Walker Tienkung Humanoid Robot SDK Development Document

Preface

- Thank you for choosing the humanoid robot products (hereinafter referred to as "this product" or "the robot") of [UBTECH ROBOTICS CORP. LTD.] (hereinafter referred to as "The company").
- Before you officially use this product, please carefully read this manual and strictly follow the instructions in it. Using this product means that you have read and agreed to comply with all the terms and specifications in this manual and its appendices.
- This manual covers three models of the Walker Tienkung series robots (Walker Tienkung,Walker Tienkung - Voice & Vision ,Walker Tienkung -Embodied intelligence). When you understand and use the robot, you should refer to the actual device you received.
- Thank you for your understanding and support. We wish you a pleasant experience!

1. Legal Statement

- Users shall be responsible for their own usage behaviors and promise to use this product only for legal and proper purposes.
- When using this product, users shall comply with the laws and regulations of the location. It is strictly prohibited to use the product to harm or intimidate others or animals, or use it as a weapon and its supporting tools, or engage in any other activities that violate laws and regulations.
- The company shall not be liable for any property losses, personal injuries or safety hazards caused by violating the usage specifications in this manual.
- To the fullest extent permitted by law, The company makes no express or implied warranties for this product, including but not limited to the warranty of fitness for a particular purpose.
- To the fullest extent permitted by law, The company shall not be liable for any indirect, consequential, accidental, incidental, special or punitive damages, and the total liability shall not exceed the amount you paid for purchasing the product and the

amount paid to The company.

- The company has the final interpretation right of the above terms and shall comply with relevant laws and regulations. The company reserves the right to update and modify these terms without prior notice.

2. Notes

2.1 Usage Conditions

- This product is only for people aged 18 and above and shall be used in accordance with the provisions of this manual.
- This product is positioned as a humanoid robot for scientific research and teaching. Do not use it in scenarios other than those specified in this manual.
- Please note that users with limited sensory or cognitive abilities or lack of experience are prohibited from using this product independently without the assistance of a professional (a person who has participated in the delivery training organized by The company) and familiarity with this manual.
- If this product needs to be repaired, please contact the official after - sales service. It is prohibited for users or third - party institutions to disassemble or repair the product by themselves. If the after - sales repair of the product by The company is affected due to self - disassembly or repair by the user or a third - party institution, the user shall be responsible for it, and The company will no longer assume the corresponding after - sales responsibility.

2.2 Operational Safety

- Please use all parts and accessories of the robot correctly. Do not modify or disassemble them privately.
- Do not put your fingers, hair or clothes close to the joints, gaps and moving parts of the robot to prevent pinching or entanglement.
- Do not touch the surface of the motor during operation to prevent burns.
- Do not cover the sensor areas (such as the torso and head) to avoid affecting perception and control.
- Avoid using the robot in harsh environments such as humid, high - temperature or strong - magnetic environments. The normal operating temperature is 0 - 30 ℃, and the humidity is below 75% RH.
- It is prohibited to move or transport the robot when it is powered on or the battery is not in the sleep state.
- When the robot is performing actions, people must keep a distance and avoid entering the movement range of the robot's arms and legs to prevent collision injuries.
- When moving the robot, use a special shifting machine. It is strictly prohibited to

carry or drag the robot manually by one person.

- In case of abnormal situations such as the robot losing control of movement, immediately press the "C" key on the remote control to make all the joints of the robot stiffen and stop, and take protective measures as soon as possible.
- If operations such as stopping the robot by remote control cannot take effect in time, quickly press the emergency stop button on the back of the robot to cut off the power supply to ensure safety.

2.3 Special Operation Warnings

- Without the support of a protective bracket, be cautious when performing the following operations, otherwise the robot may fall and be damaged:
  - Long - press the "A" key in the running mode to make the robot stand.
  - Press the "D" key on the remote control to return to zero.
  - Move the "G" lever on the remote control to the right and press the "C" key to disable the joints.
  - Press the emergency stop button.

In addition:

- Before long - term standby, use a protective bracket to fix the robot to avoid falling due to automatic shutdown caused by low battery.
- When the battery is too low, stop and turn off the robot in time to prevent it from falling and being damaged.
- Do not push the robot from the side when it is standing, walking or running normally.

3. Product Overview

3.1 Product Introduction

- Walker Tienkung series robots are full - sized, fully electric - driven anthropomorphic running robots jointly developed by Shenzhen Ubtech Robotics Co., Ltd. and Beijing Humanoid Robot Innovation Center Co., Ltd. They have a highly bionic torso configuration and anthropomorphic motion control capabilities. They can have up to 42 degrees of freedom at most. The arms are designed with seven - degree - of - freedom robotic arms at most, and the wrists can be equipped with high - precision six - dimensional force sensors and dexterous five - finger hands.
- The Walker Tienkung series robots can stand, walk, run and perform multiple specific actions. They can stably adapt to complex terrains such as slopes and sand, and have excellent dynamic balance and anti - interference capabilities. Equipped with powerful joint drives, high - precision IMUs and a dual - battery quick - change

system, the whole - machine battery life exceeds 3.5 hours.

- In addition, Walker Tienkung - Voice & Vision and Walker Tienkung -Embodied intelligence can integrate a voice module and a large - scale voice interaction model, support natural language interaction, and the whole - machine computing power can reach up to 550 TOPS.
- As a general humanoid robot platform, the entire Walker Tienkung series opens up the interfaces of all joints and sensors, facilitating secondary development in the industry to achieve innovation in scenarios such as scientific research and education.

3.2 Functional Features

- **Adaptive to complex terrains**: It can walk smoothly on irregular terrains such as slopes and sand.
- **Anthropomorphic running**: It has the ability to run.
- **Dynamic balance control**: It can resist external interference in real - time and maintain a stable posture.
- **Motion control mode**: A reinforcement learning motion control mode has been deployed.
- **Voice interaction control**: It is equipped with a voice module and supports users to install a large - scale voice model by themselves to realize natural voice commands and interactions.
- **Open - interface design**: It provides interfaces for all joints and sensors, supporting remote operation, data collection and secondary development.

3.3 Application Scenarios

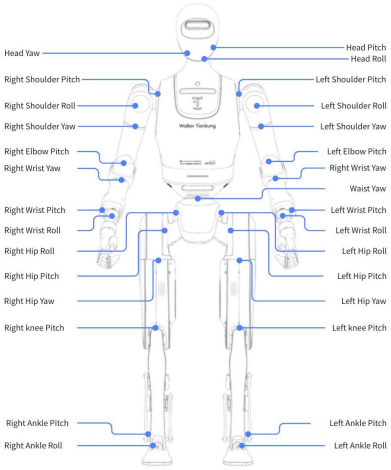
This product is designed for scientific research and teaching scenarios.

Notes:

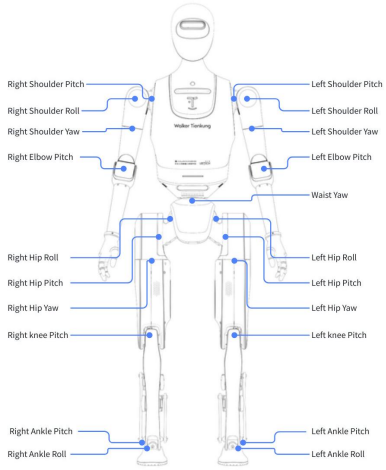
- Users can explore other application scenarios through secondary development, but they need to ensure that they comply with the laws and regulations, safety requirements and other necessary qualifications for that application scenario.
- For any use beyond the scope of scientific research and teaching, the relevant responsibilities shall be borne by the user.
- It is prohibited to directly use the original - configuration product in public places without safety protection or high - risk environments (such as rescue and medical care).

