

1. Preface

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.

UFACTORY Studio can be accessed through a browser directly without installing any software. Now it's compatible with browsers: Google Chrome/ Firefox/ Safari/ Microsoft Edge(Chromium kernel).

Apply to model: **xArm5, xArm6, xArm7, UFACTORY 850, Lite6.**

2. Glossary

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: 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.

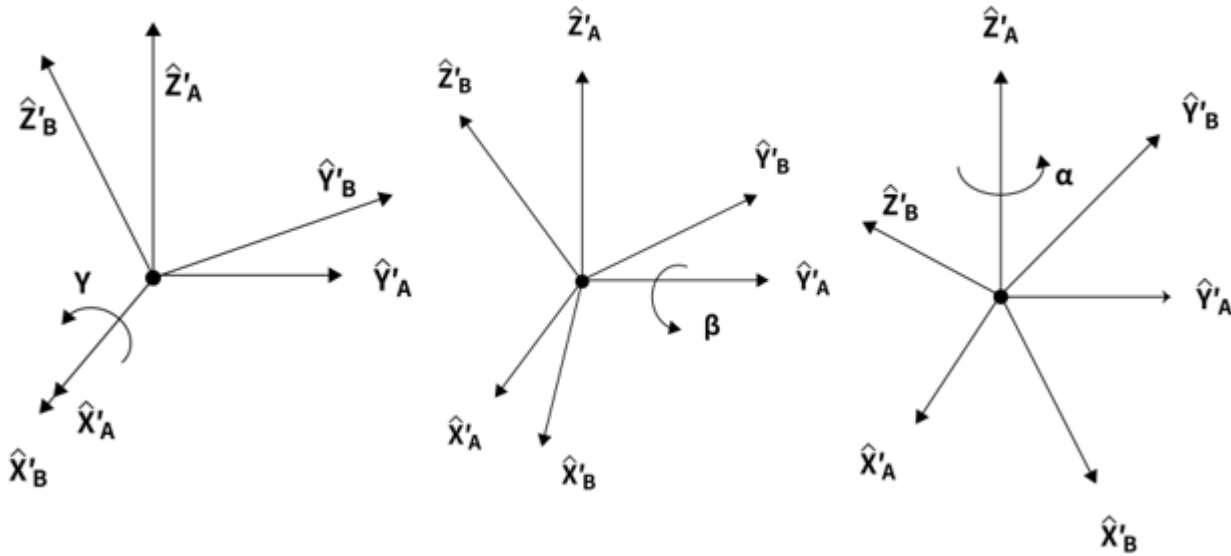
TCP Offset: Set the relative offset between the default tool coordinate system at flange center and the actual tool coordinate system, with distance unit of mm.

Roll/Pitch/Yaw(RPY): Roll/Pitch/Yaw sequentially rotates around the X / Y / Z of the selected coordinate system (base coordinate system).

The following describes the roll/pitch/yaw orientation representation of {B} relative to {A}: For example, the coordinate system {B} and a known reference coordinate system {A} are first superposed. First rotate {B} around **X-axis** by γ , then around **Y-axis** by β , and finally around **Z-axis** by α .

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.

The above description is shown in the following figure:



The equivalent rotation matrix is:

$${}^B_A\mathbf{R}_{XYZ}(\gamma, \beta, \alpha) = \mathbf{R}_Z(\alpha)\mathbf{R}_Y(\beta)\mathbf{R}_X(\gamma)$$

Note: γ corresponds to roll; β corresponds to pitch; α corresponds to yaw.

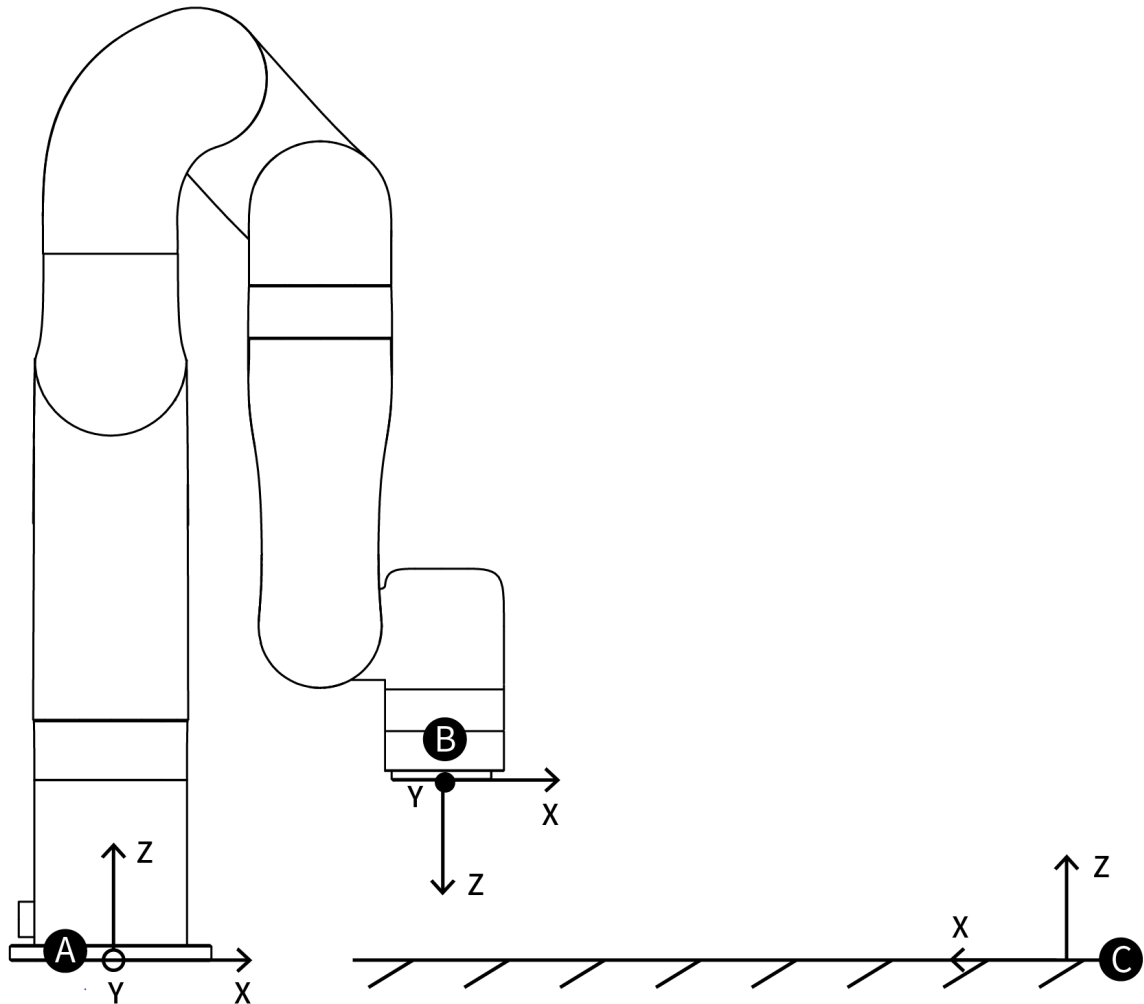
Axis-angle: Rx / Ry / Rz representation also, using 3 values to represent the pose (but not three rotation angles), which is the product of a three-dimensional rotation vector $[x, y, z]$ and a rotation angle $[\phi]$ (scalar). The characteristics of the axis angle: Assume the rotation axis is $[x, y, z]$, and the rotation angle is ϕ . Then the representation of the axial angle: $[R_x, R_y, R_z] = [x * \phi, y * \phi, z * \phi]$

- $[x, y, z]$ is a unit vector, and ϕ is a non-negative value.
- The vector length (modulus) of $[R_x, R_y, R_z]$ can be used to estimate the rotation angle, and the vector direction is the rotation direction.
- If you want to express reverse rotation, invert the rotation axis vector $[x, y, z]$, and the value of ϕ remains unchanged.
- Using ϕ and $[x, y, z]$ can also derive the attitude representation as unit quaternion $q = [\cos(\phi / 2), \sin(\phi / 2) * x, \sin(\phi / 2) * y, \sin(\phi / 2) * z]$. For example: The vector of the rotation axis represented by the base coordinate system is $[1, 0, 0]$, and the rotation angle is 180 degrees (π), then the axis angle representation of this pose is $[\pi, 0, 0]$. The rotation axis is $[0.707, 0.707, 0]$ and the rotation angle is 90 degrees ($\pi / 2$), then the axis angle posture is $[0.707 * (\pi / 2), 0.707 * (\pi / 2), 0]$.

Base Coordinate System: 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(Front and back: X axis, left and right: Y axis, up and down: Z axis).

Tool Coordinate System: 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. 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.

User System: 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.

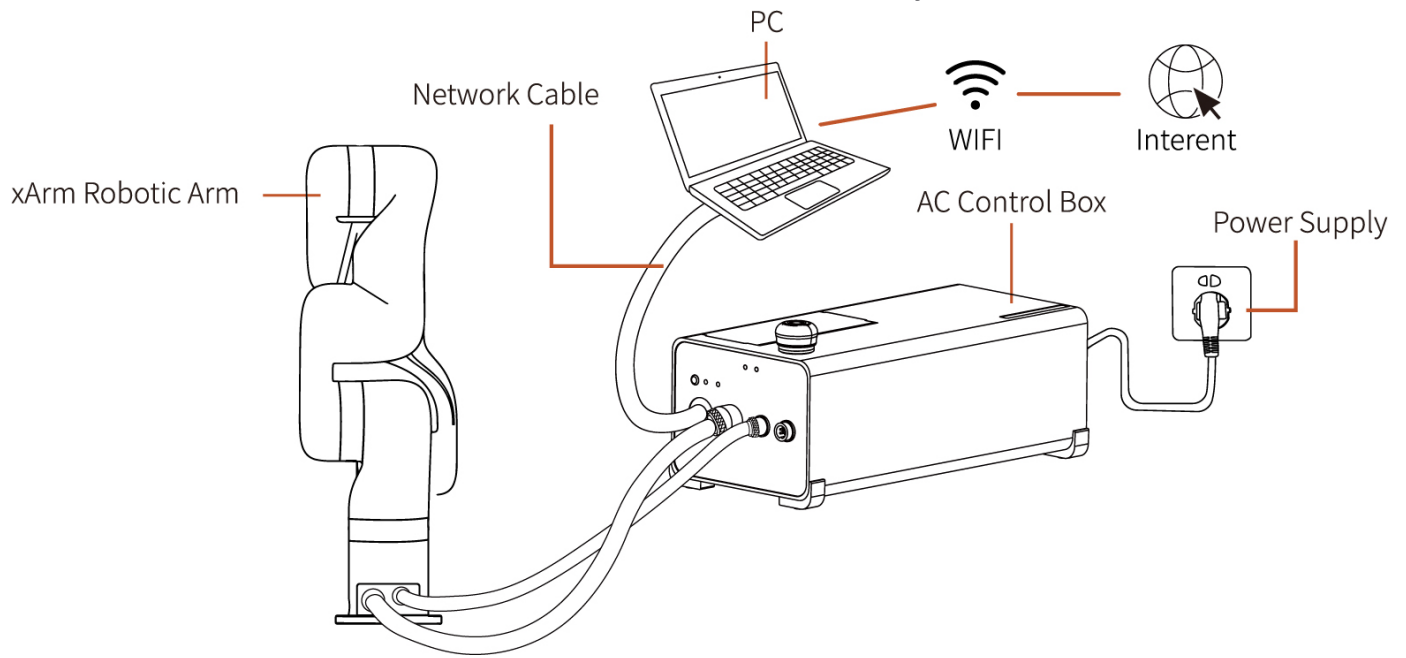
Safety Boundary: When this mode is activated, the boundary range of the cartesian space of the robotic arm can be limited. If the tool center point (TCP) exceeds the set safety boundary, the robotic arm will stop moving.

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.

3. Connection

3.1 Hardware Preparation

Recommended connection method: the control box is directly connected to the PC.



For more connection methods, please check Hardware Connection.

3.2 Software Connection

The default IP of robotic arm is 192.168.1.xxx, please make sure that the IP address of the PC and control box are on the same network segment.

Please refer to [Quick Start Guide](#) to learn how to set the IP of PC.

Enter 'IP+:18333' on the browser to access UFACTORY Studio.

For example:

The IP of control box is 192.168.1.201

Access link: 192.168.1.201:18333

Live Control

Blockly

GCode

⚙️

No End Effector

Select an end effector

Recording

You don't have a project yet

Product Information

Model xArm 7

Robot IP 192.168.1.206

Firmware Version 2.5.1

Software Version 2.5.106

Initial Position

Manual Mode

Speed 98%

J1

J2

J3

J4

J5

J6

J7

Base

Z- Z+

RZ- RZ+

X+ X- Y+ Y- XYZ

RY+ RY- RX- RX+ RXYZ

STOP



Real Sim

State: Stop(5)

Mode: Position

Payload: 0.00 Kg

Mounting: Floor

X 204.8 mm

Y 4 mm

Z 119.3 mm

Roll 180 °

Pitch 0 °

Yaw 0 °

J1 0

J2 0

J3 0

J4 0

J5 0

J6 0

J7 0

4. Live Control

The screenshot displays the UFACTORY Studio Live Control interface, which is divided into several functional panels:

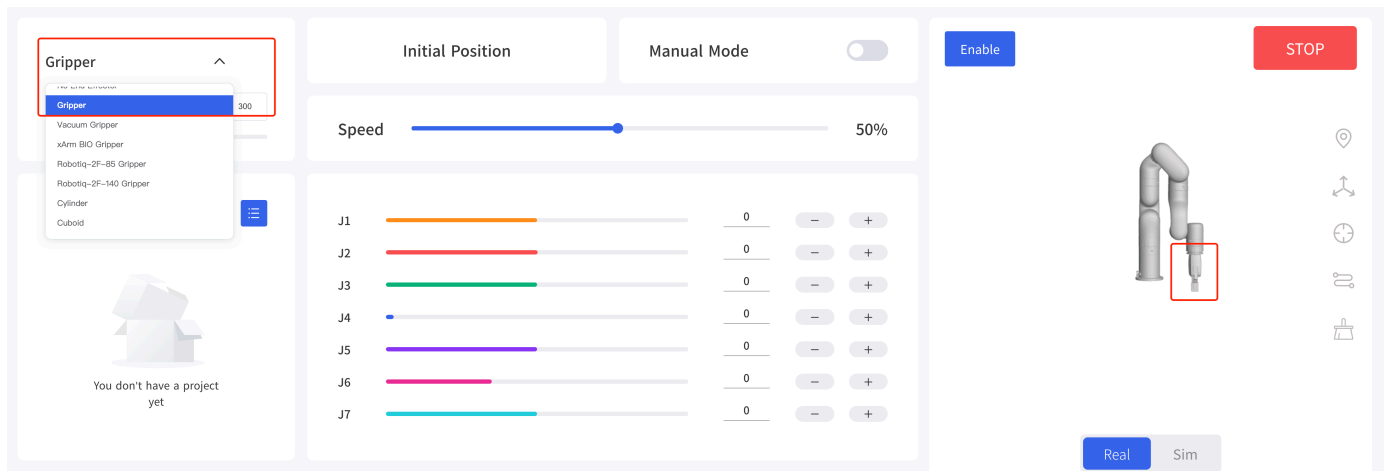
- Left Sidebar:** Contains navigation icons for 'Live Control' (selected), 'Blockly', 'GCode', and a settings gear icon.
- Top Left Panel (3.1):** Titled 'No End Effector', it includes a dropdown menu to 'Select an end effector'.
- Top Middle Panel (3.2):** Titled 'Recording', it features a play button, a pause button, a speed multiplier (x1), and a times counter (1).
- Bottom Left Panel (3.3):** Titled 'Product Information', it displays details for the 'xArm 7' robot, including its IP address (192.168.1.206), firmware version (2.5.101), and software version (2.5.109).
- Central Panel (3.4):** This is the main control area. It includes a 'Manual Mode' toggle, a 'Speed' slider set to 69%, and a list of joint positions (J1-J7) with their respective values (all 0) and directional controls. Below this is a 'Base' section with 'XYZ' and 'RPY' coordinate controls.
- Top Right Panel (3.5):** Features a red 'STOP' button.
- Right Panel (3.6):** Displays a 3D model of the xArm 7 robot. It includes a 'Real' vs 'Sim' toggle and a status bar at the bottom showing joint positions (J1-J7) and their values (all 0).

4.1 End Effector

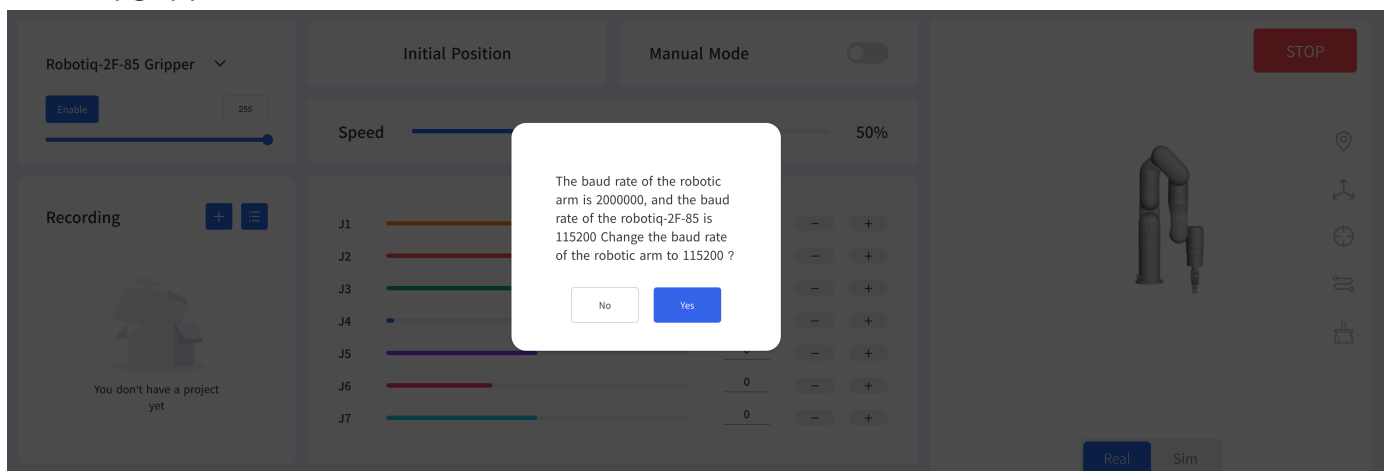
The end effectors currently supported are:

- xArm Gripper
- xArm Vacuum Gripper
- xArm BIO Gripper
- Robotiq-2F-85 Gripper
- Robotiq-2F-140 Gripper
- Gripper Lite
- Vacuum Gripper Lite.

No End Effector by default. Take xArm Gripper as an example:

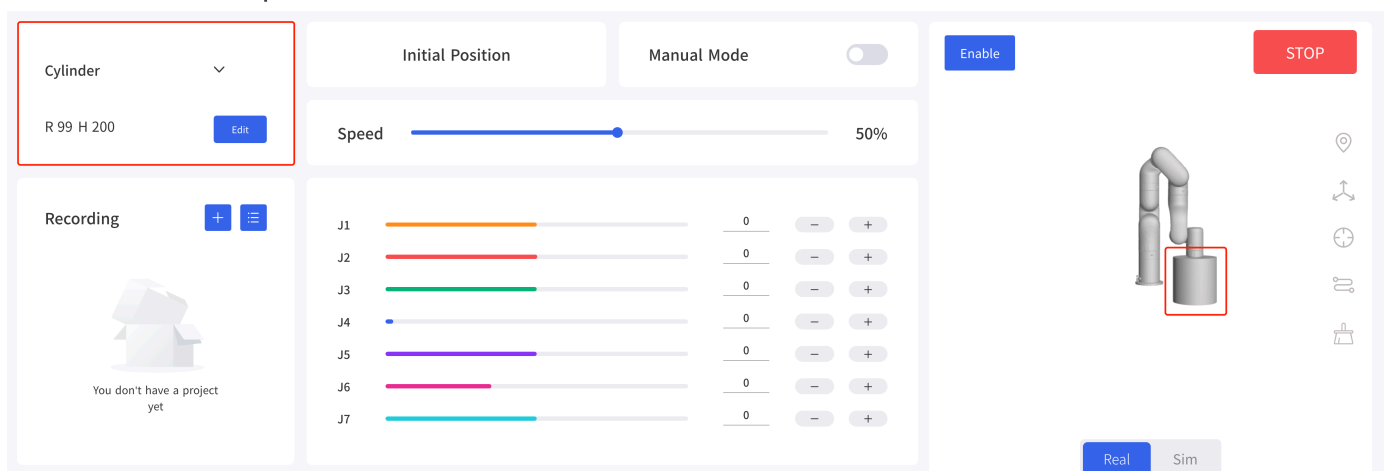


It will set the robotic arm as default baud rate when switching the end effector. For example: the default baud rate of xArm Gripper is 2000000, the default baud rate of Robotiq gripper is 115200.



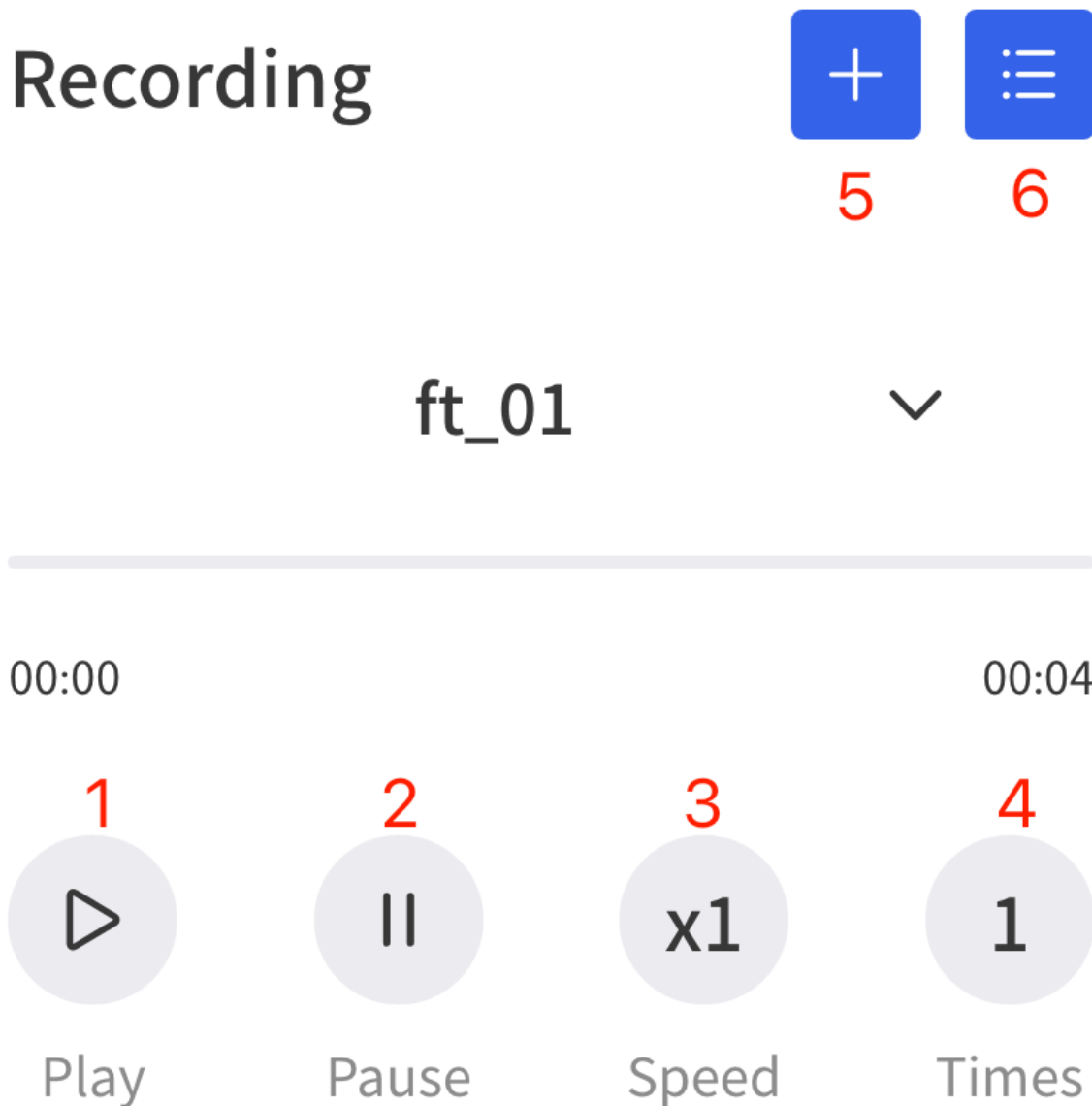
When installing other end effectors (not officially provided) at the end of the robotic arm, please choose 'Other'.

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.



4.2 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 (x1, x2, x4). A recorded trajectory can be imported into Blockly projects.

