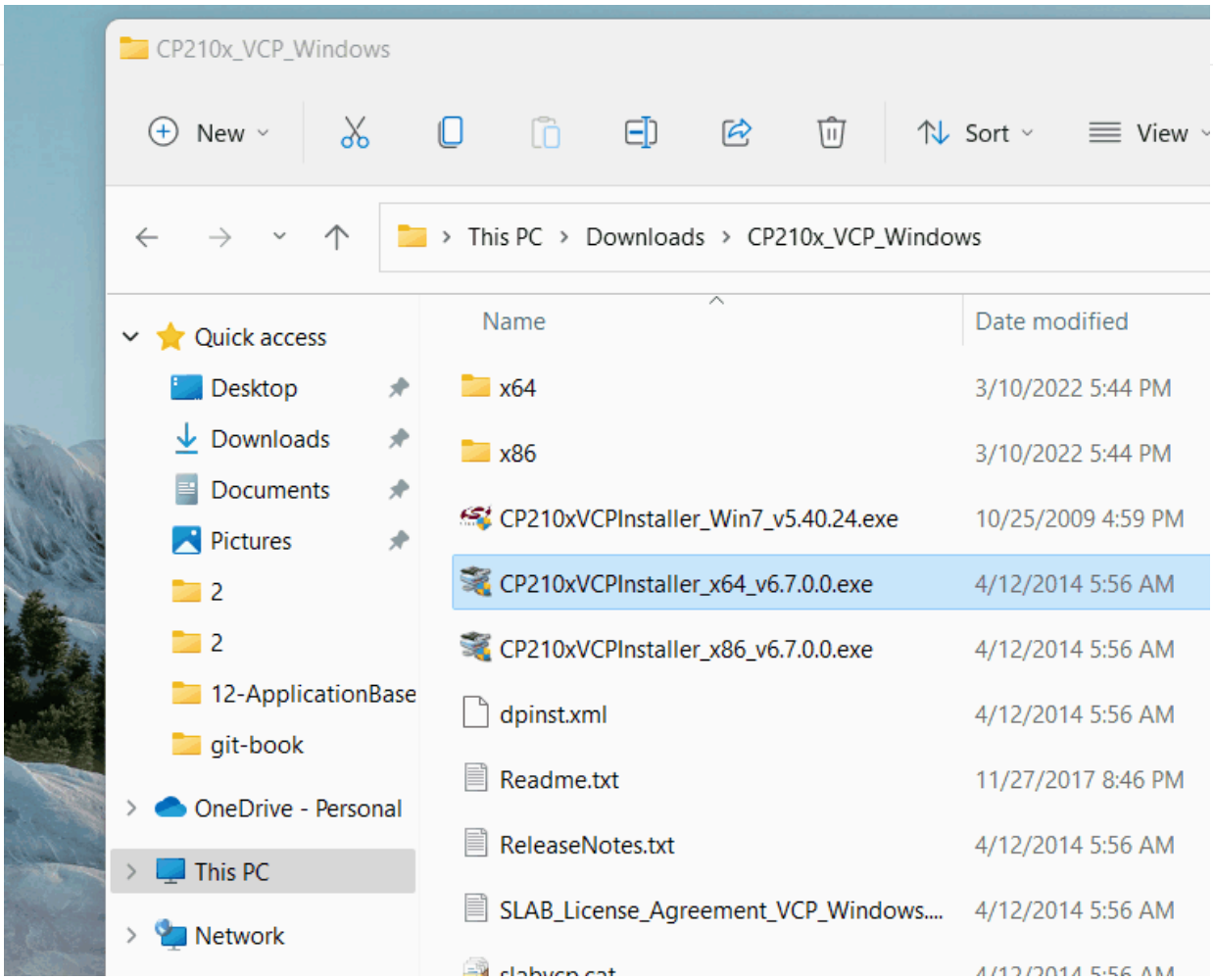


1.2 Serial Port Driver Installation

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**

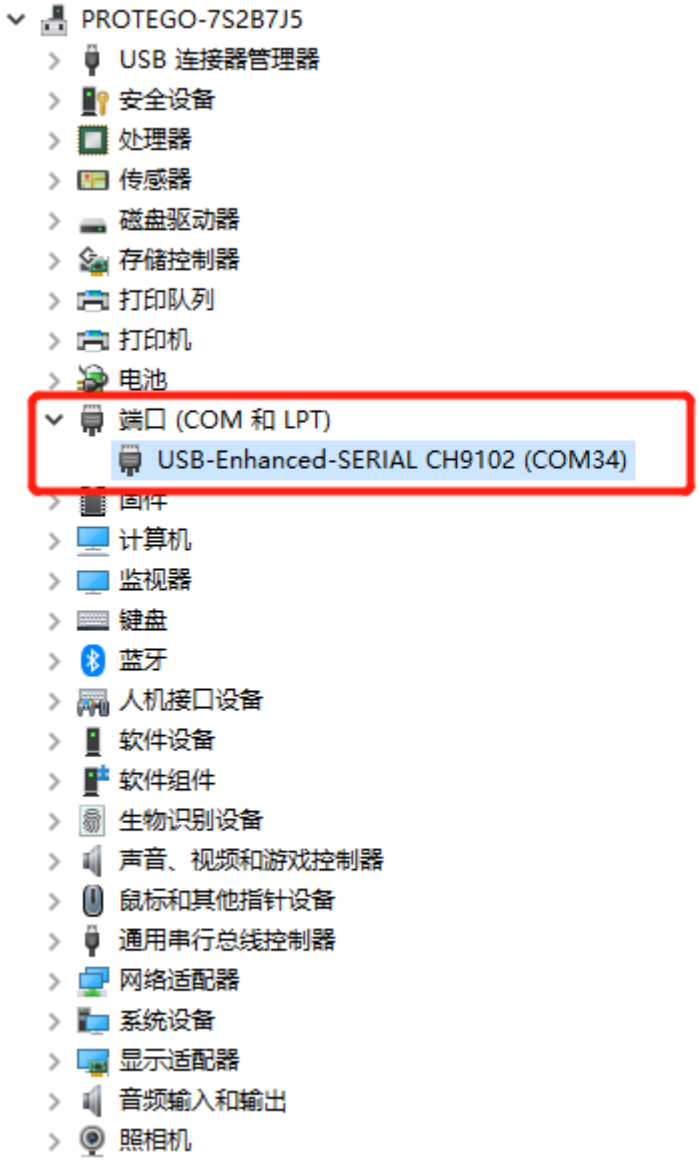
> **Notice:**
>
> - An error may be reported during installation of CH9102_VCP_SER_MacOS. However, the installation already completes. Just ignore the error.
> - Make sure that settings of the system \"Preferred settings -> Security and privacy ->General\" is correct and allow the user to get it from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

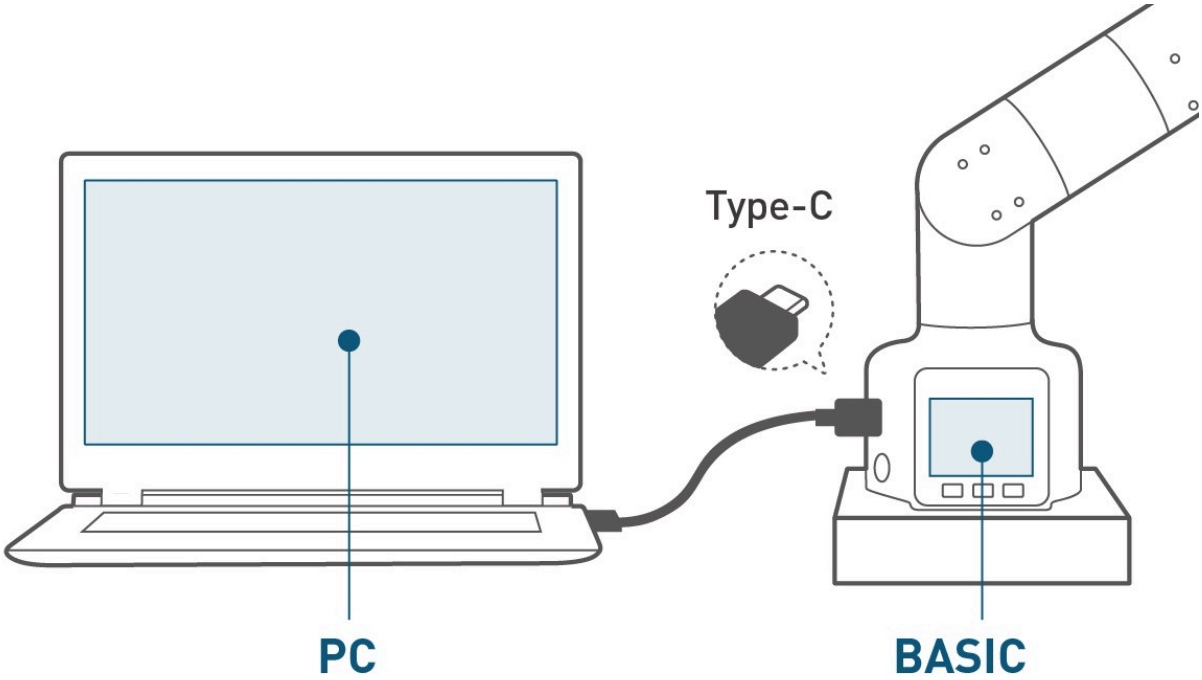
2 Burning and Updating Firmwares

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

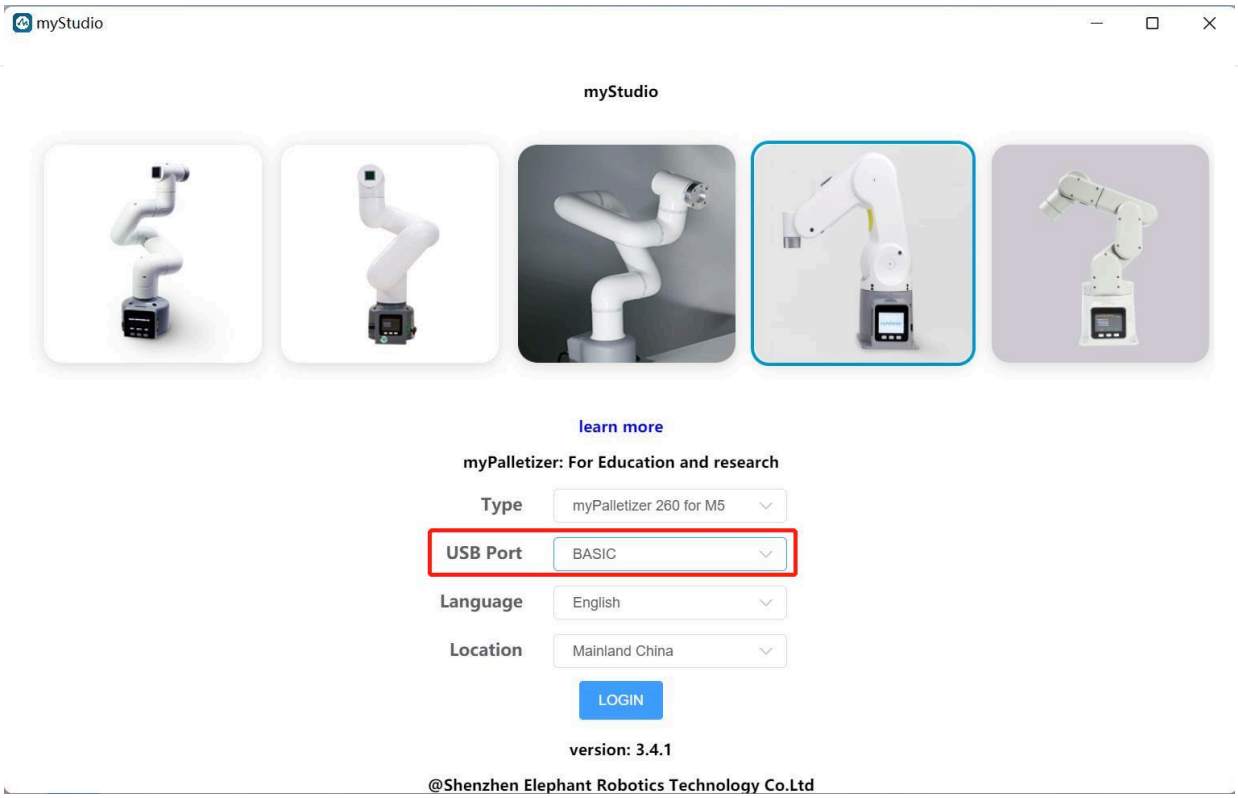
2.1 Burning M5Stack-basic Firmwares

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

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

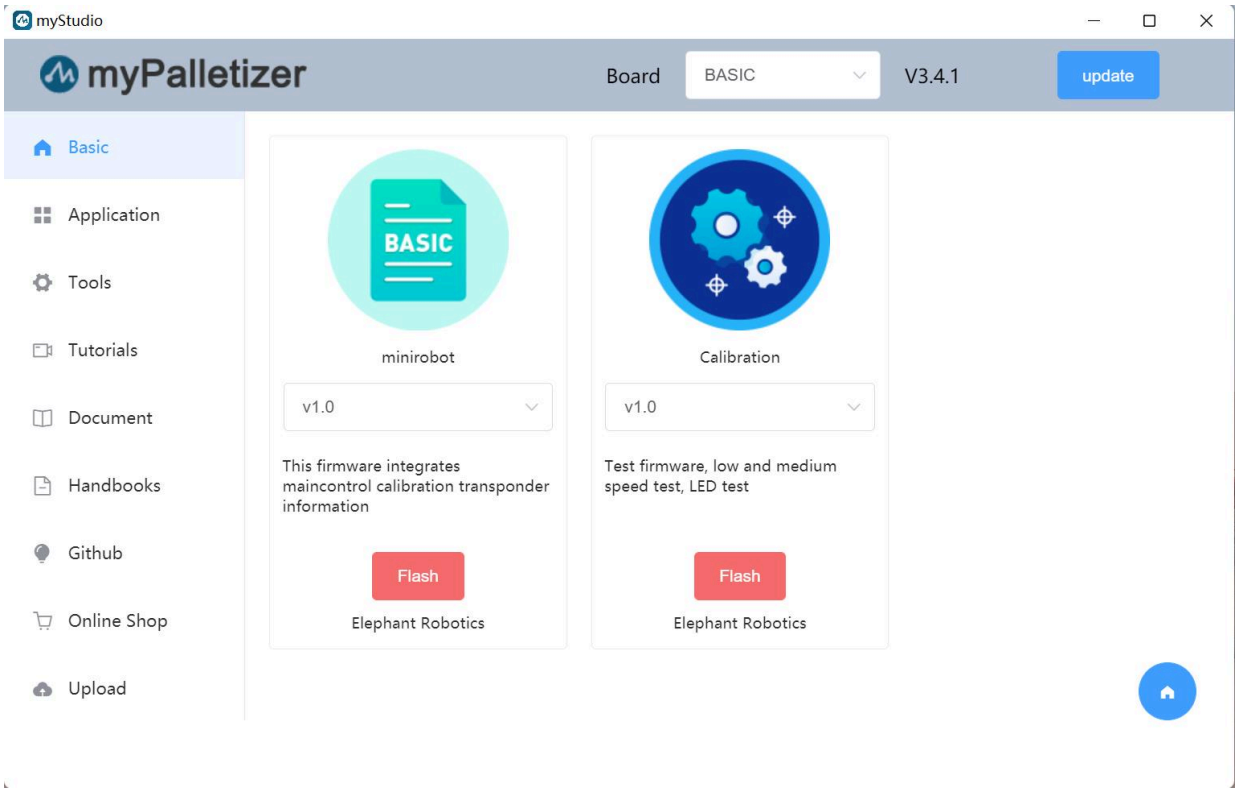


Step 2: Select `USB Port`. After connecting with PC, `USB Port` shows `M5Stack-basic`. The figure below takes myPalletizer 260 M5 as an example.



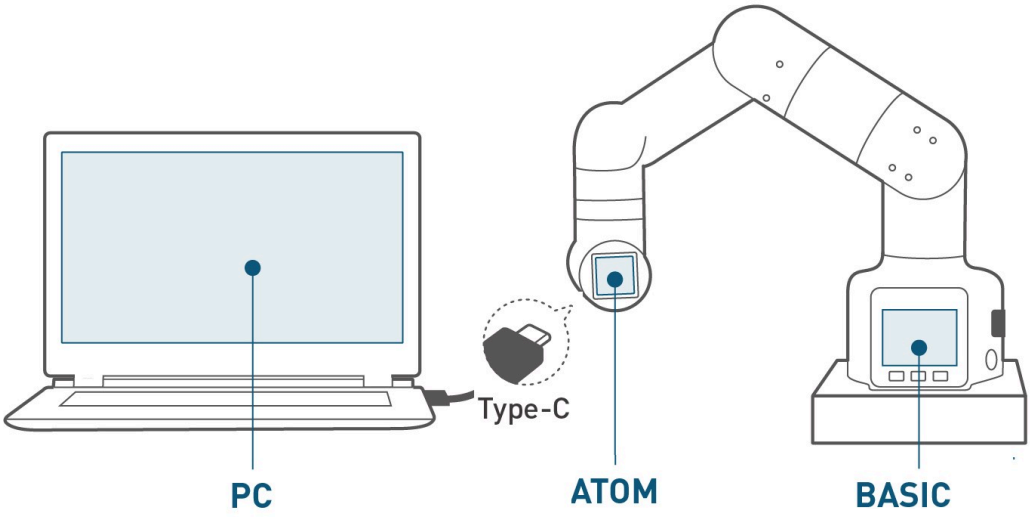
- Click on `LOGIN` -> `M5Stack-basic`, and then you can burn firmwares.

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

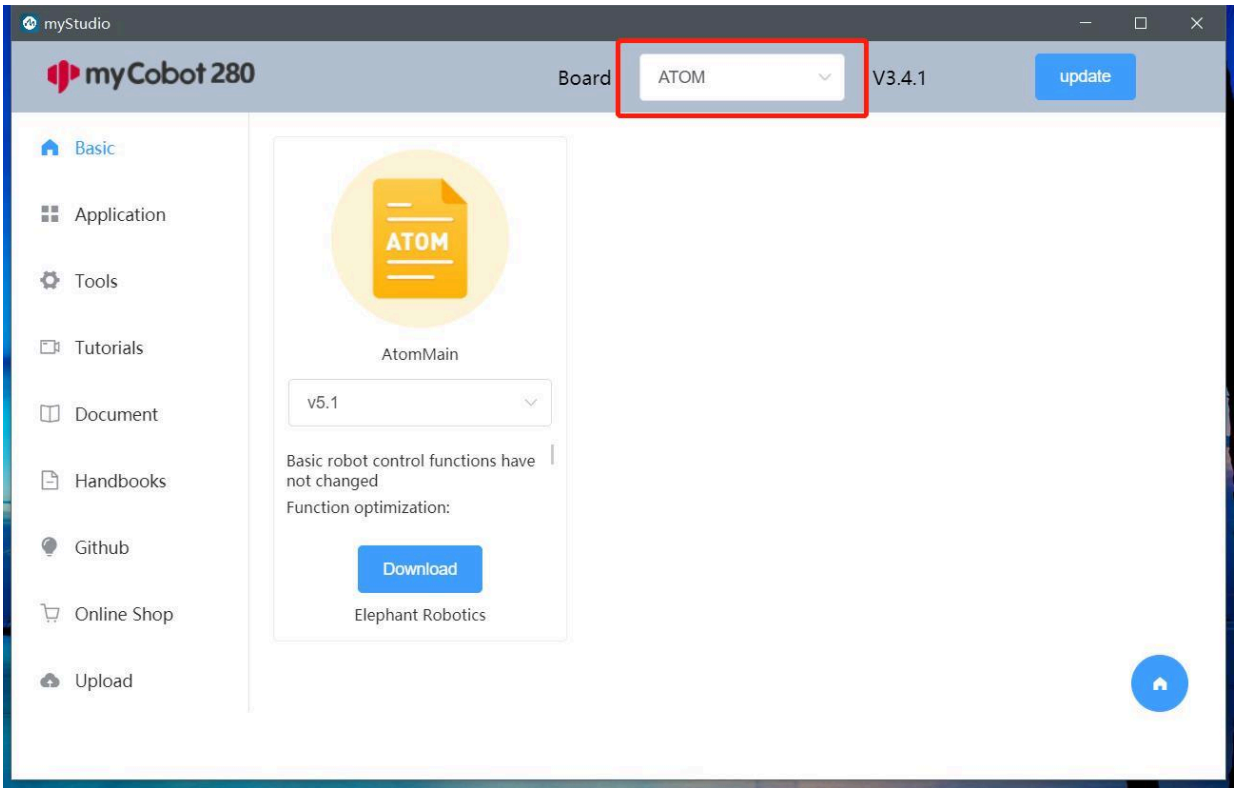


2.2 Burning Atom Firmware

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.

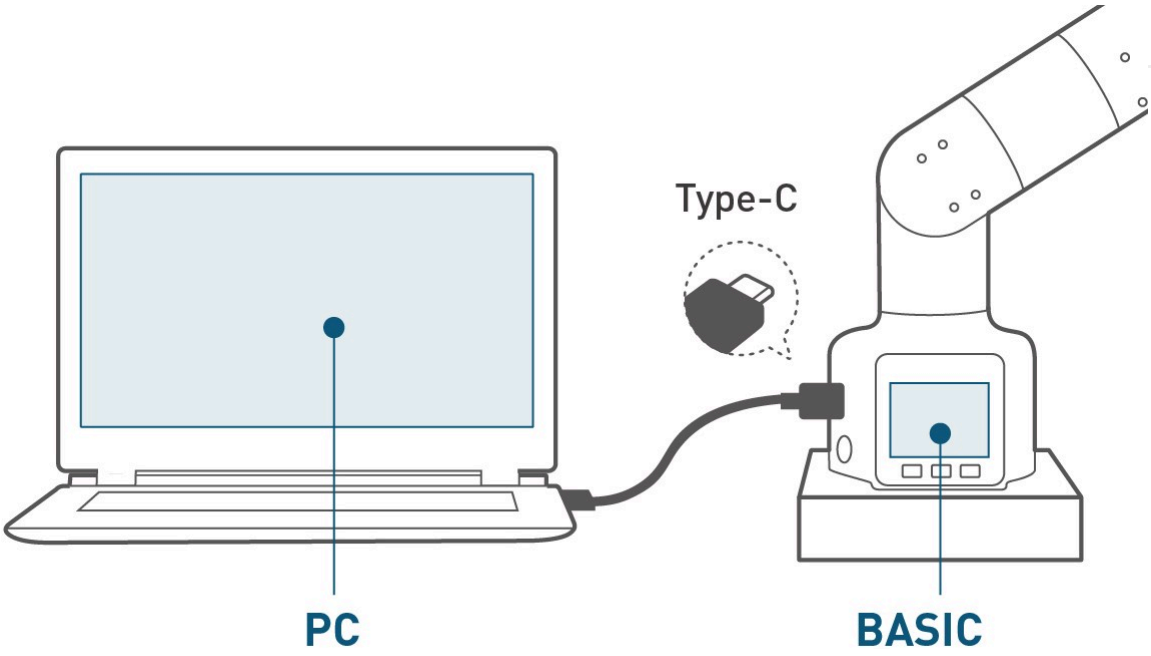


2.3 Firmware Burning for myCobot 320 series

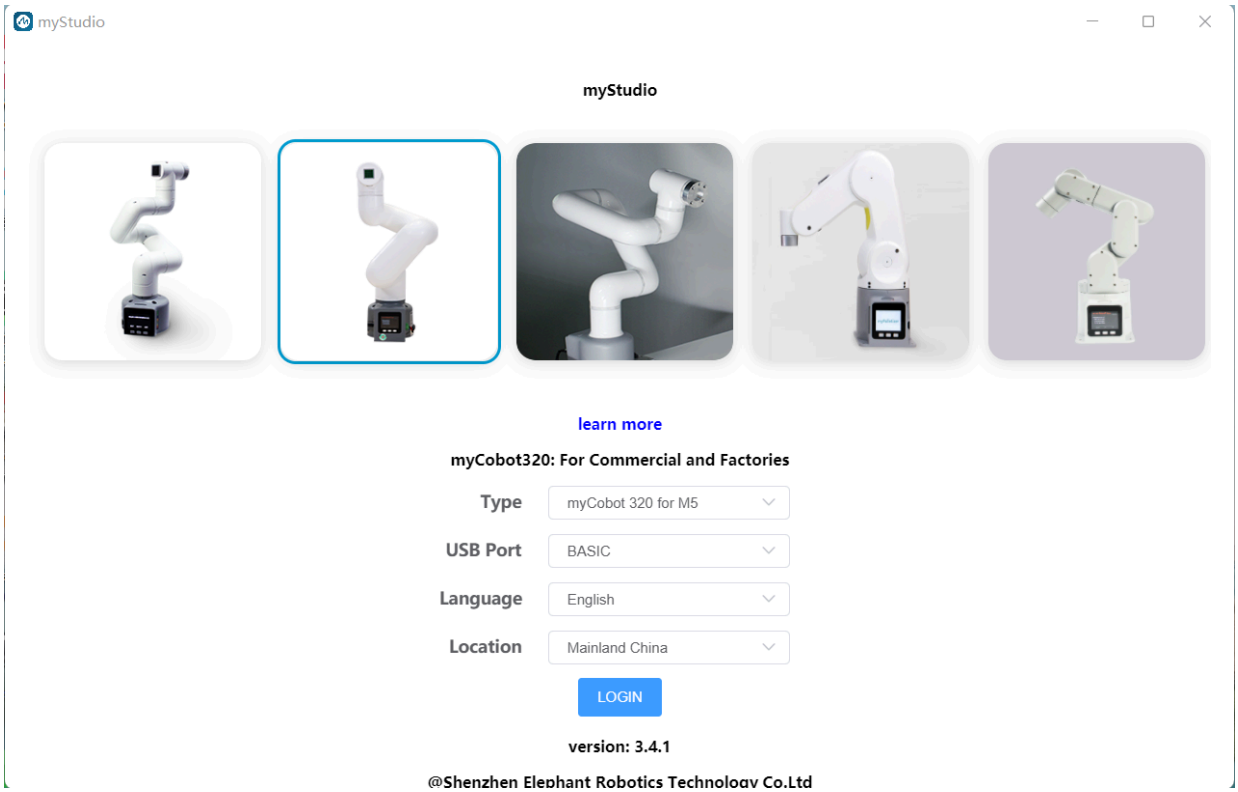
M5Stack-basic and picoMain are required to be burnt on myCobot 320 series. It should be noted that the two firmwares are burnt by different type-c interfaces.

1.Burning M5Stack-basic: miniRobot

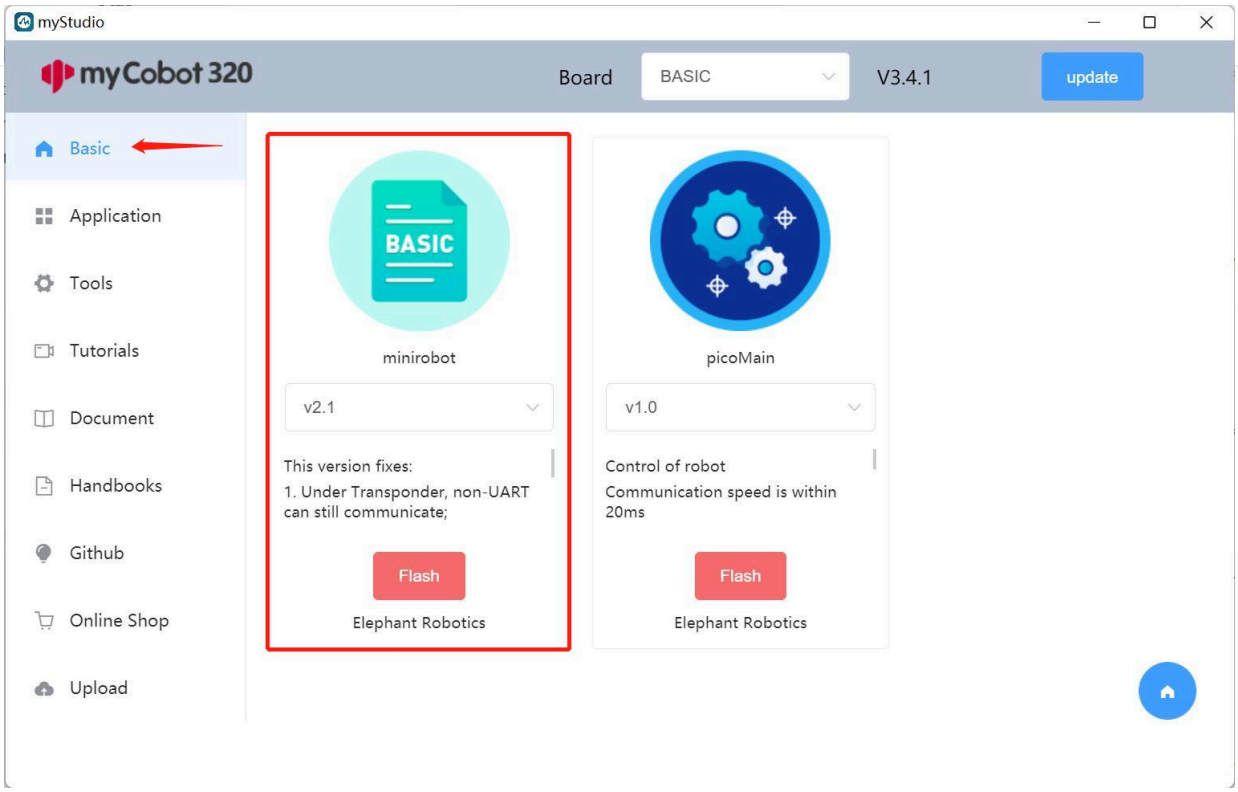
Step 1: Connect M5Stack-basic and PC.



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



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



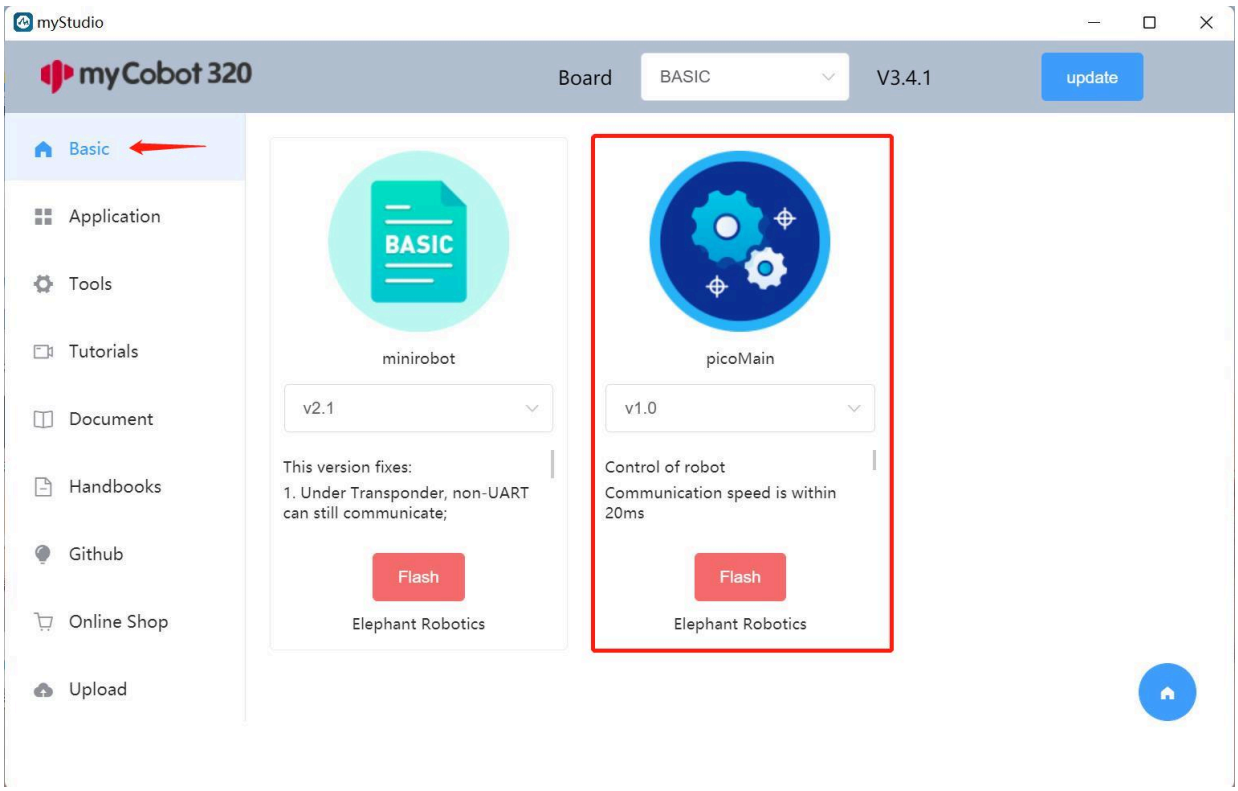
2.Burning Pico Firmware: picoMain

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.



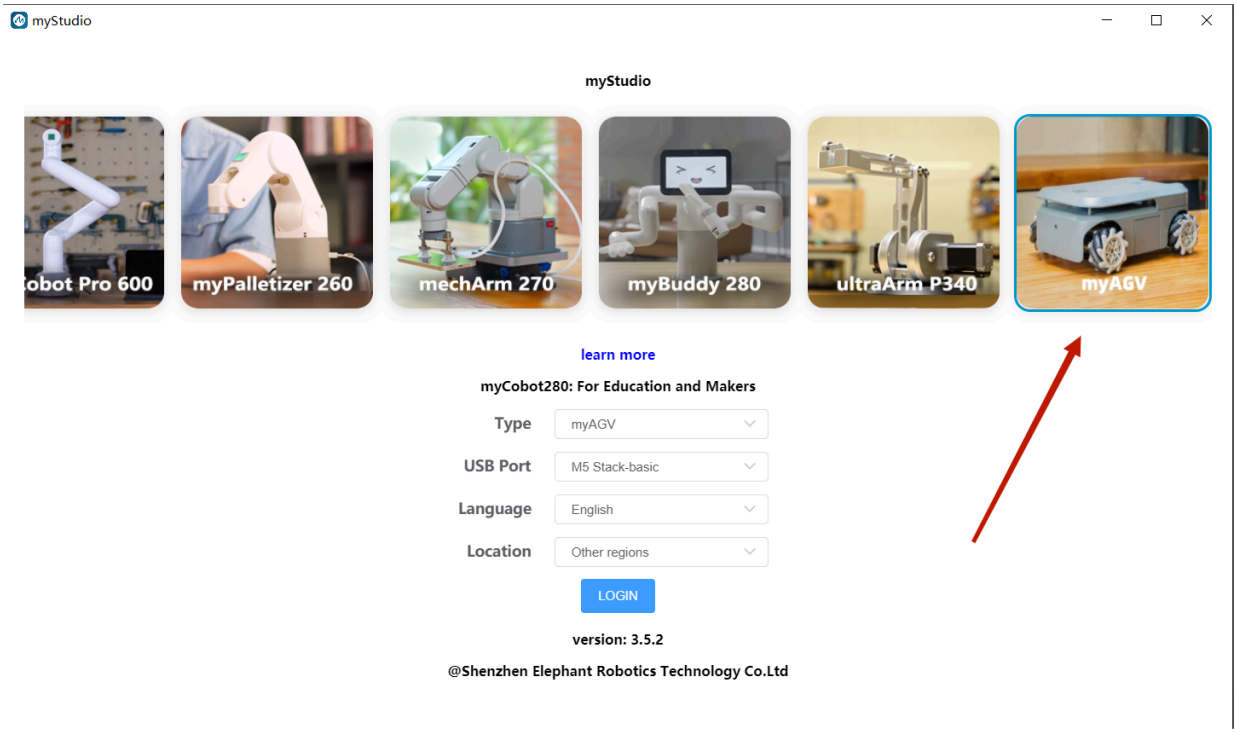
2.4 myAGV Firmware burning

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.

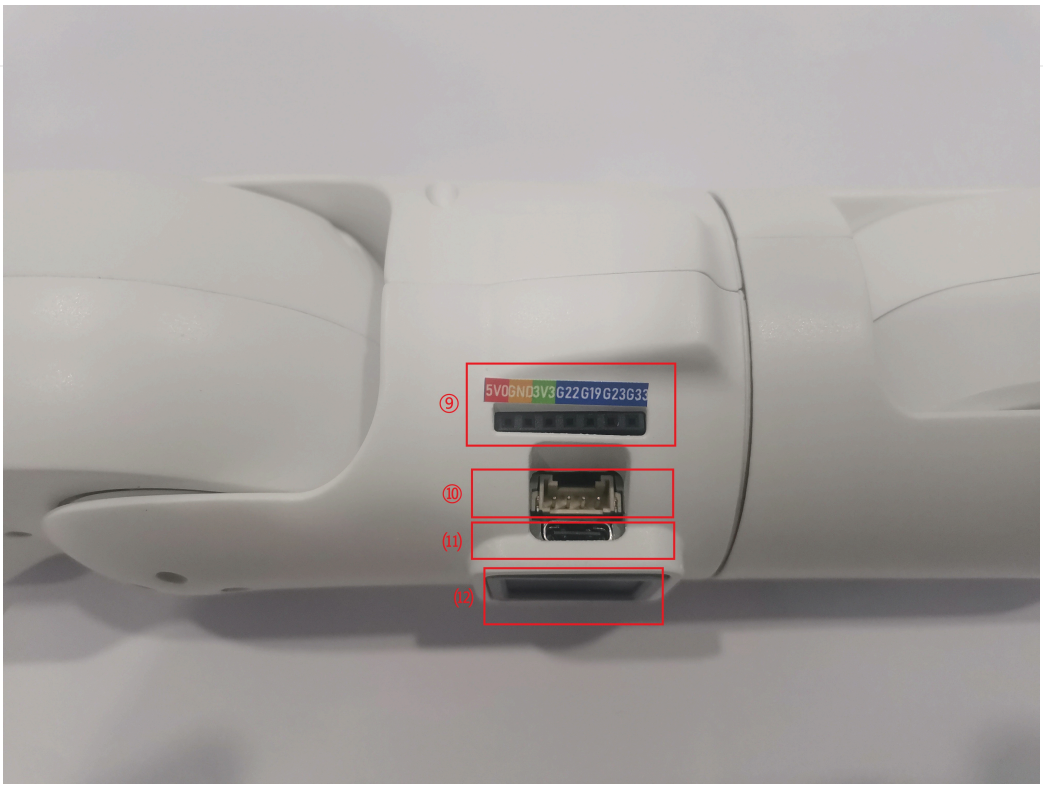


Step 3: After logging in, click 'Basic' and select picoMain to burn.

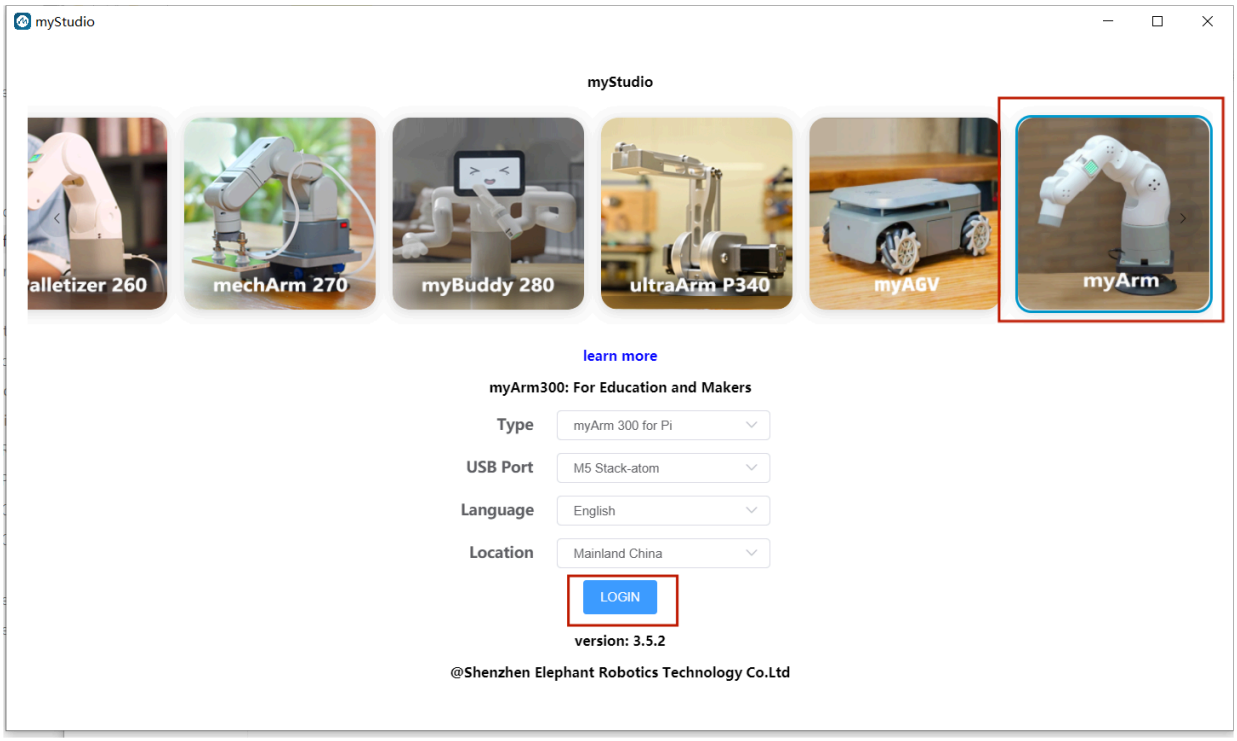
2.5 myArm Firmware burning

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.



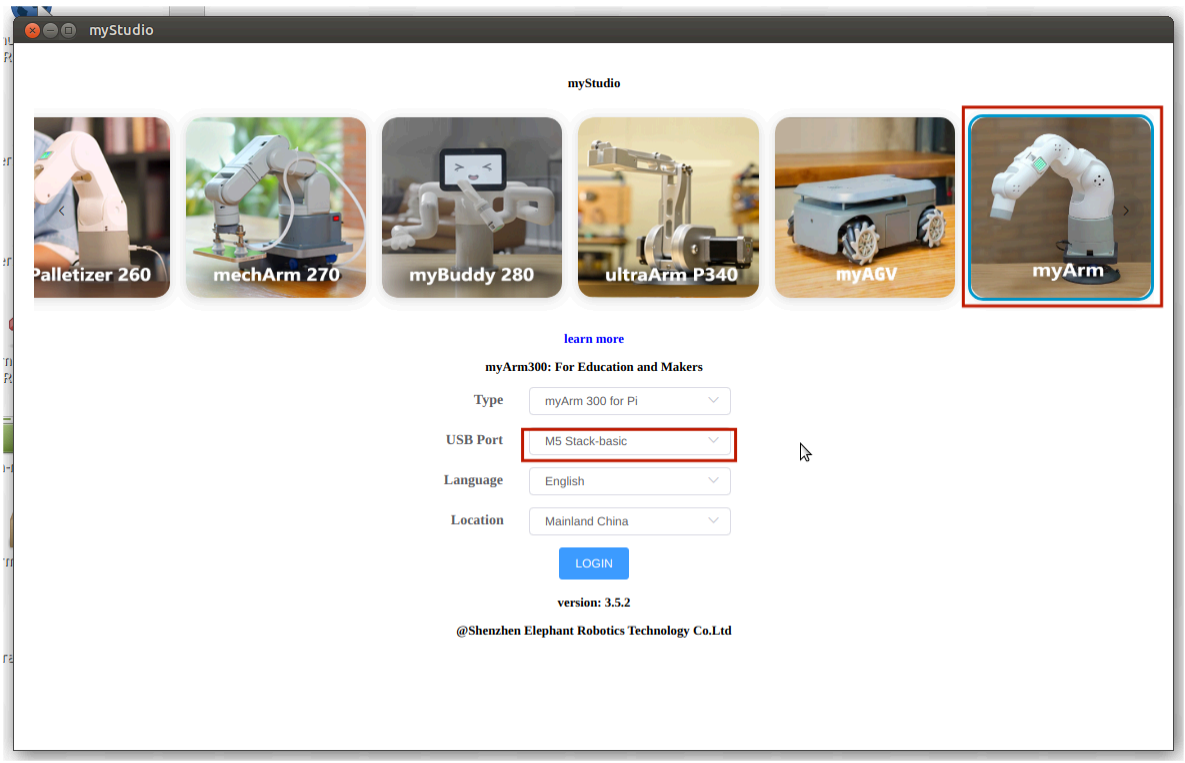
Step 3: After login, click 'Basic', select AtomMain, click download, and then click the burn button to burn.

2.Burning Pico Firmware: picoMain

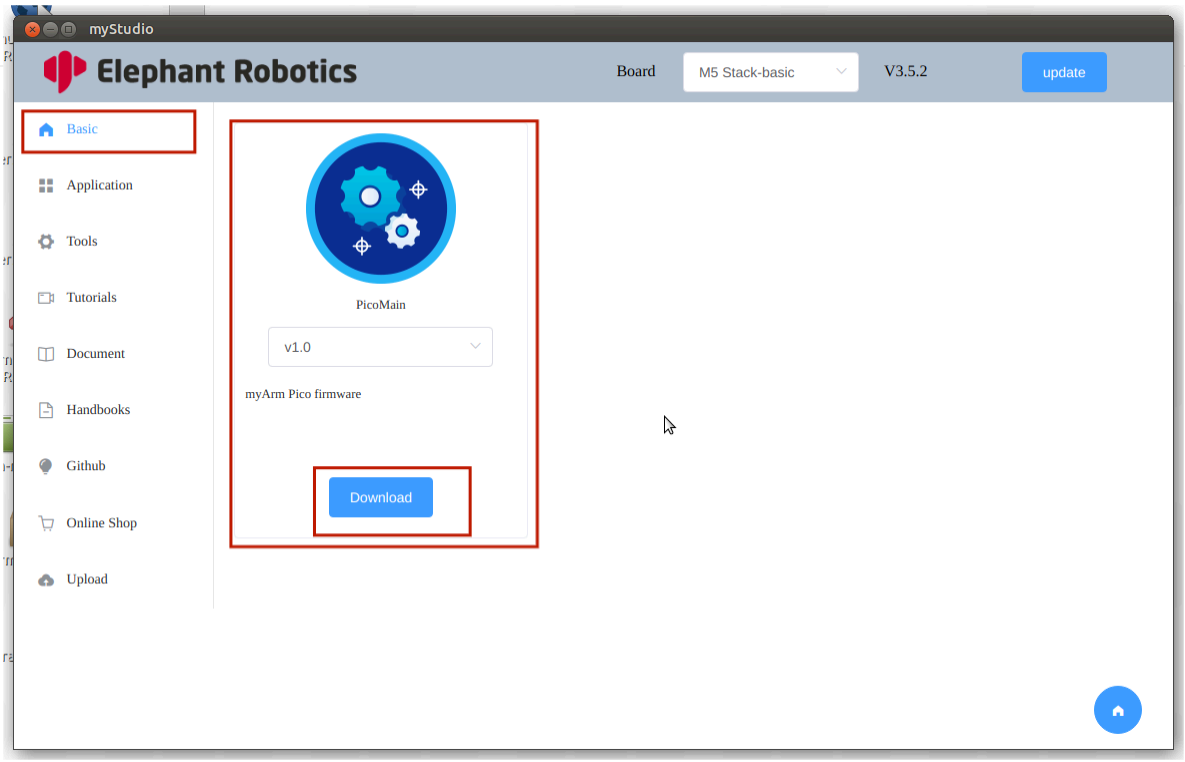
Step 1: Turn on the computer and connect the screen.

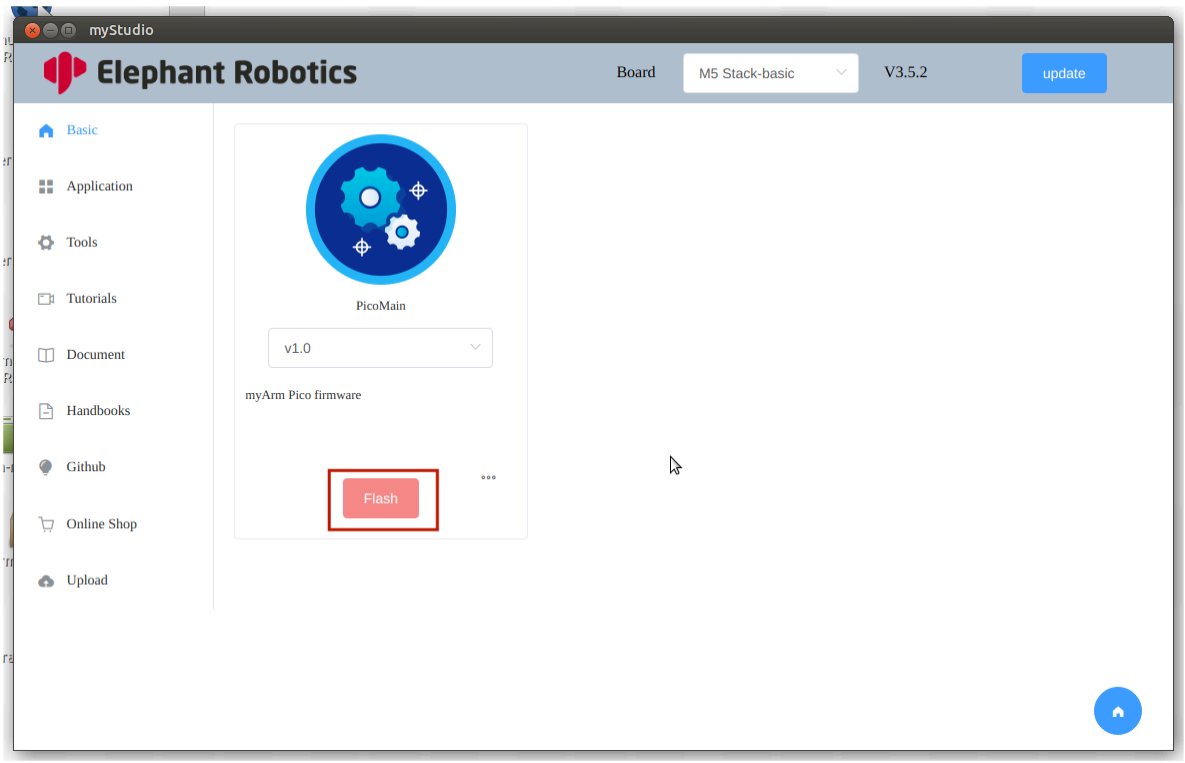
For detailed tutorials, please refer to：initial-setup

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

Factory firmware introduction

We divide the devices into two categories: **micro-controller** and **micro-CPU.** (Only introduce myCobot series, myPalletizer series, mechArm series products)

- **Micro-controller devices**
  - **myCobot 280-M5**
  - **myCobot 320-M5**
  - **myPalletizer 260- M5**
  - **mechArm 270-M5**
- **Micro-CPU devices**
  - **myCobot 280-Pi**
  - **myCobot 320-Pi**
  - **mechArm 270-Pi**

The difference between the micro-CPU and the micro-controller mainly focuses on three aspects: hardware structure, application fields and instruction set characteristics:

- Hardware structure. The micro-CPU is a single-chip CPU, while the micro-controller integrates a CPU and other circuits in an integrated circuit chip to form a complete microcomputer system. Beside the CPU, the micro-controller includes a RAM, ROM, serial interface, parallel interface, timer, and interrupt scheduling circuit.
- Application fields. The micro-CPU is usually used as a CPU in a microcomputer system, and they are designed for such application.This is just the advantage of the micro-CPU. However, the micro-controller is usually used in control-oriented applications, where system design seeks miniaturize and minimizes the number of components. The micro-controller is suitable for those applications in which control of input/output devices is implemented with very few components, while the micro-CPU is suitable for information processing in a computer system.
- Instruction set characteristics. Due to different applications, the instruction set of the micro-controller is also different from the one of the micro-CPU. The instruction set of the micro-CPU enhances the processing function, making it have a powerful addressing mode and the instructions suitable for manipulating large amounts of data. The instructions of the micro-CPU allow operation of nibbles, bytes, words, and even double words. By using address pointers and offsets, the micro-CPU provides an addressing mode that allows access of large amounts of data. The auto-increment and auto-decrement modes make it easy to access data in the units of bytes, words, or double words.

Here you can gradually familiarize yourself with the robot arm through simple routines.

1 Factory firmware for micro-controller devices - miniRoboFlow

miniRoboFlow has four main functions:

- **Maincontrol**
  - Robot drag teaching: The operator can drag robot joints directly to make them do ideal postures, and then save the actions in the robot through **button operation**. The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.
- **Calibration**

    - Calibrating the robot arm is the precondition for precise control of the robot arm, and setting joint zero and initializing the potential of the motor are basic jobs for subsequent advanced development.
- **Transponder**
    - Communication timeliness is vital to the micro-controller robot arm.For such arm, we often send control instructions to **M5Stack-basic** at the bottom. Through communication forwarding, the end effector analyzes the instructions and then implements target actions. At present, **myCobot280** has three methods of communication: **serial port, Bluetooth, and WIFI**.
- **Information**
    - Link test is a detection function that uses the motor in the robot arm and the connection state of **Atom**. The function allows the user to remove equipment faults easily. During the link test, the connection state of the equipment for the robot arm, including the **connection of the servo** and the **communication state of Atom** can be seen. In micro-controller devices, the versions of their current firmwares are shown on M5Stack-basic.

2 Factory firmware for micro-CPU devices- python demo

At present, the python demos that have been opened to micro-CPU devices are:

- **drag_trial_teaching**
    - drag teachingThe operator can drag robot joints directly to make them do ideal postures, and then save the actions in the robot through **button operation**. The cobot is a system that has this function earlier. This kind of teaching avoids various disadvantages of traditional teaching, so it is a prospective technology for robot applications.
- **rasp_mycobot_test_gui**
    - This python demo is a test tool for micro-CPU devices, including functions **robot arm calibration** and **connection detection**.
    - **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.
    - **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.

Drag teaching

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

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

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

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

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

Drag-and-Record Demonstration

1 Applicable Robotic Arms

- myCobot 280 M5
- myCobot 320 M5
- myPalletizer 260 M5
- mechArm 270 M5

2 Steps to Operate the Arms

Step1: Burn the latest **atomMain** for **Atom.**

Step 2: Burn**minirobot**for**M5Stack-basic**, and press **Maincontrol**.

Step 3: Press **Record**.

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

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

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

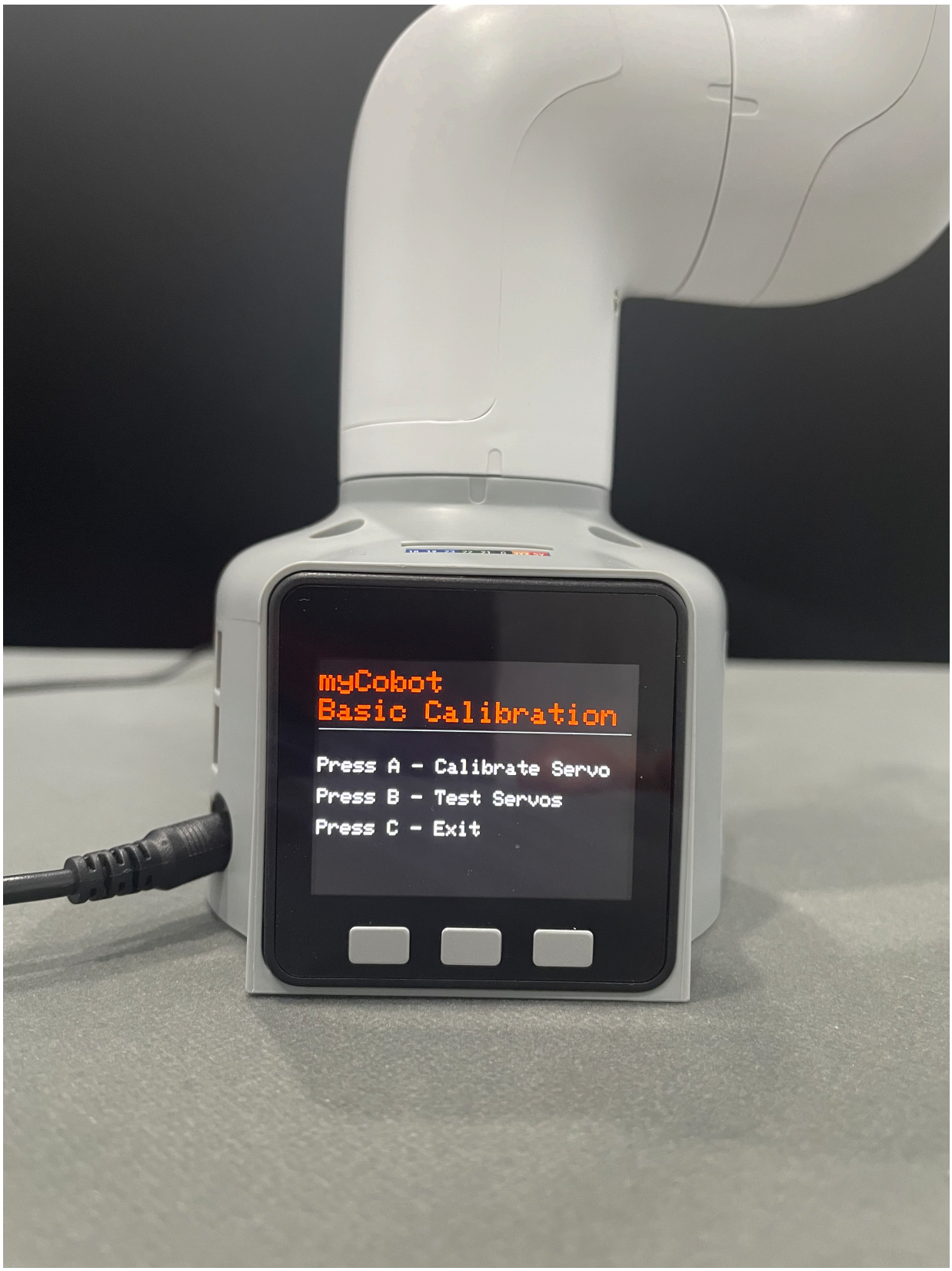
> **Notice:**
>
> **Pause**: pause the movement
>
> **Stop**: stop the movement
>
> **Play**: resume the movement

3 Tutorial Video

Address: https://www.bilibili.com/video/BV16t4y167vw/

Robot arm calibration

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



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

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

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

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

Calibration

1Applicable Robotic Arms

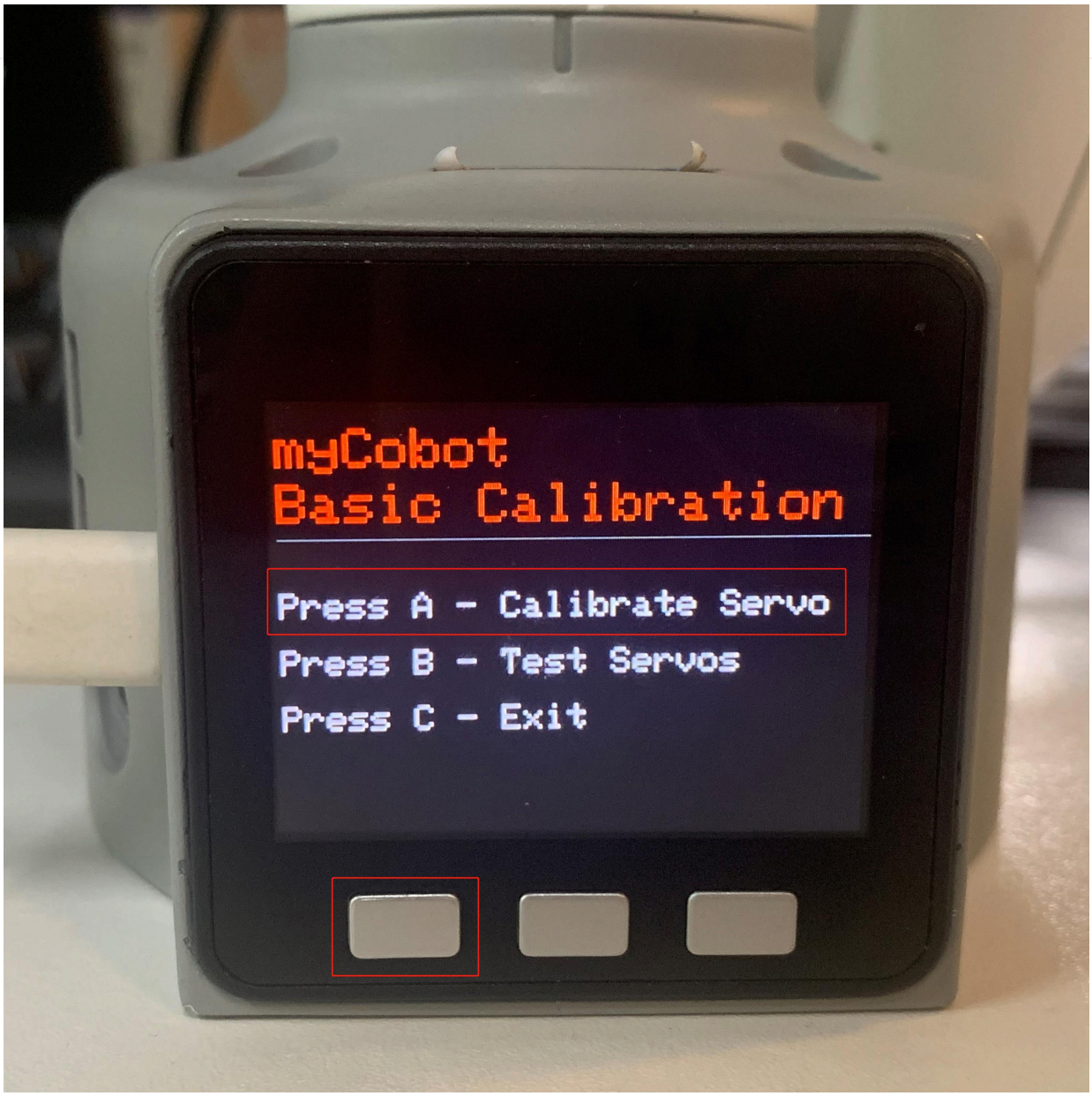
- myCobot 280 M5
- myCobot 320 M5
- myPalletizer 260 M5
- mechArm 270 M5

2 Steps to Operate the Arm

Step 1: Burn the latest **atomMain** for **Atom**.

Step 2: Burn the **minirobot** for **M5Stack-basic**, and press **Calibration**.

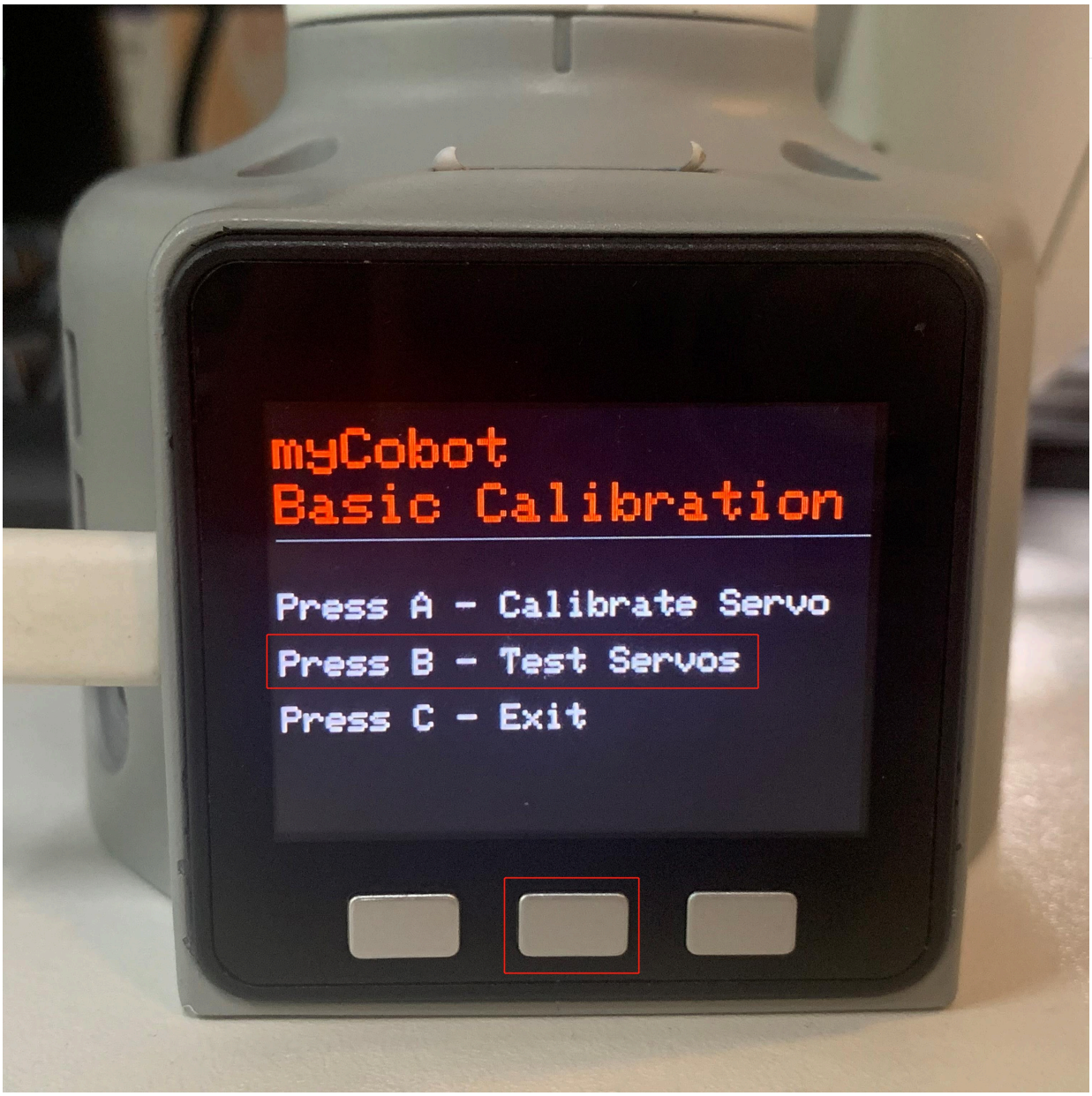
Step 2: Press **A** to start robot arm calibration.



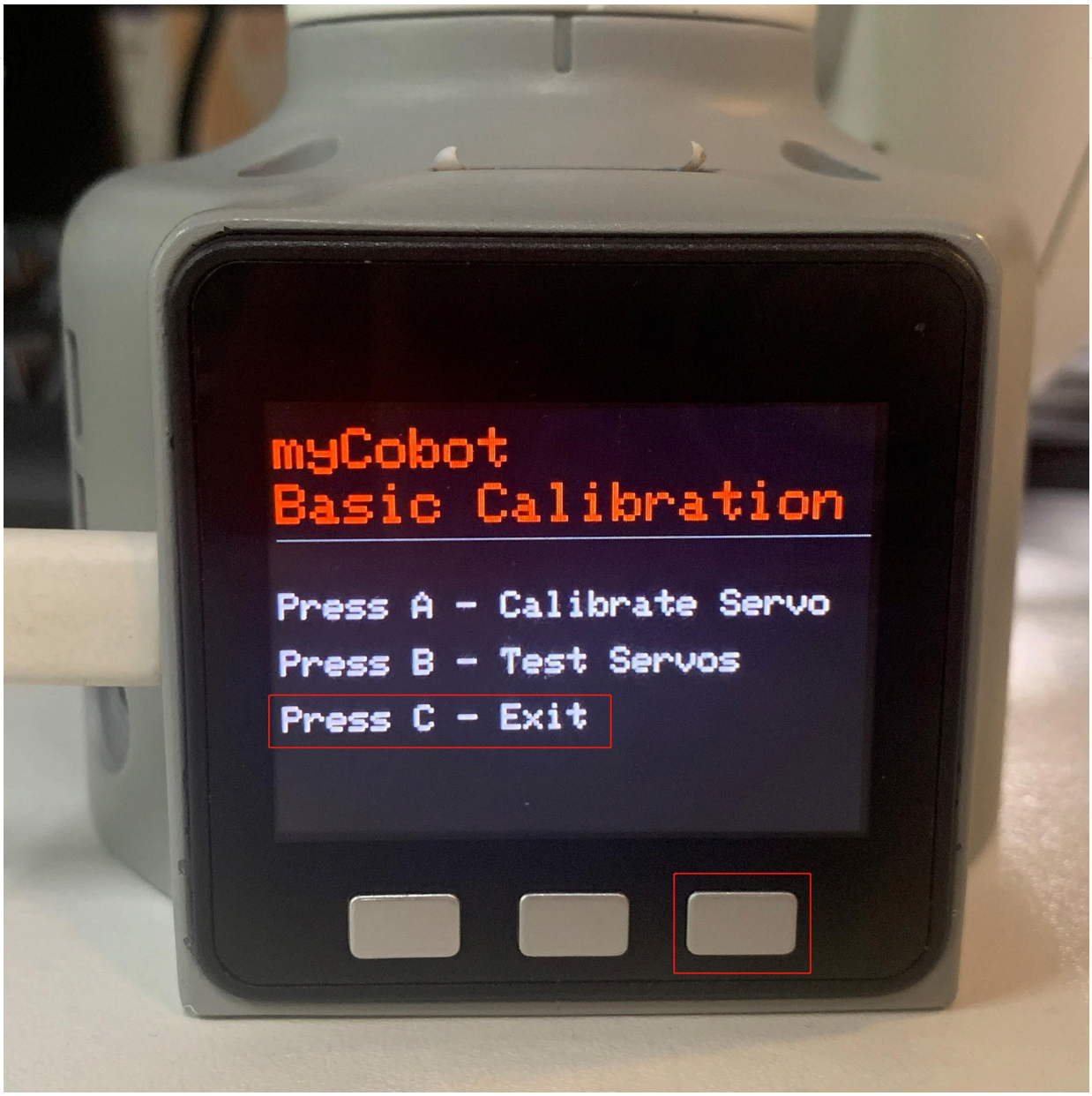
Step 3: Align each joint with zero-position line in sequence.

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

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


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

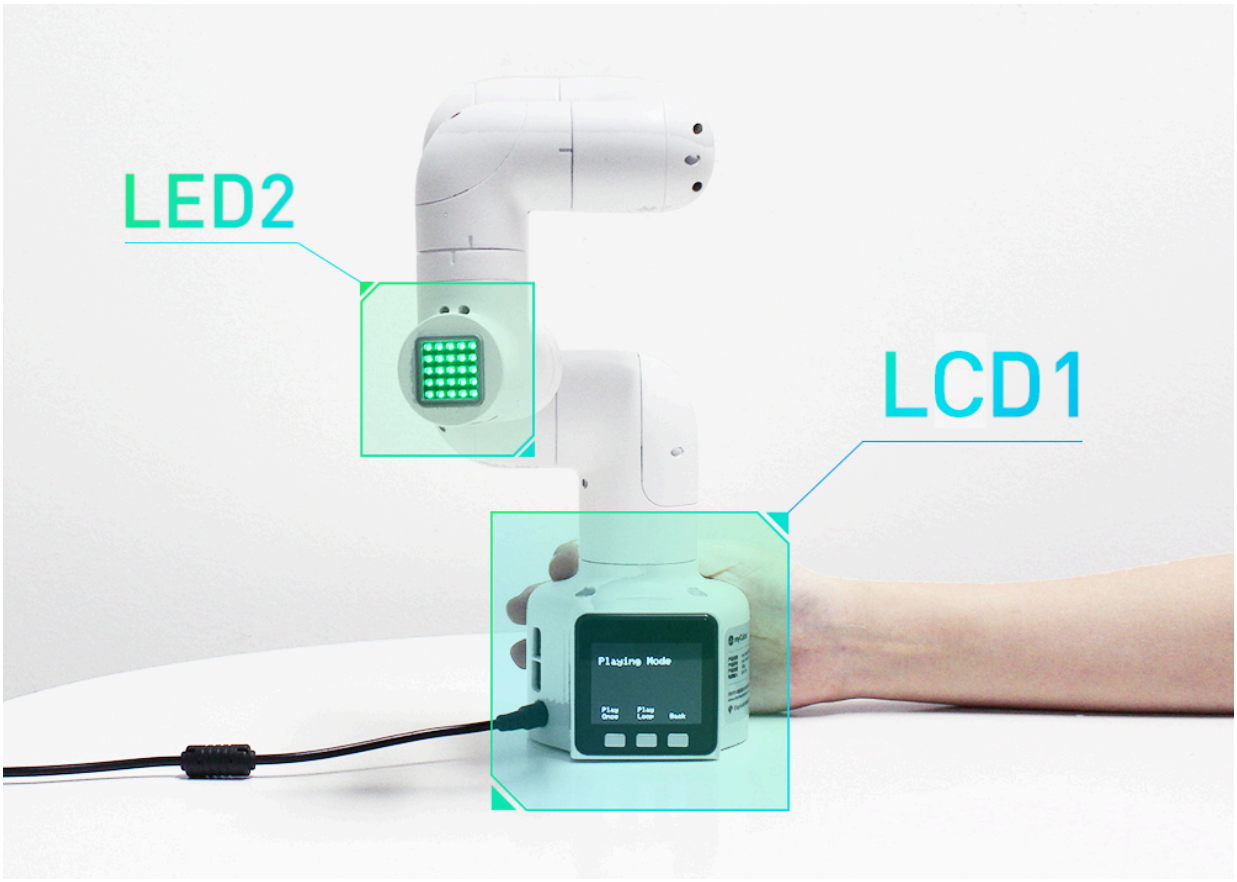
Step 6: Press **C** to exit this function. Calibration completes.



3 Tutorial Video

Video Address: https://www.bilibili.com/video/BV1FT4y1P7BV/

Communication forwarding



Communication timeliness is vital to the micro-controller robot arm. For such arm, we often send control instructions to M5Stack-basic of the stand. Through communication forwarding, the end effector analyzes the instructions and then implements target actions.

This function is mainly used for the customer to develop robot arms independently in different environments.

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

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic**.
- Choose the **Transponder** function (It is unnecessary to burn **M5Stack-basic** for micro-CPU devices)
- press the detection key to detect whether the communication between M5Stack-basic and the end effector Atom is normal
- press the exit button to exit this function.

In this chapter, we can perform real-time detection of the communication between M5Stack-basic and the end effector Atom to see whether it is normal.

Communication

1 Applicable Robotic Arm

- myCobot 280 M5
- myCobot 320 M5
- myPalletizer 260 M5
- mechArm 270 M5

2 Steps to Operate

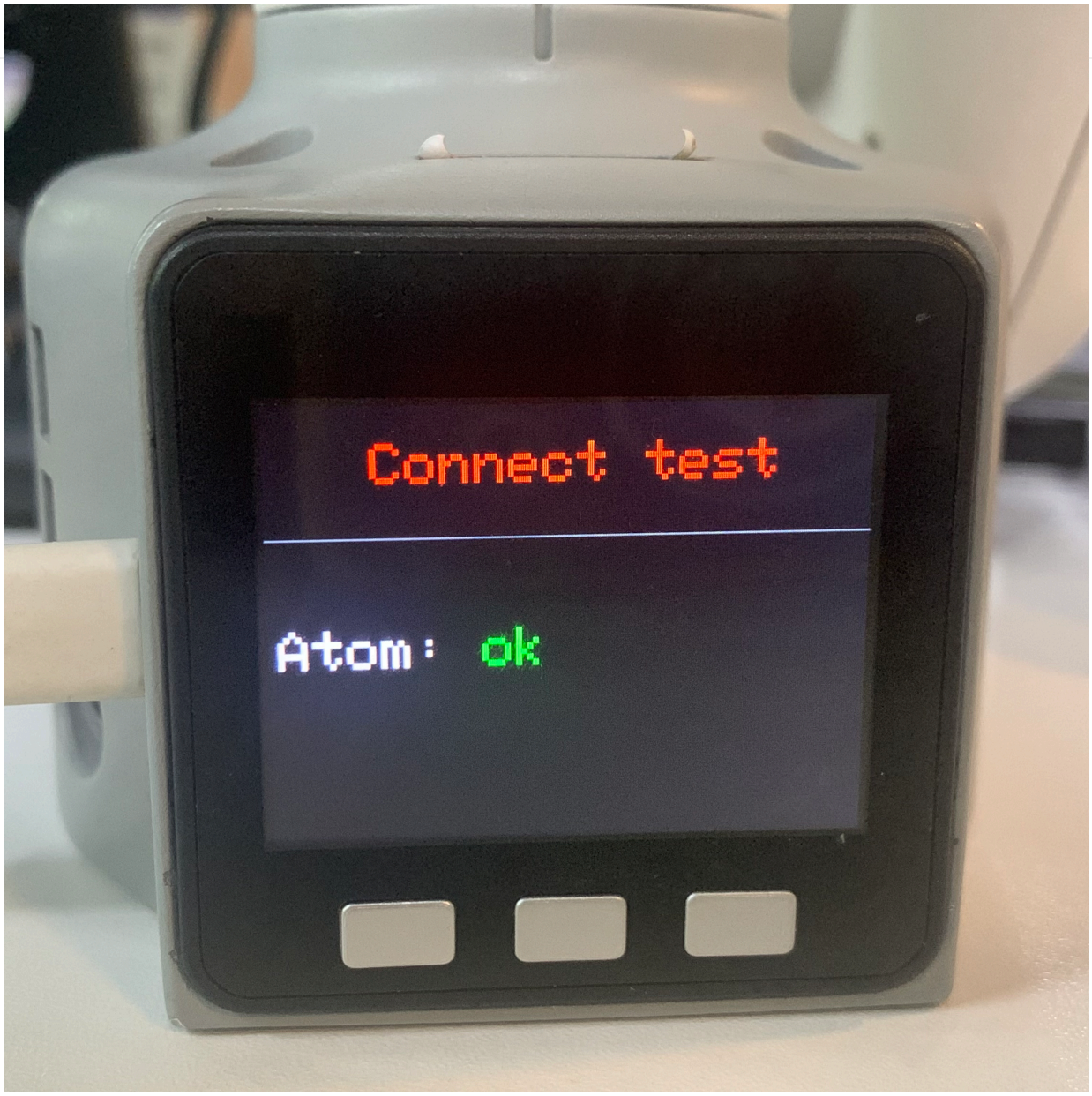
Step 1: Burn the latest **atomMain** for **Atom**.

Step 2: Burn the **minirobot** for **M5Stack-basic**, and press the **Transponder**.

Step 2: Press **A** to detect the connection of Atom.

> **Notice:** `ok` means in a normal state, otherwise `no` is displayed.

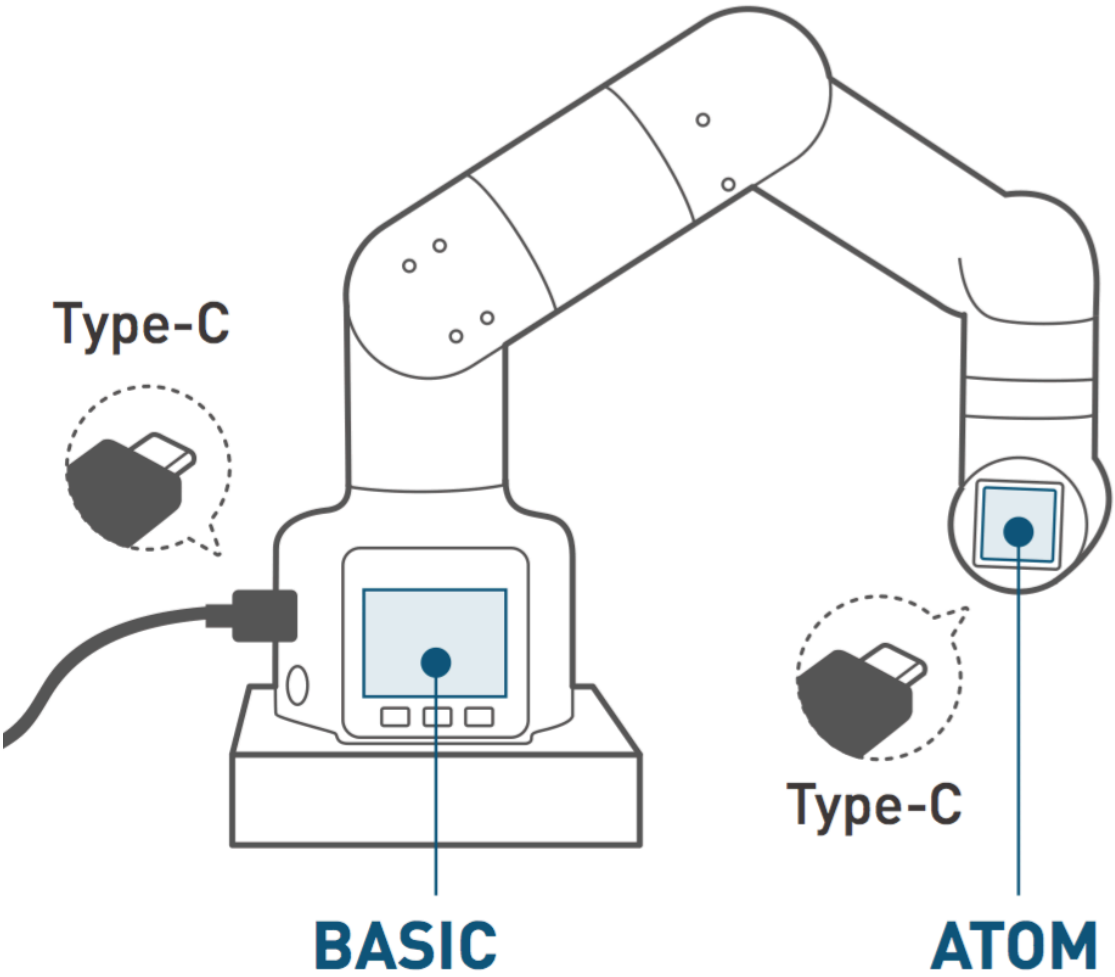

myCobot
Basic Transponder
Press A - Atom ConnectTest
Press C - Exit



Step 3: Press **C** to exit.


myCobot
Basic Transponder
Press A - Atom ConnectTest
Press C - Exit

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.

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

- Burn the latest version of **atomMain** for **Atom**, and the **minirobot** for **M5Stack-basic.**
- Choose the **Information** function (It is unnecessary to burn **M5Stack-basic** for micro-CPU devices)
- press the detection key to detect the connection of the devices
- press the firmware view key to check the version of the current firmware
- press the exit button to exit this function.

In this chapter, we will learn how to use the detection function for different types of devices.

Connection Detection

1 Applicable Robotic Arms

- myCobot 280 M5
- myCobot 320 M5
- myPalletizer 260 M5
- mechArm 270 M5

2 Steps to Operate the Arms

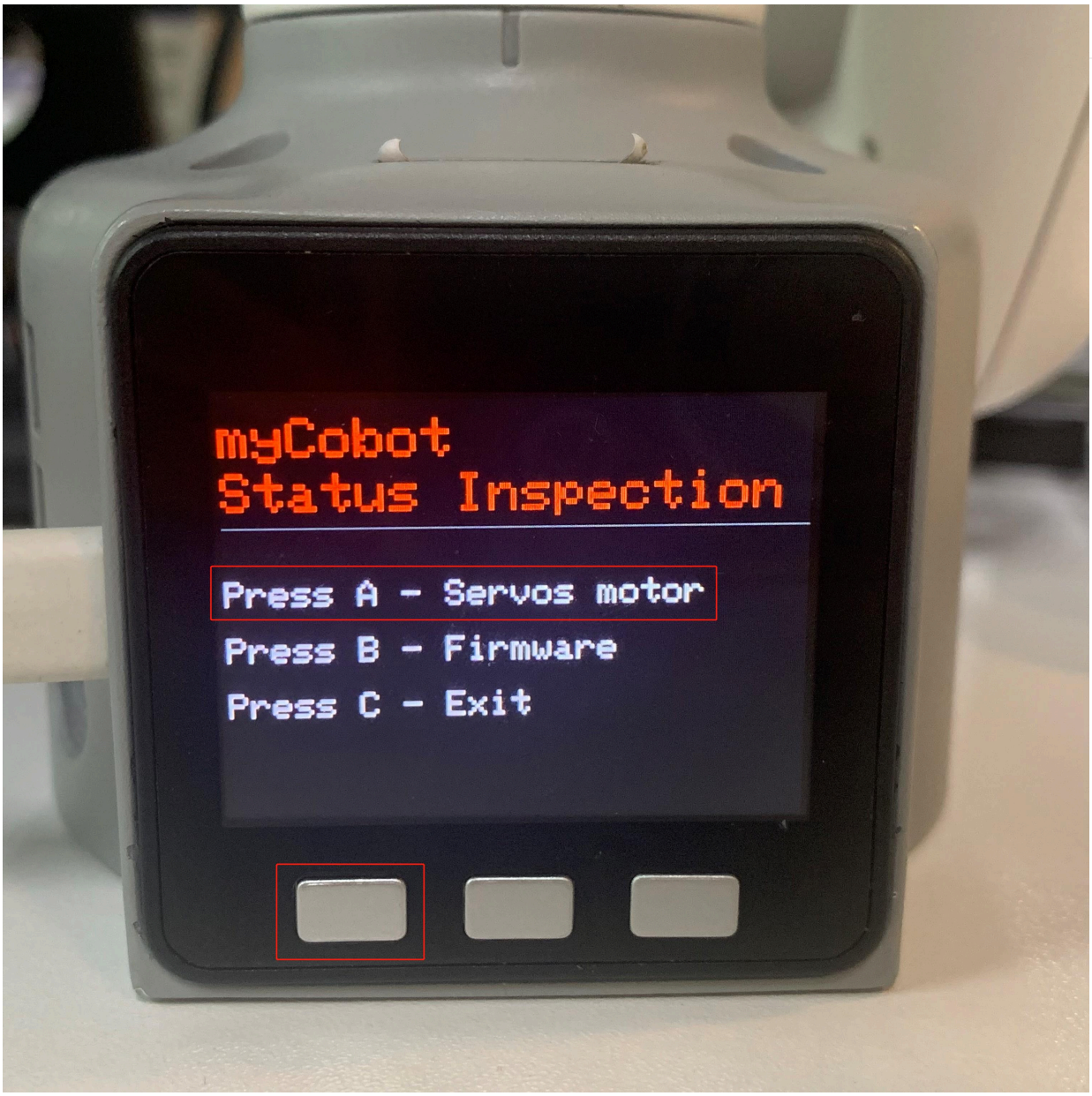
Step 1: Burn the latest version of **atomMain** for **Atom**.

