优 必 选

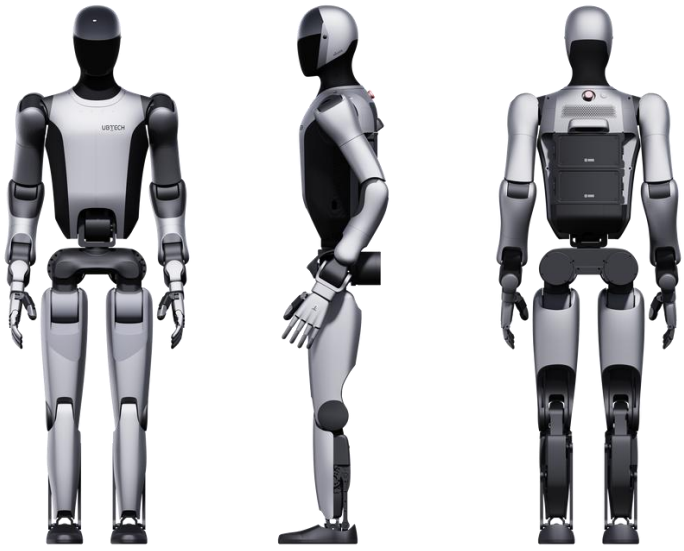
UBTECH

UBTECH Walker S2 SDK Secondary Development Document [External]

Version Management

> The content of this document may change due to product updates. Please visit https://ew9vfg36e3.feishu.cn/docx/DtPxddKKroKJtRx3eOmcbMuhnlh?from=from_copylink for the latest version.
>
> Revision Record

Revision Record				
No.	**Date**	**Revision**	**Modification Description**	**Author**
1	2025/10/31	V1.0	First version of public documentation	SDK Project Team



1. About Ubtech Walker S2

Walker S2 is an industrial humanoid robot launched by Ubtech, integrating multiple embodied AI technologies. It possesses core capabilities including high payload, autonomous navigation, and multi-scenario adaptation.

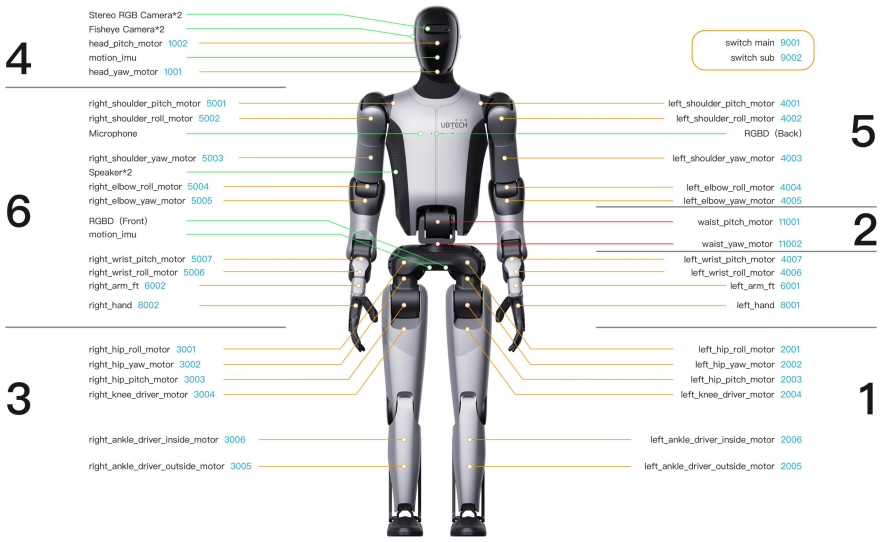
The Walker S2 robot has a total of 42 degrees of freedom (DoF), driven by 42 integrated joint motors for precise motion control. Its structure includes: 2 DoF in the head, 7 DoF per arm (shoulder, elbow, wrist), 2 DoF in the waist, 6 DoF per hand (3rd generation hand), and 6 DoF per leg (hip, knee, ankle).

1.1. General Specifications

Parameter Category	Specific Parameter
Height	176cm
Arm Span	177cm (excluding dexterous hand)
Weight	70kg (excluding dexterous hand)
DoF per Leg	6
DoF in Waist	2
DoF per Hand (excl. hand)	7

DoF in Head	2
Payload Capacity	Max 15kg for both arms
Computing Configuration	X86 + NVIDIA Jetson Orin
Sensor Configuration	Depth Camera, 6-Axis Force Sensor, IMU, Fisheye Camera
Display & Interaction	4-inch round interactive screen, battery status indicator, microphone array, speaker
Power Supply Method	Li-ion Battery DC 48V
Battery Charging Voltage/Current	Voltage: DC 54V Current: 8A
Comprehensive Endurance	2.5h
Intelligent OTA Update	Yes
Operating System	Ubuntu + ROSA 2.0

1.2. Joint Parameters



Body Part	Name	Motion Range (rad)	Motion Range (°)	Rated Speed (rpm)	Max Speed (rpm)	Rated Torque (Nm)	Peak Torque (Nm)
Head	Head Pitch Motor	lower: -0.6807 upper: 0.5061	lower: -39.00 upper: 29.00	30	50	3	4.5
	Head Yaw Motor	lower: -1.6406 upper: 1.6406	lower: -94.00 upper: 94.00	30	50	3	4.5
Left Arm	Left Shoulder Pitch Motor	lower: -2.8274 upper: 2.8274	lower: -162.00 upper: 162.00	20	30	27	80
	Left Shoulder Roll Motor	lower: -1.85 upper: 0.0873	lower: -106.00 upper: 5.00	20	30	27	80
	Left Shoulder Yaw Motor	lower: -2.8972 upper: 2.8972	lower: -166.00 upper: 166.00	20	30	17	45
	Left Elbow Roll Motor	lower: -2.6180 upper: 0.0	lower: -150.00 upper: 0.00	20	30	17	45

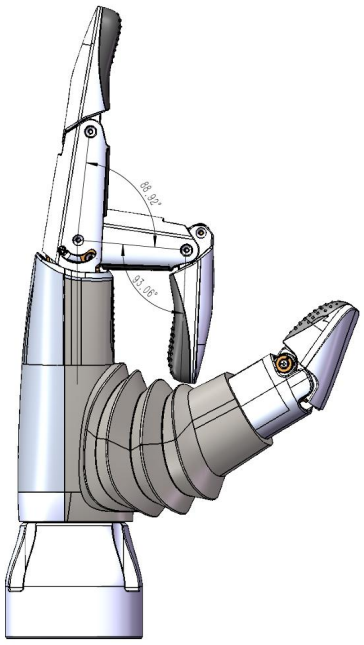
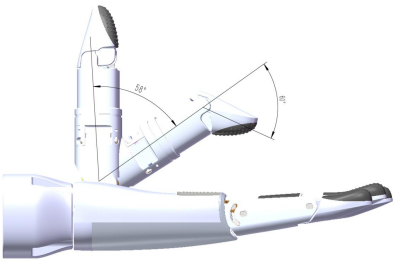
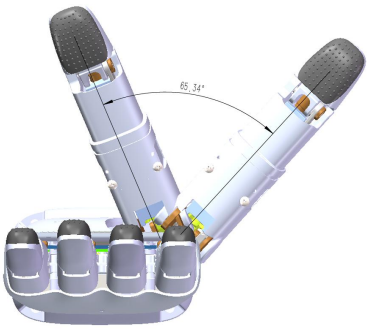
	Left Elbow Yaw Motor	lower: -2.9147 upper: 2.9147	lower: -167.00 upper: 167.00	20	30	17	20
	Left Wrist Pitch Motor	lower: -1.5882 upper: 1.5882	lower: -91.00 upper: 91.00	20	30	17	20
	Left Wrist Roll Motor	lower: -1.9897 upper: 1.9897	lower: -114.00 upper: 114.00	20	30	17	20
Right Arm	Right Shoulder Pitch Motor	lower: -2.8274 upper: 2.8274	lower: -162.00 upper: 162.00	20	30	27	80
	Right Shoulder Roll Motor	lower: -1.85 upper: 0.0873	lower: -106.00 upper: 5.00	20	30	27	80
	Right Shoulder Yaw Motor	lower: -2.8972 upper: 2.8972	lower: -166.00 upper: 166.00	20	30	17	45
	Right Elbow Roll Motor	lower: -2.6180 upper: 0.0	lower: -150.00 upper: 0.00	20	30	17	45

	Right Elbow Yaw Motor	lower: -2.9147 upper: 2.9147	lower: -167.00 upper: 167.00	20	30	17	20
	Right Wrist Pitch Motor	lower: -1.5882 upper: 1.5882	lower: -91.00 upper: 91.00	20	30	17	20
	Right Wrist Roll Motor	lower: -1.9897 upper: 1.9897	lower: -114.00 upper: 114.00	20	30	17	20
Waist	Waist Pitch Motor	lower: -1.5533 upper: 0.6109	lower: -89.00 upper: 35.00	15	20	79	265
	Waist Yaw Motor	lower: -2.7925 upper: 2.7925	lower: -160.00 upper: 160.00	20	37	35	85
Left Leg	Left Hip Roll Motor	lower: -0.4014 upper: 0.7679	lower: -23.00 upper: 44.00	50	80	75	225
	Left Hip Yaw Motor	lower: -1.0472 upper: 0.7854	lower: -60.00 upper: 45.00	50	80	22	65

	Left Hip Pitch Motor	lower: -0.5236 upper: 1.9024	lower: -30.00 upper: 109.00	50	80	75	225
	Left Knee Drive Motor	lower: -2.2515 upper: 0.0524	lower: -129.00 upper: 3.00	50	80	75	225
	Left Ankle Drive Inside Motor	lower: -1.1868 upper: 0.6632	lower: -68.00 upper: 38.00	50	80	22	65
	Left Ankle Drive Outside Motor	lower: -0.6632 upper: 1.1868	lower: -38.00 upper: 68.00	50	80	22	65
Right Leg	Right Hip Roll Motor	lower: -0.7679 upper: 0.4014	lower: -44.00 upper: 23.00	50	80	75	225
	Right Hip Yaw Motor	lower: -0.7854 upper: 1.0472	lower: -45.00 upper: 60.00	50	80	22	65
	Right Hip Pitch Motor	lower: -0.5236 upper: 1.9024	lower: -30.00 upper: 109.00	50	80	75	225

