Atom SDK Application Development Guide

Original Instructions

Issue: V1.0

Date: 2025.09.26

SHENZHEN DOBOT CORP LTD

Copyright © SHENZHEN DOBOT CORP LTD 2025. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without the prior written consent of SHENZHEN DOBOT CORP LTD (hereinafter referred to as "Dobot").

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Dobot makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Dobot be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robot is used on the premise of fully understanding the robot and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related instructions, damages or losses may happen in the using process. Dobot shall not be considered as a guarantee regarding all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robot.

SHENZHEN DOBOT CORP LTD

Address: Room 1003, Building 2, Chongwen Garden, Nanshan iPark, Liuxian Blvd, Nanshan District, Shenzhen, Guangdong Province, China

Website: www.dobot-robots.com

Preface

Purpose

This document introduces the product appearance, electrical interfaces, basic operations, SDK configuration, service interfaces, and reference examples of the Dobot Atom series robot, facilitating users to perform secondary development based on actual application scenarios.

Intended audience

This document is intended for:

- Customer
- Sales Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Revision history

Date	Version	Revision history
2025/09/26	V1.0	The first release

Symbol conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury.
WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robot arm damage.
NOTICE	Indicates a potentially hazardous situation which, if not avoided, could result in robot arm damage, data loss, or unanticipated result.
NOTE	Provides additional information to emphasize or supplement important points in the main text.

 Copyright © SHENZHEN DOBOT CORP LTD

Content

 Copyright © SHENZHEN DOBOT CORP LTD

 Copyright © SHENZHEN DOBOT CORP LTD

1. Overview

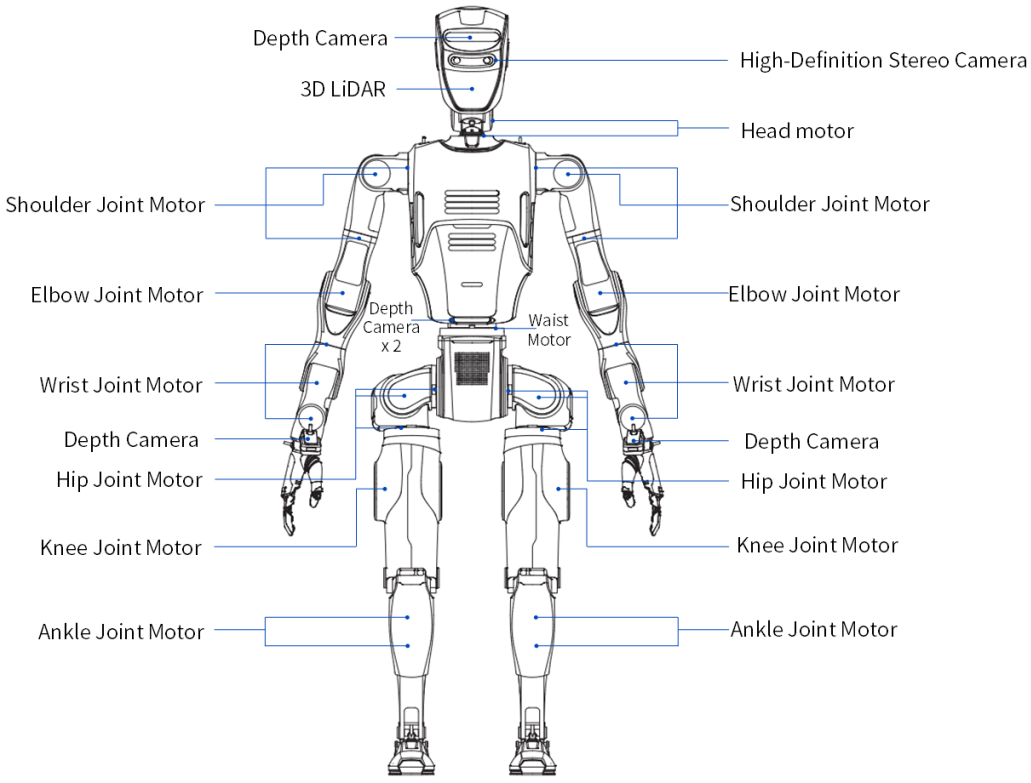
The Dobot ATOM series is the world's first "Dexterous Manipulation + Straight-Knee Walking" embodied AI humanoid robot, independently developed by Dobot. The robot features a full joint hollow routing and a 1:1 full-scale humanoid arm design. It is equipped with a 7-DOF industrial-grade bionic collaborative arm (±0.05mm repeat positioning accuracy), a synchronized humanoid head, and a five-fingered dexterous hand. Powered by embodied AI operation models and AI edge computing, it enables “collection, training, and inference” for specific application scenarios.

The Dobot ATOM robot is ideal for continuous and repetitive tasks in scenarios where equipment positions are not fixed, products vary in specifications, operations involve a high degree of similarity, and there is a need to navigate through short, confined spaces with agile turning capabilities. Examples include industrial tasks like automotive assembly line material handling, beverage preparation in coffee shops, and pharmacy nighttime dispensing.

1.1 Product Specifications

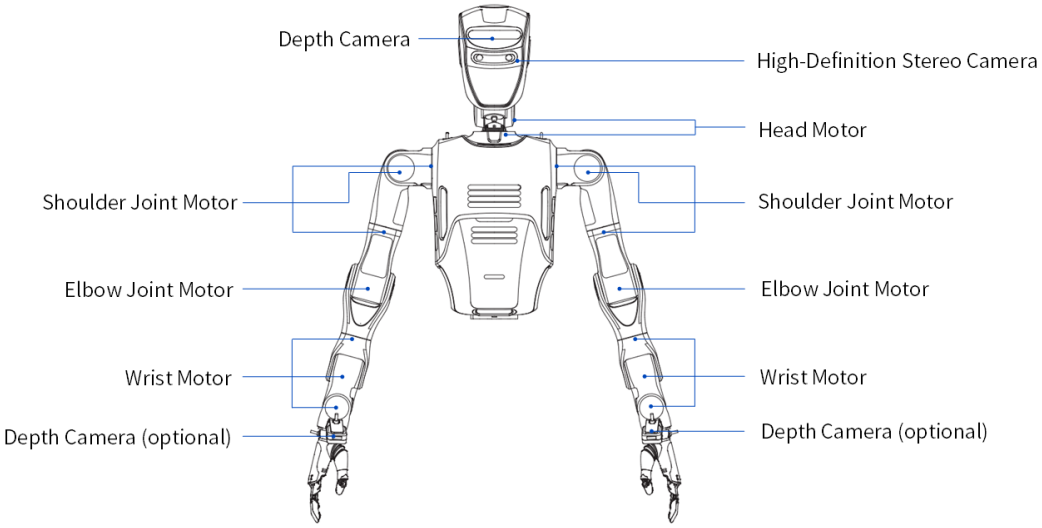
1.1.1 Product Appearance

DOBOT Atom - Max / DOBOT Atom - Trainer



 Copyright © SHENZHEN DOBOT CORP LTD

DOBOT Atom D - Data Collection



1.1.2 Technical specifications

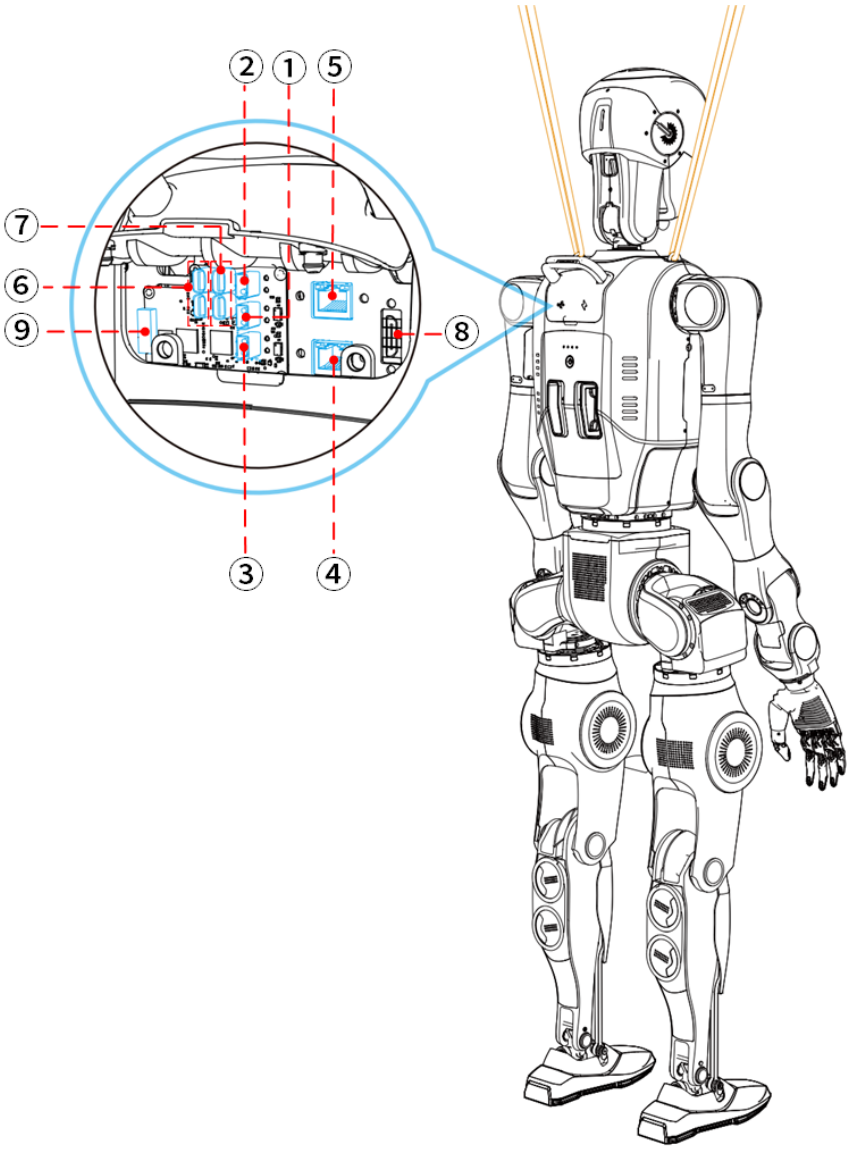
Product model	DOBOT Atom - Max	DOBOT Atom - Trainer	DOBOT Atom D - Data Collection
Height	About 1650mm	About 1650mm	About 650mm
Weight (Without dexterous hand/gripper)	About 62kg	About 62kg	About 20kg
Total Degrees of Freedom (Without dexterous hand/gripper)	29	29	16
Head Degrees of Freedom	2	2	2
Single Arm Degrees of Freedom (Without dexterous hand/gripper)	7	7	7
Waist Degrees of Freedom	1	1	0
Single Leg Degrees of Freedom	6	6	0
Arm's Reach (Without dexterous hand/gripper)	600 mm	600 mm	600 mm
Single Arm Weight (Without dexterous hand/gripper)	About 6.8kg	About 6.8kg	About 6.5kg
Single Arm Payload (Without dexterous hand/gripper)	3.5 kg	3.5 kg	3.5 kg
Single Arm Repeated Positioning Accuracy	±0.05mm	±0.05mm	±0.05mm
Maximum End Speed of the Arm	1.5 m/s	1.5 m/s	1.5 m/s
Audio Equipment	360° Omnidirectional Microphone x 1 Neodymium Speakers x 2	360° Omnidirectional Microphone x 1 Neodymium Speakers x 2	-
Maximum Walking Speed	1.5 m/s	1.5 m/s	-
Full Joint Hollow	Yes	Yes	Yes

 Copyright © SHENZHEN DOBOT CORP LTD

Routing			
Basic Computing Power	Intel i5	Intel i5	Intel i5
High Computing Power Module 1500 TOPS	Intel i9 (24 Cores 32 Threads) High-Memory Dedicated Graphics Card (FP32 GPU Computing Power: 41.15 TFLOPS)	Intel i9 (24 Cores 32 Threads) High-Memory Dedicated Graphics Card (FP32 GPU Computing Power: 41.15 TFLOPS)	-
Battery Life	About 2h	About 2h	About 2h
Charging	About 2h	About 2h	About 2h
End Effector	6-DoF Dexterous Hand x 2	0	-
Head Sensor	Depth Camera x 1 High-Definition Stereo Camera x 1	Depth Camera x 1 High-Definition Stereo Camera x 1	Depth Camera x 1 High-Definition Stereo Camera x 1
Wrist Sensor	Depth Camera x 2	Depth Camera x 2	-
Waist sensor	Depth Camera x 2	Depth Camera x 2	-
Head LiDAR	3D LiDAR × 1	3D LiDAR × 1	-
Secondary Development	Support	Support	Support
Warranty	1 year	1 year	1 year

 Copyright © SHENZHEN DOBOT CORP LTD

1.2 Electrical Interface



 Copyright © SHENZHEN DOBOT CORP LTD

No.	Interface Type	Quantity	Abbreviation	Description
①	XT30	1	VBAT	8A battery power output (directly connected to the battery power here), can also be directly connected to a charger for user convenience.
②	XT30	1	19V	Reserved, 19V/2A power output.
③	XT30	1	5V	5V/2A power output.
④	RJ45	1	1000 BASE-T	Gigabit Ethernet (connected to a switch); connect this port when you need to use PC1's API for secondary development.
⑤	RJ45	1	1000 BASE-T	Gigabit Ethernet (connected to PC2); connect this port for stable high-speed wired network when doing secondary development via PC2.
⑥	Type-C (Reinforced)	2	Type-C	Supports USB2.0 host, 5V/0.5A power output (connected to PC1, expanded to 2 ports via hub chip), external docking station can be connected for peripherals such as keyboard, mouse, etc.
⑦	Type-C (Reinforced)	2	Type-C	Supports USB2.0 host, 5V/0.5A power output (connected to PC2, expanded to 2 ports via hub chip), external docking station can be connected for peripherals such as keyboard, mouse, etc.
⑧	HDMI	1	/	Reserved, connected to PC2 for easy external monitor connection.
⑨	GPIO★	3	DI	12V – 24V digital input interface.
		3	DO	19V/0.1A digital output interface.

★ The functions corresponding to each pin number of the GPIO module are as follows:

Pin Number	Function	Pin Number	Function
1	DO1	7	DI1
2	GND	8	GND
3	DO2	9	DI2
4	GND	10	GND
5	DO3	11	DI3
6	GND	12	GND

 Copyright © SHENZHEN DOBOT CORP LTD

1.3 Debugging Interface

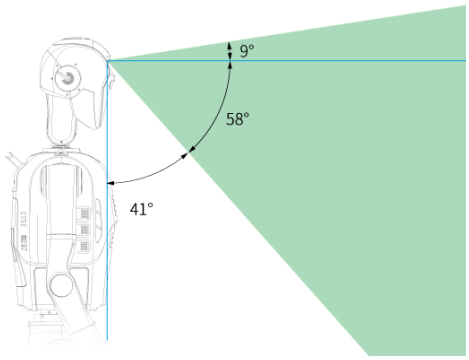
PC1: Dedicated for Dobot Atom motion control program and not available for external development.

PC2: Developers can only use PC2 for secondary development. The IP address for PC2 is 192.168.8.13, and the port is 3390. Please contact technical support to obtain the initial username and password.