Step 2: Burn the **minirobot** for **M5Stack-basic**, and press the **Information**.

Step 3: Press **A** to start connection detection.

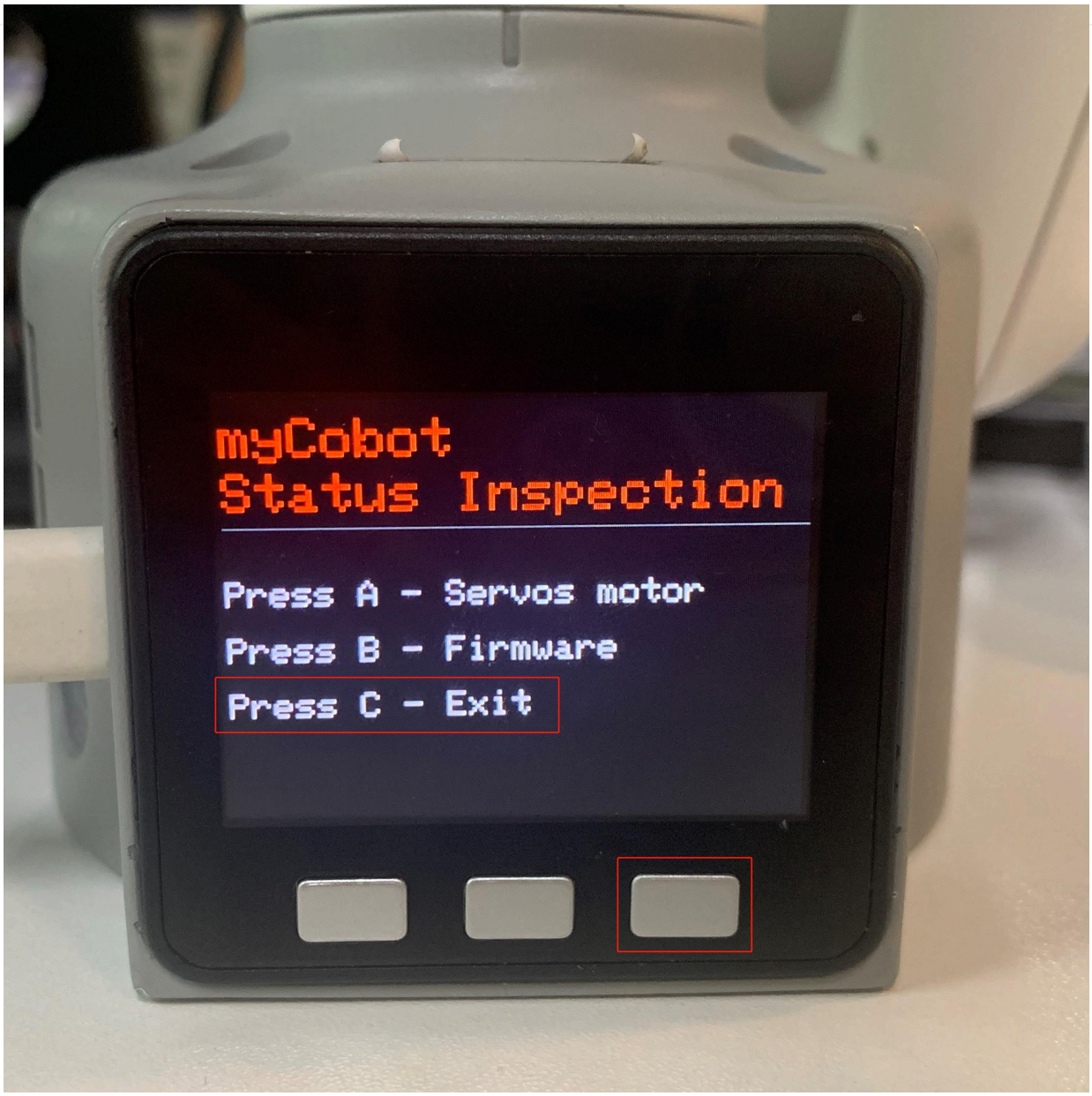
`atom: ok` means that Atom is connected normally.

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


myCobot
Status Inspection
Press A - Servos motor
Press B - Firmware
Press C - Exit

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

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



Use for the first time

Step 1: Unpacking

After receiving the packing box of the robot, first check that the box is intact. **If it is lost or damaged, contact the logistics company and the local supplier.** After opening the box, check the inside items by referring to the packing list.

1 Items included in myCobot robot arm [standard kit]

- myCobot robot arm (Model: myCobot -280)
- Product catalog
- Power supply
- USB-Type C
- Jumper
- M4*35, cup head hexagon socket, full thread, stainless steel screw
- Allen wrench

2 Confirming the operational environment and indicators

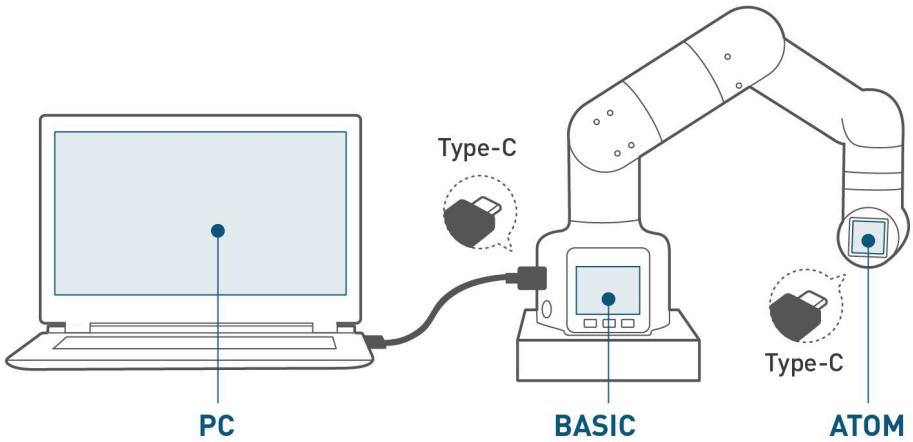
Install the robot system in the environment that meets the conditions described in the table in order to give full play to and maintain the performance of this machine and to use it safely.

	Environment	ndex	
	Temperature	-10~45��	
	Relative humidity	20%~70%	
	Indoor and outdoor requirements	Indoor	
	other environmental requirements	Do not expose it to the sunshine Keep it away from dust, oil fume, salt, iron chips, etc Keep it away from flammable and corrosive liquids and gases It should not come into contact with water Do not apply shock, vibration, etc Keep it away from strong electromagnetic interference sources	

Step 2: Connection

Connect the power adapter as shown in the figure.

Connecting the computer



Step 3: Power supply

The power for myCobot must be supplied by an **external power supply** to provide it with sufficient power:

- **rated voltage**: 8.4-12V
- **rated current**: 3-5A
- **plug type**: DC 5.5mm x 2.1

Note: **Do not supply power for myCobot only using TypeC inserted into Basic**

Use the power supply officially provided by the manufacturer to avoid damage to the robot arm.

Step 4: Fixing myCobot

1 How to fix myCobot

During the movement of the robot arm, if **the bottom surface of myCobot is not connected to the table top or other bottom surface**s, myCobot will **shake or overturn**.

There are three common ways to fix the robot arm:

- 1.Fix it on a stand with a LEGO interface by using a LEGO inserting key

  We provide two types of stands: flat sucking disc and G-clamp, which can be found in myCobot peripheral stand.

  - Flat stand **adaptive models**: myCobot 280

    - 1.Install a sucking disc on the four corners of the stand and tighten them.
    - 2.Connect the flat stand and the bottom of the robot arm by using accompanying LEGO parts.
    - 3.Fix the four sucking discs on a flat and smooth surface before use.

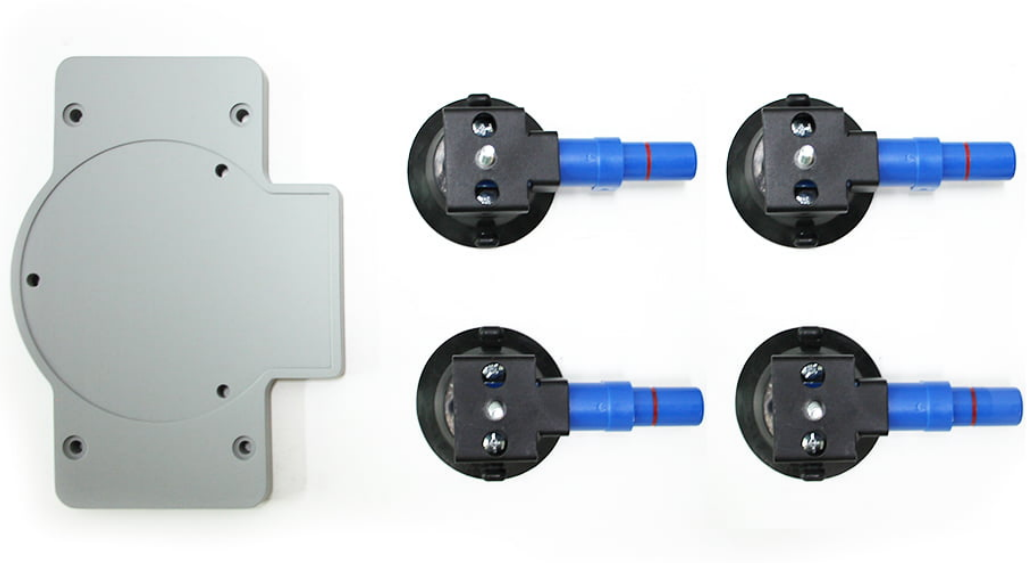
  Tip: A small amount of **non-conductive liquid** can be applied under the sucking discs to fill the gap between the sucking discs and the table top for the best suction effect.

- G-stand **adaptive models**: myCobot 280 and myPalletizer 260

- 1.Fix the stand to the edge of the table with a G-clamp.
- 2.Connect the stand and the bottom of the robot arm by using accompanying LEGO parts.

- 3.Do not use it until it becomes stable.

- 2.Pass **a screw provided for the robotic arm stand** through such stand, and fasten it to a threaded stand.
  - Sucking disc stand **adaptive models**: myCobot 320



Suitable for flat and smooth surfaces

- 1.First open the sucking discs of the self-priming pump, align the four holes in the stand bottom plate with the self-priming pump, then place the nuts in the four holes and tighten them, and finally fix the bottom plate and sucking discs.
- 2.Remove the cover at the bottom of the sucking discs, place it on a smooth table, and press the sucking discs to pump air. After pumping air from the four sucking discs, you can test whether the bottom plate has been fixed, and then tighten the nuts again.

- 3.Next, fix the robot arm to the bottom plate and align the robot arm with the groove of the bottom plate. After placing them in position, place three M6x90 screws in three holes of the robot arm, place the corresponding nuts on the bottom plate at the same position, and then tighten the screws with an Allen key to fix the robot arm on the bottom plate.

2 Connecting the screw hole of myCobot stand

The robot should be fixed on a solid stand before it can be used normally. Stand weight requirements: fixed stand or mobile stand.

Before installation, make sure that there are corresponding threaded holes on the fixed stand.

Before formal installation, confirm the following:

- The environment in which the stand is to be installed should meet the requirements in the table \"Working Environment and Conditions\" above.
- The installation position should not be smaller than the working range of the robot, and sufficient space should be reserved for installation, use, maintenance and repair.
- Put the stand in a suitable position.
- Installation-related tools, such as screws, wrenches, etc. should be ready.

After confirming the above points, move the robot onto the stand mounting table, adjust the robot position, and align the fixing holes of the robot stand with the holes in the table. After aligning the holes, align the screws with the holes and tighten them.

Notice: When adjusting the robot position on the table, don\'t push and pull the robot on the table directly as much as possible to avoid scratches. When manually moving the robot, don\'t apply external force to the vulnerable parts of the body so as to avoid unnecessary damage to the robot.

Video Tutorial

Opening the box of myCobot quickly Opening the box of myCobot Pro quickly

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

5 Development and Use based on Drag-and-drop Programming Software

What's drag-and-drop programming?

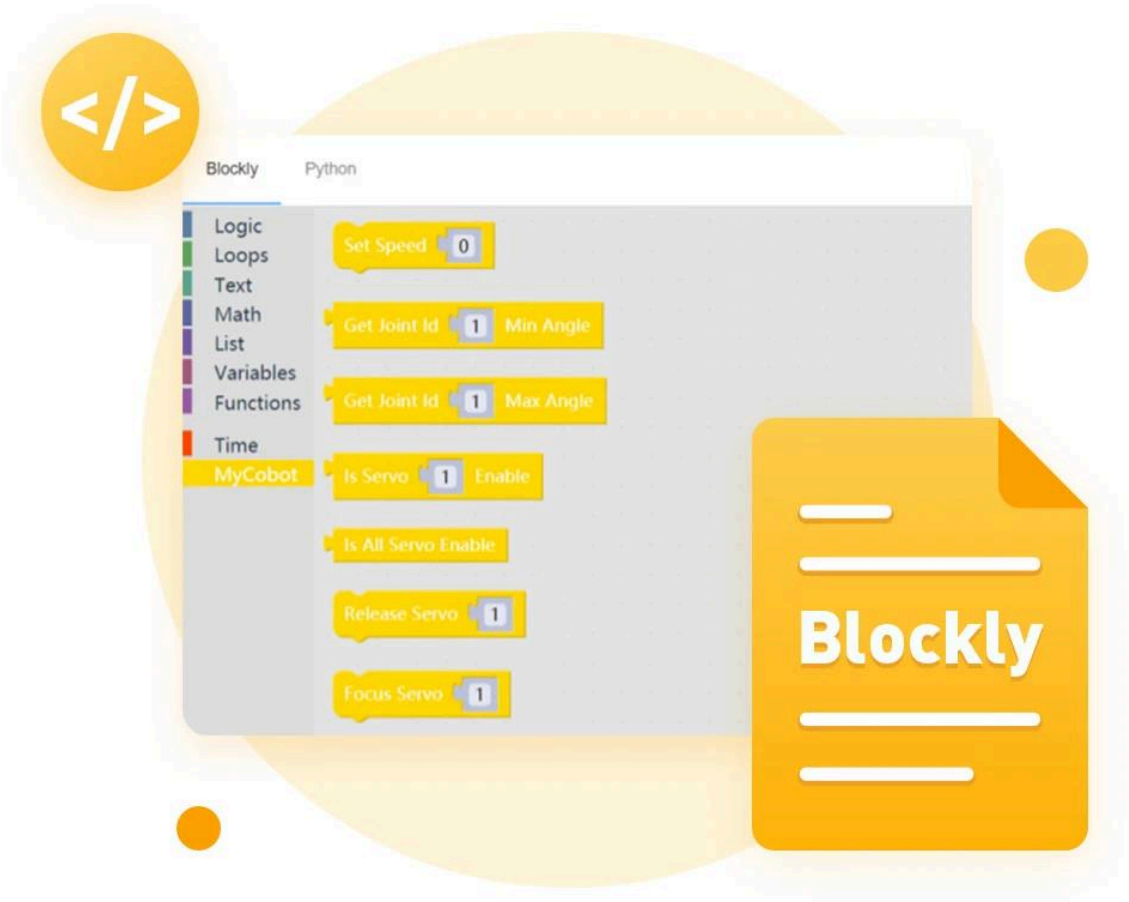
"Drag-and-drop" programming is a basic technology which allows programming by dragging and dropping code building blocks or using other vision clues instead of writing text-based codes manually. Therefore, it makes complex and abstract programming languages easy to understand.

Hadi Partovi, the founder of Code.org, once said "that drag-and-drop programming allows you to understand programming basics easily without carefully studying character arrangement. Once knowing the basic concepts to use drag and drop, you will immediately want to learn how to use them to do actual things." This completely follows the basic logic for learning programming.

A number of programs can help you get started with drag and drop programming, which include Google Blocky, Microsoft MakeCode, MIT Scratch, etc.

What are myBlockly and UIFlow respectively

Both **myBlockly** and **UIFlow** implement the model of drag-and-drop programming. The former is developed by the R&D team of Shenzhen Elephant Robotics Company, and the latter is a graphical programming platform developed by M5Stack.



myBlockly ▶ is a piece of puzzle programming software based on python environments and pymycobot dependent libraries, which allows the user to do programming in a block building way to control mycobot. It is suitable for programming learners and can train their programming logic skills. myblockly also provides a python display interface, which can convert the myblockly program built by the user to a python language one to help the user learn python statements.

However **UIFlow** ▶ is a programming tool specially designed for M5 hardware system, which uses Blockly graphical programming and Python code programming. It not only provides programming learners with friendly programming experience, but also delivers more professional and crucial programming knowledge.

First-Time Use

Preparing for the first-time use

Follow the steps below to download Python and burn firmware via MyStudio.

Step 1 Download Python. myBlockly is used in Python environment. Make sure Python is downloaded on PC (**Reference for information about configuration of Python environment**).

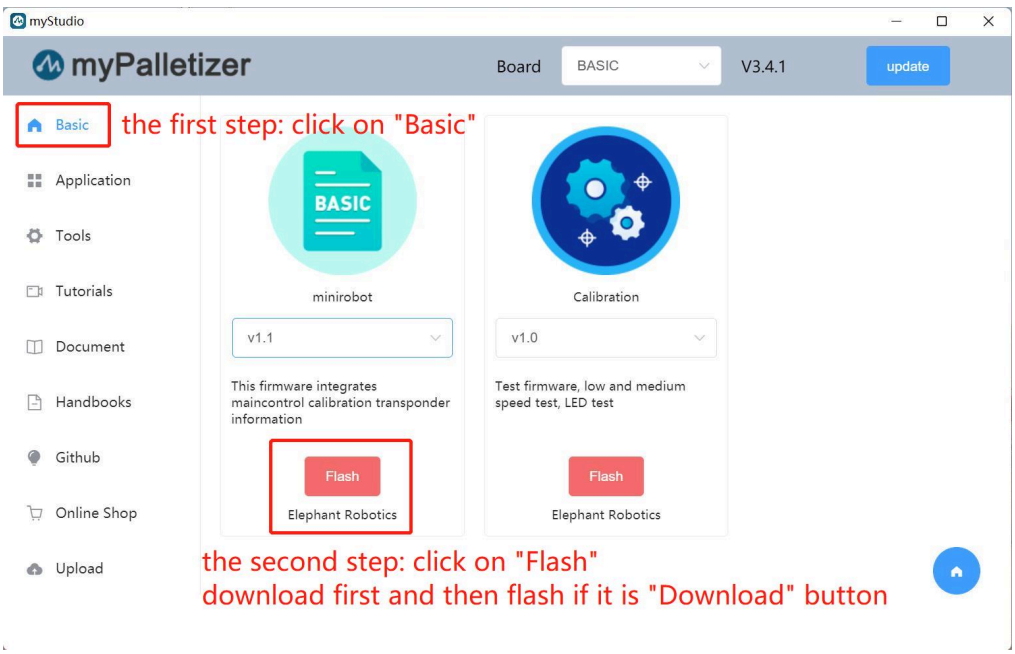
Step 2 Burning firmware.

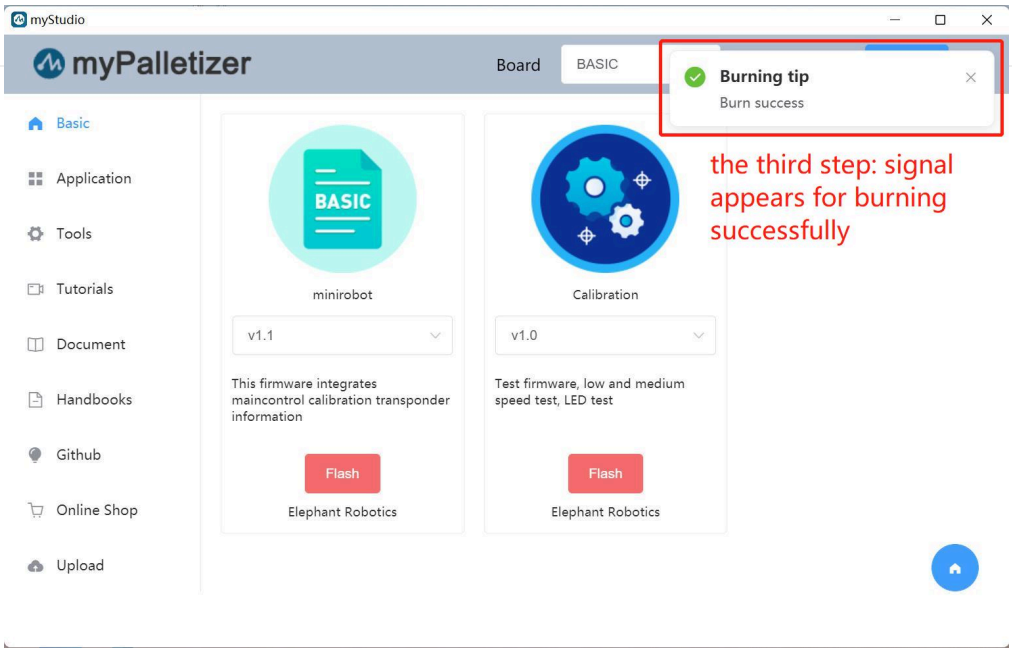
- Before burning firmware, download corresponding serial port driver according to the USB chip on your PC. CP210X is suitable for CP2104 version and CP34X is suitable for CH9102 version. You can install both of them if you are unable to confirm the type of USB chip. Go to the address below to download 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.
> - Go to **4.1.1 MyStudio Installing driver** for specific information about serial port driver installation.
> - 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.

Step 3 M5Stack-basic burning is not required on Raspberry Pi series and JETSON NANO series, and their lasted atomMain on Atom is factory burnt.

Step 4 For M5Stack series, miniRobot is required to be burnt on M5Stack-basic via MyStudio. The lasted atomMain on Atom is factory burnt. Go to **4.1 MyStudio** for its download and installation. Follow the steps below to burn miniRobot.





Step 5 Press "Transponder" button on M5Stack-basic, and then press "Atom ConnectTest".

Installing myBlockly

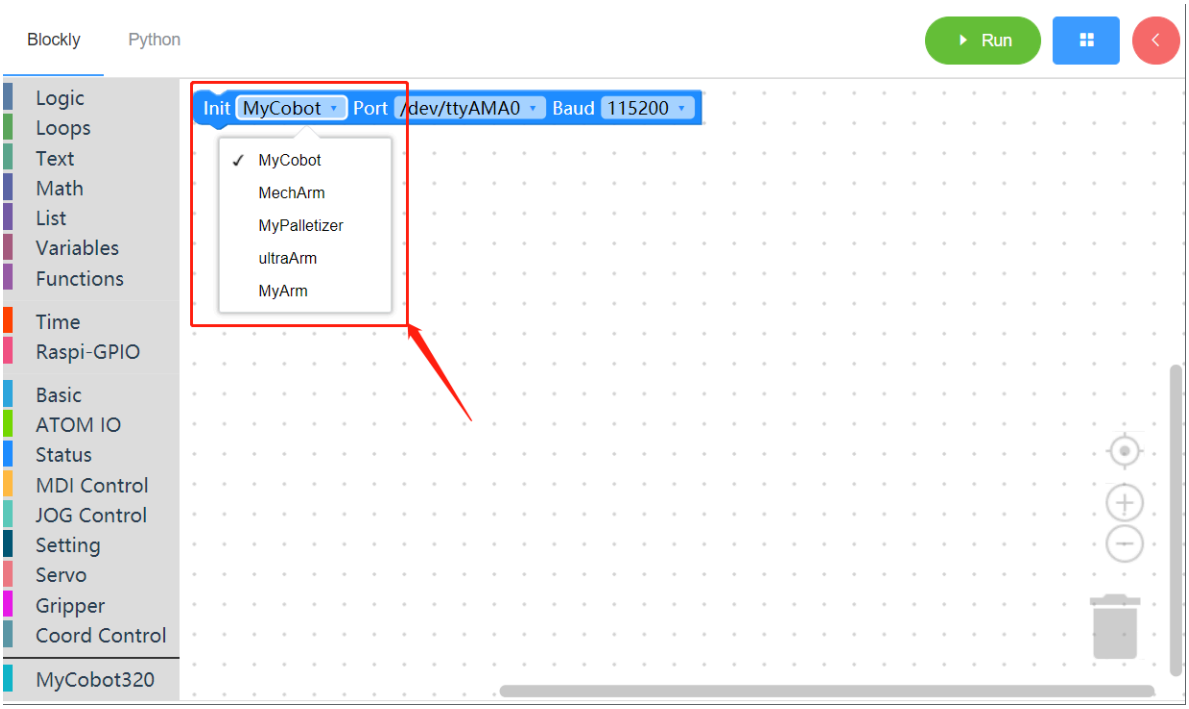
Go to the addresses below to download myBlockly.

- **Github**
- **Official Web**

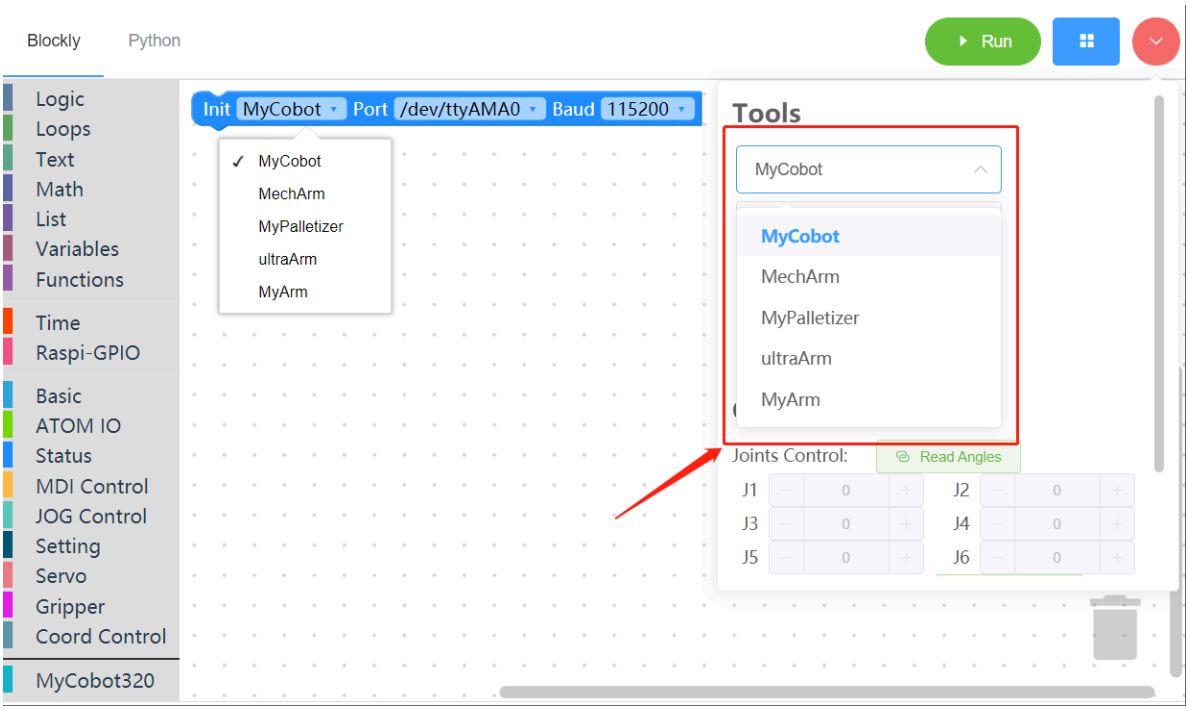
> **Notice:**
>
> Make sure to download the latest version.

Before use

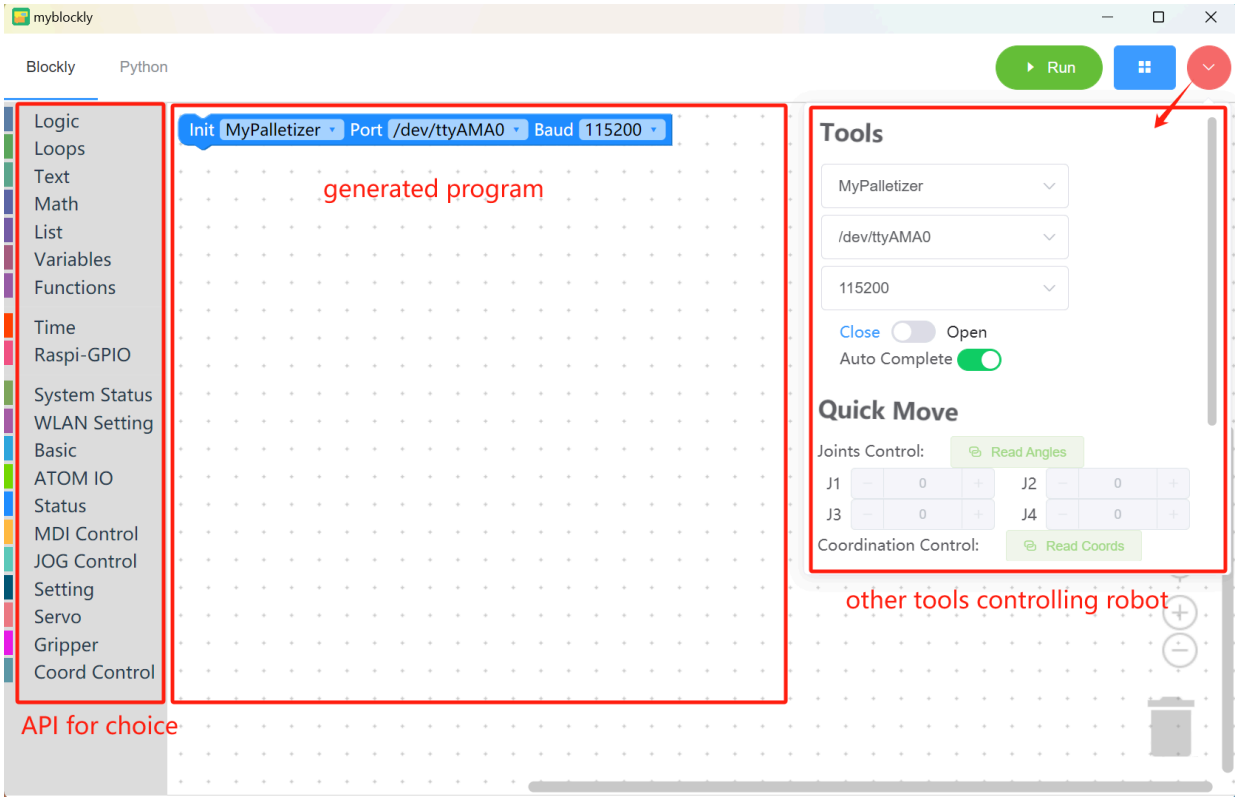
- Before start to programming, be sure to select the **corresponding machine model** , otherwise it is easy to cause hardware damage



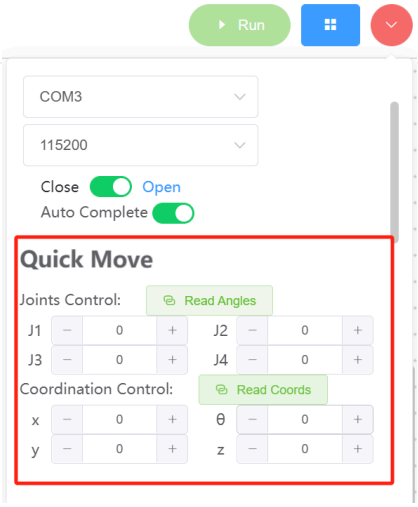
- When controling robot by quick move，be sure to select the **corresponding machine model** , otherwise it is easy to cause hardware damage



myBlockly View

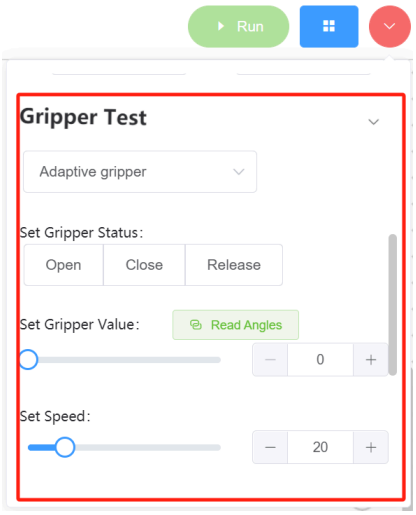


- **API blocks:**
  - Display API used to generate programs
  - Contains method modules required for programming, which can be concatenated by placing the mouse into the program editing area
- **Generated program:**
  - Before running the program, it is necessary to select the correct model, port, and baud rate in the initialization module or other tools controlling, otherwise the program cannot run normally.
  - Drag and drop the required module methods to the area and concatenate them to implement your own program.
  - If the baud rate and serial port have been modified in the right toolbar, but the current address is still/dev/ttyAMA0, it is due to the myBlockly version issue. You need to update the software version on the official website first (the latest version will change the information in the editing area after selecting the serial port and baud rate in the toolbar).
- **Other tools controlling robots:**
  - Click on the pink icon in the top right corner to select the corresponding robot type, serial port and baud rate.
  - Other tools — Quickly move



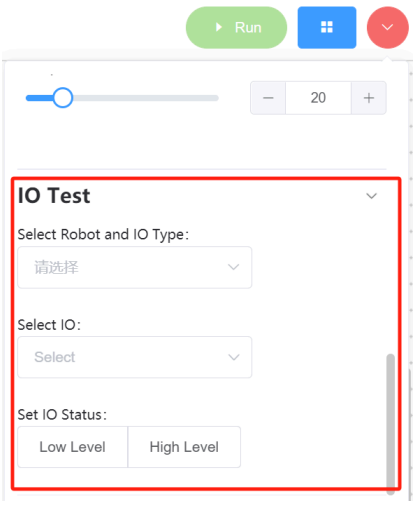
Find the Quick Move module below in the other toolbars. Tap `Joints Control` or `Coordination Control` for joints' real-time angles and coords. Tap `+` or `-` to control robot movements.

- Other tools — Gripper Test (At present, the gripper test function only adapts to the following models: MyCobot 280 series, MeCharm 270 series, MyPalletizer 260 series, MyARM 300 PI.)



Find the Gripper Test module in other toolbars. You can set the type of gripper and the movement speed of the gripper. Click the 'Read Angle' button to obtain the current value of the gripper. Set gripper status: open, close, release. Pay attention to the movement of the gripper.（Please refer to the specific testing steps--Gripper Test）

- Other tools— IO Test（At present, the IO test function only adapts to the following models: MyCobot 280 series, MeCharm 270 series, MyPalletizer 260 series, MyARM 300 PI.）



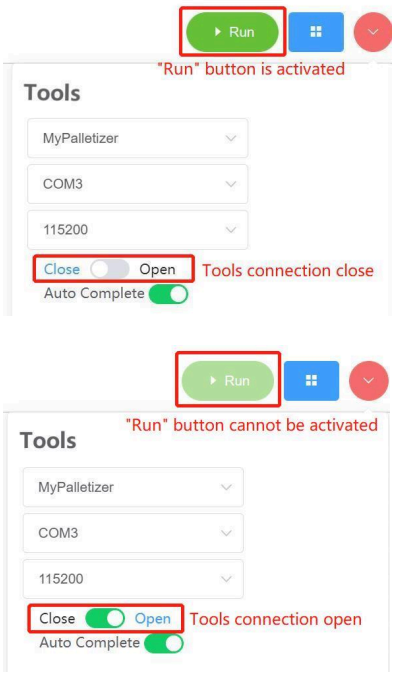
Find the IO test module in other toolbars. Select the machine and IO type, and choose the IO value. You can click the low or high button to change the current selected IO port status.（Please refer to the specific testing steps--IO test）

Notice:

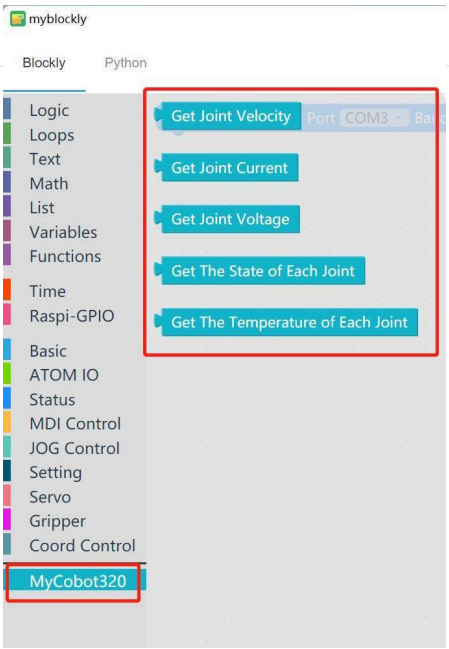
1. Baud rate for M5Stack series and Raspberry Pi series is 115200 and 1000000 respectively.

Robot	Port	Baudrate
260 M5	Win: COM*; Linux: /dev/ttyUSB*;	115200
270 M5	Win: COM*; Linux: /dev/ttyUSB*;	115200
280 M5	Win: COM*; Linux: /dev/ttyUSB*;	115200
320 M5	Win: COM*; Linux: /dev/ttyUSB*;	115200
260 PI	/dev/ttyAMA0	1000000
270 PI	/dev/ttyAMA0	1000000
280 PI	/dev/ttyAMA0	1000000
320 PI	/dev/ttyAMA0	115200
280 Jetson Nano	/dev/ttyTHS1	1000000
280 Arduino	Win: COM*; Linux: /dev/ttyUSB* or /dev/ttyACM* ;	1000000

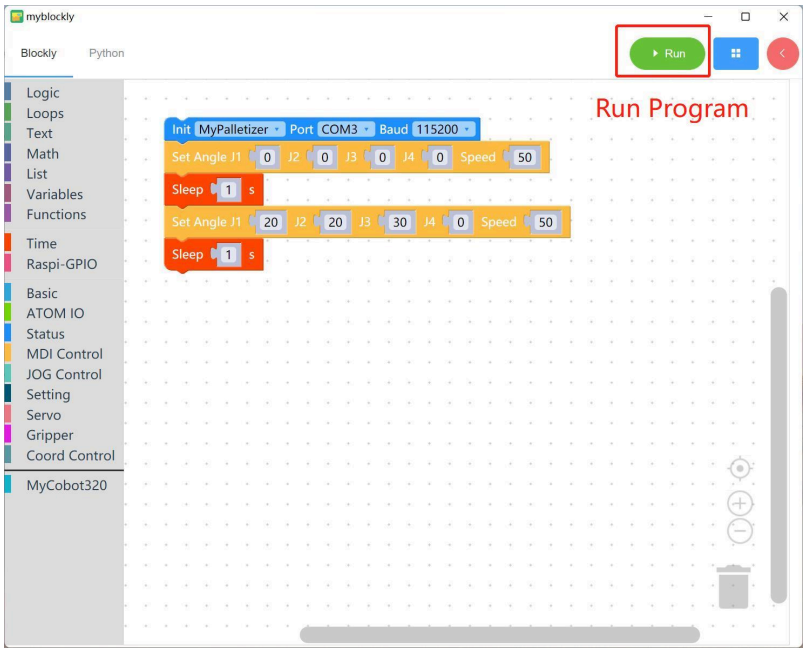
1. Go to **4.1.1 Download and Installation of myStudio** for serial port and baud rate of different types of robots.
2. Check connection of `Tools` in top right corner if program cannot run successfully.



3. API for MyCobot 320 is specifically set in the bottom left corner. It does not affect any program.



Running Program

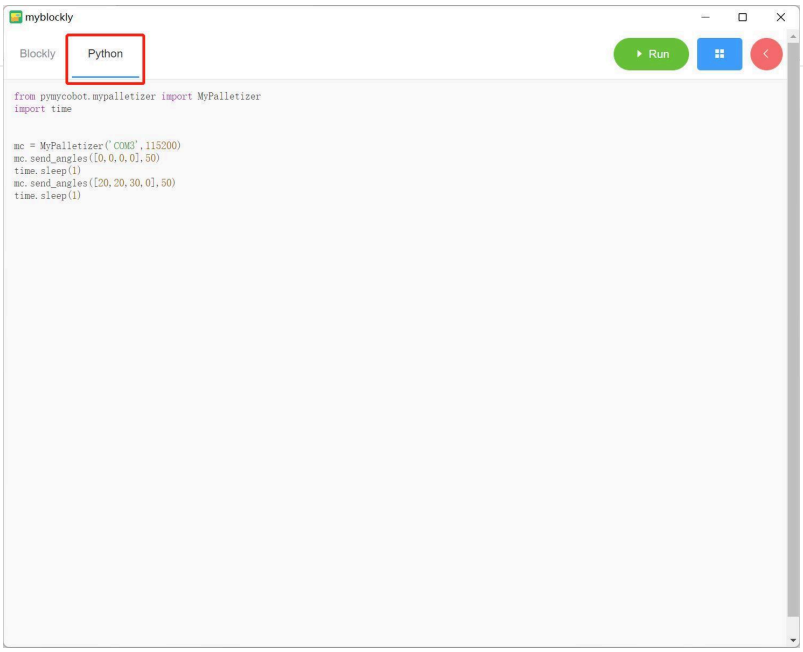


Drag API blocks from the left side to integrate them with each other for program edition. Then, tap "Run" button to run the program.

Notice:

`Sleep` block is required to provide no less than 0.5 second for a robot to move. Otherwise the robot cannot run successfully.

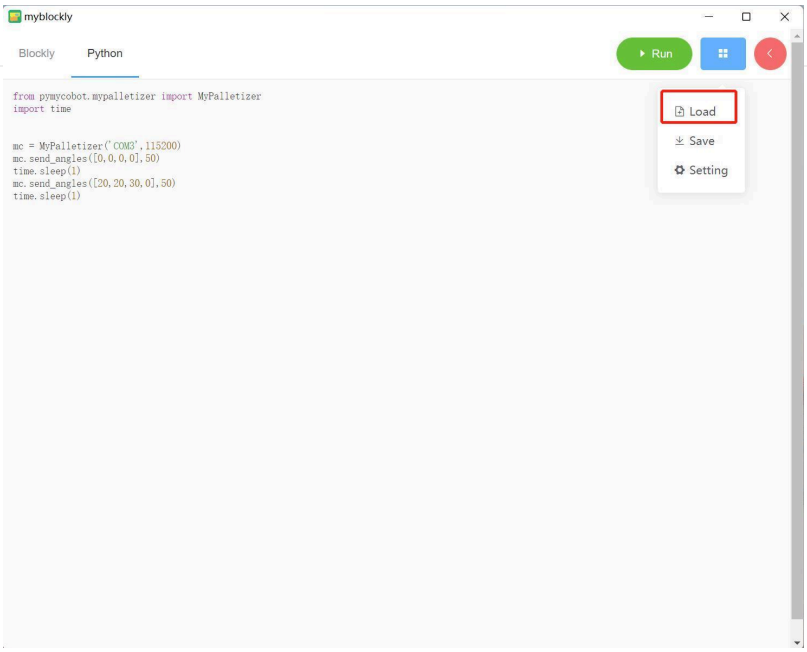
Click on `Python` for Python codes corresponding to blocks edited in myBlockly.



Saving and Loading Programs

- myBlockly files are saved as "*.json" format. Click on the blue button in the top left corner, and then click on "Save".

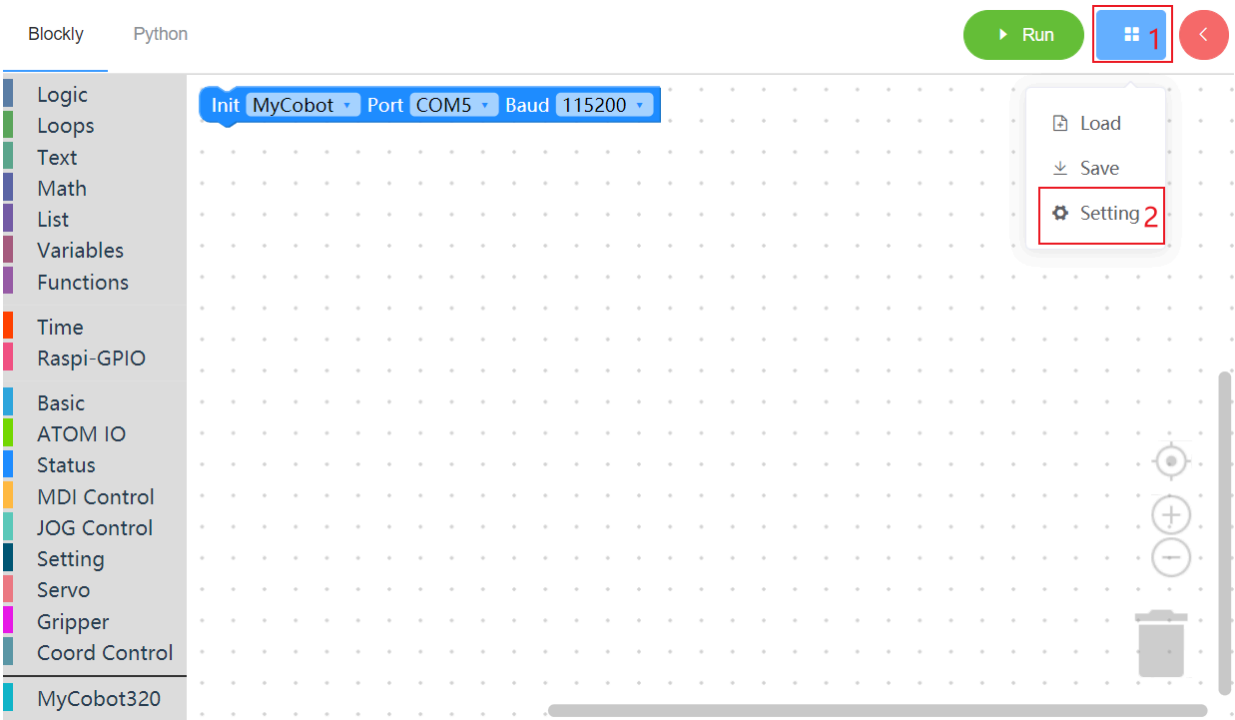
- Click on "Load" button to import existing programs.

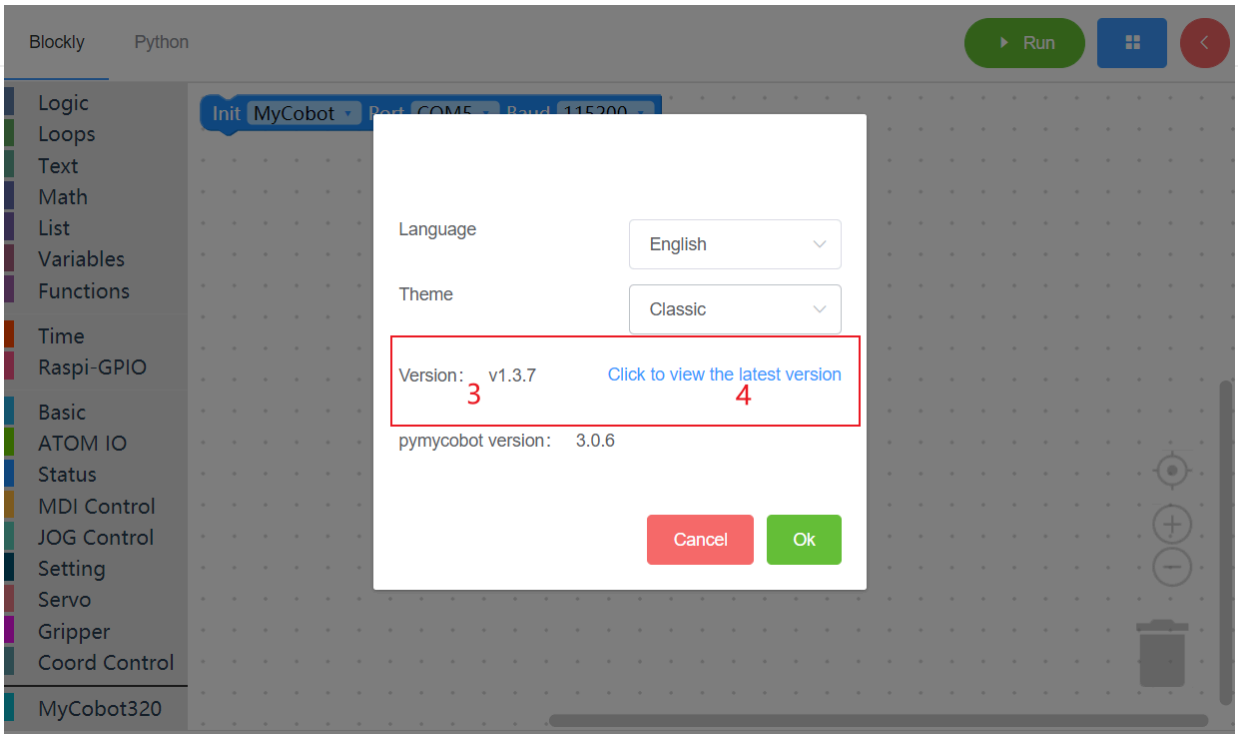


How to install and update the software

The installation and update of myBlockly need to go to the official website to download the latest version.

Inside the software, you can update the software through the following steps.





- 1 Mouse move here
- 2 Click setting
- 3 This is your current version
- 4 Click this link to view the latest version and download

And if you need to know the detailed update log of the software, you can go to github to view.

How to run the sample code

We provide a wealth of sample codes, please select the code you are interested in, then find the sample building blocks from the left menu bar, drag them to the workspace, and click Run to see the result.

- Setting the Color of RGB Light Panel
- Setting All Arms to Starting Point
- Controlling Single-Joint Motion
- Controlling Multi-Joint Motion
- Swinging Arms Left and Right
- Let Robot Dance
- The Use of Gripper
- The Use of Sucking Pump

How to view the drive library open interface

Myblockly implements all interfaces of pymycobot. You only need to find the corresponding block of the interface you want to use in the menu bar on the left side of the workspace, and then drag it to the workspace to use it.

Setting the Color of RGB Light Panel

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

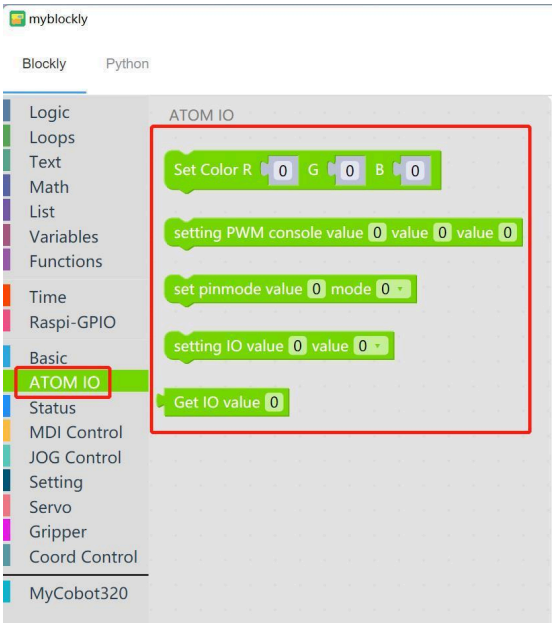
Other series: Make sure the robot is in normal status.

Purpose for this section

This section includes instructions for controlling the RGB light panel.

Introduction to API

- `Set Color R() G() B()`



- Applicable to myCobot 280 series, myCobot 320 series, myCobot Pro 600 series and myPalletizer 260 series

- Parameters:

  Set the color of RGB light panel. R `x` , G `x` and B `x` means red, green and blue respectively.

  Brightness Range:

  R: 0-255

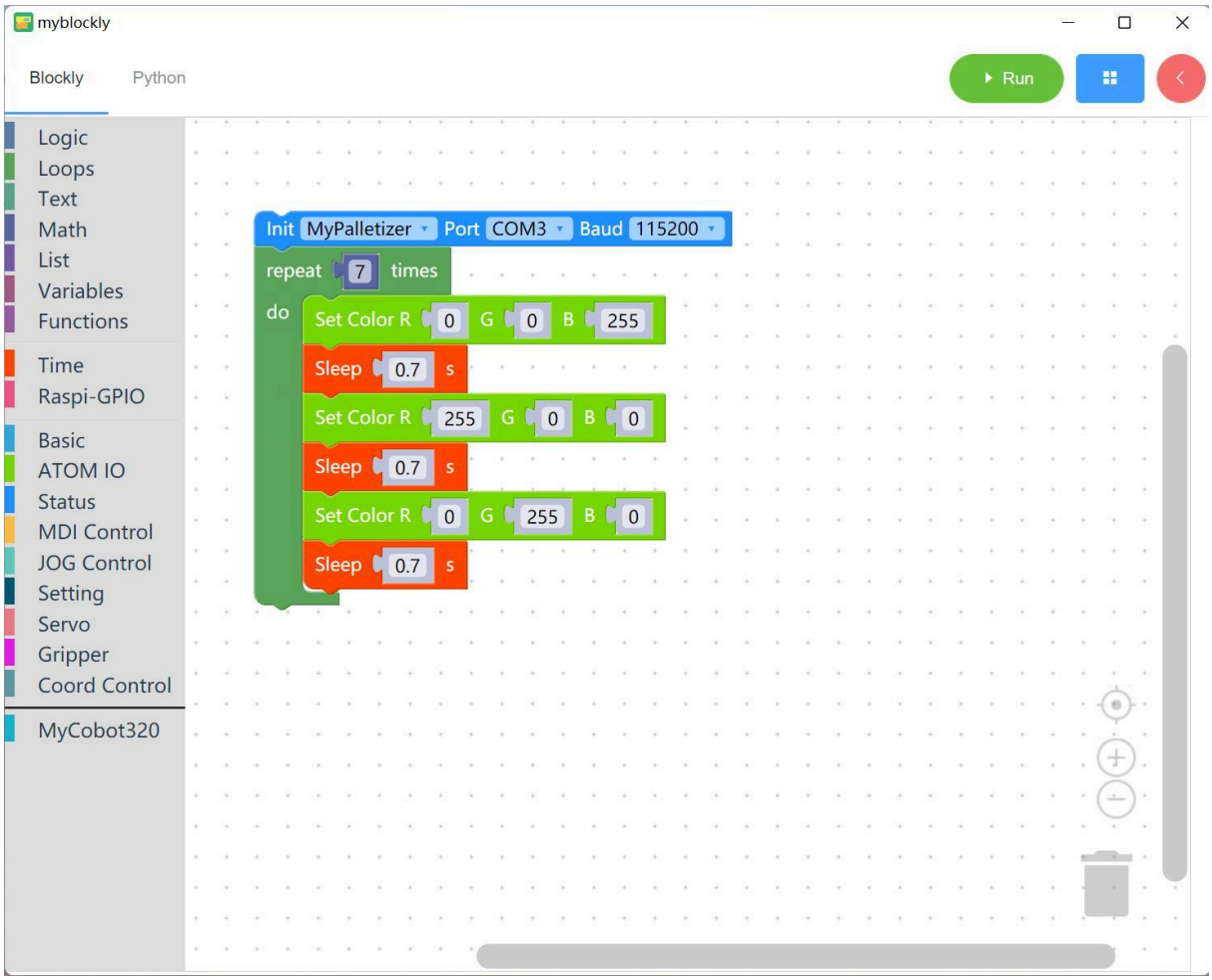
  G: 0-255

  G: 0-255

- Function: Controlling RGB light panel

Simple Demo

- program for display



- Change in color of RGB light panel:

  The color of RGB light panel changes in sequence of blue, red, then green. The whole process loops 7 times.

Setting All Arms to Starting Point

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

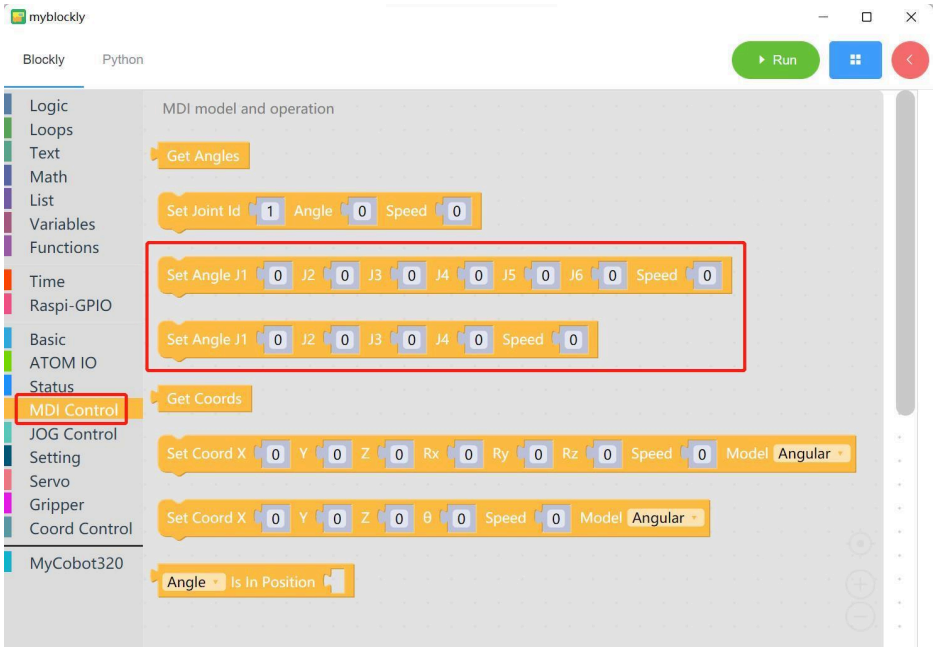
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for setting all arms to starting point.

Introduction to API

- `Set Angle ()`



- Applicable to six-axis robots (myCobot 280 series, mechArm series and myCobot 320 series)

- Applicable to four-axis robots (myPalletizer series)

- Parameters:

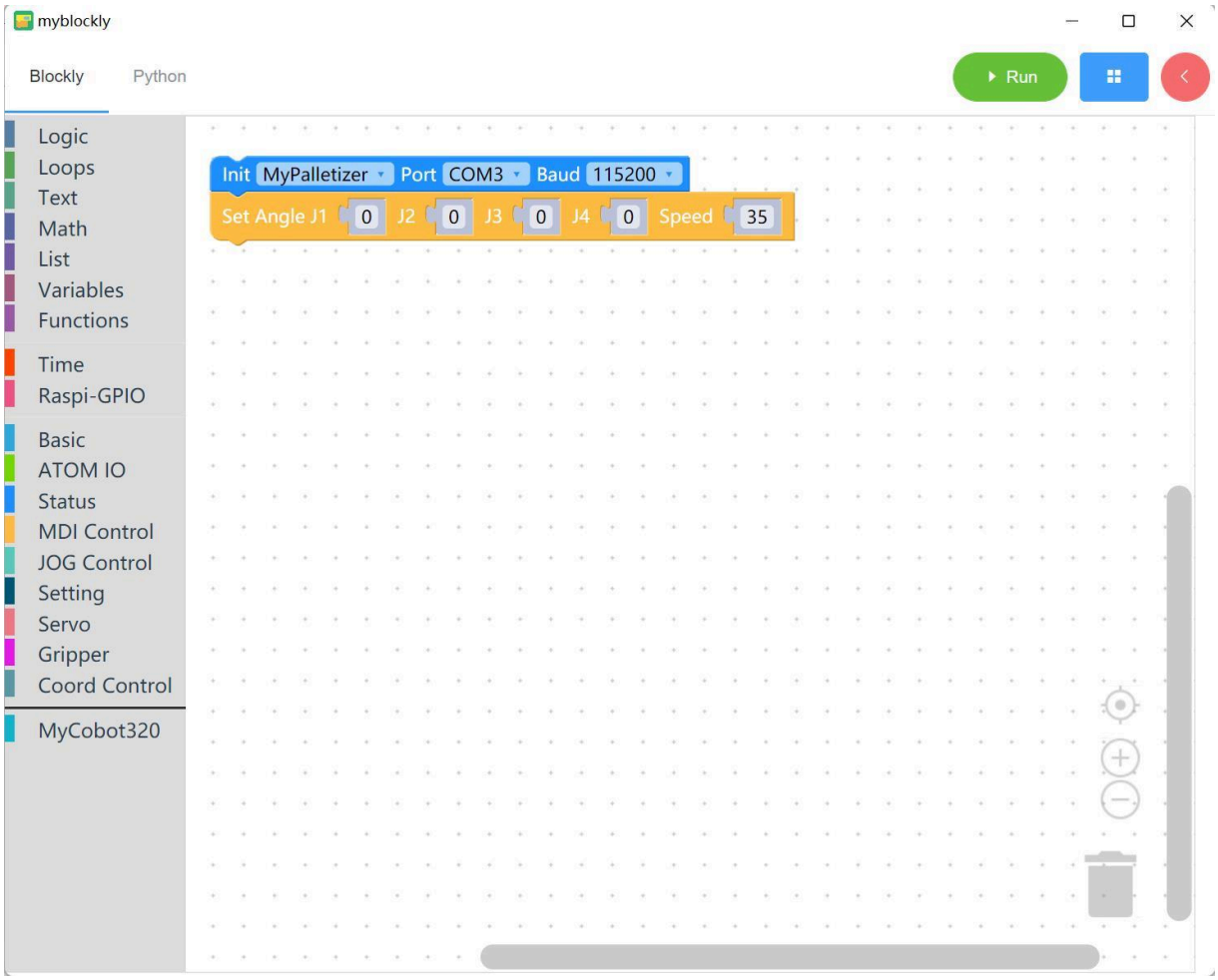
  Joint angle: if arms are set to starting point, set parameters of angles as `0`

  Speed: Refer to **2 Products Profile**

- Function: Set all arms to the starting point (Angle `0`)

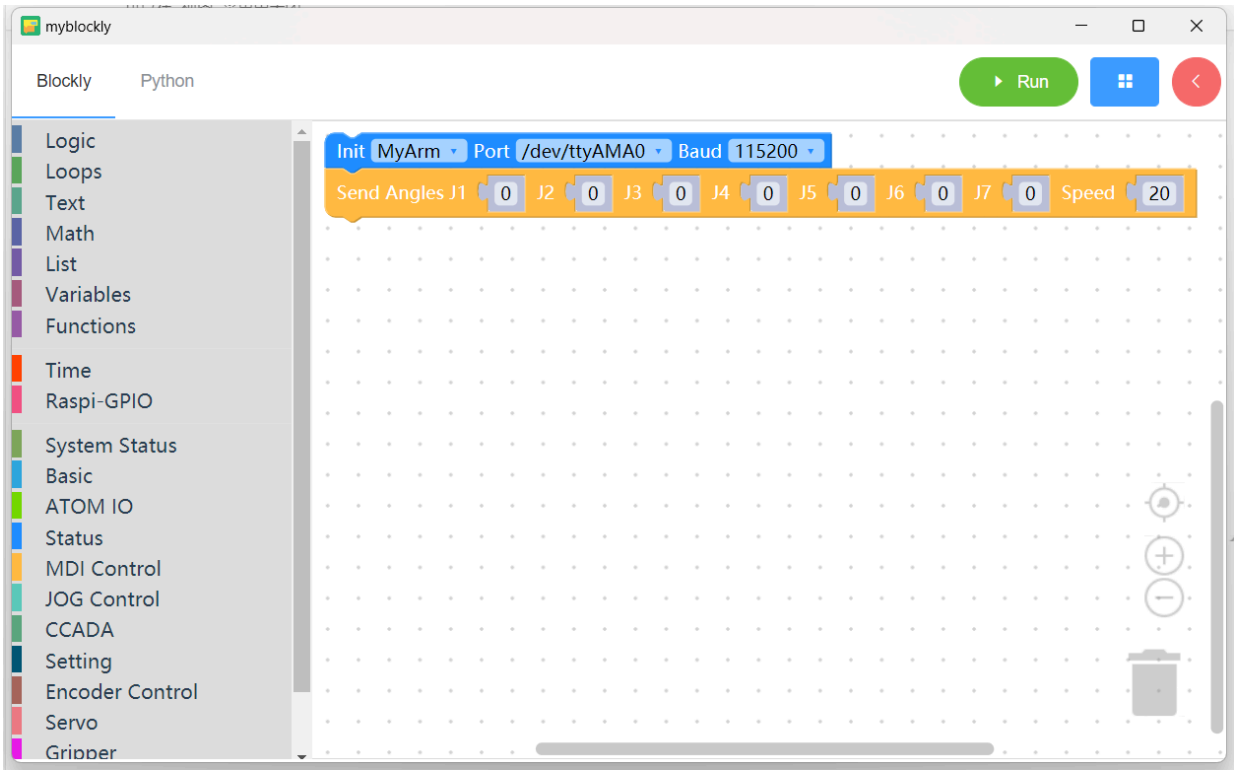
MyPalletizer

Simple demo



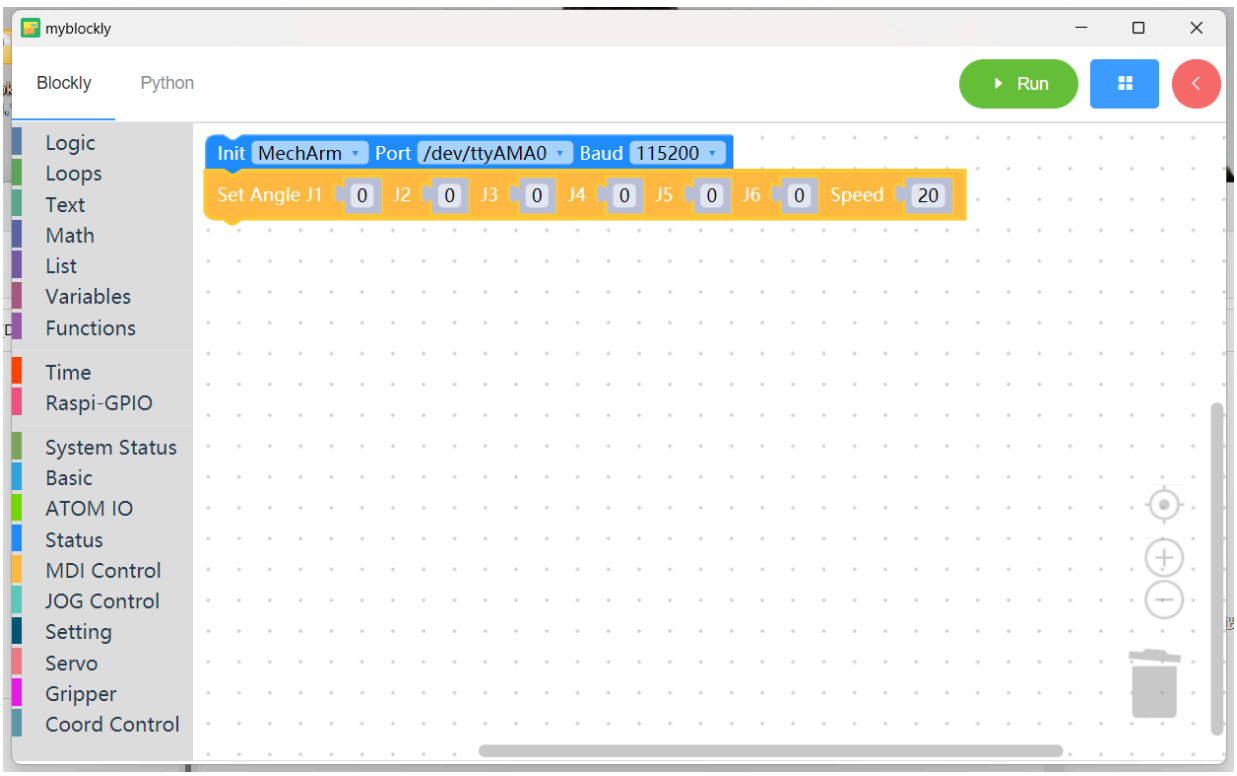
MyArm

Simple demo



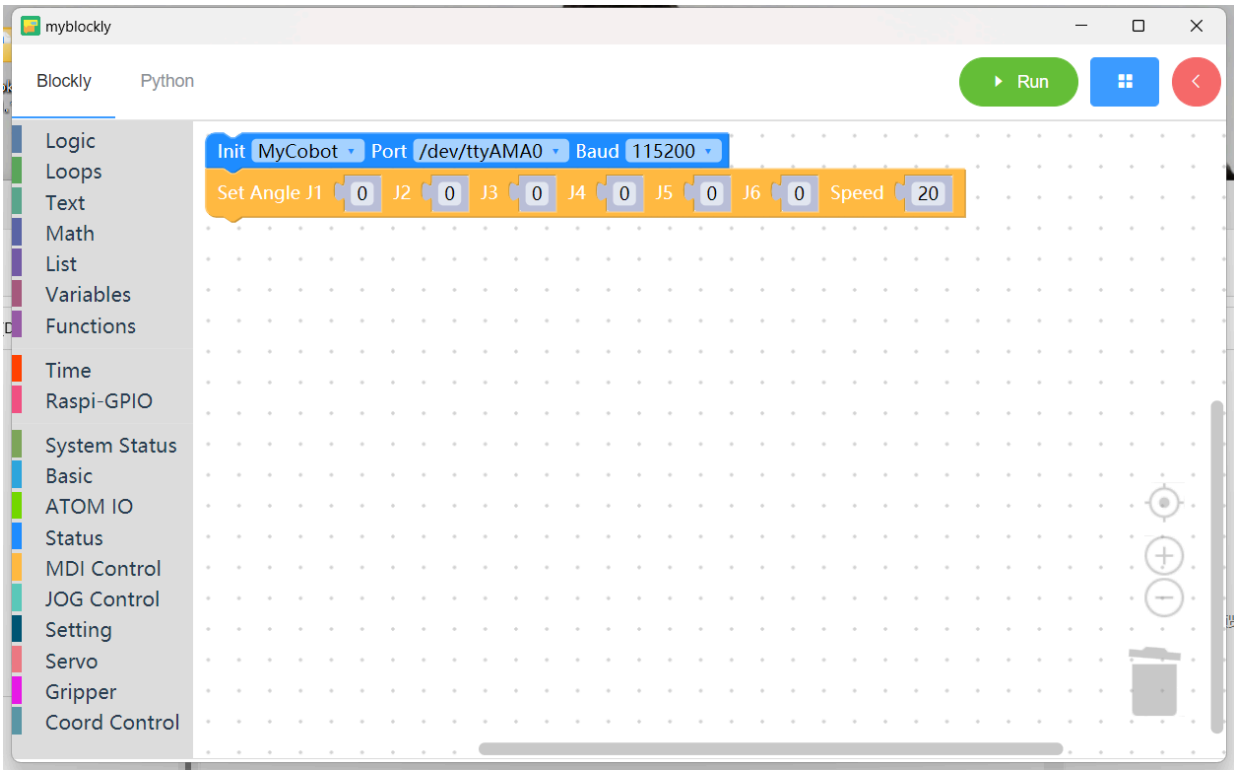
MechArm

Simple demo



MyCobot

Simple demo



- Motion:Six

  All arms move to its starting point in 35 seconds.

Controlling Single-Joint Motion

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

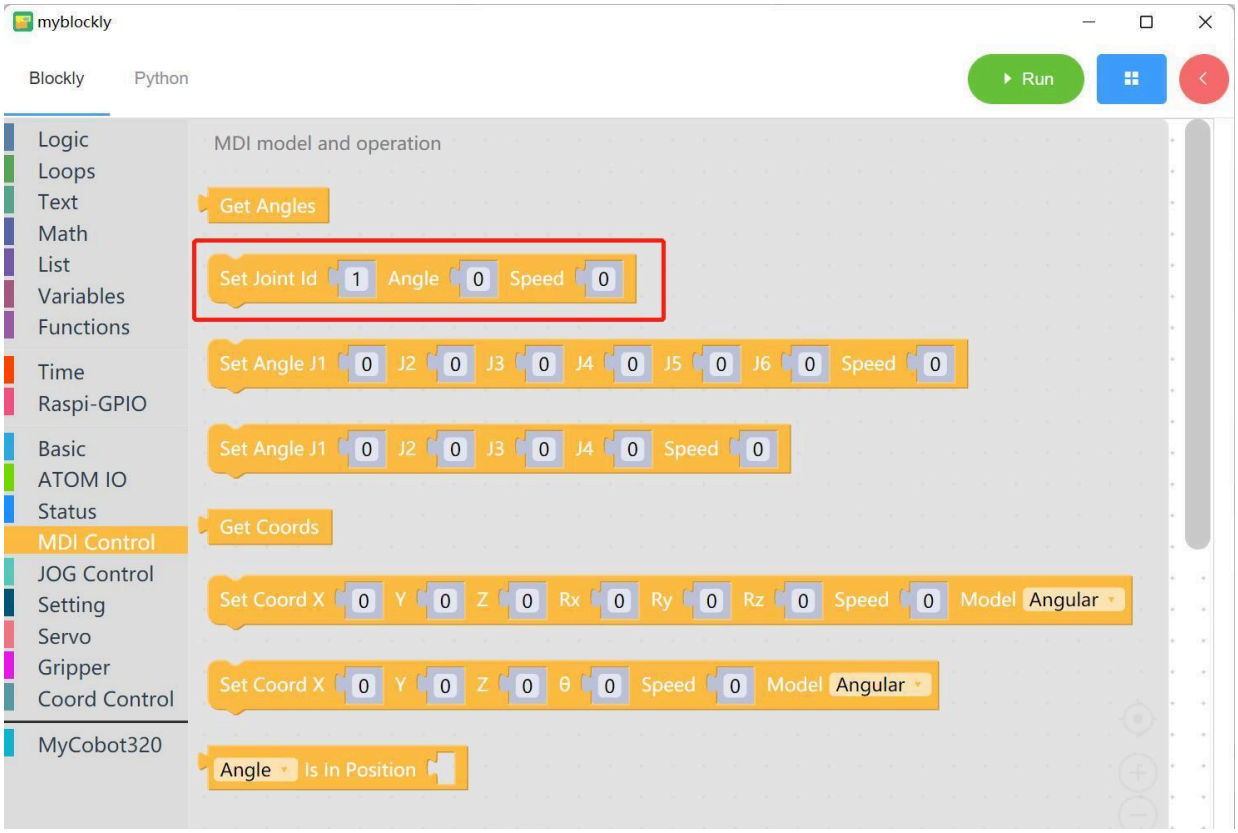
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for controlling single-joint motion.

Introduction to API

- Set Joint Id `x` Angle `x` Speed `x`



- Applicable to myCobot 280 series, myCobot 320 series, myCobot Pro 600 series and myPalletizer 260 series

- Parameters:

  Joint Id range: 1-6 (myCobot 280 series, myCobot 320 series and myCobot Pro 600 series)

  1-4 (myPalletizer 260 series)

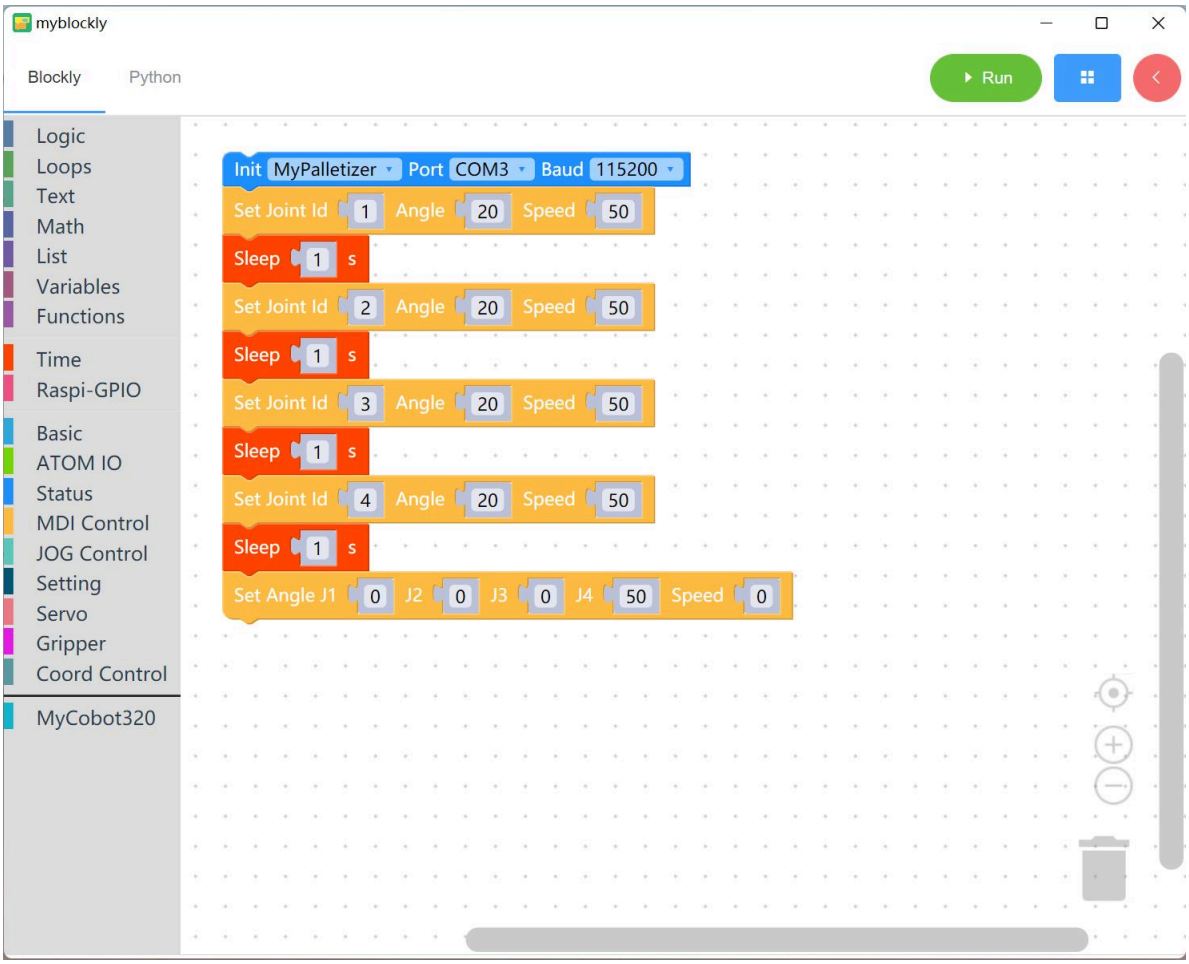
  Angle: Refer to **2 Products Profile**

Speed range: 1-100

- Function: controlling single-joint motion

Simple Demo

- Program for display:



- Motion:

  Joint 1 move to 20 degree angle at the speed of 50,

  Joint 2 move to 20 degree angle at the speed of 50 in 1 second,

  Joint 3 move to 20 degree angle at the speed of 50 in 1 second,

Joint 4 move to 20 degree angle at the speed of 50 in 1 second,

all arms move to starting point in 1 second.

Controlling Multi-Joint Motion

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

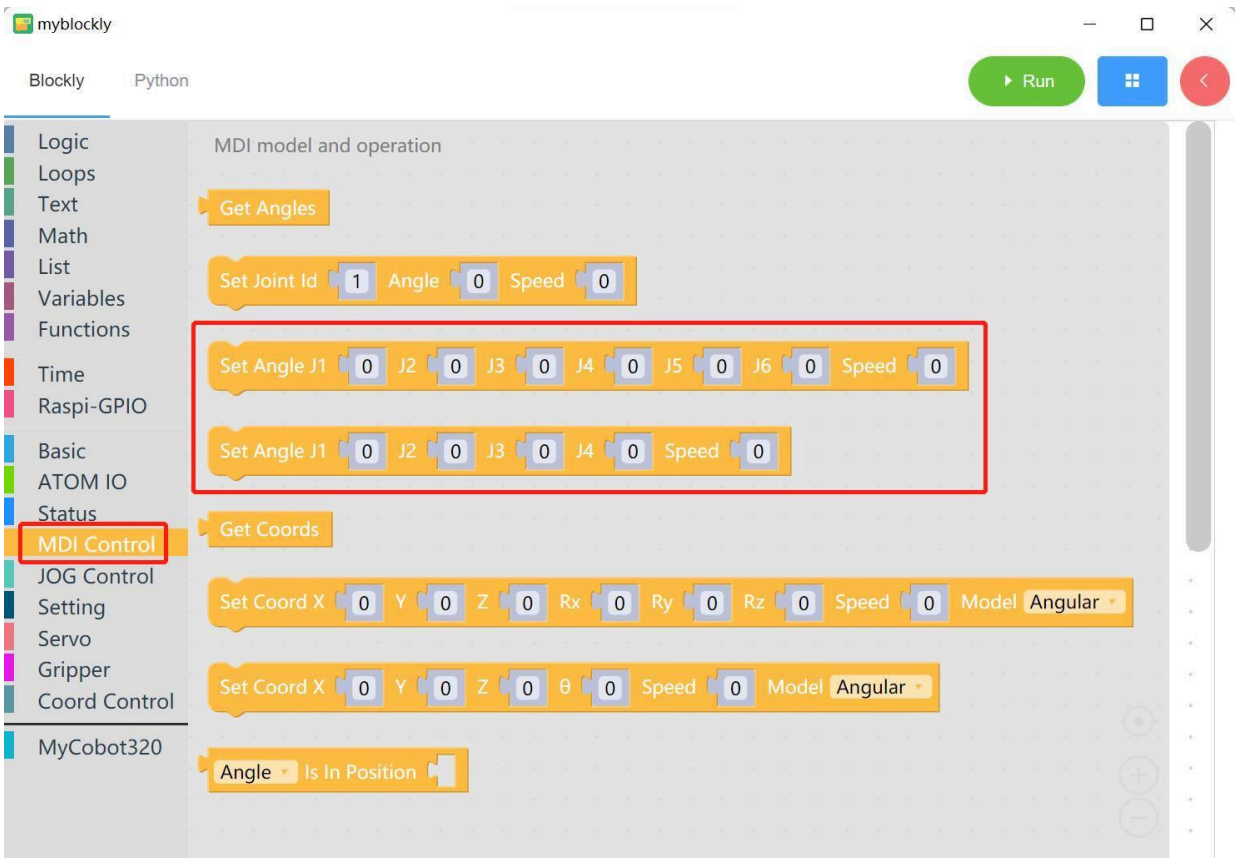
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for controlling multi-joint motion.