3.4 Structural introduction of Walker Tienkung

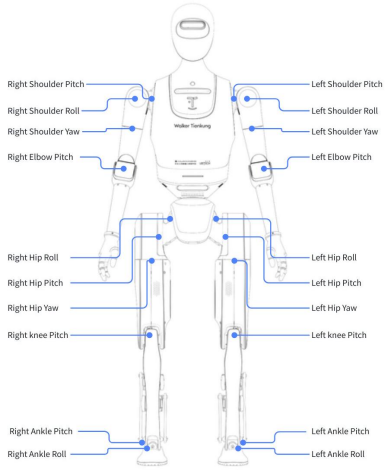
1. **Walker Tienkung -Embodied intelligence**



2. Walker Tienkung - Voice & Vision



3. Walker Tienkung



3.5 Whole-body joint parameters of Walker Tienkung

Affiliat	Nam	Range of	Torque Coeffici	Maximum	KP/KD	Rated Torqu	Peak Torqu	Joint	Applicable

ed Part	e	Motion (°)	ent (Nm/A)	Rotational Speed (rpm)	Range	e (Nm)	e (Nm)	Name	Models
Head	Head Roll	-26° ~ +26°	2.16	64	Kp: 0~2000 kd: 0~300	3	9	1	Walker Tienkung - Embodied intelligence
	Head Pitch	-25° ~ +25°	2.16	64	Kp: 0~2000 kd: 0~300	3	9	2	Walker Tienkung - Embodied intelligence
	Head Yaw	-90° ~ +90°	2.16	64	Kp: 0~2000 kd: 0~300	3	9	3	Walker Tienkung - Embodied intelligence
Waist	Waist Yaw	-170° ~ +170°	3.207	88	Kp: 0~2000 kd: 0~300	35	91	31	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision
Left Ar	Left Shoulde	-170° ~	3.207	88	Kp: 0~2	35	91	11	Walker Tienkung -

m	r Pitch	+170°			000 kd: 0~300				Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Left Shoulder Roll	-15° ~ +150°	3.37	120	Kp: 0~2000 kd: 0~300	20	60	12	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Left Shoulder Yaw	-170° ~ +170°	2.43	73	Kp: 0~2000 kd: 0~300	10	30	13	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung

	Left Elbow Pitch	-150° ~ +15°	2.43	73	Kp: 0~2000 kd: 0~300	10	30	14	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Left Wrist Yaw	-170° ~ +170°	1.71	146	Kp: 0~2000 kd: 0~300	5	15	15	Walker Tienkung - Embodied intelligence
	Left Wrist Pitch	-45° ~ +60°	2.16	72	Kp: 0~2000 kd: 0~300	3	9	16	Walker Tienkung - Embodied intelligence
	Left Wrist Roll	-95° ~ +75°	2.16	72	Kp: 0~2000 kd: 0~300	3	9	17	Walker Tienkung - Embodied intelligence
Right Arm	Right Shoulde	-170° ~ +170°	3.207	88	Kp: 0~2000 kd:	35	91	21	Walker Tienkung - Embodi

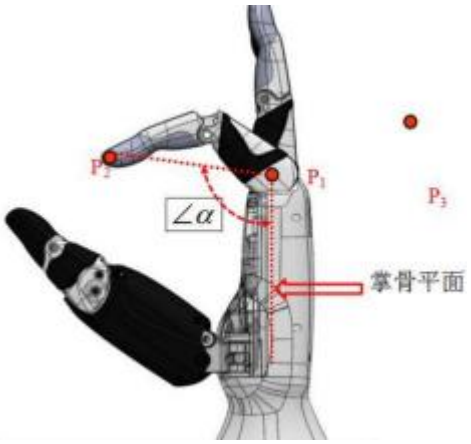
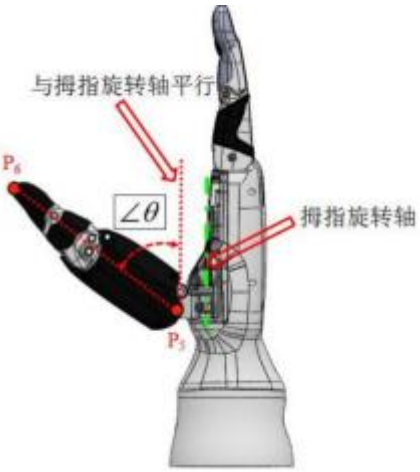
	r Pitch				0~300				ed intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Right Shoulder Roll	-150° ~ +15°	3.37	120	Kp: 0~2000 kd: 0~300	20	60	22	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Right Shoulder Yaw	-170° ~ +170°	2.43	73	Kp: 0~2000 kd: 0~300	10	30	23	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Righ	-150° ~	2.43	73	Kp:	10	30	24	Walker

	t Elbow Pitch	+15°			0~2000 kd: 0~300				Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Right Wrist Yaw	-170° ~ +170°	1.71	146	Kp: 0~2000 kd: 0~300	5	15	25	Walker Tienkung - Embodied intelligence
	Right Wrist Pitch	-45° ~ +60°	2.16	72	Kp: 0~2000 kd: 0~300	3	9	26	Walker Tienkung - Embodied intelligence
	Right Wrist Roll	-75° ~ +95°	2.16	72	Kp: 0~2000 kd: 0~300	3	9	27	Walker Tienkung - Embodied intelligence
Legs	Hip Roll	-45° ~ +45°	2.3	125	Kp: 0~2000 kd: 0~3	/	124	Left 51 Right	Walker Tienkung - Embodied intellige

					00			61	nce/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Hip Pitch	-160° ~120°	2	125	Kp: 0~2000 kd: 0~300	/	300	Left 52 Right 62	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Hip Yaw	-60° ~60°	2.3	125	Kp: 0~2000 kd: 0~300	/	124	Left 53 Right 63	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Knee Pitc	0° ~137°	2	125	Kp: 0~2000	/	300	Left 54	Walker Tienkung -

	h				kd: 0~300			Right 64	Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Ankle Pitch	-70° ~30°	2.6	128	Kp: 0~2000 kd: 0~300	/	36	Left 55 Right 65	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung
	Ankle Roll	-30° ~30°	2.6	128	Kp: 0~2000 kd: 0~300	/	36	Left 56 Right 66	Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision/ Walker Tienkung

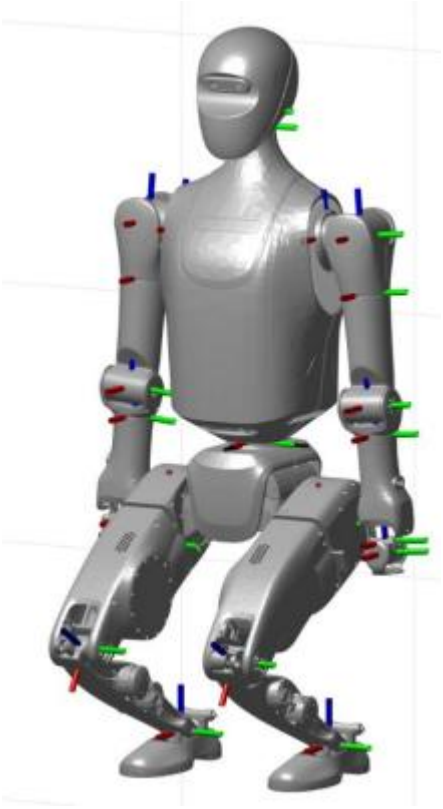
Parameter description of the dexterous hand. It is only applicable to the dexterous hand of Walker Tienkung -Embodied intelligence.

Angle	Range of motion (°)	Legend description
Pinky Ring finger, Middle finger Index finger	+19° ~ +176.7°	
Bending angle of the thumb	-13° ~ +53.6°	
Rotation angle of the thumb	+90° ~ +165°	

3.2 ## **Overall robot URDF, coordinate system, joint rotation axis, and joint zero point**

