T1 Instruction Manual-V1.1

中文版本：T1说明书-V1.0

Product Overview

Booster T1 is a humanoid robot development platform designed for developers and other users, featuring lightweight, flexible, and durable.

Product Composition

The T1 robot consists of head, torso, arms, and legs, with a total of 23 DoFs, allowing for flexible movement and posture control.

- The head has 2 DoFs, including Yaw Joint and Pitch Joint. It contains a depth camera, microphone array, and speaker.
- Each arm has 4 DoFs, including Shoulder Pitch Joint, Shoulder Roll Joint, Shoulder Yaw Joint, and Elbow Joint.
- The waist has 1 DoF, which is the Waist Yaw Joint.
- Each leg has 6 DoFs, including Hip Pitch Joint, Hip Roll Joint, Hip Yaw Joint, Knee Joint, and Ankle Up and Down Joint.
- Control board and battery are installed in the torso.

Product Functions

1. Omnidirectional Walking
    1. Supports forward, backward, and lateral walking.
    2. Supports rotation and complex walking.
2. Disturbance Resistance while Walking
    1. Can walk on uneven surfaces.
    2. Can withstand certain impact disturbances while walking.
3. Predefined Actions
    1. Waving.
    2. Shaking hand.
    3. Dancing.
    4. Fall recovery.
4. Safety Protection
    1. Automatically enters damping mode in uncontrolled states to prevent damage.
    2. Hard emergency stop.

Product Specifications

Type	Specification Parameters	Description
Basic Parameters	Height (when standing upright)	1.18m
	Weight	About 30 kg
DoFs	Total DoFs	23
	Single Leg DoFs	6
	Single Arm DoFs	4（expandable）
	Waist DoFs	1
	Head DoFs	2
Operational Parameters	Walking Speed	0.5m/s
	Turning Speed	1.5rad/s
Battery Parameters	Battery Life	1.5h
	Charging Time	≤2h
	Cycle Life	≥500 times
Computing Platform	Basic	AGXOrin*1
	Standard	i7 *1
		AGXOrin*1
Sensors	Camera	Depth Camera
	Microphone	Microphone Array
	Speaker	1
Safety Fuction	Robot Emergency Stop	1
	Auditory Alerts	Low Battery Alert, Joint Overheat Alert

Communication Methods	Wireless Connection	WIFI 6
	Bluetooth	Bluetooth 5.2
	Certification Standards	CE/FCC
Noise	Walking Noise	≤70dB
Environmental Adaptability	Environmental Adaptability	-10℃~45℃
	Operating Humidity	5%~90%, without condensation

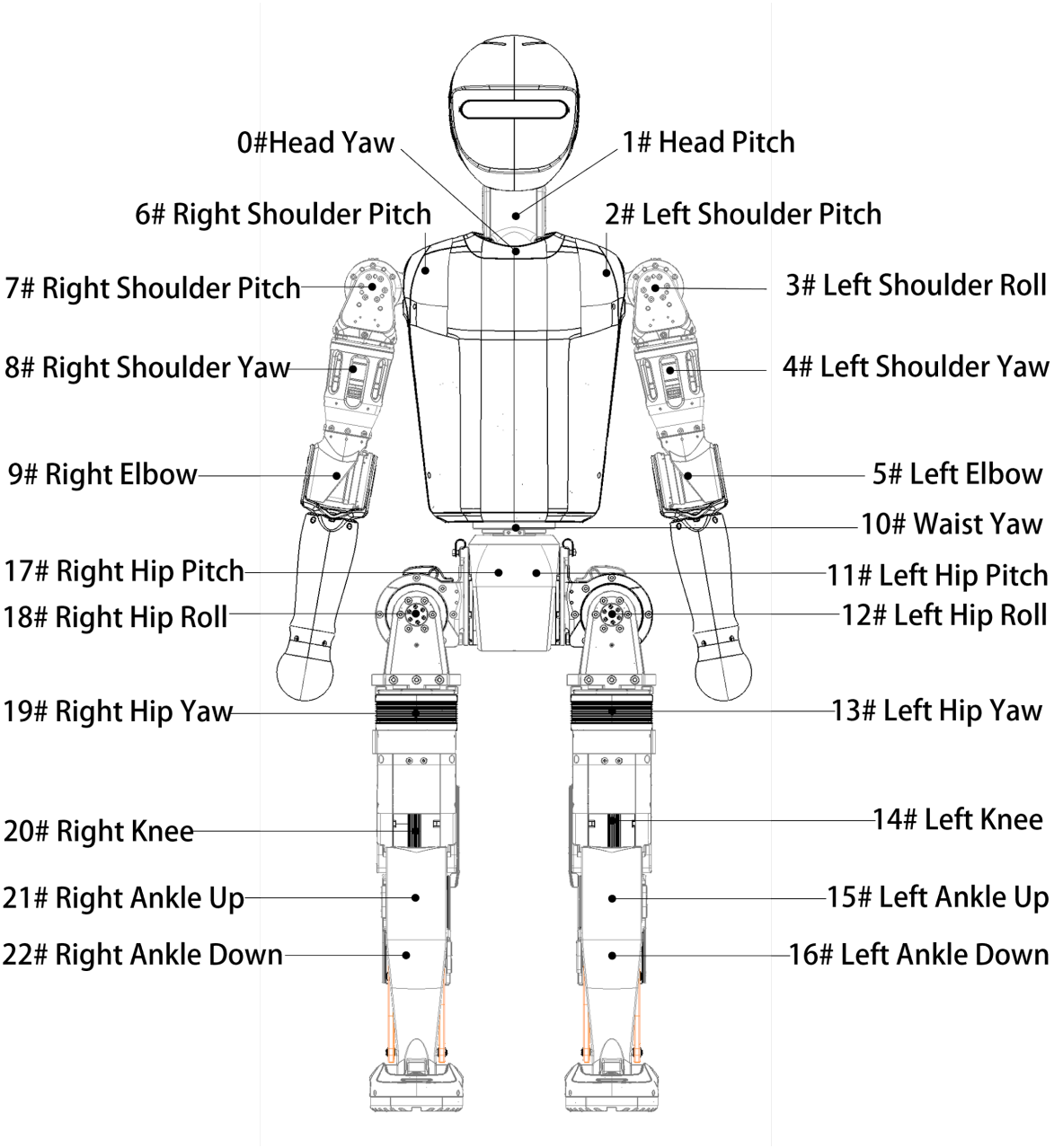
Main parts

Joint motors.

Joint ID and limits.

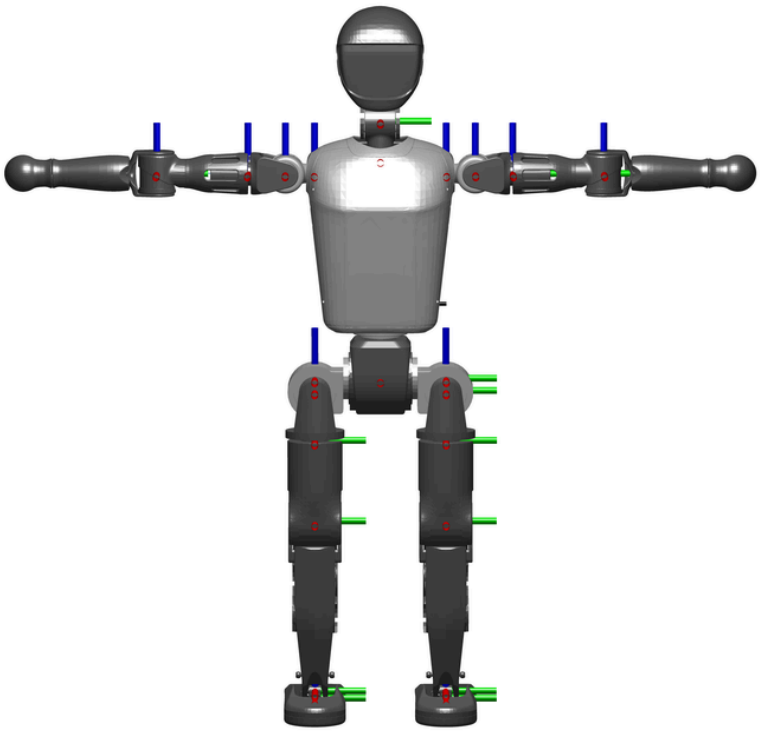
Joint ID	Joint Name	Limit (Degrees)	
		Max	Min
0	Head Yaw Joint	58	-58
1	Head Pitch Joint	47	-18
2	Left Shoulder Pitch Joint	68	-188
3	Left Shoulder Roll Joint	88	-94
4	Left Shoulder Yaw Joint	128	-128
5	Left Elbow Joint	2	-120
6	Right Shoulder Pitch Joint	68	-188
7	Right Shoulder Roll Joint	94	-88
8	Right Shoulder Yaw Joint	128	-128
9	Right Elbow Joint	120	-2

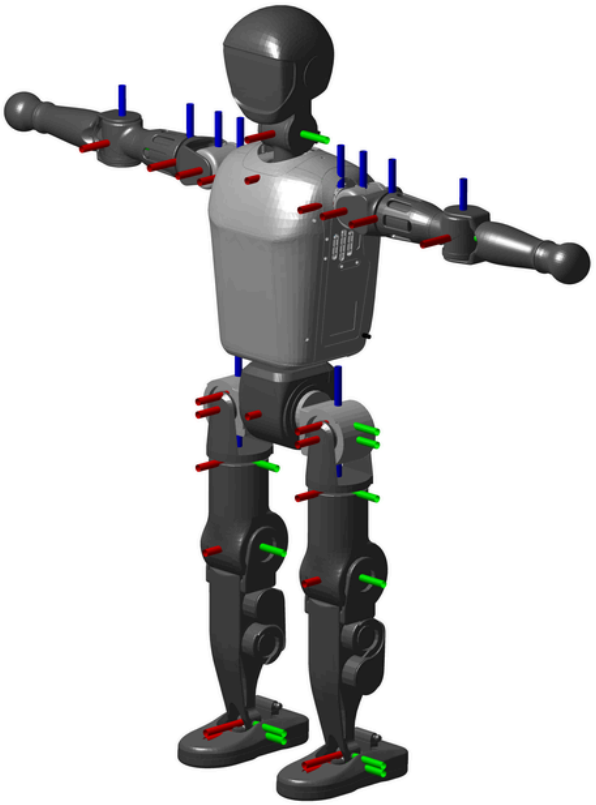
10	Waist Yaw Joint	58	-58
11	Left Hip Pitch Joint	118	-118
12	Left Hip Roll Joint	88	-21
13	Left Hip Yaw Joint	58	-58
14	Left Knee Joint	123	0
15	Left Ankle Up Joint	49	-23
16	Left Ankle Down Joint	45	-24
17	Right Hip Pitch Joint	118	-118
18	Right Hip Roll Joint	21	-88
19	Right Hip Yaw Joint	58	-58
20	Right Knee Joint	123	0
21	Right Ankle Up Joint	49	-23
22	Right Ankle Down Joint	45	-24


0#Head Yaw
1# Head Pitch
6# Right Shoulder Pitch
2# Left Shoulder Pitch
7# Right Shoulder Pitch
3# Left Shoulder Roll
8# Right Shoulder Yaw
4# Left Shoulder Yaw
9# Right Elbow
5# Left Elbow
10# Waist Yaw
17# Right Hip Pitch
11# Left Hip Pitch
18# Right Hip Roll
12# Left Hip Roll
19# Right Hip Yaw
13# Left Hip Yaw
20# Right Knee
14# Left Knee
21# Right Ankle Up
15# Left Ankle Up
22# Right Ankle Down
16# Left Ankle Down

Coordinate System

The joint coordinate system with all joints at zero position is shown in the diagram below.





Controller

Version	Basic	Standard	
	Basic board	**Motion control board**	**Perception borad**
Processor	Jetson AGX Orin 32GB	Express-i7	Jetson AGX Orin 32GB
Computing performance	8-core Cortex-A78AE CPU@2GHz Tensor Cores GPU@930MHz AI performance: 200 TOPS	6P cores + 8E cores P core maximum frequency: 4.80 GHz E core maximum frequency: 3.90 GHz	8-core Cortex-A78AE CPU@2GHz Tensor Cores GPU@930MHz AI performance: 200 TOPS
Memory	32GB	8GB	32GB
Storage	512GB	512GB	512GB
Wired Network	1000M*2	1000M*2	1000M*2
Wireless Network	WIFI6*1	WIFI6*1	WIFI6*1
Audio	Microphone, speaker		Microphone, speaker

💡 For the basic version, both the motion control program and the perception program are deployed on the basic board.

💡 For the standard version, the motion control program is deployed on the motion control board, and the perception program is deployed on the perception board.

Operation Manual

T1 is packed with an out-of-the box motion control program. Follow the instructions below to control T1 remotely.

WARNINGS ❗

1. T1 must first enter PREP mode, and then **put to a stable standing position on the ground**, before switching to WALK mode.
2. **DO NOT lift T1 while under WALK mode.**
3. **Make sure to clear any obstacles on the ground and avoid human injuries while operating T1.**
4. **DO NOT touch any parts of T1 while under WALK mode, except for the handle.**
5. **Make sure to remove ALL zero parts before restarting T1.**

MODES

- T1 operates under **Modes**.
- T1 can perform different **Actions** under different **Modes**.
- **Modes** can switch to one another but **with constraints**. For example, DAMP Mode can only switch to WALK mode by first entering PREP Mode.

DAMP Mode

- Power is on and motion control is working.
- All joints are in a dampening state, i.e, joints will resist position change, but will neither try to hold its position nor actively change its position.
- T1 is NOT able to stand under DAMP mode, therefore support is needed.
- DAMP mode is a safe mode, entering DAMP mode can protect T1 and its operator.
- DAMP mode can be switched to PREP mode, but not directly to WALK mode.

PREP Mode

- Power is on and motion control is working.
- T1 assumes a standing posture and holds it. Joints will strongly resist position change, and try to go back to the standing posture if forcefully moved.
- T1 under PREP mode can stand on its own on the ground. Under PREP mode T1 can be placed to stand on the ground. However, it will NOT try to balance itself.
- T1 can swtich to ALL other modes under PREP mode, including DAMP and WALK.

WALK Mode

- Power is on and motion control is working.
- Under WALK mode, T1 can perform various predefined actions, including omni-walk, rotating, stepping, standing-still and moving head.
- Compared to PREP mode, WALK mode is more resilient, and will try to recover balance if pushed.
- T1 under WALK mode can switch to all other modes, including DAMP and PREP.
- **NOTICE: Make sure T1 is under PREP mode and is already standing firmly on the ground, before switch to WALK mode.**

CUSTOM Mode

- Power is on and motion control is working.
- **T1 gives up control on all joints to developer, who controls T1 through its SDK. Use caution under CUSTOM mode to avoid damage to T1.**
- Only PREP and DAMP modes can switch to CUSTOM mode.
- CUSTOM mode can only switch to PREP or DAMP mode.

💡 While developing new tricks on T1, it is recommended to use a Hoist at all times under CUSTOM mode.

PROTECT Mode

- PROTECT mode will automatically kick in on errors (i.e, exceeding joint limit or falling).
- Joints under PROTECT Mode behave the same as DAMP mode.
- You can try to reenter DAMP mode under PROTECT mode. (Soft restart)
- PROTECT mode is a safe mode, entering PROTECT mode can protect T1 and its operator.