1.4 Camera and Radar Field of View

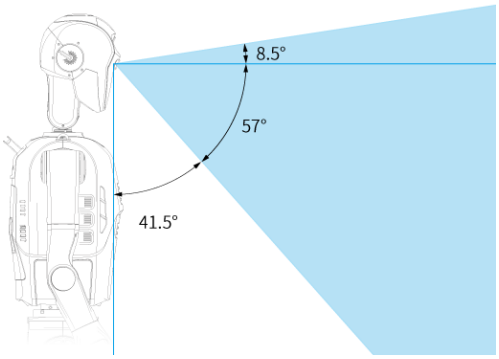
Head Depth Camera Field of View

The DOBOT ATOM - Max / DOBOT ATOM - Trainer / DOBOT ATOM - Data Collection are equipped with a head depth camera, enabling the robot to perceive and understand its surroundings with greater accuracy. The head depth camera has a field of view as shown in the figure below:



Head High-Definition Stereo Camera Field of View

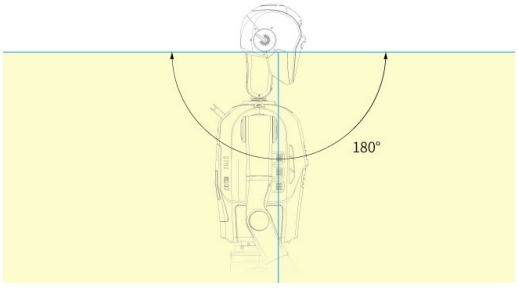
The DOBOT ATOM - Max / DOBOT ATOM - Trainer / DOBOT ATOM - Data Collection are equipped with a head high-definition stereo camera, combined with a depth camera, enhancing the robot's spatial perception and obstacle detection capabilities. The head high-definition stereo camera has a field of view as shown in the figure below:



Head LiDAR Field of View

The DOBOT ATOM - Max / DOBOT ATOM - Trainer are equipped with head LiDAR, enabling real-time acquisition of precise environmental data and providing the robot with advanced environmental perception capabilities. The head LiDAR has a field of view as shown in the figure below:

 Copyright © SHENZHEN DOBOT CORP LTD



1.5 Joint Order Names and Joint Limits

Joint Number	Joint Name	Limits (radians)
00 Left leg part Corresponds to single arm axis 1 (Increasing sequentially downwards)	left_hip_pitch_joint	[-2.5656, 3.0194]
01	left_hip_roll_joint	[-0.2618, 1.6581]
02	left_hip_yaw_joint	[-2.7925, 2.7925]
03	left_knee_joint	[0, 2.4871]
04	left_ankle_pitch_joint	[-0.5236, 0.8727]
05	left_ankle_roll_joint	[-0.4800, 0.4800]
06 Right leg part Corresponds to single arm axis 1 (Increasing sequentially downwards)	right_hip_pitch_joint	[-2.5656, 3.0194]
07	right_hip_roll_joint	[-1.6581, 0.2618]
08	right_hip_yaw_joint	[-2.7925, 2.7925]
09	right_knee_joint	[0, 2.4871]
10	right_ankle_pitch_joint	[-0.5236, 0.8727]
11	right_ankle_roll_joint	[-0.4800, 0.4800]
12 Torso part	torso_joint	[-2.8448, 2.8448]
13	left_shoulder_pitch_joint	[-2.97, 2.97]

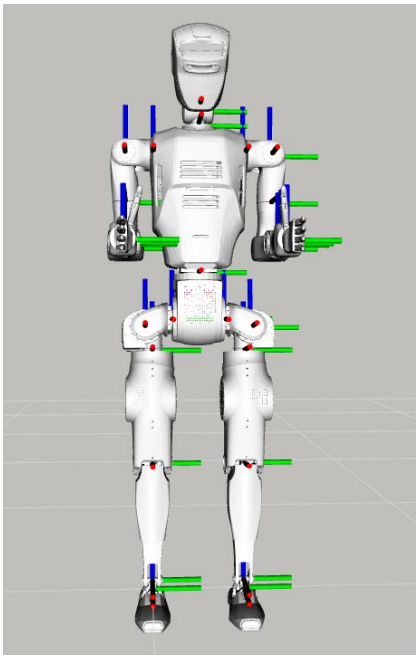
 Copyright © SHENZHEN DOBOT CORP LTD

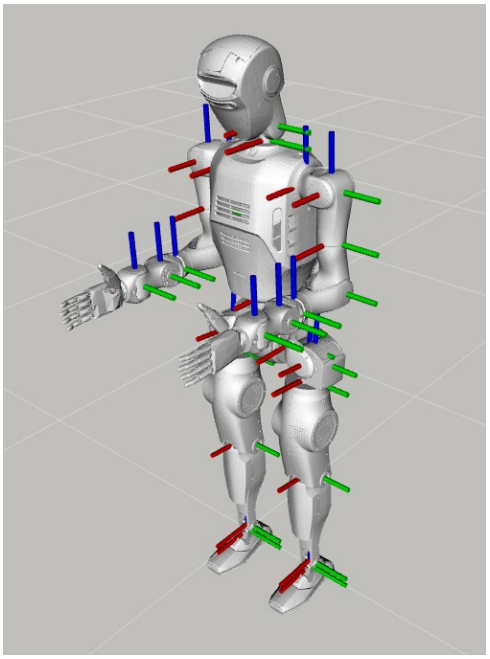
Left leg part Corresponds to single arm axis 1 (Increasing sequentially downwards)		
14	left_shoulder_roll_joint	[-0.43, 3.14]
15	left_shoulder_yaw_joint	[-2.97, 2.97]
16	left_elbow_pitch_joint	[-0.87, 3.14]
17	left_elbow_roll_joint	[-2.97, 2.97]
18	left_wrist_pitch_joint	[-1.57, 1.57]
19	left_wrist_roll_joint	[-1.57, 1.57]
20 Right leg part Corresponds to single arm axis 1 (Increasing sequentially downwards)	right_shoulder_pitch_joint	[-2.97, 2.97]
21	right_shoulder_roll_joint	[-0.43, 3.14]
22	right_shoulder_yaw_joint	[-2.97, 2.97]
23	right_elbow_pitch_joint	[-0.87, 3.14]
24	right_elbow_roll_joint	[-2.97, 2.97]
25	right_wrist_pitch_joint	[-1.57, 1.57]
26	right_wrist_roll_joint	[-1.57, 1.57]
27 Head	head_yaw_joint	[-1.57, 1.57]
28	head_pitch_joint	[-1.57, 0.6981]

 Copyright © SHENZHEN DOBOT CORP LTD

1.6 Coordinate Systems

When all joints are at zero degrees, the coordinate systems for each joint are as shown in the following figures:





 Copyright © SHENZHEN DOBOT CORP LTD

2. Operation Instructions

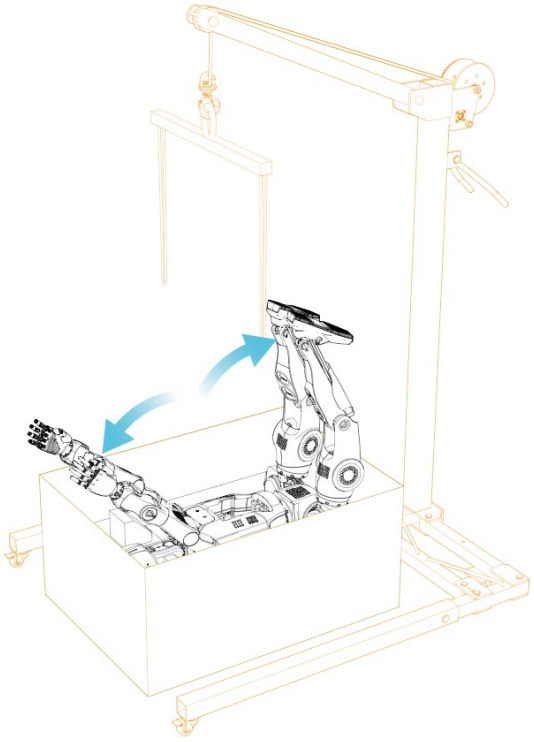
2.1 Basic Operations

2.1.1 Suspension before Powering On

1. Robot Preparation

Position the packaging box correctly beneath the hoist, Then, extend the robot's arms and legs outward.

> **NOTICE**
>
> During the entire hoisting process, the robot's arms and legs must be held by personnel to prevent impact or damage.

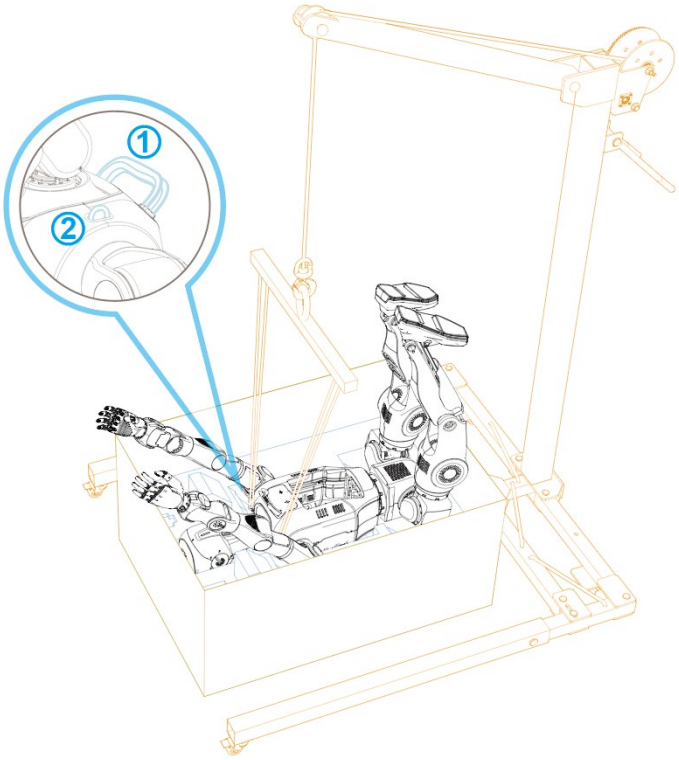


2. Suspension Preparation

① Grab the fixed ring on the back of the robot to slightly lift it away from the inner lining of the packaging box.

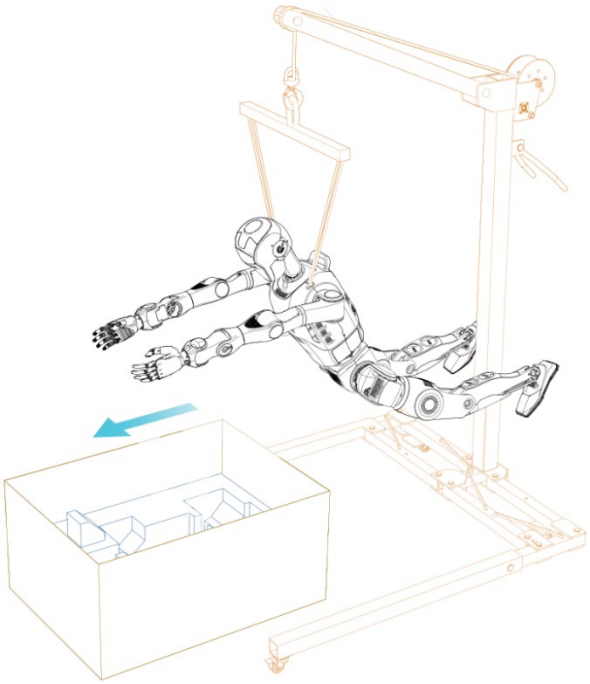
② Attach the hoist hook to the fixed rings on both shoulders of the robot.

 Copyright © SHENZHEN DOBOT CORP LTD



3. Raise the Robot

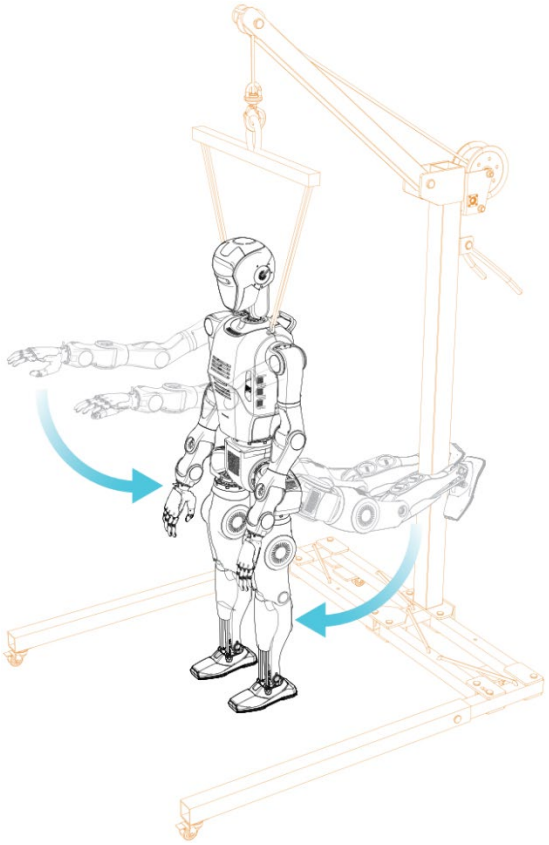
Use the hoist to slowly raise the robot, and once the robot is out of the packaging box, remove the packaging box.



 Copyright © SHENZHEN DOBOT CORP LTD

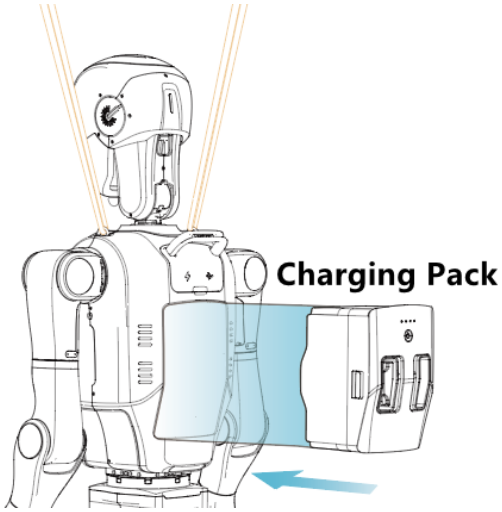
4. Straighten the Robot

As the hoist raises the robot, use your hands to support the robot's arms and legs to slowly lower it down, until the robot's two feet are close to the ground but not touching it (in a suspended state). Ensure that the robot's arms and legs are naturally positioned, with no entanglement in the joints.



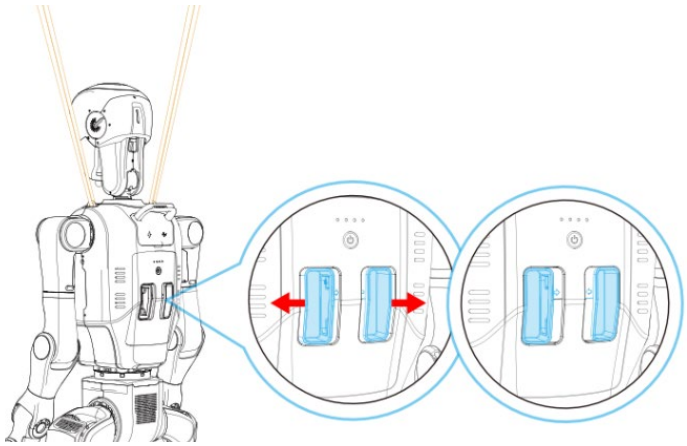
5. Install the Battery Pack

① Hold the battery pack with one hand, pinch the latch in the middle with the other hand, and push the fully charged battery pack into the battery slot on the back of the robot.



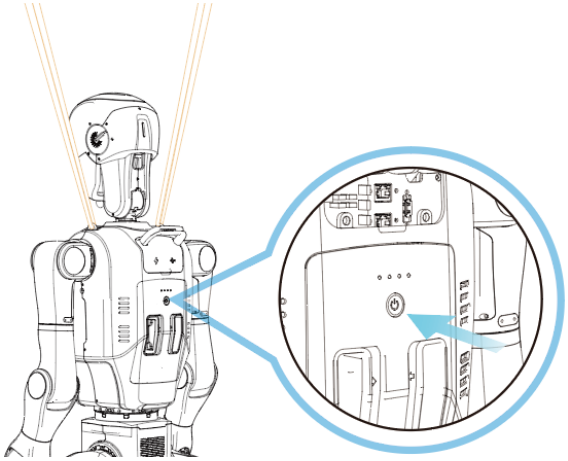
 Copyright © SHENZHEN DOBOT CORP LTD

② Gently push latches to both sides to ensure they are securely fixed in place. After installation, lightly pull the battery pack to ensure there is no looseness.