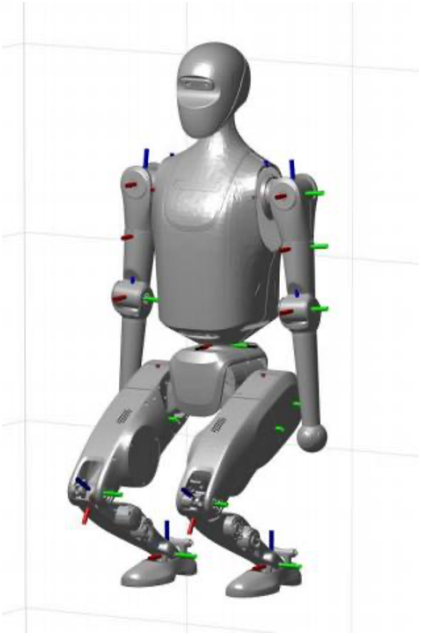
3.2.1 ### **Joint zero point**

1. Walker Tienkung -Embodied intelligence is shown in the following figure:



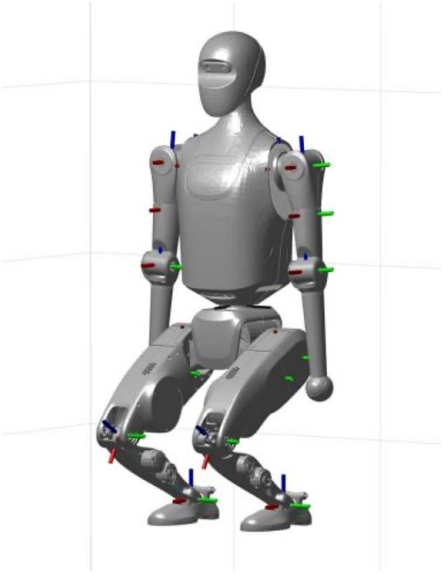
Walker Tienkung -Embodied intelligence

1. Walker Tienkung - Voice & Vision is shown in the following figure:



Walker Tienkung - Voice & Vision

2. Walker Tienkung is shown in the following figure:



Walker Tienkung

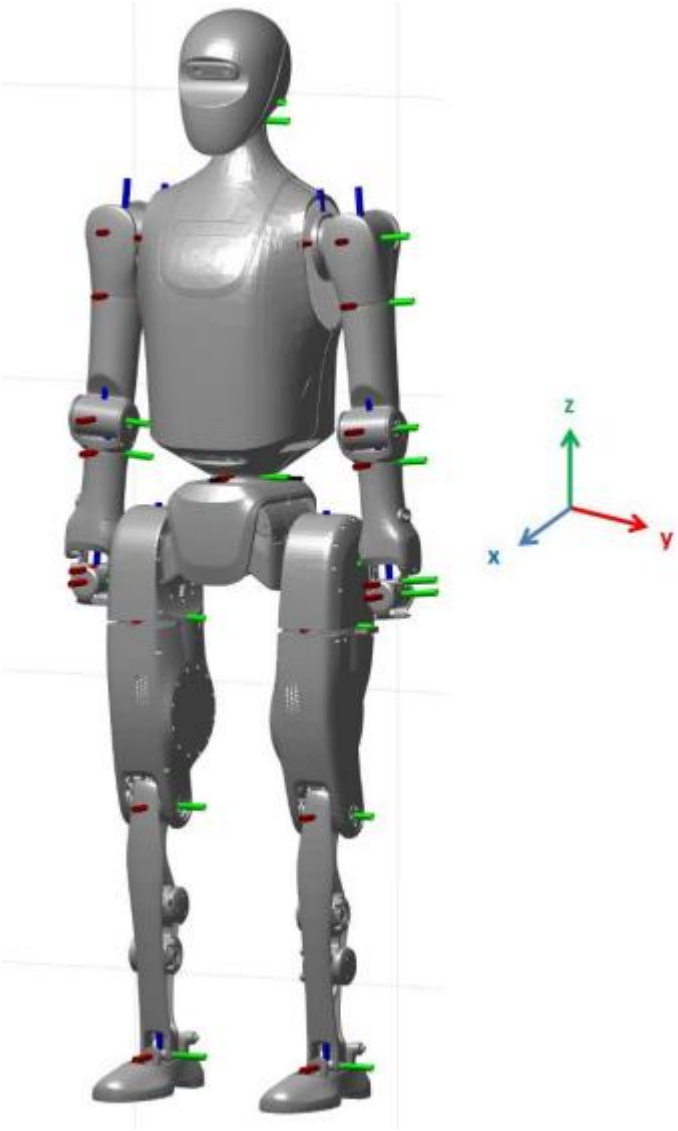
3.1.1 ### **Overall Machine Coordinate System**

1. The X-axis (Roll) of the overall machine coordinate system: The positive direction is towards the front of the robot.

2. The Y-axis (Pitch) of the overall machine coordinate system: The positive direction is towards the left of the robot.

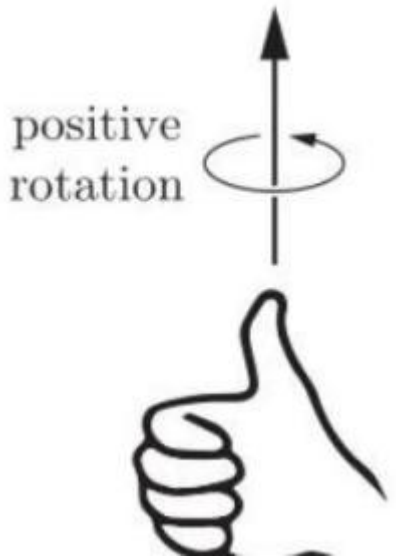
3. The Z-axis (Yaw) of the overall machine coordinate system: The positive direction is towards the top of the robot.

The directions of the overall machine coordinate systems of the three models are all the same.



3.5.1 3.1.1 ### Rotation Axes & Positive Directions

1. For the axis of each rotation axis of the robot, based on the state of each axis of the robot in the above figure, find the positive direction of the rotation axis corresponding to the positive direction of the overall coordinate system of the robot.
2. Examples:
    a. For example, the axis of the waist joint is vertical and close to the overall Z - axis (Yaw). So the positive direction of the waist joint is upward, and the positive direction of the joint rotation can be confirmed by the right - hand rule.
    b. For example, the axis of the hip joint Pitch of the leg is close to the overall Y - axis (Pitch). So the positive direction of the hip joint Pitch is to the left, and the positive direction of the joint rotation can be confirmed by the right - hand rule.



4. Quick Operation Guide

⚠️ Attention: Currently, there are two ways to start the robot: auto-start and manual program start.

If auto-start is enabled, you need to perform step 3.d **Press the A key for self-check** and do not need to perform step **5. Program Start**;

If auto-start is disabled and you start the program manually, you do not need to perform step 3.d. Press the A key for self-check and need to perform step **5. Program Start**.

1. **1. Confirm that the robot is suspended on the protective bracket;**
2. **2. Press the power button on the remote control briefly and then immediately press and hold it to turn on the remote control;**
3. **3. Power on the robot:**
    a. Press the main power switch;
    b. Press the on/off key briefly;
    c. Rotate the emergency stop button clockwise until it pops out;
    d. **(On the premise that auto-start is enabled)** After waiting for the robot to complete the boot process (i.e., the boot sound ends), press the A key on the remote control to start the self-check. After the self-check is successful (i.e., the overall status light turns blue-green and stays on constantly), enter the operable Ready state.
4. **Steps to configure Wi-Fi:**
    a. Connect the debugging Ethernet port on the back of the robot to the user's computer with an Ethernet cable;
    b. Configure the Ethernet port address of the user's computer to

192.168.41.xx/[255.255.255.0](255.255.255.0) (do not configure it to [192.168.41.1](192.168.41.1), [192.168.41.2](192.168.41.2), or [192.168.41.3](192.168.41.3));

c. Open the terminal and enter ssh ubuntu@192.168.41.x to log in to the main control board that needs to be configured (the motion control x86, Orin1, and Orin2 are [192.168.41.1](192.168.41.1), [192.168.41.2](192.168.41.2), and [192.168.41.3](192.168.41.3) respectively);

d. In the terminal from the previous step, continue to enter the following commands in sequence to configure the IP:

5. **(On the premise that auto-start is turned off)** Program startup

a. To start manually, please ensure that the auto-start function has been turned off first.

```
Bash
sudo systemctl stop proc_manager.service
```

b. First, open the first terminal and enter the following commands in sequence to start the **master control** node:

```
Bash
#Open the first window and start the main control node
sudo su
cd ros2ws
source install/setup.bash
ros2 launch body_control body.launch.py
```

c. If you need to start the **Motion Control - Reinforcement Learning** node, you should change it to enter the following commands in sequence:

```
Plain Text
sudo su
cd ros2ws
source install/setup.bash
ros2 run rl_control rl_control_node
```

You can start using the SDK for development and debugging at this time.

5. Shutdown:

1. Confirm that the robot has stopped and returned to the standing state.
2. Press the "C" key on the remote control to make the robot freeze.
3. Fix the robot on the bracket and lift it up.
4. Press the emergency stop button.