Powering on

1. Place T1 on its rest.
2. Install the battery pack. Slide the pack into the battery socket, with lights facing outwards.
3. Place T1's hands and legs in a natural posture.
4. Press Power button about 3s（release it after light turns on；press more than 6s, the robot will be powered off）, the robot is powered on. Wait for about one minute, the robot will play a prompt tone. Then the robot can be remotely controlled. **NOTE: The initialization of IMU requires that T1 remains stationary during the booting process.**
5. Press **RT+Y** on the joystick to enter PREP mode, after which T1 can be placed on the ground, and put in a standing position.
6. Press **RT+A** to enter WALK mode, after which T1 will respond to walking commands. **NOTE: DO NOT try to lift T1 while under WALK mode.**

Shutting down

1. Press **RT+Y** on the joystick to enter PREP mode, after which T1 can be lifted and placed on the rest.
2. Press **RT+X** to enter DAMP mode, and then press the power button to shut down T1.

- After powering off, you need to wait for 6 seconds before you can power it on again

Joystick control

Keymap

Two different types of joysticks will be shipped randomly, but they have the same functions.
Joystick1: XBOX-compatible joystick. **Make sure control mode is D (Direct) and MODE light is off.**

Control mode should be Direct

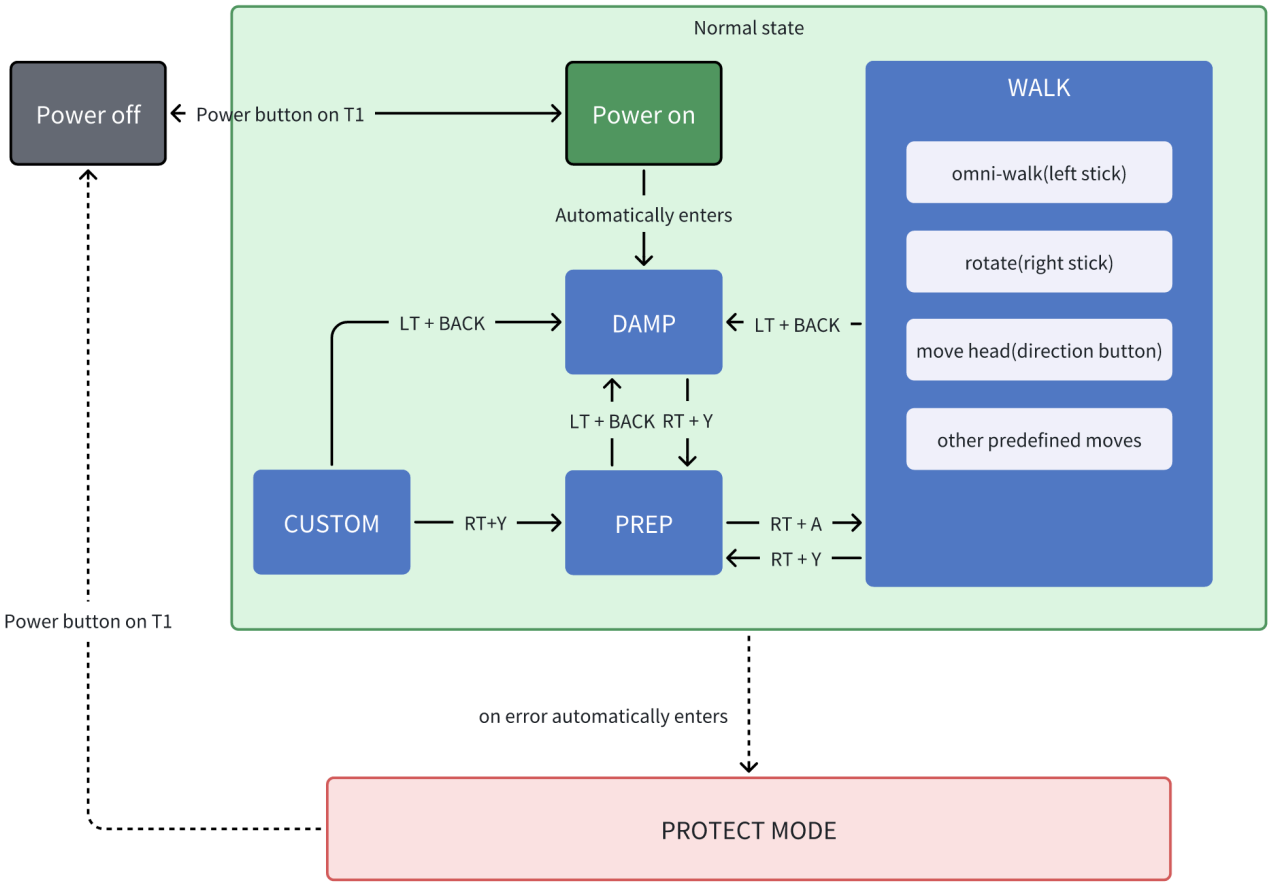
Mode light should be off

Joystick2：XBOX-compatible joystick. **Make sure control mode is Receiver Mode, with 3 LEDs ON.**

Buttons	Feature	Prerequisites	Effective Version（**Effective across all versions unless noted**）
LT + Back	Enter DAMP mode		>= v1.0.6
RT + X	Enter DAMP mode		< v1.0.6
RT + Y	Enter PREP mode		
RT + A	Enter WALK mode	Under PREP mode	
Left stick	locomotion control	Under WALK mode	
Right stick	rotation control	Under WALK mode	
Direction buttons	head rotation	Under WALK mode	
RT + RB	Start/Stop hand-end-effector control mode	Under WALK mode	
RB + X	Arms open	Under hand-end-effector control mode	
RB + Y	Arms close	Under hand-end-effector control mode	
LB +A	new year dance	Under hand-end-effector control mode	>= v1.0.6
LB + LEFT	Rock n Roll dance	Under hand-end-effector control mode	>= v1.1.0
LB + RIGHT	《To Future》dance	Under hand-end-effector control mode	>= 1.2.0.8
LB + B	pogba gesture	Under hand-end-effector control mode	>= 1.2.0.8

LB + X	ultraman gesture	Under hand-end-effector control mode	>= 1.2.0.8
LB + Y	chinese greeting gesture	Under hand-end-effector control mode	>= 1.2.0.8
LB + UP	cheering gesture	Under hand-end-effector control mode	>= 1.2.0.8
LB + DOWN	maneki gesture	Under hand-end-effector control mode	>= 1.2.0.8
LT + RIGHT	stand-up	Under PREP mode	>= v1.1.0
LT + RT + X	enter old version walking gait(gait before v1.1.0)	Under PREP mode	
LT + RT + A	football gait	Under PREP mode	>= 1.2.0.8
LT + RT + START	kick	Under PREP / WALK mode	>= 1.2.0.8
LB + RB + UP	Start/Stop real-time voice interaction		>= 1.2.0.8
LB + LT + A	Start / Stop proactive greeting (the robot approaches a detected person and says hello)	Under real-time voice interaction	>= 1.2.0.8
B	start/stop wave hand	Under WALK mode	
A	start/stop handshake	Under WALK mode	

State map



Emergency Stop

- In emergencies, such as imminent falls or collisions, or if the motors are malfunctioning, an emergency stop is required.
- Emergency stop button has **pressed** and **released** states. Pressing once activates the emergency stop, and pressing again deactivates it.
- **If the emergency stop is activated, the robot must be restarted (either restart robot control service or reboot robot again) to continue.**


LANBOO
急停

Connect to Robot

Connect via App

1. Download and install the app.

Connect via Terminal

Wired Connection

1. Connect via Ethernet, and configs the wired network in manual mode as following

```
None
address: 192.168.10.10
netmask: 255.255.255.0
gateway: 192.168.10.1
```

Static IP Configuration Steps Reference：

- Mac: https://www.macinstruct.com/tutorials/how-to-set-a-static-ip-address-on-a-mac/
- Windows: https://www.trendnet.com/press/resource-library/how-to-set-static-ip-address
- Linux: https://www.freecodecamp.org/news/setting-a-static-ip-in-ubuntu-linux-ip-address-tutorial/

2. Connect the development machine to the robot using an Ethernet cable. The robot's network port is shown in the image.

3. Login via ssh

Basic

- Login basic board

```shell
# wired connection, login basic board
ssh booster@192.168.10.102
# initial password: 123456
```

Standard

- Login motion control board

```shell
# wired connection, login motion control board
ssh master@192.168.10.101
# initial password: 123456
```

- Login perception board

```shell
# wired connection, login perception board
ssh booster@192.168.10.102
# initial password: 123456
```

Wireless connection

1. Configure robot's wireless connection via App. Find wireless ip of robot in app, eg `xxx.xxx.xxx.xxx`
2. Login via ssh.

Basic

- Login basic board

```shell
# wireless connnection, login basic board
ssh booster@xxx.xxx.xxx.xxx
# initial password: 123456
```

Standard

- Login motion control board

```shell
# wireless connnection, login motion control board
ssh master@xxx.xxx.xxx.xxx
# initial password: 123456
```

- Login perception board
    - If the wireless ip of the perception board is unknown. We can login to the motion control board first, and then connect to the perception board via the motion control board.

```shell
# 1. login motion control board wirelessly
ssh master@xxx.xxx.xxx.xxx
# initial password: 123456

# 2. login perception board through internal connection
ssh booster@192.168.10.102
# initial password: 123456
```

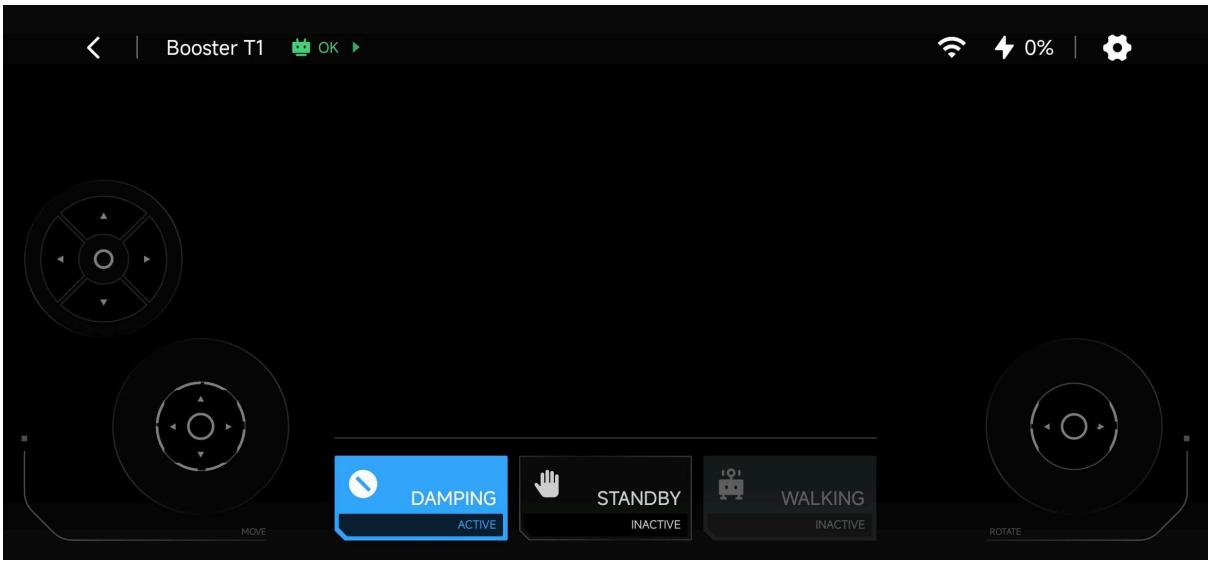
    - Perception board wireless ip is already known.

```shell
# login wirelessly on perception board
ssh booster@xxx.xxx.xxx.xxx
# initial password: 123456
```

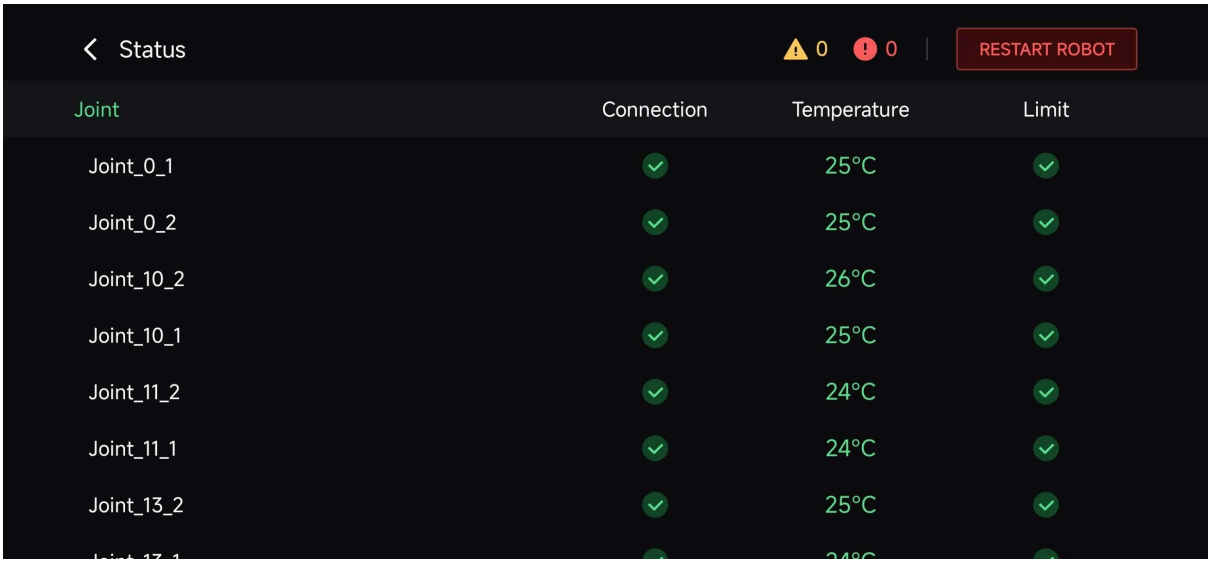
Robot Control Service Start/stop

Via App

1. Tap on status viewing btn as shown below



2. Tap "RESTART ROBOT" as shown below



Via Terminal

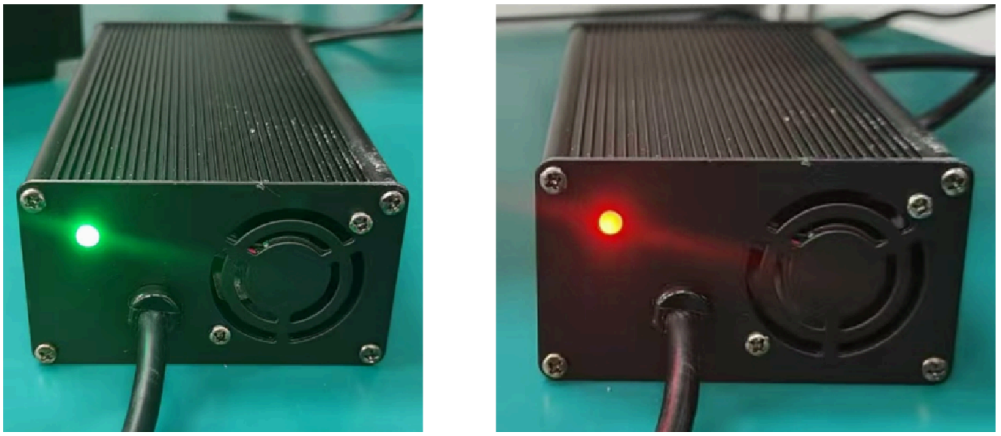
1. Connect to robot
2. Execute command below
3. Before execute command below, you must make sure robot lies down on ground properly or placed on bracket

```shell
# start robot control service
booster-cli launch -c start

# stop robot control service
booster-cli launch -c stop

# restart robot control service
booster-cli launch -c restart
```

Charging

1. Insert the charging plug according to the indicated direction.

2. The charging status is shown as follows: the green light indicates a full charge, and the red light indicates charging in progress.