Introduction to API

- `Set Angle ()`



- Applicable to six-axis robots (myCobot 280 series, mechArm series and myCobot 320 series)

- Applicable to four-axis robots (myPalletizer series)

- Parameters:

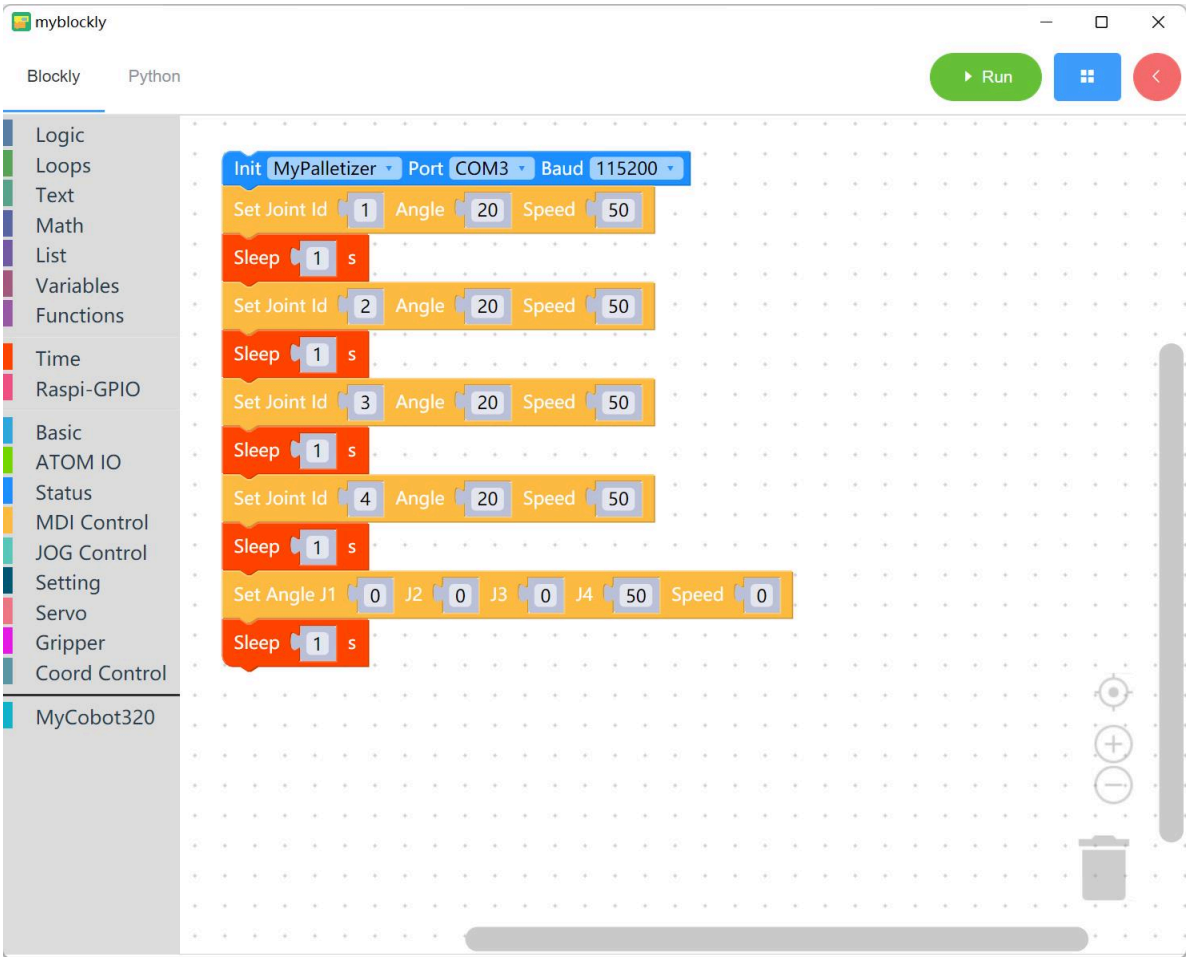
Joint angle: set angles within its due range according to your needs (refer to **2 Products Profile**)

Speed: set speed within its due range according to your needs (refer to **2 Products Profile**)

- Function: multiple joints move to corresponding degree at the preset speed.

Simple Demo

- Program for display:



- Motion:

  Arms move to the starting point,

  after 2 seconds, Joint 1, Joint 2, Joint 3 and Joint 4 move to 30 degree, 30 degree, -30 degree and 50 degree respectively at the speed of 50,

  after 2 seconds, arms move to the starting point at the speed of 50,

  after 2 seconds, Joint 1, Joint 2, Joint 3 and Joint 4 move to -30 degree, 0 degree, 30 degree and -50 degree respectively at the speed of 50,

  program is over in 2 seconds.

Swinging Arms Left and Right

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for swinging arms left and right.

Introduction to API

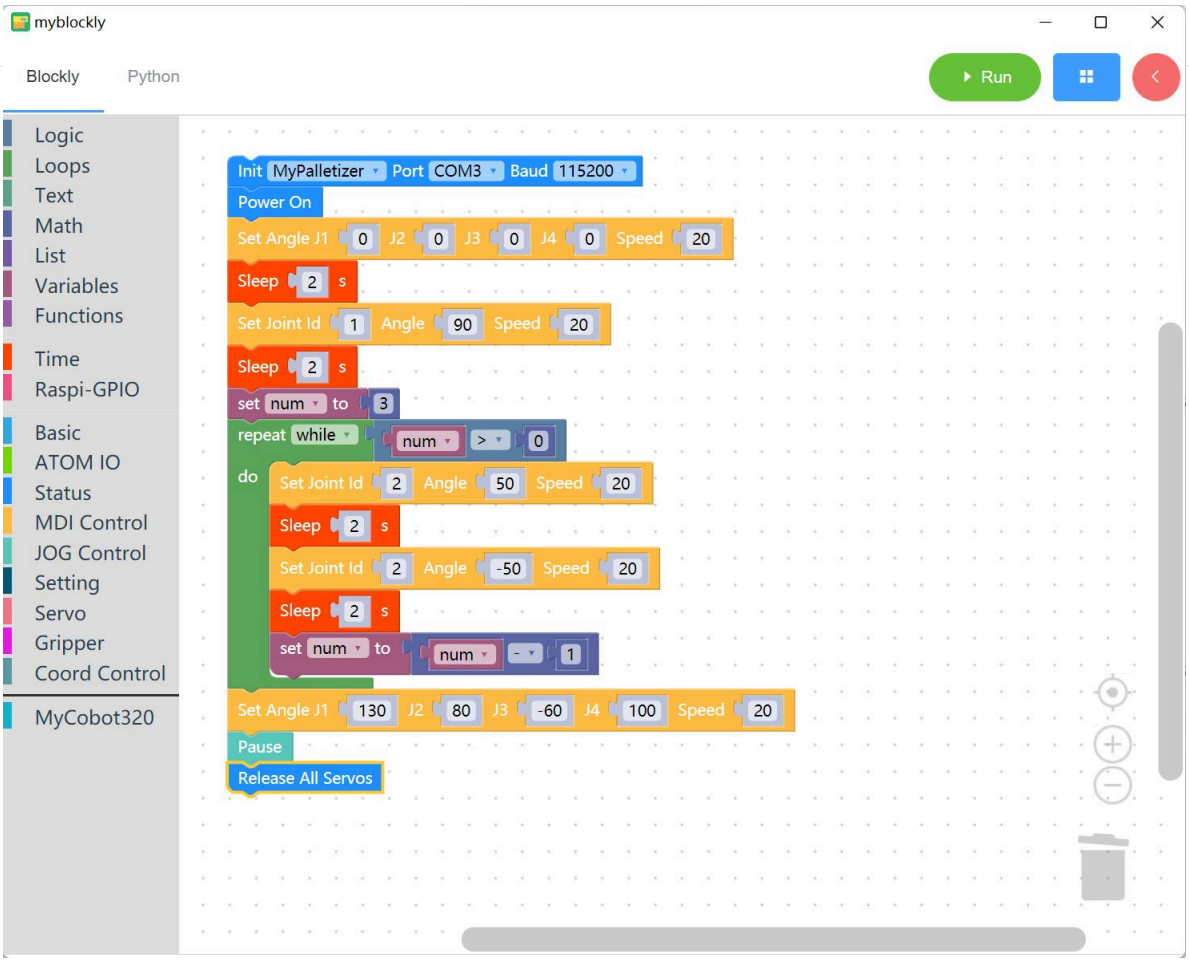
- `power_on()`

- Applicable to myCobot 280 series, mechArm series, myCobot 320 series and myPalletizer series
- Function: initiate system
- `release_all_servos()`

- Applicable to myCobot 280 series, mechArm series, myCobot 320 series and myPalletizer series
- Function: release all servo motors

Simple Demo

- Program for display:



- Motion:

  Initiating system, arms move to starting point, after 2 seconds,

  Joint 1 move to 50 degree at the speed of 20, after 2 seconds,

  Joint 2 move to -50 degree at the speed of 20, after 2 seconds,

  the above process loops again,

  Joint 1, Joint 2, Joint 3 and Joint 4 move to 130 degree, 80 degree, -60 degree and 100 degree respectively.

  Motion stops, and then all servos release.

Let Robot Dance

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

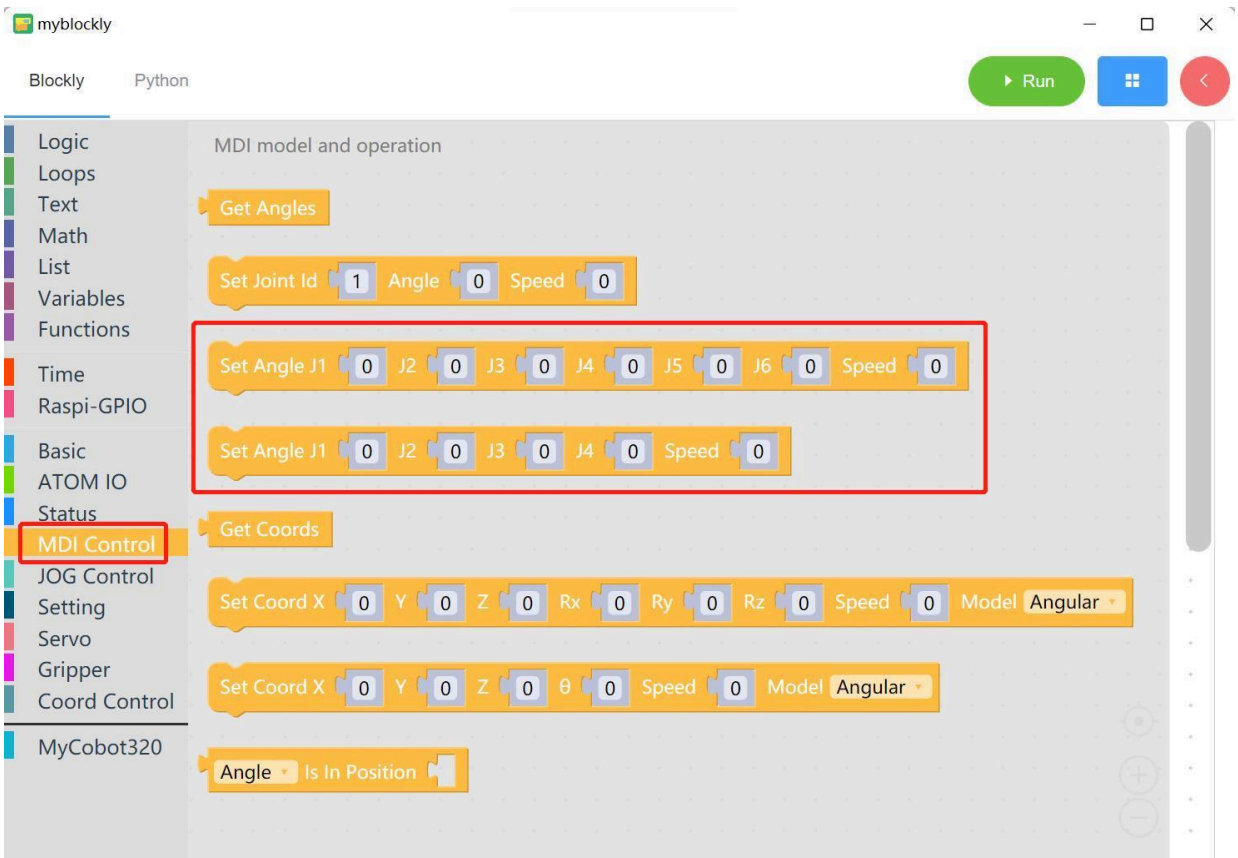
Other series: Make sure the robot is in normal status.

Purpose for this section

This section introduces instructions for swinging arms left and right.

Introduction to API

- `Set Angle`



- Applicable to six-axis robots (myCobot 280 series, mechArm series and myCobot 320 series)

- Applicable to four-axis robots (myPalletizer series)

- Parameters:

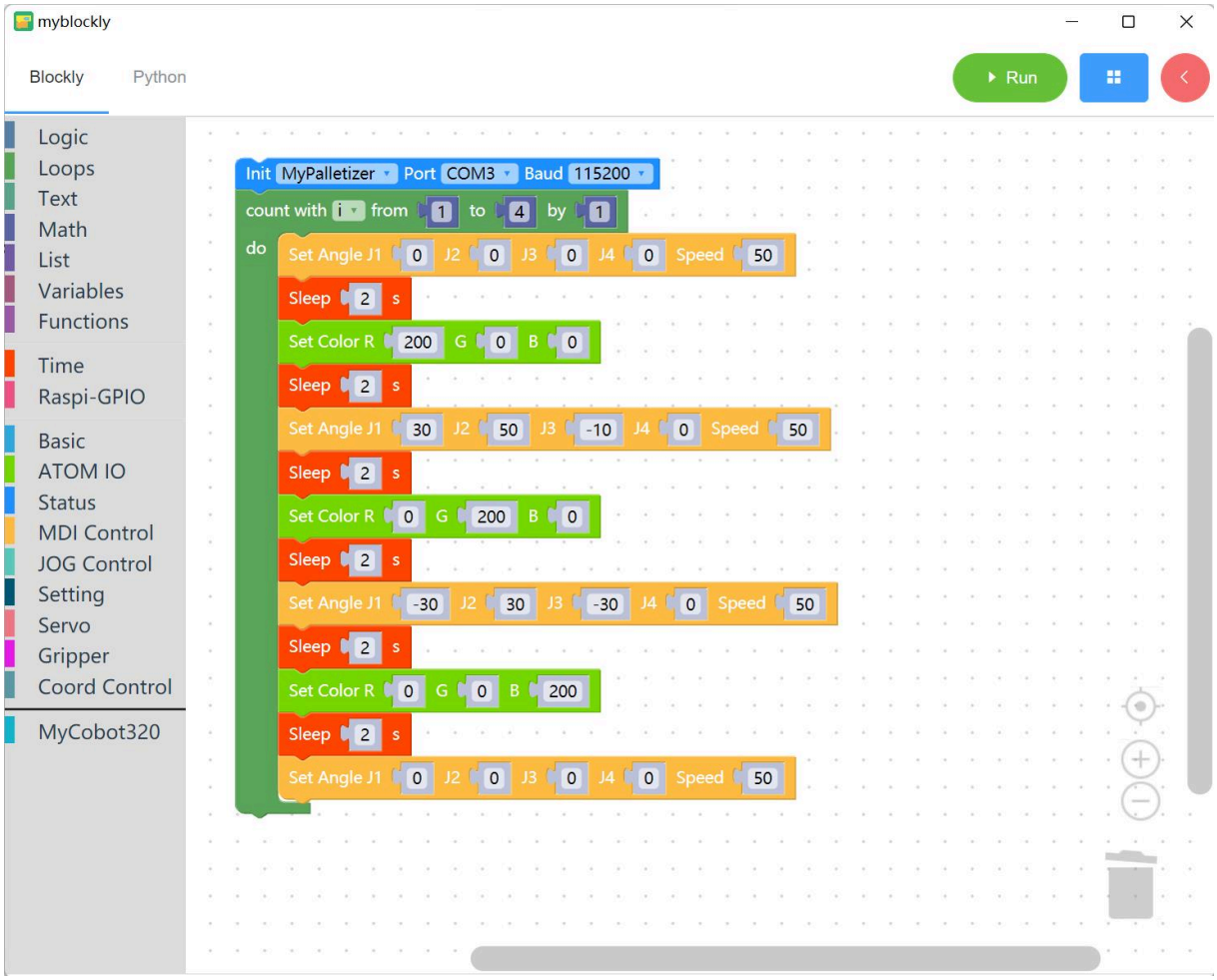
Joint angle: set angles within its due range according to your needs (refer to **2 Products Profile**)

Speed: set speed within its due range according to your needs (refer to **2 Products Profile**)

- Function: multiple joints move to corresponding degree at the preset speed.

Simple Demo

- Program for display:



- Motion:

  All arms move to starting point, RGB light changes to red,

  after 2 seconds, Joint 1, Joint 2 and Joint 3 move to 30 degree, 50 degree and -10 degree at the speed of 50 respectively,

  after 2 seconds, RGB light changes to green,

  after 2 seconds, Joint 1, Joint 2 and Joint 3 move to -30 degree, 30 degree and -30 degree at the speed of 50 respectively,

  after 2 seconds, RGB light changes to blue,

  after 2 seconds, all arms move to starting point.

The Use of Gripper

Preparation

M5Stack series: Make sure the robot is connected with PC (Go to **5.1 myBlockly** for more information).

Other series: Make sure the robot is in normal status.

This section takes myPalletizer 260 M5Stack as an example to explain the use of gripper. Go to **2.8 Accessories** for more information about gripper.

Purpose for this section

This section introduces instructions for using gripper.

Introduction to API

- `set_gripper_ini()`

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Function: Set the current position to zero
- `set_gripper_state(flag, speed)`

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - flag (int): `0` means open, and `1` means close
  - speed (int): range from 0 to 100
- Function: set gripper switch state
- `set_gripper_value(value, speed)`

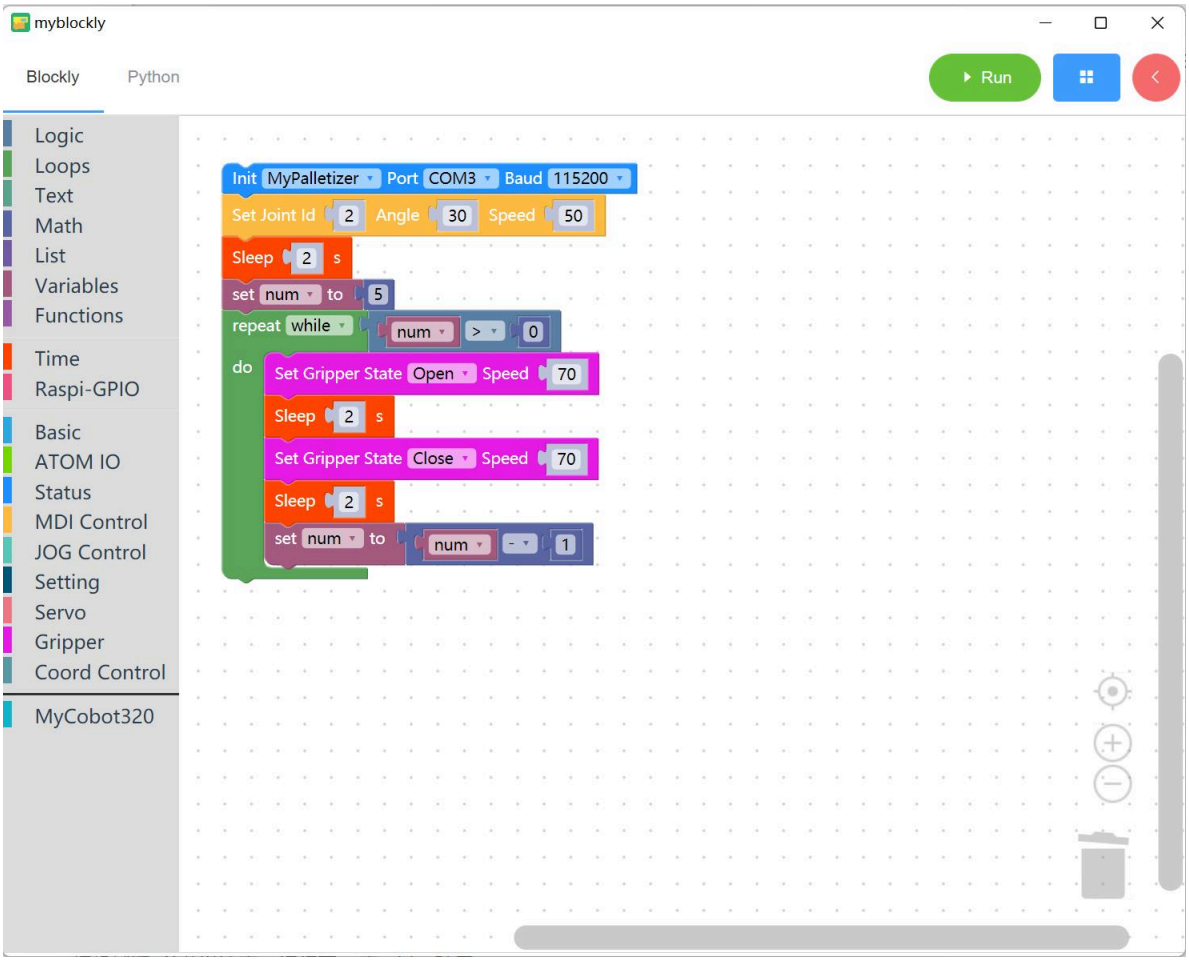
- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - `value()` : range from 0 to 100
  - `speed()` : range from 0 to 100
- Function: set gripper value
- `is_gripper_moving()`

Is Gripper Moving

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Function: Judge whether the gripper is moving or not

Simple Demo

- Program for display



- Motion:

  Joint 2 move to 30 degree at the speed of 50,

  after 2 seconds, gripper opens at the speed of 70,

  after 2 seconds, gripper closes at the speed of 70,

  the process loops 5 times.

The Use of Sucking Pump

Preparation

M5Stack series: Make sure robot is connected with PC (Go to **5.1 myBlockly** for more information).

Other series: Make sure the robot is in normal status.

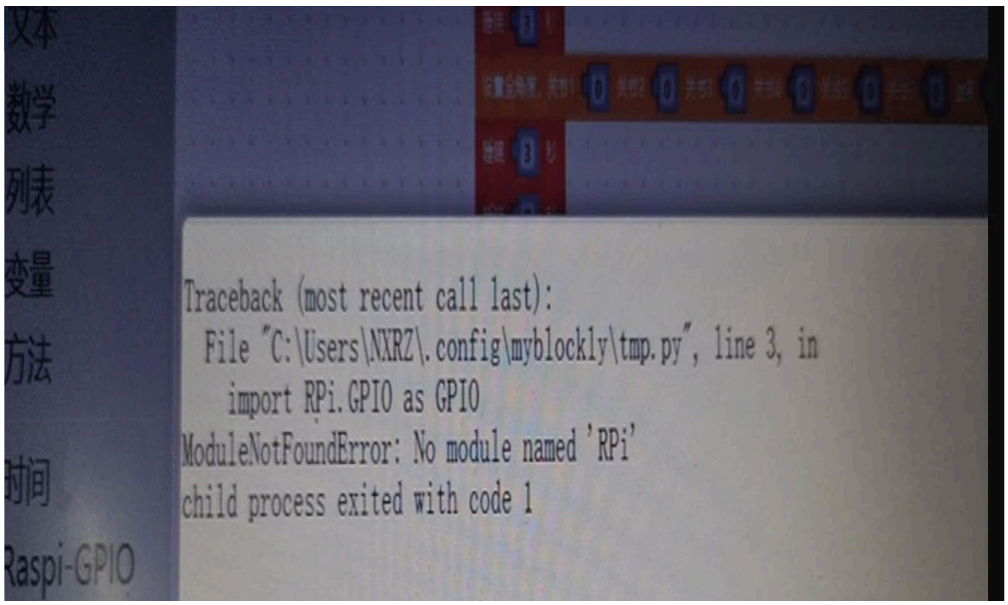
This section takes myPalletizer 260 M5Stack as an example to explain the use of suction pump. Go to **2.8 Accessories** for more information about sucking pump.

Purpose for this section

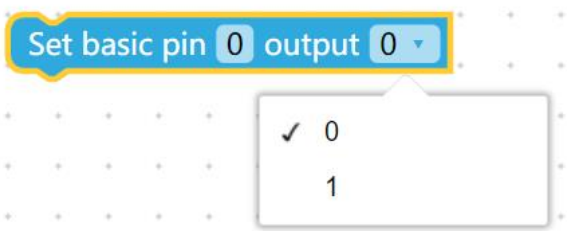
This section introduces instructions for using sucking pump.

Introduction to API

> **Notice:** M5Stack version is unable to use the blocks belonging to Raspberry Pi version. Otherwise, the system may report an error.



- `Set Basic pin () output ()` (for M5Stack version)



- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - `pin ()` : the numerical part of the numbers marked at the bottom of the equipment
  - `output ()` : `0` means setting to the running state, and `1` means setting to the stop state

- Function: set the working state of the preset bottom pin

- `get_basic_input(pin_no)` (for M5Stack version)

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - `pin()` : the numerical part of the numbers marked at the bottom of the equipment
- Function: get the working state of the bottom pin number

- `set mode ()` (for Raspberry Pi version)

- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - `mode()` : "BCM" or "BOARD"
- Function: set Raspberry Pi GPIO Pin Mode

- `set pin() mode()` (for Raspberry Pi version)

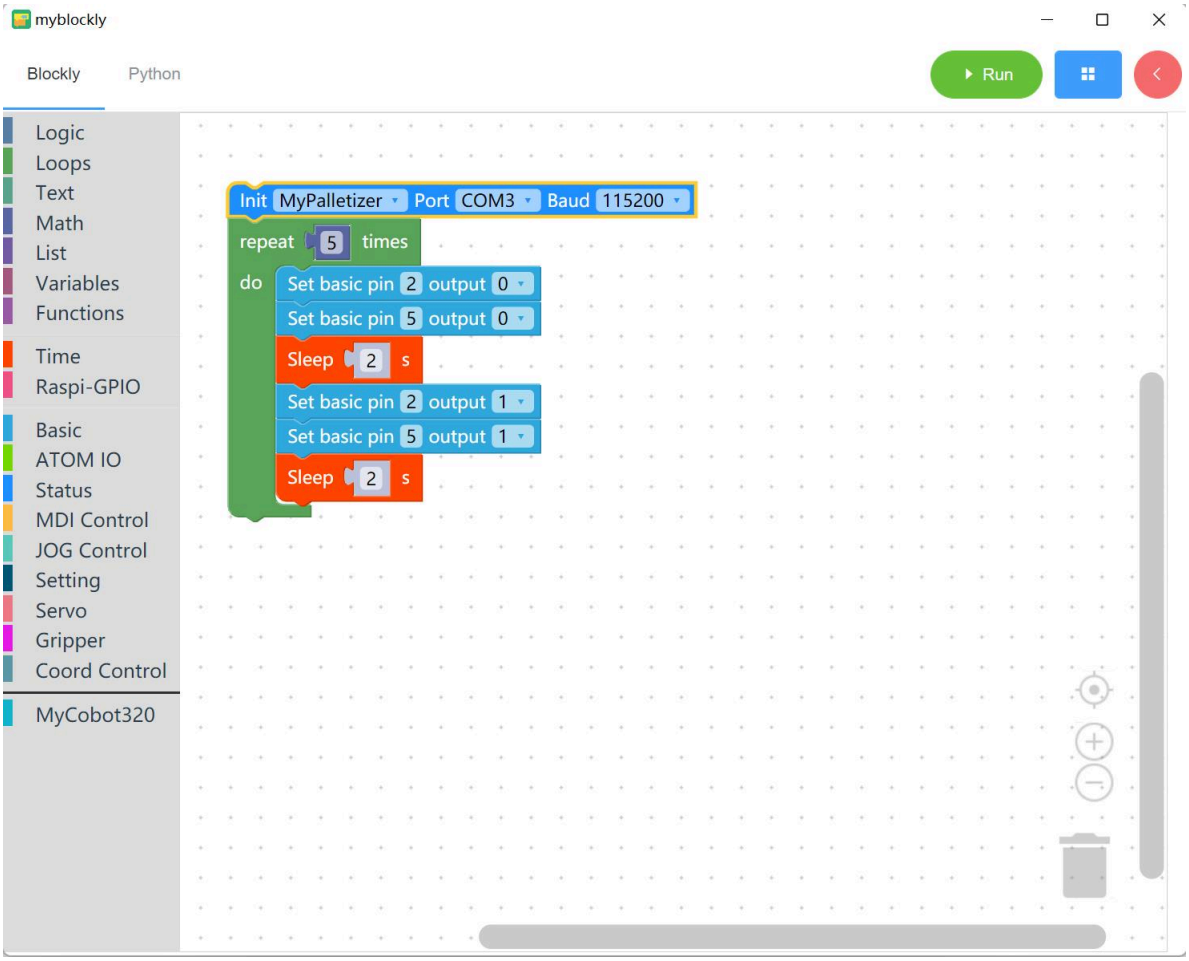
- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - `pin ()` : the numerical part of the numbers marked at the bottom of the equipment
  - `mode ()` : `in` means signal import, and `OUT` means signal output
- Function: set signal of the preset bottom pin

- `set pin() output()` (for Raspberry Pi version)

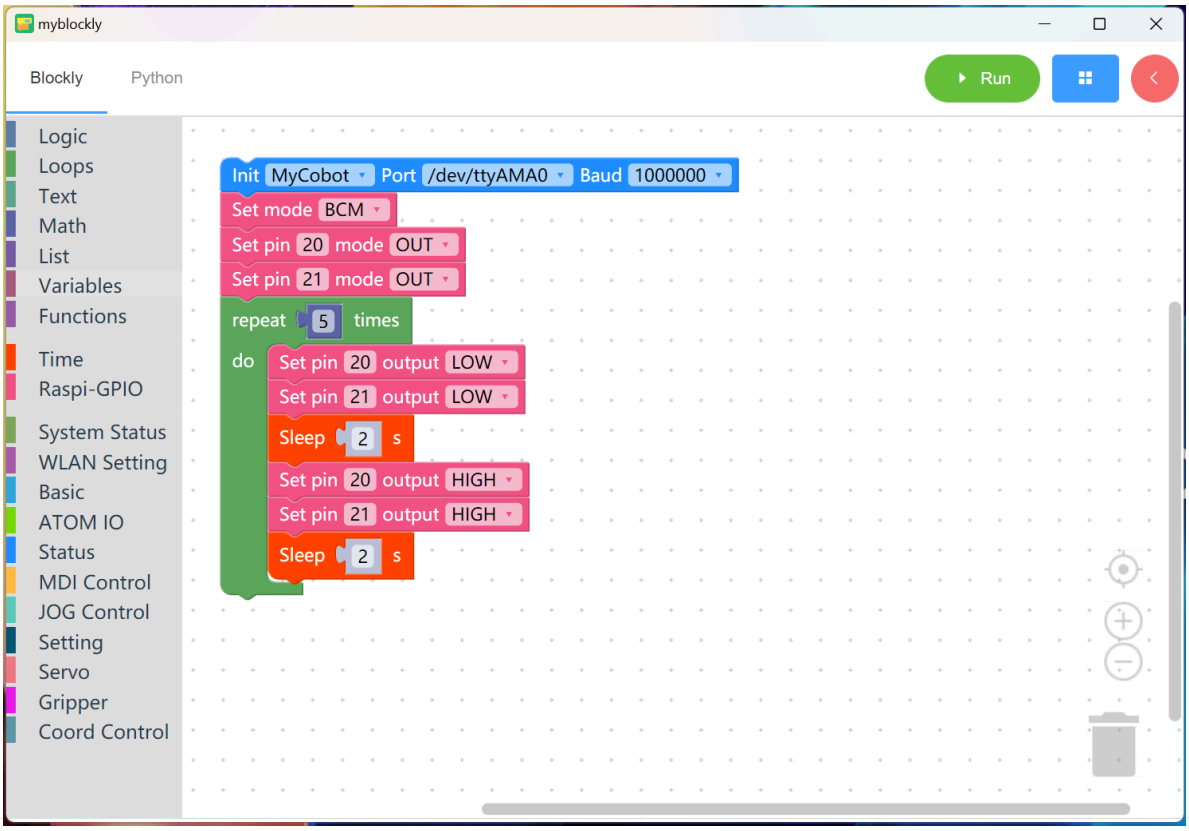
- Applicable to myCobot 280 series, mechArm 270 series, and myPalletizer 260 series
- Parameter:
  - `pin()` : the numerical part of the numbers marked at the bottom of the equipment
  - `output()` : `HIGH` means high level of sucking pump working state, and `LOW` means low level of sucking pump working state
- Function: set the working state of bottom pin to high level or low level

Simple Demo

- Program for display: **Vertical suction pump V1.0** *The graphic code is as follows: (for M5 version)*



The graphic code is as follows: (for Raspberry Pi version)



- Motion:

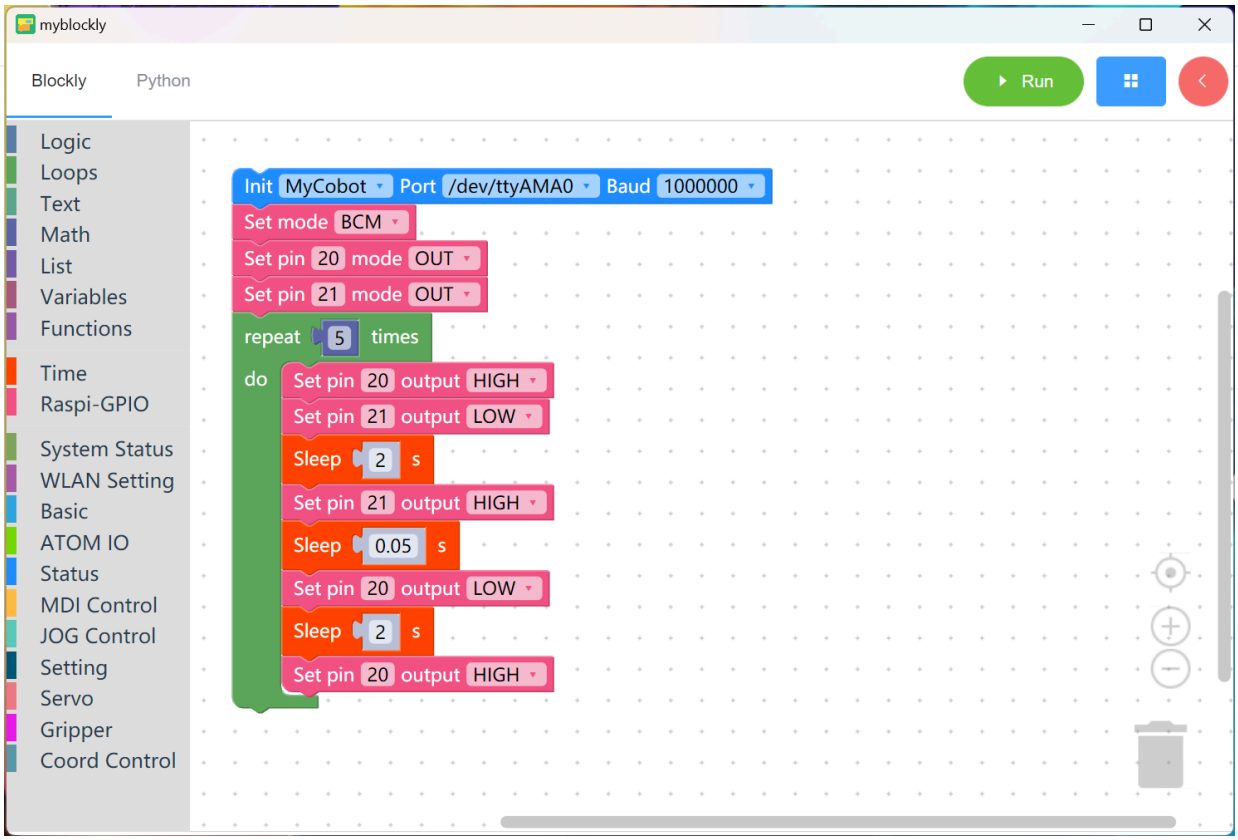
  Sucking pump vibrates and work to suck objects,

  after 2 seconds, sucking pump put objects down,

  after 2 seconds, it vibrates again. The whole process loops 5 times.

Vertical suction pump V2.0 *The graphic code is as follows: (for M5 version)*

The graphic code is as follows: (for Raspberry Pi version)



- Motion: First, close the solenoid valve and open the air release valve to prepare for starting the suction pump; After two seconds, close the vent valve. After 0.05 seconds, open the solenoid valve and the suction pump will vibrate, starting to work. The suction pump will pick up an object and after two seconds, put it down. Close the air release valve and repeat the previous action until the program is completed.

Gripper Test

Before you start

M5Stack series: Make sure the robot is connected to the computer (for details, please refer to **5.1myBlockly**)

Others: Make sure the machine is normal

To use the gripper test function, myblockly needs to be updated to v1.5.5

Learning content in this chapter

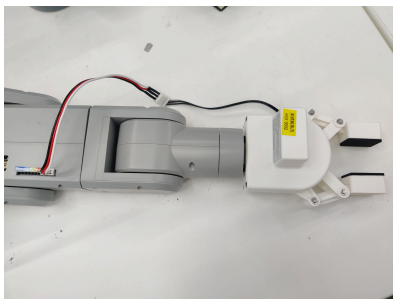
How to use myBlockly to quickly test whether the gripper is normal.

Currently, the gripper test function is only compatible with the following models: myCobot 280 series, mecharm 270 series, myPalletizer 260 series, myArm 300 Pi.

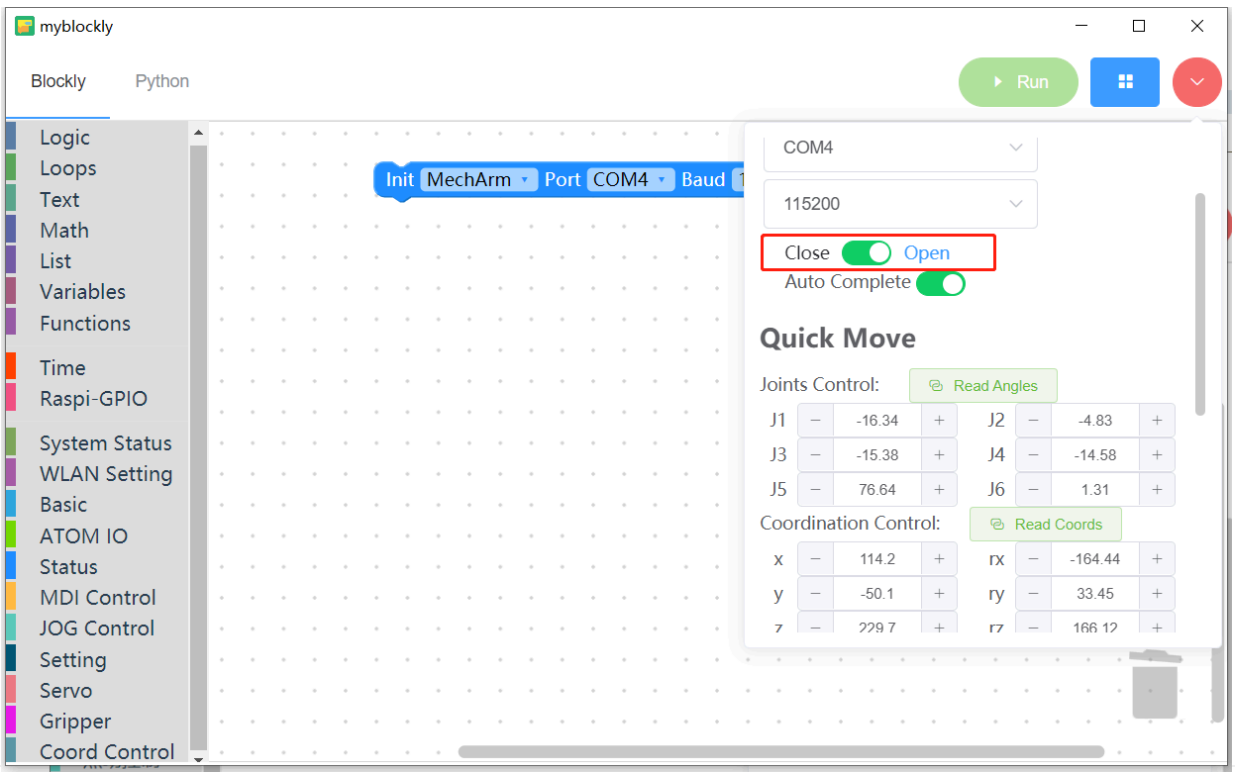
The grippers include adaptive grippers, electric grippers, and pneumatic grippers. Different grippers are compatible with different robot arm models. For details, please refer to **2.7 Accessories**.

Here, mecharm 270 M5 and adaptive grippers are taken as examples.

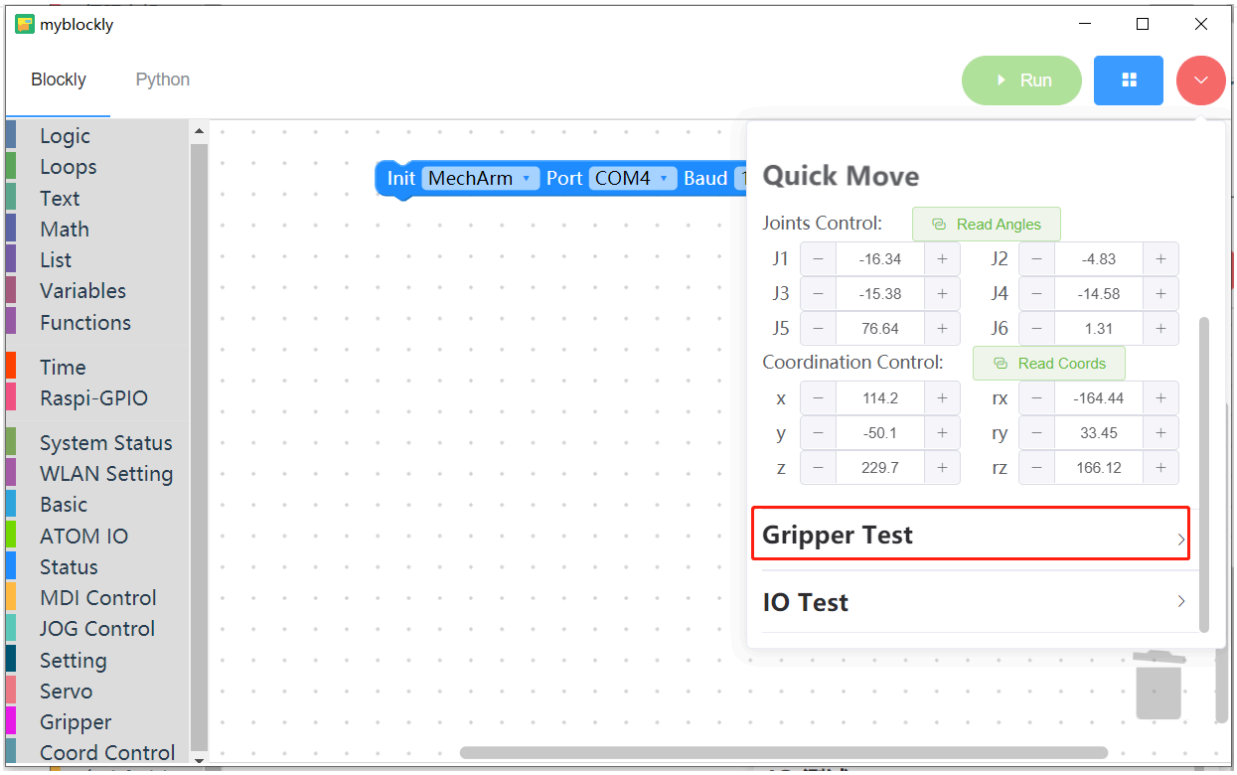
First install the gripper on the machine



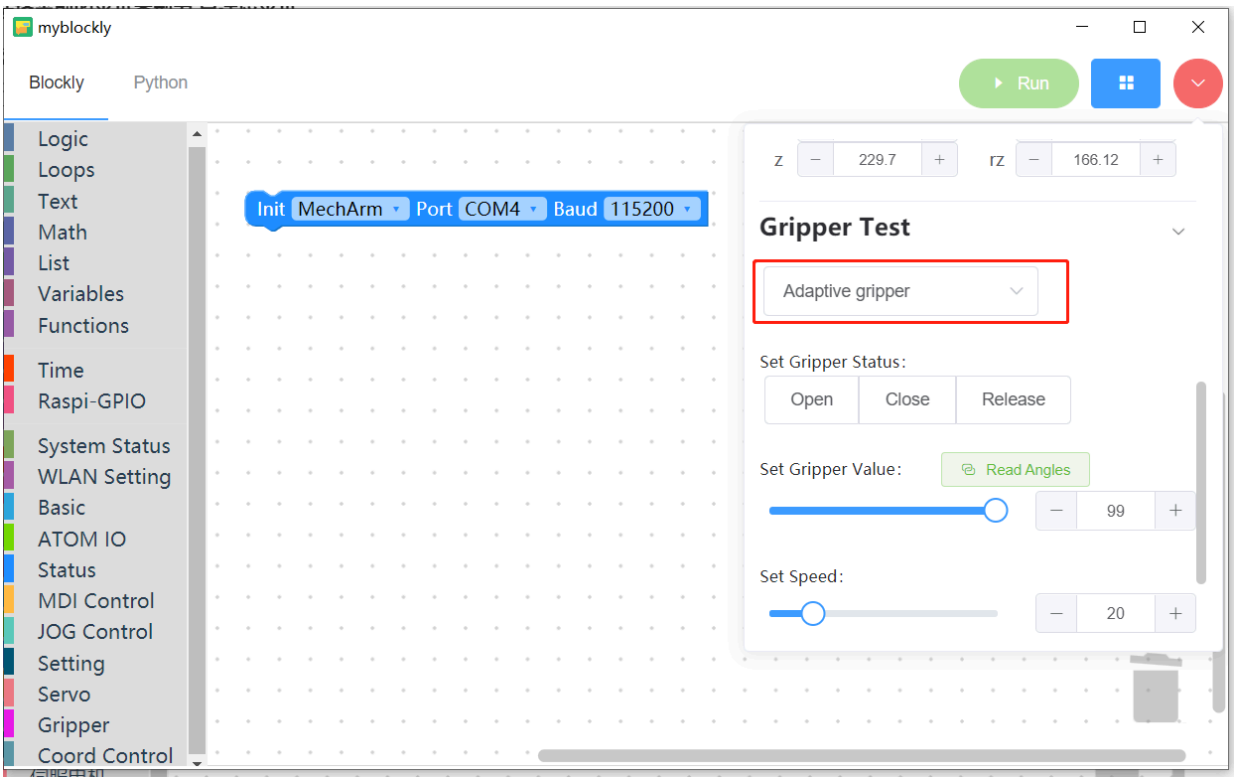
myblockly connects to the machine serial port



Open Gripper Test



Select the current gripper type as Adaptive gripper



Then you can try the following:

- Set the gripper state: Open, Closed, Released. Pay attention to the gripper movement.
- Click the Read Angles button to get the current gripper value.
- You can fine-tune the gripper value.
- You can set the gripper movement speed

> Tips: When the gripper type is Dexterous Hand, the gripper value cannot be set, only its status can be set.

IO test

Preparation before starting

M5Stack series: Make sure that the robotic arm is connected to the computer (please refer to **[5.1Myblockly] (https://docs.elephantrobotics.com/docs/gitbook/5-proGramingApplication-Myblockly- UIFL-UIFL OW-MIND/5.1-Myblockly/)**)

Others: Make sure the machine is normal

To use the IO test function, myblockly needs to be updated to V1.5.5

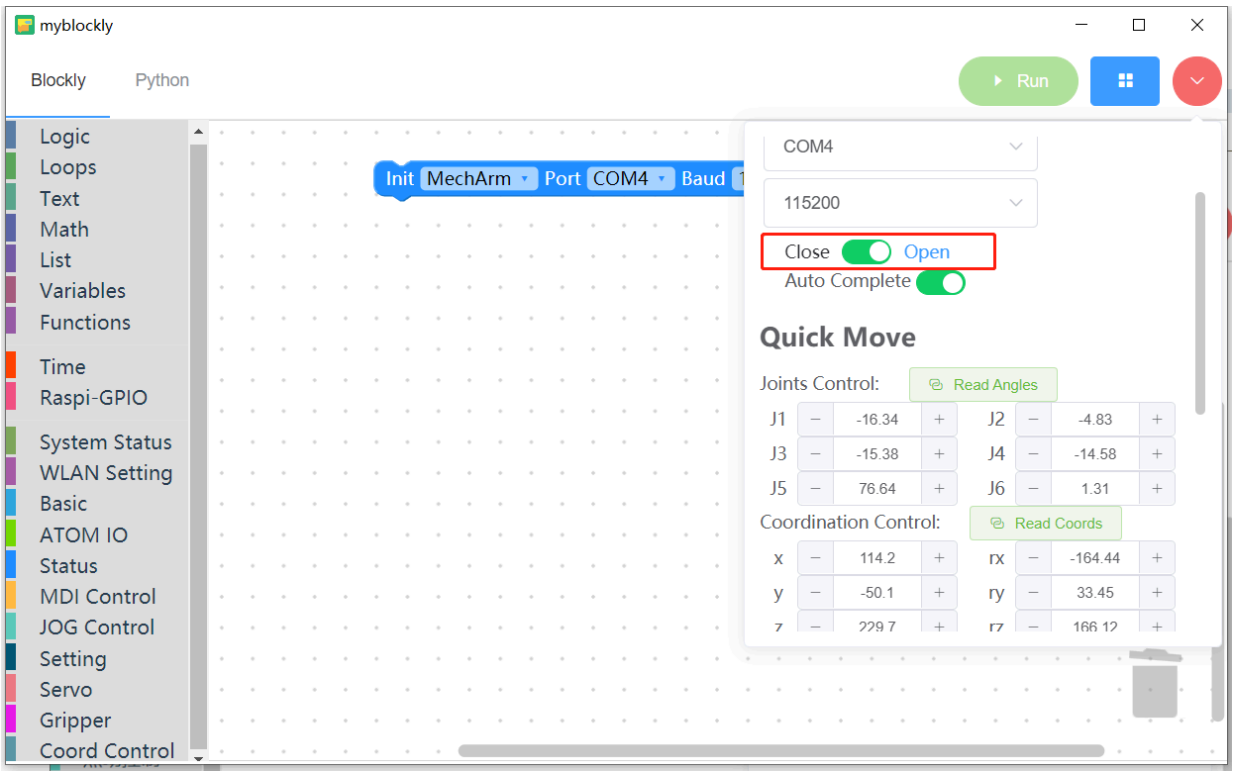
Learning content in this chapter

How to quickly test whether IO is normal to use myblockly.

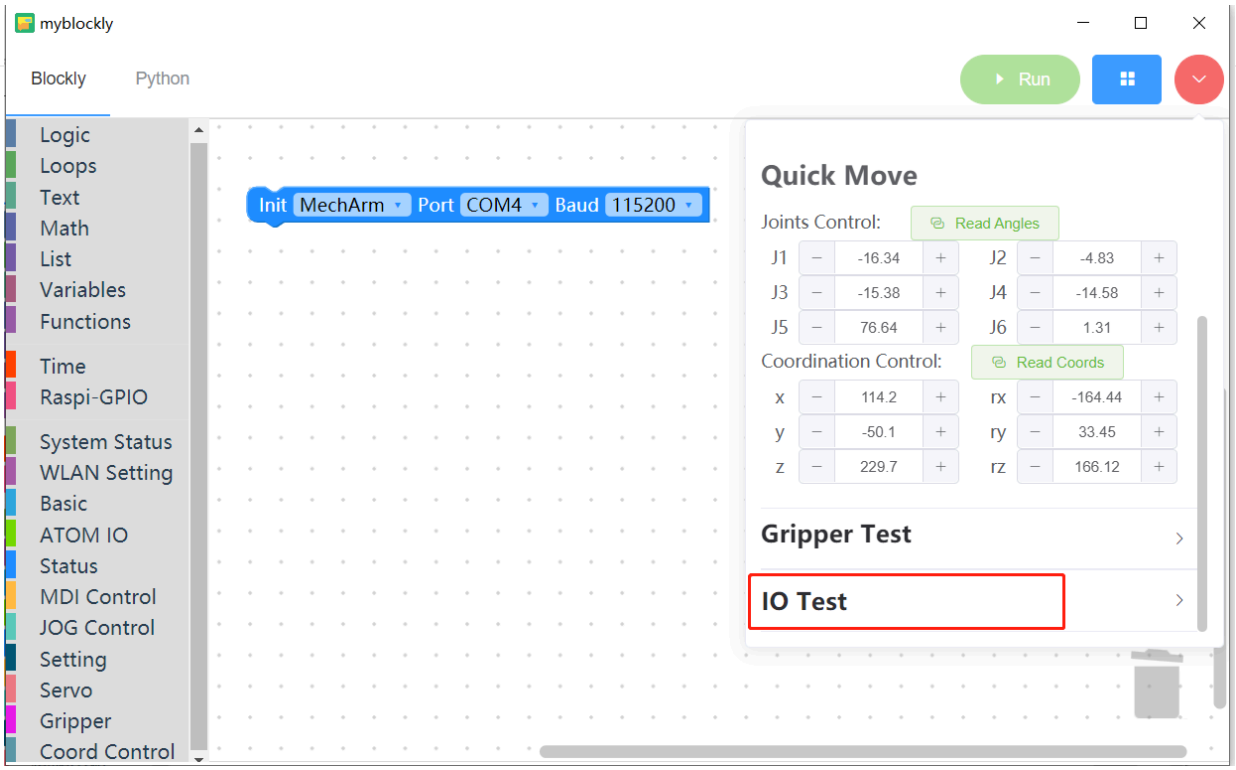
At present, the io test function only adapts to the following models: MyCobot 280 series, MeCharm 270 series, MyPalletizer 260 series, MyARM 300 PI.

Take Mecharm 270 M5 as an example.

myblockly connect the machine serial port

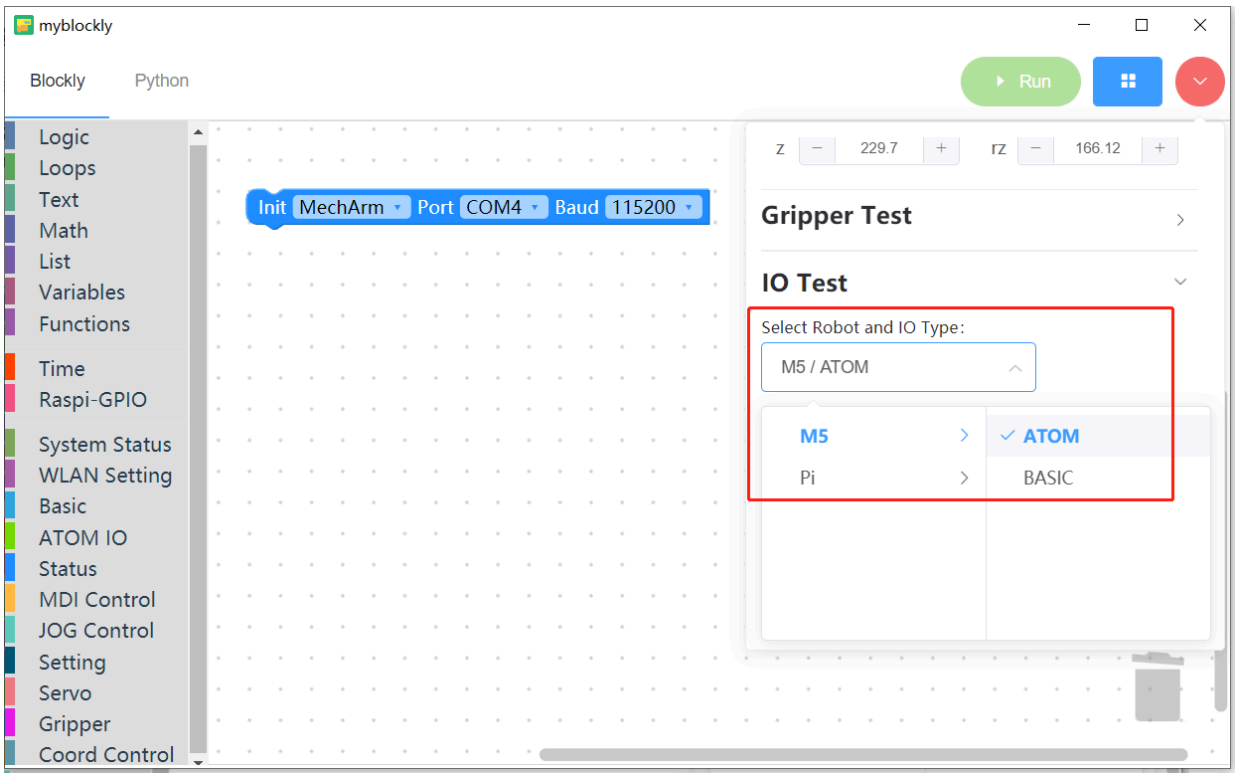


Open IO Test



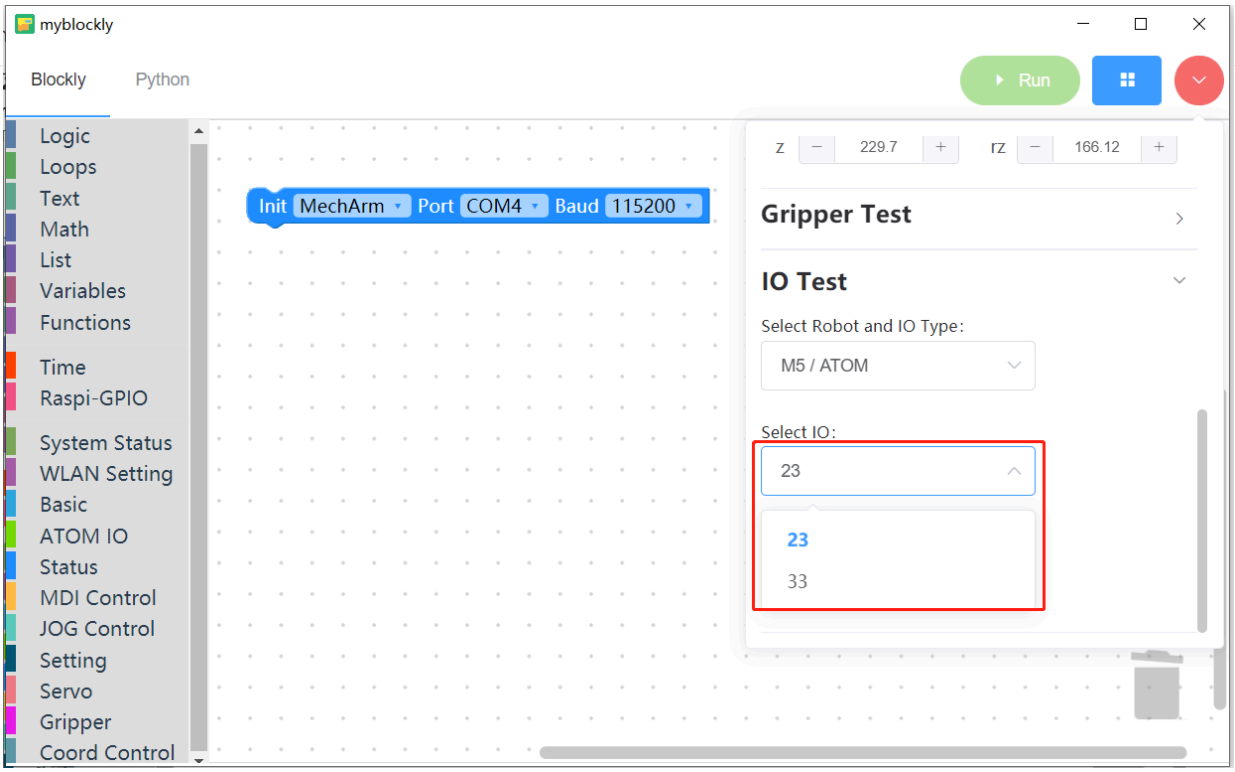
Select IO type

Our current models are M5 models, so choose the M5 type. ATOM and BASIC in the M5 classification can be selected. Here we choose atom



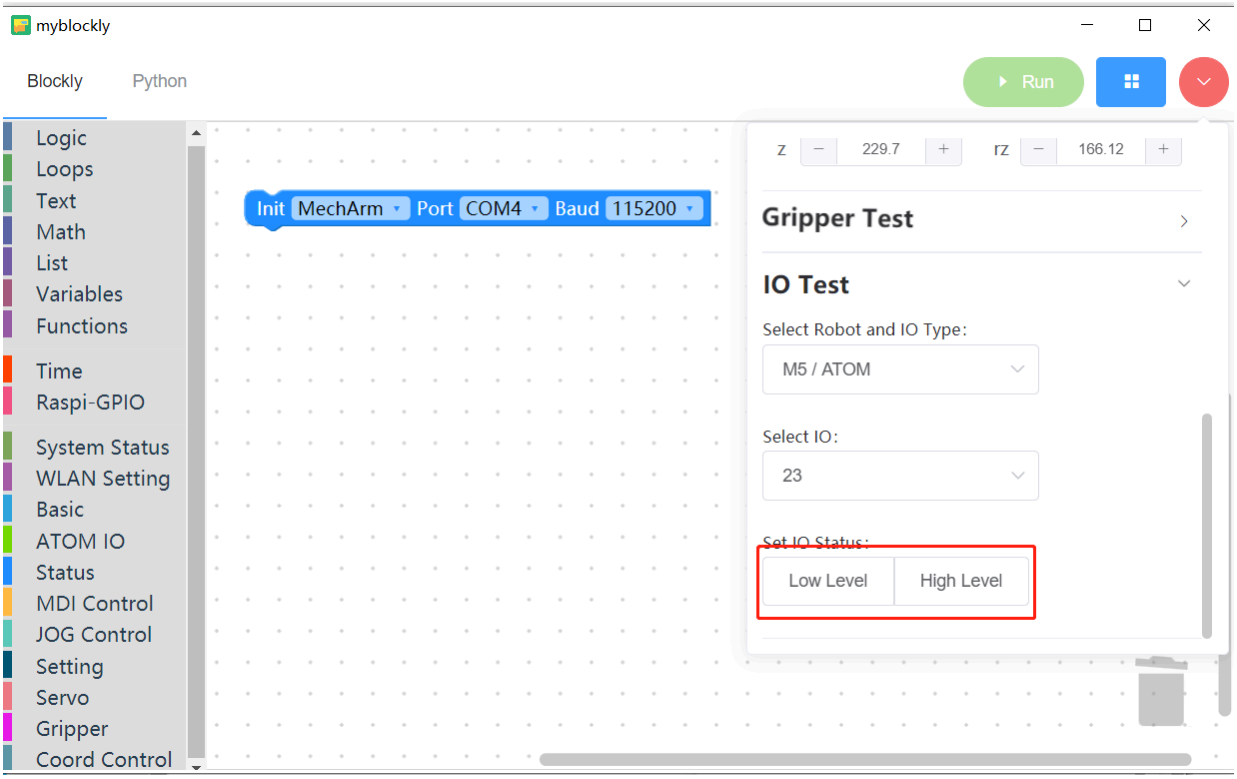
Select ATOM IO

meacharm 270 m5 Atom IO port is currently only 23 and 33 open

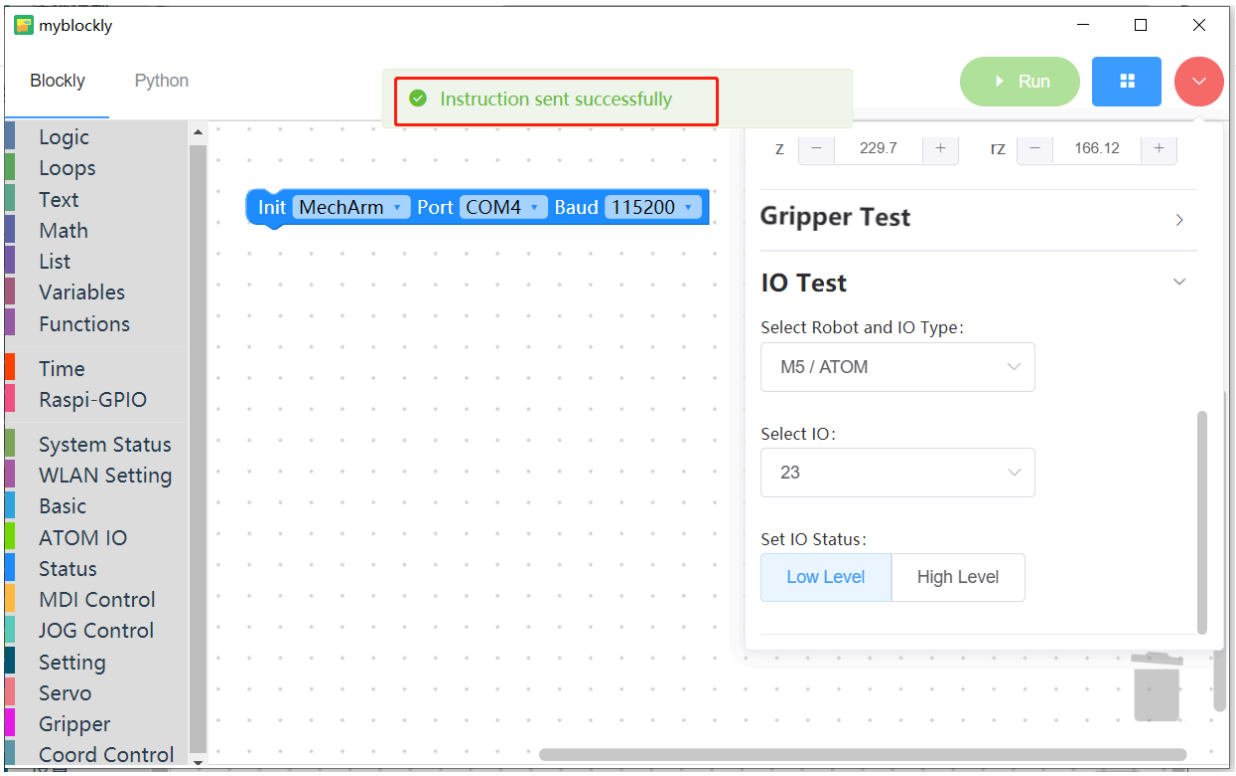


Change IO Status

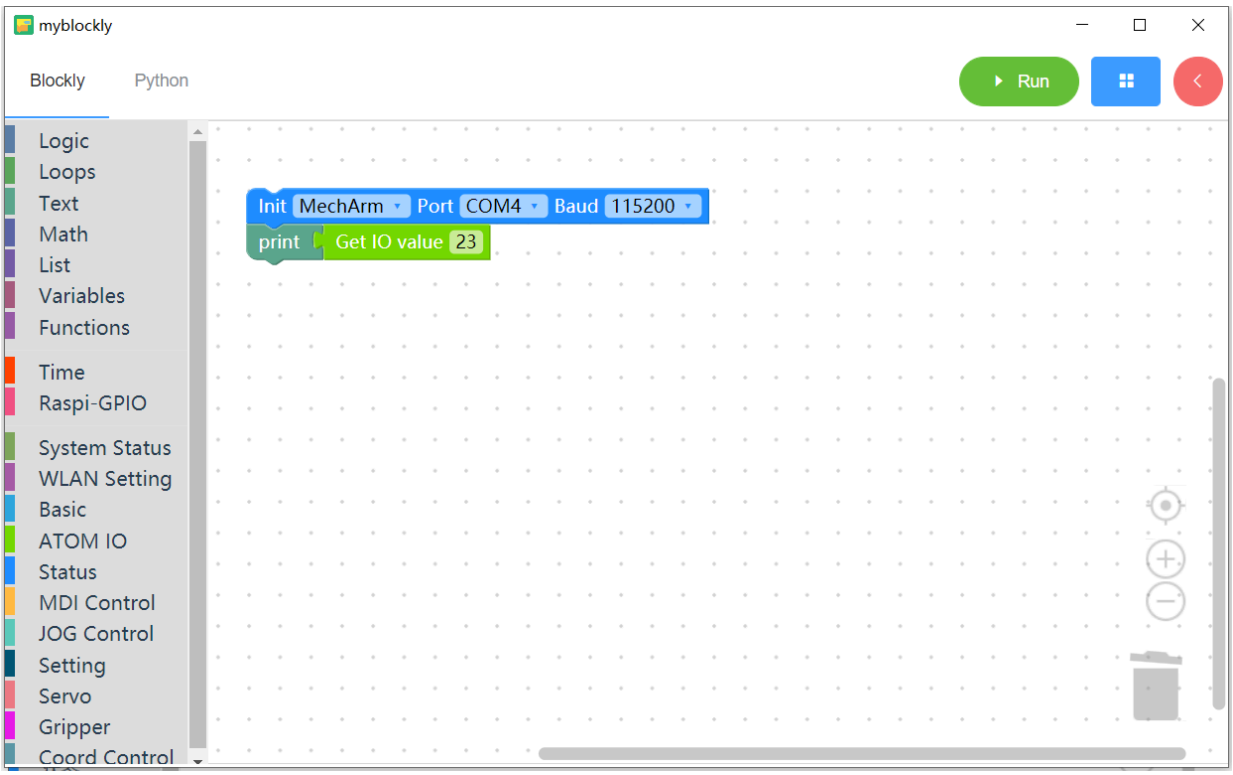
You can click the `Low Level` `High Level` button to change the current selected IO port status



Set the high and low level instructions to send successfully



After changing the IO status, you can read the atom IO level through the building block to see if it is successful



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)

Environment Building

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

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

1 Download and Installation of Python

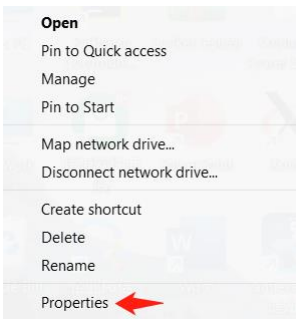
Python is applicable to:

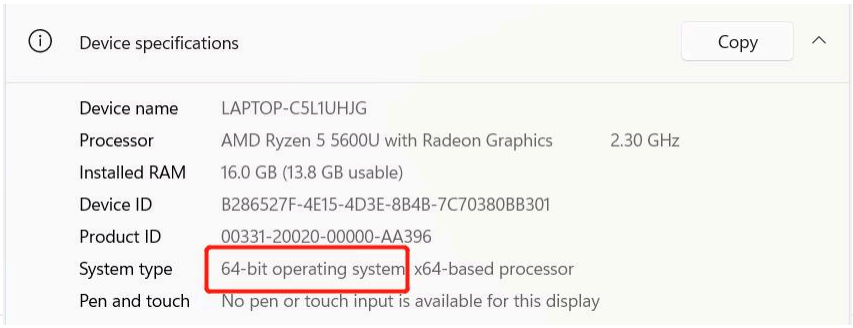
- myCobot 280：
  - myCobot 280 M5
  - myCobot 280 PI
  - myCobot 280 Jetson Nano
  - myCobot 280 for Arduino
- myCobot 320：
  - myCobot 320 M5
  - myCobot 320 PI
- myPalletizer 260：
  - myPalletizer 260 M5
  - myPalletizer 260 PI
- mechArm-270：
  - mechArm-270 M5
  - mechArm-270 PI

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

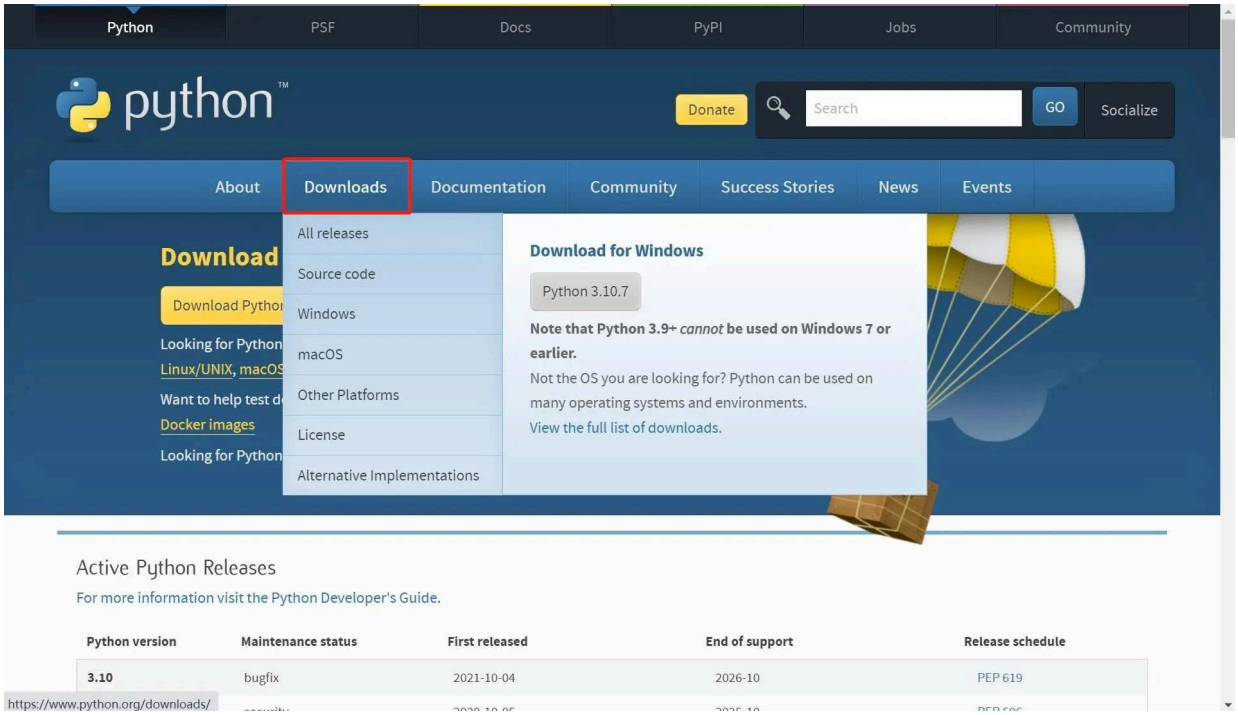
1.1 Installing Python

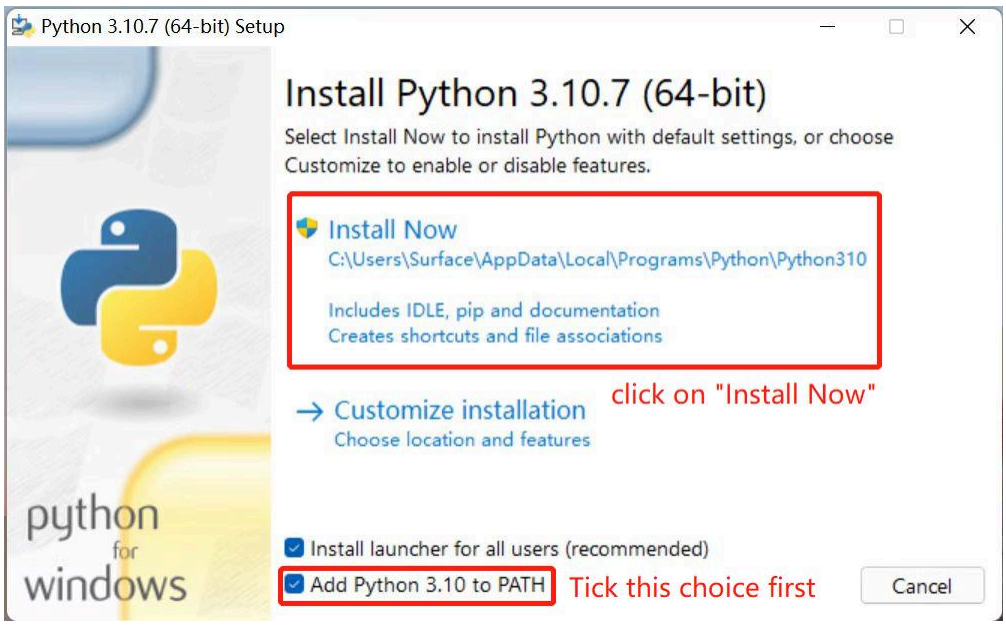
> **Notice:** Before installation, check the operation system of PC. Press right button on the `My Computer` icon and then select `Properties`. Install the corresponding Python.

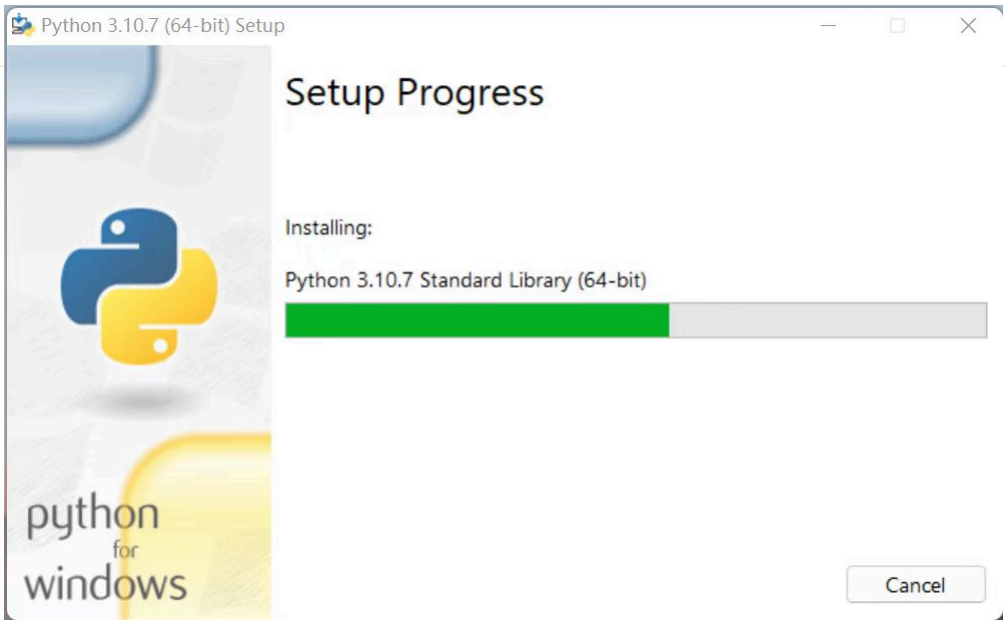




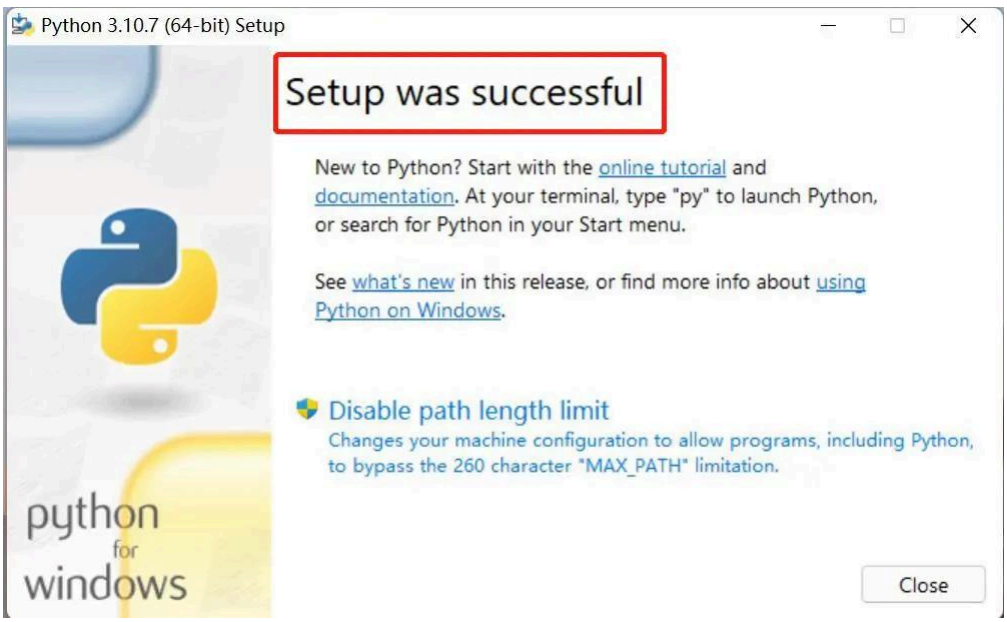
- **Go to http://www.python.org/download/ to download Python.**
- **Click on `Downloads`, and then download begins. Tick `Add Python 3.10 to PATH`. Click on `Install Now`, and then installation begins.**







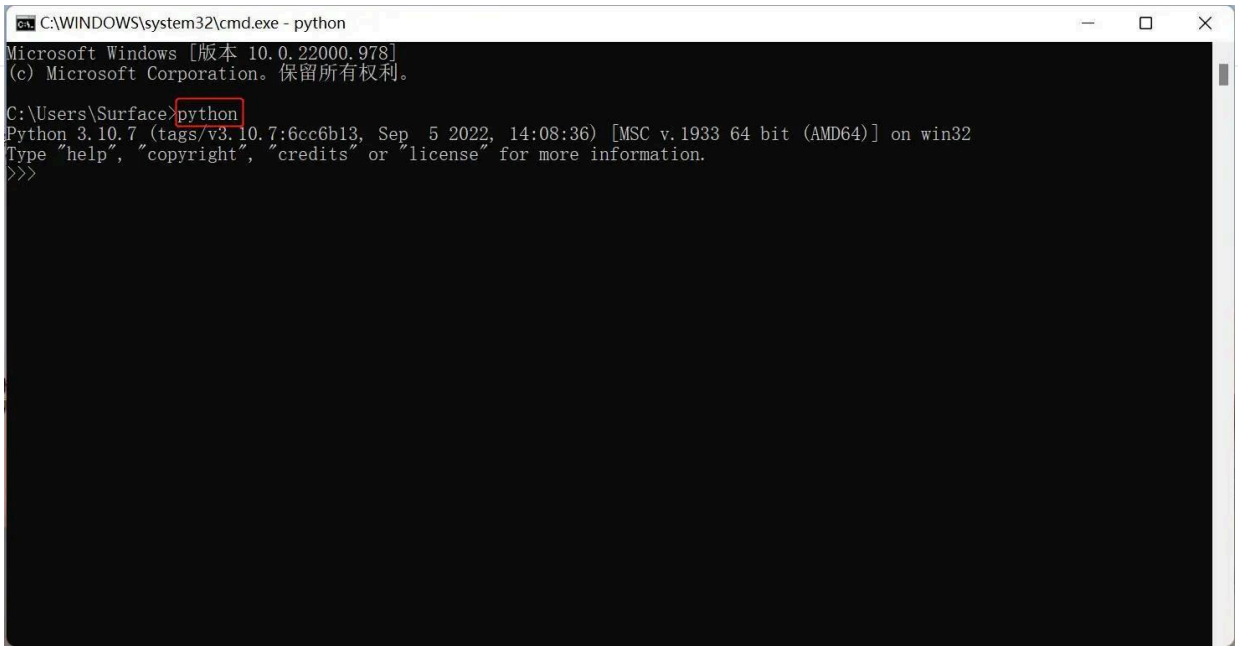
- **Download and installation complete.**



1.2 Running Python

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

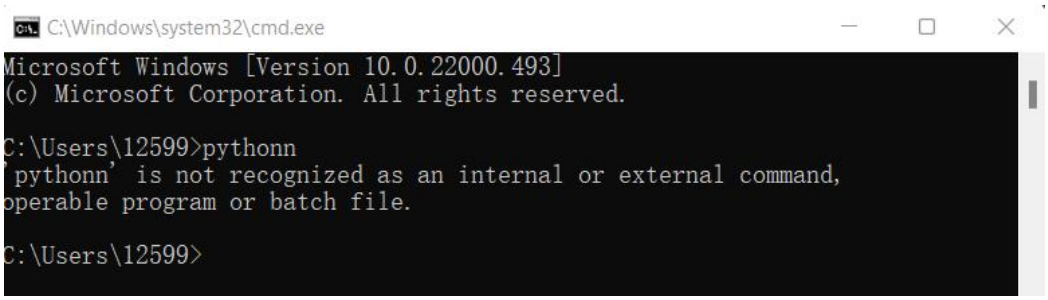
Successful Installation:



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

Error Report:

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



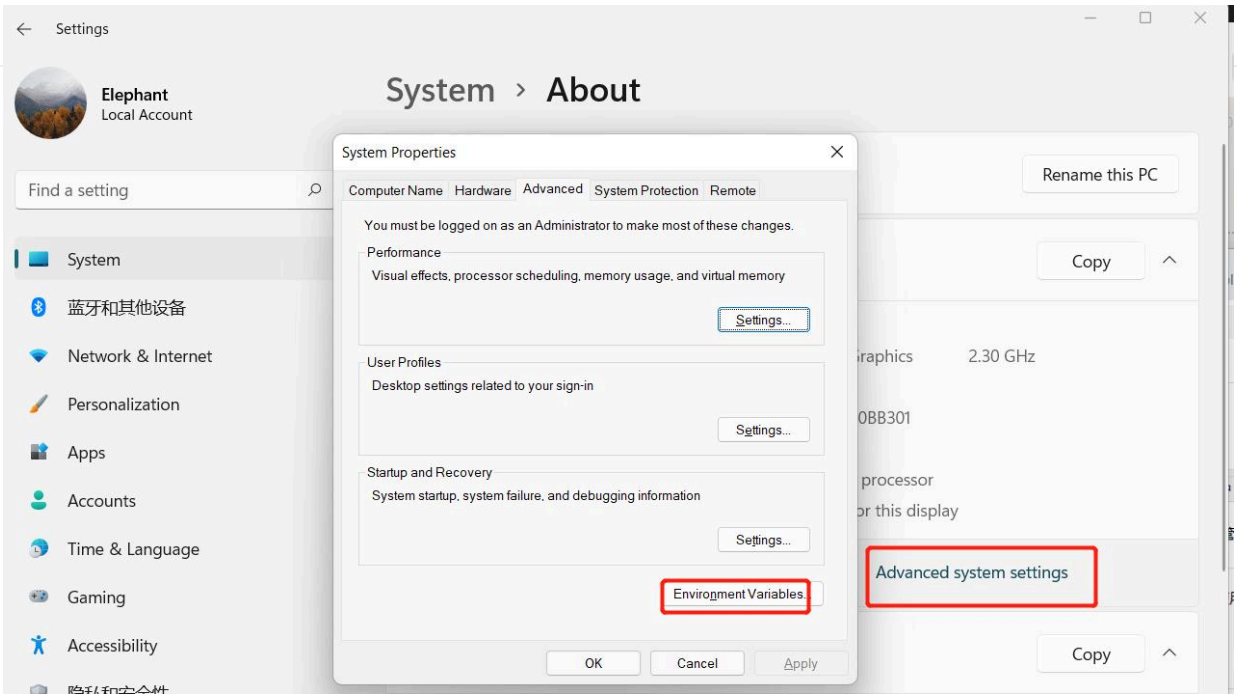
> **Notice:** Generally, the error results from lack of environment configuration. Refer to **1.3 Environment Configuration** to solve problems.

