



UFACTORY

# UFACTORY xARM

## USER MANUAL

Translated Version



S H E N Z H E N   U F A C T O R Y   C O . ,   L T D

V 2.3.0

## Table

Table .....	2
Preface .....	5
User Manual Information.....	5
Product Information.....	5
Main Contents of the Manual.....	6
Terms and Definitions.....	6
xArm Motion Parameters.....	9
Unit Definition.....	10
Additional Information.....	11
Safety Precautions .....	11
xArm User Manual–Hardware Section.....	18
1. Hardware Installation Manual.....	18
1.1. The Hardware Composition of xArm.....	18
1.2. Robot Installation.....	21
1.3. Power Supply for the Robotic Arm.....	28
2. Electrical Interface.....	31
2.1. AC Control Box .....	31
2.2. DC Control Box .....	33
2.3. Electrical Alarms and Cautions.....	35
2.4. End-Effector I/O.....	36
2.5. Control Box Electrical IO.....	41
2.6. Communication Interface.....	52
2.7. Ethernet TCP/IP.....	52
3. End-Effector .....	54
3.1. Gripper .....	54
3.2. Vacuum Gripper .....	57

xArm User Manual–Software Section.....	60
1. UFactory studio.....	60
1. 1 Hardware Preparation.....	60
1. 2 Connect to the Robotic Arm.....	62
1. 3 UFactory studio Introduction.....	68
1. 4 Robotic Arm Setting.....	69
1. 5 Live Control.....	101
1. 6 Blockly Graphical Programming.....	114
1. 7 Python IDE.....	136
1. 8 GCode .....	138
2. xArm Motion Analysis .....	140
2. 1. Motion of the Robotic Arm.....	141
2. 1. 1. Joint Motion.....	141
2. 2. xArm5 Motion Characteristics.....	153
2. 3. Singularity .....	154
3. Typical Examples.....	157
3. 1. The Use of xArm Vacuum Gripper.....	157
3. 2. The Use of xArm Gripper.....	159
3. 3. The Use of the Digital IO.....	160
4. Robotic Arm Motion Mode and State Analysis.....	161
Appendix .....	168
Appendix1–Error Reporting and Handling .....	168
1. 1 Joints Error Message and Error Handling.....	168
1. 2 Control Box Error Code and Error Handling.....	170
1. 3 Gripper Error Code & Error Handling.....	174
1. 4 Python SDK Error Code & Error Handling.....	175
Appendix2–Technical Specifications .....	178

1.1	xArm5/6/7 Common Specifications .....	178
1.2	xArm 5 Specifications .....	179
1.3	xArm 6 Specifications .....	180
1.4	xArm 7 Specifications .....	181
	Appendix3-FAQ.....	183
	Appendix4-The xArm Software/Firmware Update Method. ....	184
1.	Online upgrade: when PC has network connection.....	184
2.	Offline upgrade: when PC has no network connection.....	185
	Appendix5- Maintenance and Inspection.....	187
	Appendix6- Repair.....	187
	Appendix7-Product Information .....	189
1.1	Product Mark .....	189
1.2	Applied Standards .....	189
1.3	EMC.....	190
1.4	Use Environment .....	190
1.5	Transport, Storage and Handling.....	191
1.6	Power box placement height .....	191
1.7	Power Connection.....	191
1.8	Special Consumables. ....	192
1.9	Stop Categories.....	192
1.10	Stopping Time and Stopping Distance .....	192
1.11	Maximum Speed .....	193
1.12	Maximum Payload.....	193
1.13	Specifications .....	194
	Appendix8-DH Parameters of xArm Series.....	195

# Preface

## User Manual Information

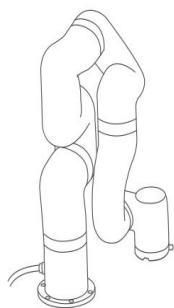
Translated Version V2.3.0.

Apply to Model: XI1300 XI1301 XI1302 XI1303 XI1304 XI1305

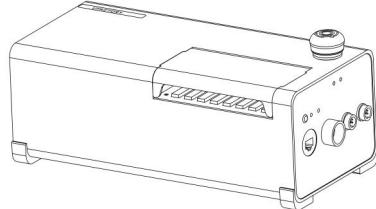
## Product Information

Package contains:

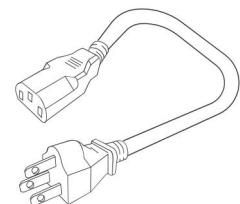
1. Robotic Arm x 1
2. Control Box x 1
3. Power cable for the Control Box x 1
4. Power cable for the Robotic Arm x 1
5. Communication cable for the Robotic Arm x 1
6. Ethernet Cable x1
7. Robotic Arm end effector adapter cable x1



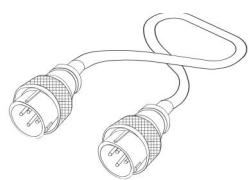
1



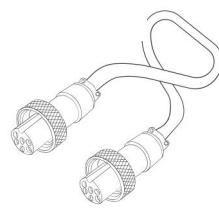
2



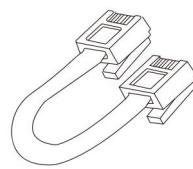
3



4



5



6

# Main Contents of the Manual

xArm User Manual Hardware Section

- (1) xArm hardware installation
- (2) Electrical interface
- (3) xArm end-effector

xArm User Manual Software Section

- (1) UFactory studio instructions
- (2) xArm motion analysis
- (3) Typical examples

Appendix

- (1) xArm error reporting and handling
- (2) xArm technical specifications
- (3) FAQ
- (4) The xArm software/firmware update method
- (5) Maintenance and Inspection
- (6) After-sales service

## Terms and Definitions

The following terms and definitions apply to this manual.

Control Box	The control box, core part of the robotic arm, is the integration of the robotic arm control system.
End Effector	The end effector, installed on the front end of the wrist of the robotic arm, is used to install special tools (such as grippers, vacuum gripper, etc.), which can directly perform work tasks.
Enable Robotic Arm	Power on the robotic arm and turn on the motor of the robotic arm. After the robotic arm is enabled, it can start to move normally.
TCP	Tool center point.
TCP Motion	TCP motion is the Cartesian space motion, with target position in Cartesian space coordinate and the end follows the specified trajectory (arc, line, etc.).

TCP Payload (End Payload)	The payload weight refers to the actual (end tool +other object) weight in Kg; the X / Y / Z-axis indicates the position of the center of mass of the TCP relative to the default tool coordinate system, with unit of mm.
TCP Offset (Tool Center Point Offset)	Set the relative offset between the default tool coordinate system at flange center and the actual tool coordinate system, with distance unit of mm.
Roll/Pitch/Yaw	<p>Roll / Pitch / Yaw sequentially rotates around the X / Y / Z of the selected coordinate system (base coordinate system).</p> <p>The following describes the roll/pitch/yaw orientation representation of {B} relative to {A}:</p> <p>For example, the coordinate system {B} and a known reference coordinate system {A} are first superposed. First rotate {B} around <math>\hat{X}_A</math> by <math>\gamma</math>, then around <math>\hat{Y}_A</math> by <math>\beta</math>, and finally around <math>\hat{Z}_A</math> by <math>\alpha</math>.</p> <p>Each rotation is around a fixed axis of the reference coordinate system {A}. This method is called the XYZ fixed angle coordinate system, and sometimes they are defined as the roll angle, pitch angle, and yaw angle.</p> <p>The above description is shown in the following figure:</p> <p>The equivalent rotation matrix is:</p> ${}^B_R_{XYZ}(\gamma, \beta, \alpha) = R_Z(\alpha)R_Y(\beta)R_X(\gamma)$ <p>Note: <math>\gamma</math> corresponds to roll; <math>\beta</math> corresponds to pitch; <math>\alpha</math> corresponds to yaw.</p>

Axis-Angle	<p>Rx / Ry / Rz representation also, using 3 values to represent the pose (but not three rotation angles), which is the product of a three-dimensional rotation vector [x, y, z] and a rotation angle [phi (scalar)].</p> <p>The characteristics of the axis angle:</p> <p>Assume the rotation axis is [x, y, z], and the rotation angle is phi.</p> <p>Then the representation of the axial angle:</p> <p><math>[Rx, Ry, Rz] = [x * \phi, y * \phi, z * \phi]</math></p> <p>Note:</p> <ol style="list-style-type: none"> <li>1. [x, y, z] is a unit vector, and phi is a non-negative value.</li> <li>2. The vector length (modulus) of [Rx, Ry, Rz] can be used to estimate the rotation angle, and the vector direction is the rotation direction.</li> <li>3. If you want to express reverse rotation, invert the rotation axis vector [x, y, z], and the value of phi remains unchanged.</li> <li>4. Using phi and [x, y, z] can also derive the attitude representation as unit quaternion <math>q = [\cos(\phi / 2), \sin(\phi / 2) * x, \sin(\phi / 2) * y, \sin(\phi / 2) * z]</math>.</li> </ol> <p>For example:</p> <p>The vector of the rotation axis represented by the base coordinate system is [1, 0, 0], and the rotation angle is 180 degrees (<math>\pi</math>), then the axis angle representation of this pose is [<math>\pi</math>, 0, 0].</p> <p>The rotation axis is [0.707, 0.707, 0] and the rotation angle is 90 degrees (<math>\pi / 2</math>), then the axis angle posture is [0.707 * (<math>\pi / 2</math>), 0.707 * (<math>\pi / 2</math>), 0].</p>
The Base Coordinate System  (Please refer to the figure 1)	<p>The base coordinate system is a Cartesian coordinate system based on the mounting base of the robotic arm and used to describe the motion of the robotic arm.</p> <p>(Front and back: X axis, left and right: Y axis, up and down: Z axis)</p>
Tool Coordinate System  (Please refer to the figure 1)	<p>Consists of tool center point and coordinate orientation. If the TCP offset is not set, the default tool coordinate system is located at flange center.</p> <p>For tool coordinate System based motion: The tool center point is taken as the zero point, and the trajectory of the robotic arm refers to the tool coordinate system.</p>

User Coordinate System (Please refer to the figure 1)	The user coordinate system can be defined as any other reference coordinate system rather than the robot base.
Manual Mode	In this mode, the robotic arm will enter the ‘zero gravity’ mode, since the gravity is compensated, the user can guide the robotic arm position directly by hand.
Teach Sensitivity	Teach sensitivity range is from 1 to 5 level. The larger the set value, the higher the teach sensitivity level, and the less the force required to drag the joint in the manual mode.
Collision Sensitivity	The collision sensitivity range is from 0 to 5 level. When it is set to 0, it means that collision detection is not enabled. The larger the set value, the higher the collision sensitivity level, and the smaller the force required to trigger the collision protection response of the robotic arm.
GPIO	General-purpose input and output. For the input, you can check the potential of the pin by reading a register; For the output, you can write a certain register to make this pin output high or low potential;
Safety Boundary	When this mode is activated, the boundary range of the cartesian space of the robotic arm can be limited. If the tool center point (TCP) exceeds the set safety boundary, the robotic arm will stop moving.
Reduced Mode	When this mode is activated, the maximum linear velocity of the Cartesian motion of the robotic arm, the maximum joint speed, and the range of the joint motion will be limited.

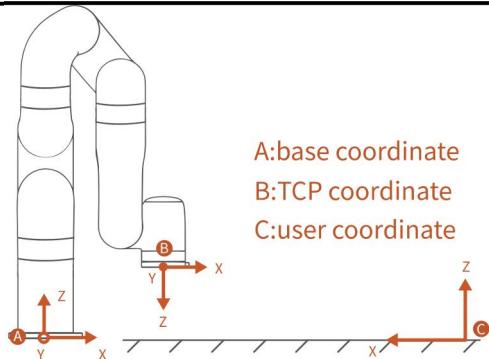


Figure 1

## xArm Motion Parameters

The parameters of the robotic arm are shown in Table 1.1 and Table 1.2.

Table 1.1 Working range of each joint of the robotic arm

	Robotic Arm	xArm 5	xArm 6	xArm 7
Maximum Speed		180° /s	180° /s	180° /s
Working Range	Joint 1	±360°	±360°	±360°
	Joint 2	-118° ~120°	-118° ~120°	-118° ~120°
	Joint 3	-225° ~11°	-225° ~11°	±360°
	Joint 4	-97° ~180°	±360°	-11° ~225°
	Joint 5	±360°	-97° ~180°	±360°
	Joint 6	None	±360°	-97° ~180°
	Joint 7	None	None	±360°

Table 1.2 Range of various motion parameters of the robotic arm

	TCP Motion	Joint Motion
Speed	0~1000mm /s	0~180° /s
Acceleration	0~50000mm /s <sup>2</sup>	0~1145° /s <sup>2</sup>
Jerk	0~100000mm /s <sup>3</sup>	0~28647° /s <sup>3</sup>

Note:

1. In the TCP motion (Cartesian space motion) commands (`set_position()` function of the SDK), If a motion command involves both position transformation and attitude transformation, the attitude rotation speed is generally calculated automatically by the system. In this situation, the specified speed parameter is the maximum linear speed, range from: 0 ~ 1000mm / s.
2. When the expected TCP motion only changes the attitude (roll, pitch, yaw), with position (x, y, z) remains unchanged, the specified speed is the attitude rotation speed, so the range 0 to 1000 corresponds to 0 to 180 ° / s.

## Unit Definition

The Python / Blockly examples and the units standard in the communication protocol are shown in Table 1.3.

Table 1.3. Default units in Python / Blockly example and

## Communication Protocol

Parameter	Python-SDK	Blockly	Communication
X (Y/Z)	millimeter (mm)	millimeter (mm)	millimeter (mm)
Roll (Pitch/Yaw)	degree (°)	degree (°)	radian (rad)
J <sub>1</sub> (J <sub>2</sub> /J <sub>3</sub> /J <sub>4</sub> /J <sub>5</sub> /J <sub>6</sub> /J <sub>7</sub> )	degree (°)	degree (°)	radian (rad)
TCP Speed	mm/s	mm/s	mm/s
TCP Acceleration	mm/s <sup>2</sup>	mm/s <sup>2</sup>	mm/s <sup>2</sup>
TCP Jerk	mm/s <sup>3</sup>	mm/s <sup>3</sup>	mm/s <sup>3</sup>
Joint Speed	° /s	° /s	rad/s
Joint Acceleration	° /s <sup>2</sup>	° /s <sup>2</sup>	rad/s <sup>2</sup>
Joint Jerk	° /s <sup>3</sup>	° /s <sup>3</sup>	rad/s <sup>3</sup>

## Additional Information

For UFactory studio software download and xArm developer manual, please refer to the UFACTORY official website.

<https://www.ufactory.cc/download/>

## Safety Precautions

### ● Introduction

This chapter contains essential safety information, integrators and users of xArm must follow the instructions and pay special attention to the content with warning signs. Due to the complexity of the robotic arm system and its degree of danger, please ensure you fully understand the content of this manual and strictly adhere to the instructions. When using SDK (Python/ROS/C++) and graphical interface (UFactory studio), please read the relevant interface instructions demonstrated in this operation manual.

UFACTORY devotes to providing reliable and safety information, but these contents do not constitute warranties by UFACTORY. UFACTORY will not have or accept any liability, obligation, or responsibility

whatsoever for any loss, destruction, or damage arising from or in respect of any use or misuse of xArm.

## ● Validity and Responsibility

The information does not cover how to design, install, and operate a complete robotic arm application, nor does it cover all peripheral equipment that can influence the safety of the complete system. The complete system must be designed and installed under the safety requirements outlined in the standards and regulations of the country where the robotic arm installed.

The integrators of the xArm are responsible for the compliance of applicable safety laws and regulations in the country, to prevent any hazards in the operating environment.

Safety precautions include but are not limited to:

- Making a risk assessment for the complete system. Make sure to have a safe distance between people and xArm when interacting with the xArm.
- Interfacing other machines and additional safety devices if defined by the risk assessment.
- For software programming, please read the interface documentations carefully and set up the appropriate safety functions in the software.
- Specifying instructions for use to prevent unnecessary property damage or personal injury caused by improper operation.

## ● Limitations on Liability Exceptions

Any information given in this manual regarding safety must not be construed as a warranty by UFACTORY that the xArm will not cause

injury or damage even if all safety instructions are complied with.

## ● Safety Alarms in this Manual



**DANGER**

### **DANGER:**

This indicates an imminently hazardous electrical situation, which if not avoided, could result in death or serious damage to the device.



**WARNING**

### **WARNING:**

This indicates a potentially hazardous situation which, if not avoided, could result in death or serious damage to the device.



**HIGH TEMPERATURE**

### **HIGH TEMPERATURE**

This indicates a potential hot surface, which if touched, could result in personal injury.



**NOTICE**

### **NOTICE**

If not avoided, could result in personal injury or damage to the equipment.



**CAUTION**

### **CAUTION:**

If not avoided, could result in personal injury or damage to the equipment.

## ● Safety Precautions

### Overview

This section contains some general warnings and cautions on installation and application planning for the robotic arm. To prevent

damage to the machine and associated equipment, users need to learn all the relevant content and fully understand the safety precautions. We do not control or guarantee the relevance or completeness of such information in this manual, for which users should conduct self-assessment of their specific problems.

### General Alarms and Cautions



DANGER

1. Make sure to use the correct installation settings in this manual for the robotic arm and all the electrical equipment.
2. Please follow the instructions in this manual, installation, and commissioning needs to be performed by professionals in accordance with the installation standards.
3. Make sure the robotic arm and tool are properly and securely bolted in place.
4. The integrity of the device and system must be checked before each use (e. g. the operational safety and the possible damage of the robotic arm and other device systems).
5. Preliminary testing and inspection for both robotic arm and peripheral protection system before production is essential.
6. The operator must be trained to guarantee a correct operation procedure when using SDK(Python/ROS/C++) and graphical interface UFactory studio.
7. A complete safety assessment must be recorded each time the robotic arm is re-installed and debugged.
8. When the robotic arm is in an accident or abnormal operation, the emergency stop switch needs to be pressed down to stop the movement, and the posture of the robotic arm will

slightly brake and fall.

9. The xArm joint module has brakes inside, which will remain manipulator's pose when a power outage occurs.
10. When the robotic arm is in operation, make sure no people or other equipment are in the working area.
11. When releasing the brakes of xArm, please take protective measures to prevent the robotic arm or operator from damage or injury.
12. When connecting the xArm with other machinery, it may increase risk and result in dangerous consequences. Make sure a consistent and complete safety assessment is conducted for the installation system.



**HIGH TEMPERATURE**

1. The robotic arm and Control Box will generate heat during operation. Do not handle or touch the robotic arm and Control Box while in operation or immediately after the operation.

2. Never stick fingers to the connector of the end-effector.

1. Make sure the robotic arm's joints and tools are installed properly and safely, and check the status for all circuits.

2. Make sure that there is enough space for the manipulator to move freely.

3. Make sure that there is no obstacle in the robotic arm's working space.

4. The Control Box must be placed outside the working range of the robotic arm to ensure the emergency stop button can be pressed once an emergency occurs.

5. If the robotic arm is in operation and needs an emergency stop, make sure the restart/reset



**CAUTION**

motions will not collide with any obstacle.

6. Do not modify the robotic arm (or Control Box). Any modification may lead to unpredictable danger to the integrators. The authorized restructuring needs to be in accordance with the latest version of all relevant service manuals. If the robotic arm is modified or altered in any way, UFACTORY (Shenzhen) Technology Co., Ltd. disclaims all liability.
7. Users need to check the collision protection and water-proof measures before any transportation.



**NOTICE**

When the xArm cooperates with other machinery, a comprehensive safety assessment of the entire collaboration system should be performed. It is recommended that any equipment that may cause mechanical damage to xArm be placed outside the working range during application planning.

## Operator Safety

In the operation of the robotic arm system, we must ensure the safety of the operators first, with the general precautions listed in the table below. Please take appropriate measures to ensure the safety of operators.



**CAUTION**

1. Each operator who uses the robotic arm system should read the product user manual carefully. Users should fully understand the standardized operating procedures with the robotic arm, and the solution to the robotic arm running error.
2. When the device is running, even if the robotic arm seems to stop, the robotic arm may be waiting for the signal and in the upcoming action status. Even in such a state, it should

- be considered as the robotic arm is in action.
3. A line should be drawn to mark the range of motion of the robotic arm to let the operator acknowledge the robotic arm, including its end tools (such as gripper and suction cup, etc) operating range.
  4. Check the robotic arm regularly to prevent loosening of the bolts that may cause undesirable consequences.
  5. Be careful when the robotic arm is running too fast.
  6. Be careful about dropping items that can be caused by accidental power off or unstable clamping of the robotic arm.

# xArm User Manual—Hardware Section

## 1. Hardware Installation Manual

### 1.1. The Hardware Composition of xArm

#### 1.1.1. Hardware Composition

The composition of robotic arm hardware includes:

- Robotic Arm (Figure 2-1)
- Control Box (Figure 2-2)
- Robotic Arm Signal Cable (Figure 2-3)
- Robotic Arm Power Supply Cable (Figure 2-4)
- Control Box Power Supply Cable (Figure 2-5)

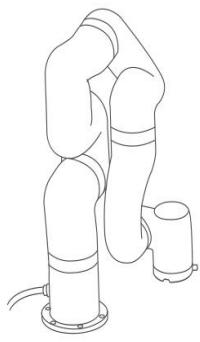


Figure 2-1

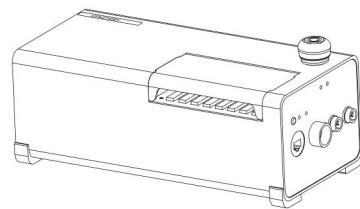


Figure 2-2



Figure 2-3

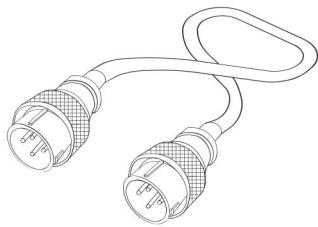


Figure 2-4

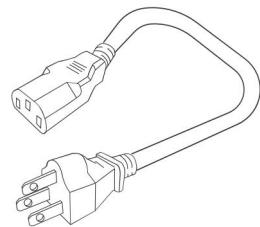


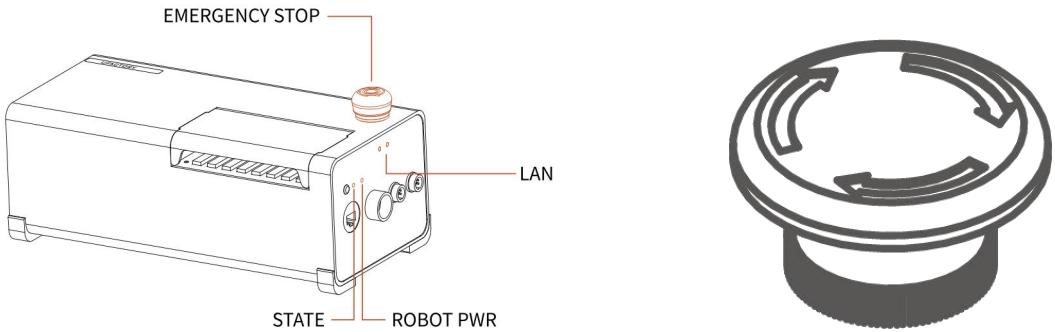
Figure 2-5

The xArm robotic arm system consists of a base and rotary joints, and each joint represents a degree of freedom. From the bottom to the top, in order, Joint 1, Joint 2, Joint 3, etc. The last joint is known as the tool side and can be used to connect end-effector (e. g. gripper, vacuum gripper, etc).

Refer to technical specifications for joint Figures (See appendix-2) .

### 1.1.2. Emergency Stop Button

By pressing the emergency stop button of the Control Box, a command will be sent to the Control Box for software deceleration to stop all activities of the robotic arm and clear all the cached commands in the Control Box; the power supply for the robotic arm will be removed within 300ms. The emergency stop should not be used as a risk reduction measure. When an emergency occurs during the operation of the robotic arm, users need to press the emergency stop, and the posture of the robotic arm will slightly brake and fall. The emergency stop button is shown below:



**Emergency Stop:** press the emergency stop button to power off the xArm, and the power indicator will go out.

**Power-on:** when the button is rotated in the direction indicated by the arrow, the button is pulled up, the xArm power indicator lights up, and the arm is powered.

**Note:**

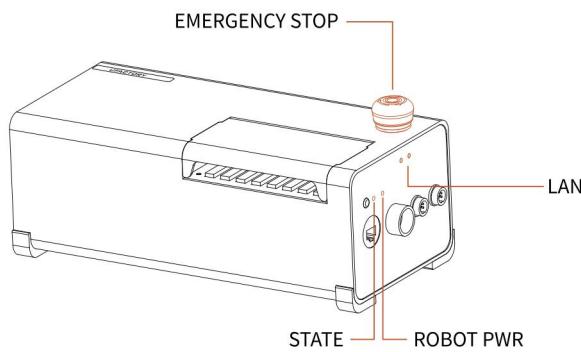
After pressing the emergency stop button, the following operations should be performed to re-start the xArm:

1. Power up the xArm (Turn the emergency stop button in the direction of the arrow)
2. Enable the xArm (enable the servo motor)

UFactory studio: enable the robotic arm: click the button:  
[Enable Robot]

Python-SDK: enable the robotic arm: `motion_enable (true)`

### 1.1.3. Control Box Description



Control Box Buttons and Indicators	Parameter Name	Function
ROBOT power indicator	ROBOT PWR	The light is on, indicating that the xArm is powered on.
Control Box power status indicator	STATE	The light flashes, indicating that the control box is powered on.
Network port indicator	LAN	The light is on, indicating that the xArm is communicating normally.
Emergency stop button	EMERGENCY STOP	Press the button to power off the xArm; Rotate the button, the ROBOT power indicator of the xArm lights up.

## 1.2. Robot Installation

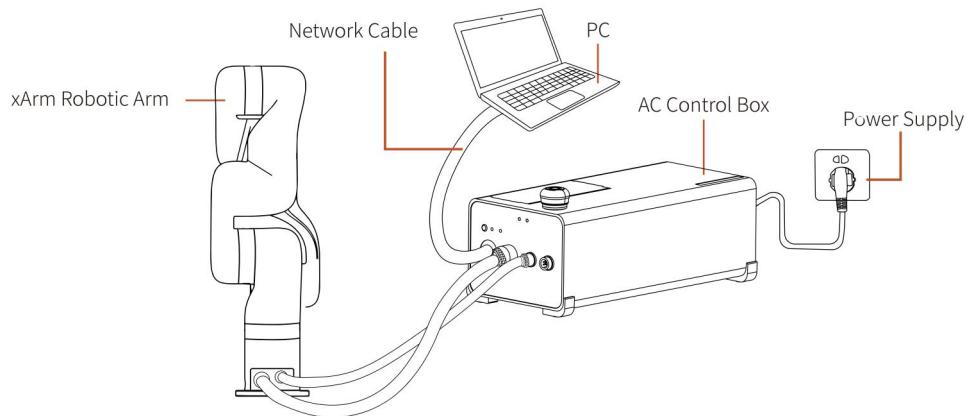
### 1.2.1. Safety Guidelines for the Robot Environment

 <b>DANGER</b>	<ol style="list-style-type: none"> <li>1. Make sure the arm is properly and safely installed in place. The mounting surface must be shockproof and sturdy.</li> <li>2. To install the arm body, check that the bolts are tight.</li> <li>3. The robotic arm should be installed on a sturdy surface that is sufficient to withstand at least 10 times the full torsion of the base joint and at least 5 times the weight of the arm.</li> </ol>
 <b>WARNING</b>	<ol style="list-style-type: none"> <li>1. The robotic arm and its hardware composition must not be in direct contact with the liquid, and should not be placed in a humid environment for a long time.</li> <li>2. A safety assessment is required each time installed.</li> <li>3. When connecting or disconnecting the arm cable, make sure that the external AC is disconnected. To avoid any electric shock hazard, do not</li> </ol>

connect or disconnect the robotic arm cable when the robotic arm is connecting with external AC.

### 1.2.2. Robot Installation

1. Brief installation steps:
  - a. Define a robotic arm workspace
  - b. Fix the robotic arm base
  - c. Connect the robotic arm with the Control Box
  - d. Connect the Control Box with cable
  - e. Install end-effector

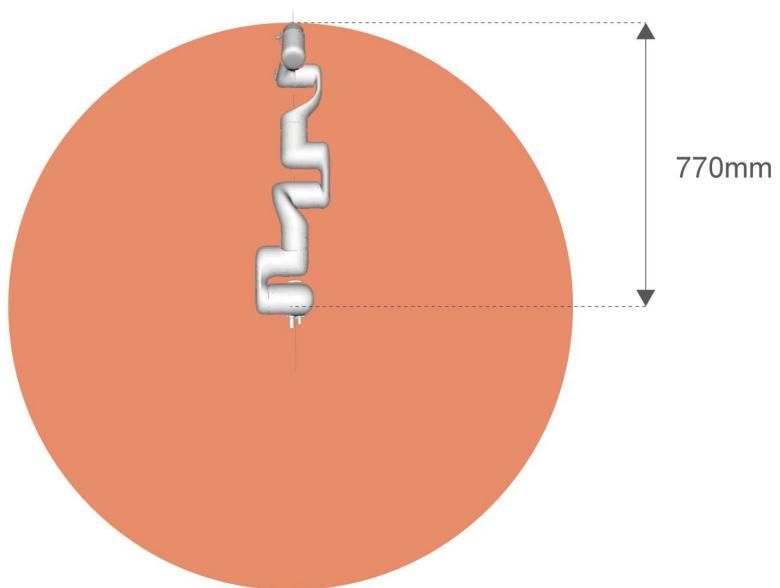
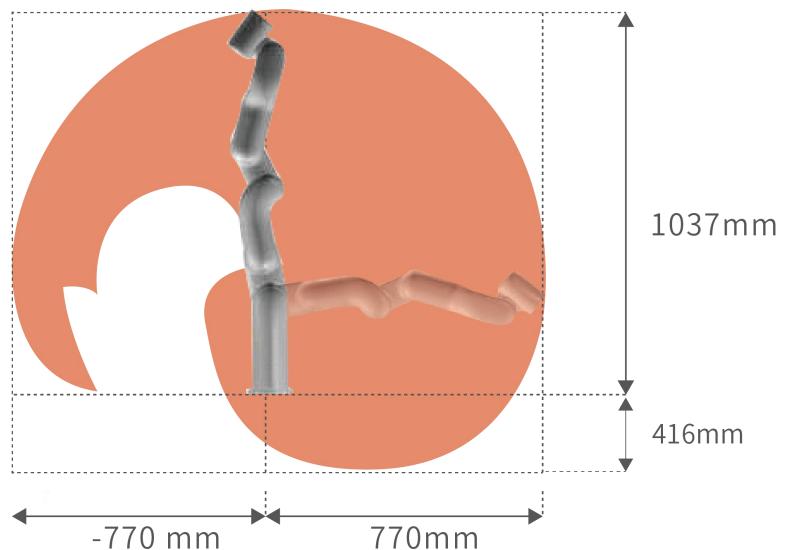


#### 1.2.2.1. Define a Robotic Arm Workspace

The robotic arm workspace refers to the area within the extension of the links. The figure below shows the dimensions and working range of the robotic arm. When installing the robotic arm, make sure the range of motion of the robotic arm is taken into account, so as not to bump into the surrounding people and equipment (the end-effector not included in the working range).

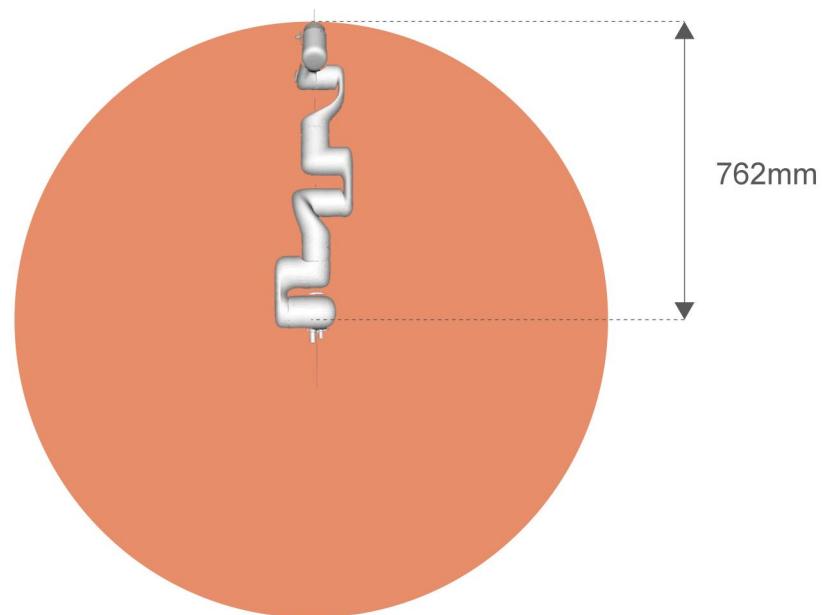
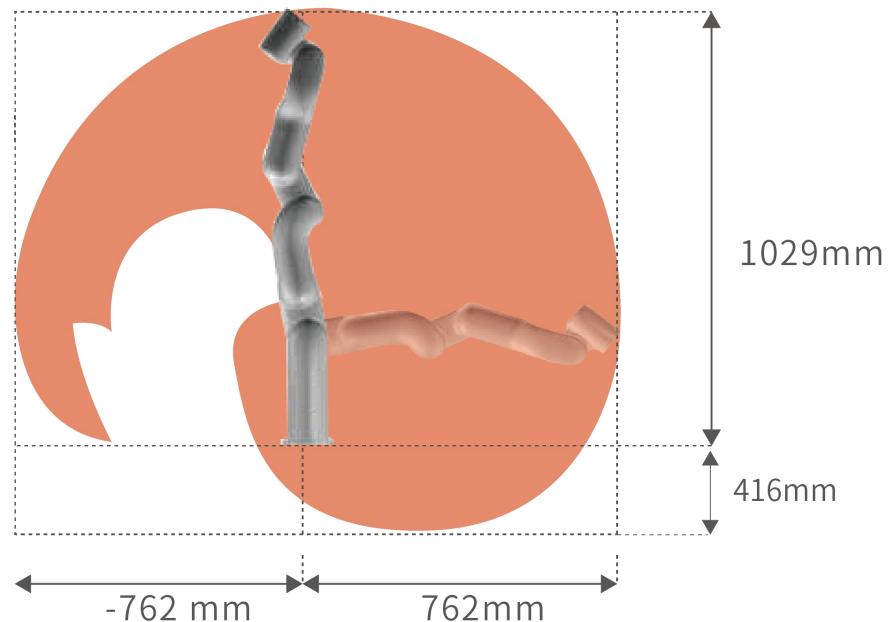
### Working space of xArm7 (unit: mm)

Note : The following working range diagrams are only for safety assessment.



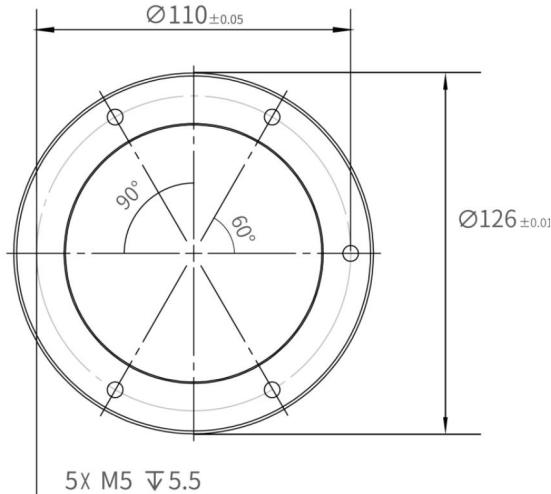
### Working space of xArm5 and xArm6 (unit: mm)

Note : The following working range diagrams are only for safety assessment.



### 1.2.2.2. Robot Installation

The robotic arm has five M5 bolts provided and can be mounted through five  $\varnothing 5.5$  holes in the base of the robotic arm. It is recommended to tighten these bolts with a torque of  $20N \cdot m$ .



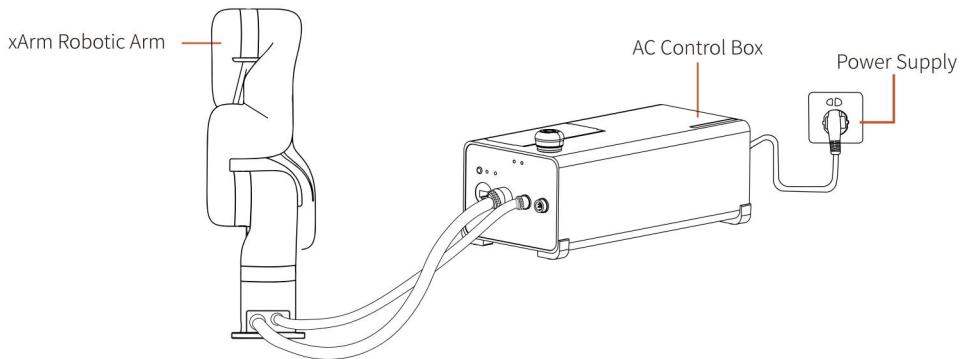
Robot Base Mounting (unit: mm)

### 1.2.2.3. Robotic Arm is Connected to the Control Box

Plug the connector of the Robotic Arm Power Supply Cable and the Robotic Arm Signal Cable into the interface of the Robotic Arm. The connector is a foolproof design. Please do not unplug and plug it violently;

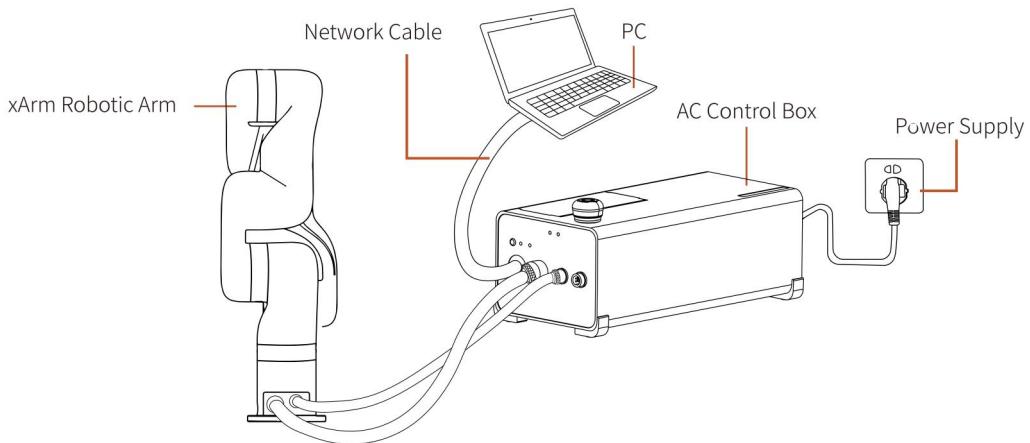
Plug the Robotic Arm Power Supply Cable and the Robotic Arm Signal Cable into the Control Box;

Plug the Control Box Power Cable into the AC (110V–240V) interface on the Control Box and the other end into the socket (as shown in Figure below).



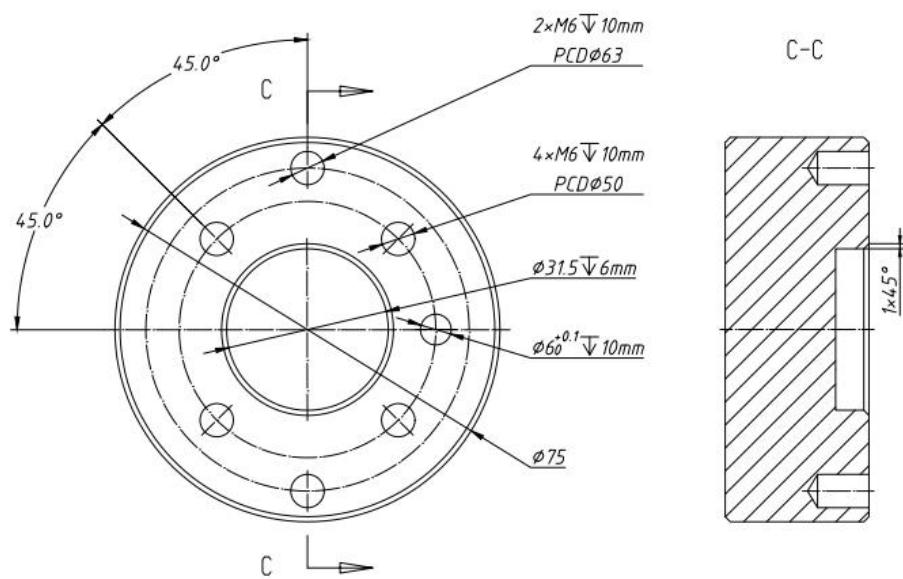
#### 1.2.2.4. Control Box Networking

Plug the Network Cable into the interface marked LAN on the Control Box, and plug the other end of the Network Cable into the computer.

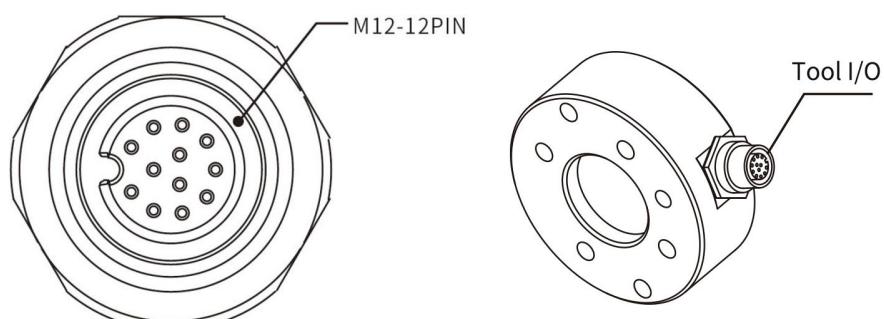


#### 1.2.2.5. End-effector Installation

The End-effector flange has 6 M6 threaded holes and one  $\Phi 6$  positioning hole, where the end-effector of two different sizes can be mounted. If the effector does not have a positioning hole, the orientation of the end-effector must be documented in a file format, to avoid errors and unexpected results when re-installing the end-effector. The end-effector flange referenced ISO 9409-1-50-4-M6 standard.



Mechanical dimensions of end-effector flange (unit: mm)



Drawing of tool I/O



**DANGER**

1. Make sure the tool is properly and safely bolted in place.
2. If the end-effector does not have a locating hole, the orientation of the end-effector must be archived as a file.
3. Make sure that the tool is safely constructed

such that it cannot create a hazardous situation by dropping a part unexpectedly.

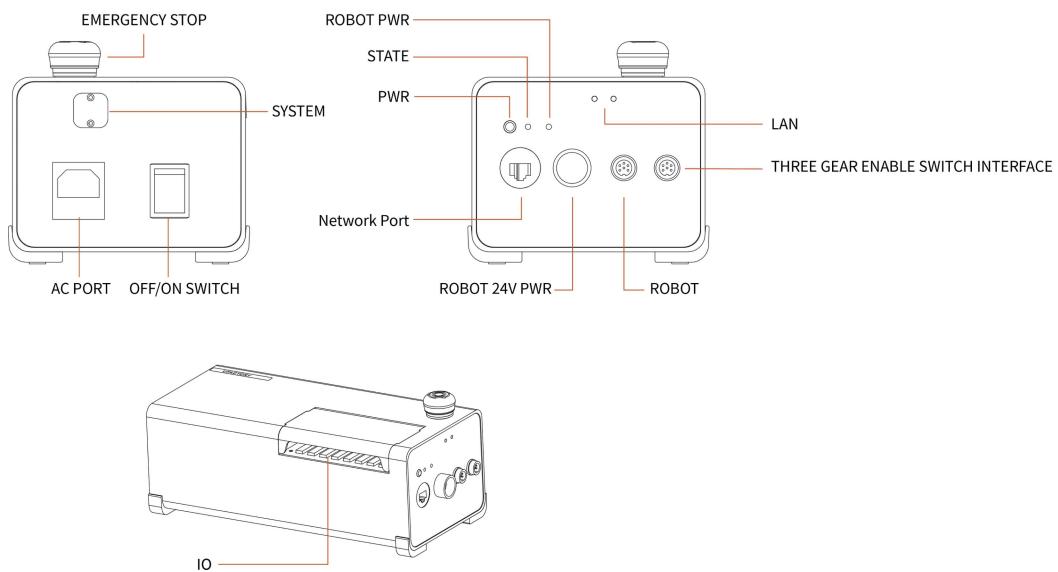
4. Pay attention to the operation specifications of sharp end-effector tools.
5. If the installed end-effector exceeds the robotic arm mounting surface at the zero position of the robotic arm, a safety assessment is required for the zero return operation.

## 1.3. Power Supply for the Robotic Arm

### 1.3.1. Preparation before Power On

- Ensure the power cable and the communication wire are properly connected between the Control Box and the robotic arm.
- Ensure the network cable or RS-485 cable is properly connected.
- Ensure the power cable for the Control Box is properly connected.
- Ensure the xArm will not hit any personnel or equipment within the working range.

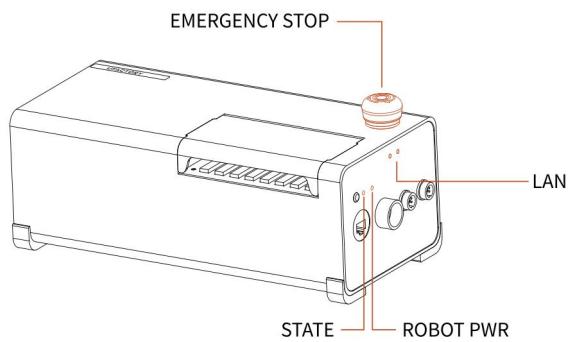
### 1.3.2. Power On



1. Turn on the OFF/ON button and ensure the indicator lights are lit.
2. Press the power button, when the status indicator (CONTROLLER) lights up, the control box is turned on.
3. Rotate the emergency stop button in the direction indicated by the arrow and is pulled up, at which point the xArm power indicator (ROBOT PWR) lights up.
4. Use the UFactory studio / SDK command to complete the operation of enabling the robotic arm. (enable the servo motor)

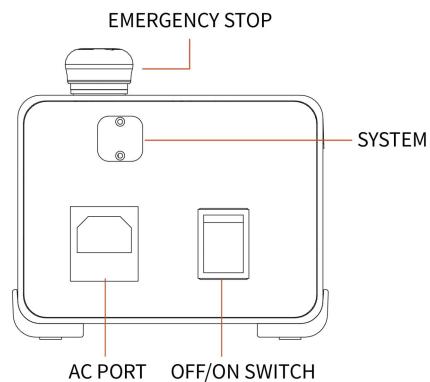
### 1.3.3. Shut Down the Robotic Arm System

1. Shutdown Sequence
  - (1) Press the EMERGENCY STOP button to power off the robotic arm.
  - (2) Ensure the power indicator light is off.



2. Shutdown the control box

- (1) Turn off the power supply of the control box. (The power switch takes about 5 seconds to turn off the power of the control box. Please do not restart the control box within 5 seconds after turning off the power supply)



**WARNING**

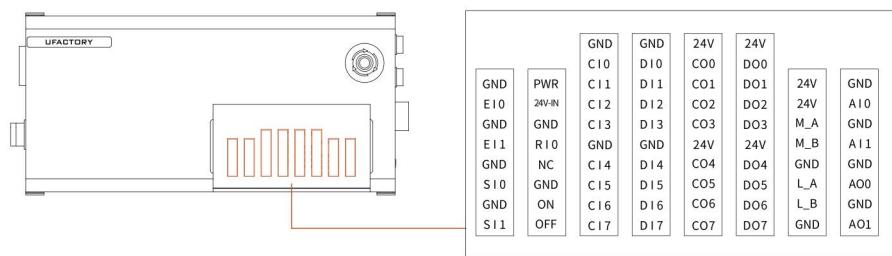
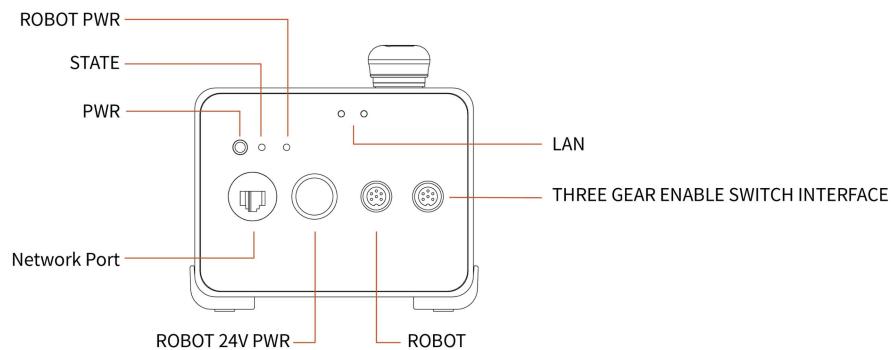
Unplugging the power cord directly from the wall outlet to shut down the system may result in damage to the file system of the control box, which may result in robotic arm malfunction.