5. Long - press the on/off key for 6 seconds.

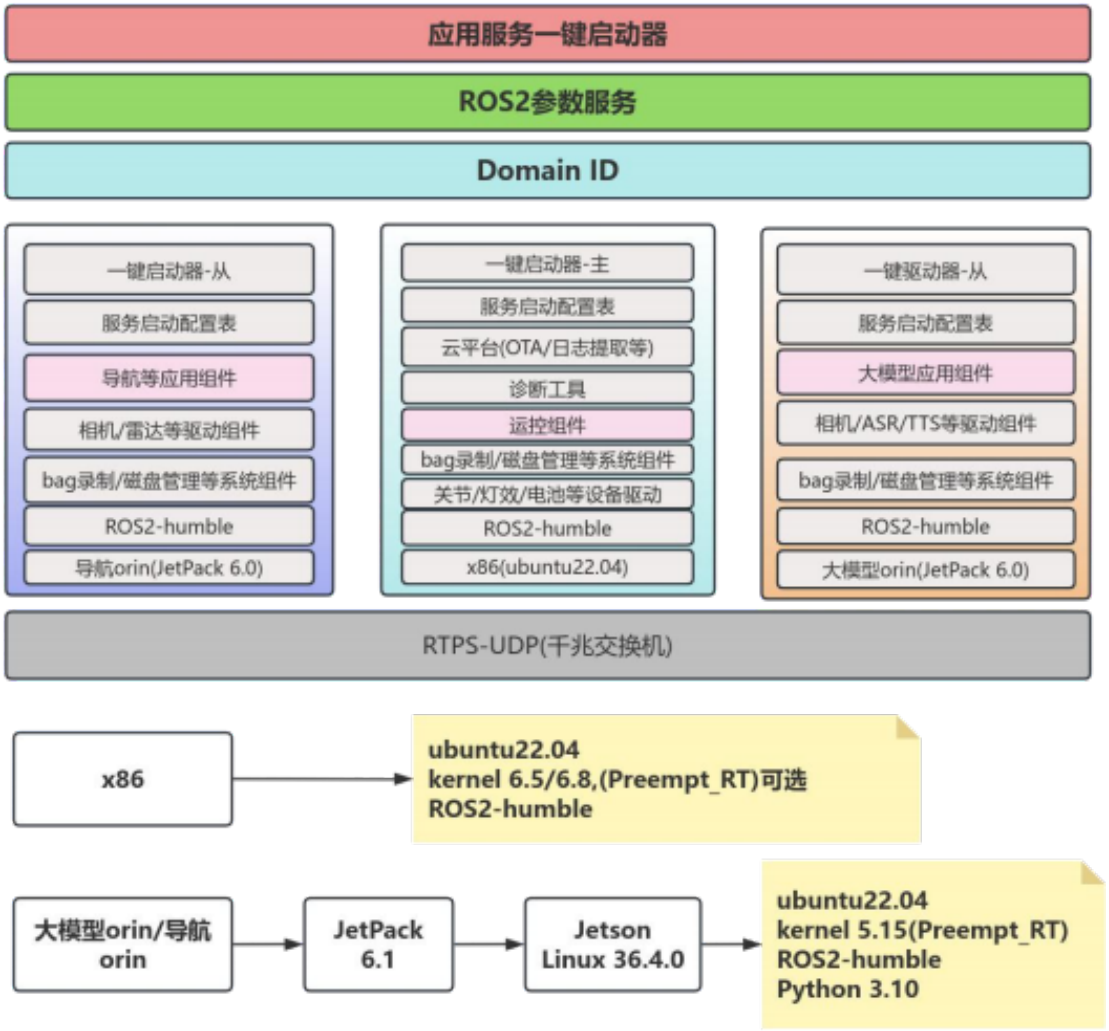
6. Press the main on/off key. At this time, all status lights will be completely extinguished. First, briefly press and then immediately long - press the power key on the remote control to turn off the remote control.

7. SDK Overview

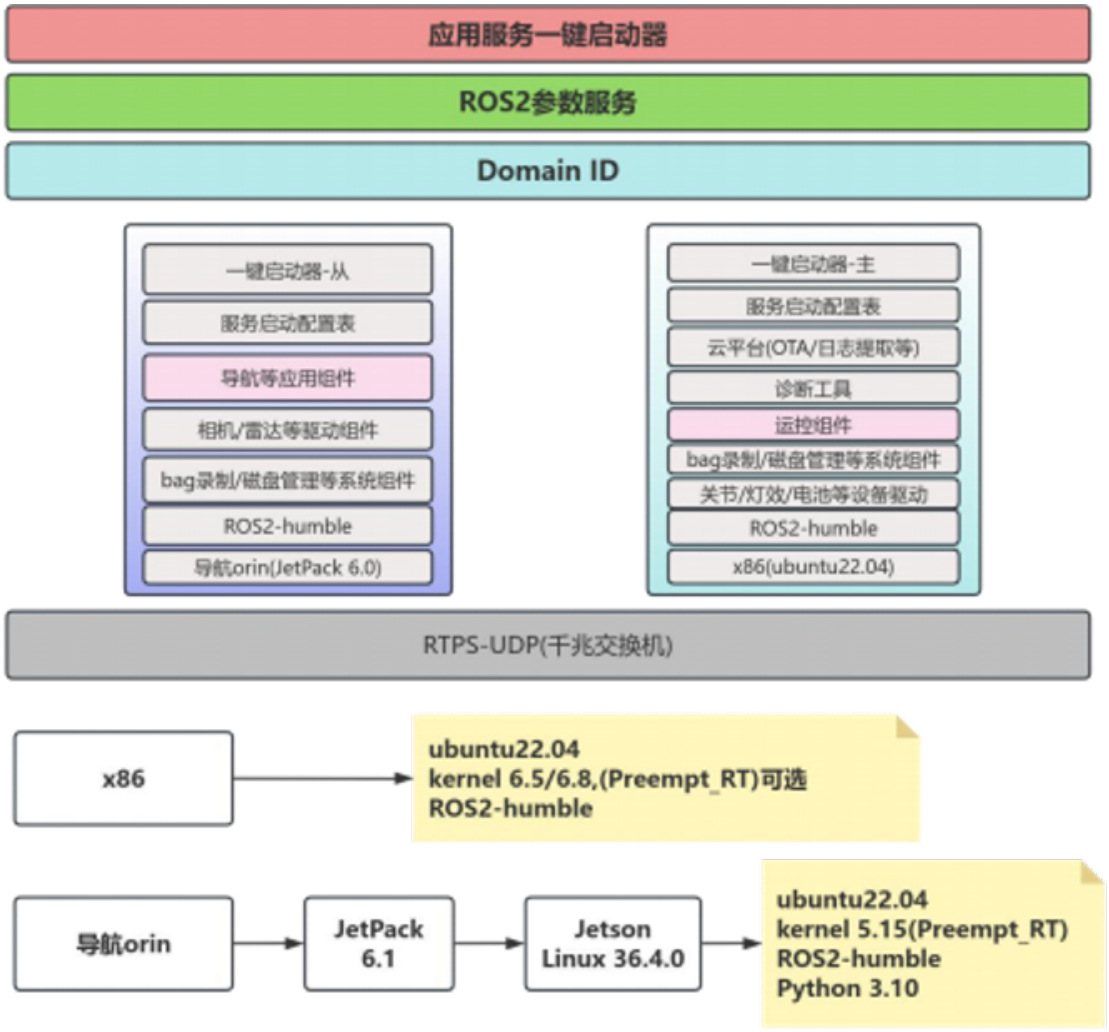
The SDK of the Walker Tienkung humanoid robot provides a wealth of interfaces, covering motor control of various parts, IMU (Inertial Measurement Unit), voice, battery, remote control, etc. It can be used to write and deploy robot applications, aiming to help developers quickly and flexibly build their own applications to precisely control and use the robot to meet the needs of different application scenarios.