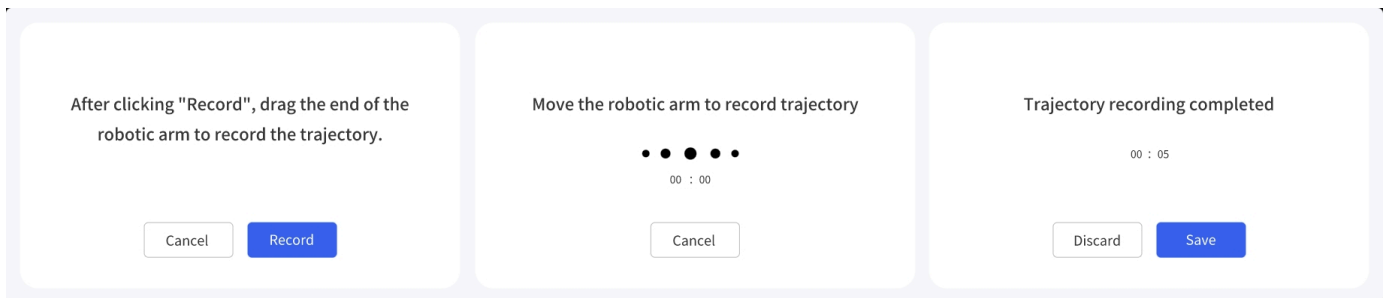


1. Play: play the trajectory.
2. Pause: pause the trajectory.

3. Speed: set playback speed.
4. Times: set playback times.
5. Create new file:
6. Import/Delete File: file operation.

Create New File:

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.



Import/Delete file:

My Projects

[⬆ Import](#)
[🗑 Select](#)

ID	Title	Duration
1	test	00:02
2	01	00:06
3	002	00:03

[Close](#)

4.3 Product Information

This page display the product information, such as: Model, IP, Firmware Version, Software Version, State, Mode, Payload, Mounting, etc.

State: Stop(5)	X	Y	Z
Mode: Position	204.8 mm	4 mm	119.3 mm
Payload: 0.00 Kg	Roll	Pitch	Yaw
Mounting: Floor	180 °	0 °	0 °

J1

0

J2

0

J3

0

J4

0

J5

0

J6

0

J7

0

- IP:The IP of the controller.
- Mode:Robotic arm mode, default: position mode.
- Payload:Robotic arm payload, default: 0kg.
- Mounting:Robotic arm mounting way, default: Floor.
- TCP Coordinate:Current TCP coordinates.
- Joint Angle:Current joint angle.

4.4 Position & Joint Control

Initial Position

Align

Manual Mode

Speed

100%

Initial Position: Long press for continuous motion, the robotic arm will back to the initial position.

Default initial position:

- xArm/850: [0,0,0,0,0,0,0]

- Lite6: `[0,9.9,31.8,0,21.9,0]`

Align: Align the head. (xArm7 don't provide such a function)

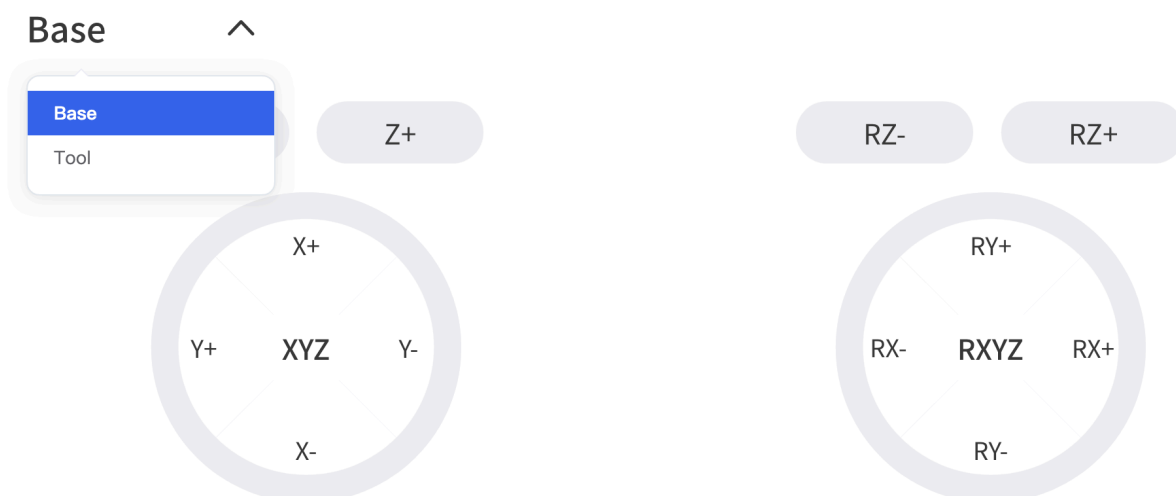
Manual Mode: By turning on the Manual Mode, the joint can be driven freely by hand. Turn on the joint manual mode, you can manually drag the robot links to reach the target position, making it easier to record the robot's motion trajectory, thereby reducing the development workload. When danger occurs, you can also use the manual mode to manually drag the robot away from the danger zone.

- 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.
- 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.
- The SN address of the robotic arm can be checked next to the power signal interface of the base.

Speed: It is used to adjust the motion speed of the live control interface of xArm.

- 50%=115mm/s, default
- 1% = 2.3mm/s
- 100% = 230mm/s

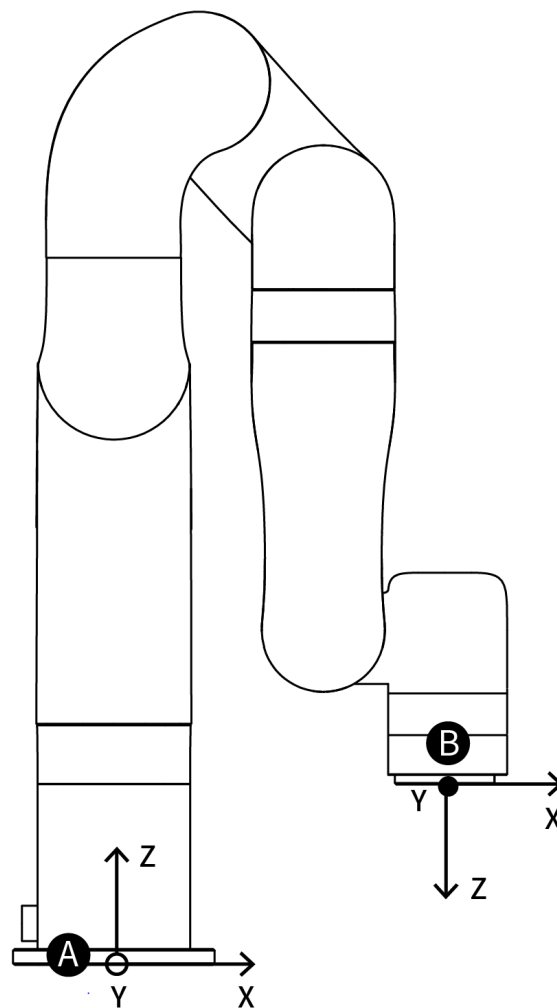
Note: The maximum speed of the live control interface is not the actual maximum motion speed of the robotic arm. If you want the program to run at high speed, you can add a speed command in the Blockly motion program.



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.

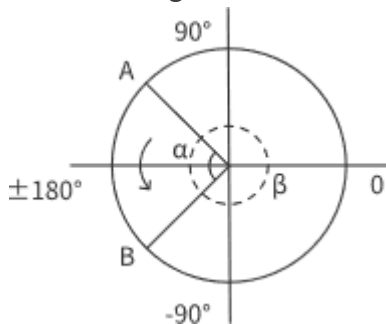
TCP Coordinate:



A: Base Coordinates B: TCP Coordinates

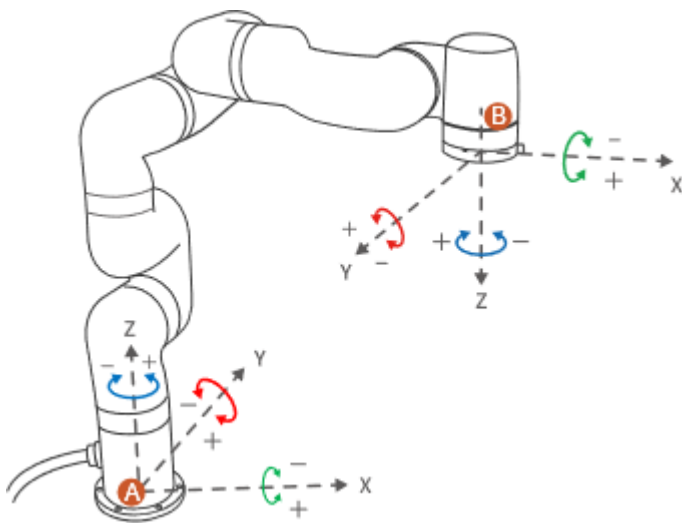
The default TCP coordinate system is defined at the centre point of the end flange of the robotic arm, and it is the result of rotating (180°, 0°, 0°) around the X/Y/Z-axis of the base coordinate system in order. The spatial orientation of the TCP coordinate system changes according to the changes of 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 α angle. If the robotic arm needs to be moved in the direction of the β angle, a new position between the angles of β should be inserted, and the angle that formed by the inserted point and A should be smaller than α .



- The +180° and -180° 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:

$$R_{rpy}(r, p, y) = RZ(y) \cdot RY(p) \cdot RX(r)$$



A: Base Coordinates B: TCP Coordinates(no offset)

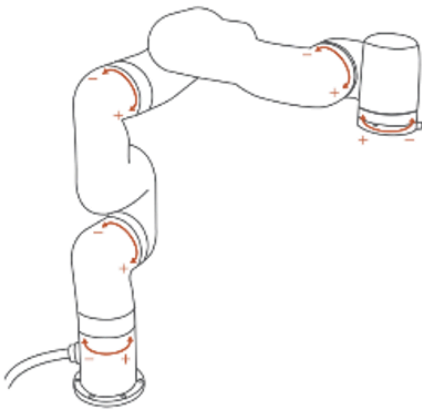
DANGER: You must check the TCP offset before recording the Cartesian position.

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.

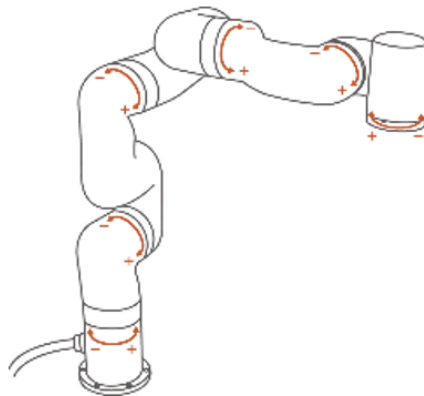
The joint motion reaches the target point with the fastest path, the end trajectory is not a straight line, and the speed unit is °/s. After the target point is set, the corresponding poses are unique for TCP and the joints along the trajectory.

J1		0	<input type="button" value="-"/>	<input data-bbox="1364 582 1422 604" type="button" value="+"/>
J2		0	<input type="button" value="-"/>	<input data-bbox="1364 645 1422 667" type="button" value="+"/>
J3		0	<input type="button" value="-"/>	<input data-bbox="1364 707 1422 730" type="button" value="+"/>
J4		0	<input type="button" value="-"/>	<input data-bbox="1364 770 1422 792" type="button" value="+"/>
J5		0	<input type="button" value="-"/>	<input data-bbox="1364 833 1422 855" type="button" value="+"/>
J6		0	<input type="button" value="-"/>	<input data-bbox="1364 896 1422 918" type="button" value="+"/>
J7		0	<input type="button" value="-"/>	<input data-bbox="1364 958 1422 981" type="button" value="+"/>

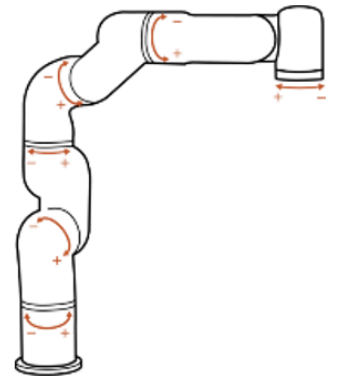
To confirm the direction of joint rotation, please refer the figure below:



xArm 5




xArm 6



xArm 7

4.5 Enable & STOP button

A blue rectangular button with rounded corners containing the word "Enable" in white text.A red rectangular button with rounded corners containing the word "STOP" in white text.

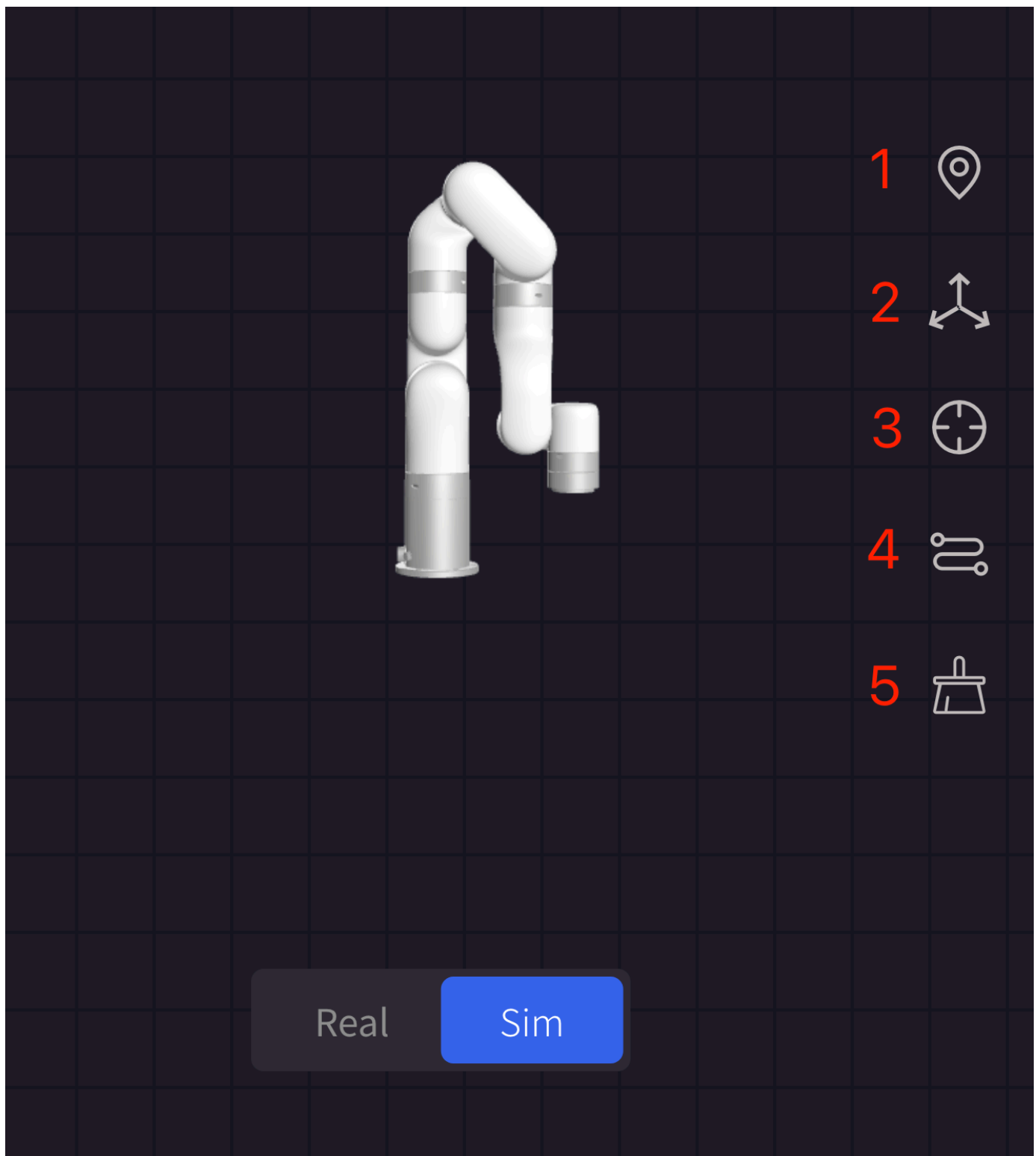
Enable: Enable Robotic Arm. This button will disappear after the robotic arm is enabled.

STOP: The robotic arm will stop immediately and clear all cache commands. It's a **software stop**, the power is still on.

4.6 Real & Simulation robotic arm

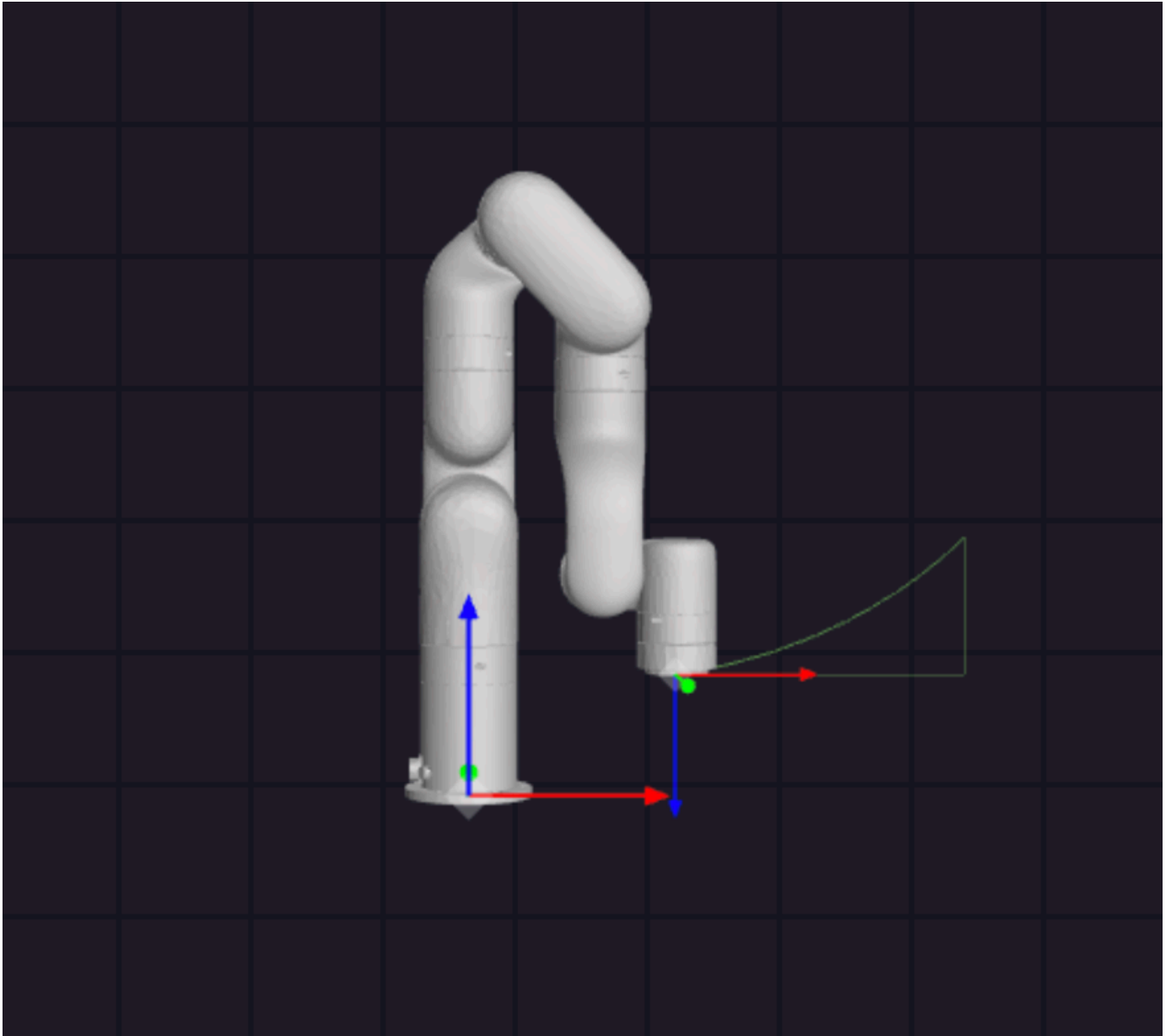
Switch to real or simulation robotic arm, both mode needs to connect to a real robotic arm.

When switch to simulation robotic arm, the robot will not move but the settings will apply to real robotic arm. For example, set tcp payload as 0.6kg in sim mode, the tcp payload will be 0.6kg when switching back to real mode.



1. Reset Perspective: Reset to default Perspective.
2. Base: Current base coordinate system.
3. TCP: Current TCP coordinate system.
4. Plot Path: Plot trajectory path.
5. Clean Path: clean trajectory path.

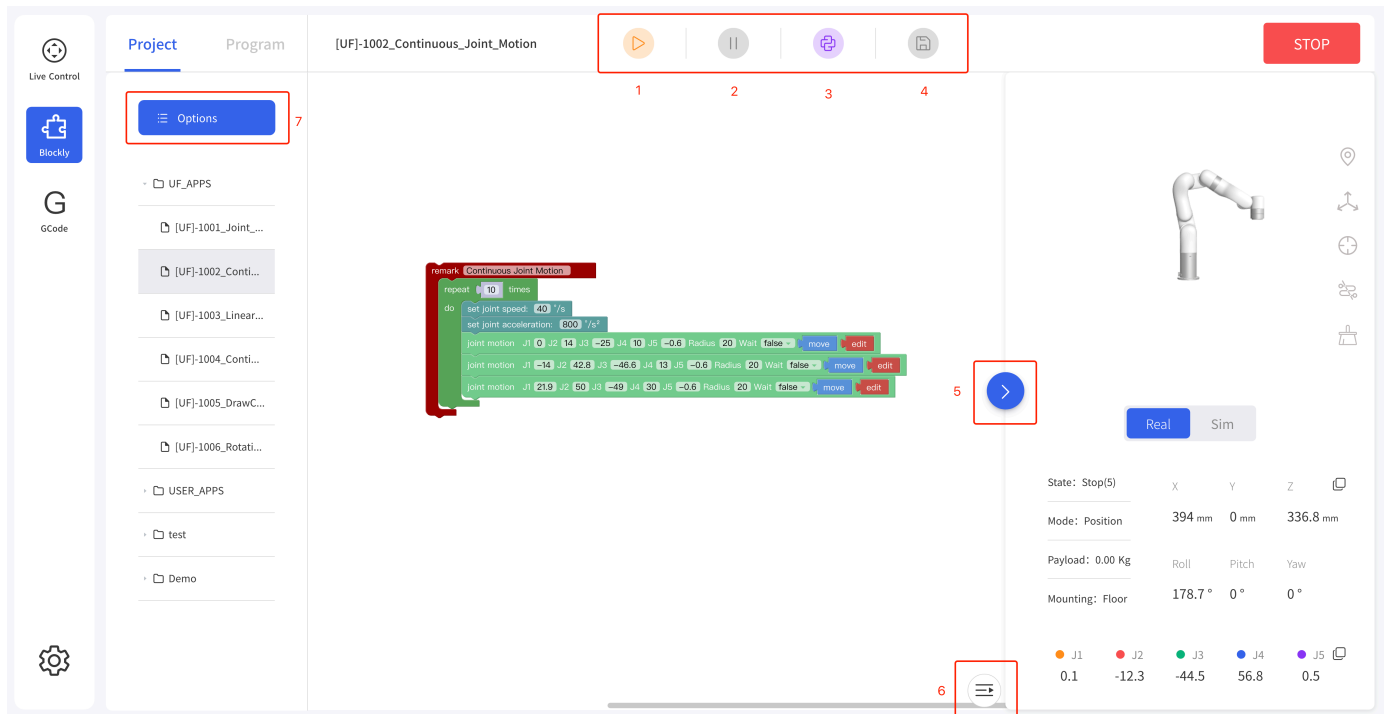
The picture below showing the Base and TCP coordinates, and plot path.



5. Blockly

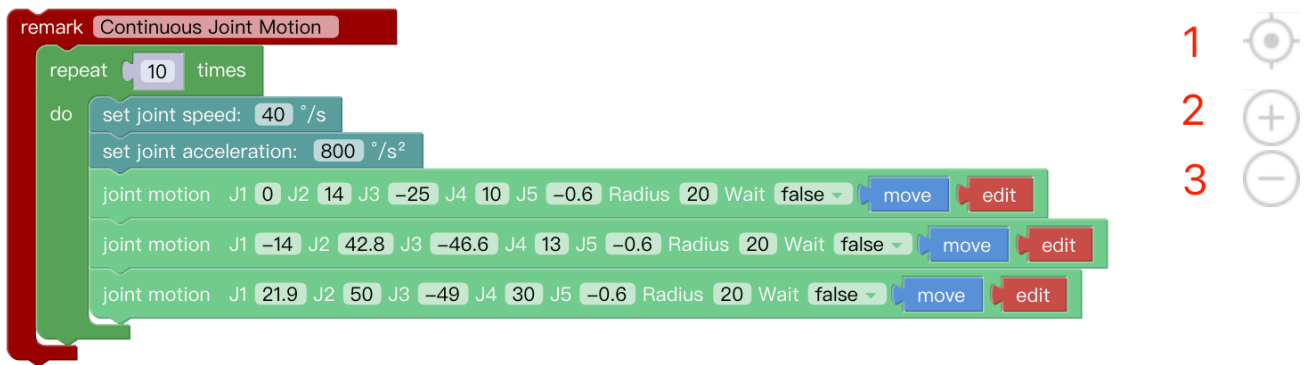
Blockly is a graphical programming tool that can be programmed to control the robotic arm by dragging and dropping code blocks without the need to write the code manually.