## 2. Electrical Interface

### 2.1. AC Control Box

#### 2.1.1. Connect the Control Box to the Robotic Arm

1. The robotic arm power supply cable connects the power port of the robotic arm and the ROBOT power port of the control box.
2. The robotic arm signal cable is connected to the signal interface of the robotic arm and the ROBOT signal interface of the control box.



#### 2.1.2. Power Connection

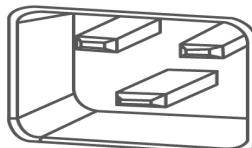
There is a standard IEC plug at the end of the control box's main cable.

Connect a local dedicated main outlet or cable to the IEC plug. The control box is powered by 100V–240V AC (the input frequency is 47–63HZ) and its internal switching power supply converts 100V–240V AC into 12V, 24V DC, which supplies power to the load of the control box and the robotic arm.

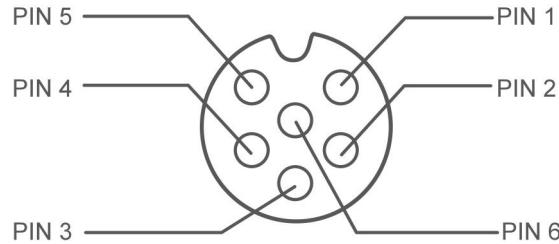
Therefore, it is necessary to check whether the connection between the robotic arm and the control box is secured before use. The hardware protection and software protection of the control box can ensure the safety of use largely. The emergency stop button of the control box allows the user to cut off the power of the robotic arm in the shortest time possible and protect the safety of both personnel and the equipment.

To power on the robotic arm, the control box must be connected to the power supply. In this process, the corresponding IEC C19 wire must be used.

Connect to the standard IEC C20 plug of the Control Box to complete the process, see the figure below.



### 2.1.3. Definition of the Robotic Arm Industrial Connector

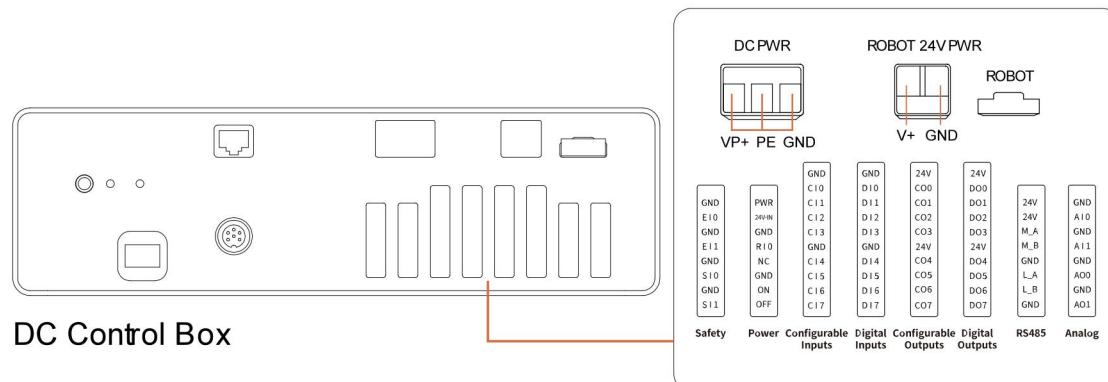


6-Pin Industrial Connector (Robot Communication )	
Industrial connector wire sequence	Functional definition
1	GND
2	485-A Arm
3	485-B Arm
4	GND
5	485-A Tool
6	485-B Tool

## 2.2. DC Control Box

### 2.2.1. External Interfaces of Control Box

Except that the DC Control Box is connected to different power sources, the other electrical interface specifications and functions are the same as the AC Control Box.



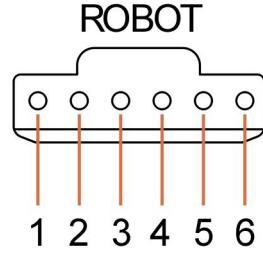
DC PWR:

VP+: Connect to 24V-72V DC Input

PE: Connect to the control housing (Leakage Protection), not a must

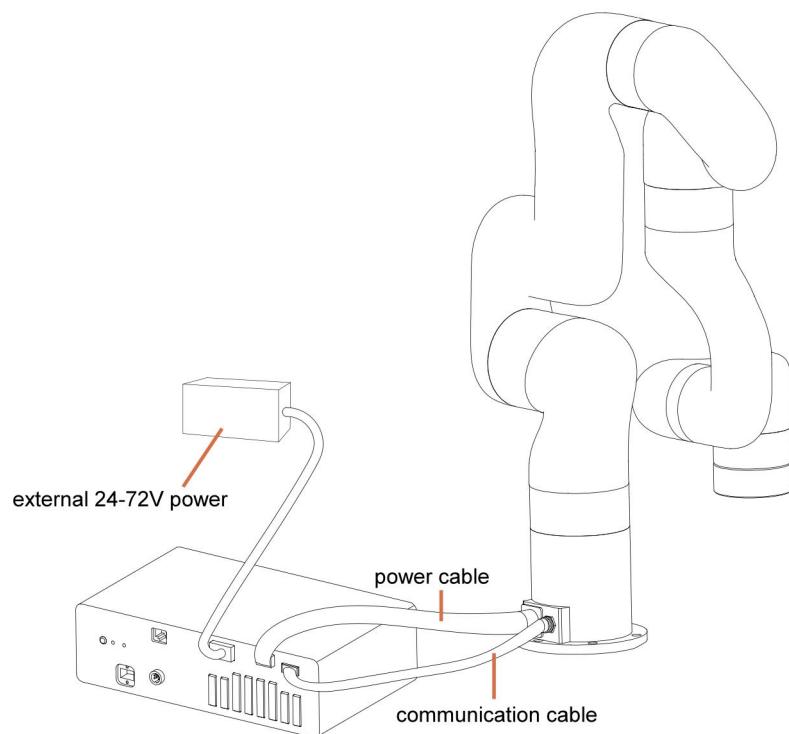
GND: Connect to the ground

ROBOT communication cable:



PIN1	GND
PIN2	485-A Arm
PIN3	485-B Arm
PIN4	GND
PIN5	485-A Tool
PIN6	485-B Tool

### 2.2.2. Cable Connection



## 2.2.3. Specification of External Power

DC1300	
Input	24V~72V DC
Output	24VDC 672Wmax
I/O power supply	24V 1.8A(Internal power supply) 24V 3A(External power supply)

## 2.3. Electrical Alarms and Cautions

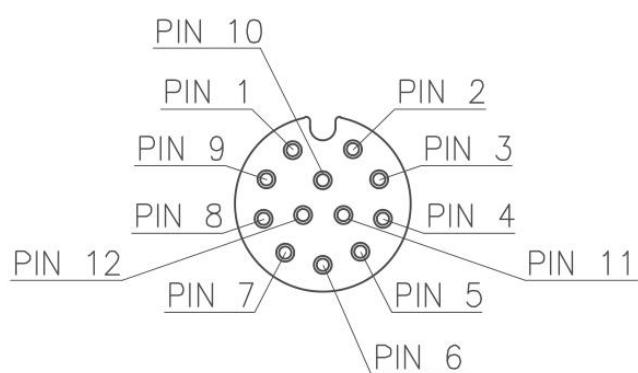
Always follow the warnings and cautions below when designing and installing a robotic arm application. These warnings and cautions are also subject to the implementation of maintenance work.

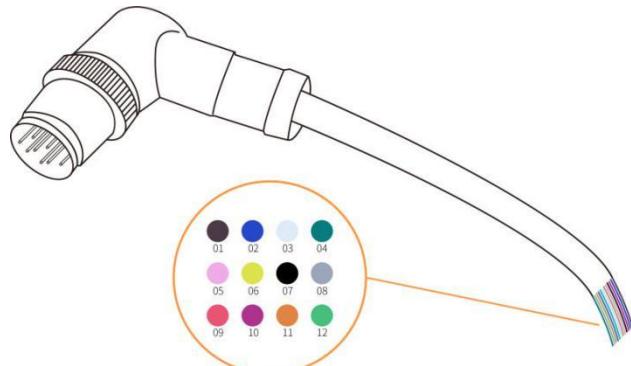
 <b>DANGER</b>	<ol style="list-style-type: none"><li>Never connect a safety signal to a non-safety PLC. Failure to follow this warning may result in serious injury or death due to an invalid safety stop function.</li></ol>
 <b>NOTICE</b>	<ol style="list-style-type: none"><li>Make sure that all the non-waterproof equipment is kept dry. If water enters the product, turn off the power supply, and contact your supplier.</li><li>Use only the original cable of the robotic arm. Do not use the robotic arm in applications where the cable needs to be bent. If you need a longer cable or flexible cable, please contact your supplier.</li><li>All GND connectors mentioned in this manual are only suitable for powering and transmitting signals.</li><li>Be careful when installing the interface cable to the I/O of the robotic arm.</li></ol>

 <b>CAUTION</b>	<ol style="list-style-type: none"> <li>1. Interfering signals above the level specified in the IEC standard will cause abnormal behaviour of the robotic arm. Extremely high signal levels or excessive exposure can cause permanent damage to the robotic arm. UFACTORY (Shenzhen) Technology Co., Ltd. is not responsible for any loss caused by EMC problems.</li> <li>2. The length of the I/O cable that used to connect the Control Box with other mechanical and plant equipment must not exceed 30 meters unless it is feasible after the extension testing.</li> </ol>
 <b>WARNING</b>	<ol style="list-style-type: none"> <li>1. When wiring the electrical interface of the Control Box, the Control Box must be powered off.</li> </ol>

## 2. 4. End-Effector I/O

At the tool side of the robotic arm, there is an avionic socket 12-pin female industrial connector. This connector provides power and control signals for the grippers and sensors used on a particular robotic arm tool. Please refer to the figure below:





There are 12 pins inside the cable with different colors, each color represents different functions, please refer to the following table:

Pin sequence	Color	Signal
1	Brown	+24V (Power)
2	Blue	+24V (Power)
3	White	0V (GND)
4	Green	0V (GND)
5	Pink	User 485-A
6	Yellow	User 485-B
7	Black	Tool Output 0 (TO0)
8	Grey	Tool Output 1 (TO1)
9	Red	Tool Input 0 (TI0)
10	Purple	Tool Input 1 (TI1)
11	Orange	Analog input 0 (AI0)
12	Light Green	Analog input 1 (AI1)

The electrical specifications are as follows:

Parameter	Min. Value	Typical Value	Max. Value	Unit
Supply Voltage in 24V Mode	20	24	30	V
Supply Current *	-	-	1800	mA

Note: \* It is strongly recommended to use a protection diode for inductive loads.



**DANGER**

Make sure that the connecting tool and the gripper do not cause any danger when the power is cut, such as dropping of the work-piece from the tool.

## 2.4.1. Digital Output

The digital output is implemented in the form of NPN with an open collector (OC). When the digital output is activated, the corresponding connector will be driven to GND. When the digital output is disabled, the corresponding connector will be open (open collector/open drain). The electrical specifications are as follows:

Parameter	Min	Typical	Max	Unit
Open-circuit Voltage	-0.5	-	30	V
Voltage when sinking 50mA	-	0.05	0.20	V
Sink Current	0	-	100	mA
Current through GND	0	-	100	mA

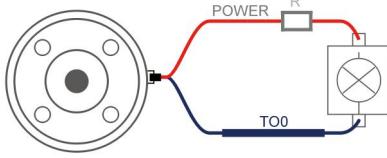


**CAUTION**

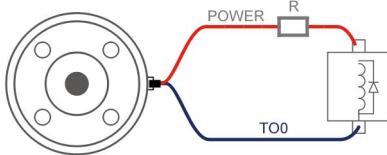
There is no current protection on the digital output of the tool, which can cause permanent damage if the specified value exceeded.

### 2.4.1.1. Tool Digital Output Usage

The following example illustrates how to use the digital output. As the internal output is an open collector, the resistor should be connected to the power supply according to the load. The size and power of the resistor depend on the specific use.



Note: It is highly recommended to use a protection diode for inductive loads as shown below.



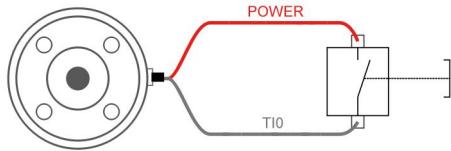
## 2.4.2. Digital Input

The digital input is already equipped with a pull-down resistor. This means that the reading of the floating input is always low. The electrical specifications are as follows:

Parameter	Min	Typical	Max	Unit
Input Voltage	-0.5	-	30	V
Logic Low Voltage	-	-	1.0	V
Logic High Voltage	1.6	-	-	V
Input Resistance	-	47k	-	$\Omega$

### 2.4.2.1. Tool Digital Input Usage

The following figure shows the connection with the simple switch.



## 2.4.3. Tool Analog Input

The tool analog input is a non-differential input. The electrical specifications are as follows:

Parameter	Min	Typical	Max	Unit
Input Voltage in Voltage Mode	-0.5	-	3.3	V

Resolution	-	12	-	Bit
Input Current in Current Mode	-	-	-	mA
Pull-down Resistors in the 4mA to 20mA Current	-	-	165	$\Omega$
Resolution	-	12	-	Bit

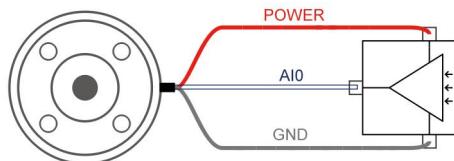


**CAUTION**

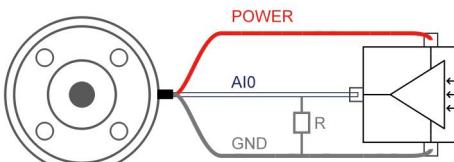
1. In the current/voltage mode, the analog input does not provide over-voltage protection. Exceeding the limits in the electrical code may result in permanent damage to the input port.
2. In current mode, the pull-down resistance depends on the range of the input current.

#### 2. 4. 3. 1. Non-differential Analog Input

The following figures show how the analog sensor can be connected to a non-differential output.



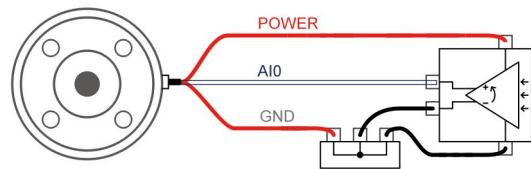
Voltage mode



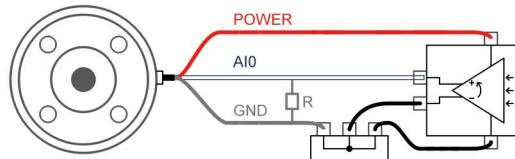
Current mode

#### 2. 4. 3. 2. Differential Analog Input

The following figures show how the analog sensor is connected to the differential output. Connect the negative output to GND (0V), and it can work like a non-differential sensor.



Voltage Mode



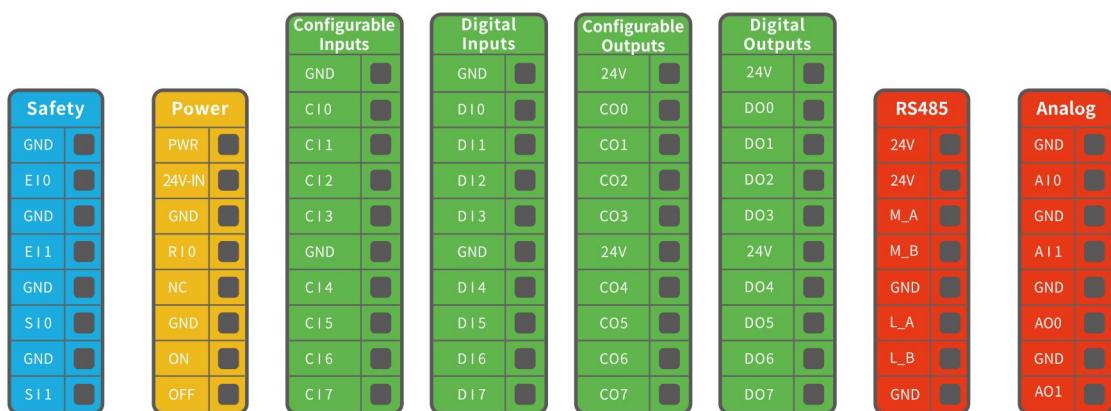
Current Mode

## 2.5. Control Box Electrical I/O

This chapter explains how to connect devices to the electrical I/O outside of the control box.

The I/Os are extremely flexible and can be used in many different devices, including pneumatic relays, PLCs, and emergency stop buttons.

The figure below shows the electrical interface layout inside the control box.



## 2.5.1. General Specifications for all Digital I/O

This section describes the electrical specifications for the following 24V digital I/Os for the Control Box.

- Dedicated safety I/O.
- Configurable I/O.

For the specific configuration functions of IO, see the following table:

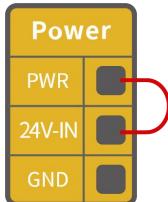
Configurable Function	CI0-CI7	DIO-DI07
General Input	Yes	Yes
Stop Moving	Yes	No
Safeguard Reset	Yes	No
Offline Task	Yes	Yes
Manual Mode	Yes	Yes
Reduced Mode	Yes	No
Enable Robot	Yes	Yes

Configurable Function	C00-C07	D00-D07
General Output	Yes	Yes
Motion Stopped	Yes	Yes
Robot Moving	Yes	Yes
Error	Yes	Yes
Warning	Yes	Yes
Collision	Yes	Yes
Manual Mode	Yes	Yes
Reduced Mode	Yes	Yes
Offline Task Running	Yes	Yes
Robot Enabled	Yes	Yes
Emergency Stop is Pressed	Yes	Yes

It is very important to install xArm according to the electrical specifications.

All the I/O must comply with the specifications. The digital I/O can

be powered by a internal 24V power supply or by an external power supply by configuring the power junction box. In the following figure, PWR is the internal 24V power output. The lower terminal (24V-IN) is the 24V input external power input for I/O. The default configuration is to use internal power, see below.



If larger current is needed, connect the external power supply as shown below.



The electrical specifications for the internal and external power supplies are as follows.

Terminal	Parameter	Min. Value	Typical Value	Max. Value	Unit
Built-in 24V Power Supply					
[PWR – GND]	Voltage	23	24	30	V
[PWR – GND]	Current	0	-	1.8	A
External 24V Input Requirement					
[24V – 0V]	Voltage	20	24	30	V
[24V – 0V]	Current	0	-	3	A

The digital I/O electrical specifications are as follows.

Terminal	Parameter	Min. Value	Typical Value	Max. Value	Unit
Digital Output					
[COx]	Current*	0	-	100	mA
[COx]	Voltage Goes Down	0	-	0.5	V
[COx]	Open Drain	0	-	0.1	mA
[COx]	Function	-	NPN (OC)	-	Type
Digital Input					
[EIx/SIx/CIx/RI]	Voltage	0	-	30	V

[EIx/SIx/CIx/RI]	OFF Area	15	-	30	V
[EIx/SIx/CIx/RI]	ON Area (low level)	0	-	5	V
[EIx/SIx/CIx/RI]	Current (0–0.5)	3	-	8	mA
[EIx/SIx/CIx/RI]	Function	-	-	-	Type
Note: ** For resistive or inductive loads up to 1H.					



**CAUTION**

There is no current protection on the digital output of the Control Box. If the specified values exceeded, permanent damage may result.

## 2.5.2. Dedicated Safety I/O

This section describes the dedicated safety inputs and their configurations of the safety I/O. Please follow the universal specifications in Section 2.4.1.

Safety devices and equipment must be installed to comply with the safety instructions and risk assessment (see Chapter 1).

All safety I/Os exist in pairs (redundancy) and must be kept in two separate branches. A single I/O failure should not result in the loss of safety features. There are two fixed safety inputs:

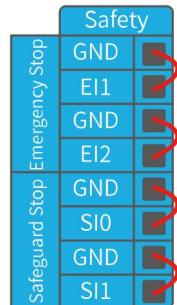
- The robotic arm emergency stop input is only used for the emergency stop of the device.
- The protective stop input is used for all types of safety protection.

The functional differences are as follows.

	Emergency Stop	Protective Stop
Stops the motion of the robotic	Yes	Yes
Program execution	Stop	Suspend
Reset	Manual	Auto or manual
Usage frequency	Not frequent	No limit
Need re-initiation	Only releasing the brake	No

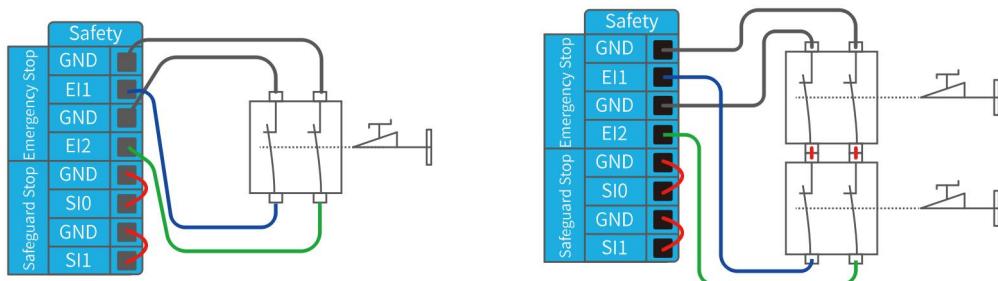
### 2.5.2.1. Default Safety Configuration

The robotic arm has been configured by default and can be operated without any additional safety equipment, as the figure below. If there is a problem with the robotic arm, please check the following figure for the correct connection.



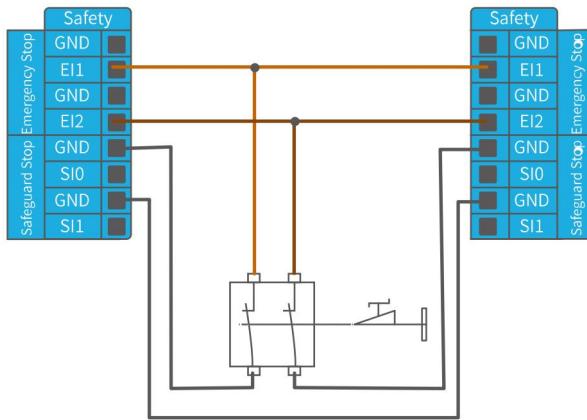
### 2.5.2.2. Connect to the Emergency Stop Button

In most applications, one or more additional emergency stop buttons are required. The figure below shows how to connect one or more emergency stop buttons.



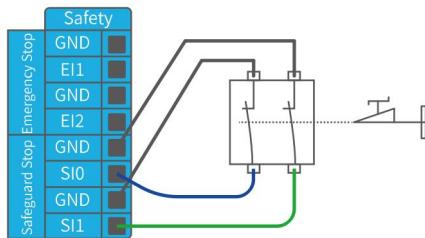
### 2.5.2.3. Share Emergency Stop with other Machines

When a robotic arm is used with other machines, it requires to set up a common emergency stop circuit in most of the time. The following figure shows that two robotic arms share an emergency stop button (the connection method shown in the figure below also applies to multiple robotic arms sharing an emergency stop button).

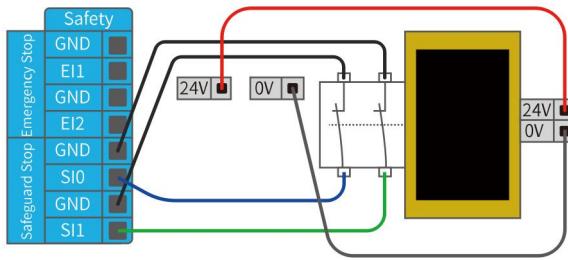


### 2.5.2.4. Automatically Recoverable Protective Stop

The door switch is an example of a basic protective stop device. When the door is open, the robotic arm stops. See the figure below.

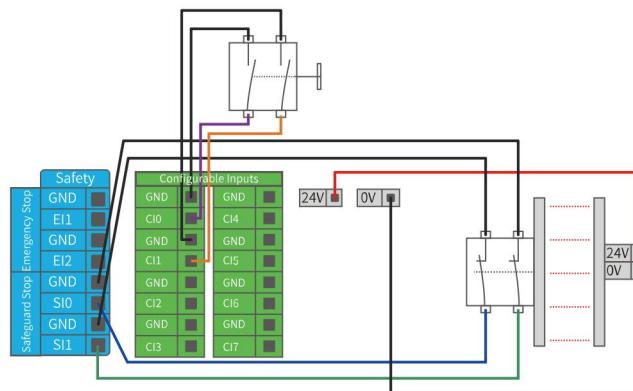


This configuration is only for applications where the operator is unable to close the door from behind. Configurable I/O can be used to set the reset button outside the door, as to reactivate the movement of the robotic arm. Another example of an automatic recovery is the use of a safety pad or a safety laser scanner, see the figure below.



### 2.5.2.5. Protective Stop with Reset Button

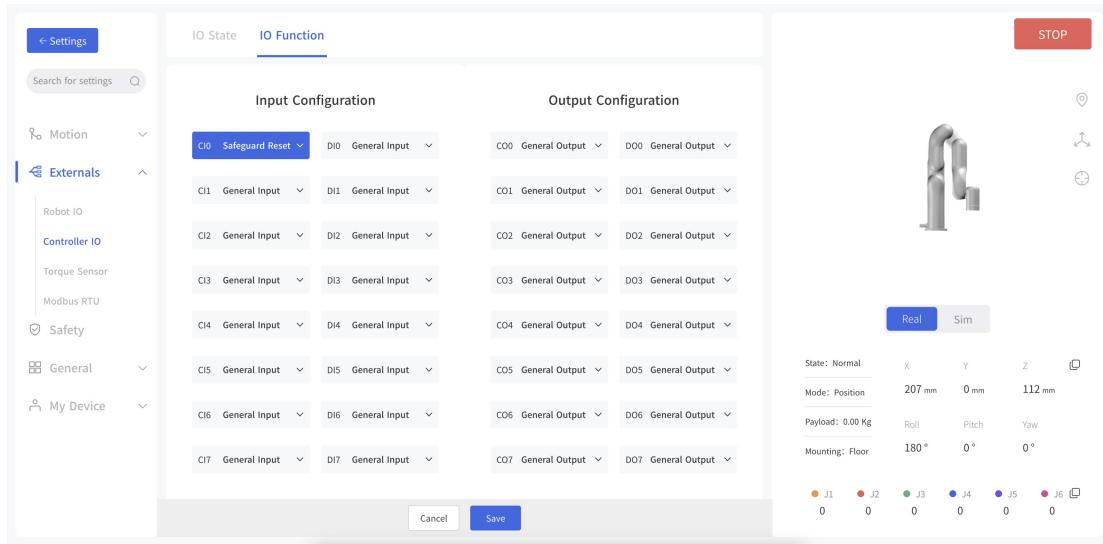
If you use a protective interface to interact with the light curtain, you need to reset from outside the safety zone. The reset button must be a two-channel button. In the example shown below, the I/O of the reset configuration is “CI0” . (the corresponding configuration must also be done in UFactory Studio)



How to realize the protection reset function with reset button:

1. Configure “CI0” as the safeguard reset in UFactory studio. The specific steps are as follows:

Enter “Settings” – “Externals” –“Controller I/O” – “I/O Function” – Configure CI0 as safeguard reset –“Save”.



2. If xArm needs to resume motion, connect SI0 and SI1 to GND, and trigger the motion of xArm by connecting CIO0 to GND; if xArm needs to pause the motion, disconnect SI0 and SI1 from GND.

Note:

DI0-DI7 are not equipped with the following three functions: stop moving, safeguard reset, and reduced mode.

### 2.5.3. General Digital I/O Function

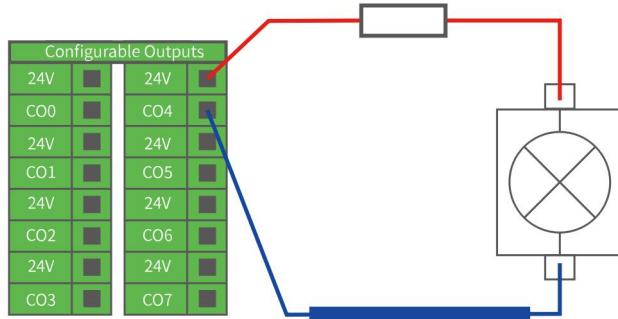
#### 2.5.3.1. Configurable Output

The digital output is implemented in the form of NPN. When the digital output is enabled, the corresponding connector will be driven to GND. When the digital output is disabled, the corresponding connector will be open (OC/OD).

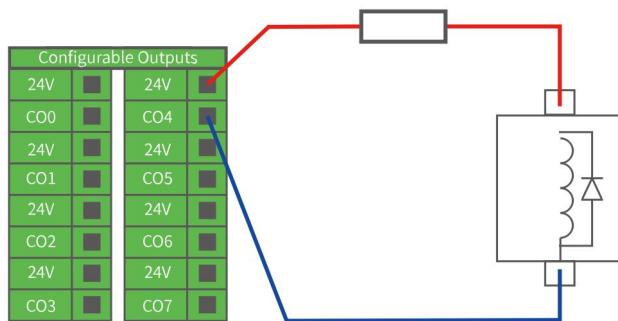
Users must follow the electrical specifications set in section 2.4.1 ‘universal specification’ .

The following example shows how to use the digital output, as the

internal output is an open-drain (OD) output, so you need to connect the resistor to the power supply according to the load. The resistance and power of the resistor depend on the specific use.



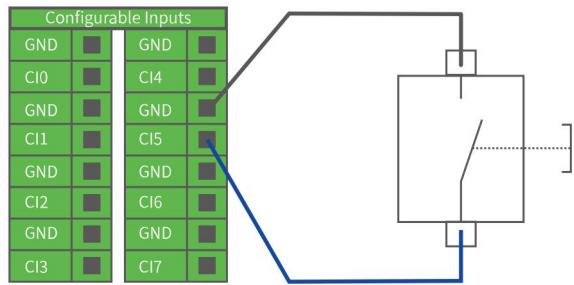
Note: It is highly recommended to use a protection diode for inductive loads as shown below.



### 2.5.3.2. Configurable Input

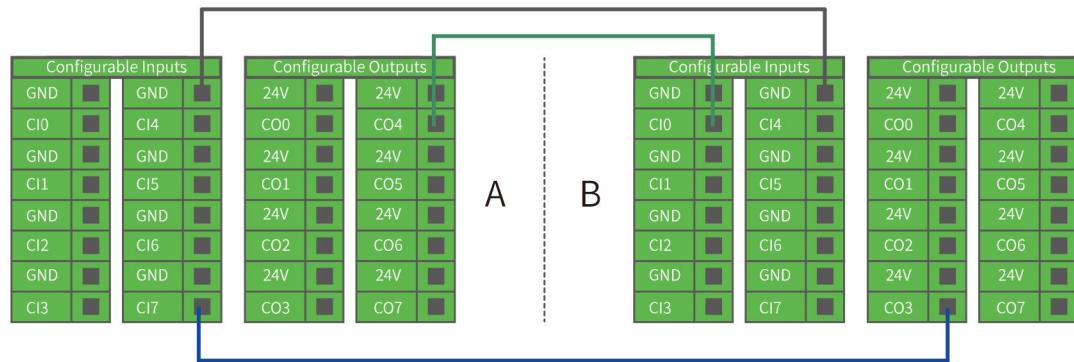
The digital input is implemented in the form of a weak pull-up resistor. This means that the reading of the floating input is always high.

Users must follow the electrical specifications set in the 2.4.1 ‘universal specification’. This example shows how a simple button is connected to a digital input.



### 2.5.3.3. Communicate with other Machines or PLCs

If general GND (0V) is established and the machine uses open-drain output technology, digital I/O and other can be used device communication, see the figure below.



### 2.5.4. General Analog I/O

This type of interface can be used to set or measure voltage (0–10V) going into or out of other devices.

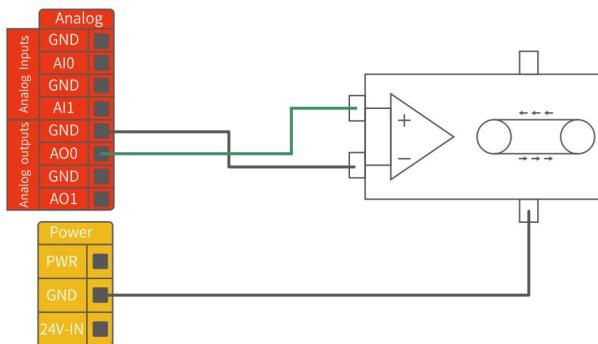
For the highest accuracy, the following instructions are recommended:

- Use the GND terminal closest to this I/O.
- The device and Control box use the same ground (GND). The analog I/O is not isolated from the control box.
- Use shielded cables or twisted pairs. Connect the shield to the “GND” terminal on the “Power” section.

Terminal	Parameter	Min. Value	Typical Value	Max. Value	Unit
Analog Input under Voltage Mode					
[AIx - AG]	Voltage	0	-	10	V
[AIx - AG]	Resistance	-	10k	-	Ω
[AIx - AG]	Resolution	-	12	12	bit
Analog Output under Voltage Mode					
[AOx - AG]	Voltage	0	-	10	V
[AOx - AG]	Current	0	-	20	mA
[AOx - AG]	Resistance	-	100k	-	Ω
[AOx - AG]	Resolution	-	12	-	bit

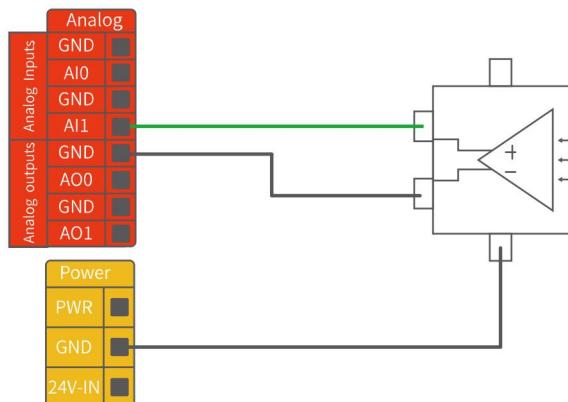
#### 2. 5. 4. 1. Analog Output

The following example shows how to use the analog speed control input to control the conveyor belt. (Connect to A00 or A01)



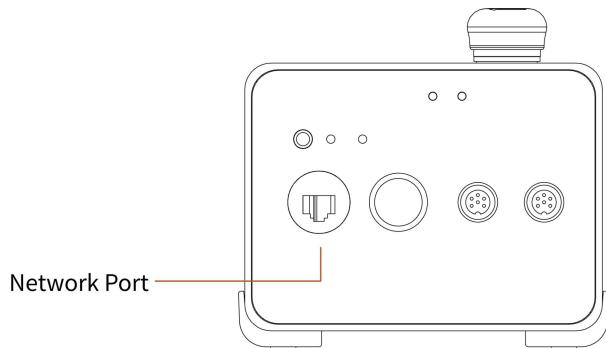
#### 2. 5. 4. 2. Analog Input

The following example shows how to connect an analog sensor. (Connect to AI0 or AI1)



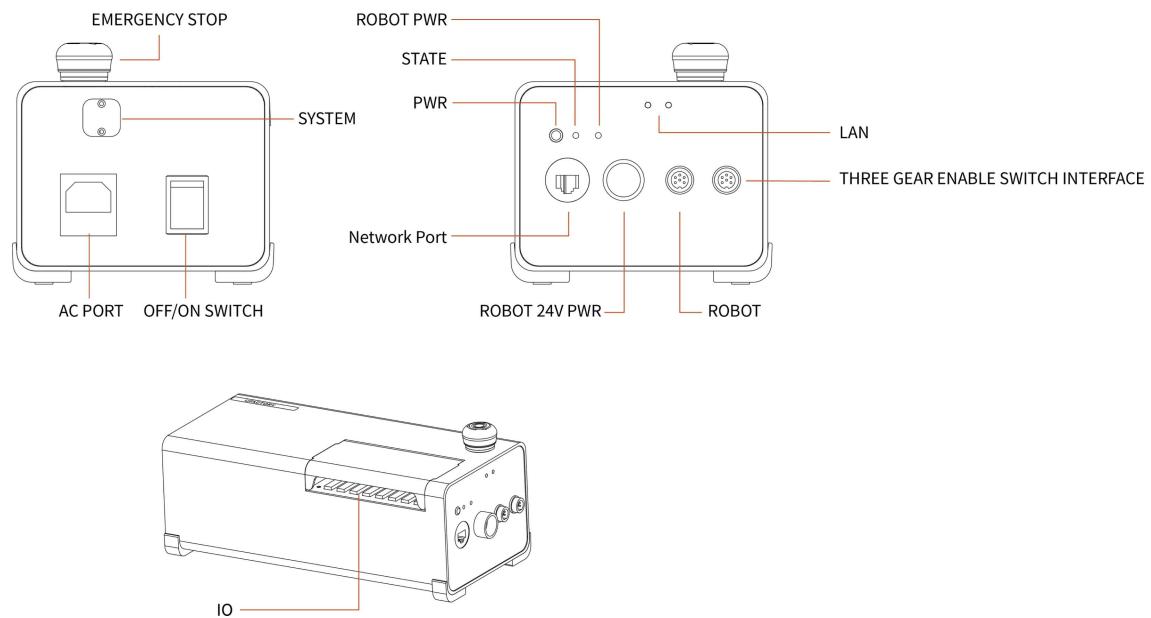
## 2. 6. Communication Interface

The Control Box provides Ethernet interface, as shown in the figure below.



## 2. 7. Ethernet TCP/IP

The control box provides a gigabit Ethernet interface.



### Ethernet connection steps:

- The control box and the computer are connected via Ethernet. One end of the network cable is connected to the network interface of the control box, and the other end is connected to the computer or

LAN network interface. If the connection is successful, the network port indicator blinks frequently.

The default network segment IP address of the control box is 192.168.1.\*(2~254). For a specific IP address, please check the control box label. When communicating with the robotic arm, the IP address of the computer should be in the same network segment with the IP address of the control box.

**Note:**

To connect with Ethernet, please check if the computer's IP address is 192.168.1.\*, check if the network proxy is enabled, and check if the robotic arm's IP address conflicts with that of other devices in the LAN. Please change the computer IP address to the same network segment and close the computer's network proxy. To test whether the computer can communicate with the robotic arm, open the command terminal and input ‘ping 192.168.1.\* (the IP address of the robotic arm)’ . If the ping is working, the communication between the computer and the robotic arm is successful.

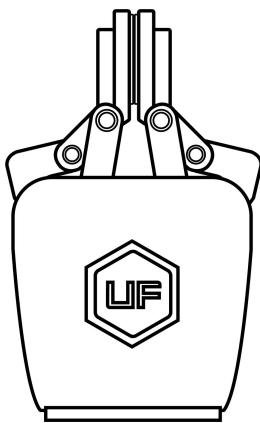
### 3. End-Effector

#### 3. 1. Gripper

The gripper is the end-effector of the robotic arm, which can grasp objects dynamically.

The value range of the gripper opening and closing is: -10 to 850. The larger the value, the greater the stroke of the gripper, meaning the smaller the value, the smaller the stroke of the gripper. If the clamping is not tight, a negative value can be set until it is tightened.

The speed of the gripper should be in 1000–5000. If a speed less than 1000 was set, the gripper may not work. The speed of the gripper opening needs to be greater than or equal to the speed of the gripper closing.



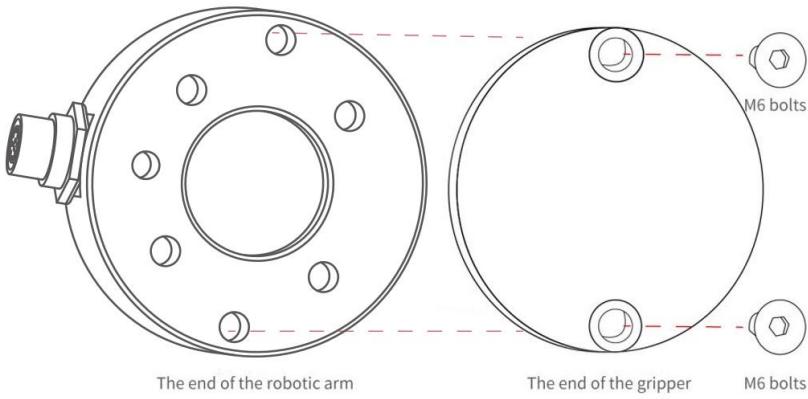
##### 3. 1. 1. Gripper Installation

**Installation of gripper:**

1. Move the robotic arm to a safe position. Avoid collision with the robotic arm mounting surface or other equipment;
2. Power off the robotic arm by pressing the emergency stop button on

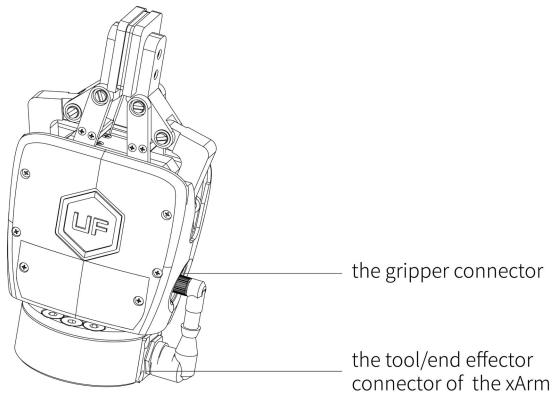
the control box;

3. Fix the gripper on the end of the robotic arm with 2 M6 bolts;
4. Connect the robotic arm and the gripper with the gripper connection cable;



**Note:**

1. When wiring the gripper connection cable, be sure to power off the robotic arm, to set the emergency stop button in the pressed state, and to ensure that power indicator of the robotic arm is off, as to avoid robotic arm failure caused by hot-plugging;
2. Due to limited length of the gripper connection cable, the gripper connector and the tool/end effector connector must be on the same side;
3. When connecting the gripper and the robotic arm, be sure to align the positioning holes at the ends of the gripper and the robotic arm. The male pins of the connecting cable are relatively thin, please be careful to avoid bending the male pins during disassembly.



### 3.1.2. The Flow of Gripper Movement

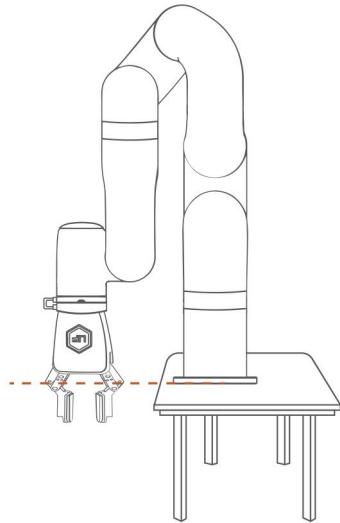
1. Enable the gripper.
2. Send out a position for clamping.
3. The current range of value:  $-10 \sim 850$ .

### 3.1.3. Precautions



DANGER

1. When the robotic arm is in the zero position, the gripper will exceed the installation surface. Please adjust the robotic arm to a posture suitable for installing the gripper during installation.
2. When a robotic arm equipped with a gripper is used for trajectory planning, it is necessary to perform a safety assessment on whether to return to the zero-point or whether the operation can be performed and to avoid collisions.



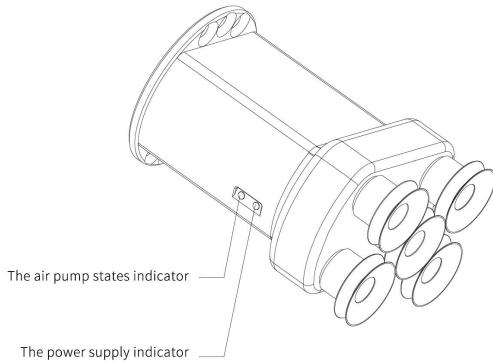
The gripper of the robotic arm in the zero position will exceed the mounting surface.

**Note:**

For detailed instructions on the xArm gripper, please refer to the xArm gripper user manual, download link:

<https://www.ufactory.cc/pages/download-xarm>

### 3. 2. Vacuum Gripper



The vacuum gripper can dynamically suck the smooth plane object with payload  $\leq 5\text{kg}$ . The vacuum gripper is equipped with 5 suction cups, which can be partially selected for use according to the size of the object surface, and the unused suction cup needs to be sealed.

**Note:**

If the surface of the object is not smooth, there will be air leakage from the suction cup which makes the object fail to be sucked up firmly.

Indicator status: When the vacuum gripper is powered on, the power supply indicator near the vacuum gripper is constantly red. When the vacuum gripper is on, the IO status indicator is constantly green.

### 3.2.1. Vacuum Gripper Installation

**Installation of vacuum gripper:**

1. Move the robotic arm to a safe position. Avoid collision with the robotic arm mounting surface or other equipment;
2. Power off the robotic arm by pressing the emergency stop button on the control box;
3. Fix the vacuum gripper on the end of the robotic arm with 2 M6 bolts;
4. Connect the robotic arm and the vacuum gripper with the vacuum gripper connection cable.

**Note:**

1. When turning on the vacuum gripper connection cable, be sure to power off the robotic arm, to set the emergency stop button in the pressed state, and to ensure that power indicator of the robotic arm is off, as to avoid robotic arm failure caused by hot-plugging;
2. Due to the length limitation of the vacuum gripper connection cable, the vacuum gripper interface and the tool IO interface must be in the same direction;

3. When connecting the vacuum gripper and the robotic arm, be sure to align the positioning holes on the two ends of the interface. The male pins of the connecting cable are relatively thin to avoid bending the male pins during disassembly.

### 3.2.2. Turn On/Off Vacuum Gripper

Example:

Blockly:



Python-SDK:

```
arm.set_vacuum_gripper(True, wait=False) #Turn on vacuum gripper  
arm.set_vacuum_gripper(False, wait=False) #Turn off vacuum gripper
```

Note:

1. Python-SDK and UFactory studio provide wrapped functions that can be called to turn on/off the vacuum gripper.
2. For detailed instructions on the xArm vacuum gripper, please refer to the xArm vacuum gripper user manual, download link:  
<https://www.ufactory.cc/pages/download-xarm>

# xArm User Manual-Software Section

## 1. UFactory studio

UFactory studio is a graphical user application for controlling the robotic arm. With this application, you can set parameters, move the robotic arm in live control, and create a motion trajectory by simply drag and drop the code blocks of Blockly. UFactory studio allows users to plan the motion trajectory for the robotic arm without programming skills.

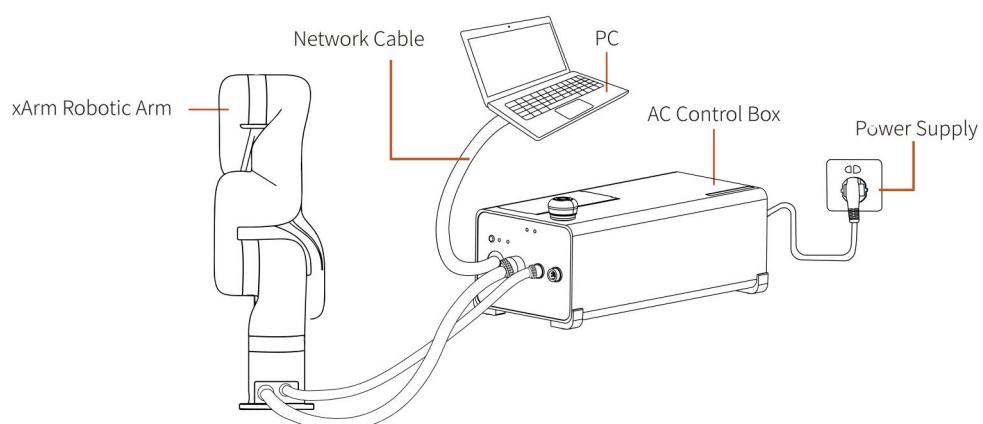
### Note:

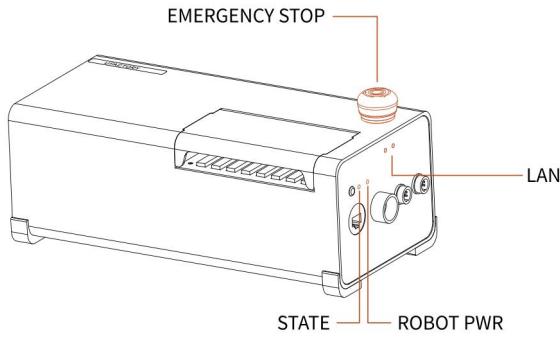
- 1) Installation systems supported by the UFactory studio client: Windows, Mac, Linux (Ubuntu16.04)
- 2) On Windows (Mac/Linux) computers, iPad, Android tablets and Raspberry Pi 4B, you can access UFactory studio through a browser.