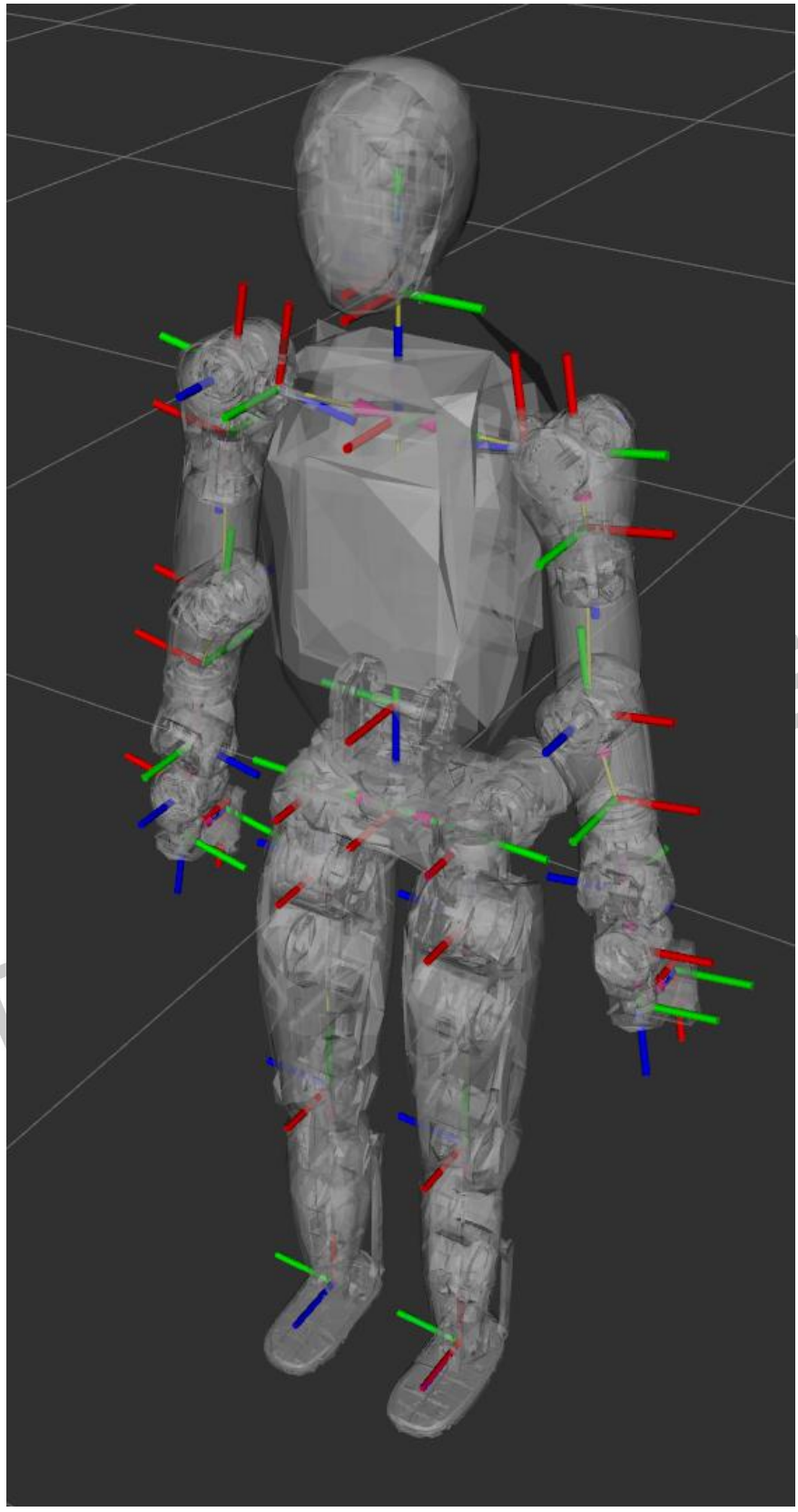
	Right Knee Drive Motor	lower: -2.2515 upper: 0.0524	lower: -129.00 upper: 3.00	50	80	75	225
	Right Ankle Drive Inside Motor	lower: -1.1868 upper: 0.6632	lower: -68.00 upper: 38.00	50	80	22	65
	Right Ankle Drive Outside Motor	lower: -0.6632 upper: 1.1868	lower: -38.00 upper: 68.00	50	80	22	65
3rd Gen Dexterous Hand(Note 1: Left/Right hand identical. Fingertips are passive joints, no motor)(Note 2: Drive is linear; Speed and Torque correspond to linear velocity and thrust respectively)	Four Fingers Palmar Flexion Motor	lower: 0.0 upper: 1.46	+0.00~+83.66	25mm/s	39mm/s	Rated thrust 45.53N at rated torque	Peak thrust 100N at peak torque
	Thumb Flexion Motor	lower: 0.0 upper: 1.04	+0.00~+59.59				
	Thumb Rotation Motor	lower: 0.0 upper: 0.96	+0.00~+55.00				

Joint Part (3rd Gen Dexterous Hand)	Motion Range (°)	Illustration Description

Little Finger, Ring Finger, Middle Finger, Index Finger	Palmar rotation angle 88.92° (1.52 rad), Fingertip passive rotation 93.06° (1.62 rad)	
Thumb Flexion Angle	Thumb palmar angle 58° (1.01 rad), Fingertip passive rotation 60° (1.05 rad)	
Thumb Rotation Angle	Thumb abduction angle 65.34° (1.14 rad)	

1.3. Coordinate System, Joint Rotation Axes, and Joint Zero Points

When all joints are at zero degrees, the coordinate systems are as shown in the figure below. Red is the x-axis, green is the y-axis, and blue is the z-axis.



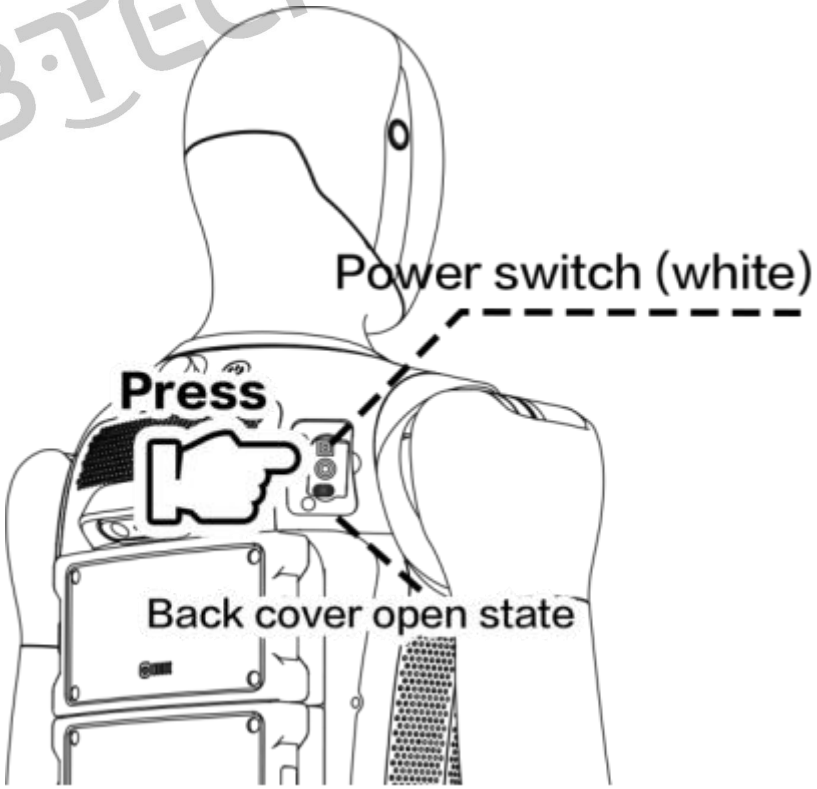
2. Quick Operation Guide

2.1. Risk Warning

- Confirm the robot is suspended on the protective frame, with feet at least 20cm off the ground, before debugging.
- Ensure there are no obstacles within the robot's motion range during movement to avoid collisions.
- Do not touch the robot's limbs during movement.
- Do not press the power button while the robot is in a standing state, unless in an emergency, as this will deactivate the servos, causing the legs to lose support and the robot to fall.
- In case of emergencies, press the emergency stop button on the robot's back and hoist the robot onto the display frame.

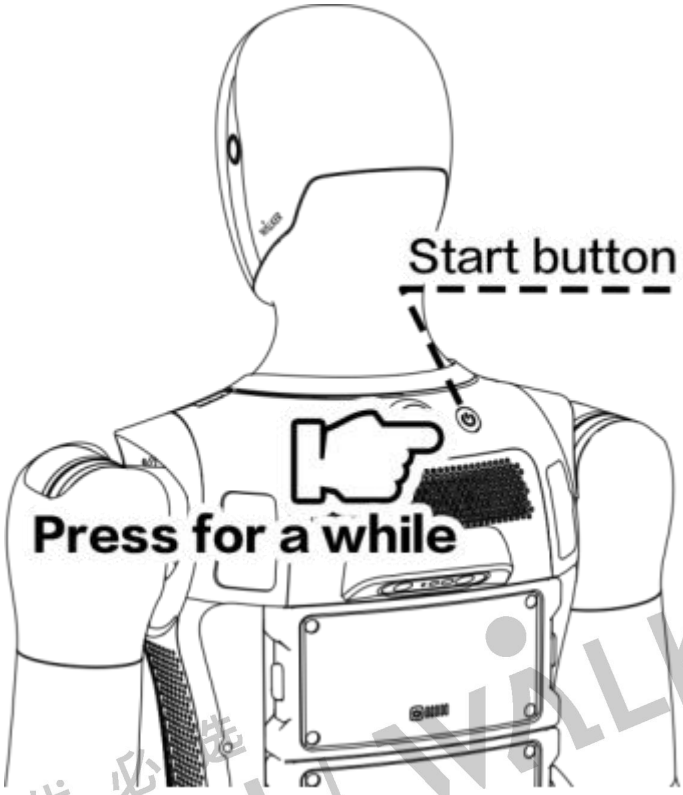
2.2. Power On

- Before powering on, ensure the robot is hung on the support frame: The robot's feet should hang naturally, the frame should be raised to its highest point to ensure the feet do not touch the ground or other objects during leg reset. Ensure the robot's arms hang naturally, with the inner elbows and palms facing forward, the head facing down and forward, the emergency stop button on the back is released, and the battery is connected.
- Open the back cover panel and press the power switch on the robot's back. (Figure-1)



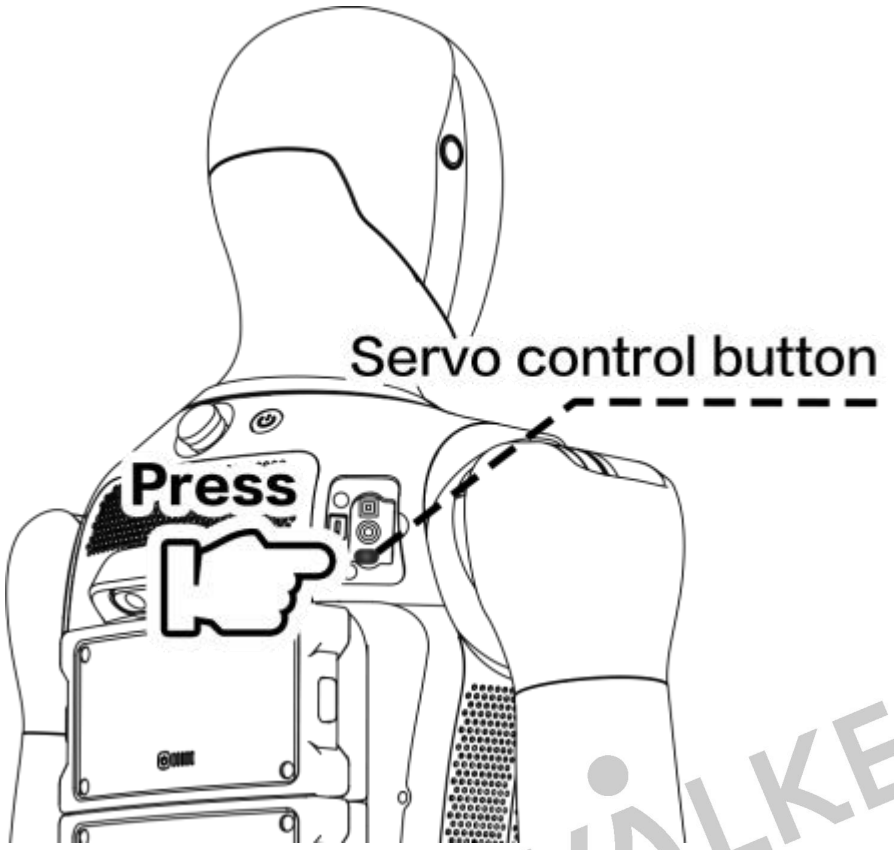
(Figure-1)

- Press and hold the startup button on the robot's back. The robot has started booting when the head display lights up. Boot time is approximately 1 minute. (Figure-2)



(Figure-2)

- Press the servo control button on the robot's back; the LED lights on the legs will illuminate, indicating the robot startup is complete. (Figure-3)



(Figure-3)

2.3. Connect to the Robot

1. Connect to the Walker S2 robot via Ethernet cable: Use a network cable to connect to the robot's Ethernet port.

2. Then open network settings, find the network adapter connected to the robot, go to IPv4 settings, change the IPv4 method to Manual, set the Address to `192.168.11.99`, set the Netmask to `255.255.255.0`, click Apply, and wait for the network to reconnect.