1.3 Environment Variable Configuration

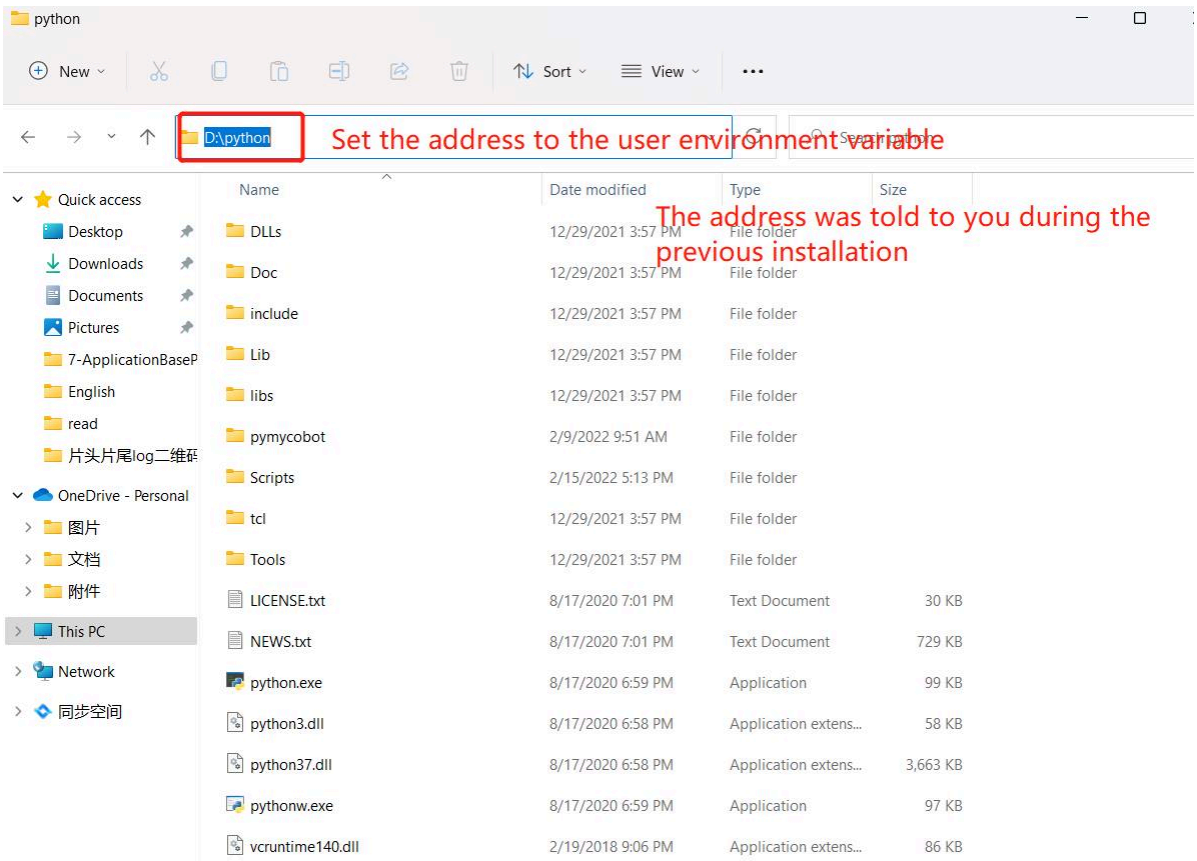
Windows follows the path set by a Path environment variable in search of **python.exe** . Otherwise, an error will be reported. If you fail to tick `Add Python 3.9 to PATH` during installation, you need to manually add the path where python.exe is located into environment variable or download python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

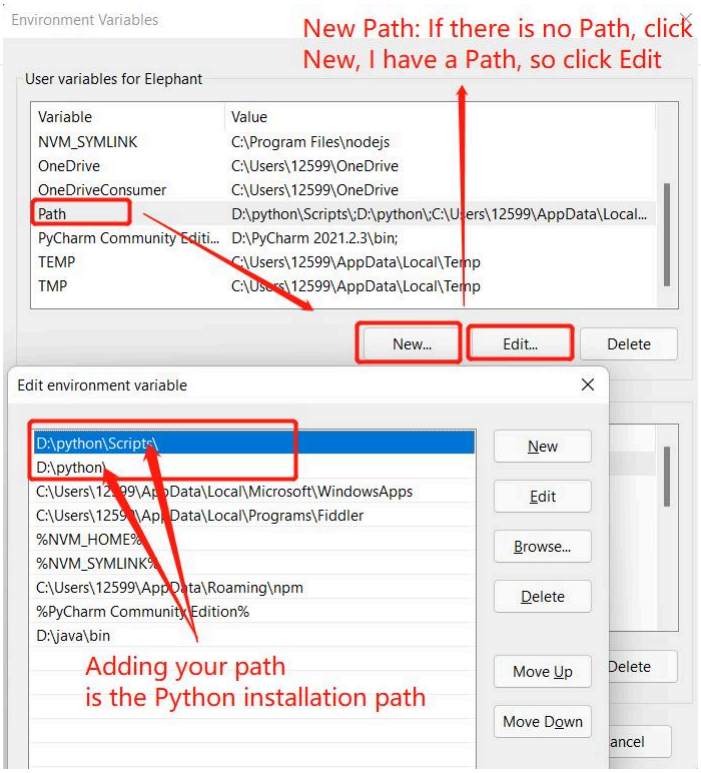
Follow the steps below to add python into environment variable manually.

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

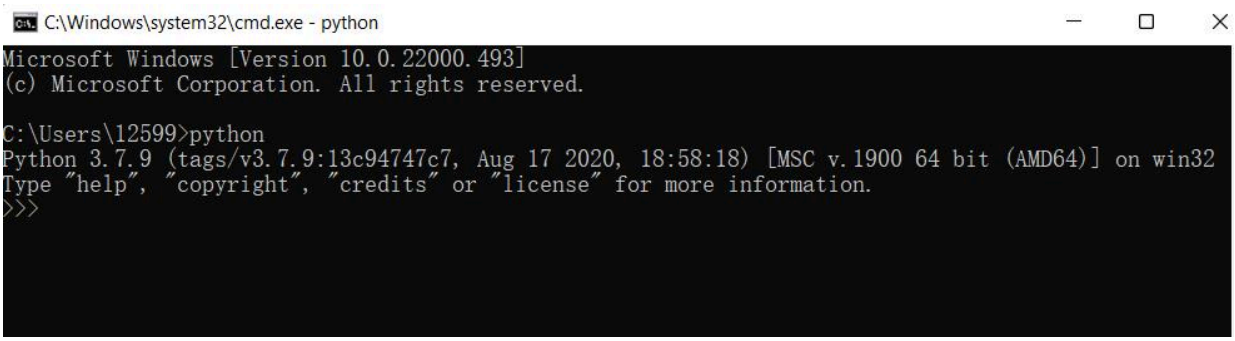


- The environment variables include user variables and system variables. For user variables, users can utilize their own downloaded programs via `cmd` command. Write the absolute path of the target program into the user variables.





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



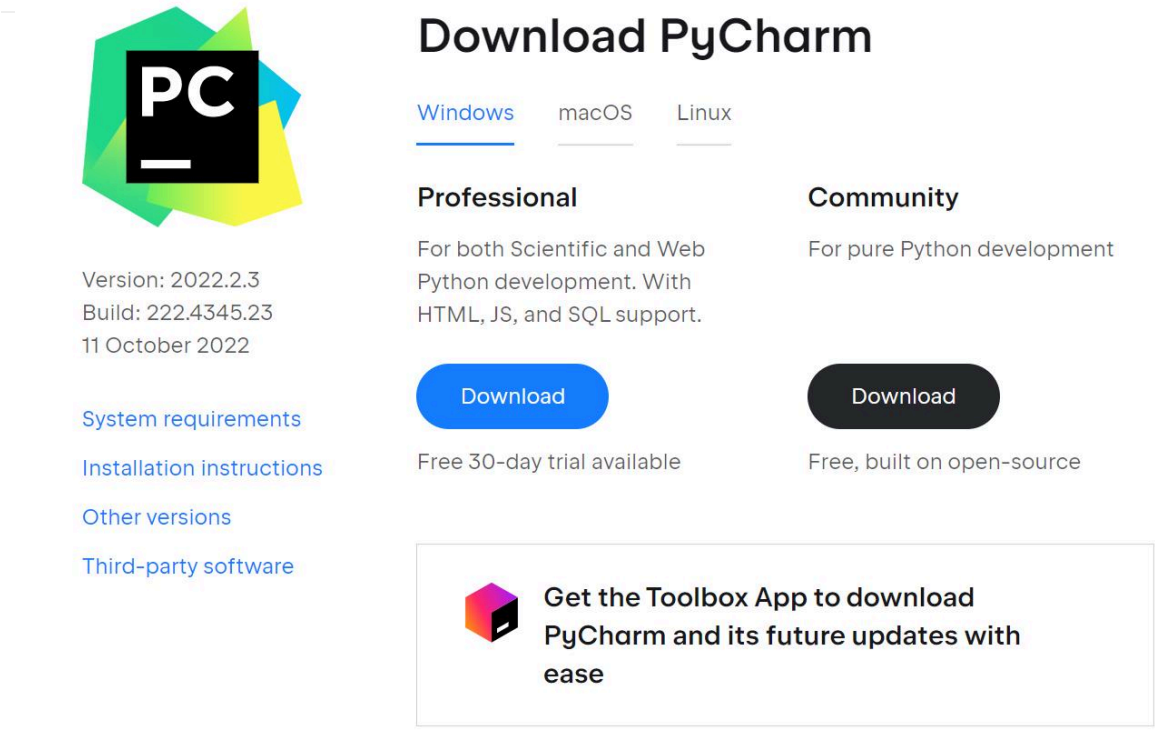
2 Installation of PyCharm

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

Go to **PyCharm** to download PyCharm.

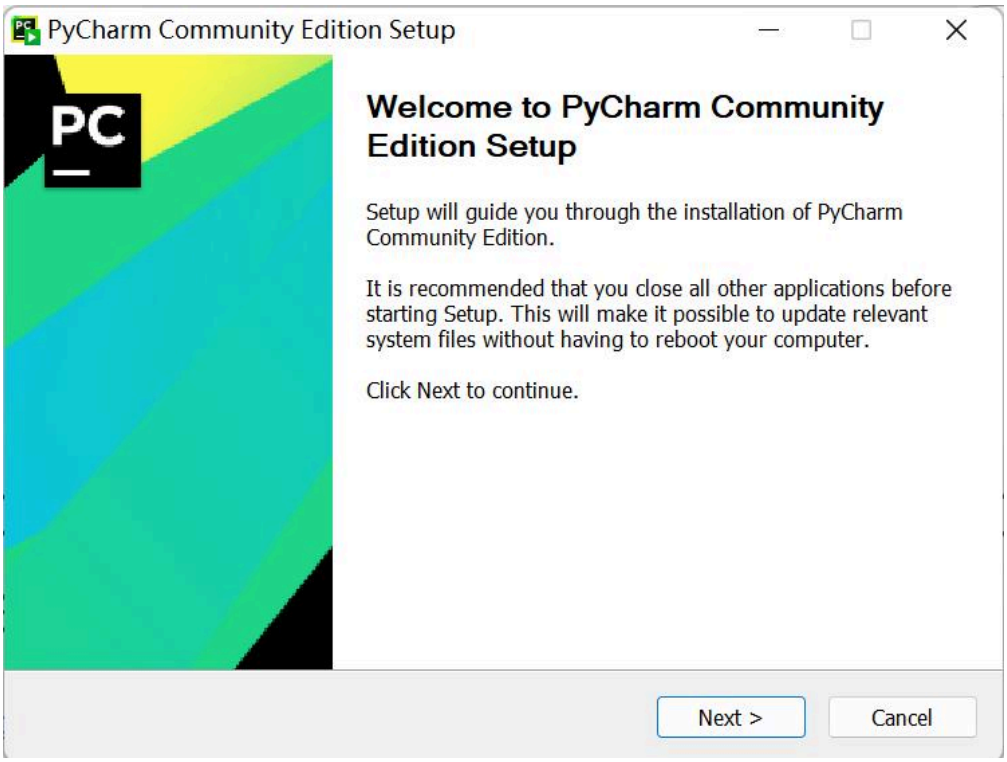
2.1 Download and Installation

Official website view:

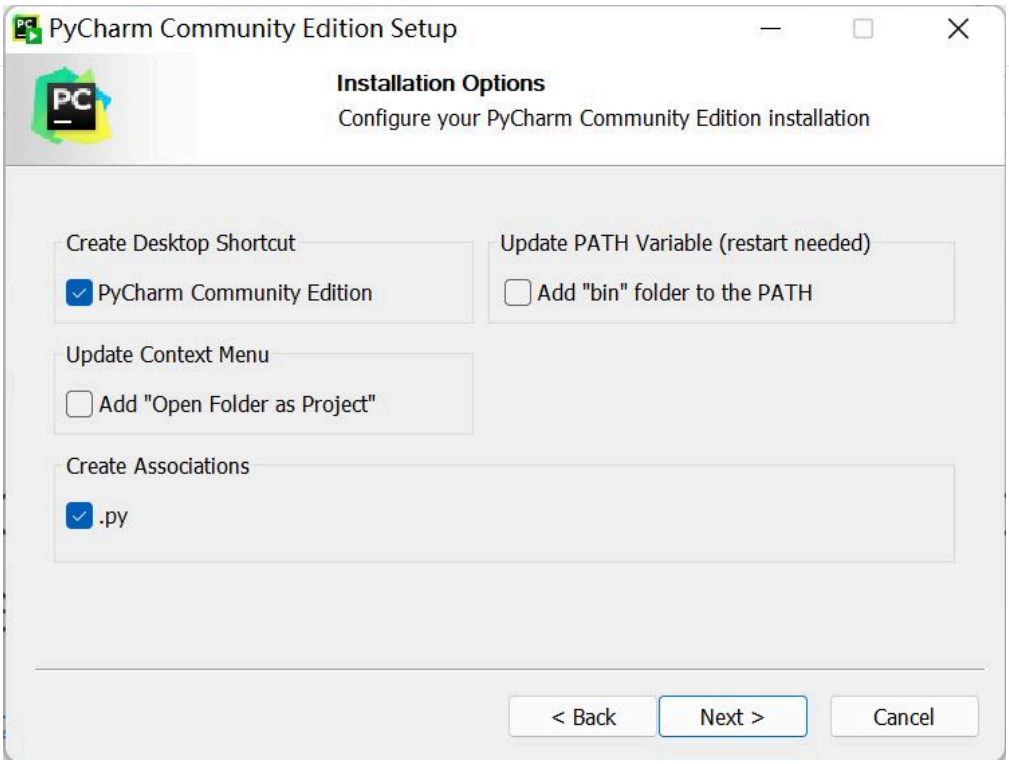


It is recommended to install the free version.

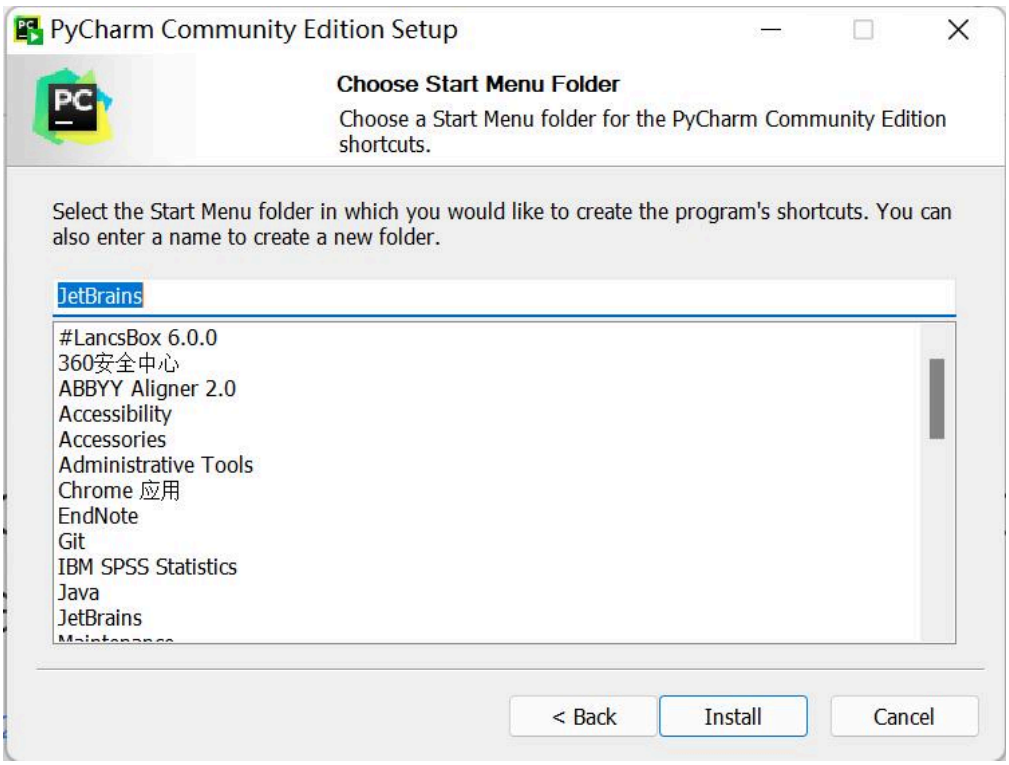
- Click on `Next` :



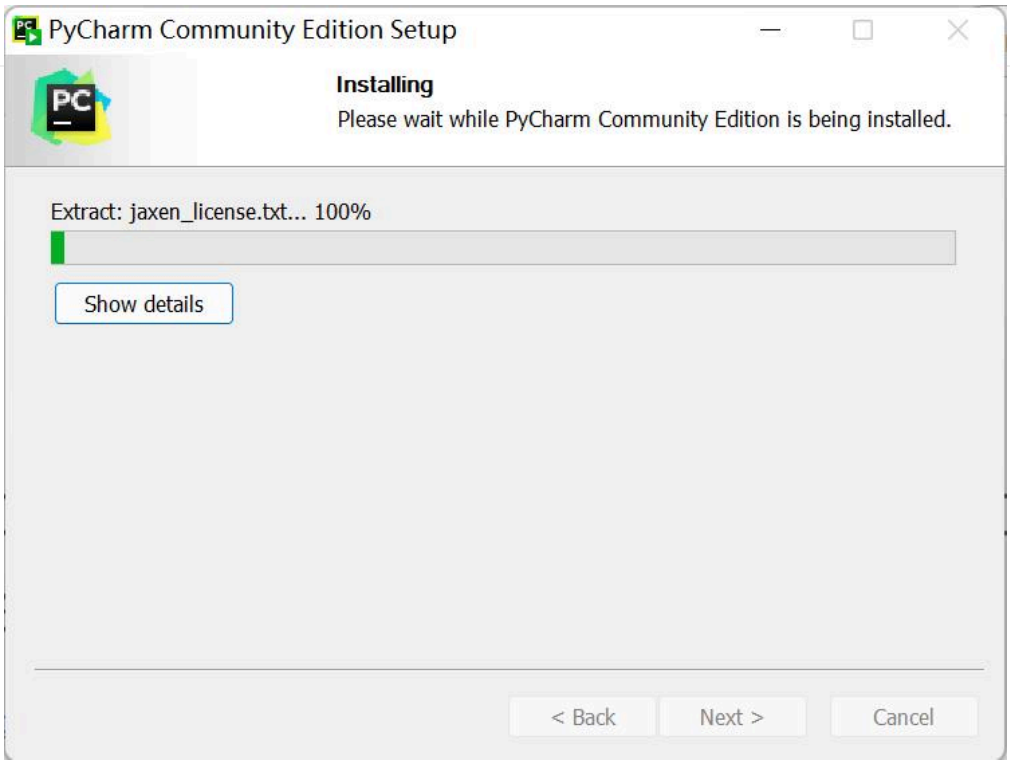
- Select options according to your needs and then select `Next` :



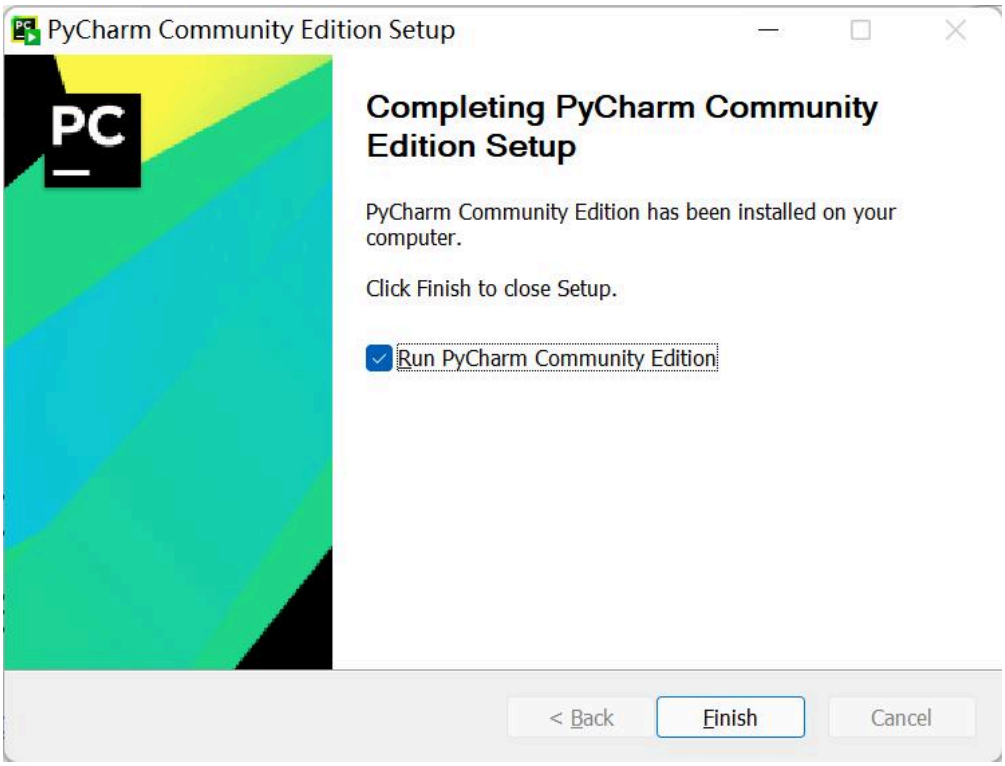
- Tap `Install` :



- Installing:

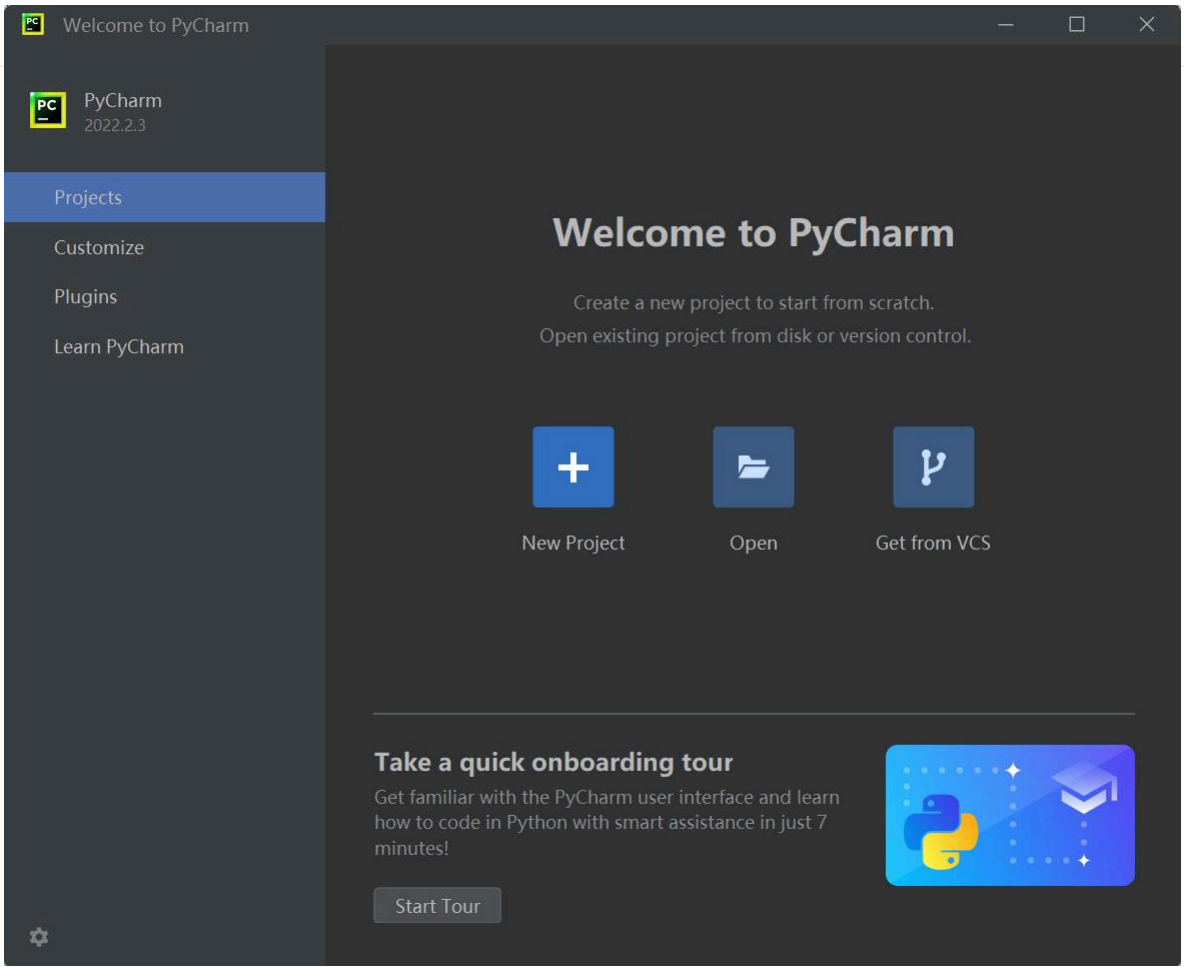


- Tap `Finish`

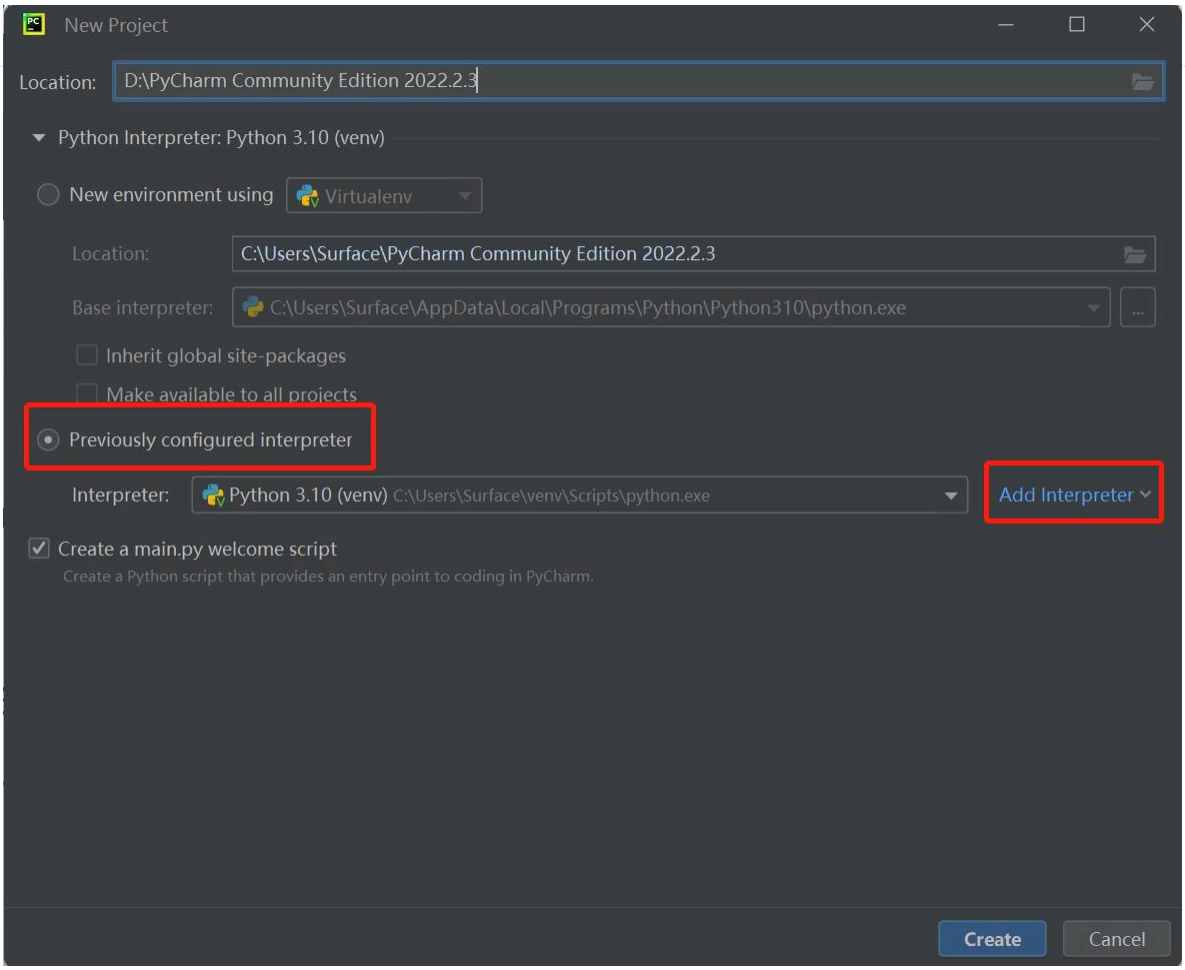


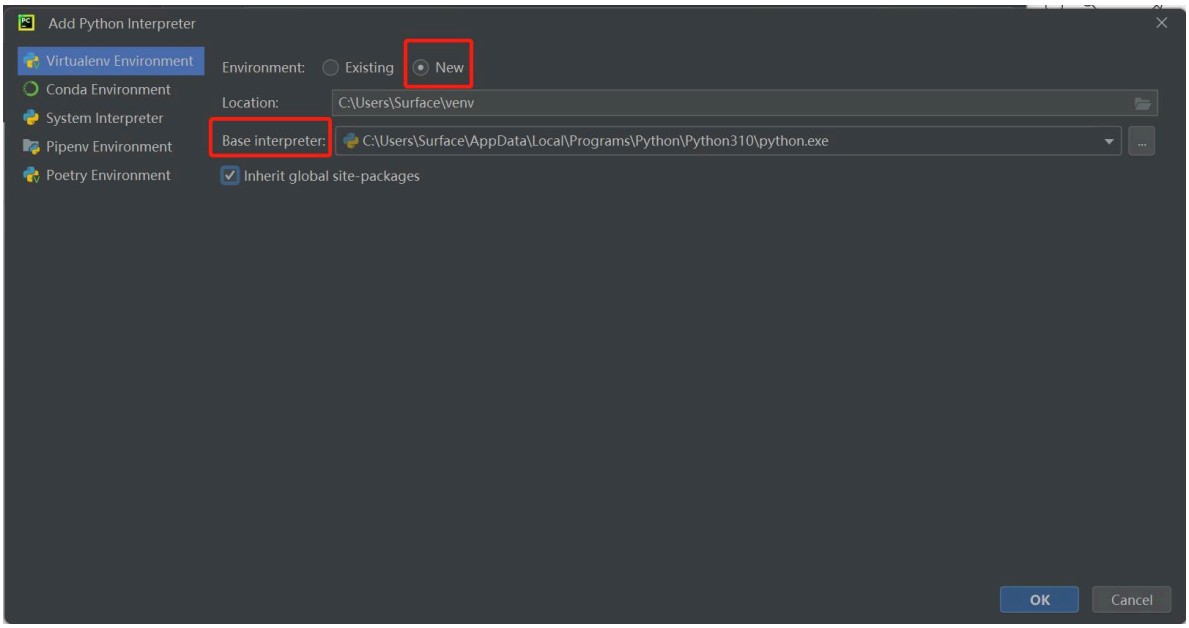
2.2 Create a new project

- Click `+New Project` :

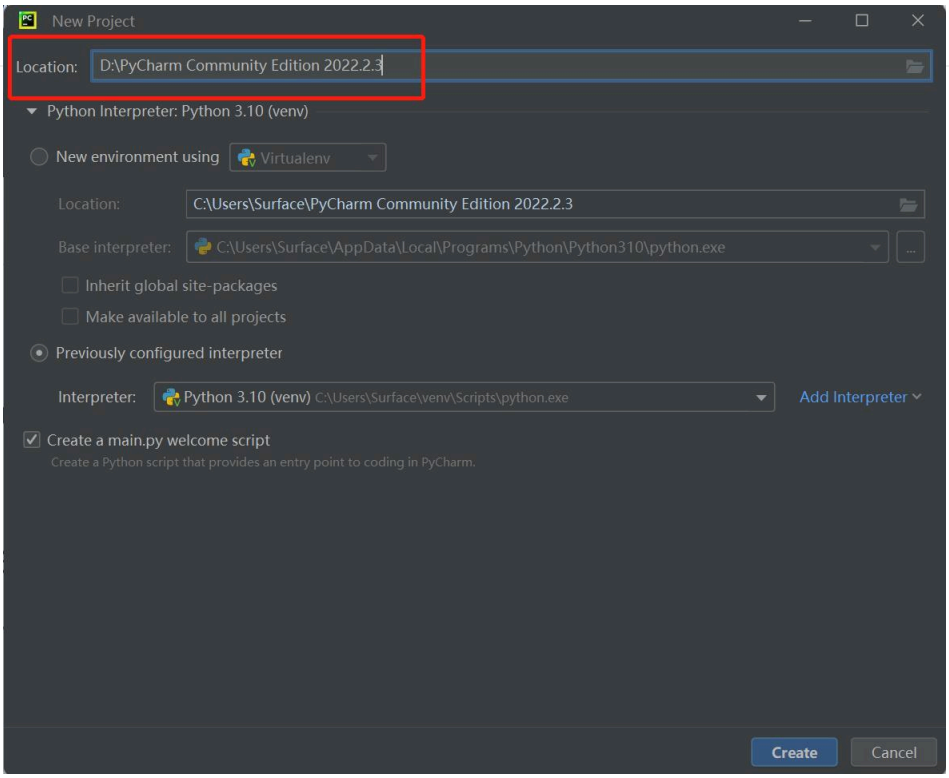


- The `Interpreter` is used to interpret python programs. Select `Add Interpreter` -> `New` to add base interpreter.

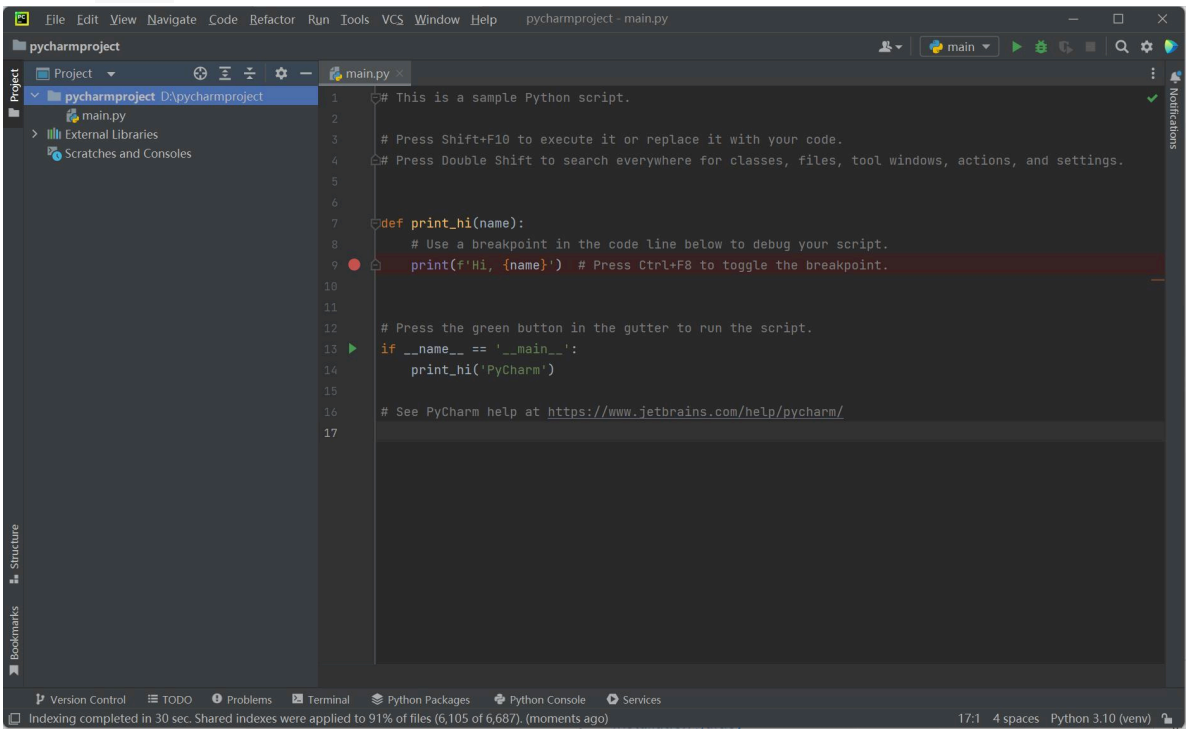




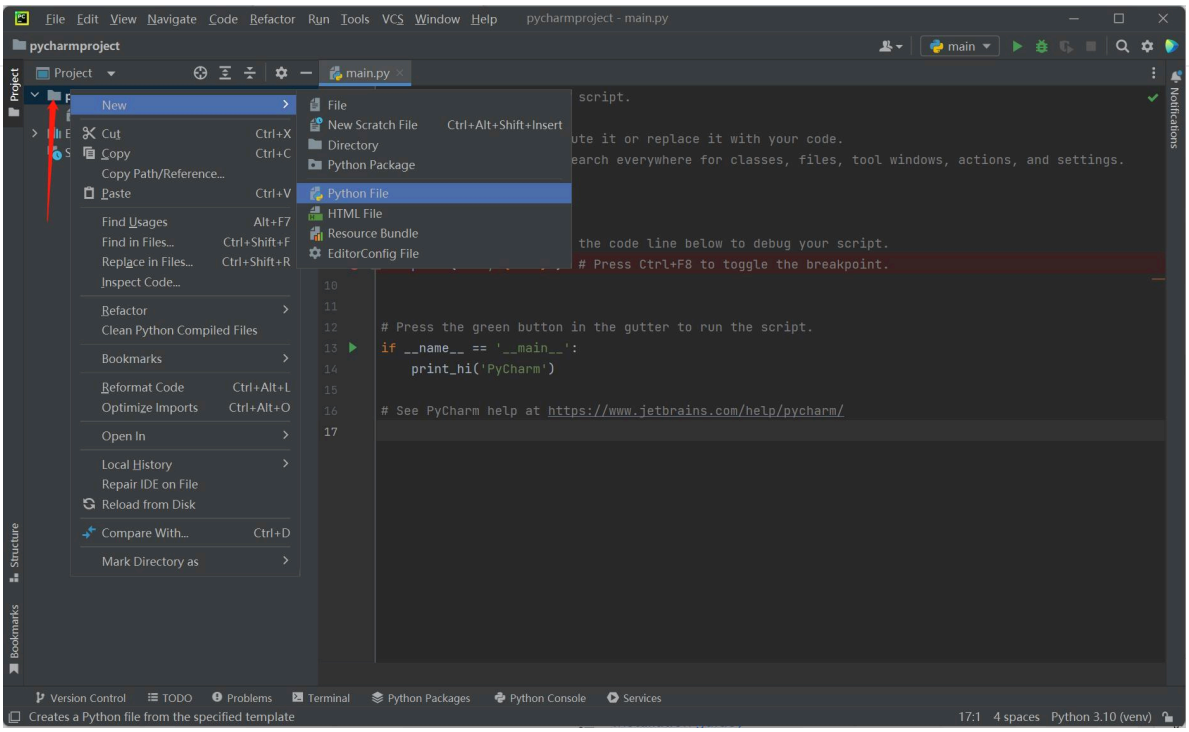
- `Location` refers to the place where to save python file. Choose a file to put your programs.



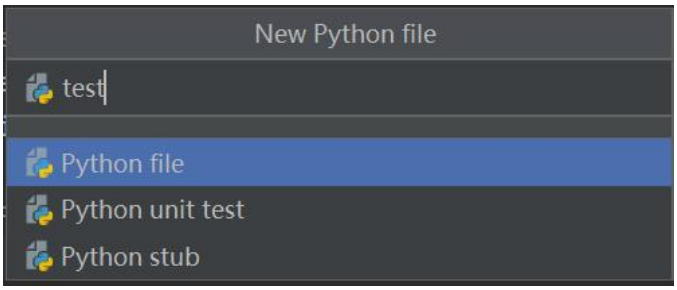
- Click on `Create` and a sample appears:



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



- Type name for the new file.



3 Preparations

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

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

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

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

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

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

- Update pymycobot

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

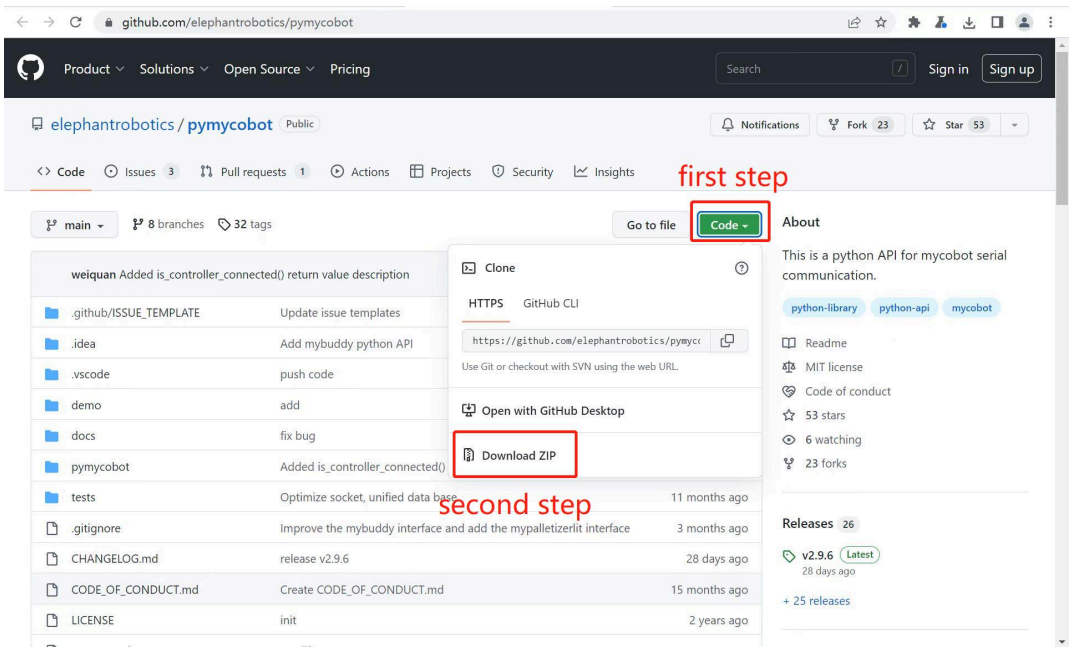
4 Import of pymycobot

This part takes MyPalletizer 260-M5 as an example to introduce how to control a robot via python.

- Import pymycobot library for MyPalletizer :

```
from pymycobot.mypalletizer import MyPalletizer
```

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



5 Simple Demo

Create a new Python file, and type the following codes to set the color of RGB light panel. The codes are suitable to myCobot 280-M5, myCobot 320-M5 and myPalletizer 260.

> **Notice:** The baud rates are different according to types of devices. Refer to **calculator device manager** to check the corresponding number.

- **Codes for MyCobot:**

```
# demo.py
from pymycobot.mycobot import MyCobot

from pymycobot import PI_PORT, PI_BAUD      # When using the Raspberry Pi version of mycobot, you can refer to these two v
import time
#The above codes are required to be written, which means importing the project package

# MyCobot class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is:  115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MyCobot("/dev/ttyUSB0", 115200)
#          or mc = MyCobot("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MyCobot("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot(PI_PORT, PI_BAUD)
#
# Initiate MyCobot
# Create object code here for windows version
mc = MyCobot("COM3", 115200)

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

Run the example file:

```
python3 demo.py
```

- **Codes for MyPalletizer:**

```
from pymycobot.mypalletizer import MyPalletizer
import time
##The above codes are required to be written at the beginning, which means importing the project package


#initiate MyPalletizer
mc = MyPalletizer("COM3", 115200)


#     MyPalletizer initiation requires two parameters:
#   The first is the serial port string, such as:
#       linux:  "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is:  115200
#
#    Example:
#       MyPalletizer-M5:
#           linux:
#              mc = MyPalletizer("/dev/ttyUSB0", 115200)
#          or mc = MyPalletizer("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MyPalletizer("COM3", 115200)


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

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

MyPalletizer 260

[toc]

Python API usage instaructions

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

```
# Example
from pymycobot import MyPalletizer260

# M5 version
mc = MyPalletizer260('COM3')

# PI version
# mc = MyPalletizer260('/dev/ttyAMA0', 1000000)
# Gets the current angle of all joints
angles = mc.get_angles()
print(angles)

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

1. System Status

1.1 `get_system_version()`

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

1.2 `get_basic_version()`

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

2. Overall Status

2.1 `power_on()`

- **function:** atom open communication (default open)

2.2 `power_off()`

- **function:** Power off of the robotic arm

2.3 `is_power_on()`

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

2.4 `release_all_servos()`

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

2.5 `focus_servo(servo_id)`

- **function:** Power on designated servo
- **Parameters:**
  - `servo_id:` int, 1-4

2.6 `is_controller_connected()`

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

2.7 `set_free_mode()`

- **function:** set to free mode
- **Parameters:**
  - `1` : open free mode
  - `0` : close free mode

2.8 `is_free_mode()`

- **function:** Check if it is free mode
- **Return value:**
  - `1` : free mode
  - `0` : on-free mode

3.MDI Mode and Operation

3.1 `get_angles()`

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

3.2 `send_angle(id, degree, speed)`

- **function:** send one degree of joint to robot arm
- **Parameters:**
  - `id` : Joint id(`genre.Angle`), range int 1-4
  - `degree` : degree value(`float`)
  - `speed` ： the speed and range of the robotic arm's movement 1~100

Joint Id	Range
1	-162 ~ 162
2	-2 ~ 90
3	-92 ~ 60
4	-180 ~ 180

3.3 `send_angles(angles, speed)`

- **function：** Send all angles to all joints of the robotic arm
- **Parameters:**
  - `angles` : a list of degree value(`List[float]`), length 4
  - `speed` : (`int`) 1 ~ 100

3.4 `get_coords()`

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

3.5 `send_coord(id, coord, speed)`

- **function:** send one coord to robot arm
- **Parameters:**
  - `id` :send one coord to robot arm, 1-4 corresponds to [x, y, z, rx]
  - `coord` : coord value(`float`)
  - `speed` : (`int`) 1-100

Coord Id	Range
x	-260 ~ 260
y	-260 ~ 260
z	-15 ~ 357.58
rx	-180 ~ 180

3.6 `send_coords(coords, speed)`

- **function:**: Send overall coordinates and posture to move the head of the robotic arm from its original point to your specified point
- **Parameters:**
  - coords: ： a list of coords value `[x,y,z,rx]` ,length 4
  - speed `(int)` : 1 ~ 100

3.7 `pause()`

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

3.8 `sync_send_angles(angles, speed,timeout=15)`

- **function：** Send the angle in synchronous state and return when the target point is reached
- **Parameters:**
  - `angles` : a list of degree value(`List[float]`), length 4
  - `speed` : (`int`) 1 ~ 100
  - `timeout` ： default 15 s

3.9 `sync_send_coords(coords, speed,timeout=15)`

- **function：** Send the coord in synchronous state and return when the target point is reached
- **Parameters:**
  - `coords` : a list of coord value(`List[float]`), length 4
  - `speed` : (`int`) 1 ~ 100
  - `timeout` : default 15 s

3.10 `get_angles_coords()`

- **function：** Get joint angles and coordinates
- **Return value:** A list with a length of 8. The first four digits are angle information, and the last four digits are coordinate information.

3.11 `is_paused()`

- **function:** Check if the program has paused the move command

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

3.12 `resume()`

- **function:** resume the robot movement and complete the previous command

3.13 `stop()`

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

3.14 `is_in_position(data, flag)`

- **function** : judge whether in the position.
- **Parameters:**
  - data: Provide a set of data that can be angles or coordinate values. If the input angle length range is 4, and if the input coordinate value length range is 4
  - flag data type (value range 0 or 1)
    - `0` : angle
    - `1` : coord
- **Return value**:
  - `1` - true
  - `0` - false
  - `-1` - error

3.15 `is_moving()`

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

4. JOG Mode and Operation

4.1 `jog_angle(joint_id, direction, speed)`

- **function:** jog control angle
- **Parameters**:
  - `joint_id` : Represents the joints of the robotic arm, represented by joint IDs ranging from 1 to 4
  - `direction(int)` : To control the direction of movement of the robotic arm, input `0` as negative value movement and input `1` as positive value movement
  - `speed` : 1 ~ 100

4.2 `jog_coord(coord_id, direction, speed)`

- **function:** jog control coord.
- **Parameters:**
  - `coord_id` : (`int`) Coordinate range of the robotic arm: 1~4
  - `direction` : (`int`) To control the direction of machine arm movement, `0` - negative value movement, `1` - positive value movement
  - `speed` : 1 ~ 100

4.3 `jog_absolute(joint_id, angle, speed)`

- **function:** Jog absolute angle
- **Parameters:**
  - `joint_id` : (`int`) 1-4
  - `angle` : -180 ~ 180
  - `speed` : (`int`) 1 ~ 100

4.4 `jog_increment(joint_id, increment, speed)`

- **function:** Single joint angle increment control
- **Parameters**:
  - `joint_id` : 1-4
  - `increment` : Incremental movement based on the current position angle
  - `speed` : 1 ~ 100

4.5 `set_encoder(joint_id, encoder, speed)`

- **function**: Set a single joint rotation to the specified potential value
- **Parameters**
  - `joint_id` : (`int`) 1-4
  - `encoder` : 0 ~ 4096
  - `speed` : 1 ~ 100

4.6 `get_encoder(joint_id)`

- **function**: Set a single joint rotation to the specified potential value
- **Parameters**
  - `joint_id` : (`int`) 1-4
- **Return value:** (`int`) Joint potential value

4.7 `set_encoders(encoders, speed)`

- **function**: Set the six joints of the manipulator to execute synchronously to the specified position.
- **Parameters**
  - `joint_id` : (`int`) 1-4
  - `encoder` : 0 ~ 4096
  - `speed` : 1 ~ 100

4.8 `get_encoders()`

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

5. Running status and Settings

5.1 `get_speed()`

- **function:** Get speed
- **Return value**： `int` joint speed

5.2 `set_speed(speed)`

- **function:** Set speed
- **Parameters:**
  - `speed` : (int) 1-100
- **Return value**： `float` Angle value

5.3 `get_joint_min_angle(joint_id)`

- **function:** Gets the minimum movement angle of the specified joint
- **Parameters:**
  - `joint_id` : Enter joint ID (range 0-3)
- **Return value**： `float` Angle value

5.4 `get_joint_max_angle(joint_id)`

- **function:** Gets the maximum movement angle of the specified joint
- **Parameters:**
  - `joint_id` : Enter joint ID (range 0-3)
- **Return value:** `float` Angle value

5.5 `set_joint_min(id, angle)`

- **function:** Set minimum joint angle limit
- **Parameters:**
  - `id` : Enter joint ID (range 0-3)
  - `angle` : Refer to the limit information of the corresponding joint in the send_angle() interface, which must not be less than the minimum value

5.6 `set_joint_max(id, angle)`

- **function:** Set maximum joint angle limit
- **Parameters:**
  - `id` : Enter joint ID (range 0-3)
  - `angle` : Refer to the limit information of the corresponding joint in the send_angle() interface, which must not be greater than the maximum value

6. Joint motor control

6.1 `is_servo_enable(servo_id)`

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

6.2 `is_all_servo_enable()`

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

6.3 `set_servo_calibration(servo_id)`

- **function:** The current position of the calibration joint actuator is the angle zero point
- **Parameters**:
  - `servo_id` : 1 - 4

6.4 `release_servo(servo_id)`

- **function:** Set the specified joint torque output to turn off
- **Parameters**:
  - `servo_id` : 1 ~ 4
- **Return value:**
  - `1` : release successful
  - `0` : release failed
  - `-1` : error

6.5 `focus_servo(servo_id)`

- **function**: Set the specified joint torque output to turn on
- **Parameters**: `servo_id` : 1 ~ 4
- **Return value:**
  - `1` : focus successful
  - `0` : focus failed
  - `-1` : error

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

- **function:** Set the data parameters of the specified address of the steering gear
- **Parameters**:
  - `servo_id` : (`int`) joint id 1 - 4
  - `data_id` : (`int`) Data address
  - `value` : (`int`) 0 - 4096

- `mode` : 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.

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

- **function:** Read the data parameter of the specified address of the steering gear.
- **Parameters**：
  - `servo_id` : (`int`) joint id 1 - 4
  - `data_id` : (`int`) Data address
  - `mode` : 0 - indicates that value is one byte(default), 1 - 1 represents a value of two bytes.
- **Return value:** 0 ~ 4096

6.8 `joint_brake(joint_id)`

- **function:** Make it stop when the joint is in motion, and the buffer distance is positively related to the existing speed
- **Parameters**：
  - `joint_id` : (`int`) joint id 1 - 4

7. Robotic arm end IO control

7.1 `set_color(r, g, b)`

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

7.2 `set_digital_output(pin_no, pin_signal)`

- **function:** set IO statue
- **Parameters**
  - `pin_no` (int): Pin number
  - `pin_signal` (int): 0 / 1

7.3 `get_digital_input(pin_no)`

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

7.4 `set_pin_mode(pin_no, pin_mode)`

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

8. Robotic arm end gripper control

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

- **function**: Adaptive gripper enable
- **Parameters**:
  - `flag (int)` : 0 - open 1 - close, 254 - release
  - `speed (int)` : 1 ~ 100
  - `_type_1 (int)` :
    - `1` : Adaptive gripper (default state is 1)
    - `2` : A nimble hand with 5 fingers
    - `3` : Parallel gripper
    - `4` : Flexible gripper

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

- **function**: Set the gripper value
- **Parameters**:
  - `gripper_value (int)` : 0 ~ 100
  - `speed (int)` : 1 ~ 100
  - `gripper_type (int)` :
    - `1` : Adaptive gripper (default state is 1)
    - `2` : A nimble hand with 5 fingers
    - `3` : Parallel gripper
    - `4` : Flexible gripper

8.3 `set_gripper_calibration()`

- **function**: Set the current position of the gripper to zero

8.4 `is_gripper_moving()`

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

8.5 `get_gripper_value()`

- **function**: Get the value of gripper
- **Parameters**:
  - `gripper_type` : (int) default 1

    - 1: Adaptive gripper
    - 3: Parallel gripper
    - 4: Flexible gripper
- **Return value**：gripper value (int)

9. Set bottom IO input/output status

9.1 `set_basic_output(pin_no, pin_signal)`

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

9.2 `get_basic_input(pin_no)`

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

10. WLAN Setting

10.1 `set_ssid_pwd(account, password)`

- **function:** Change connected wifi. (Apply to m5)
- **Parameters:**
  - `account` (`str`) new wifi account
  - `password` (`str`) new wifi password

10.2 `get_ssid_pwd()`

- **function:** Get connected wifi account and password. (Apply to m5)
- **Return value:** (account, password)

10.3 `set_server_port(port)`

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

11. TOF

11.1 `get_tof_distance()`

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

12. Raspberry pi -- GPIO

12.1 `gpio_init()`

- **function**: Init GPIO module, and set BCM mode.

12.2 `gpio_output(pin, v)`

- **function**: Set GPIO port output value.
- **Parameters**
  - `pin` (`int`) Pin number.
  - `v` (`int`): 0 / 1

13. utils (module)

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

```
from pymycobot import utils
```

13.1 `utils.get_port_list()`

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

13.2 `utils.detect_port_of_basic()`

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

MyPalletizer 260 Socket

> Note: raspberryPi version Only supports python3 The robotic arm that uses this class of premise has a server and has been turned on.

Use TCP/IP to control the robotic arm

1. Client

```
# demo
from pymycobot import MyPalletizerSocket
# Port 9000 is used by default
mc = MyPalletizerSocket("192.168.10.10",9000)

res = mc.get_angles()
print(res)

mc.send_angles([0,0,0,0],20)
...
```

2. Server

Server file is in the `demo folder` ,For details, please check the Server_260.py file in the demo folder

3.1 socket control

> Note: Most of the methods are the same as the class MyPalletizer260, only the new methods are listed here.

3.1 `set_gpio_mode(mode)`

- **function**: Set pin coding method.
- **Parameters**
  - `mode` (`str`) "BCM" or "BOARD".

3.2 `set_gpio_out(pin_no, mode)`

- **function**: Set the pin as input or output.
- **Parameters**
  - `pin_no` (`int`) pin id.
  - `mode` (`str`) "in" or "out"

3.3 `set_gpio_output(pin_no, state)`

- **function**: Set the pin to high or low level.
- **Parameters**
  - `pin_no` (`int`) pin id.
  - `state` (`int`) 0 or 1

3.4 `get_gpio_in(pin_no)`

- **function**: Get pin level status.
- **Parameters**
  - `pin_no` (`int`) pin id.

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

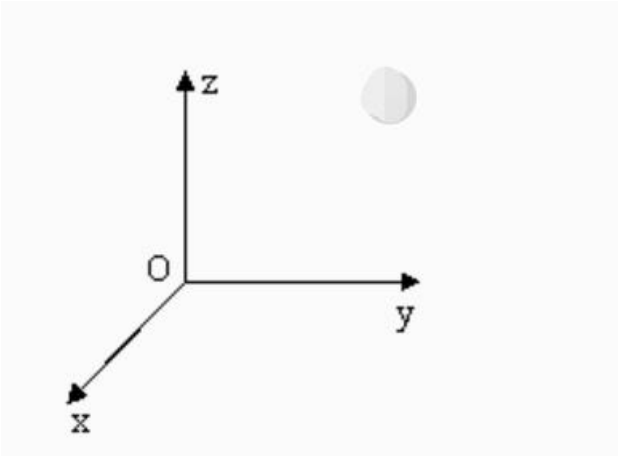
Simple Demo

```
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

```
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)
```

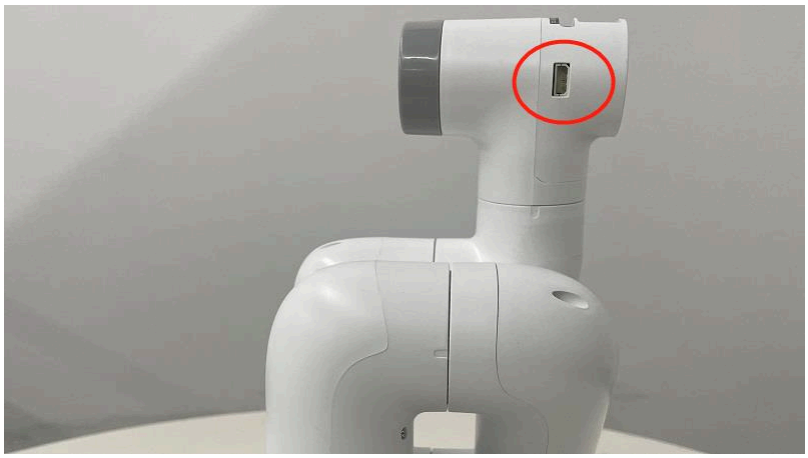
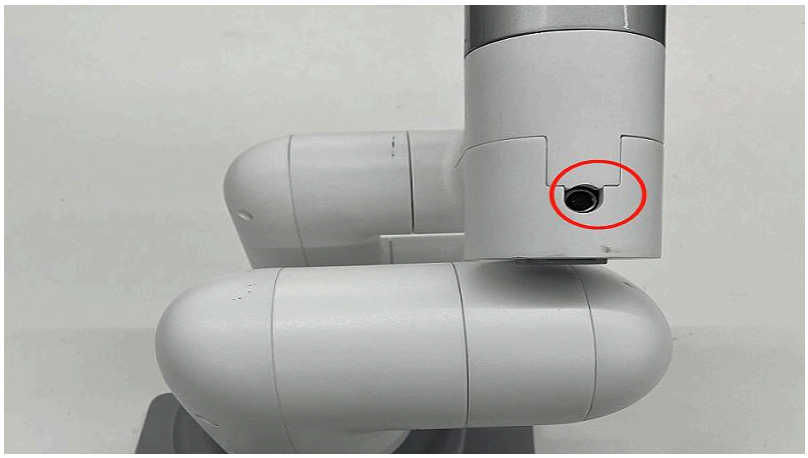
MyArm

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

```
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

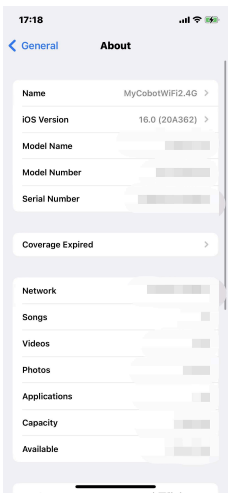
1.1 Create a default network

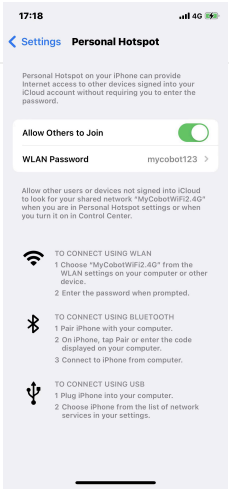
When using TCP/IP with the myCobot 280 m5 , it connects to a "MyCobotWiFi2.4G" network using the default password "mycobot123".

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.

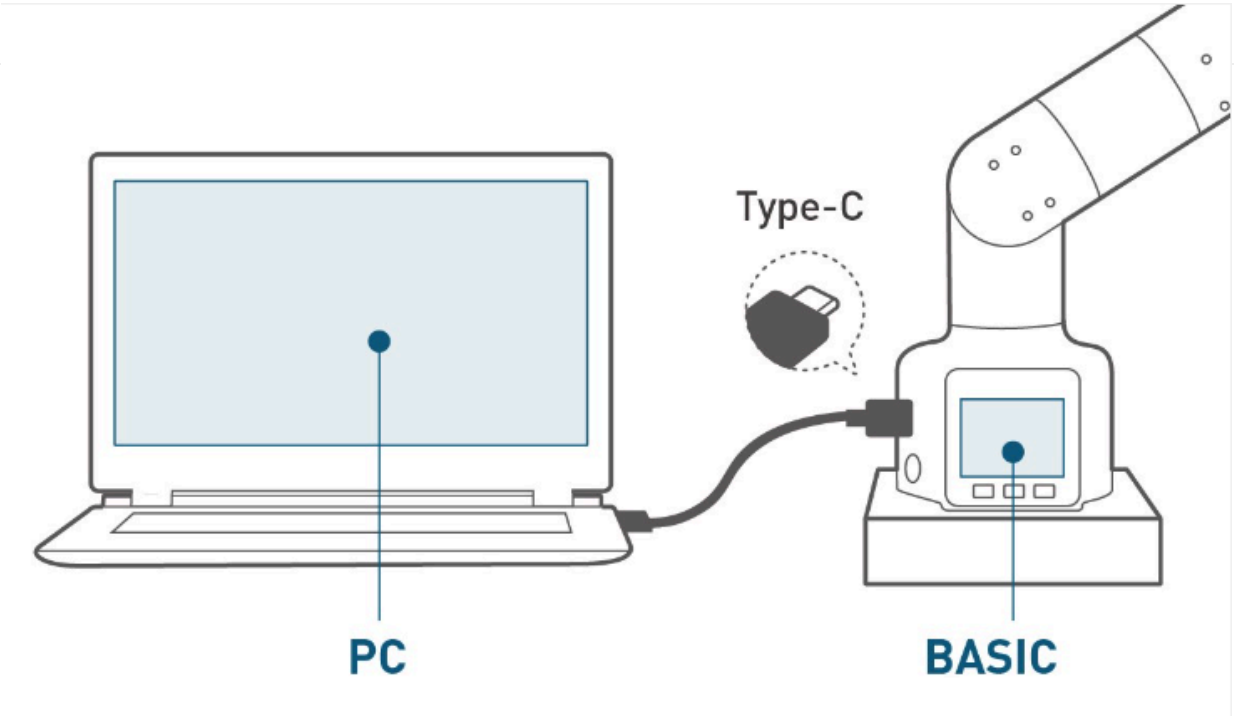
If the screen shows WIFI Connected, IP and Port, it means that robotic arm is successfully connected with WIFI.

1.3 Connect to another network

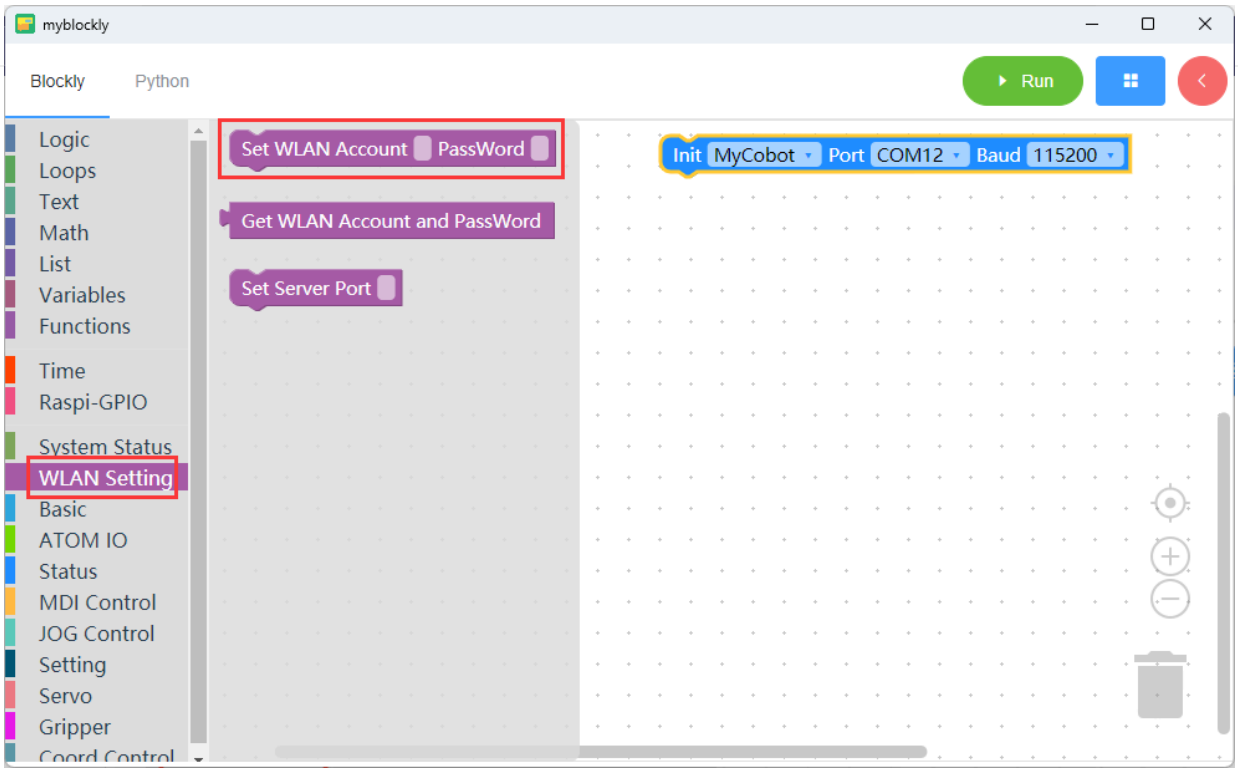
If you need to connect to another network, you can use the myBlockly software to configure the new network settings.

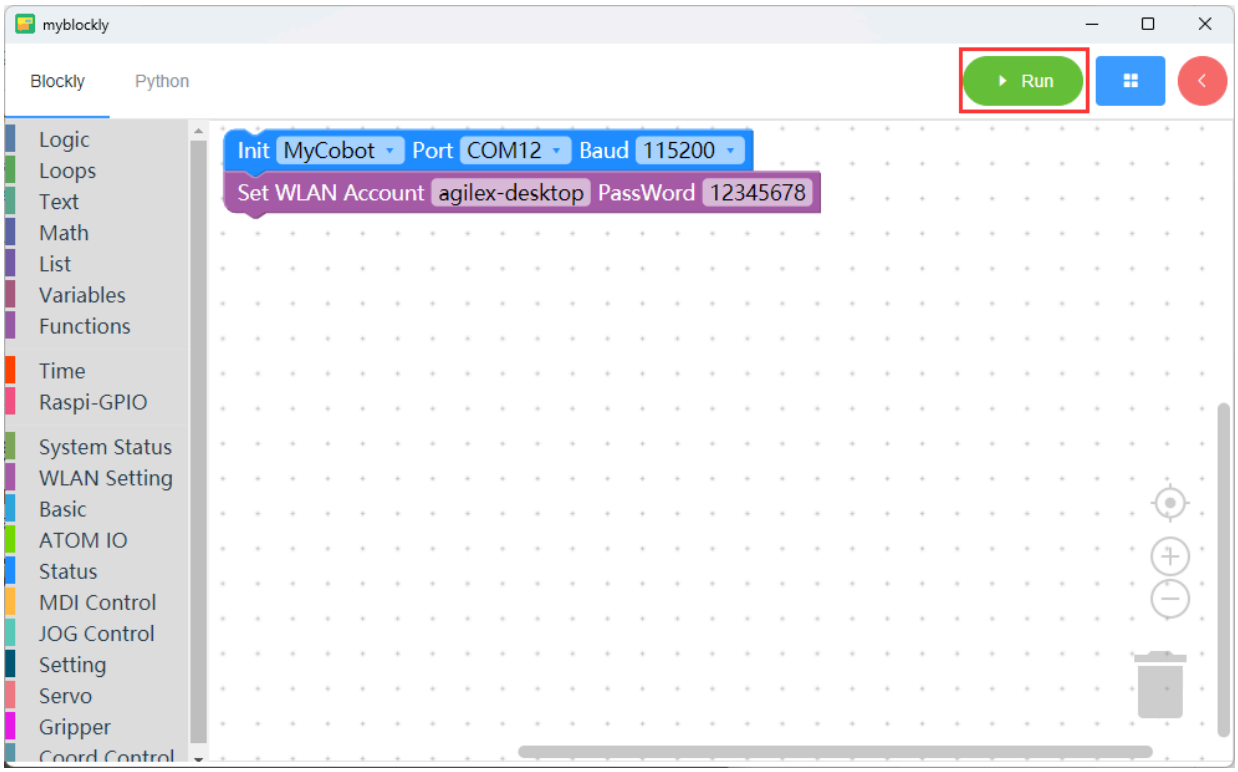
Note: The myCobot 280 m5 cannot save the connected Wi-Fi account and password when powered off. After a power cycle, it will reconnect to the default Wi-Fi account "MyCobotWiFi2.4G" with the password "mycobot123". To connect to a different network, you will need to reconfigure the Wi-Fi account and password.

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)

...
```

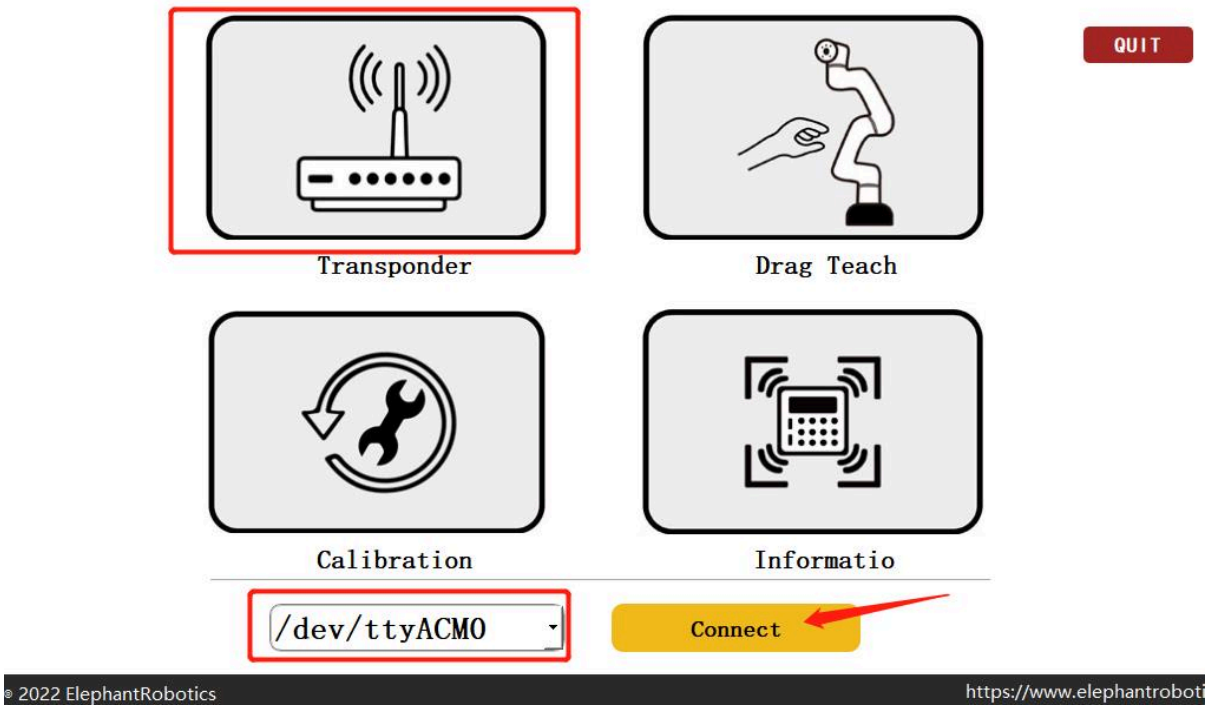
mybuddy

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

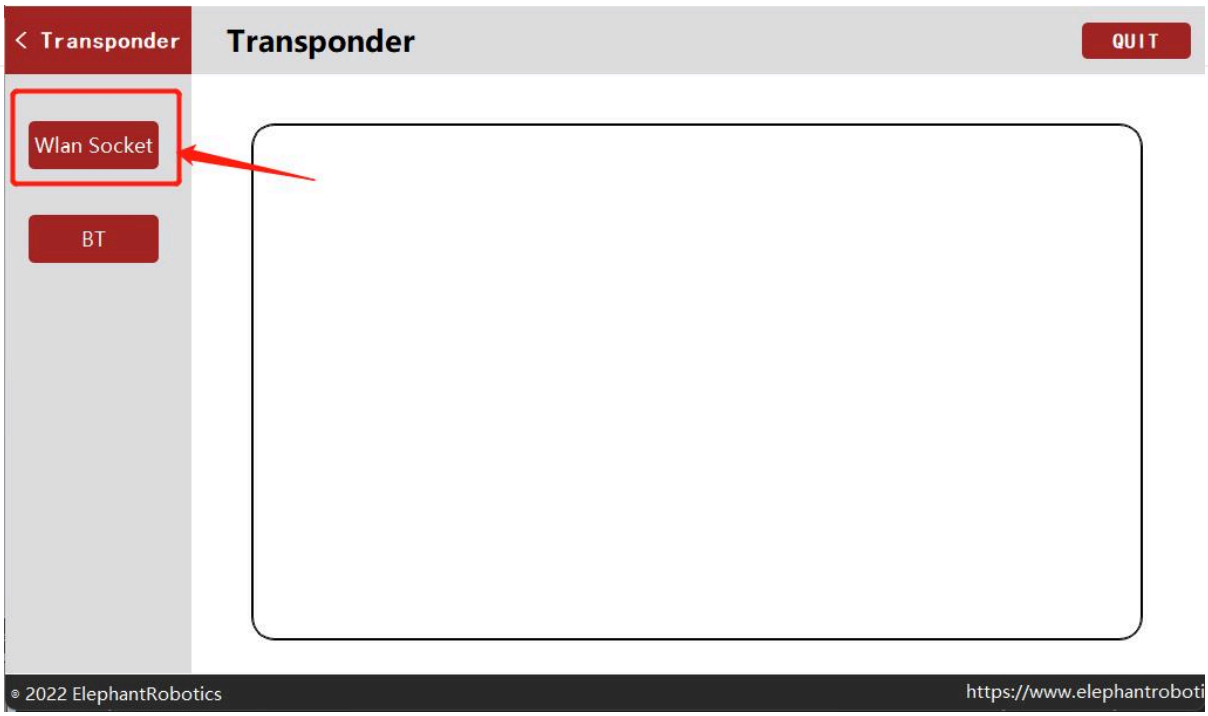
start the server

1.Double click to open this software on the desktop

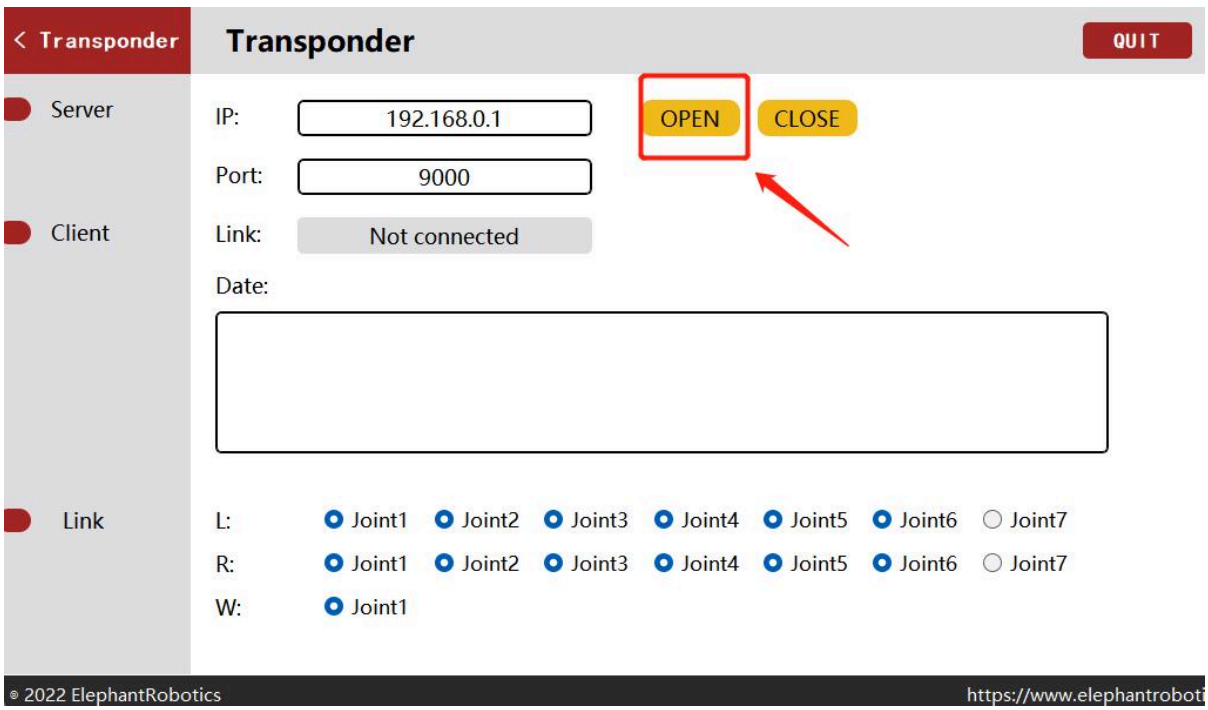
2.Select Transponder, the port is /dev/ttyACM0, click Connect



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))
```

The movement of the machine can be controlled by the gamepad, and the grasping of objects can be realized with the gripper or the suction pump.

> Note: The handle needs to be purchased separately, please consult the official customer service for details

The corresponding functions of the handle buttons are as follows:

myCobot:

- **1**: RX coordinate value increases
- **2**: RX coordinate value decreases
- **3**: RY coordinate value increases
- **4**: RY coordinate value decreases
- **5**: X coordinate value increases
- **6**: X coordinate value decreases
- **7**: Y coordinate value decreases
- **8**: Y coordinate value increases
- **9**: Z coordinate value decreases
- **10**: Z coordinate value increases
- **11**: RZ coordinate value decreases
- **12**: RZ Coordinate value increases
- **13**: Wake up the handle. If the handle is not used for a long time after connection, it will enter sleep mode. You need to press this button to continue using it.
- **X**: Click the button to open the jaws
- **Y**: Click the button to close the jaws
- **A**: Click the button to turn on the suction pump
- **B**: Click the button to turn off the suction pump

- **Left 1**: Press and hold for 2s to initialize the robot to the joint zero position state.
- **Left 2**: Press and hold for 2s, the robot stops torque output and relaxes all joints.
- **Right 1**: Press and hold for 2s to initialize the robot to the initial point of movement.
- **Right 2**: Press and hold for 2s, the robot turns on torque output and all joints are locked.

myArm

- **1**: RX direction coordinate value increases
- **2**: RX direction coordinate value decreases
- **3**: RY direction coordinate value decreases
- **4**: RY direction coordinate value increases
- **5**: X direction coordinate value increases
- **6**: X direction coordinate value decreases
- **7**: Y direction coordinate value decreases
- **8**: Y direction coordinate value increases
- **9**: Currently not functional
- **10**: Currently not functional
- **11**: Currently not functional
- **12**: Currently not functional
- **13**: Wake up the handle. After the handle is not used for a long time after connection, it will enter sleep mode. You need to press this button to wake up.
- **14**: Check the connection status of the machine. The atom LED flashes green three times to indicate that the machine is normal; flashes red three times to indicate that the state is abnormal.
- **X**: Z direction coordinate value increases
- **Y**:Z direction coordinate value decreases
- **A**: open suction gripper
- **B**: Shut down the suction gripper
- **Left 1**: Press and hold for 2s to initialize the robot to the joint zero state.
- **Left 2**: Press and hold for 2s, the robot stops torque output and relaxes all joints.
- **Right 1**: Press and hold for 2s to initialize the robot to move to the initial point.
- **Right 2**: Press and hold for 2s, the robot turns on the torque output, and all joints are locked

instructions

1.Connecting devices

Connect the MyCobot and handle to the computer.

2.Install required libraries

Download code: https://github.com/elephantrobotics/pymycobot

Open the terminal, Enter the `pymycobot/demo/handle_control` folder, and run the following command。

```
pip3 install -r requirements.txt
```

3.Modify port number

myCobot

Edit the myCobot280_handle_control.py file

```
import pygame
import time
from pymycobot import MyCobot280
import threading
# Change com7 to the actual port number detected by your computer

mc = MyCobot280("com7", 115200)
...
```

Run the program.