UFactory studio is now compatible with browsers:

Google Chrome/ Firefox / Safari /Microsoft Edge (Chromium kernel)

### 1.1 Hardware Preparation





Before using UFactory studio, you must ensure that the hardware is installed correctly and all the protective measures for the workplace environment have been implemented.

1. The robotic arm is fixed on the plane; protective measures are in place within the range of motion.
2. Check if the connection between the control box and the robotic arm, power supply, and network cable is stable.
3. Check if the main power of the control box is on. If the ON/OFF light is on it means power is on.
4. Check if the control box is turned on, if the status indicator of the control box is on, it means the control box is turned on.
5. Check if the network is connected. If the network indicator in the middle of the control box flashes frequently, it means the network communication is normal.
6. Check if the robotic arm is powered and the emergency stop button is disabled. If the power indicator of the robotic arm lights up, it means the power is on.

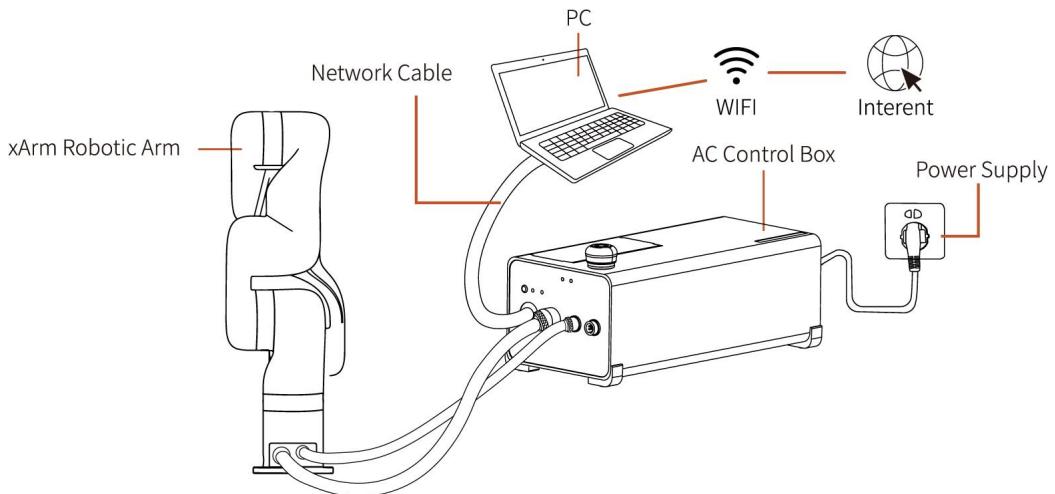
## 1.2 Connect to the Robotic Arm

### 1.2.1 The Robotic Arm Network Settings

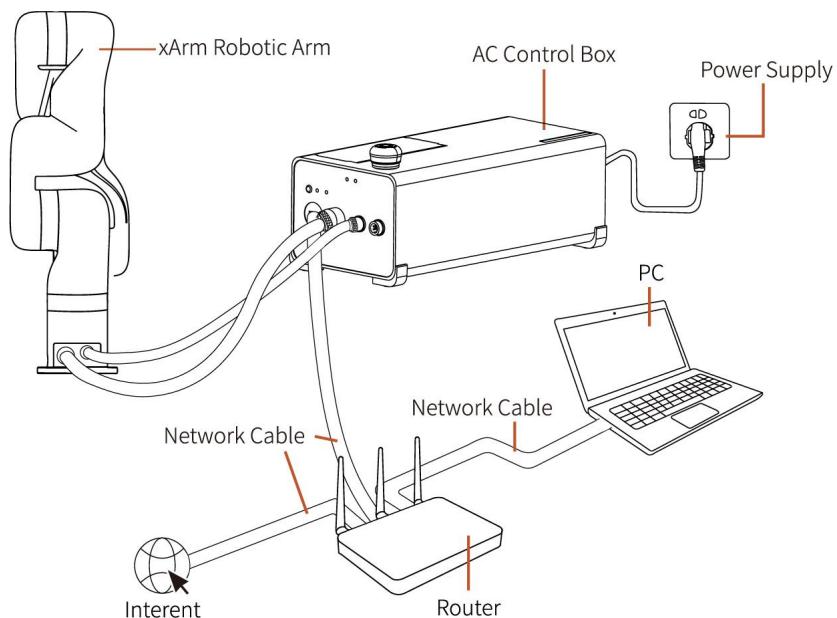
There are four ways of network settings for the robotic arm. You can choose the appropriate network setting method according to your scenario:

- (1) The control box is directly connected to the PC.

Note: Recommended connection method.

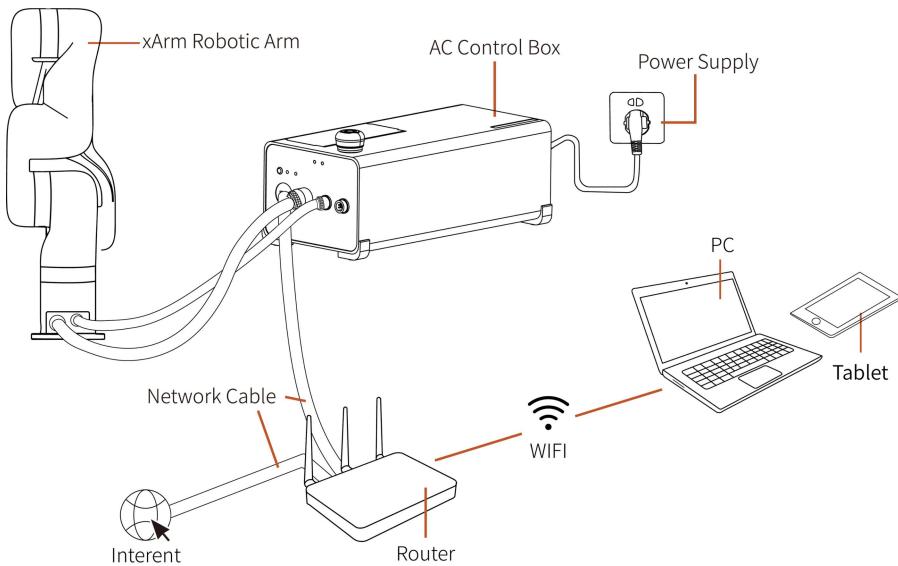


- (2) The control box, PC and router are connected by Ethernet cable.



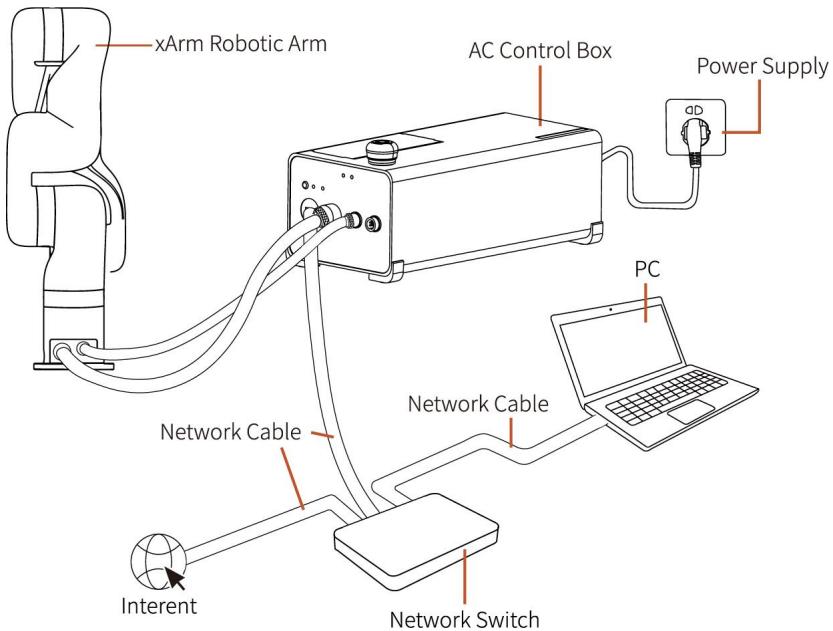
(3) PC and router are connected by wireless network, and control box and router are connected by Ethernet cable.

Note: It is not recommended because of the delay and packet loss of wireless connection.



(4) The control box, PC and network switch are connected by Ethernet

cable.



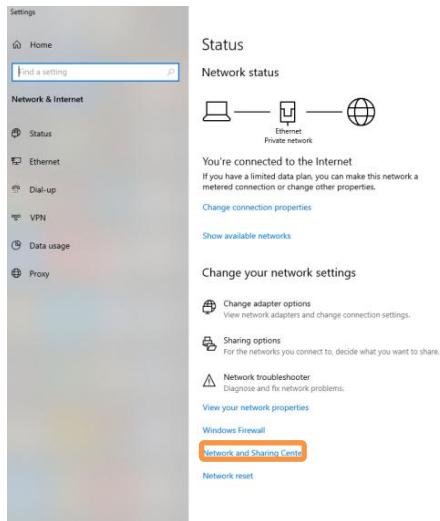
## 1.2.2 IP Configuration

Before connecting the robotic arm with UFactory studio (communication with the robotic arm), make sure that the IP address of the computer and the IP address of the control box are on the same network segment. When the control box is shipped from the factory, the default IP address is 192.168.1.xxx (The factory IP address of the device has been marked on the side of the control box). Therefore, to successfully communicate with the control box, the IPV4 network segment on the computer must be between 192.168.1.1-192.168.1.255 (cannot be the same as the IP address tail number of the control box).

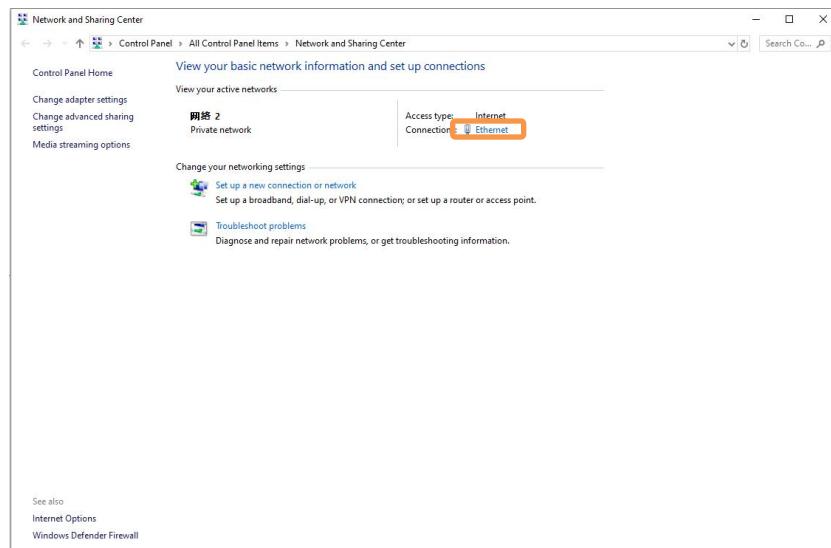
### ● IP Setting

Network and Sharing Center → Ethernet → Properties → IPV4

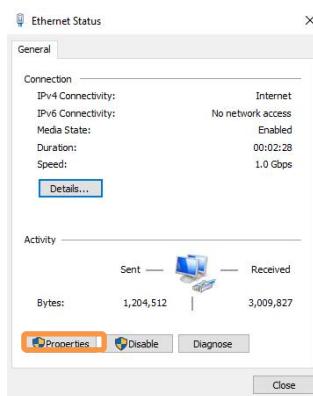
Step1: Open the “Network and Sharing Center”



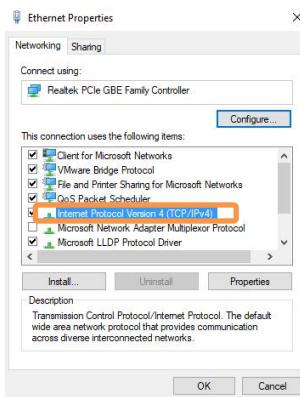
Step2: Open the “Ethernet”



Step3: Open the “Properties”

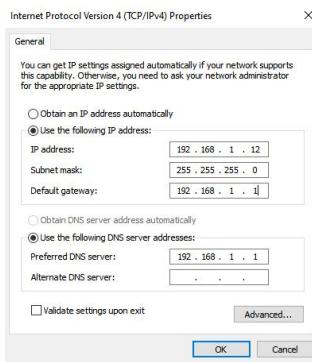


Step4: Open the “IPV4”

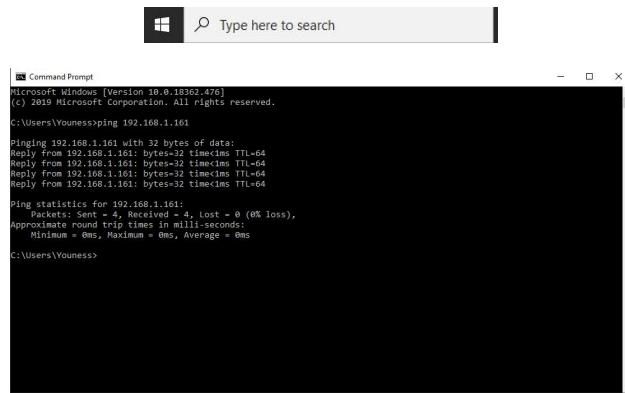


Step5:

Then check whether the computer IP is within 192.168.1.1–192.168.1.255 (the tail number should be 1 to 255, and can not be the same as the IP address of the control box). If not, please modify the computer's IP.



Step6:



### 1.2.3 Connect to the Robotic Arm

There are the following two ways to communicate with the robotic arm.

1. If you access UFactory studio software, you can communicate with the robotic arm through the following steps:

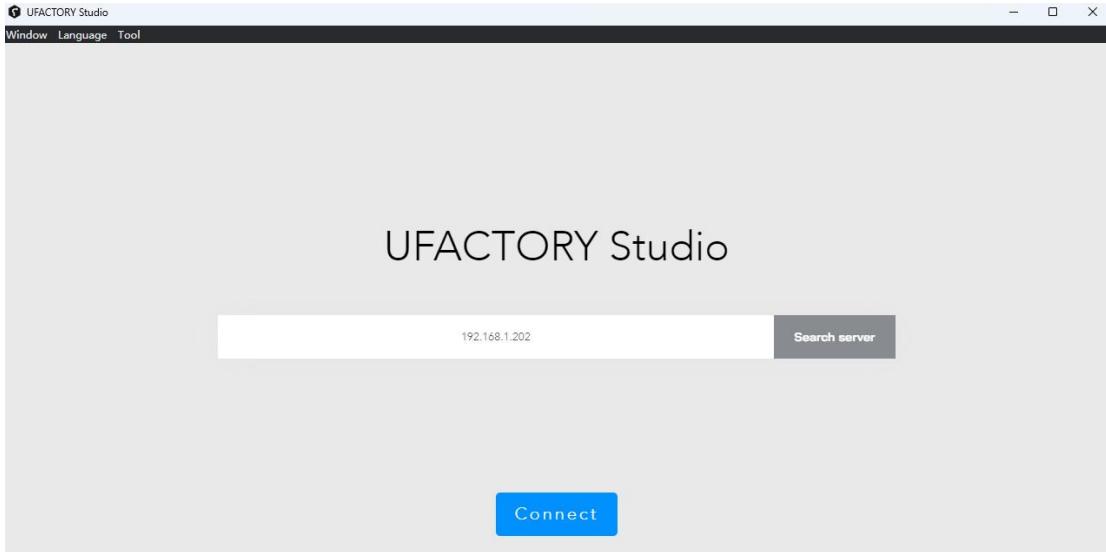
(1) Download UFactory studio

UFactory studio download address:

<https://www.ufactory.cc/ufactory-studio/>

(2) Install UFactory studio software

(3) Open the UFactory studio software, and enter the IP address of the control box in the search box (the default IP address of the device has been marked on the side of the control box)

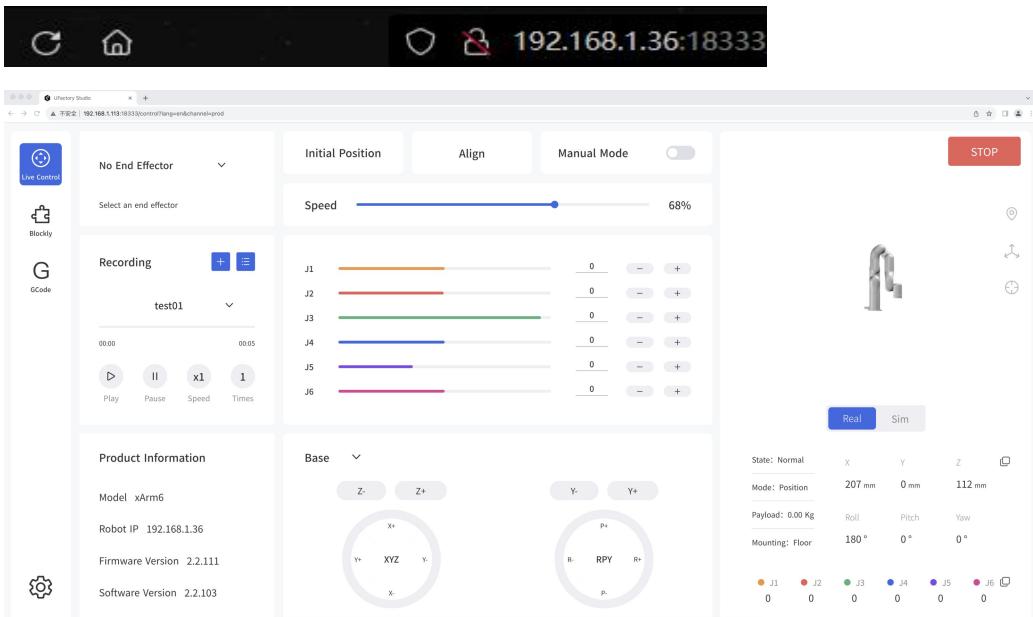


2. If you use a browser to access UFactory studio, you can communicate with the robot through the following steps:

(1) Open the browser

(2) Enter in the search bar: the IP address of the control box:  
18333

For example, if the IP address of the control box is 192.168.1.202, enter 192.168.1.202:18333 in the search bar to access UFactory studio.



## 1.2.4 Return to the Search Interface

Window Language Tool Help

PC: Click 【Tool】 - 【Search】 to return to the search interface.

## 1.3 UFactory studio Introduction

UFactory studio mainly consists of 3 main functional modules:



Live Control

**Live Control:** Used to control the position of the robot arm in real time, adjust the motion attitude of the robot arm, end tool, and track recording.



Blockly

**Blockly:** A graphical programming tool that allows users to achieve programming for the control of the robotic arm , I/O, or end-effector by simply drag and drop the code blocks.



GCode

**GCode:** Entering the GCode programming page, users can use GCode to control the movement of the robotic arm.



**Settings:** It is used to set the parameters of the robot arm, to pass the settings, safety settings, system software upgrade, etc.

### 1.3.1 Toolbar

Window Language Tool Help

**Window:** To adjust its size, you can make a selection in the 【Window】 drop-down menu or adjust the size by dragging the border of the window.

**Language:** Switch language in the upper right corner of the toolbar - 【Language】 may switch between Simplified Chinese /

English.

**Tool:** 【Tools】 - 【Search】 to return to the interface of ‘search the robotic arm’ .

【Tools】 - 【Check for Updates】 to check the software updates.

**Help:** 【Help】 Use the drop-down window to download the manuals of the robotic arm, contact technical support, open forums, and visit GitHub.

## 1.4 Robotic Arm Setting

Click the 【Settings】 button on the home page to enter the robotic arm setting interface. Set the desired parameters according to the actual situation.

### 1.4.1 Motion Settings

#### 1.4.1 Parameters



##### 1.4.1.1 Joint Motion

**Acceleration:** The acceleration of joint motion. The larger the value, the less time it takes to reach the set speed. The range

is recommended to be within 20 times the maximum operating speed [20\*180° /s].

**Joint step:** Set the step length for fine adjustment of single joint rotation in Live-control.

#### 1.4.1.2 Linear Motion

**Acceleration:** The acceleration of linear motion. The larger the value, the less time it takes to reach the set speed. It is recommended to be set within 20 times the maximum speed value for a smooth trajectory.

**Position step:** Set the step length for fine cartesian position (X/Y/Z) adjustment in Live-control.

**Attitude step:** Set the step length for fine adjustment of TCP orientation in Live-control.

#### 1.4.1.3 Collision Detection:

- When the deviation of the torque detected by the joint exceeds a certain normal range during the movement of the robotic arm, the robotic arm will automatically stop to prevent the robotic arm or the operator from being injured. The collision sensitivity range is 1 to 5 levels. The larger the value is set, the higher the collision sensitivity level is, and the smaller the additional torque required for the robotic arm to trigger collision protection. If the load or installation direction is not set accurately, it may cause false alarms. During certain high loads

or high speed movements, if you confirm that the load or installation direction is set accurately, you can try to lower the collision sensitivity, but it is not recommended to lower it to less than 3.

#### 1.4.1.4 Initial Position

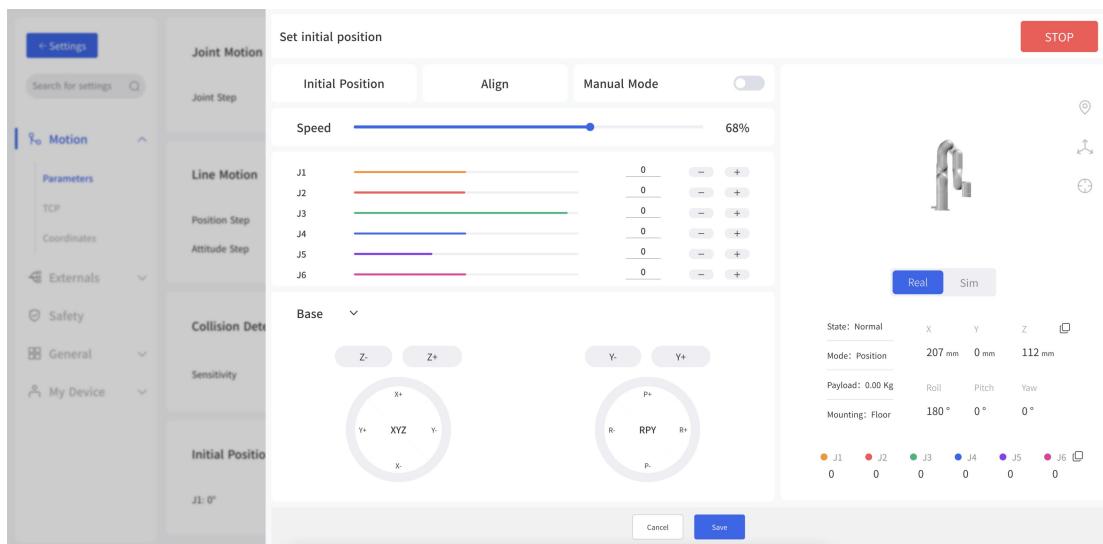
Setting the Initial Position of the robotic arm can help the user to return the robotic arm to a relatively safe position when planning the motion trajectory.

Steps for setting the initial position:

1. Click 【Settings】 button on the homepage.
2. Enter 【Motion】 , then click the 【Set】 button next to the Initial Position.
3. Set the initial position of the xArm in Live-control.

【Confirm】 : Save the changes.

【Cancel】 : Cancel the changes.



## 1.4.2 TCP Settings

Set TCP Payload and TCP Offset according to the actual situation.

### 【TCP Payload】

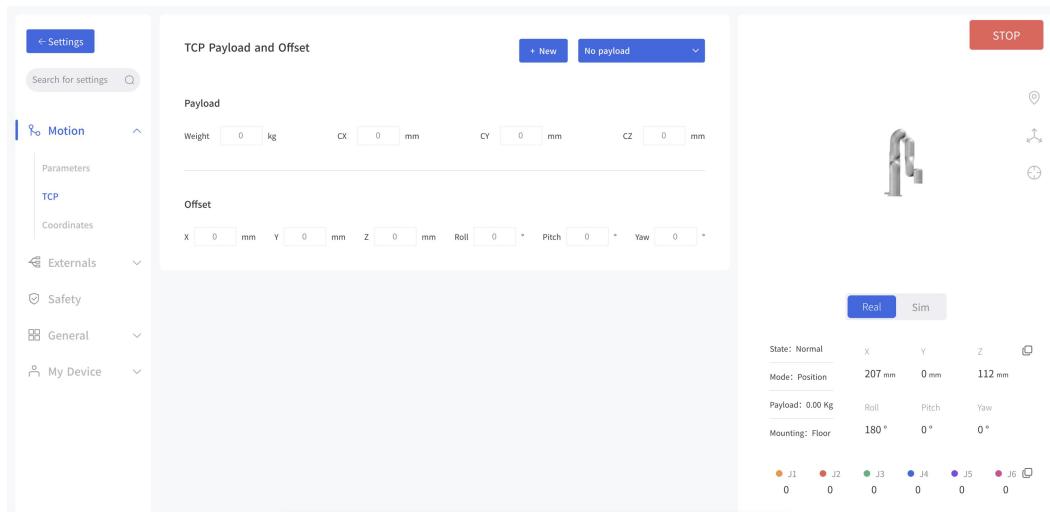
- The load weight refers to the actual mass (end-effector + object) in Kg; X/Y/Z-axis represents the position of the centre of gravity of payload in mm, this position is expressed in default TCP coordinate located at flange center (Frame B in the above figure). If there is virtually no load at the end, both TCP payload and centre of gravity must be set to 0.

### 【TCP Offset】



Setting the Tool Coordinate Offset with respect to the initial tool frame located at the center of the flange (Frame B in the above figure). The position coordinates X, Y, and Z determine the position of TCP, while Roll, Pitch, andYaw determine the orientation. When the specified value is zero, TCP coincides with the centre point of the tool output flange.

### 1.4.2.1 TCP Payload



On this page, the current payload of the robotic arm can be set and the additional TCP payload data can be recorded. The additional TCP payload data can be referenced during Blockly programming.



【Set as default】

- Set the payload data to the payload of the current robotic arm and display the current payload at the top, which is used for controlling the entire robotic arm and is related to the normal use of manual mode and collision detection.

【New】 : Create new payload data.

【Select】 : Select the payload data to be deleted in the next step.



【Delete】 : Delete the selected payload data. Note: the current default payload data cannot be deleted.

【Save】 : Save for the newly added payload record, setting the default payload, and deleting the payload record.

【Cancel】 : Cancel saving the newly added payload record, setting the default payload, or deleting the payload record.

#### 1.4.2.2 Create New TCP Payload and Offset

There are two ways to create a new TCP payload:

Manual input or Automatic identification. Manually inputting can be selected if the weight of the payload and the approximate center of gravity of the payload are known. The center of gravity of the payload is set based on the initial tool coordinates (the coordinates of point B shown in the above figure).

The current robotic arm must be mounted on a steady floor if

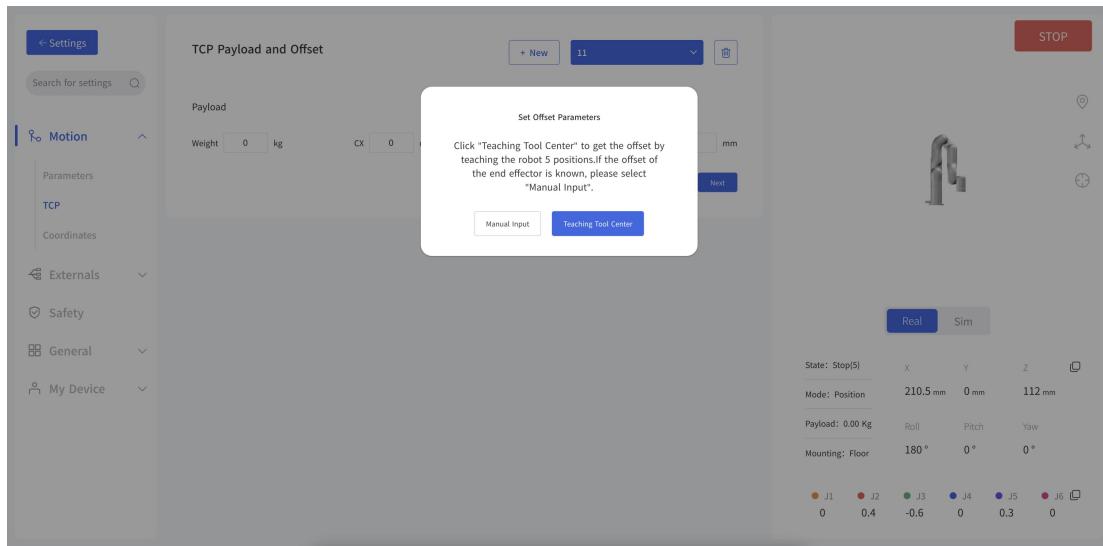
automatic identification is selected. The robotic arm needs to run a series of action commands to calculate the parameters of TCP payload. In addition, it is important to ensure the safety of equipment and personnel near the robotic arm.

Note: Once the name of the new payload has been determined, it cannot be changed.

The screenshot shows the 'Set Payload Parameters' interface. At the top, there are two buttons: 'Manual Input' (white) and 'Identification' (blue, selected). The main area is titled 'TCP Payload and Offset' with tabs for 'Payload' (selected), 'Offset', and 'Coordinates'. On the left, there's a sidebar with sections like 'Motion', 'TCP', 'Coordinates', 'Externals', 'Safety', 'General', and 'My Device'. A central dialog box is titled 'Payload Parameter Identification' with the sub-instruction: 'During the payload parameter identification process, the robotic arm will perform a series of movements, as shown in the right model. Click "Next Step" and the robotic arm will start to move.' Below the dialog are 'Back' and 'Next Step' buttons. To the right, there's a 3D model of a robotic arm and a detailed status panel. The status panel shows:

State:	Stop(5)	X:	207 mm	Y:	0 mm	Z:	112 mm
Mode:	Position	Roll:		Pitch:		Yaw:	
Payload:	0.00 Kg						
Mounting:	Floor	180°	0°	0°			
J1	0	J2	0	J3	0	J4	0
J5	0	J6	0				

After completing the TCP load settings, click 【Next】 to start the TCP offset settings. When creating a new TCP offset, there are two ways to set the new TCP offset parameters, as shown in the figure below:



## 1) Manual Input

When the TCP offset parameter of the end effector is known, you can choose to manually input its TCP offset parameter.

## 2) Teaching Tool Center

When the TCP offset parameter of the end effector is unknown, click the [Teach and Acquire] button to obtain the TCP offset parameters by teaching 5 points.

**Teaching Tool Center Point Position**

Next, move the tool center of the robotic arm to the same point from 4 different angles. Ensure that the 4 poses are diverse and not on the same plane, as shown in the figure below. If the operation is successful, the robot arm will calculate the position automatically.

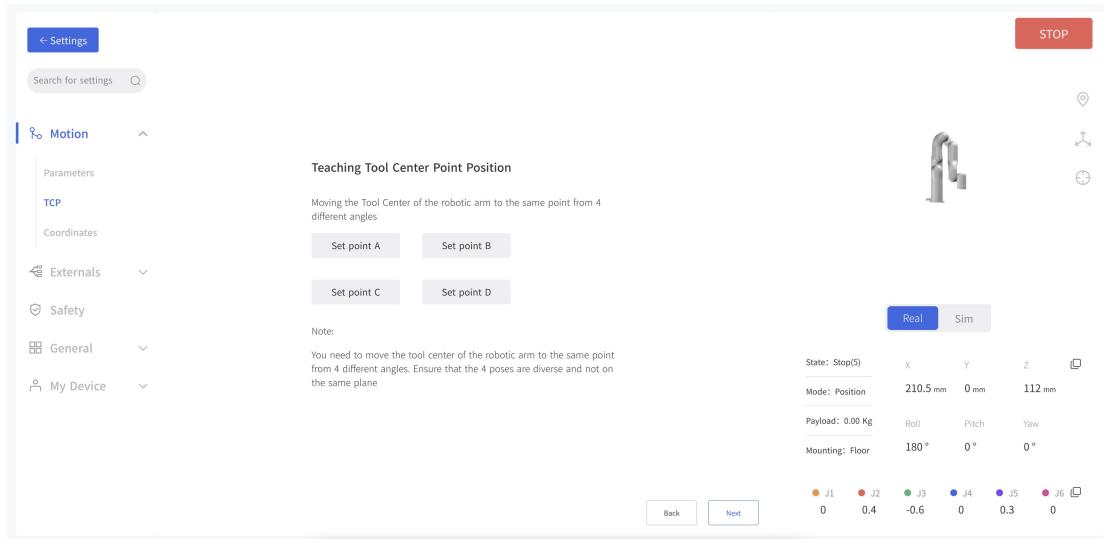
**Teaching**

Note:

1. Before teaching, please make sure that the currently offset is [0,0,0,0,0,0]
2. In the process of teaching, please do not change the offset or the user coordinate system.

I have read and understood the above information

Back
Next
Cancel
Set Point



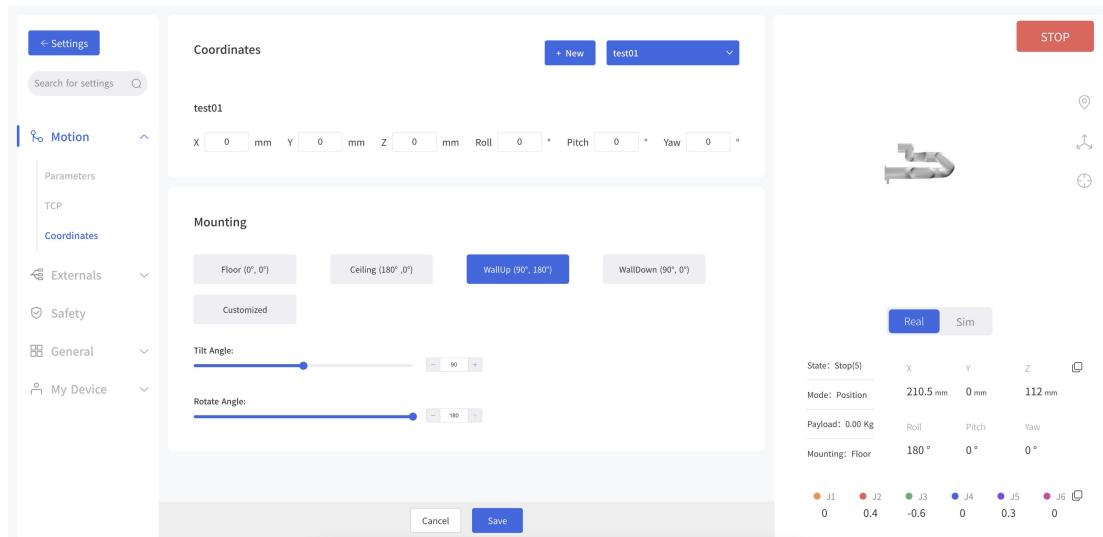
**【Delete】** : Delete the selected offset data.

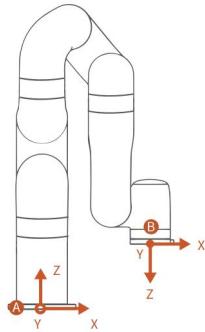
Note: the current default offset data cannot be deleted.

**【Save】** : Save for the newly added offset record, setting the default offset, and deleting the offset record.

**【Cancel】** : Cancel saving the newly added offset record, setting the default offset, and deleting the offset record.

### 1. 4. 3 Coordinates



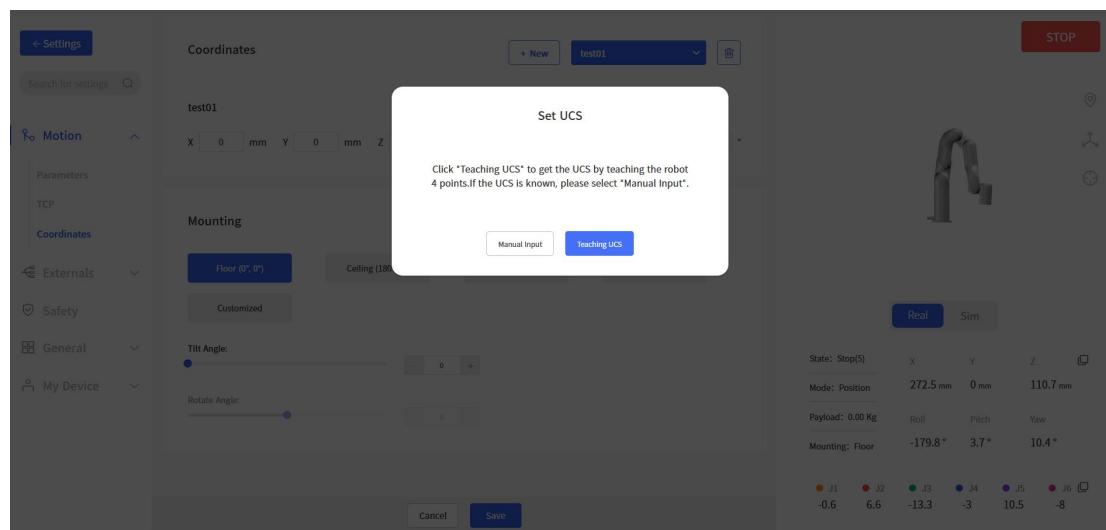


A:Base coordinate system  
B:Tool coordinate system

In this interface, the user can set the coordinate offset to customize the user coordinate system. X, Y, Z are coordinate values that are offset relative to the base coordinate system. Roll, Pitch, Yaw represents the angular values of orientation relative to the base coordinate system. After this offset setting, user coordinate system becomes the world origin instead of robot base.

#### 【New】 : Create a new user coordinate offset.

When creating a new base coordinate offset, there are two ways to set the new base coordinate offset parameters, as shown in the figure below:



#### 1) Manual Input

When the base coordinate offset parameter is known, you can choose to manually input its base coordinate offset parameter.

#### 2) Teaching UCS

When the base coordinate offset parameter is unknown, click the 【Teaching UCS】button to obtain the base coordinate offset parameters by teaching 3 points.

**Teaching Direction of UCS**

Next, you need to teach 3 points P-X-Y in sequence on the working plane. As shown in the figure below, the robot will calculate the UCS direction of the working plane based on the 3 points taught.

**Teaching UCS**

Note:

1. Before teaching UCS, please make sure that the coordinate system is [0,0,0,0,0]
2. In the process of teaching UCS, please do not change the TCP offset or the coordinate system.

I have read and understood the above information

Back      Next

Back      Next

State: Normal	X	Y	Z
Mode: Position	210.5 mm	0 mm	112 mm
Payload: 0.00 Kg	Roll	Pitch	Yaw
Mounting: Floor	180°	0°	0°

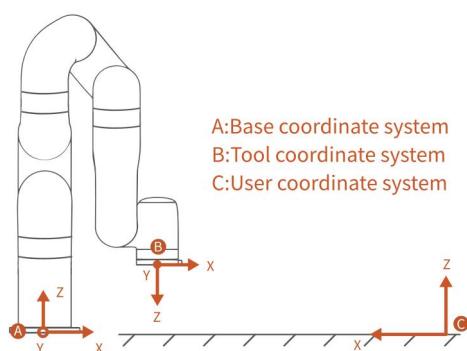
Real      Sim

Back      Next

【Cancel】: Cancel the selection.

【Save】: Save the modified data.

【Discard】: Discard the modified data.



### Example:

When expressed in coordinate system {A}:

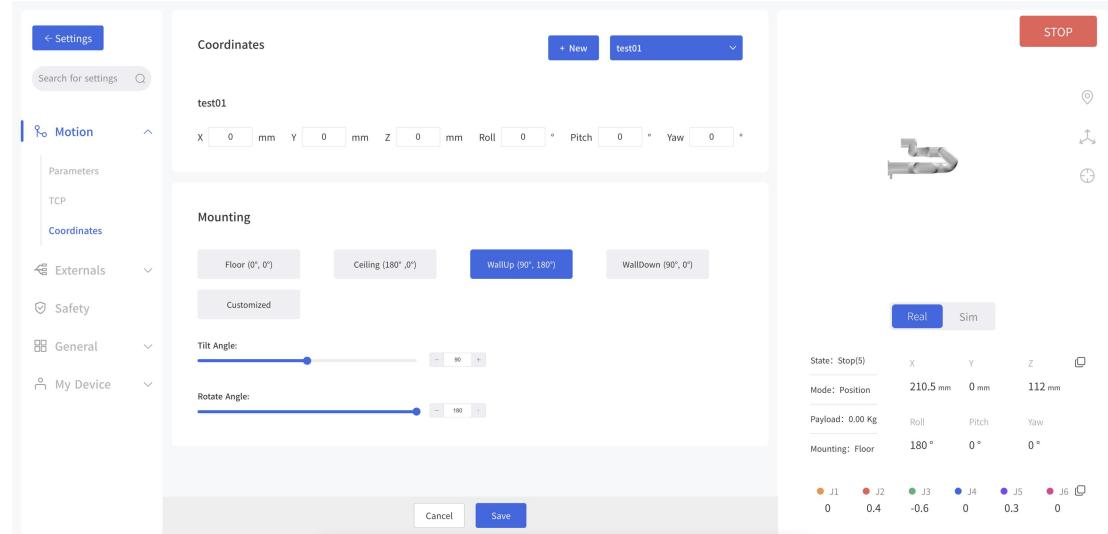
B is (207, 0, 112, 180, 0, 0),  $D_{AC} = 1000\text{mm}$ , if user want to set the world reference coordinate system to {C}, just express the position and orientation of user coordinate system {C} in coordinate system {A}.

As figure shown, the offset of the base coordinate system should be (1000, 0, 0, 0, 0, 180).

Former TCP coordinates of B (207, 0, 112, 180, 0, 0) in base coordinate system, after user coordinate system offset setting:

Becomes: B' (793, 0, 112, 180, 0, 180)

### 1.4.3.1 Mounting



Setting the mounting direction of the robotic arm is mainly to inform the control box of the current relationship between the actual mounting direction of the robotic arm and the direction of gravity.

If the mounting direction of the robotic arm is set incorrectly, the robotic arm will not be able to accurately recognize the direction of gravity, which will cause the robotic arm to frequently trigger a collision warning and stop motion, and will also result in uncontrolled motion of the robotic arm after entering manual mode.

xArm with SN of XF1300/XI1300/XS1300 and later versions, the built-in

IMU of the robot arm will detect the direction of gravity. When the deviation between the installation direction you set and the installation direction detected by the IMU exceeds 10° , the software will pop-up prompts.

#### 【Floor (0 ° , 0 ° )】

- The default method is horizontal installation, and the horizontally mounted robotic arm does not need a tilt angle and a rotation angle.

#### 【Ceiling (180 ° , 0 ° )】

- For ceiling-mount, users simply need to set the mounting method as ceiling, and it is not necessary to set the angle of rotation.

#### 【WallUp (90 ° , 180 ° )】

- Indicates that the robotic arm is wall-mounted and the end of the robotic arm is facing up.

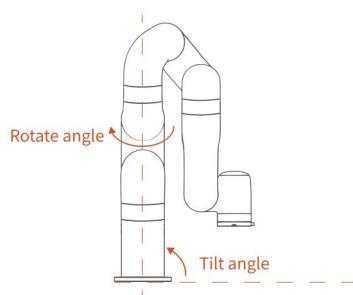
#### 【WallDown (90 ° , 0 ° )】

- Indicates that the robotic arm is wall-mounted and the end of the robotic arm is facing down.

#### 【Customized】

- Mount at other angles. For mounting at a certain angle. It is necessary to set the tilt angle and the rotation angle according to the actual situation.

How to determine the tilt angle and rotation angle?



The initial position of the robotic arm:

- On the horizontal plane, when the user is facing the robotic arm side, the initial position is on the left-hand side of the user in a downward direction.

**Tilt angle:** The initial position of the robotic arm and the base of the robotic arm to be mounted should be in a tilt angle, which ranges from 0 to 180° .

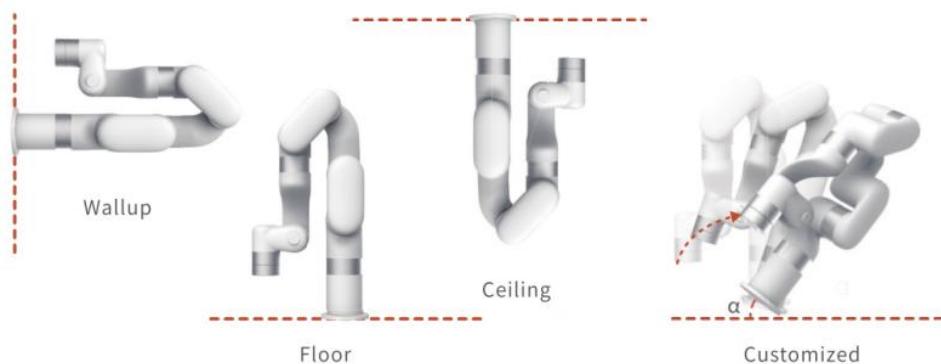
**Rotation angle:** The initial position of the robotic arm and the end direction of the robotic arm to be mounted should be used as the rotation angle.

**The method of determining the rotation angle ± direction:**

Hold it with your right hand and point your thumb in the direction of the robotic arm which is vertically mounted. The direction where your four fingers point is the positive direction and vice versa.

The range of rotation angle:

±180°



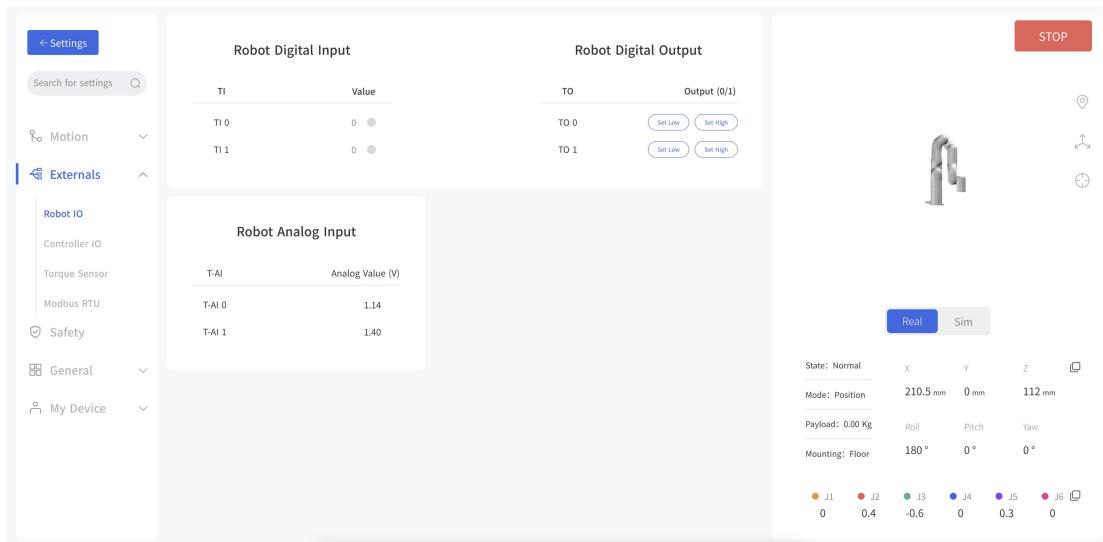
**DANGER**

Make sure the robotic arm is properly placed according to the actual use.

Must be mounted on a sturdy, shock-resistant surface to avoid the risk of rollover of the robotic arm.