3. Via SSH, log in to the ROS 2 container on PC1 or PC2 and run the Demo program or a custom program (`ssh ubt@192.168.11.2/3 -p 2222` Password: `Ubtubt@9880`).

2.4. Zeroing / Starting the Controller

1. Confirm the robot is suspended on the protective frame, with feet at least 20cm off the ground, and the emergency stop button is released.

2. Use the remote control for zeroing / controller startup (Operation method see section 2.6).

3. After successful zeroing / controller startup, you can now enter developer mode via command and read various sensor values through the SDK.

2.5. Enter Developer Mode

1. Ensure the robot controller has been zeroed/started.
2. SSH remotely into the ROS 2 Docker container development environment.

Host	Architecture	Address	Username@Password
PC1	x86	192.168.11.2:2222	ubt@Ubtubt@9880
PC2	arm	192.168.11.3:2222	ubt@Ubtubt@9880

```Bash
ssh -p 2222 ubt@192.168.11.2/3
# Enter password: Ubtubt@9880
```

3. Enter Developer Mode
   - In the ROS 2 container, run the command:

     ```Bash
     # Request service to enter developer mode (true to enter, false to exit)
     ros2 service call /sys/task/developer_mode std_srvs/srv/SetBool "{data: true}"
     ```

   - When it returns `success=True`, entry/exit is successful, and `message` shows `developer enable`.

     ```
     response:
     std_srvs.srv.SetBool_Response(success=True, message='Developer mode enabled.')
     ```

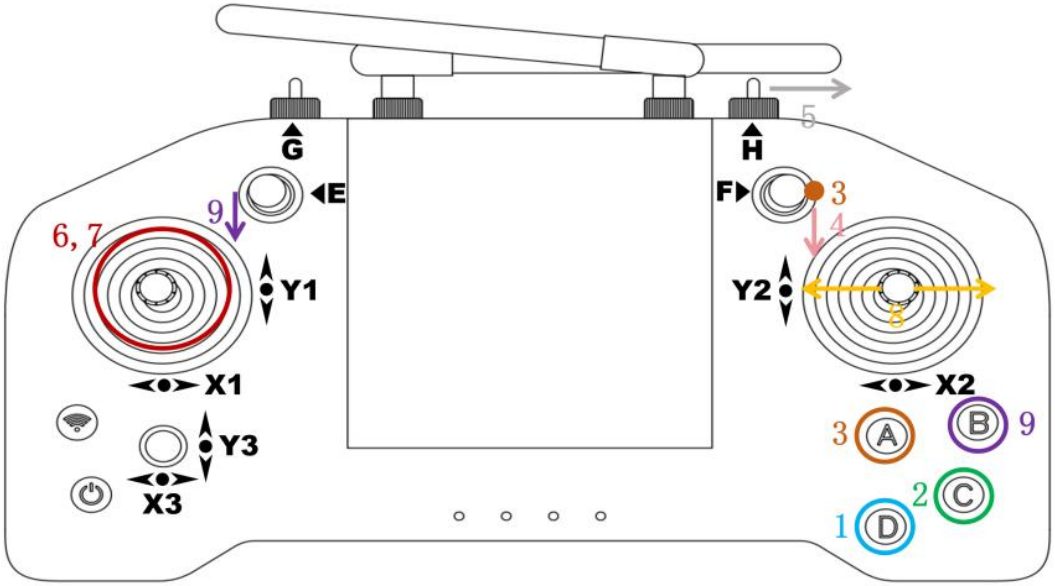
   - You can also check the robot's current status via topic:

     ```Bash
     # true is developer mode, false is normal mode
     ros2 topic echo /sys/state/walker_mode
     ```

   - When it returns `data: true`, you can start using the SDK for development and debugging. If entering developer mode fails, please check:
     - Is the emergency stop button released?
     - Was the robot controller started successfully

2.6. Remote Control Operation

1. Joystick Button Definitions



2. Remote Control Button Functions

Remote Control Button Function Description			
serial number	**Function**	**Button**	**Remarks**
1	Return to Zero + Retract Leg	Press F key down + D	The joint hardens from softness and enters the locked position mode, allowing the sling to be released.
2	Locking Mode	F key up + C	Robot's whole body joints are stiff, in wooden man mode (Note: the robot can stand, but will fall over with a touch)
3	Damping Mode	F key down + C	The damping mode does not mean being completely unable to stand still, nor does it mean being completely able to stand on its own, and requires human

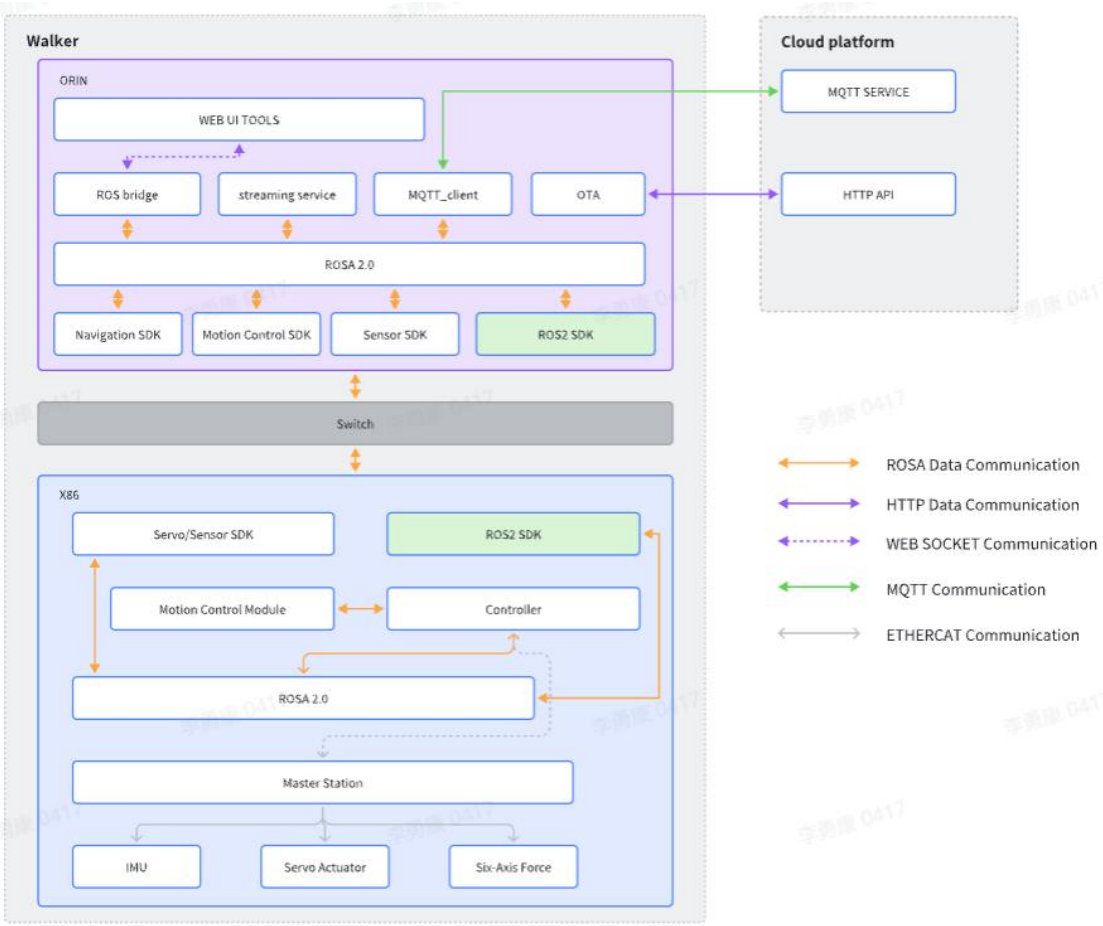
			assistance.
4	Force-controlled standing mode	H - Middle Dial Lever +A	1. In this mode, it is not allowed to push the robot to walk. 2. In this mode, the robot can stand autonomously and will balance itself after being pushed. 3. During the task flow, the force control mode can be entered to interrupt the task flow. 4. After the robot returns to the force control standing mode, the task flow is reset to its initial state. 5. Only the standing mode can enter the locked position mode.
5	Walking mode (start stepping, respond to speed commands)	H - Move the lever to the left and then reset it (reset: move the lever to the middle)	Pre-state reserved for running mode, preprocessing stage, can push the handlebar to walk
6	Running	H - Move the lever to the right and then reset it	Tip: To stop running, you need to first switch to walking mode (treadmill state), then press the "H left toggle and then reset" button, then press A to stop. You can push the handlebar to walk.
7	Move left and right	X1 - Left Joystick: Left/Right	The distance of the putter and the magnitude of the speed are mapped proportionally
8	Move forward and backward	Y1 - Left Joystick: Up and Down	
9	Turn counterclockwise, turn clockwise	X2 - Right Joystick: Left and Right	Left - Counterclockwise Right - Clockwise
10	Task Flow	E - Up lever (perform the previous action) E - In the middle	Only the force control mode can enter the task flow mode Press E a few times, and it will

		(execute the current action) E - Down lever (perform the next action) G - Push the lever to the right (start performing the action)	skip several processes
11	Power On & Shutdown	Power On: Press F+D : When the legs are bent, the robot can be lowered. Please hold the machine firmly and stand still for 20 seconds (check if stillness is required), then press H middle +**A Stand**; Shutdown: After securing the ropes during the racking process**, first press F↓+C**. At this time, the machine will lose its center of gravity, so please make sure to hold the machine firmly, and then the machine can be lifted.	After the rope is properly fixed, be sure to**press C**, otherwise the machine leg will shake, which will in turn affect the detection of IMU data.

3. Application Development

3.1. Software Architecture Description

- System Architecture Diagram



3.1.1. Cloud Platform

The Cloud Platform provides unified remote management and operational support for Ubtech robots, helping users achieve efficient device monitoring, remote upgrades, and multi-robot collaboration services. Main functions include:

1. Operational Data Collection & Fault Detection
    a. The platform automatically collects key data during robot operation (does not involve user privacy).
    b. Through big data analysis and intelligent algorithms, enables fault detection, statistics, and early warning, improving the stability and maintainability of the robot system.
2. System OTA Upgrade
    a. Supports remote online upgrades, ensuring the robot system can iterate continuously.
    b. Through unified version management and distribution mechanisms, enables rapid deployment and maintenance of new features.
3. IoT Communication & Task Scheduling
    a. Uses MQTT protocol to establish an efficient communication channel between the cloud and robots.

b. Provides remote fault monitoring, task distribution, and multi-robot collaborative scheduling functions, supporting centralized management in complex scenarios.

3.1.2. ROSA 2.0

Ubtech's independently developed Robot Operating System Application framework ROSA 2.0 deeply integrates various robot functions and controls, providing solid support for the flexible application of robots in various scenarios, ensuring the autonomous controllability and security of underlying algorithms.