5.1 Interface Overview



1. Run: Run the Blockly program.
2. Pause: Pause the Blockly program.
3. Convert to Python: Convert to Python code.
4. Save: Save change to Blockly code.
5. Zoom Log: Zoom in/out the log page.
6. Zoom 3D Models: Zoom in/out the 3D models page.
7. File Operation: new, import, download, copy, paste, rename, delete.

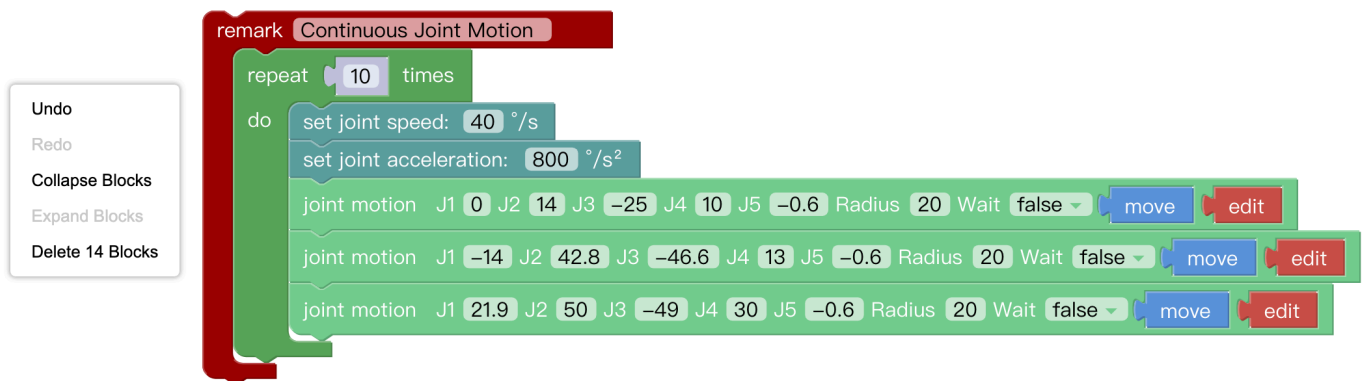
5.2 Blockly Workspace



1. Reset Workspace: Reset to default workspace.
2. Zoom In: Zoom in the code block.
3. Zoom Out: Zoom out the code block.

Right Click Mouse Event in the Workspace

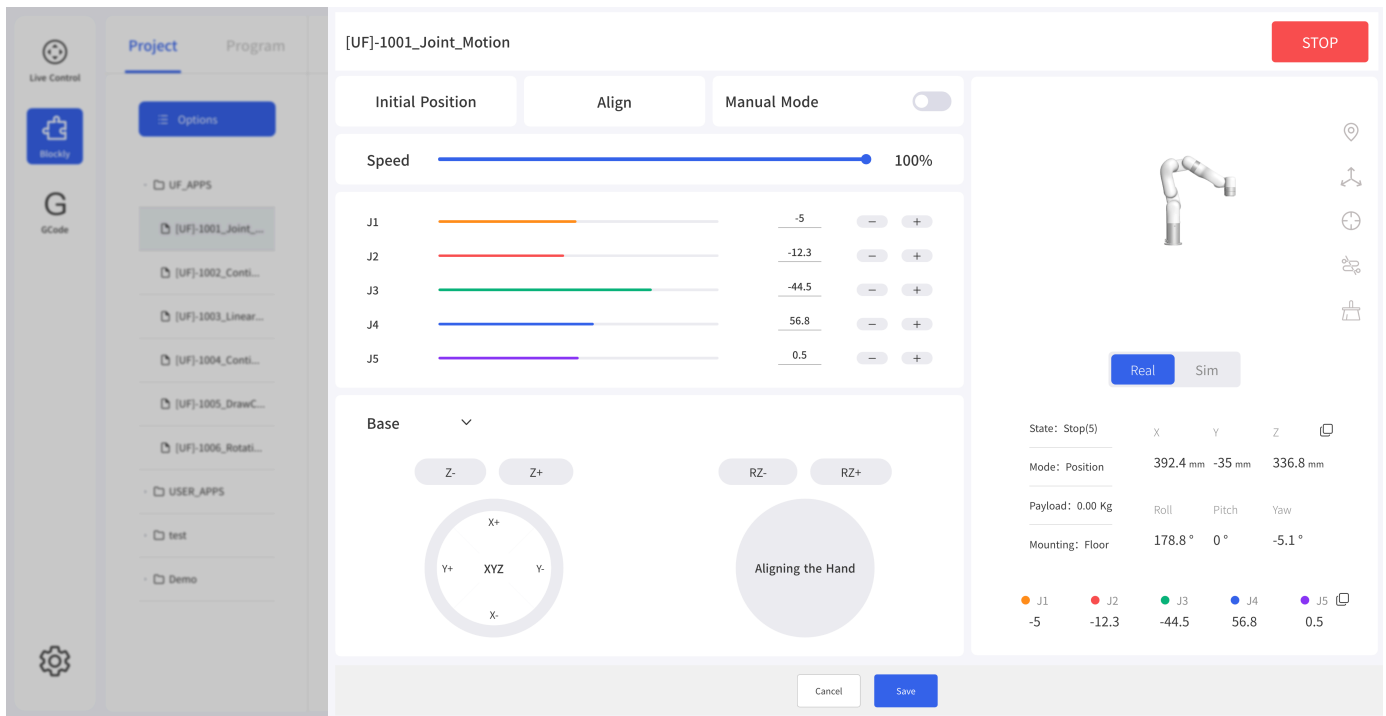
Right-click on the blank workspace of the non-code block, the function is mainly for all code blocks:



1. Undo: Undo the previous operation.
2. Redo: Restore the last undo operation.
3. Collapse Blocks: Collapse all code blocks.
4. Expand Blocks: Display all collapsed commands.
5. Delete 14 Blocks: Delete all code blocks.

Move/Wait/Edit





For some common functions of the motion commands:

1. Move: The robotic arm will move to the current position
2. Edit: The robotic arm will move to the current position and jump to the live control page.
3. Wait (true/false): indicating whether to wait for the execution of a command before sending the next one.

5.3 Blockly Code Block

5.3.1 Settings

Project	Program	[UF]-1003_Linear_Motion
		<div>set TCP speed: 230 mm/s</div> <div>set TCP acceleration: 1000 mm/s²</div> <div>set joint speed: 41 °/s</div> <div>set joint acceleration: 200 °/s²</div> <div>set TCP payload No payload Weight 0 X 0 Y 0 Z 0</div> <div>set TCP offset No offset X 0 Y 0 Z 0 Roll 0 Pitch 0 Yaw 0</div> <div>set coordinate system offset Base Coordinate System X 0 Y 0 Z 0 Roll 0 Pitch 0 Yaw 0</div> <div>set collision sensitivity 3</div> <div>counter reset</div> <div>counter plus</div> <div>get counter</div> <div>when counter > 0 do</div>
Setting		
Motion	>	
IO	∨	
Controller IO		
Tool IO		
Tool		

Set TCP speed: Set the speed of the linear motion in mm/s.

Set TCP acceleration: Set the acceleration of the linear motion in mm/s².

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

Set joint acceleration: Set the acceleration of joint motion in °/s². 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() weigh() XYZ: Set the load of the current project, refer Settings-TCP Payload from the drop-down list.

Set tcp offset() XYZRPY: Set the end offset of the current project, reference Settings-TCP Offset from the drop-down list.

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

Set collision sensitivity: Set collision detection sensitivity level.

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.

Get counter: Get the value of counter.

When counter (>)0 do: Execute when event.

5.3.2 Motion

Setting

Motion

Motion

Motion[Variable]

IO

Controller IO

Tool IO

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.

Align: Align the head.

Joint motion: Set each joint value for the joint movement, with the unit of degree.

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

Move (forward)()mm wait(): Indicates that the robotic arm makes relative linear motion forward/backward/left/right based on the current position, in mm.

Move tool line XYZRPY: This command is a relative motion relative to TCP coordinates.

Move circle: 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).

Motion with Variable: The command passes through the Cartesian motion and supports variable values.

Setting

Motion

Motion

Motion[Variable]

5.3.3 IO

5.3.3.1 Controller IO

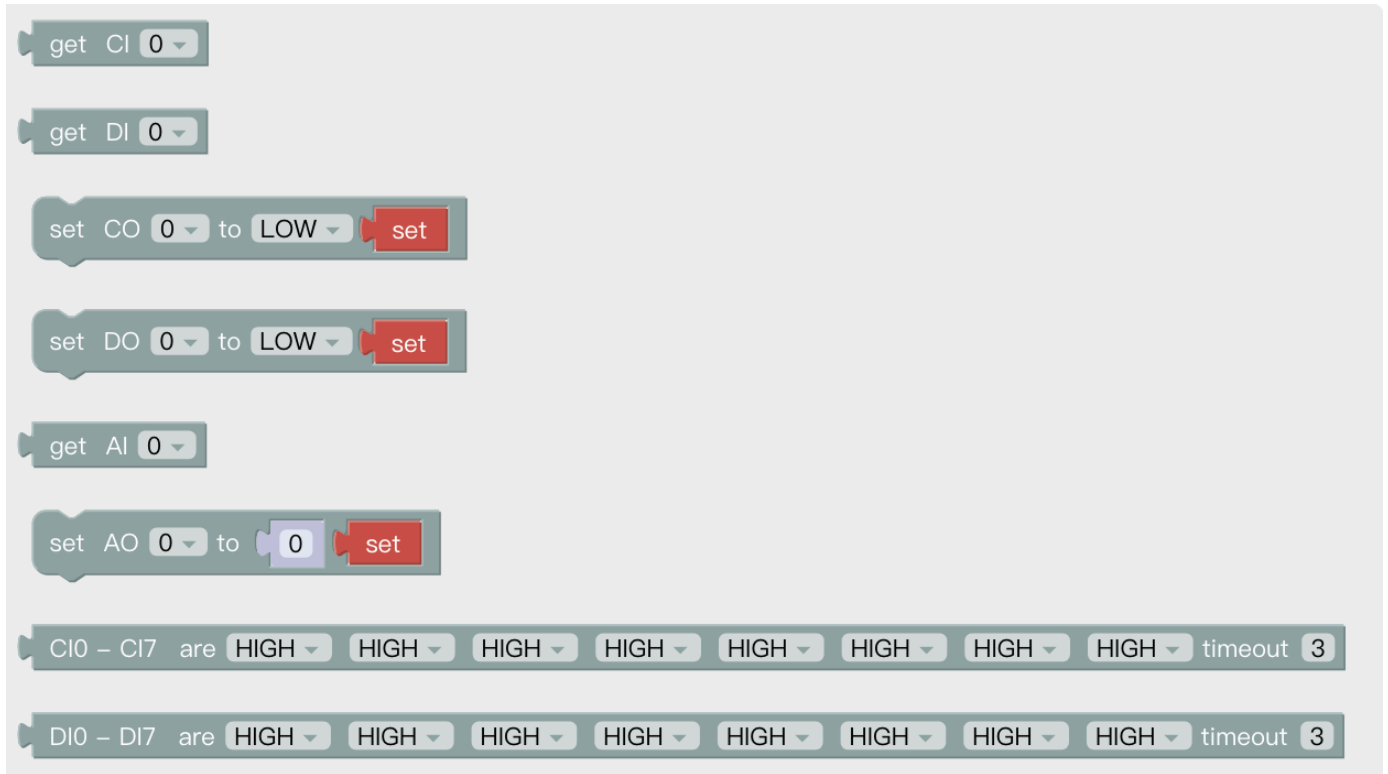
The IO interface is made up of a control box interface and an end tool interface, which can be used to acquire, set, and monitor IO interface operations.

The control box has 16 digital input, 16 digital output, 2 analog input and 2 analog output.

The end tool has 2 digital input, 2 digital output, 2 analog input and 2 analog output.

The control box digital IO is **low-level-triggered**.

The end tool digital IO is **high-level-triggered**.



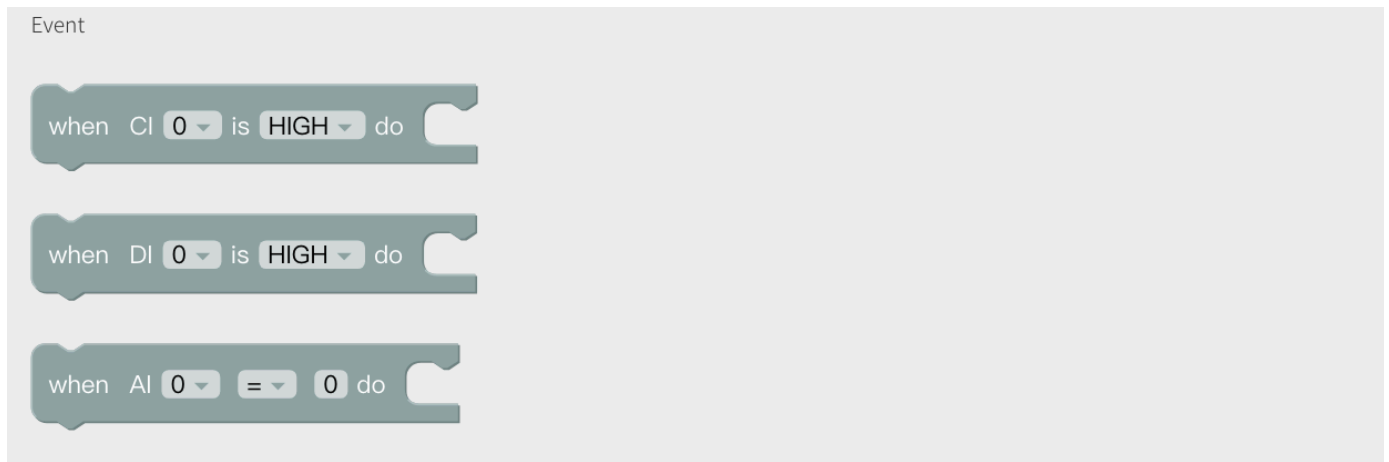
get CI(0): Acquire the I/O interface data of the code block.

set CO(0): 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 =, ≠, >, ≥, <, ≤.

5.3.3.2 Tool IO

Setting

Motion >

IO v

Controller IO

Tool IO

The image shows a sequence of Blockly code blocks for Tool IO. It starts with 'get TI 0' and 'get T-AI 0' blocks. Then there is a 'set TO 0 to LOW' block with a red 'set' button. This is followed by a 'TI 0 - TI 1 are LOW LOW timeout 3' block. Below these is a 'Position Trigger' section with a 'when X= 201.5 Y= 0 Z= 140.5 tolerance= 0 set TO 0 to LOW' block. Finally, there is an 'Event' section with two blocks: 'when TI 0 is HIGH do' and 'when T-AI 0 = 0 do'.

Similar to Controller IO.

5.3.4 Tool

Gripper

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

Gripper

set gripper Pos 300 Speed 5000 Wait true set

set gripper [variable] Pos 300 Speed 5000 Wait true

object is picked by gripper, timeout 3

Gripper

set gripper OPEN set

object is picked by gripper, timeout 3

BIO Gripper

Open/Close Bio Gripper.

BIO Gripper

initialize BIO gripper set

set BIO gripper OPEN Speed 300 Wait false set

object is picked by bio gripper timeout 3

Robotiq Gripper

Initialize Robotiq gripper, set robotiq gripper pos and force.

Robotiq Gripper

initialize robotiq gripper

set

set robotiq gripper Pos 255 Speed 255 Force 255 Wait true set

Vacuum Gripper

Get Vacuum gripper state: 0/1. 0: Picked 1: Release Object is (picked) timeout(): wait until timeout

Vacuum Gripper

get vacuum gripper state

object is picked timeout 3

set vacuum gripper ON object detection false set

5.3.5 Externals

Direct Drive Linear Motor

Set Linear Motor: Speed, position.

Direct Drive Linear Motor

set linear motor Pos 0 Speed 100 Wait true move edit

Modbus RTU

Get/Set Standard Modbus RTU command: Robot Arm, Control Box, it will add CRC automatically.

Modbus RTU

set RS-485 Port Robot Arm Modbus RTU Commands ☐ Send

get RS-485 Port Robot Arm Modbus RTU Commands ☐

For example: Set xArm Gripper to pos=600, and get the current pos.

set RS-485 Port Robot Arm Modbus RTU Commands 0x08,0x10,0x07,0x00,0x00,0x02,0x04,0x00,0x00,0x02,0x58 Send

wait 1

variable printing gripper pos= get RS-485 Port Robot Arm Modbus RTU Commands 08 03 07 02 00 02

[2024-12-14 14:34:18][127] xArm-Python-SDK Version:1.14.7

gripper pos=08 03 04 00 00 02 58

Torque Sensor

Get/Set torque sensor force control. Direction: Fx, Fy, Fz, Tx, Ty, Tz Value: Fx(-105,105), Fy(-105,105), Fz(-140,140) TxTyTz(-2.8,2.8)

Torque Sensor

set torque sensor force control base Fx value 0 duration 10 s set

get torque sensor value Fx

Modbus TCP

Definition of register please refer to: [Modbus TCP](#). When detects the value is changed, trigger the event.

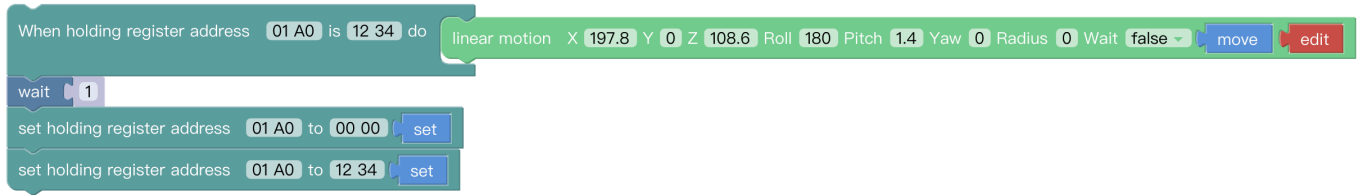
Modbus TCP

set holding register address 01 00 to 00 00 set

get holding register address 01 00

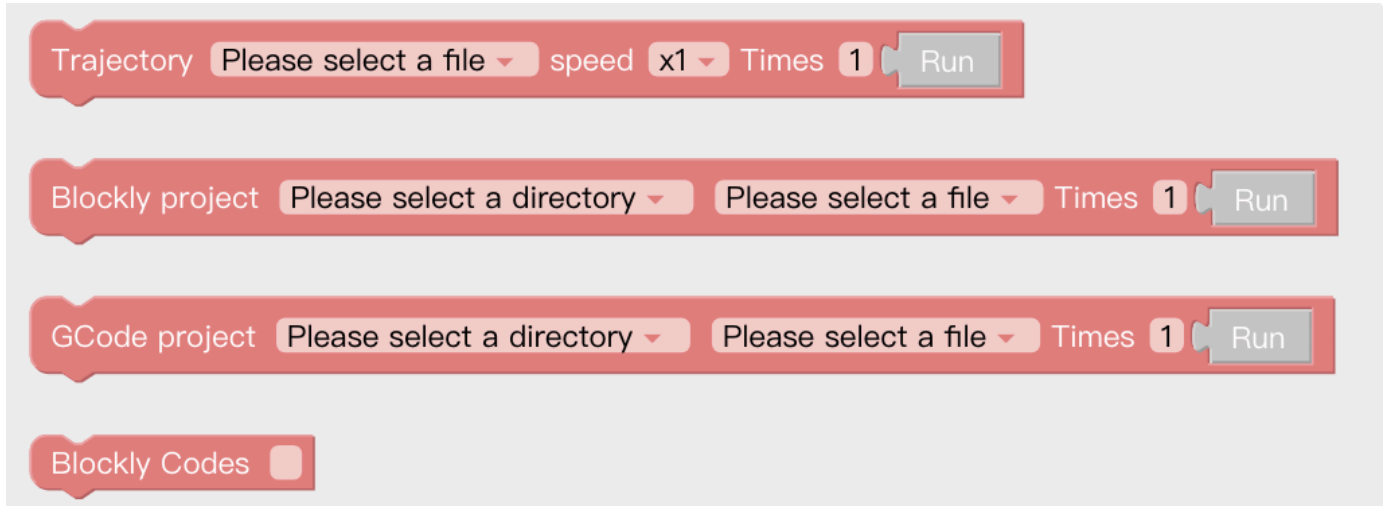
When holding register address 01 00 is 00 00 do

For example, set register address 01 A0 as 12 34, it will trigger linear motion.



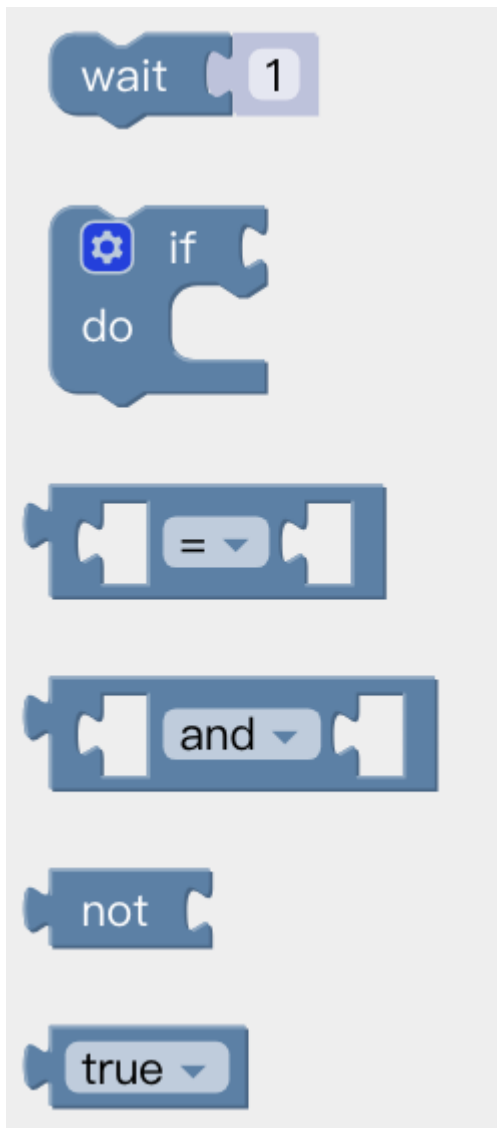
5.3.6 Import

Import the trajectory/Blockly file and set the times of executions. Copy Blockly Codes.



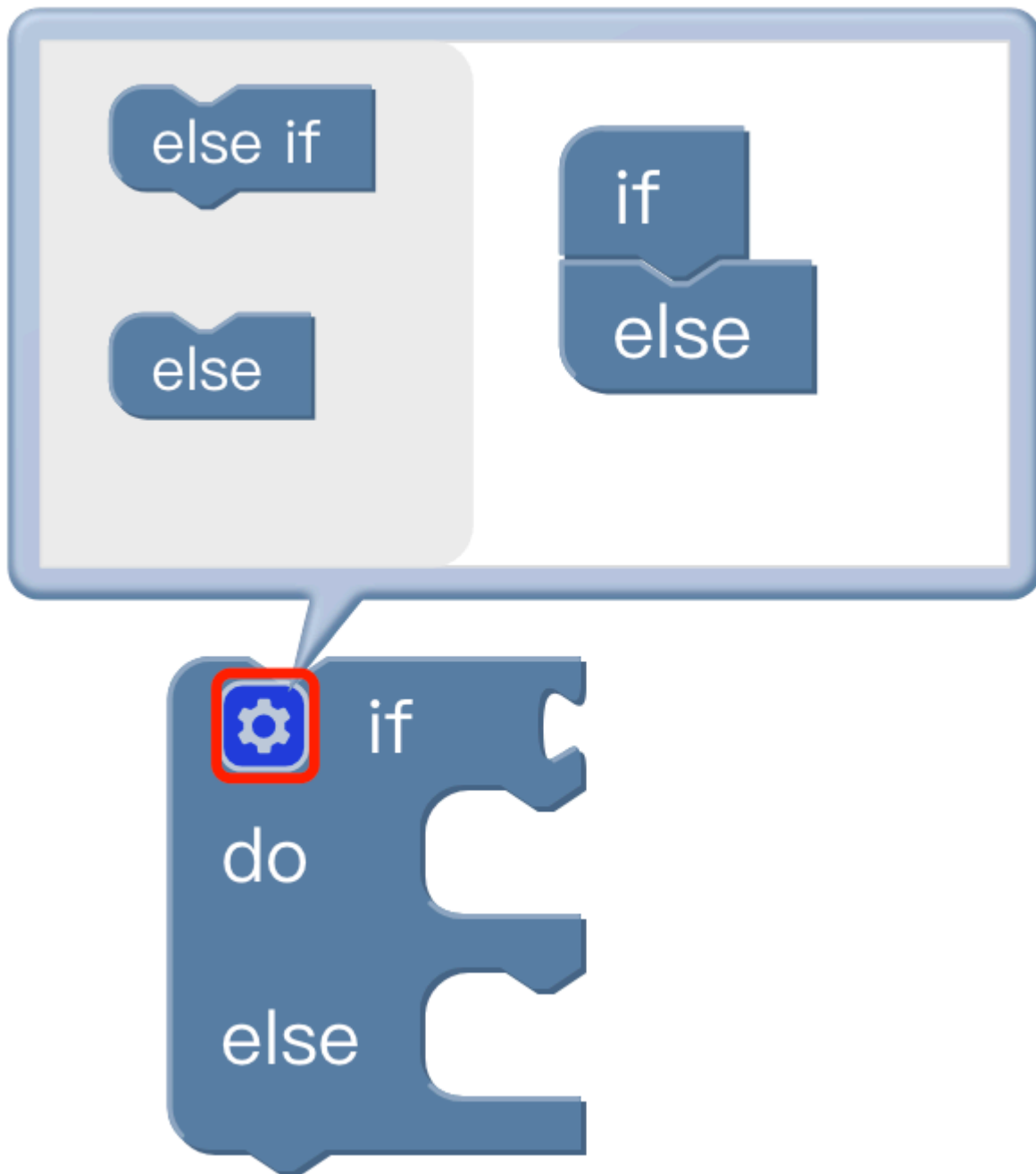
5.3.7 Logic

Wait: Wait seconds and send next commands.