## 1.4.2 Externals

### 1.4.2.1 Robot IO

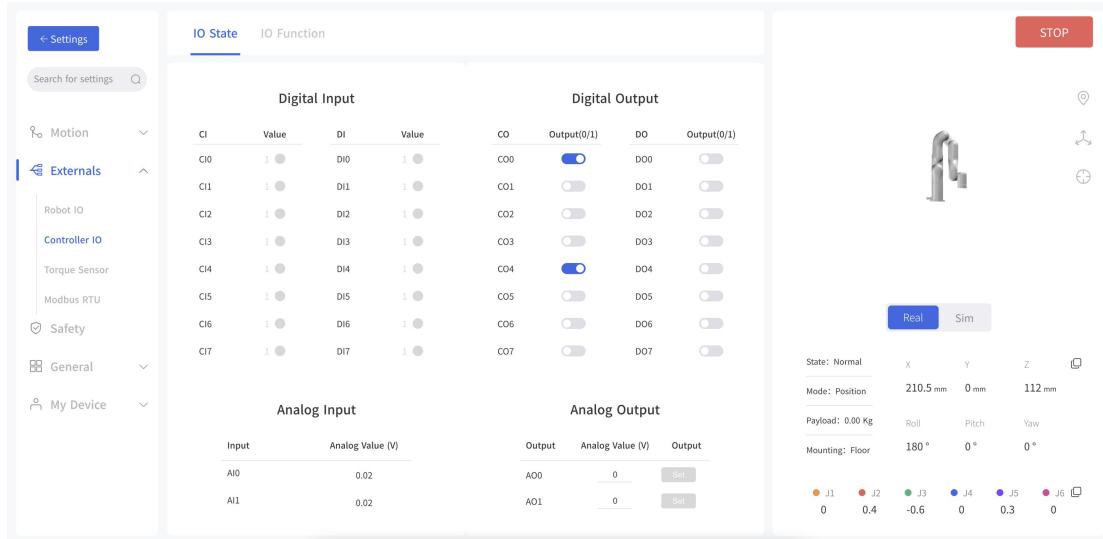


Through this page, you can monitor the digital input and analog input status of the external device to the robot arm, and set the digital output at the end of the robot arm.

### 1.4.2.2 Controller IO

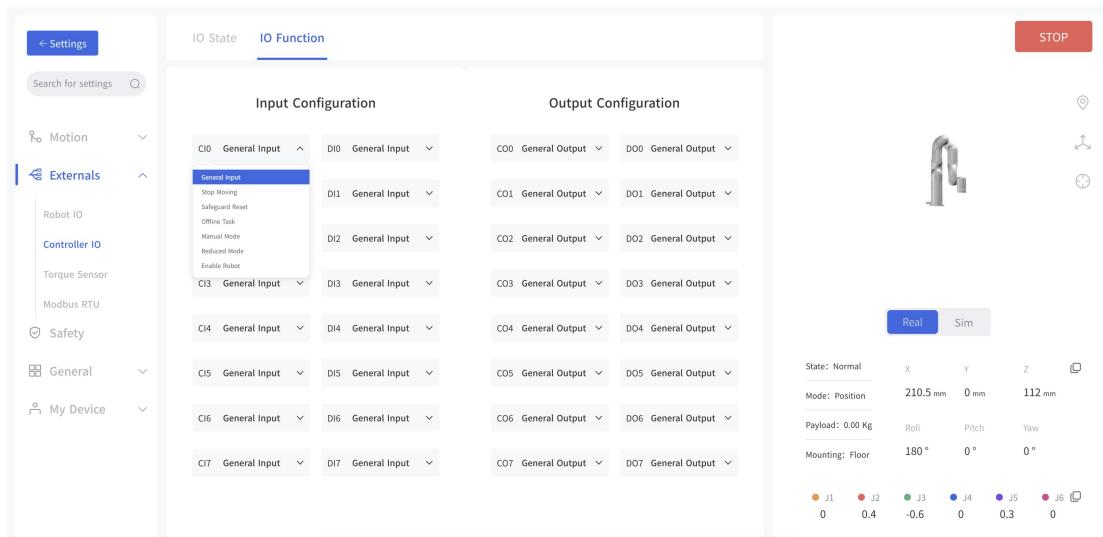
The control box of the robotic arm is equipped with 32 digital input and output signals, which can be set in the Blockly project and SDK only when IO is set to General Input / Output, otherwise the custom setting will not take effect.

## 1.4.2.2.1 IO State



In this interface, the IO input status and IO output status of the control box can be monitored, and the IO output status of the control box can be controlled by clicking the button.

## 1.4.2.2.2 IO Function



The following functions (if configured), can be triggered by low-level input signals.

**【General Input】** The user can use the IO freely in Blockly or SDK program only when the controller input is set as a general input, otherwise it will cause a function conflict. For example, if CI 0 is configured as an offline task, CI 0 should not be used in any program.

**【Stop Moving】** Trigger IO, the robotic arm stops moving.

**【Safeguard Reset】** Trigger IO to resume the motion of the robotic arm in the protection stop state. Should work with SI, refer to [2.4.2.5. Protective Stop with Reset Button](#)

**【Offline Task】** Offline Task can add multiple Blockly to be triggered through I/O. As shown in the figure above, CI0 is set as Offline Task and a Blockly project is added. Click **【Add】** to add a Blockly project, and click  to delete the project.

**【Manual Mode】** When set as Manual Mode, the robotic arm can be dragged freely when the input signal remains low level.

**【Reduced Mode】** The IO is triggered and the robotic arm enters the reduced mode.

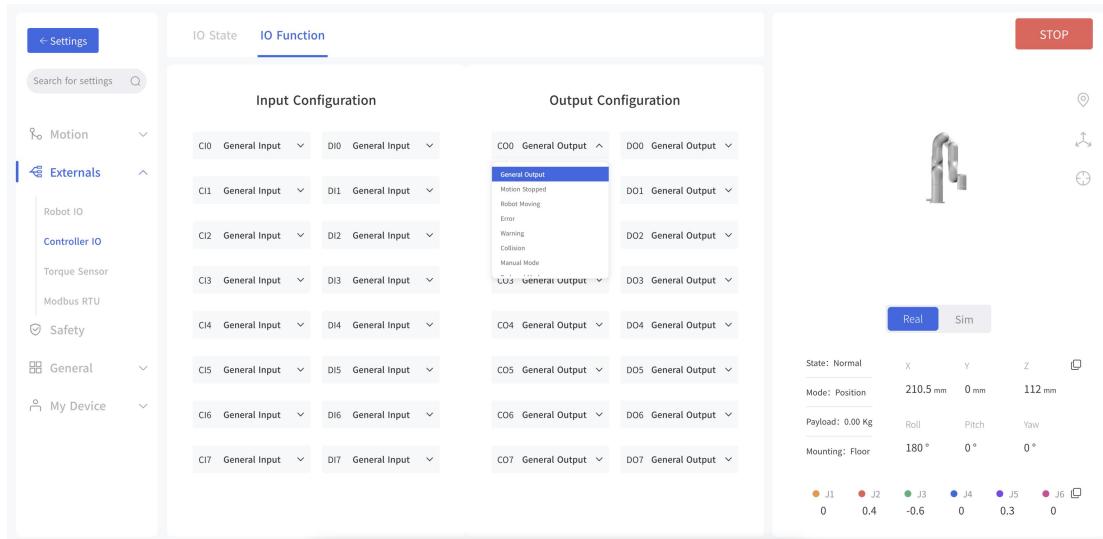
**【Enable Robot】** Enable the robotic arm by triggering IO.

**【Save】** Save the changes.

**【Cancel】** Discard the changes.

Note:

DIO-DI7 are not equipped with the following three functions: stop moving, safeguard reset, and reduced mode.



The below functions can be configured for each output.

**【General Output】** The user can use the IO freely in Blockly or SDK program only when the controller output is set as general output, otherwise it will cause function conflict. For example, if C0 0 is configured as motion stopped, C0 0 should not be used in any program.

**【Motion Stopped】** The system enters an emergency stop state and outputs a high signal.

The actions that conform to the emergency stop are:

- When the Emergency Stop button of the control box is pressed, the power supply of the robotic arm is cut off.
- Enter emergency stop via CI.
- Stop button of UFactory studio and Emergency stop code block of Blockly.
- Emergency stop API of SDK.

**【Robot Moving】** When the robotic arm is moving, the output is high.

**【Error】** When the robotic arm reports an error, the output is high.

**【Warning】** When the robotic arm issues a warning, the output is high.

**【Collision】** When the robotic arm reports an error of collision, the output is high.

**【Manual mode】** When the Manual Mode is turned on, the output is high.

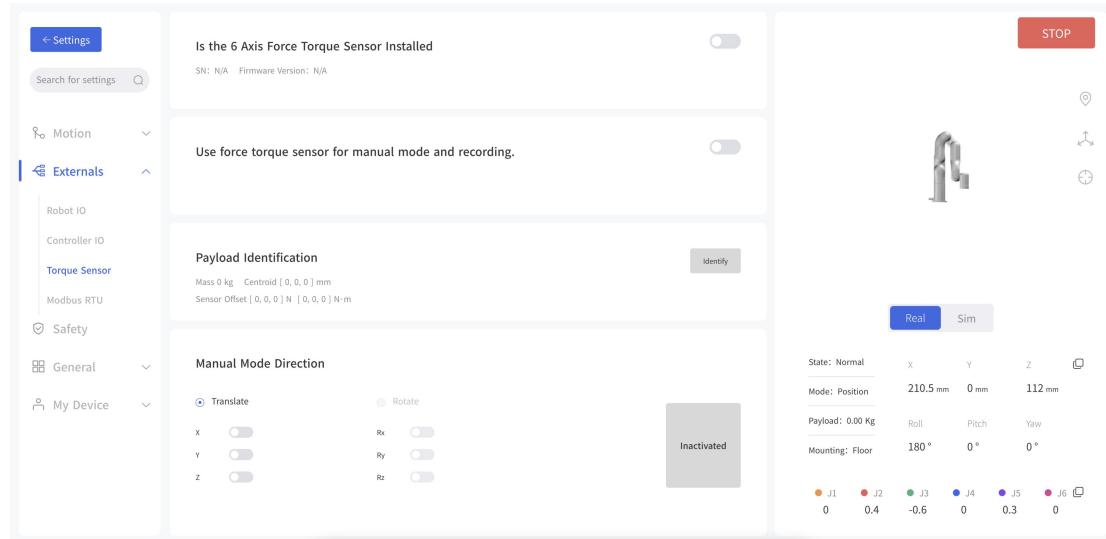
**【Offline task running】** When the robotic arm is running the offline projects, the output is high.

**【Reduced Mode】** When the reduced mode is turned on, the output is high.

**【Robot Enabled】** When the robot is enabled, the output is high.

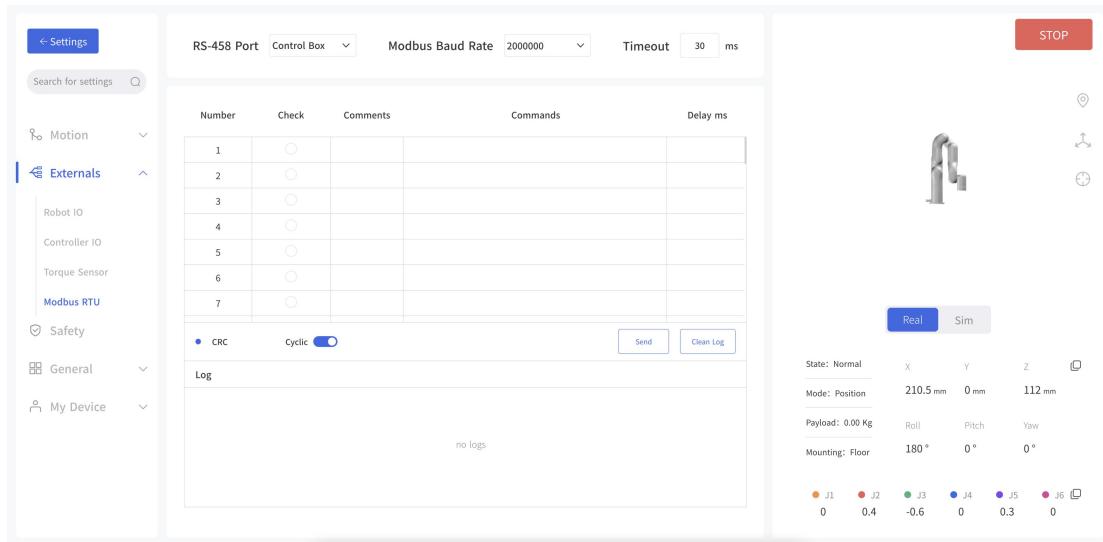
**【Emergency Stop is Pressed】** When the emergency stop button on the robotic arm control box is pressed, the output is high.

### 1.4.2.3 Torque Sensor



This page allows you to do load recognition of the torque sensor and set the manual mode direction of the torque sensor.

#### 1.4.2.4 Modbus RTU

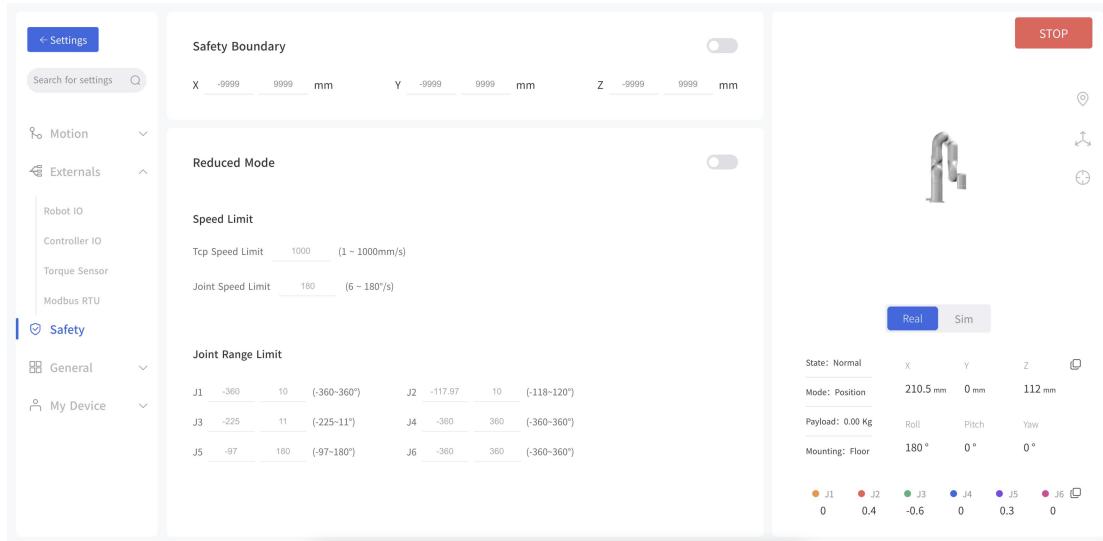


In the Modbus RTU interface, the user can send commands to control the robot gripper and get the position information of the gripper.

1. Selects the robot arm Modbus or control box Modbus.
2. Sets the baud rate, the default baud rate is 2000000.
3. Enter commands in the "Commands" box, for example:  
0x08, 0x03, 0x07, 0x02, 0x00, 0x02, note that the program will do CRC checksum automatically.
4. Click Send and you can see the sent and received information in the debug box on the left.

If you want to send in a loop, you need to set the delay time, turn on the loop function and click send.

### 1.4.3 Safety Settings



#### Safety Boundary

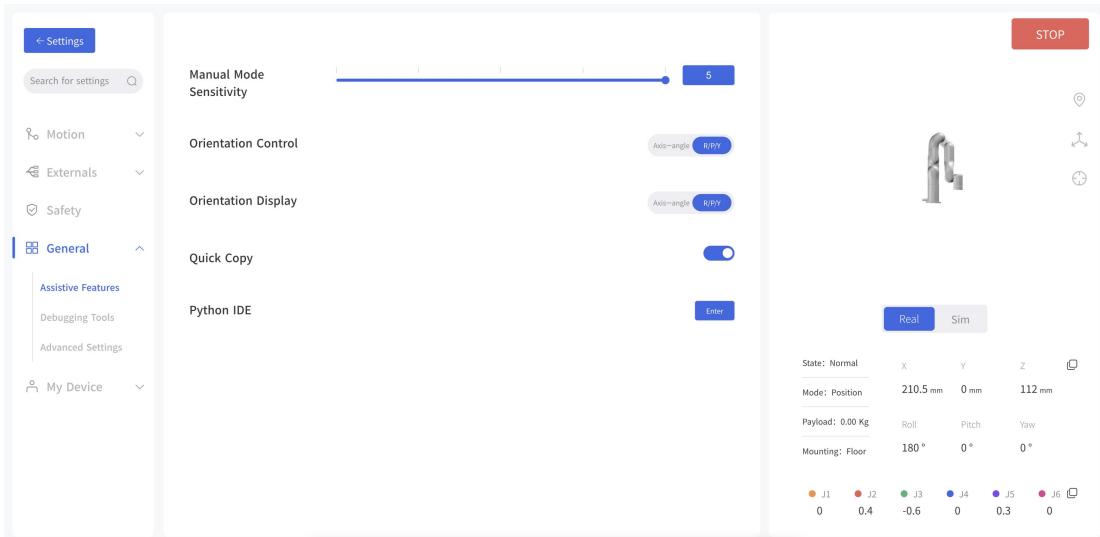
- When this mode is turned on, the working range of the robotic arm in Cartesian space can be limited. If the tool center point (TCP) of the robotic arm exceeds the set safety boundary, the robotic arm will stop moving. The user can then adjust the robotic arm back into the restricted space.

#### Reduced Mode

- When this mode is turned on, the maximum linear speed, maximum joint speed, and joint range of the robotic arm in Cartesian space will be limited.

## 1.4.4 General

### 1.4.4.1 Assistive Features



#### 【Orientation Control】

xArm supports adjusting the rotation of the robot arm through the axis-angle and R/P/Y. Generally, it is recommended to use the axis-angle since the axis-angle control is more intuitive. The choice here determines the TCP control mode of the UFactory studio Live Control page. The left side of the figure below is the axis-angle control, the button is displayed as Rx/Ry/Rz; the right side is the R/P/Y control, and the button is displayed as R/P/Y.



## 【Quick Copy】

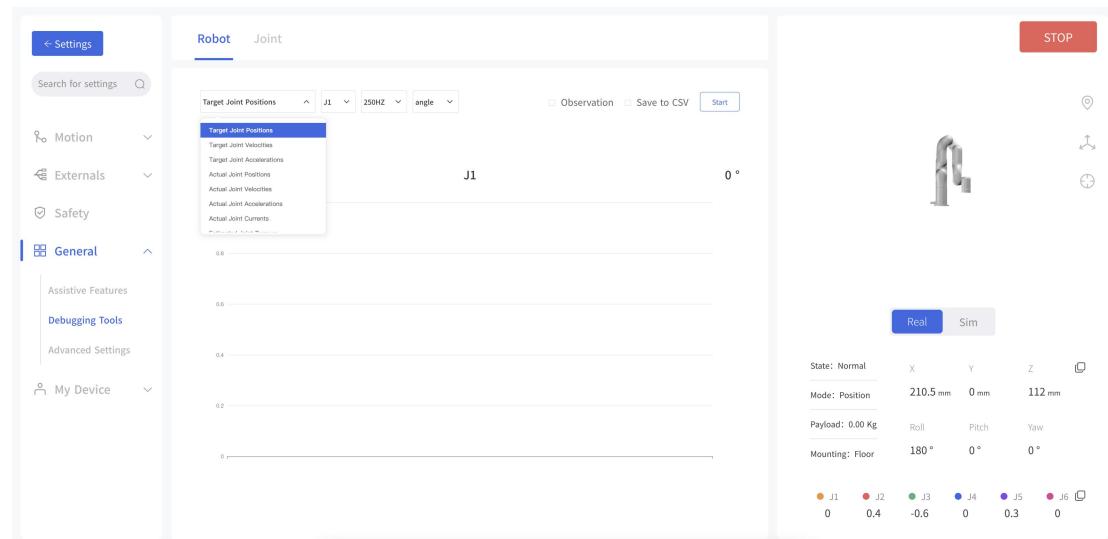
After turning on this button, the TCP coordinates and joint angle values of the xArm can be copied on the real-time control interface.

## 【Python IDE】

Enter PythonIDE

### 1.4.4.2 Debugging Tool

#### 1.4.4.2.1 Robot



Note: This function should be completed under the guidance of technical support.

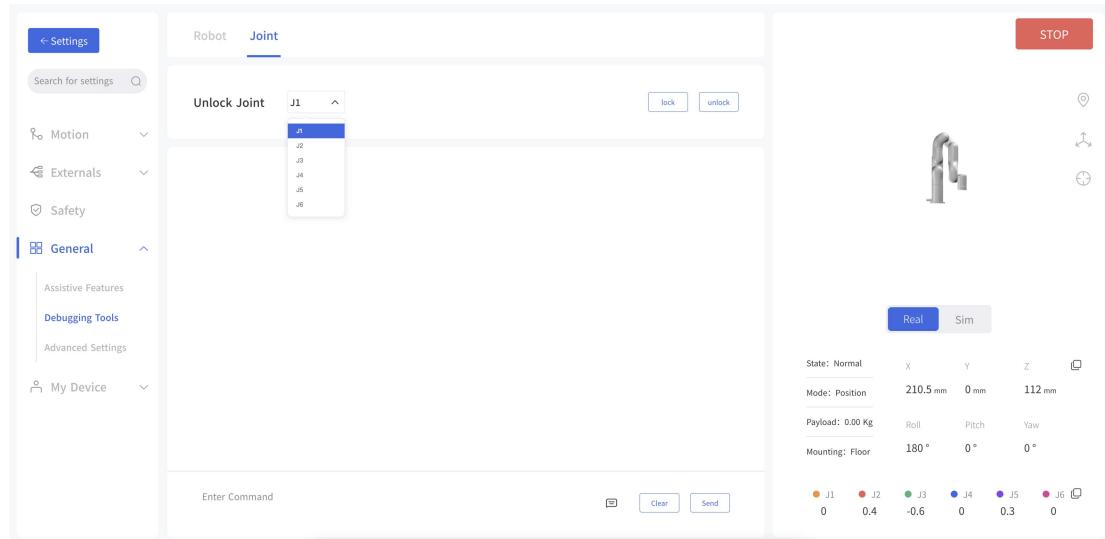
(Please contact the technical support by the email: support@ufactory.cc)

This page can help technical support remotely analyze and solve problems by observing changes in some parameter values of the robotic arm and drawing graphics to technical support. At the same time, you can also check the record CSV and download it after the observation and send it to technical support. The values that can be selected for observation include: Target Joint Positions, Target Joint Velocities, Target Joint Accelerations, Actual Joint Positions, Actual Joint Velocities, Actual Joint Accelerations, Actual Joint Currents

Target Joint Accelerations, Actual Joint Positions, Actual Joint

Velocities, Actual Joint Accelerations, Actual Joint Currents, Estimated Joint Torques, etc.

#### 1.4.4.2.2 Joint



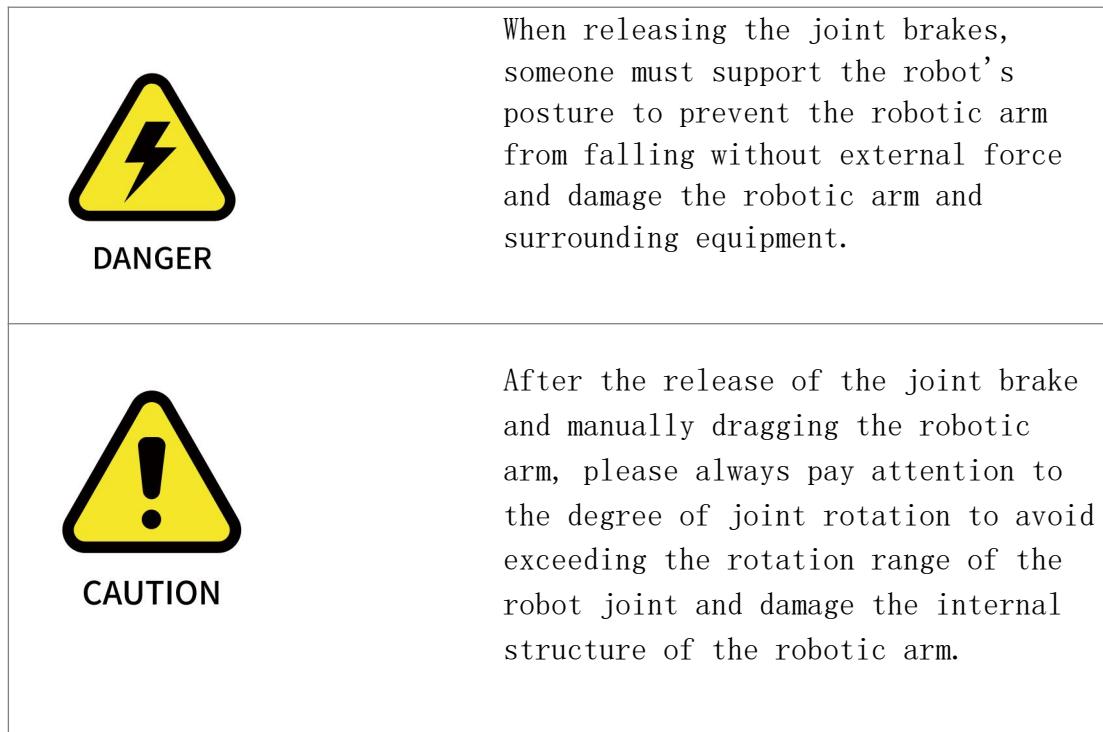
Click **【unlock】** to unlock a single joint. The unlocked joint does not have any force to provide and thence external force support is needed. At this time, the joint can be dragged by hand to rotate. After confirming the position, please re-lock all the joints manually.

Note:

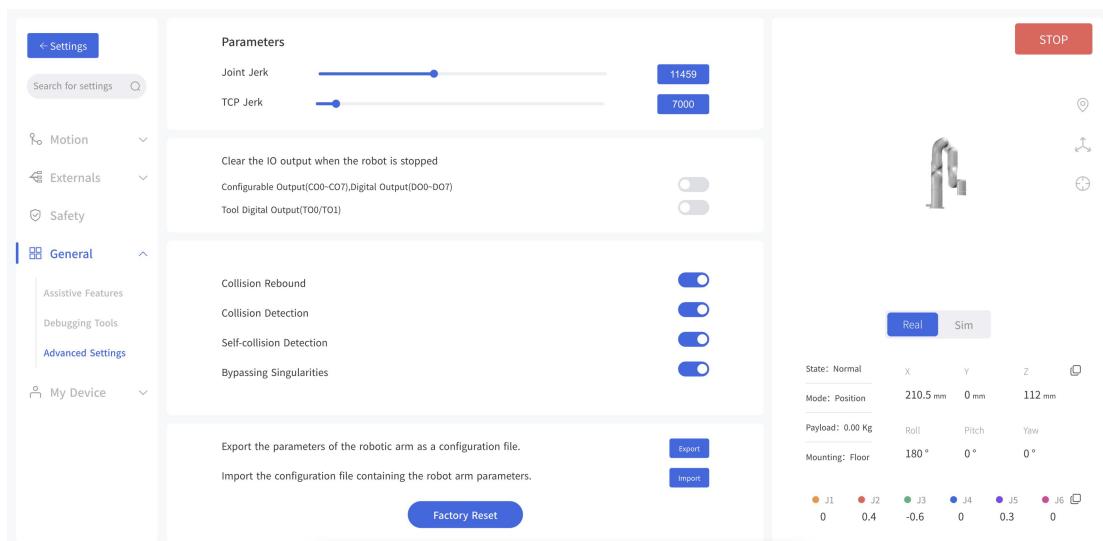
1. Please ensure to hold the robotic arm by hand when unlocking the joint to prevent it from falling down due to the inadequate provision of force, and take measures to protect the surrounding environment and peripheral facilities.
2. The operation of the unlocking joint is mainly used to adjust the posture of the robotic arm to a relatively safe position when the error is reported by the robotic arm. Attention should be paid to adjusting the joint into the range manually when it exceeds the range

of the joint.

3. In the "simulated robotic arm mode", clicking the unlock joint button will also unlock the real joints of the robotic arm.



#### 1.4.4.3 Advanced Settings



If you want to modify the joint jerk and TCP jerk of the robotic arm, you can modify time here.

Note:

1. The jerk affects the acceleration performance of the robotic arm.

In general, we do not recommend modifying this parameter.

2. If the robotic arm is not enabled, the jerk cannot be modified.

3. If an error warning occurs on the robotic arm, the jerk cannot be modified.

4. When the robotic arm is moving, the jerk can not be modified.

#### **Clear the IO output when the robot is stopped**

- After turning on 【Clear IO output when the robot is stopped】 if the robotic arm receives a stop command, 【Controller Digital Output】 or 【Tool Digital Output】 will be set to the invalid state. Otherwise, the 【Controller Digital Output】 or 【Tool Digital Output】 will not be affected by the stop command.

#### **Collision Rebound**

- When this mode is turned on, the robotic arm will rebound backward for a certain distance after it collides with an obstacle. If collision Detection is open, when this mode is turned off, the robotic arm will stay at the position where collision is detected.

#### **Collision Detection**

- Turn on collision detection

#### **Self-collision detection**

- When the mode is turned on, it will prevent the xArm from causing self-collision.

## ● Bypassing Singularities

Turn on Bypassing Singularities

### Configuration File

- Click the 【Export】button to export the parameters of the robotic arm as a configuration file.

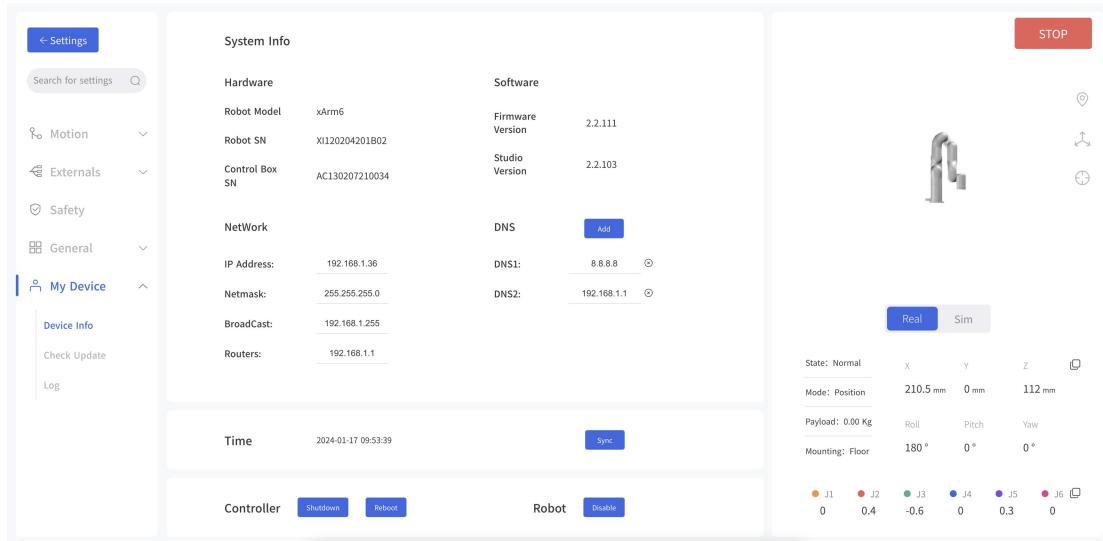
The robotic arm parameters that can be exported mainly include: motion parameters, TCP offset, TCP payload, IO settings, safety boundary, installation methods, coordinate systems, and advanced parameters.

- Click the 【Import】button to import the configuration file containing the parameters of the robotic arm.
- Click the 【Factory Reset】button, and the robotic arm will restore the factory settings mode.

Note:

- (1) When multiple robotic arms need to share a set of configuration parameters, click the 【Export】button to export the configuration file of a robotic arm that has been set. Then click the 【Import】button to import the configuration file to other robotic arms.
- (2) When the control box fails and needs to be repaired, you can export and save the configuration file of the robotic arm to prevent the original data from being lost or changed during the repair process.
- (3) The parameters of the robotic arm will change after the factory reset. Please export the configuration file of the robotic arm before the factory reset.

## 1.4.5 My Device



### 1.4.5.1 Device Info

- Display the IP address of the connected robotic arm, the firmware version of the arm, and the UFactory studio software version, the degree of freedom (number of axis) of the current robotic arm, and SN address of the robotic arm can be checked.

#### Network Settings

- Display the IP address of the robotic arm, subnet mask, broadcast address, and default gateway. The DNS address can be modified and added.

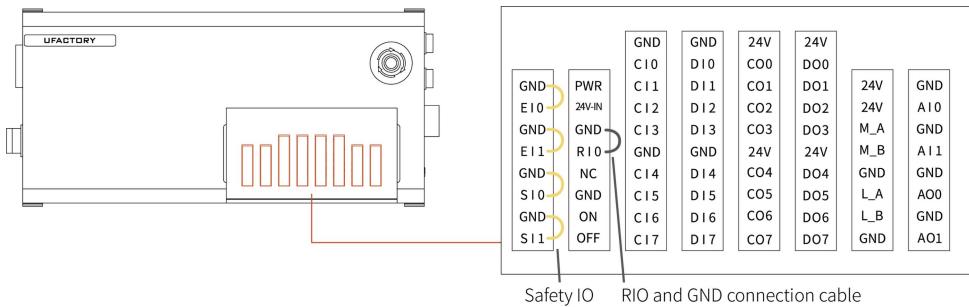
#### Note:

If you change the IP address, be sure to mark it on the control box. If you forget or lose the modified IP address, you can use the following method to reset the IP.

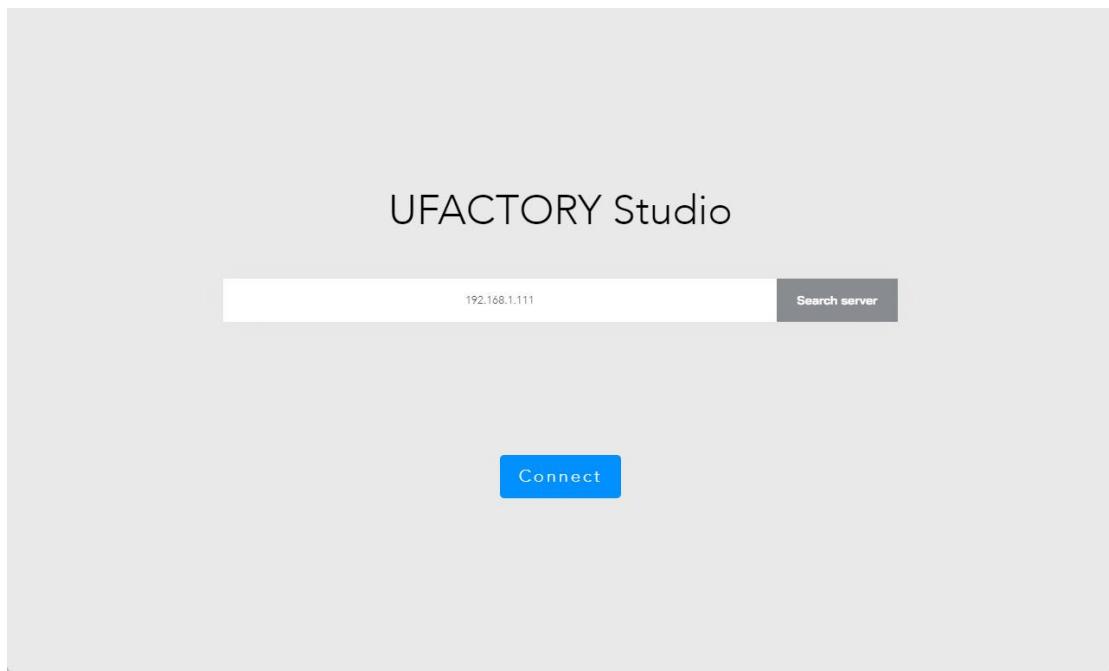
#### 1.4.5.1.1 Reset IP

Steps to reset IP:

1. Press the emergency stop button and turn off the power of the control box.
2. Connect RIO to GND with a cable.



3. Turn on the power of the control box. After hearing the sound of "beep", it means that the IP address of the control box has been reset successfully. The reset IP is 192.168.1.111.
4. Please unplug the cable connecting RIO and GND and wait for the control box to start up (60 seconds).
5. Enter 192.168.1.111 in the UFactory studio search box, connect the robotic arm.



- If you need to modify the IP, just modify the IP in [Settings] → [System Settings] → [Network Settings]. (For example: the modified IP is 192.168.1.36)

**System Info**

Hardware		Software	
Robot Model	xArm6	Firmware Version	2.2.111
Robot SN	XI120204201B02	Studio Version	2.2.103
Control Box SN	AC130207210034		

**NetWork**

		DNS	Add
IP Address:	192.168.1.36	DNS1:	8.8.8.8
Netmask:	255.255.255.0	DNS2:	192.168.1.1
BroadCast:	192.168.1.255		
Routers:	192.168.1.1		

**Real**   **Sim**

**Robot Status**

State:	Stop(5)	X	Y	Z
Mode:	Position	234.3 mm	0 mm	197.4 mm
Payload:	0.00 Kg	Roll	Pitch	Yaw
Mounting:	Floor	180°	0°	-45.2°

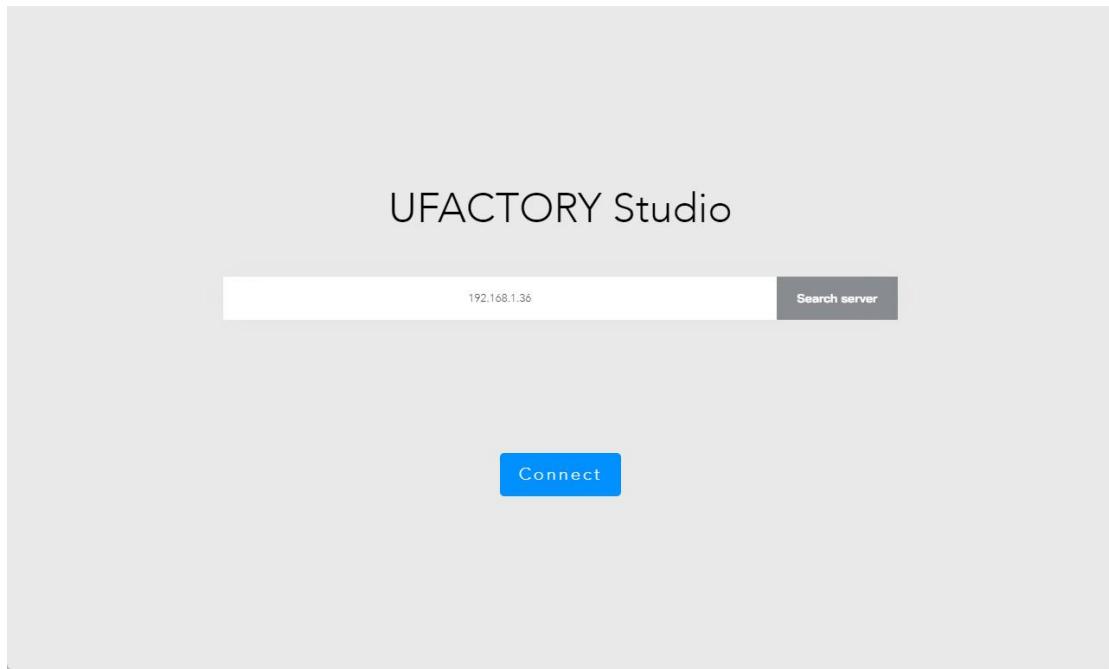
**Robot Control**

Controller	Shutdown	Reboot	Robot	Disable

**Joint Angles**

J1	J2	J3	J4	J5	J6
0	-28.9	-3.5	0	32.4	45.2

- Restart the control box, enter your modified IP in the UFactory studio search box, and connect the robotic arm.



**Note:**

1. If you need to reset the IP, the xArm firmware version must be  $\geq$  V1.5.0.
2. If you do not unplug the cable connecting RIO and GND, the next time you restart the control box, no matter what IP address you modify, the IP address of the control box will be automatically changed to 192.168.1.111, so after modifying the IP, Be sure to unplug the cable connecting RIO and GND.

#### Access to Control Box Shutdown

- Access to [Settings]-[My Device]-[Device Info]

#### Shutdown / Reboot

The control box can be shut down or restarted. Note that the shutdown / reboot button does not turn off the power supply and the main power supply to the robotic arm.

## 【Shutdown】

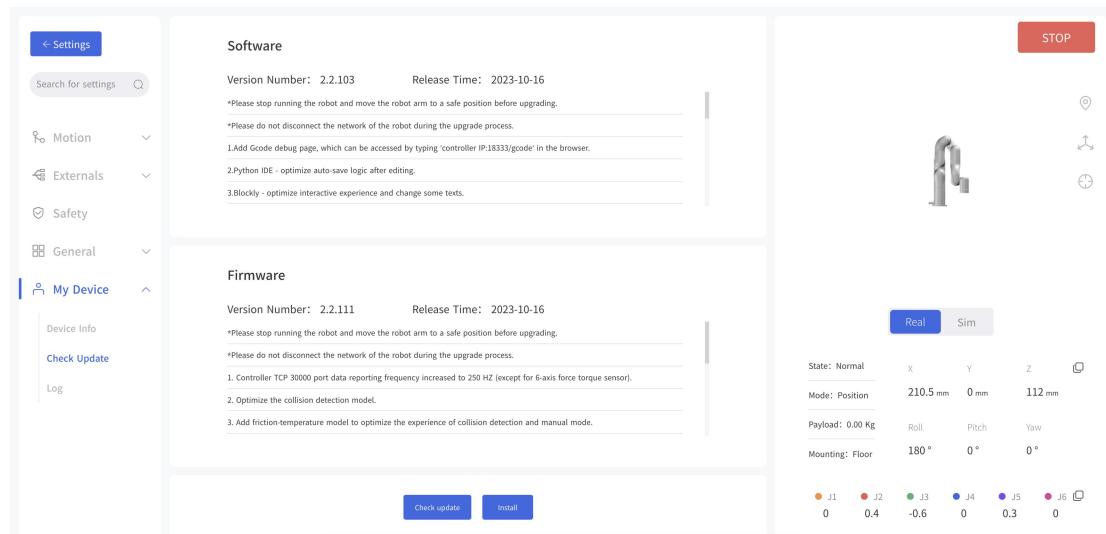
- Click this button, the page will go back to the 【Search the IP Address of the Control Box】 page, and the control box will shutdown. This operation is equivalent to long pressing the Power button of the control box, and the shutdown process takes 2 to 3 seconds.

## 【Reboot】

- Click this button, the control box of the robotic arm will restart. After the restart action is completed, the pop-up window will close and the robotic arm will be automatically reconnected. The operation is equivalent to the shutdown and startup process of the control box, and restart process takes 2 to 4 minutes.

Note: The 【xArm shutdown】 and 【xArm Reboot】 buttons do not affect the main power supply of the control box and the power supply of the robotic arm.

### 1.4.5.2 Check Update



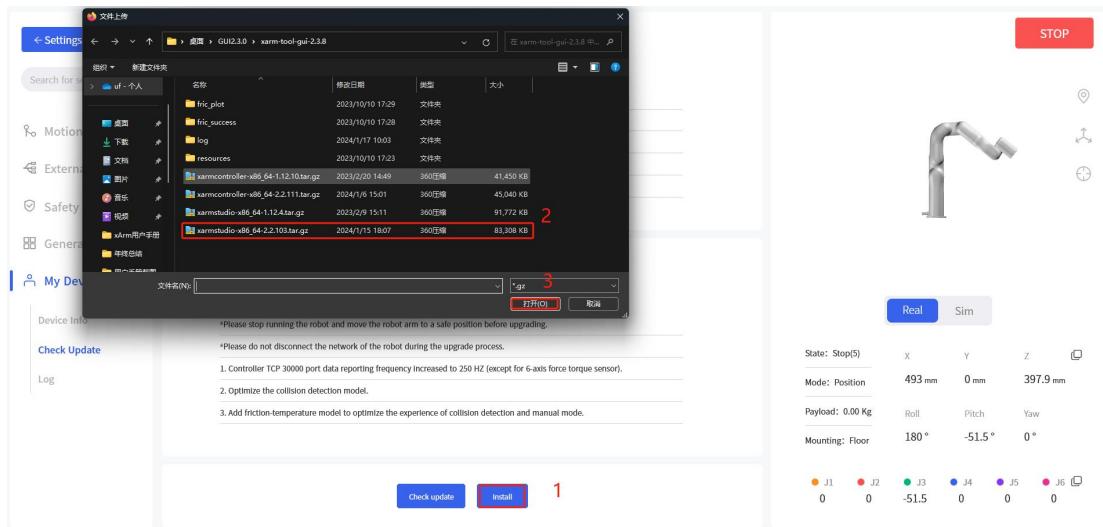
When updating software and firmware, make sure that the local area

network where the computer and control box are located can communicate with the external network. In addition, make sure the control box can communicate with external internet.

**Check update** : Click to get the latest UFactory studio and xArm firmware version information for your controller.

**【Install】** : Click to go to the offline installation window for UFactory studio and xArm firmware, select the upgrade package you downloaded in advance to update the firmware and studio to the latest.

Click to download the latest offline upgrade installation package



#### Note:

For detailed steps on updating UFactory studio and xArm firmware, please refer to Appendix 4-xArm Software/Firmware Update Method.

#### 1.4.5.3 Log

The error log of the control box, servo error log and end effector

error log can be checked.

Click the 【Download】button to download the error log.

The screenshot shows the robot control interface with the following sections:

- Left Sidebar:** Includes "Settings" (with a search bar), "Motion", "Externals", "Safety", "General", and "My Device" (with "Device Info", "Check Update", and "Log" sub-sections).
- Middle Section:** A table titled "Error Log" with columns "Time", "Error\_Code", and "Error\_Content". It lists several entries from January 2024, mostly related to emergency stop buttons being pushed.
- Right Section:** Real-time robot status including "State: Normal", "Mode: Position", "Payload: 0.00 Kg", "Mounting: Floor", and joint angles J1-J6.
- Bottom Right:** Buttons for "Real" and "Sim" modes, and a "STOP" button.

## 1.5 Live Control

The screenshot shows the live control interface with the following sections:

- Left Sidebar:** Includes "Live Control", "Blockly", "GCode", and "Settings" (with a gear icon).
- Middle Section:** "Recording" panel for "test01" showing speed (68%) and joint positions (J1-J6) over time (00:00 to 00:05). It includes "Play", "Pause", "Speed", and "Times" controls.
- Bottom Left:** "Product Information" section showing Model: xArm6, Robot IP: 192.168.1.36, Firmware Version: 2.2.111, and Software Version: 2.2.103.
- Bottom Right:** Real-time robot status including "State: Normal", "Mode: Position", "Payload: 0.00 Kg", "Mounting: Floor", and joint angles J1-J6.
- Bottom Right:** Buttons for "Real" and "Sim" modes, and a "STOP" button.

### 1.5.1 Page Introduction

#### o IP address

Displays the IP address of the currently linked robotic arm controller.

- o **Payload**

Displays the current default load parameters of the arm Robotic arm status

- o **Normal:** The robotic arm is not in motion, but is ready to go into motion.

- o **Motion:** The robotic arm is in motion.

- o **Stop:** The robotic arm is in motion stop.

- o **Pause:** The robotic arm is in motion pause.

- o **mode**

The mode the robotic arm is currently in, default is **【position mode】**  
Position information

The position information that the robot arm is currently in, contains [X, Y, Z, R, P, Y], or [X, Y, Z, Rx, Ry, Rz] if the axis angle display mode is selected real robotic arm/simulation robotic arm

- o **Simulation robotic arm:**

It means the robotic arm is connected and the current one is simulation robotic arm. You can control the virtual robot arm movement through UFactory studio interface.

- o **Real arm:**

means the arm is connected and the arm is currently a real arm. You can control the movement of the real robot arm through UFactory studio interface. The virtual robot arm will react to the position and attitude of the real robot arm in real time.

Note: A robot arm can only be in one mode (real robot arm mode / simulation robot arm mode).

### o Align

When this button is clicked, the tool flange will be leveled, i.e. pitch and roll will be adjusted to fixed values of  $0^\circ$  and  $180^\circ$ .

### o Initial point

It means all joint angle values are 0. Long press the Initial point button to make the robot arm return to zero point attitude. You can set the initial point by clicking 【Settings】 - 【Motion】 - 【Parameters】 - 【Initial Point Position】 in the home page.

【Initial point】 point press for step motion, long press for continuous motion.