3.2. ROS 2 SDK Overview

The full version of the Ubtech SDK, developed by Ubtech Robotics, provides rich interfaces covering motor control for the head, arms, elbows, hands, and legs, as well as the use of 6-axis force sensors, IMU (Inertial Measurement Unit), and cameras. It is used for writing and deploying robot applications, aiming to help developers quickly and flexibly build their own applications to precisely control and use the robot to meet needs in different application scenarios. You can follow the interfaces and examples we provide, along with this development guide, to complete secondary development for Walker S2.

3.3. ROS 2 SDK Acquisition

At this stage, to obtain the ROS SDK demo program, please contact the Ubtech company contact person at `yongkang.li@ubtrobot.com`. In the future, we will provide a public download link on the official website.

3.3.1. ROS 2 SDK Build

- Ensure correct network connection, then enter the robot's ROS 2 container:

```Bash
ssh -p 2222 ubt@192.168.11.2/3
# Enter password: Ubtubt@9880
```

- You can then download and build our ROS 2 demo program:

```Bash
# Copy the demo program to the /debug directory
scp -P 2222 -r ubt_ros2_demo ubt@192.168.11.3:/debug/
cd /debug/ubt_ros2_demo
source /opt/ros/humble/setup.bash
# Build the ubt_ros2demo program
colcon build
```

- In this project structure:
  - `example` stores the example demo code for the SDK.
  - `low_level` contains low-level SDK calls, `common` contains related common interfaces, such as audio playback, etc.

3.3.2. Run and Test

After completing the code build above, enter:

```
Bash
source /debug/ubt_ros2_demo/install/setup.bash
```

Please run programs related to motor control and status subscription on the motion control board (`ubt@192.168.11.2 -p 2222`), and use `sudo su` to become the root user.

You can run our demo programs. Also, you can open a terminal and enter:

```
Bash
ros2 topic list
```

You can see the following topics and can print the corresponding data using `ros2 topic echo`.

```
/parameter_events
/rosout
/sensor/camera/body_back_rgbd/color/raw
/sensor/camera/body_back_rgbd/depth/info
/sensor/camera/body_back_rgbd/depth/raw
/sensor/camera/body_back_rgbd/mix/raw
/sensor/camera/fisheye_left/image/info
/sensor/camera/fisheye_left/image/raw
/sensor/camera/fisheye_left/sn
/sensor/camera/fisheye_right/image/info
/sensor/camera/fisheye_right/image/raw
/sensor/camera/fisheye_right/sn
/sensor/camera/rgb_camera_manager/transition_event
/sensor/camera/stereo_left/image/info
/sensor/camera/stereo_left/image/raw
/sensor/camera/stereo_left/sn
/sensor/camera/stereo_right/image/info
/sensor/camera/stereo_right/image/raw
/sensor/camera/stereo_right/sn
/sys/node_state
/sys/speech/asr
```

- Demo Program Examples

Example source code is located in the `/example/src` folder. We provide the following example programs:

- `audio_player`: Plays a fixed audio file using Walker S2's built-in speaker.
- `powerboardstate_subscriber`: Reads Walker S2 power board status information.
- `batterystate_subscriber`: Subscribes to Walker S2 power status information.
- `pub_arm_command`: Publishes arm control commands.
- `pub_hand_command`: Publishes dexterous hand control commands.
- `pub_head_command`: Publishes head control commands.
- `sub_hands_state`: Subscribes to hand status information.
- `sub_robot_state`: Subscribes to robot status information (IMU, 6-axis force, joint motors, etc.).
- `camera_subscriber`: Example programs for subscribing to various cameras.

- Running Example Code

a. Take the IMU subscription demo as an example, open a terminal and enter:

```Bash
ros2 run ubt_ros2_example imu_subscriber
```

b. You can then obtain IMU (Orin) status information.

```
ubt@motion:/debug$ ros2 run ubt_ros2_example imu_subscriber
[INFO] [1758543179.788544882] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.788983814] [imu_subscriber]: Angular velocity: [x=0.000, y=-0.001, z=0.002] rad/s
[INFO] [1758543179.789053326] [imu_subscriber]: Linear acceleration: [x=0.222, y=-5.080, z=8.381] m/s^2
[INFO] [1758543179.798115075] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.798156451] [imu_subscriber]: Angular velocity: [x=-0.001, y=0.001, z=-0.000] rad/s
[INFO] [1758543179.798166355] [imu_subscriber]: Linear acceleration: [x=0.238, y=-5.072, z=8.376] m/s^2
[INFO] [1758543179.799182744] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.799280806] [imu_subscriber]: Angular velocity: [x=0.001, y=0.001, z=-0.001] rad/s
[INFO] [1758543179.799304555] [imu_subscriber]: Linear acceleration: [x=0.215, y=-5.072, z=8.395] m/s^2
[INFO] [1758543179.809380421] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.809581697] [imu_subscriber]: Angular velocity: [x=0.001, y=-0.002, z=-0.002] rad/s
[INFO] [1758543179.809593327] [imu_subscriber]: Linear acceleration: [x=0.229, y=-5.064, z=8.361] m/s^2
[INFO] [1758543179.814352345] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.814390257] [imu_subscriber]: Angular velocity: [x=0.001, y=-0.002, z=0.002] rad/s
[INFO] [1758543179.814399827] [imu_subscriber]: Linear acceleration: [x=0.207, y=-5.062, z=8.358] m/s^2
[INFO] [1758543179.817806568] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.817883934] [imu_subscriber]: Angular velocity: [x=-0.000, y=0.001, z=0.001] rad/s
[INFO] [1758543179.817897652] [imu_subscriber]: Linear acceleration: [x=0.222, y=-5.053, z=8.368] m/s^2
[INFO] [1758543179.823348195] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.823418228] [imu_subscriber]: Angular velocity: [x=0.000, y=-0.002, z=0.001] rad/s
[INFO] [1758543179.823501905] [imu_subscriber]: Linear acceleration: [x=0.228, y=-5.055, z=8.374] m/s^2
[INFO] [1758543179.828256738] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.828298224] [imu_subscriber]: Angular velocity: [x=-0.001, y=0.002, z=0.001] rad/s
[INFO] [1758543179.828315086] [imu_subscriber]: Linear acceleration: [x=0.231, y=-5.063, z=8.382] m/s^2
[INFO] [1758543179.831956306] [imu_subscriber]: Orientation: [w=0.963, x=-0.268, y=-0.012, z=0.000]
[INFO] [1758543179.831980577] [imu_subscriber]: Angular velocity: [x=0.000, y=0.001, z=0.000] rad/s
```

- For other detailed examples, see Module 5.

3.4. Environmental Dependencies

3.4.1. System Environment Configuration

To ensure the best development experience and compatibility, it is recommended to run self-built programs within the robot's own ROS 2 container. Development on Mac and Windows systems is currently not supported.

Walker S2 Framework Environment:

Platform	Address	System Version	ROS 2 Container
Walker S2 Orin	192.168.11.2	Ubuntu 20.04.6	humble
Walker S2 x86	192.168.11.3	Ubuntu 22.04.4	humble

4. Global Value Description

- Description: The naming ID definitions for the head, arm, waist, and leg motors of the full version Ubtech Walker S2 are as follows:

```
C++
{9001, "switch main", "Main Switch"},

{10000, "motion_imu", "Motion IMU Sensor"},
// Left Leg
{2001, "left_hip_roll_motor", "Left Hip Roll Motor"},
{2002, "left_hip_yaw_motor", "Left Hip Yaw Motor"},
{2003, "left_hip_pitch_motor", "Left Hip Pitch Motor"},
{2004, "left_knee_driver_motor", "Left Knee Drive Motor"},
{2006, "left_ankle_driver_inside_motor", "Left Ankle Drive Inside
Motor"},
{2005, "left_ankle_driver_outside_motor", "Left Ankle Drive
Outside Motor"},

// Waist
{11001, "waist_pitch_motor", "Waist Pitch Motor"},
{11002, "waist_yaw_motor", "Waist Yaw Motor"},
// Right Leg
{3001, "right_hip_roll_motor", "Right Hip Roll Motor"},
{3002, "right_hip_yaw_motor", "Right Hip Yaw Motor"},
{3003, "right_hip_pitch_motor", "Right Hip Pitch Motor"},
{3004, "right_knee_driver_motor", "Right Knee Drive Motor"},
{3006, "right_ankle_driver_inside_motor", "Right Ankle Drive
```

```
Inside Motor"},
{3005, "right_ankle_driver_outside_motor", "Right Ankle Drive
Outside Motor"},

{9002, "switch sub", "Sub Switch"},

// Head
{1001, "head_yaw_motor", "Head Yaw Motor"},
{1002, "head_pitch_motor", "Head Pitch Motor"},

// Left Arm
{4001, "left_shoulder_pitch_motor", "Left Shoulder Pitch Motor"},
{4002, "left_shoulder_roll_motor", "Left Shoulder Roll Motor"},
{4003, "left_shoulder_yaw_motor", "Left Shoulder Yaw Motor"},
{4004, "left_elbow_roll_motor", "Left Elbow Roll Motor"},
{4005, "left_elbow_yaw_motor", "Left Elbow Yaw Motor"},
{4006, "left_wrist_roll_motor", "Left Wrist Roll Motor"},
{4007, "left_wrist_pitch_motor", "Left Wrist Pitch Motor"},
{6001, "left_arm_ft", "Left Arm 6-Axis Force Sensor"},
{8001, "left_hand", "Left Hand (may not be connected to dexterous
hand, actually a gripper)"},

// Right Arm
{5001, "right_shoulder_pitch_motor", "Right Shoulder Pitch
Motor"},
{5002, "right_shoulder_roll_motor", "Right Shoulder Roll Motor"},
{5003, "right_shoulder_yaw_motor", "Right Shoulder Yaw Motor"},
{5004, "right_elbow_roll_motor", "Right Elbow Roll Motor"},
{5005, "right_elbow_yaw_motor", "Right Elbow Yaw Motor"},
{5006, "right_wrist_roll_motor", "Right Wrist Roll Motor"},
{5007, "right_wrist_pitch_motor", "Right Wrist Pitch Motor"},
{6002, "right_arm_ft", "Right Arm 6-Axis Force Sensor"},
{8002, "right_hand", "Right Hand (may not be connected to
dexterous hand, actually a gripper)"}
```

5. Interface Description - Low-Level Services

> During actual debugging, refer to the demo examples as the standard~

5.1. Motion Control

5.1.1. Motion Control Status Acquisition Interface

- Description: Acquires status information related to motion control, including the current position of motors, IMU, and 6-axis force sensor information.
- Control Method: Topic
- Topic Name: `/mc/sdk/robot_state`
- Data Definition Location: `mc_state_msgs::msg::robot_state`
- Data Format:

```
Plain Text
std_msgs/Header header
sensor_msgs/JointState joint_states
sensor_msgs/Imu[] imu_states
geometry_msgs/WrenchStamped[] ft_states
```