```
python3 myCobot280_handle_control.py
```

> Note: After running the program, first click the **Right 1** button. After the machine reaches the initial point, other operations can be performed.

myArm

Edit the myarm_handle_control.py file

```
import pygame
import time
from pymycobot import MyArm
import threading
# Change com7 to the port number detected by your computer

mc = MyArm("/dev/ttyAMA0",115200)
...
```

Finally. run the program

```
python3 myarm_handle_control.py
```

> Note: After running the program, first click the **14** button to check the machine connection status, the machine connection status is normal (if it is abnormal, you will not be able to perform other operations, please solve the abnormal connection problem first), and then click **Right 1** button, other operations can only be performed after the machine reaches the initial point.

Videos and Codes for Display

Videos given below are for reference.

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

1 Controlling RGB Light Panel

MechArm270

```
from pymycobot import MechArm270


from pymycobot import PI_PORT, PI_BAUD      # When using the Raspberry Pi version of mycobot, you can refer to these two v
import time
#The above needs to be written at the beginning of the code, which means importing the project package


# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate:
#       M5 version is: 115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code here for windows version
mc = MechArm270("COM3", 115200)


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

myArm

```
from pymycobot.myarm import MyArm
import time
#The above needs to be written at the beginning of the code, which means importing the project package

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

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

2 Controlling Arms to Move Them to Starting Point

MechArm270

```
from pymycobot.mycobot import MechArm270
from pymycobot import PI_PORT, PI_BAUD      # When using the Raspberry Pi version of mycobot, you can refer to these two v


# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       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/ttyAMA0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code here for Raspberry Pi version
mc = MechArm270(PI_PORT, PI_BAUD)


# Check whether the program can be burned into the robot arm
if mc.is_controller_connected() != 1:
    print("Please connect the robot arm correctly for program writing")
    exit(0)


# Fine-tune the robotic arm to ensure that all the bayonets are aligned in the adjusted position
# Subject to the alignment of the mechanical arm bayonet, this is only a case
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
```

3 Single-Joint Motion

MechArm270

```
from pymycobot import MechArm270
from pymycobot.genre import Angle
import time

# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate:
#       M5 version is: 115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MMechArm270 object
# Create object code for Raspberry Pi
# mc = MechArm270(PI_PORT, PI_BAUD)
# Create object code for M5
mc=MechArm270('COM3',115200)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 40)
time.sleep(3)

# Control joint 3 to move 70°
mc.send_angle(Angle.J3.value,60,40)
time.sleep(3)

# Control joint 4 movement -70°
mc.send_angle(Angle.J4.value,-70,40)
time.sleep(3)

# Control joint 1 to move 90°
mc.send_angle(Angle.J1.value,90,40)
time.sleep(3)

# Control joint 5 movement -90°
```

```
mc.send_angle(Angle.J5.value,-90,40)
time.sleep(3)


# Control joint 5 to move 90°
mc.send_angle(Angle.J5.value,90,40)
time.sleep(3)
```

4 Multi-Joint Motion

MechArm270

```
import time
from pymycobot import MechArm270
# MechArm270 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#    Example:
#       mycobot-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MechArm270("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MechArm270("COM3", 115200)
#       mycobot-raspi:
#           mc = MechArm270(PI_PORT, PI_BAUD)
#
# Initialize a MechArm270 object
# Create object code for Raspberry Pi
# mc = MechArm270(PI_PORT, PI_BAUD)
# Create object code for M5
mc=MechArm270('COM3',115200)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
time.sleep(2.5)

# Control different angles of rotation of multiple joints
mc.send_angles([90,45,-90,90,-90,90],50)
time.sleep(2.5)

# Return the robotic arm to zero
mc.send_angles([0,0,0,0,0,0],50)
time.sleep(2.5)

# Control different angles of rotation of multiple joints
mc.send_angles([-90,-45,90,-90,90,-90],50)
time.sleep(2.5)
```

5 Swaying Arms Left and Right

MechArm270

```
from pymycobot import MechArm270
from pymycobot.genre import Angle
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

# Initialize a MechArm270 object
mc = MechArm270("COM3", 115200)
# Get the coordinates of the current location
angle_datas = mc.get_angles()
print(angle_datas)



#By passing the angle parameter, let each joint of the robotic arm move to the position
mc.send_angles([0, 0, 0, 0, 0, 0], 50)
print(mc.is_paused())
# Set the waiting time to ensure that the robotic arm has reached the specified position
# while not mc.is_paused():
time.sleep(2.5)

# Move joint 1 to the 90 position
mc.send_angle(Angle.J1.value, 90, 50)

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

# set loop times
num = 5

# The following code can make the robotic arm swing left and right
while num > 0:
    # Move joint 2 to the 50 position
    mc.send_angle(Angle.J2.value, 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(Angle.J2.value, -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
```

```
mc.send_angles([88.68, -70, 60, -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()
```

6 Let Robot Dance

MechArm270

```
from pymycobot import MechArm270
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time


if __name__ == "__main__":
    # MechArm270 class initialization requires two parameters:
    #   The first is the serial port string, such as:
    #       linux: "/dev/ttyUSB0"
    #          or "/dev/ttyAMA0"
    #       windows: "COM3"
    #   The second is the baud rate::
    #       M5 version is: 115200
    #
    #    such as:
    #       mycobot-M5:
    #           linux:
    #              mc = MechArm270("/dev/ttyUSB0", 115200)
    #          or mc = MechArm270("/dev/ttyAMA0", 115200)
    #           windows:
    #              mc = MechArm270("COM3", 115200)
    #       mycobot-raspi:
    #           mc = MechArm270(PI_PORT, PI_BAUD)
    #
    # Initialize a MechArm270 object
    # Create object code for Raspberry Pi version below
    mc = MechArm270(PI_PORT, PI_BAUD)

    # set start start time
    start = time.time()
    # Let the robotic arm reach the specified position
    mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
    # Determine if it reaches the specified position
    while not mc.is_in_position([-1.49, 115, -153.45, 30, -33.42, 137.9], 0):
        # Return the robotic arm to motion
        mc.resume()
        # Let the robotic arm move for 0.5s
        time.sleep(0.5)
        # Pause arm movement
        mc.pause()
        # Determine if the move timed out
        if time.time() - start > 3:
            break

    # set start time
    start = time.time()
```

```
    # Let the exercise last for 30 seconds
    while time.time() - start < 30:
        # Let the robotic arm quickly reach this position
        mc.send_angles([-1.49, 115, -153.45, 30, -33.42, 137.9], 80)
        # Set the color of the light to [0,0,50]
        mc.set_color(0, 0, 50)
        time.sleep(0.7)
        # Let the robotic arm quickly reach this position
        mc.send_angles([-1.49, 55, -153.45, 80, 33.42, 137.9], 80)
        # Set the color of the light to [0,50,0]
        mc.set_color(0, 50, 0)
        time.sleep(0.7)
```

7 Controlling Gripper

MechArm270

```
from pymycobot import MechArm270
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time
def gripper_test(mc):
    print("Start check IO part of api\n")
    # Check if the gripper is moving
    flag = mc.is_gripper_moving()
    print("Is gripper moving: {}".format(flag))
    time.sleep(1)

    # Set the current position to (2048).
    # Use it when you are sure you need it.
    # Gripper has been initialized for a long time. Generally, there
    # is no need to change the method.
    # mc.set_gripper_ini()
    # Set joint point 1 to rotate to 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)

    #Get the position information of joint point 1
    print(mc.get_encoder(1))
    # Set the gripper to rotate to the position of 2048
    mc.set_encoder(7, 2048)
    time.sleep(3)
    # Set the gripper to rotate to the position of 1300
    mc.set_encoder(7, 1300)
    time.sleep(3)

    # Let the gripper reach the state of 2048 at a speed of 70, 2048 will report an error, so change it to 255
    mc.set_gripper_value(255, 70)
    time.sleep(3)
    # Let the gripper reach the state of 1500 at a speed of 70, 1500 will report an error, so change it to 255
    mc.set_gripper_value(255, 70)
    time.sleep(3)


    num=5
    while num>0:
        # 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 to quickly close the gripper at a speed of 70
        mc.set_gripper_state(1, 70)
        time.sleep(3)
        num-=1

    # 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/ttyAMA0"
    #       windows: "COM3"
    #   The second is the baud rate::
    #       M5 version is: 115200
    #
    #    such as:
    #       mycobot-M5:
    #           linux:
    #              mc = MechArm270("/dev/ttyUSB0", 115200)
    #          or mc = MechArm270("/dev/ttyAMA0", 115200)
    #           windows:
    #              mc = MechArm270("COM3", 115200)
    #       mycobot-raspi:
    #           mc = MechArm270(PI_PORT, PI_BAUD)
    #
    # Initialize a MechArm270 object
    # Create object code for Raspberry Pi version below
    mc = MechArm270(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)
```

8 Controlling Sucking Pump

MyCobot 280

280-M5 version (the video below takes the M5 version as an example):

```python
from pymycobot import MyCobot280
from pymycobot import PI_PORT, PI_BAUD  # When using the Raspberry Pi version of mycobot, these two variables can be refer
import time

# MyCobot280 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#    such as:
#       mycobot-M5:
#           linux:
#              mc = MechArm270("/dev/ttyUSB0", 115200)
#          or mc = MyCobot280("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MyCobot280("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot280(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot280 object
# Create object code here for windows version
mc = MyCobot280("COM3", 115200)

# The position of the robot arm movement
angles = [
            [92.9, -10.1, -60, 5.8, -2.02, -37.7],
            [92.9, -53.7, -83.05, 50.09, -0.43, -38.75],
            [92.9, -10.1, -87.27, 5.8, -2.02, -37.7]
         ]

# Turn on the suction pump
def pump_on():
    # make position 2 work
    mc.set_basic_output(2, 0)
    # make position 5 work
    mc.set_basic_output(5, 0)

# stop the suction pump
def pump_off():
    # Stop position 2 from working
    mc.set_basic_output(2, 1)
    # Stop position 5 from working
    mc.set_basic_output(5, 1)

# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
time.sleep(3)
```

```
# Turn on the suction pump
pump_on()
mc.send_angles(angles[2], 30)
time.sleep(2)

# absorb small blocks
mc.send_angles(angles[1], 30)
time.sleep(2)
mc.send_angles(angles[0], 30)
time.sleep(2)
mc.send_angles(angles[1], 30)
time.sleep(2)

# Turn off the suction pump
pump_off()
mc.send_angles(angles[0], 40)
time.sleep(1.5)
```

280-Pi version:

```
from pymycobot import MyCobot280
from pymycobot import PI_PORT, PI_BAUD  # # When using the Raspberry Pi version of mycobot, these two variables can be ref
import RPi.GPIO as GPIO
import time

# MyCobot280 class initialization requires two parameters:
#   The first is the serial port string, such as:
#       linux: "/dev/ttyUSB0"
#          or "/dev/ttyAMA0"
#       windows: "COM3"
#   The second is the baud rate::
#       M5 version is: 115200
#
#    such as:
#       mycobot-M5:
#           linux:
#              mc = MyCobot280("/dev/ttyUSB0", 115200)
#          or mc = MyCobot280("/dev/ttyAMA0", 115200)
#           windows:
#              mc = MyCobot280("COM3", 115200)
#       mycobot-raspi:
#           mc = MyCobot280(PI_PORT, PI_BAUD)
#
# Initialize a MyCobot280 object
# Create object code here for Raspberry Pi version
mc = MyCobot280(PI_PORT, PI_BAUD)

# The position of the robot arm movement
angles = [
            [92.9, -10.1, -60, 5.8, -2.02, -37.7],
            [92.9, -53.7, -83.05, 50.09, -0.43, -38.75],
            [92.9, -10.1, -87.27, 5.8, -2.02, -37.7]
        ]
# Initialize the suction pump
GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

# Turn on the suction pump
def pump_on():
    # Make the 20 work
    GPIO.output(20, 0)
    # Make the 21 work
          GPIO.output(21, 0)

# Stop the suction pump
def pump_off():
    # Stop the 20 from working
    GPIO.output(20, 1)
    # Stop the 21 from working
```

```
        GPIO.output(21, 1)


# Robotic arm recovery
mc.send_angles([0, 0, 0, 0, 0, 0], 30)
time.sleep(3)

# Turn on the suction pump
pump_on()
mc.send_angles(angles[2], 30)
time.sleep(2)

# absorb small blocks
mc.send_angles(angles[1], 30)
time.sleep(2)
mc.send_angles(angles[0], 30)
time.sleep(2)
mc.send_angles(angles[1], 30)
time.sleep(2)

# Turn off the suction pump
pump_off()
mc.send_angles(angles[0], 40)
time.sleep(1.5)
```

C++

Using C++ language, you can make developments (coordinate control, angle control, io control, gripper control, etc.) freely through the C++ dynamic library developed by our company, and control some of the robots which have been developed by us.

What is C++?

C++ is the inheritance of C language. It may be used to carry out procedural programming of C language, object-based programming characterized by abstract data types, and object-oriented programming characterized by inheritance and polymorphism.

While C++ is good at object-oriented programming, it also can be used to carry out process-based programming, so C++ can adapt to the size of the problem.

C++ not only has the practical characteristics of efficient computer operation, but also strives to improve the programming quality of large-scale programs and the problem description ability of programming languages.

Applicable equipment:

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

- myCobot 320
  - myCobot 320 M5

Preconditions for use：

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

Programming development

Some integrated development environments (IDE)

Visual Studio (Visual C++)

Dev C++

C++ Builder

Compiler

Ultimate++

Digital Mars

C-Free

MinGW

C++ Environment building

1 Confirming development goals

MycobotCpp is an interface program used for serial communication with mycobot. It calls the mycobot library developed by our company, which contains simple use cases. If you want to make developments via C++ to control the robot that have been developed by us, it is your choice.

Available for: Robot Arm Models: myCobot 280-M5 and myCobot 320-M5.

The software required to run MycobotCpp: vs2019, qt5.12.10, and vsaddin (qt plugin).

2 Windows Environment Configuration

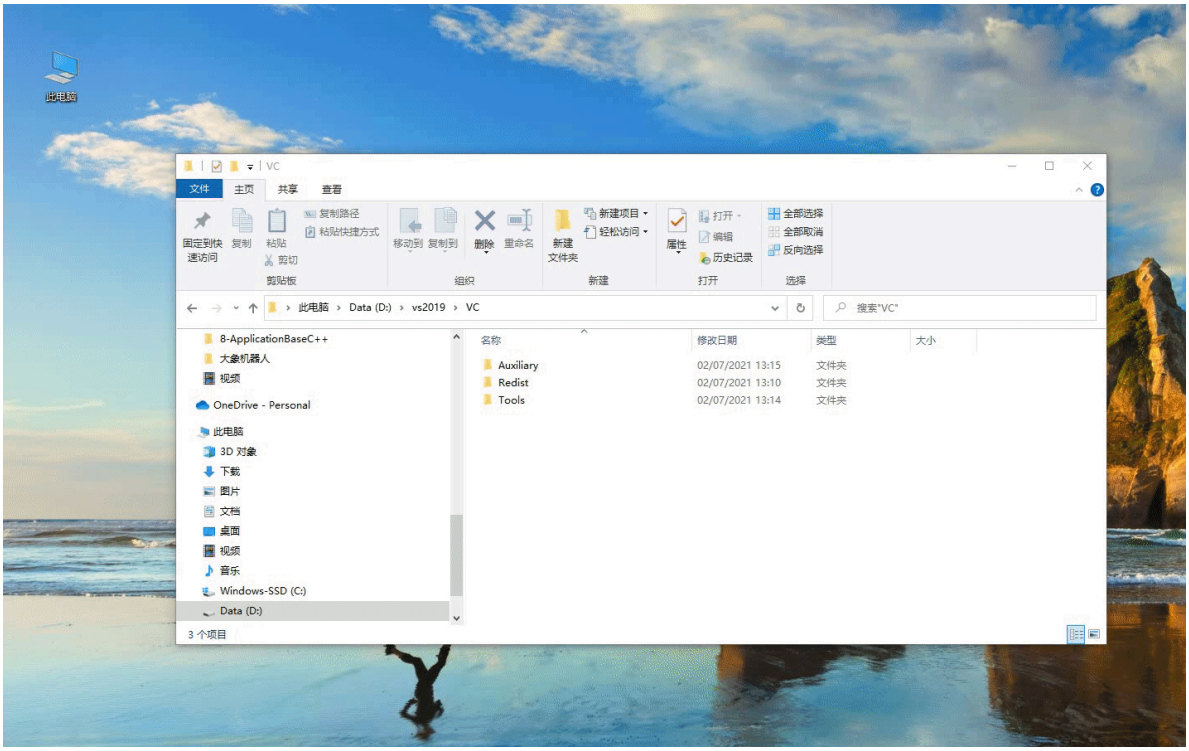
2.1 Installing vs2019

- Downloading: First download vs2019 from the official website.

- Installation: After installation is complete, the interface shown in the figure below will appear. "Universal Windows platform development, desktop development using C++, ASRNET and Web development" are mainly chosen (This is just a suggestion, you can choose according to your own needs. The installation time of vs2019 is long).

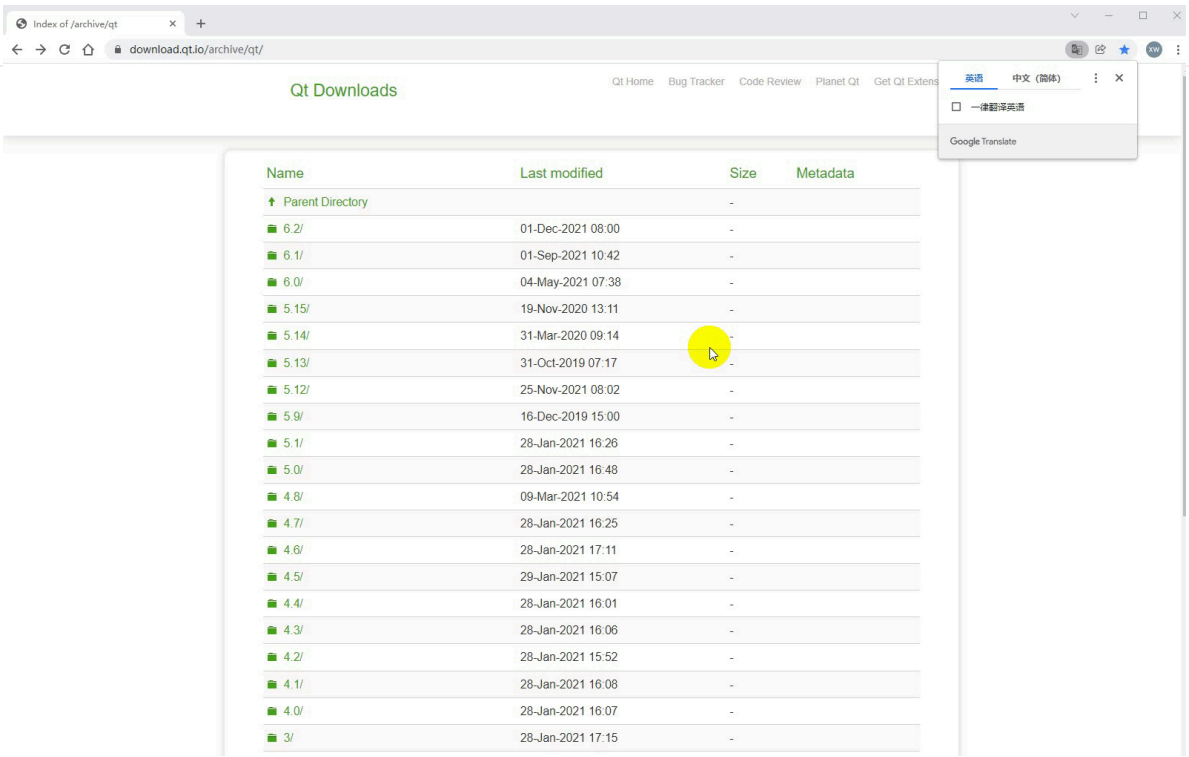
- Environment variable configuration: This Computer -> Right-click Properties -> Advanced System Settings -> Environment Variables -> Look at System Variables, click New-> Variable Name: VCINSTALLDIR variable

value: Find the directory where Redist is located (eg: D:2019), as shown in the following figure.

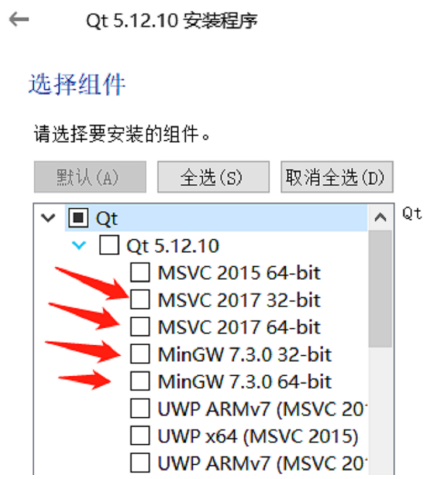


2.2 Installing qt5.12.10

- Downloading: Download qt5.12.10 and above versions. They are all available. The specific operation is shown in the figure below.



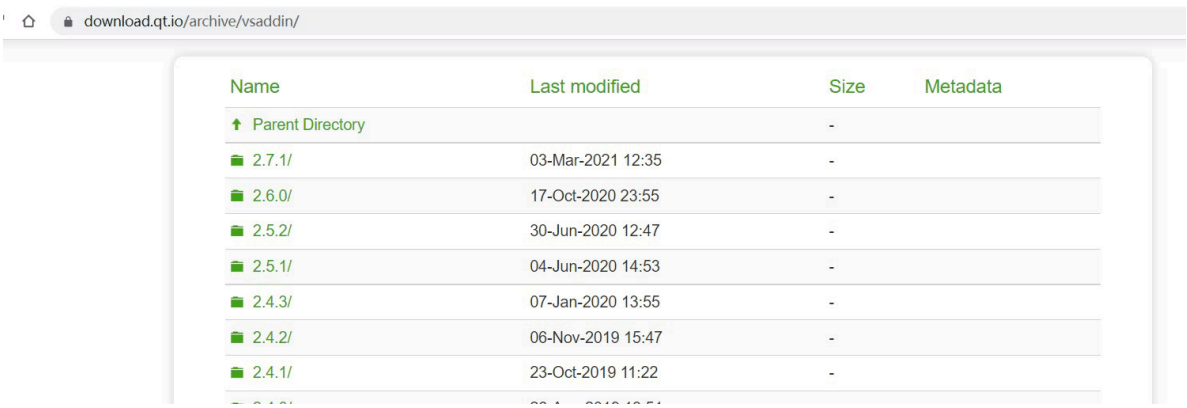
- Installation: First log into the qt account. if you haven't an qt account, register first. Next, the interface for selecting components will appear. You may select MinGW and MSVC on Windows, as shown in the following figure.



- Environment variable configuration: This Computer -> Right-click Properties -> Advanced System Settings -> Environment Variables -> Look at System Variables, click New-> Variable Name: QTDIR variable value: the directory where msvc2017_64 is located (eg: D:5.12.10\5.12.102017_64. For details, check the installation path in your computer), as shown in the following figure.

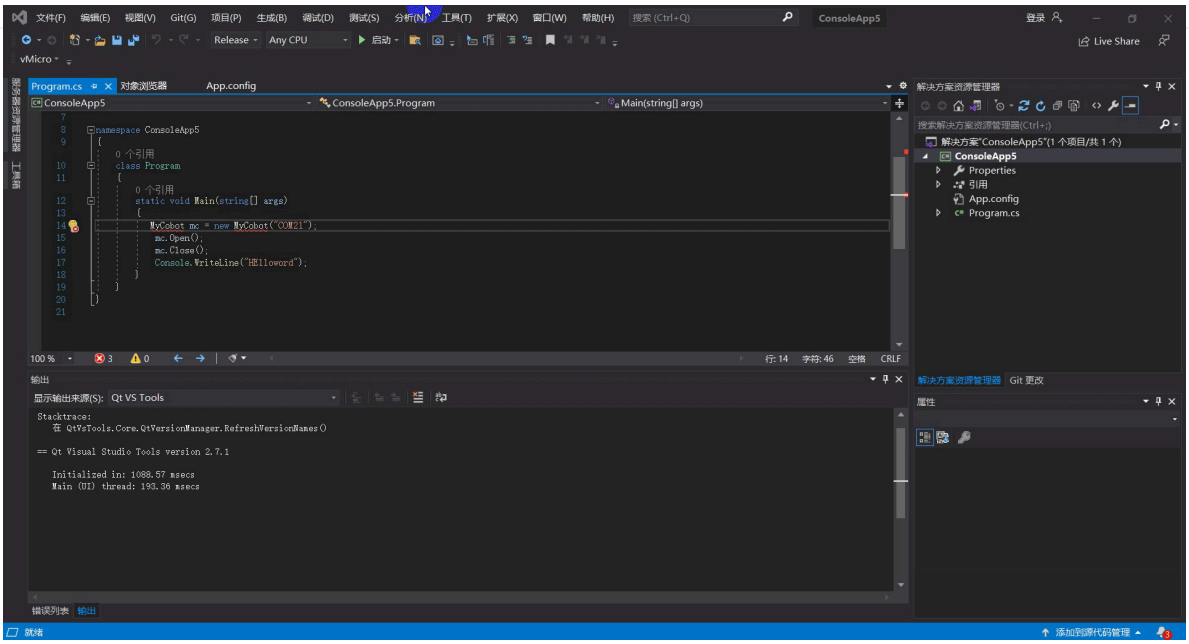
2.3 Installing qt plugin vsaddin

- Downloading: First select the vsaddin version corresponding to vs2019. The specific operation is shown in the figure below.



- installation: direct installation.

- configuration: vs2019 menu bar extension -> QT VS Tools -> QT Versions -> add new qtversion. For Path, select the path where msvc2017_64 is located (eg: D:5.12.10\5.12.102017_64). The specific operation is shown in the following figure:



3 Linux Environment Configuration

3.1 Installing qt5.12.10

- Downloading:

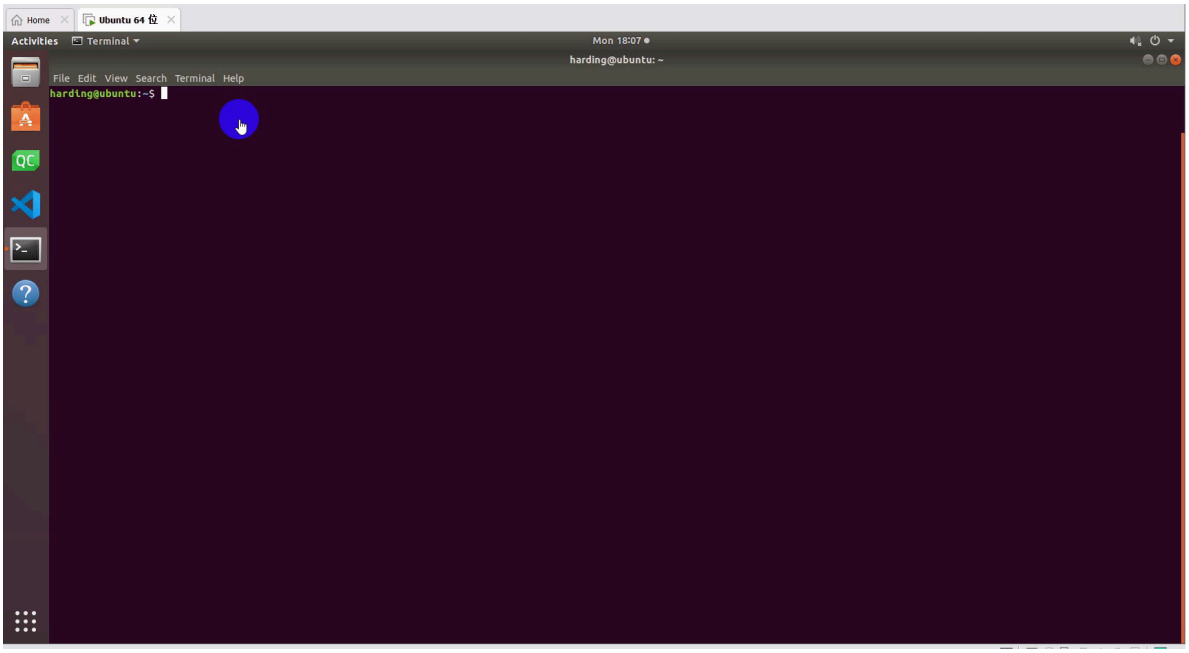
The download address is the same as Windows. Select the installation package on Linux. For details, see 8.1.2.2 above.

- Installation:

Command line installation: run ./"installation package name". If there is no execute permission, add execute permission: sudo chmod +x "installation package name", and then enter the graphical interface the same as Windows;

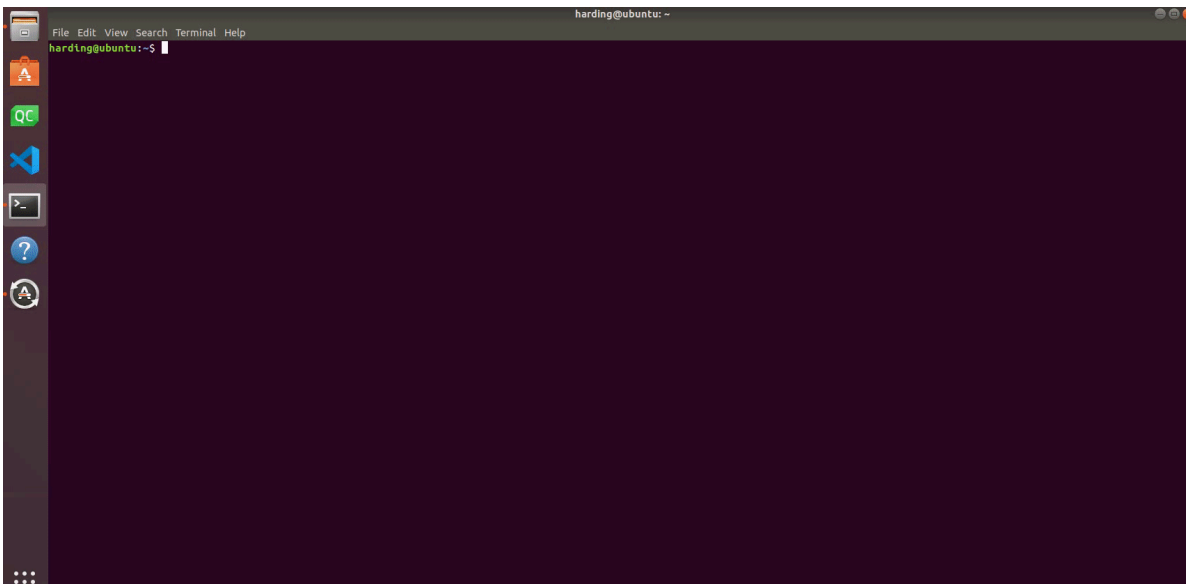
graphical interface installation: the same as Windows.

It is recommended to install qt directly with ordinary user rights. After the installation is successful, you can execute qmake –version, and the following interface appears: installation is successful:



- Configuration:

  Open the configuration file, install qt: vi ~/.bashrc for ordinary users, and install qt: vi ~/.profile for root users. Add in the configuration file: export QTDIR="the directory where qt is located" (eg: export QTDIR=$HOME/Qt/5.12.10/gcc_64), as shown in the following figure:



Compiling and running of mycobotCpp

1 Downloading

1.1 Downloading source code

Download MycobotCpp on github.

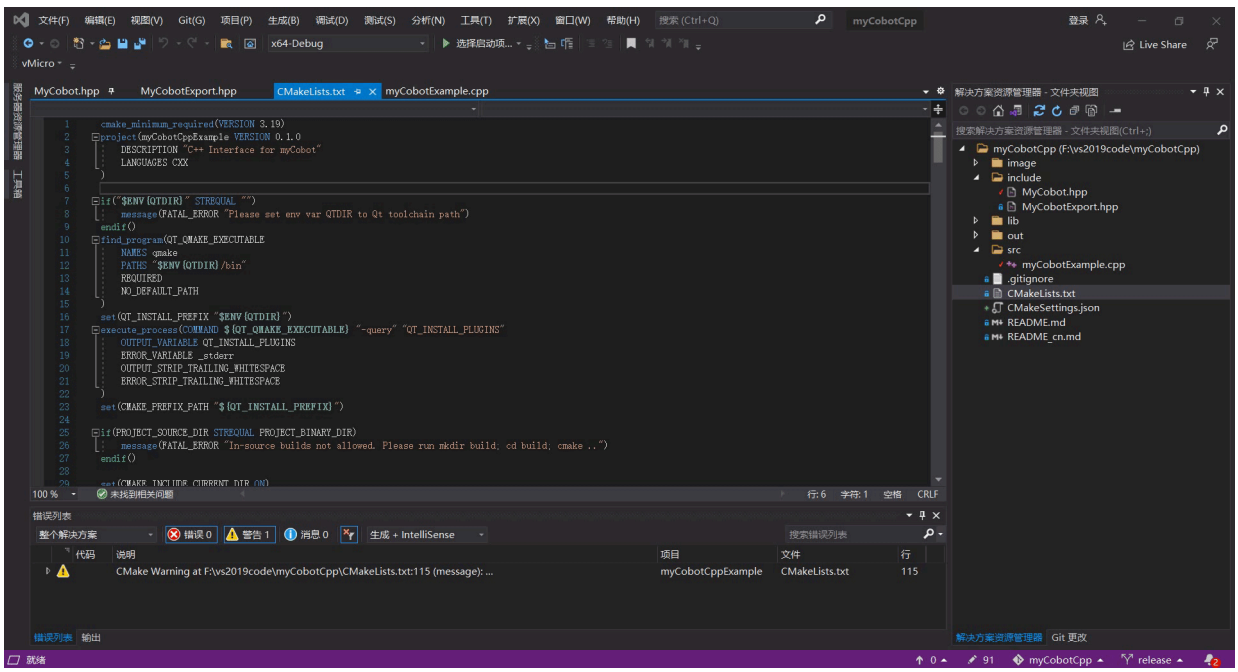
1.2 Dynamic library downloading

Dependency library downloading (Download the latest version, and pay attention to select Windows or Linux; the suffix.zip is the library required by Windows, and .tar.gz is the library required by Linux).

2 Running in Windows

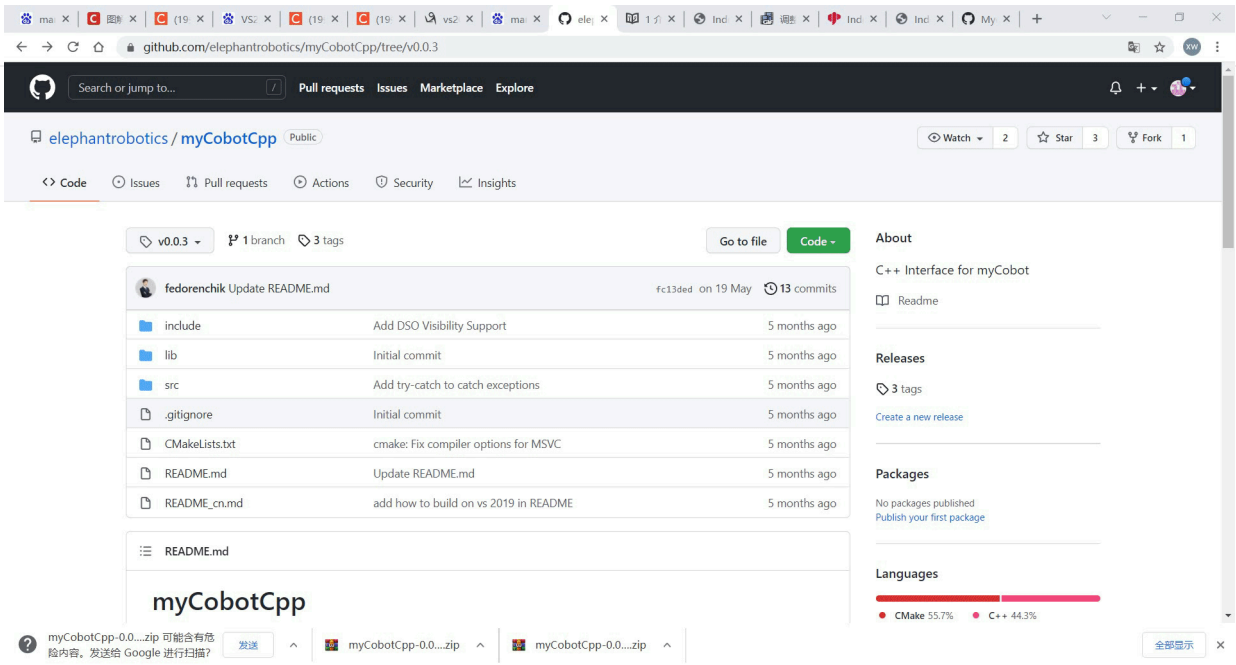
2.1 Compiling

In vs2019, open MycobotCpp, select x64-Release to compile (which is next to the startup item. If there is no x64-Release, click the drop-down box -> Manage configuration to add it), at the same time select release in the configuration of cmake setting, and finally click Generate. Note: Be sure to select x64-Release to compile, as shown in the following figure.



2.2 Running

- Add library files: add .lib to myCobotCpp/lib. Add .lib and .dll to the directory, of myCobotCppExample.exe, at the same level, such as out/build/x64-Release/bin (The two files .lib and .dll are in the myCobotCpp-0.0.3-windows-msvc-x86_64. zipped package).
- Running: Select the startup item (which is next to the green play button and also to the running button), drop down to select myCobotCppEXample.exe (bin.exe), and click Run, as shown in the following figure:



2.3 Common problems and solutions.

Run-time error:

- If myCobotCpp.dll is missing, put myCobotCpp.dll previously saved in the lib directory to the directory where mycobotcppexample.exe is located.
- If QT5Core.dll is reported to be missing, open the qt command (search for QT in the menu bar) and select msvc 2017 64 -bit, execute the directory where windeployqt –release myCobotCppExample.exe is located (for example: windeployqt –release D:201964-Release). If the VS installation path cannot be found after executing the command here, check the settings of VS environment variable.
- After executing the above steps, if the qt5serialport.dll file is reported to be missing, copy the file in the qt installation directory (such as: D:5.12.10\5.12.102017_64) to the directory where myCobotCppExample.exe is located.

3 Running in Linux

3.1 Compiling and building

- mkdir build && cd build
- cmake ..
- cmake –build .

3.2 Run

- copy all .so files to the lib directory;

- command line run: ./bin/myCobotCppExample (Here it is running in the build directory).

4 For example, running on Ubuntu20.04

4.1 Compiling

- mkdir build && cd build

- cmake ..

- cmake –build .

4.2 run

- copy all .so files to the lib directory (Note: after downloading, unzip it. Do not unzip it in Windows, and then copy to Ubuntu. Unzip it directly in Ubuntu, such as: tar -xvf and then drag the file directly to the terminal);

- Soft-link libQt5SerialPort.so.5 (in the QT installation directory, such as: /home/"username"/Qt5.12.10/5.12.10 /gcc_64/lib) to mycobotcpp/build/bin (Do not copy directly). The command is as follows (Be careful to choose your path): ln -s /home/"username"/Qt5.12.10/5.12.10/gcc_64/ lib/libQt5SerialPort.so.5 /home/“username”/myCobotCpp/build/bin/libQt5SerialPort.so.5

4.3 Common problems and solutions

- Error during compilation: QTDIR is not found. Solution: Check whether QTDIR is configured correctly. You may check it in the command line output: echo $QTDIR

- run-time error: Serial port problem: The serial port cannot be opened. Solution: Modify the permission for the serial port of the robot arm. You cannot directly chmod..., because the permission must be modified every time of restart. Modify the file directly:

- cd /etc/udev/rules.d

- sudo gedit 20-usb-serial.rules

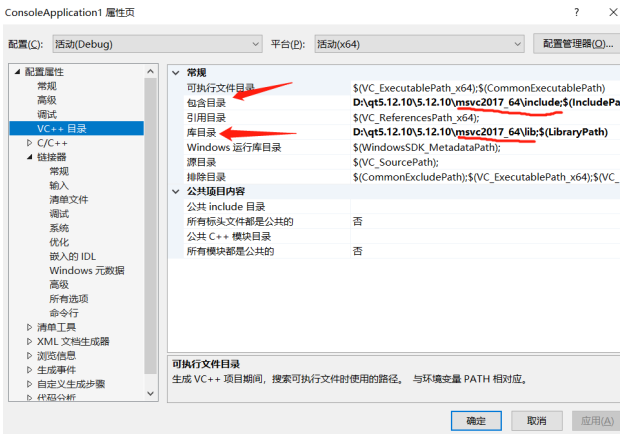
- Add in the file: KERNEL=="ttyUSB*" MODE="0777"

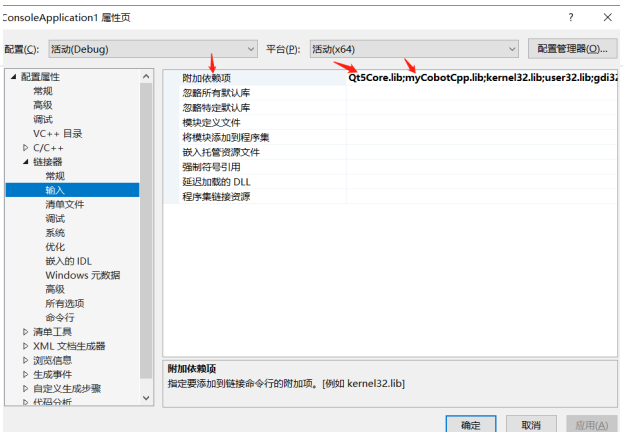
- The file cannot be found: such as libQt5SerialPort.so.5 could not be opened or found.

  Solution: Check the running step 2 above.

Notice

If you don't use cmake to compile, and if you use it directly in MFC, do configurations as shown in the figure below.




ConsoleApplication1 属性页
配置(C): 活动(Debug)
平台(P): 活动(x64)
配置管理器(O)...
配置属性
常规
高级
调试
VC++ 目录
C/C++
链接器
常规
输入
清单文件
调试
系统
优化
嵌入的 IDL
Windows 元数据
高级
所有选项
命令行
清单工具
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
附加依赖项
Qt5Core.lib;myCobotCpp.lib;kernel32.lib;user32.lib;gdi32
忽略所有默认库
忽略特定默认库
模块定义文件
将模块添加到程序集
嵌入托管资源文件
强制符号引用
延迟加载的 DLL
程序集链接资源
附加依赖项
指定要添加到链接命令行的附加项。[例如 kernel32.lib]
确定
取消
应用(A)

Joint control

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

1 Single joint control

1.1 Sending the angle of single joint

WriteAngle(Joint joint, double value, int speed = DefaultSpeed)

Return value: no parameter

parameter description: Parameter 1: joint number (1-6) Parameter 2: angle (-170°- 170°) Parameter 3: speed (0-100), the example with 30 as a default

example:

```
mycobot::MyCobot::I().WriteAngle(mycobot::Joint::J1, 10, 30);
```

2 Multi-joint control

2.1 Get the angles of all joints

GetAngles()

Return value: Angles type

parameter description: no example:

```
mycobot::Angles angles= mycobot::MyCobot::I().GetAngles();
```

2.2 Send the angles of all joints

WriteAngles(const Angles& angles, int speed = DefaultSpeed)

Return value: no

parameter description: Parameter 1: all angles (std::array, ranging from -170° to 170°) Parameter 2: speed (0-100), the example with 30 as a default

```
mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteAngles(goal_angles, 30);
mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteAngles(goal_angles, 30);
```

3 Complete use cases

```
int main(int argc, char* argv[])
{
    try {
        QCoreApplication a(argc, argv);
        using namespace std::chrono_literals;
        if (!mycobot::MyCobot::I().IsControllerConnected()) {
            std::cerr << "Robot is not connected\n";
            exit(EXIT_FAILURE);
        }
        std::cout << "Robot is connected\n";
        mycobot::MyCobot::I().PowerOn();
        mycobot::MyCobot::I().StopRobot();
        std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n";
        mycobot::Angles angles = mycobot::MyCobot::I().GetAngles();
        std::this_thread::sleep_for(200ms);
        mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", " << angles[mycobot::J3] << ", "
            << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]";
        mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 };
        mycobot::MyCobot::I().WriteAngles(goal_angles);
        while (!mycobot::MyCobot::I().IsInPosition(goal_angles, false)) {
            angles = mycobot::MyCobot::I().GetAngles();
            std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
                << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
                << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
            std::this_thread::sleep_for(200ms);
        }

        mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
        std::this_thread::sleep_for(5000ms);
        mycobot::MyCobot::I().StopRobot();

        std::cout << "\n";
        exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
        std::cerr << "System error. Exiting.\n";
        exit(EXIT_FAILURE);
    } catch (...) {
        std::cerr << "Unknown exception thrown. Exiting.\n";
        exit(EXIT_FAILURE);
    }
}
```

Coordinate control

Coordinate control is to make the robot arm move to a specified point with a specified posture, which is divided into x, y, z, rx, ry, rz. X, Y and Z represents the position of the robot arm head in space (The coordinate system is Cartesian coordinate system); [rx,ry,rz] represents the posture of such head at this point (The coordinate system is Euler coordinates).

1 Single parameter control

1.1 Sending single parameter coordinates

WriteCoord(Axis axis, double value, int speed = DefaultSpeed)

Return value: no

parameter description: Parameter 1: coordinate number (Axis enumeration type, int: 1-6 (X-RZ)); Parameter 2: coordinate (X, Y and Z range from -300 to 300.00 in mm; RX, RY, RZ ranges from -180 to 180); Parameter 3: speed (0-100), the example with 30 as a default

example:

```
mycobot::MyCobot::I().WriteCoord(mycobot::Axis::X, 10, 30);
```

2 Multi-parameter control

2.1 Get all coordinates

GetCoords() Return value: Coords type parameter description: no example: mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();

2.2 Send all coordinates

WriteCoords(const Coords& coords, int speed = DefaultSpeed)

Return value: no

parameter description: Parameter 1: coordinates (X, Y and Z range from -300 to 300.00 in mm; RX, RY and RZ range from -180 to 180); Parameter 2: speed (0-100), the example with 30 as a default

example:

```
mycobot::Coords goal_coords = {5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteCoords(goal_coords, 30);
mycobot::Coords goal_coords = {5, 5, 5, 5, 5, 5 };
mycobot::MyCobot::I().WriteCoords(goal_coords, 30);
```

3 Complete use cases

```
int main(int argc, char* argv[])
{
    try {
        QCoreApplication a(argc, argv);
        using namespace std::chrono_literals;
        if (!mycobot::MyCobot::I().IsControllerConnected()) {
            std::cerr << "Robot is not connected\n";
            exit(EXIT_FAILURE);
        }
        std::cout << "Robot is connected\n";
        mycobot::MyCobot::I().PowerOn();
        mycobot::MyCobot::I().StopRobot();
        std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n";
        mycobot::Angles angles = mycobot::MyCobot::I().GetAngles();
        std::this_thread::sleep_for(200ms);
        mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", " << angles[mycobot::J3] << ", "
            << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]";
        mycobot::Angles goal_angles = { 5, 5, 5, 5, 5, 5 };
        mycobot::MyCobot::I().WriteAngles(goal_angles);
        while (!mycobot::MyCobot::I().IsInPosition(goal_angles, false)) {
            angles = mycobot::MyCobot::I().GetAngles();
            std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
                << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
                << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
            std::this_thread::sleep_for(200ms);
        }

        mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
        std::this_thread::sleep_for(5000ms);
        mycobot::MyCobot::I().StopRobot();

        std::cout << "\n";
        exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
        std::cerr << "System error. Exiting.\n";
        exit(EXIT_FAILURE);
    } catch (...) {
        std::cerr << "Unknown exception thrown. Exiting.\n";
        exit(EXIT_FAILURE);
    }
}
```

io control

There are pins on the M5Stack-basic at the bottom of the robot arm and on the Atom at the end. The adsorption pump and other tools can be controlled by setting high and low levels of the pins through io control. For the numbers of input and output pins for each type of robot arm, refer to the following description table.

Table of Description of Input and Output Pins on the M5Stack-basic

Robot arm model	Input pin number	Output pin number
myCobot 280-M5	35、36	2、5、26
myCobot 320-M5	35、36	5、15

Table of Description of Input and Output Pins on the Atom

Robot arm model	Input pin number	Output pin number
myCobot 280-M5	19、22	23、33
myCobot 320-M5	无	无

1 M5Stack-basic io control (m5)

1.1 Setting the high and low levels of output io

SetBasicOut(int pin_number, int pin_signal)

Return value: no

parameter description: Parameter 1: pin number (basic output pin number); Parameter 2: state (0 - low level; 1- high level)

Case: set the output pin 2 to high level

```
mycobot::MyCobot::I().SetBasicOut(2, 1);
```

1.2 Getting the state of input io

GetBasicIn(int pin_number)

Return value: pin state (0-low level; 1-high level)

Parameter description: pin number (basic input pin number)

Case:

```
mycobot::MyCobot::I().GetBasicIn(35);
```

2 Atom io Control

Note: 320m5 has no atom io, so the module API is not used.

2.1 Setting the high and low levels of output io

SetDigitalOut(int pin_number, int pin_signal)

Return value: no

parameter description: Parameter 1: pin number (atom output pin number); Parameter 2: state (0 - low level; 1- high level)

Case:

```
mycobot::MyCobot::I().SetDigitalOut(23, 1);
```

2.2 Getting the state of input io

GetDigitalIn(int pin_number)

Return value: pin state (0-low level; 1-high level)

Parameter description: pin number (atom input pin number)

Case:

```
mycobot::MyCobot::I().GetDigitalIn(19);
```

3 Complete use cases

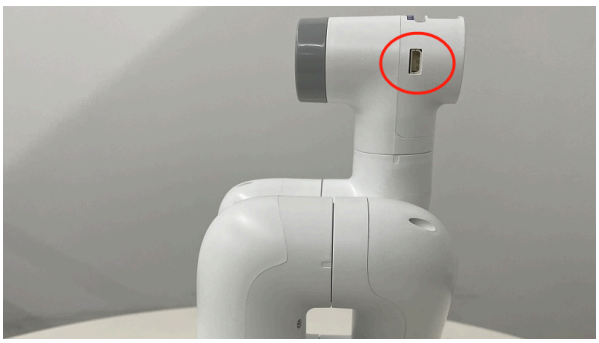
```
int main(int argc, char* argv[])
{
    try {
        QCoreApplication a(argc, argv);
        using namespace std::chrono_literals;
        if (!mycobot::MyCobot::I().IsControllerConnected()) {
            std::cerr << "Robot is not connected\n";
            exit(EXIT_FAILURE);
    }
    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn();
    mycobot::MyCobot::I().SleepSecond(1);

    mycobot::MyCobot::I().SetBasicOut(2, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(5, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(26, 1);
    mycobot::MyCobot::I().SleepSecond(1);

    /*for (int i = 0; i < 2; i++) {
        std::cout << "35= " << mycobot::MyCobot::I().GetBasicIn(35) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "36= " << mycobot::MyCobot::I().GetBasicIn(36) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/

    /*mycobot::MyCobot::I().SetDigitalOut(23, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetDigitalOut(33, 1);
    mycobot::MyCobot::I().SleepSecond(1);*/

    /*for (int i = 0; i < 2; i++) {
        std::cout << "22= " << mycobot::MyCobot::I().GetDigitalIn(22) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "19= " << mycobot::MyCobot::I().GetDigitalIn(19) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/

    /*for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetGriper(1);
        mycobot::MyCobot::I().SleepSecond(3);
        mycobot::MyCobot::I().SetGriper(0);
        mycobot::MyCobot::I().SleepSecond(3);
    }*/

    /*for (int i = 0; i < 2; i++) {
```

```
        mycobot::MyCobot::I().SetElectricGriper(1);
        mycobot::MyCobot::I().SleepSecond(1);
        mycobot::MyCobot::I().SetElectricGriper(0);
        mycobot::MyCobot::I().SleepSecond(1);
    }*/
    /*mycobot::MyCobot::I().StopRobot();
    std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n";
    mycobot::Angles angles = mycobot::MyCobot::I().GetAngles();
    std::this_thread::sleep_for(200ms);
    mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();
    angles = mycobot::MyCobot::I().GetAngles();
    std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", " << angles[mycobot::J3] << ", "
        << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]";
    mycobot::Angles goal_angles = { 1, 0, 0, 0, 0, 0 };
    mycobot::MyCobot::I().WriteAngles(goal_angles,180);
    while (!mycobot::MyCobot::I().IsInPosition(goal_angles, false)) {
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
            << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
            << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    //mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
    std::this_thread::sleep_for(5000ms);
    mycobot::MyCobot::I().StopRobot();*/

    std::cout << "\n";
    exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
    } catch (...) {
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
    }
}
```

Gripper control

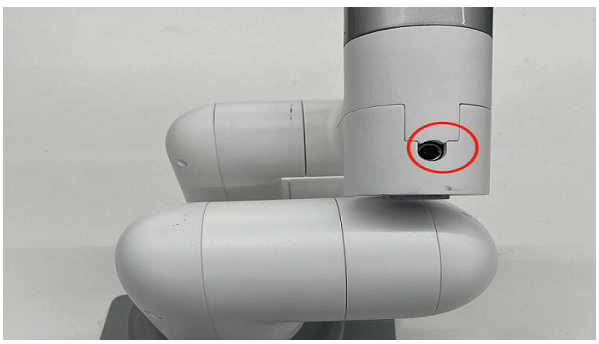
Installing the gripper:

- For an adaptive gripper, insert it on the pin on the atom, as shown in the following figure:



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

Note: myCobot 280-m5 has no electric gripper, only myCobot 320-m5 has an electric gripper



1 Adaptive gripper control

supports: myCobot280, 320 && myPalletizer 260

SetGriper(int open)

Return value: no

parameter description: gripper switch state (0-off; 1-on)

Case: Due to the delay, when the gripper is controlled for the first time, the gripping may not be successful. Therefore it is recommended to send twice

```
for (int i = 0; i < 2; i++) {<br>
    mycobot::MyCobot::I().SetGriper(1);
    mycobot::MyCobot::I().SleepSecond(3);
    mycobot::MyCobot::I().SetGriper(0);
    mycobot::MyCobot::I().SleepSecond(3);
}
```

2 Electric gripper control

Available for: myCobot320

SetElectricGriper(int open)

Return value: no

parameter description: gripper switch state (0-off; 1-on)

Case: Due to the delay, when the gripper is controlled for the first time, the gripping may not be successful. Therefore it is recommended to send twice

```
for (int i = 0; i < 2; i++) {<br>
    mycobot::MyCobot::I().SetElectricGriper(1);<br>
    mycobot::MyCobot::I().SleepSecond(1);<br>
    mycobot::MyCobot::I().SetElectricGriper(0);<br>
    mycobot::MyCobot::I().SleepSecond(1);<br>
}
```

3 Complete use cases

```
int main(int argc, char* argv[])
{
    try {
    QCoreApplication a(argc, argv);
    using namespace std::chrono_literals;
    if (!mycobot::MyCobot::I().IsControllerConnected()) {
        std::cerr << "Robot is not connected\n";
        exit(EXIT_FAILURE);
    }
    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn();

    mycobot::MyCobot::I().SleepSecond(1);

    mycobot::MyCobot::I().SetBasicOut(2, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(5, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(26, 1);
    mycobot::MyCobot::I().SleepSecond(1);

    /*for (int i = 0; i < 2; i++) {
        std::cout << "35= " << mycobot::MyCobot::I().GetBasicIn(35) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "36= " << mycobot::MyCobot::I().GetBasicIn(36) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/

    /*mycobot::MyCobot::I().SetDigitalOut(23, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetDigitalOut(33, 1);
    mycobot::MyCobot::I().SleepSecond(1);*/

    /*for (int i = 0; i < 2; i++) {
        std::cout << "22= " << mycobot::MyCobot::I().GetDigitalIn(22) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "19= " << mycobot::MyCobot::I().GetDigitalIn(19) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/

    for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetGriper(1);
        mycobot::MyCobot::I().SleepSecond(3);
        mycobot::MyCobot::I().SetGriper(0);
        mycobot::MyCobot::I().SleepSecond(3);
    }
```

```
    /*for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetElectricGriper(1);
        mycobot::MyCobot::I().SleepSecond(1);
        mycobot::MyCobot::I().SetElectricGriper(0);
        mycobot::MyCobot::I().SleepSecond(1);
    }*/
    /*mycobot::MyCobot::I().StopRobot();
    std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n";
    mycobot::Angles angles = mycobot::MyCobot::I().GetAngles();
    std::this_thread::sleep_for(200ms);
    mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();
    angles = mycobot::MyCobot::I().GetAngles();
    std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", " << angles[mycobot::J3] << ", "
        << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]";
    mycobot::Angles goal_angles = { 1, 0, 0, 0, 0, 0 };
    mycobot::MyCobot::I().WriteAngles(goal_angles,180);
    while (!mycobot::MyCobot::I().IsInPosition(goal_angles, false)) {
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
            << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
            << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    //mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
    std::this_thread::sleep_for(5000ms);
    mycobot::MyCobot::I().StopRobot();*/

    std::cout << "\n";
    exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
    } catch (...) {
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
    }
}
```

MyCobot API

When using the following function interfaces, import our API library first. Otherwise it is impossible to run successfully. To download and import the library, refer to 8.2 Compiling and running of MycobotCpp.

1 Instantiate MyCobot

1.1 **I()**

Function: Instantiate MyCobot

return value: MyCobot type, single instance of myCobot object

parameter description: none

Note: When calling the following API, it is not necessary to do separate instantiation, and just call this API

2 Overall running status of Robot

2.1 **PowerOn()**

Function: to power up the robot arm

Return value: none

parameter description: none

Note: After the robot arm is powered up, it cannot be moved manually

2.2 **PowerOff()**

Function: to cut off teh power for the robot arm

Return value: none

parameter description: none

Note: After the robot arm is powered up, if you want to move it manually, you may call this API

2.3 **SetFreeMoveMode(bool free_move = true)**

Function: to set free movement mode

Return value: none

parameter description: enable or disable free movement; true – open free movement, and false – close free automatic

Note: After free movement is enabled, you may move the robot arm manually; and at the same time the light on the atom will turn yellow, and it will turn green when it is turned off

2.4 **IsFreeMoveMode()**

Function: to check whether the current free movement mode

Return value: bool type, true - free movement is enabled, false - free movement is not enabled

parameter description: none

2.5 **IsControllerConnected()**

Function: to check whether the system is normal

return value: none

parameter description: bool type, if false is returned, the robot arm cannot be controlled

3 MDI program control mode

3.1 **IsInPosition(const Coords& coords, bool is_linear = true)**

Function: to check whether the robot arm reaches a specified point (angle or coordinate)

Return value: bool type; if false is returned , it means failing to reach the specified point; and if true is returned, it means having reached the specified point

Parameter description: Parameter 1: all angles or coordinates; Parameter 2: 0 or 1 (Coordinate is 1 (true), and

angle is 0 (false))

3.2 **IsMoving()**

Function: to detect whether the robot arm is moving

Return value: bool type, true - moving, false - not moving

Parameter description: none

3.3 **WriteAngle(Joint joint, double value, int speed = DefaultSpeed)**

Function: to send single joint angle

Return value: none

parameter description: Parameter 1: joint number (1-6), Parameter 2: angle (-170°- 170°), and Parameter 3: speed (0-100), and the default is 30

3.4 **GetAngles()**

Function: to get the angles of all joints

Return value: Angles type

Parameter description: none

3.5 **WriteAngles(const Angles& angles, int speed = DefaultSpeed)**

Function: to send the angles of all joints

Return value: none

parameter description: Parameter 1: all angles (std::array, and ranging from -170° to 170°), and Parameter 2: speed (0-100) with a default of 30

3.6 **WriteCoord(Axis axis, double value, int speed = DefaultSpeed)**

Function: to send single parameter coordinate

Return value: none

parameter description: Parameter 1: coordinate number (Axis enumeration type, int: 1-6 (X-RZ)), Parameter 2: coordinate (X, Y and Z range form -300 to 300.00mm; RX, RY and RZ range from -180 to 180), and Parameter 3: speed (0-100), and the default is 30

3.7 **GetCoords()**

Function: to get all coordinates

Return value: Coords type

Parameter description: none

3.8 **WriteCoords(const Coords& coords, int speed = DefaultSpeed)**

Function: to send all coordinates

Return value: none

Parameter description: Parameter 1: coordinate (X, Y and Z range form -300 to 300.00mm; RX, RY and RZ range from -180 to 180), and Parameter 2: speed (0-100), the default is 30

3.9 **StopRobot()**

Function: to stop the movement of the robot arm. When the robot arm is moving, you may stop it by calling this API

Return value: none

Parameter description: none

4 Running Auxiliary Information

4.1 **GetSpeed()**

Function: to get the speed of the robot arm

Return value: int type, the movement speed of the robot arm (0-100)

Parameter description: none

4.2 **SetSpeed(int percentage)**

Function: to set the movement speed of the robot arm

Return value: none

Parameter description: the movement speed of the robot arm (0-100)

4.3 **GetJointMin(Joint joint)**

Function: to read the minimum angle of a joint

Return value: double type, the minimum angle (the minimum angle that the joint can reach)

Parameter description: joint number (1-6)

4.4 **GetJointMax(Joint joint)**

Function: to read the maximum angle of a joint

Return value: double type, the maximum angle (the maximum angle that the joint can reach)

Parameter description: joint number (1-6)

4.5 **SleepSecond(unsigned time)**

Function: to wait

Return value: none

Parameter description: time unit is second

5 JOG mode and operation

5.1 **JogCoord(Axis axis, int direction, int speed = DefaultSpeed)**

Function: to make the robot arm move in the direction of a coordinate axis

Return value: none

Parameter description: Parameter 1: coordinate number (1-6, xyz rx ry rz), Parameter 2: direction (1 - positive direction, and 0 - negative direction), and Parameter 3: speed (Default is 30, ranging from 0 to 100). Notice: This API will make the robot arm move in the forward and reverse directions of the coordinate axis, and it will stop moving after reaching a limited position or when JogStop is called during the movement

5.2 **JogAngle(Joint joint, int direction, int speed = DefaultSpeed)**

Function: to make a joint move until Jogstop or reaching a limited position

Return value: none

Parameter description: Parameter 1: joint number (1-6), Parameter 2: direction (1 - positive direction, and 0 - negative direction), and Parameter 3: speed (Default 30, ranging from 0 to 100). Notice: This API will make the robot arm move in the forward and reverse directions, and it will stop moving after reaching the limited position or when JogStop is called during the movement

5.3 **JogCoordAbsolute(Axis axis, double value, int speed = DefaultSpeed)**

Function: to make a coordinate to move to a specified coordinate

Return value: none

Parameter description: Parameter 1: coordinate number (1-6, x y z rx ry rz), Parameter 2: coordinate (X, Y and Z range form -300 to 300.00mm; and RX, RY and RZ range from -180 to 180), and Parameter 3: speed (with a default of 30 and ranging from 0 to 100)

5.4 **JogAngleAbsolute(Joint joint, double value, int speed = DefaultSpeed)**

Function: to make a joint move to a specified angle

Return value: none

Parameter description: Parameter 1: joint number (1-6), Parameter 2: angle (ranging from -170 to 170), and Parameter 3: speed (with a default of 30 and ranging from 0 to 100)

5.5 **JogCoordIncrement(Axis axis, double increment, int speed = DefaultSpeed)**

Function: to make a coordinate move by a set coordinate increment

Return value: none

Parameter description: Parameter 1: coordinate number (1-6, xyz rx ry rz), Parameter 2: coordinate increment value, and Parameter 3: speed (with a default of 30, and ranging from 0 to 100). Notice: The robot arm performs discrete motion; for example, the current coordinate of X-axis is 100, and the incremental value is 50. After the movement, the coordinate of X-axis will be 150

5.6 **JogAngleIncrement(Joint joint, double increment, int speed = DefaultSpeed)**

Function: to make a joint move by a set angle increment

Return value: none

Parameter description: Parameter 1: joint number (1-6), Parameter 2: joint increment value, and Parameter 3: speed (with a default of 30, and ranging from 0 to 100). Notice: The robot arm performs discrete motion; for example, the current coordinate of the joint 1 is -100, and the incremental value is 50. After the movement, the coordinate of the joint 1 will be 50

6 Atom end IO control

6.1 **SetDigitalOut(int pin_number, int pin_signal)**

Function: to set the high and low levels of output io

Return value: none

Parameter description: Parameter 1: pin number (atom output pin number), and Parameter 2: state (0-low level, and 1-high level)

6.2 **GetDigitalIn(int pin_number)**

Function: to get state of input io

Return value: pin state (0-low level, and 1-high level)

Parameter description: pin number (atom input pin number)

6.3 **SetGriper(int open)**

Function: to control the adaptive gripper

Return value: none

Parameter description: gripper switch state (0-off, and 1-on)

6.4 **SetElectricGriper(int open)**

Function: to control the electric gripper

Return value: none

Parameter description: gripper switch state (0-off, and 1-on)

7 Stand M5Stack-basic IO control

7.1 **SetBasicOut(int pin_number, int pin_signal)**

Function: to set the high and low levels of output io

Return value: none

Parameter description: Parameter 1: pin number (basic output pin number), and Parameter 2: state (0-low level, and 1-high level)

7.2 **GetBasicIn(int pin_number)**

Function: to get state of input io

Return value: pin state (0-low level, and 1-high level)

Parameter description: pin number (basic input pin number)

Use cases

In this case, three output pins of m5 are set to high level. and then make the robot arm moves to zero point. The program ends when the robot arm reaches the zero point. The myCobotExample.cpp in the project is a use case. You may modify it based on your needs:

```
int main(int argc, char* argv[])
{
    try {
    QCoreApplication a(argc, argv);
    using namespace std::chrono_literals;
    if (!mycobot::MyCobot::I().IsControllerConnected()) {
        std::cerr << "Robot is not connected\n";
        exit(EXIT_FAILURE);
    }
    std::cout << "Robot is connected\n";
    mycobot::MyCobot::I().PowerOn();

    mycobot::MyCobot::I().SleepSecond(1);



    mycobot::MyCobot::I().SetBasicOut(2, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(5, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetBasicOut(26, 1);
    mycobot::MyCobot::I().SleepSecond(1);



    /*for (int i = 0; i < 2; i++) {
        std::cout << "35= " << mycobot::MyCobot::I().GetBasicIn(35) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "36= " << mycobot::MyCobot::I().GetBasicIn(36) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/



    /*mycobot::MyCobot::I().SetDigitalOut(23, 1);
    mycobot::MyCobot::I().SleepSecond(1);
    mycobot::MyCobot::I().SetDigitalOut(33, 1);
    mycobot::MyCobot::I().SleepSecond(1);*/



    /*for (int i = 0; i < 2; i++) {
        std::cout << "22= " << mycobot::MyCobot::I().GetDigitalIn(22) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
        std::cout << "19= " << mycobot::MyCobot::I().GetDigitalIn(19) << std::endl;
        mycobot::MyCobot::I().SleepSecond(1);
    }*/



    /*for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetGriper(1);
        mycobot::MyCobot::I().SleepSecond(3);
        mycobot::MyCobot::I().SetGriper(0);
        mycobot::MyCobot::I().SleepSecond(3);
```

```
    }*/



    /*for (int i = 0; i < 2; i++) {
        mycobot::MyCobot::I().SetElectricGriper(1);
        mycobot::MyCobot::I().SleepSecond(1);
        mycobot::MyCobot::I().SetElectricGriper(0);
        mycobot::MyCobot::I().SleepSecond(1);
    }*/
    mycobot::MyCobot::I().StopRobot();
    std::cout << "Robot is moving: " << mycobot::MyCobot::I().IsMoving() << "\n";
    mycobot::Angles angles = mycobot::MyCobot::I().GetAngles();
    std::this_thread::sleep_for(200ms);
    mycobot::Coords coords = mycobot::MyCobot::I().GetCoords();
    angles = mycobot::MyCobot::I().GetAngles();
    std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", " << angles[mycobot::J3] << ", "
        << angles[mycobot::J4] << ", " << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]";
    mycobot::Angles goal_angles = { 1, 0, 0, 0, 0, 0 };
    mycobot::MyCobot::I().WriteAngles(goal_angles,180);
    while (!mycobot::MyCobot::I().IsInPosition(goal_angles, false)) {
        angles = mycobot::MyCobot::I().GetAngles();
        std::cout << "[" << angles[mycobot::J1] << ", " << angles[mycobot::J2] << ", "
            << angles[mycobot::J3] << ", " << angles[mycobot::J4] << ", "
            << angles[mycobot::J5] << ", " << angles[mycobot::J6] << "]" << std::flush;
        std::this_thread::sleep_for(200ms);
    }

    //mycobot::MyCobot::I().JogAngle(mycobot::Joint::J1, 1, 5);
    std::this_thread::sleep_for(5000ms);
    mycobot::MyCobot::I().StopRobot();

    std::cout << "\n";
    exit(EXIT_SUCCESS);
    } catch (std::error_code&) {
    std::cerr << "System error. Exiting.\n";
    exit(EXIT_FAILURE);
    } catch (...) {
    std::cerr << "Unknown exception thrown. Exiting.\n";
    exit(EXIT_FAILURE);
    }
}
```

C#

Using C# language, you can make developments (coordinate control, angle control, io control, gripper control, etc.) freely through the c# dynamic library provided by our company, and control some of the robots which have been developed by us.

Available for: **myCobot280, 320 and myPalletizer 260**.

What is C#?

C# is an object-oriented programming language derived from C and C++ released by Microsoft, and an advanced programming language running on .NET Framework and .NET Core (completely open source and cross-platform).

C# is obviously different from Java. It draws on a feature of Delphi and is directly integrated with COM (Component Object Model); and it is the protagonist of .NET windows network framework of Microsoft.

C# enables C++ programmers to develop programs efficiently, and because native functions written with C/C++ may be called, the original powerful functions of C/C++ will never be affected. Because of this inheritance

relationship, C# is very similar to C/C++, and the developers familiar with similar languages are able to quickly turn to C#.

Applicable equipment:

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

- myCobot 320
  - myCobot 320 M5

Preconditions for use：

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

Programming development

Some integrated development environments (IDE)

Visual Studio (Visual C#)

MonoDevelop

C# Environment building

1 Confirming development goals

Mycobot.csharp is a program used for serial communication with a robot. It contains simple use cases. If you want to make developments using C# to control the robots that have been developed by us, it is your choice.

Available for: **myCobot280, 320 and myPalletizer 260**.

Run the software recommended by Mycobot.csharp: vs2019 (Windows development), and MonoDevelop (Development on the Raspberry Pi robot arm).

2 Windows Environment Configuration

2.1 Installing vs2019

- Downloading: First download vs2019 from the official website.

- Installation: After installation is complete, the interface shown in the figure below will appear. .NET desktop development is mainly chosen (This is just a suggestion, you can choose according to your own needs. The installation time of vs2019 is long).

3 Environment configuration of Raspberry Pi robot arm

3.1 Installing monodevelop

- Installation

  Execute the following commands in order to install it. You may also view the instructions on the official website.

  sudo apt install apt-transport-https dirmngr

  sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E0328081BFF6A14DA29AA6A19B38D3D831EF echo "deb https://download.mono-project.com/repo/ubuntu vs-bionic main" | sudo tee /etc/apt/sources.list.d/mono-official-vs.list

  sudo apt update

  sudo apt-get install monodevelop

- Test

  Test whether the installation is successful. Check this document.

Compiling and Running of a MycobotCpp Case

1 Downloading

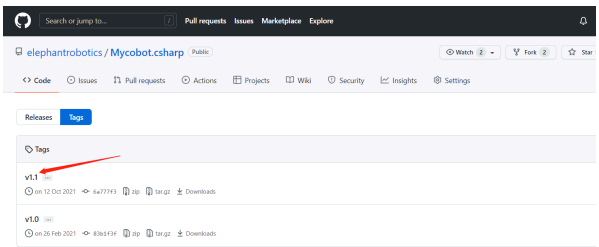
1.1 Downloading source code

Download Mycobot.csharp from github.

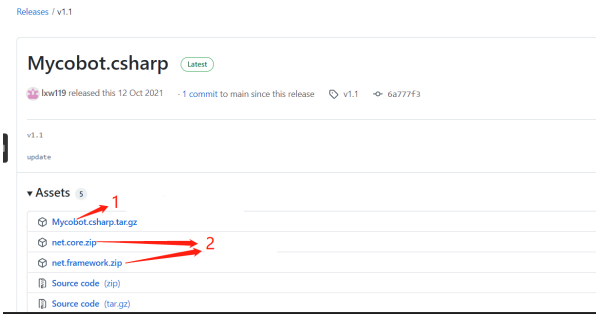
1.2 Downloading a dynamic library

To run the case, you need to use this dynamic library, which encapsulates the API to control the robot arm:

- Select the latest version, as shown in the following figure:



- The dynamic library is divided into two versions: Windows (Windows is divided into .net and .net framework. For the way to distinguish them, see running under Windows below) and Raspberry Pi system, as shown in the following figure.

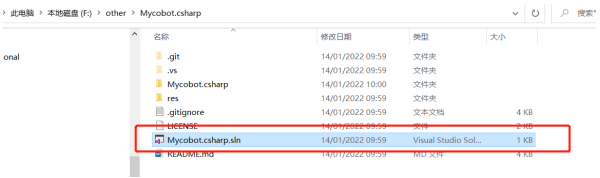


1 适用于树莓派机械臂系统

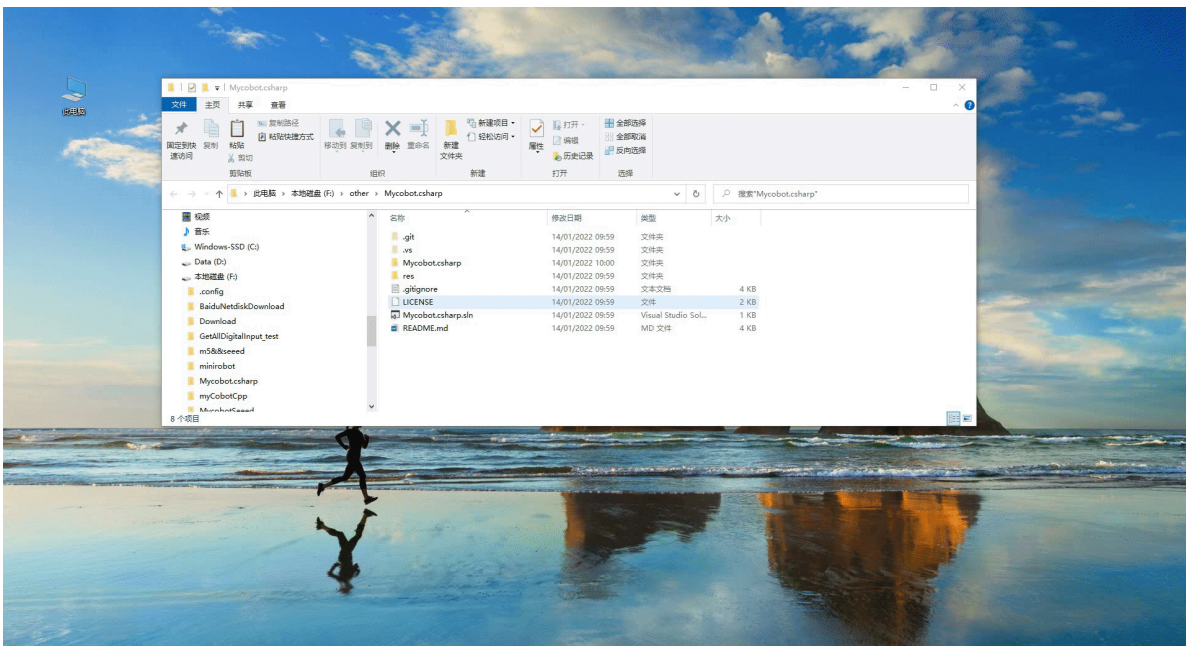
2 Running in Windows

2.1 Directly running the Mycobot.csharp case downloaded from github.

- Double-click Mycobot.csharp.sln to open it (Make sure that vs2019 has been installed in the computer. If not, see 9.1 Environment building)

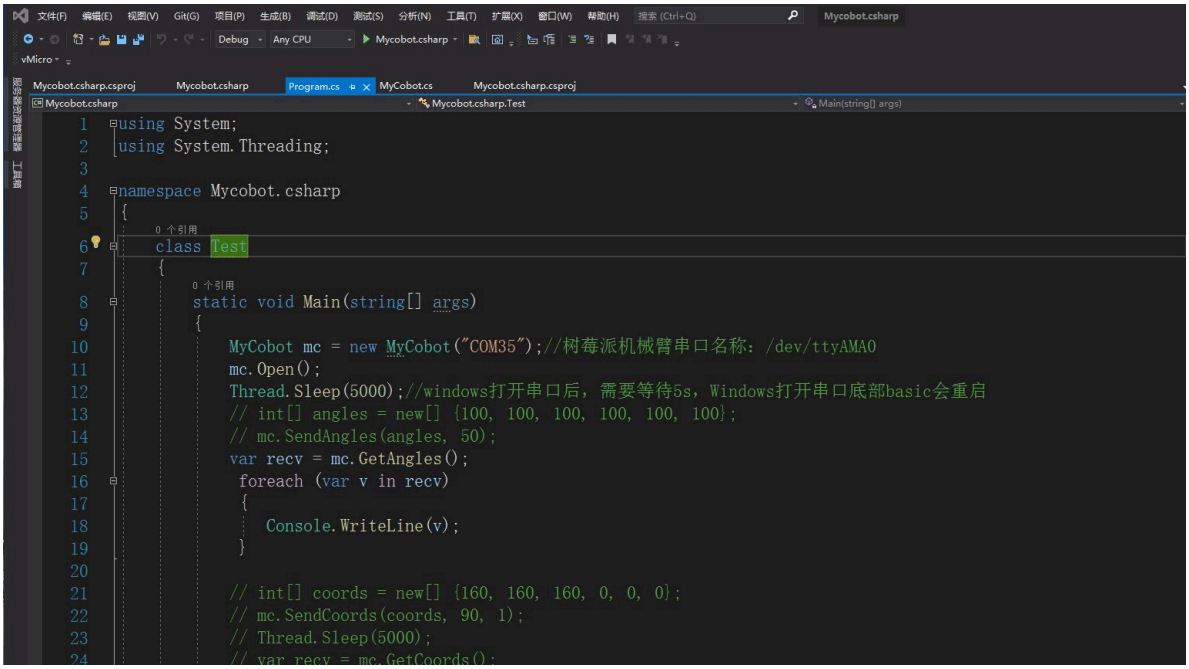


- Compile and run the project, and check the serial port number of the robot arm. If it is inconsistent with the example, modify the serial port number, as shown in the picture below.

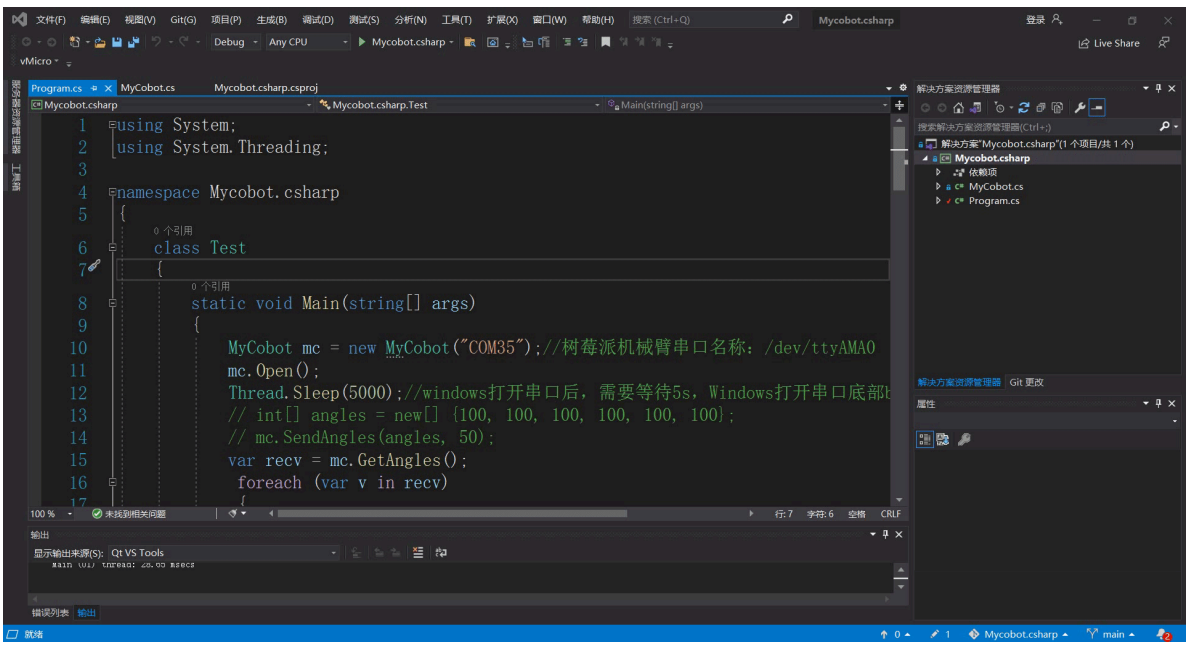


2.2 Calling a Mycobot.csharp dynamic library in your own project

- Check the target frame of the project, and then download the dynamic library corresponding thereto. If the target frame is .net core, download net core/Mycobot.csharp.dll; if it is .net framework, download net framework/Mycobot.csharp.dll)

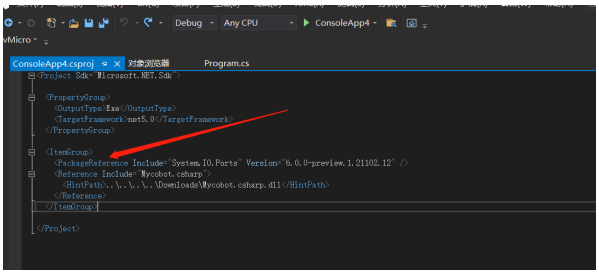


- Import Mycobot.csharp.dll into the project.

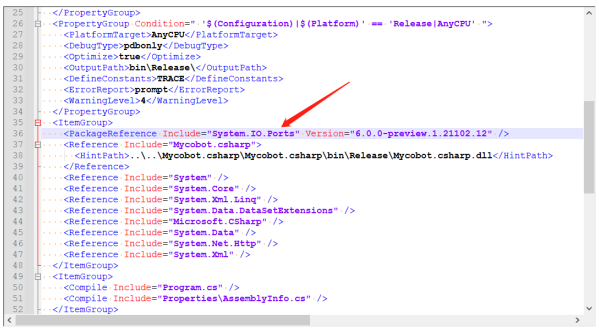


- Add system.io.ports to .csproj (the project name, and the file is located in the directory of the project), as shown in the picture below:

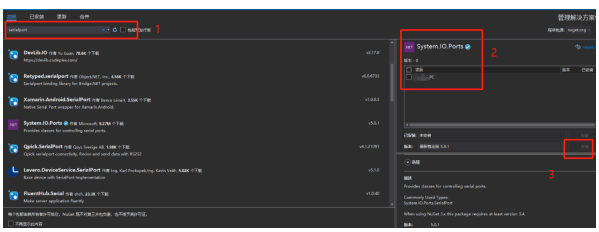
frame: .net core



frame: .net framework



In versions prior to vs2019, SerialPort can be used by simply using System.IO.Ports. If you get the error: No type name found in the namespace, you need to configure the dll for your project as follows: Tools - >Nuget Package Manager (N) - > Manage solution Nuget package (N) - > Browse, search for the corresponding dll(e.g. SerialPort) in the left bar, check the item you want to add on the right, click Download and install.

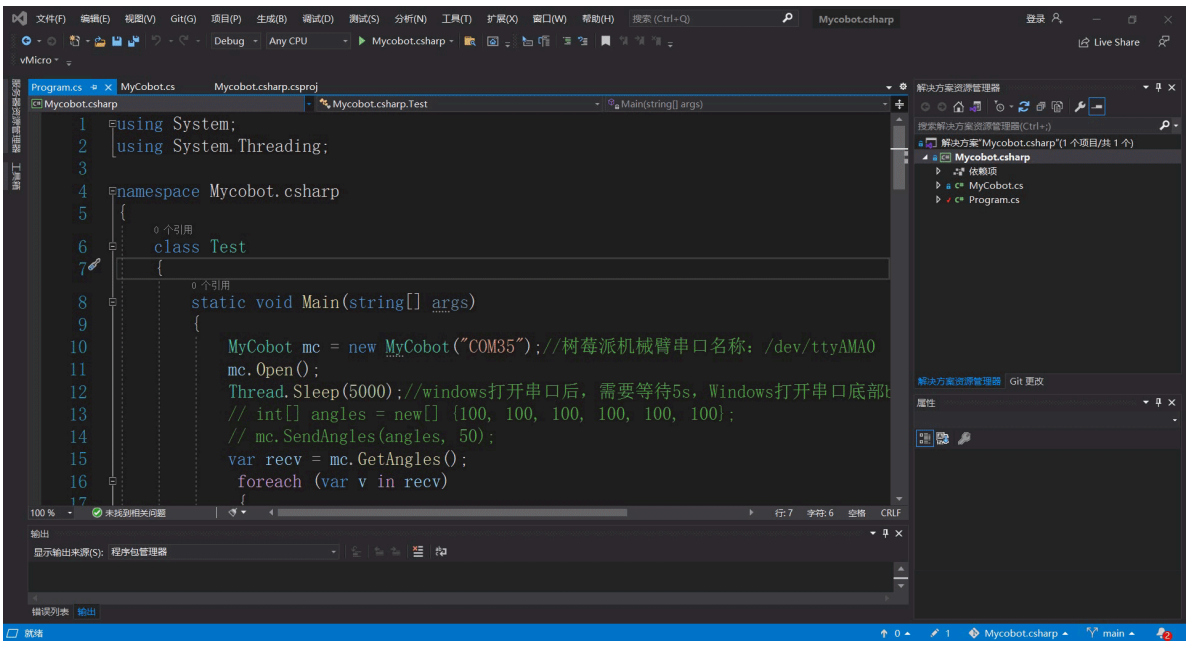


- For the use of library functions, see 9.7 Mycobot API, 9.8 Use Cases and the separate use of joints, coordinates, etc. in the following chapters.

2.3 Problems

Problems that may be encountered during use:

- Problem: System.Runtime, Version=5.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a' or one of its dependencies… Solution: update your sdk (if .net core, update to 5.0 and choose, if .net framework update to 4.0 and choose 4.7.2), watch the following animation:



- Problem: System.IO.FileNotFoundException: "Could not load file or assembly 'System.IO.Ports, Version=6.0.0.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2ddd51'. Solution: See the step 3 of 9.2.2.2 above.

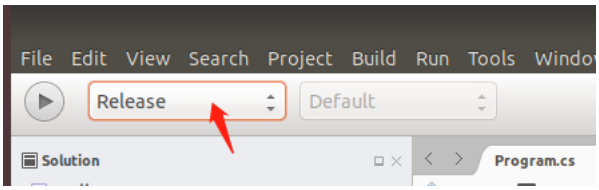
3 Running in the Raspberry Pi robot arm

- Create a C# console application;

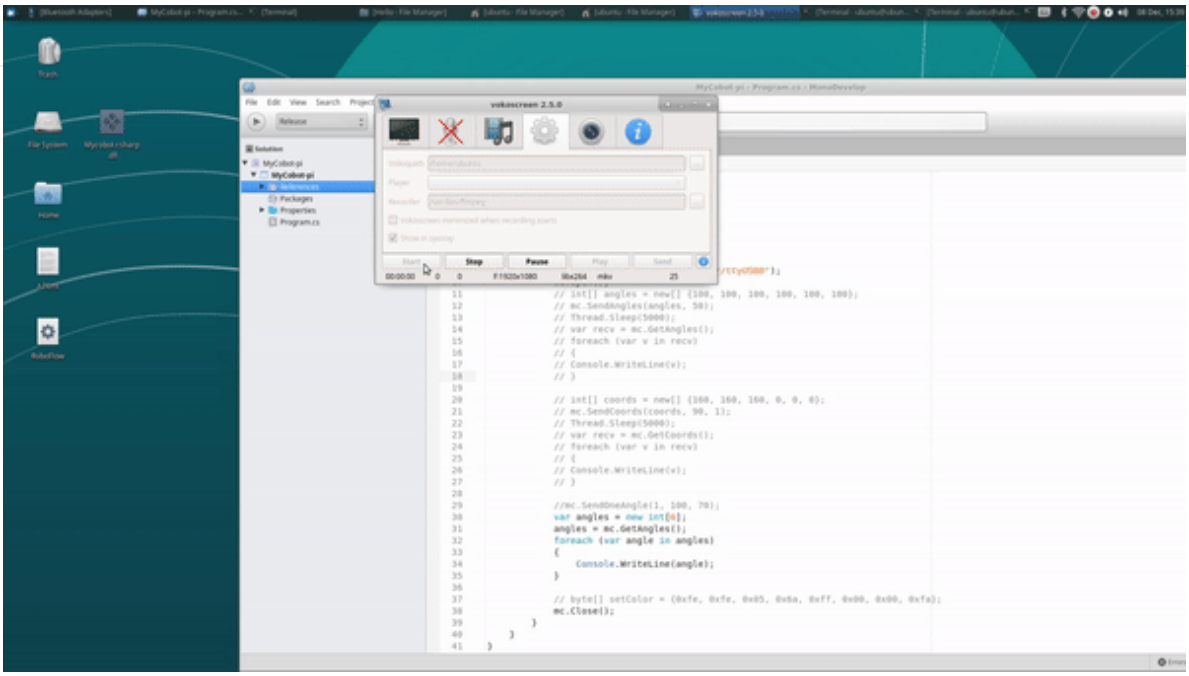
- copy the program.cs and paste it to the application;

- change the port number in program.cs to /dev/ttyAMA0 (MyCobot mc = new MyCobot("/dev/ttyAMA0"));

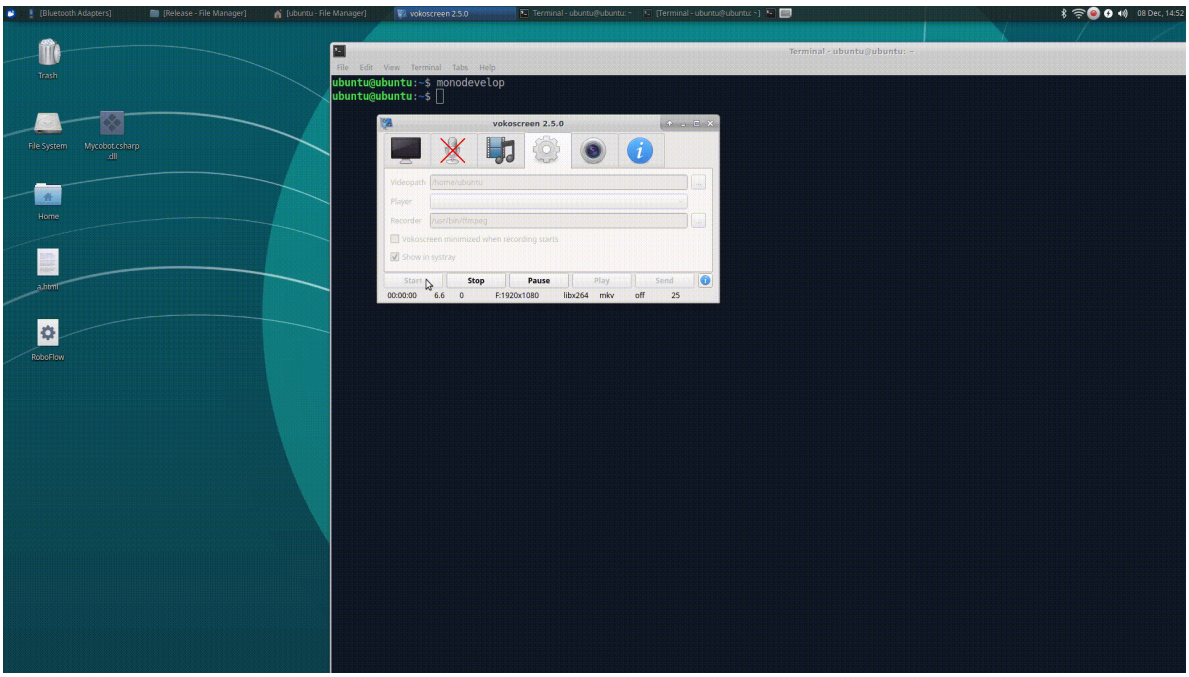
- change the compilation method to Release ;



- Add the Mycobot.csharp.dll library file to the project, library: ReFerences–>Edit References–>.Net Assembly–>Browse(path for .dll)



- Run it. Note: compile and run it. For the whole operation process, watch the the following animation.



Joint control

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

1 Single joint control

1.1 Sending the angle of single joint

SendOneAngle(int jointNo, int angle, int speed)

Return value: none

Parameter description: Parameter 1: joint number (1-6), Parameter 2: angle (ranging from -170° to 170°), and Parameter 3: speed (0-100)

Case:

```
mc.SendOneAngle(1, 100, 70); ## 9.3.2 Multi-joint control
```

2 Multiple joint control

2.1 Get the angles of all joints

GetAngles()

Return value: return an array of int type, int[], length: 6

Description of parameters: none

Case:

```
var recv = mc.GetAngles();
```

2.2 send the angles of all joints

SendAngles(int[] angles, int speed)

Return value: none

Parameter description: Parameter 1: the angles of all joints (ranging from -170° to 170°), and Parameter 2: speed (0-100)

Case:

```
int[] angles = new[] {100, 100, 100, 100, 100, 100};
mc.SendAngles(angles ,30);
int[] angles = new[] {100, 100, 100, 100, 100, 100};
mc.SendAngles(angles,30);
```

3 Complete use cases

The program.cs in the project is a complete use case program, which can be modified as needed on this basis.