## 1.5.2 Emergency Stop

**STOP**

Click on the emergency stop button to immediately stop the current motion and clear all cached commands.

**Note:**

The “STOP” button in UFactory studio is different from the one on the control box.

1. The “STOP” button in UFactory studio allows the robotic arm to stop the current motion and clear all cache commands immediately. It is a software stop, and the power is still on.

2. The Emergency STOP button on the control box: Send out a stop command to cut off the power supply of the robotic arm, and thence the posture of the robotic arm will slightly brake and fall.

### 1.5.3 Position Control

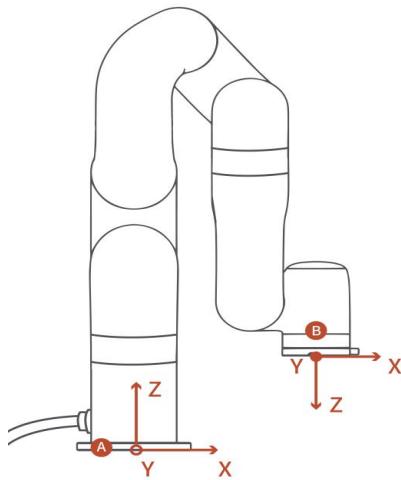
#### 1.5.3.1 Linear Motion

Users can control the motion of the robotic arm based on the base coordinate system and TCP coordinate system. The trajectory of tool center point in the Cartesian space is a straight line. Each joint performs a more complex movement to keep the tool in a straight path. The TCP path is unique once the target point is confirmed, and the corresponding posture in the execution process is random.

X, Y, and Z control the position of TCP in base or tool coordinate system, in the unit of mm. While Roll/Pitch/Yaw controls the TCP orientation in the unit of degree.

Linear motion and arc linear motion belong to the Cartesian space trajectory planning, which needs to be solved by inverse kinematics. Therefore, there may be no solution, multiple solutions, and approximated solutions; and due to the nonlinear relationship between the joint space and Cartesian space, the joint motion may exceed its maximum speed and acceleration limits.

### 1.5.3.2 TCP Coordinate System

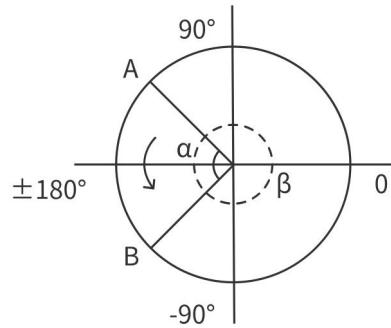


A: Base coordinate system      B: Tool coordinate system

The default TCP coordinate system is defined at the centre point of the end flange of the robotic arm, and it is the result of rotating [180° , 0° , 0° ] around the X/Y/Z-axis of the base coordinate system in order. The spatial orientation of the TCP coordinate system changes according to the changes of the joint angles.

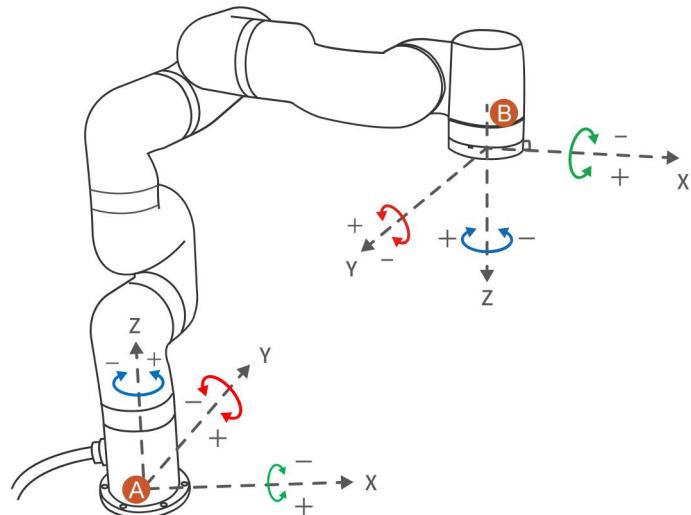
- Roll/Pitch/Yaw respectively rotates around X/Y/Z of the base coordinate system, and the final TCP orientation is the result of the three rotations in exact order. The robotic arm will always choose the shortest way to reach target orientation. In particular, it is important to strictly control the magnitude of the deflection angle between the two points to control the direction of rotation, and if necessary, insert a third point between the two points. As shown in figure 6.4, if a deflection is needed from position point A to point B, the robotic arm moves in the direction of  $\alpha$  angle. If the robotic arm needs to be moved in the direction of the  $\beta$  angle, a new position between the angles of  $\beta$  should be inserted, and the angle that formed by

the inserted point and A should be smaller than  $\alpha$ .



- The  $+180^\circ$  and  $-180^\circ$  points of the Roll/Pitch/Yaw are coinciding in the space, and the valid range is  $\pm 180^\circ$ , so it is possible to have both  $\pm 180^\circ$  when the robotic arm is reporting the position.
- Roll angle, pitch angle, and yaw angle (RPY). The RPY rotation matrix ( $X$ ,  $Y'$ ,  $Z''$  rotation) is determined by the following formula:

$$R_{rpy}(\gamma, \beta, \alpha) = R_Z(\alpha) \cdot R_Y(\beta) \cdot R_X(\gamma)$$



A: Base coordinates

B: TCP coordinates (If no offset)



**DANGER**

1. You must check the TCP offset before recording the Cartesian position.

### 1.5.3.3 Speed Setting

It is used to adjust the motion speed of the live control interface of xArm. (Note that the maximum speed of the live control interface is not the actual maximum motion speed of the robotic arm. If you want the program to run at high speed, you can add a speed command in the Blockly motion program).

#### Joint Operating Speed

- The range is  $1^\circ /s \sim 180^\circ /s$ . When the robotic arm is in operation, the actual maximum speed will be influenced by the payload, speed, and the pose, and the maximum speed would not be an absolutely reachable value.

**Note:** the speed at which the joint runs between each command is not continuous, and the robotic arm will have a brief pause between joint command.

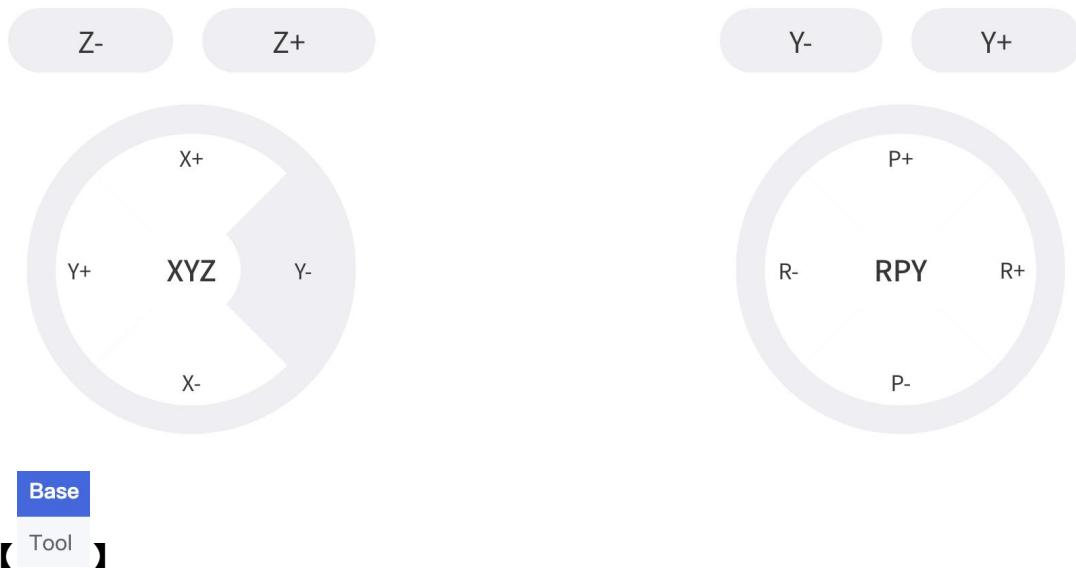
#### TCP Operating Speed

- The Cartesian speed range is from 1mm/s to 1000mm/s. The actual maximum speed is also affected by the payload, speed, and posture of the robotic arm. If the set speed is close to the limit speed, the robotic arm will slow down or cause an error mechanism. When a command involves displacement and rotation, at the same time, the time required for the displacement motion and the rotational motion

depends on the one that takes more time, but in principle, it is better to separate the displacement from the rotation command.

### 1.5.3.4 Operation Mode

#### 1.5.3.4.1 xArm 6 (xArm 7) Operation Interface



- It can switch the control functions between the base coordinate system and the tool coordinate system.  
【Position/Attitude Real-time Display】
  - X / Y / Z represents the coordinates of the tool center point (TCP) position of the robotic arm under the base coordinate offset. Roll/Pitch/Yaw under the Attitude indicates the angle value rotated under the base coordinate offset, which is a description of the azimuth obtained by rotating three times around the selected coordinate system in a certain order.

#### 【Real-time Position Control】

- X/Y/Z controls the X/Y/Z-axis of the selected coordinate system respectively. Click for step motion and long press for continuous

motion.

#### 【Real-time Attitude Control】

- Roll/Pitch/Yaw controls the Roll/Pitch/Yaw of the selected coordinate system respectively. Click for step motion and long press for continuous motion.

The step can be set by clicking 【Settings】 - 【Motion】 - 【Parameters】 - 【Attitude Step】 on the homepage.

### 1. 5. 3. 4. 2 xArm 5 Operation Interface



#### 【Aligning the Hand】

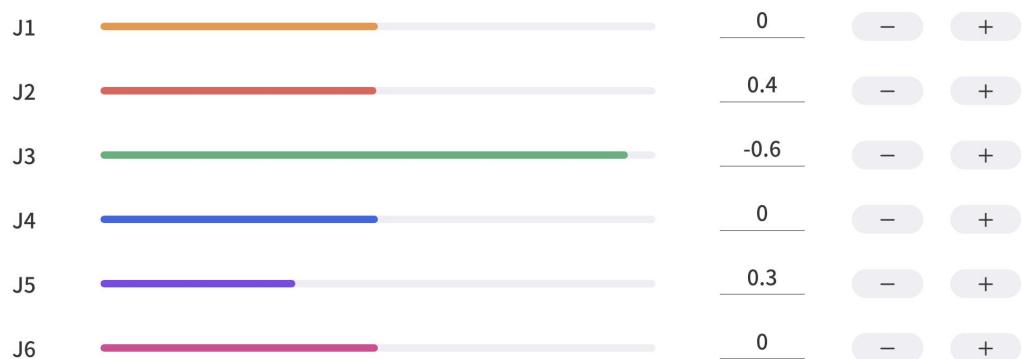
- After clicking this button, the tool flange will be adjusted to a horizontal attitude, that is, pitch and roll will be adjusted to the fixed values of  $0^{\circ}$  and  $180^{\circ}$ .

### 1. 5. 3 Joint Motion

The robotic arm consists of joint modules. The position of the end-effector is controlled by coordinating the rotation angle of each joint.

The joint motion reaches the target point with the fastest path, the end trajectory is not a straight line, and the speed unit is ° /s.

After the target point is set, the corresponding poses are unique for TCP and the joints along the trajectory.



You can copy the joint angle value of the robotic arm by clicking this button.

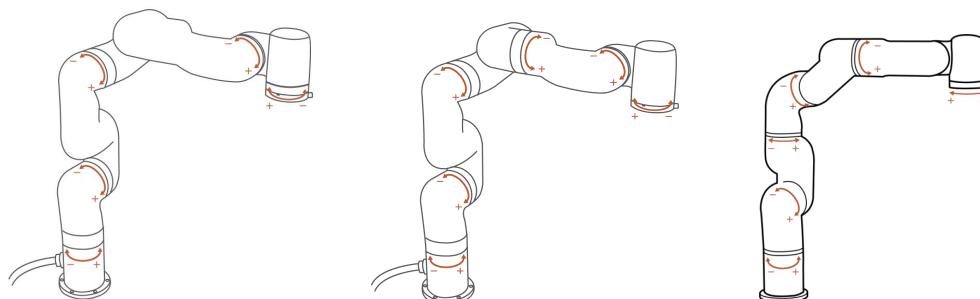
The progress bar represents the range of joints, the text represents the current joint and its degree.

#### Operation mode:

Click **【+】** or **【-】** for the step angles, users can set the step angle in **【Settings】 - 【Motion Settings】 - 【Joint Motion】 - 【Joint Step】**.

Press-and-hold **【+】** or **【-】** for continuous joint motion in a positive or negative direction, which will stop when the mouse is released.

To confirm the direction of joint rotation, please refer the figure below:



xArm 5

xArm 6

xArm 7

### 1. 5. 3. 1 Manual Mode

By turning on the Manual Mode, the joint can be driven freely by hand.

- Turn on the joint manual mode, you can manually drag the robot links to reach the target position, making it easier to record the robot's motion trajectory, thereby reducing the development workload. When danger occurs, you can also use the manual mode to manually drag the robot away from the danger zone.
- Drag sensitivity can be set in 【General】 - 【Assistive Features】 - 【Manual Mode Sensitivity】

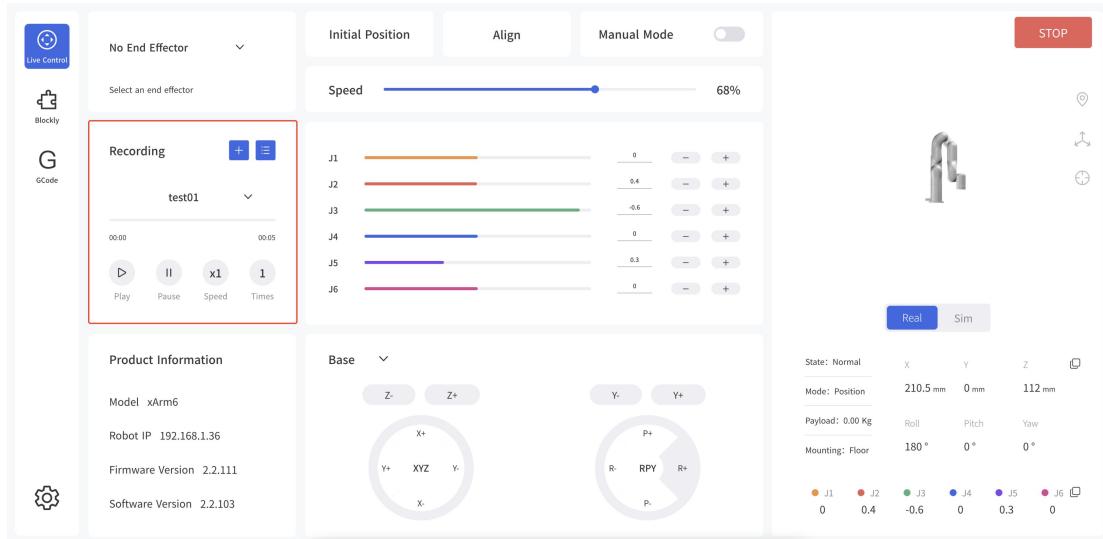
**Teach sensitivity:**

- The level of Teach sensitivity is from 1 to 5. The higher the set value, the smaller the force required to drag the joint in manual mode.

Note:

1. Before opening the manual mode, you must ensure that the installation method of the robotic arm and the payload setting of the robotic arm are consistent with the actual situation, otherwise it will be dangerous.
2. The serial number of robotic arm and the control box need to be matched before Manual Mode can be turned on. The SN of the control box can be checked in 【Settings】 - 【My Device】 - 【Device Info】 .
3. The SN address of the robotic arm can be checked next to the power signal interface of the base.

## 1.5.4 Recording



The position of the joint is obtained and recorded by 250HZ to record the motion trajectory of the robotic arm in free driving, and the maximum recording time is 5 minutes. The playback will completely repeat the motion trajectory during recording, and the playback speed of the trajectory can be set ( $x_1$ ,  $x_2$ ,  $x_4$ ). A recorded trajectory can be imported into Blockly projects.

**【 + 】** Create a new recording file.

**【 Record 】** Manual Mode will be turned on accordingly by clicking on the button, and the robotic arm can be dragged directly for trajectory recording. When starting recording, be sure to pay attention to the load state of the robotic arm, so as to avoid the big difference between the actual load and the set load of the robotic arm, resulting in its self-motion.

**【 00:04 】** Display recording time.

**【 End Recording 】** Stop recording.

**【Times】** Set playback times.

【Speed】 Set playback speed.

My Projects

Export Delete Cancel Deselect All

Select	ID	Title	Duration
<input checked="" type="checkbox"/>	1	test01	00:05

Close

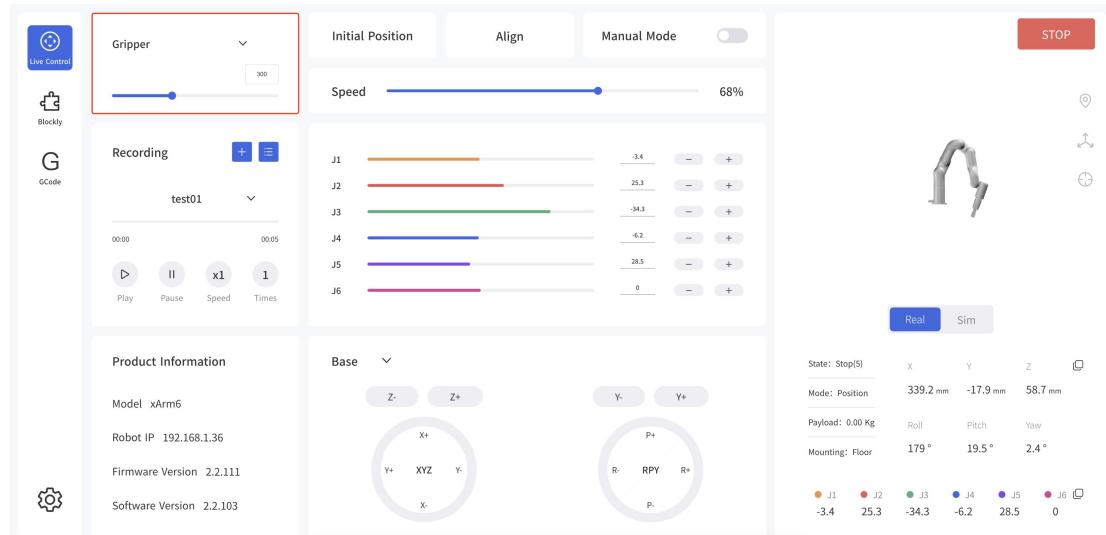
### 1. 5. 5 End Effector

- When the end effector provided in the option is installed at the end of the xArm, select the corresponding end effector.

The end effectors currently supported by xArm are: xArm Gripper, xArm vacuum Gripper, xArm BIO Gripper, Robotiq-2F-85 Gripper, Robotiq-2F-140 Gripper.

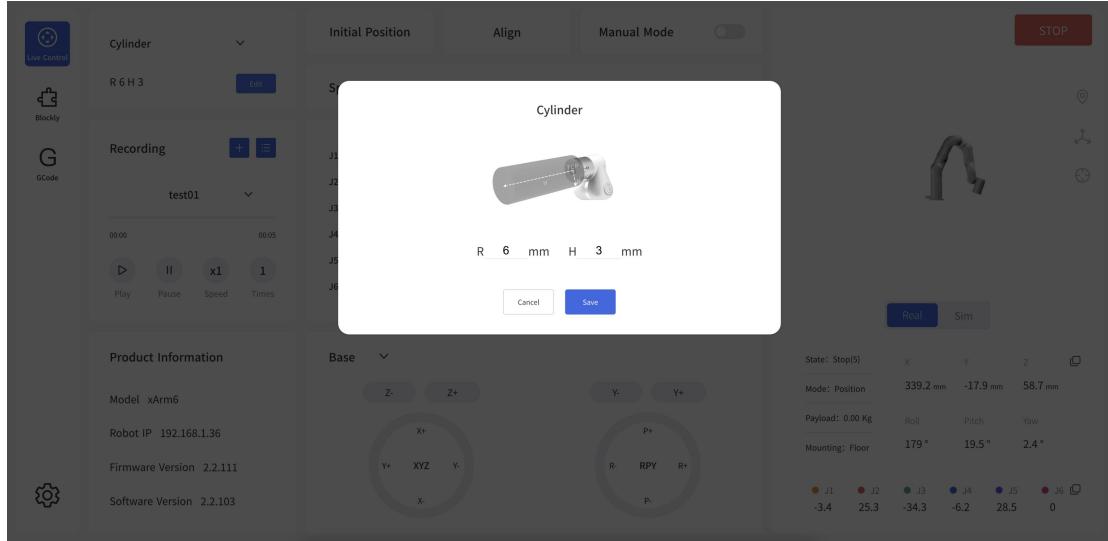
Take the xArm Gripper as an example.

#### xArm Gripper

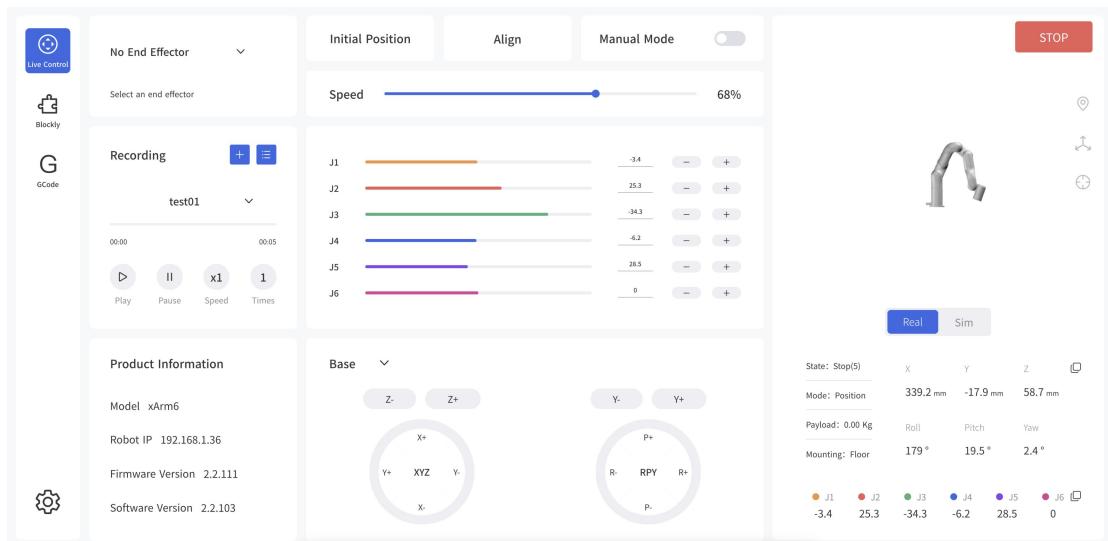


- When installing other end effectors (not officially provided) at

the end of the robotic arm, please choose **[other]**.



1. You can choose a 3D model (cylinder/cuboid) that can wrap the end effector and use it as the self-collision prevention model of the end effector.
- When no end effector is installed at the end of the robotic arm, select **[No End Effector]**

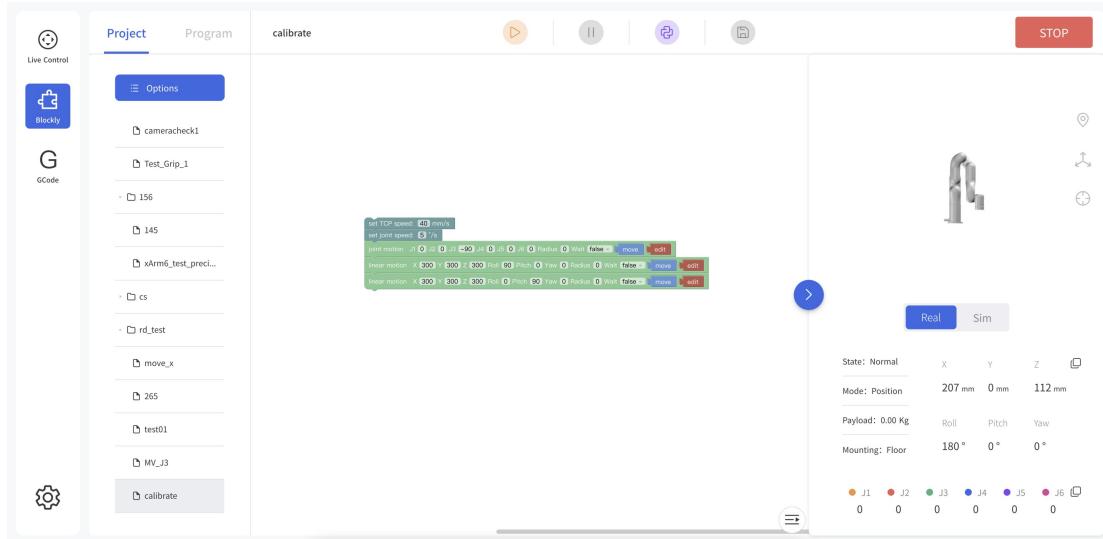


## 1.6 Blockly Graphical Programming

Blockly is a graphical programming tool that can be programmed to

control the robotic arm by dragging and dropping code blocks without the need to write the code manually.

### 1.6.1 Interface Overview



**【Project】** Click to expand to display all created items, the currently open item "xxx" is displayed when it folds.

**【Program】** Click to expand the different programming modules, such as Motion, Setting, Logic, Loop

**【▶】** Run the Blockly program

**【✚】** Click to convert Blockly projects to Python code.

**【💾】** Save changes to Blockly code.

New Folder

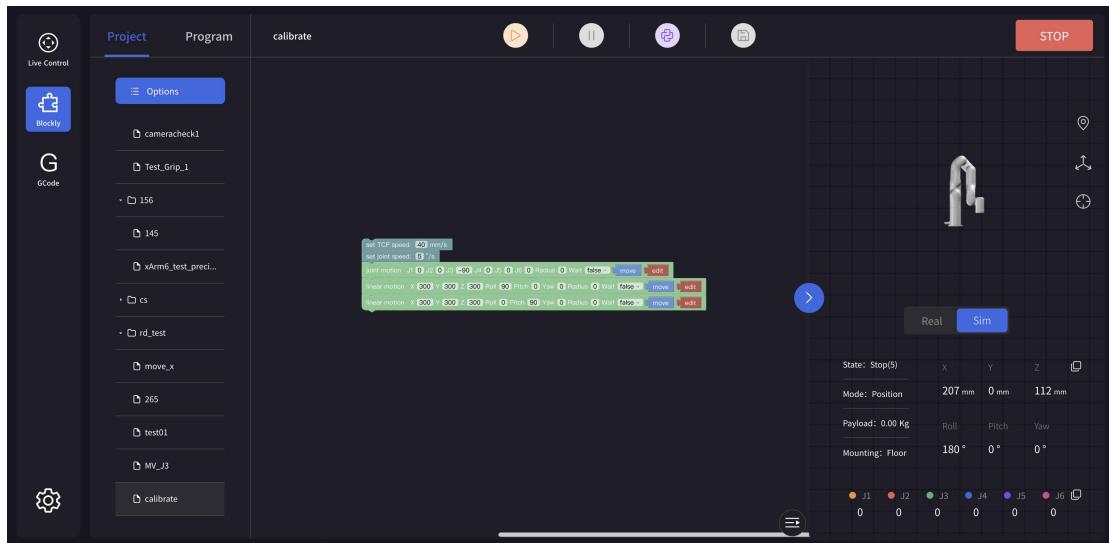
New File

Import

Download

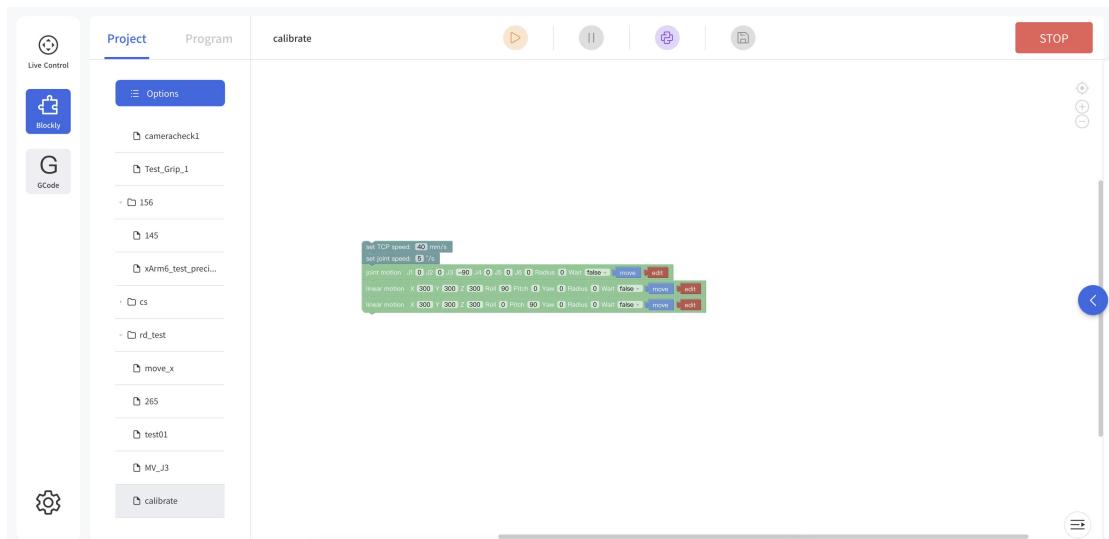
Rename

**【Delete】** Right mouse click on the Blockly file name to download, rename, new folder, new file, import or delete the file



Note: When the robotic arm is in the simulation mode, you can also run the Blockly motion program to observe the motion of the virtual robotic arm.

## 1.6.2 Blockly Workspace



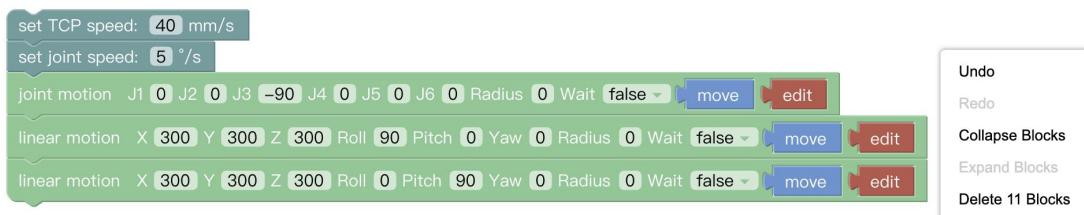
Drag the code block into the action panel, the code execution is top-down, users can drag and drop the code block with the blocks attached from behind together.

 Return to the default size and code block at centered position

【+】 Zoom in on the code block.

【-】 Zoom out on the code block.

### 1.6.2.1 The Right Click Mouse Event in the Workspace



Right-click on the blank workspace of the non-code block, the function is mainly for all code blocks:

【Undo】 : Undo the previous operation.

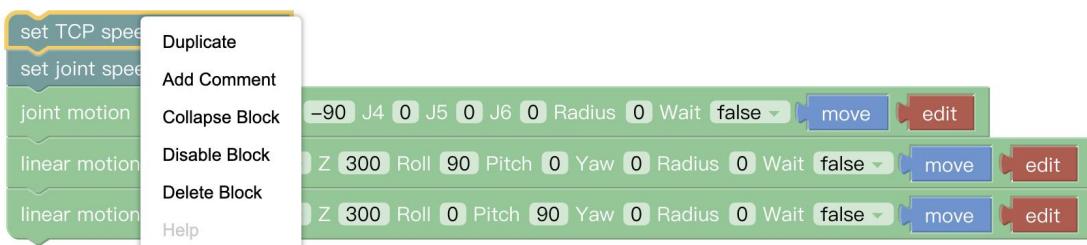
【Redo】 : Restore the last undo operation.

【Collapse Blocks】 : Collapse all code blocks.

【Expand Blocks】 : Display all collapsed commands.

【Delete 22 Blocks】 : Delete all code blocks.

### 1.6.2.2 The Right Click Mouse Event of the Code Block

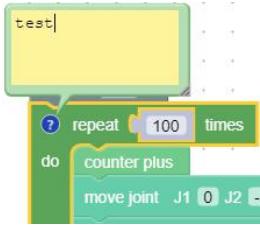


Right-click in the code block, the function of each module pop up:

【Duplicate】 : Copy all code blocks of the current workspace, copy/cut shortcuts with the keyboard and paste them into other files.

【Add Comment】 : Users can add a description to the code block, which is identified by the symbol . Click to open/close

the description pop-up window, as shown in the following figure.



**【External input】**: The location for setting the text box is displayed at the far right.

**【Internal Input】**: The location for the setting of the text box to be displayed in the default middle position.

**【Collapse Block】**: Folds the code block of the current workspace.

**【Disable Block】**: Stop the execution of the running command of the current code block. The opposite is **【Enable Block】**.

**【Delete 82 Blocks】**: Delete the current code block selected by the mouse click.

**【Help】**: Jump to the Help Page of the corresponding code block.

### 1. 6. 2. 3 Move/Wait/Edit

For some common functions of the motion commands, click **【move】** and the robotic arm will move to the current position; click **【edit】**, Then the robotic arm will move to the current position and open the live control interface; **Wait (true/false)**, indicating whether to wait for the execution of a command before sending the next one.

### 1. 6. 3 Blockly Code Block

**Setting:** Used to set the running speed, acceleration, collision sensitivity, load, etc. of the robotic arm.

**Motion:** Common motion commands including linear motion, joint motion,

linear motion with arc, sleep time, zero point, and emergency stop.

**Application:** You can import Blockly other projects.

**GPIO:** External input signal triggered event (suitable for plc).

**End-Effector:** Contains common commands to control the end-effector, such as gripper, vacuum gripper.

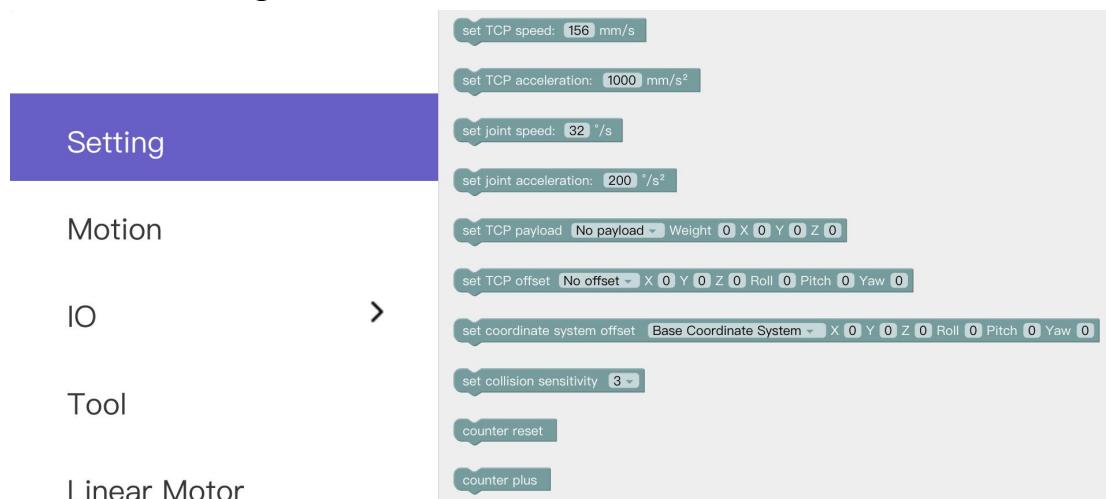
**Logic:** Contains commonly used logic commands.

**Loop:** Contains common loop commands such as multiple loops, infinite loops, and breaking loops.

**Math:** Contains commands for mathematical operations.

**Advanced:** Includes location notes and message reminders.

## 1. 6. 4 Setting



【Set TCP speed () mm/s】

- Set the speed of the linear motion in mm/s.

【Set TCP acceleration () mm/s<sup>2</sup>】

- Set the acceleration of the linear motion in mm/s<sup>2</sup>.

【Set joint speed () ° /s】

- Set the speed of joint movement in ° /s.

**【Set joint acceleration () ° /s<sup>2</sup>】**

- Set the acceleration of joint motion in ° /s<sup>2</sup>. The default speed and acceleration values in the code block are the speed and acceleration values set currently, which can be modified manually.

**【Set tcp load () weight () XYZ】**

- Set the load of the current project, refer Settings-TCP Payload from the drop-down list.

**【Set tcp offset () X Y Z R P Y】**

- Set the end offset of the current project, reference Settings-TCP Offset from the drop-down list.

**【Set world offset () X Y Z Roll Pitch Yaw】**

- Set the base coordinate offset of the current project. The drop-down list refers to the data of the Setting-Base Coordinate Offset.

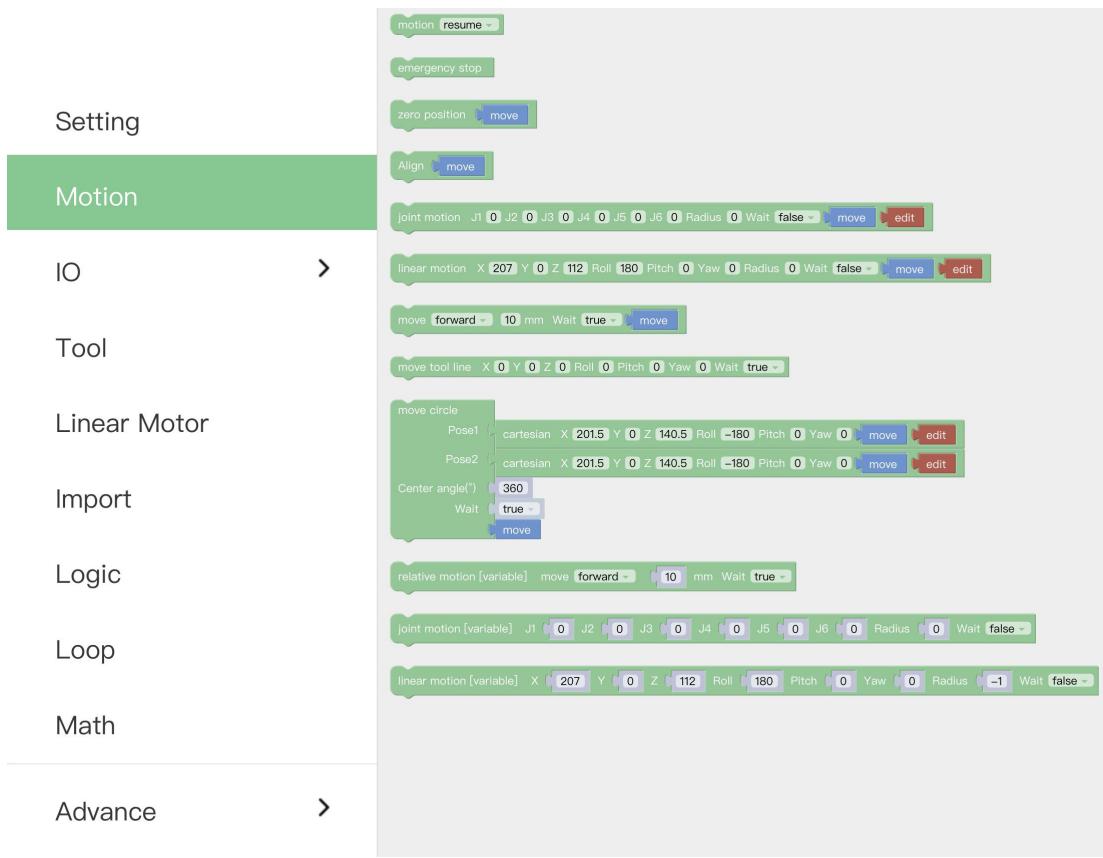
**【Counter reset】**

- This command resets the counter in the control box to 0

**【Counter plus】**

- Each time the command is run, the counter of the control box will be incremented by 1. It can be used to calculate the number of times the program actually cycles.

## 1.6.5 Motion



### 【sleep () s】

- After receiving this command, the robotic arm will stop moving for the set time, and then continue to execute the following commands. It is mainly used in motion programs that need to do the continuous motion. It is used to buffer more motion commands for successful continuous motion calculation.

### 【Motion ()】

- With this command, operators can set the state of the robotic arm (movement, pause, stop). It is used to control the state of the robotic arm. It is mainly used in condition-triggered programs.

### 【Emergency stop】

- The robotic arm immediately stops moving and clears the command cache.

## 【Zero position】

- The robotic arm returns to a posture where the joint value are 0.

【move joint J1() J2() J3 () J4() J5() J6() J7() Radius()  
Wait(true/false) , [move] , [edit]】

- Set each joint value for the joint movement, with the unit of degree.

【move(arc) line X() Y()Z() Roll() Pitch() Yaw() Radius()  
Wait(true/false) [move] [edit]】

- Set the Cartesian coordinate target value of the linear motion and the TCP rotation angle in mm and ° .

【Move (front/back/left/right) (true/false) () mm 】

- Indicates that the robotic arm makes relative linear motion forward/backward/left/right based on the current position, in mm

【move tool line X() Y()Z() Roll() Pitch() Yaw() Radius()  
Wait(true/false) 】

- This command is a relative motion relative to TCP coordinates.

【move circle position 1 to position 2】

- From current position, the whole circle is determined by current position and position1 and position2, “center angle” specifies how much of the circle to execute.

【center angle (° ) () 】

- Indicates the degree of the circle. When it is set to 360, a whole circle can be completed, and it can be greater than or less than 360; (Note: To achieve smooth track motion, you need to set Wait = false).

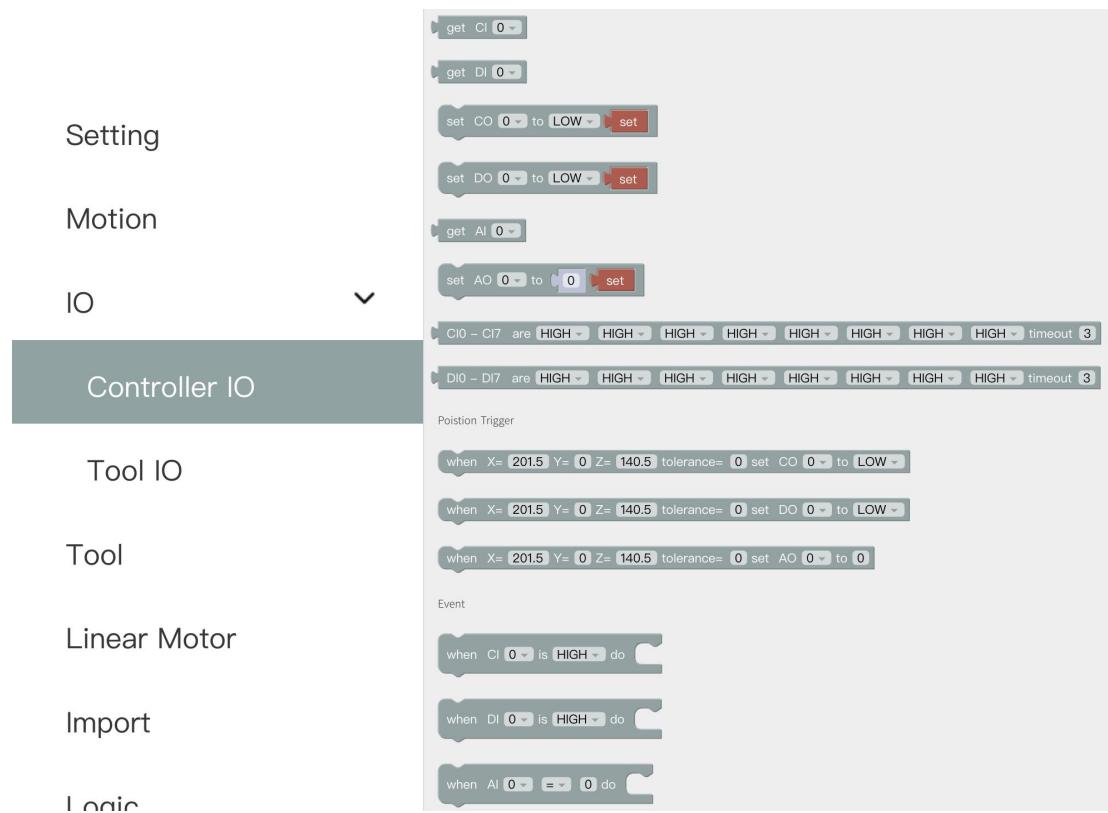
【move joint [variable] J1() J2() J3 () J4() J5() J6() J7()  
Wait(true/false) 】

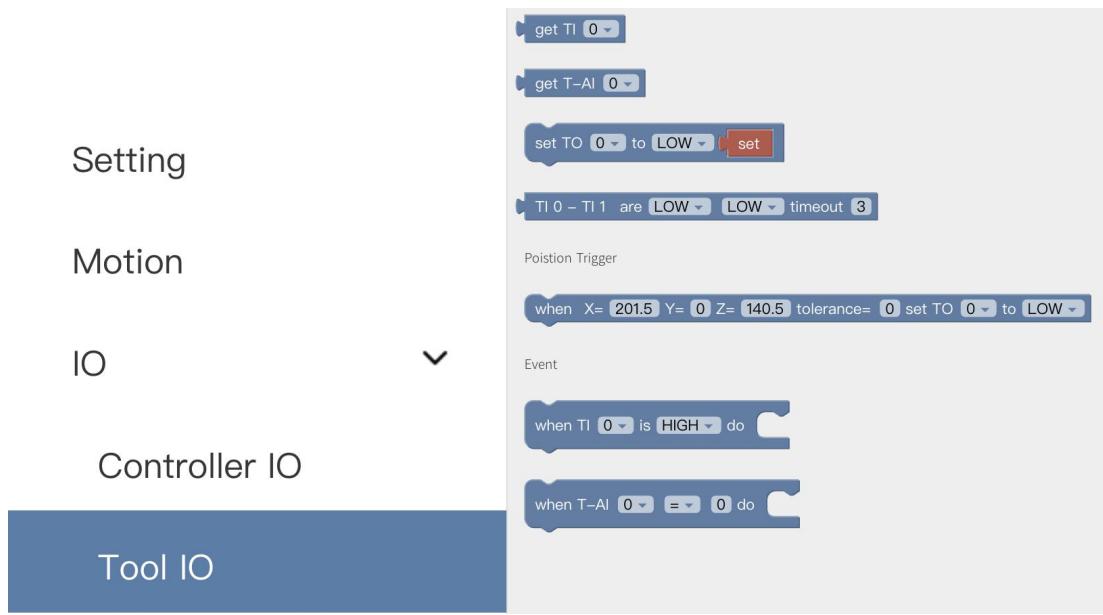
- The command passes through the joint motion and supports variable values.

```
【move (arc) line [variable] X() Y() Z() Roll() Pitch() Yaw()
Radius() Wait(true/false)】
```

- The command passes through the Cartesian motion and supports variable values.

### 1.6.6 GPIO (Control Box and End tool interface)





The IO interface is made up of a control box interface and an end tool interface, which can be used to acquire, set, and monitor IO interface operations. The control box has 8 digital input interfaces, 8 digital output interfaces, 2 analog input interfaces, and 2 analog output interfaces. The end tool has 2 digital input interfaces and 2 digital output interfaces. 2 analog input interfaces. The control box digital IO is low-level-triggered. The end tool digital IO is high-level-triggered.

### 【get I/O】

- Acquire the I/O interface data of the code block.

### 【set I/O】

- Set the I/O interface of the code block, click 【Set】 to run the command.

### 【set I/O when (X, Y, Z, tolerance)】

- When the robotic arm reaches the specified position (the area of the sphere specified with the trigger position point (X, Y, Z) as the center (the radius of the sphere is the tolerance radius)), IO

is triggered. This command can be used to trigger IO at a specific location.

X, Y, Z represent the coordinate value of the specified position to be reached by the robot arm, with the unit of mm.

The digital IO is triggered as soon as the system detects that the TCP position enters a spherical area centered at (X, Y, Z) with the specified radius. If the tolerance radius is not set, when the robotic arm passes the specified point at a speed other than 0, it may miss the trigger because it cannot be accurately detected.

**【when digital I/O is (High/Low) do】**

- Executes the commands contained in this code block when the condition is met.