- Code Example:

```
C++
#include "rclcpp/rclcpp.hpp"
#include "mc_state_msgs/msg/robot_state.hpp"
#include "std_msgs/msg/string.hpp"
#include <iostream>
#include <memory>

class RobotStateSubscriber : public rclcpp::Node
{
public:
  RobotStateSubscriber() : Node("sub_robot_state")
  {
    // Create QoS configuration, using system default sensor
data QoS
    rclcpp::QoS qos_settings(10);

qos_settings.reliability(RMW_QOS_POLICY_RELIABILITY_BEST_EFFOR
T);

qos_settings.durability(RMW_QOS_POLICY_DURABILITY_VOLATILE);
    qos_settings.history(RMW_QOS_POLICY_HISTORY_KEEP_LAST);

    subscription_ = this-
>create_subscription<mc_state_msgs::msg::RobotState>(
      "/mc/sdk/robot_state",
      qos_settings,
      std::bind(&RobotStateSubscriber::topic_callback, this,
std::placeholders::_1));
```

```
  }

private:
  void topic_callback(const
mc_state_msgs::msg::RobotState::SharedPtr msg) const
  {
    RCLCPP_INFO(this->get_logger(),
"================================");

    for (size_t i = 0; i < msg->joint_states.name.size(); i++)
{
      std::cout << "Joint: " << msg->joint_states.name[i]
                << " Joint position: " << msg-
>joint_states.position[i] << std::endl;
    }

    for (const auto& item : msg->imu_states) {
      std::cout << "Imu: " << item.header.frame_id
                << ", acc:" << item.linear_acceleration.x << "
"
                << item.linear_acceleration.y << " " <<
item.linear_acceleration.z
                << ", gyro:" << item.angular_velocity.x << " "
                << item.angular_velocity.y << " "
                << item.angular_velocity.z << std::endl;
    }

    for (const auto& item : msg->ft_states) {
      std::cout << "Ft: " << item.header.frame_id
                << ", force: " << item.wrench.force.x << " "
                << item.wrench.force.y << " " <<
item.wrench.force.z
                << ", torque: " << item.wrench.torque.x << " "
                << item.wrench.torque.y << " " <<
item.wrench.torque.z << std::endl;
    }
  }


rclcpp::Subscription<mc_state_msgs::msg::RobotState>::SharedPt
r subscription_;
};

int main(int argc, char const* argv[])
```

```
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<RobotStateSubscriber>());
  rclcpp::shutdown();
  return EXIT_SUCCESS;
}
```

5.1.2. Motion Control Interface

- Description: Interface for position control of motors. Requires providing the motor name, control mode, and target position.
- Control Method: Topic
- Topic Name: `/mc/sdk/robot_command`
- Data Definition Location: `mc_task_msgs::msg::joint_cmd.hpp` and `mc_task_msgs::msg::robot_command.hpp`
- Data Format:

```
Plain Text
RobotCommand
std_msgs/Header header
JointCmd[] joint_cmd

JointCmd
int8 BEGIN=-1
int8 MODE_EFFORT=0
int8 MODE_VELOCITY=1
int8 MODE_POSITION=2
int8 CUSTOM_MODE_1 = 7
int8 CUSTOM_MODE_2 = 8
int8 CUSTOM_MODE_3 = 9

string name
int8 control_mode
float64 position
float64 velocity
float64 effort
float64 v1
float64 v2
float64 v3
```

- Code Example:

```
C++
```

```
#include <mc_task_msgs/msg/robot_command.hpp>
#include <mc_task_msgs/msg/joint_cmd.hpp>

#include <chrono>
#include <rclcpp/rclcpp.hpp>
#include <std_msgs/msg/header.hpp>
#include <cmath>

int main(int argc, char **argv) {
  rclcpp::init(argc, argv);
  auto node = rclcpp::Node::make_shared("pub_head_command");
  auto cmd_publisher_ = node-
>create_publisher<mc_task_msgs::msg::RobotCommand>("/mc/sdk/ro
bot_command", 10);
  rclcpp::Rate rate(500);
  double time_cnt = 0.0;

  while (rclcpp::ok()) {
    mc_task_msgs::msg::RobotCommand cmd;
    cmd.header.stamp = node->now();

    mc_task_msgs::msg::JointCmd head_yaw_cmd;
    head_yaw_cmd.name = "head_yaw_joint";
    head_yaw_cmd.control_mode =
mc_task_msgs::msg::JointCmd::MODE_POSITION;
    head_yaw_cmd.position = sin(time_cnt) * 0.5;
    cmd.joint_cmd.push_back(head_yaw_cmd);

    mc_task_msgs::msg::JointCmd head_pitch_cmd;
    head_pitch_cmd.name = "head_pitch_joint";
    head_pitch_cmd.control_mode =
mc_task_msgs::msg::JointCmd::MODE_POSITION;
    head_pitch_cmd.position = sin(time_cnt) * 0.5;
    cmd.joint_cmd.push_back(head_pitch_cmd);

    cmd_publisher_->publish(cmd);

    time_cnt += 0.002;
    rate.sleep();
  }

  rclcpp::shutdown();
  return EXIT_SUCCESS;
}
```

5.2. IMU Orin

5.2.1. Status Acquisition Interface

- **Description:** Acquires IMU sensor data from the Orin, including acceleration, angular velocity, and orientation quaternion.
- **Control Method:** Topic
- **Topic Name:** `/sensor/imu/orin`
- **Data Type:** `sensor_msgs::msg::Imu`
- **Data Format:**

```
Plain Text
std_msgs/Header header       # Message header

geometry_msgs/Quaternion orientation      # Orientation
quaternion
float64[9] orientation_covariance         # Orientation
covariance matrix

geometry_msgs/Vector3 angular_velocity    # Angular velocity
vector
float64[9] angular_velocity_covariance    # Angular velocity
covariance matrix

geometry_msgs/Vector3 linear_acceleration    # Linear
acceleration vector
float64[9] linear_acceleration_covariance    # Linear
acceleration covariance matrix
```

- **Subscribe to Topic:**

```
Bash
ros2 run ubt_ros2_example imu_subscriber
```

- **Example Code:**

```
C++
#include "rclcpp/rclcpp.hpp"
#include "sensor_msgs/msg/imu.hpp"

class ImuSubscriber : public rclcpp::Node
{
 public:
```

```
  ImuSubscriber() : Node("imu_subscriber")
  {
    subscription_ = this-
>create_subscription<sensor_msgs::msg::Imu>(
        "/sensor/imu/orin",  // IMU topic name
        rclcpp::QoS(rclcpp::KeepLast(10)).best_effort(),
        std::bind(&ImuSubscriber::topic_callback, this,
std::placeholders::_1));
  }

 private:
  void topic_callback(const sensor_msgs::msg::Imu::SharedPtr
msg)
  {
    RCLCPP_INFO(this->get_logger(),
                "Orientation: [w=%.3f, x=%.3f, y=%.3f,
z=%.3f]",
                msg->orientation.w, msg->orientation.x, msg-
>orientation.y,
                msg->orientation.z);

    RCLCPP_INFO(this->get_logger(),
                "Angular velocity: [x=%.3f, y=%.3f, z=%.3f]
rad/s",
                msg->angular_velocity.x, msg-
>angular_velocity.y,
                msg->angular_velocity.z);

    RCLCPP_INFO(this->get_logger(),
                "Linear acceleration: [x=%.3f, y=%.3f, z=%.3f]
m/s^2",
                msg->linear_acceleration.x, msg-
>linear_acceleration.y,
                msg->linear_acceleration.z);
  }

  rclcpp::Subscription<sensor_msgs::msg::Imu>::SharedPtr
subscription_;
};

int main(int argc, char *argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<ImuSubscriber>());
```

```
  rclcpp::shutdown();
  return 0;
}
```

5.3. RGBD Camera

5.3.1. Body Back Orbbec Gemini 2L Camera

Walker-S2 is equipped with Gemini 2L cameras on the back of the body and the front of the waist, providing the following topics and services:

- **Default Provided Topics:**
  - `/sensor/camera/body_back_rgbd/color/info`: `sensor_msgs/msg/CameraInfo`
  - `/sensor/camera/body_back_rgbd/color/raw`: `shm_msgs/msg/Image1m`
  - `/sensor/camera/body_back_rgbd/depth/info`: `sensor_msgs/msg/CameraInfo`
  - `/sensor/camera/body_back_rgbd/depth/raw`: `shm_msgs/msg/Image1m`
  - `/sensor/camera/body_back_rgbd/mix/raw`: `shm_msgs/msg/Vec2Image1m`
- **Example:**
  - **Description:** Acquires data from the body back Gemini 2L.
  - **Topic Name:** `/sensor/camera/body_back_rgbd/color/raw`
  - **Type:** `shm_msgs::msg::Image1m`
  - **Data Format:**

```
Plain Text
shm_msgs/Header header # Message header
uint32 height                # Image height
uint32 width                 # Image width
shm_msgs/String encoding       # Pixel encoding
uint8 is_bigendian    # Is big-endian
uint32 step           # Total byte length per row
uint8[1048576] data          # Actual matrix data
uint32 DATA_MAX_SIZE=1048576

shm_msgs/Header
    builtin_interfaces/Time stamp
    shm_msgs/String frame_id
```

```
shm_msgs/String
    char[256] data
    uint8 size 0
    uint8 MAX_SIZE=255
```

- **Subscribe to Topic:**

```
Bash
ros2 run ubt_ros2_example headfrontcolor_subscriber # Subscribe to color topic
```

- **Example Code:**

```
C++
#include "rclcpp/rclcpp.hpp"
#include "shm_msgs/msg/image1m.hpp"

class BodyBackRgbdColorSubscriber : public rclcpp::Node
{
 public:
  BodyBackRgbdColorSubscriber() : Node("body_back_color_subscriber")
  {
    subscription_ = this->create_subscription<shm_msgs::msg::Image1m>(
        "/sensor/camera/body_back_rgbd/color/raw",  // Topic name
        rclcpp::QoS(rclcpp::KeepLast(10)).best_effort(),

std::bind(&BodyBackRgbdColorSubscriber::topic_callback, this,
                  std::placeholders::_1));
  }

 private:
  void topic_callback(const shm_msgs::msg::Image1m::SharedPtr msg)
  {
    RCLCPP_INFO(this->get_logger(), "Current Time: %d sec %u nanosec",
                msg->header.stamp.sec, msg->header.stamp.nanosec);
    RCLCPP_INFO(this->get_logger(), "Frame id: %s",
                reinterpret_cast<char *>(msg->header.frame_id.data.data()));
```

```
    RCLCPP_INFO(this->get_logger(), "Height * Width:  %u
* %u", msg->height,
                msg->width);
    RCLCPP_INFO(this->get_logger(), "Encoding: %s",
                reinterpret_cast<char *>(msg-
>encoding.data.data()));
    RCLCPP_INFO(this->get_logger(), "Bigendian: %u", msg-
>is_bigendian);
    RCLCPP_INFO(this->get_logger(), "Step: %u", msg-
>step);
    RCLCPP_INFO(this->get_logger(), "Matrix data
length: %zu",
                sizeof(msg->data) / sizeof(msg->data[0]));
  }

  rclcpp::Subscription<shm_msgs::msg::Image1m>::SharedPtr
subscription_;
};

int main(int argc, char *argv[])
{
  rclcpp::init(argc, argv);

rclcpp::spin(std::make_shared<BodyBackRgbdColorSubscriber>
());
  rclcpp::shutdown();
  return 0;
}
```

- **Default Provided Services:**
  - `/body_back_rgbd/get_color_auto_white_balance`: `orbbec_camera_msgs/srv/GetInt32`
  - `/body_back_rgbd/get_color_device_info`: `orbbec_camera_msgs/srv/GetDeviceInfo`
  - `/body_back_rgbd/get_color_exposure`: `orbbec_camera_msgs/srv/GetInt32`
  - `/body_back_rgbd/get_color_gain`: `orbbec_camera_msgs/srv/GetInt32`
  - `/body_back_rgbd/get_color_sharpness`: `orbbec_camera_msgs/srv/GetInt32`
  - `/body_back_rgbd/get_color_white_balance`: `orbbec_camera_msgs/srv/GetInt32`