If() do():

- Click the setting button on the command block, then the command block will pop up a selection box;
- At this point, drag the 'else' code block to the bottom of the 'if' code block, and combine the two code blocks;
- Click the setting button, the selection box is retracted, if/else sentence setting is completed.



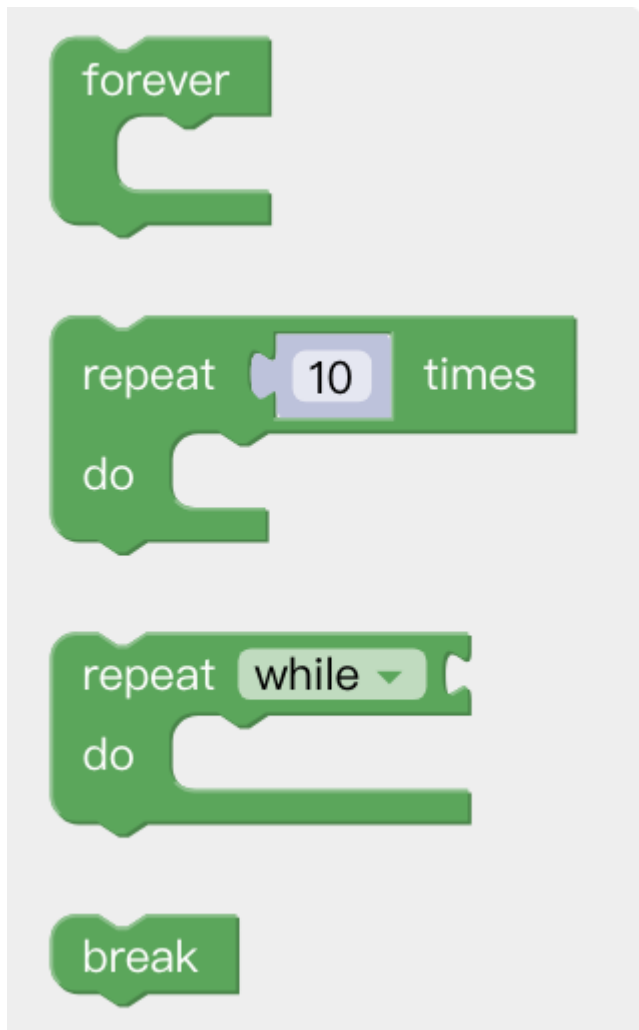
5.3.8 Loop

Forever: The command contained in the loop will be executed in infinite loop.

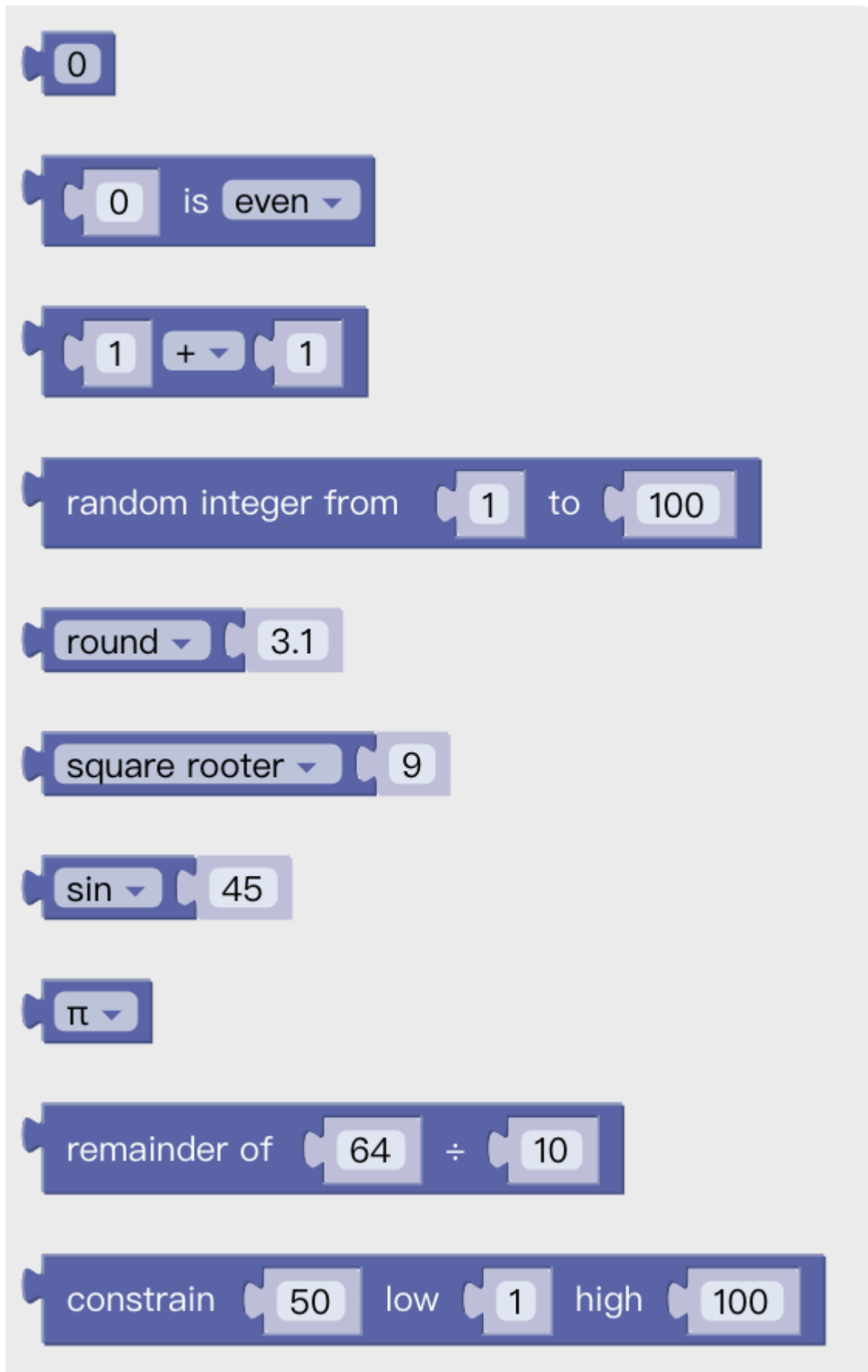
Repeat() times do: The command contained in the loop will be executed X times.

Repeat while/until do: When the condition is not met, it jumps out of the loop.

Break: Terminate the loop.



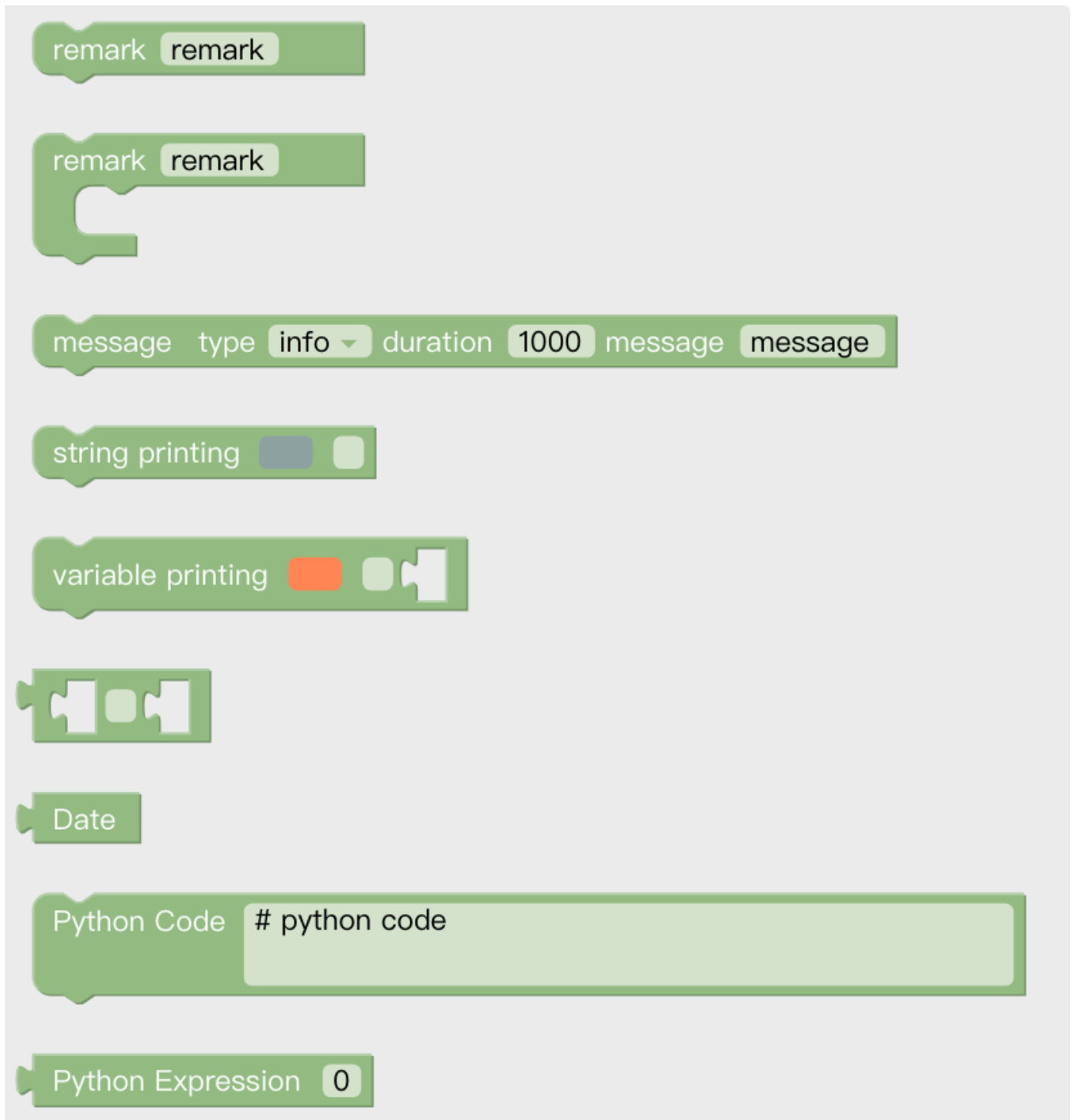
5.3.9 Math



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

5.3.10 Advance

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.

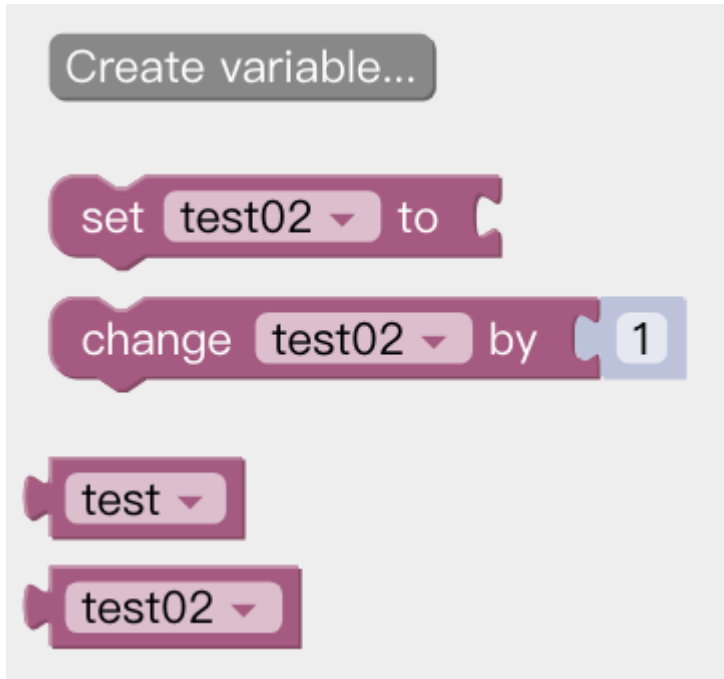
String printing[]: Users can print the entered string below and set the font and the color.

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.

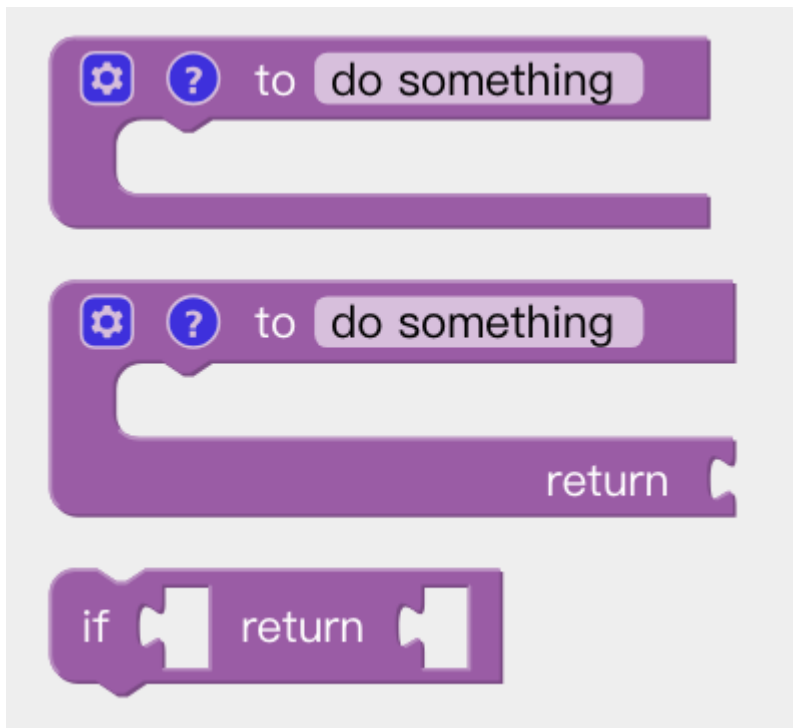
Python Code: You can write custom python code to turn it into a block in Blockly and use it in your program

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).

Function



to (do something): Users can define a new function without a return value.

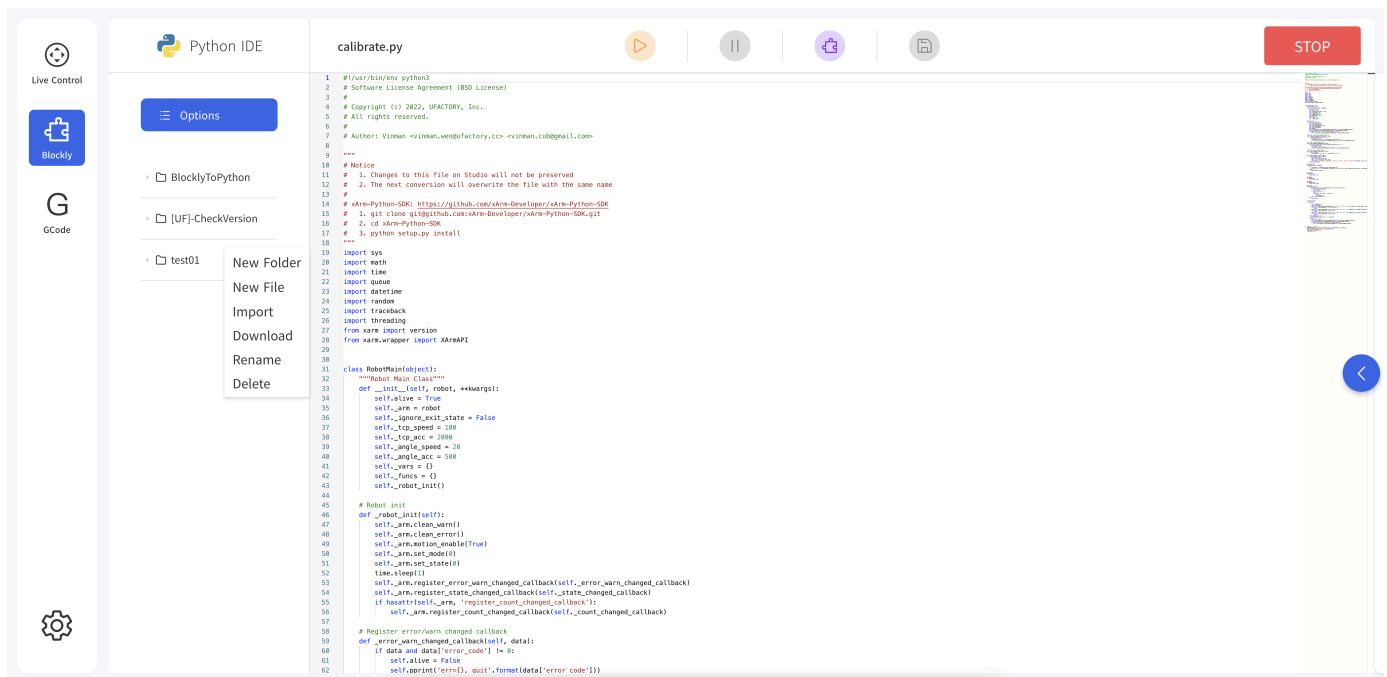
to (do something) return []: Users can define a new function with a return value.

if [] return []: Conditional judgment sentence that can only be placed in the built-in function.

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

5.4 Python IDE

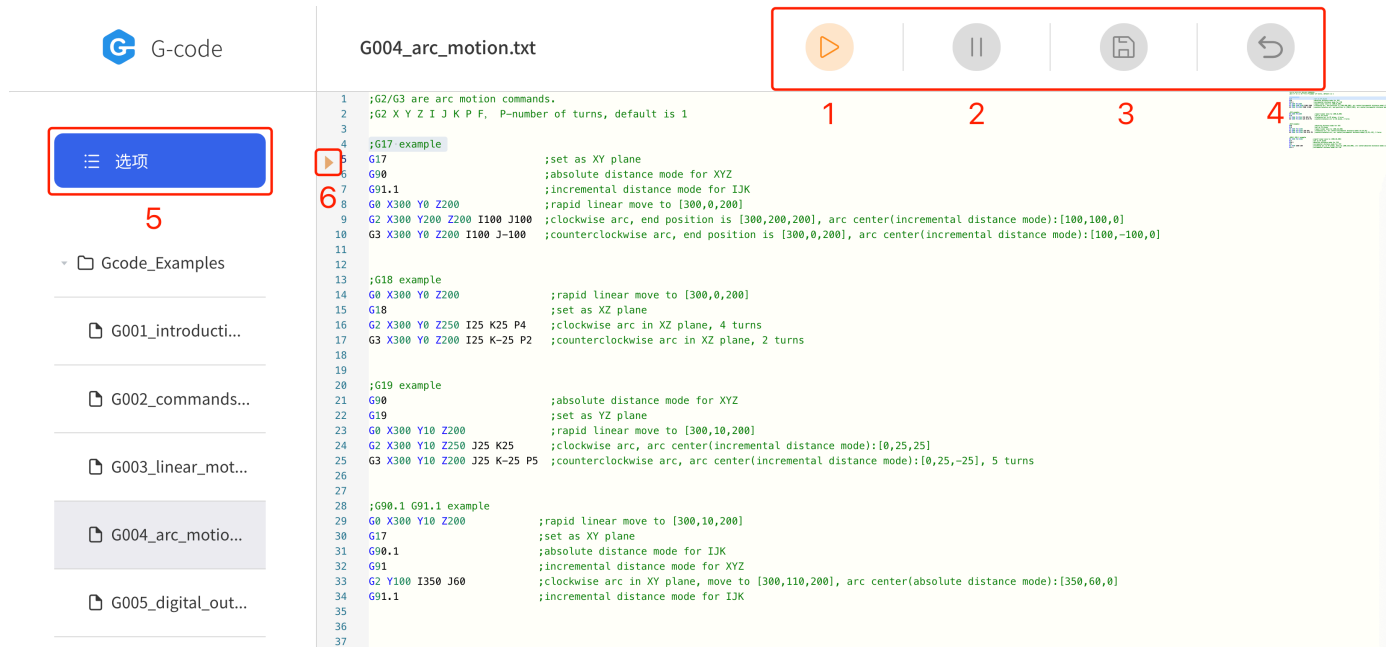
Python IDE is a Python development integration environment that can directly use xArm-Python-SDK API and check the Blockly projects converted into Python code.



6. Gcode

UFACTORY Gcode refers to the RS-274 standard and is compatible with LinuxCNC Gcode: <http://linuxcnc.org/>.

6.1 Interface Overview



1. Run: Run the Gcode program.
2. Pause: Pause the running Gcode program.
3. Save: Save change to Gcode program.
4. Withdraw: Withdraw 1 step.
5. File Operation: New, Import, Download, Rename, Delete.
6. Run: Run 1 gcode command.

6.2 G Command

G Command	Format	Description
G0	G0 X Y Z A B C	Rapid Move

G Command	Format	Description
G1	G1 X Y Z A B C F	Linear Move
G2	G2 X Y Z R P FG2 X Y Z I J K P F	Arc Move, clockwise arcR-radius, I- X offset, J- Y offset, K- Z offset, P-number of turns, default is 1
G3	G3 X Y Z R P FG3 X Y Z I J K P F	Arc Move, counterclockwise arc
G4		Dwell, Unit: s
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
G90.1		absolute distance mode for I, J & K offsets
G91		incremental distance mode
G91.1		incremental distance mode for I, J & K offsets

6.3 M Command

M Command	Format	Description
M2/M30		end program
M62	M62 P	turn on digital output synchronized with motion
M63	M63 P	turn off digital output synchronized with motion
M64	M64 P	turn on digital output immediately

M Command	Format	Description
M65	M65 P	turn off digital output immediately
M67	M67 E Q	set an analog output synchronized with motion
M68	M68 E Q	set an analog output immediately P: IONUM(0-15, 0-7: CO0-CO7, 8-15: DO0-DO7) E: IONUM(0-1), Q: value(0-10)
M100	M100 P Q	enable or disable the robotP1-enable, P0-disable, Q-joint ID(8 by default, stands for all joints).
M101		clear error
M102		clear warning
M103	M103 P	set mode
M104	M104 P	set state
M115	M115 P Q	set TGPIOP: IONUM 0/1/2/3/4Q: 0/ 1/ 10/ 11 Q0: turn off(low level) tool digital output synchronized with motion(wait=True). Q1: turn on(high level) tool digital output synchronized with motion(wait=True). Q10: turn off tool digital output immediately(wait=False). Q11: turn on tool digital output immediately(wait=False).
M116	M116 P Q	control the end effectorP1: xArm Gripper, Q-positionP2: xArm Vacuum Gripper Q0:open(wait=True),Q1:close(wait=True), Q10:open(wait=False), Q11:close(wait=False)P3: xArm BIO Gripper Q0: close, Q1: openP4/P5: Robotiq-2F-85 Gripper, Robotiq-2F-140 Gripper, Q:position(0~255)P11: Lite6 Gripper Q0:close(wait=True), Q1:open(wait=True), Q10:close(wait=False),Q11:open(wait=False)P12: Lite6 Vacuum Gripper Q0:close(wait=True), Q1:open(wait=True Q10:close(wait=False),Q11:open(wait=False)

6.4 Firmware Request

- Firmware Version: v2.2.0+
- Port: 504
- Response: 5 bytes

1. byte0: return value. 0 is success
2. byte1: mode and states
3. byte2: error code
4. byte3&byte4: buffer

- Recommend to send 1-non-empty data at a time(with line breaks)

```
sock.send(b'G0 X300\n')
```

python

- Need to receive the response, otherwise the buffer will be full.

```
sock.recv(5)
```

python

- A maximum of 2000 commands can be accepted.

7. Settings

7.1 Motion

7.1.1 Parameters

Joint Motion

Joint Step

1°

Line Motion

Position Step

1mm

Attitude Step

1°

Collision Detection

Sensitivity

3

Initial Position

J1: 0°

J2: 0°

J3: 0°

J4: 0°

J5: 0°

J6: 0°

J7: 0°

Set >

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

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.

Collision Detection Sensitivity: 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.