6. Activate the battery pack

Before powering on the battery pack, please adjust the robot's head to a straight position (level and facing forward). Then short-press the battery pack's power button once, and long-press for 3 seconds. The four indicator lights illuminate in sequence and then flash, indicating that the battery pack has been successfully activated.



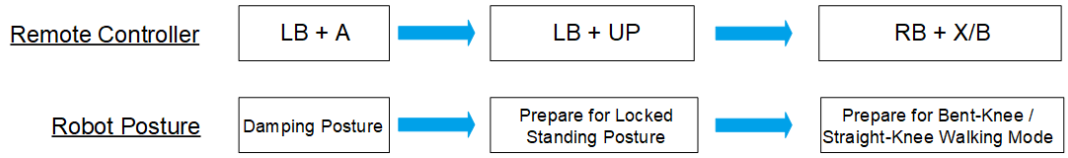
NOTICE

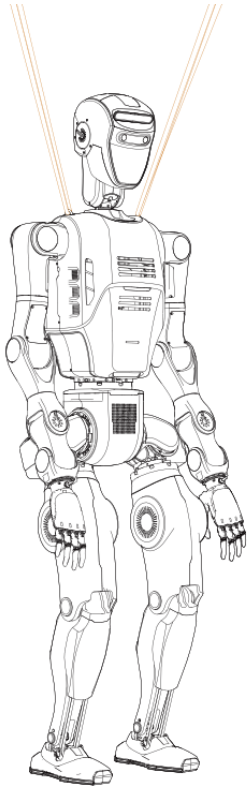
If the robot's head is not properly positioned (e.g., the head is drooping), powering on may trigger a limit alarm.

7. Start the robot

After the robot passes the self-check, it enters the zero-torque state. At this point, you can connect to the App to check if the robot has any errors. If there are no errors, you can connect the remote controller and switch the robot's mode according to the following procedure.

 Copyright © SHENZHEN DOBOT CORP LTD





Prepare for Bent-Knee Walking Mode

NOTICE

After entering the Prepare for Bent-Knee / Straight-Knee Walking Mode, the robot will interpolate within 10 seconds to transition to the default joint angles of the bent-knee model or straight-knee model (non-operable). During this process, the robot needs to be hoisted to suspend in the air or have personnel hold it to prevent it from falling.

8. Lower the Suspension Rope

Make ATOM's feet touch the ground, press the RB+A button on the remote controller to enter the Bent-Knee / Straight-Knee Walking Mode (the specific walking mode entered depends on the prepared mode in the previous step).

NOTICE

During the descent of the suspension rope, personnel must hold onto Dobot ATOM to ensure it remains upright and stable, with the suspension rope slack and free of tension.

 Copyright © SHENZHEN DOBOT CORP LTD

9. Release the Suspension Rope

Once Dobot ATOM's movement is confirmed normal, untie the rope and control the robot via remote controller.

> **NOTICE**
>
> In case of an emergency, press LB+A to put the robot into damping state, and the robot will slowly fall to the ground.

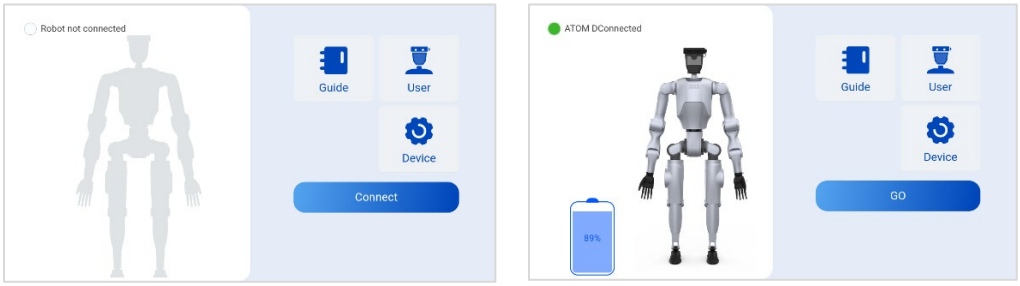
2.1.2 Connect to Dobot Ex app

1. Please contact Dobot technical support to obtain the **Dobot Ex** App installation package, download and install it.
2. Double-click to open the Dobot Ex App and log in using the enterprise account and password provided by the sales.

> **NOTE**
>
> If you do not have an enterprise account, please contact DOBOT sales service personnel to obtain one.

3. Power on Dobot ATOM by following the Suspension before Powering On procedure.
4. Turn on the phone's WiFi switch and connect to the WiFi named "ATOM+SN code" (default password: 12345678).
5. On the initial interface of the Dobot Ex App, click the "Connect" button. When the interface displays the ATOM appearance and battery level, it indicates that the connection has been completed.

 Copyright © SHENZHEN DOBOT CORP LTD

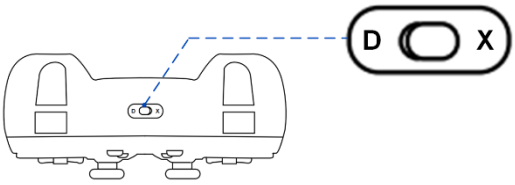


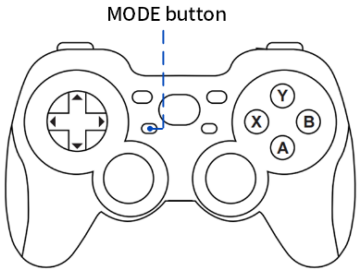
Robot not connected　　Robot connected

6. Once the robot is connected, you can access the "Device" page to review information about the robot, including any alarm details. The robot can only switch to motion mode if there are currently no errors.

2.2 Connect to Remote Controller

Before using the remote controller, switch the top button to the "**X**" mode, and press the **MODE** button to turn off the indicator light. The Dobot ATOM robot will automatically connect to the remote controller after powering on. You can control the robot's motion direction and motion state using the joystick and buttons on the remote controller.





NOTICE

- The remote controller must be strictly set to "X" mode. After switching mode, the robot needs to be powered on again.
- During the use of the remote controller, the MODE indicator light must always remain off.

2.2.1 Suspension before Powering Off

1. Before powering off, please make sure to rehang the ATOM onto the protective frame and secure it with a rope to keep it suspended. Ensure the robot is in the initial position after a successful start, with no arm operations and standing still.
2. Press the **LB+A** button on the remote controller to control the robot into damping state.
3. First, short-press the battery pack's power button once, then long-press for 3 seconds to power it off (the four power indicators will dim one by one).

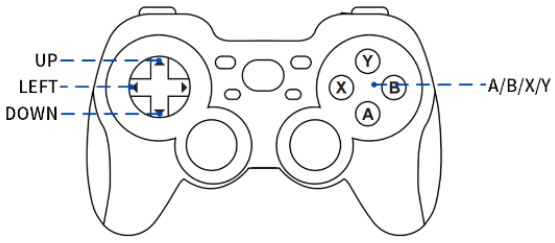
NOTICE

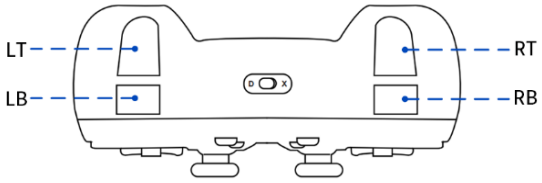
- If the robot is not used for a long time, please remove the battery pack in time.
- Ensure the robot is suspended on the protective frame and shut down in a damping state; otherwise, the robot may fall to the ground after powering off, which could pose a safety hazard.
- Be careful of the robot joints pinching your hand.

2.3 Remote Controller Instructions

2.3.1 Mode Concepts and Button Descriptions

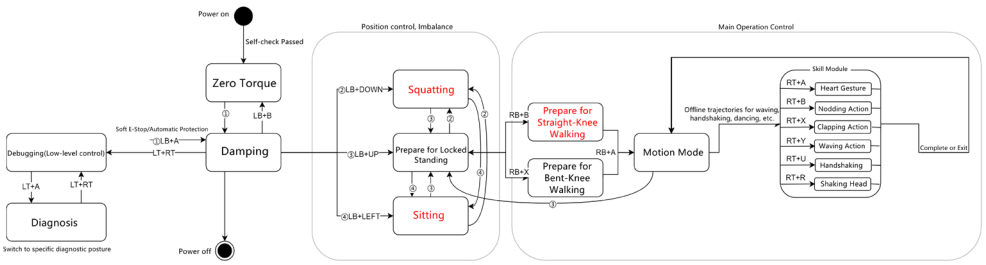
The robot's states can be switched using the following combined buttons on the remote controller.





Button	Robot status	Description
LB+B	Zero Torque State	All robot motors will stop active movement, and there is no damping sensation during swinging.
LB+A	Damping State	All robot motors will stop active movement, and there is a noticeable damping sensation during swinging.
LT+RT	Debugging State	Development and debugging based on the underlying control SDK. Acquire real-time operation data (battery, motors, IMU, etc.) and directly drive the motors.
LT+A	Diagnostic State	Controls joint positions to switch the robot to a specific diagnostic posture. Press **LT+B** to switch from diagnostic mode to debugging mode.
LB+UP	Prepare for Locked Standing Posture	The robot slowly assumes the prepared posture before the motion mode within 5 seconds (without balance control).
LB+DOWN	Squatting state	The robot slowly assumes a squatting posture within 5 seconds (without balance control).
LB+LEFT	Sitting State	The robot slowly assumes a sitting posture within 5 seconds (without balance control).
RB+B	Prepare for Straight-Knee Walking	The robot slowly assumes a prepared posture for straight-knee walking within 5 seconds (without balance control).
RB+X	Prepare for Bent-Knee Walking	The robot slowly assumes a prepared posture for bent-knee walking within 5 seconds (without balance control).
RB+X → RB+A	Bent-Knee Walking Mode	First press **RB+X** to control the robot into a prepared posture for bent-knee walking, then press **RB+A** to enter the bent-knee walking mode, to be used as a prepared posture before the motion state and skill state.
RB+B → RB+A	Straight-Knee Walking Mode	First, press **RB+B** to control the robot into the prepared posture before straight-knee walking, then press **RB+A** to enter the straight-knee walking mode, to be used as a prepared posture for motion state and skill state.

2.3.2 Mode Switching Logic



NOTE

- The feature marked in red is under development.
- Only standing with straight knees and standing with bent knees can enter operation control. Before entering operation control, the robot needs to be placed on the ground with the body level, and then switch to straight-knee walking or bent-knee walking mode using the remote controller.

2.4 Voice Assistant Description

The Dobot Atom Voice Assistant supports a hybrid architecture of local and cloud modes, with local mode being the default. It allows customers to enhance their own knowledge base on top of the built-in language model to enable dialogues for specific scenarios.

NOTICE

- The voice assistant is only supported by the Dobot Atom - Max and the Dobot Atom - Trainer.
- The Dobot Atom Voice Assistant currently supports only local mode (cloud mode is in development).

2.4.1 Turning On and Off

The Dobot Atom Voice Assistant feature is disabled by default. To enable it, first connect robot to App, and then follow this path:

[Device] > [Voice Assistant] > [On/Off]

2.4.2 Mode Description

Dobot Atom supports two conversation modes: **Voice-Triggered Dialogue** and **Button-Triggered Dialogue**.

Switching Method

APP: [Device] > [Voice Assistant] > [Voice-Triggered Dialogue / Button-Triggered Dialogue]

Voice-Triggered Dialogue

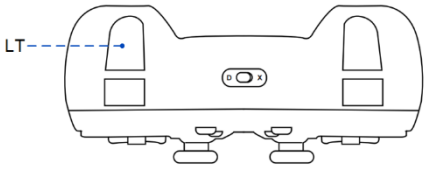
Wake Word	Hi Atom
Feature	Once the robot is activated, it can engage in multi-turn natural conversations. This mode is suitable for relatively quiet and ordinary conversational settings.
Introduction	Once successfully activated with a wake word, the robot will respond to questions and can then engage in unlimited multi-turn conversations. The conversation

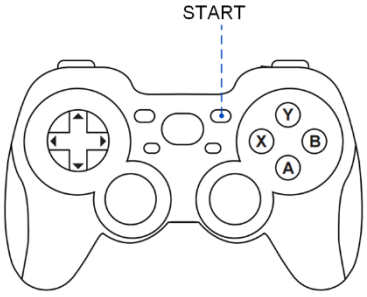
 Copyright © SHENZHEN DOBOT CORP LTD

	ends if there is no sound for more than 15 seconds, and it needs to be reactivated for further use (up to 5 wake words can be set in the App).

Button-Triggered Dialogue

Feature	Force a voice recognition attempt, suitable for noisy environments.
Introduction	There is no need for a wake word; simply press and hold the LT+Start buttons on the remote controller (as indicated in the diagram) to begin recording. Release the LT+Start buttons to stop recording. The sounds generated during this phase will be forcibly recognized and sent to the voice recognition framework.



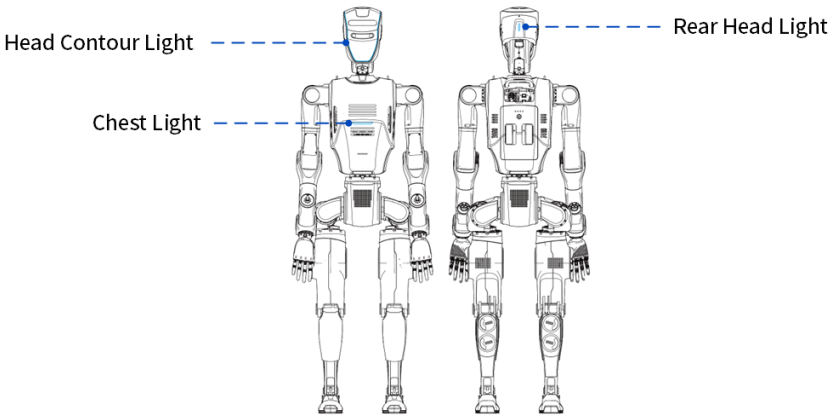


How to Improve Recognition Success Rate

Standing directly in front of the robot (facing the microphone opening) at a distance of within 1 meter.

 Copyright © SHENZHEN DOBOT CORP LTD

2.5 All Light Signals on the Robot



2.5.1 Chest Light and Rear Head Light

Light Status	Device Status
Off	The device is not powered on, or there is an issue with the power distribution board
Slow blue blinking	The device is in the process of starting up.
Solid blue	The device has successfully started, and all systems are functioning normally
Fast red blinking	Disconnection of CAN board or servo. Please contact technical support to troubleshoot the issue.
Slow red blinking	Other than CAN board or servo disconnection, alerts such as servo over-limit error, BMS failure, or enablement failure require contacting technical support for troubleshooting.

2.5.2 Head Contour Light

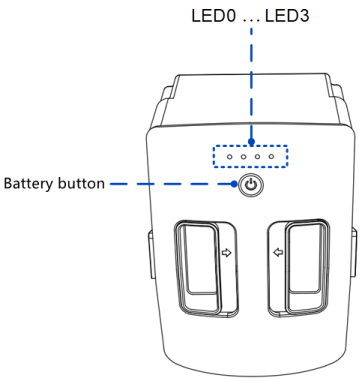
Light Status	Device Status
Blue light is breathing (bright > dim > bright)	Voice module is recording
Solid blue	Voice module is in a non-recording state

 Copyright © SHENZHEN DOBOT CORP LTD

2.5.3 Battery Light Indications

NOTE

- The battery indicator lights are labeled LED0 to LED3 from left to right, with LED0 on the far left and LED3 on the far right.
- Battery indicator icon meanings: Off ○, Slow flash ◎.