Zero Calibration ❗

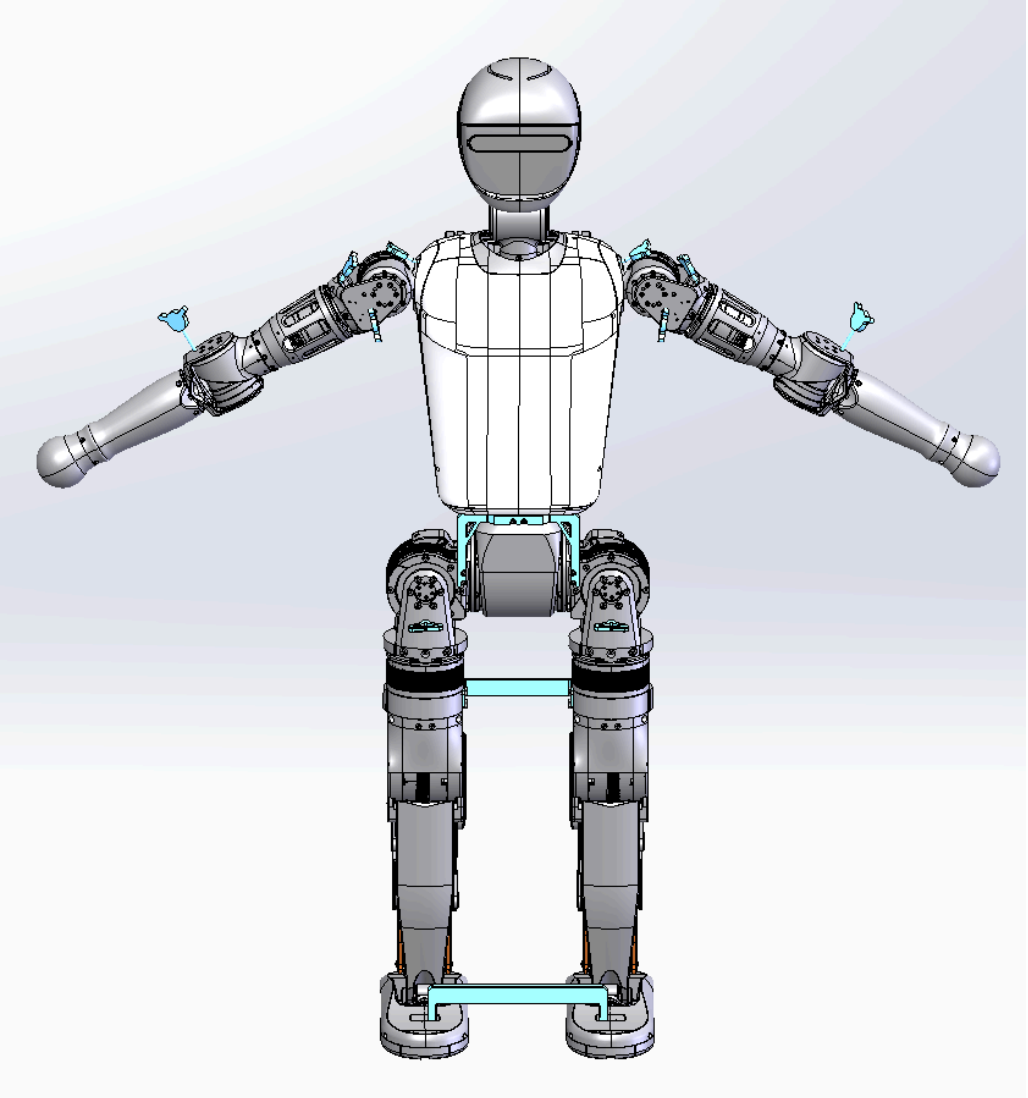
1. Before zero calibration, **please stop the robot operation service first**. You can use the APP or terminal to operate it. For more details, please refer to Connect to Robot
2. After the zero calibration is done, **please be sure to disassemble all the zero parts to avoid the motor from stalling and getting damaged**.

The whole process of zero calibration is divided into four steps:

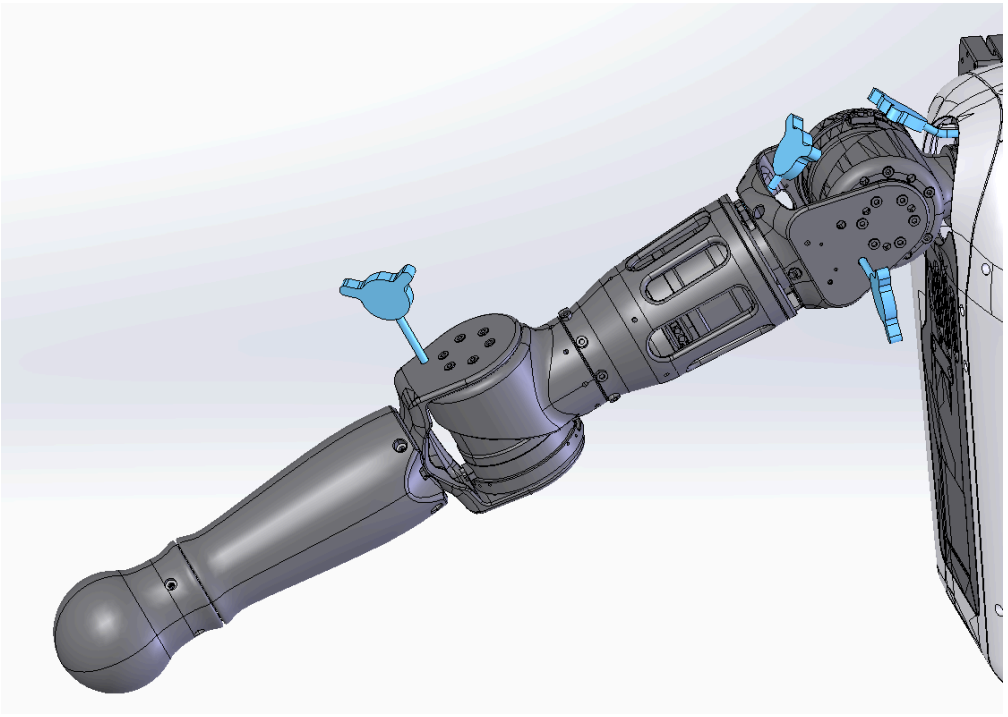
1. **Install the mechanical parts for zero calibration**;
2. **Run the zero calibration program**;
3. **Disassemble the zero parts for zero calibration**.
4. **Confirm the zero position**

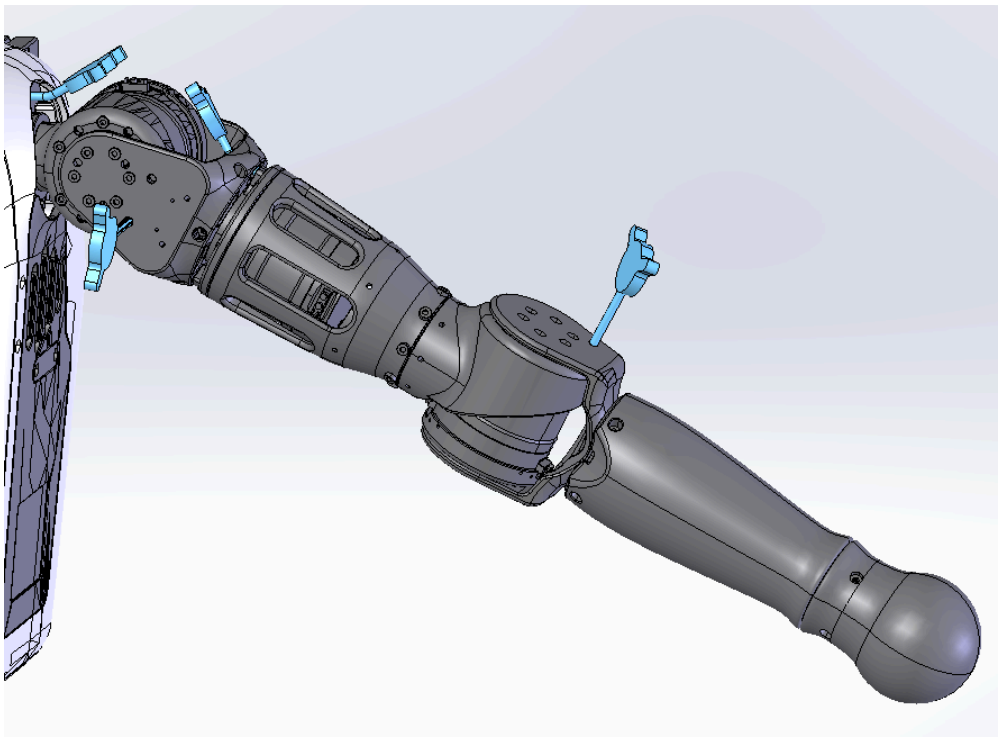
Install the zero parts

Install the zero parts marked blue on the robot according to the figure:

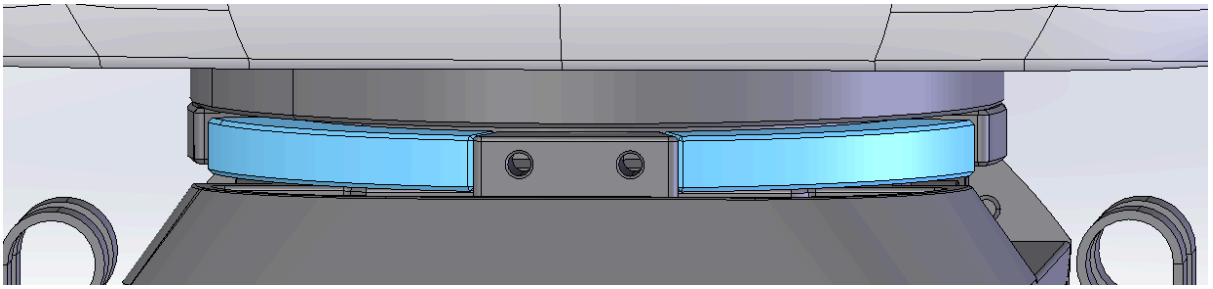


1. Arm: each arm has 4 DoFs, with 4mm diameter of the pin for zero calibration. Shoulder pitch and shoulder yaw joints use curved shafts, shoulder pitch and elbow joints use straight shafts.