- `/body_back_rgbd/get_depth_exposure`: `orbbec_camera_msgs/srv/GetInt32`
- `/body_back_rgbd/get_depth_gain`: `orbbec_camera_msgs/srv/GetInt32`
- `/body_back_rgbd/set_color_auto_exposure`: `std_srvs/srv/SetBool`
- `/body_back_rgbd/set_color_auto_white_balance`: `std_srvs/srv/SetBool`
- `/body_back_rgbd/set_color_exposure`: `orbbec_camera_msgs/srv/SetInt32`
- `/body_back_rgbd/set_color_gain`: `orbbec_camera_msgs/srv/SetInt32`
- `/body_back_rgbd/set_color_mirror`: `std_srvs/srv/SetBool`
- `/body_back_rgbd/set_color_sharpness`: `orbbec_camera_msgs/srv/SetInt32`
- `/body_back_rgbd/set_color_white_balance`: `orbbec_camera_msgs/srv/SetInt32`
- `/body_back_rgbd/set_depth_auto_exposure`: `std_srvs/srv/SetBool`
- `/body_back_rgbd/set_depth_exposure`: `orbbec_camera_msgs/srv/SetInt32`
- `/body_back_rgbd/set_depth_gain`: `orbbec_camera_msgs/srv/SetInt32`
- `/body_back_rgbd/set_depth_mirror`: `std_srvs/srv/SetBool`
- `/body_back_rgbd/toggle_color`: `std_srvs/srv/SetBool`
- `/body_back_rgbd/toggle_depth`: `std_srvs/srv/SetBool`
- The data format for the Orbbec camera service interfaces is as follows. For details, please refer to the official Orbbec SDK: https://github.com/orbbec/OrbbecSDK_ROS2/tree/v2-main

5.3.2. Waist Front Orbbec Gemini 2L Camera

- **Description:** Acquires data from the waist front Gemini 2L.
- **Default Provided Topics:**
  - `/sensor/camera/waist_front_rgbd/color/info`: `sensor_msgs/msg/CameraInfo`
  - `/sensor/camera/waist_front_rgbd/color/raw`: `shm_msgs/msg/Image1m`
  - `/sensor/camera/waist_front_rgbd/depth/info`:

`sensor_msgs/msg/CameraInfo`

- `/sensor/camera/waist_front_rgbd/depth/raw`: `shm_msgs/msg/Image1m`
- `/sensor/camera/waist_front_rgbd/mix/raw`: `shm_msgs/msg/Vec2Image1m`

- **Example:**
  - **Topic Name:** `/sensor/camera/waist_front_rgbd/color/raw`
  - **Type:** `shm_msgs::msg::Image1m`
  - **Data Format:**

```
Plain Text
shm_msgs/Header header # Message header
uint32 height                # Image height
uint32 width                 # Image width
shm_msgs/String encoding       # Pixel encoding
uint8 is_bigendian    # Is big-endian
uint32 step           # Total byte length per row
uint8[1048576] data          # Actual matrix data
uint32 DATA_MAX_SIZE=1048576

shm_msgs/Header
    builtin_interfaces/Time stamp
    shm_msgs/String frame_id

shm_msgs/String
    char[256] data
    uint8 size 0
    uint8 MAX_SIZE=255
```

  - **Subscribe to Topic:**

```
Bash
ros2 run ubt_ros2_example waistfrontcolor_subscriber
```

- **Default Provided Services:**
  - `/waist_front_rgbd/get_color_auto_white_balance`: `orbbec_camera_msgs/srv/GetInt32`
  - `/waist_front_rgbd/get_color_device_info`: `orbbec_camera_msgs/srv/GetDeviceInfo`
  - `/waist_front_rgbd/get_color_exposure`: `orbbec_camera_msgs/srv/GetInt32`

- `/waist_front_rgbd/get_color_gain`: `orbbec_camera_msgs/srv/GetInt32`
- `/waist_front_rgbd/get_color_sharpness`: `orbbec_camera_msgs/srv/GetInt32`
- `/waist_front_rgbd/get_color_white_balance`: `orbbec_camera_msgs/srv/GetInt32`
- `/waist_front_rgbd/get_depth_exposure`: `orbbec_camera_msgs/srv/GetInt32`
- `/waist_front_rgbd/get_depth_gain`: `orbbec_camera_msgs/srv/GetInt32`
- `/waist_front_rgbd/set_color_auto_exposure`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/set_color_auto_white_balance`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/set_color_exposure`: `orbbec_camera_msgs/srv/SetInt32`
- `/waist_front_rgbd/set_color_gain`: `orbbec_camera_msgs/srv/SetInt32`
- `/waist_front_rgbd/set_color_mirror`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/set_color_sharpness`: `orbbec_camera_msgs/srv/SetInt32`
- `/waist_front_rgbd/set_color_white_balance`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/set_depth_auto_exposure`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/set_depth_exposure`: `orbbec_camera_msgs/srv/SetInt32`
- `/waist_front_rgbd/set_depth_gain`: `orbbec_camera_msgs/srv/SetInt32`
- `/waist_front_rgbd/set_depth_mirror`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/toggle_color`: `std_srvs/srv/SetBool`
- `/waist_front_rgbd/toggle_depth`: `std_srvs/srv/SetBool`

5.4. Stereo Camera

5.4.1. Stereo Camera

1. **Fisheye Camera Interface**

- **Default Provided Topics:**
  - `/sensor/camera/fisheye_left/image/info`: `sensor_msgs/msg/CameraInfo`
  - `/sensor/camera/fisheye_left/image/raw`: `shm_msgs/msg/Image2m`
  - `/sensor/camera/fisheye_left/sn`: `sensor_task_msgs/msg/SensorSN`
  - `/sensor/camera/fisheye_right/image/info`: `sensor_msgs/msg/CameraInfo`
  - `/sensor/camera/fisheye_right/image/raw`: `shm_msgs/msg/Image2m`
  - `/sensor/camera/fisheye_right/sn`: `sensor_task_msgs/msg/SensorSN`
- **Example:**
  - **Topic Name:** `/sensor/camera/fisheye_left/image/raw`
  - **Topic Type:** `shm_msgs::msg::Image2m`
  - **Data Format:**

```
Plain Text
shm_msgs/Header header # Message header
uint32 height                # Image height
uint32 width                 # Image width
shm_msgs/String encoding       # Pixel encoding
uint8 is_bigendian    # Is big-endian
uint32 step           # Total byte length per row
uint8[2097152] data          # Actual matrix data

uint32 DATA_MAX_SIZE=2097152
```

  - **Example Code:**

```
C++
#include "rclcpp/rclcpp.hpp"
#include "shm_msgs/msg/image2m.hpp"

class FishEyeLeftSubscriber : public rclcpp::Node
{
 public:
  FishEyeLeftSubscriber() :
Node("fish_eye_left_subscriber")
  {
    subscription_ = this-
>create_subscription<shm_msgs::msg::Image2m>(
        "/sensor/camera/fisheye_left/image/raw",  //
```

```
Topic name

rclcpp::QoS(rclcpp::KeepLast(10)).best_effort(),

std::bind(&FishEyeLeftSubscriber::topic_callback, this,
                  std::placeholders::_1));
  }

 private:
  void topic_callback(const
shm_msgs::msg::Image2m::SharedPtr msg)
  {
    RCLCPP_INFO(this->get_logger(), "Current Time: %d
sec %u nanosec",
                msg->header.stamp.sec, msg-
>header.stamp.nanosec);
    RCLCPP_INFO(this->get_logger(), "Frame id: %s",
                reinterpret_cast<char *>(msg-
>header.frame_id.data.data()));
    RCLCPP_INFO(this->get_logger(), "Height *
Width:  %u * %u", msg->height,
                msg->width);
    RCLCPP_INFO(this->get_logger(), "Encoding: %s",
                reinterpret_cast<char *>(msg-
>encoding.data.data()));
    RCLCPP_INFO(this->get_logger(), "Bigendian: %u",
msg->is_bigendian);
    RCLCPP_INFO(this->get_logger(), "Step: %u", msg-
>step);
    RCLCPP_INFO(this->get_logger(), "Matrix data
length: %zu",
                sizeof(msg->data) / sizeof(msg-
>data[0]));
  }


rclcpp::Subscription<shm_msgs::msg::Image2m>::SharedPtr
subscription_;
};

int main(int argc, char *argv[])
{
  rclcpp::init(argc, argv);
```

```
rclcpp::spin(std::make_shared<FishEyeLeftSubscriber>())
;
  rclcpp::shutdown();
  return 0;
}
```

2. **Stereo Camera Interface**
   - **Default Provided Topics:**
     - `/sensor/camera/stereo/depth/info`: `sensor_msgs/msg/CameraInfo`
     - `/sensor/camera/stereo_left/image/info`: `sensor_msgs/msg/CameraInfo`
     - `/sensor/camera/stereo_left/image/raw`: `shm_msgs/msg/Image2m`
     - `/sensor/camera/stereo_left/sn`: `sensor_task_msgs/msg/SensorSN`
     - `/sensor/camera/stereo_right/image/info`: `sensor_msgs/msg/CameraInfo`
     - `/sensor/camera/stereo_right/image/raw`: `shm_msgs/msg/Image2m`
     - `/sensor/camera/stereo_right/sn`: `sensor_task_msgs/msg/SensorSN`
   - **Example:**
     - **Topic Name:** `/sensor/camera/stereo_left/image/raw`
     - **Topic Type:** `shm_msgs::msg::Image2m`
     - **Data Format:**

```
Plain Text
shm_msgs/Header header # Message header
uint32 height                # Image height
uint32 width                 # Image width
shm_msgs/String encoding       # Pixel encoding
uint8 is_bigendian    # Is big-endian
uint32 step           # Total byte length per row
uint8[2097152] data          # Actual matrix data

uint32 DATA_MAX_SIZE=2097152
```

     - **Example Code:**

```
C++
#include "rclcpp/rclcpp.hpp"
#include "shm_msgs/msg/image2m.hpp"

class StereoLeftSubscriber : public rclcpp::Node
```

```
{
 public:
  StereoLeftSubscriber() :
Node("stereo_left_subscriber")
  {
    subscription_ = this-
>create_subscription<shm_msgs::msg::Image2m>(
        "/sensor/camera/stereo_left/image/raw",  //
Topic name

rclcpp::QoS(rclcpp::KeepLast(10)).best_effort(),

std::bind(&StereoLeftSubscriber::topic_callback, this,
                  std::placeholders::_1));
  }

 private:
  void topic_callback(const
shm_msgs::msg::Image2m::SharedPtr msg)
  {
    RCLCPP_INFO(this->get_logger(), "Current Time: %d
sec %u nanosec",
                msg->header.stamp.sec, msg-
>header.stamp.nanosec);
    RCLCPP_INFO(this->get_logger(), "Frame id: %s",
                reinterpret_cast<char *>(msg-
>header.frame_id.data.data()));
    RCLCPP_INFO(this->get_logger(), "Height *
Width:  %u * %u", msg->height,
                msg->width);
    RCLCPP_INFO(this->get_logger(), "Encoding: %s",
                reinterpret_cast<char *>(msg-
>encoding.data.data()));
    RCLCPP_INFO(this->get_logger(), "Bigendian: %u",
msg->is_bigendian);
    RCLCPP_INFO(this->get_logger(), "Step: %u", msg-
>step);
    RCLCPP_INFO(this->get_logger(), "Matrix data
length: %zu",
                sizeof(msg->data) / sizeof(msg-
>data[0]));
  }
```

```
rclcpp::Subscription<shm_msgs::msg::Image2m>::SharedPtr
subscription_;
};

int main(int argc, char *argv[])
{
  rclcpp::init(argc, argv);

rclcpp::spin(std::make_shared<StereoLeftSubscriber>());
  rclcpp::shutdown();
  return 0;
}
```

5.5. Battery

5.5.1. Battery Status

- **Description:** Acquires battery status information. The reporting frequency is 1Hz.
- **Charge Status:** `charge_status` `idle` idle status, `charging` charging (currently only these two states).
- **Control Method:** Topic
- **Topic Name:** `/emb/battery_state`
- **Type:** `emb_task_msgs/msg/BatteryState`
- **Data Format:**

```
Plain Text
BatteryInfo[] batteries_states
string IDLE=idle
string CHARGING=charging
string DISCHARGING=discharging
string FULL=full
string charge_status                    # Charge/discharge
status

float32   voltage                       # Total battery voltage
float32   current                       # Total battery current
float32   temperature                   # Battery temperature
float32   maxdifvol                     # Maximum voltage
difference
float32   batsoc                        # Total battery
capacity (State of Charge)
```

```
float32   remainchargetime              # Remaining charging
time
uint16    healthstatus                  # Health status
float32   remainuselife                 # Battery remaining
life (cycle count)
```

- healthstatus: The specific health states are as listed below.