Battery indicator light				Alarm category	Alarm description
LED0	LED1	LED2	LED3		
◎	○	○	○	Voltage-related issues	Overvoltage/undervoltage in cells/battery packs, battery failure
○	◎	○	○	Current-related issues	Overcurrent/short circuit during charging/discharging
○	○	◎	○	Temperature-related issues	Abnormal temperature in cell NTC, PCB NTC, AFE chip, etc.
○	○	○	◎	Other issues	Open circuit, cell mismatch, FET driver issues, sensor anomalies, BMS system errors

 Copyright © SHENZHEN DOBOT CORP LTD

3. Application Development

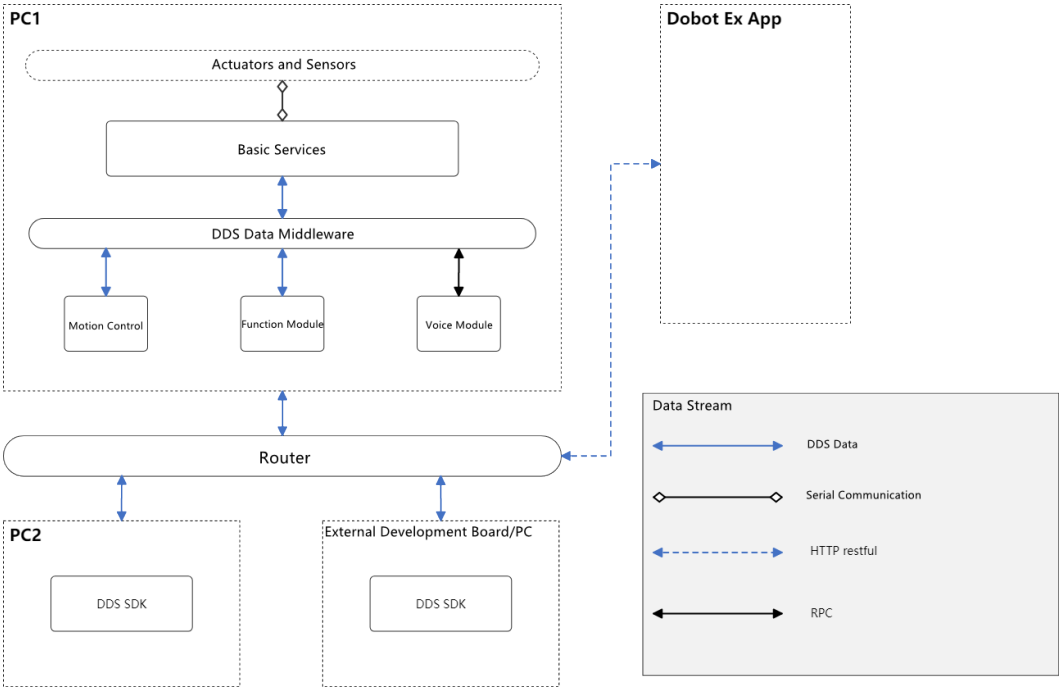
3.1 SDK Overview

The Dobot Atom SDK is an integrated development kit designed to support developers in conducting secondary development based on the Atom robot.

The SDK uses DDS as its messaging middleware, consistent with the communication mechanism used in ROS2, allowing seamless communication between the two systems. Consequently, developers can perform secondary development on the Atom robot using either the ROS2 framework or the Dobot Atom SDK.

- **Subscribe/Publish:** The receiver subscribes to a specific message, and the sender dispatches messages to the receiver based on the subscription list. This model is primarily used for medium to high frequency or continuous data interactions.
- **Request/Response:** This is a query mode used to obtain data or execute operations through requests. It is utilized for low-frequency interactions or data interactions when switching functions.

3.2 Architecture Description



Atom Robot

- Communication between various functional modules is primarily implemented using DDS.
- Data from sensors such as motors and radars are collected via serial ports and then forwarded to the DDS middleware.
- The Dobot Atom - Max and Dobot Atom - Trainer robots come with two built-in computing units: PC1 and PC2. PC1 is dedicated to the Atom robot's motion control program and is not open for external access. Developers can only use PC2 for secondary development. The IP address for PC2 is 192.168.8.13, and the port is 3390. Please contact technical support to obtain the initial username and password.

Dobot Ex App

- User accounts for the Dobot Ex App are created on the Dobot backend and are provided to customers individually.
- The Dobot Ex App connects to the Atom robot via WiFi, enabling users to view the status of the Atom robot or modify certain configurations through the App.

3.3 Development Language

Supports development in C++ language.

3.4 SDK Acquisition

3.4.1 SDK Download Address

https://github.com/Dobot-Arm/dobot_atom_sdk

3.4.2 URDF Download Address

https://github.com/Dobot-Arm/dobot_atom_urdf

3.4.3 ROS2 Download Address

https://github.com/Dobot-Arm/dobot_atom_ros2

3.5 Development Environment Description and Configuration

3.5.1 Environment Dependencies

System Environment

Development is recommended under Ubuntu 20.04 and is not currently supported on Mac or Windows systems. PC1 runs official services and does not support development; only PC2 is accessible for development purposes.

- **OS:** Ubuntu 20.04 LTS
- **Compiler:** GCC 9.4.0

 Copyright © SHENZHEN DOBOT CORP LTD

Network Environment

Connect the user's computer and the Atom switch to the same network. It is recommended that new users connect the computer to the Atom switch using an Ethernet cable and adapter and set the network card communicating with the robot to 192.168.8. Experienced users can configure the network environment themselves.

3.5.2 Installation and Compilation Examples

SDK Installation

```
mkdir build
cd build
cmake ..
make
```

3.5.3 Compiling the Built-in Examples

NOTICE

When running these examples, the robot will move. Please ensure your environment is safe before proceeding with any operations.

High-Level RPC Example

```
cd build
./rpc_test
```

Low-Level DDS Example

NOTICE

When running these examples, please ensure the robot is switched to debugging mode.

```
cd build
./bridge_test
```

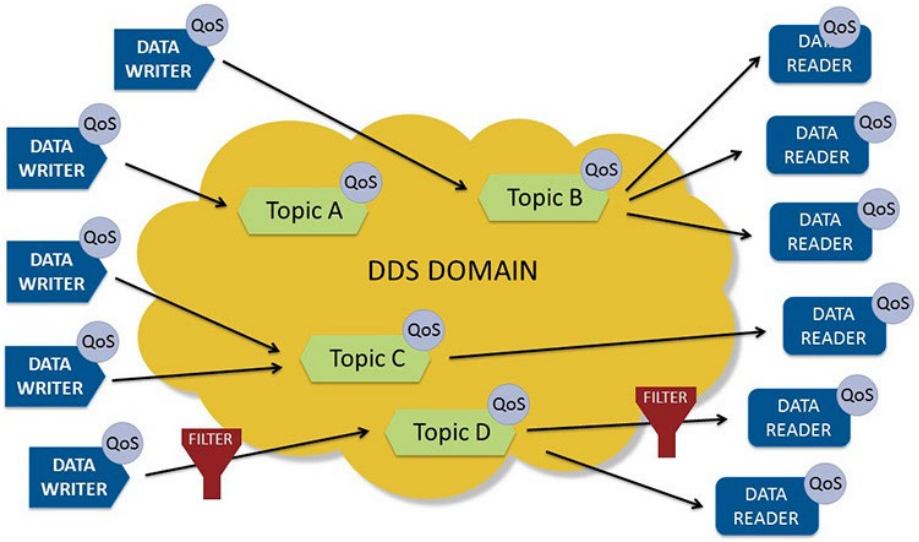
 Copyright © SHENZHEN DOBOT CORP LTD

4. Software Service Interface

4.1 DDS Communication Interface

4.1.1 Introduction

DDS (short for Data Distribution Service), is a middleware and a distributed communication standard published by OMG. It uses a publish/subscribe model and offers various Quality of Service (QoS) policies to ensure the real-time, efficient, and flexible distribution of data. This capability meets the demands of various distributed real-time communication applications.



This document provides the interface specifications for Dobot Atom's interaction using DDS. Since DDS interfaces are defined through IDL files and serialized for generation, all interfaces in this document are defined by IDL files.

All topics that need to be published in real-time start with **rt/**.

Due to the diverse product forms of the Dobot Atom, it employs a separate control scheme for the upper and lower body, meaning the upper and lower limbs are controlled using different topics.

Upper Limb Control: Waist, left arm, right arm, head

Lower Limb Control: left leg, right leg.

The "**underlying layer**" mentioned herein refers to the programs that manage joints, BMS (Battery Management System), IMU (Inertial Measurement Unit), and controllers. If personnel are familiar with reinforcement learning or model-based theories and can program using related technologies, the interfaces described herein can provide direct communication with this low-level program, enabling higher-level control of the Dobot Atom.

The algorithm state machine is a program built into PC1, which, under certain conditions, also controls the upper and lower limbs. For detailed information, see the State Machine ID section.

Dexterous Hand Array Sorting: Right hand (thumb flexion, index finger, middle finger, ring finger, little finger, thumb rotation), left hand (thumb flexion, index finger, middle finger, ring finger, little finger, thumb rotation).

 Copyright © SHENZHEN DOBOT CORP LTD

4.1.2 Shared Sub-IDL File

Joint Information: motor_state.idl

```
cat motor_state.idl
struct MotorState_ {
uint8 mode;                 // Mode
float q;                    // Angular Position
float dq;                   // Angular Velocity
float ddq;                  // Angular Acceleration
float tau_est;              // Estimated Torque
float q_raw;                // Raw Angular Position
float dq_raw;               // Raw Angular Velocity
float ddq_raw;              // Raw Angular Acceleration
uint8 mcuTemp;              // Servo Control Board Temperature
uint8 mosTemp;              //MOSFET Temperature
uint8 motorTemp;            // Motor Temperature
uint8 busVoltage;           // Bus Voltage Data
};
```

Joint Command: motor_cmd.idl

```
cat motor_cmd.idl
struct MotorCmd_ {
uint8 mode;                 // Mode
float q;                    // Position
float dq;                   // Velocity
float tau;                  // Torque
float kp;                   // Proportional Gain
float kd;                   // Differential Gain
};
```

Imu Information: imu_state.idl

```
cat imu_state.idl
struct IMUState_ {
float quaternion[4];        // Quaternion (4D)
float gyroscope[3];         // Gyroscope data (3D)
float accelerometer[3];     // Accelerometer data (3D)
float rpy[3];               // Euler angles (roll, pitch, yaw)
uint8 temperature;          // Temperature sensor data
};
```

 Copyright © SHENZHEN DOBOT CORP LTD

wireless_remote Description

It holds a value when the corresponding key is pressed, and reverts to 0 when released.

wireless_remote[2]， From high bit to low bit, the corresponding buttons are: '', '', 'LT', 'RT', 'SELECT', 'START', 'LB', 'RB'.

wireless_remote[3]， From high bit to low bit, the corresponding buttons are: 'LEFT', 'DOWN', 'RIGHT', 'UP', 'Y', 'X', 'B', 'A'.

wireless_remote[4->7]， Stores the floating-point value of the LX key, converted to a range between [-1,0]; wireless_remote[4] is the low byte of the value.

wireless_remote[8->11]， Stores the floating-point value of the RX key, in the range [0,1]; wireless_remote[8] is the low byte of the value.

wireless_remote[12->15]， Stores the floating-point value of the RY key, converted to a range between [-1,0]; wireless_remote[12] is the low byte of the value.

wireless_remote[16->19]， Stores the floating-point value of the LY key, in the range [0,1]; wireless_remote[16] is the low byte of the value.

4.1.3 Switching the Underlying State

Function Description

Due to safety precautions, if joint control is required, use this interface to set the underlying state to a non-zero torque mode. Otherwise, the control instructions will not be sent to the joints.

Topic Name

set/fsm/id

IDL Document Content

```
cat set_fsm_id.idl
struct SetFsmId_ {
        uint16 id;                      //For Algorithm State Machine
};
```

4.1.4 Upper Limb State

Function Description

The underlying information of the upper limb.

Topic Name

rt/upper/state

 Copyright © SHENZHEN DOBOT CORP LTD

IDL Document Content

```
cat upper_state.idl
#include "imu_state.idl"
#include "motor_state.idl"
#include "bms_state.idl"
struct UpperState_ {
    char robot_type[16];                /*Type defined according to the product*/
    boolean is_upper_control;           /*true: upper limb has control, false: upper limb lacks control*/
    uint16 fsm_id;                      /*Corresponding algorithm state machine*/
    MotorState_ motor_state[17];
    BmsState_ bms_state;
    octet wireless_remote[40];          /*Key values for the remote controller*/
    uint32 reserve;                     /*Reserved for Dobot*/
};
```

NOTE

The *is_upper_control* indicates whether teleoperation is enabled, with *true* meaning that teleoperation is activated.

4.1.5 Upper Limb Control

Function Description

Upper limb control commands.

Topic Name

rt/upper/cmd

IDL Document Content

```
cat upper_cmd.idl
#include "motor_cmd.idl"

struct UpperCmd_ {
    MotorCmd_ motor_cmd[17];            //Motor commands, including 20 MotorCmd_ structures
};
```

 Copyright © SHENZHEN DOBOT CORP LTD

4.1.6 Lower Limb Status

Function Description

Information on lower limb status.

Topic Name

rt/lower/state

IDL Document Content

```
cat lower_state.idl
#include "imu_state.idl"
#include "motor_state.idl"
#include "bms_state.idl"

struct LowerState_ {
        uint16 fsm_id;              /*Corresponding Algorithm State Machine*/
    IMUState_ imu_state;
    MotorState_ motor_state[12];
    BmsState_ bms_state;
    octet wireless_remote[40];
        uint32 reserve;                     /*Reserved for Dobot*/
};
```

4.1.7 Lower Limb Control

Function Description

Lower limb control commands.

Topic Name

rt/lower/cmd

IDL Document Content

```
    cat lower_cmd.idl
    #include "motor_cmd.idl"

    struct LowerCmd_ {
        MotorCmd_ motor_cmd[12];      // Motor Commands, including 12
MotorCmd_ structures
    };
```

4.1.8 Dexterous Hand Status

Function Description

Only the angle data for each finger of the dexterous hand.

Topic Name

rt/hands/state

IDL Document Content

```
cat hands_state.idl
#include "motor_state.idl"

struct HandsState_
{
        MotorState_ hands[12];
};
```

4.1.9 Dexterous Hand Control

Function Description

Controls only the angle of each finger on the dexterous hand.

Topic Name

rt/hands/cmd

IDL Document Content

```
cat hands_cmd.idl
#include "motor_cmd.idl"
struct HandsCmd_
{
        MotorCmd_ hands[12];
} ;
```

4.2 Underlying Service Interface

The underlying communication enables data exchange between the client PC (PC2 / external PC) and the robot, utilizing the DDS protocol.

The interface becomes active when the state machine switches to the debugging state (fsm id: 2).

 Copyright © SHENZHEN DOBOT CORP LTD

WARNING

When using the underlying service interface, please ensure the robot is set to debugging mode first; otherwise, there is a risk of anomalies.

4.2.1 Underlying Control Command Interface

The Bridge class encapsulates DDS communication and provides interfaces for "issuing control commands" and "retrieving the latest status".