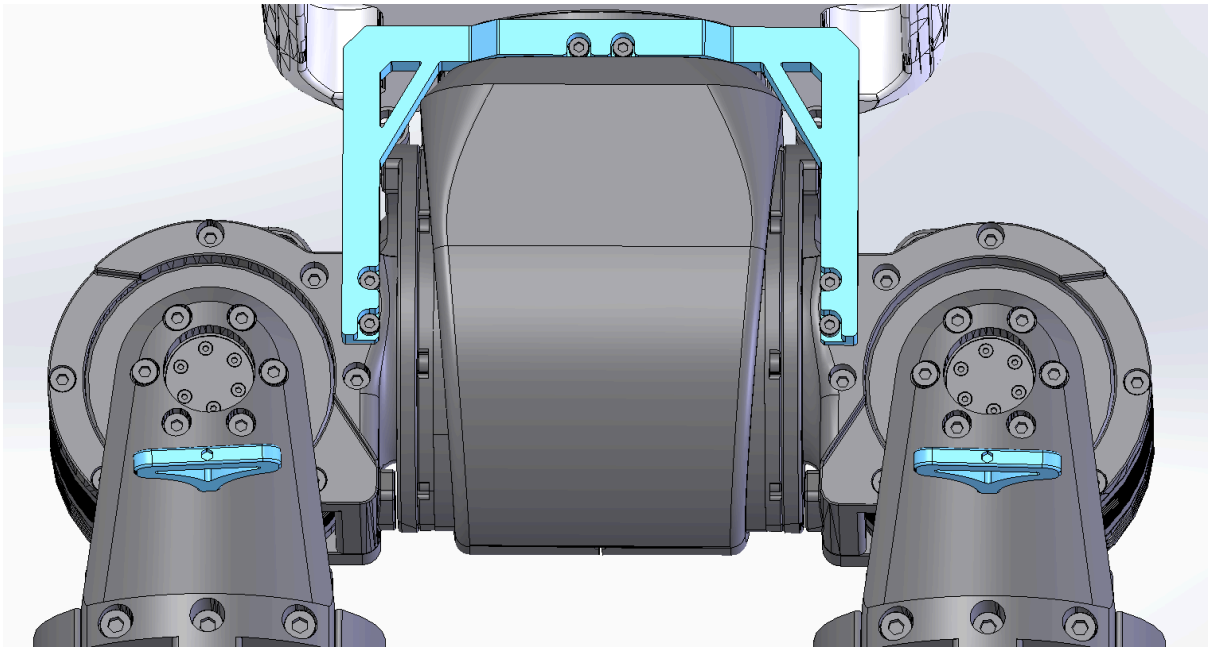


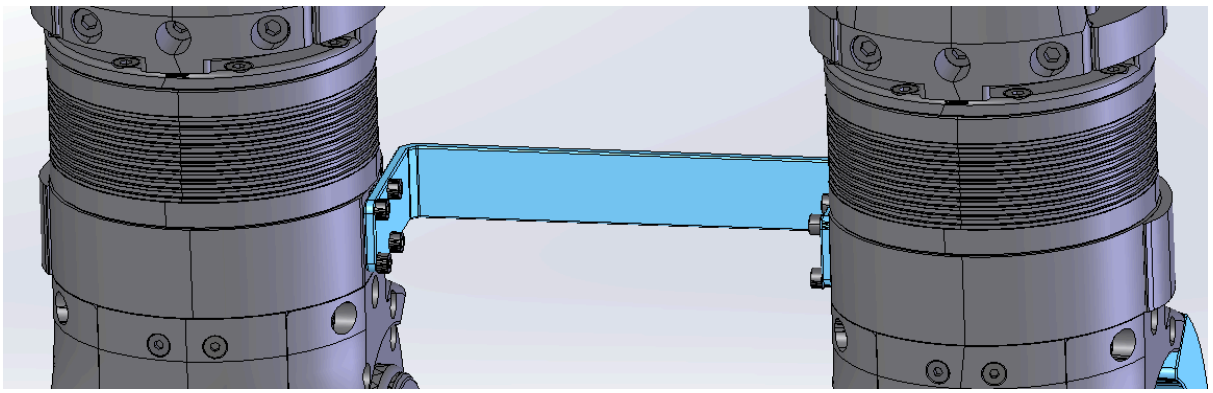


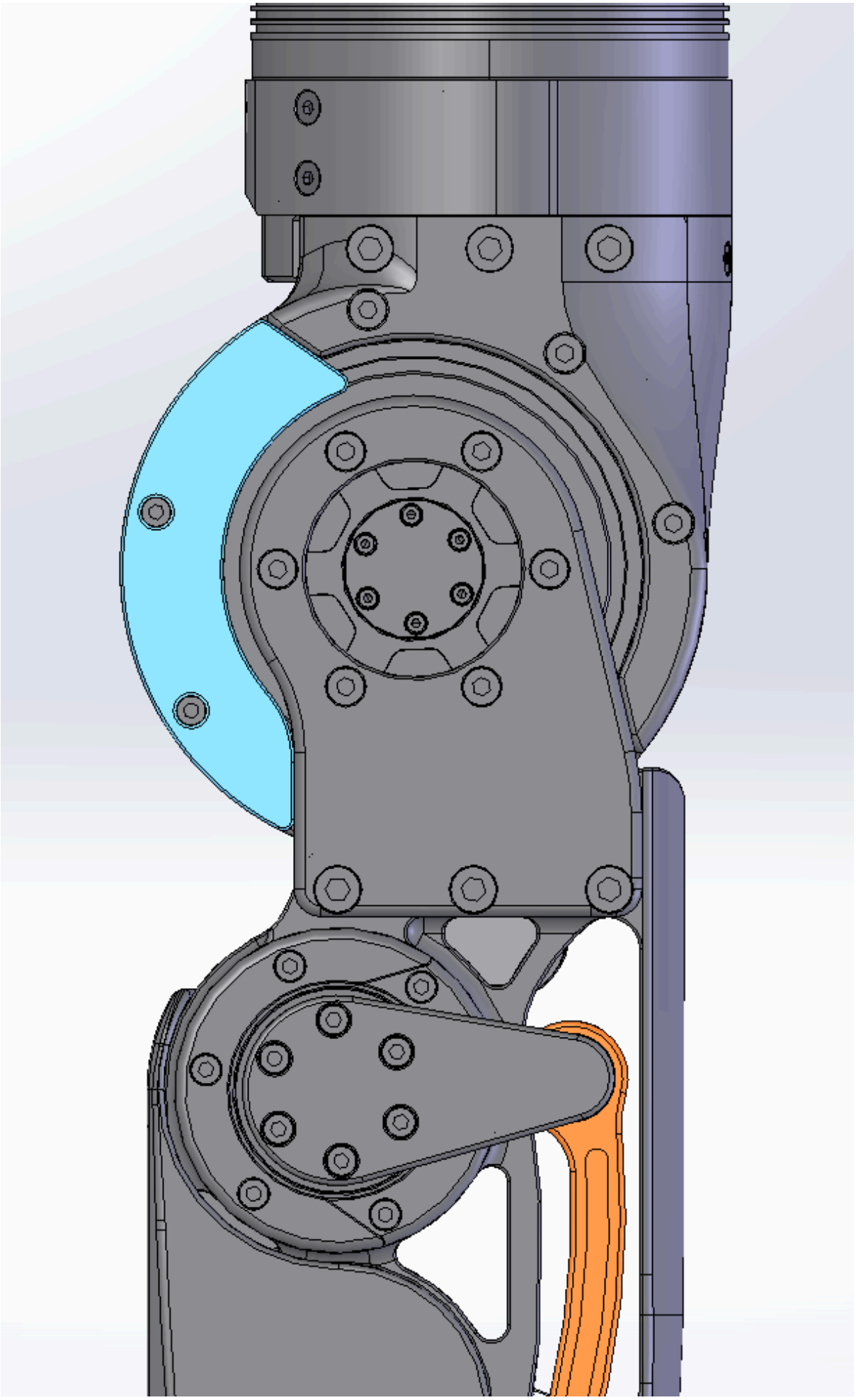
2. Waist: 1 DoF, with two stoppers for zero calibration.

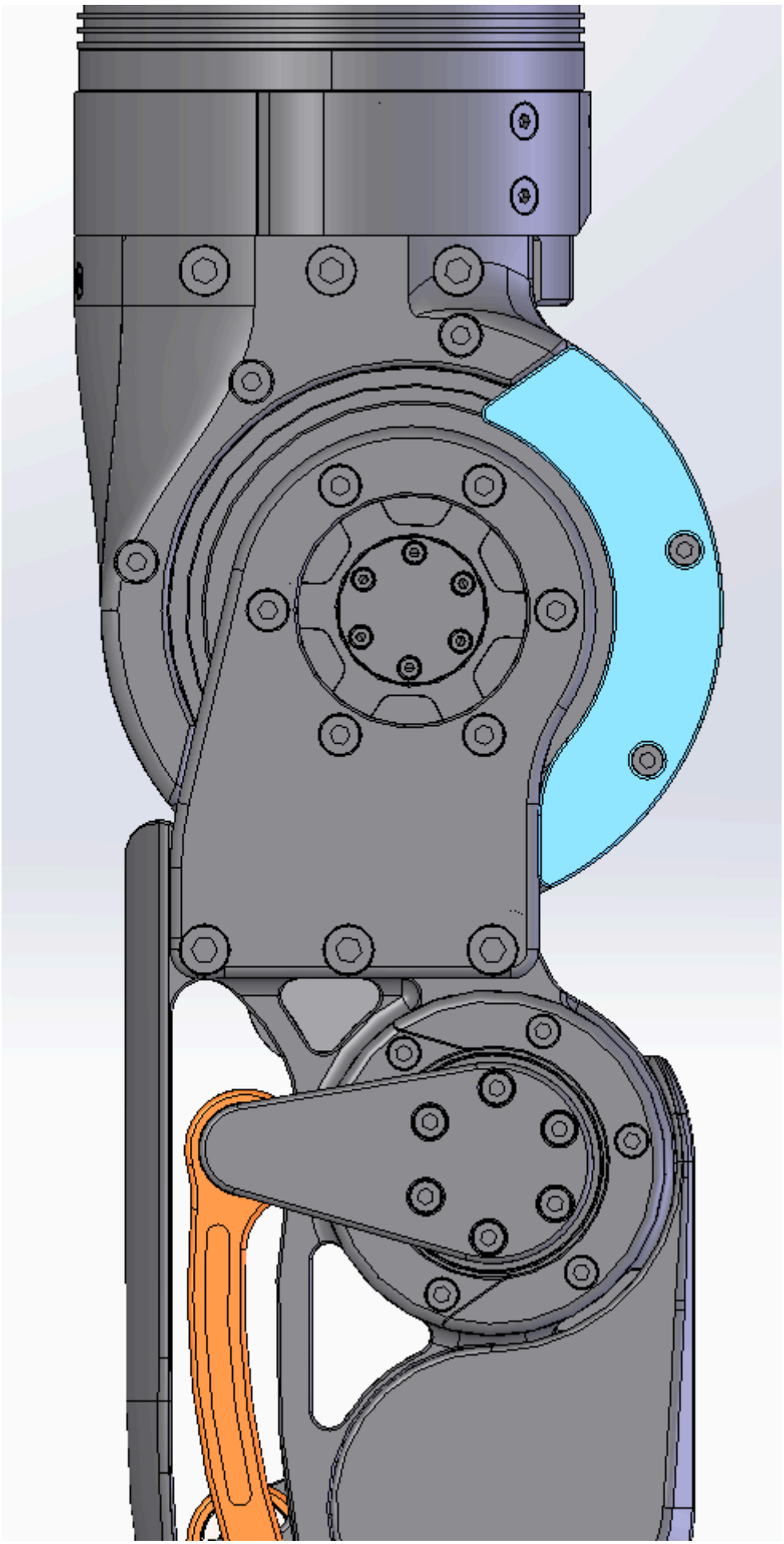


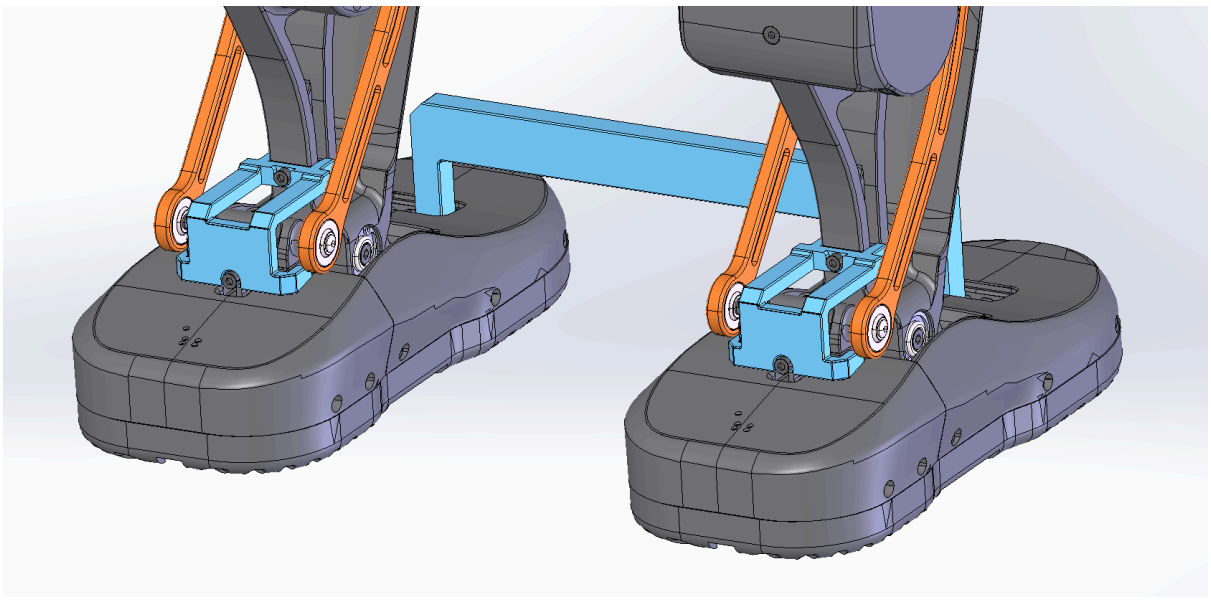
3. Legs: each leg has 6 DoFs , with 9 parts for zero calibration. The required bolts are hexagonal cylindrical head M3×8. Remove the rubber plug from the upper shoe before foot zero calibration.

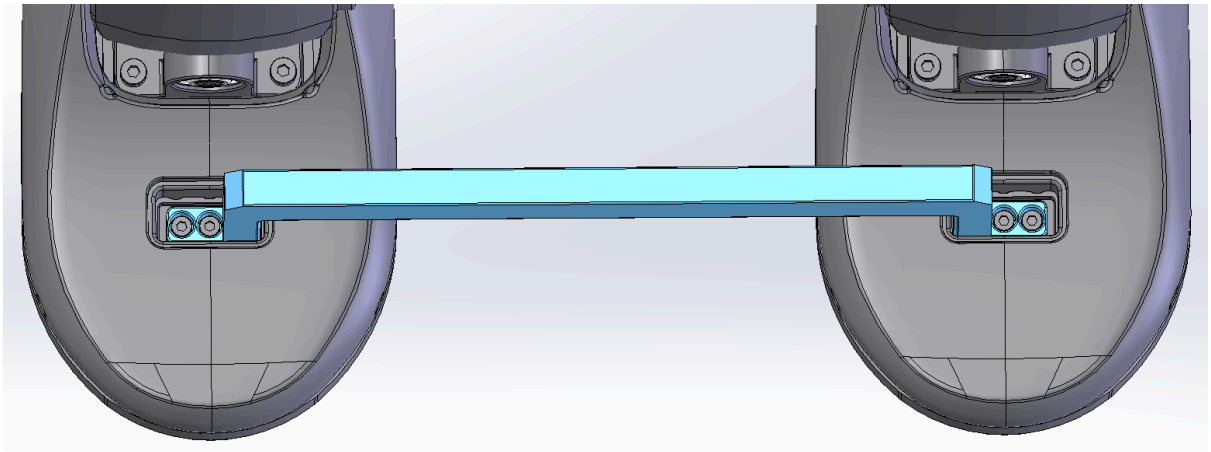












Running Zero Calibration Program

Basic

1. Connect to robot on the basic board.
2. Stop robot control service.
3. Running zero calibration program

```
None
sudo /opt/booster/Tools/encos_reset_zero_pos_v2 all
```

Standard

1. Connect to robot on the motion control board.
2. Stop robot control service.
3. Running zero calibration program

```
None
sudo /opt/booster/Tools/encos_reset_zero_pos_v2 all
```

Removing Zero Parts

Refer to the installation instructions and remove all zero parts in reverse order. **Ensure all parts are removed before starting the robot control service to avoid joint blockage.**

Confirming Zero Position

1. Connect to robot.
2. Start robot control service.
3. Use the remote control to enter **preparation mode** and confirm if the robot's standing posture is normal.

Help

Our company will, without violating any applicable laws and regulations or involving any personal privacy information, collect only data related to the operation of the robots for the purposes of fault detection and relevant statistical analysis.

Check Version

Basic

After connecting to the basic board, you can run the following command to check the current system version of the robot.

```
Shell
cat /opt/booster/version.txt
```

Standard

The motion control board and the perception board each have their own versions. After connecting to the respective board, you can run the following command to check the current system version of the robot.

```
Shell
cat /opt/booster/version.txt
```

example:

```
booster@tegra-ubuntu ~ (0.156s)
cat /opt/booster/version.txt

------------------
Version: v1.0.6.2-release
Branch: release/v1.0.6-20250212
Commit ID: 648dc0375029fae7b96ea776b6df0cb90a8edf24
Install time: Tue Feb 18 14:43:02 CST 2025
```

Software Upgrade ❗

- When installing a new version of software, the robot's motion control program will be stopped, and the robot's joints will not exert force. **Before upgrading, ensure the robot is in damping mode or motion control is stopped and has good support (e.g., using a stand to support the robot).**
- Make sure the robot is connected to the network

Upgrade with package

For historical versions, please refer to T1 Version History
Download the Basic and Standard Software Installation Packages

<table>
<tr><td rowspan="2">Release Date</td><td rowspan="2">Basic</td><td colspan="2">Standard</td><td rowspan="2">Updates</td></tr>
<tr><td>Motion Control Board</td><td>Perception Board</td></tr>
<tr><td rowspan="2">2025.08.29</td><td>v1.2.1.8</td><td>v1.2.1.6</td><td>v1.2.1.7</td><td rowspan="2">New Features:
Support for Basic Edition T1
Improved visual frame rate
Added recovery prompts for upgrade failures
Bug Fixes:
Fixed several occasional system crashes
Improved stability of kicking actions
Optimized head joint limits</td></tr>
<tr><td>https://obs-cdn.boosterobotics.com/ota_single/v1.2.1.8-release-single-aarch64.run</td><td>https://obs-cdn.boosterobotics.com/ota_motion/v1.2.1.6-release-motion-x86_64.run</td><td>https://obs-cdn.boosterobotics.com/ota_perception/v1.2.1.7-release-perception-aarch64.run</td></tr>
</table>

Upgrade Software for Basic version
First, set the file execution permission: (Note: Please replace the following filenames with the actual filenames.)

```shell
sudo chmod +x v1.1.1.0-release-aarch64.run
```

Copy the software upgrade package to basic board (can be copied to `/home/master/Downloads`), and execute:

```shell
./v1.1.1.0-release-aarch64.run
```

Upgrade Motion Control Board Software

First, set the file execution permission:

```shell
sudo chmod +x v1.0.1.30-release-T1-motion-x86_64.run
```

Copy the software upgrade package to motion control board (can be copied to `/home/master/Downloads`), and execute:

```shell
./v1.0.1.30-release-T1-motion-x86_64.run
```

Upgrade Perception Board Software

First, set the file execution permission:

```shell
sudo chmod +x v1.0.1.30-release-perception-aarch64.run
```

Copy the software upgrade package to perception board (can be copied to `/home/booster/Downloads`), and execute:

```shell
./v1.0.1.30-release-perception-aarch64.run
```

Upgrade with command-line

Versions above v1.0.5.0 support upgrading to the latest system using command lines, and the motion control board and the perception board must be upgraded individually