7.1 Basic Framework - ROS2

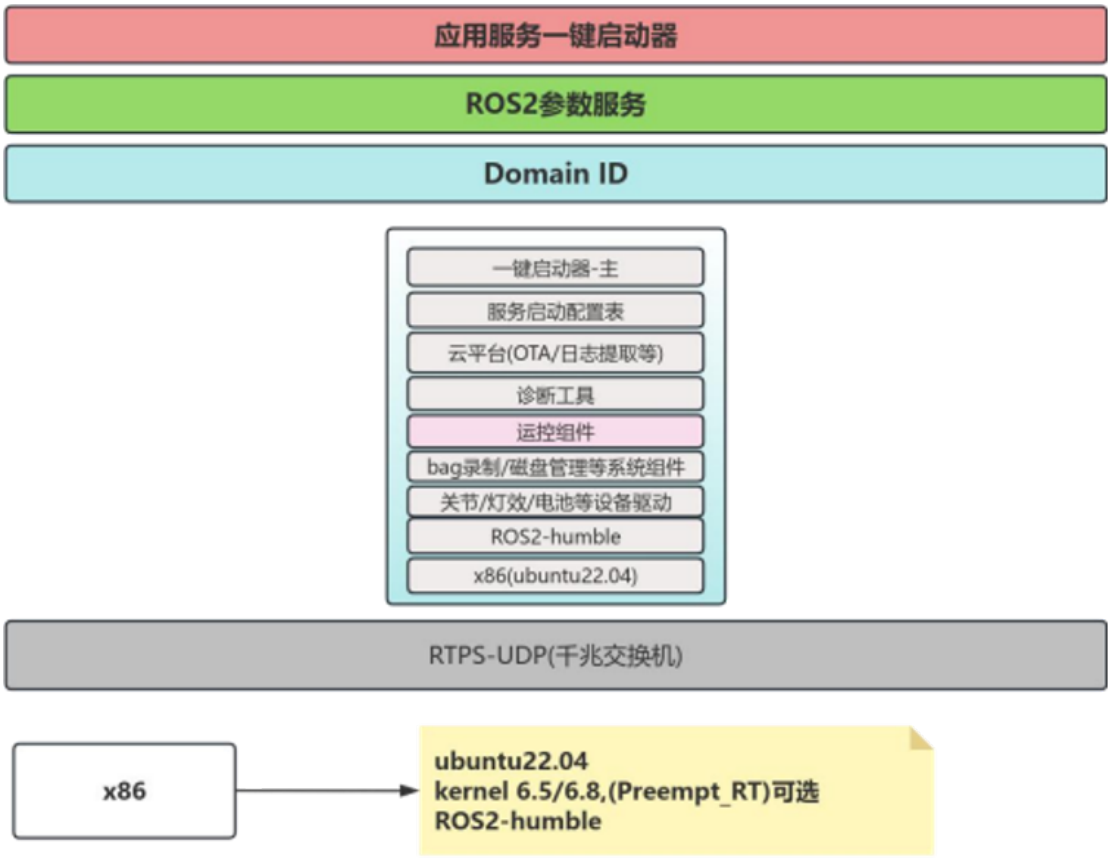
1. Basic framework of the Walker Tienkung -Embodied intelligence System



1. Basic framework of the Walker Tienkung - Voice & Vision System



1. Basic framework of the Walker Tienkung system



7.2 System Environment Dependencies

To ensure the best development experience and compatibility, it is recommended to develop on the Ubuntu 22.04 system. Currently, development on Mac and Windows systems is not supported.

7.3 Description of the operating environment

7.3.1 Overall parameter configuration path

```Bash
#x86 Global configuration parameter path
/home/ubuntu/data/param
#orin Global configuration parameter path
/home/nvidia/data/param
```

7.3.2 Configuration Parameters and Environment Variables

Location of configuration parameters：`/home/ubuntu/data/param/ros2_config.txt`

```
Python
ROS_LOG_DIR =/home/ubuntu/logs
ROS2_BAG_DIR =/home/ubuntu/bags
FASTRTPS_DEFAULT_PROFILES_FILE
=/home/ubuntu/data/param/dds_profile.xml RMW_IMPLEMENTATION
=rmw_fastrtps_cpp
ROS_DOMAIN_ID =0
```

Note:

ROS_LOG_DIR: Log storage location

ROS2_BAG_DIR: Bag file storage location

FASTRTPS_DEFAULT_PROFILES_FILE: DDS configuration file. By default, using Wi-Fi for DDS communication debugging is prohibited.

RMW_IMPLEMENTATION: Type of middleware used

ROS_DOMAIN_ID: Domain ID

Environment variable: /home/ubuntu/data/param/ros2_setup.bash

The above configuration parameters are automatically exported as environment variables when logging in to the terminal. You can also manually export them by executing the following commands.

```
Plain Text
source /home/ubuntu/data/param/ros2_setup.bash
```

8. Global value description

8.1 Definition of whole-body joint IDs

Message name： MotorName

Data definition location： MotorName.msg

Note: The naming ID definitions of all the motors on the Walker Tienkung series robots are as follows, which can be mapped according to the joint parts of different models of the Walker Tienkung.

```
Bash
代码块
# head 00 + id
# The head motors are all applicable to Walker Tienkung -Embodied
intelligence.
uint16 MOTOR_HEAD_1 = 1
```

```
uint16 MOTOR_HEAD_2 = 2
uint16 MOTOR_HEAD_3 = 3

# arm-left 10 + id
# Left arm motor, models 11 - 14 are suitable for all series of
Walker Tienkung.
uint16 MOTOR_ARM_LEFT_1 = 11
uint16 MOTOR_ARM_LEFT_2 = 12
uint16 MOTOR_ARM_LEFT_3 = 13
uint16 MOTOR_ARM_LEFT_4 = 14
# Left arm motor, models 15 - 17 are suitable for Walker Tienkung
-Embodied intelligence.
uint16 MOTOR_ARM_LEFT_5 = 15
uint16 MOTOR_ARM_LEFT_6 = 16
uint16 MOTOR_ARM_LEFT_7 = 17

# arm-right 20 + id
# Right arm motor, models 21 - 24 are suitable for all series of
Walker Tienkung.
uint16 MOTOR_ARM_RIGHT_1 = 21
uint16 MOTOR_ARM_RIGHT_2 = 22
uint16 MOTOR_ARM_RIGHT_3 = 23
uint16 MOTOR_ARM_RIGHT_4 = 24
# Right arm motor, suitable for Walker Tienkung -Embodied
intelligence (models 25 - 27)
uint16 MOTOR_ARM_RIGHT_5 = 25
uint16 MOTOR_ARM_RIGHT_6 = 26
uint16 MOTOR_ARM_RIGHT_7 = 27

# body (including waist) 30 + id
# Waist motor, suitable for Walker Tienkung -Embodied
intelligence/Walker Tienkung - Voice & Vision.
uint16 MOTOR_WAIST = 31

# leg-left 50 + id
# Left leg motor, models 51 - 56 are suitable for all series of
Walker Tienkung.
uint16 MOTOR_LEG_LEFT_1 = 51
uint16 MOTOR_LEG_LEFT_2 = 52
uint16 MOTOR_LEG_LEFT_3 = 53
uint16 MOTOR_LEG_LEFT_4 = 54
uint16 MOTOR_LEG_LEFT_5 = 55
uint16 MOTOR_LEG_LEFT_6 = 56
# leg-right 60 + id
```

```
# Right leg motor, models 61 - 66 are suitable for all series of
Walker Tienkung.
uint16 MOTOR_LEG_RIGHT_1 = 61
uint16 MOTOR_LEG_RIGHT_2 = 62
uint16 MOTOR_LEG_RIGHT_3 = 63
uint16 MOTOR_LEG_RIGHT_4 = 64
uint16 MOTOR_LEG_RIGHT_5 = 65
uint16 MOTOR_LEG_RIGHT_6 = 66
```

9. Interface description

9.1 Product serial number

9.1.1 Command line reading

```
Plain Text
xsn
```

9.1.2 C++ Integration

1. In CMakeLists.txt:

```
Plain Text
find_library(XSYS_LIB
NAMES xsys libxsys.so
PATHS /usr/local/lib
REQUIRED
)
target_link_libraries( <your_app> PRIVATE
${XSYS_LIB}
)
```

2. In the .cpp file:

```
C++
#include <xsys/xsys.h >
const std::string& serial_number = x::sys::ProductSerial Number();
```

9.1.3 ROS Interface

```
YAML
string SERVICE_NAME = /xsys/get_serial_number
```

```
---
string serial_number
```

9.2 One-click Startup

9.2.1 One-click startup function

One-key manual startup. Before starting, ensure that the auto-start service (proc_manager.service) has been disabled and stopped (use "ps -ef | grep ros" to check if there are any ros-related processes. If so, kill them):

Manually start the process manager:

ROS2：

Walker Tienkung

```
Bash
cd ros2ws
source install/setup.bash
ros2 launch proc_manager tg2.0_lite.launch.py
```

Walker Tienkung - Voice & Vision

```
Python
cd ros2ws
source install/setup.bash
ros2 launch proc_manager tg2.0_plus.launch.py
```

Walker Tienkung -Embodied Intelligence

```
Python
cd ros2ws
source install/setup.bash
ros2 launch proc_manager tg2.0_pro.launch.py
```

9.2.2 Enable/Disable Auto-start Function

The auto-start function is disabled by default. Use the following methods to enable or disable it.