```
using System;


namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("/dev/ttyUSB0");
            mc.Open();
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            // Thread.Sleep(5000);
            // var recv = mc.GetAngles();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            mc.SendOneAngle(1, 100,70);

            // byte[] setColor = {0xfe, 0xfe, 0x05, 0x6a, 0xff, 0x00, 0x00, 0xfa};
            mc.Close();
        }
    }
}
```

Coordinate control

Coordinate control is to make the robot arm move to a specified point with a specified posture, which is divided into x, y, z, rx, ry, rz. X, Y and Z represents the position of the robot arm head in space (The coordinate system is Cartesian coordinate system); [rx,ry,rz] represents the posture of such head at this point (The coordinate system is Euler coordinates).

1 Single parameter coordinate

1.1 Sending single parameter coordinates

SendOneCoord(int coord, int value, int speed)

Return value: none

Parameter description: Parameter 1: coordinate number(1-6(x, y, z, rx, ry, rz)); Parameter 2: coordinate (X, Y and Z range from -300 to 300.00mm; RX, RY and RZ range from -180 to 180); Parameter 3: speed (0-100).

Case:

```
mc.SendOneCoord(1, 160, 30);
```

2 Multiple parameter coordinates

2.1 Getting all coordinates

GetCoords()

Return value: return an array of int type, int[], length: 6

Description of parameters: none

Case:

```
var recv = mc.GetCoords();
```

2.2 Sending single parameter coordinates

SendCoords(int[] coords, int speed, int mode)

Return value: none

Parameter description: Parameter 1: all coordinates (X, Y and Z range from -300 to 300.00mm; RX, RY and RZ range from -180 to 180); Parameter 2: speed (0-100); Parameter 3: mode (0 - angular, and1 - linear)

Case:

```
int[] coords = new[] {160, 160, 160, 0, 0, 0};
mc.SendCoords(coords ,30);
int[] coords = new[] {160, 160, 160, 0, 0, 0};
mc.SendCoords(coords,30);
```

3 Complete use cases

The program.cs in the project is a complete use case program, which can be modified as needed on this basis.

using System;

```
namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("/dev/ttyUSB0");
            mc.Open();
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            // Thread.Sleep(5000);
            // var recv = mc.GetAngles();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
            // foreach (var v in recv)
            // {
                // Console.WriteLine(v);
            // }

            mc.SendOneAngle(1, 100,70);

            // byte[] setColor = {0xfe, 0xfe, 0x05, 0x6a, 0xff, 0x00, 0x00, 0xfa};
            mc.Close();
        }
    }
}
```

io control

There are pins on the M5Stack-basic at the bottom of the robot arm and on the Atom at the end. The adsorption pump and other tools can be controlled by setting high and low levels of the pins through io control (For the numbers of the pins, see the pin labels attached at M5Stack-basic and Atom pins. The pins are common for input and output).

1 M5Stack-basic io control (m5)

1.1 Setting the high and low levels of output io

SetBasicOut(byte pin_number, byte pin_signal)

Return value: none

Parameter description: Parameter 1: pin number (basic output pin number); Parameter 2: state (0 - low level; 1- high level)

Case:

```
mc.SetBasicOut(2, 1);
Thread.Sleep(100);
mc.SetBasicOut(5, 1);
Thread.Sleep(100);
```

1.2 Getting the state of input io

GetBasicIn(byte pin_number)

Return value: pin state (0-low level; 1-high level)

Parameter description: pin number (basic input pin number)

Case: set the output pin 2 to high level

```
Console.WriteLine(mc.GetBasicIn(35));
Thread.Sleep(100);
Console.WriteLine(mc.GetBasicIn(36));
Thread.Sleep(100);
```

2 Atom io Control

Note: 320m5 has no atom io, so the module API is not used

2.1 Setitng the high and low levels of output io

SetDigitalOut(byte pin_number, byte pin_signal)

Return value: none

Parameter description: Parameter 1: pin number (atom output pin number); Parameter 2: state (0 - low level; 1- high level)

Case:

```
mc.SetDigitalOut(23, 0);
Thread.Sleep(100);
mc.SetDigitalOut(33, 0);
Thread.Sleep(100);
```

2.2 Getting the state of input io

GetDigitalIn(byte pin_number)

Return value: pin state (0-low level; 1-high level)

Parameter description: pin number (atom input pin number)

Case:

```
Console.WriteLine(mc.GetDigitalIn(19));
Thread.Sleep(100);
Console.WriteLine(mc.GetDigitalIn(22));
Thread.Sleep(100);
```

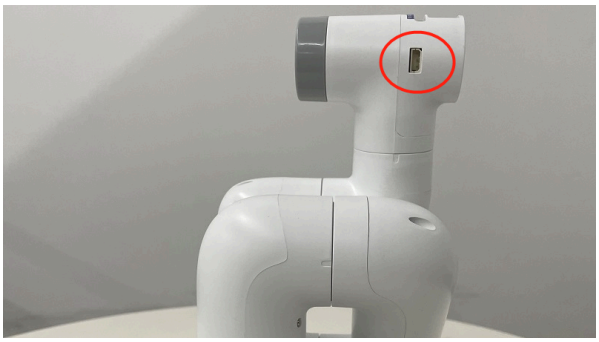
3 Complete use cases

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");
            mc.Open();
            Thread.Sleep(5000);

            //set basic output io
            /*mc.SetBasicOut(2, 1);
            Thread.Sleep(100);
            mc.SetBasicOut(5, 1);
            Thread.Sleep(100);
            mc.SetBasicOut(26, 1);
            Thread.Sleep(100);*/

            //get basic input io
            Console.WriteLine(mc.GetBasicIn(35));
            Thread.Sleep(100);
            Console.WriteLine(mc.GetBasicIn(36));
            Thread.Sleep(100);

            //set atom output io
            /*mc.SetDigitalOut(23, 0);
            Thread.Sleep(100);
            mc.SetDigitalOut(33, 0);
            Thread.Sleep(100);*/

            //get m5 input io
            /*Console.WriteLine(mc.GetDigitalIn(19));
            Thread.Sleep(100);
            Console.WriteLine(mc.GetDigitalIn(22));
            Thread.Sleep(100);*/

            mc.Close();
        }
    }
}
```

Gripper control

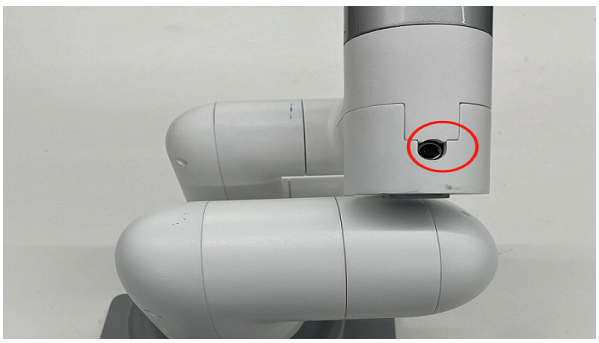
Installing the gripper:

- For an adaptive gripper, insert it on the pin on the atom, as shown in the following figure:



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

Notice: myCobot280 and myPalletizer 260 have no electric gripper, only myCobot320 has an electric gripper.



1 Adaptive gripper control

supports：**myCobot280, 320&&myPalletizer 260**

1.1 **setGripperValue(byte angle, byte speed)**

Return value: none

Parameter description: Parameter 1: gripper opening and closing angles (ranging from 0 to 100; 0–closed; 100-maximum open angle); Parameter 2: gripper opening and closing speeds (0-100)

Case:

```
mc.setGripperValue(0, 10);
Thread.Sleep(3000);
mc.setGripperValue(50, 100);
Thread.Sleep(3000);
```

1.2 **getGripperValue()**

Return value: int type, returning the gripper angle (0 - closed; 100 - maximum open angle)

Parameter description: none

Case:

```
Console.WriteLine(mc.getGripperValue());
```

2 Electric gripper control

Available for: **myCobot320**

2.1 **setEletricGripper(int state)**

Return value: none

Parameter description: gripper switch state (0-off; 1-on)

Case:

```
mc.setEletricGripper(0);
```

3 Complete use cases

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");
            mc.Open();
            Thread.Sleep(5000);

            //set gripper open or close 0--close 100-open max 0-100
            mc.setGripperValue(0, 10);
            Thread.Sleep(3000);
            mc.setGripperValue(50, 100);
            Thread.Sleep(3000);

            //set electric gripper
            mc.setEletricGripper(0);
            Thread.Sleep(100);
            mc.setEletricGripper(1);
            Thread.Sleep(100);

            //get gripper state 0--close 1--open
            Console.WriteLine(mc.getGripperValue());
            mc.Close();
        }
    }
}
```

myCobot API

Before using the following function interfaces, import our API library first. Otherwise it is impossible to run successfully. To download and import the library, refer to 9.2 Compiling and Running of a MycobotCpp Case.

1 Preconditions for controlling the robot arm

1.1 MyCobot(string port, int baud=115200)

Function: to instantiate MyCobot

Return value: none

Parameter description: Parameter 1: serial port number ("COM*" on Windows (such as COM30) Parameter 2: baud rate (default 115200)

Note: If you want to call the following API, you need to instantiate it first

1.2 Open()

Function: to open the serial port

Return value: none

Parameter description: none

Note: for communication with the robot arm, the serial port should be opened first

1.3 Close()

Function: to close the serial port

Return value: none

Parameter description: none

Note: When the program ends, it is better to close the serial port

2 Overall running status of Robot

2.1 PowerOn()

Function: to power up the robot arm

Return value: none

Parameter description: none

Note: After the robot arm is powered up, it cannot be moved manually

2.2 PowerOff()

Function: to cut off teh power for the robot arm

Return value: none

parameter description: none

Note: After the robot arm is powered up, if you want to move it manually, you may use this api to cut off the power of the robot arm

3 MDI program control mode

3.1 SendOneAngle(int jointNo, int angle, int speed)

Function: to send single joint angle

Return value: none

Parameter description: Parameter 1: joint number (1 - 6); Parameter 2: angle (ranging from -170° to 170°); Parameter 3: speed (0-100)

3.2 GetAngles()

Function: to get the angles of all joints

Return value: return an array of int type, int[], length: 6

Description of parameters: none

3.3 SendAngles(int[] angles, int speed)

Function: to send the angles of all joints

Return value: none

Parameter description: Parameter 1: the angles of all joints (ranging from -170° to 170°); Parameter 2: speed (0-100)

3.4 GetCoords()

Function: to get all coordinates

Return value: return an array of int type, int[], length: 6

Description of parameters: none

3.5 SendCoords(int[] coords, int speed, int mode)

Function: to send multi-parameter coordinates

Return value: none

Parameter description: Parameter 1: All coordinates (The values of X, Y and Z range from -300 to 300.00mm; the values of RX, RY and RZ range from -180 to 180); Parameter 2: speed (0-100); Parameter 3: mode (0 - angular, and 1 - linear)

3.6 SendOneCoord(int coord, int value, int speed)

Function: to send single parameter coordinates

Return value: none

Parameter description: Parameter 1: coordinate number (1-6(x, y, z, rx, ry, rz)); Parameter 2: coordinate (The values of X, Y and Z range from -300 to 300.00mm; the values of RX, RY and RZ range from -180 to 180); Parameter 3: speed (0-100)

4 Atom end IO control

4.1 SetDigitalOut(byte pin_number, byte pin_signal)

Function: to set the high and low levels of output io

Return value: none

Parameter description: Parameter 1: pin number (atom output pin number), and Parameter 2: state (0-low level, and 1-high level)

4.2 GetDigitalIn(byte pin_number)

Function: to get the state of input io

Return value: pin state (0-low level, and 1-high level)

Parameter description: pin number (atom input pin number)

4.3 setGripperValue(byte angle, byte speed)

Function: to control the adaptive gripper

Return value: none

Parameter description: Parameter 1: gripper opening and closing angles (ranging from 0 to 100; 0–closed; 100-maximum open angle); Parameter 2: gripper opening and closing speeds (0-100)

4.4 SetElectricGriper(byte open)

Function: to control the electric gripper

Return value: none

Parameter description: gripper switch state (0-off, and 1-on)

4.5 getGripperValue()

Function: to get the angle of the adaptive gripper

Return value: int type, return gripper angle (0 - closed; 100 - maximum open angle)

Parameter description: none

5 Stand M5Stack-basic IO control

5.1 SetBasicOut(byte pin_number, byte pin_signal)

Function: to set the high and low levels of output io

Return value: none

Parameter description: Parameter 1: pin number (atom output pin number), and Parameter 2: state (0-low level, and 1-high level)

5.2 GetBasicIn(byte pin_number)

Function: to get state of input io

Return value: pin state (0-low level, and 1-high level)

Parameter description: pin number (basic input pin number)

Use cases

In this case, first get the current angles of all joints, then make the joint 1 returns to zero point, and finally get the values of two input pins on M5Stack-basic. The program.cs in the project is a complete use case program, which can be modified as needed on this basis:

```
using System;
using System.Threading;

namespace Mycobot.csharp
{
    class Test
    {
        static void Main(string[] args)
        {
            MyCobot mc = new MyCobot("COM57");
            mc.Open();
            Thread.Sleep(5000);
            // int[] angles = new[] {100, 100, 100, 100, 100, 100};
            // mc.SendAngles(angles, 50);
            var recv = mc.GetAngles();
             foreach (var v in recv)
             {
                Console.WriteLine(v);
             }

            // int[] coords = new[] {160, 160, 160, 0, 0, 0};
            // mc.SendCoords(coords, 90, 1);
            // Thread.Sleep(5000);
            // var recv = mc.GetCoords();
            // foreach (var v in recv)
            // {
            // Console.WriteLine(v);
            // }

            mc.SendOneAngle(1,0, 70);
            Thread.Sleep(100);
            /*var angle = new int[6];
            angle = mc.GetAngles();
            foreach (var v in angle)
                Console.WriteLine(v);
            // byte[] setColor = {0xfe, 0xfe, 0x05, 0x6a, 0xff, 0x00, 0x00, 0xfa};*/

            //set basic output io
            /*mc.SetBasicOut(2, 1);
            Thread.Sleep(100);
            mc.SetBasicOut(5, 1);
            Thread.Sleep(100);
            mc.SetBasicOut(26, 1);
            Thread.Sleep(100);*/

            //get basic input io
            Console.WriteLine(mc.GetBasicIn(35));
            Thread.Sleep(100);
            Console.WriteLine(mc.GetBasicIn(36));
            Thread.Sleep(100);
```

```
            //set atom output io
            /*mc.SetDigitalOut(23, 0);
            Thread.Sleep(100);
            mc.SetDigitalOut(33, 0);
            Thread.Sleep(100);*/

            //get m5 input io
            /*Console.WriteLine(mc.GetDigitalIn(19));
            Thread.Sleep(100);
            Console.WriteLine(mc.GetDigitalIn(22));
            Thread.Sleep(100);*/

            //set gripper open or close 0--close 100-open max 0-100
            /*mc.setGripperValue(0, 10);
            Thread.Sleep(3000);
            mc.setGripperValue(50, 100);
            Thread.Sleep(3000);*/

            //get gripper state 0--close 1--open
            /*Console.WriteLine(mc.getGripperValue());*/
            mc.Close();
        }
    }
}
```

Arduino

What is Arduino?

Arduino is an easy-to-use open source electronic prototype platform that includes hardware (various Arduino-compliant development boards) and software (Arduino IDE and related development pakages).

The hardware (or called development board) consists of a microcontroller (MCU), flash memory (Flash), and a set of general-purpose input/output interfaces (GPIO), etc. You can understand it as a microcomputer motherboard.

The software is mainly composed of Arduino IDE on the PC side, related board support packages (BSP) and rich third-party function libraries. Using Arduino IDE, users may easily download the BSP related to the development board you have and the required function library for writing your program.

What can the MyCobotBasic library do?

MyCobotBasic library is an open source robot control library developed by our company, which can be used only after using the robot developed by our company. Using this library, you can control our robot through Bluetooth, WiFi, serial port, etc. It also supports functions such as external sensors, IIC communication, and LED lights. You can DIY different application scenarios according to your own needs, or you can refer to the MiniRobot sample code or control cases such as angles, coordinates, and grippers we provide. The MiniRobot sample code includes Bluetooth, WiFi, drag teaching, distance sensor and other control-related content.

Applicable equipment:

- myCobot 280
  - myCobot 280 M5
  - myCobot 280 for Arduino
- myCobot 320
  - myCobot 320 M5
- myPalletizer 260
  - myPalletizer 260 M5
- mechArm 270
  - mechArm 270 M5

Preconditions for use

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

Arduino Environment Building

1 Arduino IDE download

Arduino IDE Download Address

- Arduino Official Website
- Windows X64
- Mac OS X
- Linux ARM 64

2 Installed drivers

Before burning the program, M5Core host (including M5Stack-basic/GRAY/M5GO/FIRE/FACES) / **micro control types of equipment** users according to the operating system you are using, please download the corresponding driver package. After decompressing the package, select the installation package of the corresponding OPERATING system bit.

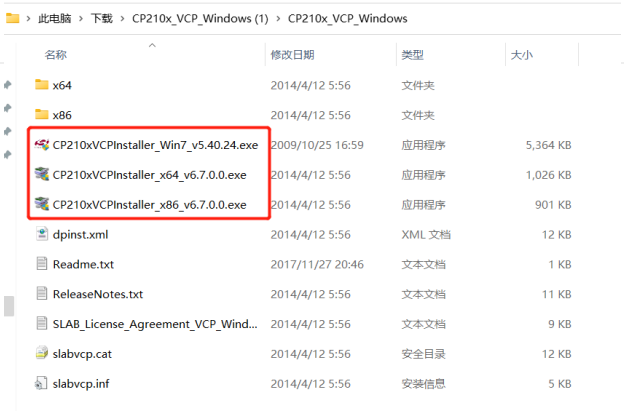
For Mac OS, ensure correct settings of the system **Preferred settings --> Security and privacy --> General** before installation, and allow the user to get it from App Store or an approved developer.

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

CP210X

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

After extracting the zip package, select the corresponding installation package to install according to your computer **operating system** (win10 and win11 select x64 or x86 for installation).

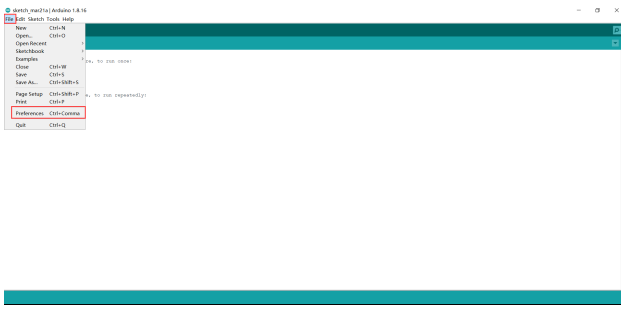


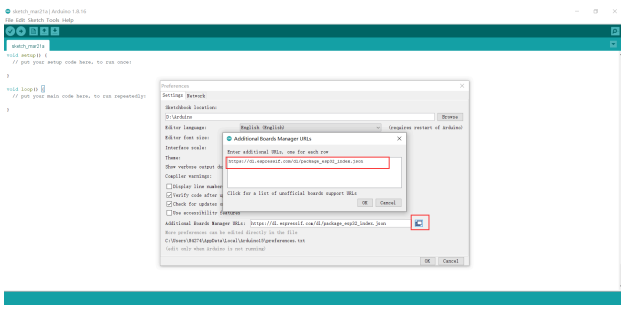
CP34X

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

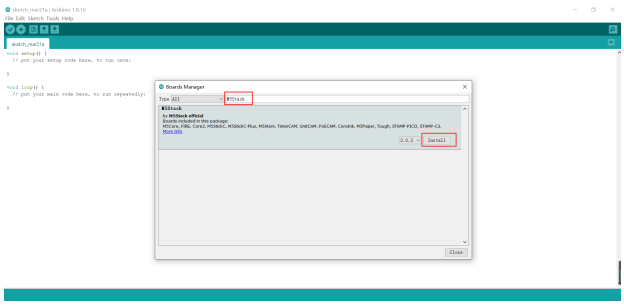
3 Add board

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

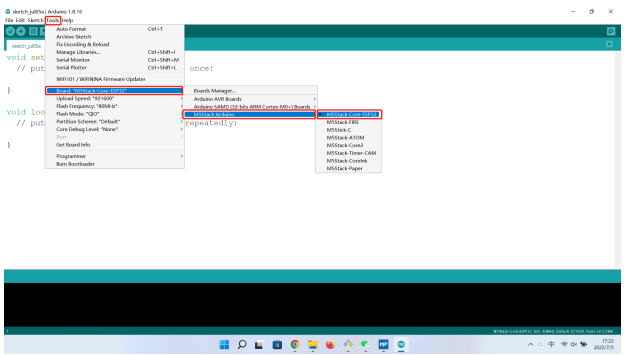




- After adding, select the **Tools --> Board --> Boards Manager**, in the new pop-up dialog, input and search **M5Stack** , click Install (in case of search failure, you can try to restart **Arduino** program), as shown below:



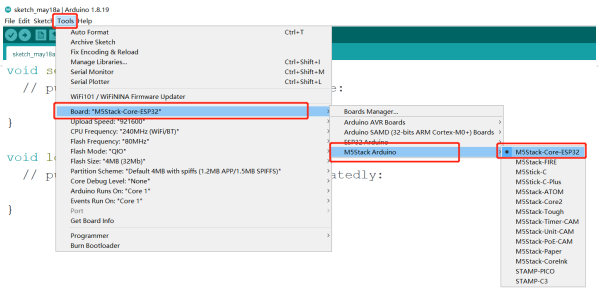
- After adding, select **Tools --> Board**, check whether it is successful, as shown below:



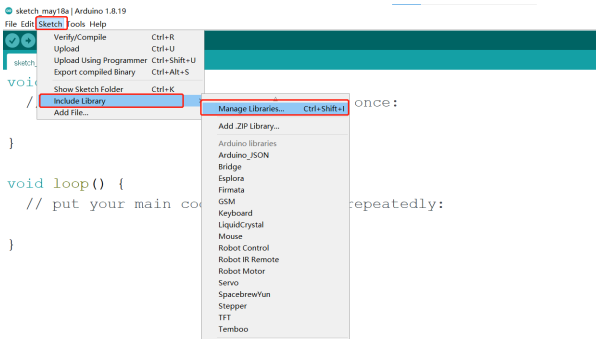
4 Add related libraries

4.1 Install the M5Stack library

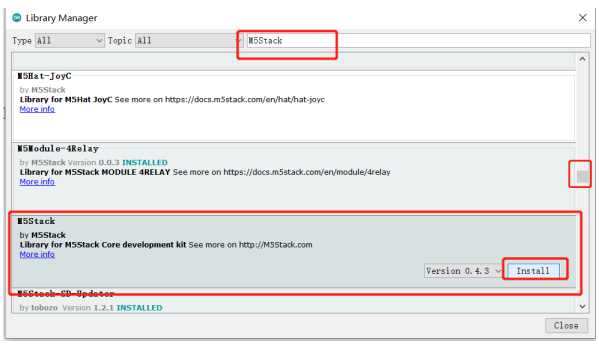
1. Tools --> Development Board --> M5Stack Arduino select **M5Stack-Core-ESP32**, as shown in the following figure：



2. Project --> Load Library --> Manage Library In the search box, enter **M5Stack**, as shown in the figure below：



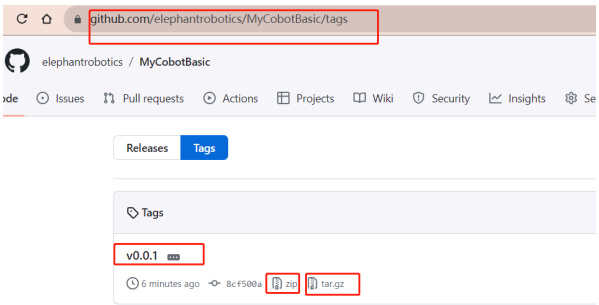
3. After finding it, click Install, scroll down, **M5Stack** is at the back, you can see the location of the drop-down slider in the picture for details, as shown in the figure below：



4.2 Install the MyCobotBasic library

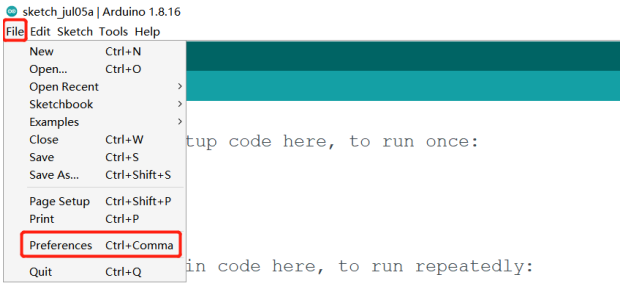
Note: Please download the latest library, the first version is v0.0.1.

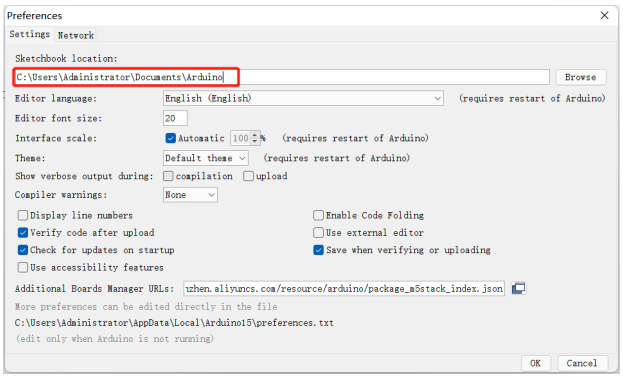
- **MycobotBasic**(After the Mycobot280-Arduino model is imported, you can refer to 10.3-arduinolib_use for use). Please see the figure below for details, .zip is suitable for Windows systems, and .tar.gz is suitable for Linux systems:

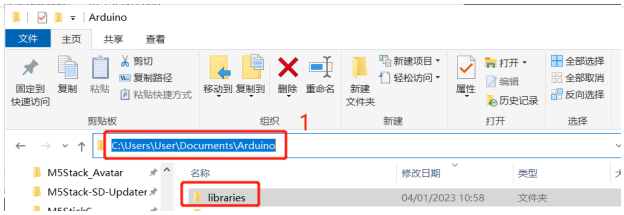


- Library Installation Instructions

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

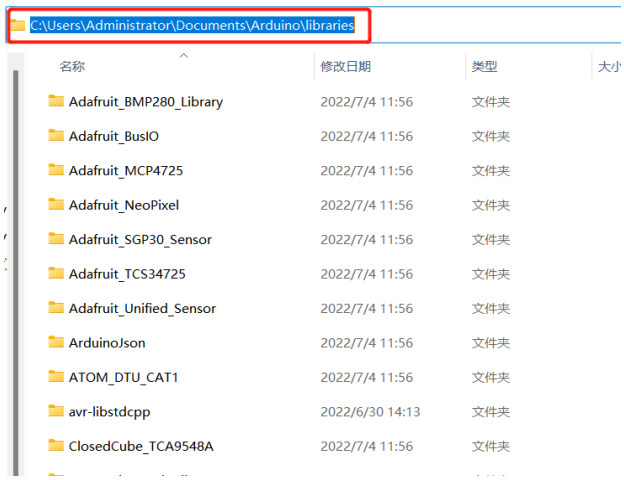






1 Copy the path here and press enter to find the libraries folder

Unzip it into the corresponding folder in the **libraries** directory. If you are using **Arduino**, don't overwrite it, just add to the existing **Library**.



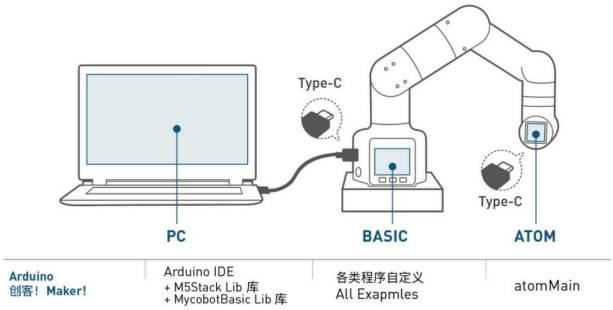
Congratulations, you have built a **Arduino** related development environment.

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

Simple use of Arduino

1 Connecting a device

Take **myCobot 280-M5** for example. Connect the M5Stack-basic on the stand of the robot arm with a PC by using a Tyep-C data cable.



2 Firmware requirements

• ATOM: burn the latest version of AtomMain by using **MyStudio**.

• M5Stack-basic: no requirements

3 Detecting link

Open the computer device manager to see whether there is a device. If no device is detected, replace the USB cable. If "it cannot be used" is shown, click to download **CP210X driver or CP34X**. After the downloading is complete, unzip and install the required version of the driver for use.

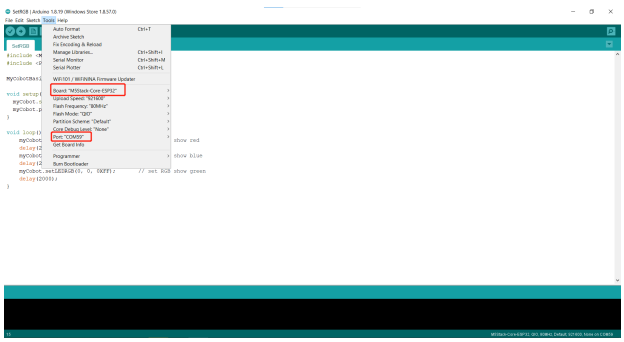
Open **Arduino IDE --> Tools --> Port** to check whether there is a device. If no device is detected, replace the USB cable for test, or check whether the driver is installed successfully.

4 Starting development

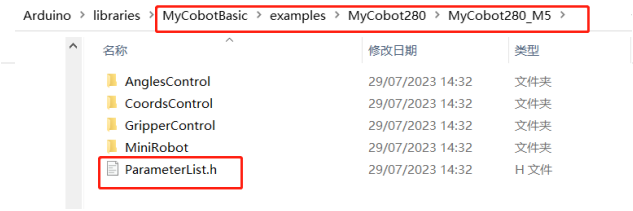
Take burning an official demo as an example. Open **Arduino IDE --> File --> Examples --> MyCobotBasic**, and you will see all project examples. Choose to burn a simple demo, for example **MyCobotBasic --> MyCobot280--> MyCobot280-M5--> AnglesControl**.

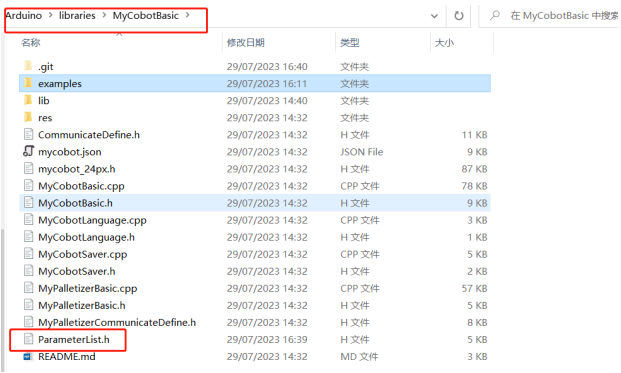
Open AnglesControl.ino from the example file.

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



If you are using the myCobot280-M5, **Please use the ParameterList.h under the MyCobot280-M5 folder to replace the ParameterList.h under the MyCobotBasic folder**. Please refer to the figure below for details:

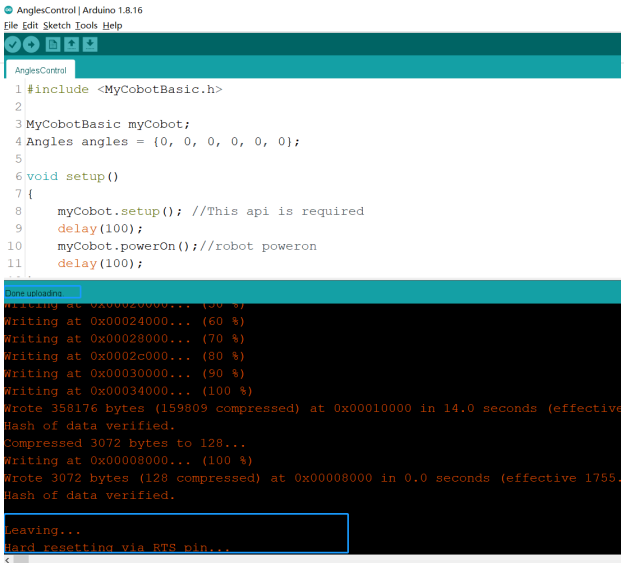




Note: When using different models, please use the "ParameterList.h" file in the respective case directory to replace the "MyCobotBasic\ParameterList.h" file Click Download and wait for the progress bar at the bottom right to finish.

Click upload and wait for the progress bar at the bottom right to complete

Wait until the bottom right shows that the uploading is successful, which means that the downloading of the program has been completed.



At this point we can see the **robot** start working.

The interface and drivers for the use of the **basic buttons with the screen** can be found in the following documents:

button: https://docs.m5stack.com/en/api/core/button

screen: https://docs.m5stack.com/en/api/core/lcd

5 Part of the case introduction

5.1 At present, different models have angle, coordinate, and gripper control. MyCobot320 supports adaptive gripper and electric gripper control.

5.2 miniRobot:

MyCobot280, 320m5, and mechArm270-M5 use cases can perform zero calibration, drag teaching, communication, etc. (on this basis, use RoboFlow, python, myblockly, etc. to control the robotic arm), information acquisition (obtain the connection status of the servo atom , and basic, atom firmware version).

5.3 Transponder:

myCobot280-Arduino uno and Mega2560 development board use case, the function is mainly communication, on this basis, use RoboFlow, python, myblockly, etc. to control the robotic arm and perform io control.

5.4 MyPalletizerRoboFlow:

MyPalletizer260 use case, can perform zero calibration, drag teaching, communication (on this basis, use RoboFlow, python, myblockly, etc. to control the robotic arm), information acquisition (obtain the connection status of the servo atom, and the firmware version of basic and atom).

Note: Arduino environment configuration and case compilation can be found in our gitbook documentation（ https://docs.elephantrobotics.com/docs/gitbook-en/10-ArduinoEnv/ ） and the videos on Bilibili （ https://www.bilibili.com/video/BV1Vi4y1c7DQ/ ）。

10.3 Use of the Arduino library

Support robotic arm types: **myCobot280-Arduino**

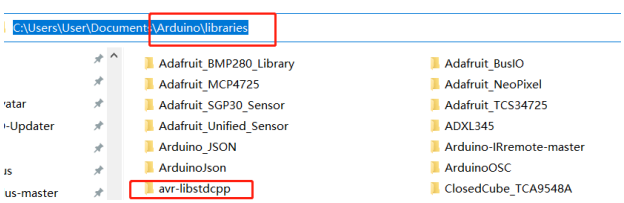
Use Cases: eg:Open **C:\Users\User\Documents\Arduino\libraries\MyCobotBasic\examples\MyCobot280\MyCobot280_Arduino\Mega\AnglesControl\AnglesControl.ino**, This case requires the board to be burned first and then connected to the robot arm, otherwise the upload will fail.

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

10.3.1 Modifications before compilation

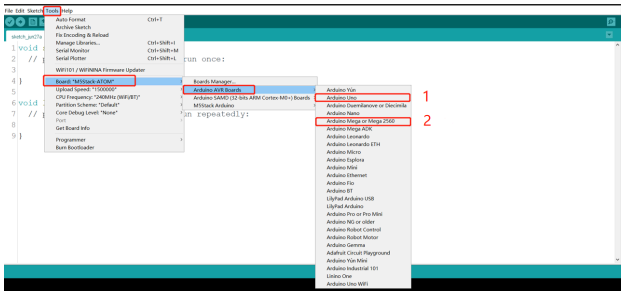
Import the library file, if your robot arm is myCobot280-Arduino, please check if the board is Mega2560 or Uno, if it is:

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



10.3.2 Selection of boards before compilation

1.Board for Uno、Mega2560, Tools --> Board --> Arduino AVR Boards --> Arduino Uno （or Arduino MEAG or Mega2560), see the following chart for details:

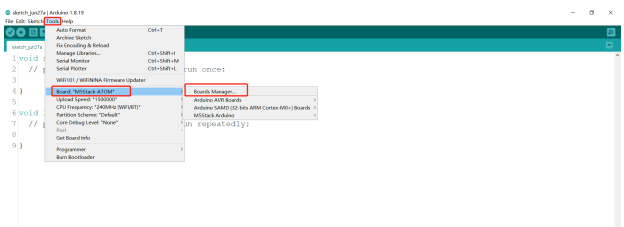


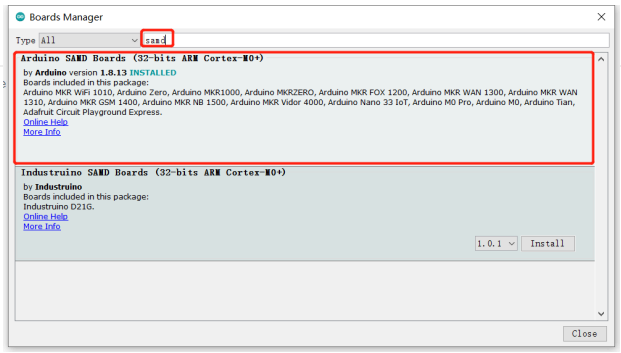
1 When using uno,select

2 When using Mega2560, select

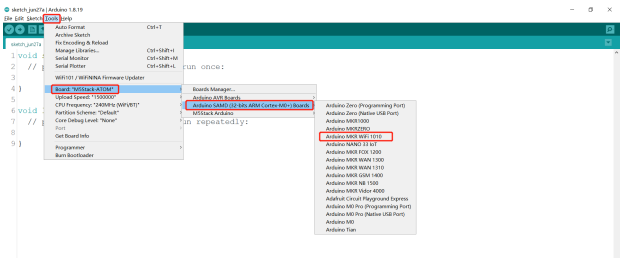
2.Board for mkr wifi1010

Boards manager search samd, if not installed, install, first Tools --> Board --> Boards Manager, then search samd, see the following figure:





Board for mkr wifi1010, Tools --> Board --> Arduino SAMD --> Arduino MKR WiFi1010



Arduino API

1. Overall Status

`powerOn();`

- Function: The robot is powered on, and the robot can be controlled after the power is turned on（open by default）
- Return Value: None

`powerOff();`

- Function: Robot power off
- Return Value: None

`isPoweredOn();`

- Function: Atom status inquiry, return Atom connect status
- Return Value: Power on is TRUE, power off is FALSE

`getAtomVersion();`

- Function: Get Atom firmware version
- Return Value: Values of type int,the data needs to be /10. For example, if the version number read is 12, it needs to be divided by 10, and the final version number is 1.2

`setFreeMoveMode(bool mode);`

- Function: Set the free movement mode. After the free movement mode is turned on, the LED at the end will be yellow. Press and hold the atom at the end to move the robot manually
- Parameter Description:
- mode: mode, 0/1, 0--off free movement, 1--open free movement
- Return Value: None

2. MDI Mode and Robot Control (Manual Data Input)

`getAngles();`

- Function: Read all joint angles, when used one Angles should be defined to receive data that was read. Angles are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 angles to store Angle variables, it is used in the same way as arrays.
- Return Value: Arrays of type Angles

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

- Function: Send a single joint Angle
- Parameter Specification：

Joint Number= joint, range from 1-6; Specified Angle Value= value, range approximately from -170°- + 170°; Specified Speed= speed, range from 1-100

- Return Value: None

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

- Function: Synchronize joint angles, send joint angles at the same time. Specified Angles is a container with a capacity of 6 data, can be viewed as an array. Use a for loop to assign values, or assign values separately.
- Angles[0] = Specified Angle, Angles[2] = Specify Angle, range from 0-90 (the value range should be the same as writeAngle) unit°

Movement Speed = speed, range from 0–100 unit %

- Return Value: None

```
getCoords();
```

- Function: Read x,y,z,rx,ry,rz of the end of myCobot, a Coords tempcoords should be defined when used to received angles that was read. Coords are defined in terms of variables or functions built into library functions. We can define a memory space that is 6 tempcoords to store Angle variables, it is used in the same way as arrays.
- Return Value: An array of type Coords. You need to define variables of type Coords.

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

- Function: Send the specific value of the individual coordinate parameters x/y/z, the ends are going to move in a single direction.
- Parameter Specification:

Value of Moving Path Coordinate = value range from -300–300 (The position coordinates of axis=Axis::X, aixs=Axis::Y and axis=Axis::Z are respectively X,Y,Z, the units would be mm. Position coordinate value range is not uniform, axis=Axis::RX, aixs=Axis::RY and axis=Axis::RZ are respectively RX,RY,RZ ranging from-180°-180°, if the value is beyond the range it will return the clue “inverse kinematics no solution”)

Specified Speed = speed range from 1-100 unit %

- Return Value: None

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

- Function: To send the specified coordinate parameter, which should be of type Coords, declare a variable of type Coords, which is used in the same way as an array
- Parameter Specification

coords[0] = X, coords[1] = Y, coords[2] = Z,

X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue ”inverse kinematics no solution” will be given) unit mm

RX,RY,RZ range from -180-180

Specified Speed = speed, range from 1-100 unit %

- Return Value: None

`checkRunning();`

- Function: Check whether the equipment is in motion
- Return Value: In motion is TRUE，on the contrary it's FALSE

`setEncoder(int joint, int encoder);`

- Function: Set a single joint to rotate to a specified potential value
- Parameter description: joint number = joint value range 1-7 (joint number 7 is generally a gripper, gripper potential value range: 1325-2048); steering gear potential value = encoder value range 0-4096 (the range should be positively related to the range of each joint)
- Return Value: None

`getEncoder(int joint);`

- Function: Get the specified joint potential value
- Parameter Specification: Servo Motor Number = joint range from 1-7
- Return Value: Int type, range from 0-4096

`setEncoders(Angles angleEncoders, int speed);`

- Function: Set the six joints to run synchronously to the specified position
- Parameter Specification: Need to define a variable of type Angles: angleEncoders, it is used in the same way as arrays. Assign a value to the array angleEncoders, values range from 0-4096 (The range should be positively related to the range of each joint) , the length range of the array is 6. Specified Speed = speed, range from 1-100 unit %
- Return Value: None

`getEncoders();`

- Function: Get all joint potential values
- Return Value: Arrays of type Angles, range from 0-4096

`getServoSpeeds();`

- Function: Get all servo speeds

- Return Value: Arrays of type Angles, Velocity is 0 when not in motion

3. JOG Mode

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

- Function: Control the movement of a single joint in one direction
- Parameter Specification:

  Joint/Servo Motor Number = joint, range from 1-6;

  Direction of Joint Motion = Direction, range from -1/1;

  Specified Speed = speed, range from 1-100 unit %
- Return Value: None

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

- Function: Control myCobot moves in one direction in Cartesian space
- Parameter Specification:

  Direction Selection = axis, range from X,Y,Z,RX,RY,RZ;

  Direction of Joint Motion = Direction, range from -1/1;

  Specified Speed = speed, range from 1-100 unit %
- Return Value: None

`jogStop();`

- Function: Stops the specified direction of motion that has started
- Return Value: None

`ProgramPause();`

- Function: Program pause
- Return Value: None

`ProgramResume();`

- Function: Program continues to run
- Return Value: None

`TaskStop();`

- Function: Program stop
- Return Value: None

4. Running Status and Settings

`getSpeed();`

- Function: Read the current running speed
- Return Value: Int tape, range from 1-100, unit %

`setSpeed(int percentage);`

- Function: Set the running speed
- Parameter Specification: Percentage, range from 1-100, unit %

`getJointMin(int joint);`

- Function: Read the joint minimal limit Angle
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: Float type of the array

`getJointMax(int joint);`

- Function: Read the joint maximal limit Angle
- Parameter Specification: Joint Number = joint, range from 1-6
- Return Value: Float type of the array

`setMovementType(MovementType movement_type);`

- Function: Set the movement method
- Parameter Specification: Movements are non-linear path movement (movej) and linear path movement (movel)
- Return Value: None

`getMovementType();`

- Function: Read movement method
- Return Value: The non-linear method returns 0; the linear method returns 1

5. Joint Servo Control

`isServoEnabled(int joint);`

- Function: Check the single joint connection status
- Parameter Specification: Joint Number = joint, range from 1-6

- Return Value: Connection status, 0/1, 1--connected, 0--not connected

```
isAllServoEnabled();
```

- Function: Check whether all joins are properly connected
- Return Value: Connection status, 0/1, 1--connected, 0--not connected

```
getServoData(int joint, byte data_id);
```

- Function: Read the system parameters of joint
- Parameter Specification:

  Joint servo serial number = joint, range from 1-6;

  Data Address = data_id, refer to the following Figure for address

- Return Value: Refer to the following Figure for the value range

Address	Function	Range of values	Initial value	Explanation of values
20	LED alarm conditions	0-254	0	Corresponding bit setting 1 to turn on the flashing light alarm Corresponding bit setting 0 to turn off the flashing light alarm
21	Position loop P scale factor	0-254	123 Joints take value 8, 456 take value 5.	Proportionality factor of control motor
22	Position loop D differentiation factor	0-254	123 Joints take value 20, 456 take value 13.	Differential coefficient of control motor
23	Position loop I integration factor	0-254	0	Integral coefficient of the control motor
24	Minimum start-up force	0-1000	0	Set the minimum output starting torque of the servo, set 1000 = 100% * Plugging torque

```
setServoData(int joint, byte data_id, byte data);
```

- Function: Set the system parameters of joint
- Parameter Specification:

  Joint servo serial number = joint, range from 1-6;

  Data Address = data_id, refer to the address range in the figure above;

data = the range of values in the above diagram

- Return Value: None

`setServoCalibration(int joint);`

- Function:Joint Zero Calibration, the corresponding potential value is 2048
- Parameter Specification: joint number = joint, range from 1-6

`releaseServo(byte servo_no);`

- Function: Relax/disable a certain joint of the robot
- Parameter Specification: servo_no is 1-6
- Return Value: None

`focusServo(byte servo_no);`

- Function: Enable a certain joint of the robot
- Parameter Specification: servo_no is 1-6
- Return Value: None

`getServoVoltages();`

- Function: Get all servo voltages
- Return Value: Array of type Angles, reference range from 8.4-12.0

`getServoStatus();`

- Function: Get the status of all servos
- Return Value: 0 means all states are normal; 1 means voltage overvoltage/undervoltage; 2 means magnetic code state abnormal; 4 means temperature overheating; 8 means current overcurrent; 32 means load overload; when the number that appears is not equal to the above abnormal numbers, for example: 3 means voltage overvoltage/undervoltage and magnetic code state abnormal, 7 means voltage overvoltage/undervoltage, magnetic code state abnormal and temperature overheating

`getServoTemps();`

- Function: Get all servo temperatures
- Return Value: Array of type Angles, reference range from 0-255

6. Atom IO Control

`setPinMode(byte pin_no, byte pin_mode);`

- Function: Sets the state mode of the specified pin of atom
- Parameter Specification:

Pin Serial Number = `pin_no`, reference range from: 19、22、23、26、32、33

Output Mode = pin_mode, reference range from: 0、1

- Return Value: None

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

- Function: Set the color of the RGB lights of atom
- Parameter Specification:

Parameter of Red Light = r, range from 0x00 – 0xFF;

Parameter of Green Light = g, range from 0x00 – 0xFF;

Parameter of Blue Light = b, range from 0x00 – 0xFF;

- Return Value: None

`setGripperState(byte mode, int sp);`

- Function: Set the opening and closing state of the Gripper
- Parameter Description:
- mode, jaw opening and closing mode, range 0/1, 0--the jaws are opened to the maximum, 1--the jaws are closed to the minimum
- sp, clamping jaw opening and closing speed, range 1-100

`setGripperValue(int data, int sp);`

- Function: Set the Gripper opening and closing angle
- Parameter Description:
- data, clamping jaw opening and closing angle, range 0-100, 0--closed to the minimum angle, 100--open to the maximum angle
- sp, clamping jaw opening and closing speed, range 1-100
- Return Value: None

`setGripperIni();`

- Function: Set jaw zero point
- Return Value: None

`getGripperValue();`

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

`isGripperMoving();`

- Function: Detects if the jaws are in motion
- Return Value: 0 not in motion, 1 in motion

```
void setEletricGripper(bool mode);
```

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

```
void InitEletricGripper();
```

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

```
void setGripperMode(bool mode);
```

- Note: This interface is only available for MyCobot320 robot
- Function: Set Adaptive Gripper Control Mode
- Parameter Description:
- mode: mode, 0/1, 0--485 communication control, 1--io control (in io mode, it can only be turned on or off, and the angle cannot be set. Pins 23 and 33, when turned on or off, both pins The feet need to be set to different states, one must be high and one low)
- Return value: None

```
bool getGripperMode();
```

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

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

- Function: Setting the operating state of IO pins
- Parameter Specification: 0 input; 1 output; 2 `pull_up_input`
- Return Value: None

```
getDitialInput(byte pin_no);
```

- Function: Read input
- Parameter Specification: Pin Serial Number = pin_no Range of values: 19、22、23、26、32、33
- Return Value: None

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

- Function: Set ATOM end IO to output PWM signal with specified duty cycle
- Parameter Specification:
- pin_no: IO Serial Number
- freq: Clock Frequency
- pin_write: Duty Cycle 0-256; 128 means 50%
- Return Value: None

7. Coordinate Control Mode

`setToolReference(Coords coords);`

- Function: Set coordinate system of tool
- Parameter Specification:

  X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

  RX,RY,RZ range from -180.00-180.00
- Return Value: None

`setWorldReference(Coords coords);`

- Function: Set coordinate system of world
- Parameter Specification:

  X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

  RX,RY,RZ range from -180.00-180.00
- Return Value: None

`getToolReference();`

- Function: Get coordinate system of tool

- Return Value:

X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

```
getWorldReference();
```

- Function: Get coordinate system of world
- Return Value:

X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

```
setReferenceFrame(RFType rftype);
```

- Function: Set coordinate system of frame
- Parameter Specification:

RFType::BASE takes the robot base as the base coordinate, RFType::WORLD takes the world coordinate system as the base coordinate

- Return Value: None

```
getReferenceFrame();
```

- Function：Get coordinate system of flange
- Return Value: X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

```
setEndType(EndType end_type)
```

- Function: Set coordinate system of the end
- Parameter Specification:

EndType::FLANGE set the end as flange, EndType::TOOL set the end to the tool end

- Return Value: None

```
getEndType();
```

- Function: Get coordinate system of the end
- Return Value: X,Y,Z range from -300.00-300.00 (Value range is not defined. If the value is beyond the range, the clue "inverse kinematics no solution" will be given) unit mm

RX,RY,RZ range from -180.00-180.00

JavaScript

JavaScript is a scripting language that runs on the client side; it does not need to be compiled and is interpreted and executed one by one by the js interpreter during runtime.

What is JavaScript?

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as part of a web page, and it allows client-side scripts to interact with the user and generate dynamic pages. It is an interpreted programming language with object-oriented functions.

JavaScript is a very well-known programming language, and it was originally used twenty years ago with the purpose of vivifying web pages. It is also an important part of the skill set for web developers. The JavaScript scripting language is not dependent on an operating system, and it only requires the support of a browser. Therefore, a JavaScript script can be taken to any machine after it has been written, provided that the browser on the machine supports the JavaScript scripting language. JavaScript is already supported by most browsers.

JavaScript is easy to learn but difficult to master, and it is used for a variety of purposes, from simply enhancing the functions of a website to running cool games and web-based software.

Applicable equipment：

- myCobot 280

    - myCobot 280 M5
    - myCobot 280 for Arduino

- myCobot 320
    - myCobot 320 M5

- myPalletizer 260
    - myPalletizer 260 M5

- mechArm-270
    - mechArm-270 M5

Preconditions for use：

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)

Programming development

Development environment

- **Node**

    - **Windows Node**

    - **MacOs Node**

    - **Linux Node**

Preconditions for use：

- Burn the latest version of atomMain for Atom.
- Burn the minirobot for M5Stack-basic, and choose the transponder function. And It is unnecessary to burn Basic for Pi series.

Preparations before development

> node environment building

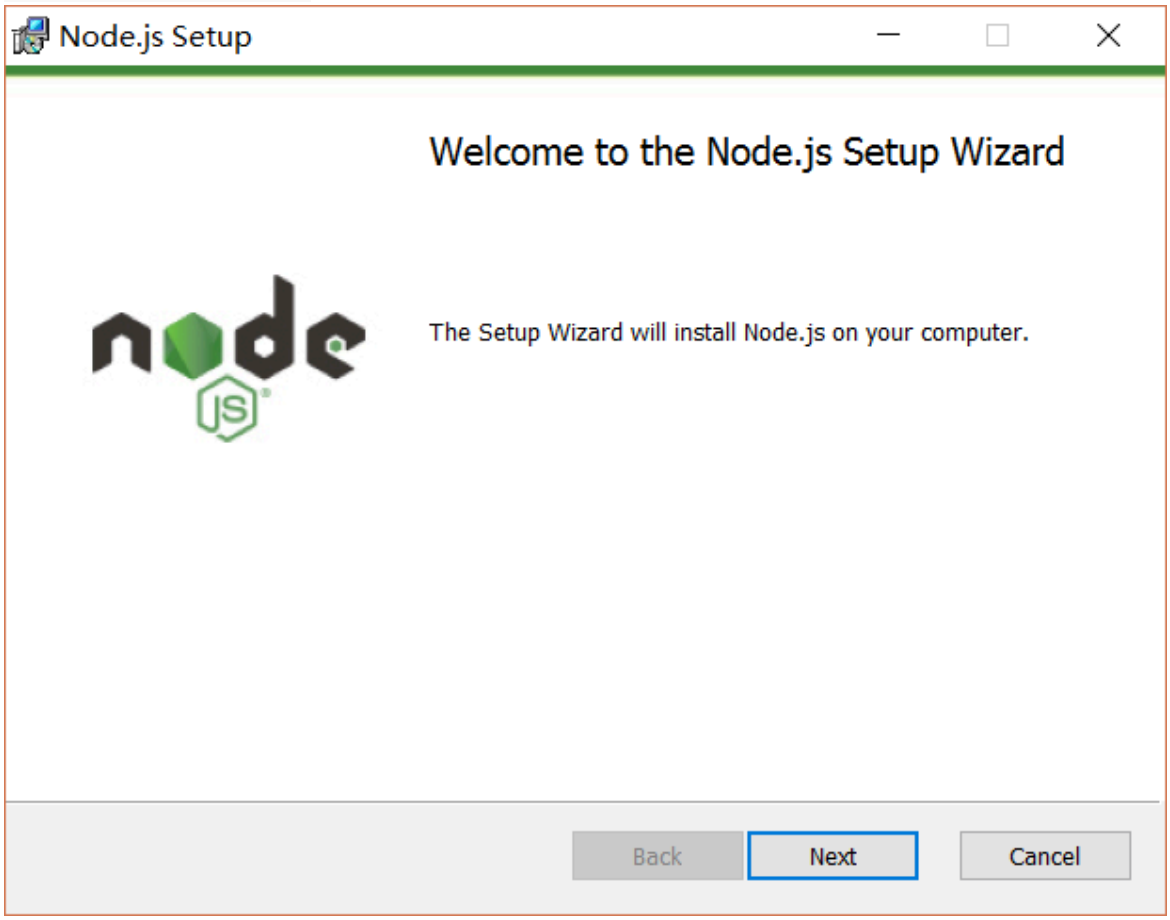
windows 32-bit downloading address

windows 64-site downloading address

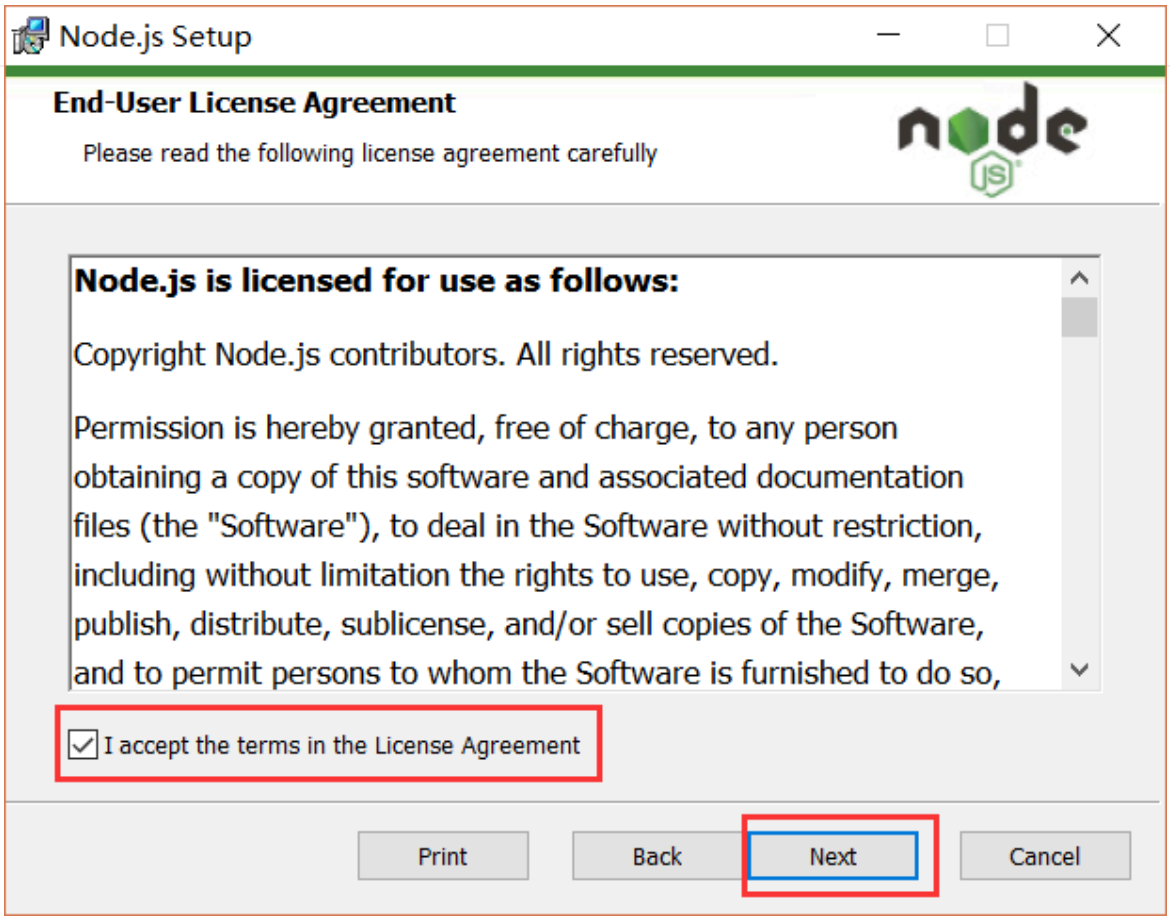
macOs downloading address

1 windows Node environment building

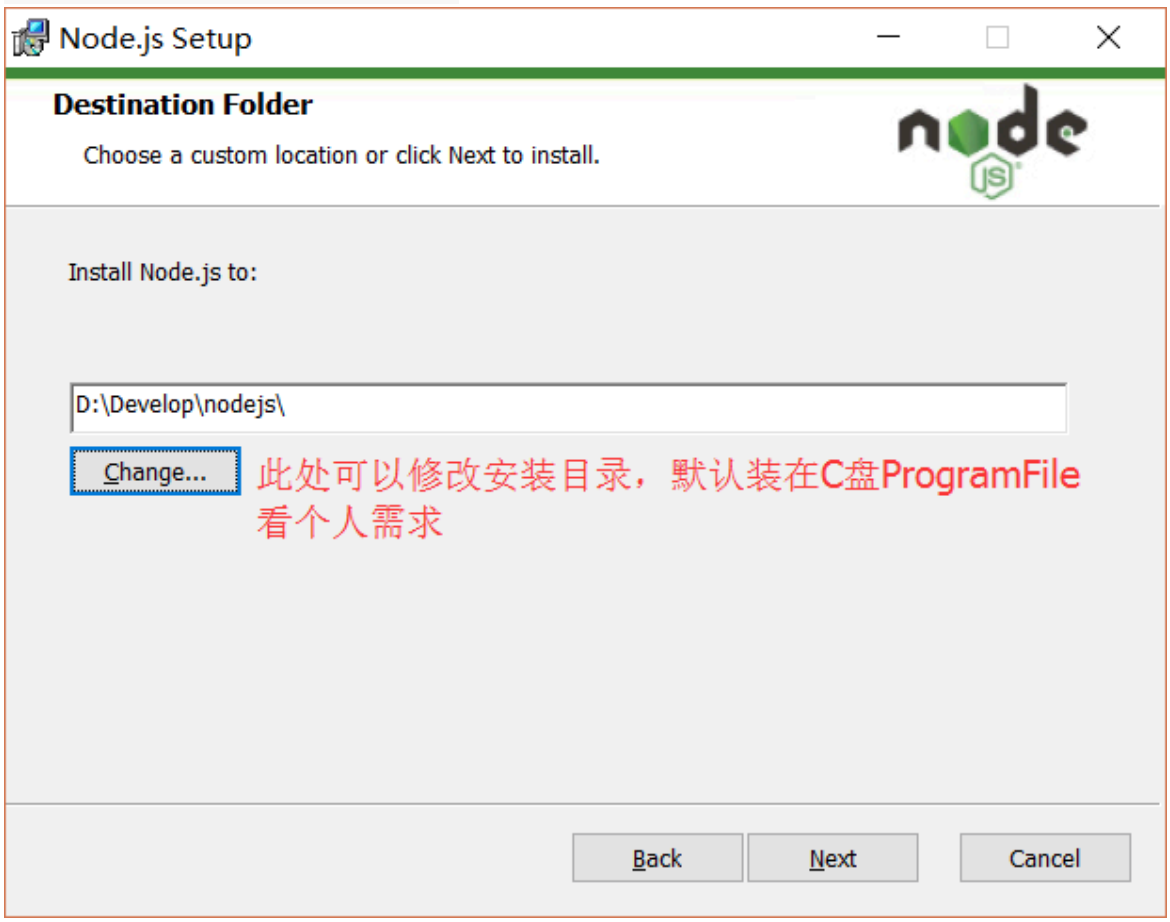
Step 1: After downloading is completed, double-click the downloaded installation package to start installing Node.js. Click the Next button



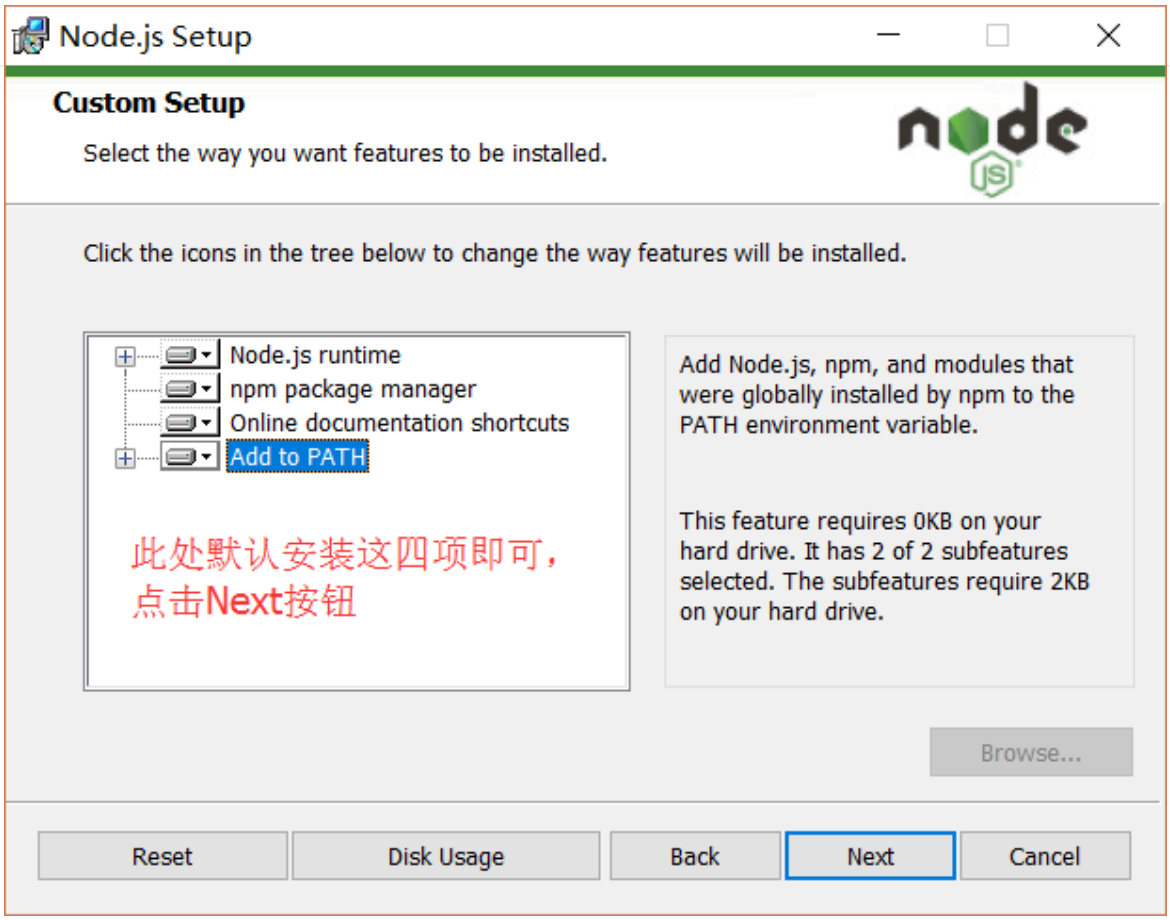
Step 2: Check the option in the lower left red box, and click Next



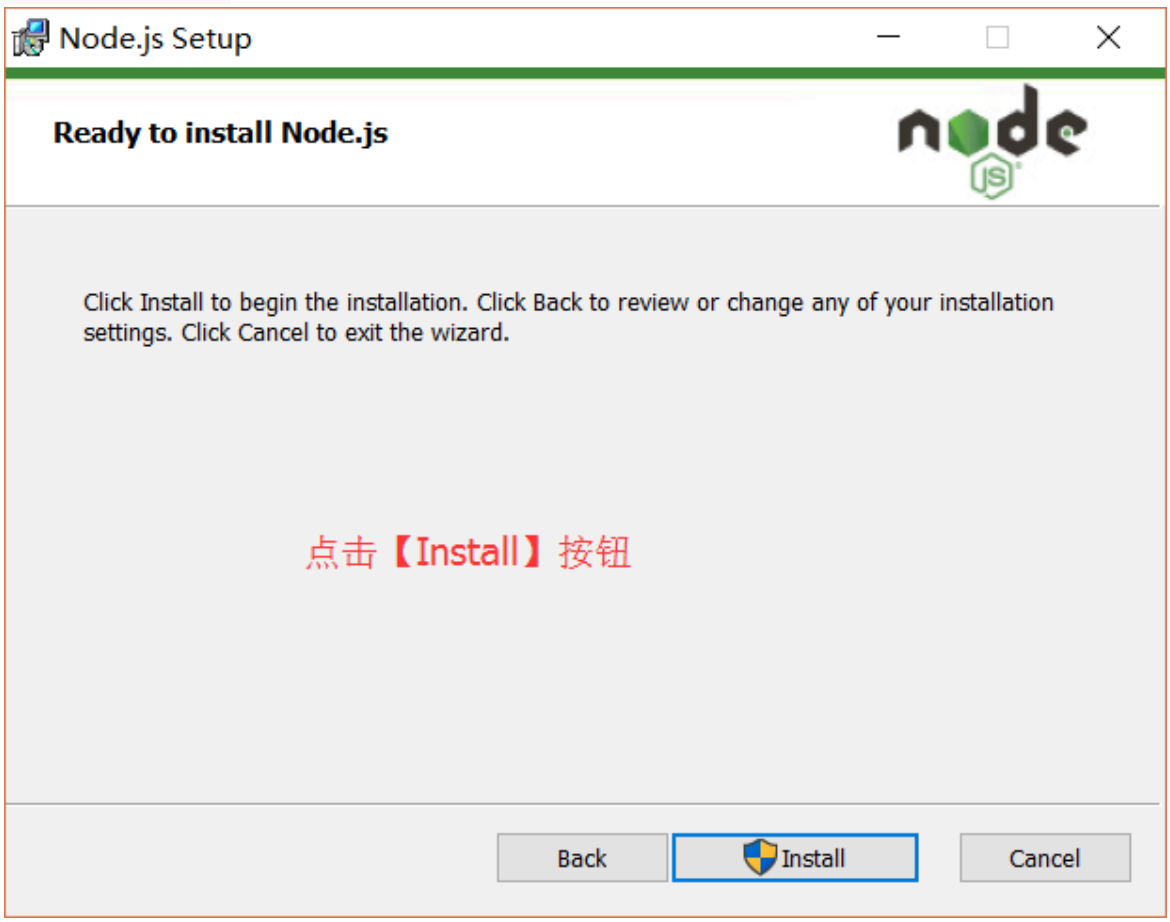
Step 3: customize the installation directory



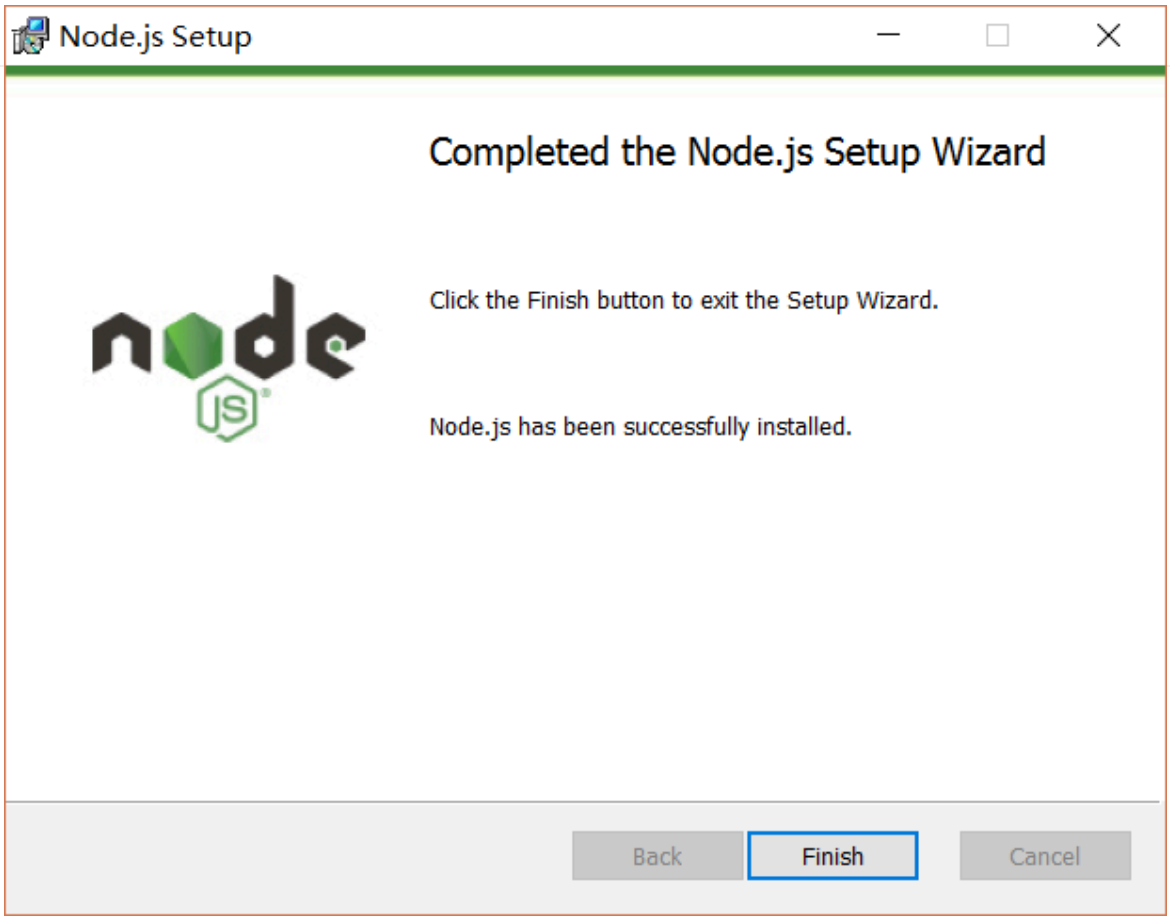
Step 4: click the Next button (default)



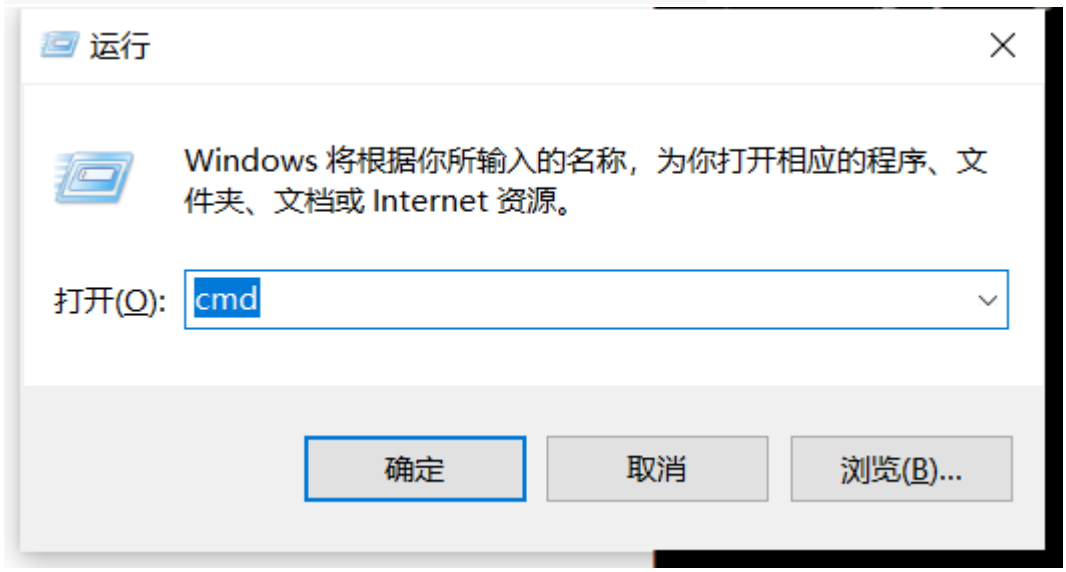
Step 5: click Install



Step 6: click Finish to complete the installation

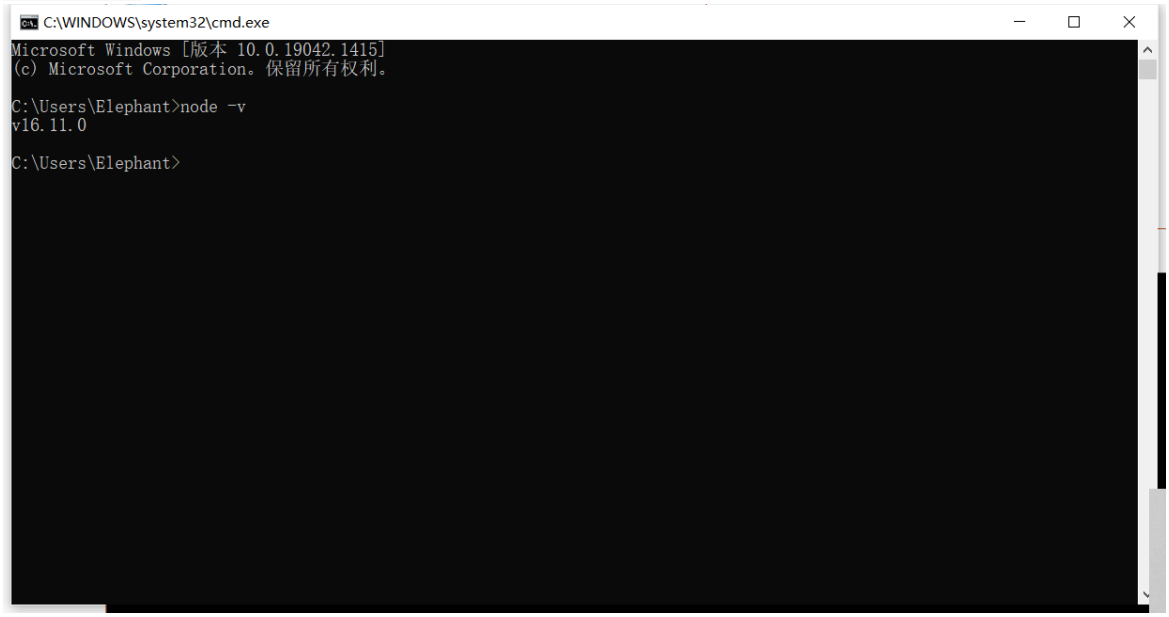


Step 7: open and urn with `win+r`, input `cmd` to a command indicator



Step 8:

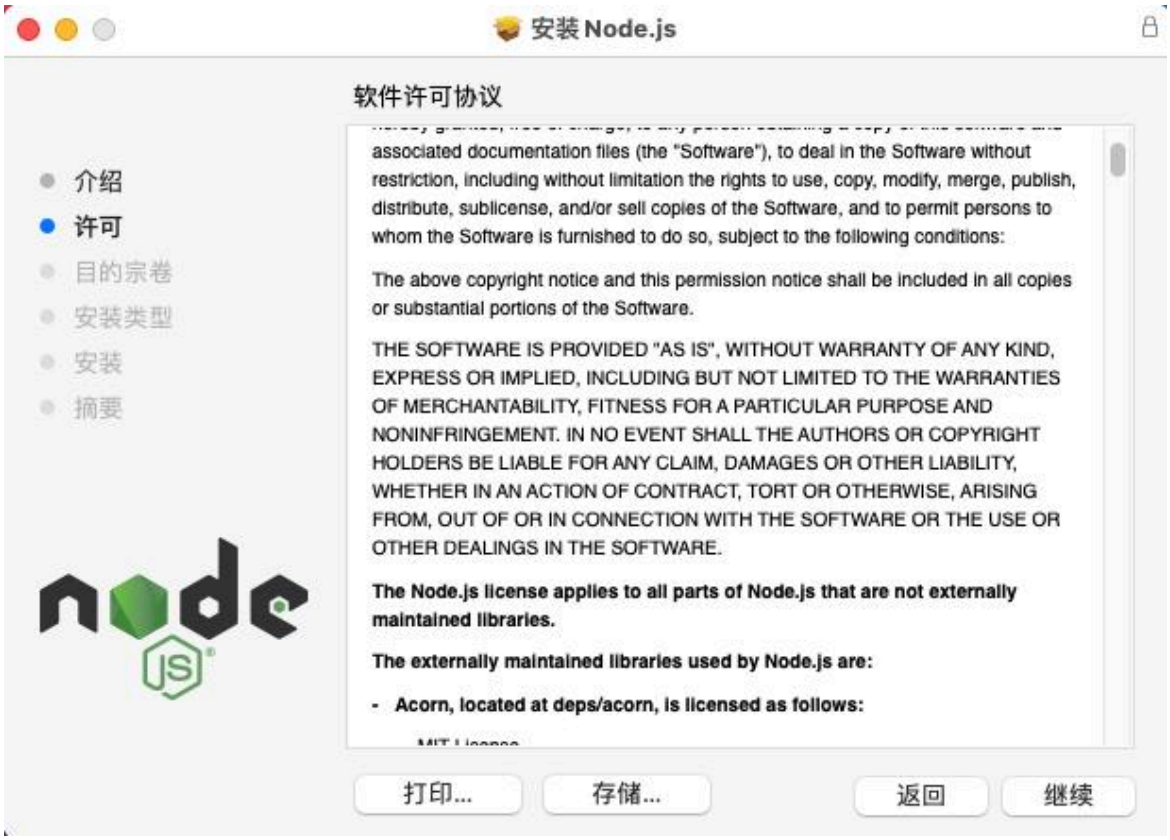
input node -v get the node version; when the version is displayed, its means that the installation is completed successfully.



2 MacOs node environment building

Step 1: After downloading is completed, double-click the downloaded installation package to start installing Node.js. Click the Next button, click Continue

Step 2: Click Continue again



Step 3: Click Agree to go to the next step

Step 4: Click Customize, choose the installation address, or click Install to continue the installation and input your password to install

Step 5: When successful installation is prompted, click Close to exit The installation process

Right-click on the desktop and select a ``punching terminal``, enter the terminal and input node -v. If the node version number is displayed, it means that installation is done successfully

3 Linux node environment building

Step 1: The official Node website has changed the linux downloading version to a compiled version, so we can directly download and unzip it for use:

```
# wget https://nodejs.org/dist/v10.9.0/node-v10.9.0-linux-x64.tar.xz    // download
# tar xf  node-v10.9.0-linux-x64.tar.xz       // unzip
# cd node-v10.9.0-linux-x64/                  // enter the unzipping directory
# ./bin/node -v                               // execute node command, and check version
v10.9.0
```

Step 2: The bin directory of the unzipped file contains commands such as node, npm, etc. We can use the ln command to set up a soft connection:

```
ln -s /usr/software/nodejs/bin/npm   /usr/local/bin/
ln -s /usr/software/nodejs/bin/node   /usr/local/bin/
```

Installing source code (Node.js)

Step 1: In the following part, we will introduce the installation of Node.js in Ubuntu Linux using source code. For other Linux systems, such as Centos, perform the installation steps below. Get Node.js source code from ``Github``.