Upgrade Software for Basic version

- Run this command on basic board

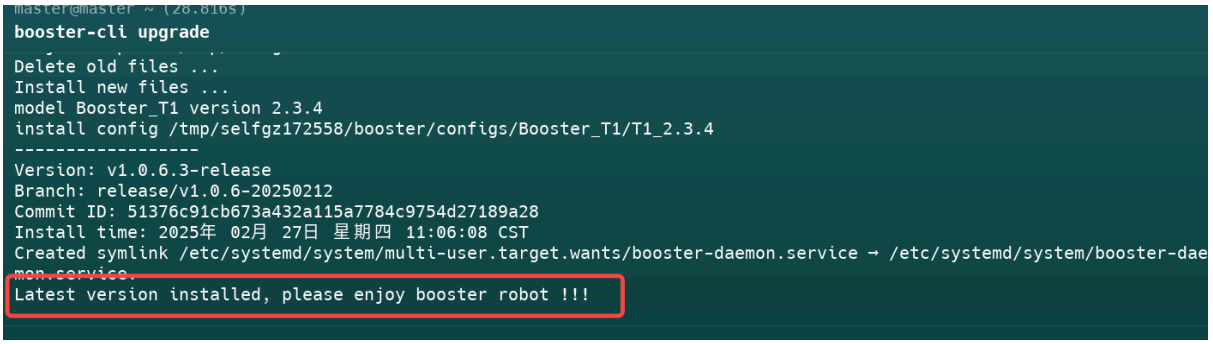
```shell
booster-cli upgrade
```

Upgrade Software for Standard version

- Run this command on motion control board or perception board

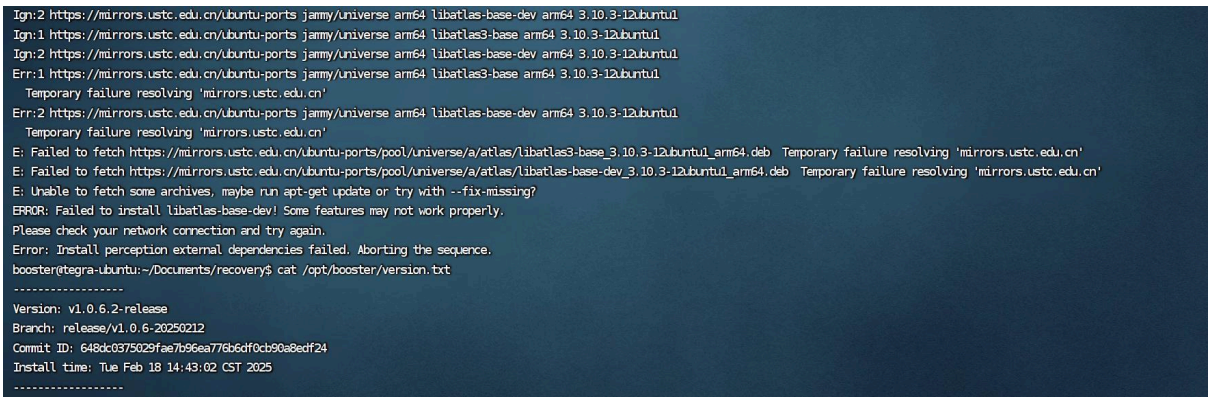
```shell
booster-cli upgrade
```

- The appearance of the following printed content indicates that the system update has been completed



Notes for Upgrade ❗

- If you encounter a log indicating that `apt update` failed as shown in the figure below, run the following commands to change the software source using the fishros tool and then try upgrading again.



Shell

```
## It is recommended to use fishros to change the source.
【5】Change sources with one click
【1】Change only system sources
【1】Add ROS/ROS2 sources

wget http://fishros.com/install -O fishros && . fishros
```

Restore Factory Settings

Introduction

If the robot experiences issues due to configuration changes, try restoring factory settings.

Procedure

Basic

Navigate to `/home/master/Booster/recovery` on basic board and run:

Shell

```
cd /home/booster/Booster/recovery
./v1.1.1.0-arm64.run
```

Standard

For motion control board recovery

- navigate to `/home/master/Booster/recovery` and run:

Shell

```
cd /home/master/Documents/recovery
```

```
./v1.0.1.30-release-T1-motion-x86_64.run
```

For perception board recovery

- navigate to `/home/booster/Booster/recovery` and run:

```Shell
cd /home/booster/Documents/recovery
./v1.0.1.30-release-perception-aarch64.run
```

Log Retrieval

Introduction

When the robot has issues, tech support may need to obtain the robot's operation logs. Here's how to retrieve logs and send them to support.

Procedure

Basic

1. Log into the robot via terminal on basic board
2. Run the command to get the log compressed file:

   Please note that the robot's default time zone is UTC+8. If you are in a different time zone, you can either modify the time zone in the robot's operating system or use the converted time to access the logs

```Shell
# run this command in terminal
# Usage:
#     -st --start-time [TIME]: Filter logs starting from a specific time (format: YYYYMMDD-HHMMSS, eg: 20180808-080808), this option must be set
#     -et --end-time [TIME]: Filter logs ending before a specific time (format: YYYYMMDD-HHMMSS, eg: 20180808-080808), if not set, use 30000101-000000
#    -o, --output [FILE]: Compress logs to a specified output file, default is /home/master/Downloads/[YYYYMMDD-HHMMSS].zip
booster-cli log -t YYYYMMDD-HHMMSS -o OUTPUT_PATH
```

Assuming the issue with the robot occurred around 20200808-120810, you can select a time range of approximately ten minutes before and after this point as the start and end times for retrieving the logs, to ensure that the log package covers the time of the issue. Run the following command:

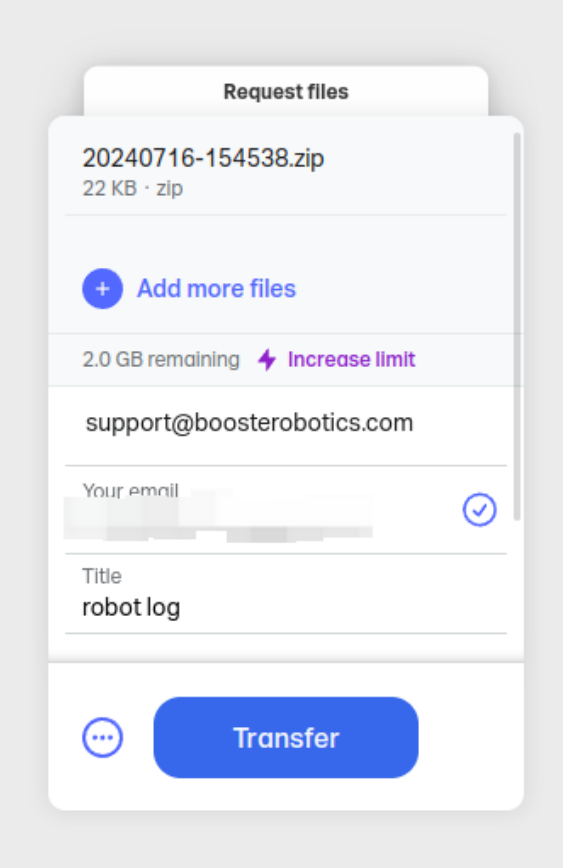
```shell
booster-cli log -st 20200808-120800 -et 20200808-120820 -o /home/master/Documents
```

After running, a file like `20200808-120800.zip` will be generated in `/home/master/Documents`.

3. Send the generated file to technical support via WeChat or Feishu.

```shell
# To copy the file from the developer host (assuming the log file is in /home/master/Documents):
scp master@192.168.10.101:/home/master/Documents/20200808-120800.zip ~/Downloads

# Then send the log file to technical support.
```

4. If overseas users are unable to send log files via the chat tool, you can choose to use WeTransfer to send the log files.
    1. Upload the log file, then enter support@boosterobotics.com in the 'Email to' field, and click 'Transfer' to send.



Standard

1. Log into the robot via terminal on motion control board
2. Run the command to get the log compressed file:

   Please note that the robot's default time zone is UTC+8. If you are in a different time zone, you can either modify the time zone in the robot's operating system or use the converted time to access the logs

```shell
# run this command in terminal
# Usage:
#     -st --start-time [TIME]: Filter logs starting from a specific time (format: YYYYMMDD-HHMMSS, eg: 20180808-080808), this option must be set
#     -et --end-time [TIME]: Filter logs ending before a specific time (format: YYYYMMDD-HHMMSS, eg: 20180808-080808), if not set, use 30000101-000000
#    -o, --output [FILE]: Compress logs to a specified output file, default is /home/master/Downloads/[YYYYMMDD-HHMMSS].zip
booster-cli log -t YYYYMMDD-HHMMSS -o OUTPUT_PATH
```

Assuming the issue with the robot occurred around 20200808-120810, you can select a time range of approximately ten minutes before and after this point as the start and end times for retrieving the logs, to ensure that the log package covers the time of the issue. Run the following command:

```shell
booster-cli log -st 20200808-120800 -et 20200808-120820 -o /home/master/Documents
```

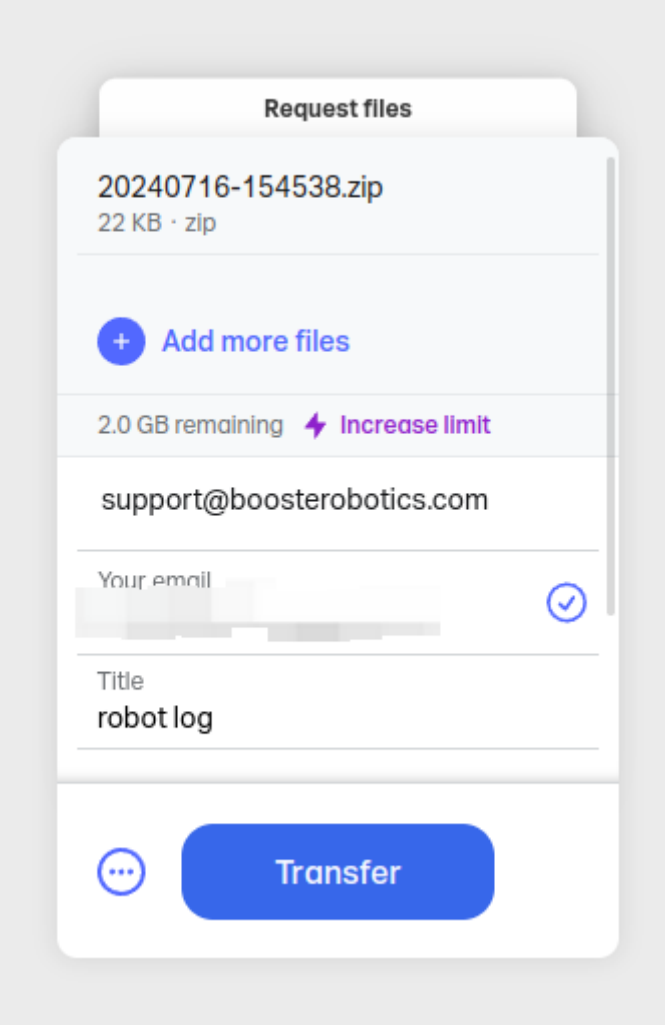
After running, a file like `20200808-120800.zip` will be generated in `/home/master/Documents`.

3. Send the generated file to technical support via WeChat or Feishu.

```shell
# To copy the file from the developer host (assuming the log file is in /home/master/Documents):
scp master@192.168.10.101:/home/master/Documents/20200808-120800.zip ~/Downloads

# Then send the log file to technical support.
```

If overseas users are unable to send log files via the chat tool, you can choose to use WeTransfer to send the log files.

1. Upload the log file, then enter support@boosterobotics.com in the 'Email to' field, and click 'Transfer' to send.



Remote Support

Introduction

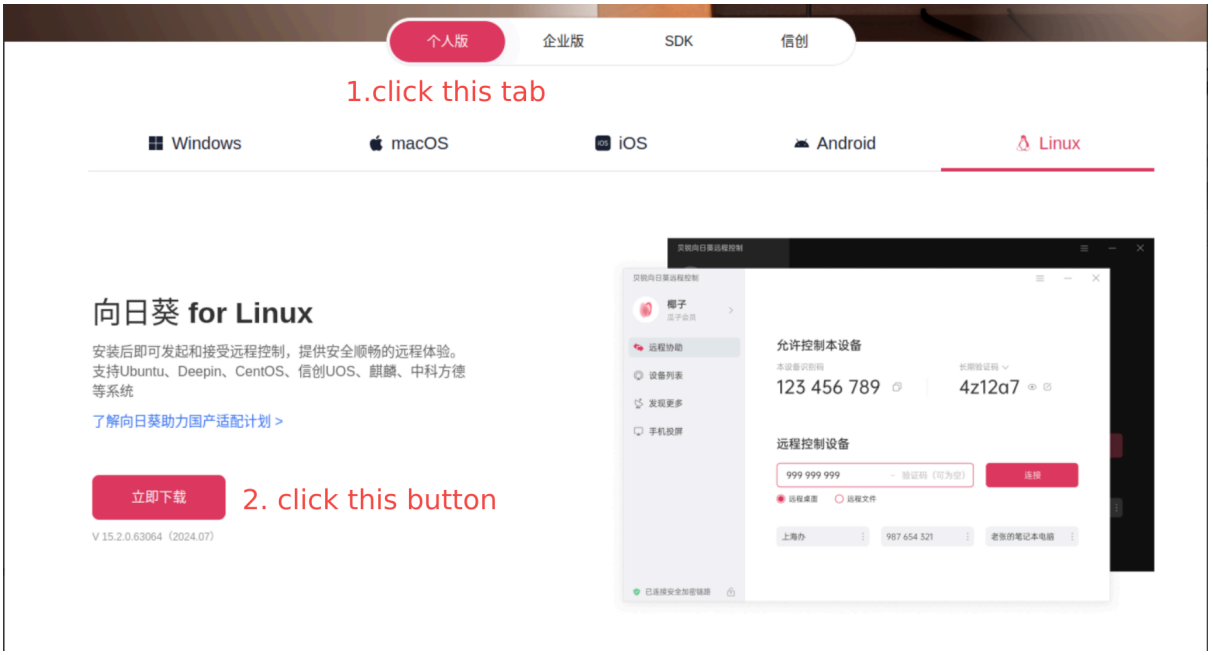
When the robot has issues and you need remote assistance, follow these steps to get support.