1. Enable self-start.

```
Bash
sudo systemctl enable proc_manager.service
```

2. Disable auto-start.

```Bash
sudo systemctl disable proc_manager.service
```

9.2.3 Service Startup Configuration Instructions

The configuration of the self-starting service is as follows, and the configuration is located at：

`/home/ubuntu/rosws/install/share/diagnose/param/proc_manager.json`， The definition is as follows.：

Among them：

1. config：(Define global configuration parameters)：

```JSON
{
  "config": {
    "audio_player": "192.168.41.2", // The voice playback device is defined on [192.168.41.2](192.168.41.2) (Orin).
    "audio_path": "/home/nvidia/data/speech", // Define the audio directory for voice playback as /home/nvidia/data/speech.
    "audio_list": ["0.mp3", "1.mp3", "2.mp3", "3.mp3", "4.mp3"] // Define the list of voice files controlled by the remote control.
  },
  ...
}
```

2. proc：(Define the service process)：

```JSON
{
  ...
  "proc": [
    {
      "name": "power_board", // Service name
      "enable": true, // true：enable，false：disable
      "shell": "roslaunch power_board power_board.launch", // shell：Shell command to be executed
      "boot_start": true, // Does the startup automatically start: True: Yes, false: No
      "ready_topic": "power_board_state" // Service status ready's topic ，Waiting for the topic signal identification service to
```

```
start and complete.
    }, {
      "name": "usb_s bus",
      "enable": true,
      "shell": "roslaunch usb_sb us s bus.launch",
      "boot_start": true
    }, {
      "name": "body_control",
      "enable": true,
      "shell": "roslaunch body_control body_no_s bus.launch",
      "boot_start": false,
      "ready_topic": "bodycontrol_state",
      "ready_timeout": 20
    }, {
      "name": "motion",
      "enable": true,
      "shell": "roslaunch motion_control motion.launch"
    }
  ],
  ...
}
```

9.3 IMU

9.3.1 Format 1 for Obtaining IMU Sensor Information --- Standard Format

Description: Obtain the data information of the IMU sensor, including acceleration, angular velocity, pose quaternion, and Euler angles. It is applicable to all series of Walker Tienkung.

- Control method ：topic
- Topic name：/imu
- Data definition location： sensor_msgs::msg::Imu
- Data format：

```
Plain Text
std_msgs/Header header
geometry_msgs/Quaternion orientation
float64[9] orientation_covariance
geometry_msgs/Vector3 angular_velocity
float64[9] angular_velocity_covariance
```

```
geometry_msgs/Vector3 linear_acceleration
float64[9] linear_acceleration_covariance
```

- Example commands：

```
Bash
# View topic messages
ros2 topic echo /imu
```

9.3.2 Get IMU sensor information format 2 - Custom format

Description: Obtain the data information of the IMU sensor, including acceleration, angular velocity, pose quaternion, Euler angles, and error alarms. It is applicable to all series of Walker Tienkung.。

- Control method ：topic
- Topic name：/imu/status
- Data definition location： bodyctrl_msgs::msg::Imu
- Data format：

```
Plain Text
std_msgs/Header header
geometry_msgs/Quaternion orientation
geometry_msgs/Vector3 angular_velocity
geometry_msgs/Vector3 linear_acceleration
bodyctrl_msgs/Euler euler
uint32 error

float64[3] angular_velocity_covariance
float64[3] orientation_covariance
float64[3] linear_acceleration_covariance
```

error The definition of incorrect field values is as follows.：

```
Plain Text
error = 33072,        The device is offline.
```

```
Bash
#View topic messages
ros2 topic echo /imu/status
```

9.4 Joint Temperature Acquisition Interface

Instruction ：All joint temperatures are acquired and published in the form of a topic once per second. ;

- Data definition location： bodyctrl_msgs::msg::MotorStatus1
- Data format：

```
Plain Text
std_msgs/Header header
MotorStatus1[] status

MotorStatus1 It is defined as follows.:
uint16 name # MotorName
float32 motortemperature
float32 mostemperature
```

1. **Get the temperature information of the head joints, applicable to Walker Tienkung -Embodied intelligence.**

Topic name：/head/motor_status

```
Bash
ros2 topic echo /head/motor_status
```

2. **Get the temperature information of the double-arm joints, applicable to all series of Walker Tienkung.**

Topic name：/arm/motor_status

```
Bash
ros2 topic echo /arm/motor_status # The topic of Walker Tienkung - Voice & Vision.
```

3. **Obtain the temperature information of leg joints, suitable for all series of Walker Tienkung.**

Topic name：/leg/motor_status

```
Bash
ros2 topic echo /leg/motor_status # The topic of The topic of Walker Tienkung - Voice & Vision.
```

9.5 Head Joints

9.5.1 Status Acquisition Interface

1. Get head joint information

Description: Get the status information of the head joints, including the current position, speed, current, and temperature of the joints. It is applicable to the Walker Tienkung -Embodied intelligence.

Control method：topic

Topic name：/head/status

Data definition location： bodyctrl_msgs::msg::MotorStatusMsg.msg

Data format：

```
Plain Text
std_msgs/Header header
MotorStatus[] status

MotorStatus The definition is as follows.:
uint16 name        #MotorName
float32 pos         #rad
float32 speed       #rad
float32 current     #A
float32 temperature #MosTemperature
uint32 error
```

error The definition of the error field value is as follows.：

error = 33072,	Device Disconnected
error = 33073,	Joint Position Out of Limit
error = 1,	Joint Motor Overheating
error = 2,	Overcurrent
error = 3,	Undervoltage
error = 4,	Joint MOS Overheating
error = 5,	Stalling

error = 6,	Overvoltage
error = 7,	Phase Loss
error = 8,	Encoder Error

```
Bash
ros2 topic echo /head/status
```

9.5.2 Control Interface

1. Position mode

- Description: The position control interface for the head joints requires providing the desired position, desired velocity, and maximum current. It is applicable to the Walker Tienkung -Embodied intelligence.
- Control method：topic
- Topic name：`/head/cmd_pos`
- Data definition location：`bodyctrl_msgs::msg::CmdSetMotorPosition.msg`
- Data format：

```
Plain Text
std_msgs/Header header
SetMotorPosition[] cmds

#SetMotorPosition.msg
uint16 name   # MotorName
float32 pos   # rad
float32 spd   # rad/s
float32 cur   # A
```

1. Force-position hybrid mode

- Note: The force-position hybrid control interface for the head joints requires the provision of the desired position, desired velocity, feedforward torque, kp, and kd coefficients. It is applicable to the Walker Tienkung -Embodied intelligence.
- Control method：topic
- Topic name：`/head/cmd_ctrl`
- Data definition location：`bodyctrl_msgs::msg::CmdMotorCtrl.msg`
- Data format：

```
Plain Text
std_msgs/Header header
MotorCtrl[] cmds

# MotorCtrl.msg
uint16 name
float32 kp
float32 kd
float32 pos
float32 spd
float32 tor
```

1. Speed mode

- Note: The joint speed control interface requires the provision of the desired speed and maximum current. It is applicable to the Walker Tienkung -Embodied intelligence.
- Control method：topic
- Topic name：/head/cmd_vel
- Data definition location：bodyctrl_msgs::msg::CmdSetMotorSpeed.msg
- Data format：

```
Plain Text
std_msgs/Header header
SetMotorSpeed[] cmds

#SetMotorSpeed.msg
uint16 name # MotorName
float32 spd # rad/s
float32 cur # A
```