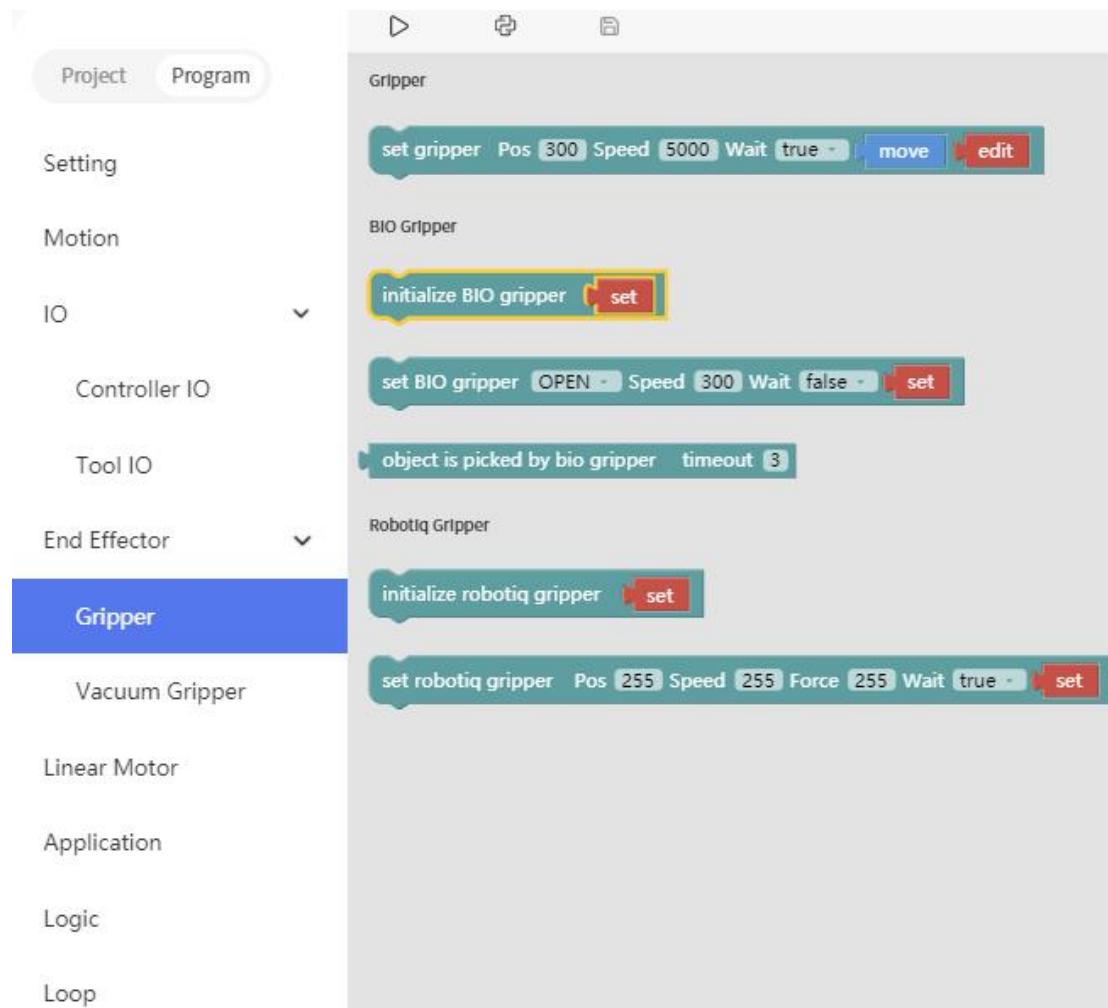
**【 when the analog IO value satisfies the set condition do】**

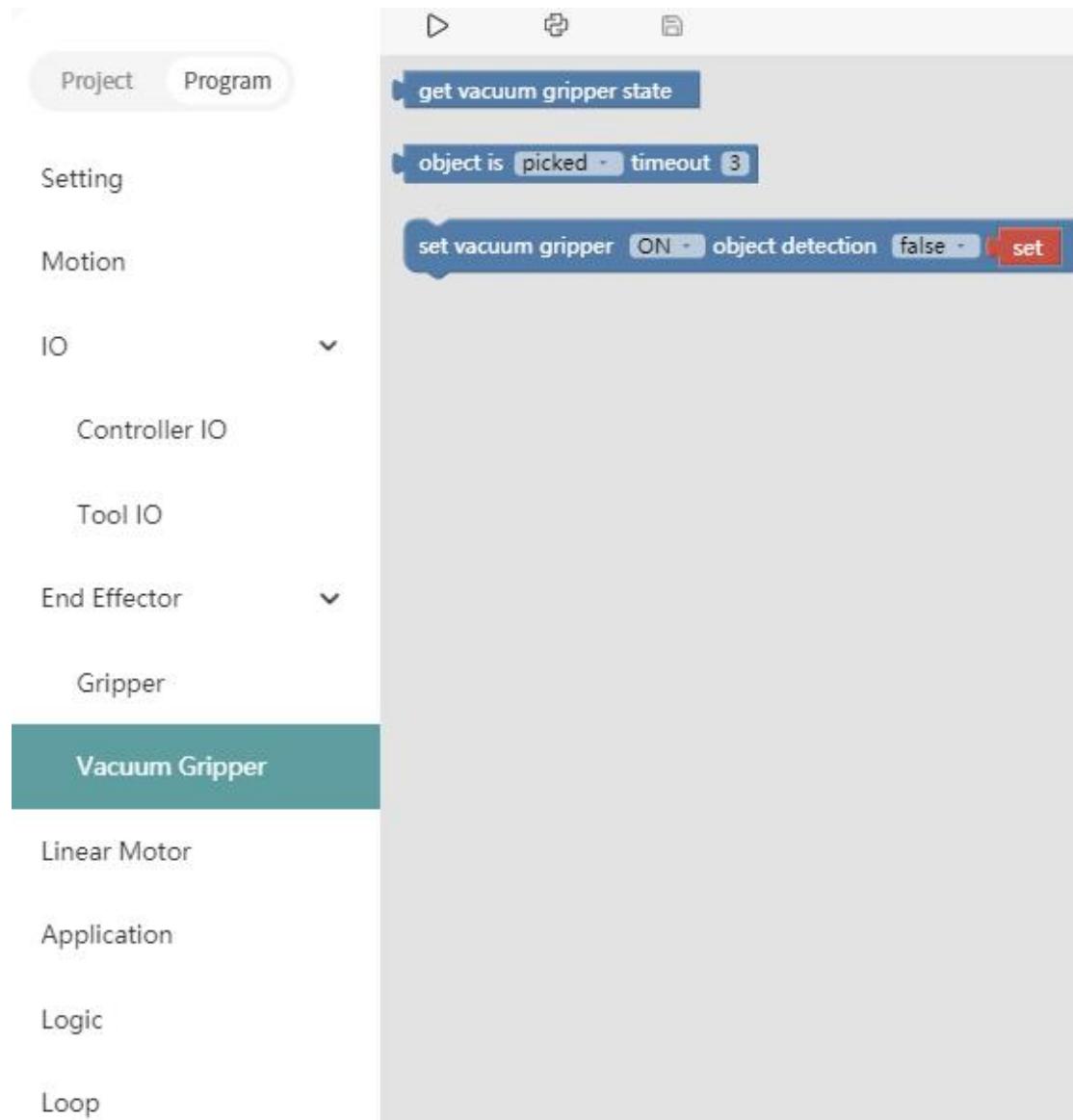
- When the monitored analog IO value meets the condition, the commands contained in the code block will be executed, and the condition are =, ≠, >, ≥, <, ≤.

**IO trigger logic of UFactory studio:**

1. UFactory studio obtains the IO state every 100ms, and uses the IO state value obtained for the first time as the initial value.
2. Compare the IO state obtained the second time with the IO state obtained last time. If the IO state changes, a callback meeting the condition is triggered.

## 1.6.7 End Effector





【set xarm gripper Pos () Speed () Wait (true / false)[move][edit】

- Set the position and the opening and closing speed of the gripper.

【set bio gripper Speed () Wait (true / false)[move][edit】

- Set the opening and closing speed of the gripper.

【set robotiq gripper Pos () Speed () Wait (true / false)[move][edit】

- Set the position of robotiq gripper, opening and closing speed, and the strength of the gripping object.

【initialize gripper】

- Enable gripper.

【object is (picked/release)】

- Detect whether the vacuum gripper has picked (released) the object.

If it is detected that the vacuum gripper has picked (released) the object, then jump out of this command and execute the next command.

If the timeout period is exceeded, the vacuum gripper has not yet picked (released) the object, it will also jump out of the command and execute the next command.

**【get xarm vacuum gripper state】**

- Obtain whether the vacuum gripper picks the object or not. When the vacuum gripper state is 1, it indicates that the object is picked successfully; when the vacuum gripper state is 0, it indicates that the object fails to be picked.

**【set xarm vacuum gripper (ON/OFF) object detection (true/false)**

**[set]】**

- Set the vacuum gripper to be on and off.

[object detection] = true: detect whether the object is picked, if not, it will jump out of the entire program.

[object detection] = false: do not detect whether the object is picked.

## 1.6.8 Application



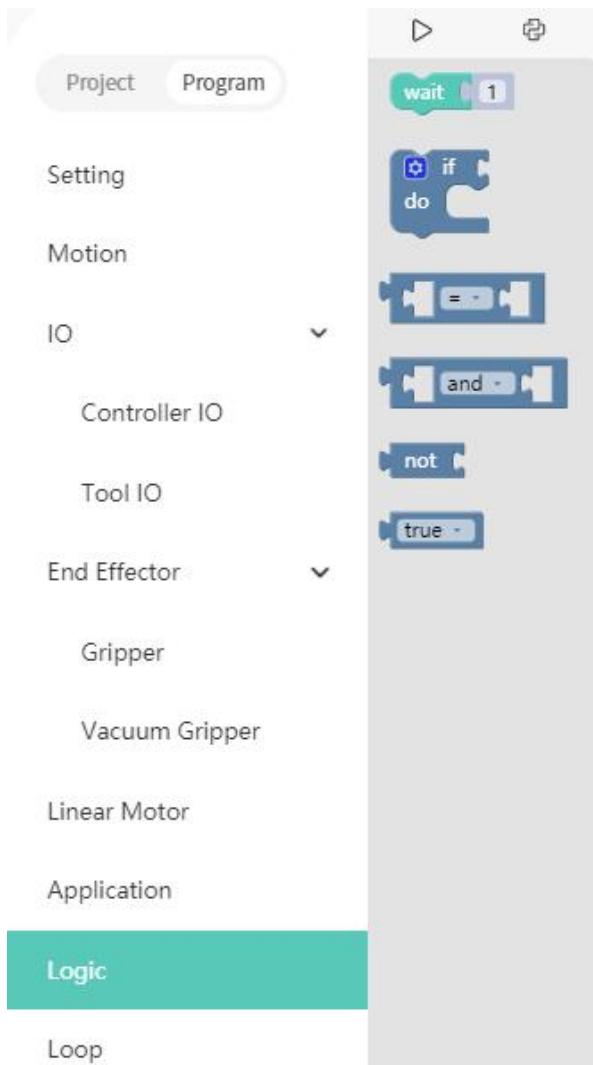
**【Run Trajectory (path) Times [1]】**

- Users can import the trajectory recording file and set the times of executions.

**【Import other APP】**

- Users can import Blockly of other projects.

## 1.6.9 Logic



【wait ()】

- Wait for the next command to be sent, with the unit of seconds.

【if (Condition 1) Run (Command 1)】

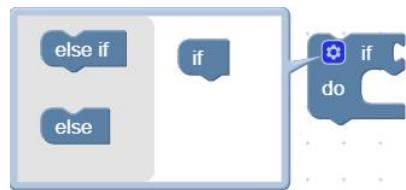
- If Condition 1 is true, then Command 1 will be run. Otherwise, it will be skipped.

The setting method of the if/else sentence:

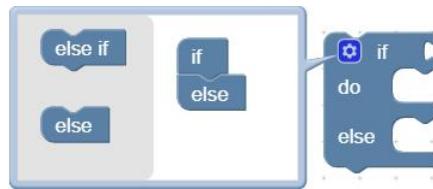


1. Click the setting button on the command block , then the

command block will pop up a selection box, as shown below:



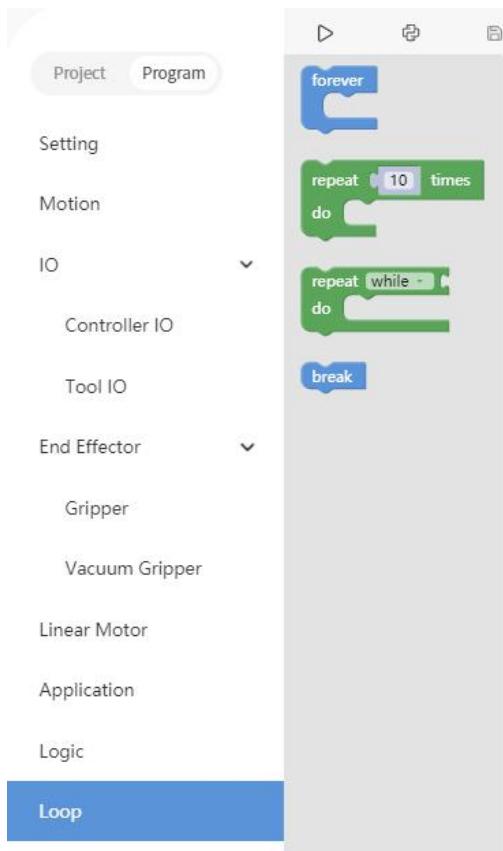
2. At this point, drag the [else] code block to the bottom of the [if] code block, and combine the two code blocks, as shown below:



3. Click the setting button , the selection box is retracted, if /else sentence setting is completed, as shown below:



## 1.6.10 Loop



【forever】

- The command contained in the loop will be executed in infinite loop.

【repeat() times do】

- The command contained in the loop will be executed X times.

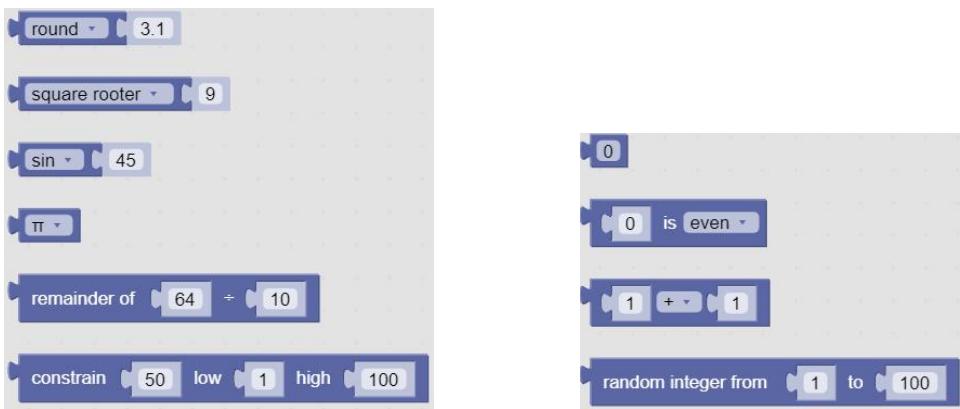
【repeat while/until do】

- When the condition is not met, it jumps out of the loop.

【break】

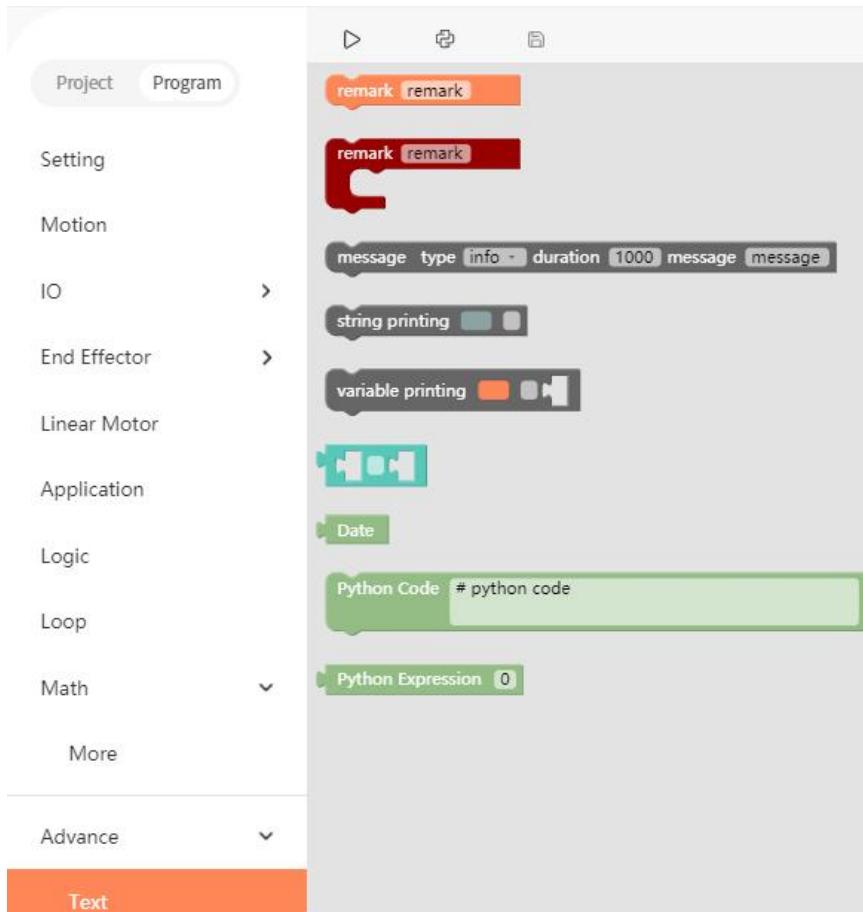
- Terminate the loop.

## 1. 6. 11 Math



You can use the above code block to do some complex operations such as addition, subtraction, multiplication, and division, exponential operations.

## 1. 6. 12 Text



【remark】

- Remark the code block, which serves as an indicator and can change the color.

#### 【message type】

- Types available are: (information/success/warning/error), duration indicates the time interval the message is displayed, the unit is in second; the message indicates the content of the prompt message.

#### 【string printing[]】

- Users can print the entered string below and set the font and the color.

#### 【variable printing】

- Users can print the added variable and set the font and the color.

#### 【Date】

- The date and time on which the command was run can be output.

#### 【Python Code】

- You can write custom python code to turn it into a block in Blockly and use it in your program



The screenshot shows the Blockly Python interface. At the top, there are parameters for 'set joint speed' (21 °/s) and 'set joint acceleration' (500 °/s<sup>2</sup>). Below these are fields for 'joint motion' with values J1: 45, J2: 9.9, J3: 31.8, J4: 0, J5: 21.9, J6: 0, Radius: 0, Wait: false. To the right of these fields are buttons for 'move' and 'edit'. A green box labeled 'Python Code' contains the following code:

```

set joint speed: 21 °/s
set joint acceleration: 500 °/s²

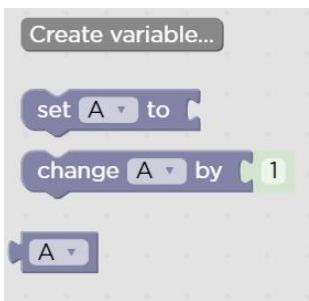
joint motion J1 45 J2 9.9 J3 31.8 J4 0 J5 21.9 J6 0 Radius 0 Wait false move edit

Python Code
#####
# get the angle of the robot
angle_ret=arm.get_servo_angle()[1]

## compare the angle of joint 1, if angel of joint 1 less than 0 degree, go to joint pos (0,9.9,31.8,0,21.9,0)
if angle_ret[0]>0 or angle_ret ==0:
    ret=arm.set_servo_angle(angle=[0.0,9.9,31.8,0.0,21.9,0.0])
    print('angle of joint 1 equal or bigger than 0 °, ret')
else:
    print('angle J1 less than 0° ')

```

### 1.6.13 Variable



【Create variable】

- New variables can be added. After adding a variable, there are three commands by default (set the value of the variable, change the value of the variable by adding or subtracting, variable).

【Rename variable】

- Rename the variable.

【Delete variable】

- Delete the variable.

### 1.6.14 Function



【to (do something)】

- Users can define a new function without a return value.

【to (do something) return []】

- Users can define a new function with a return value.

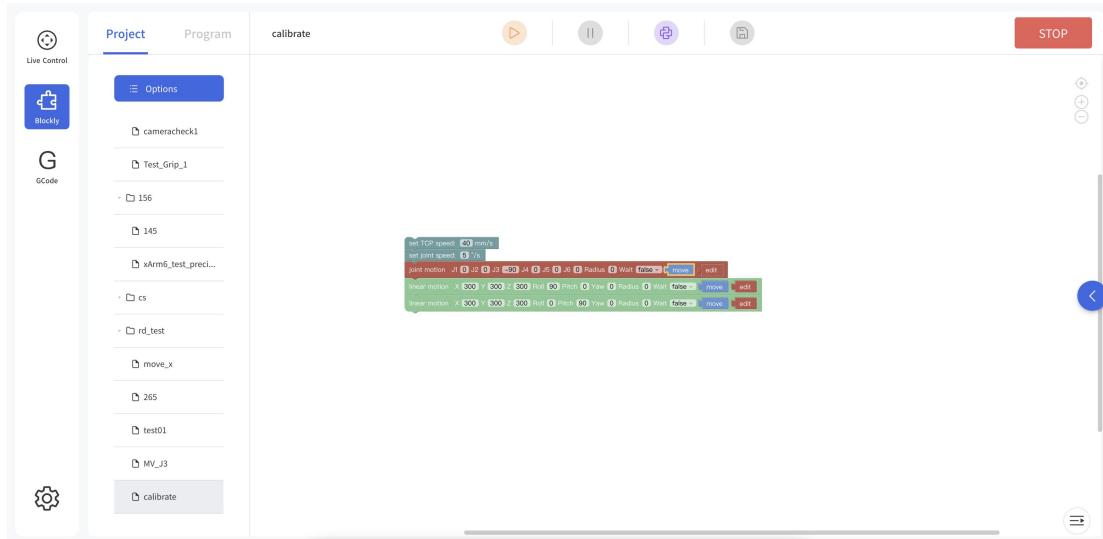
【if [] return []】

- Conditional judgment sentence that can only be placed in the built-in function.

Note:

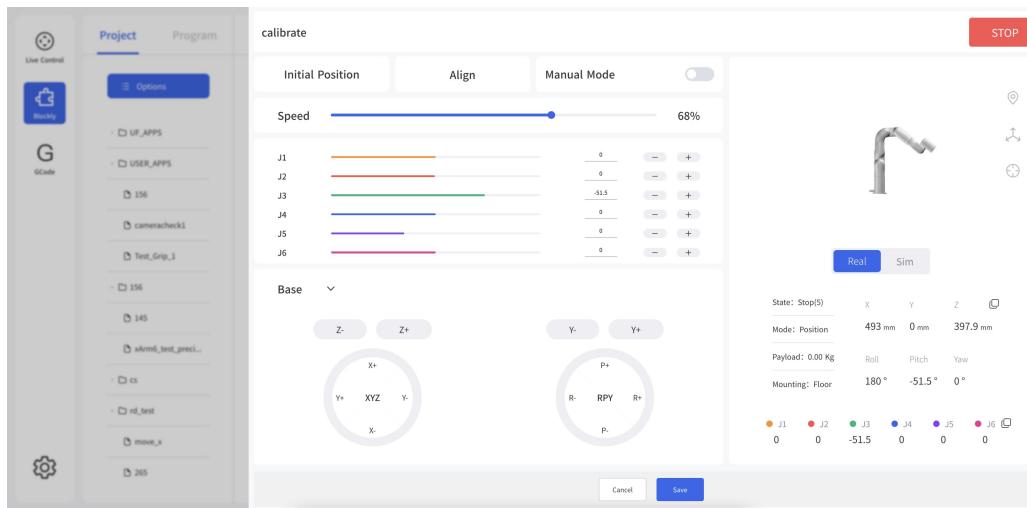
1. The defined function should be placed in front of the main programs.

## 1.6.15 Set & Edit Motion Coordinates



Long press **【Move】** button to move the robotic arm to the position of the current command.

Click **【edit】** to pop up the live control interface to re-edit the motion coordinates of the current command.



Click **【Save】** to save the changes and close the pop-up window.

Click **【Cancel】** to cancel the changes and close the pop-up window.

Note: In the command, there are sequential points such as A / B / C / D. Etc. If the user clicks **【move】** to skip point B from point A to point C, a safety assessment must be carried out to avoid damage to peripheral facilities

Due to the complexity of Cartesian commands, Cartesian spatial trajectory planning needs to be solved by inverse kinematics.

Therefore, there may be no solution, multiple solutions, approximate solutions. When the solution of the Cartesian command from point A to point B is not ideal, insert a third joint command between the two points if necessary.

### 1.6.16 Path Planning Guidelines

 <b>DANGER</b>	<ul style="list-style-type: none"><li>If the robotic arm is collided during the movement, resulting in stopping, the robotic arm will report an error at this time, and the error must be cleared before it can be used normally. Be sure to do a safety assessment before moving again to prevent collisions.</li></ul>
 <b>CAUTION</b>	<ul style="list-style-type: none"><li>When the robotic arm is in certain positions, there may be a situation where the linear motion is unsolvable. At this time, the route needs to be re-planned. For details, please refer to "xArm Kinematics-Linear Motion".</li></ul>

### 1.7 Python IDE



**【Python】** Click to enter Python IDE ,Python IDE is a Python development integration environment that can directly use xArm-Python-SDK API and

check the Blockly projects converted into Python code.

```

1  #!/usr/bin/env python3
2  # Software License Agreement (BSD License)
3  #
4  # Copyright (c) 2012, UFAC/INTech, Inc.
5  # All rights reserved.
6  #
7  # Author: Vilmar Vilmarwengfactory.rca <vilmar.cub@gmail.com>
8  #
9  #
10 # Notes: changes to this file on Studio will not be preserved
11 # Z. The next conversion will overwrite the file with the same name
12 #
13 # xmc-Python-SDK: https://github.com/xmc-Developer/xmc-Python-SDK
14 # 1. git clone https://github.com/xmc-Developer/xmc-Python-SDK.git
15 # 2. cd xmc-Python-SDK
16 # 3. python setup.py install
17 #
18 #
19 import sys
20 import math
21 import time
22 import serial
23 import random
24 import threading
25 from time import sleep
26 import version
27 from arduino import ArduAPI
28
29 class Robot:
30     def __init__(self, robot, **kwargs):
31         self._robot = robot
32         self._x = 0
33         self._y = 0
34         self._angle = 0
35         self._max_speed = 100
36         self._min_speed = -100
37         self._angle_speed = 20
38         self._angle_acc = 500
39         self._fracs = 0
40         self._last_update = 0
41         self._last_error = 0
42         self._last_error_code = 0
43         self._last_error_time = 0
44
45     def robot_initialize(self):
46         self._robot.initialize()
47         self._x = 0
48         self._y = 0
49         self._angle = 0
50         self._max_speed = 100
51         self._min_speed = -100
52         self._angle_speed = 20
53         self._angle_acc = 500
54         self._fracs = 0
55         self._last_update = 0
56         self._last_error = 0
57         self._last_error_code = 0
58         self._last_error_time = 0
59
60     def _robot_initialize(self):
61         self._robot.initialize()
62         self._x = 0
63         self._y = 0
64         self._angle = 0
65         self._max_speed = 100
66         self._min_speed = -100
67         self._angle_speed = 20
68         self._angle_acc = 500
69         self._fracs = 0
70         self._last_update = 0
71         self._last_error = 0
72         self._last_error_code = 0
73         self._last_error_time = 0
74
75     def _error_change(self, data):
76         if data and data['error_code'] != 0:
77             print("Error: " + str(data['error_code']))
78             self.errorprint("Error: " + str(data['error_code']))
79
80     def error_change(self, data):
81         if data and data['error_code'] != 0:
82             print("Error: " + str(data['error_code']))
83             self.errorprint("Error: " + str(data['error_code']))
84
85     def errorprint(self, err):
86         self._errorprint(err)
87         self._formatdata("error code")
```

New Folder  
New File  
Import  
Download  
Rename  
Delete

**【 Delete 】** Right mouse click on the Python file name to download, new folder, new file, import, rename or delete the file

**【 】** Run the program

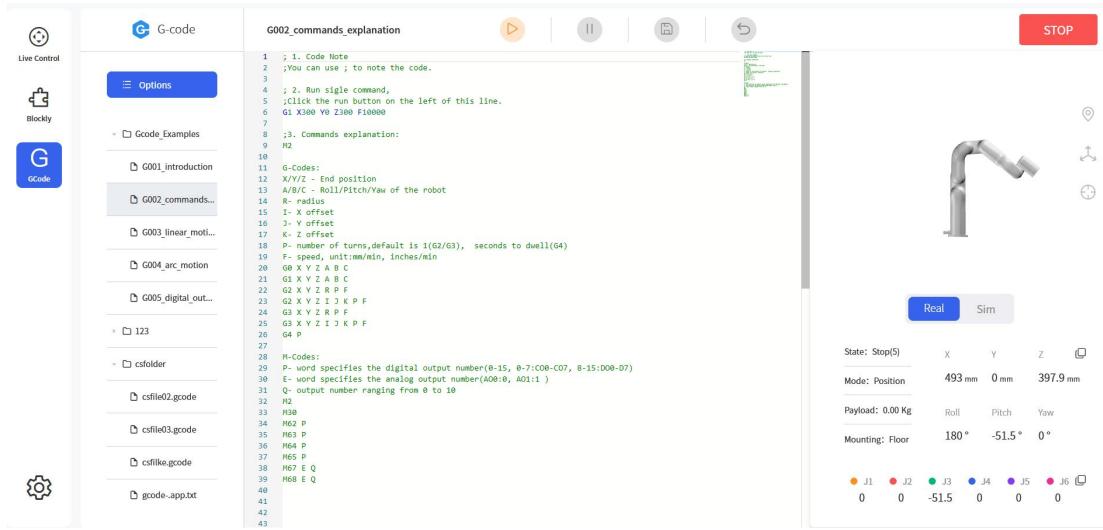
**【 】** Go back Blockly

**【 】** Save changes to Python code.

#### Control Box Command Caching Mechanism:

The current control box can cache 2048 commands. If more than 2048 commands need to be sent, user have to control the cached number and control the volume. When awaiting commands of the control box exceed the maximum buffer amount (2048), a warning code will be returned. The warning code is decimal 11, and the command will be discarded. The commands issued should not exceed 256 command caches. It is recommended to keep the number of cached/sent commands under 256 all the time.

## 1.8 GCode



Firmware version: V2.1.102+

TCP port: 504

G-Codes

G0, Rapid Move.

G1, Linear Move.

G2, Arc Move, clockwise arc.

G3, Arc Move, counterclockwise arc.

G4, Dwell.

G17, Z-axis, XY-plane.

G18, Y-axis, XZ-plane.

G19, X-axis, YZ-plane.

G20, to use inches for length units.

G21, to use millimeters for length units.

G90, absolute distance mode

G91, incremental distance mode

G90.1, absolute distance mode for I, J & K offsets.

G91.1, incremental distance mode for I, J & K offsets.

## M-Codes

M2/M30, end program  
M62, turn on digital output synchronized with motion.  
M63, turn off digital output synchronized with motion.  
M64, turn on digital output immediately.  
M65, turn off digital output immediately.  
M67, set an analog output synchronized with motion.  
M68, set an analog output immediately.

### Note:

1. Response:
  - byte0:return value. 0 is success
  - byte1:mode and state
  - byte2:error code
  - byte3&byte4:buffer
2. Recommend to send 1 non-empty data at a time (with line breaks)  
sock.send(b'G0 X300\\n')
3. Need to receive the response, otherwise the buffer will be full.  
sock.recv(5)

## 2. xArm Motion Analysis

In this section, we mainly use Python / Blockly examples to explain a few typical motions in the list below.

Motion	Joint Motion	Linear Motion	Arc linear motion	Circular Motion	xArm5 Motion
--------	--------------	---------------	-------------------	-----------------	--------------

### About Python-SDK:

For all interfaces with `is_radian`, the default value of `is_radian` is the value at the time of instantiation.

That is, the value of “`is_radian`” set when `xArmAPI()` is created.

Here are three examples to illustrate:

1. `arm = xArmAPI('192.168.1.226',)`
2. `arm = xArmAPI('192.168.1.226', is_radian=False)`
3. `arm = xArmAPI('192.168.1.226', is_radian=True)`

When the `xArmAPI()` interface is created in method 1, the default value of `is_radian` is False, the unit is ° ;

When the `xArmAPI()` interface is created in method 2, `is_radian = False`, the unit is ° ;

When the `xArmAPI()` interface is created in method 3, `is_radian = True`, the unit is rad ;

### About Blockly:

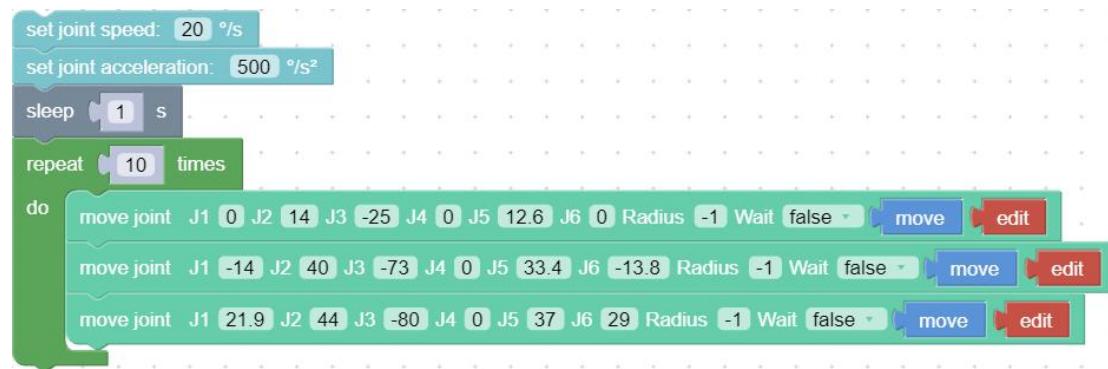
All units for angles use degrees (° ).

## 2.1. Motion of the Robotic Arm

### 2.1.1. Joint Motion

To achieve point-to-point motion in joint space (unit: degree), the speed is not continuous between each command.

Blockly example:



【Set joint speed() ° /s】 : Set the speed of joint movement in ° /s.

【Set joint acceleration() ° /s<sup>2</sup>】 : Set the acceleration of joint motion in ° /s<sup>2</sup>.

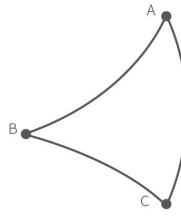
【move joint J1() J2() J3() J4() J5() J6() J7(),Radius()】 : Set each joint angle for the joint movement, the unit is ° .

【Wait (true / false)】 : indicates whether to wait for the execution of this command before sending the next command.

【Move】 : The robotic arm will move to the current position.

【Edit】 : Open the live control interface and adjust the coordinates of the current point.

The motion trajectory of the robotic arm in the above example is as follows:



Python example:

```
arm.set_servo_angle(angle=[0.0, 7.0, -71.2, 0.0, 0.0, 0.0], speed=8, mvacc=1145, wait=True)
arm.set_servo_angle(angle=[0.0, 7.0, -51.2, 0.0, 0.0, 0.0], speed=8, mvacc=1145, wait=True)
arm.set_servo_angle(angle=[0.0, 7.0, -91.2, 0.0, 0.0, 0.0], speed=8, mvacc=1145, wait=True)
```

The interface `set_servo_angle` is described in Table 2.1:

Table 2.1 Description of `set_servo_angle`

set_servo_angle		
description	set joint angle for joint motion	
parameter	servo_id	joint ID, 1-7, None or 8 means all joints: a) 1- (Number of axes) Joint number of the robotic arm E.g. : <code>arm.set_servo_angle (servo_id = 1, angle = 45, is_radian = False)</code> b) None (8) represents all joints E.g. : <code>arm.set_servo_angle (angle = [30, -45, 0, 0, 0, 0, 0], is_radian = False)</code>
	angle	angle Joint angle or list of joint angles (the unit of the default joint angle is <code>is_radian = False</code> , degrees (° )) a) If <code>servo_id</code> is 1- (joint number) E.g. : <code>arm.set_servo_angle (servo_id = 1, angle = 45, is_radian = False)</code> b) If <code>servo_id</code> is None or 8, E.g. : <code>arm.set_servo_angle (angle = [30, 45, 0, 0, 0, 0, 0], is_radian = False)</code>
	speed	joint speed (the default unit is ° / s): Unit: if <code>is_radian = True</code> , the unit is rad / s; if <code>is_radian = False</code> , the unit is ° / s;
	mvacc	joint acceleration (default unit is ° / s²) Unit: if <code>is_radian = True</code> , the unit is rad / s²; if <code>is_radian = False</code> , the unit is ° / s²;
	is_radian	roll / pitch / yaw Whether it is measured in radian (default <code>is_radian = False</code> )

		If is_radian = True, the unit of roll / pitch / yaw is radian; If is_radian = False, the unit of roll / pitch / yaw is degree (°);
	wait	If wait = True, wait for the current commands to finish before sending the next commands; If wait = False, send the next commands directly;
	mvtime	0, reserved;

Note:

1. If the joint angle is to be set in radian, then is\_radian = True;

ex: code = arm.set\_servo\_angle (servo\_id = 1, angle = 1.57, is\_radian = True)

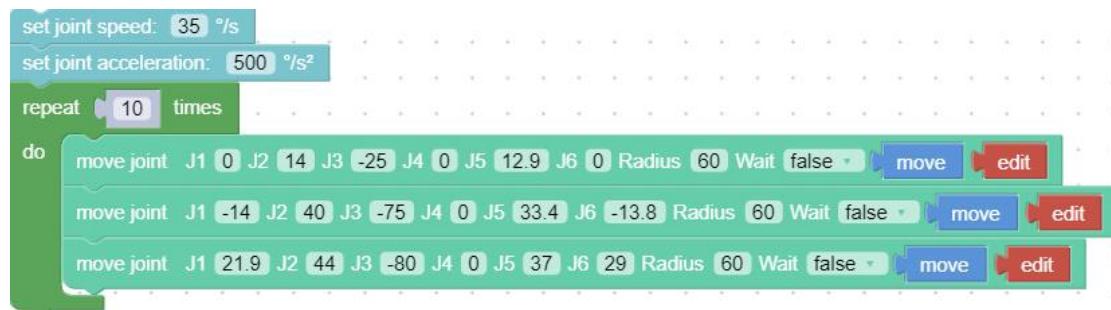
2. To wait for the robotic arm to complete the current commands before returning, wait = True;

ex: code = arm.set\_servo\_angle (servo\_id = 1, angle = 45, is\_radian = False, wait = True)

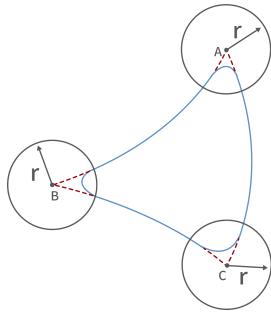
## Continuous Joint Motion

Inserting an arc transition between two joint motion commands is a way to plan the continuous joint motion of the robotic arm.

**Blockly:**



The motion trajectory of the robotic arm in the above example is as follows:



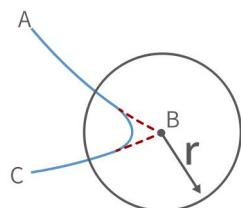
### Key parameter description

**Radius = 60**

Radius =60 in the "move joint" command refers to setting the radius of the transition arc  $R = 60\text{mm}$ , which is used to achieve a smooth transition of the arc in a joint motion.

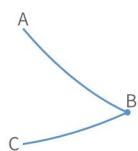
The parameters of Radius can be set as Radius> 0, Radius = 0, Radius = -1, different parameters correspond to different trajectories.

(1) Radius> 0. For example, setting Radius = 60, the turning trajectory is as shown in the arc in the figure below, which can achieve a smooth turning effect.



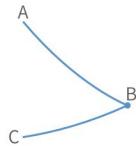
Note: The radius of the arc is smaller than  $D_{AB}$  and  $D_{BC}$ .

(2) Radius = 0. There is no arc transition at the turn, it will be a sharp turn with no deceleration, as shown in the figure below.



(3) Radius <0. There is no arc transition at the turn, this speed

will not be continuous between this and next motion, as shown in the figure below, speed will decelerate to 0 at point B before moving to C.



Note: Radius <0 cannot realize continuous motion. If you need to plan a continuous motion of the robotic arm, please make sure Radius>0.

**Wait = false**

The wait in the "move joint" command indicates whether it is necessary to wait for the execution of this command before sending the next command.

**Note:** If you need to plan for speed continuous motion, make sure wait = false, to buffer the commands to be blended.

### 2.1.1. Linear Motion and Arc Linear Motion

#### 2.1.1.1. Linear Motion

##### Characteristics of Linear Motion

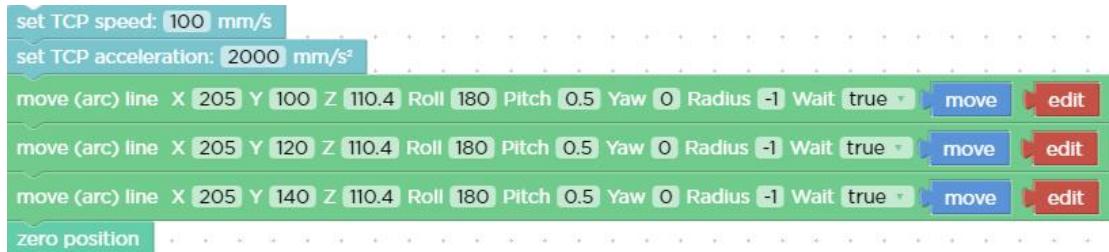
The concept of linear motion

- Straight linear motion between Cartesian coordinates (unit: mm), the speed is not continuous between each command.
- Users can control the motion of the robotic arm based on the base coordinate system and TCP coordinate system. The trajectory of tool center point in the Cartesian space is a straight line. Each joint performs a more complex movement to keep the tool in a straight path. The TCP path is unique once

the target point is confirmed, and the corresponding posture in the execution process is random. X, Y, and Z control the position of TCP in base or tool coordinate system, in the unit of mm. While Roll/Pitch/Yaw controls the TCP orientation in the unit of degree.

- Linear motion and circular linear motion belong to the Cartesian space trajectory planning, which needs to be solved by inverse kinematics. Therefore, there may be no solution, multiple solutions, and approximated solutions; and due to the nonlinear relationship between the joint space and Cartesian space, the joint motion may exceed its maximum speed and acceleration limits.

#### Blockly example:



【Set TCP speed () mm/s】 : Set the speed of the linear motion in mm/s.

【Set TCP acceleration () mm/s<sup>2</sup>】 : Set the acceleration of the linear motion in mm/s<sup>2</sup>.

【move(arc) line X() Y()Z() Roll() Pitch() Yaw() Radius()  
Wait(true/false) 【move】 【edit】】 : Indicates the Cartesian coordinate value of the linear motion and the TCP rotation angle in mm and ° .

Note: Cartesian motion is TCP straight-line motion.

#### Python example:

```
arm.set_tcp_jerk(2000)
```

```

arm.set_position(x=205.0, y=100.0, z=110.4, roll=180.0, pitch=0.5, yaw=0.0, speed=100,
radius=-1.0, wait=True)

arm.set_position(x=205.0, y=120.0, z=110.4, roll=180.0, pitch=0.5, yaw=0.0, speed=100,
radius=-1.0, wait=True)

arm.set_position(x=205.0, y=140.0, z=110.4, roll=180.0, pitch=0.5, yaw=0.0, speed=100,
radius=-1.0, wait=True)

arm.reset()

```

The interface `set_position()` is described in Table 2.2:

Table 2.2 `set_position` description

set_position		
Description	Sets the Cartesian coordinate value of the linear motion	
Parameter	x	coordinate x, (unit: mm)
	y	coordinate y, (unit: mm)
	z	coordinate z, (unit: mm)
	roll	attitude roll (default unit is °) : Unit: if <code>is_radian</code> = True, the unit is rad; if <code>is_radian</code> = False, the unit is ° ;
	pitch	attitude pitch (default unit is °) : Unit: if <code>is_radian</code> = True, the unit is rad; if <code>is_radian</code> = False, the unit is ° ;
	yaw	attitude yaw (default unit is °) : Unit: if <code>is_radian</code> = True, the unit is rad; if <code>is_radian</code> = False, the unit is ° );
	radius	radius: if it is a linear motion, <code>radius &lt; 0</code> / <code>radius = None</code> ; if it is arc linear motion(blended), <code>radius &gt; 0</code> ;
	is_radian	if <code>is_radian</code> = True, the unit of roll / pitch / yaw is rad; if <code>is_radian</code> = False, the unit of roll / pitch / yaw is ° ;
	speed	TCP motion speed (mm / s, rad / s);
	mvacc	TCP motion acceleration (mm / s <sup>2</sup> , rad / s <sup>2</sup> );
	mvtime	0, reserved;
	relative	if <code>relative</code> = True, it is relative motion; if <code>relative</code> = True, it is not relative motion;
	wait	if <code>wait</code> = True, wait for the current commands to finish before sending the next commands; if <code>wait</code> = False, send the next commands directly;

Note: If it is xArm5, roll and pitch must be set to roll = ± 180 ° and pitch = 0 ° .

### 2.1.1.2. Arc Linear Motion

Characteristics of Arc Linear Motion:

Arc linear motion (Lineb), inserting arc transitions between two straight lines, is a way to plan the continuous movement of the robotic arm. The following figure is a simple example of continuous motion using a circular arc linear motion planning robotic arm.

Note: When the xArm firmware version  $\geq 1.6.0$ , if you need to plan Lineb motion, you need to adjust the TCP speed below 200mm/s for debugging, otherwise there will be a high security risk.

Blockly:



Key parameter description

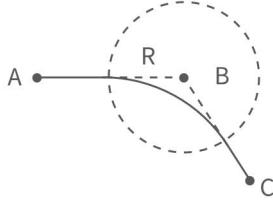
Radius = 5

Radius = 5 in the "move (arc) line" command refers to setting the radius of the transition arc between two straight lines  $R = 5\text{mm}$ , which is used to achieve a smooth transition of the arc in a straight motion.

The parameters of Radius can be set as Radius > 0, Radius = 0, Radius = -1, different parameters correspond to different trajectories.

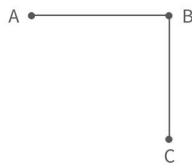
(4) Radius > 0. For example, setting Radius = 5, the turning trajectory is as shown in the black arc in the figure below, which

can achieve a smooth turning effect.



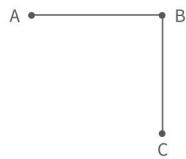
Note: The radius of the arc is smaller than  $D_{AB}$  and  $D_{BC}$ .

(5) Radius = 0. There is no arc transition at the turn, it will be a sharp turn with no deceleration, as shown in the figure below.



Note: If the motion of the robotic arm is a reciprocating linear motion, you need to set radius=0. If the radius>0, the robotic arm may report a motion planning error.

(6) Radius <0. There is no arc transition at the turn, this speed will not be continuous between this and next motion, as shown in the figure below, speed will decelerate to 0 at point B before moving to C.



Note: Radius <0 cannot realize continuous motion. If you need to plan a continuous movement of the robotic arm, please make sure Radius $\geqslant$ 0.

**Wait = false**

The wait in the "move (arc) line" command indicates whether it is necessary to wait for the execution of this command before sending the next command.

**Note:** If you need to plan for speed continuous motion, make sure wait = false, to buffer the commands to be blended.

### Python example:

```
arm.reset(wait=True)
arm.set_pause_time(0.5)

while True:
    arm.set_position(x=400, y=-100, z=250, roll=180, pitch=0, yaw=0, radius=50, speed=200,
wait=False)
    arm.set_position(x=400, y=100, z=250, roll=180, pitch=0, yaw=0, radius=50, speed=200,
wait=False)
    arm.set_position(x=300, y=0, z=250, roll=-180, pitch=0, yaw=0, radius=50, speed=200,
wait=False)
```

set\_position interface: refer to Table 2.2.

The set\_pause\_time interface is described in Table 2.3:

Table 2.3 set\_pause\_time description

set_pause_time		
Description	Set the robotic arm pause time	
Parameter	sltime	pause time, unit: second (s);
	wait	whether to wait, default is False;

### 2.1.2. Circular and Arc Motion

The circular motion calculates the trajectory of the spatial circle according to the coordinates of three points, which are (starting point, pose 1, pose 2).

The calculation method of three-point drawing circle:

Use the current point as the starting point, and then set two position points. Three points define a circle. Make sure these three points are not in a common line.