Bridge.h

```
namespace Atom
{
    class Bridge
    {
    public:
        Bridge();
        ~Bridge();

        /* Command */
        void SetNewestLegCommand(const LegCommand &motor_command);
        void SetNewestHandCommand(const HandCommand
&hand_command);
        void SetNewestArmCommand(const ArmCommand &arm_command);
        void SetNewestFsmCommand(const int &fsm_id);

        /* State */
        const std::shared_ptr<const BaseState> GetNewestBaseStatePtr()
{ return base_state_buffer_.GetData(); }
        const std::shared_ptr<const JointState> GetNewestJointStatePtr()
{ return joint_state_buffer_.GetData(); }
        const std::shared_ptr<const ArmJointState> GetNewestArmStatePtr()
{ return arm_state_buffer_.GetData(); }
        const std::shared_ptr<const DexterousHandState>
GetNewestHandStatePtr() { return hand_state_buffer_.GetData(); }
        const std::shared_ptr<const RemoteCommand>
GetNewestRemoteCommandPtr() { return remote_command_buffer_.GetData(); }

    private:
        ......
    };
}   // namespace Atom

#endif    // BRIDGE_H
```

 Copyright © SHENZHEN DOBOT CORP LTD

motor_command.h

```
    ......

    namespace Atom
    {
        struct LegCommand {
            Eigen::Matrix<float, Atom::kLegDofs, 1> q_ref = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();             // angle
            Eigen::Matrix<float, Atom::kLegDofs, 1> dq_ref = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();           // velocity
            Eigen::Matrix<float, Atom::kLegDofs, 1> kp = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();                 // position stiffness coefficients
            Eigen::Matrix<float, Atom::kLegDofs, 1> kd = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();                 // velocity stiffness coefficients
            Eigen::Matrix<float, Atom::kLegDofs, 1> tau_forward =
Eigen::Matrix<float, Atom::kLegDofs, 1>::Zero();   // forward torque
        };

        ......
    }
```

robot_state.h

```
    ......

    namespace Atom
    {
        struct JointState {
            Eigen::Matrix<float, Atom::kLegDofs, 1> q = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();      // angle
            Eigen::Matrix<float, Atom::kLegDofs, 1> dq = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();     // velocity
            Eigen::Matrix<float, Atom::kLegDofs, 1> tau = Eigen::Matrix<float,
Atom::kLegDofs, 1>::Zero();   // torque
        };

        ......
    }
```

 Copyright © SHENZHEN DOBOT CORP LTD

4.2.2 Low-Level Control Example

```
#include "common/bridge.h"
......

int main(int argc, char *argv[])
{
    ......

// Initialize the bridge for communication and the command structures for legs, hands, and arms.
Atom::Bridge bridge;
Atom::LegCommand leg_command;
Atom::HandCommand hand_command;
Atom::ArmCommand arm_command;

// Define timing parameters for the control loop.
// The control loop will run at a frequency of 1/control_dt.
const float init_duration = 5.0;   // time unit: s
const float control_dt = 0.01;      // time unit: s
const int s2us = 1e6;
float time = 0.0;
Timer timer;

// Set the initial position for the robot's arms and legs.
// The robot will move to this position at the beginning of the program.
Eigen::Vector<float, Atom::kArmDofs> q_arm_init;
Eigen::Vector<float, Atom::kLegDofs> q_leg_init;
q_leg_init.setZero();
q_arm_init.setZero();
for (int i = 0; i < Atom::kLegDofs; ++i) { q_leg_init[i] = q_val[0][i]; }
for (int i = 0; i < Atom::kArmDofs; ++i) { q_arm_init[i] = q_val[0][i + Atom::kLegDofs]; }
Eigen::Vector<float, Atom::kLegDofs> q_leg_ref;
Eigen::Vector<float, Atom::kArmDofs> q_arm_ref;

int row = 0;

// Main control loop.
    while (1) {
        // Get the current joint states from the robot.
        const std::shared_ptr<const Atom::JointState> js_tmp_ptr = bridge.GetNewestJointStatePtr();      // leg
        const std::shared_ptr<const Atom::ArmJointState> ajs_tmp_ptr = bridge.GetNewestArmStatePtr();   // arm
        timer.Tic();

        if (js_tmp_ptr && ajs_tmp_ptr) {
            time += control_dt;
            if (time < init_duration) {
                // Interpolate to the initial position over a specified duration.
                // This ensures a smooth start for the robot's motion.
                q_leg_ref = (q_leg_init - js_tmp_ptr->q) * time / init_duration + js_tmp_ptr->q;
```

 Copyright © SHENZHEN DOBOT CORP LTD

```
                        q_arm_ref = (q_arm_init - ajs_tmp_ptr->q) * time / init_duration
+ ajs_tmp_ptr->q;
                    } else {
                        // Follow the predefined trajectory from the file.
                        for (int i = 0; i < Atom::kLegDofs; ++i) { q_leg_ref[i] =
q_val[row][i]; }
                        for (int i = 0; i < Atom::kArmDofs; ++i) { q_arm_ref[i] =
q_val[row][i + Atom::kLegDofs]; }
                        row++;
                        if (row == q_val.size() - 1) { row = 0; }    // Loop back to the
beginning of the trajectory.
                    }
                    // Set the PID gains for the legs from the configuration file.
                    for (int j = 0; j < Atom::kLegDofs; j++) {
                        leg_command.kp(j) = controlParams[j]["kp"].get<double>();
                        leg_command.kd(j) = controlParams[j]["kd"].get<double>();
                    }
                    // Set the PID gains for the arms from the configuration file.
                    for (int j = 0; j < Atom::kArmDofs; j++) {
                        arm_command.kp(j) = controlParams[j +
Atom::kLegDofs]["kp"].get<double>();
                        arm_command.kd(j) = controlParams[j +
Atom::kLegDofs]["kd"].get<double>();
                    }

                    // Set and send the commands to the robot.
                    leg_command.q_ref = q_leg_ref;
                    leg_command.dq_ref.setZero();
                    leg_command.tau_forward.setZero();
                    bridge.SetNewestLegCommand(leg_command);

                    hand_command.q_ref.setZero();
                    hand_command.dq_ref.setZero();
                    bridge.SetNewestHandCommand(hand_command);

                    arm_command.q_ref = q_arm_ref;
                    arm_command.dq_ref.setZero();
                    arm_command.tau_forward.setZero();
                    bridge.SetNewestArmCommand(arm_command);
                }

                // Wait for the next control cycle.
                timer.Toc();
                timer.usDelay(control_dt * s2us - timer.usDuration());
            }

            return 0;
        }
```

 Copyright © SHENZHEN DOBOT CORP LTD

4.2.3 Joint Motor Sequence

Joint Number	Joint Name
00 Left leg, corresponding to single arm axis 1 (Increasing sequentially downward)	left_hip_pitch_joint
01	left_hip_roll_joint
02	left_hip_yaw_joint
03	left_knee_joint
04	left_ankle_pitch_joint
05	left_ankle_roll_joint
06 Right leg, corresponding to single arm axis 1 (Increasing sequentially downward)	right_hip_pitch_joint
07	right_hip_roll_joint
08	right_hip_yaw_joint
09	right_knee_joint
10	right_ankle_pitch_joint
11	right_ankle_roll_joint
12 Torso part	torso_joint
13 Left arm, corresponding to single arm axis 1 (Increasing sequentially downward)	left_shoulder_pitch_joint
14	left_shoulder_roll_joint
15	left_shoulder_yaw_joint
16	left_elbow_pitch_joint
17	left_elbow_roll_joint

 Copyright © SHENZHEN DOBOT CORP LTD

18	left_wrist_pitch_joint
19	left_wrist_roll_joint
20 Right arm, corresponding to single arm axis 1 (Increasing sequentially downward)	right_shoulder_pitch_joint
21	right_shoulder_roll_joint
22	right_shoulder_yaw_joint
23	right_elbow_pitch_joint
24	right_elbow_roll_joint
25	right_wrist_pitch_joint
26	right_wrist_roll_joint
27 Head	head_yaw_joint
28	head_pitch_joint

4.3 High-level Motion Service Interface

The primary functions of the high-level motion service interfaces are **full-body control** and **upper limb control** of the robot.

- **Full-body control:** It controls the robot to switch modes and move, and provides an RPC interface.
- **Upper limb control:** It enables independent control of the upper limb motors to achieve operation while the robot's built-in movement controller is running, providing a DDS interface.

4.3.1 High-Level Full-Body Control RPC Interface

Interface name	**GetFsmId**
Interface definition	RpcErrorCode GetFsmId(int32_t &fsm_id);
Function overview	Obtain current state machine ID.
Parameter	fsm_id: Retrieved current state machine ID.

 Copyright © SHENZHEN DOBOT CORP LTD

Return value	Returns 0 on successful execution; otherwise, returns the relevant error code. Please refer to the Error Codes section for more details.
Note	For details on the state machine ID, please refer to the State Machine ID section.

Interface name	**SetFsmId**
Interface definition	RpcErrorCode SetFsmId(int32_t fsm_id);
Function overview	Set state machine ID.
Parameter	fsm_id: Target state machine ID.
Return value	Returns 0 on successful execution; otherwise, returns the relevant error code. Please refer to the Error Codes section for more details.
Note	For details on the state machine ID, please refer to the State Machine ID section.

Interface name	**SwitchUpperLimbControl**
Interface definition	RpcErrorCode SwitchUpperLimbControl(bool is_on);
Function overview	Switch the upper limb control.
Parameter	is_on: true indicates switching upper limb control to interface control, false indicates disabling the switch and returning to the full-body motion control.
Return value	Returns 0 on successful execution; otherwise, returns the relevant error code. Please refer to the Error Codes section for more details.
Note	Switching is only allowed to be enabled in control states such as "Ready" and during walking; upon entering a state where switching is not permitted, it will automatically be disabled.

Interface name	**SetVel**

 Copyright © SHENZHEN DOBOT CORP LTD

Interface definition	RpcErrorCode SetVel(float vx, float vy, float vyaw, float duration = 1.0);
Function overview	Set speed command.
Parameter	vx: Forward and backward motion speed, positive for forward, in units of m/s. vy: Left and right motion speed, positive for left, in units of m/s. vyaw: Rotational speed, positive for counterclockwise, in units of rad/s. duration: Duration of the speed command, stops after the timeout, in units of seconds.
Return value	Returns 0 on successful execution; otherwise, returns the relevant error code. Please refer to the Error Codes section for more details.
Note	The low-level system will automatically clamp the set parameters to a reasonable range. Remote controller commands have higher priority. To stop the entire robot's motion: set the speed command to zero using SetVel(0, 0, 0).

4.3.2 Error Codes (RpcErrorCode)

ID	Description
0	No error
1	Socket creation failed
2	Communication connection failed
3	Failed to read response
4	Failed to parse response data
5	Incorrect response data format
6	Internal server error
100	Illegal state transition request; please make the call within allowed states
101	Illegal request to switch upper-limb control; please make the call within allowed states
102	Illegal request to set speed command; please make

 Copyright © SHENZHEN DOBOT CORP LTD

	the call within allowed states

4.3.3 Upper Limb Motor Control DDS Interface

Refer to the Underlying Service Interface.

4.4 State Machine ID (FsmId)

ID	State	Description
0	Zero-torque state	All robot motors cease active motion, and there is no damping sensation when swinging.
1	Damping state	All robot motors cease active motion, but there is a noticeable damping sensation when swinging.
2	Debugging state	Development and debugging are based on the underlying control SDK. It acquires real-time operation data (e.g., battery, motors, IMU) and directly drives the motors.
3	Diagnostic state	Controls joint positions to transition the robot into specific diagnostic postures.
100	Prepare for locked standing	The robot will slowly move into a prepared posture for motion mode within 5 seconds (no balance control).
101	Squatting state	The robot will slowly move into a squatting posture within 5 seconds (no balance control).
102	Sitting state	The robot will slowly move into a sitting posture within 5 seconds (no balance control).

ID	State	Description
301	Bent-knee walking mode	The robot will slowly move to the starting position of the model within 5 seconds (no balance control), then transition into the model using the remote controller. Once in the model, the robot will maintain balance, serving as the prepared posture before the motion state and skill state.

 Copyright © SHENZHEN DOBOT CORP LTD

4.5 Audio Service Interface

The following describes the RPC interfaces provided by the server. All interfaces return a type of RpcErrorCode, and the actual call results or data are returned via reference parameters.

Interface name	**GetDevState**
Interface definition	RpcErrorCode GetDevState(int32_t& devstate)
Function overview	Retrieve device status
Output parameters	devstate: Device status value

Example	enum SpeakAudioStateEnum
	{
	eSpeakAudioInit = 1, // Not Initialized
	eSpeakAudioPlaying = 2, // Playing
	eSpeakAudioPause = 3, // Paused
	eSpeakAudioStop = 4 // Stopped
	};

Interface name	**SetVolume**
Interface definition	RpcErrorCode SetVolume(int32_t volume, bool& result)
Function overview	Set device volume
Input parameters	volume: Target volume value volume:0-128
Output parameters	result: Indicates whether the setting was successful

Interface name	**GetVolume**
Interface definition	RpcErrorCode GetVolume(int32_t& volume)

 Copyright © SHENZHEN DOBOT CORP LTD

Function overview	Get the current device volume
Output parameters	volume: Current volume value

Interface name	**PlayAudio**
Interface definition	RpcErrorCode PlayAudio(bool& result)
Function overview	Play the audio.
Output parameters	result: Whether the playback was successful

Interface name	**StopAudio**
Interface definition	RpcErrorCode StopAudio(bool& result)
Function overview	Stop audio playback.
Output parameters	result: Whether the stop operation was successful.

Interface name	**PauseAudio**
Interface definition	RpcErrorCode PauseAudio(bool& result)
Function overview	Pause audio playback.
Output parameters	result: Indicates whether the pause was successful

Interface name	**ContinueAudio**
Interface definition	RpcErrorCode ContinueAudio(bool& result)

 Copyright © SHENZHEN DOBOT CORP LTD

Function overview	Resume audio playback.
Output parameters	result: Indicates whether the resume operation was successful.