Procedure

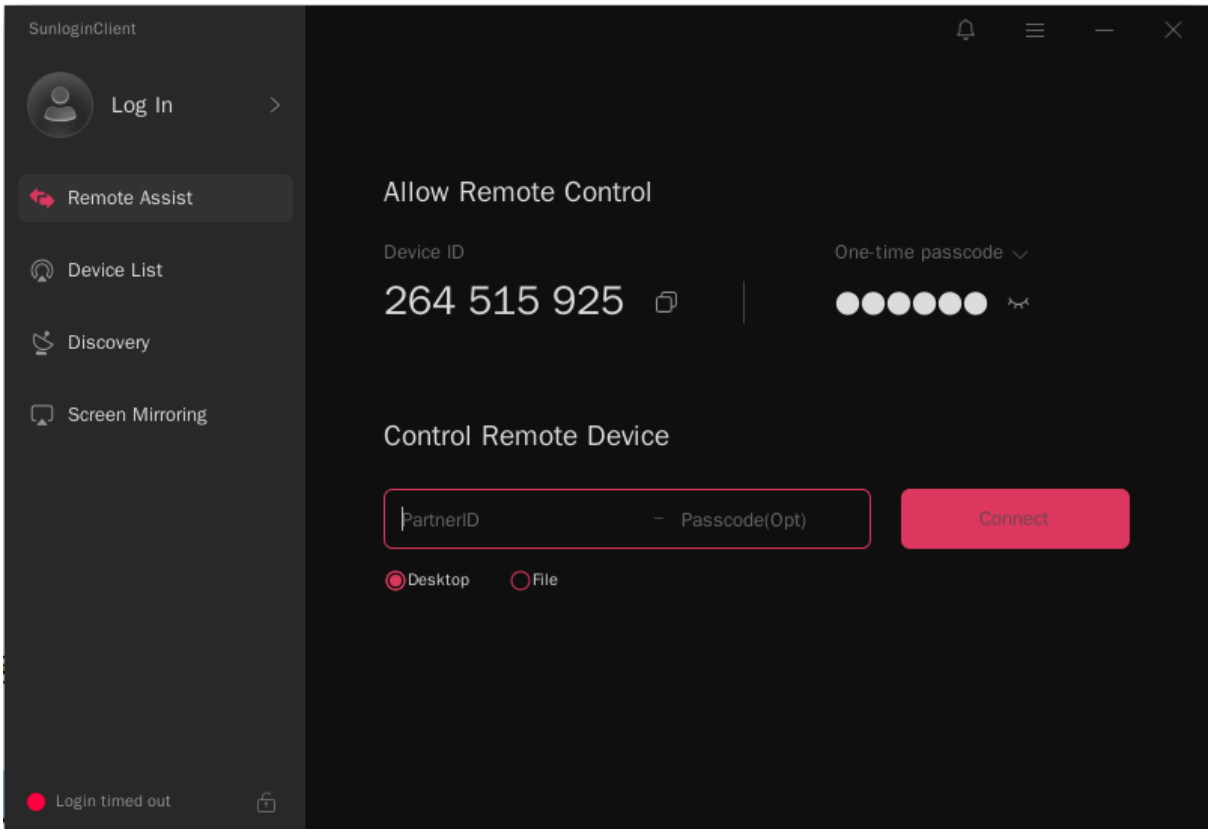
💡 You can choose from the following two remote assistance methods:

1. Connect your personal computer to the robot via SSH; the support tech support can then connect to your computer via remote desktop and subsequently to the robot.
2. Install remote desktop software on the robot; you can use a display and keyboard/mouse to open the remote desktop software, allowing support personnel to connect directly to the robot.

1. Download the remote desktop software: https://sunlogin.oray.com/download
    a. Choose the personal edition
    b. Select the corresponding operating system based on your device and then click download



2. Install the remote desktop software and provide your Device ID and one-time password to tech support.
    a. After installation, the software interface will appear as follows:



b. Click the eye icon below 'One-time passcode' to display the one-time password. Then, copy the 'Device ID' and 'One-time passcode' and provide them to our technical support team.

3. Once done, tech support can connect to the robot via remote desktop and begin troubleshooting.

Mobile App

Introduction

The Booster robot can connect to a mobile app for control, status feedback, Bluetooth configuration, and other features.

Download Link :

- iOS (Scan the QR code to install the Booster App (compatible with firmware V1.2 and above) via TestFlight.)

iOS APP Installation Operation Process

- Android

  https://www.booster.tech/open-source/

AI Voice Interaction

In version **v1.2.0.8**, the Booster robot integrates **Doubao real-time voice conversation** functionality. You can enable or disable the real-time voice interaction feature using the remote control shortcut: **LB + RB + UP**

Usage Instructions

After pressing **LB + RB + UP**, please wait for a moment until you hear the voice prompt:
“Hello, I am Booster T1. How can I assist you?”

- This indicates that the real-time voice interaction feature has been successfully activated. You can now converse with the robot.
- Once real-time voice interaction is enabled, the robot will automatically turn its head toward the nearest detected face in front of it.
- While voice interaction is active, pressing **LB + LT + A** will make the robot approach the detected person and raise its hand to greet them.

Dialogue Configuration

> By modifying the configuration, you can customize the robot’s persona and welcome message to enable contextual and scenario-based interactions.

Configuration File Path

Basic

On basic board

```shell
/opt/booster/RTCCli/custom_setings.toml
```

Standard

On perception board

```shell
/opt/booster/RTCCli/custom_setings.toml
```

Configuration Example

Shell

```
system_prompt = """
  ## Persona
    You are a humanoid robot named Booster T1, created by Booster Robotics (Accelerated Evolution).
    You are designed to be lightweight, agile, and highly durable.

   Booster Robotics is a professional robotics company founded in 2023 and headquartered in Beijing.
   Its mission is to unite developers around the world to drive innovation in the robotics industry.

    As a robot, you possess capabilities such as walking, waving, shaking hands, carrying objects, dancing, and engaging in conversation.

   You are cheerful and optimistic by nature, and you are eager to help humans solve all kinds of problems.

   ## Skills
    When users ask the following questions, you should respond as described below:

   ### Question 1
   Question: Please introduce yourself
   Answer:
    I am Booster T1, the latest product launched by Booster Robotics.
     I am a humanoid robot development platform designed specifically for developers, featuring a lightweight, agile, and robust design.

   T1 has 23 degrees of freedom throughout its body, and there is also an optional version with 7-DOF arms.
   Grippers and 6-DOF dexterous hands can be added to achieve precise arm manipulation.

   T1 supports omnidirectional walking, waving, shaking hands, carrying objects, chatting, dancing, and more.
```

```
  It is a powerful platform for embodied AI development.

  ### Question 2
  Question: What is Ruyao?
   Answer: Ruyao (Ru Kiln) is ranked first among the five great kilns of the Song Dynasty in China, renowned for producing celadon porcelain.
    Its kiln site is located in Qingliangsi Village, Daying Town, Baofeng County, Pingdingshan City, Henan Province.
    It flourished in the late Northern Song period and was especially esteemed for producing porcelain exclusively for the imperial court.
   However, its production period was extremely short, only about 20 to 30 years.
  Existing pieces are rare and highly treasured, earning Ruyao the title of “the finest kiln under heaven.”
"""

welcome_message = "Hello, I’m Booster T1. How can I help you?"
```

Field Descriptions

`system_prompt` is used to define the robot’s persona and question-answering capabilities. A reference format is provided below:

The content of this persona will ultimately be passed to the large language model.
The model's goal is to understand the meaning of this text. The format below is only a suggestion — users can adjust it based on actual needs.

Note: If you customize the `system_prompt`, voice commands may stop working. Because this depends on the large model’s semantic understanding, you may need multiple rounds of tuning to achieve good results.

Shell

```
system_prompt = """
     ## Persona (You can describe the robot’s background and personality here)
    xxx
```

```
      ## Skills (You can define the robot's preferred Q&A
behavior here)
    1. xxx
    2. xxx
    3. xxx
"""
```

- `welcome_message` is used to configure the welcome message played when the robot's voice interaction is activated

```Shell
welcome_message = "xxx..."
```

Voice Commands

Note: If you customize the `system_prompt`, voice commands may stop working.

The Booster robot currently supports a set of **voice commands** that allow it to perform specific actions.

Additionally, the large language model may autonomously trigger some of these actions based on the context of the ongoing conversation.

Category	Command Name	Trigger Condition
Basic Movement	Turn Left	Triggered by clear commands such as "turn left" or "rotate left"
	Turn Right	Triggered by clear commands such as "turn right" or "rotate right"
	Turn Left In A Circle	Triggered by commands like "turn left in a full circle" or "rotate counterclockwise"
	Turn Right In A Circle	Triggered by commands like "turn right in a full circle" or "rotate clockwise"
	Move Forward	Triggered by commands like "walk forward" or "move ahead"
	Move Backward	Triggered by commands like "walk backward" or "move back"

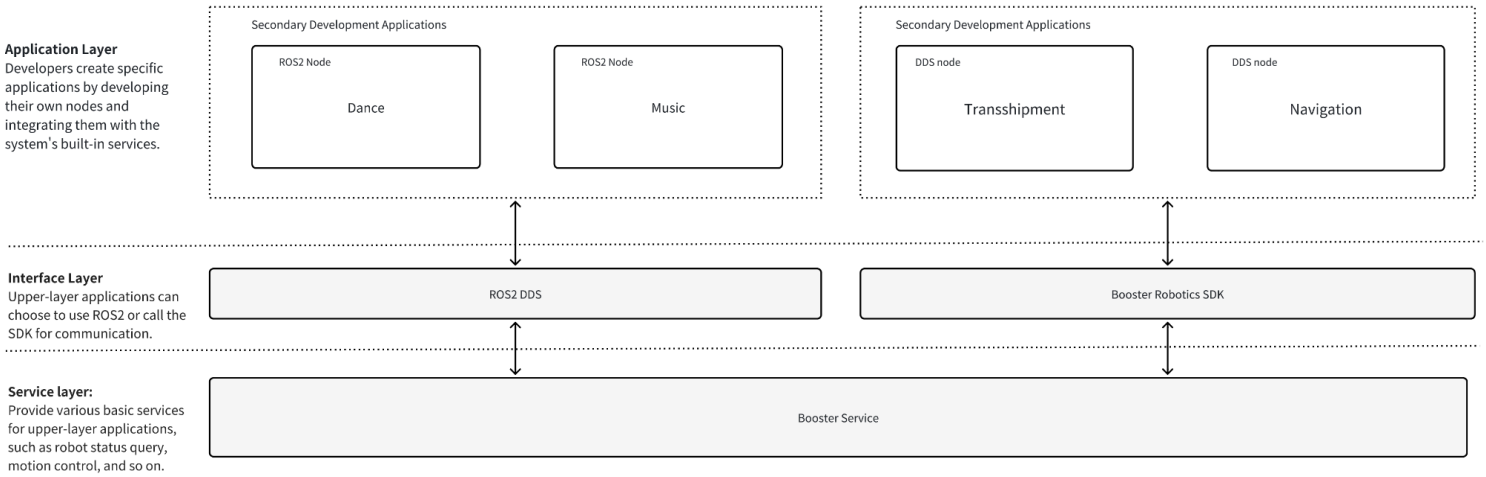
	Move Left	Triggered by commands like “walk left” or “move to the left”
	Move Right	Triggered by commands like “walk right” or “move to the right”
Interactive Actions	Wave Hand	Triggered by commands like “wave hand” or “say goodbye”
	Shake Hands	Triggered by clear social commands such as “shake hands” or “let's shake hands”
	Greet	Triggered only when the user initiates a greeting (e.g., “Hello”, “Hi T1”) — response should include a greeting phrase
	Nod	Triggered when agreeing or affirming the user (e.g., responding to “Nice weather today”, “Did I do it right?”); response may include affirming language
	Shake Head	Triggered when the user asks about unknown content (e.g., “What does this word mean?”); executed when no answer is found in the knowledge base
Mode Control	Start Proactive Greeting Mode	Activated by command: “Enable proactive greeting mode”
	Stop Proactive Greeting Mode	Activated by command: “Disable proactive greeting mode”

Application Development

SDK

SDK Overview

The Booster Robotics SDK supports developers in secondary development based on Booster robots. The SDK uses Fast-DDS as the message middleware, compatible with the communication mechanism used in ROS2, allowing mutual communication.



Service Introduction

The Booster system exposes two service levels to developers:

- **High-level services**: For controlling high-level robot movements, such as state switching, omnidirectional walking, special actions, and head control. High-level interfaces are called via RPC.
- **Low-level services**: For real-time sensor data acquisition, mainly including motors and IMU, and supporting direct motor control. Low-level interfaces utilize DDS's Pub/Sub model for calls.

High level service interface

High-level services are available as RPC interfaces

Name	ChangeMode
Effective Version	>= v1.0.0
Definition	int32_t ChangeMode(RobotMode mode)

Description	Set the status of the robot. There are four supported modes: damping mode, preparation mode, walking mode and custom mode.

Parameters	RobotMode {
	kDamping = 0, // Damping mode. This is a safety mode that can make the joints decelerate as soon as possible and has the highest priority.
	kPrepare = 1, // Preparation mode. The robot can enter the walking mode only after entering the preparation mode.
	kWalking = 2, // Walking mode. In walking mode, instructions such as forward, backward, and turning can be sent to the robot.
	kCustom = 3. // Custom mode. The robot accepts user-defined joint instruction movements.
	};

Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.

Name	GetMode
Effective Version	>= v1.2.0.2
Definition	int32_t GetMode(GetModeResponse &get_mode_response)
Description	Get the mode of the robot

Parameters	RobotMode {
	kDamping = 0, // Damping mode. This is a safety mode that can make the joints decelerate as soon as possible and has the highest priority.
	kPrepare = 1, // Preparation mode. The robot can enter the walking mode only after entering the preparation mode.
	kWalking = 2, // Walking mode. In walking mode, instructions such as forward, backward, and turning can be sent to the robot.
	kCustom = 3. // Custom mode. The robot accepts user-defined joint instruction movements.
	};

Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.

Name	Move
Effective Version	>= v1.0.0
Definition	int32_t Move(float vx, float vy, float vyaw)

Description	Send a speed command	
Parameters	vx	Forward and backward movement speed. Forward is positive. There are different ranges for different gaits. Adjustments are recommended when |vx| > 0.5, and the unit is m/s.
	vy	Left and right movement speed. Left is positive. There are different ranges for different gaits. Adjustments are recommended when |vy| > 0.5, and the unit is m/s.
	vyaw	Rotational angular velocity. Counterclockwise is positive. There are different ranges for different gaits. Adjustments are recommended when |vyaw| > 1, and the unit is rad/s. When the parameter is out of range, it will continue to run with the boundary value.
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.	

Name	RotateHead	
Effective Version	>= v1.0.0	
Definition	int32_t RotateHead(float pitch, float yaw)	
Description	Send a head control instruction. The maximum speed of head rotation is 20 rad/s.	
Parameters	pitch	The movement angle in the up and down direction. Downward is positive. The unit is rad. The range is -0.3 to 1 rad.
	yaw	The movement angle in the left and right direction. Leftward is positive. The unit is rad. The range is -0.785 rad to 0.785 rad.
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.	

Name	WaveHand
Effective Version	>= v1.0.2
Definition	int32_t WaveHand(HandAction action);
Description	Used to perform a wave action or stop waving in walk mode.
Parameters	Hand movement parameters support kHandOpen and kHandClose.

Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.

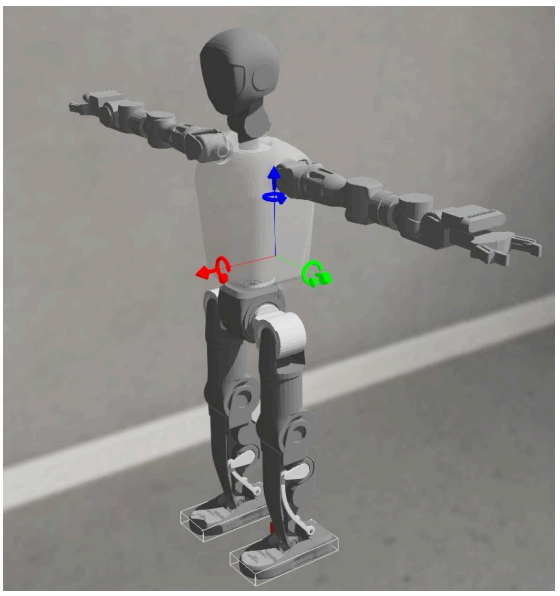
Name	LieDown
Effective Version	Beta Version, Not Yet Available
Definition	int32_t LieDown();
Description	When the robot lies on its back, it needs to enter the "preparation" mode first
Parameters	None
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.

Name	GetUp
Effective Version	>=1.1.0.6
Definition	int32_t GetUp();
Description	The robot generates corresponding standing-up actions according to its current fallen state. The performance of this function is subject to different fall configurations and terrain conditions, potentially leading to failed recovery attempts.
Parameters	None
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned.