Serial Number	Failure Level	fault code	Failure Phenomenon
1	Level 1 fault of the head plate	0x0000	No fault code
2		0x0001	Light sensor malfunction
3		0x0002	Parameter setting error
4		0x0003	Communication instruction error
5		0x0004	Light board malfunction
6		0x0005	Receiving failed
7		0x0006	Failed to send
8		0x0007	CAN communication timeout
9		0x0008	CAN physical bus error
10		0x0009	CAN peripheral initialization error
11	Level 1 Fault	0x1000u	No fault
12		0x1001u	Battery communication failure
13		0x1002u	Ethernet communication failure
14		0x1003u	Ultrasonic proximity warning fault code

15		0x1004u	Remote control communication interrupted
16		0x1005u	Wireless charging connection error
17		0x1006u	Left leg level 1 overcurrent fault code
18		0x1007u	Right leg level 1 overcurrent fault code
19		0x1008u	Hardware Level 1 Overcurrent Fault Code
20		0x1009u	System-wide level 1 overcurrent fault code
21		0x100Au	Flash failure
22		0x100Bu	Light sensor not connected fault
23		0x100Cu	Lamp panel display malfunction
24		0x100Du	RTC clock information fault
25		0x100Eu	UDP communication command error
26		0x100Fu	Communication command error
27	Secondary Fault	0x2001u	Left leg secondary overcurrent fault code
28		0x2002u	Right leg secondary overcurrent fault code
29		0x2003u	Hardware Secondary

			Overcurrent Fault Code
30		0x2004u	Whole body secondary overcurrent fault code
31		0x2005u	UDP reception failed
32		0x2006u	Battery CAN Bus Error
33		0x2007u	Head plate CAN bus error
34		0x2008u	Battery overcharge fault
35		0x2009u	Battery over-temperature fault
36		0x200Au	Battery overcurrent fault
37		0x200Bu	Battery short circuit fault
38	Level 3 Fault	0x3001u	Battery power overvoltage fault
39		0x3002u	Battery power undervoltage fault
40		0x3003u	ORIN power port voltage abnormal fault
41		0x3004u	Startup failed
42		0x3005u	Left leg level 3 overcurrent fault code
43		0x3006u	Right leg level 3 overcurrent fault code
44		0x3007u	Hardware Level 3 Overcurrent Fault Code
45		0x3008u	Whole body level 3 overcurrent fault code

46		0x3009u	Hardware overcurrent fault

- **Code Example:**

```
C++
#include "emb_task_msgs/msg/battery_state.hpp"
#include "rclcpp/rclcpp.hpp"

class BatterySubscriber : public rclcpp::Node
{
 public:
  BatterySubscriber() : Node("battery_subscriber")
  {
    subscription_ = this-
>create_subscription<emb_task_msgs::msg::BatteryState>(
        "/emb/battery_state",
rclcpp::QoS(rclcpp::KeepLast(10)).best_effort(),
        std::bind(&BatterySubscriber::topic_callback, this,
                  std::placeholders::_1));
  }

 private:
  void topic_callback(const
emb_task_msgs::msg::BatteryState::SharedPtr msg)
  {
    int size = msg->batteries_states.size();
    RCLCPP_INFO(this->get_logger(), "-------------------------
----");
    for (int i = 0; i < size; i++)
    {
      RCLCPP_INFO(this->get_logger(), "------Battery %d
state:-----", i + 1);
      RCLCPP_INFO(this->get_logger(), "  Charge status: %s",
                  msg-
>batteries_states[i].charge_status.c_str());
      RCLCPP_INFO(this->get_logger(), "  Voltage: %.3f V",
                  msg->batteries_states[i].voltage);
      RCLCPP_INFO(this->get_logger(), "  Current: %.3f A",
                  msg->batteries_states[i].current);
      RCLCPP_INFO(this->get_logger(), "
Temperature: %.3f °C",
                  msg->batteries_states[i].temperature);
      RCLCPP_INFO(this->get_logger(), "  Max voltage
```

```
diff: %.3f V",
                  msg->batteries_states[i].maxdifvol);
      RCLCPP_INFO(this->get_logger(), "  Battery
SOC: %.3f %%",
                  msg->batteries_states[i].batsoc);
      RCLCPP_INFO(this->get_logger(), "  Remaining charge
time: %.3f s",
                  msg->batteries_states[i].remainchargetime);
      RCLCPP_INFO(this->get_logger(), "  Health status: %u",
                  msg->batteries_states[i].healthstatus);
      RCLCPP_INFO(this->get_logger(), "  Remaining use
life: %.3f times",
                  msg->batteries_states[i].remainuselife);
    }
  }


rclcpp::Subscription<emb_task_msgs::msg::BatteryState>::Shared
Ptr
      subscription_;
};

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

5.5.2. Power Board Status

- **Description:** Acquires power board status information. The reporting frequency is 4.8Hz.
    - Time is BCD encoded.
- **Control Method:** Topic
- **Topic Name:** `/emb/powerboard_innerdata`
- **Type:** `emb_task_msgs::msg::InnerData`
- **Data Format:**

```
Plain Text
float32    adc_orin_value
```

```
float32    adc_orin_ibus_value       # Orin current A
float32    adc_arm_ibus_value
float32    adc_ibus_value
float32    adc_leftleg_ibus_value
float32    adc_rightleg_ibus_value
float32    adc_waist_ibus_value      # Waist current A
float32    adc_charge_det_value
float32    adc_vdc1_value
float32    adc_mos_temp
float32    adc_1v5
float32    temptature
float32    vrefint
float32    adc_x86_ibus_value        # x86 current A
float32    adc_rk_ibus_value     # rk current A
float32    adc_3vref_value           # 3v reference voltage V
uint16     err_code

uint8 second         # Second
uint8 minute         # Minute
uint8 hour           # Hour
uint8 day            # Day
uint8 month          # Month
uint16 year          # Year
```

- **Code Example:**

```
C++
#include "emb_task_msgs/msg/inner_data.hpp"
#include "rclcpp/rclcpp.hpp"

class InnerDataSubscriber : public rclcpp::Node
{
 public:
  InnerDataSubscriber() : Node("inner_data_subscriber")
  {
    subscription_ = this-
>create_subscription<emb_task_msgs::msg::InnerData>(
        "/emb/powerboard_innerdata",  // Topic name
        rclcpp::QoS(rclcpp::KeepLast(10)).best_effort(),
        std::bind(&InnerDataSubscriber::topic_callback, this,
                  std::placeholders::_1));
  }

 private:
```

```
  void topic_callback(const
emb_task_msgs::msg::InnerData::SharedPtr msg)
  {
    RCLCPP_INFO(this->get_logger(), "------ Inner Data ------
");
    RCLCPP_INFO(this->get_logger(), "Orin Voltage: %.3f V",
                msg->adc_orin_value);
    RCLCPP_INFO(this->get_logger(), "Orin Current: %.3f A",
                msg->adc_orin_ibus_value);
    RCLCPP_INFO(this->get_logger(), "Arm Current: %.3f A",
                msg->adc_arm_ibus_value);
    RCLCPP_INFO(this->get_logger(), "Total Current: %.3f A",
                msg->adc_ibus_value);
    RCLCPP_INFO(this->get_logger(), "Left Leg Current: %.3f
A",
                msg->adc_leftleg_ibus_value);
    RCLCPP_INFO(this->get_logger(), "Right Leg Current: %.3f
A",
                msg->adc_rightleg_ibus_value);
    RCLCPP_INFO(this->get_logger(), "Waist Current: %.3f A",
                msg->adc_waist_ibus_value);
    RCLCPP_INFO(this->get_logger(), "Charge Voltage: %.3f V",
                msg->adc_charge_det_value);
    RCLCPP_INFO(this->get_logger(), "Total Voltage: %.3f V",
                msg->adc_vdc1_value);
    RCLCPP_INFO(this->get_logger(), "MOSFET Temp: %.3f °C",
msg->adc_mos_temp);
    RCLCPP_INFO(this->get_logger(), "5V Output Voltage: %.3f
V", msg->adc_1v5);
    RCLCPP_INFO(this->get_logger(), "Chip Temp: %.3f °C", msg-
>temptature);
    RCLCPP_INFO(this->get_logger(), "Reference Voltage: %.3f
V", msg->vrefint);
    RCLCPP_INFO(this->get_logger(), "X86 Current: %.3f A",
                msg->adc_x86_ibus_value);
    RCLCPP_INFO(this->get_logger(), "RK Current: %.3f A",
                msg->adc_rk_ibus_value);
    RCLCPP_INFO(this->get_logger(), "3V Reference
Voltage: %.3f V",
                msg->adc_3vref_value);
    RCLCPP_INFO(this->get_logger(), "Error Code: %u", msg-
>err_code);
  }
```

```
rclcpp::Subscription<emb_task_msgs::msg::InnerData>::SharedPtr
subscription_;
};

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

5.6. Audio Interface

5.6.1. Audio Playback

- **Description:** Play existing audio files through the speaker or synthesize speech via TTS.
- **Control Method:** Action
- **Audio Playback:** Controlled via goal. When `type=1`, it's TTS speech synthesis playback; when `type=0`, it's specified file playback.
- **Action Name:** `/sys/speech/tts`
- **Data Format:** `sys_task_msgs/action/tts`

```
Plain Text
# Goal
uint8 FILE = 0
uint8 TTS = 1
uint8 type 1

# all valid
bool is_break true # Whether to interrupt

# Only file is valid
string file_path "" # File path

# Only tts is valid
string text "" # TTS synthesis text
string speaker "male_01" # Voice selection, default male_01
int32 speed 50 # TTS speech speed
int32 volume 100 # TTS speech volume
```

```
int32 pitch 50 # TTS speech pitch
string language "zh" # TTS language
string format "wav" # Audio file format supports .wav format
bool need_save true # Whether to cache

---
# Result
rosa_msgs/NodeState result
---
```

- **Example Demo Run:**

Transfer the audio file to the `/debug` directory.

```
Bash
ros2 run ubt_ros2_example audio_player --ros-args -p
file_path:=/path/to/your/file.wav
```