Interface name	**Pc2SwitchTTS**
Interface definition	RpcErrorCode Pc2SwitchTTS(bool flag, bool& result)
Function overview	Toggle TTS function.
Input parameters	flag: Enable or disable TTS
Output parameters	result: Operation result

Interface name	**Pc2PlayTTS**
Interface definition	RpcErrorCode Pc2PlayTTS(const std::string text, bool& result)
Function overview	Play TTS audio for the specified text.
Input parameters	text: The text content
Output parameters	result: Whether the playback was successful

Interface name	**Pc2SwitchASR**
Interface definition	RpcErrorCode Pc2SwitchASR(bool flag, bool& result)
Function overview	Toggle the ASR function.
Input parameters	flag：Enable or disable ASR
Output	result: Operation result

 Copyright © SHENZHEN DOBOT CORP LTD

parameters	

Interface name	**Pc2PlayASR**
Interface definition	RpcErrorCode Pc2PlayASR(const std::string language, uint32_t timeout, std::string& result)
Function overview	Start ASR recognition.
Input parameters	• language: The language type • timeout：The timeout period
Output parameters	result：Recognized text • Timeout function timeout • http interface exception Http fail • Invalid response content Invalid json • udp multicast connection failed Socket fail • Audio device failed to load Device load fail

Interface name	**Pc2PlayDify**
Interface definition	RpcErrorCode Pc2PlayDify(const std::string text, std::string& result)
Function overview	Generate and play speech using the Dify model.
Input parameters	text: The text of the question

 Copyright © SHENZHEN DOBOT CORP LTD

Output parameters	result: Returned response text • http interface exception Http fail • Invalid response content Invalid json • Audio device failed to load Device load fail

Interface name	**Pc1MicTTS**
Interface definition	RpcErrorCode Pc1MicTTS(const std::string wavPath, const std::string text, bool& result)
Function overview	Play TTS audio for the specific text through the Mic and generate an audio file.
Input parameters	• wavPath: The save path • text: The text content
Output parameters	result: Whether the playback was successful

Interface name	**Pc1MicServer**
Interface definition	RpcErrorCode Pc1MicServer(bool flag, bool& result)
Function overview	Start or stop the Mic multicast service.
Input parameters	flag: true= start, false = stop
Output parameters	result: Whether the operation was successful

 Copyright © SHENZHEN DOBOT CORP LTD

Interface name	**Pc1MicIVW**
Interface definition	RpcErrorCode Pc1MicIVW(const uint32_t timeout, std::string& result)
Function overview	Initiate wake recognition using the Mic.
Input parameters	timeout: The timeout period (s)
Output parameters	result: • Key wake words • Timeout: function timeout • udp multicast connection failed Socket fail • aikit interface exception Function fail • Audio device load failure Device load fail

Interface name	**Pc1MicESR**
Interface definition	RpcErrorCode Pc1MicESR(const uint32_t timeout, std::string& result)
Function overview	Use Mic to start command word recognition.
Input parameters	timeout: The timeout period (s)
Output parameters	Result: • Recognized result • Timeout: function timeout

 Copyright © SHENZHEN DOBOT CORP LTD

	• udp multicast connection failed
	Socket fail
	• aikit interface exception
	Function fail
	• Audio device load failure
	Device load fail

 Copyright © SHENZHEN DOBOT CORP LTD

5. Reference Examples

5.1 DDS Communication Example

```
#include "common/bridge.h"
......

int main(int argc, char *argv[])
{
    ......

    // Initialize the bridge for communication and the command structures for legs, hands, and arms.
    Atom::Bridge bridge;
    Atom::LegCommand leg_command;
    Atom::HandCommand hand_command;
    Atom::ArmCommand arm_command;

    // Define timing parameters for the control loop.
    // The control loop will run at a frequency of 1/control_dt.
    const float init_duration = 5.0;    // time unit: s
    const float control_dt = 0.01;      // time unit: s
    const int s2us = 1e6;
    float time = 0.0;
    Timer timer;

    // Set the initial position for the robot's arms and legs.
    // The robot will move to this position at the beginning of the program.
    Eigen::Vector<float, Atom::kArmDofs> q_arm_init;
    Eigen::Vector<float, Atom::kLegDofs> q_leg_init;
    q_leg_init.setZero();
    q_arm_init.setZero();
    for (int i = 0; i < Atom::kLegDofs; ++i) { q_leg_init[i] = q_val[0][i]; }
    for (int i = 0; i < Atom::kArmDofs; ++i) { q_arm_init[i] = q_val[0][i + Atom::kLegDofs]; }
    Eigen::Vector<float, Atom::kLegDofs> q_leg_ref;
    Eigen::Vector<float, Atom::kArmDofs> q_arm_ref;

    int row = 0;

    // Main control loop.
    while (1) {
        // Get the current joint states from the robot.
        const std::shared_ptr<const Atom::JointState> js_tmp_ptr = bridge.GetNewestJointStatePtr();      // leg
        const std::shared_ptr<const Atom::ArmJointState> ajs_tmp_ptr = bridge.GetNewestArmStatePtr();   // arm
        timer.Tic();

        if (js_tmp_ptr && ajs_tmp_ptr) {
            time += control_dt;
            if (time < init_duration) {
                // Interpolate to the initial position over a specified duration.
                // This ensures a smooth start for the robot's motion.
```

```
                    q_leg_ref = (q_leg_init - js_tmp_ptr->q) * time / init_duration +
js_tmp_ptr->q;
                    q_arm_ref = (q_arm_init - ajs_tmp_ptr->q) * time / init_duration
+ ajs_tmp_ptr->q;
                } else {
                    // Follow the predefined trajectory from the file.
                    for (int i = 0; i < Atom::kLegDofs; ++i) { q_leg_ref[i] =
q_val[row][i]; }
                    for (int i = 0; i < Atom::kArmDofs; ++i) { q_arm_ref[i] =
q_val[row][i + Atom::kLegDofs]; }
                    row++;
                    if (row == q_val.size() - 1) { row = 0; }    // Loop back to the
beginning of the trajectory.
                }
                // Set the PID gains for the legs from the configuration file.
                for (int j = 0; j < Atom::kLegDofs; j++) {
                    leg_command.kp(j) = controlParams[j]["kp"].get<double>();
                    leg_command.kd(j) = controlParams[j]["kd"].get<double>();
                }
                // Set the PID gains for the arms from the configuration file.
                for (int j = 0; j < Atom::kArmDofs; j++) {
                    arm_command.kp(j) = controlParams[j +
Atom::kLegDofs]["kp"].get<double>();
                    arm_command.kd(j) = controlParams[j +
Atom::kLegDofs]["kd"].get<double>();
                }

                // Set and send the commands to the robot.
                leg_command.q_ref = q_leg_ref;
                leg_command.dq_ref.setZero();
                leg_command.tau_forward.setZero();
                bridge.SetNewestLegCommand(leg_command);

                hand_command.q_ref.setZero();
                hand_command.dq_ref.setZero();
                bridge.SetNewestHandCommand(hand_command);

                arm_command.q_ref = q_arm_ref;
                arm_command.dq_ref.setZero();
                arm_command.tau_forward.setZero();
                bridge.SetNewestArmCommand(arm_command);
            }

            // Wait for the next control cycle.
            timer.Toc();
            timer.usDelay(control_dt * s2us - timer.usDuration());
        }

        return 0;
    }
```

5.2 ROS2 Communication Example

```
#include "rclcpp/rclcpp.hpp"
#include "dobot_atom/msg/upper_state.hpp"

class SimpleUpperStateReader : public rclcpp::Node
{
public:
    SimpleUpperStateReader() : Node("simple_upper_state_reader")
    {
        // Subscribe to /upper/state topic
        upper_state_sub_ =
this->create_subscription<dobot_atom::msg::UpperState>(
            "/upper/state", 10,
            std::bind(&SimpleUpperStateReader::upperStateCallback, this,
std::placeholders::_1));

        RCLCPP_INFO(this->get_logger(), "Simple upper state reader started,
waiting for data...");
    }

private:
    void upperStateCallback(const dobot_atom::msg::UpperState::SharedPtr
msg)
    {
        RCLCPP_INFO(this->get_logger(), "Received upper state data:");
        RCLCPP_INFO(this->get_logger(), "    Timestamp: %ld.%09ld",
msg->stamp.sec, msg->stamp.nanosec);
        RCLCPP_INFO(this->get_logger(), "    Joint count: %zu",
msg->q.size());

        // Print first 3 joint positions (if available)
        for (size_t i = 0; i < std::min(msg->q.size(), size_t(3)); ++i) {
            RCLCPP_INFO(this->get_logger(), "    Joint %zu position: %.4f", i,
msg->q[i]);
        }
    }

    rclcpp::Subscription<dobot_atom::msg::UpperState>::SharedPtr
upper_state_sub_;
};

int main(int argc, char * argv[])
{
    rclcpp::init(argc, argv);
    rclcpp::spin(std::make_shared<SimpleUpperStateReader>());
    rclcpp::shutdown();
    return 0;
}
```

 Copyright © SHENZHEN DOBOT CORP LTD

5.3 RPC Example

```
#include "rpc/algs_rpc_client.h"
#include <iostream>
#include <thread>

int main()
{
    // Initialize the RPC client for communication with the robot's algorithm
server.
    Atom::AlgsRpcClient rpc;
    int sw = 0;

    while (true) {
        // Main loop to interact with the user and send commands to the robot.
        if (sw == 0) {
            // Prompt the user to select a command.
            std::cout << "Enter: 1 = FSM, 2 = SetVel." << std::endl;
            std::cin >> sw;
            if (std::cin.fail()) {
                std::cin.clear();
// Clear error flags
                std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');  // Discard invalid input
                std::cout << "Invalid input. Please enter a number." <<
std::endl;
                continue;
            }
            std::cout << "Test Start..." << std::endl;
        }
        switch (sw) {
            case 1: { // Finite State Machine (FSM) control
                int32_t fsm_id = 0;
                rpc.GetFsmId(fsm_id);
                std::cout << "Enter number for fsm_id: ";
                std::cin >> fsm_id;
                // Set the FSM ID to change the robot's state.
                Atom::RpcErrorCode errorCode = rpc.SetFsmId(fsm_id);
                if (errorCode != Atom::RpcErrorCode::SUCCESS) { std::cout
<< "SetFsmId errorCode: " << errorCode << std::endl; }
                std::this_thread::sleep_for(std::chrono::seconds(2));
                break;
            }
            case 2: { // Velocity control
                int32_t fsm_id = 0;
                rpc.GetFsmId(fsm_id);
                // Check if the robot is in a state that allows velocity control.
                if (301 != fsm_id && 302 != fsm_id) {
                    std::cout << "SetVel failed. Please enter fsm 301 or 302."
<< std::endl;
                    sw = 0;
                    break;
                }
                const float kVel = 0.3;        // Desired velocity.
```

 Copyright © SHENZHEN DOBOT CORP LTD

```
                        const float duration = 2.0; // Duration of the velocity command.
                        // Send the velocity command to the robot.
                        Atom::RpcErrorCode errorCode = rpc.SetVel(kVel, 0.0, 0.0,
duration);
                        if (errorCode != Atom::RpcErrorCode::SUCCESS) { std::cout
<< "\033[31mSetVel failed.\033[0m" << std::endl; }
                        sw = 0;
                        break;
                    }
                    default:
                        std::cout << "not support" << std::endl;
                        sw = 0;
                        break;
                }
            }
            return 0;
        }
```

5.4 LiDAR Example

5.4.1 LiDAR Introduction

The LiDAR module of the Dobot Atom is located on its head, and the model is RoboSense Airy.

The Airy is a novel short-range 3D LiDAR developed by RoboSense specifically for eliminating blind spots. It is primarily used in applications such as robotic environment perception, autonomous vehicle environment perception, UAV mapping, and smart city fields.

LiDAR details page:

https://www.robosense.ai/IncrementalComponents/Airy

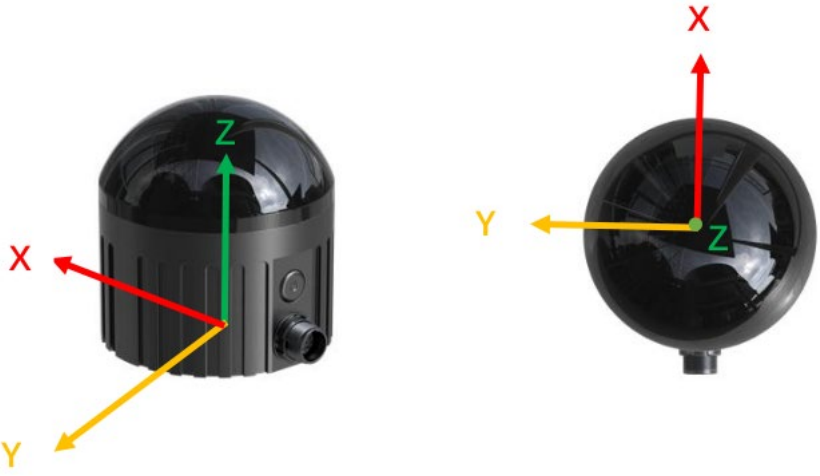
LiDAR resources page:

https://www.robosense.ai/resources-161

Official GitHub repository:

https://github.com/RoboSense-LiDAR/rslidar_sdk

The coordinate system of the Airy is as shown in the figure below:



5.4.2 Parameter Settings

Communication Connection

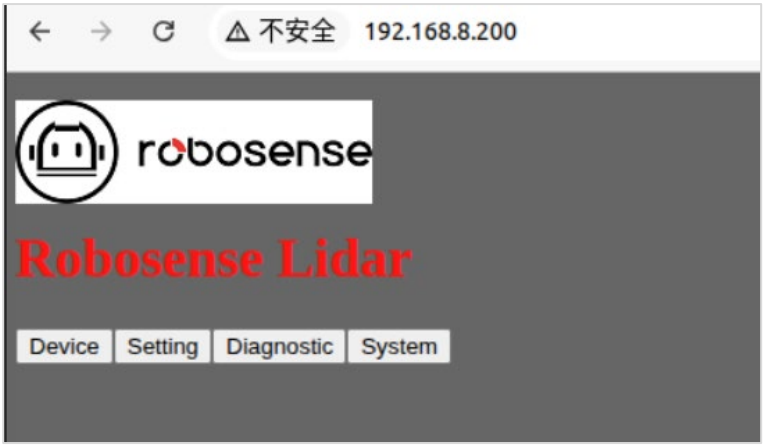
The radar's IP address, set by default when shipped with the robot, is: **192.168.8.200**.

Connect the computer to the robot using an Ethernet cable and set the computer's IP within the 192.168.8 subnet, with the Netmask as 255.255.255.0 and Gateway as 192.168.8.1. You can verify communication with the radar by pinging 192.168.8.200.

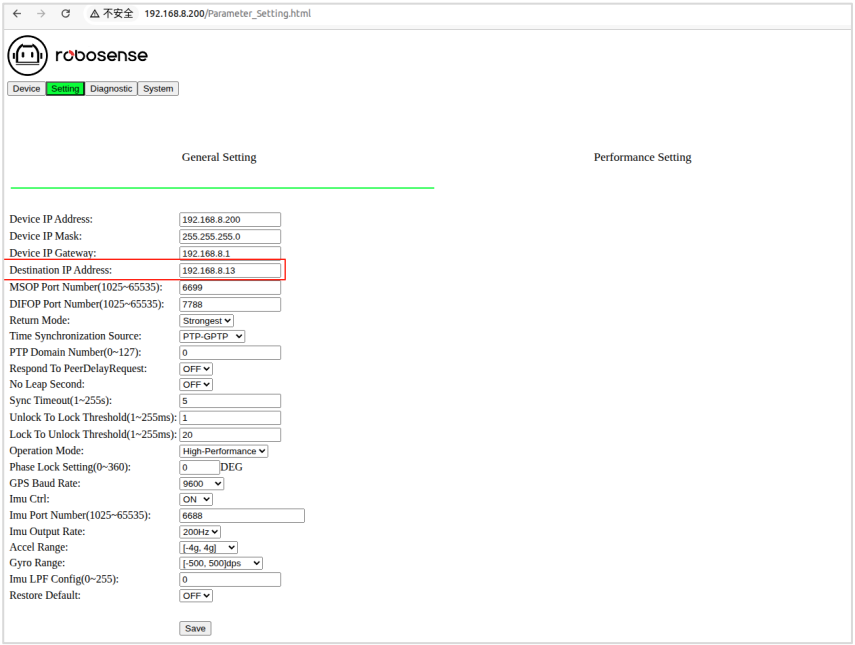
Target IP Setting

The Airy supports parameter configuration, operation information/status viewing, and firmware upgrades via a web interface.

1. Open a web browser on the computer connected to the LiDAR and access the product's IP address: http://192.168.8.200.



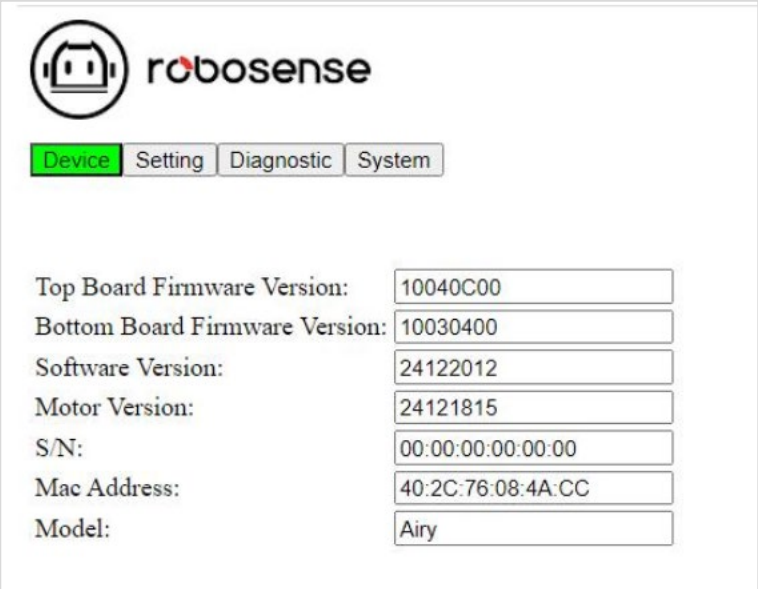
2. Click on **Setting** and modify the "**Destination IP Address**" to the target machine's IP. The default setting is the IP of Dobot Atom PC2.