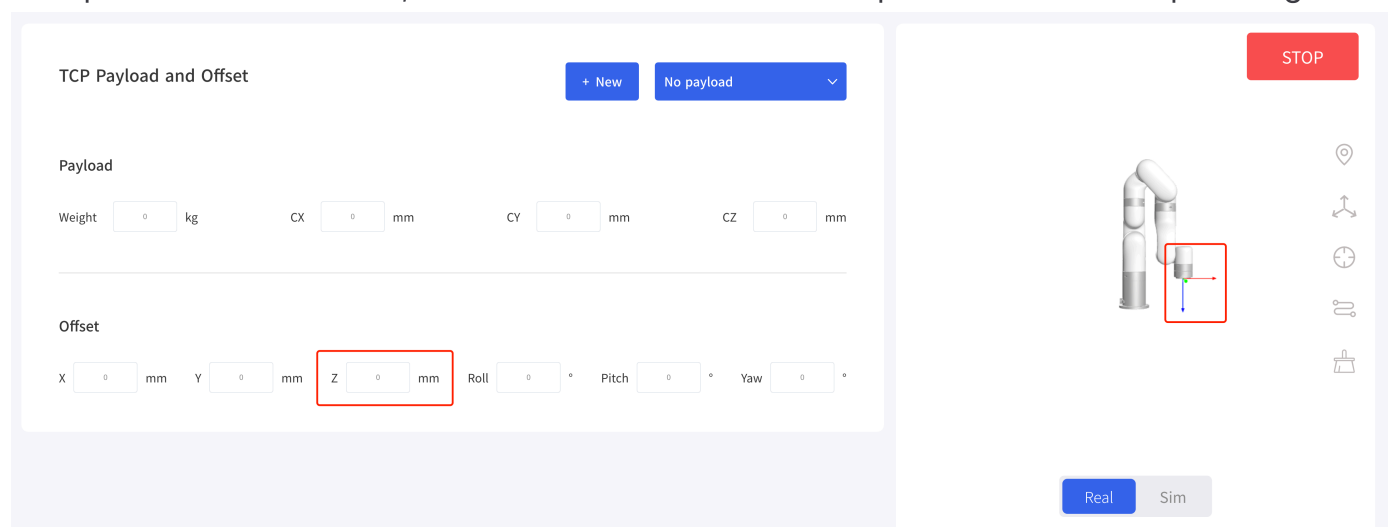
Initial Position: Setting the Initial Position of the robotic arm can help the user to return the robotic arm to a relatively safe position when planning the motion trajectory.

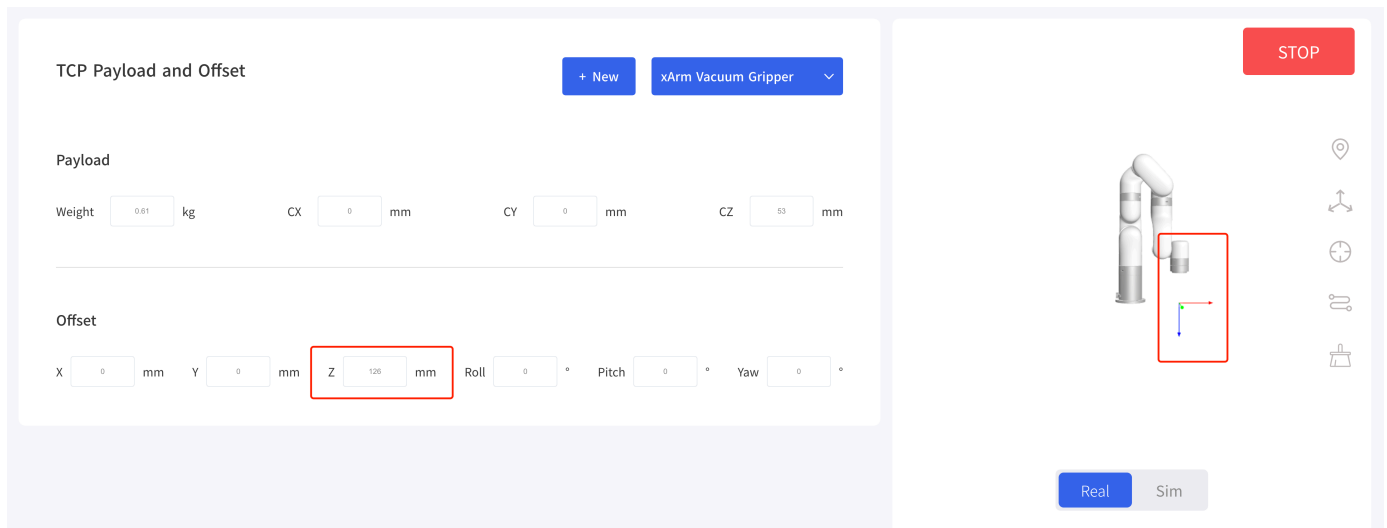
7.1.2 TCP

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, and Yaw determine the orientation. When the specified value is zero, TCP coincides with the centre point of the tool output flange.





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.

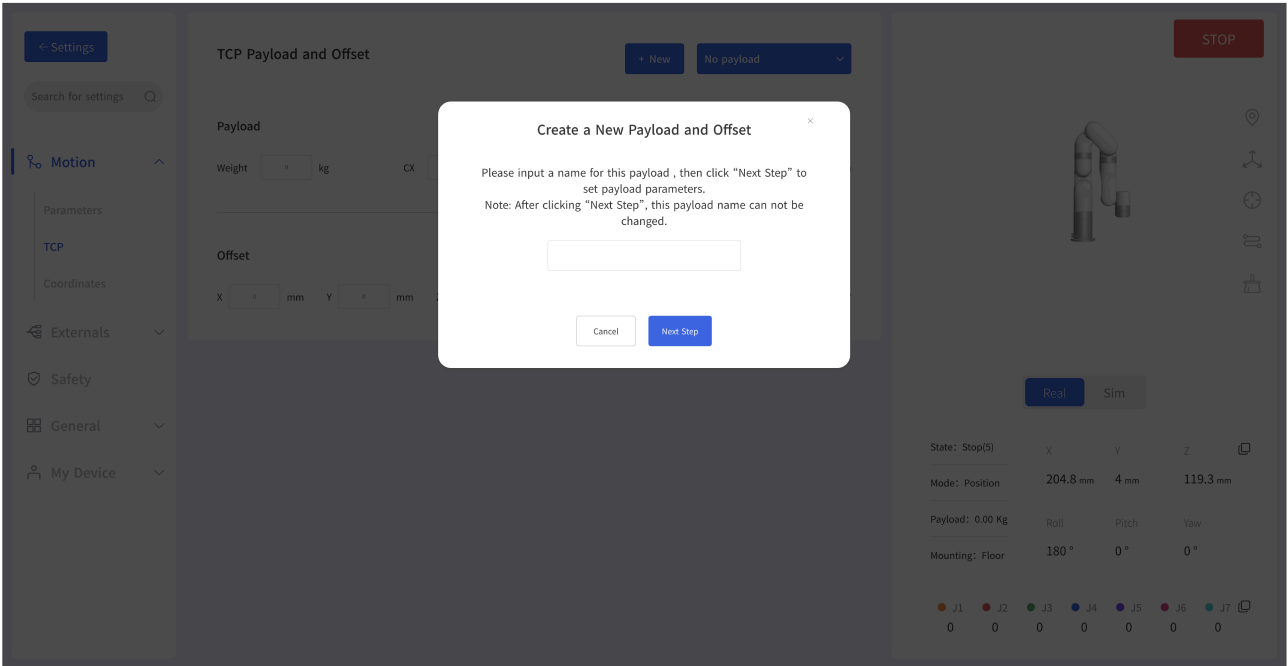
Default TCP payload: [kg,Cx,Cy,Cz][x,y,z,roll,pitch,yaw]

- No Payload: [0,0,0,0], [0,0,0,0,0,0]
- xArm Vacuum Gripper: [0.61,0,0,53], [0,0,126,0,0,0]
- xArm Gripper: [0.82,0,0,48], [0,0,172,0,0,0]
- xArm BIO Gripper: [0.72,22.39,3.22,23.55], [159.5,0,59.5,0,0,0]
- Robotiq-2F-85 Gripper: [0.925,0,0,58], [0,0,174,0,0,0]
- Robotiq-27-140 Gripper: [1.025,0,0,73], [0,0,244,0,0,0]

Create New TCP Payload and Offset

- Method 1: Manual Input When the TCP offset parameter of the end effector is known, you can choose to manually input its TCP offset parameter.

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



Set Offset Parameters

Click "Teaching Tool Center" to get the offset by teaching the robot 5 positions.If the offset of the end effector is known, please select "Manual Input".

Manual Input

Teaching Tool Center

- Method 2: Automatic identification. 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. Teaching 5 points to get the TCP offset.

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.

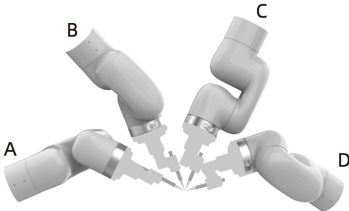
Teaching

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

7.1.3 Coordinates

Base Coordinates

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.

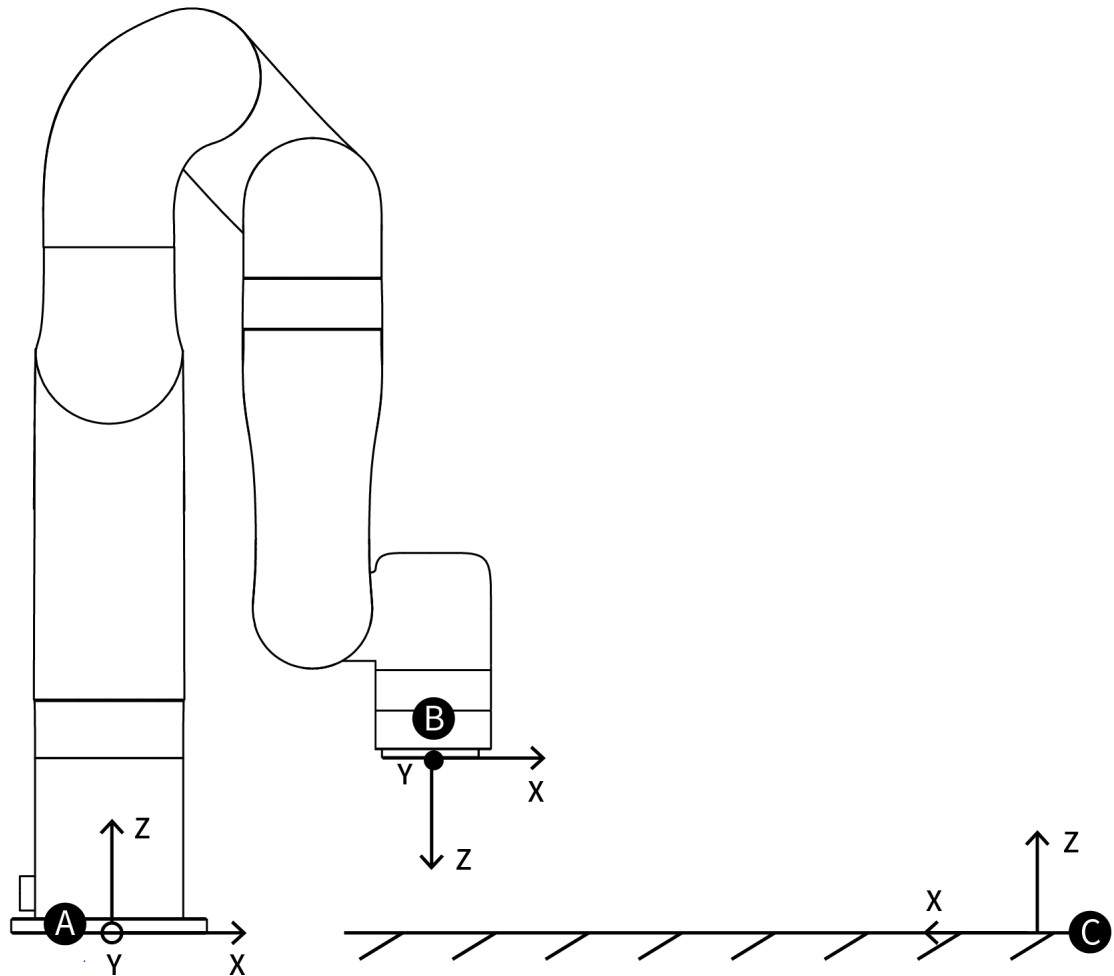
Create a new Coordinates

- Method 1: Manual Input
- Method 2: Teaching UCS. 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.

[Back](#)[Next](#)

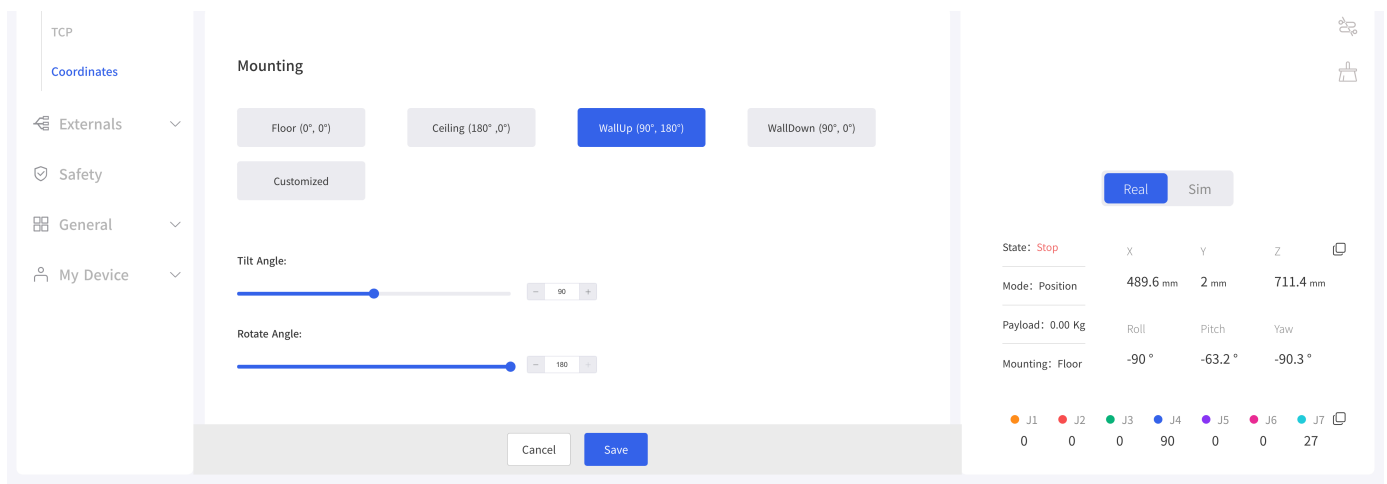


A:Base Coordinate B:Tool Coordinate C:User Coordinate

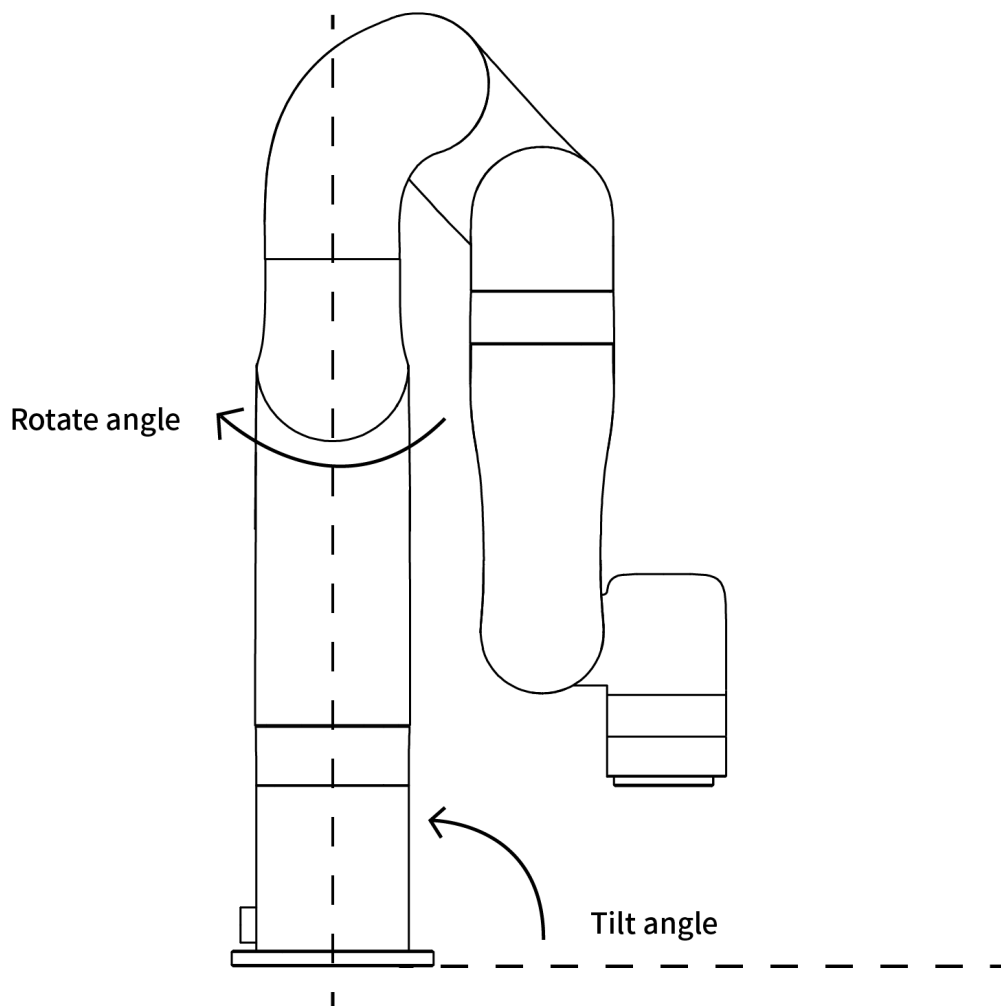
Mounting

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.



How to determine the tilt angle and rotation angle?

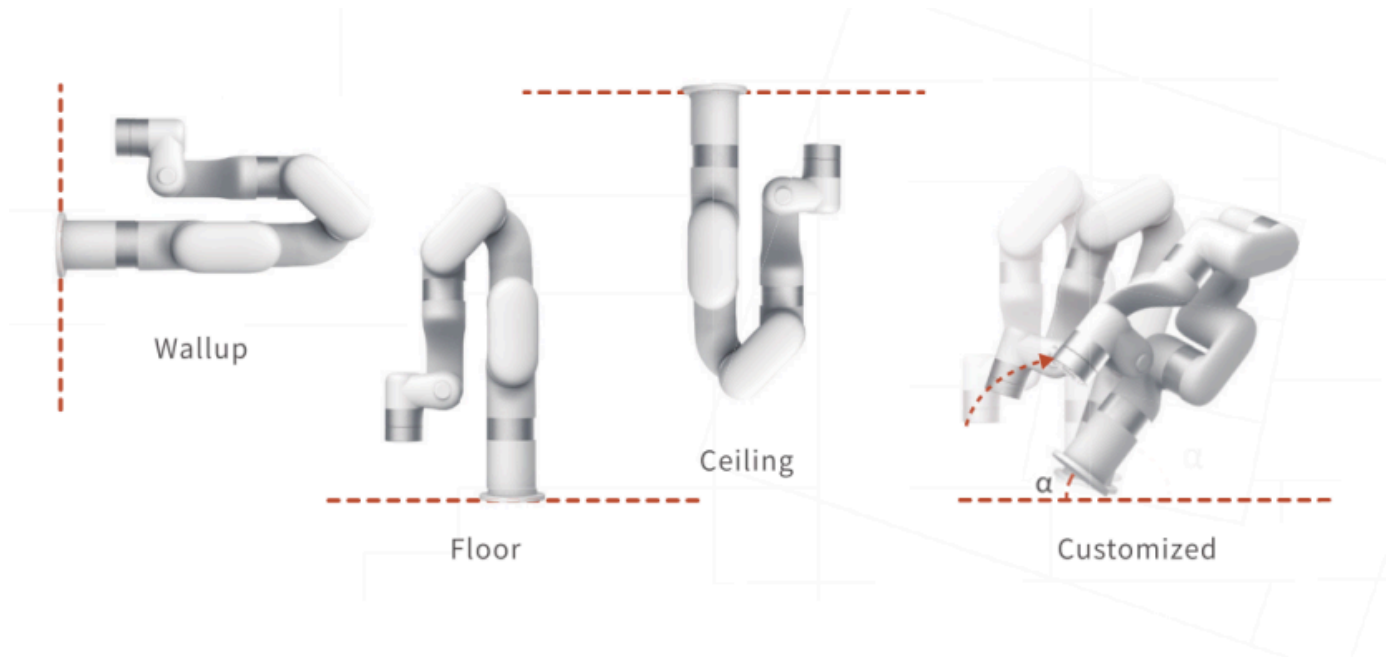


The initial position of the robotic arm: On the horizontal plane, when the user is facing the robotic arm side, the initial position is on the left-hand side of the user in a downward direction.

- **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 \pm 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: $\pm 180^\circ$



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.

7.2 Externals

7.2.1 End Effector IO

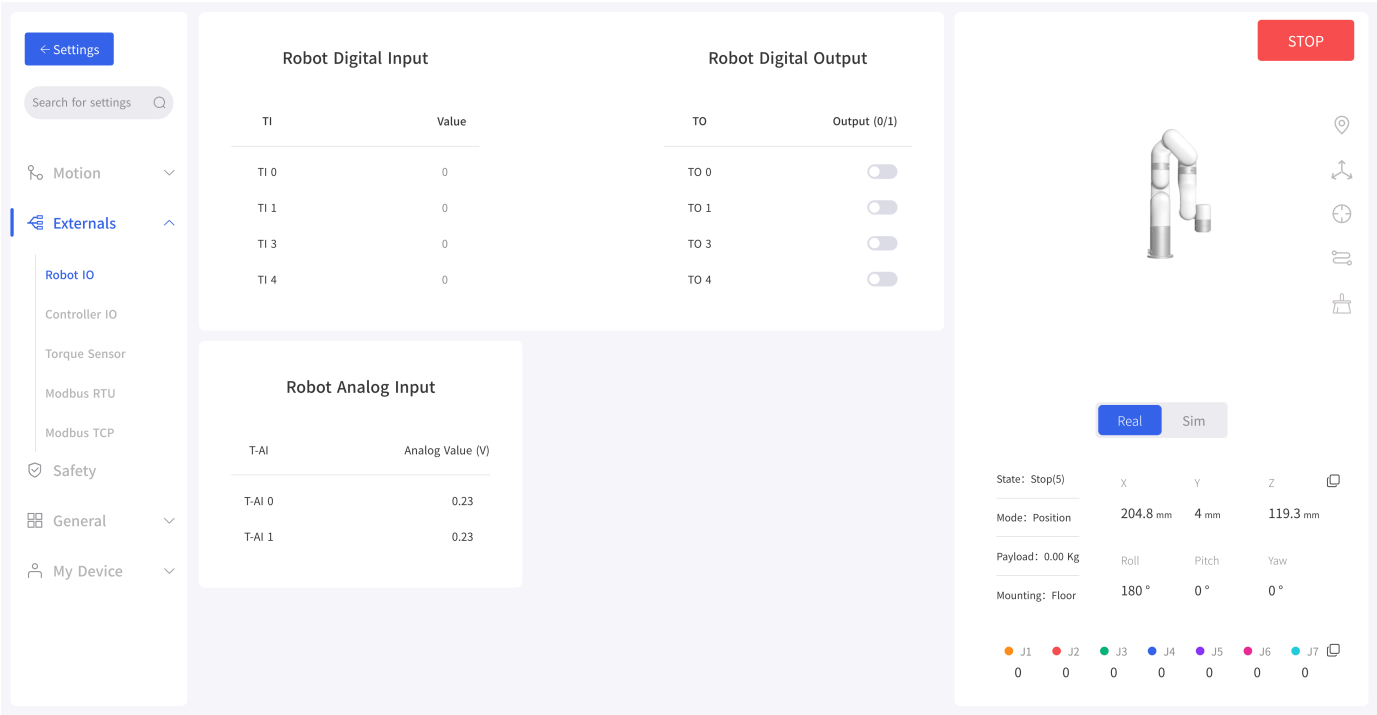
Monitor the digital input and analog input status of the external device to the robot arm, and set the digital output at the end of the robot arm.

Update Frequency: 5HZ.

Digital Input: TI0, TI1, TI2, TI3, TI4, low level by default

Digital Output: TO0, TO1, TO2, TO3, TO4.

Robot Analog Input: TAI0, TAI1, 0V by default, [0-3.3V].



7.2.2 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.

IO State

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.

Digital Input: CI0-CI7, DI0-DI7, High level by default.

Digital Output: CO0-CO7, DO0-DO7, Low level by default.

Analog Input: AI0, AI1.

Analog Output: AO0, AO1, 0V by default, [0-10V].