```
$ sudo git clone https://github.com/nodejs/node.git
Cloning into 'node'...
```

Modify the permission of the directory.

```
$ sudo chmod -R 755 node
```

Step 3: Create a compiling file ``using ./``configure and follows:

```
$ cd node
$ sudo ./configure
$ sudo make
$ sudo make install
```

Step 4: Check the node version

```
$ node --version
v0.10.25
```

1 Downloading a project file

> Open the command prompt

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

2 Initializing a development program

> Note: Open the command prompt in the downloaded project file.

```
<!-- initialize the program, then install and run all plug-ins required therefor -->
npm i
```

3 Initializing the program

```
<!-- Import plugins -->
const mycobot = require("mycobot")

<!-- initialize the program, mycobot.connect(serial port number, serial baud rate) -->
const obj = mycobot.connect("COM15",115200)

<!-- write in the first command, power up the robot arm and keep its current posture -->
obj.write(mycobot.powerOn())
```

IO Control

```
<!-- initialize the program -->
const mycobot = require('mycobot')

const obj = mycobot.connect('COM15',115200)

<!-- set the color of ATOM indicator lamp-->
<!-- Note: set three parameters to 0-255 -->
obj.write(mycobot.setColor(125,11,9))

<!-- set the angle and coordinate of the current robot arm to the running start point of the robot arm -->
obj.write(mycobot.setGripperInit())
```

1 Single joint control

```
<!-- initialize the program -->
const mycobot = require('mycobot')

const obj = mycobot.connect('COM15',115200)

<!-- set the angle of single robot arm mycobot.sendAngle(robot arm ID, angle value, the running speed at which the robot a
<!-- Note:When setting the value to change the angle, pay attention to the number of robot arm joints. if the number is 4,
<!-- For setting of the angles of four-axis and six-axis robot arms, see Digram 1-3 -->

obj.write(mycobot.sendAngle(1,110,10))

<!-- set the coordinate of a single robot arm mycobot.sendCoord(robot armID, coordinate value, the the movement speed at w

obj.write(mycobot.sendCoord(1,20,10))
```

2 Multi-joint control

Note: When operating multiple joints, fill in the parameters corresponding to the number of joints of the robot arm.

```
<!-- initialize the program -->
const mycobot = require('mycobot')

const obj = mycobot.connect('COM15',115200)

<!-- set the angle of the multi-joint robot arm mycobot.sendAngles([angle of joint 1, angle of joint 2, angle of joint 3,
<!-- set the coordinate of the multi-joint robot arm mycobot.sendCoords([coordinate of joint 1, coordinate of joint 2, coo

obj.write(mycobot.sendCoords([22.5,12,-22,45],20))
```

3 Standard parameters of angles and coordinates of four-axis and six-axis robot arms

- Four-axis robot arm

Joint ID	Limited Values
1	-160~160
2	0~90
3	-90~45
4	Unlimited Values

- Six-axis robot arm

Joint ID	Limited Values
1	-170~170
2	-170~170
3	-170~170
4	-170~170
5	-170~170
6	Unlimited Values

Gripper control

```
<!-- initialize the program -->
const mycobot = require('mycobot')

const obj = mycobot.connect('COM15',115200)

<!-- set the open and closed states of the gripper mycobot.setGripperState(open and closed states , running speed)-->
<!-- Note: reference values for open and closed states:0-open, 1-closed, and running speed is limited to 0-100 -->
obj.write(mycobot.setGripperState(0,10))

<!-- set the angle of the gripper mycobot.setGripperValue(angle value, speed)-->
<!-- Note: the angle and speed values are limited to 0-100 -->
obj.write(mycobot.setGripperValue(80,20)))
```

What is js?

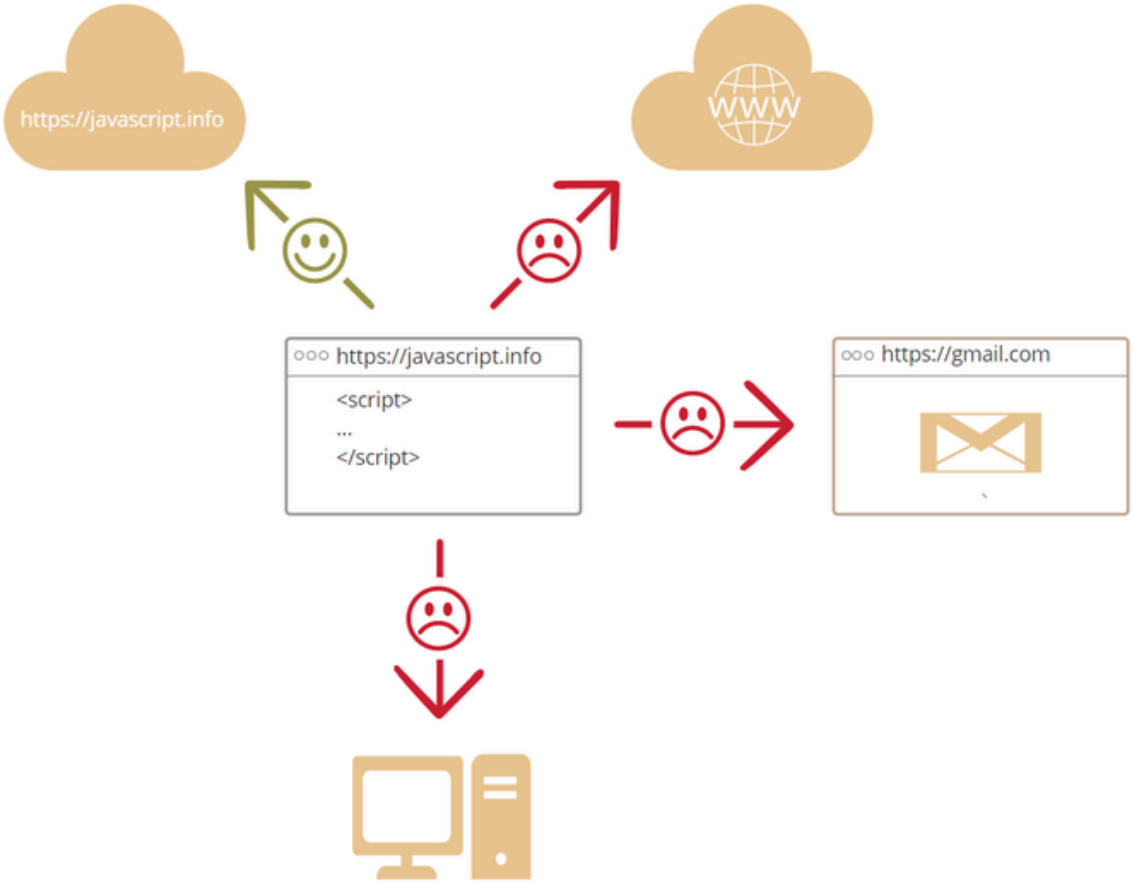
```
The original purpose of JavaScript was to "vivify webpages".

This programming language is called a script. They can be written in HTML and executed automatically when a page is being

Scripts exist and execute as plain text. They can run without special preparations or compilation.
```

What can JavaScript in a browser do?

```
Modern JavaScript is a "safe" language. It doesn't provide low-level access to memory or CPU, because it was originally cr
The power of JavaScript is largely dependent on the environment in which it is executed.
For example, Node.js allows JavaScript to read and write arbitrary files, perform network requests, etc. JavaScript in a b
For example, JavaScript in the browser can do the following:

- insert a new HTML in a webpage, modify contents and styles of an existing webpage;
- respond to user actions, mouse clicks or movements, and keyboard strokes;
- send network requests to a remote server to download or upload files (so-called AJAX and COMET technologies);
- get or modify cookies, ask questions and send messages to visitors;
- remember client's data (local storage).
```

What can\'t JavaScript in a browser do?

```
For its users'(information) security, the power of JavaScript in browsers are limited. This is mainly to prevent malicious
Examples of these restrictions are:
- JavaScript in webpages cannot read, write, copy and execute files or programs on the user's disk. It hasn't the function
Modern browsers allow JavaScript to do some file-related operations, but this operation is limited.
only when the user uses a certain action, JavaScript can manipulate a file.
For example, "dragging" a file into the browser, or selecting a file via the input tag. There are many ways for JavaScript
So, a JavaScript-enabled webpage shouldn't surreptitiously launch a webcam to watch you and send your information to the A
- Different browser tabs are basically unrelated to each other.
Sometimes, they also have some relationships with each other. For example, a tab opens another new tab via JavaScript. But
This is the "same origin policy". In order to solve the problem of "same-origin policy", both tabs must contain some speci
- JavaScript can easily communicate with the server of the current webpage domain name via the Internet.
But the ability to get data from the server of other websites/domains is limited.
Although this is possible, it requires explicit protocol (in HTTP headers) from the remote server.
This is also for the user's data security.
```



What makes javascript different

```
+ Full integration with HTML/CSS.
+ Use simple tools to accomplish simple tasks.
+ Supported by all

The browser that meets these three browser technologies is JavaScript only.

That's why JavaScript is different! That's also why it's the most common tool for creating browser interfaces.

In addition, JavaScript also supports the creation of servers, mobile applications, etc.
```

Case 1

```
<!-- Initialization procedure -->
const mycobot = require('mycobot')
const obj = mycobot.connect('COM15',115200)

<!-- Suppose a four axis manipulator is connected -->
const name = "myPallizer"
<!-- If the connected equipment is a six axis manipulator -->
if(name == "myCobot"){
    <!-- Sets the angle of each of the six joints -->
    obj.write(mycobot.sendAngles([-1.49,115,-153.45,30,-33.42,137.9],80))
    <!-- And judge whether the coordinate is reached -->
    if(obj.write(mycobot.isInPosition([-1.49,115,-153.45,30,-33.42,137.9],0)) == 1){
        <!-- Resume manipulator movement -->
        obj.write(mycobot.programResume())
        <!-- Wait 0.5 seconds -->
        settimeout(() =>{
            <!-- Pause the manipulator movement -->
            obj.write(mycobot.programPause())
        },500)
    }
}else{
    <!-- Sets the angle of each of the four joints -->
    obj.write(mycobot.sendAngles([-1.49,45,-23,30],80))
    <!-- Sets the color of the atom indicator -->
    obj.write(mycobot.setColor(0,0,50))
    <!-- Wait 0.7 seconds -->
    settimeout(() =>{
        <!-- Set the angles of the four joints again -->
        obj.write(mycobot.sendAngles([-1.49,60,11,30],80))
        <!-- Sets the color of the atom indicator -->
        obj.write(mycobot.setColor(0,50,0))
    },700)

}
```

Case 2

```
<!-- Initialization procedure -->
const mycobot = require('mycobot')
const obj = mycobot.connect('COM15',115200)

<!-- Gets the angle of all current joints -->
const botData = obj.write(mycobot.getAngles())
<!-- Outputs the angle of all current joints -->
console.log(botdata)
<!-- Sets the angle of all current joints -->
obj.write(mycobot.sendAngles([0,0,0,0],50))
<!-- Print whether the robot arm currently reaches this position -->
console.log(obj.write(mycobot.isInPosition([0,0,0,0,0,0],0)) == 1)

<!-- Wait for 3 seconds -->
settimeout(() =>{
    <!-- Sets the angle of the first joint -->
    obj.write(mycobot.sendAngle(1,90,50))
},3000)

<!-- Wait two seconds -->
settimeout(() =>{
    <!-- Set the second joint angle to 50 degrees -->
    obj.write(mycobot.sendAngle(2,50,50))
},2000)
<!-- Wait 1.5 seconds -->
settimeout(() =>{
    <!-- Set the joint triangulation to 50 degrees -->
    obj.write(mycobot.sendAngle(3,-50,50))
},1500)
<!-- Wait 1.5 seconds -->
settimeout(() =>{
    <!-- Set all joint angles of the manipulator -->
    obj.write(mycobot.sendAngles([88.68, -138.51, 155.65, -128.05, -9.93, -15.29],50))
},1500)
<!-- Wait for 2.5 seconds -->
settimeout(() =>{
    <!-- Set the manipulator to free mode -->
    obj.write(mycobot.releaseAllServos())
},2500)
```

myCobot

This is javascript API for mycobot.

MyCobot

Import to your project:

```
// basic demo
var mycobot = require("mycobot")

// obj Based on SerialPort
var obj = mycobot.connect("COM15",115200)

obj.write(mycobot.powerOn())

// Receive returned data
// obj.on("data",(data)=>{
//     res = mycobot.processReceived(data)
//     console.log("res:", res)
// })
```

Overall status

connect

- **Prototype**: `connect(port, baud)`
- **Description**:Create objects, connect devices (default open).

powerOn

- **Prototype**: `powerOn()`
- **Description**:Atom open communication (default open).

powerOff

- **Prototype**: `powerOff()`
- **Description**: Atom turn off communication.

isPowerOn

- **Prototype**: `isPowerOn()`
- **Description**: Adjust robot arm whether power on.
- **Returns**
  - `1` : power on

- `0` : power off

releaseAllServos

- **Prototype**: `releaseAllServos()`
- **Description**: Set robot arm into free moving mode.

MDI mode and operation

getAngles

- **Prototype**: `getAngles()`
- **Description**: Get the degree of all joints.
- **Returns**: `object` : A float list of all degree.

sendAngle

- **Prototype**: `sendAngle(id, degree, speed)`
- **Description**: Send one degree of joint to robot arm.
- **Parameters**
  - `id` : Joint id
  - `degree` : degree value(`number`)
  - `speed` : (`number`) 0 ~ 100
- **Example**

```
obj.write(mycobot.sendAngle(1,50,50))
```

sendAngles()

- **Prototype**: `sendAngles(angles, speed)`
- **Description**: Send the degrees of all joints to robot arm.
- **Parameters**
  - `degrees` : a list of degree value(`Array`).
  - `speed` : (`number`) 0 ~ 100
- **Example**

```
obj.write(mycobot.sendAngles([0,0,0,0,0,0],60))
```

getCoords

- **Prototype**: `getCoords()`
- **Description**: Get the Coords from robot arm, coordinate system based on base.
- **Returns**: `object` : A float list of coord - `mycobot:[x, y, z, rx, ry, rz]; mypalletizer:[x, y, z, θ]`

sendCoord

- **Prototype**: `sendCoord(id, coord, speed)`
- **Description**: Send one coord to robot arm.
- **Parameters**
  - `id` : coord id
  - `coord` : coord value(`number`)
  - `speed` : (`number`) 0 ~ 100
- **Example**

```
obj.write(mycobot.sendCoord(x,20,50))
```

sendCoords

- **Prototype**: `sendCoords(coords, speed, mode)`
- **Description**: Send all coords to robot arm.
- **Parameters**
  - `coords` : a list of coords value(`Array`).
  - `speed` : (`number`) 0 ~ 100
  - `mode` : `0` - angular, `1` - linear
- **Example**

```
obj.write(mycobot.sendCoords([160, 160, 160, 0, 0, 0], 70, 0))
```

isInPosition

- **Prototype**: `isInPosition(data, flag)`
- **Description**: Judge whether in the position.
- **Parameters**
  - `data` : A data list, angles or coords.
  - `flag` : Tag the data type, `0` - angles, `1` - coords.
- **Returns**
  - `1` - true
  - `0` - false

JOG mode and operation

jogAngle

- **Prototype**: `jogAngle(jointId, direction, speed)`
- **Description**: Jog control angle
- **Parameters**
  - `jointId` : (`int`) 1 ~ 6

- `direction` : `0` - decrease, `1` - increase
- `speed` : 0 ~ 100

jogCoord

- **Prototype**: `jogCoord(coordId, direction, speed)`
- **Description**: Jog control coord.
- **Parameters**
  - `coordId` : (`int`) 1 ~ 6
  - `direction` : `0` - decrease, `1` - increase
  - `speed` : 0 ~ 100

jogStop

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

programPause

- **Prototype**: `programPause()`
- **Description**: Pause movement.

programResume

- **Prototype**: `programResume()`
- **Description**: Recovery movement.

stop

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

setEncoder

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

getEncoder

- **Prototype**: `getEncoder(jointId)`
- **Description**:Obtain the specified joint potential value.
- **Parameters**: `jointId` : 1 ~ 6

- **Returns**: `encoder` : 0 ~ 4096

setEncoders

- **Prototype**: `setEncoders(encoders, speed)`
- **Description**: Set the six joints of the manipulator to execute synchronously to the specified position.
- **Parameters**:
  - `encoders` : A encoder list, length 6.
  - `speed` : 0 - 100

getEncoders

- **Prototype**: `getEncoders()`
- **Description**: Get the all joints of the manipulator.
- **Returns**: the list of encoder (`Array`)

Running status and Settings

getSpeed

- **Prototype**: `getSpeed()`
- **Description**: Get speed.
- **Returns**: speed

setSpeed

- **Prototype**: `setSpeed(speed)`
- **Description**: Set speed.
- **Parameters**: speed: 0 ~ 100

getJointMin

- **Prototype**: `getJointMin(jointId)`
- **Description**: Gets the minimum movement angle of the specified joint
- **Parameters**: `jointId`
- **Returns**: angle value (`float`)

getJointMax

- **Prototype**: `getJointMax(jointId)`
- **Description**: Gets the maximum movement angle of the specified joint
- **Parameters**: `jointId`
- **Returns**: angle value (`float`)

Servo control

isServoEnable

- **Prototype**: `isServoEnable(servoId)`
- **Description**: Determine whether all steering gears are connected
- **Parameters**: `servoId` 1 ~ 6
- **Returns**
  - `0` : disable
  - `1` : enable

isAllServoEnable

- **Prototype**: `isAllServoEnable()`
- **Description**: Determine whether the specified steering gear is connected
- **Returns**
  - `0` : disable
  - `1` : enable

setServoData

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

getServodata

- **Prototype**: `getServodata(servo_no, dataId)`
- **Description**: Read the data parameter of the specified address of the steering gear.
- **Parameters**:
  - `servo_no` : Serial number of articulated steering gear, 1 - 6.
  - `dataId` : Data address.
- **Returns**: `value` : 0 - 4096
  - `0` : disable
  - `1` : enable
  - `-1` : error

setServoCalibration

- **Prototype**: `setServoCalibration(servo_no)`
- **Description**: The current position of the calibration joint actuator is the angle zero point, and the corresponding potential value is 2048.

- **Parameters**:
  - `servo_no` : Serial number of articulated steering gear, 1 - 6.

releaseServo

- **Prototype**: `releaseServo(servoId)`
- **Description**: Power off designated servo
- **Parameters**: `servoId` : 1 ~ 6

focusServo

- **Prototype**: `focusServo(servoId)`
- **Description**: Power on designated servo
- **Parameters**: `servoId` : 1 ~ 6

Atom IO

setColor

- **Prototype**: `setColor(r, g, b)`
- **Description**: Set the color of the light on the top of the robot arm.
- **Parameters**
  - `r` : 0 ~ 255
  - `g` : 0 ~ 255
  - `b` : 0 ~ 255

setPinMode

- **Prototype**: `setPinMode(pin_no, pinMode)`
- **Description**: Set the state mode of the specified pin in atom.
- **Parameters**
  - `pin_no` : Pin number.
  - `pinMode` : 0 - input, 1 - output, 2 - inputPullup

setDigitalOutput()

- **Parameters**
  - `pin_no` :
  - `pinSignal` : 0 / 1

getDigitalInput()

- **Parameters**: `pin_no`
- **Return**: signal value

getGripperValue

- **Prototype**: `getGripperValue()`
- **Description**: Get gripper value
- **Return**: gripper value

setGripperState

- **Prototype**: `setGripperState(flag, speed)`
- **Description**: Set gripper switch state
- **Parameters**
  - `flag` : 0 - open, 1 - close
  - `speed` : 0 ~ 100

setGripperValue

- **Prototype**: `setGripperValue(value, speed)`
- **Description**: Set gripper value
- **Parameters**
  - `value` : 0 ~ 100
  - `speed` : 0 ~ 100

setGripperIni

- **Prototype**: `setGripperIni()`
- **Description**: Set the current position to zero, set current position value is `2048` .

isGripperMving

- **Prototype**: `isGripperMving()`
- **Description**: Judge whether the gripper is moving or not
- **Returns**
  - `0` : not moving
  - `1` : is moving

M5Stack-basic

setBasicOutput

- **Prototype**: `setBasicOutput(pin_no, pinSignal)`
- **Description**: Set bottom pin.
- **Parameters**
  - `pin_no` : Pin number.

- `pinSignal` : 0 / 1

getBasicOutput

- **Prototype**: `getBasicOutput(pin_no)`
- **Description**: get bottom pin.
- **Parameters**
  - `pin_no` : Pin number.

ROS/ROS2 Introduction

ROS is the abbreviation of robot operating system. **ROS** is a highly flexible software architecture for writing robot software programs.

Notice:

- Currently, **myCobot 280 series**、**myCobot 320 series** 、**myPalletizer 260 series**、**mechArm 270 series** support the use of ROS. for specific situation of the development of various equipment, refer to **equipment development**.
- Burn the corresponding firmware by using **mystudio**. Burn the minirobot in M5Stack-basic, choose the transponder function, and burn the latest version of atomMain in atom.

ROS Logo：

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**.

Due to the following disadvantages of **ROS**:

- Limited real-time communication
- System stability does not yet meet industrial-grade requirements
- No security measures
- Only supports Linux(ubuntu)
- The performance of the core mechanism is not optimized to occupy resources

Therefore, **ROS** cannot really enter the industry, and naturally cannot be commercialized. To solve this problem, the community proposed **ROS 2**. It makes **ROS** have the characteristics of productization, including real-time performance, adaptability to all platforms, suitable for hardware with low performance (MCU+RTOS), distributed, data encryption and support for modern programming languages.

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

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

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

- myCobot PRO 600

- 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)

Device Description:

- Among the devices used above, mechArm-270 PI、myCobot 280-Pi, myCobot 280-JetsonNano, and myCobot 320-Pi versions have their own Ubantu (V-18.04 / V-20.04) system and have a built-in development environment, so they can be used directly without setting up management.
- MyCobot 280-M5, myCobot 320-M5, myCobot 280-Arduino, myPalletizer 260, and mechArm-270-M5 versions need to be built in an environment, but in ROS/ROS2, you only need to build a ROS environment or build a ROS2 environment.

MoveIt Introduction

MoveIt is currently the most advanced software for movement operations of robot arms and has been used for more than 100 robots. It integrates the latest achievements in motion planning, control, 3D perception, motion control, control and navigation, provides an easy-to-use platform for developing advanced robot applications, and provides an integrated software platform for design, and integration assessment of new robot products in the fields of industry, commerce, R&D, etc.

MoveIt Logo :

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

- myCobot PRO 600

- mechArm-270
  - mechArm-270 PI
  - mechArm-270 M5

Preconditions for use:

- **M5** series version， the bottom **M5Stack-basic** is programmed to miniRobot , select the **Transponder** function, and the end **ATOM** is programmed to the latest version of atomMain (the factory default has been programmed)
- **Pi \ jetsonnano** series, **ATOM** burns the latest version of **atomMain** (factory default already burnt)

ROS Introduction

ROS is an open-source meta-operating system used for robots. It provides an operating system with expected services, including hardware abstraction, low-level device control, implementation of common functions, messages transfered between processes, and package management. It also provides the tools and library functions needed to obtain, compile, write, and run codes across computers.

The "graph" of ROS runtime is a loosely coupled peer-to-peer process network based on a ROS communication infrastructure. ROS implements several different communication methods, including a services mechanism based on synchronous RPC-style communication, a topics mechanism based on asynchronous streaming media data, and a parameter server for data storage.

ROS is not a real-time framework, but it can be embedded in real-time programs. Willow Garage's PR2 robot uses a system called pr2_etherCAT to send or receive ROS messages in real time. ROS may also be seamlessly integrated with Orocos real-time toolkits.

1 Design goals and characteristics of ROS

Many people ask "what are differences between ROS and other robot software platforms?" This question is difficult to answer. Because ROS is not a framework that integrates most functions or features. In fact, the main goal of ROS is to provide code reuse support for robot R&D. ROS is a framework (i.e. nodes) for distributed processes, which are encapsulated in program and function packages that can be easily shared and distributed. ROS also supports a federated system similar to a code repository, and this system also enables the collaboration and distribution of a project. This design enables the development and realization of a project to to be decided completely independently from the file system to the user interface (no limit by ROS). At the same time, all projects can be integrated with the basic tools of ROS.

To support the primary goals of sharing and collaboration, the ROS framework has several other features:

- Lean: ROS is designed to be as lean as possible so that codes written for ROS can be used with other robot software frameworks. The inevitable conclusion from this is that ROS can be easily integrated into other robot software platforms: ROS can already be integrated with OpenRAVE, Orocos and Player.
- ROS insensitive libraries: The preferred development model of ROS is written with clean library functions that do not depend on ROS.
- Language independence: The ROS framework can simply be implemented in any modern programming language. ros has implemented Python version, C++ version and Lisp version. It also has experimental libraries for Java and Lua versions.
- Loose coupling: The function modules in ROS are encapsulated in independent function packages or meta-function packages, which are easy to share. The modules in the function package are run in units of nodes. With ROS standard IO used as the interface, developers does not need to pay attention to the internal implementation of the module, as long as they understand the interface rules, they can achieve reuse and point-to-point loose coupling between modules.
- Convenient testing: ROS has a built-in unit/integration testing framework called rostest, which can easily install or uninstall test modules.
- Scalable: ROS is applicable to large runtime systems and large development processes.
- Free and open-source: It has many developers and many function packages.

2 Why ROS is used?

Through ROS, we can realize the simulated control of robot arms in a virtual environment.

we will visualize robot arms through **rviz**, operates our robot arms in a variety of ways, and plan and executes the action path of robot arms through **MoveIt** to achieve the effect of freely controlling the robot arms.

We will learn how to control our products through the platform in ros in the following chapters.

M5 Version:

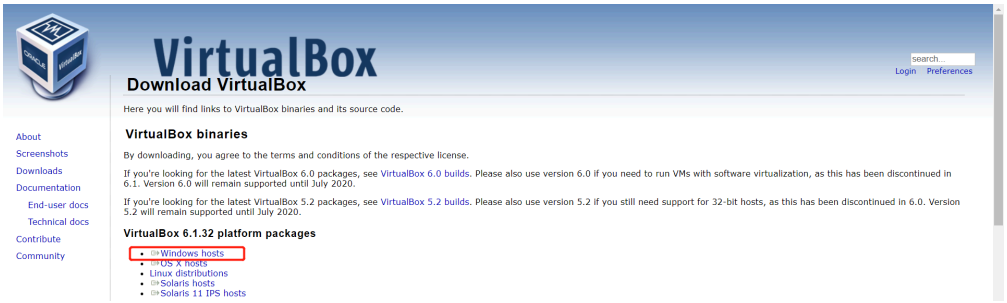
Install different versions of ubuntu system in Linux

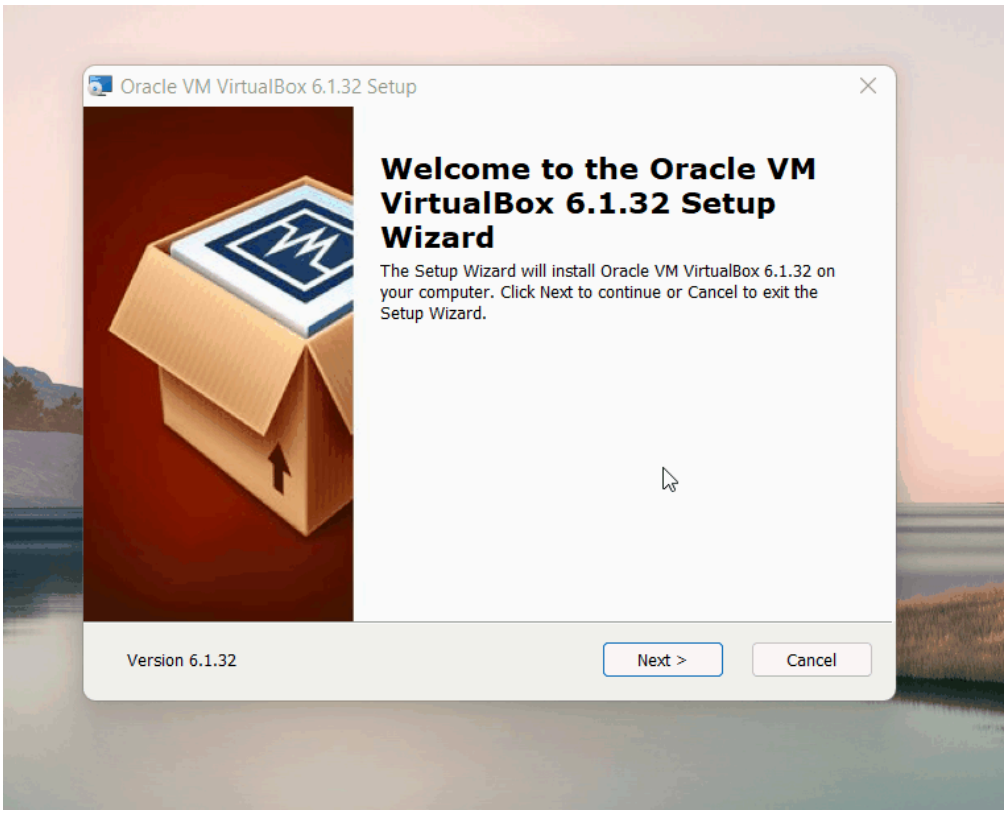
1 Virtual machine installation

Go to Official website to download the virtual machine Virtual Box or go to Official website to download the virtual machine VMware.

Of course, if you already have your virtual machine, you may skip this step.

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



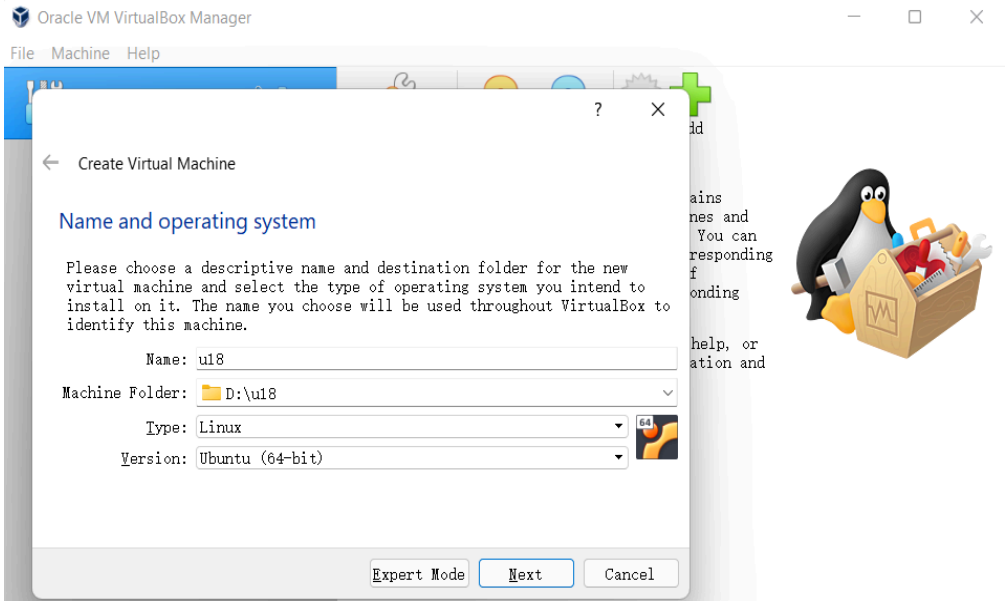


2 Creating a virtual machine

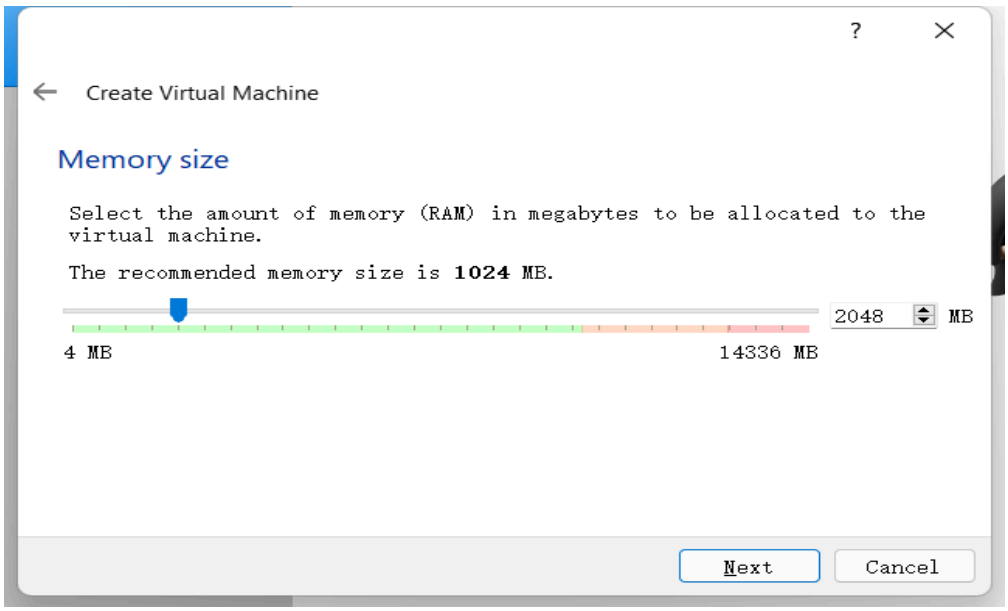
2.1 Creating a virtual machine

Select Create in control

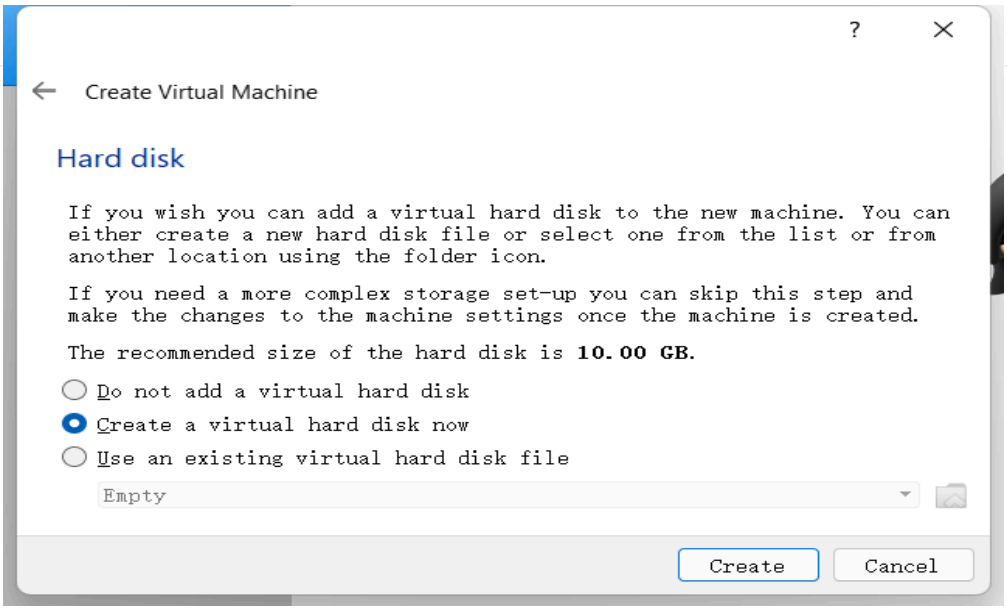
- Input the name of the virtual machine and the location where it is stored, select the type of virtual machine as Linux, select the ubuntu64-bit version, and go to the next step.



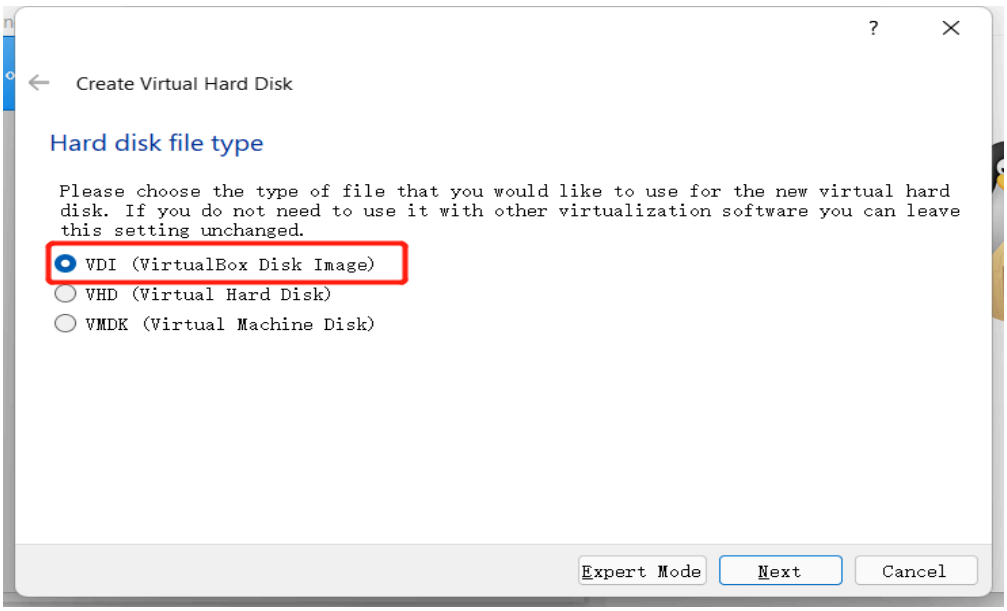
- Configure a memory size according to your own needs, and then go to the next step.



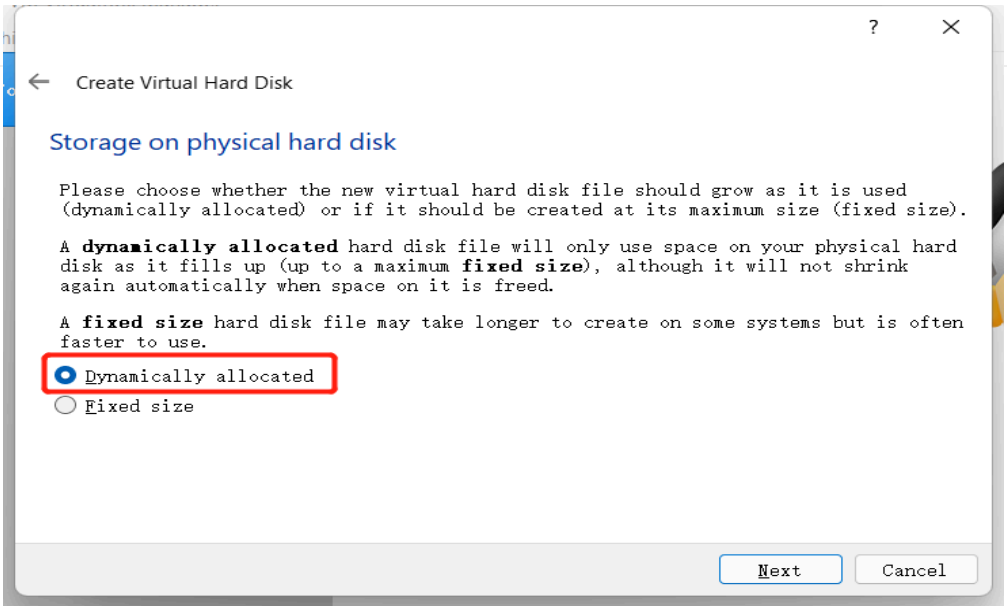
- Choose **Create a virtual hard disk now** to create it.

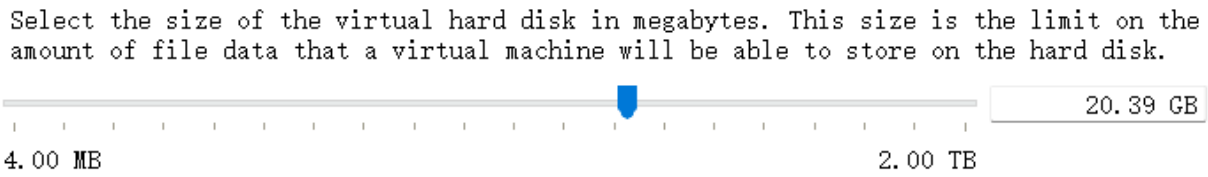


- Select the virtual hard disk type as **VDI** type, and go to the next step.



- Allocate the size of the virtual hard disk. Since an ubuntu system needs to be installed, and the operation will be performed in the system, it is recommended that the size should not be less than 20G.





3 Importing the ubuntu system

3.1 Downloading the ubuntu system

Select the ubuntu version according to your own needs and install it:

Note: ROS2 needs to download **version 20.04**.

- Version 16.04
- Version 18.04
- Version 20.04

The installation method and process of the three versions are the same. Here, we take the version 18.04 version for an example.

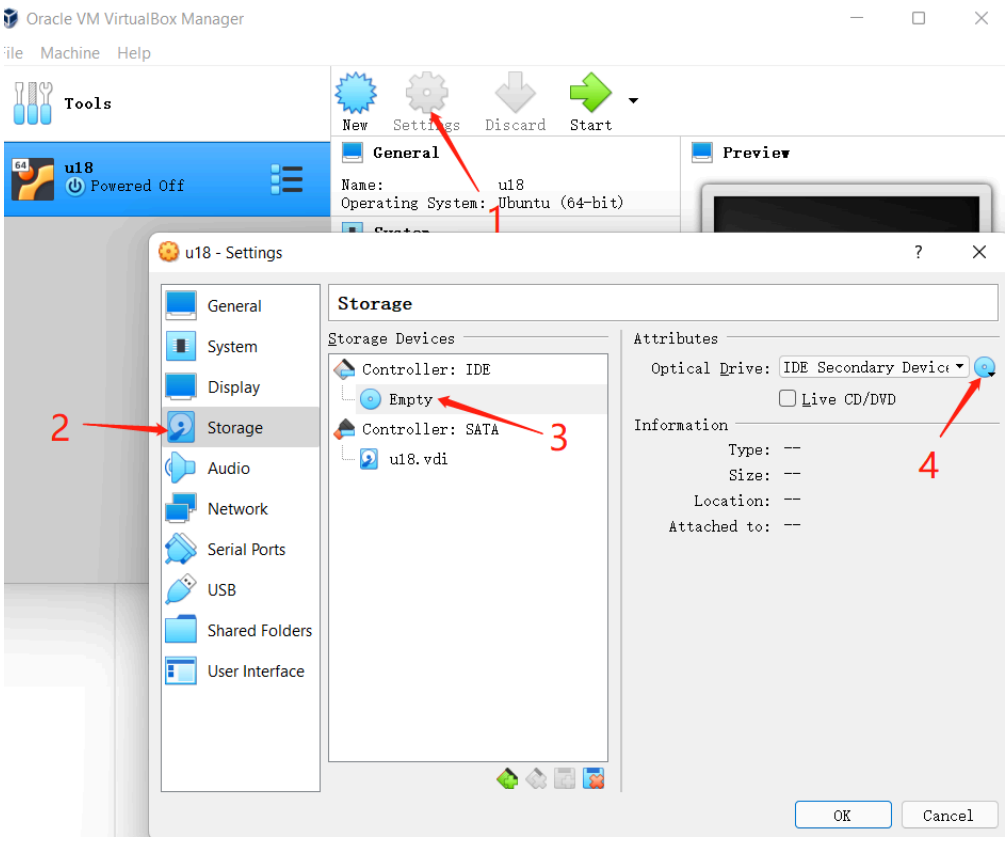

If you need help burning these images to disk, see the Image Burning Guide.

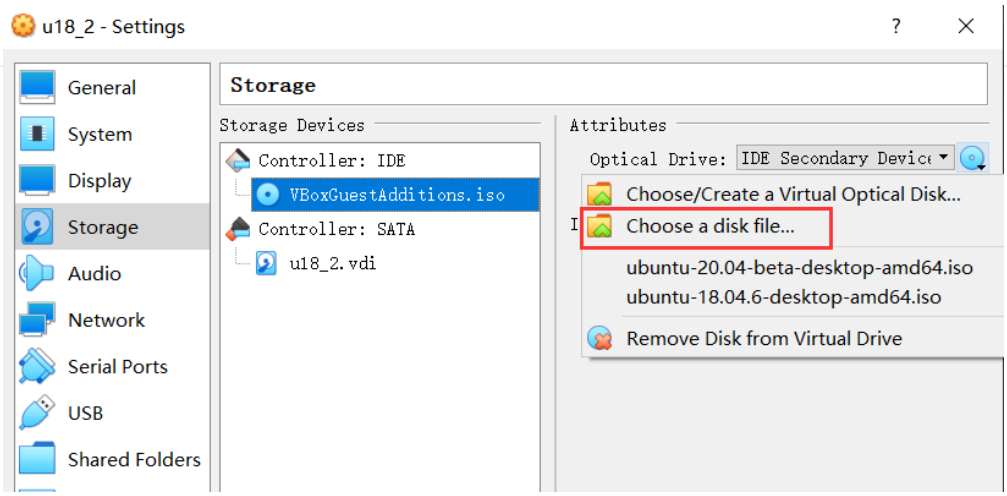
Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS-metalink	2020-02-12 13:42	296	
MD5SUMS-metalink.gpg	2020-02-12 13:42	916	
SHA256SUMS	2021-09-16 21:58	202	
SHA256SUMS.gpg	2021-09-16 21:58	833	
ubuntu-18.04.6-desktop-amd64.iso	2021-09-15 20:42	2.3G	Desktop image for 64-bit PC (AMD64) computers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

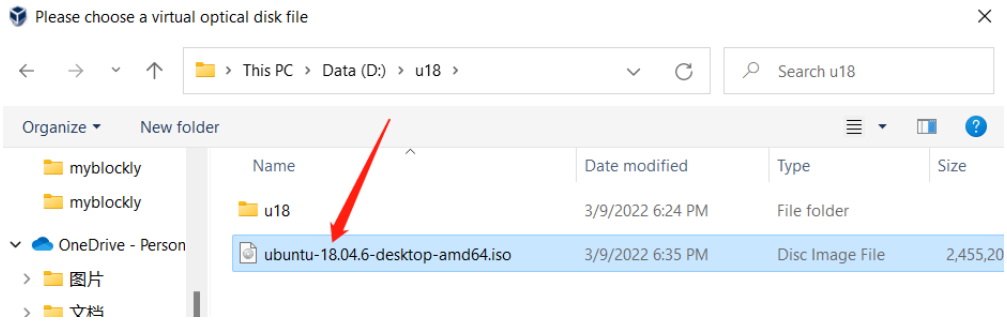
After the downloading is complete, there will be a file shown in the figure:

3.2 Importing ubuntu into a visual machine

Find the previously installed virtual machine in the Virtual box, enter **Setup**, and assign a CD to the controller in **Storage**:





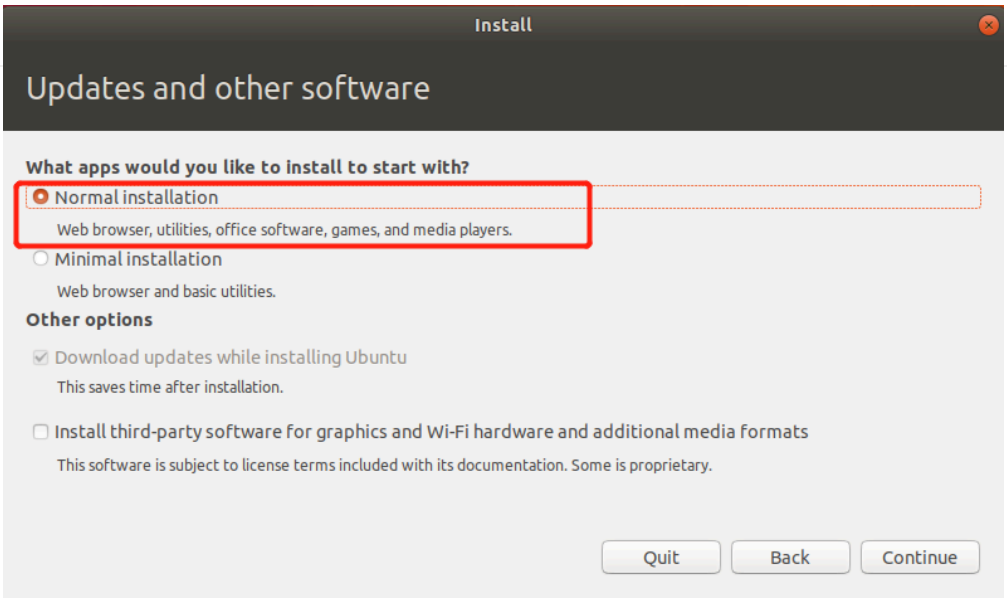


Then open the virtual machine for ubuntu installation, and click start.

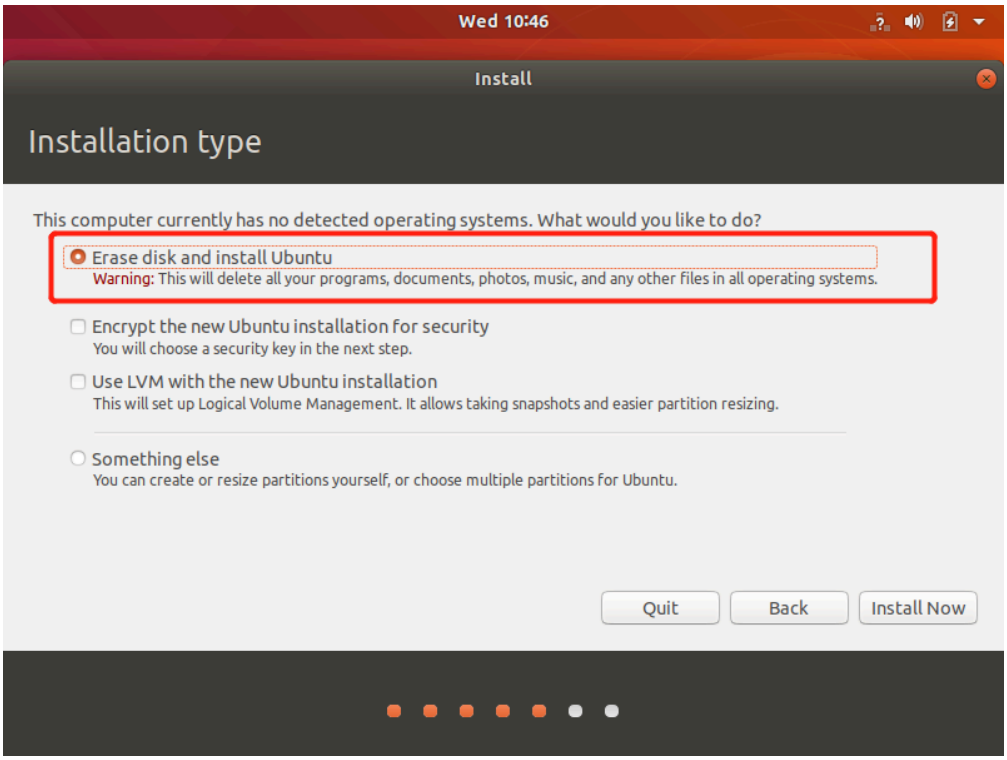
3.3 Installing ubuntu

- Wait for the system to start, enter a **welcome** interface, select "Chinese (Simplified)", and click the "Install Ubuntu" button;

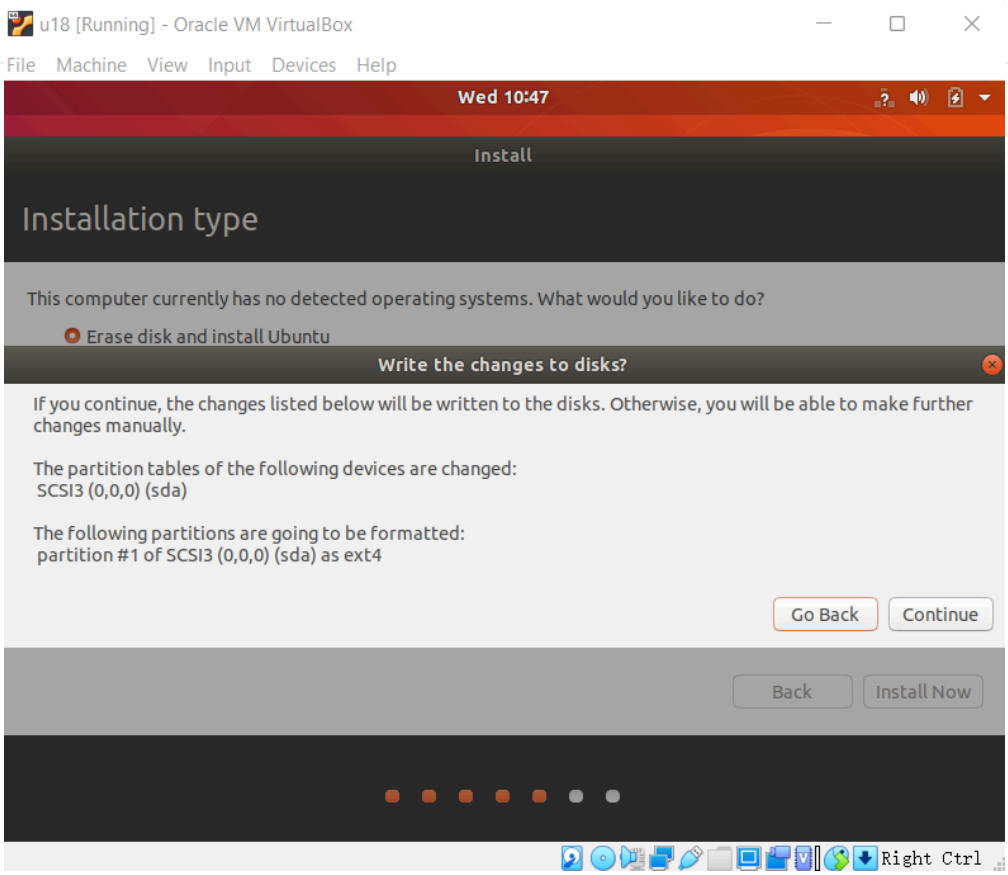
- Click “Continue” button;



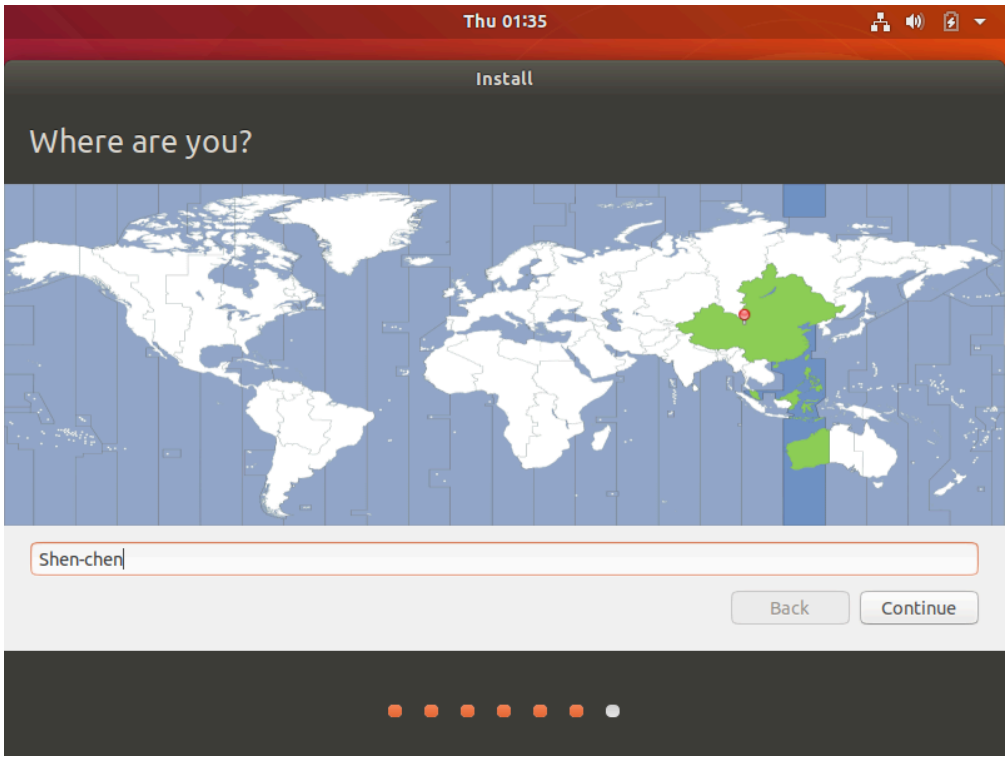
- Select the "Clear entire disk and install Ubuntu" option, and click the "Install Now" button;



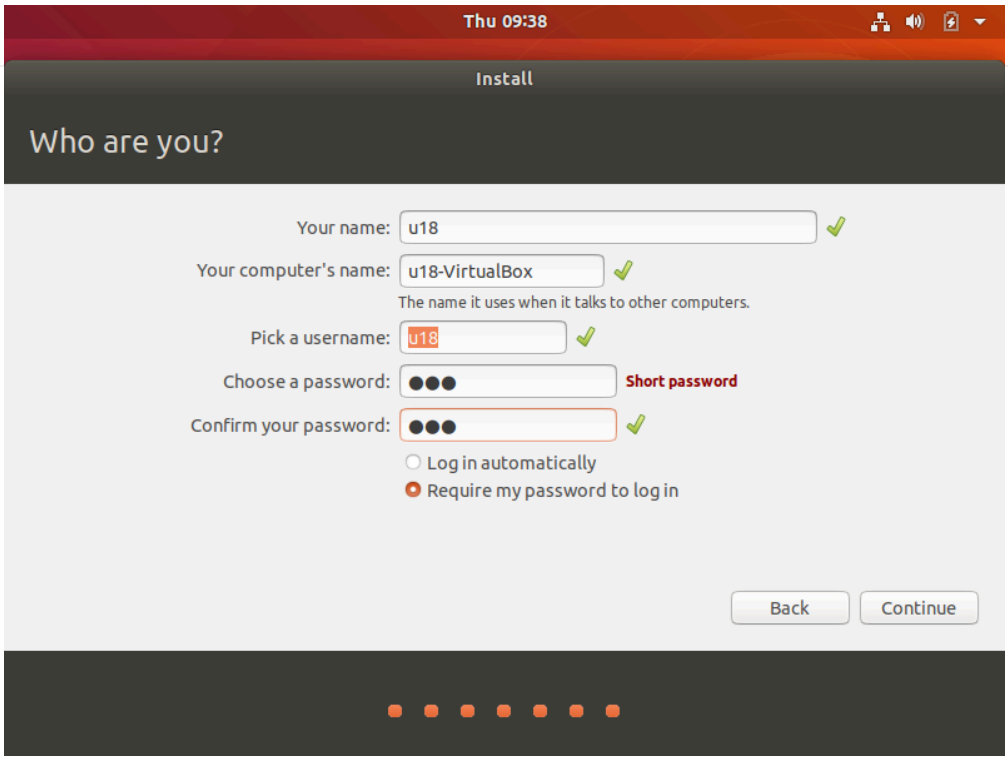
- Click the "Continue" button in the pop-up dialog box;



- Set a geographic position and click the “Continue” button;

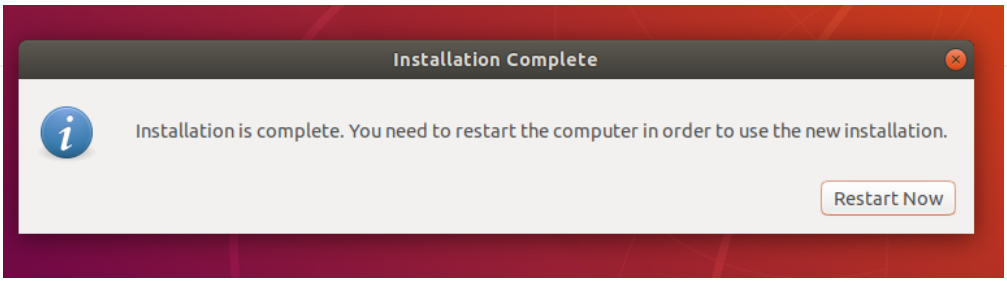


- Set a user name and password, and click the “Continue” button;



- Enter the system installation interface, and wait patiently;

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



1 ROS Environment building

1.1 Installing ROS

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

For **myCobot 280-M5** and **myCobot 320-M5 devices**, refer to the installation methods and procedures described below. For **myCobot 280-PI** and **myCobot 320-PI** devices, you only need to install the mycobot_ros installation package.

1.1.1 Selecting a version

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

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

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

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

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

1.1.2 Installing ROS

1 Adding a software source

There is no ROS software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

- official source:

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

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, Tsinghua University is:

```
sudo sh -c '. /etc/lsb-release && echo "deb http://mirrors.tuna.tsinghua.edu.cn/ros/ubuntu/ `lsb_release -cs` main" > /etc
```

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

2 Setting a key

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

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

The execution result is displayed as follows:

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

3 Installation

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

```
sudo apt-get update
```

Execute **Installing ROS**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command according to your Ubuntu version:

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

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

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

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

```
Fetched 523 MB in 2min 34s (3,401 kB/s)
E: Failed to fetch http://us.archive.ubuntu.com/ubuntu/pool/universe/w/wxwidgets
3.0/libwxbase3.0-0v5_3.0.4+dfsg-3_amd64.deb  Undetermined Error [IP: 91.189.91.3
9 80]
```

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

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

- 将sources.list中的官方软件源全部替换成下面的阿里云软件源：

Ubuntu 16.04 version：

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

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

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

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

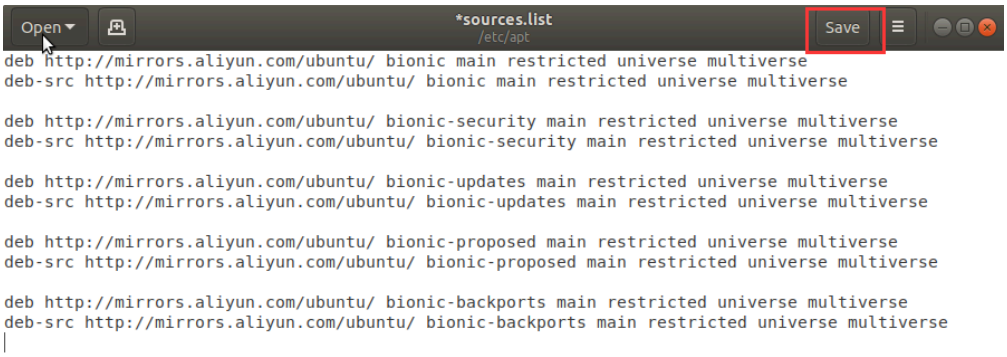
Ubuntu 18.04 version：

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

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

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

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

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

Ubuntu 20.04 version：

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

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

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

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

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

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



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

```
sudo apt-get update
```

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

```
# Ubuntu 16.04
sudo apt install ros-kinetic-desktop-full
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-desktop-full
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-desktop-full
```

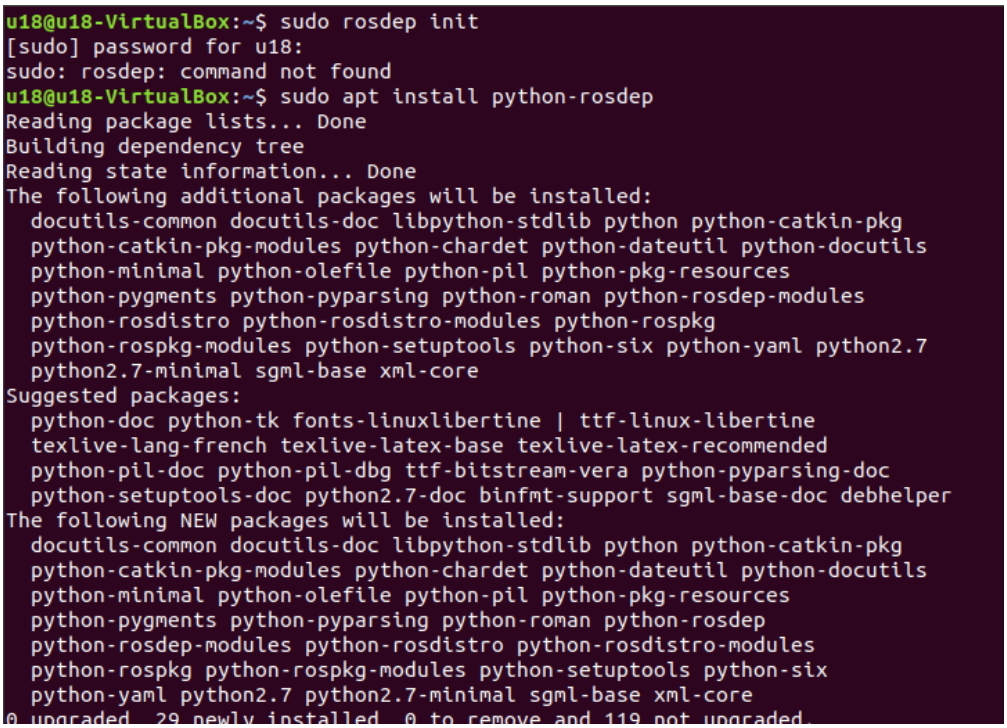
The installation process takes a long time, so please wait with patience.

4 Configuring the ROS environment to the system

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

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

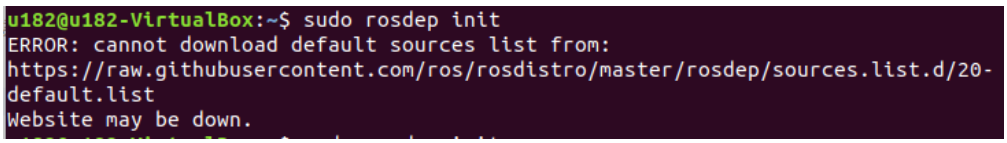
If the system of the Ubuntu installed by you is version 20.04, use the command `sudo apt install python3-rosdep` to it install. After completion, execute the rosdep initialization command.



initialize rosdep:

```
sudo rosdep init
```

If the following error message appears:

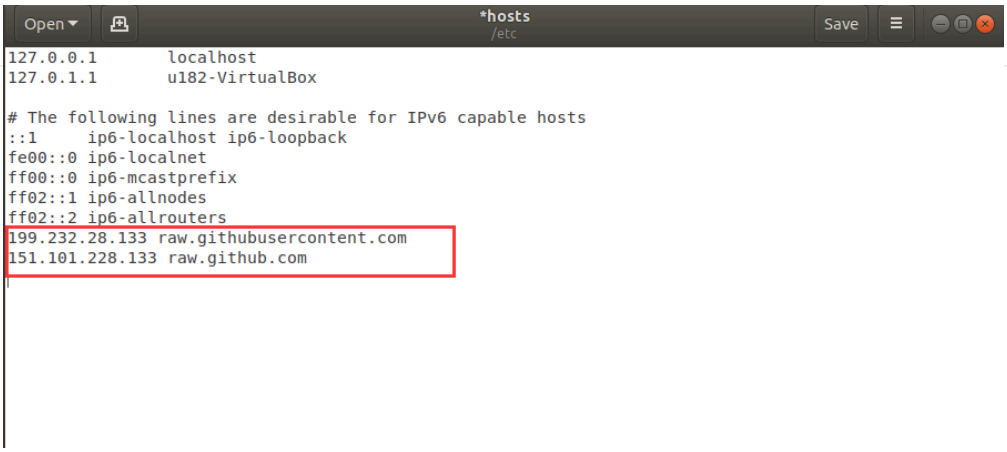


Solution: Modify the hosts file and enter the following command in the console terminal:

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

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

```
199.232.28.133 raw.githubusercontent.com
151.101.228.133 raw.github.com
```



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

```
sudo rosdep init
```

```
rosdep update
```

After the initialization is completed, in order to avoid re-validating the ROS function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

5 Set up the ros environment

Bash

Execute the following commands:

```
# Ubuntu 16.04
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Zsh

If you replace bash with zsh, then:

```
# Ubuntu16.04
echo "source /opt/ros/kinetic/setup.bash" >> ~/.zshrc
```

```
# Ubuntu18.04
echo "source /opt/ros/melodic/setup.bash" >> ~/.zshrc
```

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

```
source ~/.zshrc
```

6 Installing a ROS extra dependency

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

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

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

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

```
# Ubuntu 16.04
sudo apt install ros-kinetic-joint-state-publisher-gui
```

```
# Ubuntu 18.04
sudo apt install ros-melodic-joint-state-publisher-gui
```

```
# Ubuntu 20.04
sudo apt install ros-noetic-joint-state-publisher-gui
```

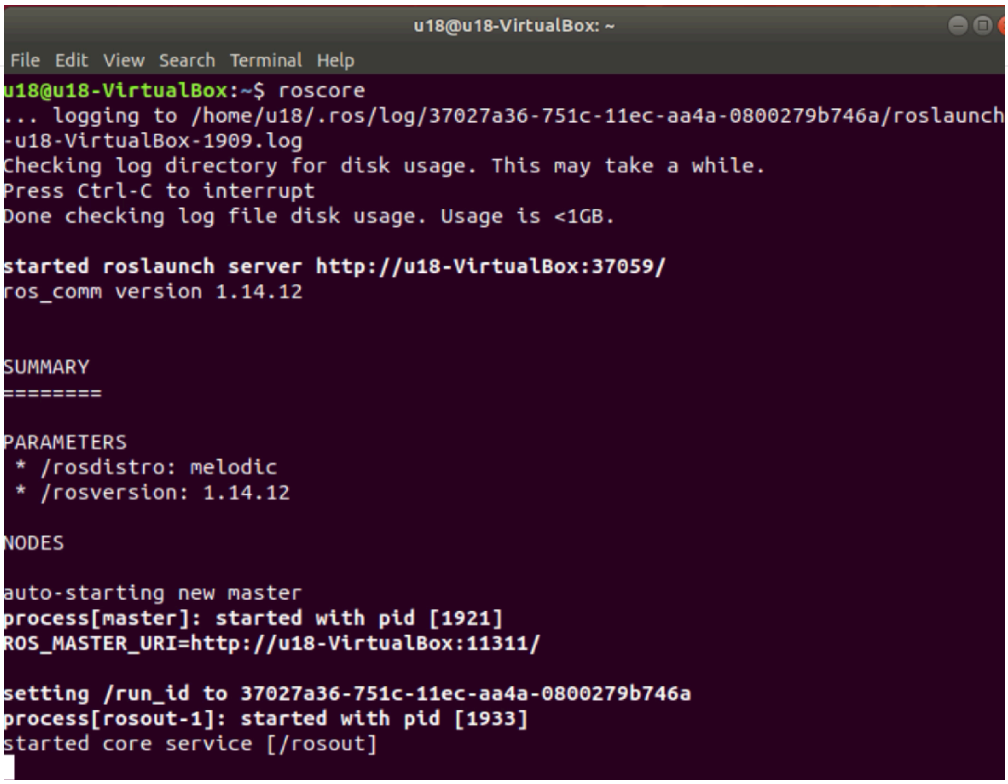
1.1.3 Verifying ROS

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

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

```
roscore
```

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



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

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

1.2 MoveIt Installation

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

1.2.1 Updating the software source list

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

```
sudo apt-get update
```

1.2.2 Installing MoveIt

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

```
# Ubuntu16.04
sudo apt-get install ros-kinetic-moveit
```

```
# Ubuntu 18.04
sudo apt-get install ros-melodic-moveit
```

```
# Ubuntu20.04
sudo apt-get install ros-noetic-moveit
```

1.3 git Installation

1.3.1 Adding a software source

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

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

1.3.2 Updating the software source list

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

```
sudo apt-get update
```

1.3.3 Installing git

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

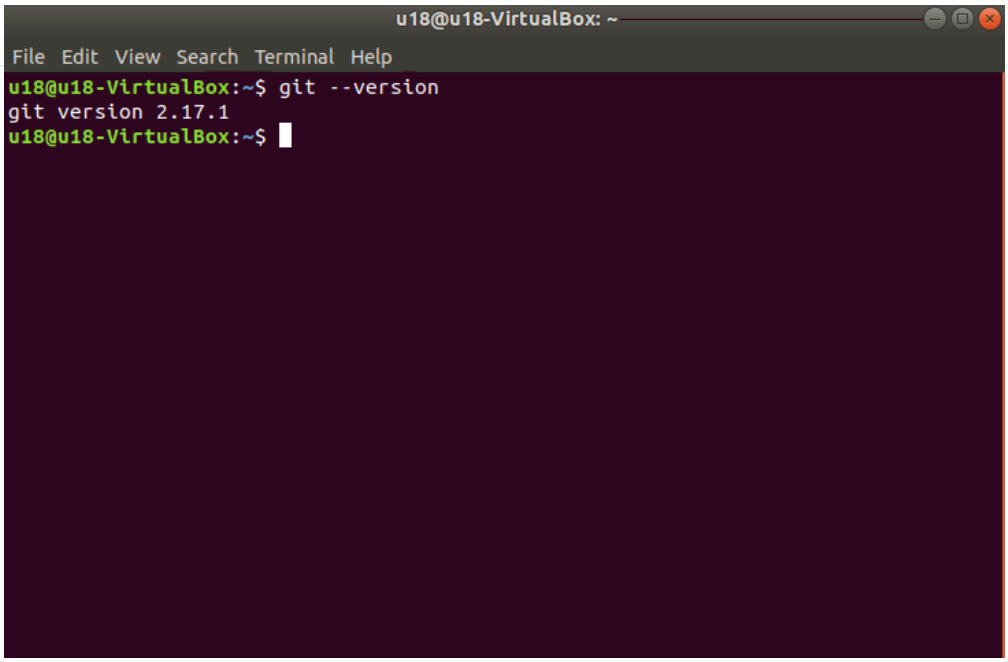
```
sudo apt-get install git
```

1.3.4 Verifying git

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

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.



1.3.5 Use

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

- https://git-scm.com/book/zh/v2
- https://www.runoob.com/git/git-tutorial.html

2 mycobot_ros Installation

`mycobot_ros` is a ROS package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros

2.1 Precondition

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

Here is an **example command for creating a workspace**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command in the command line:

```
mkdir -p ~/catkin_ws/src  # Create a folder
cd ~/catkin_ws/src        # Enter the folder
catkin_init_workspace     # Initialize the current directory into a ROS workspace
cd ..                     # Return to the parent directory
catkin_make               # Build the code in the workspace
```

Add workspace environment

Bash

The official default ROS1 workspace is `catkin_ws` .

```
# Ubuntu 16.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 18.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

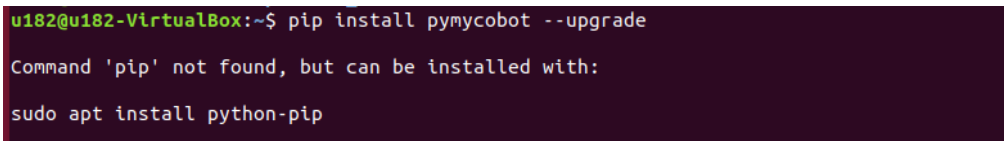
```
source ~/.bashrc
```

2.2 Installing the package

NOTE:

- This package depends on ROS and MoveIT. Before using it, make sure that you have installed ROS and MoveIT successfully.
- The interaction of this package with a real robot arm depends on PythonApi - `pymycobot`.
- The Api project location is: https://github.com/elephantrobotics/pymycobot
- Quick installation: `pip install pymycobot --upgrade`

  When executing the pip install pymycobot --upgrade command, if the following error message appears:

  Enter the following command to install pip at the prompt

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

- If your Ubuntu system is version 20.04, please execute the command `sudo apt install python3-pip` to install pip

  After the pip installation is complete, the terminal executes again

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

- The installation method depends on Git, so make sure that Git have been installed in your computer.

The official default ROS1 workspace is `catkin_ws`.

```
cd ~/catkin_ws/src  # Enter the src folder of the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd ..       # return to the workspace
catkin_make # Build the code in the workspace
source devel/setup.bash # add environment variables
```

PI Version:

The Raspberry Pi version comes with an Ubuntu (V-20.04) system and a built-in development environment, so you don't need to build and manage it, Just update the mycobot_ros package.

`mycobot_ros` is a ROS1 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

ROS1 Project address: http://github.com/elephantrobotics/mycobot_ros

Robotic arm API driver library address: https://github.com/elephantrobotics/pymycobot

1 Update the mycobot_ros package

In order to ensure that users can use the latest official package in time (new users do not need to update), they can go to the `/home/er/catkin_ws/src` folder through the file manager and open the console terminal (shortcut `Ctrl+Alt+T`) , enter the following command to update:

```
# Clone the code on github
cd ~/catkin_ws/src
git clone https://github.com/elephantrobotics/mycobot_ros.git # Please check the attention section below before deciding w
cd ..     # Back to work area
catkin_make # Build the code in the workspace
source devel/setup.bash # add environment variable
```

Note: If the `mycobot_ros` folder already exists in the `/home/er/catkin_ws/src (equivalent to ~/catkin_ws/src)` directory, you need to delete the original `mycobot_ros` before executing the above command. Among them, ubuntu in the directory path is the user name of the virtual machine. If it is inconsistent, please modify it.

So far, the ROS1 environment construction has been completed.Please refer to 13.1.3 Rviz Introduction and Use for the use of ROS1.

1 Service and Topic

We provide some services and topics to interact with mycobot.

1.1 Service

> Note: myCobot Pro 600 and myCobot Pro 630 devices do not support Service usage.

Service is a request-response communication mechanism. One node can provide a service, and another node can request the service and wait for a response. Services are usually used to perform time-consuming operations or tasks that require interaction, such as performing calculations and obtaining data.

Related commands and instructions:

command	Detailed description
rosservice list	Display active service information
rosservice info [service name]	Display information about the specified service
rosservice type [service name]	Show service type
rosservice find [service name]	Find a service for a specified service type
rosservice uri [service name]	show ROSRPC URI service
rosservice args [service name]	show service parameters
rosservice call [service name] [parameters]	Request service with input parameters

1.2 Topic

Topic is a publish-subscribe communication mechanism. Nodes can publish messages to a topic, or subscribe to a topic to receive messages. Multiple nodes can publish and subscribe to the same topic at the same time to broadcast and receive information. Topic is mainly used for data transmission with low real-time requirements, such as sensor data, status information, etc.

Related commands and instructions:

Command	Detailed description
rostopic list	Display active topic list
rostopic echo [topic name]	Display the message content of the specified topic in real time
rostopic find [type name]	Display threads with messages of the specified type
rostopic type [topic name]	Displays the message type of the specified topic
rostopic bw [topic name]	Display the message bandwidth of the specified topic（bandwidth）
rostopic hz [topic name]	Display the message data publishing cycle of the specified topic
rostopic info [topic name	Display information about the specified topic
rostopic pub [topic name] [message type] [parameters]	Post a message with the specified topic name

The difference between service and topic:

	topic	service
Synchronization	Asynchronous	Synchronous
communication model	pub/sub	server/client
underlying protocol	ROSTCP/ROSUDP	RPC
Feedback Mechanism	No	Yes
buffer	Yes	No
Real-time	Weak	Strong
Node Relationship	Many-to-Many	One-to-Many
Applicable Scenarios	Data Transmission	Logical Processing

you can go toservice and topic learn more about the use of these two features

2 Introduction to msg and srv

- msg：The msg file is a simple text file describing the fields of a ROS message. They are used to generate source code for messages in different languages (c++ or python, etc.).
- srv：srv files are used to describe services. It consists of two parts: the request (request) and the response (response). msg files are stored in the msg directory of the package, and srv files are stored in the srv directory.

2.1 rosmsg

rosmsg is a command line tool for displaying information about ROS message types.

rosmsg demo:

```
rosmsg show       # Show message description
rosmsg info       # Display message information
rosmsg list       # list all messages
rosmsg md5        # Display md5 encrypted message
rosmsg package    # Display all messages under a feature pack
rosmsg packages   # List feature packs that contain messages
```

- rosmsg list will list all msgs in the current ROS
- rosmsg packages List all packages containing messages
- rosmsg package List all msgs under a package

```
//rosmsg package  # Package names
rosmsg package turtlesim
```

- rosmsg show Show message description

```
//rosmsg show # message name
rosmsg show turtlesim/Pose
# result:
float32 x
float32 y
float32 theta
float32 linear_velocity
float32 angular_velocity
```

- rosmsg info Works the same as rosmsg show
- rosmsg md5 A check algorithm to ensure the consistency of data transmission

2.2 rossrv

rossrv is a command-line tool for displaying information about ROS service types, and uses a syntax that is highly similar to rosmsg.