- **Example Code:**

```
C++
#include <rclcpp/rclcpp.hpp>
#include <rclcpp_action/rclcpp_action.hpp>
#include "sys_task_msgs/action/tts.hpp"
#include <future>  // for std::promise / std::future

using namespace std::chrono_literals;

class SpeechClient : public rclcpp::Node
{
 public:
  SpeechClient() : Node("simple_speech_client")
  {
    // Create Action client
    action_client_ =
rclcpp_action::create_client<sys_task_msgs::action::Tts>(
        this, "/sys/speech/tts");

    // Wait for Action Server to be available
    if (!action_client_->wait_for_action_server(20s))
    {
      RCLCPP_ERROR(this->get_logger(),
                   "Action server not available after
waiting");
      return;
```

```
    }

    // Declare and get parameter
    this->declare_parameter<std::string>("file_path", "");
    this->get_parameter("file_path", file_path_);

    if (file_path_.empty())
    {
      RCLCPP_ERROR(this->get_logger(), "File path parameter
not provided");
      return;
    }

    // Send goal request
    send_goal();
  }

  std::shared_future<void> get_result_future()
  {
    return result_promise_.get_future();
  }

 private:
  void send_goal()
  {
    sys_task_msgs::action::Tts::Goal goal_msg;
    goal_msg.type = 0;
    goal_msg.is_break = true;
    goal_msg.file_path = file_path_;

    RCLCPP_INFO(this->get_logger(), "Sending goal with file
path: %s",
                file_path_.c_str());


rclcpp_action::Client<sys_task_msgs::action::Tts>::SendGoalOpt
ions options;

    // Goal response callback
    options.goal_response_callback =

[this](std::shared_ptr<rclcpp_action::ClientGoalHandle<
                   sys_task_msgs::action::Tts>> goal_handle) {
          if (!goal_handle)
```

```
          {
            RCLCPP_ERROR(this->get_logger(), "Goal was
rejected by server");
            result_promise_.set_value();  // End early
          }
          else
          {
            RCLCPP_INFO(this->get_logger(), "Goal accepted by
server");
          }
        };

    // Feedback callback (can be ignored or printed here)
    options.feedback_callback =
        [this](rclcpp_action::ClientGoalHandle<
                    sys_task_msgs::action::Tts>::SharedPtr,
               const std::shared_ptr<const
sys_task_msgs::action::Tts::Feedback>
                    feedback) {
          RCLCPP_INFO(this->get_logger(), "Feedback
received...");
        };

    // Result callback
    options.result_callback =
        [this](const rclcpp_action::ClientGoalHandle<
              sys_task_msgs::action::Tts>::WrappedResult
&result) {
          switch (result.code)
          {
            case rclcpp_action::ResultCode::SUCCEEDED:
              RCLCPP_INFO(this->get_logger(), "Result received
successfully");
              break;
            case rclcpp_action::ResultCode::ABORTED:
              RCLCPP_ERROR(this->get_logger(), "Goal was
aborted");
              break;
            case rclcpp_action::ResultCode::CANCELED:
              RCLCPP_ERROR(this->get_logger(), "Goal was
canceled");
              break;
            default:
              RCLCPP_ERROR(this->get_logger(), "Unknown result
```

```
code");
                break;
            }
            result_promise_.set_value();  // Notify main it can
exit
        };

    action_client_->async_send_goal(goal_msg, options);
  }

  std::string file_path_;
  rclcpp_action::Client<sys_task_msgs::action::Tts>::SharedPtr
action_client_;

  std::promise<void> result_promise_;  // For exit control
};

int main(int argc, char **argv)
{
  rclcpp::init(argc, argv);
  auto node = std::make_shared<SpeechClient>();

  // Wait for result callback notification
  auto future = node->get_result_future();
  rclcpp::spin_until_future_complete(node, future);

  rclcpp::shutdown();
  return 0;
}
```

- **Error Codes:**

```
C++
// Success
int32 SUCCESS = 1001000
string SUCCESS_STR = "Success"

// Interrupted
int32 INTERRUPTED = 1001001
string INTERRUPTED_STR = "Interrupted"

// websocket invalid
int32 WEBSOCKET_INVALID = 6001001
string WEBSOCKET_INVALID_STR = "Websocket cannot connect"
```

```
// file invalid
int32 FILE_INVALID = 5001002
string FILE_INVALID_STR = "Invalid file passed in"

// Large model upload image failed
int32 LLM_UPLOAD_IMAGE_FAILED = 5002003
string LLM_UPLOAD_IMAGE_FAILED_STR = "LLM upload image failed"

// Large model CHAT timeout
int32 LLM_CHAT_TIMEOUT = 5002004
string LLM_CHAT_TIMEOUT_STR = "LLM chat timeout"

// Large model CHAT failed
int32 LLM_CHAT_FAILED = 5002005
string LLM_CHAT_FAILED_STR = "LLM chat failed"
```

5.7. Dexterous Hand - 3rd Generation

5.7.1. Status Acquisition Interface

- **Description:** Acquires dexterous hand joint status information.
- **Acquisition Method:** Topic
- **Topic Names:** `/mc/left_hand/joint_states` and `/mc/right_hand/joint_states`
- **Data Definition Location:** `sensor_msgs::msg::JointState`
- **Data Format:**

```
Plain Text
std_msgs/Header header
        builtin_interfaces/Time stamp
                int32 sec
                uint32 nanosec
        string frame_id

string[] name
float64[] position
float64[] velocity
float64[] effort
```

- **Code Example:**

```C++
#include "rclcpp/rclcpp.hpp"
#include "sensor_msgs/msg/joint_state.hpp"
#include "std_msgs/msg/string.hpp"
#include <iostream>
#include <memory>

class HandsStateSubscriber : public rclcpp::Node
{
public:
  HandsStateSubscriber() : Node("sub_hands_state")
  {
    // Create QoS configuration, using system default sensor
data QoS
    rclcpp::QoS qos_settings(10);

qos_settings.reliability(RMW_QOS_POLICY_RELIABILITY_BEST_EFFOR
T);

qos_settings.durability(RMW_QOS_POLICY_DURABILITY_VOLATILE);
    qos_settings.history(RMW_QOS_POLICY_HISTORY_KEEP_LAST);

    // Subscribe to left hand joint states
    left_hand_subscription_ = this-
>create_subscription<sensor_msgs::msg::JointState>(
      "/mc/left_hand/joint_states",
      qos_settings,
      [this](const sensor_msgs::msg::JointState::SharedPtr
msg) {
        this->left_hand_callback(msg);
      });

    // Subscribe to right hand joint states
    right_hand_subscription_ = this-
>create_subscription<sensor_msgs::msg::JointState>(
      "/mc/right_hand/joint_states",
      qos_settings,
      [this](const sensor_msgs::msg::JointState::SharedPtr
msg) {
        this->right_hand_callback(msg);
      });
  }

private:
```

```
  void left_hand_callback(const
sensor_msgs::msg::JointState::SharedPtr msg) const
  {
    RCLCPP_INFO(this->get_logger(), "======= Left Hand Joint
States =======");

    for (size_t i = 0; i < msg->name.size(); i++) {
      std::cout << "Joint: " << msg->name[i]
                << " Position: " << msg->position[i] <<
std::endl;
    }
  }

  void right_hand_callback(const
sensor_msgs::msg::JointState::SharedPtr msg) const
  {
    RCLCPP_INFO(this->get_logger(), "======= Right Hand Joint
States =======");

    for (size_t i = 0; i < msg->name.size(); i++) {
      std::cout << "Joint: " << msg->name[i]
                << " Position: " << msg->position[i] <<
std::endl;
    }
  }


rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr
left_hand_subscription_;

rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr
right_hand_subscription_;
};

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

5.7.2. Control Interface

- **Description:** Controls the 3rd generation dexterous hand.
- **Control Method:** Topic
- **Topic Names:** `/mc/left_hand/command` or `/mc/right_hand/command`
- **Type:** `mc_task_msgs::msg::JointCommand`
- **Data Format:**

```
Plain Text
std_msgs/Header header
        builtin_interfaces/Time stamp
                int32 sec
                uint32 nanosec
        string frame_id
int32[] mode
float64[] position
float64[] velocity
float64[] torque
float64[] acceleration
string[]  names
float64[] kp
float64[] kd
int32 POSITION_MODE=1
int32 VELOCITY_MODE=2
int32 TORQUE_MODE=3
int32 RAW_POSITION_MODE=4
```

- **Code Example:**

```
C++
#include <mc_task_msgs/msg/joint_command.hpp>

#include <chrono>
#include <rclcpp/rclcpp.hpp>
#include <std_msgs/msg/header.hpp>
#include <cmath>

int main(int argc, char **argv) {
  rclcpp::init(argc, argv);
  auto node = rclcpp::Node::make_shared("pub_hand_command");

  // Create publishers for left and right hand controllers
  auto left_hand_publisher = node-
>create_publisher<mc_task_msgs::msg::JointCommand>(
    "/mc/left_hand/command", 10);
```

```
  auto right_hand_publisher = node-
>create_publisher<mc_task_msgs::msg::JointCommand>(
    "/mc/right_hand/command", 10);

  // Define joint names for left hand
  std::vector<std::string> left_joint_names = {
    "left_thumb_swing",
    "left_thumb_mcp",
    "left_index_mcp",
    "left_middle_mcp",
    "left_ring_mcp",
    "left_little_mcp"
  };

  // Define joint names for right hand
  std::vector<std::string> right_joint_names = {
    "right_thumb_swing",
    "right_thumb_mcp",
    "right_index_mcp",
    "right_middle_mcp",
    "right_ring_mcp",
    "right_little_mcp"
  };

  rclcpp::Rate rate(500);
  double time_cnt = 0.0;

  while (rclcpp::ok()) {
    // Publish joint commands for left hand
    mc_task_msgs::msg::JointCommand left_cmd;
    left_cmd.header.stamp = node->now();
    left_cmd.names.resize(left_joint_names.size());
    left_cmd.position.resize(left_joint_names.size());
    left_cmd.mode.resize(left_joint_names.size());

    for (size_t i = 0; i < left_joint_names.size(); i++) {
      left_cmd.names[i] = left_joint_names[i];
      left_cmd.position[i] = sin(time_cnt + i * 0.2) * 0.3;
// Add phase difference for each joint
      left_cmd.mode[i] = 5;
    }

    // Publish joint commands for right hand
    mc_task_msgs::msg::JointCommand right_cmd;
```

```
    right_cmd.header.stamp = node->now();
    right_cmd.names.resize(right_joint_names.size());
    right_cmd.position.resize(right_joint_names.size());
    right_cmd.mode.resize(right_joint_names.size());

    for (size_t i = 0; i < right_joint_names.size(); i++) {
      right_cmd.names[i] = right_joint_names[i];
      right_cmd.position[i] = sin(time_cnt + i * 0.2) * 0.3;
// Add phase difference for each joint
      right_cmd.mode[i] = 5;
    }

    // Publish commands
    left_hand_publisher->publish(left_cmd);
    right_hand_publisher->publish(right_cmd);

    time_cnt += 0.002;
    rate.sleep();
  }

  rclcpp::shutdown();
  return EXIT_SUCCESS;
}
```

- **Note:**

a. The dexterous hand control only supports position control, and the positions in the msg must be assigned in the order given in the demo.

b. If only controlling some joints, set the control value for uncontrolled joints to 0 to place them at the zero position.

6. FAQ

Q1: Does it support starting up while lying down/sitting?

A1: No, the current version of Walker S2 only supports starting up while suspended. Before startup, pay attention to the positions of the upper limbs and ankles. Abnormal joint positions may cause the product to operate abnormally.

Q2: Does it support force control?

A2: The S2 model only supports position control, not force control.

Q3: For SDK secondary development, are there version requirements for the machine?

A3: It is recommended to perform secondary development on the main version 0.3.0 or above. Do not perform secondary development on lower versions or custom

versions.

If you have other questions, please provide feedback through the following channels:

- Customer Service Hotline: 400-6666-700 (*Only within China, Beijing Time 09:00 - 18:30*)
- WeChat Official Account: Ubtech Service