 Copyright © SHENZHEN DOBOT CORP LTD

Other Parameter Settings

For more parameter configurations, please refer to the official Product User Guide.



5.4.3 Data Acquisition

NOTICE

During the data acquisition process, ensure that the computer's IP address is set to the radar's target IP.

Data Acquisition via Host Computer

Download the Host Computer

Download the **RSView** software from the Laser Resources page.

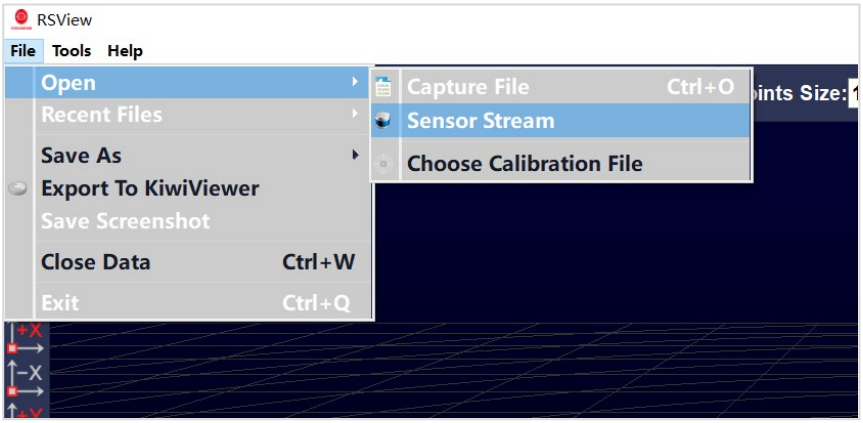
- **Windows:** After the download is complete, extract the files and navigate to the bin directory. Then, click on RsView.exe to launch the host computer software.
- **Ubuntu:** After downloading, extract the files into the home directory.

```
cd rsview
./run_rsview.sh    #Enter the password
```

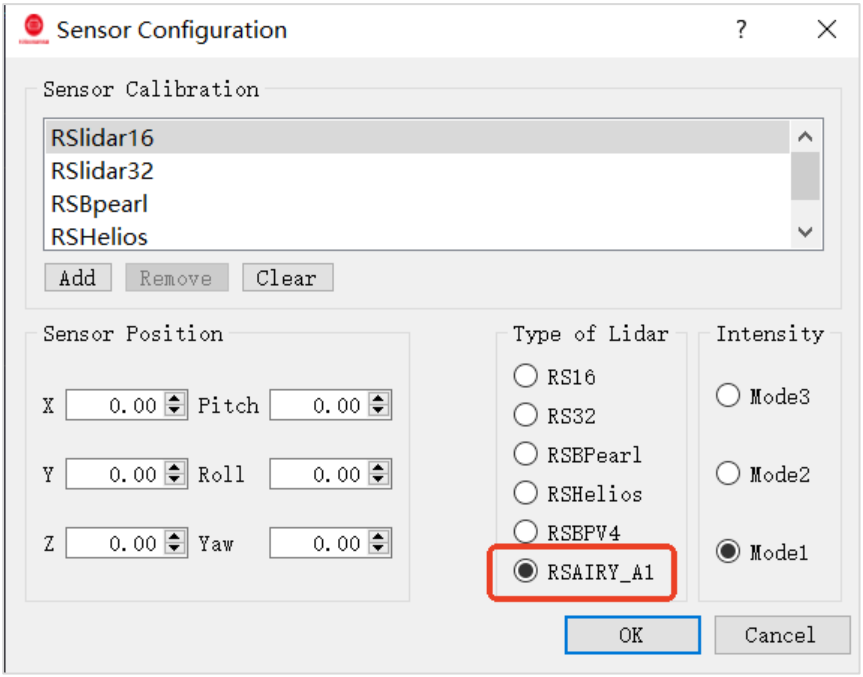
 Copyright © SHENZHEN DOBOT CORP LTD

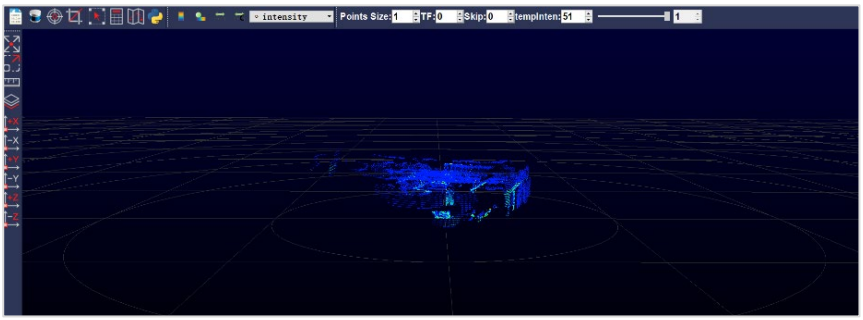
Data Acquisition Process

1. Click on File > Open > Sensor Stream.



2. In the **Type of Lidar** column, check **RSAIRY_A1** and click **OK,** the point cloud will then be displayed.





 Copyright © SHENZHEN DOBOT CORP LTD

Data Acquisition via SDK

Function Introduction

The **rslidar_sdk** is a radar driver software package developed by RoboSense for the Ubuntu environment, including:

- Radar driver core rs_driver
- ROS extension features
- ROS2 extension features

Repository address: https://github.com/RoboSense-LiDAR/rslidar_sdk

Data Acquisition via rs_driver

The **rs_driver** is a driver library for RoboSense radar, included in the rslidar_sdk package (located at rslidar_sdk/src/rs_driver). It can also be downloaded separately. The rs_driver provides a development library and example code for obtaining radar data.

- **Download and install the rs_driver separately**

```
git clone https://github.com/RoboSense-LiDAR/rs_driver.git
sudo apt-get install libpcap-dev libeigen3-dev libboost-dev libpcl-dev
#Dependency installation, optional
cd rs_driver
mkdir build && cd build
cmake ..
make -j4
sudo make install
```

- **Data Acquisition Demo**

The rs_driver/demo/demo_online.cpp offers a routine for obtaining radar data using rs_driver. To compile the demo, make sure to enable the compile option COMPILE_DEMOS=ON in step 5 of the compilation process.

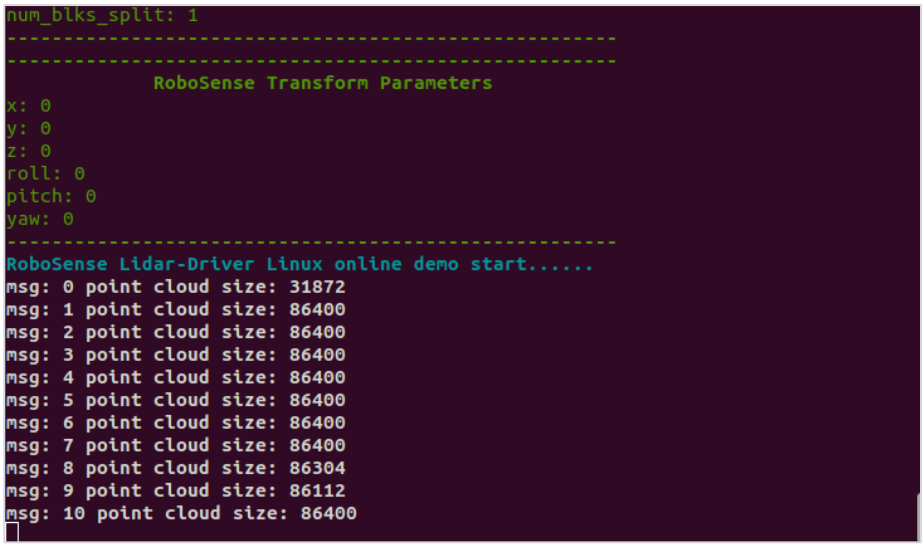
```
cmake .. -DCOMPILE_DEMOS=ON
make -j4
```

After compilation, run the demo in the build directory.

```
./demo/demo_online
```

You will see the following output in the terminal:

 Copyright © SHENZHEN DOBOT CORP LTD



Acquiring Data via ROS Driver

The rslidar_sdk automatically detects whether the current environment is ROS1 or ROS2 and adjusts the compilation method accordingly. It is not recommended to compile rslidar_sdk in a system that has both ROS1 and ROS2 installed.

Place the rslidar_sdk project in the src folder of the corresponding workspace, and then navigate to the workspace directory with cd.

- **ROS1**

```
catkin_make
source devel/setup.bash
roslaunch rslidar_sdk start.launch
```

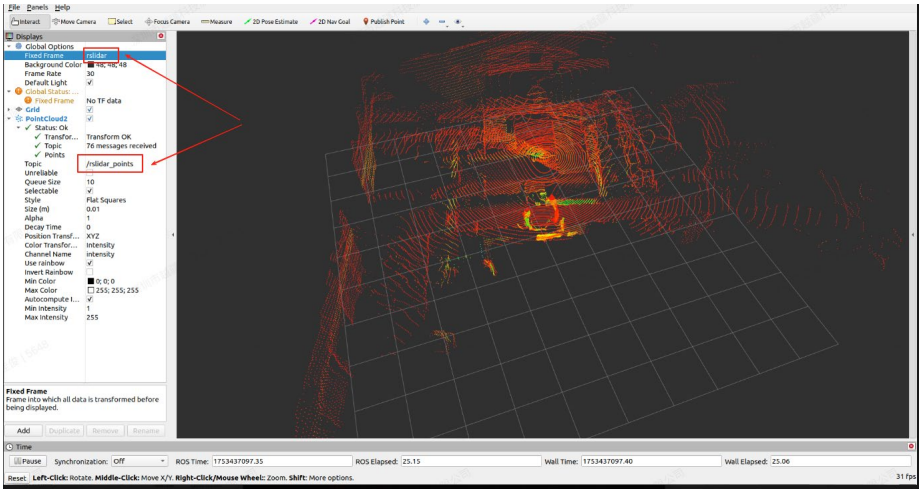
- **ROS2**

```
colcon build
source install/setup.bash
ros2 launch rslidar_sdk start.py
```

After running, the LiDAR node will output the topics /rslidar_points and /rslidar_imu_data.

You can view the point cloud data using Rviz.

 Copyright © SHENZHEN DOBOT CORP LTD



5.5 Depth Camera Example

5.5.1 Depth Camera Introduction

The Dobot Atom has two depth cameras: the RealSense D455FC on the head and the RealSense D435CB on the waist.

Official product homepage:

https://realsenseai.com//stereo-depth-cameras/

Official document:

https://dev.realsenseai.com/docs/docs-get-started

Official GitHub repository:

https://github.com/IntelRealSense

Here are the parameters for the two camera models in the table below:

 Copyright © SHENZHEN DOBOT CORP LTD

Model	D435CB	D455FC
Suitable Environment	Indoor/Outdoor	Indoor/Outdoor
Image Sensor Technology	Global Shutter	Global Shutter
Stereo Baseline	50mm	95mm
Depth Field of View (H×V)	87°×58°	87°×58°
Max Depth Resolution @Frame Rate	1280×720@30fps	1280×720@30fps
Max Frame Rate @Depth Resolution	90fps@848×480	90fps@848×480
Z-Accuracy Error	<2% (at 2m)	<2% (at 4m)
RGB Sensor Technology	Rolling Shutter	Global Shutter
RGB Resolution @Frame Rate	1920×1080@30fps	1280×800@30fps
RGB Field of View (H×V)	69°×42°	90°×65°
Inertial Measurement Unit (IMU)	None	Yes
Minimum Distance at Max Resolution	~28cm	~52cm
Ideal Working Range	0.3 – 3m	0.6 – 6m
Main Components	D430 Module, D4/D4V3 Board, RGB Sensor	D450 Module, D4V3 Board
Dimensions (mm)	90×25×25	124×29×26
Interface	Type - C	Type - C
Note	None	Equipped with 750nm Near-Infrared Filter

 Copyright © SHENZHEN DOBOT CORP LTD

5.5.2 Data Acquisition

Data Acquisition via Host Computer

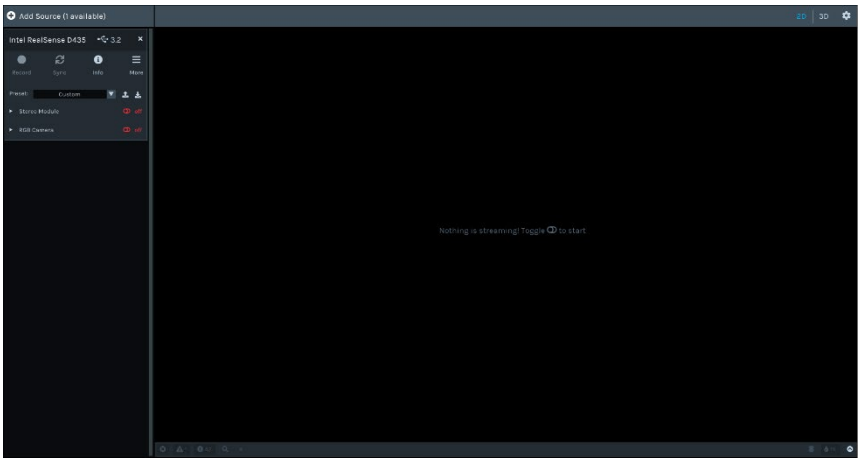
Download the Host Computer

Windows: Download Intel.RealSense.Viewer.exe

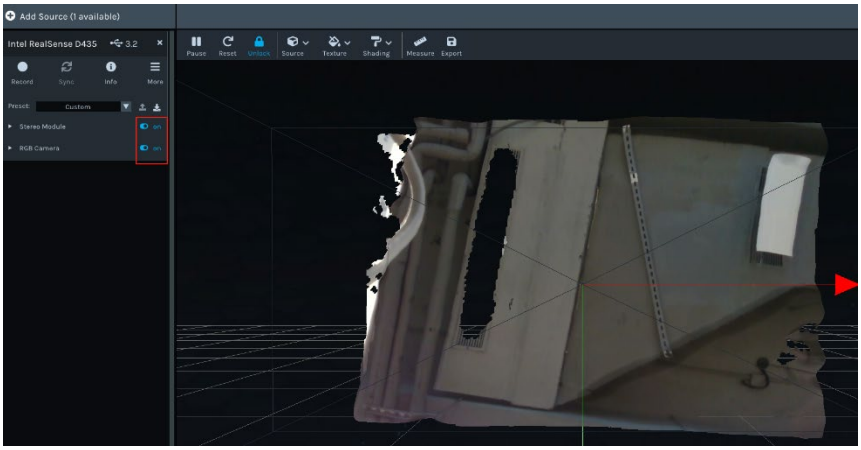
Linux: Refer to the official tutorial.

Data Acquisition Process

1. On Windows, double-click the .exe file to open the host computer software. On Linux, open the terminal and type realsense-viewer to launch the host computer software.