```
rossrv show       # Display service message details
rossrv info       # Display information about service messages
rossrv list       # List all service information
rossrv md5        # Display md5 encrypted service messages
rossrv package    # Display all service messages under a package
rossrv packages   # Show all packages that contain service messages
```

- rossrv list Will list all srv messages in the current ROS
- rossrv packages List all packages that contain service messages
- rossrv package List all msgs under a package

```
//rossrv package # Package names
rossrv package turtlesim
```

- rossrv show Show message description

```
//rossrv show # message name
rossrv show turtlesim/Spawn
# result:
float32 x
float32 y
float32 theta
string name
---
string name
```

- rossrv info The effect is the same as rossrv show
- rossrv md5 Use md5 checksum (encryption) for service data

3 Introduction to URDF

- Unified Robot Description Format，Unified Robot Description Format, abbreviated as URDF. The urdf package in ROS contains a C++ parser for URDF, and URDF files describe robot models in XML format. *URDF cannot be used alone, it needs to be combined with Rviz or Gazebo. URDF is just a file that needs to be rendered into a graphical robot model in Rviz or Gazebo.

3.1 urdf file description

Code example:

Only part of the code is intercepted here for display:

```
<?xml version="1.0"?>
<robot  xmlns:xacro="http://www.ros.org/wiki/xacro" name="mycobot_ai_with" >

  <xacro:property name="width" value=".2" />

  <link name="env">
  <inertial>
    <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="10"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
  </inertial>
    <visual>
      <geometry>
         <!--- 0.0 0 -0.04  1.5708 3.14159-->
       <mesh filename="package://mycobot_description/urdf/mycobot/suit_env.dae"/>
      </geometry>
      <origin xyz = "0 0 0.0" rpy = "1.5708 0 -1.5708"/>
    </visual>
  </link>

  <link name="joint1">
    <inertial>
    <origin xyz="0 0 0.1" rpy="0 0 0"/>
      <mass value="0.2"/>
      <inertia
        ixx="1.0" ixy="0.0" ixz="0.0"
        iyy="1.0" iyz="0.0"
        izz="1.0"/>
  </inertial>
    <visual>
      <geometry>
         <!--- 0.0 0 -0.04  1.5708 3.14159-->
       <mesh filename="package://mycobot_description/urdf/mycobot/joint1.dae"/>
      </geometry>
      <origin xyz = "0.0 0 0.02 " rpy = " 0 0 -1.5708"/>
    </visual>
    <collision>
      <geometry>
         <!--- 0.0 0 -0.04  1.5708 3.14159-->
       <mesh filename="package://mycobot_description/urdf/mycobot/joint1.dae"/>
        </geometry>
        <origin xyz = "0.0 0 0.02 " rpy = " 0 0 -1.5708"/>
    </collision>
  </link>

  <joint name="vision_joint" type="fixed">
    <axis xyz="0 0 0"/>
```

```
    <limit effort = "1000.0" lower = "-3.14" upper = "3.14159" velocity = "0"/>
    <parent link="env"/>
    <child link="joint1"/>
    <origin xyz= "0 0 0" rpy = "0 0 0"/>
  </joint>

<link name="world"/>
<joint name="fixed" type="fixed">
  <parent link="world"/>
  <child link="env"/>
</joint>

</robot>
```

It can be seen that the urdf file is not complicated, it is mainly composed of two parts, `link` and `joint`, which are repeated continuously.

3.2 link section

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

3.2.1 Attributes

name： The name used to describe the link itself

3.2.2 element

- `<inertial>` (optional)
  - Inertia properties of connecting rods
  - `<origin>` (optional，defaults to identity if not specified)
    - Defines the reference coordinate of the inertial reference system relative to the connecting rod coordinate system. The coordinate must be defined at the center of gravity of the connecting rod, and its coordinate axis may not be parallel to the main axis of inertia.
    - xyz (optional, defaults to zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
    - rpy(optional: defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
  - `<mass>` Mass properties of connecting rods
  - `<inertia>` 3×3 rotational inertia matrix, consisting of six independent quantities: ixx, ixy, ixz, iyy, iyz, izz。
- `<visual>` (optional)
  - Visual properties of the connecting rod. It is used to specify the shape of the link display (rectangle, cylinder, etc.). There can be multiple visual elements in the same link, and the shape of the link is formed by two elements. In general, the model is more complex and can be drawn through soildwork to generate stl calls, and simple shapes such as adding end effectors can be directly written. At the same time, the position of the geometry can be adjusted according to the gap between the theoretical model and the actual model.
  - `<namel>` (optional) The name of the connecting rod geometry.
  - `<origin>` (optional，defaults to identity if not specified)
    - The geometry coordinate system relative to the coordinate system of the connecting rod.

        - xyz (optional: defaults to zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
        - rpy (optional: defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.

- `<geometry>` （required）
    - The shape of the visualization, which can be one of the following:
    - `<box>` A rectangle with elements including length, width, and height. The origin is in the center.
    - `<cylinder>` Cylinder, elements include radius and length. center of origin.
    - `<sphere>` Sphere, element containing the radius. The origin is in the center.
    - `<mesh>` The grid, as determined by the file, also provides a scale to define its boundaries. Collada .dae files are recommended, .stl files are also supported, but must be a local file.
- `<material>` (optional)
    - Visualize the component's material. It can be defined outside the link tag, but it must be inside the robot tag. When defining outside the link tag, the name of the link must be quoted.
    - `<color>` (optional) Color, consisting of red/green/blue/alpha, in the range [0,1].
    - `<texture>` (optional) Material properties, defined by the file.
- `<collision>` (optional)
    - Collision properties of the link. Collision properties differ from visual properties of connecting rods, and simple collision models are often used to simplify calculations. The same link can have multiple collision attribute labels, and the collision attribute representation of the link is composed of the set of geometric shapes defined by it.
    - `<name>` (optional) Specifies the name of the connecting rod geometry
    - `<origin>` (optional，defaults to identity if not specified)
        - The reference coordinate system of the collision component is relative to the reference coordinate system of the link coordinate system.
        - xyz (optional, default zero vector) Represents the offset in the x , y , z x,y,zx,y,z directions, in meters.
        - rpy (optional, defaults to identity if not specified) Indicates the rotation of the coordinate axis in the RPY direction, in radians.
    - `<geometry>` Same as the geometry element description above

Detailed elements and the role of each element can go to official documentation to view

3.3 joint part

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

3.3.1 properties of joint:

name：

Specifies a unique name for the joint

type：

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

- revolute - A hinged joint that rotates along an axis, the range of which is specified by the upper and lower bounds.
- Continuous - A continuous hinged joint that rotates around an axis with no upper and lower bounds.
- Prismatic - A sliding joint that slides along an axis, the range of which is specified by upper and lower limits. +Fixed - this is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint

does not require axes, calibration, dynamics, limits or safety_controller。
- Floating - This joint allows motion in all 6 degrees of freedom.
- Plane - This joint allows movement in a plane perpendicular to the axis.

3.3.2 elements of joint

- `<origin>` (optional，defaults to identity if not specified) In the transformation from parent link to child link, the joint is located at the origin of the child link. Modifying this parameter can adjust the position of the connecting rod. It can be used to adjust the error between the actual model and the theoretical model, but it is not recommended to modify it greatly, because this parameter affects the connecting rod stl The position of , easily affects the collision detection effect.
  - xyz (optional: default to zero vector) Represents the offset in the x , y , z x,y,zx,y,z axis directions, in meters.
  - rpy (optional: default to zero vector) Represents the angle of rotation around a fixed axis: roll is around the x-axis, pitch is around the y-axis, and yaw is around the z-axis, expressed in radians.
- `<parent>` (required)
  - The name of the parent link is a mandatory attribute.
  - link The name of the parent link is the name of the link in the robot structure tree.
- `<child>` (required)
  - The name of the child link is a mandatory attribute.
  - link The name of the child link is the name of the link in the robot structure tree.
- `<axis>` (optional: defaults to (1,0,0))
  - The joint's axis is in the joint's coordinate system. This is the axis of rotation (revolute joint), the axis of movement of the prismatic joint, and the standard plane of the planar joint. This axis is specified in the joint coordinate system. Modifying this parameter can adjust the axis around which the joint rotates. It is often used to adjust the rotation direction. If the model rotation is opposite to the actual one, just multiply by -1. Fixed and floating joints do not need this element.
  - xyz(required) x , y , z x, y, zx, y, z components representing axis vectors, as normalized vectors.
- `<calibration>` (optional)
  - The reference point of the joint, used to correct the absolute position of the joint.
  - rising (optional) When the joint is moving forward, the reference point triggers a rising edge.
  - falling (optional) When the joint is moving forward, the reference point triggers a falling edge.
- `<dynamics>` (optional)
  - This element is used to specify the physical properties of the joint. Its value is used to describe the modeling performance of the joint, especially during simulation.

`<limit>` (Required when the joint is a rotation or translation joint)

- This element is a joint kinematics constraint.
- lower (optional, default to 0) Specify the attribute of the lower bound of the joint's motion range (the unit of the revolute joint is radians, and the unit of the prismatic joint is meters). This attribute is ignored for continuous joints.
- upper (optional, defaults to 0) Specify the attribute of the upper bound of the joint's motion range (the unit of the revolute joint is radians, and the unit of the prismatic joint is the meter). This attribute is ignored for continuous joints.
- effort (required) This property specifies the maximum force at which the joint will run.
- velocity (required) This property specifies the maximum speed of the joint runtime.

`<mimic>` (optional)

- This tag is used to specify a defined joint to mimic an existing joint. The value of this joint can be calculated using the following formula: `value = multiplier * other_joint_value + offset`
- joint(required) The name of the joint to mimic.
- multiplier(optional) Specify the multiplier factor in the above formula.
- offset(optional) Specify the offset term in the above formula. Default value is 0

`<safety_controller>` (optional)

- This element is a security control limit. The data under this element will be read into move_group, but it is invalid in practice. Move_group will skip this limit and directly read the parameter content under limit. At the same time, setting this element may cause planning failure.
- soft_lower_limit (optional, defaults to 0) This attribute specifies the lower bound of the joint security control boundary, which is the starting limit point of the joint security control. This value needs to be greater than the lower value in the above limit.
- soft_upper_limit (optional, defaults to 0) This attribute specifies the upper bound of the joint security control boundary, which is the starting limit point of the joint security control. This value needs to be less than the upper value in the above limit.
- k_position(optional, defaults to 0) This attribute is used to describe the relationship between position and velocity.
- k_velocity(required) This property is used to describe the relationship between force and velocity.

Detailed elements and the role of each element can go to http://wiki.ros.org/urdf/XML/joint to view.

Brief introduction and use of rviz

rviz is a 3D visualization platform in ROS. On one hand, it can realize the graphical display of external information, and on the other hand, it can also release control information to an object through rviz, realizing the monitoring and control of a robot.

1 Installation of rviz and the introduction to its interface

When installing ros, if you perform a complete installation, rviz is already installed, and you may try to run it directly; if it is not fully installed, you may install rviz separately:

```
# Ubuntu16.04
sudo apt-get install ros-kinetic-rviz
```

```
# Ubuntu18.04
sudo apt-get install ros-melodic-rviz
```

```
# Ubuntu20.04
sudo apt-get install ros-noetic-rviz
```

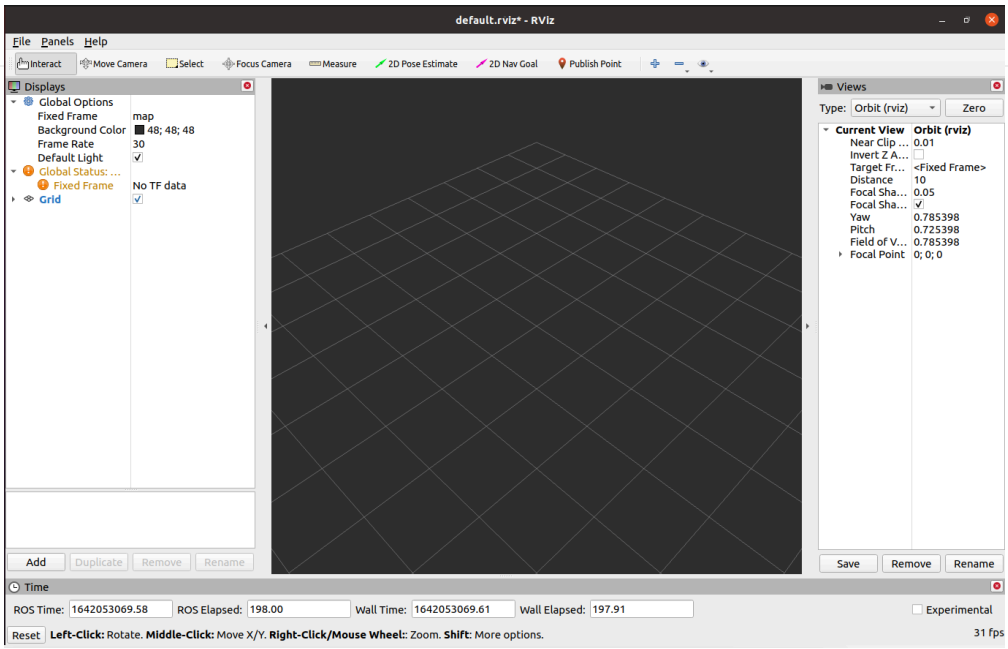
After the installation is complete, open a new terminal (shortcut key: `Ctrl+Alt+T`) and enter the following command:

```
roscore
```

Then open a new terminal (shortcut key: `Ctrl+Alt+T`) and input the following command to open rviz.

```
rosrun rviz rviz
# or
rviz
```

Open rviz, and the following interface will be displayed:



1.1 Introduction of all areas

- There is a list of monitors on the left. The monitor is a device that draws something in a 3D world and may have some options available in the display list.
- On the top is a toolbar, which allows the user to use various function buttons to select tools with multiple functions.
- The middle part is the 3D view: It is a main screen where various data can be viewed in three dimensions. The background color, fixed frame, grid, etc. of the 3D view can be set in detail in the Global Options and Grid items displayed on the left.
- Below is the time display area, including system time and ROS time.
- The right side is the observation angle setting area where different observation angles can be set.

We only give a rough introduction in this part. If you want to know more details, go to User Guide.

2 mycobot_ros installation and update

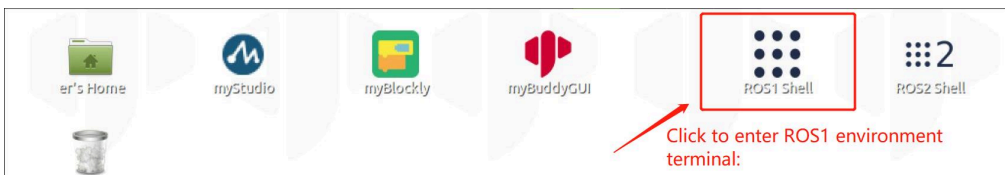
- **M5 Version:** Please refer to the end of the 13.1.1 Environment Building chapter.
- **PI版本(Ubuntu 20.04)：**

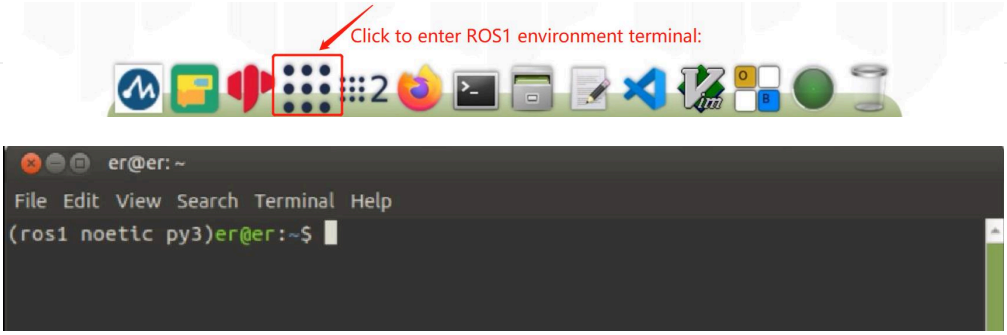
`mycobot_ros` is a ROS package from ElephantRobotics that works with all types of desktop robots.

The address of the project: https://github.com/elephantrobotics/mycobot_ros

The official default ROS1 workspace is `catkin_ws`.

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 ~/catkin_ws/src  # Enter the src folder of the workspace
# Clone the code on github
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd ..       # return to the workspace
catkin_make # Build the code in the workspace
source devel/setup.bash # add environment variables
```

Note: If the `/home/er/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, `er` is the user name of the VM. If they are different, change them.

3 Simple use

Start using launch file

This example is built on what you have already done Environment building and you have successfully copied the company's code from GitHub to your virtual machine.

Open a new terminal (shortcut key: `Ctrl+Alt+T`)

Input the command to **configure the ROS environment**.

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

Input again:

- mycobot 280-M5 version:

```
roslaunch mycobot_280 test.launch
```

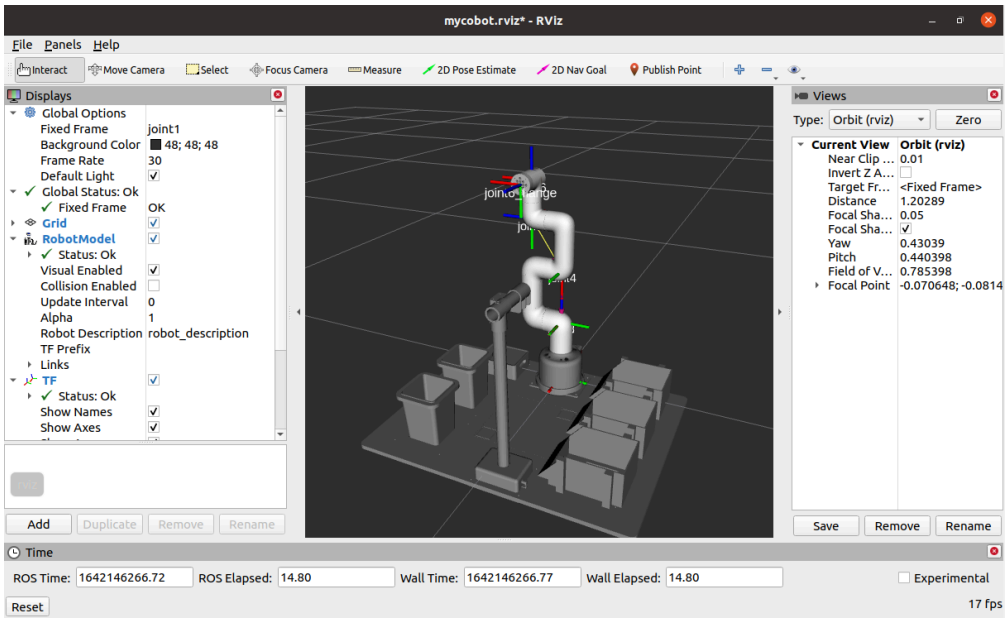
- mycobot 280-Pi version:

```
roslaunch mycobot_280pi test.launch
```

- mycobot 280-JetsonNano version:

```
roslaunch mycobot_280jn test.launch
```

Open rviz, and then you will obtain the following result:



If you want to know more information about rviz, go to Official documents.

4 M5 Version Prerequisites

- Type Ctrl+Shift+T in the command terminal to open another terminal window in the same directory to view the device name:

```
# View the device name of the robotic arm
ls /dev/ttyUSB* # old version myCobot280 M5


# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # new version myCobot280 M5
```

- Grant the serial port permission to the robotic arm:

```
# The default device name is /dev/ttyUSB0, if the device name is not the default value, it needs to be modified.
sudo chmod 777 /dev/ttyUSB0 # old version myCobot280 M5


sudo chmod 777 /dev/ttyACM0 # new version myCobot280 M5
```

Then enter the user password(**Note:** The password is not displayed, just enter it correctly).

Control and following of the robot arm

1 Slider Control

Open a command line and run:

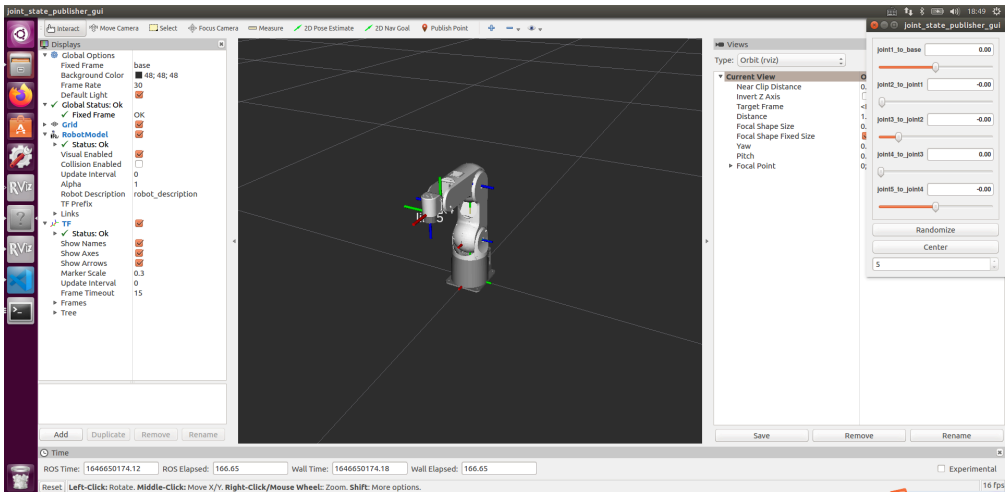
- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
roslaunch mypalletizer_260 slider_control.launch port:=/dev/ttyUSB0 baud:=115200
```

- mypalletzier 260-Pi version:

```
# The default serial port name of the mechArm version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mypalletizer_260_pi slider_control.launch port:=/dev/ttyAMA0 baud:=1000000
```

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 mycobot to move with the model, you need to open another command line and run:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port
rosrun mypalletizer_260 slider_control.py _port:=/dev/ttyUSB0 _baud:=115200
```

- mypalletzier 260-Pi version:

```
# The default serial port name of the mechArm version is "/dev/ttyAMA0", and the baud rate is 1000000".
rosrun mypalletizer_260_pi slider_control.py _port:=/dev/ttyAMA0 _baud:=1000000
```

Note: Since the robot arm will move to the current position of the model when the command is input, make sure that the model in rviz does not appear to be worn out before you use the command.

Do not drag the slider quickly after connecting the robot arm to prevent damage to the robot arm.

2 Model Following

In addition to the above controls, we can also let the model move by following the real robot arm. Open a command line and run:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port
rosrun mypalletizer_260 follow_display.py _port:=/dev/ttyUSB0 _baud:=115200
```

- mypalletzier 260-Pi version:

```
# The default serial port name of the mechArm version is "/dev/ttyAMA0", and the baud rate is 1000000".
rosrun mypalletizer_260_pi follow_display.py _port:=/dev/ttyAMA0 _baud:=1000000
```

Then open another command line and run:

- mypalletizer 260-M5 version:

```
roslaunch mypalletizer_260 follow.launch
```

- mypalletzier 260-Pi version:

```
roslaunch mypalletizer_260_pi follow.launch
```

It will **open rviz to show the model following effect.**

3 GUI control

On the basis of the previous contents, this package also **provides a simple GUI control interface.** This method is used for interaction between real robot arms. Connect to mycobot.

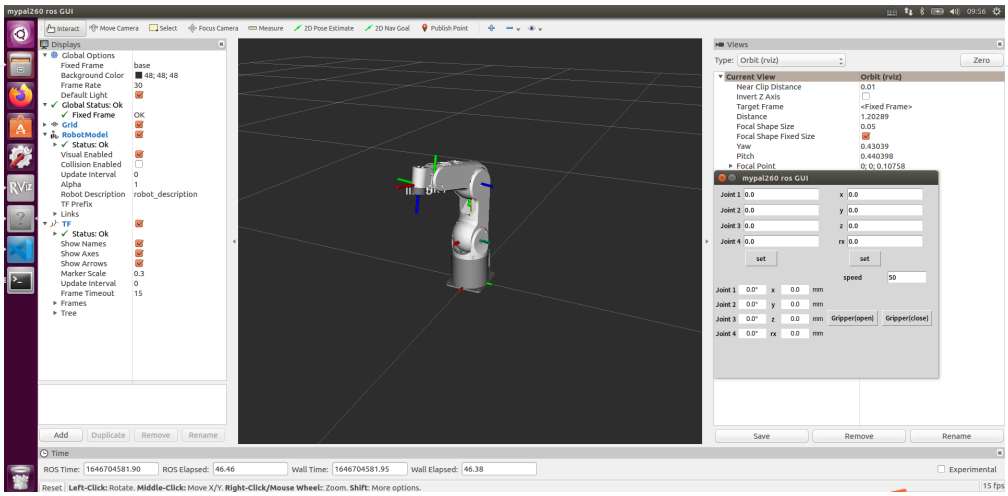
Open a command line:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port
roslaunch mypalletizer_260 simple_gui.launch port:=/dev/ttyUSB0 baud:=115200
```

- mypalletzier 260-Pi version:

```
# The default serial port name of the mechArm version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mypalletizer_260_pi simple_gui.launch port:=/dev/ttyAMA0 baud:=1000000
```

Running effect: 

4 Keyboard control

Keyboard control is added in `mypalletizer_260` package, and real-time Synchronization is performed in rviz. This function depends on pythonApi, so be sure to connect with the real robot arm.

Open a command line and run:

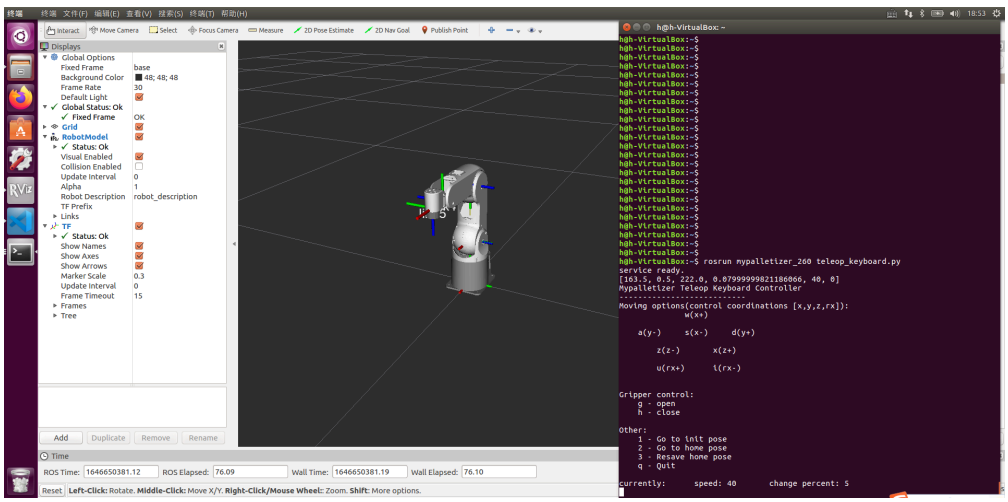
- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial port
roslaunch mypalletizer_260 teleop_keyboard.launch port:=/dev/ttyUSB0 baud:=115200
```

- mypalletzier 260-Pi version:

```
# The default serial port name of the mechArm version is "/dev/ttyAMA0", and the baud rate is 1000000".
roslaunch mypalletizer_260_pi teleop_keyboard.launch port:=/dev/ttyAMA0 baud:=1000000
```

Running effect is as follows:



mypalletizer information will be output in the command line as follows:

```
SUMMARY
========

PARAMETERS
 * /mypalletizer_services/baud: 125200
 * /mypalletizer_services/port: /dev/ttyUSB0
 * /robot_description: <?xml version="1....
 * /rosdistro: kinetic
 * /rosversion: 1.12.17

NODES
  /
    mypalletizer_services (mypalletizer_communication/mypal_services.py)
    real_listener (mypalletizer_260/listen_real.py)
    robot_state_publisher (robot_state_publisher/robot_state_publisher)
    rviz (rviz/rviz)

ROS_MASTER_URI=http://localhost:12312

process[robot_state_publisher-1]: started with pid [14764]
process[rviz-2]: started with pid [14765]
process[mypalletizer_services-3]: started with pid [14766]
process[real_listener-4]: started with pid [14782]
[INFO] [1646649869.148017]: start ...
[INFO] [1646649869.156531]: /dev/ttyUSB0,125200

Mypalletizer Status
-------------------------------
Joint Limit:
    joint 1: -160 ~ +160
    joint 2: 0 ~ +90
    joint 3: 0 ~ +60
    joint 4: -180 ~ +180


Connect Status: True

Servo Infomation: unknown

Servo Temperature: unknown

Atom Version: unknown

[INFO] [1646649869.427899]: ready
```

Then open another command line and run:

- mypalletizer 260-M5 version:

```
rosrun mypalletizer_260 teleop_keyboard.py
# or
rosrun mypalletizer_260 teleop_keyboard.py _speed:=70
```

- mypalletizer 260-Pi version:

```
rosrun mypalletizer_260_pi teleop_keyboard.py
# or
rosrun mypalletizer_260 teleop_keyboard.py _speep:70
```

You will see the following output in the command line:

```
Mypalletizer Teleop Keyboard Controller
---------------------------
Movimg options(control coordinations [x,y,z,rx]):
              w(x+)

    a(y-)     s(x-)     d(y+)

        z(z-)       x(z+)

        u(rx+)      i(rx-)



Gripper control:
    g - open
    h - close

Other:
    1 - Go to init pose
    2 - Go to home pose
    3 - Resave home pose
    q - Quit

currently:    speed: 40    change percent: 5
```

In this terminal, you can control the state of the robot arm and move it using the keys in the command line.

Parameters supported by this script:

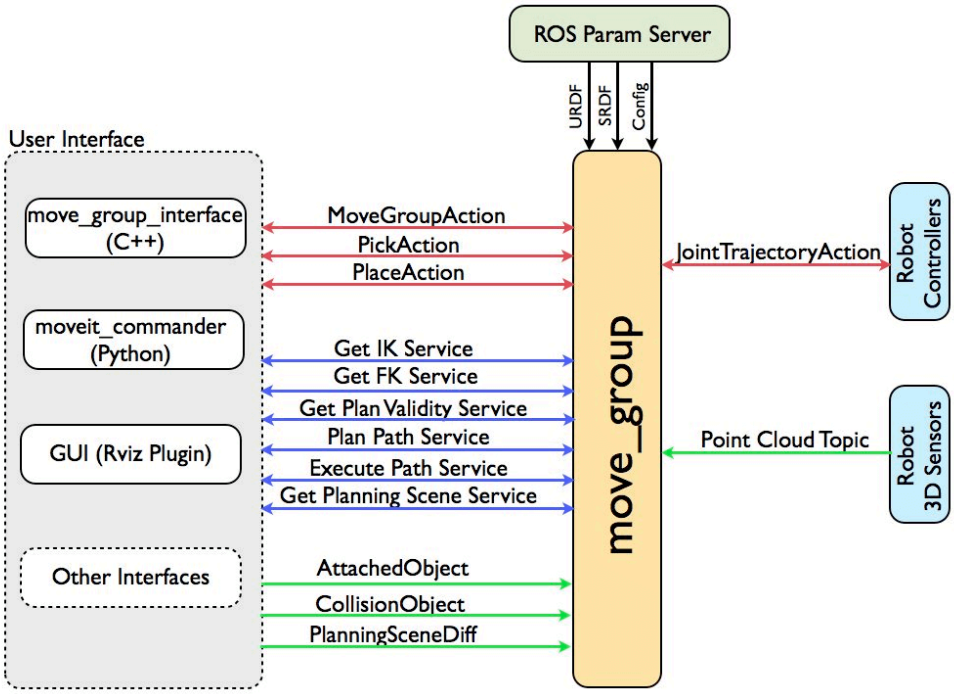
- _speed: the movement speed of the robot arm
- _change_percent: movement distance percentage

MoveIt

1 Introduction to MoveIt

MoveIt is an integrated development platform in ROS, which consists of a variety of functional packages for manipulating robot arms, including motion planning, operation, control, inverse kinematics, 3D perception, collision detection, etc.

The following figure shows the high-level structure of the main node **move_group** provided by Moveit. It is like a combiner: all the individual components are integrated together, providing a series of actions and services for users to use.



2 User interface

The user may access the operations and services provided by move_group in three ways:

- In C++, you may use move_group easily by using move_group_interface package.
- In Python, use the moveit_commander package.
- Via GUI: use Rviz (ROS visualization tool) of Motion-commander.

move_group can be configured using the ROS parameter server, from which the robot's URDF and SRDF can also be obtained.

3 Configuration

move_group is a ROS node. It uses the ROS parameter server to obtain three kinds of information:

- URDF - move_group looks for the robot_description parameter in the ROS parameter server to get the robot's URDF.
- SRDF - move_group looks for the robot_description_semantic parameter in the ROS parameter server to get the robot's SRDF. SRDF is typically created by the user using an MoveIt Setup Assistant.
- MoveIt configuration - move_group will look in the ROS parameter server for additional MoveIt-specific configurations, including joint constraint, kinematics, motion planning, perception, and other information. w The

configuration files for these components are automatically generated by the MoveIt Setup Assistant and stored in the configuration directory of the robot's corresponding MoveIt configuration package.

Mypalletizer 260 Moveit

`mypalletizer_260` has integrated the MoveIt section.

Open the command line and run:

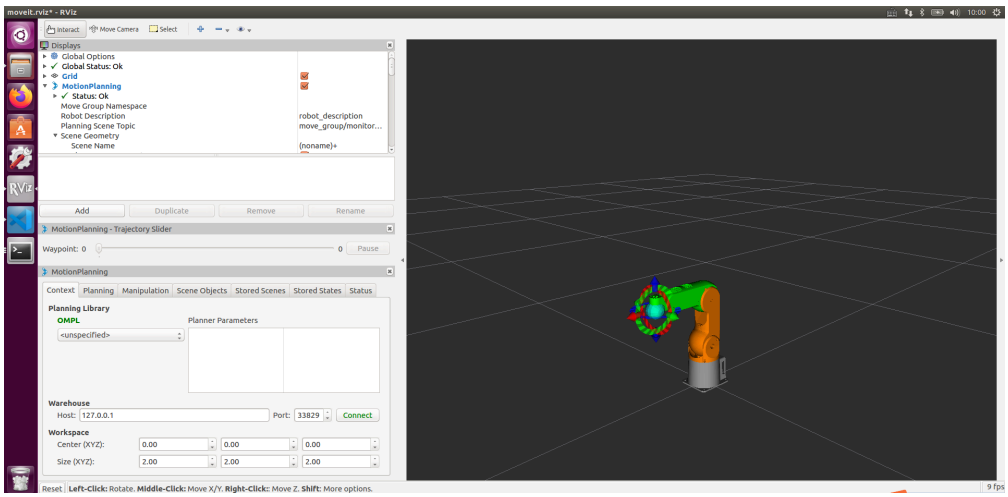
- mypalletizer 260-M5 version:

```
roslaunch mypalletizer_260_moveit mypal_moveit.launch
```

- mypalletizer 260-Pi version:

```
roslaunch mypalletizer_260_pi_moveit mypal_moveit.launch
```

The operation effect is as follows:



If you want a real robot arm to execute a plan synchronously, you need to open another command line and run:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
rosrun mypalletizer_260_moveit sync_plan.py _port:=/dev/ttyUSB0 _baud:=115200
```

- mypalletizer 260-Pi version:

```
# The default serial port name of the mypalletizer 260-Pi version is "/dev/ttyAMA0", and the baud rate is 1000000".
rosrun mypalletizer_260_pi_moveit sync_plan.py _port:=/dev/ttyAMA0 _baud:=1000000
```

ROS2 introduction

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

1 Design goals and features of ROS2

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

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

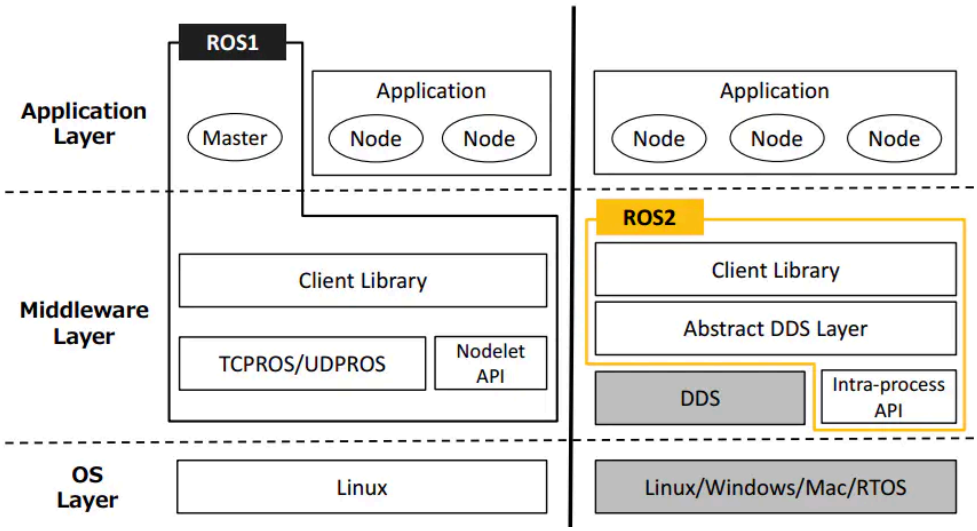
2 Release Version

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

ROS2 version	release date	Maintenance deadline	Ubuntu version
Dashing	2019.5	2021.5	Ubuntu 18.04 (Bionic Beaver)
Eloquent	2019.11	2020.11	Ubuntu 18.04 (Bionic Beaver)
Foxy	2020.6	2023.5	Ubuntu 20.04(Focal Fossa)
Galactic	2021.5	2022.11	Ubuntu 20.04(Focal Fossa)
Humble	2022.5	2027.5	Ubuntu 22.04(Jammy Jellyfish)

3 Comparison of ROS and ROS2

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



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

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

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

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

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

Environment Installation

M5 Version:

Notice: When installing the virtual machine system, please install the **Ubuntu 20.04 version of the system**, The installation method is consistent with ubuntu 18.04.

To install different versions of ubuntu system in Linux, please refer to 13.1.1 Environment Building chapter.

1 ROS Environment building

1.1 Installing ROS2

Building a basic development environment requires the installation of the robot operating system (short for ROS2) and a git version manager. The installation methods and procedures are described respectively below.

For **myCobot 280-M5**、**myCobot 320-M5 devices**、**myPalletizer 260-M5** and **mechArm 270-M5**, refer to the installation methods and procedures described below. For **myCobot 280-PI**、**myCobot 320-PI**、**myPalletizer 260-PI** and **mechArm 270-PI** devices, you only need to install the mycobot_ros installation package.

1.1.1 Selecting a version

There is a one-to-one relationship between ROS2 and ubuntu. Different versions of ubuntu correspond to different versions of ROS2. The reference website is as follows: http://docs.ros.org/en/foxy/Releases.html

Here are the ROS versions supported by Ubuntu:

ROS2 version	release date	Maintenance deadline	Ubuntu version
Foxy	June 5, 2020	May 2023	Ubuntu 20.04(Focal Fossa)
Galactic	May 23, 2021	November 2022	Ubuntu 20.04(Focal Fossa)
Humble	May 23, 2022	May 2027	Ubuntu 22.04(Jammy Jellyfish)

Install the ROS2 version corresponding to the Ubuntu version you have installed.

If the versions are different, the downloading will fail. The system we choose here is Ubuntu 20.04(recommend), so the corresponding ROS2 version is ROS2 Foxy.

> NOTE: At present, we do not provide any reference for installing ROS in windows. If necessary, refer to http://docs.ros.org/en/foxy/Installation/Alternatives/Windows-Development-Setup.html

1.1.2 Installing ROS

1 Adding a software source

There is no ROS2 software source in the software source list of Ubuntu itself, so you need to Configure the ROS software source into the software list repository first, and then download ROS2. Open a console terminal (shortcut key: Ctrl+Alt+T) and input the following command:

- official source:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.or
```

- If the download speed is slow, it is recommended to select a nearby mirror source to replace the above command. For example, huawei cloud is:

```
echo "deb [arch=$(dpkg --print-architecture)] https://repo.huaweicloud.com/ros2/ubuntu/ $(lsb_release -cs) main" | sudo te
```

2 Setting a key

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

```
sudo apt install curl gnupg2 -y
curl -s https://gitee.com/ohhuo/rosdistro/raw/master/ros.asc | sudo apt-key add -
```

3 Installation

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

```
sudo apt-get update
```

Execute **Installing ROS2**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command according to your Ubuntu version:

```
# Ubuntu 20.04 foxy version
sudo apt install ros-foxy-desktop
```

```
# Ubuntu 20.04 galactic version
sudo apt install ros-galactic-desktop
```

```
# Ubuntu 22.04 humble version
sudo apt install ros-humble-desktop
```

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

Refresh environment variables after installation is complete:

```
source /opt/ros/foxy/setup.bash
```

1.1.3 Set up the ros2 environment

In order to avoid re-validating the ROS2 function path every time the terminal window is closed, we can **set the path to an environment variable**, so that each time you open a new terminal, the ROS2 function path will automatically take effect. Execute the following commands in sequence in the terminal, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

Execute the following commands:

```
# Ubuntu 20.04 foxy version
# Add the ros2 environment to the environment variables of the current console
echo "source /opt/ros/foxy/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 20.04 galactic version
echo "source /opt/ros/galactic/setup.bash" >> ~/.bashrc
```

```
# Ubuntu 22.04 humble version
echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Installing a ROS2 extra dependency

Input the following command in the terminal to install a ROS2 extra dependency, and open a console terminal (shortcut key: `Ctrl+Alt+T`):

```
sudo apt install python3-argcomplete -y
```

```
sudo apt install ros-foxy-xacro
```

```
sudo apt-get install python3-colcon-common-extensions
```

```
# Ubuntu 20.04 foxy version
sudo apt install ros-foxy-joint-state-publisher-gui
```

```
# Ubuntu 20.04 galactic version
sudo apt install ros-galactic-joint-state-publisher-gui
```

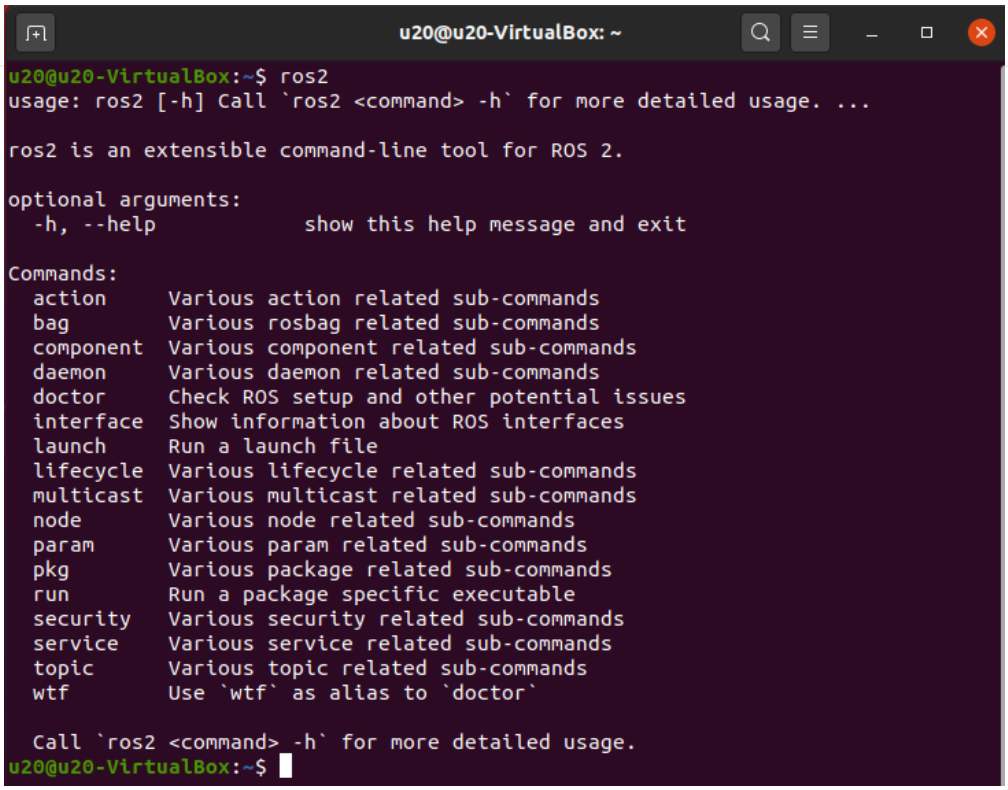
```
# Ubuntu 22.04 humble version
sudo apt install ros-humble-joint-state-publisher-gui
```

1.1.4 Verify installation

To verify whether ROS2 has been installed successfully, open a console terminal (shortcut key: `Ctrl+Alt+T`) and execute the following command in the terminal:

```
ros2
```

When the following interface is displayed, it means that ROS2 has been installed successfully.



2 git Installation

2.1 Updating the software source list

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

```
sudo apt-get update
```

2.2 Installing git

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

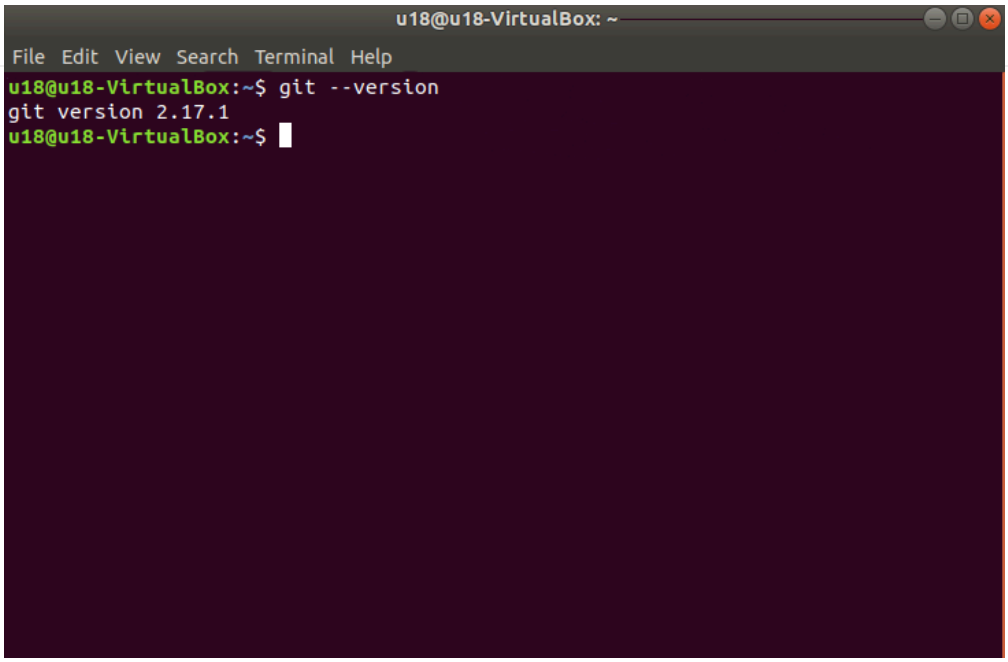
```
sudo apt-get install git
```

2.3 Verify installation

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

```
git --version
```

The git version number can be displayed in the terminal as follows, that is, the installation is successful.



3 mycobot_ros2 Installation

`mycobot_ros2` is a ROS2 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros2

3.1 Precondition

Before installing the package, make sure you have a ros2 workspace.

Here is an **example command for creating a workspace**. Open a console terminal (shortcut key: `Ctrl+Alt+T`) and input the following command in the command line:

```
# Create a folder
mkdir -p ~/colcon_ws/src
# compile workspace
cd ~/colcon_ws
colcon build
```

Add workspace environment

The official default ROS2 workspace is `colcon_ws`.

```
echo "source ~/colcon_ws/install/setup.bash" >> ~/.bashrc
```

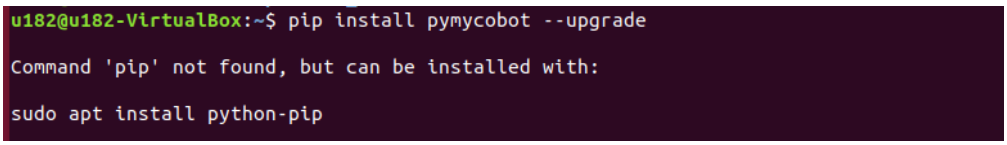
```
source ~/.bashrc
```

3.2 Installation

NOTE:

- This package depends on ROS2 and MoveIT2. Before using it, make sure that you have installed ROS2 and MoveIT2 successfully.
- The interaction of this package with a real robot arm depends on PythonApi - `pymycobot`.
- The Api project location is: https://github.com/elephantrobotics/pymycobot
- Quick installation: `pip install pymycobot --upgrade`

  When executing the pip install pymycobot --upgrade command, if the following error message appears:

  Enter the following command to install pip at the prompt

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

- If your Ubuntu system is version 20.04, please execute the command `sudo apt install python3-pip` to install pip

  After the pip installation is complete, the terminal executes again

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

- The installation method depends on Git, so make sure that Git have been installed in your computer.

The official default ROS2 workspace is `colcon_ws`.

```
cd ~/colcon_ws/src  # Enter the src folder of the workspace
# Clone the code on github
git clone --depth 1 https://github.com/elephantrobotics/mycobot_ros2.git
cd ..       # return to the workspace
colcon build --symlink-install # Build the code in the workspace, --symlink-install: Avoid having to recompile python scri
source install/setup.bash # add environment variables
```

PI Version

The Raspberry Pi version comes with an Ubuntu (V-20.04) system and a built-in development environment, so you don't need to build and manage it, Just update the mycobot_ros2 package.

`mycobot_ros2` is a ROS2 package launched by Elephant Robotics and applicable to its mycobot series of desktop six-axis robot arms.

Project address: http://github.com/elephantrobotics/mycobot_ros2

Robotic arm API driver library address: https://github.com/elephantrobotics/pymycobot

1 Update the mycobot_ros2 package

In order to ensure that users can use the latest official package in time, they can go to the `/home/er/colcon_ws/src` folder through the file manager and open the console terminal (shortcut `Ctrl+Alt+T`) , enter the following command to update:

```
# Clone the code on github
cd ~/colcon_ws/src
git clone https://github.com/elephantrobotics/mycobot_ros2.git # Please check the attention section below before deciding
cd ..      # Back to work area
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 `mycobot_ros2` folder already exists in the `/home/er/colcon_ws/src (equivalent to ~/colcon_ws/src)` directory, you need to delete the original `mycobot_ros2` before executing the above command. Among them, `er` in the directory path is the user name of the virtual machine. If it is inconsistent, please modify it.

So far, the ROS2 environment construction has been completed. Please refer to 13.2.3 Rviz Introduction and Use for the use of ROS2.

Basic tool

In this chapter, you will learn about the common command tools of ROS2.

1 Topics

ROS 2 breaks complex systems down into many modular nodes. Topics are a vital element of the ROS graph that act as a bus for nodes to exchange messages. Topics are one of the main ways in which data is moved between nodes and therefore between different parts of the system.

Specific reference: Official Tutorials

- **topics help**

```
ros2 topics -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **Node Relationship Diagram**

```
rqt_graph
```

- **Learn about topic-related commands**

```
ros2 topics -h
```

- **topics list**

```
ros2 topic list
ros2 topic list -t # Display the corresponding message type
```

- **View topic content**

```
ros2 topic echo <topic_name>
ros2 topic echo /turtle1/cmd_vel
```

- **Display topic-related information, type**

```
ros2 topic info <topic_name>
# Output /turtle1/cmd_vel topic related information
ros2 topic info /turtle1/cmd_vel
```

- **Display interface related information**

```
ros2 interface show <msg_type>
# Output geometry_msgs/msg/Twist interface related information
ros2 interface show geometry_msgs/msg/Twist
```

- **Issue an order**

```
ros2 topic pub <topic_name> <msg_type> '<args>'
# Issue speed command
ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.
# Issue speed commands at a certain frequency
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y:
```

- **See how often topics are posted**

```
ros2 topic hz <topic_name>
# Output /turtle1/cmd_vel publish frequency
ros2 topic pub --rate 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y:
```

2 Nodes

Each node in ROS should be responsible for a single, module purpose (e.g. one node for controlling wheel motors, one node for controlling a laser range-finder, etc). Each node can send and receive data to other nodes via topics, services, actions, or parameters. A full robotic system is comprised of many nodes working in concert. In ROS 2, a single executable (C++ program, Python program, etc.) can contain one or more nodes.

Specific reference: Official Tutorials

- **nodes help**

```
ros2 nodes -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View the node list**

```
ros2 node list
```

- **View Node Relationship Diagram**

```
rqt_graph
```

- **Remapping**

```
ros2 run turtlesim turtlesim_node --ros-args --remap __node:=my_turtle
ros2 node list
```

- **View node information**

```
ros2 node info <node_name>
ros2 node info /my_turtle
```

3 Services

Services are another method of communication for nodes in the ROS graph. Services are based on a call-and-response model, versus topics' publisher-subscriber model. While topics allow nodes to subscribe to data streams and get continual updates, services only provide data when they are specifically called by a client.

Specific reference: Official Tutorials

- **services help**

```
ros2 service -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View the service list**

```
ros2 service list
# Display service list and message type
ros2 service list -t
```

- **View the message types received by the service**

```
ros2 service type <service_name>
ros2 service type /clear
```

- **Find services that use a certain message type**

```
ros2 service find <type_name>
ros2 service find std_srvs/srv/Empty
```

- **View Service Message Type Definitions**

```
ros2 interface show <type_name>.srv
ros2 interface show std_srvs/srv/Empty.srv
```

- **Call the service command to clear the walking track**

```
ros2 service call <service_name> <service_type>
ros2 service call /clear std_srvs/srv/Empty
```

- **Spawn a new turtle**

```
ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: 'turtle2'}"
```

4 Parameters

A parameter is a configuration value of a node. You can think of parameters as node settings. A node can store parameters as integers, floats, booleans, strings, and lists. In ROS 2, each node maintains its own parameters. For more background on parameters, please see the concept document.

Specific reference: Official Tutorials

- **parameters help**

```
ros2 param -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- **View service list**

```
ros2 param list
```

- **Get the parameter value**

```
ros2 param get <node_name> <parameter_name>
ros2 param get /turtlesim background_g
```

- **Set parameter values**

```
ros2 param set <node_name> <parameter_name> <value>
ros2 param set /turtlesim background_r 150
```

- **Export parameter values**

```
ros2 param dump <node_name>
ros2 param dump /turtlesim
```

- **Import parameters independently**

```
ros2 param load <node_name> <parameter_file>
ros2 param load /turtlesim ./turtlesim.yaml
```

- **Start the node and import parameters at the same time**

```
ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>
ros2 run turtlesim turtlesim_node --ros-args --params-file ./turtlesim.yaml
```

5 Actions

Actions are one of the communication types in ROS 2 and are intended for long running tasks. They consist of three parts: a goal, feedback, and a result.

Actions are built on topics and services. Their functionality is similar to services, except actions are preemptable (you can cancel them while executing). They also provide steady feedback, as opposed to services which return a single response.

Actions use a client-server model, similar to the publisher-subscriber model (described in the topics tutorial). An “action client” node sends a goal to an “action server” node that acknowledges the goal and returns a stream of feedback and a result.

Specific reference: Official Tutorials

- **action help**

```
ros2 action -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

Press G|B|V|C|D|E|R|T to achieve rotation, press F to cancel

- **View the server and client of the node action**

```
ros2 node info /turtlesim
```

- **View action list**

```
ros2 action list
ros2 action list -t # show action type
```

- **view action info**

```
ros2 action info <action>
ros2 action info /turtle1/rotate_absolute
```

- **View action message content**

```
ros2 interface show turtlesim/action/RotateAbsolute
```

- **Send action target information**

```
ros2 action send_goal <action_name> <action_type>
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
# With feedback information
ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 0}" --feedback
```

6 RQt

RQt is a graphical user interface framework that implements various tools and interfaces in the form of plugins. One can run all the existing GUI tools as dockable windows within RQt! The tools can still run in a traditional standalone method, but RQt makes it easier to manage all the various windows in a single screen layout.

Specific reference: Official Tutorials

You can run any RQt tools/plugins easily by:

```
rqt
```

- **rqt help**

```
rqt -h
```

- **Start turtlesim and keyboard control**

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

- Action Type Browser: / Plugins -> Actions ->Action Type Browser
- parameter reconfiguration: / Plugins -> configuration ->Parameter Reconfigure
- Node grap: /Node Graph
- control steering: /Plugins -> Robot Tools -> Robot Steering
- service invocation: /Plugins -> Services -> Service Caller
- Service Type Browser: Plugins -> Services -> Service Type Browser
- message release: Plugins -> Topics -> Message Publisher
- Message Type Browser: Plugins -> Topics -> Message Type Browser
- topic list: Plugins -> Topics -> Topic Monitor
- draw a graph: Plugins -> Visualization -> Plot
- **View logs: rqt_console**

```
ros2 run rqt_console rqt_console
ros2 run turtlesim turtlesim_node
ros2 topic pub -r 1 /turtle1/cmd_vel geometry_msgs/msg/Twist "{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z
```

7 TF2

tf2 is the transform library, which lets the user keep track of multiple coordinate frames over time. tf2 maintains the relationship between coordinate frames in a tree structure buffered in time and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time.

Specific reference: Official Tutorials

Let's start by installing the demo package and its dependencies.

```
sudo apt-get install ros-foxy-turtle-tf2-py ros-foxy-tf2-tools ros-foxy-tf-transformations
```

- **follow**
- launch starts 2 little turtles, the first little turtle automatically follows the second one

```
ros2 launch turtle_tf2_py turtle_tf2_demo.launch.py
```

- Control the movement of the first little turtle through the keyboard

```
ros2 run turtlesim turtle_teleop_key
```

- **View TF tree**

```
ros2 run tf2_tools view_frames.py
evince frames.pdf
```

- **View the relationship between two coordinate systems**

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

- **View TF relationships on rviz**

```
ros2 run rviz2 rviz2 -d $(ros2 pkg prefix --share turtle_tf2_py)/rviz/turtle_rviz.rviz
```

8 URDF

URDF is the Unified Robot Description Format for specifying robot geometry and organization in ROS.

Specific reference: Official Tutorials

- **Complete syntax**

```
<robot>
    # describe:
    # Parameters: name=""
    #  Child node:
        <link>
            # Description:
            # Parameters: name=""
            # Child node:
                <visual>
                    # describe:
                    # Parameters:
                    # child nodes:
                        <geometry>
                            # description
                            # parameters
                            # Child node:
                                <cylinder />
                                    # Description:
                                    # Parameters:
                                        # length="0.6"
                                        # radius="0.2"
                                <box />
                                    # description
                                    # Parameters:size="0.6 0.1 0.2"
                                <mesh />
                                    #  Description
                                    #Parameters: filename="package://urdf_tutorial/meshes/l_finger_tip.dae"
                        <collision>
                            # Description: collision element, prioritized
                            # parameters
                            # child node
                                <geometry>
                        <inertial>
                            # description
                            # parameters
                            # Child nodes:
                                <mass />
                                    # description: mass
                                    # Parameters: value=10
                                <inertia />
                                    # Description: Inertia
                                    # Parameters: i+"Cartesian product of xyz" (9 in total)="0.4"
                        <origin />
                            # Description:
                            # Parameters:
                                # rpy="0 1.5 0"
                                # xyz="0 0 -0.3"
                        <material />
                            # Description
                            # Parameters: name="blue"
```

```
        <joint>
            # Description
            # Parameters:
                # name=""
                # type=""
                    # fixed
                    # prismatic
            # child node
                <parent />
                    # Description
                    # Parameters: link=""
                <child />
                    # Description:
                    # Parameters: link=""
                <origin />
                    # Description:
                    # Parameters: xyz="0 -0.2 0.25"
                <limit />
                    # Description
                    # Parameters:
                        # effort="1000.0"    maximum effort
                        # lower="-0.38"      Joint upper limit (radians)
                        # upper="0"          Joint lower limit (radians)
                        # velocity="0.5"     Maximum velocity
                <axis />
                    # Description:  Press ? axis rotation
                    # Parameters: xyz="0 0 1", along the Z axis
        <material>
            # Description:
            # Parameters: name="blue"
            # child node:
                <color />
                    # description:
                    # Parameters: rgba="0 0 0.8 1"
```

- **Install dependent libraries**

```
sudo apt install ros-foxy-joint-state-publisher-gui ros-foxy-joint-state-publisher
sudo apt install ros-foxy-xacro
```

- **Download the source code**

```
cd ~/dev_ws
git clone -b ros2 https://github.com/ros/urdf_tutorial.git src/urdf_tutorial
```

- **Compiling the source code**

```
colcon build --packages-select urdf_tutorial
```

- **Running the example**

```
ros2 launch urdf_tutorial display.launch.py model:=urdf/01-myfirst.urdf
```

9 Launch

The launch system in ROS 2 is responsible for helping the user describe the configuration of their system and then execute it as described. The configuration of the system includes what programs to run, where to run them, what arguments to pass them, and ROS-specific conventions which make it easy to reuse components throughout the system by giving them each a different configuration. It is also responsible for monitoring the state of the processes launched, and reporting and/or reacting to changes in the state of those processes.

Launch files written in Python, XML, or YAML can start and stop different nodes as well as trigger and act on various events.

Specific reference: Official Tutorials

Setup

Create a new directory to store your launch files:

```
mkdir launch
```

Writer the launch file

Let’s put together a ROS 2 launch file using the turtlesim package and its executables. As mentioned above.

Copy and paste the complete code into the launch/turtlesim_mimic_launch.py file:

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    return LaunchDescription([
        Node(
            package='turtlesim',
            namespace='turtlesim1',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            namespace='turtlesim2',
            executable='turtlesim_node',
            name='sim'
        ),
        Node(
            package='turtlesim',
            executable='mimic',
            name='mimic',
            remappings=[
                ('/input/pose', '/turtlesim1/turtle1/pose'),
                ('/output/cmd_vel', '/turtlesim2/turtle1/cmd_vel'),
            ]
        )
    ])
```

Run the ros2 launch file

To run the launch file created above, enter into the directory you created earlier and run the following command:

The syntax format is:

```
ros2 launch <package_name> <launch_file_name>
```

```
cd launch
ros2 launch turtlesim_mimic_launch.py
```

- **launch help**

```
ros2 launch -h
```

- **running node**

```
ros2 launch turtlesim multisim.launch.py
```

- **Check the parameters of the launc file**

```
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py -s
ros2 launch turtlebot3_fake_node turtlebot3_fake_node.launch.py --show-arguments
ros2 launch turtlebot3_bringup robot.launch.launch.py -s
```

- **Run the launch file with parameters**

```
ros2 launch turtlebot3_bringup robot.launch.launch.py usb_port:=/dev/opencr
```

- **Run the node and debug**

```
ros2 launch turtlesim turtlesim_node.launch.py -d
```

- **Only output node description**

```
ros2 launch turtlesim turtlesim_node.launch.py -p
```

- **running components**

```
ros2 launch composition composition_demo.launch.py
```

10 Run

run is used to run a single node, component program

- **run help**

```
ros2 run -h
```

- **running node**

```
ros2 run turtlesim turtlesim_node
```

- **Run node with parameters**

```
ros2 run turtlesim turtlesim_node --ros-args -r __node:=turtle2 -r __ns:=/ns2
```

- **Run component container**

```
ros2 run rclcpp_components component_container
```

- **running components**

```
ros2 run composition manual_composition
```

11 Package

A package can be considered a container for your ROS 2 code. If you want to be able to install your code or share it with others, then you'll need it organized in a package. With packages, you can release your ROS 2 work and allow others to build and use it easily.

Package creation in ROS 2 uses ament as its build system and colcon as its build tool. You can create a package using either CMake or Python, which are officially supported, though other build types do exist.

Specific reference: Official Tutorials

Creating a workspace

Create a new directory for every new workspace. The name doesn't matter, but it is helpful to have it indicate the purpose of the workspace. Let's choose the directory name ros2_ws, for "development workspace":

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
```

- **pkg help**

```
ros2 pkg -h
```

- **List Feature Packs**

```
ros2 pkg executable turtlesim
```

- **Output a function package executable program**

```
ros2 pkg executable turtlesim
```

- **Create a Python package**

Make sure you are in the src folder before running the package creation command.

```
cd ~/ros2_ws/src
```

The command syntax for creating a new package in ROS 2 is:

```
ros2 pkg create --build-type ament_python <package_name>
# you will use the optional argument --node-name which creates a simple Hello World type executable in the package.
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

- **Build a package**

Putting packages in a workspace is especially valuable because you can build many packages at once by running colcon build in the workspace root. Otherwise, you would have to build each package individually.

```
# Return to the root of your workspace:
cd ~/ros2_ws
# Now you can build your packages:
colcon build
```

- **Source the setup file**

To use your new package and executable, first open a new terminal and source your main ROS 2 installation.

Then, from inside the ros2_ws directory, run the following command to source your workspace:

```
source install/setup.bash
```

Now that your workspace has been added to your path, you will be able to use your new package's executables.

- **Use the package**

To run the executable you created using the --node-name argument during package creation, enter the command:

```
ros2 run my_package my_node
```

Brief introduction and use of rviz2

rviz is a three-dimensional visualization platform in ROS. On the one hand, it can realize the graphical display of external information, and on the other hand, it can also release control information to the object through rviz, so as to realize the monitoring and control of the robot.

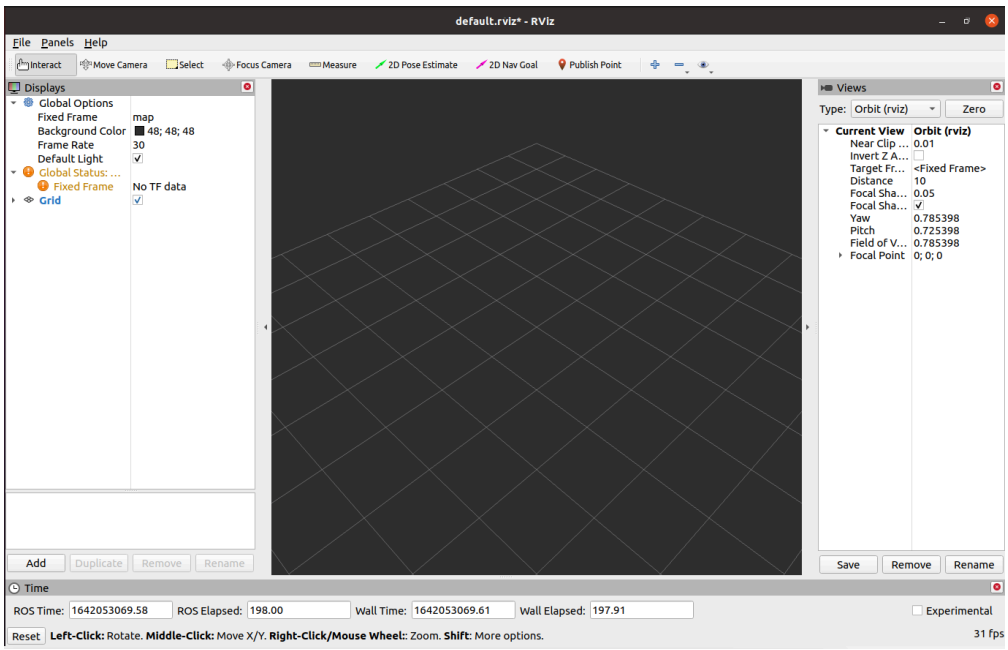
1 Introduction to rviz2

The successful installation of ros2 indicates that rviz2 is also successfully installed together, because the installation of ros2 includes rviz2.

Open a new terminal (shortcut`Ctrl+Alt+T`)enter the command to open rviz2

```
rviz2
```

Open rviz2 and display the following interface:



2 Introduction of each area

- On the left is the list of monitors, a monitor is something that draws something in the 3D world and may have some options available in the display list.
- Above is the toolbar, which allows users to use various function buttons to select tools with multiple functions
- The middle part is the 3D view: it is the main screen where various data can be viewed in 3D. The background color, fixed frame, grid, etc. of the 3D view can be set in detail in the Global Options and Grid items displayed on the left.
- Below is the time display area, including system time and ROS time.
- The right side is the observation angle setting area, and different observation angles can be set.

We only give a rough introduction in this part. If you want to know more detailed content, you can go to the user guide to view it.

3 mycobot_ros2 installation and update

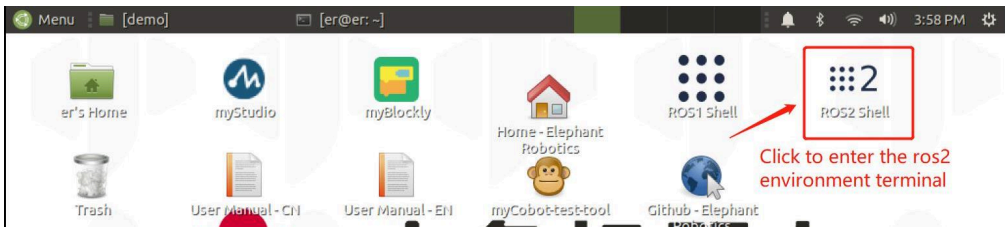
- **M5 verrsion:** Please refer to the end of the 13.2.1 Installation of ROS2 chapter.
- **PI版本(Ubuntu 20.04)：**

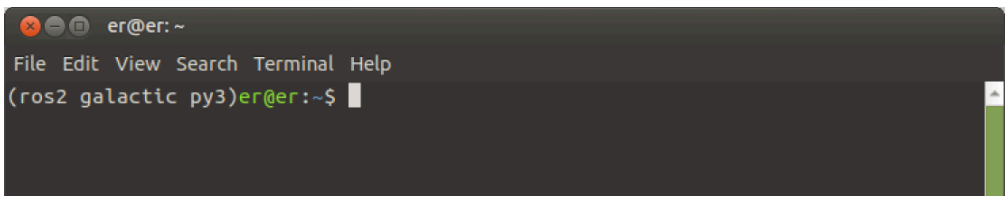
`mycobot_ros2` is a ROS2 package from ElephantRobotics that works with all types of desktop robots.

The address of the project: https://github.com/elephantrobotics/mycobot_ros2

The official default ROS2 workspace is `colcon_ws` .

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 ~/colcon_ws/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.

4 Simple use

Launch via launch.py file

This example is based on the fact that you have completed the environment setup and successfully copied the company's code from GitHub.。

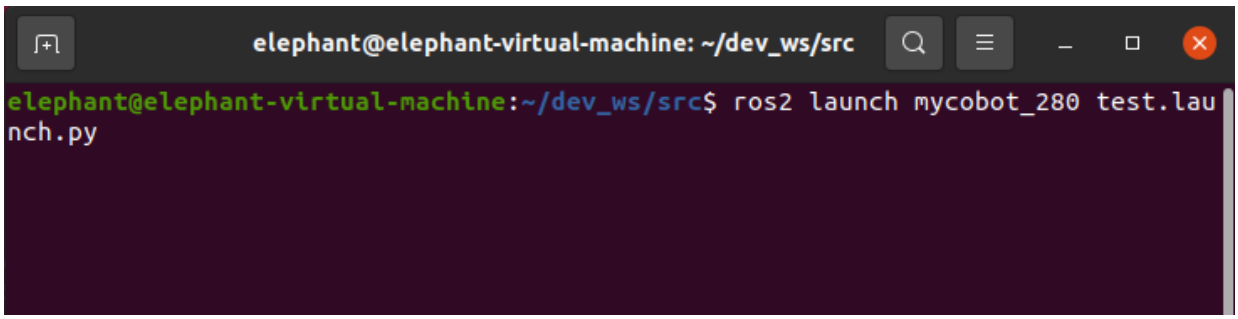
Open a console terminal (shortcut`Ctrl+Alt+T`) Enter the following command to **configure the ROS2 environment** .

```
cd ~/colcon_ws
colcon build --symlink-install
source install/setup.bash
```

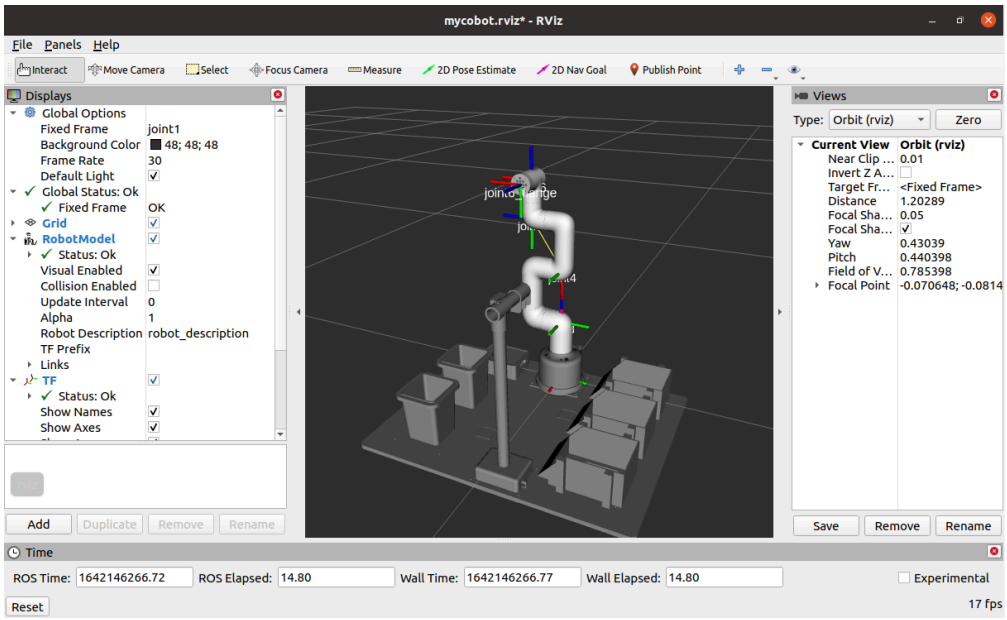
Enter again:

- mycobot 280-M5 version:

```
ros2 launch mycobot_280 test.launch.py
```



Open rviz and get the following result:



If you want to know more information about rviz, you can go to the official documentation to view it

5 M5 Version Prerequisites

- Type Ctrl+Shift+T in the command terminal to open another terminal window in the same directory to view the device name:

```
# View the device name of the robotic arm
ls /dev/ttyUSB* # old version myCobot280 M5

# If the terminal does not display the /dev/ttyUSB related name, you need to use the following command
ls /dev/ttyACM* # new version myCobot280 M5
```

- Grant the serial port permission to the robotic arm:

```
# The default device name is /dev/ttyUSB0, if the device name is not the default value, it needs to be modified.
sudo chmod 777 /dev/ttyUSB0 # old version myCobot280 M5

sudo chmod 777 /dev/ttyACM0 # new version myCobot280 M5
```

Then enter the user password(**Note:** The password is not displayed, just enter it correctly).

Control and following of the robot arm

1 Slider Control

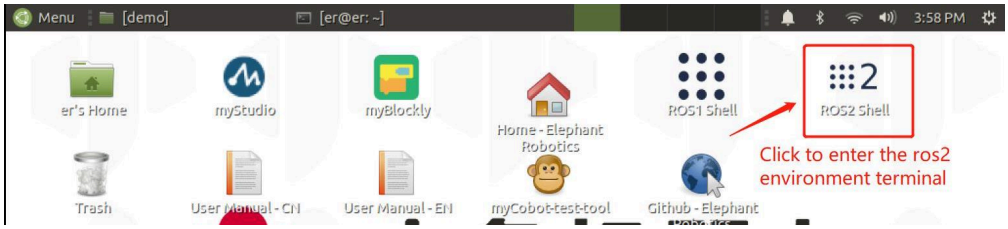
Open a command line and run:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
ros2 launch mypalletizer_260 slider_control.launch.py
```

- mypalletizer 260-PI version:

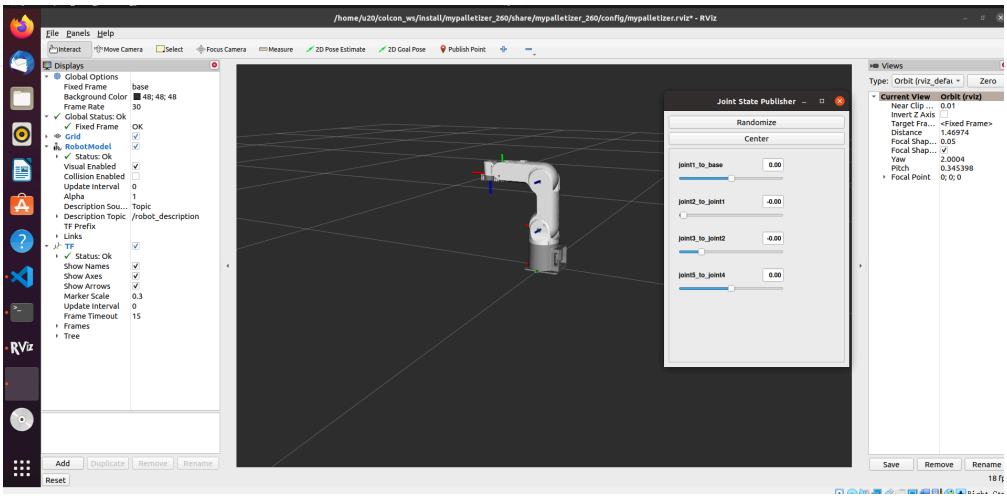
Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



Then run the command:

```
# The default serial port name of mypalletizer 260-PI version is "/dev/ttyAMA0", and the baud rate is 1000000.
ros2 launch mypalletizer_260_pi 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 mypalletizer 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 Following

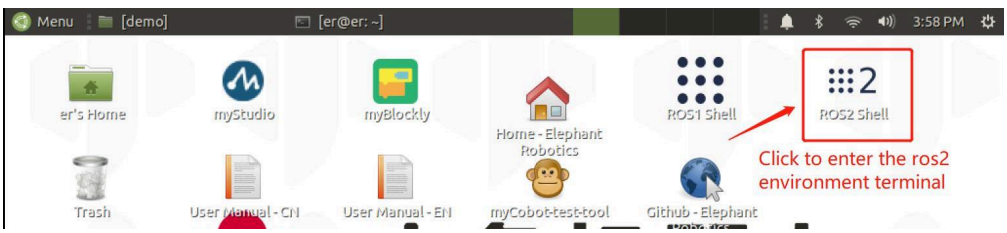
In addition to the above controls, we can also let the model move by following the real robot arm. Open a command line and run:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
ros2 launch mypalletizer_260 mycobot_follow.launch.py
```

- mypalletizer 260-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



Then run the command:

```
# The default serial port name of mypalletizer 260-PI version is "/dev/ttyAMA0", and the baud rate is 1000000
ros2 launch mypalletizer_260_pi mycobot_follow.launch.py
```

It will **open rviz to show the model following effect.**

3 GUI control

On the basis of the previous contents, this package also **provides a simple GUI control interface.** This method is used for interaction between real robot arms. Connect to mycobot.

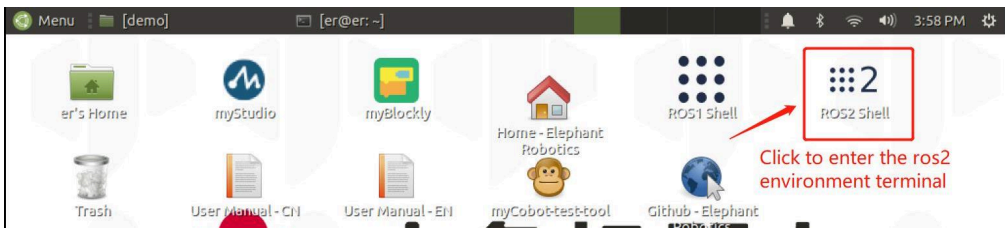
Open a command line:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
ros2 launch mypalletizer_260 simple_gui.launch.py
```

- mypalletizer 260-PI version:

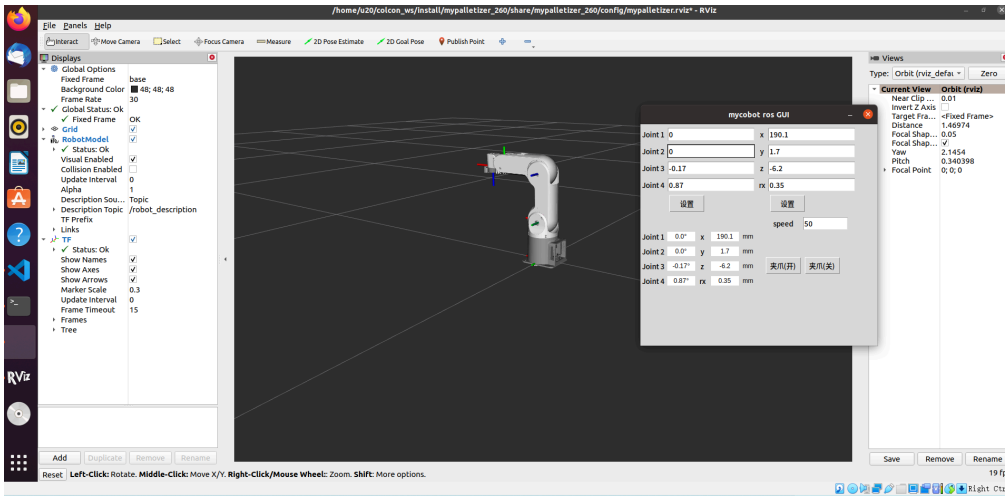
Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



Then run the command:

```
# The default serial port name of mypalletizer 260-PI version is "/dev/ttyAMA0", and the baud rate is 1000000.
ros2 launch mypalletizer_260_pi simple_gui.launch.py
```

Running effect:



4 Keyboard control

Keyboard control is added in `mypalletizer_260` package, and real-time Synchronization is performed in rviz. This function depends on pythonApi, so be sure to connect with the real robot arm.

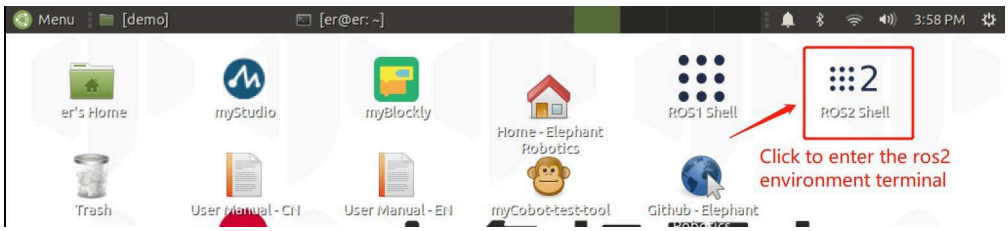
Open a command line and run:

- mypalletizer 260-M5 version:

```
# The default serial port name of mypalletizer 260-M5 version is "/dev/ttyUSB0", and the baud rate is 115200. The serial p
ros2 launch mypalletizer_260 teleop_keyboard.launch.py
```

- mypalletizer 260-PI version:

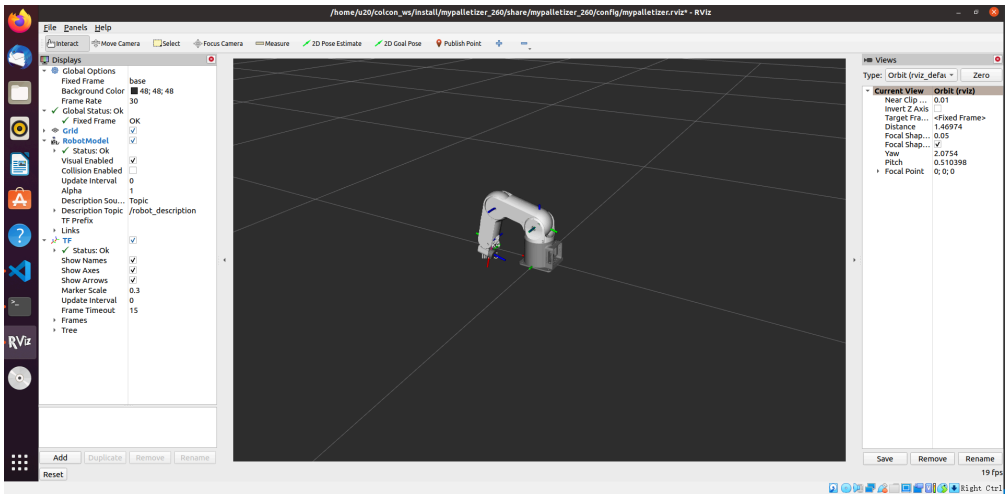
Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



Then run the command:

```
# The default serial port name of mypalletizer 260-PI version is "/dev/ttyAMA0", and the baud rate is 1000000.
ros2 launch mypalletizer_260 teleop_keyboard.launch.py
```

Running effect is as follows:



mypalletizer information will be output in the command line as follows:

```
[INFO] [launch]: All log files can be found below /home/u20/.ros/log/2022-08-01-14-53-35-880311-u20-VirtualBox-5591
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [robot_state_publisher-1]: process started with pid [5594]
[INFO] [rviz2-2]: process started with pid [5596]
[INFO] [follow_display-3]: process started with pid [5598]
[robot_state_publisher-1] Parsing robot urdf xml string.
[robot_state_publisher-1] Link link1 had 1 children
[robot_state_publisher-1] Link link2 had 1 children
[robot_state_publisher-1] Link link3 had 1 children
[robot_state_publisher-1] Link link4 had 1 children
[robot_state_publisher-1] Link link5 had 0 children
[robot_state_publisher-1] [INFO] [1659336816.100317312] [robot_state_publisher]: got segment base
[robot_state_publisher-1] [INFO] [1659336816.100434612] [robot_state_publisher]: got segment link1
[robot_state_publisher-1] [INFO] [1659336816.100443491] [robot_state_publisher]: got segment link2
[robot_state_publisher-1] [INFO] [1659336816.100451008] [robot_state_publisher]: got segment link3
[robot_state_publisher-1] [INFO] [1659336816.100457862] [robot_state_publisher]: got segment link4
[robot_state_publisher-1] [INFO] [1659336816.100464597] [robot_state_publisher]: got segment link5
[rviz2-2] [INFO] [1659336816.529758865] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-2] [INFO] [1659336816.530082788] [rviz2]: OpenGl version: 3.1 (GLSL 1.4)
[rviz2-2] [INFO] [1659336816.620671305] [rviz2]: Stereo is NOT SUPPORTED
[follow_display-3] [INFO] [1659336816.635957580] [follow_display]: port:/dev/ttyUSB0, baud:115200
[rviz2-2] Parsing robot urdf xml string.
```

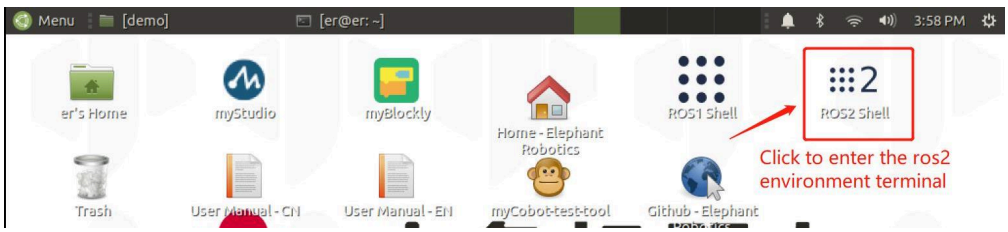
Then open another command line and run:

- mypalletizer 260-M5 version:

```
ros2 run mypalletizer_260 teleop_keyboard
```

- mypalletizer 260-PI version:

Click the `ROS2 Shell` icon on the desktop or the corresponding icon in the lower column of the desktop to open the ROS2 environment terminal:



Then run the command:

```
ros2 run mypalletizer_260_pi teleop_keyboard
```

You will see the following output in the command line:

```
Mycobot Teleop Keyboard Controller
---------------------------
Movimg options(control coordinations [x,y,z,rx]):
              w(x+)

    a(y-)     s(x-)     d(y+)

        z(z-)       x(z+)

        u(rx+)      i(rx-)



Gripper control:
    g - open
    h - close

Other:
    1 - Go to init pose
    2 - Go to home pose
    3 - Resave home pose
    q - Quit

currently:    speed: 30    change percent: 2
```

In this terminal, you can control the state of the robot arm and move it using the keys in the command line.

Parameters supported by this script:

- _speed: the movement speed of the robot arm
- _change_percent: movement distance percentage

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！

💬