← Settings

Search for settings

Motion

Externals

Robot IO

Controller IO

Torque Sensor

Modbus RTU

Modbus TCP

Safety

General

My Device

IO State

IO Function

Digital Input

CI	Value	DI	Value
CI0	1	DI0	1
CI1	1	DI1	1
CI2	1	DI2	1
CI3	1	DI3	1
CI4	1	DI4	1
CI5	1	DI5	1
CI6	1	DI6	1
CI7	1	DI7	1

Analog Input

Input	Analog Value (V)
AI0	0.00
AI1	0.01

Digital Output

CO	Output(0/1)	DO	Output(0/1)
CO0	1	DO0	1
CO1	0	DO1	0
CO2	0	DO2	0
CO3	0	DO3	0
CO4	0	DO4	0
CO5	0	DO5	0
CO6	0	DO6	0
CO7	0	DO7	0

Analog Output

Output	Analog Value (V)	Output
A00	0	Set
A01	1.00	Set

STOP

Real Sim

State: Stop(5)

X

Y

Z

Mode: Position

204.8 mm

4 mm

119.3 mm

Payload: 0.00 Kg

Roll

Pitch

Yaw

Mounting: Floor

180 °

0 °

0 °

J1

J2

J3

J4

J5

J6

J7

0

0

0

0

0

0

0

IO Function

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

IO State

IO Function

Input Configuration

Output Configuration

CI0

General Input

General Input

Stop Moving

Safeguard Reset

Offline Task

Manual Mode

Reduced Mode

Enable Robot

CI3

General Input

General Input

CI4

General Input

General Input

CI5

General Input

General Input

CI6

General Input

General Input

CI7

General Input

General Input

DI0

General Input

General Input

DI1

General Input

General Input

DI2

General Input

General Input

DI3

General Input

General Input

DI4

General Input

General Input

DI5

General Input

General Input

DI6

General Input

General Input

DI7

General Input

General Input

CO0

General Output

General Output

CO1

General Output

General Output

CO2

General Output

General Output

CO3

General Output

General Output

CO4

General Output

General Output

CO5

General Output

General Output

CO6

General Output

General Output

CO7

General Output

General Output

DO0

General Output

General Output

DO1

General Output

General Output

DO2

General Output

General Output

DO3

General Output

General Output

DO4

General Output

General Output

DO5

General Output

General Output

DO6

General Output

General Output

DO7

General Output

General Output

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.

Offline Task: Offline Task can add multiple Blockly to be triggered through I/O.

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.

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

7.2.3 Torque Sensor

Is the 6 Axis Force Torque Sensor Installed

SN: N/A Firmware Version: N/A

Use force torque sensor for manual mode and recording.

Payload Identification

Mass 0 kg Centroid [0, 0, 0] mm
Sensor Offset [0, 0, 0] N [0, 0, 0] N·m

Identify

Manual Mode Direction

☒ Translate

X ☐

Y ☐

Z ☐

☐ Rotate

Rx ☐

Ry ☐

Rz ☐

Inactivated

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

7.2.4 Linear Motor

This module will only be seen when connecting to the linear motor. Control: Position, Speed.

7.2.5 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.

RS-485 Port

Robot Arm

Modbus Baud Rate

2000000

Timeout

5

ms

Number	Check	Comments	Commands	Delay ms
1	<input checked="" type="radio"/>		0x08,0x03,0x07,0x02,0x00,0x02	
2	<input type="radio"/>			
3	<input type="radio"/>			
4	<input type="radio"/>			
5	<input type="radio"/>			
6	<input type="radio"/>			
7	<input type="radio"/>			
8	<input type="radio"/>			

☒ CRC

Cyclic

☐

Send

Clean Log

Log

[12:17:15:208]Send--> 0x08,0x03,0x07,0x02,0x00,0x02

[12:17:15:217]Receive<-- 08 03 04 00 00 02 27

- For example:
- Selects the robot arm Modbus or control box Modbus.
 - Sets the baud rate, the default baud rate is 2000000.
 - Enter commands in the "Commands" box, for example: 0x08,0x03,0x07,0x02,0x00,0x02, note that the program will do CRC checksum automatically.

0x08,0x03,0x07,0x02,0x00,0x02

bash

- 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.

7.2.6 Modbus TCP

This page allows you to send standard Modbus TCP command, the IP is controller ip, port 502, can not be modified.

IP Address

192.168.1.206

Port

502

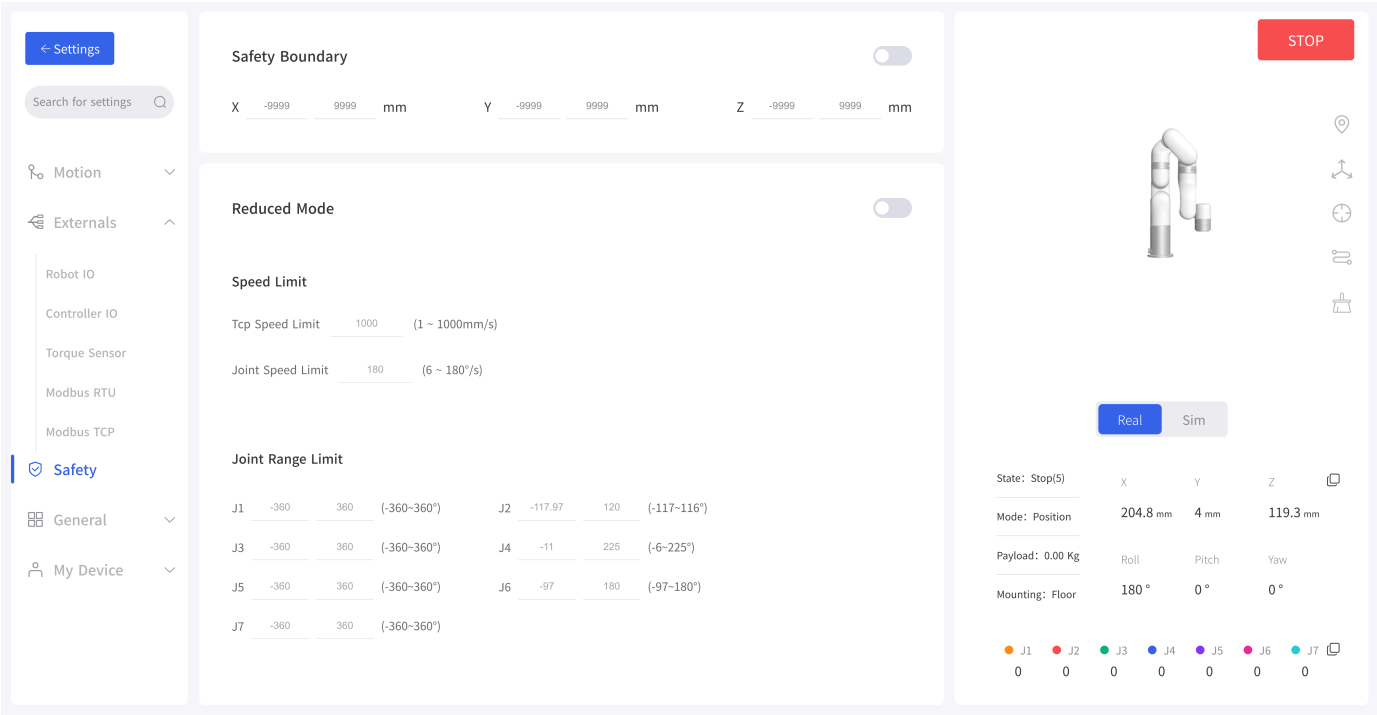
Number	Check	Comments	Commands	Delay ms
1	<input checked="" type="checkbox"/>		00 01 00 00 00 06 01 06 00 04 03 E8	
2	<input type="checkbox"/>			
3	<input type="checkbox"/>			
4	<input type="checkbox"/>			
5	<input type="checkbox"/>			
6	<input type="checkbox"/>			
7	<input type="checkbox"/>			
8	<input type="checkbox"/>			
Cyclic <input type="checkbox"/>				<div>Send</div> <div>Clean Log</div>
Log				
<div>[12:17:44:761]Send--> 00 01 00 00 00 06 01 06 00 04 03 E8</div> <div>[12:17:44:772]Receive<-- 00 01 00 00 00 06 01 06 00 04 03 E8</div>				

For example:
Send '00 01 00 00 00 06 01 06 00 04 03 E8', set controller analog output AO1 as 1V.

00 01 00 00 00 06 01 06 00 04 03 E8

bash

7.3 Safety



7.3.1 Safety Boundary

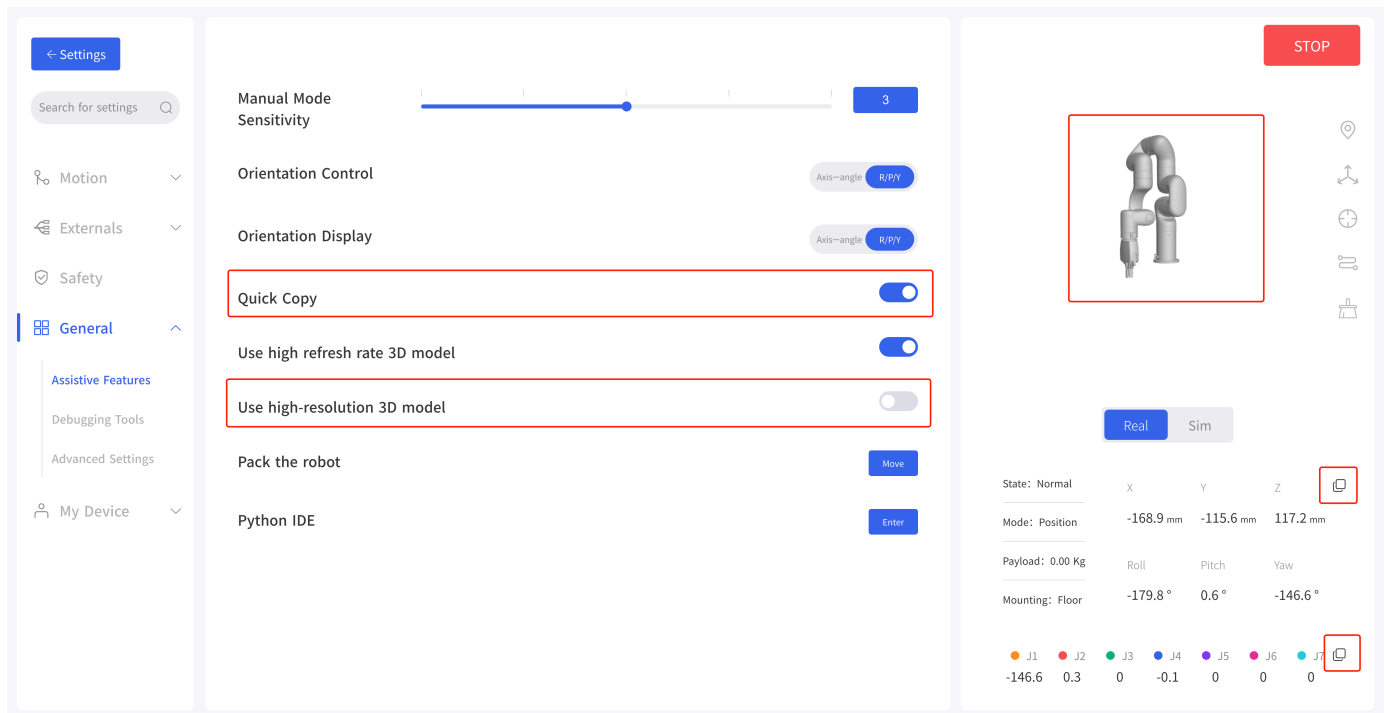
When this mode is turned on, the working range of the robotic arm in Cartesian space can be limited. If the tool center point (TCP) of the robotic arm exceeds the set safety boundary, the robotic arm will stop moving. The user can then adjust the robotic arm back into the restricted space.

7.3.2 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.

7.4 General

7.4.1 Assistive Features



Manual Mode Sensitivity: Adjust manual mode

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.

Orientation Display: Similar to orientation control.

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.

Use high refresh rate 3D model: switch to high refresh rate 3D model.

Use high-resolution 3D model: Switch to high-resolution 3D model.

Pack the robot: Move the robotic arm to pack position.

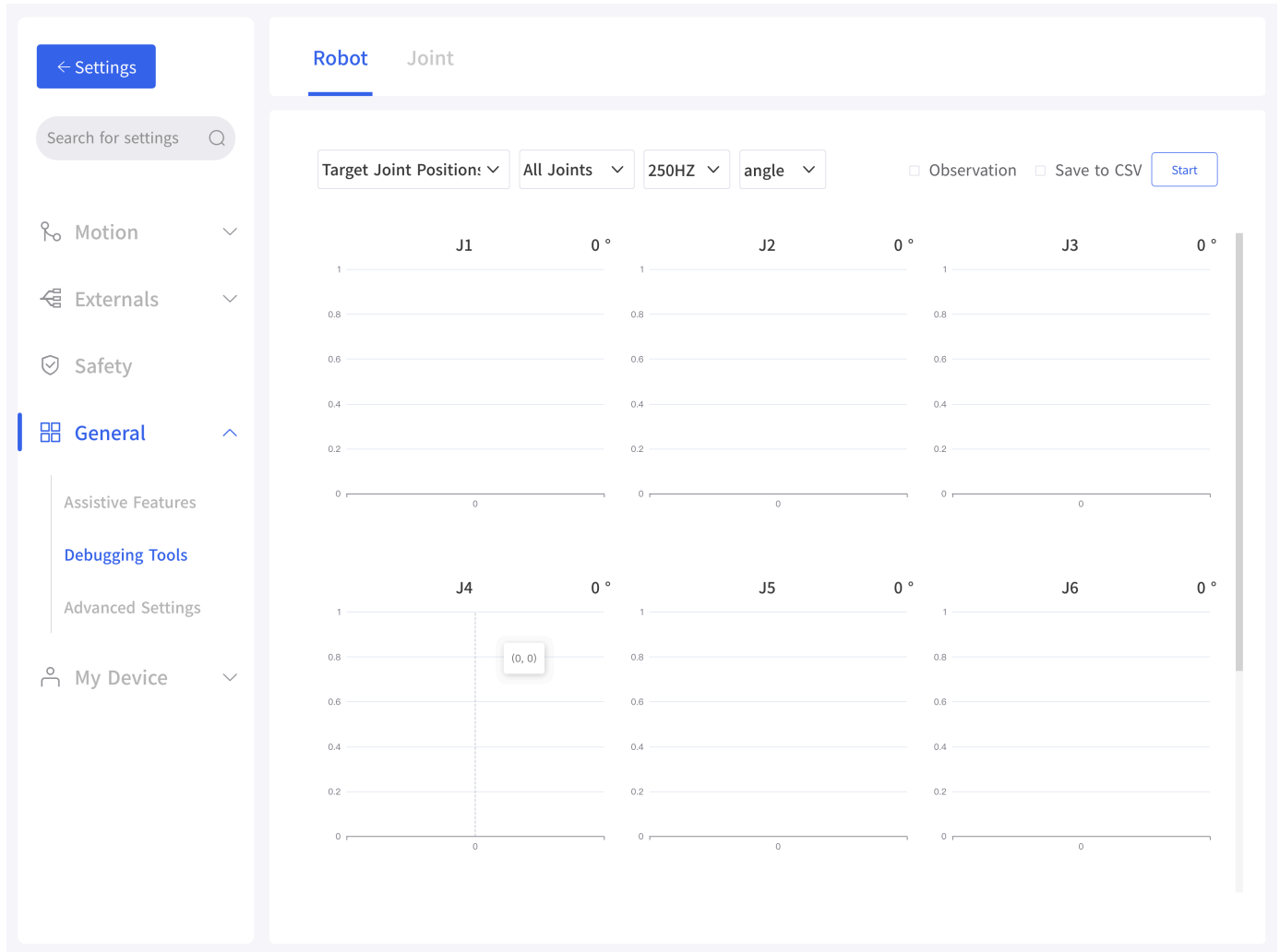
Python IDE: Enter into python IDE module.

7.4.2 Debugging Tools

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. You can also check the record CSV and download it after the observation and send it to technical support.

Robot



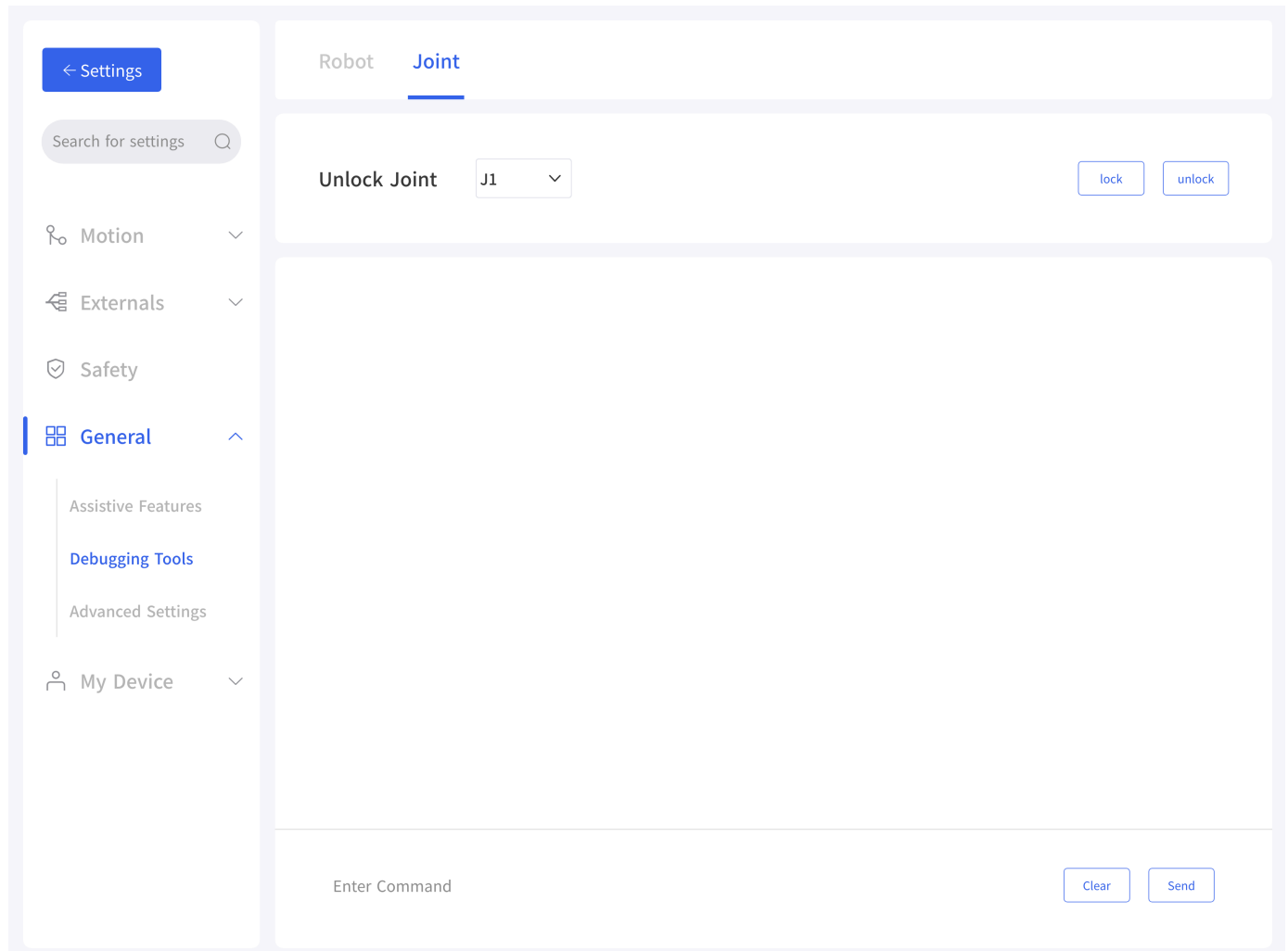
Data Item:

- Target Joint Positions
- Target Joint Velocities
- Target Joint Accelerations
- Actual Joint Positions
- Actual Joint Velocities
- Actual Joint Accelerations
- Actual Joint Currents
- Estimated Joint Torques
- Target TCP Pose
- Target TCP Speed
- Actual TCP Pose

- Actual TCP Speed
- Estimated TCP Torques

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.



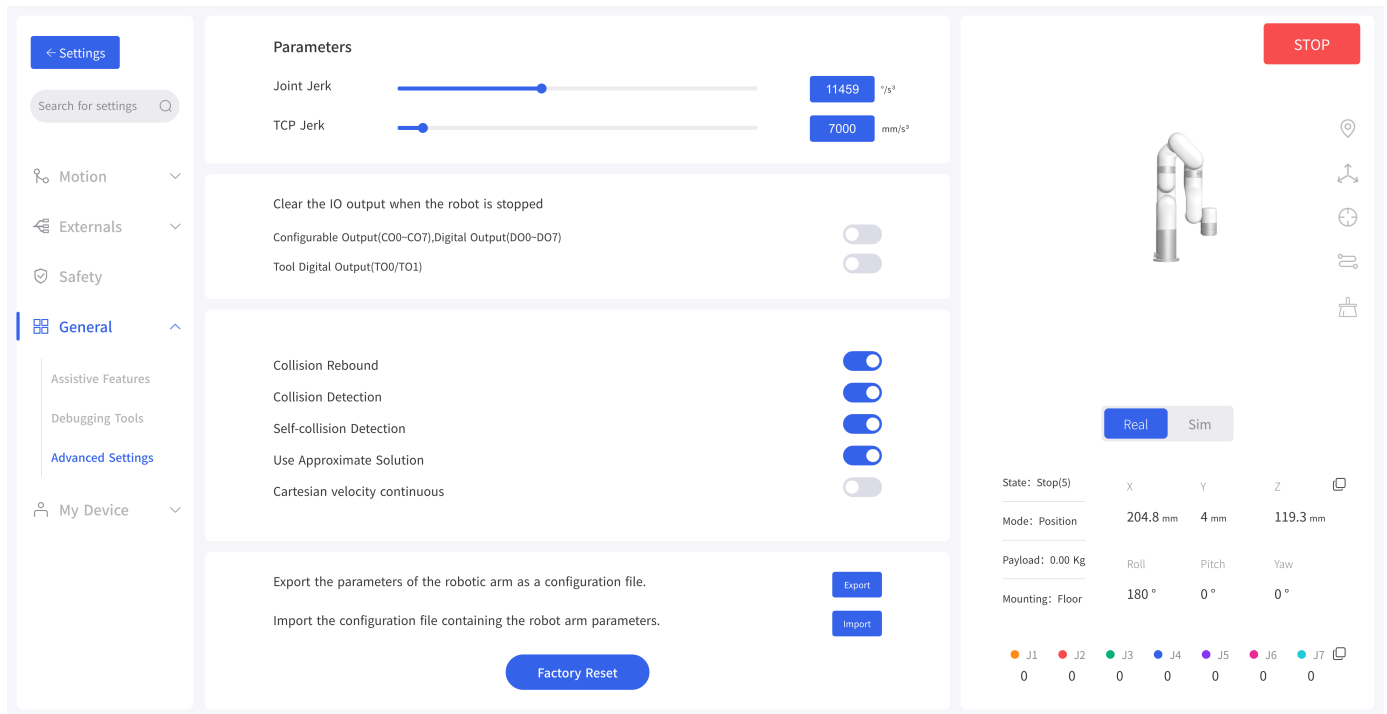
- Please ensure to hold the robotic arm by hand when unlocking the joint to prevent it from falling down due to the inadequate provision of force, and take measures to protect the surrounding environment and peripheral facilities.
- The operation of the unlocking joint is mainly used to adjust the posture 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.
- In the 'simulated robotic arm mode', clicking the unlock joint button will also unlock the real joints of the robotic arm.

DANGER: When releasing the joint brakes, someone must support the robot's posture to prevent the robotic arm from falling without external force and damage the robotic arm

and surrounding equipment.

CAUTION: After the release of the joint brake and manually dragging the robotic arm, please always pay attention to the degree of joint rotation to avoid exceeding the rotation range of the robot joint and damage the internal structure of the robotic arm.

7.4.3 Advanced Settings



You must enter password to access this page, the default password: admin.

Joint Jerk: 6~28647, the default is 11459°/s³.

TCP Jerk: 1~100000, the default is 7000mm/s³.

- The jerk affects the acceleration performance of the robotic arm. In general, we do not recommend modifying this parameter.
- If the robotic arm is not enabled, the jerk cannot be modified.
- If an error warning occurs on the robotic arm, the jerk cannot be modified.
- When the robotic arm is moving, the jerk can not be modified.

Clear the IO output when the robot is stopped: The robotic arm will be reset to the default status after sending STOP command.

- Configurable Output(CO0~CO7), Digital Output(DO0~DO7)
- Tool Digital Output(TO0/TO1)

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/off collision detection

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

Use Approximate Solution: Use approximate solution to pass Singularities.

Cartesian velocity continuous: Set cartesian velocity continuous.

Export: 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.

Import: Import the configuration file containing the parameters of the robotic arm.