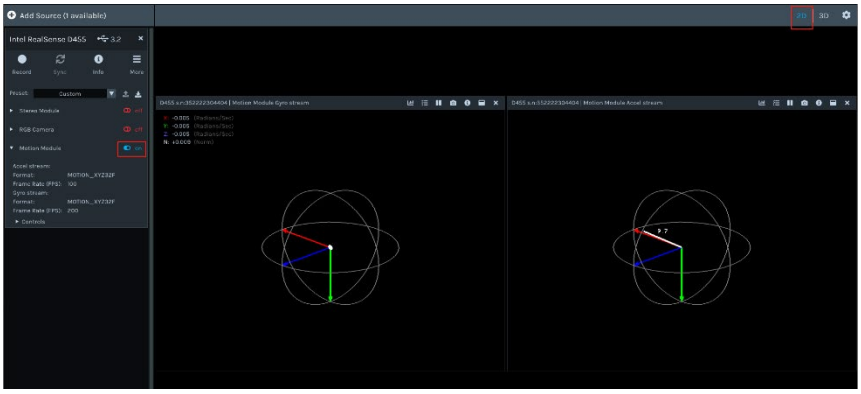


2. Turn on the corresponding module switch to view the data.



3. The D455FC integrates inertial sensors, so it also provides inertial data.

 Copyright © SHENZHEN DOBOT CORP LTD



Acquire Data via ROS

Official ROS driver repository:

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

For more details, refer to the readme. It is generally recommended to use the SDK for development.

Acquire Data via SDK

Official SDK repository:

https://github.com/IntelRealSense/librealsense

For development examples, refer to:

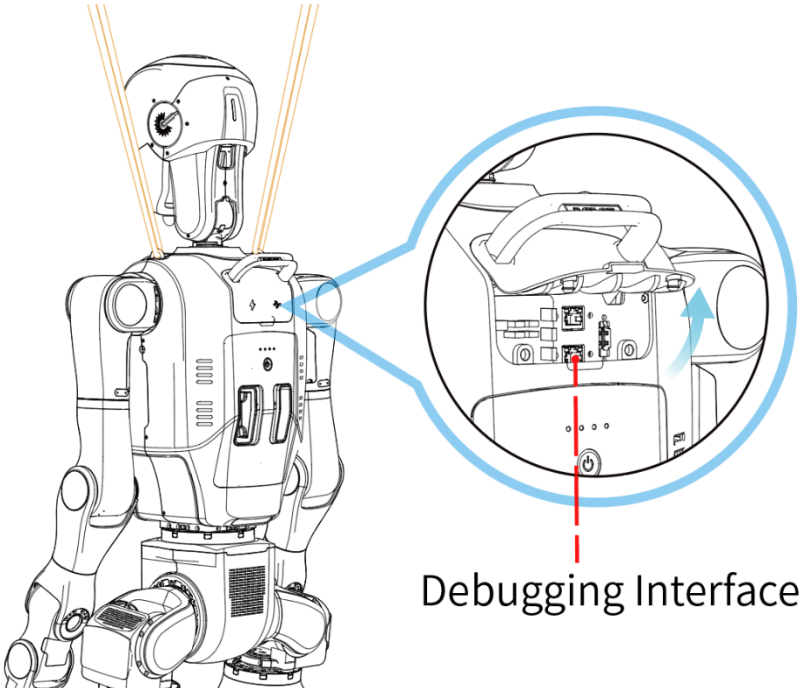
https://github.com/IntelRealSense/librealsense/tree/master/examples

 Copyright © SHENZHEN DOBOT CORP LTD

6. Debugging Instructions

1. Hang the ATOM robot and press the **LB+A** button on the remote controller to control it in damping state.
2. Press the **LT+RT** button on the remote controller, and ATOM will enter debugging state. Under debugging state, the robot defaults to a damping state. Development and debugging can be performed using the underlying control SDK to directly drive the motors. It supports customized teleoperation, deployment and control of reinforcement learning models, as well as real-time monitoring of battery, motor, and IMU data.

   The interface for debugging is located under the fixing ring on the back of the robot, and can be accessed by uncovering the silicone shell, as shown in the figure below:



3. Press the **LT+A** button, and ATOM will enter diagnostic state and assume a specific diagnostic posture. Press the **LT+B** button again, and ATOM will return to debugging state.

> **NOTE**
>
> This process is used to confirm whether ATOM has successfully entered the debugging state or for hardware troubleshooting.

 Copyright © SHENZHEN DOBOT CORP LTD

7. Common Errors and Interpretations

7.1 Power Supply Error

ID	Level	Error Description	Solution
256	1	Lower Limb Power Supply Error	Check if the hardware is functioning properly, or contact a technical support engineer
257	1	Upper Limb Power Supply Error	
258	1	24V1 Power Supply Error	
259	1	24V2 Power Supply Error	
260	1	12V Power Supply Error	
261	1	Peripheral Power Supply Error	
262	1	Charging Port Power Supply Error	

7.2 Voltage Error

ID	Level	Error Description	Solution
263	1	Lower Limb Voltage Too Low	Check if the hardware is functioning properly, or contact a technical support engineer
264	1	Lower Limb Voltage Too High	
265	1	Upper Limb Voltage Too Low	
266	1	Upper Limb Voltage Too High	

 Copyright © SHENZHEN DOBOT CORP LTD

7.3 Battery Error

ID	Level	Error Description	Solution
512	1	Cell Overvoltage	If a single cell voltage exceeds 4.2V, please check if the hardware is functioning properly, or contact a technical support engineer.
513	1	Cell Undervoltage	If a single cell voltage falls below 2.5V, please check if the hardware is functioning properly, or contact a technical support engineer.
514	1	Battery Failure	If a single cell voltage falls below 2V, please check if the hardware is functioning properly, or contact a technical support engineer.
515	1	Battery Pack Overvoltage	When charging, if the battery pack voltage exceeds 70.5V, please check if the hardware is functioning properly or contact a technical support engineer.
516	1	Battery Pack Undervoltage	If the battery pack voltage falls below 40V, also check the hardware or contact a technical support engineer.
517	1	Charging Overcurrent	During charging, if the current exceeds 12A for more than 20ms, check if the hardware is functioning properly or contact a technical support engineer.
518	1	Discharging Overcurrent	During discharging, if the current exceeds 100A for more than 20ms, check if the hardware is functioning properly or contact a technical support engineer.
519	1	Charging Short Circuit	During charging, if the current exceeds 15A for more than 1.6ms, please check if the hardware is functioning properly or contact a technical support engineer.
520	1	Discharging Short Circuit	During discharging, if the current exceeds 120A for more than 200μs, check if the hardware is functioning properly or contact a technical support engineer.
521	1	Chip Overtemperature	If the internal temperature monitoring of the BMS chip is abnormal, please check if the hardware is functioning properly or contact a technical support engineer.
522	1	Battery Overtemperature	If the battery temperature exceeds 50°C during charging or 60°C during discharging, check if the hardware is functioning properly or contact a technical support engineer.
523	1	Battery Low Temperature	If the battery temperature is below 2°C during charging or below -18°C during discharging, please check if the hardware is functioning

 Copyright © SHENZHEN DOBOT CORP LTD

			properly or contact a technical support engineer.
524	1	PCB Overtemperature	If the NTC temperature on the PCB exceeds 85°C, please check if the hardware is functioning properly or contact a technical support engineer.
525	1	Open Circuit Alarm	If the battery sampling line has come off or there is an open circuit in the cell series connection, please check if the hardware is functioning properly or contact a technical support engineer.
526	1	Cell Mismatch	If the maximum voltage difference between the cells exceeds 0.234V, please check if the hardware is functioning properly or contact a technical support engineer.
527	1	MOSFET Driver Malfunction	If there is a MOSFET timeout, typically due to a soft-start error, please check if the hardware is functioning properly or contact a technical support engineer.
528	1	MCU Error Alarm	If there is an abnormal communication between the MCU and the BMS chip, please check if the hardware is functioning properly or contact a technical support engineer.
529	1	Sensor Alarm	If the ADC self-diagnosis of the MP2797 chip is abnormal, please check if the hardware is functioning properly or contact a technical support engineer.

7.4 File Parsing Error

ID	Level	Error Description	Solution
1280	1	Failed to parse alarm_host.json file	System reports a file error. Please contact a technical support engineer.
1281	1	Failed to parse alarm_servo.json file	Servo reports a file error. Please contact a technical support engineer.
1282	1	Failed to parse Offset.json file	Joint offset file error. Please contact a technical support engineer.
1283	1	Failed to parse equipment_config.json file	Peripheral configuration file error. Please contact a technical support engineer.
1284	1	Failed to parse robotType.json file	Robot type file error. Please contact a technical support engineer.
1285	1	Failed to parse snCode.json file	SN code file error. Please contact a technical support engineer.
1286	1	Failed to parse is_virtual.json file	Virtual axis file error. Please contact a technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD

1287	1	Failed to parse offset.json.default file	Factory joint offset backup file error. Please contact a technical support engineer.
1288	1	Failed to parse Offset.json.bak file	User joint offset backup file error. Please contact a technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD

7.5 File Modification Error

ID	Level	Error Description	Solution
1289	1	Failed to modify Offset.json file	Joint offset file error. Please contact a technical support engineer.
1290	1	Failed to modify offset.json.default file	Factory joint offset file error. Please contact a technical support engineer.
1291	1	Failed to modify offset.json.bak file	User joint offset backup file erroer. Please contact a technical support engineer.
1292	1	Failed to modify equipment_config.json file	Peripheral configuration file error. Please contact a technical support engineer.
1293	1	Failed to modify is_virtual.json file	Virtual axis file error. Please contact a technical support engineer.

7.6 Other Errors

ID	Level	Error Description	Solution
1294	1	Enable Failure	Enable failure, please contact technical support engineer.
1295	1	can1 Board Offline	Joint offset file error, please contact technical support engineer.
1296	1	can 2 Board Offline	Factory joint offset file error, please contact technical support engineer.
1297	1	Position Exceeded Limit	Servo position exceeds limit, please contact technical support engineer.
1298	1	Speed Exceeded Limit	Servo speed exceeds limit, please contact technical support engineer.
1299	1	Torque Exceeded Limit	Servo torque exceeds limit, please contact technical support engineer.
1300	1	Zero Calibration Failure	Zero calibration failure, please contact technical support engineer.
1301	1	Bus Communication Error	Bus communication error, please contact technical support engineer.
1302	1	Mode Switching Failure	Mode switching failure, please contact technical support engineer.
1303	1	imu Zero Calibration Failure	imu zero calibration failure, please contact technical support engineer.
1304	1	Log File Compression Failure	Log file compression failure, please contact technical support engineer.
1305	1	Joint Offset Modification Failure	Joint offset modification failure, please contact technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD

1306	1	Dexterous Hand Configuration File Error	Dexterous hand configuration file error, please contact technical support engineer.
1307	1	Position Limit File Error	Position over-limit file error, please contact technical support engineer.
1308	1	Speed Limit File Error	Speed over-limit file error, please contact technical support engineer.
1309	1	Torque Limit File Error	Torque over-limit file error, please contact technical support engineer.
1310	1	Position Limit File Modification Failed	Position over-limit file modification failure, please contact technical support engineer.
1311	1	Speed Limit File Modification Failed	Speed over-limit file modification failure, please contact technical support engineer.
1312	1	Torque Limit File Modification Failed	Torque over-limit file modification failure, please contact technical support engineer.
1314	1	AtomOs.version File Parsing Failed	AtomOs.version file parsing failure, please contact technical support engineer.
1315	1	Abnormal power Heartbeat	Abnormal power heartbeat, please contact technical support engineer.
1316	1	Abnormal bms Heartbeat	Abnormal bms heartbeat, please contact technical support engineer.
8752	1	Hardware Overcurrent Protection	The actual current value collected by the servo is greater than 90% of the hardware's maximum sampled current. Check whether the hardware is functioning properly, or contact technical support engineer.
8992	1	Software Overcurrent Protection	Power down and restart, or contact technical support engineer.
9040	1	Module Overload Protection	Check whether the hardware is functioning properly, or contact technical support engineer.
9088	1	Zero-point Current Offset Excessive Protection	Reset the zero point current offset.
12592	1	Drive Input Phase Loss Protection	Check whether the hardware is functioning properly, or contact technical support engineer.
12816	1	PN Overvoltage Protection	Servo detects bus voltage value greater than the set threshold, check whether the hardware is functioning properly, or contact technical support engineer.
12832	1	PN Undervoltage Protection	Servo detects bus voltage value lower than the set threshold, check whether the hardware is functioning properly, or contact technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD

12928	1	Servo and Motor Power Mismatch Error	Check whether the hardware is functioning properly, or contact technical support engineer.
13184	1	Drive Output Phase Sequence Error Protection	Connect using the correct phase sequence wiring.
17168	1	Drive Overheat Protection	Check whether the hardware is functioning properly, or contact technical support engineer.
17169	1	Thermistor Damaged or Not Installed	Check whether the hardware is functioning properly, or contact technical support engineer.
21008	1	Drive Board Identification Error Protection	Check whether the hardware is functioning properly, or contact technical support engineer.
21120	1	FPGA Configuration Error	System error, please contact technical support engineer.
21121	1	Internal Error	System error, please contact technical support engineer.
21122	1	STO Safety Wiring Error Protection	System error, please contact technical support engineer.
21569	1	Drive Board Connection Error Protection	Check whether the hardware is functioning properly, or contact technical support engineer.
25378	1	DI Function Duplicate Assignment	Check the DI group parameters to ensure the same function is not assigned to different DIs.
25379	1	DO Function Assignment Exceeds Limit	Check if the communication cables are securely connected.
25382	1	Software Position Upper/Lower Limit Setting Error	Correctly set the values for 0x607D-01h and 0x607D-02h.
25383	1	Home Offset Setting Error	Correctly set the values for 0x607D-01h and 0x607D-02h, ensuring the home offset value 0x607C is between them.
28961	1	Stall Motor Overheat Protection	Check if the brake is released properly or if the motor is jammed due to mechanical error.
29056	1	Motor Overload Protection	Check if the robot's end load exceeds the robot's rated load or consult technical personnel to replace the motor.
29568	1	Encoder Communication Disconnection Protection	Check encoder wiring or replace/repair the encoder.
29569	1	Encoder Error Fault	Check whether the hardware is functioning properly, or contact technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD

29570	1	Encoder Battery Error Protection	Please contact technical support engineer.
29571	1	Encoder Battery Power Supply Error	System error, please contact technical support engineer.
29572	1	Main Encoder CRC Check Error	System error, please contact technical support engineer.
29573	1	Auxiliary Encoder Disconnection Error	System error, please contact technical support engineer.
29574	1	Auxiliary Encoder Internal Error	System error, please contact technical support engineer.
29575	1	Auxiliary Encoder CRC Check Error	System error, please contact technical support engineer.
29576	1	pwm Output buffer Error	Replace the driver.
29577	1	Brake Power Supply Error	Check if the brake is connected or damaged.
29584	1	Excessive Joint Position Deviation	1. Check the production parameters. 2. Check the encoder installation. 3. Check the drive structure.
30080	1	Servo Bus Communication Error	System error, please contact technical support engineer.
30081	1	XML Configuration File Not Programmed	System error, please contact technical support engineer.
30082	1	Synchronization Cycle Configuration Error	System error, please contact technical support engineer.
30083	1	Master Station Synchronization Signal Loss	System error, please contact technical support engineer.
30084	1	Excessive Synchronization Cycle Error	System error, please contact technical support engineer.
33920	1	Exceeds Maximum Speed Protection	Check if the motor wiring sequence is correct.
33921	1	Excessive Speed Deviation Protection	Check that the hardware is functioning properly and restart the system, or contact a technical support engineer.
33922	1	Motor Stall Protection	Check if the motor angle is normal or if the motor is damaged.
34321	1	Excessive Position Deviation Protection	System error, please contact technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD

34322	1	Position Command Over-Limit Protection	Please enter the correct parameters.
34323	1	Full Closed-Loop Excessive Position Deviation	Check whether the hardware is functioning properly, or contact technical support engineer.
65282	1	Angle Identification Failure	Check whether the hardware is functioning properly, or contact technical support engineer.
65344	1	Parameter Identification Failure	Check whether the hardware is functioning properly, or contact technical support engineer.

 Copyright © SHENZHEN DOBOT CORP LTD