1. Zero the joints.

- Note: The zeroing interface for the head joints is applicable to Walker Tienkung - Embodied intelligence.
- Control method：topic
- Topic name：/head/cmd_set_zero
- Data definition location：std_msgs::msg::String
- Data format：

Send the joint IDs as strings to achieve zero calibration;

Head joint IDs: 1, 2, 3;

9.6 Lumbar joints

9.6.1 Status Acquisition Interface

1. Obtain information about the lumbar joints.

- Description: Obtain the status information of the lumbar joint, including the joint's current position, speed, current, and temperature. It is applicable to Walker Tienkung -Embodied intelligence/Walker Tienkung - Voice & Vision.
- Control method：topic
- Topic name：/waist/status
- Data definition location：bodyctrl_msgs::msg::MotorStatusMsg.msg
- Data format：

```
Go
std_msgs/Header header
MotorStatus[] status

MotorStatusThe definition is as follows:
uint16 name    # MotorName
float32 pos    # rad
float32 speed  # rad
float32 current # A
float32 temperature #
uint32 error
```

error The definition of the incorrect field values is as follows.：

error = 33072,	Device Disconnected
error = 33073,	Joint Position Out of Limit
error = 1,	Joint Motor Overheating
error = 2,	Overcurrent
error = 3,	Undervoltage

error = 4,	Joint MOS Overheating
error = 5,	Stalled Rotor
error = 6,	Overvoltage
error = 7,	Phase Loss
error = 8,	Encoder Error

9.6.2 Control Interface

1. Position mode

- Note: This is a joint position control interface. It requires the provision of the desired position, desired speed, and maximum current. It is applicable to the Walker Tienkung -Embodied intelligence/Walker Tienkung - Voice & Vision.
- Control method：topic
- Topic name：`/waist/cmd_pos`
- Data definition location：`bodyctrl_msgs::msg::CmdSetMotorPosition.msg`
- Data format：

```
Plain Text
std_msgs/Header header
SetMotorPosition[] cmds

# SetMotorPosition.msg
uint16 name  # MotorName
float32 pos  # rad
float32 spd  # rad/s
float32 cur  # A
```

```
Bash
ros2 topic pub /waist/cmd_pos
bodyctrl_msgs/msg/CmdSetMotorPosition "{header: {stamp: {sec: 0,
nanosec: 0 }, frame_id: ''},cmds: [{name: 31, pos: -0.3, spd: 0.2,
cur: 8.0 }]}"
ros2 topic pub /waist/cmd_pos
bodyctrl_msgs/msg/CmdSetMotorPosition "{header: {stamp: {sec: 0,
nanosec: 0 }, frame_id: ''},cmds: [{name: 31, pos: 0.0, spd: 0.2,
```

```
cur: 8.0 }]}"
# The header can also be omitted when using the ros2 topic pub
command. In the subsequent parts of this document, the header will
be temporarily omitted from the commands that include the header:
ros2 topic pub /waist/cmd_pos
bodyctrl_msgs/msg/CmdSetMotorPosition "{cmds: [{name: 31, pos: -
0.3, spd: 0.2, cur: 8.0 }]}"
ros2 topic pub /waist/cmd_pos
bodyctrl_msgs/msg/CmdSetMotorPosition "{cmds: [{name: 31, pos:
0.0, spd: 0.2, cur: 8.0 }]}"
```

2. Force-position hybrid mode

- Note: The force-position hybrid control interface for joints requires the provision of desired position, desired velocity, feedforward torque, kp and kd coefficients. It is applicable to Walker Tienkung -Embodied intelligence/Walker Tienkung - Voice & Vision.
- Control method：topic
- Topic name：`/waist/cmd_ctrl`
- Data definition location：`bodyctrl_msgs::msg::CmdMotorCtrl.msg`
- Data format：

```
Plain Text
std_msgs/Header header
MotorCtrl[] cmds

# MotorCtrl.msg
uint16 name
float32 kp
float32 kd
float32 pos
float32 spd
float32 tor
```

```
Bash
ros2 topic pub /waist/cmd_ctrl bodyctrl_msgs/msg/CmdMotorCtrl
"{cmds: [{name: 31,kp: 30.0,kd: 10.0,pos: -0.5,spd: 0.0,tor:
0.0}]}"
ros2 topic pub /waist/cmd_ctrl bodyctrl_msgs/msg/CmdMotorCtrl
"{cmds: [{name: 31,kp: 30.0,kd: 10.0,pos: 0.0,spd: 0.0,tor:
0.0}]}"
```

3. Speed mode

- Note: The speed control interface for joints requires the provision of the desired speed and maximum current. It is applicable to Walker Tienkung -Embodied intelligence/Walker Tienkung - Voice & Vision.
- Control mode：topic
- Topic name：/waist/cmd_vel
- Data definition location：bodyctrl_msgs::msg::CmdSetMotorSpeed.msg
- Data format：

```
Plain Text
std_msgs/Header header
SetMotorSpeed[] cmds

#SetMotorSpeed.msg
uint16 name # MotorName
float32 spd # rad/s
float32 cur # A
```

4. Zero the lumbar joints.

- Instruction: This is a zeroing interface for the waist joint. When this interface is called, the current position of the waist motor will be set as the zero position. When the "D" button on the remote control is pressed to return to zero, the waist motor will turn to the current position. It generally needs to be used in conjunction with a zeroing tool. Use it with caution if there is no zeroing tool. It is applicable to the Walker Tienkung -Embodied intelligence/Walker Tienkung - Voice & Vision.
- Control method：topic
- Topic name：/waist/cmd_set_zero
- Data definition location：std_msgs::msg::String
- Data format：

Send the joint ID as a string to achieve zero calibration;

Waist joint ID: 31

9.7 Double-arm joints

9.7.1 Status Acquisition Interface

1. Obtain the information of the arm joints.

- Description: Obtain the status information of the arm joints, including the current

position, speed, current, and temperature of the joints. It is applicable to all series of Walker Tienkung.

- Control mode：topic
- Topic name：/arm/status
- Data definition location：bodyctrl_msgs::msg::MotorStatusMsg.msg
- Data format：

```Go
std_msgs/Header header
MotorStatus[] status

MotorStatusThe definition is as follows.:
uint16 name    # MotorName
float32 pos    # rad
float32 speed  # rad
float32 current # A
float32 temperature
uint32 error
```

error The definition of incorrect field values is as follows.：

error = 33072,	Device offline
error = 33073,	Joint position out of limit
error = 1,	Joint motor overheating
error = 2,	Overcurrent
error = 3,	Undervoltage
error = 4,	Joint MOS overheating
error = 5,	Stalling
error = 6,	Overvoltage
error = 7,	Phase loss

error = 8,	Encoder error

9.7.2 Control Interface

1. Position mode

- Description: The position control interface for joints requires providing the desired position, desired velocity, and maximum current. It is applicable to all series of Walker Tienkung.
- Control method：topic
- Topic name：/arm/cmd_pos
- Data definition location：bodyctrl_msgs::msg::CmdSetMotorPosition.msg
- Data format：

```
Python
std_msgs/Header header
SetMotorPosition[] cmds

# SetMotorPosition.msg
uint16 name  # MotorName
float32 pos  # rad
float32 spd  # rad/s
float32 cur  # Electric current
```

```
Bash
ros2 topic pub /arm/cmd_pos bodyctrl_msgs/msg/CmdSetMotorPosition
"{cmds: [{name: 12, pos: 0.3, spd: 0.2, cur: 8.0 }]}"
ros2 topic pub /arm/cmd_pos bodyctrl_msgs/msg/CmdSetMotorPosition
"{cmds: [{name: 22, pos: -0.3, spd: 0.2, cur: 8.0 }]}"
```

2. Force-position hybrid mode

- Description: The force-position hybrid control interface for joints requires the provision of desired position, desired velocity, feedforward torque, kp and kd coefficients, and is applicable to all series of Walker Tienkung.
- Control method:topic
- Topic name:/arm/cmd_ctrl
- Data definition location：bodyctrl_msgs::msg::CmdMotorCtrl.msg
- Data format:

```
Plain Text
std_msgs/Header header
MotorCtrl[] cmds

# MotorCtrl.msg
uint16 name
float32 kp
float32 kd
float32 pos
float32 spd
float32 tor
```

```
Bash
ros2 topic pub /arm/cmd_ctrl bodyctrl_msgs/msg/CmdMotorCtrl
"{cmds: [{name: 13, kp: 30.0, kd: 10.0, pos: 0.2, spd: 0.0, tor:
0.0}]}"
ros2 topic pub /arm/cmd_ctrl bodyctrl_msgs/msg/CmdMotorCtrl
"{cmds: [{name: 23, kp: 30.0, kd: 10.0, pos: 0.2, spd: 0.0, tor:
0.0}]}"
```