#### Set the center angle:

1. If  $0 < \text{center angle } (\circ) < 360^\circ$  or  $\text{center angle } (\circ) > 360^\circ$ , the

motion path of the robotic arm is a circular arc of the corresponding degree;

**center angle =  $60^\circ$** , the motion trajectory of the robotic arm is:



2. The center angle ( $^\circ$ ) =  $360^\circ$ , the movement track of the robotic arm is a complete circle;
3. If you want to draw multiple circles continuously (for example, draw 10 circles continuously), set center angles equal to  $3600^\circ$ ;

### Blockly example:



**【move circle position 1 to position 2】**: From current position, the whole circle is determined by current position and position1 and position2, “center angle” specifies how much of the circle to execute.

Note: (1) The starting point, pose 1 and pose 2 determine the three reference points of a complete circle. If the motion path of the robotic arm is a circular arc, then pose 1 and pose 2 are not necessarily end points or passing points;

(2) If you want the robot arm to change its posture during the movement, set the roll, pitch,

and yaw of pose 2 to the desired posture when completing the trajectory;

**【center angle (° ) ()】**: Indicates the degree of the circle. When it is set to 360, a whole circle can be completed, and it can be greater than or less than 360;

Note: To achieve smooth motion, you need to set Wait = false.

### **Example explanation:**

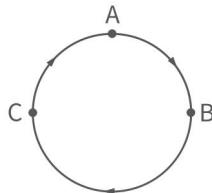
In this example, the central angle is set to  $3600^\circ$ , which means that the robotic arm can draw ten circles at a time, and the robotic arm still stays at the starting point after drawing a circle.

### **Judgment of the direction of the robotic arm motion**

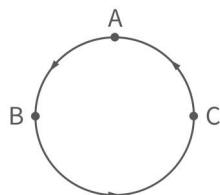
In the above example, the starting point, pose 1 and pose 2 are:

A (300, 0, 400, 180, 0, 0)    B (350, 50, 400, 180, 0, 0)    C (350, -50, 400, 180, 0, 0)

- The robotic arm draws a circle in a clockwise direction, and the trajectory of the robotic arm is as follows:



- If the positions of point B and C are swapped, point B is (350, -50, 400, 180, 0, 0), point C is (350, 50, 400, 180, 0, 0), the robotic arm will draw a circle in a counterclockwise direction. The trajectory of the robotic arm is as follows:



### **Python example:**

```
arm.set_servo_angle(angle=[0.0, -45.0, 0.0, 0.0, -45.0, 0.0], speed=20, mvacc=500, wait=True)
arm.set_position(*[300.0, 0.0, 400.0, 0.0, -90.0, 180.0], speed=300, mvacc=2000, radius=-1.0,
wait=True)
```

```
move_circle([350.0, 50.0, 400.0, 180.0, -90.0, 0.0], [350.0, -50.0, 400.0, 180.0, -90.0, 0.0],
1000.0, speed=300, mvacc=2000, wait=True)
```

set\_servo\_angle interface: see Table 2.1.

set\_position interface: see Table 2.2.

The move\_circle interface is described in Table 2.4:

Table 2.4 move\_circle description

move_circle		
Description	This motion calculates the trajectory of a space circle based on three-point coordinates. The three-point coordinates are (current starting point, pose 1, pose 2)	
Parameter	pose1	Cartesian coordinates [x(mm), y(mm), z(mm), roll(rad or °), pitch(rad or °), yaw(rad or °)];
	pose2	Cartesian coordinates [x(mm), y(mm), z(mm), roll(rad or °), pitch(rad or °), yaw(rad or °)];
	percent	Percentage of arc moved
	is_radian	If is_radian = True, the unit of roll / pitch / yaw is rad; If is_radian = False, the unit of roll / pitch / yaw is ° ;
	speed	TCP motion speed (mm / s, rad / s);
	mvacc	TCP motion acceleration (mm / s <sup>2</sup> , rad / s <sup>2</sup> );
	mvtime	0, reserved;
	wait	If wait = True, wait for the current commands to be sent before sending the next commands; If wait = False, send the next commands directly;

## 2.2. xArm5 Motion Characteristics

- Cartesian space

The movement of xArm5 is relatively special. Due to the structural limitation, the actual flexible degrees of freedom of linear and circular motions in Cartesian space is 4, which is [x, y, z, yaw], similar to a SCARA manipulator with four degrees of

freedom. Before starting Cartesian control, it is necessary to ensure that the end flange surface of xArm5 and the base are completely parallel. If mounted on horizontal plane, the roll and pitch should be [ $\pm$  180 degrees, 0 degrees], otherwise the trajectory is likely to have no solution.

- Joint space

In joint space, the robotic arm has 5 degrees of freedom to control and can switch to joint commands when different orientations are required at the end. Then use the joint command again to return the flange and the base to a horizontal attitude, and you can switch back to Cartesian control. A quick way to set a cartesian controllable attitude is: Just set the angle of J4 equal to -(J2 angle + J3 angle).

## 2.3. Singularity

### 1. Concept

Singularities occur when the axes of any two joints of a robotic arm are on the same straight line. At the singularity point, the robot's degrees of freedom will be degraded, which will cause the angular velocity of some joints to be too fast, leading to loss of control. A common situation is that when the wrist joint (the penultimate one) is at or near the axis of the first joint, singularity point will also appear (see Figure 2.1), so the robotic arm should try to avoid passing directly the central area near the base, which is likely to cause 1st Joint speed too high.

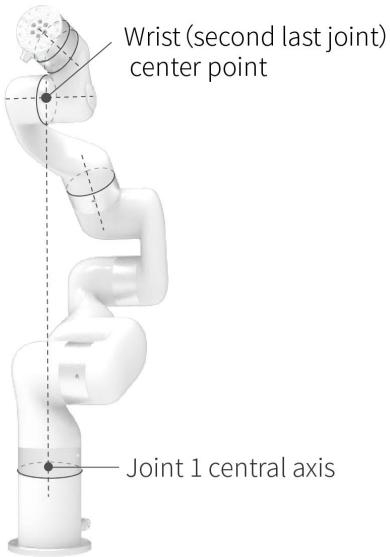


Figure 2.1 xArm6 singularity

## 2. Characteristics

The characteristic of the singularity is that the planning movement cannot be performed correctly. Coordinate-based planned movements cannot be explicitly translated into joint motions of each axis. When the robot performs motion planning (linear, circular, etc., excluding joint movements) near the singularity point, it will stop to avoid high instantaneous speed of the joint when it passes the singularity point. Therefore, try to avoid the singularity point or pass the singularity point through joint motion.

## 3. Processing method for singularity point

Case 1: Singularity encountered during robot teaching

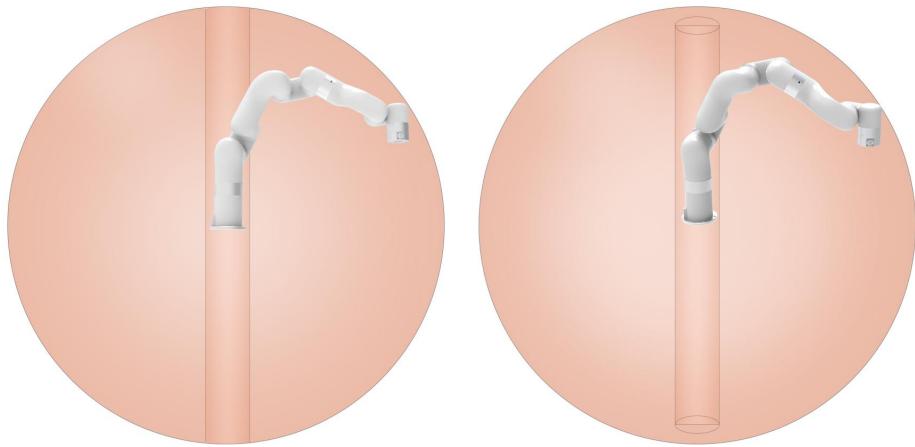
- a) Switch the robot coordinate system to a joint coordinate system, and pass the singularity point through joint motion.

Case 2: Singularities encountered while the program is running

- a) When encountering a singularity point while running the program, you can modify the position and attitude of the robot and re-plan the path to the target point.

**Note:**

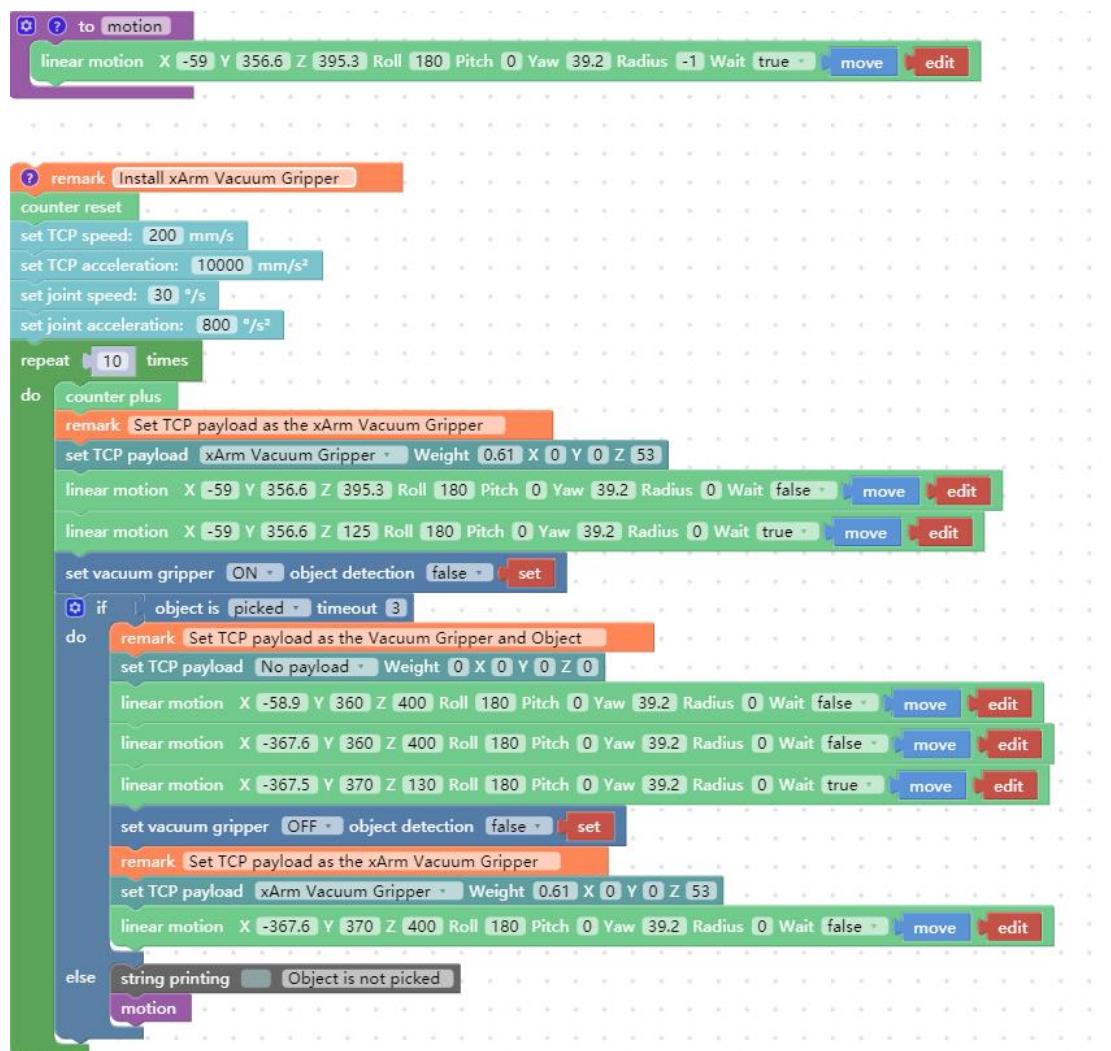
It is important to consider the cylindrical volume directly above and directly below the base of the robotic arm when a mounting place for the robotic arm is chosen. Moving the wrist joint (second last joint) close to the cylindrical volume should be avoided if possible, because it causes the joints to move fast even though the robotic arm is moving slowly, causing the robot to work inefficiently and making it difficult to conduct a risk assessment.



### 3. Typical Examples

There are 10 examples built into Blockly in UFactory Studio, which you can refer to for programming. Here are some of the more representative examples.

#### 3.1. The Use of xArm Vacuum Gripper



The role of this program: execute this program to control the vacuum gripper to suck the target object at the specified position, and then place the target object at the target position.

Explanation of main commands:

**【object is (picked/release)】**

- Detect whether the vacuum gripper has picked (released) the object, if it is detected that the vacuum gripper has picked (released) the object, then jump out of this command and execute the next command. If the timeout period is exceeded, the vacuum gripper has not yet picked (released) the object, it will also jump out of the command and execute the next command.

**【set xarm vacuum gripper (ON/OFF) object detection (true/false)**

**[set]】:**

- Set the vacuum gripper to be on and off.

**[object detection] = true:** detect whether the object is sucked, if not, it will jump out of the entire program.

**[object detection] = false:** do not detect whether the object is sucked.

- Cyclic motion count: By adding **【Counter plus】**, each time the command is run, the counter of the Control Box will be incremented by 1. It can be used to calculate the number of times the program cycles.

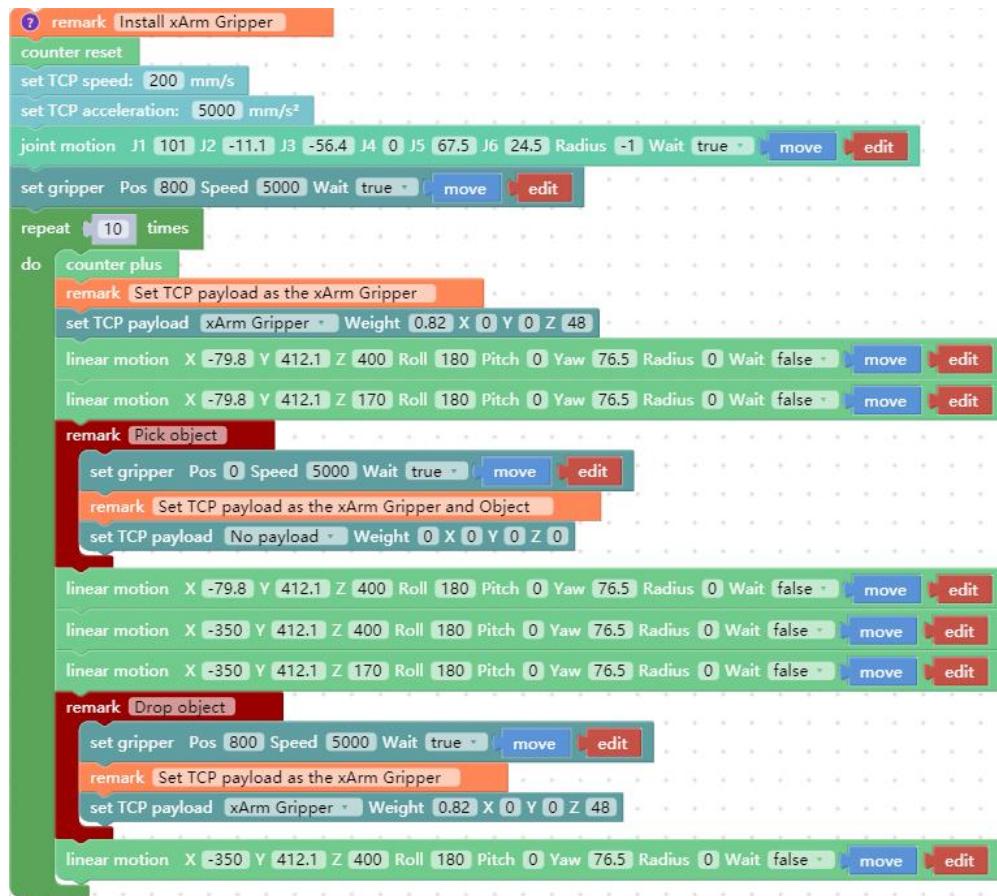
**【Counter reset】:** This command resets the counter in the Control Box to 0.

**Note:** If the robotic arm needs to count the cyclic motion, it is recommended to use the counter for counting.

**Note:**

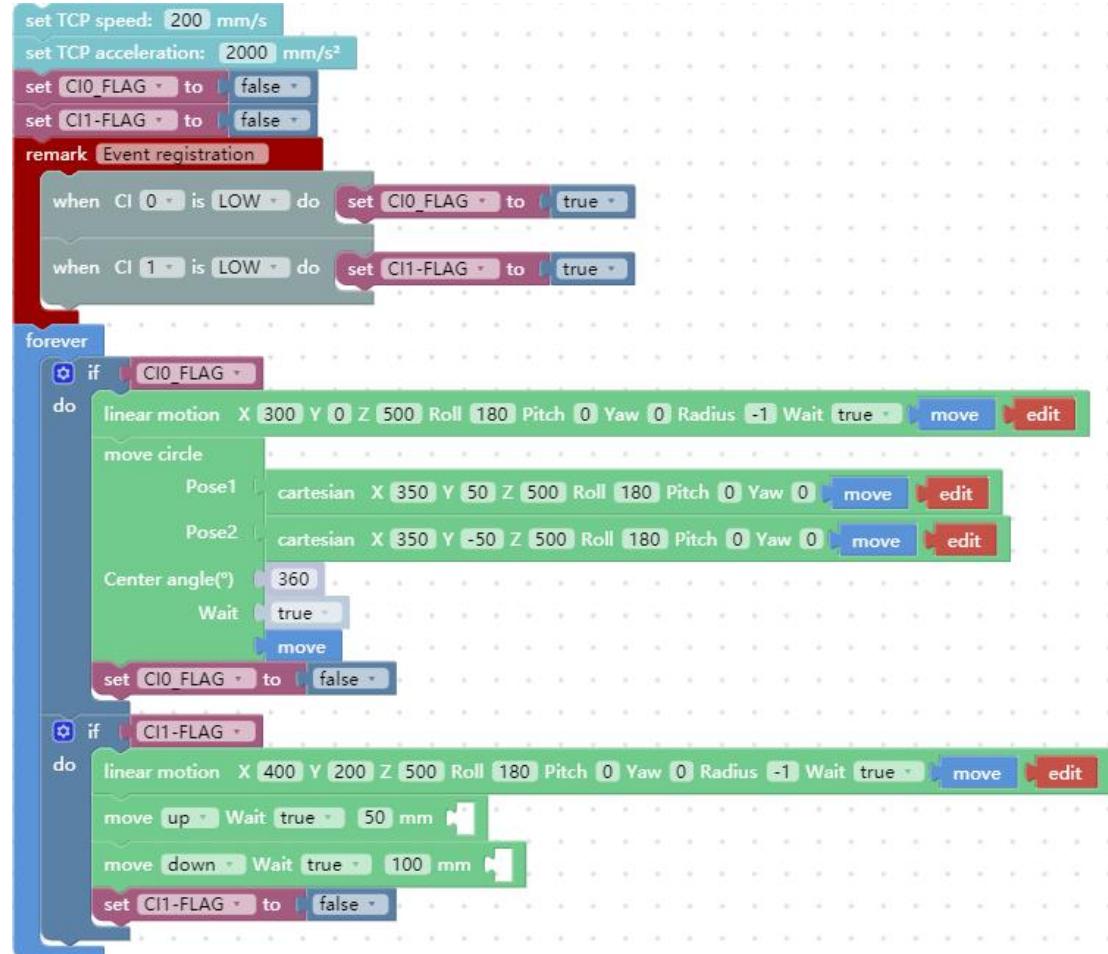
The defined function should be placed in front of the main programs, as shown in the figure above.

### 3.2. The Use of xArm Gripper



The role of this program: execute this program to control the gripper to grip the target object at the specified position, and then place the target object at the target position.

### 3.3. The Use of the Digital IO



The role of this program: If you need to use digital IO to control the motion of the robotic arm, you can trigger the digital IO to perform the corresponding motion.

## 4. Robotic Arm Motion Mode and State Analysis

### 4.1 Analysis of Robotic Arm Movement Mode

4 motion modes of the control box: (Python SDK: set\_mode ())

- Mode 0: Position control mode.

The control box enters by default after startup.

- Joint Motion: to achieve the point-to-point motion of joint space (unit:degree/radian), the speed between each command is discontinuous.

Click to see code:[set\\_servo\\_angle\(\)](#)

- Linear Motion: to achieve linear motion between Cartesian coordinates (unit: mm), the speed between each instruction is discontinuous.

Click to see code:[set\\_position\(\)](#) [set\\_position\\_aa\(\)](#)

- Arc Linear Motion: to achieve linear motion between Cartesian coordinates (unit: mm), inserting an arc between two straight lines for a smooth transition, and the speed between each command is continuous.

Click to see code:[move\\_arc\\_lines\(\)](#)

- Circular Motion: Circular motion calculates the trajectory of the spatial circle according to the three-point coordinates, the three-point coordinates are starting point, parameter 1 and parameter 2.

Click to see code: [move\\_circle\(\)](#)

- Mode 1: Servoj mode.

- Servoj motion: move to the given joint position with the fastest speed ( $180^\circ /s$ ) and acceleration (unit: degree/radian). This command has no buffer, only execute the latest received target point, and the user needs to enter the servoj mode to use. In servoj mode, the maximum receiving frequency of the control box is 250 Hz (the maximum receiving frequency of the version before 1.4.0 is 100 Hz). If the frequency of sending commands exceeds 250Hz, the redundant commands will be lost. The xArm-Python-SDK interface function we provide also reserves the speed, acceleration and time settings, but they will not work at present. The suggested way of use: If you want to plan your track, you can use this command to issue a smoothed track point with interpolation at a certain frequency (preferably 100Hz or 200 Hz), similar to the position servo control command. (Note: this execution is similar to the step response, for safety considerations, do not give a distant target position at once). Using this mode requires detailed position planning for each axis and motion estimation of the robotic arm, which is difficult to develop.

Click to see code: [set\\_servo\\_angle\\_j\(\)](#)

- Servo\_cartesian motion: move to the given cartesian position with the fastest speed (1 m/s) and acceleration

(unit: mm). This command has no buffer, only execute the latest received target point, and the user needs to enter the servo<sub>j</sub> mode to use. In servo<sub>j</sub> mode, the maximum receiving frequency of the control box is 250 Hz (the maximum receiving frequency of the version before 1.4.0 is 100 Hz). If the frequency of sending commands exceeds 250 Hz, the redundant commands will be lost. The xArm-Python-SDK interface function we provide also reserves the speed, acceleration and time settings, but they will not work at present. The suggested way of use: If you want to plan your track, you can use this command to issue a smoothed track point with interpolation at a certain frequency (preferably 100 Hz or 200 Hz), similar to the position servo control command. (Note: this execution is similar to the step response, for safety considerations, do not give a distant target position at once). It is recommended that the frequency of user issuing commands be controlled within the range of 30 Hz–250 Hz. If the frequency is lower than 30 Hz, the motion of the robotic arm may be discontinuous. Using this mode requires planning the fine position of each axis and predicting the motion behavior of the robotic arm, which is difficult to develop.

Click to see code:[servo\\_cartesian\(\)](#), [servo\\_cartesian\\_aa\(\)](#)

## ● Mode 2: Joint teaching mode.

In this mode, the robotic arm will enter the zero gravity mode, and the user can freely drag the links of the robotic arm to complete the teaching function. If the drag teaching

is completed, switch back to mode 0.

Note for safe use: Before turning on the joint teaching mode, be sure to confirm that the installation direction of the robotic arm and the TCP load are set correctly, otherwise the arm may not be able to remain stationary due to inaccurate gravity compensation

Click to see code:[record\\_trajectory\(\)](#), [playback\\_trajectory\(\)](#)

- Mode 4: Joint velocity control mode.

Click to see code:[vc\\_set\\_joint\\_velocity\(\)](#)

- Mode 5: Cartesian velocity control mode

Click to see code:[vc\\_set\\_cartesian\\_velocity\(\)](#)

- Mode 6: Joint online trajectory planning mode

Command sent by `set_servo_angle()` In this mode, every time a motion command is received, the current motion command will be interrupted, and then the motion command will be planned and executed from the current position, and the latter motion command can be interrupted Ongoing movement guide

Click to see code: [mode\(6\)](#)

- Mode 7: Cartesian online trajectory planning mode

Command sent by `set_position()` or `set_position_aa()` In this mode, every time a motion command is received, the current motion command will be interrupted, and then the motion command will be planned and executed from the current position, and the latter motion command can be interrupted Ongoing movement guide

Note: When using Cartesian online planning mode, the ‘`is_tool_coord`’ in `set_position_aa()` must be False, that is,

Cartesian online planning mode can only use the base coordinate system as the reference coordinate system, not the tool coordinate system for relative motion.

Click to see code:[mode\(7\)](#)

Operators who design the robotic arm motion path must be qualified with the following conditions:



**CAUTION**

1. The operator should have a strong sense of security consciousness, and sufficient knowledge on robotic arm operations.
2. The operator should have in-depth knowledge on the robotic arm and understands the joint motion mode and the linear motion mode.
3. The operator should have safety knowledge on emergencies.
4. When planning the path for the robotic arm, the risk assessment must be done and the operator should be cautious.

## 4.2 Analysis of the Motion Status of the Robotic Arm

3 states that the control box can set: (Python SDK: `set_state()`)

- State 0: Start motion.

Can be understood as ready for motion or stand-by. In this state, the robotic arm can normally respond to and execute motion commands. If the robotic arm recovers from an error, power outage, or stop state (state 4), remember to set the state to 0 before continuing to send motion commands.

Otherwise the commands sent will be discarded.

- State 3: Paused state.

Pause the currently executing motion and resume the motion at the interruption by setting state 0 again.

- State 4: Stop state.

Terminates the current motion and clears the cached subsequent commands. Need to set state 0 to continue the motion.

6 states that the control box can get: (Python SDK: `get_state ()`)

- State 1: In motion.

The robotic arm is executing motion commands and is not stationary.

- State 2: Standby.

The control box is already in motion ready state, but no motion commands are cached for execution.

- State 3: Pausing.

The robotic arm is set to pause state, and the motion commands buffer may not be empty.

- State 4: Stopping.

This state is the state entered by default upon power-on. Stop and on commands can be executed until state is set to 0.

- State 5: System reset.

The user just enters the state after the mode switch or changes some settings (such as TCP offset, sensitivity, etc.). The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.

- State 6: Stop.

Generally, use studio will generate state 6, and state 6 is also a kind of stop.

# Appendix

## Appendix1-Error Reporting and Handling

### 1.1 Joints Error Message and Error Handling

- Error processing method: Re-power on, the steps are as follows:
  1. Turn the emergency stop button on the control box
  2. Enable the robotic arm
- UFactory studio enable method: Click the guide button of the error pop-up window.
- xArm-Python-SDK enable method: [Refer to Error Handling Mode.](#)
- xArm-ROS-library: Users can view related documents at  
[https://github.com/xArm-Developer/xarm\\_ros](https://github.com/xArm-Developer/xarm_ros)  
[https://github.com/xArm-Developer/xarm\\_ros2](https://github.com/xArm-Developer/xarm_ros2)
- If the problem remains unsolved after power on/off for multiple times, please contact UFACTORY team for support.

Software Error Code	Error Handling
S0	Joint Communication Error Please restart the xArm with the Emergency Stop Button on the Control Box. If multiple reboots do not work, please contact technical support.

S10	Abnormal Current Detection Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S11	Joint Overcurrent Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S12	Joint Overspeed Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S14	Position Command Overlimit Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S15	Joints Overheat If the robot arm is running for a long time, please stop running and restart the xArm after it cools down.
S16	Encoder Initialization Error Please ensure that no external force pushes the robot arm to move when it's powered on. Please restart the xArm with the Emergency Stop Button on the Control Box.
S17	Single-turn Encoder Error Please restart the xArm with the Emergency Stop Button on the Control Box.
S18	Multi-turn Encoder Error Please click "Clear Error", then push the power switch of the Control Box to OFF, wait 5 seconds and then power on again.
S19	Low Battery Voltage Please contact technical support.
S20	Driver IC Hardware Error Please re-enable the robot.
S21	Driver IC Initialization Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S22	Encoder Configuration Error Please contact technical support.
S23	Large Motor Position Deviation Please check whether the xArm movement is blocked, whether the payload exceeds the rated payload of xArm, and whether the acceleration value is too large.
S26	Joint N Positive Overrun Please check if the angle value of the joint N is too large.

S27	<p><b>Joint N Negative Overrun</b>            Please check if the angle value of the joint N is too large, if so, please click Clear Error and manually unlock the joint and rotate the joint to the allowed range of motion.</p>
S28	<p><b>Joint Commands Error</b>            The xArm is not enabled, please click Enable Robot.</p>
S33	<p><b>Drive Overloaded</b>            Please make sure the payload is within the rated load.</p>
S34	<p><b>Motor Overload</b>            Please make sure the payload is within the rated load.</p>
S35	<p><b>Motor Type Error</b>            Please restart the xArm with the Emergency Stop Button on the xArm Control Box.</p>
S36	<p><b>Driver Type Error</b>            Please restart the xArm with the Emergency Stop Button on the xArm Control Box.</p>
S39	<p><b>Joint Overvoltage</b>            Please reduce the acceleration value in the Motion Settings.</p>
S40	<p><b>Joint Undervoltage</b>            Please reduce the acceleration value in the Motion Settings.            Please check if the control box emergency stop switch is released.</p>
S49	<p><b>EEPROM Read and Write Error</b>            Please restart the xArm with the Emergency Stop Button on the xArm Control Box.</p>
S52	<p><b>Initialization of Motor Angle Error</b>            Please restart the xArm with the Emergency Stop Button on the xArm Control Box.</p>
For alarm codes that are not listed in the above table: Power on again. If the problem remains unsolved after power on/off for multiple times, please contact technical support.	

## 1.2 Control Box Error Code and Error Handling

### 1.2.1 Control Box Error Code

If there is an error in the hardware of the robotic arm/the software of the Control Box/in sending commands, an error or warning will be

issued. This error/warning signal will be fed back when the user sends any command; that is, the feedback is passive and not actively reported.

After the above error occurs, the robotic arm will stop working immediately and discard the Control Box cache command. Users need to clear these errors manually to allow normal operation. Please re-adjust the motion planning of the robotic arm according to the reported error message.

Software Error Code	Error Handling
C1	The Emergency Stop Button on the Control Box is Pushed in to Stop please release the Emergency Stop Button, and then click "Enable Robot"
C2	The Emergency IO of the Control Box is triggered Please ground the 2 EIs of the Control Box, and then click "Enable Robot".
C3	The Emergency Stop Button of the Three-state Switch is pressed Please release the Emergency Stop Button of the Three-state Switch, and then click "Enable Robot".
C11-C17	Power on again.
C19	End Module Communication Error Please check whether gripper is installed and the baud rate setting is correct.
C21	Kinematic Error Please re-plan the path.
C22	Self-collision Error, Please Re-plan the Path.  If the robotic arm continues to report self-collision errors, please go to the "live control" interface to turn on the "manual mode" and drag the robotic arm back to the normal position.
C23	Joints Angle Exceed Limit  Please go to the "Live Control" page and press the "Initial POSITION" button to let the robot back to the Initial position.
C24	Speed Exceeds Limit  Please check if the xArm is at singularity point, or reduce the speed and acceleration values.
C25	Planning Error  Please re-plan the path or reduce the speed.

C26	Linux RT Error Please contact technical support.
C27	Command Reply Error Please retry, or restart the xArm with the Emergency Stop Button on the Control Box. If multiple reboots are not working, please contact technical support.
C29	Other Errors Please contact technical support.
C30	Feedback Speed Exceeds Limit Please contact technical support.
C31	Abnormal current in the robotic arm 1. Check whether the robotic arm collides. 2. Check whether the mass and center of mass set at "Settings"--"TCP Settings"--"TCP Payload" match the actual payload. 3. Check whether the mounting direction set at "Settings"--"Mounting" matches the actual situation. 4. Check whether the TCP payload parameters set in your program match the actual payload. 5. Reduce the motion speed of the robotic arm. 6. Go to "Settings"--"Motion"--"Sensitivity Settings" to lower the collision sensitivity.
C32	Three-point Drawing Circle Calculation Error please reset the arc command.
C33	Controller IO Error If the error occurs repeatedly, please contact technical support.
C34	Recording Timeout The track recording duration exceeds the maximum duration limit of 5 minutes. It is recommended to re-record.
C35	Safety Boundary Limit The xArm reaches the safety boundary. Please let the xArm work within the safety boundary.
C36	The number of delay commands exceeds the limit 1. Please check whether there are too many position detection or IO delay commands. 2. Increase the tolerance of the position detection command.
C37	Abnormal Motion in Manual Mode Please check whether the TCP payload setting of the robotic arm and the installation method of the robotic arm match the actual settings.
C38	Abnormal Joint Angle Please stop the xArm by pressing the Emergency Stop Button on the Control Box.

C39	Control Box Power Board Master and Slave IC Communication Error Please contact technical support.
C50	Six-axis Force Torque Sensor Error Please check the sensor error code, locate the problem, and power on again. If it cannot be resolved, please contact technical support.
C51	Six-axis Force Torque Sensor Mode Setting Error Please make sure that the robotic arm is not in Manual Mode, check whether the given value of this command is 0/1/2
C52	Six-axis Force Torque Sensor Zero Setting Error Please check the sensor communication wiring and whether the power is normal.
C53	Six-axis Force Torque Sensor Overload Please reduce the payload or applied external force.
C110	Robot Arm Base Board Communication Error. Please contact technical support.
C111	Control Box External 485 Device Communication Error Please contact technical support.
For alarm codes that are not listed in the above table: Power on again. If the problem remains unsolved after power on/off for multiple times, please contact technical support.	

### 1.2.2 Control Box Error Code

The error does not affect the normal operation of the robotic arm, but it may affect the user's program operation. Once the warning occurs, the arm will set the warning flag and return it together in the command reply. Otherwise, no other operations will be performed. The robotic arm will still operate normally.

Error code	Description	Error Handling
11	Buffer overflow	Control the volume of command cache
12	Command parameter abnormal	Check sent command
13	Unknown Command	Check sent command
14	Command no solution	Check sent command

### 1.3 Gripper Error Code & Error Handling

The user can re-power on the robotic arm as an error handling, the steps are as follows (all the following steps are needed) :

1. Re-powering the robotic arm via the emergency stop button on the control box.
2. Enable the robotic arm.
  - a. UFactory studio enable method: Click the guide button of the error pop-up window or the ‘STOP’ red button in the upper right corner.
  - b. xArm-Python-SDK enable method: [Refer to Error Handling Method.](#)
  - c. xArm\_ROS library:

ROS      [https://github.com/xArm-Developer/xarm\\_ros](https://github.com/xArm-Developer/xarm_ros)

ROS2    [https://github.com/xArm-Developer/xarm\\_ros2](https://github.com/xArm-Developer/xarm_ros2)

3. Re-enable the gripper.

If the problem remains unsolved after power on/off multiple times, please contact UFACTORY team for support.

Software Error Code	Error Handling
G9	<p>Gripper Current Detection Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.</p>
G11	<p>Gripper Current Overlimit Please click “OK” to re-enable the Gripper.</p>
G12	<p>Gripper Speed Overlimit Please click “OK” to re-enable the Gripper.</p>

G14	Gripper Position Command Overlimit Please click “OK” to re-enable the Gripper.
G15	Gripper EEPROM Read and Write Error Please click “OK” to re-enable the Gripper.
G20	Gripper Driver IC Hardware Error Please click “OK” to re-enable the Gripper.
G21	Gripper Driver IC Initialization Error Please click “OK” to re-enable the Gripper.
G23	Gripper Large Motor Position Deviation Please check if the movement of the Gripper is blocked, if not, please click “OK” to re-enable the Gripper.
G25	Gripper Command Over Software Limit Please check if the gripper command is set beyond the software limit.
G26	Gripper Feedback Position Software Limit Please contact technical support.
G33	Gripper Drive Overloaded Please contact technical support.
G34	Gripper Motor Overload Please contact technical support.
G36	Gripper Driver Type Error Please click “OK” to re-enable the Gripper.
For alarm codes that are not listed in the above table: enable the robotic arm and gripper. If the problem remains unsolved after power on/off for multiple times, please contact technical support.	

## 1.4 Python SDK Error Code & Error Handling

Software Error Code	Error Handling
A-9	Emergency Stop
A-8	The TCP position command is out of the robot arm's motion range. Please adjust the TCP position command.
A-2	xArm is not ready. Please check whether the robot is enabled and the state is set correctly.

A-1	xArm is disconnect or not connect. Please check the network.
A1	There are errors that have not been cleared. Please clear the errors and try again.
A2	There are warnings that have not been cleared. Please clear the warnings and try again.
A3	Get response timeout. Please check the firmware version and the network.
A4	TCP reply length error. Please check the network.
A5	TCP reply number error. Please check the network.
A6	TCP protocol flag error. Please check the network.
A7	The TCP reply command does not match the sending command. Please check the network.
A8	Send command error. Please check the network.
A9	xArm is not ready. Please check whether the errors have been cleared, whether the robot arm has been enabled, and whether the robot arm status is set correctly.
A11	Other error. Please contact technical support.
A12	Parameter error.
A20	Tool IO ID error.
A22	The end tool Modbus baud rate is incorrect.
A23	The end tool Modbus reply length error.
A31	Trajectory read/write failed.
A32	Trajectory read/write timeout.
A33	Playback trajectory timeout.
A41	Vacuum gripper wait timeout.
A100	Waiting for completion timeout.
A101	Too many failures to detect the status of the end effector.
A102	There are errors in the end effector
A103	The end effector is not enabled

For alarm codes that are not listed in the above table: enable the robotic arm and gripper. If the problem remains unsolved after power on/off for multiple times, please contact technical support.

### xArm-Python-SDK Error Handling:

When designing the robotic arm motion path with the Python library, if the robotic arm error (see Appendix for Alarm information) occurs, then it needs to be cleared manually. After clearing the error, the robotic arm should be motion enabled.

Python library error clearing steps: (Please check GitHub for details on the following interfaces)

- a. error clearing: `clean_error()`
- b. Re-enable the robotic arm: `motion_enable(true)`
- c. Set the motion state: `set_state(0)`

## Appendix2–Technical Specifications

### 1.1 xArm5/6/7 Common Specifications

xArm		
Cartesian Range	X	± 700 mm
	Y	± 700 mm
	Z	-400 mm ~ 951.5 mm
	Roll/Yaw/Pitch	± 180 °
Maximum Joint Speed	180 ° /s	
Reach	700 mm	
Repeatability	± 0.1 mm	
Max Speed of End-effector	1 m/s	
*Ambient Temperature Range	0~50 ° C*	
Power Consumption	Min 8.4 W, Typical 200 W, Max 500 W	
Input Power Supply	24 V DC, 16.5 A	
ISO Class Cleanroom	5	
Robotic Arm Mounting	Any	
Programming	UFactory studio/Python/C++/ROS	
Robotic Arm Communication Protocol	Private TCP	
End-effector I/O Interface	2 Digital inputs, 2 Digital outputs, 2 Analog inputs	
End-effector Communication Protocol	Modbus TCP	
Footprint	Ø 126 mm	
Materials	Aluminium, Carbon Fiber	
End Tool Flange	DIN ISO 9409-1-A50/63 (M5*6)	
Control Box		
	AC Control Box	DC Control Box
Input	100~240V AC 50/60 Hz	24V DC
Output	24V DC 20.8 A	24V DC 16.5 A
Control Box Communication Protocol	Private TCP	
Control Box Communication Model	Ethernet	
Control Box I/O Interface	8*CI+8*DI (Digital In) 8*CO+8*DO (Digital Out)  2*AI (Analog In) 2*AO (Analog Out)  1*RS-485 Master  1*RS-485 Slave	8*CI (Digital In) 8*CO (Digital Out)  2*AI (Analog In) 2*AO (Analog Out)
Weight	3.9 kg	2.6 kg

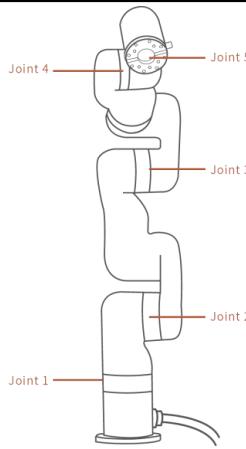
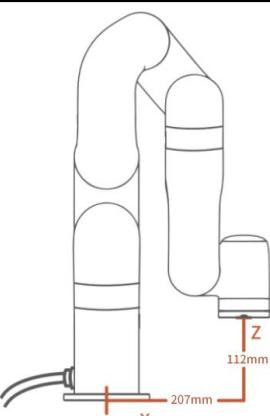
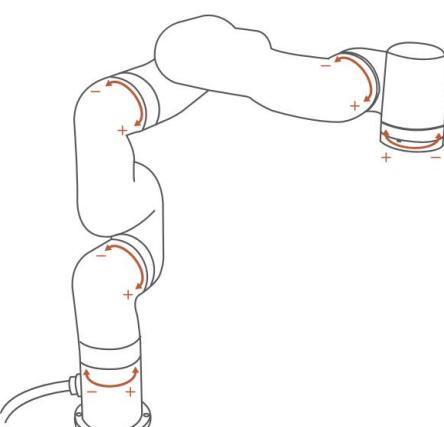
Dimension (L*W*H)	285*135*101 mm	262*160*75 mm
-------------------	----------------	---------------

### xArm accessories parameters:

Gripper	
Nominal Supply Voltage	24V DC
Absolute Maximum Supply Voltage	28V DC
Quiescent Power (Minimum Power Consumption)	1.5 W
Peak Current	1.5 A
Working Range	84 mm
Maximum Clamping Force	30 N
Weight (g)	802 g
Communication Mode	RS-485
Communication Protocol	Modbus TCP
Programmable Gripping Parameters	Position, Speed
Feedback	Position
Vacuum Gripper	
Rated Supply Voltage	24V DC
Absolute Maximum Supply Voltage	28V DC
Quiescent Current (mA)	30 mA
Peak Current (mA)	400 mA
Vacuum	78 %
Vacuum Flow (L/min)	> 5.6 L/min
Weight (g)	610 g
Dimensions (L*W*H)	122.5 * 91.6 * 75 mm
Payload (kg)	≤5 kg
Noise Level (30cm away)	< 60 dB
Communication Mode	Digital IO
State Indicator	Power, Working State
Feedback	Air Pressure (Low or Normal)
Notes:	
1. The ambient temperature of xArm is 0–50 ° C, please reduce the temperature if continuous high-speed operation is needed.	

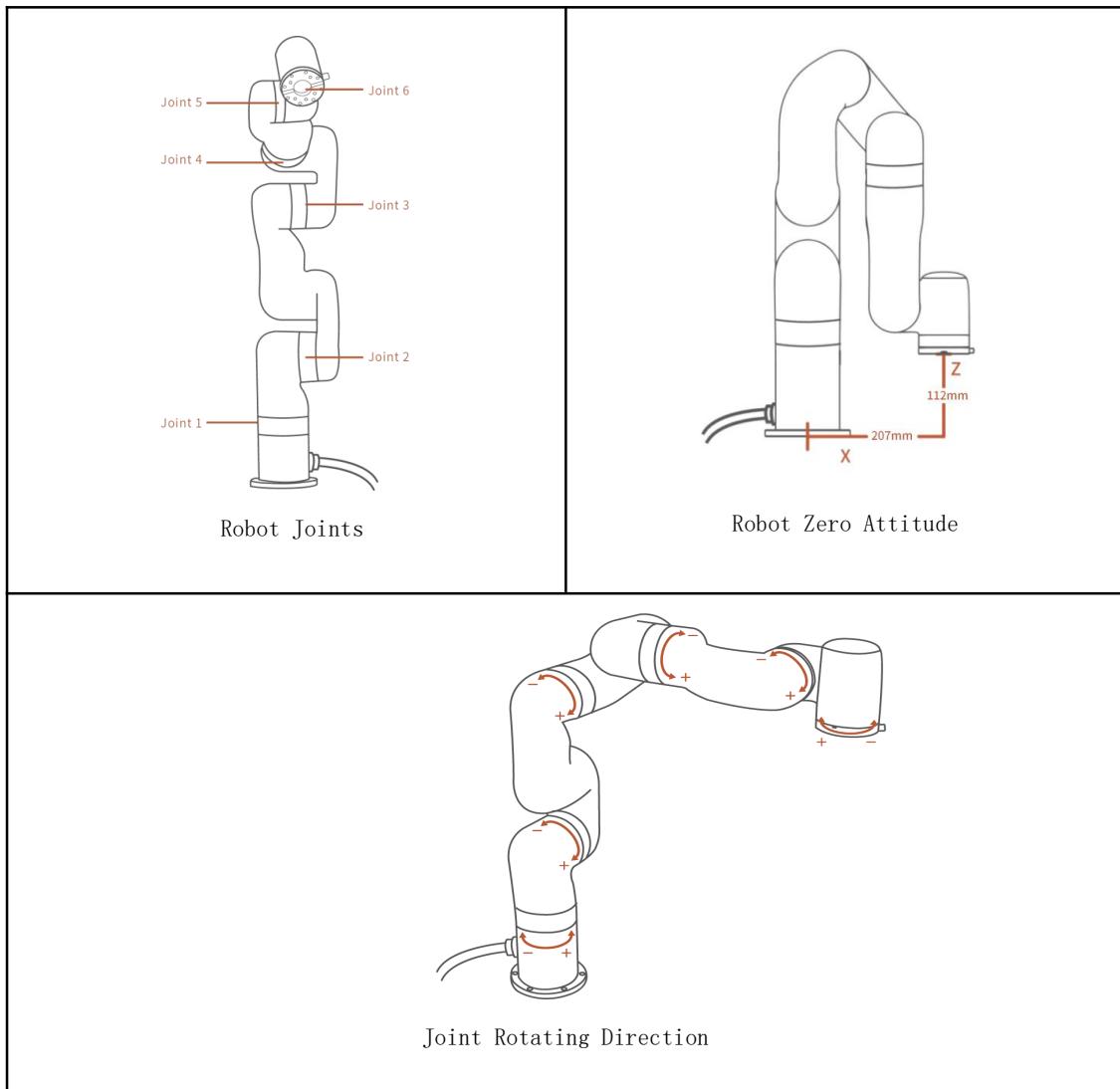
## 1.2 xArm 5 Specifications

Joint Range	1, 5	±360°
	2	-118° ~120°
	3	-225° ~11°
	4	-97° ~180°
Payload		3kg

Degrees of Freedom	5
Weight(robotic arm only)	11.2 kg
 <b>Robot Joints</b>	 <b>Robot Zero Attitude</b>
 <b>Joint Rotating Direction</b>	

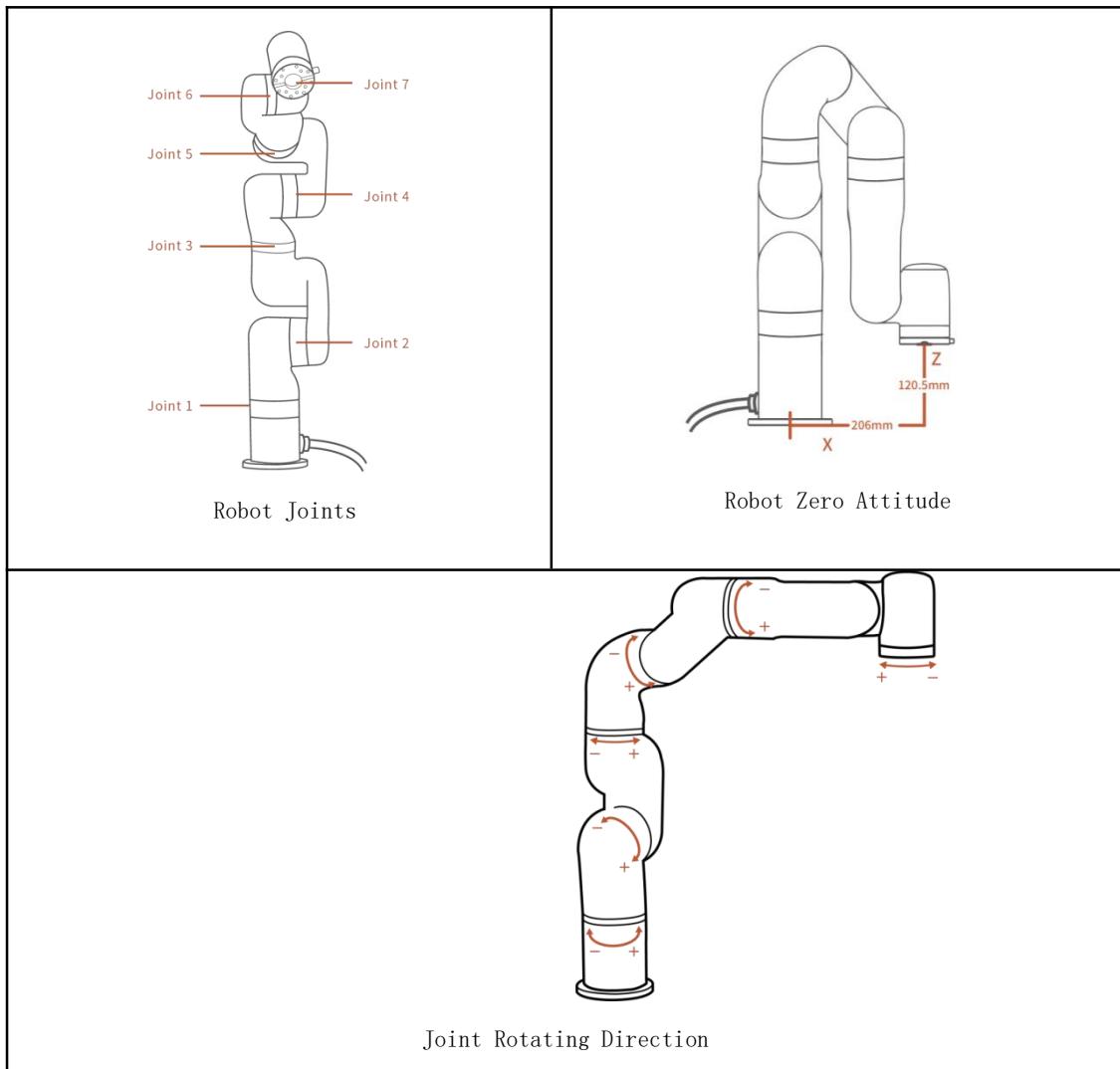
### 1.3 xArm 6 Specifications

Joint Range	1, 4, 6	$\pm 360^\circ$
	2	$-118^\circ \sim 120^\circ$
	3	$-225^\circ \sim 11^\circ$
	5	$-97^\circ \sim 180^\circ$
	Payload	5 kg
Degrees of Freedom		6
Repeatability		$\pm 0.1$ mm
Weight(robotic arm only)		12.2 kg



## 1.4 xArm 7 Specifications

Joint Range	1, 3, 5, 7	$\pm 360^\circ$
	2	$-118^\circ \sim 120^\circ$
	4	$-11^\circ \sim 225^\circ$
	6	$-97^\circ \sim 180^\circ$
	Payload	3.5 kg
Degrees of Freedom		7
Weight (robotic arm only)		13.7 kg



## Appendix3-FAQ

1. [Guide for UFactory studio displaying “Sever is not ready”](#)
2. [Guide to use the Vacuum Gripper](#)
3. [Guide to download the log file on the UFactory studio](#)
4. [Solve the problem that all joints of the xArm are at '0' in the gazebo](#)
5. [The Method of the IP Configuration](#)
6. [How to use PLC to control xArm](#)
7. [Guide to control xArm by tablet](#)
8. [Kinematic and Dynamic Parameters of xArm Series](#)
9. [The Proper Way to Power DC Control Box](#)
10. [How to get the joint current/torque data of the xArm robot](#)
11. [Guide to Update the UFactory studio and xArm Firmware](#)
12. [What should I do if I have a problem with xArm?](#)
13. [Guide to install the xArm Camera Module](#)
14. [Guide to use the Robotiq Gripper on xArm](#)
15. [Guide to run xArm at the maximum speed](#)

## Appendix4-The xArm Software/Firmware Update Method.

### Notes

- 1) It is recommended to update the xArm firmware and UFactory studio at the same time.
- 2) Please check the update notice carefully before each update. It is not recommended to update the robot arm that have been deployed to production environment.
- 3) After each update, please download the latest xArm User Manual and xArm Developer Manual from the official website to learn the latest feature. If you use xArm-Python-SDK (xArm-C++ SDK or xArm ROS) for development, please get the latest SDK code from GitHub.