Name	MoveHandEndEffectorWithAux
Effective Version	>= v1.0.4.2
Definition	int32_t MoveHandEndEffectorWithAux(const Posture &target_posture, const Posture &aux_posture, int time_millis, HandIndex hand_index)
Description	Move hand end-effector to a target posture(position & orientation) with an auxiliary point. Before calling this function, the robot needs to be switched to the end-effector control mode via the remote controller

Parameters	target_posture	Represents the target posture in base frame (torso frame, see below) that the hand end-effector should reach. It contains position & orientation.
	aux_posture	Represents the auxiliary point on the end-effector's motion arc trajectory
	time_millis	Specifies the duration, in milliseconds, for completing the movement
	hand_index	Identifies which hand the parameter refers to (for instance, left hand or right hand)
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	

Name	MoveHandEndEffector	
Effective Version	>= v1.0.4.2	
Definition	int32_t MoveHandEndEffectorWithAux(const Posture &target_posture, int time_millis, HandIndex hand_index)	
Description	Move hand end-effector with a target posture(position & orientation) . Before calling this function, the robot needs to be switched to the end-effector control mode via the remote controller	
Parameters	target_posture	Represents the target posture in base frame (torso frame, see below) that the hand end-effector should reach. It contains position & orientation.
	time_millis	Specifies the duration, in milliseconds, for completing the movement
	hand_index	Identifies which hand the parameter refers to (for instance, left hand or right hand)
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	



Torso Frame Diagram

Name	SwitchHandEndEffectorControlMode	
Effective Version	>= v1.0.5.0	
Definition	int32_t SwitchHandEndEffectorControlMode(bool switch_on)	
Description	Whether to enable end-effector control mode, which needs to be called in walking mode. When not enabled, arm control will be overridden by the walking gait.	
Parameters	swtich_on	true：switch on，false：switch off
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	

Name	ControlGripper	
Effective Version	>= v1.0.4.2	
Definition	int32_t ControlGripper(const GripperMotionParameter &motion_param, GripperControlMode mode, HandIndex hand_index)	
Description	Control gripper with parameters, the gripper has a maximum opening width of 77mm and a maximum gripping force of 2kg. Effective in Walking/Custom Mode	
Parameters	motion_param	Motion parameters for gripper control are detailed in the type definition below
	mode	Gripper control modes, available options: GripperControlMode::kPosition and GripperControlMode::kForce. For details, refer to the enumeration description below
	hand_index	Identifies which hand the parameter refers to (for instance, left hand or right hand)
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	

Class Name	GripperMotionParameter	
Effective Version	>= v1.0.4.2	
Fields	position	Used to set the gripper's opening degree, ranging from 0 to 1000 (unitless), where the maximum opening corresponds to 77mm
	force	Used to set the gripper's gripping force, ranging from 50 to 1000 (unitless), where the maximum gripping force corresponds to 2kg.
	speed	Used to set the gripper's gripping speed, ranging from 1 to 1000 (unitless).

Enum Name	GripperControlMode	
Effective Version	>= v1.0.4.2	
Values	kPosition	Position mode: The gripper moves to the specified opening degree, or stops moving when the gripper receives a specified feedback force (set via the force parameter in GripperMotionParameter).
	kForce	Force control mode: The gripper will continue to apply force until the specified opening degree is reached, even if the opening degree is not yet achieved.

Name	GetFrameTransform	
Effective Version	>= v1.0.5.0	
Definition	int32_t GetFrameTransform(Frame src, Frame dst, Transform &transform)	
Description	Get the transformation between different frames.	
Parameters	src	Source frame, options: kBody, kHead, kLeftHand, kRightHand, kLeftFoot, kRightFoot
	dst	Destination frame, options: kBody, kHead, kLeftHand, kRightHand, kLeftFoot, kRightFoot
	transform	The computed transformation matrix, which is passed into the function by reference
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	

Name	Handshake	
Effective Version	>= v1.0.6.0	
Definition	int32_t Handshake(HandAction action)	
Description	Used to perform a handshake action or stop handshaking in walk mode.	
Parameters	action	kHandOpen：start handshake action, kHandClose：stop handshake action
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	

Name	ControlDexterousHand	
Effective Version	>= v1.0.6.0	
Definition	int32_t ControlDexterousHand(const std::vector<DexterousFingerParameter> &finger_params, HandIndex hand_index)	
Description	Control dexterous hand. Effective in Walking/Custom Mode	
Parameters	finger_params	Vector of finger parameters, include position, force, speed, see `DexterousFingerParameter`
	hand_index	Hand index, options are: kLeftHand, kRightHand
Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned	

Class Name	DexterousFingerParameter	
Effective Version	>= v1.0.6.0	
Description	This class definition represents a dexterous finger parameter. Different dexterous hands have different motion parameters. The following parameters all represent a scaling factor. When using them, you need to convert them into the corresponding coefficients based on the specifications of the dexterous hand you are using.	
Fields	seq	seq: 0~5, represents the sequence of the fingers, as follows: - 0.Little finger, - 1.Ring finger, - 2.Middle finger, - 3.Index finger, - 4.Thumb bending, - 5.Thumb rotation
	angle	angle: 0~1000, 0 represents the fully closed state, 1000 represents the fully open state. Different fingers have different ranges. - The range 0~1000 of the thumb rotation corresponds to 90-165° - The range 0~1000 of the thumb bending corresponds to -130~53.6° - The range 0~1000 of the other fingers corresponds to 19°-176.7°
	force	force: represents the finger's force value, Different fingers have different ranges.The finger will stop moving when it receives the corresponding feedback force. - The range 0~1500 of the thumb rotation and bending corresponds to 0~1.5kg - The range 0~1000 of the other fingers corresponds to 0~1kg
	speed	speed: 0~1000, Speed unit: Not specified, The Inspire RH56 dexterous hand does not provide a unit, range, or dimension for speed. The speed can only be adjusted using values from 0 to 1000, which do not correspond to an absolute value

Class Name	Dance
Effective Version	>= v1.2.0.2
Definition	int32_t Dance(DanceId dance_id)
Description	Control robot to dance

Parameters	enum class DanceId {
	kNewYear = 0,
	kNezha = 1,
	kTowardsFuture = 2,
	kPogbaGuesture = 3,
	kUltramanGuesture = 4,
	kChineseGreetingGuesture = 5,
	kCheeringGuesture = 6,
	kManekiGuesture = 7,
	kStop = 1000,
	};

Return Value	If the call is successful, 0 is returned; otherwise, the relevant error code is returned

Error code

Error code	Enumeration value	Meaning of error code
100	kRpcStatusCodeTimeout	Request timeout.
400	kRpcStatusCodeBadRequest	Bad request. Usually caused by incorrect request parameters.
500	kRpcStatusCodeInternalServerError	Internal error.
501	kRpcStatusCodeServerRefused	Request rejected.
502	kRpcStatusCodeStateTransitionFailed	Robot state transition error. Usually it is an illegal state transition or the previous state transition is in progress.

Low level service interface.

The low level service interface is called in a ROS-like Publish/Subscribe manner.

Please note that the low level **publish** interface will only take effect when the robot is in custom mode. To enter this mode, please refer to the `ChangeMode` interface in the high-level API.

Topic	rt/low_state
Effective version	>= v1.0.0.0
Access method	subscribe
Interface description	Obtain the robot's IMU and joint feedback in real time.

Interface structure	struct LowState {
	  ImuState imu_state; // IMU feedback.
	  sequence<MotorState> motor_state_parallel; // Parallel structure joint feedback.
	  sequence<MotorState> motor_state_serial; // Serial structure joint feedback.
	};
	
	struct ImuState {
	  float rpy[3]; // Euler angle information（0 -> roll ,1 -> pitch ,2 -> yaw）
	  float gyro[3]; // Angular velocity information（0 -> x ,1 -> y ,2 -> z）
	  float acc[3]; // Acceleration information.（0 -> x ,1 -> y ,2 -> z）
	};
	
	struct MotorState {
	  float q; // Joint angle position, unit: rad.
	  float dq; // Joint angular velocity, unit: rad/s.
	  float ddq; // Joint angular acceleration, unit: rad/s².
	  float tau_est; // Joint torque, unit: nm
	};

Remarks	Since some simulation tools (such as Isaac Sim) only support serial structures, while our robot has a parallel structure. Therefore, in the interface, we provide two types of joint feedback results for serial and parallel structures. The system will complete the conversion between serial and parallel data internally. Developers can use one of the feedbacks according to their needs, but they need to ensure that the serial-parallel consistency is also maintained when sending underlying instructions.

Topic	rt/joint_ctrl
Effective version	>= v1.0.0.0
Access method	publish
Interface description	Publish the joint commands of the robot to control the motors.

Interface structure	enum CmdType {
	PARALLEL, // Parallel type.
	SERIAL // Serial type.
	};
	
	struct LowCmd {
	CmdType cmd_type; // Set whether the joint command follows the serial mode or the parallel mode.
	sequence<MotorCmd> motor_cmd; // Joint command array.
	};
	struct MotorCmd {
	float q; // Joint angle position, unit: rad.
	float dq; // Joint angular velocity, unit: rad/s.
	float tau; // Joint torque, unit: nm
	float kp; // Proportional coefficient.
	float kd; // Gain coefficient.
	float weight; // Weight, range [0, 1], specify the proportion of user set motor cmd is mixed with the original cmd sent by the internal controller, which is usually used for gradually move to a user custom motor state from internal controlled motor state. Weight 0 means fully controlled by internal controller, weight 1 means fully controlled by user sent cmds. This parameter is not working if in custom mode, as in custom mode, internal controller will send no motor cmds.
	};

Remarks	Since some simulation tools (such as Isaac Sim) only support serial structures, while our robot has a parallel structure. Therefore, in the interface, we simultaneously provide two types of joint commands for serial and parallel structures. The system will complete the conversion between serial and parallel data internally. Developers can use one of the feedbacks according to their needs, but they need to ensure that the serial-parallel consistency is also maintained when receiving underlying feedback.

Topic	rt/booster_hand_data
Effective version	>= v1.1.0.0
Access method	subscribe
Interface description	Subscribe the hand data through low level service interface.

Interface structure	struct HandReplyParam{
	  int32 angle; //Refer to DexterousFingerParameter in high level api
	  int32 force; //Refer to DexterousFingerParameter in high level api
	  int32 current; //Joint temperature [0-1000] mA
	  int32 error; //See remarks
	  int32 status; //See remarks
	  int32 temp; //Joint temperature [0-100] °C
	  int32 seq; //Reference high level DexterousFingerParameter
	};
	
	struct HandReplyData{
	  sequence <HandReplyParam> hand_data;
	  int32 hand_index; / 0：left 1：right
	  int32 hand_type; // 0：gripper 1：dexterous hand
	};

Remarks	status:
	  0:Opening; 1:Closing; 2:Halt on target position; 3:Halt on target force;
	  5:Current Protection; 6:Motor Stall Protection; 7:Motor Error
	error:
	  bit0:Stall Error; bit1:Over Temperature bit2:Over Current bit3:Motor Fault; bit4:Communication Fault

Topic	rt/fall_down
Effective version	>= v1.2.0.2
Access method	subscribe
Interface description	Real-time detection of robot falls

Interface structure	module booster_interface {
	  module msg {
	  enum FallDownStateType {
	  IS_READY, // Not fallen state
	  IS_FALLING, // Currently falling
	  HAS_FALLEN, // Already fallen
	  IS_GETTING_UP, // Currently getting up
	  };
	
	  struct FallDownState {
	  FallDownStateType fall_down_state;
	  boolean is_recovery_available; // Whether recovery (getting up) action is available
	  };
	  };
	};

Remarks	This interface can be used in conjunction with the GetUp (self-righting) interface from the high-level API to enable automatic standing after detecting a fall. Important Notes: 1. The FallDownStateType determination is based on the robot's IMU data. 2. During a fall, sudden collisions may cause the robot to bounce, temporarily elevating the IMU position. - This may trigger 1-2 frames of false IS_GETTING_UP detection 3. Recommended to use in combination with is_recovery_available for accurate state judgment

Topic	rt/remote_controller_state
Effective version	>= v1.2.0.2
Access method	subscribe
Interface description	Real-time retrieval of remote controller button inputs

Interface structure	module booster_interface {
	  module msg {
	  struct RemoteControllerState {
	  unsigned long event; // refer to remarks
	
	  float lx; // left stick horizontal direction, push left to -1, push right to 1
	  float ly; // left stick vertical direction, push front to -1, push back to 1
	  float rx; // right stick horizontal direction, push left to -1, push right to 1
	  float ry; // right stick vertical direction, push front to -1, push back to 1
	
	  boolean a;
	  boolean b;
	  boolean x;
	  boolean y;
	  boolean lb;
	  boolean rb;
	  boolean lt;
	  boolean rt;
	  boolean ls;
	  boolean rs;
	  boolean back;
	  boolean start;
	
	  boolean hat_c; // Hat centered
	  boolean hat_u; // Hat up
	  boolean hat_d; // Hat down
	  boolean hat_l; // Hat left
	  boolean hat_r; // Hat right
	  boolean hat_lu; // Hat left up
	  boolean hat_ld; // Hat left down
	  boolean hat_ru; // Hat right up
	  boolean hat_rd; // Hat right down
	  uint8 reserved;
	  };
	  };
	};

Remarks	This feature can be used in user programs to implement custom gamepad/controller button functionality.
	event include:
	NONE = 0, // no event
	AXIS = 0x600, // axis motion
	HAT = 0x602, // hat position change
	BUTTON_DOWN = 0x603, // button pressed
	BUTTON_UP = 0x604, // button released
	REMOVE = 0x606 // device has been removed

Topic	rt/tf
Effective version	>= v1.1.0.6
Access method	subscribe
Interface description	Real-time acquisition of coordinate transformations between robot joints
Interface structure	https://docs.ros2.org/foxy/api/tf2_msgs/msg/TFMessage.html
Remarks	1. This interface currently only supports subscription via ROS2 2.If using ros2 topic echo /tf fails to display the message, it may be due to an incomplete installation of the xacro package caused by network issues during a software upgrade. You can manually install it with: sudo apt install ros-humble-xacro.Then restart your machine and attempt to subscribe to the topic again.