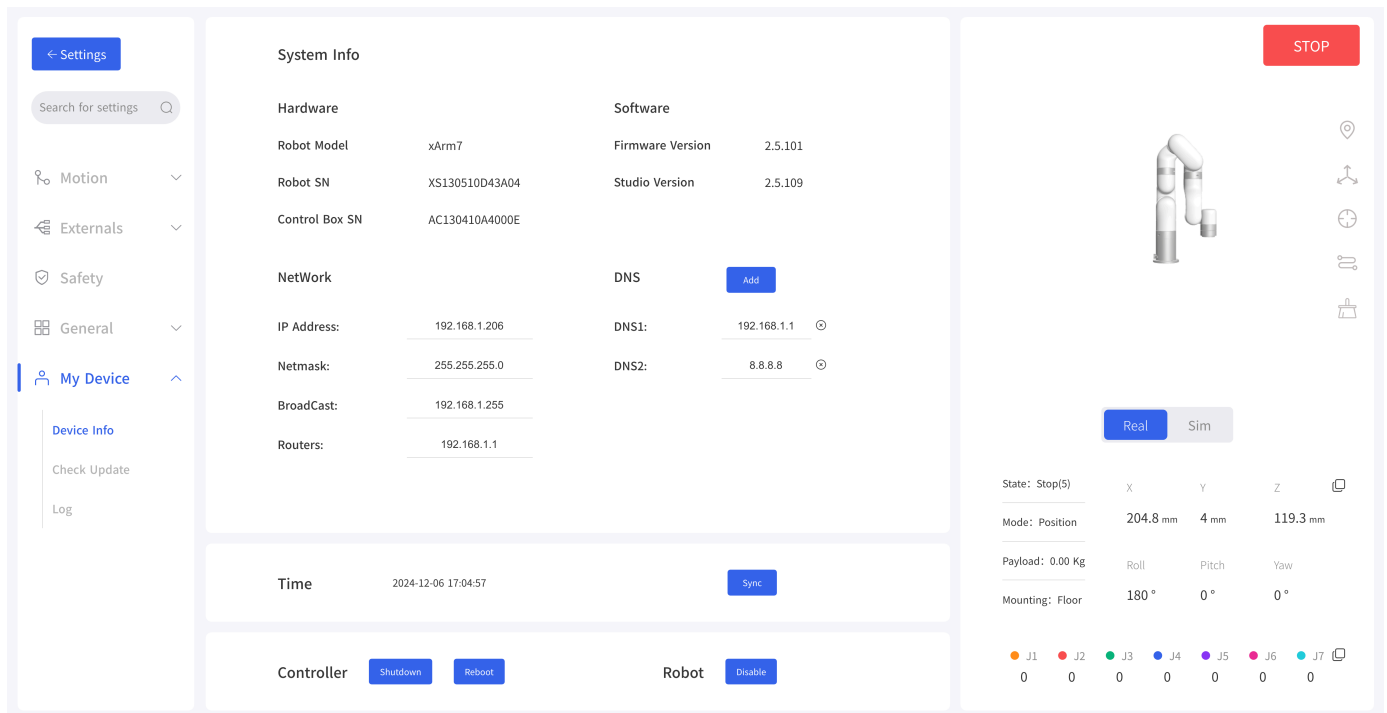
Factory Reset: The robotic arm will restore the factory settings.

Note:

- 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.
- 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.
- 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.

7.5 My Device

7.5.1 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.

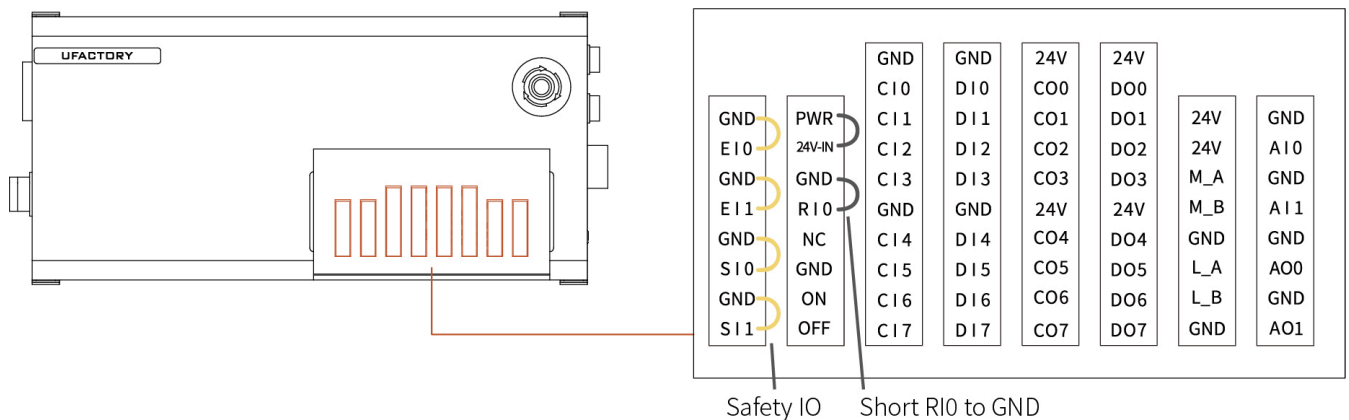
Network Settings: Display the IP address of the robotic arm, subnet mask, broadcast address, and default gateway. The DNS address can be modified and added.

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.

How to reset IP?

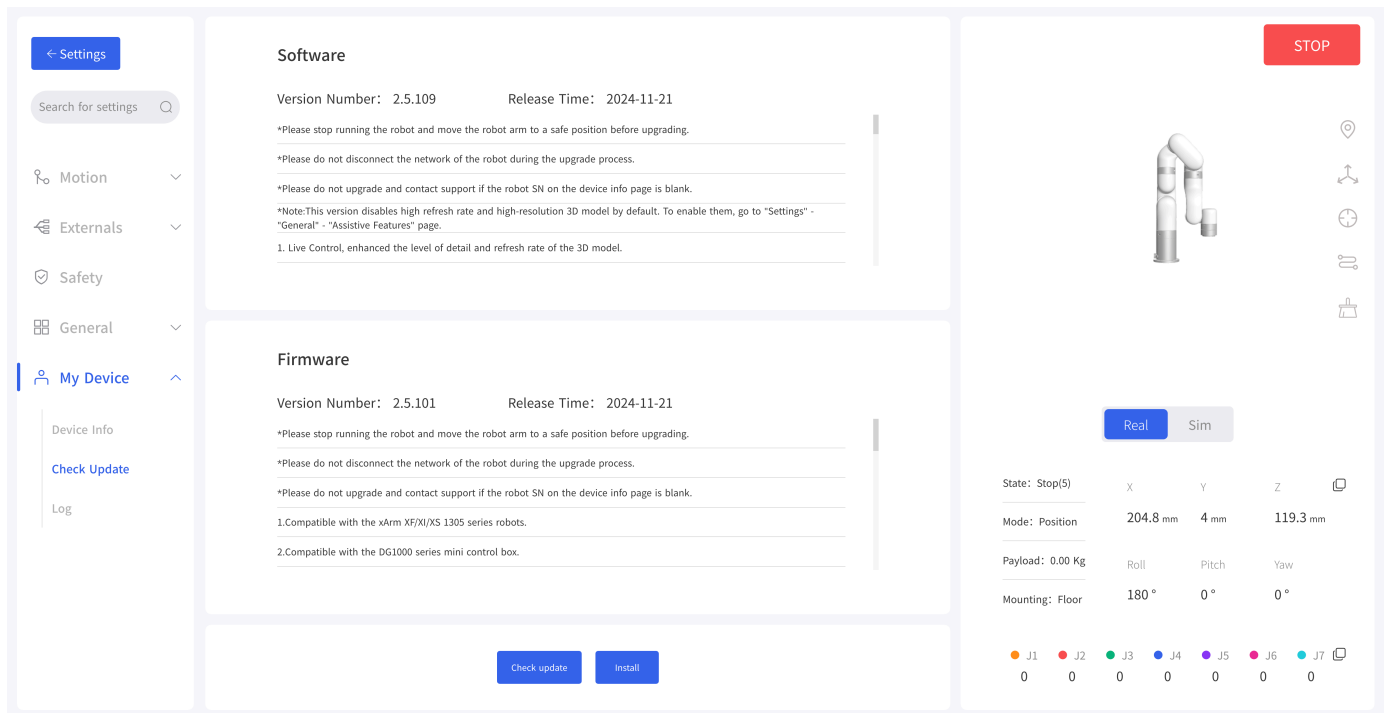
Steps:

- 1. Press the emergency stop button and turn off the power of the control box.
- 2. Connect RI0 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 RI0 and GND and wait for the control box to start up (60 seconds).
- 5.Enter 192.168.1.111:18333 in the browser to access UFACTORY Studio.

7.5.2 Check Update



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.

7.5.3 Log

← Settings

Search for settings

Motion

Externals

Safety

General

My Device

Device Info

Check Update

Log

Servo

Controller


End-effector

Linear Motor

Time	Error_Code	Error_Content
2024-11-29 16:42:40	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:39:19	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:36:07	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:32:36	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:32:07	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:31:16	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:30:35	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:30:01	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:29:28	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:28:48	C2	The Emergency Stop Button is pushed in to stop.
2024-11-29 16:26:48	C1	The Emergency Stop Button on the Control Box is pushed in to stop.

Download

STOP



Real Sim

State: Stop(5)

X

Y

Z

Mode: Position

204.8 mm

4 mm

119.3 mm

Payload: 0.00 Kg

Roll

Pitch

Yaw

Mounting: Floor

180 °

0 °

0 °

J1

J2

J3

J4

J5

J6

J7

0

0

0

0

0

0

0

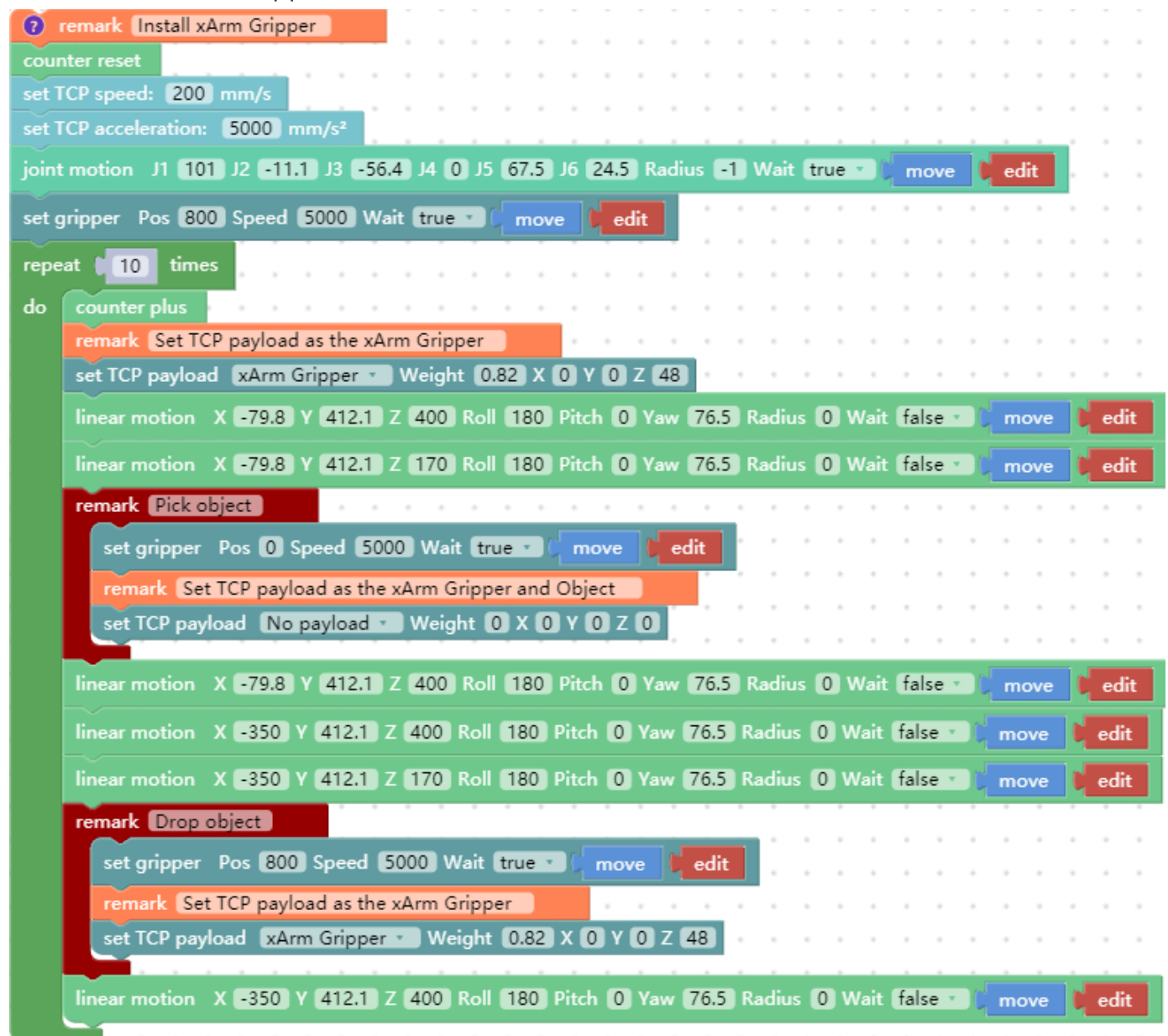
The error log of the control box, servo error log and end effector error log can be checked. Click the 'Download' button to download the error log.

8. Blockly Typical Examples

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

8.1 xArm Gripper

[UF] - 1007_xArm_Gripper.

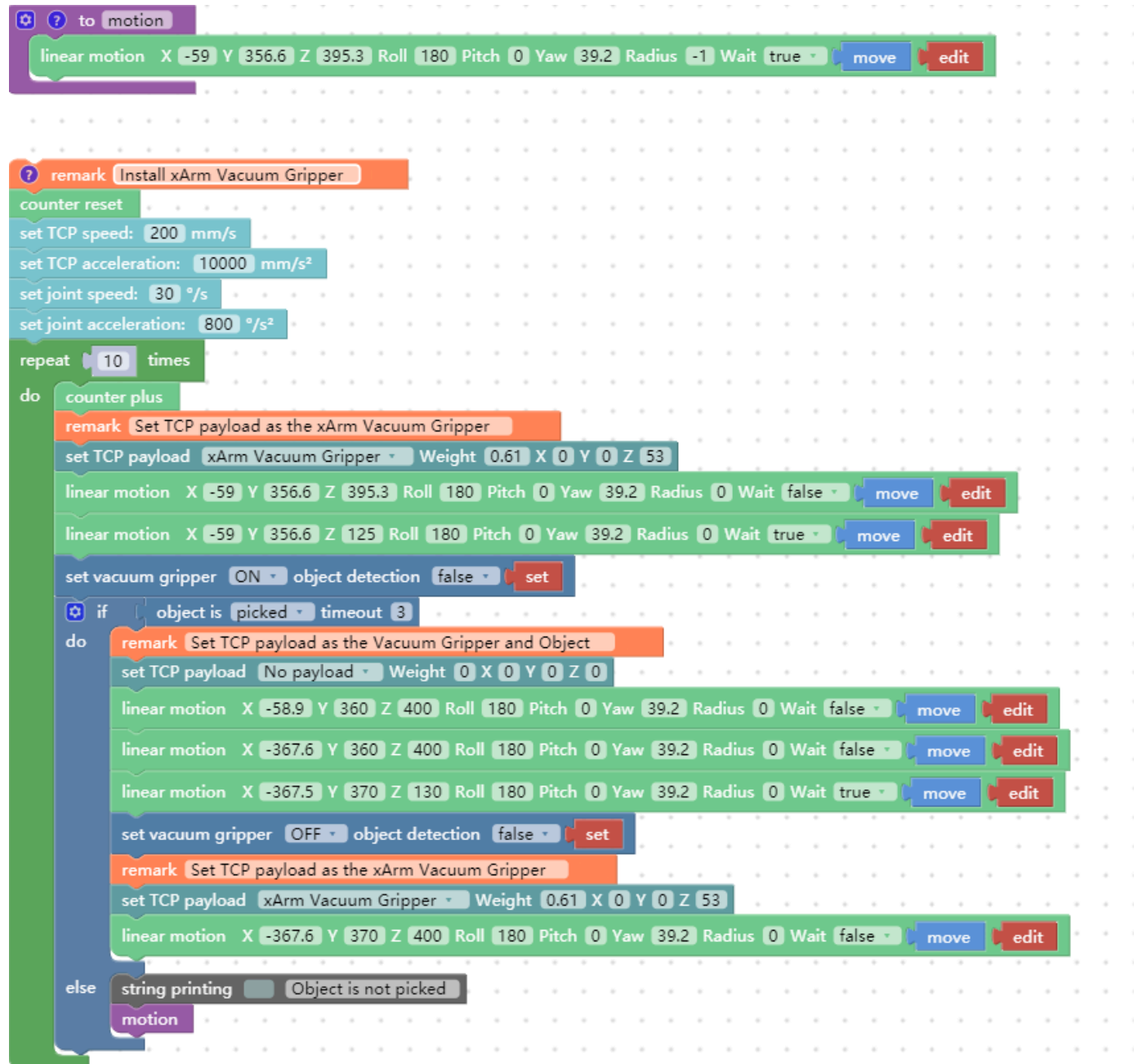


The role of this program: execute this program to control the gripper to grab the target object at the specified position, and then place the target object at the target position.

Set TCP payload: Dynamic change tcp payload according to the real application.

8.2 xArm Vacuum Gripper

[UF] - 1008_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.

Block:

- 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.

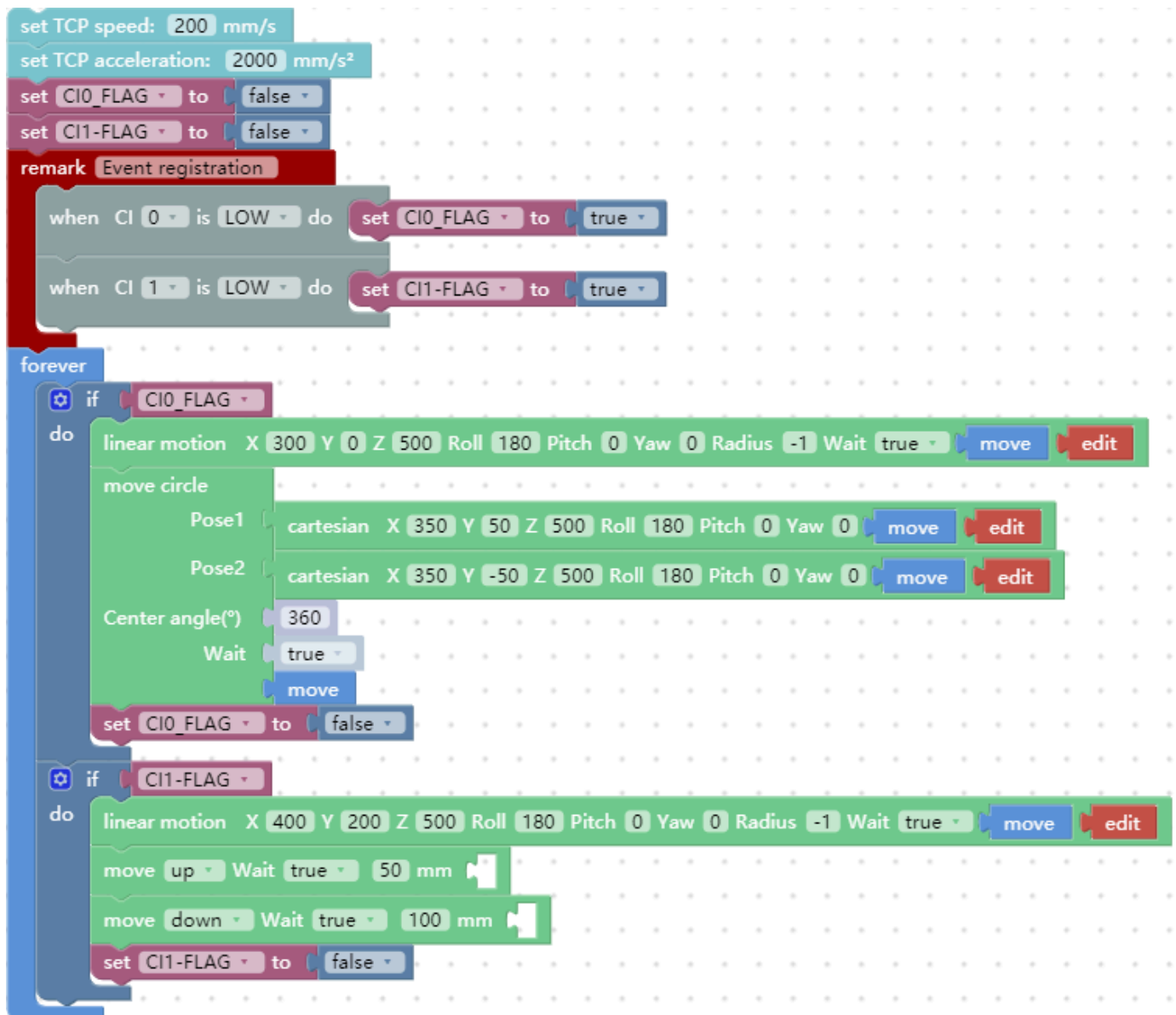
object detection = false: do not detect whether the object is sucked.

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.

8.3 Digital_IO

[UF] - 1010_Digital_IO.



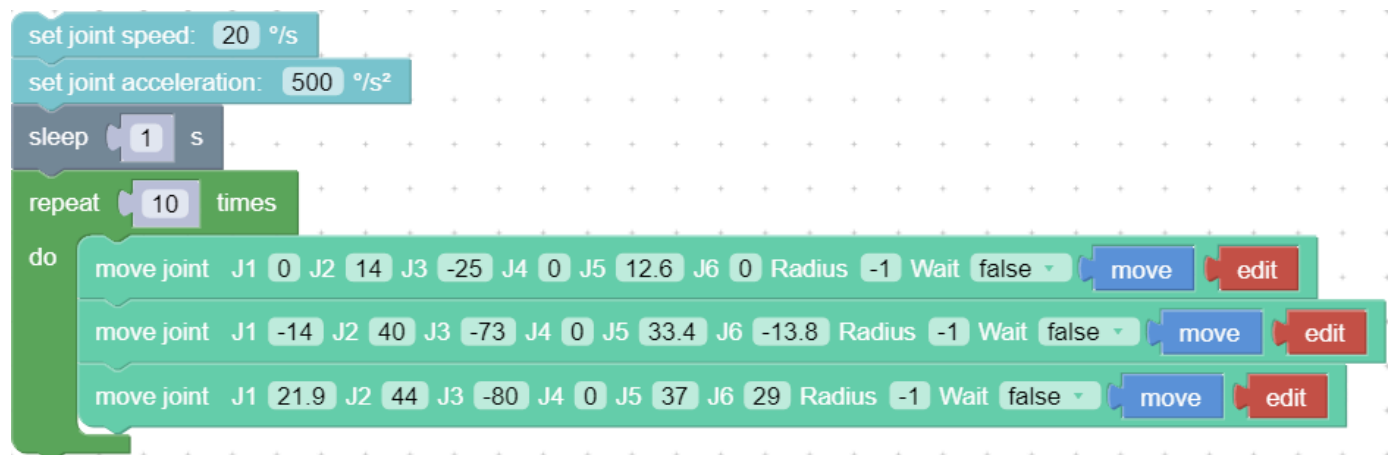
The role of this program: If you need to use digital IO to control the motion of the robotic arm, you can trigger the digital IO(**signal change**) to perform the corresponding motion.

9. Motion Characteristics

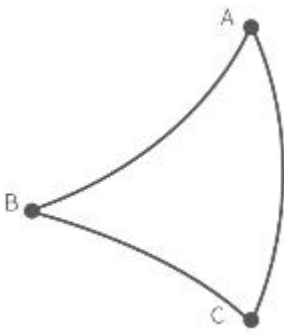
9.1 Motion of Robotic Arm

9.1.1 Joint Motion

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



- 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:

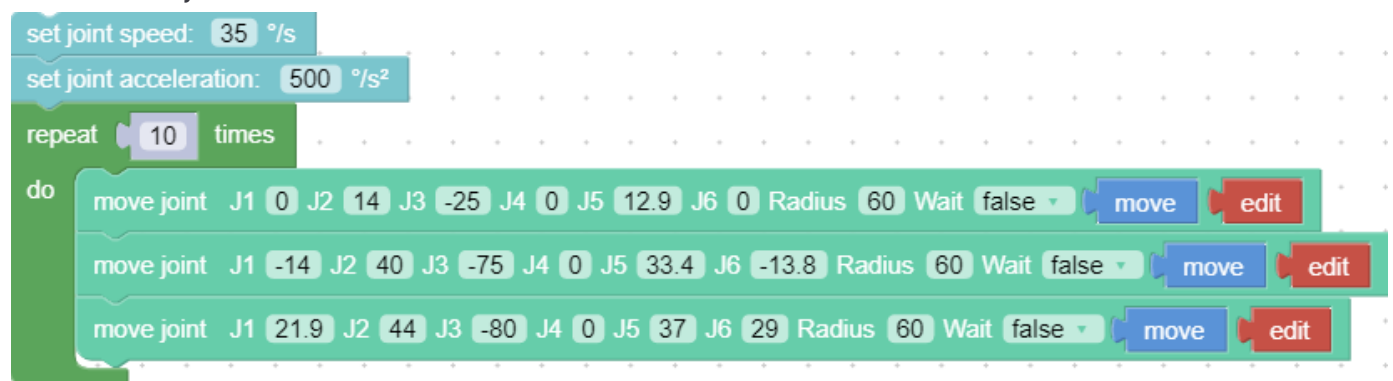
```
python
arm.set_servo_angle(angle=[0.0, 7.0, -71.2, 0.0, 0.0, 0.0], speed=8, mvacc=1145, wait=
arm.set_servo_angle(angle=[0.0, 7.0, -51.2, 0.0, 0.0, 0.0], speed=8, mvacc=1145, wait=
arm.set_servo_angle(angle=[0.0, 7.0, -91.2, 0.0, 0.0, 0.0], speed=8, mvacc=1145, wait=
```

set_servo_angle:

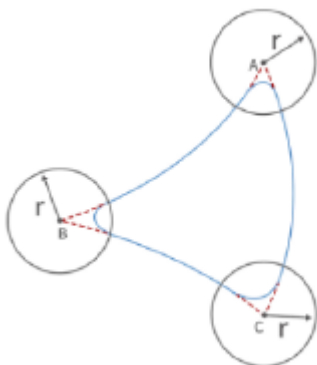
- `servo_id` : joint ID, 1-7, None or 8 means all joints.
- `angle` : Joint angle or list of joint angles (the unit of the default joint angle is `is_radian = False`, degrees ($^{\circ}$)).
- `speed` : joint speed (the default unit is $^{\circ}/s$).
- `mvacc` : joint acceleration (default unit is $^{\circ}/s^2$).
- `is_radian` : roll/pitch/yaw Whether it is measured in radian (default `is_radian = False`).
- `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.