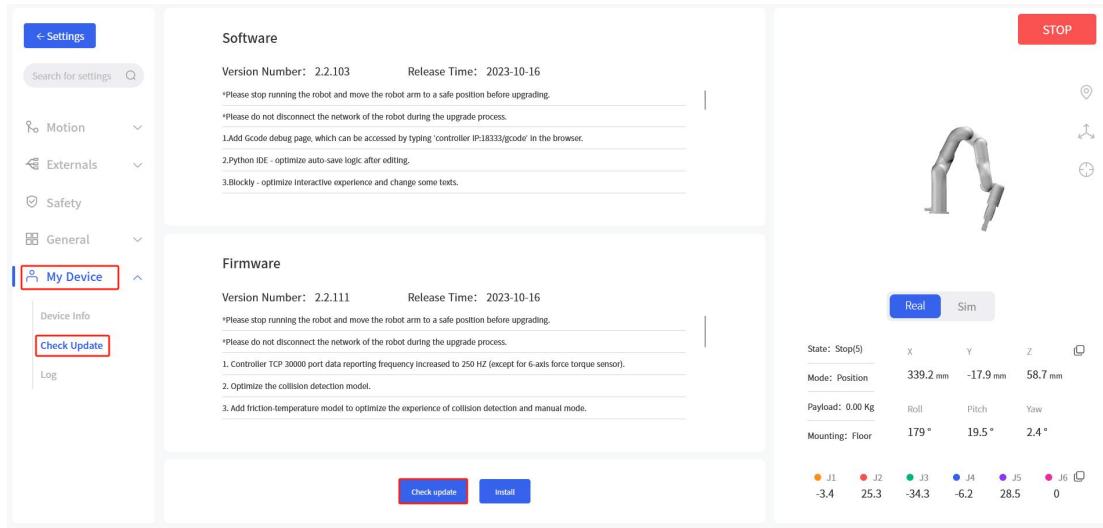
Manual download: <https://www.cn.ufactory.cc/xarm-download>

SDK: <https://github.com/xArm-Developer>

### 1. Online upgrade: when PC has network connection

#### ● Use UFactory Studio to do the online upgrade:

Go to [My Device] → [Check for Update], click "Check for Update", if there is a new version, click "Download", click "Install" to load the downloaded installation package and wait for the system to reboot. The reboot will take about 2-3 minutes.



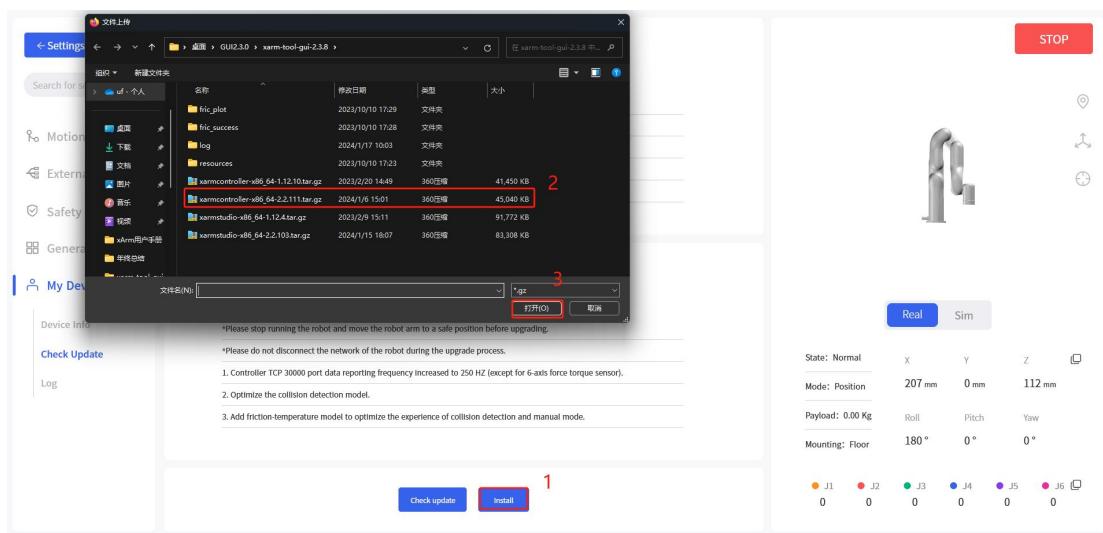
## 2. Offline upgrade: when PC has no network connection

### ● Tool download

xArm-Tool-GUI

### ● Use UFactory Studio to do offline upgrade:

Go to [My Device] – [Check Update], click "Install" to load the offline package downloaded in advance, reboot the system, it will take 2–3 minutes.



- Use `xArm-tool-gui` tool to do the offline upgrade:

#### xArm-Tool-GUI Download

Download the `xarm-tool-gui` tool, unzip and run it. Enter the IP address of the controller and click [Connect] button.

- 1) Click [Firmware] - [Offline Installation], select the offline firmware package you downloaded in advance, click [Install], the interface pop-up prompt "Install firmware successfully".
- 2) Click [xArmStudio] - [Offline Installation], select the offline package downloaded in advance, click [Install], the interface will pop up a prompt "Install Studio successfully".
- 3) Click [Reboot Control Box], wait for 2-3 minutes for the controller to finish rebooting and reconnect.



Note: If the online upgrade fails, you can try the offline upgrade. When the offline upgrade is still unsuccessful, please contact technical support (`support@ufactory.cc`).

## **Appendix5– Maintenance and Inspection**

### **1. Long-term placement**

If the robotic arm is not used for a long time ( $\geq 3$  months), you need to power on the robotic arm for 6 hours every 3 months to charge the built-in battery of the robotic arm.

When powering on the robotic arm, please release the emergency stop button on the control box, and the robotic arm does not need to be enabled.

### **2. Clean**

After the robotic arm is used for a long time, there may be dirt or grease on the carbon fiber shell (in rare cases, a small amount of grease can be seen at the joints, which will not affect the normal use or life of the joints). You can use 95% alcohol or 70% isopropanol to wipe the carbon fiber surface for cleaning.

Note:

When cleaning the carbon fiber surface, be careful not to let the liquid penetrate the joints.

## **Appendix6– Repair**

### **1. Repair work must only be done by UFACTORY.**

After repair work, checks must be done to ensure the required safety level. Checks

must adhere to valid national or regional work safety regulations. The correct functioning of all safety functions shall also be tested.

### **2. After-sales policy:**

For the detailed after-sales policy of the product, see the official website:

<https://www.ufactory.cc/warranty-and-returns/>

3. The general process of after-sales service is:

(1) Contact UFACTORY technical support (support@ufactory.cc) to confirm whether the product needs to repair and which part should be sent back to UFACTORY.

(2) After the bill of lading on UPS, we will send the invoice and label to you by mail. You need to make an appointment with the local UPS and then send the product to us.

(3) UFACTORY will check the product warranty status according to the after-sales policy.

(4) Generally, the process takes around 1-2 weeks except for shipment.

**Note:**

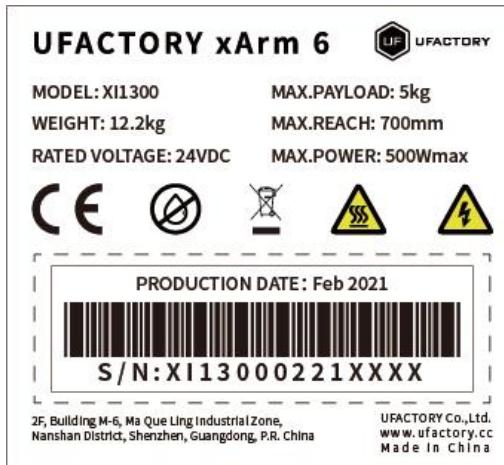
1. Please keep the original packaging materials of the product. When you need to send the product back to get repaired, please pack the product with the original box to protect the product during the transportation.

2. If you need to send the control box to get repaired, please export and save the configuration file of the robotic arm to prevent the original data from being lost or changed during the repair process .

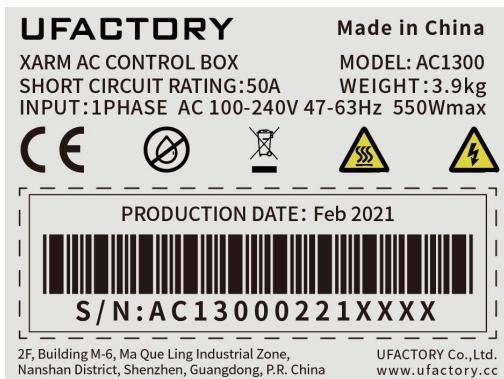
## Appendix7-Product Information

### 1.1 Product Mark

Robotic Arm



Control Box



### 1.2 Applied Standards

The xArm 6 (XI1300, XI1301, XI1302, XI1303, XI1304, XI1305) robot is certified and tested by SGS, and has passed the EU CE certification. The product meets the relevant requirements of the EU CE directive:

- MD 2006/42/EC
- EMC 2004/108/EC
- EN ISO 10218-1:2011
- EN 60204-1:2018

- EN ISO 12100:2010
- EN 61000-6-2:2005
- EN 61000-6-4:2007+A1:2011

### **1.3 EMC**

- IEC 61000-6-2:2005
- IEC 61000-6-4/A1:2010
- EN 61000-6-2:2005 [2004/108/EC]
- EN 61000-6-4/A1:2011 [2004/108/EC]

Electromagnetic compatibility (EMC)

Part 6-2: Generic standards – Immunity for industrial environments

Part 6-4: Generic standards – Emission standard for industrial environments

These standards define requirements for the electrical and electromagnetic disturbances. Conforming to these standards ensures that the xArm robots perform well in industrial environments and that they do not disturb other equipment.

- EN 61000-6-4:2019
- EN 61000-6-2:2019

Electrical equipment for measurement, control and laboratory use – EMC requirements

Part 3-1: Immunity requirements for safety-related systems and for equipment intended to perform safety-related functions (functional safety) – General industrial applications.

This standard defines extended EMC immunity requirements for safety-related functions. Conforming to this standard ensures that the safety functions of xArm robots provide safety even if other equipment exceeds the EMC emission limits defined in the IEC 61000 standards.

### **1.4 Use Environment**

- Low humidity (25%–85% non-condensing)
- Altitude: <2000m

- Ambient temperature: 0° C ~ 50° C
- Avoid direct sunlight (indoor use)
- No corrosive gas or liquid.
- No flammable materials.
- No oil mists.
- No salt sprays.
- No dust or metal powder.
- No mechanical shock, vibration.
- No electromagnetic noise.
- No radioactive materials.

## 1.5 Transport, Storage and Handling

- Move the robot to the zero position by UFactory studio, then put the xArm robot and Control Box in the original packaging.
- Transport the robot in the original packaging.
- Lift both tubes of the robot arm at the same time when moving it from the packaging to the installation place. Hold the robot in place until all mounting bolts are securely tightened at the base of the robot.
- The controller box shall be lifted by the handle.
- Save the packaging material in a dry place, you may need to pack down and move the robot in the future.

## 1.6 Power box placement height

The control box should be placed at a height of 0.6m to 1.5m

## 1.7 Power Connection

The power cut-off method of this product is a plug/socket connection, so when using this product, it is recommended to equip with a suitable switching device with sufficient breaking capacity (such as an air switch; insulation voltage: 400V AC; rated current: 10A)

## 1.8 Special Consumables.

Fuse specifications: 15A 250V 5×20mm Time-Lag glass body cartridge fuse

## 1.9 Stop Categories

A Stop Category 1 and a Stop Category 2 decelerates the robot with drive power on, which enables the robot to stop without deviating from its current path.

Safety Input	Description
Emergency Stop Button of the Control Box	Performs a Stop Category 1
Emergency Input of the Control Box	Performs a Stop Category 1.
Emergency Stop Button of the Three-Position Enabling Device	Performs a Stop Category 1.
Three-Position Enabling Device	Performs a Stop Category 2.
Safeguard Stop by CI of Control Box	Performs a Stop Category 2.

## 1.10 Stopping Time and Stopping Distance

Stop Category 1 stopping distances and times.

The table below includes the stopping distances and times measured when a Stop Category 1 is triggered. These measurements correspond to the following configuration of the robot:

- Extension: 100% (the robot arm is fully extended horizontally).
- Speed: 100% (the general speed of the robot is set to 100% and the movement is performed at a joint speed of 180 ° /s).
- Payload: maximum payload handled by the robot attached to the TCP (5 kg).

The test on the Joint 1 was carried out by performing a horizontal movement, the axis of rotation was perpendicular to the ground.

During the tests for Joint 2 and 3 the robot followed a vertical

trajectory, i.e. the axes of rotation were parallel to the ground, and the stop was performed while the robot was moving downwards.

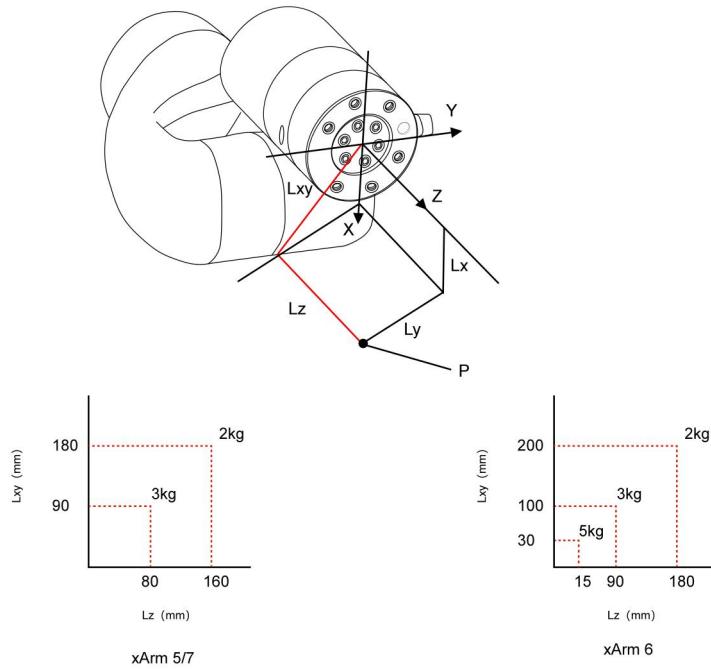
	Stopping Distance (rad)	Stopping time (ms)
Joint 1	0.62	521
Joint 2	1.12	885
Joint 3	0.67	577

## 1.11 Maximum Speed

Mode	Typical Scenarios	Maximum Speed
Teaching mode	Live Control Page of UFactory studio	250 mm/s
Automatic mode	Blockly/ IDE of UFactory studio	1000 mm/s

## 1.12 Maximum Payload

The maximum allowed payload of the robot arm depends on the center of gravity offset. The center of gravity offset is defined as the distance between the center of the tool output flange and the center of gravity.



## 1.13 Specifications

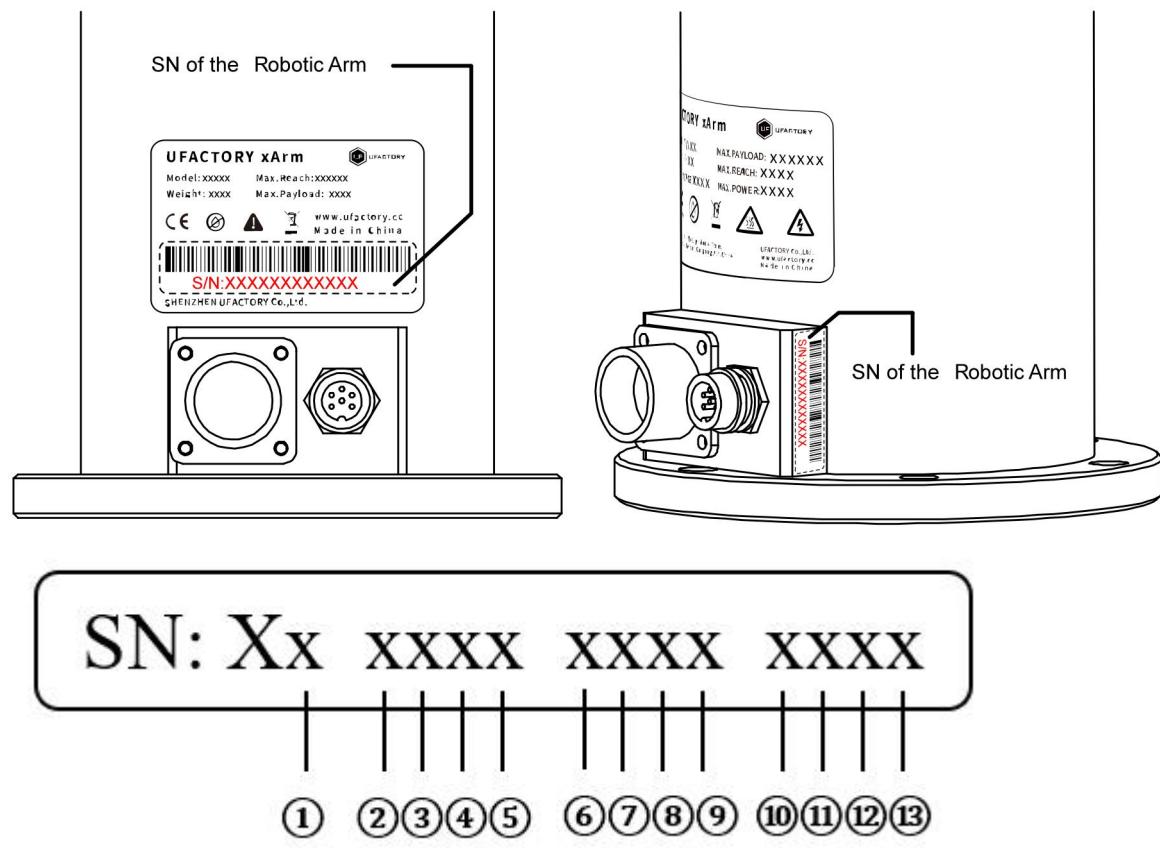
Robotic Arm Model		XI1300
Joint Range	1, 4, 6	±360°
	2	-118° ~120°
	3	-225° ~11°
	5	-97° ~180°
Payload		5kg
Maximum Reach		700mm
Degrees of Freedom		6
Repeatability		±0.1mm
Maximum Joint Speed		180° /s
Weight (robotic arm only)		12.2kg
Footprint		Ø 126 mm
Maximum Power		500W
Rated Voltage		24VDC

Control Box Model		AC1300
Size		285*135*101mm
I/O Ports		8*CI 8*DI 8*CO 8*DO 2*AI 2*AO 2*RS-485
Communication Protocol		Private TCP
Weight		3.9kg
Operating Temperature		0~50°C
Humidity		25%~85% (non-condensing)
Short Circuit Rating		50A
Input		1PHASE AC 100~240V 47~63HZ 550Wmax

## Appendix8-DH Parameters of xArm Series

First let's learn to distinguish the model of the UFACTORY xArm by the SN of the UFACTORY xArm, so that the user can obtain the kinematic and dynamic parameters of the corresponding model of the UFACTORY xArm.

- Method of distinguishing UFACTORY xArm model through SN number



①:

There are three representations for this position: S/I/F

S	I	F
xArm 7	xArm 6	xArm 5

⑪:

There are three representations for this position: L/B/0

When ⑪ position is 0, Please send the SN of the UFACTORYxArm to technical support (email: support@ufactory.cc).

When ⑪ position is L, it means that the robotic arm is model 2.

When ⑪ position is B, it means that the robotic arm is model 1/model 3

3-1: When the position ⑥⑦ is 04-12 and the position ⑧⑨ is 20, the model of the UFACTORY xArm is model 3 (such as 0420).

3-2: When the position ⑥⑦⑧⑨ is other numbers, the model of the UFACTORY Arm is model 1.

### ● Kinematic and Dynamic Parameters of UFACTORY xArm

Annotation of coordinate representation:

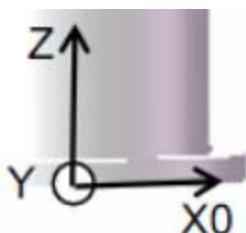
Black dot (●):

This means the direction is pointing out of the viewing surface.



White dot (○):

This means the direction is pointing into the viewing surface.



\* Please note the provided Dynamic Parameters are just for reference.

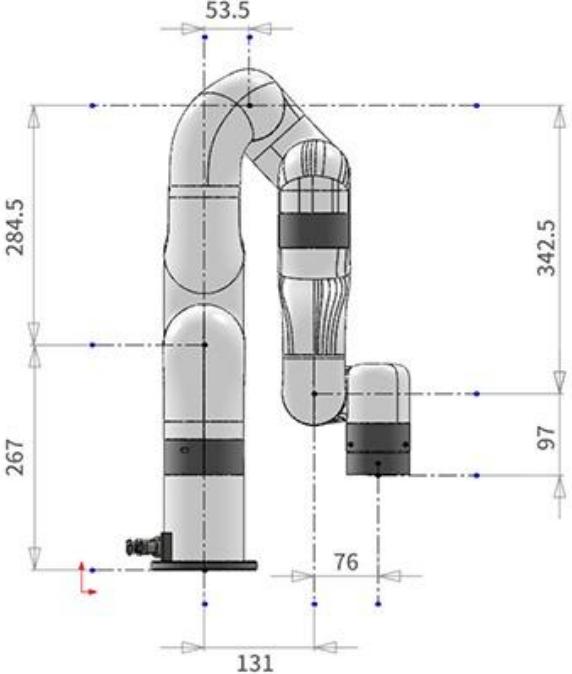
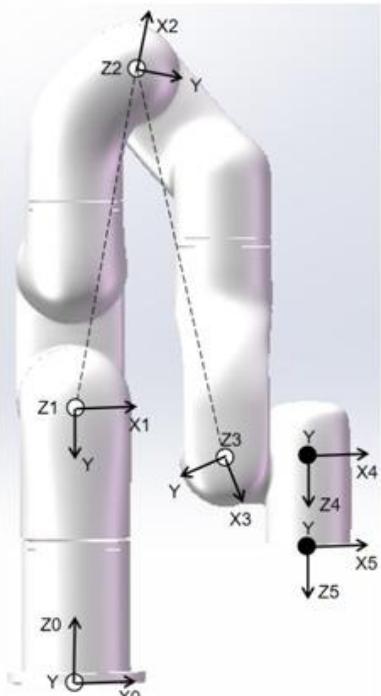
## 1. UFACTORY xArm 5

### 1) UFACTORY xArm5 Modified D-H Parameters

Size Parameters: (unit: mm)	Joint Coordinate Definition

Kiematics	theta (rad)	d (mm)	alpha (rad)	a (mm)	offset (rad)
Joint1	0	267	0	0	0
Joint2	0	0	-pi/2	0	T2_offset
Joint3	0	0	0	a2	T3_offset
Joint4	0	0	0	a3	T4_offset
Joint5	0	97	-pi/2	76	0

### 2) UFACTORY xArm5 Standard D-H Parameters

Size Parameters: (unit: mm)			Joint Coordinate Definition		
					
Kiematics	theta (rad)	d (mm)	alpha (rad)	a (mm)	offset (rad)
Joint1	0	267	-pi/2	0	0
Joint2	0	0	0	a2	T2_offset
Joint3	0	0	0	a3	T3_offset
Joint4	0	0	-pi/2	76	T4_offset
Joint5	0	97	0	0	0

Note:

'Tx\_offset' is the offset joint angle from the mathematical zero position to the mechanical zero position shown in the picture.

$$T2\_offset = -\text{atan}(284.5/53.5) = -1.3849179 \quad (-79.34995^\circ);$$

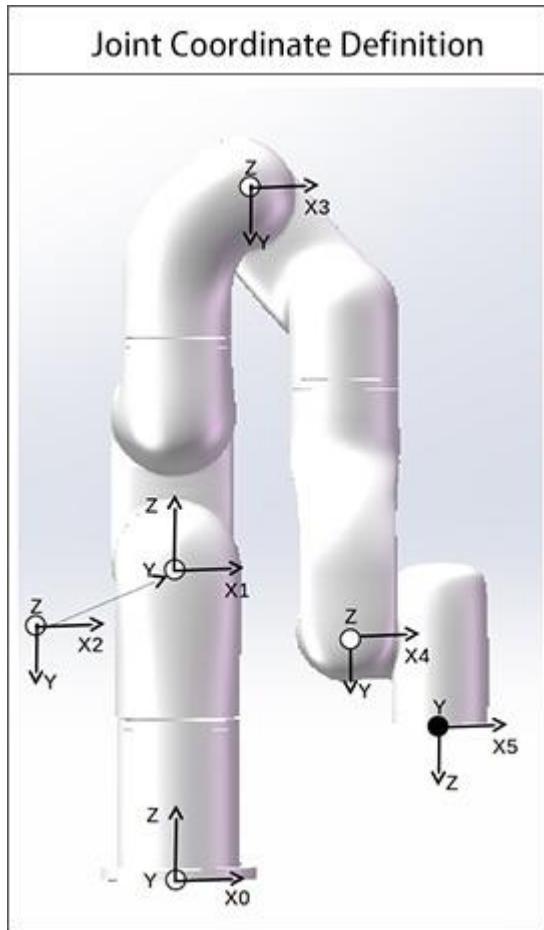
$$T3\_offset = \text{atan}(284.5/53.5) + \text{atan}(0.3425/0.0775) = 2.7331843 \quad (156.599924^\circ);$$

$$a2 = \sqrt{284.5^2 + 53.5^2} = 289.48866;$$

$$T4\_offset = -\text{atan}(342.5/77.5) = -1.3482664 \quad (-77.249974^\circ);$$

$$a_3 = \sqrt{(77.5^2 + 342.5^2)} = 351.158796;$$

### 3) UFACTORY xArm5 xArm Mass Parameters



UFACTORY xArm 5-Model 1

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2.177	[0.15, 27.24, -13.57]
Link2	2.011	[36.7, -220.9, 33.56]
Link3	2.01	[68.34, 223.66, 1.1]
Link4	1.206	[63.87, 29.3, 3.5]
Link5	0.17	[0, -6.77, -10.98]

## UFACTORY xArm 5-Model 2

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2. 46	[0. 13, 30. 1, -12. 0]
Link2	2. 21	[38. 2, -226. 6, 34. 7]
Link3	2. 16	[69. 0, 231. 8, 1. 0]
Link4	1. 354	[65. 2, 31. 8, 3. 11]
Link5	0. 17	[0, -6. 77, -10. 98]

## UFACTORY xArm 5-Model 3

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2. 382	[0. 13, 29. 4, -12. 4]
Link2	2. 164	[37. 88, -225. 4, 34. 47]
Link3	2. 121	[68. 83, 229. 85, 1. 02]
Link4	1. 317	[64. 9, 31. 2, 3. 2]
Link5	0. 17	[0, -6. 77, -10. 98]

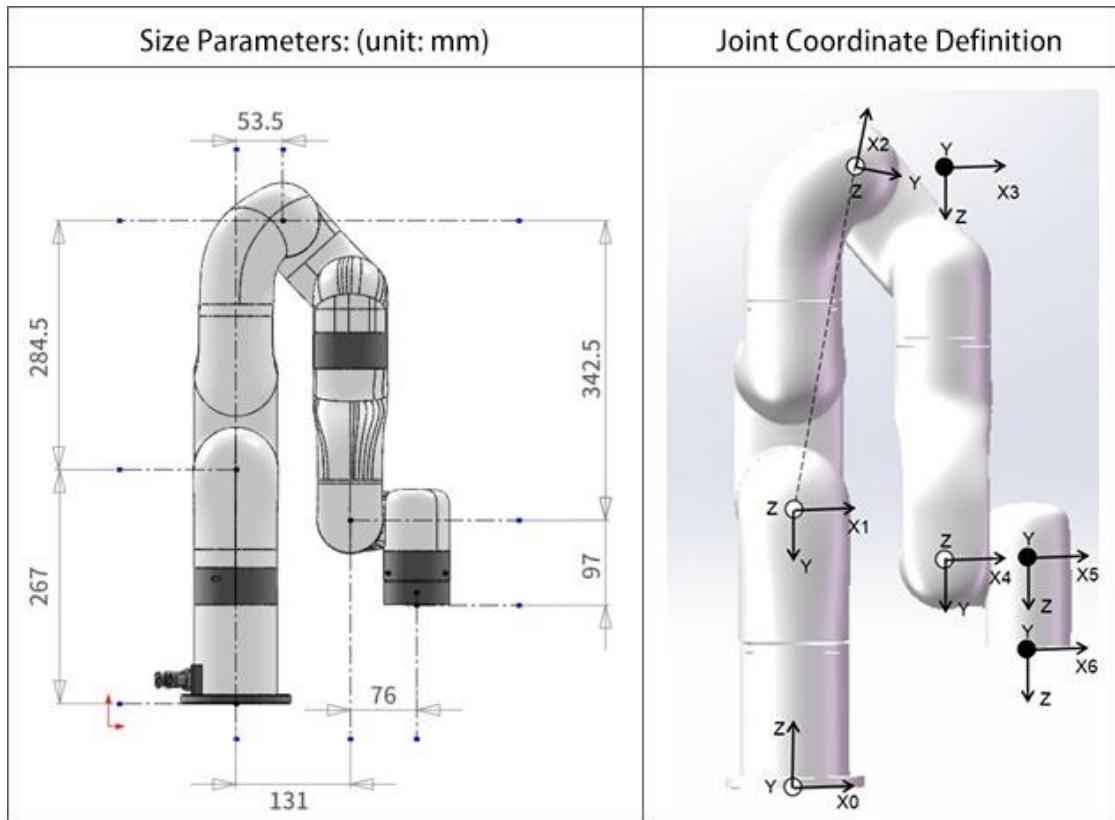
## 2. UFACTORY xArm 6

### 1) UFACTORY xArm 6 Modified D-H Parameters

Size Parameters: (unit: mm)		Joint Coordinate Definition	

Kinematics	theta (rad)	d (mm)	alpha (rad)	a (mm)	offset (rad)
Joint1	0	267	0	0	0
Joint2	0	0	-pi/2	0	T2_offset
Joint3	0	0	0	a2	T3_offset
Joint4	0	342.5	-pi/2	77.5	0
Joint5	0	0	pi/2	0	0
Joint6	0	97	-pi/2	76	0

## 2) UFACTORY xArm 6 Standard D-H Parameters



Kinematics	theta (rad)	d (mm)	alpha (rad)	a (mm)	offset (rad)
Joint1	0	267	-pi/2	0	0
Joint2	0	0	0	a2	T2_offset
Joint3	0	0	-pi/2	77.5	T3_offset
Joint4	0	342.5	pi/2	0	0
Joint5	0	0	-pi/2	76	0
Joint6	0	97	0	0	0

Note:

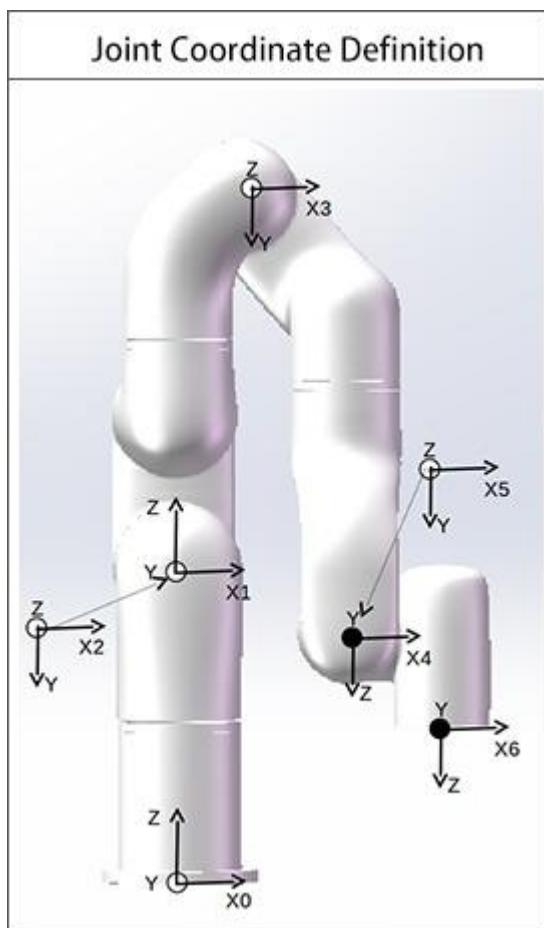
'Tx\_offset' is the offset joint angle from the mathematical zero position to the mechanical zero position shown in the picture.

$$a2 = \sqrt{(284.5^2 + 53.5^2)} = 289.48866;$$

$$T2\_offset = -\text{atan}(284.5/53.5) = -1.3849179 \quad (-79.34995^\circ);$$

$$T3\_offset = -T2\_offset = 1.3849179 \quad (79.34995^\circ);$$

### 3) UFACTORY xArm 6 Mass Parameters



### UFACTORY xArm 6-Model 1

Dynamics	Mass (Kg)	Center of Mass (mm)

Link1	2. 177	[0. 15, 27. 24, -13. 57]
Link2	2. 011	[36. 7, -220. 9, 33. 56]
Link3	1. 725	[69. 77, 113. 5, 11. 6]
Link4	1. 211	[-0. 2, 20. 0, -26. 0]
Link5	1. 206	[63. 87, 29. 3, 3. 5]
Link6	0. 17	[0, -6. 77, -10. 98]

### UFACTORY xArm 6-Model 2

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2. 46	[0. 13, 30. 1, -12. 0]
Link2	2. 21	[38. 2, -226. 6, 34. 7]
Link3	1. 925	[70. 6, 117. 2, 10. 4]
Link4	1. 36	[0. 18, 17. 7, -23. 0]
Link5	1. 354	[65. 2, 31. 8, 3. 11]
Link6	0. 17	[0, -6. 77, -10. 98]

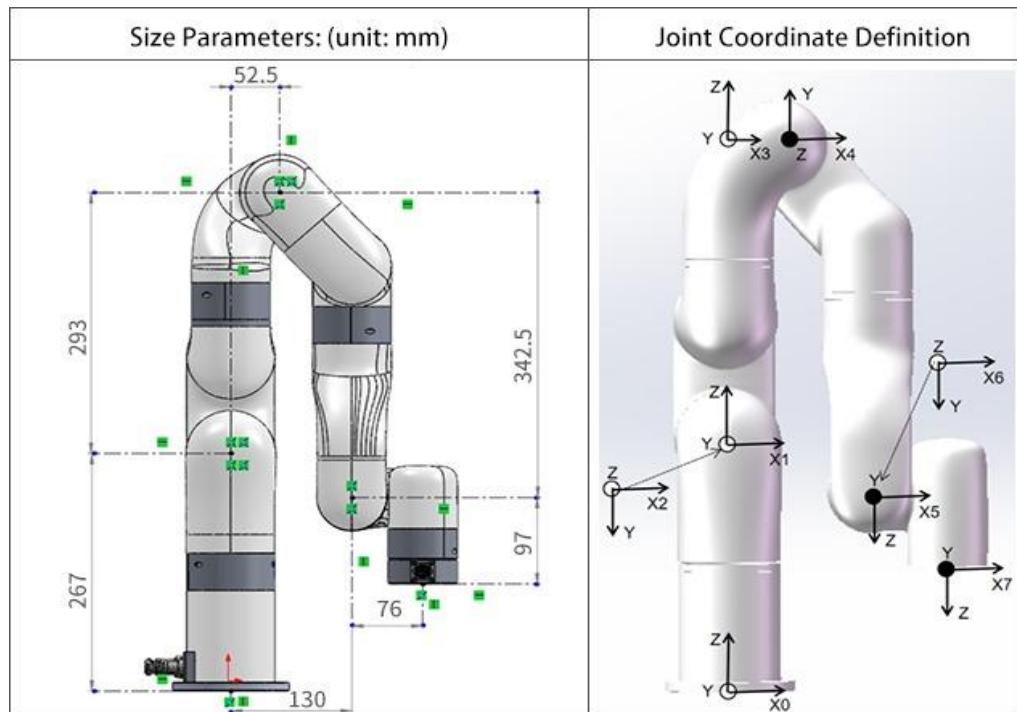
### UFACTORY xArm 6-Model 3

Dynamics	Mass (Kg)	Center of Mass (mm)

Link1	2. 382	[0. 13, 29. 4, -12. 4]
Link2	2. 267	[38. 8, -227. 8, 34. 96]
Link3	1. 875	[70. 4, 116. 3, 10. 7]
Link4	1. 319	[-0. 2, 18. 0, -22. 9]
Link5	1. 34	[65. 1, 30. 96, 3. 15]
Link6	0. 17	[0, -6. 77, -10. 98]

### 3. UFACTORY xArm 7

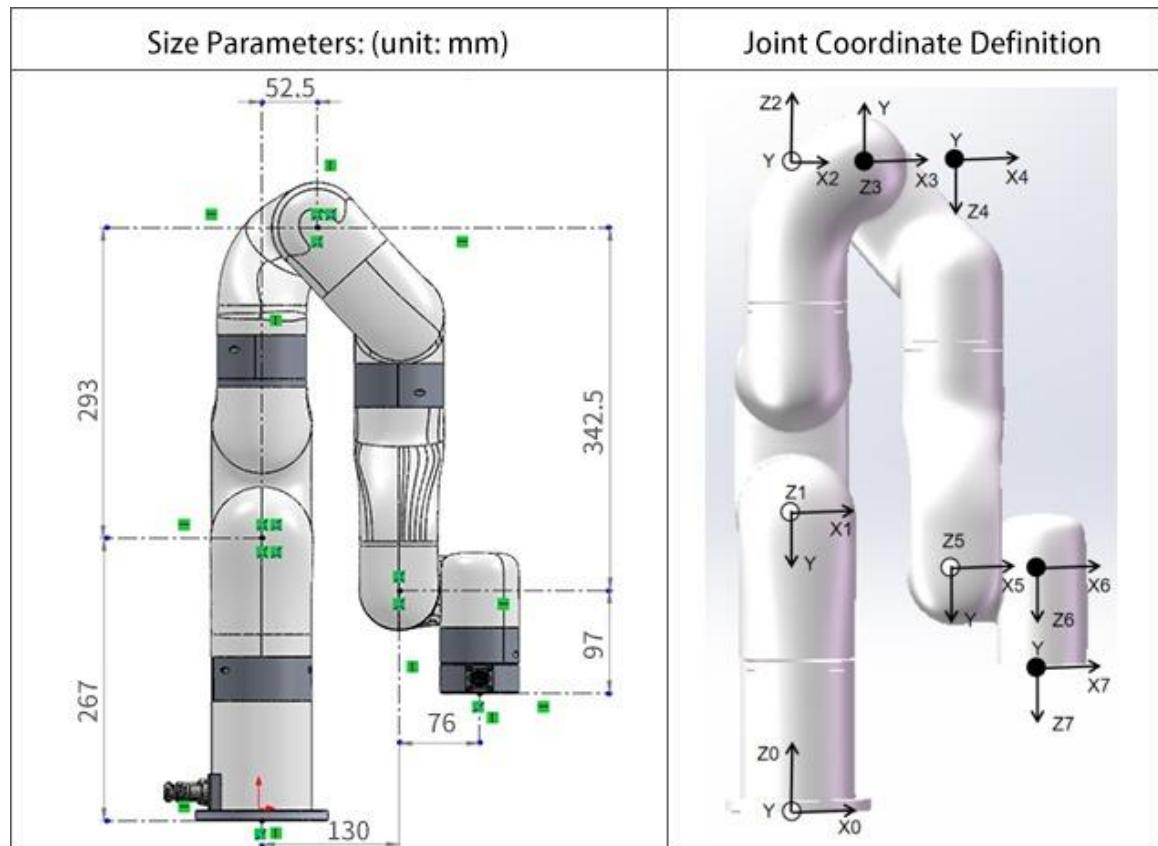
#### 1) UFACTORY xArm7 Modified D-H Parameters



Kiematics	theta (rad)	d (mm)	alpha (rad)	a (mm)	offset (rad)
Joint1	0	267	0	0	0
Joint2	0	0	-pi/2	0	0
Joint3	0	293	pi/2	0	0

Joint4	0	0	$\pi/2$	52.5	0
Joint5	0	342.5	$\pi/2$	77.5	0
Joint6	0	0	$\pi/2$	0	0
Joint7	0	97	$-\pi/2$	76	0

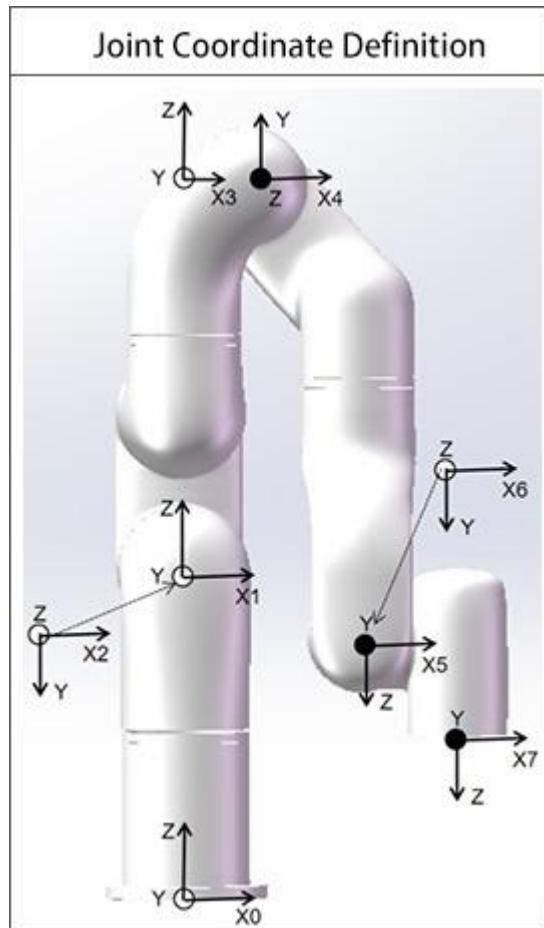
## 2) UFACTORY xArm7 Standard D-H Parameters



Kinematics	theta (rad)	d (mm)	alpha (rad)	a (mm)	offset (rad)
Joint1	0	267	$-\pi/2$	0	0
Joint2	0	0	$\pi/2$	0	0
Joint3	0	293	$\pi/2$	52.5	0
Joint4	0	0	$\pi/2$	77.5	0

Joint5	0	342. 5	$\pi/2$	0	0
Joint6	0	0	$-\pi/2$	76	0
Joint7	0	97	0	0	0

### 3) UFACTORY xArm 7 Mass Parameters



UFACTORY xArm7-Model 1

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2. 177	[0. 15. 27. 24. -13. 57]
Link2	1. 716	[0. 22, -124. 7, 18. 9]
Link3	1. 485	[46. 0, -22. 3, -8. 47]
Link4	1. 574	[69. 75, -112. 5, 13. 2]
Link5	1. 209	[-0. 35, 17. 6, -28. 4]

Link6	1. 214	[63. 65, 30. 84, 21. 7]
Link7	0. 17	[0, -6. 77, -10. 98]

### UFACTORY xArm7-Model 2

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2. 46	[0. 13, 30. 1, -12. 0]
Link2	1. 916	[0. 2, -129. 6, 16. 9]
Link3	1. 69	[46. 76, -25. 3, -7. 46]
Link4	1. 774	[70. 66, -116. 6, 11. 7]
Link5	1. 357	[-0. 3, 15. 6, -25. 3]
Link6	1. 362	[65. 0, 33. 4, 21. 3]
Link7	0. 17	[0, -6. 77, -10. 98]

### UFACTORY xArm7-Model 3

Dynamics	Mass (Kg)	Center of Mass (mm)
Link1	2. 382	[0. 13, 29. 4, -12. 4]
Link2	1. 869	[0. 2, -128. 56, 17. 35]
Link3	1. 638	[46. 6, -24. 63, -7. 68]
Link4	1. 727	[70. 5, -115. 75, 12. 0]
Link5	1. 32	[-0. 32, 16. 04, -26. 0]
Link6	1. 325	[64. 7, 32. 8, 21. 4]
Link7	0. 17	[0, -6. 77, -10. 98]

**Manufacturer:** UFactory Technology Co., Ltd

**Address:** 2F, Building M-6, Ma Que Ling Industrial Zone, Nanshan District, Shenzhen, Guangdong, China

**Website:** [www.ufactory.cc](http://www.ufactory.cc)