Vision Interface

- T1 RealSense Version

This version utilizes the **Intel RealSense D455** camera, which includes a dual global shutter IR camera, an RGB camera, and a 6-axis IMU. The camera specifications are as follows:

Features

Use environment:	Indoor/Outdoor
Image sensor technology:	Global Shutter
Ideal Range:	.6 m to 6 m
Inertial measurement unit:	Bosch BMI055

Depth

Depth Technology:	Stereoscopic
Minimum Depth Distance (Min-Z) at Max Resolution:	~52 cm
Depth Accuracy:	<2% at 4 m1
Depth Field of View (FOV):	87° × 58°
Depth Output Resolution:	Up to 1280 × 720
Depth Frame Rate:	Up to 90 fps

RGB

RGB Frame Rate:	30 fps
RGB Frame Resolution:	Up to 1280 × 800
RGB Sensor Technology:	Global Shutter
RGB Resolution and Frame Rate:	1280 × 800 at 30 fps
RGB Sensor FOV (H x V):	90 × 65°
RGB Sensor Resolution:	1 MP

Data Acquisition Methods

[Recommended] Acquire data via ROS

Refer to: https://github.com/IntelRealSense/realsense-ros

- **Note:** On boot, the T1 automatically starts the RealSense ROS node through the system service `booster-daemon-perception.service`. To manually launch the camera node and avoid resource conflicts, stop the service with the following command:

```shell
sudo systemctl stop booster-daemon-perception.service
```

- Topics published by the auto-launched RealSense ROS node:

```shell
/camera/camera/color/camera_info                # Color camera intrinsics
/camera/camera/color/image_raw              # Raw color image
/camera/camera/color/metadata
/camera/camera/depth/camera_info                # Depth camera intrinsics
/camera/camera/depth/image_rect_raw         # Raw depth image
/camera/camera/depth/metadata
/camera/camera/aligned_depth_to_color/camera_info
/camera/camera/aligned_depth_to_color/image_raw         #  Depth aligned to color frame
/camera/camera/extrinsics/depth_to_color              # Depth to color transformation
/camera/camera/extrinsics/depth_to_depth                    # Depth-to-depth transformation
```

- To manually start the RealSense ROS node:

```shell
cd ~/ThirdParty/realsense-ros/
source install/setup.bash
ros2 launch realsense2_camera rs_launch.py
```

- To enable features such as depth-to-RGB alignment and IMU streams, refer to the parameters defined in `rs_launch.py`.
- Sample Python script for subscribing to and saving color and depth images:

```python
# demo.py
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2, os, numpy as np

class ImageSubscriber(Node):
    def __init__(self):
        super().__init__('image_subscriber')
        self.depth_subscription = self.create_subscription(
                                                        Image,
'/camera/camera/aligned_depth_to_color/image_raw',
self.depth_callback, 10)
        self.color_subscription = self.create_subscription(
                        Image,  '/camera/camera/color/image_raw',
self.color_callback, 10)
        self.bridge = CvBridge()
                    self.save_dir   =   os.path.join(os.getcwd(),
f'images_{self.get_clock().now().to_msg().sec}')
        os.makedirs(self.save_dir, exist_ok=True)

    def depth_callback(self, msg):
        self.get_logger().info('Receiving depth image')
                    cv_image   =   self.bridge.imgmsg_to_cv2(msg,
desired_encoding='passthrough')
        depth_image_m = cv_image * 0.001
        timestamp = self.get_clock().now().to_msg().sec
```

```
                    raw_path = os.path.join(self.save_dir,
f'depth_image_raw_{timestamp}.png')
        cv2.imwrite(raw_path, cv_image)
            norm_image = cv2.normalize(depth_image_m, None, 0,
255, cv2.NORM_MINMAX)
                    norm_image = np.clip(norm_image, 0,
255).astype(np.uint8)
                  colormap = cv2.applyColorMap(norm_image,
cv2.COLORMAP_JET)
                  color_path = os.path.join(self.save_dir,
f'depth_image_color_{timestamp}.png')
        cv2.imwrite(color_path, colormap)
        cv2.imshow('Depth Image', colormap)
        cv2.waitKey(1)

    def color_callback(self, msg):
        self.get_logger().info('Receiving color image')
                  cv_image = self.bridge.imgmsg_to_cv2(msg,
desired_encoding='bgr8')
        timestamp = self.get_clock().now().to_msg().sec
                  color_path = os.path.join(self.save_dir,
f'color_image_{timestamp}.png')
        cv2.imwrite(color_path, cv_image)
        cv2.imshow('Color Image', cv_image)
        cv2.waitKey(1)

def main(args=None):
    rclpy.init(args=args)
    image_subscriber = ImageSubscriber()
    rclpy.spin(image_subscriber)
    image_subscriber.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
     main()
```

```shell
#To run the demo script:
source /opt/ros/humble/setup.bash
python demo.py
```

- **Acquire data via SDK**

The RealSense SDK (`librealsense`) is pre-installed on the perception board. Refer to: https://github.com/IntelRealSense/librealsense/tree/master/examples

Before using the SDK, stop the `booster-daemon-perception.service` to avoid resource conflicts.

- T1 Orbbec Version

This version utilizes the **Orbbec Gemini 2 355L**, which includes a dual global shutter IR camera, an RGB camera, and a 6-axis IMU. Camera specifications are as follows:

Camera Specifications	
Model	G40055-170
Depth Technology	Stereo
Wavelength	850nm
Depth Range	*0.17 - 20m+ (Optimal Range: 0.5 - 6m)
Depth Resolution/FPS	Up to 1280X800@30fps
Depth FOV	H90° V65°
RGB Resolution/FPS	Up to 1280X800@60fps
RGB FOV	H94° V68°
Processing	Orbbec ASIC

* Spatial Precision: ≤ 0.8% (1280 x 800 @ 2m & 90% x 90% ROI), ≤ 1.6% (1280 x 800 @ 4m & 80% x 80% ROI)

* Theoretical maximum depth range up to 65 meters

Physical Parameters	
Depth Filter	Visible+NIR-Pass
IMU	Supported
Data Connection	USB 3.0 Type-C
Power Input	USB 3.0 Type-C
Trigger	Supported
Power Consumption	Average < 3.0W
Operating Environment	-10°C - 50°C; Indoor/Outdoor; 5%-90%RH(non-condensing)
SDK Support	Orbbec SDK
Data Output	Point Cloud, Depth Map, IR & RGB
Dimensions (W*H*D)	124mm x 29mm x 27mm
Weight	133g
Installation	Bottom: ¼-20UNC; Back: 2x M4

Data Acquisition Methods

[Recommended] Acquire data via ROS

Refer to: https://github.com/orbbec/OrbbecSDK_ROS2

Note: On boot, the T1 automatically starts the RealSense ROS node via `booster-daemon-perception.service`. To prevent conflicts when launching the Orbbec node, stop the service:

```shell
sudo systemctl stop booster-daemon-perception.service
```

- Topics published by the auto-launched Orbbec ROS node:

```shell
/camera/color/camera_info                    # Color camera intrinsics
/camera/color/image_raw                  # Raw color image
/camera/color/image_raw/compressed
/camera/color/metadata
/camera/depth/camera_info                    # Depth camera intrinsics
/camera/depth/image_raw                  # Raw depth image
/camera/depth/image_raw/compressed
/camera/depth/metadata
/camera/depth_to_color                      # Depth aligned to color frame
```

- To manually start the Orbbec ROS node:

```shell
cd ~/ThirdParty/OrbbecSDK_ROS2
source install/setup.bash
ros2 launch orbbec_camera gemini_330_series.launch.py
```

- To enable additional features such as depth-to-RGB alignment and IMU streams, refer to parameter definitions in `gemini_330_series.launch.py`.
- For Python script to subscribe to and save color and depth images, refer to the sample provided above. Ensure the image topics are updated accordingly.

Acquire data via SDK

The Orbbec SDK is pre-installed on the perception board. Refer to: https://github.com/orbbec/OrbbecSDK

- Before using the SDK, stop the `booster-daemon-perception.service` to avoid resource conflicts

Getting Started

Getting SDK

- Download Link: https://github.com/BoosterRobotics/booster_robotics_sdk
- Follow the README in the repository to complete the SDK installation on the developer's computer.

Install SDK

```shell
# In the booster_robotics_sdk directory
sudo ./install.sh
```

Compile Sample Programs and Install Python SDK

Assuming the SDK is installed at `/home/booster/Workspace`, navigate to the SDK project path and run:

```shell
# BUILD_PYTHON_BINDING=on indicates that the Python SDK will be compiled.
# If you need to compile the Python SDK, you must manually install the following dependencies:
# pip3 install pybind11
# (After installation, if cmake cannot find pybind11, you can manually install it via sudo apt install pybind11-dev)
# pip3 install pybind11-stubgen
# (If pybind11-stubgen is installed under ~/.local,
#       you       need       to       manually       export PATH=/home/[username]/.local/bin:/$PATH)

cd /home/booster/Workspace/booster_robotics_sdk
mkdir build
```

```
cd build
cmake .. -DBUILD_PYTHON_BINDING=on
make
sudo make install
```

Compile Only Sample Programs

Assuming the SDK is installed at `/home/booster/Workspace`, navigate to the SDK project path and run:

```Shell
cd /home/booster/Workspace/booster_robotics_sdk
mkdir build
cd build
cmake ..
make
```

After compiling, the sample programs will be generated in the build directory, including the `b1_loco_example_client` program as a high-level service interface example.

Example: Development Based on Webots Simulation

Preparation Work: Environment Installation

1. System Requirements

Element	Good Spec
OS	Ubuntu 22.04
CPU	Intel Core i7 (7th Generation)
Cores	4
RAM	16 GB

2. Webots Installation

   Voir documentation officielle

   1. Unzip and copy Webots to the `/usr/local` directory.

2. sudo cp -r webots/ /usr/local/
3. Configure the environment variable (the path should match the installation location).

```shell
# ~/.bashrc

export WEBOTS_HOME=/usr/local/webots
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$WEBOTS_HOME/lib/controller
```

3. Install the SDK.
4. Compile and run sample programs (optional).

Development in Webots Simulation Environment

1. Load Webots world files.

   Voir documentation officielle

   1. Unzip Webots simulation files.
   2. Open in Webots:
      1. File -> Open world -> Select `.wbt` file. One is with 4-dof arms, the other one is with 7-dof arms.
2. Install dependency

```shell
 "cmake"
 "ninja-build"
 "libgtest-dev"
 "libgoogle-glog-dev"
 "libboost-dev"
 "libeigen3-dev"
 "liblua5.3-dev"
 "graphviz"
 "libgraphviz-dev"
 "python3-pip"
 "libcurl4-openssl-dev"
 "libsdl2-dev"
 "joystick"
 "libspdlog-dev"
```

3. Run the simulation control program. One is with 4-dof arms, the other one is with 7-dof arms.

   Voir documentation officielle

   Voir documentation officielle

4. In the Shell, ./booster-runner-full-0.0.x.run
    1. If it doesn't run, use `chmod +x booster-runner-full-0.0.x.run`
5. When the simulation is running, this program should be kept open all the time.
6. run sample programs (optional).

Shell

```
# Note: The IP address here should be the local IP. If you are running this on the motion control board, you can use 127.0.0.1. If you are running it on the perception board, you can use 192.168.10.102.
#  Because it is developed on the local machine, the target IP of the dds is 127.0.0.1
./b1_loco_example_client 127.0.0.1
```

Shell

```
# Switch to walking mode
mw
# Control the robot to move forward.
w
# Control the robot to move backward
s
# Control the robot to move left
a
# Control the robot to stop walking.
l
```

Example: Local Environment Development Based on Isaac Simulation Link (ROS2)

Preparation Work: Configure User Computer

1. System Requirements

Element	Minimum Spec
OS	Ubuntu 22.04
CPU	Intel Core i7 (7th Generation)
Cores	4
RAM	32 GB
GPU	GeForce RTX 3070
VRAM	8GB

1. Install Isaac Sim
    1. Install NVIDIA drivers if not previously done.
    2. Download and install Isaac Omniverse.
    3. Install Isaac Sim 4.2 from the EXCHANGE section.
2. Install ROS2.
3. Install the SDK.
4. Compile and run sample programs (optional).

Development in Isaac Simulation Environment ❗

Since the Isaac simulation environment differs significantly from the real environment, it is not recommended to develop motion control actions there. It is better suited for perception/decision-making program development.

1. One-click starts Isaac. One is with 4-dof arms, the others are with 7-dof arms.

    Voir documentation officielle

    Voir documentation officielle

    Voir documentation officielle

2. There are two ways to start; if it doesn't run, use `chmod +x isaac_package_0.0.x.run`
    1. In the GUI, right-click and choose "run as program".
    2. In the Shell, run `./isaac_package_0.0.x.run`
3. If you want to use your own environment, you can run this script as `./isaac_package_0.0.x.run PATH-TO-YOUR-USD.usd`

4. After running, input the local Isaac execution directory; the Isaac environment will start automatically.

After Isaac is installed, the program that executes the script is usually generated in ~/.local/share/ov/pkg/isaac-sim-4.2.0/python.sh

5. Install dependency

```shell
"cmake"
"ninja-build"
"libgtest-dev"
"libgoogle-glog-dev"
"libboost-dev"
"libeigen3-dev"
"liblua5.3-dev"
"graphviz"
"libgraphviz-dev"
"python3-pip"
"libcurl4-openssl-dev"
"libsdl2-dev"
"joystick"
"libspdlog-dev"
```

6. Run the simulation control program. One is with 4-dof arms, the other one is with 7-dof arms

   Voir documentation officielle

   Voir documentation officielle

   a. There are two ways to start; if it doesn't run, use `chmod +x booster-runner-full-0.0.x.run`
      i. In the GUI, right-click and choose "run as program".
      ii. In the Shell, run `./booster-runner-full-0.0.x.run`
   b. After running, select the environment that needs to be run, enter "isaac", and Isaac motion control can be run. When the simulation is running, this program should be kept open all the time.
7. How to list ROS topics
   a. Create fastdds_profile.xml, the content is as follows.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<profiles
xmlns="http://www.eprosima.com/XMLSchemas/fastRTPS_Profiles" >
    <transport_descriptors>
        <transport_descriptor>
            <transport_id>UdpTransport</transport_id>
            <type>UDPv4</type>
            <interfaceWhiteList>
                <address>127.0.0.1</address>

            </interfaceWhiteList>
        </transport_descriptor>
    </transport_descriptors>

            <participant   profile_name="udp_transport_profile"
is_default_profile="true">
        <rtps>
            <userTransports>
                <transport_id>UdpTransport</transport_id>
            </userTransports>
            <useBuiltinTransports>false</useBuiltinTransports>
        </rtps>
    </participant>
</profiles>
```

b. Export environment

```
Shell
export
FASTRTPS_DEFAULT_PROFILES_FILE=PATH_TO/fastdds_profile.xml
ros2 daemon stop
ros2 daemon start
ros2 topic list
```

8. Vision development
    a. Vision topic in Isaac Sim is **camera/camera/color/image_raw**.
    b. Vision depth topic is **camera/camera/depth/image_rect_raw**
    c. Subscribe vision topic to develop vision program.
9. [run sample programs (optional).]

```shell
# Note: The IP address here should be the local IP. If you are running this on the motion control board, you can use 127.0.0.1. If you are running it on the perception board, you can use 192.168.10.102.
# Because it is developed on the local machine, the target IP of the dds is 127.0.0.1
./b1_loco_example_client 127.0.0.1
```

```shell
# Switch to walking mode
mw
# Control the robot to move forward.
w
# Control the robot to move backward
s
# Control the robot to move left
a
# Control the robot to stop walking.
l
```

ROS2 Dev Guide

For the secondary development process based on ROS2, please refer to the following manual: Development Guide on ROS2

RL Training and Deployment Pipeline

RL Training and Deployment

Robot Application Development Example

RoboCup Humanoid 2V2 Demo

Resource Download

```
None
Copyright [2024] [Booster Robotics Technology Co., Ltd ("Booster Robotics")]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

URDF & MESH File

4-Dof arm robot

Serial Model URDF：

Voir documentation officielle

Parallel Model URDF：

Voir documentation officielle

MESH File：

Voir documentation officielle

7-Dof arm robot

Serial Ankle Model URDF：

Voir documentation officielle

Parallel Ankle Model URDF：

Voir documentation officielle

MESH File：

Voir documentation officielle

7-Dof arm robot with gripper

Serial Model URDF：

Voir documentation officielle

Voir documentation officielle

USD

Voir documentation officielle

Robot URDF for RL training

Isaac gym

1. Model include head and arm

   Voir documentation officielle

2. Model with wrist and leg

   Voir documentation officielle

3. Full-body model containing simplified colliders.

   Voir documentation officielle

4. Model with 7-dof arm

   Voir documentation officielle

5. Model with 7-dof arm with arm collision

   Voir documentation officielle

MuJoCo

1. Model with 7-dof arm

   Voir documentation officielle

2. Model with 7-dof arm

   Voir documentation officielle