How to plan 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.

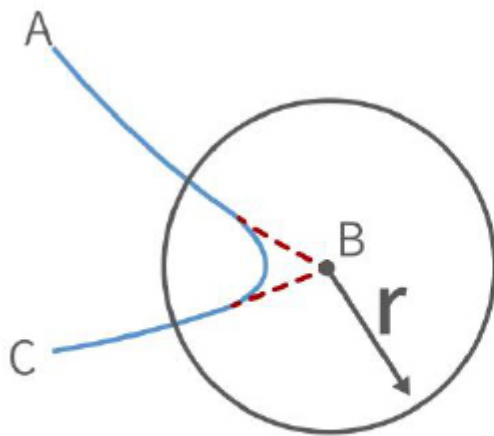


The motion trajectory of the robotic arm in the above example is as follows:

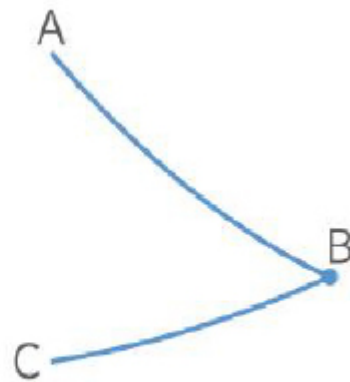


Key Parameter Radius=60 `Radius =60` in the `move joint` command refers to setting the radius of the transition arc $R = 60\text{mm}$, which is used to achieve a smooth transition of the arc in a joint motion. The parameters of Radius can be set as $\text{Radius} > 0$, $\text{Radius} = 0$, $\text{Radius} = -1$, 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 $D \sim AB \sim$ and $D \sim BC \sim$.
- (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.



radius>0



radius<=0

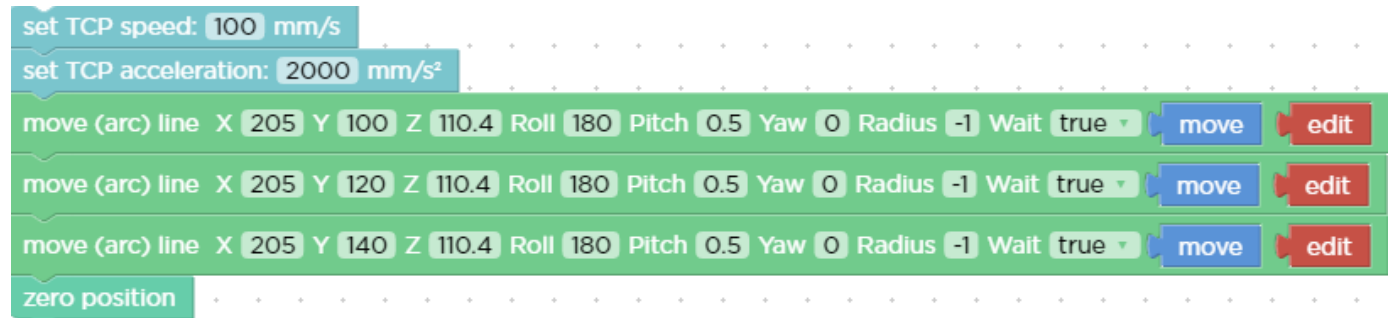
Conclusion: If you need to plan for speed continuous joint motion, make sure **wait = false**, **radius ≥ 0** to buffer the commands to be blended.

9.1.2 Linear Motion

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.



Python Example:

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

python

```
arm.set_position(x=205.0, y=100.0, z=110.4, roll=180.0, pitch=0.5, yaw=0.0, speed=100,
arm.set_position(x=205.0, y=120.0, z=110.4, roll=180.0, pitch=0.5, yaw=0.0, speed=100,
arm.set_position(x=205.0, y=140.0, z=110.4, roll=180.0, pitch=0.5, yaw=0.0, speed=100,
```

If you want to plan speed continuous linear motion, make sure **wait=false**, **radius≥0**.

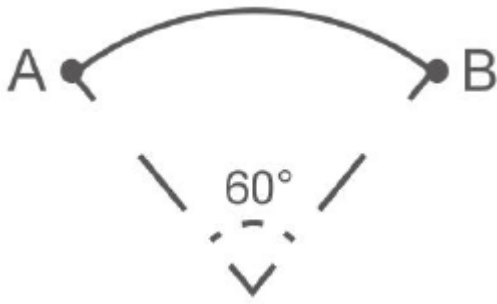
Note: If it is xArm5, roll and pitch must be set to roll,pitch = [±180,0]°.

9.1.3 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:

- 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° , the motion trajectory of the robotic arm is:



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

The screenshot shows a programming sequence in UFACTORY Studio:

- remark** Draw Circle
- set TCP acceleration:** 2000 mm/s²
- set TCP speed:** 300 mm/s
- repeat** 10 times
- do**
 - move joint** J1 0 J2 -45 J3 0 J4 0 J5 -45 J6 0 Wait true
 - move (arc) line** X 300 Y 0 Z 400 Roll 0 Pitch -90 Yaw 180 Radius -1 Wait true
 - move circle**
 - Pose1** cartesian X 350 Y 50 Z 400 Roll 180 Pitch -90 Yaw 0
 - Pose2** cartesian X 350 Y -50 Z 400 Roll 180 Pitch -90 Yaw 0
 - Center angle($^{\circ}$)** 3600
 - Wait** true

- 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;
- 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 ($^{\circ}$): 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;

Example explanation:

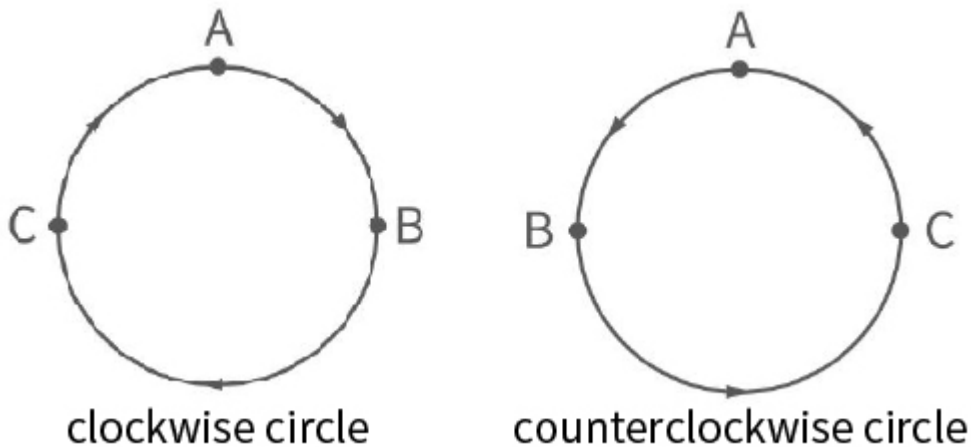
In this example, the central angle is set to 3600° , which means that the robotic arm can draw ten circles at a time, and the robotic arm still stays at the starting point after drawing a circle.

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:

```
python
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=100)
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],
```

move_circle:

- `pose1` : Cartesian coordinates, [x,y,z,roll,pitch,yaw].
- `pose2` : Cartesian coordinates, [x,y,z,roll,pitch,yaw].
- `percent` : Percentage of arc moved.
- `is_radian` : roll/pitch/yaw Whether it is measured in radian (default is_radian = False).
- `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.

9.2 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 [± 180 degrees, 0 degrees], otherwise the trajectory is likely to have no solution. When firmware version > v2.5.0, we provided **Use approximate solutions** option to remove this restriction.

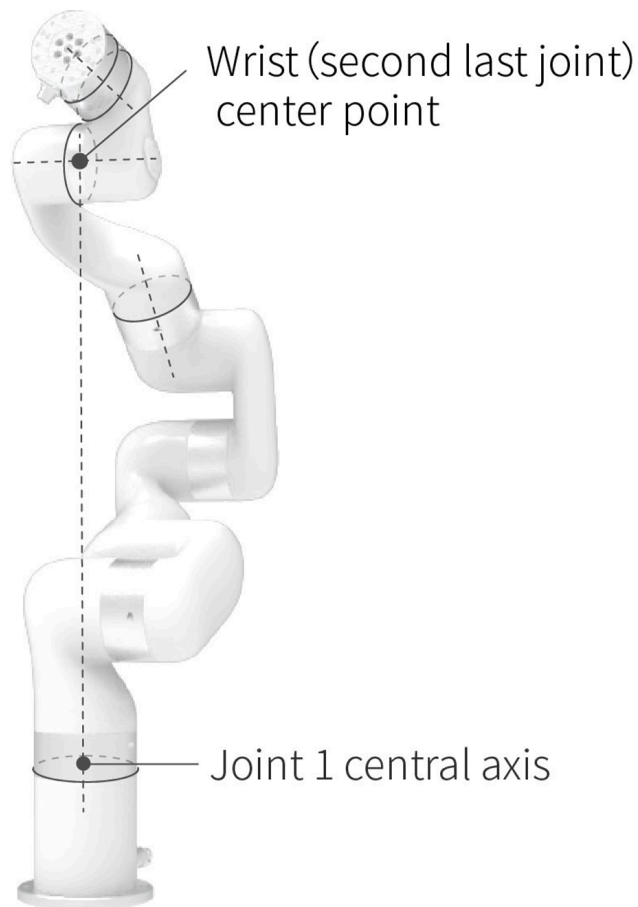
Joint space

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 $-(J2 \text{ angle} + J3 \text{ angle})$.

9.3 Singularity

What is singularity?

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.



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.

Processing method for singularity point

Case 1: Singularity encountered during robot teaching

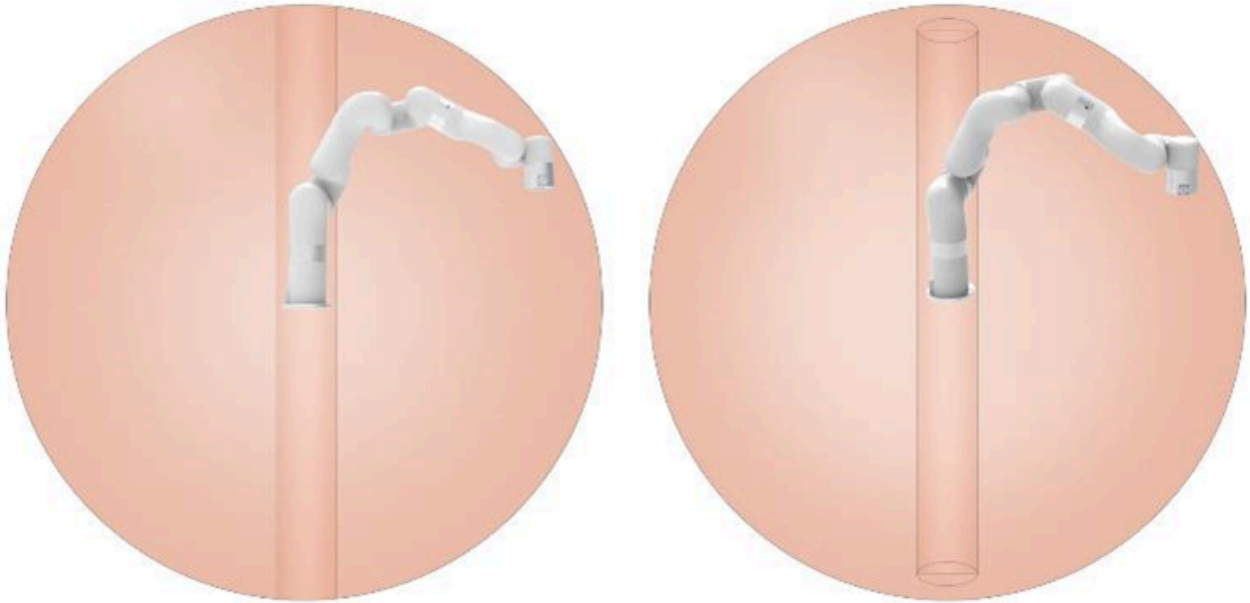
- Switch the robot coordinate system to a joint coordinate system, and pass the singularity point through joint motion.

Case 2: Singularities encountered while the program is running

- Enable **bypassing singularity** option
 - 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.



10. Robotic Arm Motion Mode and State

The controller provides 7 motion mode and 6 state, corresponding to python SDK `set_mode`, `set_state`.

10.1 Robotic Arm Mode

Mode 0: Position Control Mode

The control box enters this mode 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.

Python Example: [set_servo_angle](#)

Linear Motion

To achieve linear motion between Cartesian coordinates (unit: mm), the speed between each instruction is discontinuous.

Python Example: [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.

Python Example: [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.

Python Example: [move_circle](#)

Mode 1: Servo(ServoJ) Mode

Servo Joint Motion Move to the given joint position with the fastest speed (180°/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.

Python Example: [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 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 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.

Python Example: [servo_cartesian](#), [servo_cartesian_aa](#)

Mode 2: Manual 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.

Python Example:

```
arm.set_mode(2)
arm.set_state(0)
```

python

Mode 4: Joint Velocity Control Mode

Joint motion with specified velocity for each joint (unit: rad/s).

Python Example: [vc_set_joint_velocity](#)

Mode 5: Cartesian Velocity Mode

Linear motion of TCP with specified velocity in mm/s (position) and rad/s (orientation in axis-angular_velocity).

Python Example: [vc_set_cartesian_velocity](#)

Mode 6: Joint online 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.

Python Example: [set_servo_angle](#)

Mode 7: Cartesian online 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.

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.

Python Example: [set_position](#)

10.2 Robotic Arm State

State 0

Set: configure the robot to be STANDBY state in corresponding mode, and clear the error code as well. Note: after this setting, the feedback state will switch to 2(READY) automatically.

State 1

Feedback. The robot is in motion.

State 2

Feedback. The robot is ready to receive and execute commands.

State 3

Set and Feedback.

- set: set the robot to a PAUSED state when executing motion commands, the motion can be resumed by setting state 0.
- feedback: the robot is in PAUSE state.

State 4

Set and Feedback.

- set: set the robot to STOP state, it will terminate any execution immediately and will not receive or execute any new command until the state is set back to STANDBY.
- feedback: the robot is in STOP state, can't receive and execute any command, and will automatically switch to this state when any error occurs.

State 5

Feedback. MODE_CHANGED state, will automatically switch to this state if some critical configurations (mode, payload, TCP offset, collision sensitivity, etc) have been changed,

and cannot receive and execute any command until set state 0.

State 6

Set and Feedback. Perform a decelerated stop immediately.

11. Technical Specifications

	xArm5	xArm6	xArm7	850	Lite6
Max speed	180°/s	180°/s	180°/s	180°/s	180°/s
Joint1	±360°	±360°	±360°	±360°	±360°
Joint2	(-118,120)	(-118,120)	(-118,120)	±132°	±150°
Joint3	(-255,11)	(-255,11)	±360°	(-242,3.5)	(-3.5,300)
Joint4	(-97,180)	±360°	(-11,225)	±360°	±360°
Joint5	±360°	(-97,180)	±360°	±124°	±124°
Joint6	-	±360°	(-97,180)	±360°	±360°
Joint7	-	-	±360°	-	-

	TCP Motion	TCP Motion	Joint Motion	Joint Motion
	xArm/850	Lite6	xArm/850	Lite6
speed	1000mm/s	500mm/s	180°/s	180°/s
acc	50000mm/s ²	50000mm/s ²	1146°/s ²	1146°/s ²
jerk	100000mm/s ³	100000mm/s ³	28647°/s ³	28647°/s ³

12. Error Handling

12.1 Control Box Error Code and Handling

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; 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-plan the motion of the robotic arm according to the reported error message.

Controller 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.

Controller Error Code	Error Handling
C23	<p>Joints Angle Exceed Limit</p> <p>Please go to the "Live Control" page and press the "Initial POSITION" button to let the robot back to the Initial position.</p>
C24	<p>Speed Exceeds Limit</p> <p>Please check if the xArm is at singularity point, or reduce the speed and acceleration values.</p>
C25	<p>Planning Error</p> <p>Please re-plan the path or reduce the speed.</p>
C26	<p>Linux RT Error</p> <p>Please contact technical support.</p>
C27	<p>Command Reply Error</p> <p>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.</p>
C29	<p>Other Errors</p> <p>Please contact technical support.</p>
C30	<p>Feedback Speed Exceeds Limit</p> <p>Please contact technical support.</p>
C31	<p>Abnormal current in the robotic arm</p> <ol style="list-style-type: none"> 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	<p>Three-point Drawing Circle Calculation Error</p> <p>please reset the arc command.</p>
C33	<p>Controller IO Error</p>

Controller Error Code	Error Handling
	If the error occurs repeatedly, please contact technical support.
C34	<p>Recording Timeout</p> <p>The track recording duration exceeds the maximum duration limit of 5 minutes. It is recommended to re-record.</p>
C35	<p>Safety Boundary Limit</p> <p>The xArm reaches the safety boundary. Please let the xArm work within the safety boundary.</p>
C36	<p>The number of delay commands exceeds the limit</p> <ol style="list-style-type: none"> 1. Please check whether there are too many position detection or IO delay commands. 2. Increase the tolerance of the position detection command.
C37	<p>Abnormal Motion in Manual Mode</p> <p>Please check whether the TCP payload setting of the robotic arm and the installation method of the robotic arm match the actual settings.</p>
C38	<p>Abnormal Joint Angle</p> <p>Please stop the xArm by pressing the Emergency Stop Button on the Control Box.</p>
C39	<p>Control Box Power Board Master and Slave IC Communication Error</p> <p>Please contact technical support.</p>
C50	<p>Six-axis Force Torque Sensor Error</p> <p>Please check the sensor error code, locate the problem, and power on again. If it cannot be resolved, please contact technical support.</p>
C51	<p>Six-axis Force Torque Sensor Mode Setting Error</p> <p>Please make sure that the robotic arm is not in Manual Mode, check whether the given value of this command is 0/1/2</p>
C52	<p>Six-axis Force Torque Sensor Zero Setting Error</p> <p>Please check the sensor communication wiring and whether the power is normal.</p>
C53	<p>Six-axis Force Torque Sensor Overload</p> <p>Please reduce the payload or applied external force.</p>
C110	Robot Arm Base Board Communication Error.

Controller Error Code	Error Handling
	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.

12.2 Joint Servo Error Code and Handling

Error processing method: Re-power on, the steps are as follows:

- Release the emergency stop button on the control box.
- Enable the robotic arm.

Joint Servo 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
S12	Joint Overspeed Please restart the xArm with the Emergency Stop Button on the xArm
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

Joint Servo Error Code	Error Handling
	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	Joint N Negative Overrun 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.
S28	Joint Commands Error The xArm is not enabled, please click Enable Robot.

Joint Servo Error Code	Error Handling
S33	Drive Overloaded Please make sure the payload is within the rated load.
S34	Motor Overload Please make sure the payload is within the rated load.
S35	Motor Type Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S36	Driver Type Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S39	Joint Overvoltage Please reduce the acceleration value in the Motion Settings.
S40	Joint Undervoltage Please reduce the acceleration value in the Motion Settings. Please check if the control box emergency stop switch is released.
S49	EEPROM Read and Write Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S52	Initialization of Motor Angle Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.

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.

12.3 Python SDK Error Code and 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)

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

Python SDK 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.

Python SDK Error Code	Error Handling
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.

13. Software and Firmware Update Method

13.1 Online Update

When PC has network connection.

Method 1: UFACTORY Studio.

Enter Settings - My Device - Check 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.

Software

Version Number: 2.5.109

Release Time: 2024-11-21

*Please stop running the robot and move the robot arm to a safe position before upgrading.

*Please do not disconnect the network of the robot during the upgrade process.

*Please do not upgrade and contact support if the robot SN on the device info page is blank.

*Note: This version disables high refresh rate and high-resolution 3D model by default. To enable them, go to "Settings" - "General" - "Assistive Features" page.

1. Live Control, enhanced the level of detail and refresh rate of the 3D model.

Firmware

Version Number: 2.5.100

Release Time: 2024-11-21

*Please stop running the robot and move the robot arm to a safe position before upgrading.

*Please do not disconnect the network of the robot during the upgrade process.

*Please do not upgrade and contact support if the robot SN on the device info page is blank.

1. Compatible with the xArm XF/XI/XS 1305 series robots.

2. Compatible with the DG1000 series mini control box.

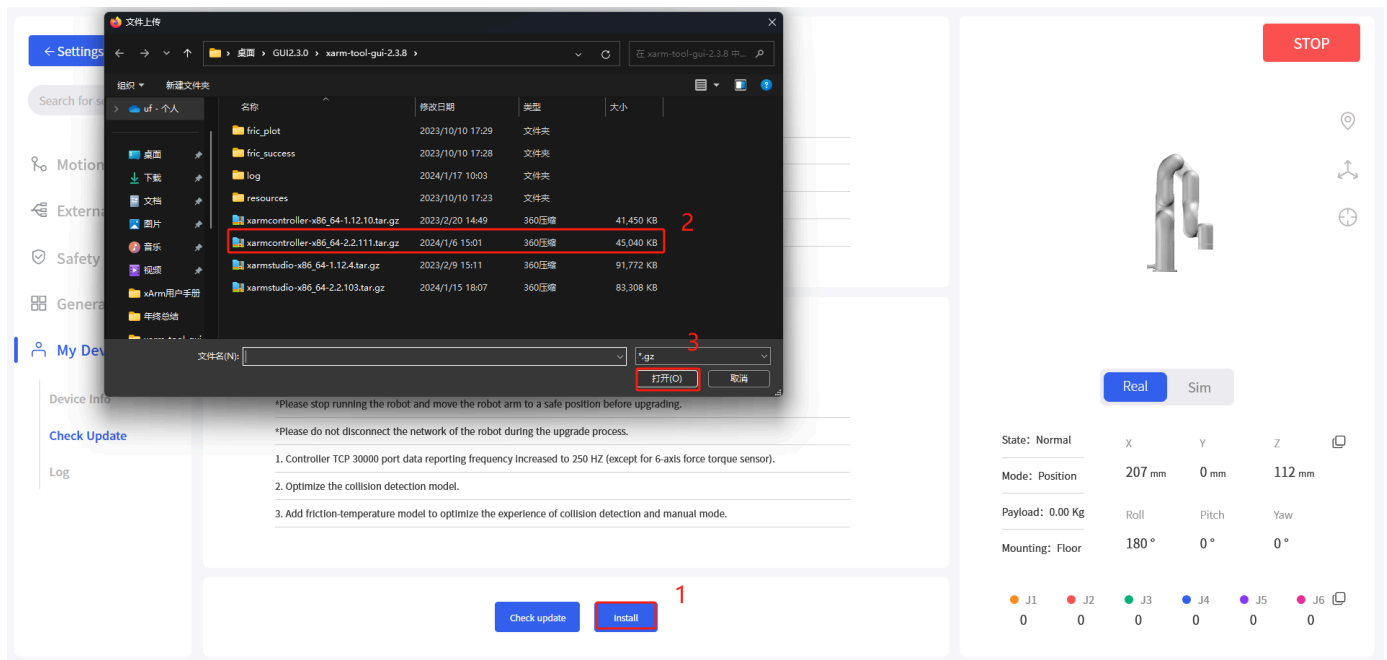


13.2 Offline Update

When PC has no network connection.

Method 1: UFACTORY Studio.

Enter Settings - My Device - Check Update, click "Install" to load the offline package downloaded in advance, reboot the system, it will take 2-3 minutes.



Method 2: xArm-Tool-GUI.

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

- Click Firmware - Offline Installation, select the offline firmware package you downloaded in advance, click 'Install', the interface pop-up prompt "Install firmware successfully".
- Click xArmStudio - Offline Installation, select the offline package downloaded in advance, click 'Install', the interface will pop up a prompt "Install Studio successfully".
- 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).