3. Speed mode

- Description: The speed control interface for joints requires the provision of the desired speed and maximum current, and is applicable to all series of Walker Tienkung. Note that once this message is published, the motor will continue to rotate at this speed until a message with an spd of 0.0 is received on this topic. Therefore, it should be used with caution.
- Control method：topic
- Topic name：`/arm/cmd_vel`
- Data definition location：`bodyctrl_msgs::msg::CmdSetMotorSpeed.msg`
- Data format：

```
Plain Text
std_msgs/Header header
SetMotorSpeed[] cmds

#SetMotorSpeed.msg
uint16 name # MotorName
float32 spd # rad/s
float32 cur # A
```

5. Zero the arm joints.

- Instruction: This is the zero-calibration interface for the arm joints. When calling this interface, the current position of the specified motor of the arm will be set as the zero position. When pressing the "D" button on the remote control to return to zero, the motor will rotate to the current position. Generally, it needs to be used in conjunction with a zero-calibration tool. Use it with caution when there is no zero-calibration tool. It is applicable to all series of the Walker Tienkung.
- Control mode：topic
- Topic name：`/arm/cmd_set_zero`
- Data definition location：`std_msgs::msg::String`
- Data format：

Send the joint IDs as strings to implement zero calibration;

Joint IDs of both arms:

Left arm: 11---17,

Right arm: 21---27;

9.8 Leg joints

9.8.1 Status Acquisition Interface

1. Obtain the information of the leg joints.

- Description: Obtain the status information of the leg joints, including the current position, speed, current, temperature, and alarms of the joints. It is applicable to all models of the Walker Tienkung series.
- Control method：topic
- Topic name：`/leg/status`
- Data definition location：`bodyctrl_msgs::msg::MotorStatusMsg.msg`
- Data format：

```
Plain Text
std_msgs/Header header
MotorStatus[] status

MotorStatusThe definition is as follows.:
uint16 name    # MotorName
float32 pos    # rad
float32 speed  # rad
float32 current # A
```

```
float32 temperature
uint32 error
```

The definitions of error field values are as follows. :

error = 33072,	Device offline
error = 33073,	Joint position out of limit
error = 1,	Joint motor overheating
error = 2,	Overcurrent
error = 3,	Low voltage
error = 4,	Joint MOS overheating
error = 5,	Stalled rotor
error = 6,	High voltage
error = 7,	Phase loss
error = 8,	Encoder error

9.8.2 Control Interface

1. Position mode

- Description: The joint position control interface requires providing the desired position, desired velocity, and maximum current. It is applicable to all series of Walker Tienkung.
- Control method：topic
- Topic name：`/leg/cmd_pos`
- Data definition location：`bodyctrl_msgs::msg::CmdSetMotorPosition.msg`
- Data format：

```
Plain Text
std_msgs/Header header
```

```
SetMotorPosition[] cmds

# SetMotorPosition.msg
uint16 name  # MotorName
float32 pos  # rad
float32 spd  # rad/s
float32 cur  # A
```

2. Force-position hybrid mode

- Note: The force-position hybrid control interface for joints requires the provision of desired position, desired velocity, feedforward torque, kp, and kd coefficients. It is applicable to all models of the Walker Tienkung series.
- Control method：topic
- Topic name：`/leg/cmd_ctrl`
- Data definition location：`bodyctrl_msgs::msg::CmdMotorCtrl.msg`
- Data format：

```
Plain Text
std_msgs/Header header
MotorCtrl[] cmds

# MotorCtrl.msg
uint16 name
float32 kp
float32 kd
float32 pos
float32 spd
float32 tor
```

3. Speed mode

- Note: The joint speed control interface requires the provision of the desired speed and maximum current. It is applicable to all models of the Walker Tienkung series.
- Control method：topic
- Topic name：`/leg/cmd_vel`
- Data definition location：`bodyctrl_msgs::msg::CmdSetMotorSpeed.msg`
- Data format：

```
Plain Text
std_msgs/Header header
```

```
SetMotorSpeed[] cmds

#SetMotorSpeed.msg
uint16 name # MotorName
float32 spd # rad/s
float32 cur # A
```

4. Zero the leg joints.

- Note: This is the zeroing interface for the leg joints. When this interface is called, the current position of the specified motor in the leg will be set as the zero position. When the "D" button on the remote control is pressed for zeroing, the motor will turn to the current position. Generally, it needs to be used in conjunction with a zeroing tool. Use it with caution if there is no zeroing tool. It is applicable to all models of the Walker Tienkung series.
- Control method：topic
- Topic name：`/leg/cmd_set_zero`
- Data definition location：`std_msgs::msg::String`
- Data format：

Send the joint IDs as strings to achieve zero calibration;

Joint IDs of the legs:

Left leg: 51---56,

Right leg: 61---66;

9.9 Voice

9.9.1 ASR Sound Pickup

- Note: This is a voice recognition interface applicable to Walker Tienkung - Embodied intelligence/Walker Tienkung - Voice & Vision.
- Receiving method：topic
- Topic name：`/xunfei/aiui_msg`
- Data types：`std_msgs::msg::String`
- Data format：JSON

```JSON
{
  "type": "aiui_event",
```

```
  "content": {
    "eventType": 1,
    "info": {
      "data": [
        {
          "params": { "sub": "iat" },
          "content": [{ "dte": "utf8", "dtf": "json", "cnt_id":
"0" } ]
        }
      ]
    },
    "result": {
      "text": {
        "bg": 0, "sn": 1, "ws": [
          {"bg": 0,"cw": [{"w": "叫","sc": 0}]},
          {"bg": 0,"cw": [{"w": "什么","sc": 0}]},
          {"bg": 0,"cw": [{"w": "名字","sc": 0}]}
        ],
        "ls": false, "ed": 0
      }
    }
  }
}
```

Start the iFlytek service first on the Orin board at [192.168.41.2](192.168.41.2). You can use tmux. :

```Bash
. ~/voice_ws/install/setup.bash
ros2 launch xunfei_dev_socket xunfei_dev_all.launch.py
```

After starting the service, stand in front of the Walker Tienkung and speak. The command "ros2 topic echo /xunfei/aiui_msg" will generate output, but the text may not be fully visible. You can try using Python code. The reference code is as follows (for reference only). :

```Python
#!/usr/bin/env python3
from datetime import datetime
import glob
import json
import os
import rclpy
from rclpy.node import Node
```

```
from std_msgs.msg import String

MAX_FILES = 100
SAVE_DIR = 'json_data'

class ASRMonitor(Node):
    def __init__(self):
        super().__init__('asr_monitor')
        self.topic_name = '/xunfei/aiui_msg'
        self.subscription = self.create_subscription(
            String,
            self.topic_name,
            self.msgs_callback,
            10
        )

        os.makedirs(SAVE_DIR, exist_ok=True)
        self.get_logger().info(f"Subscribed topics：
{self.topic_name}")

    def msgs_callback(self, msg: String):
        json_data = None
        try:
            json_data = json.loads(msg.data)
        except json.JSONDecodeError:
            self.get_logger().warn("Received invalid JSON.
Skipping.")
            return

        # If you save here, all the messages will be saved,
there's actually no need
        # self.try_save_json_file(json_data)

        self.cleanup_old_files()

        self.try_parse_and_print(json_data)

    def try_save_json_file(self, json_data):
        try:
            os.makedirs(SAVE_DIR,exist_ok=True)
            timestamp = datetime.now().strftime('%H%M%S%f')[:-3]
# Hour, minute, second, and millisecond (with 3 bits reserved)
            filename = f"{timestamp}.json"
            filepath = os.path.join(SAVE_DIR, filename)
```

```
            with open(filepath, 'w', encoding='utf-8') as f:
                json.dump(json_data, f, ensure_ascii=False,
indent=2)
            self.get_logger().info(f"Saved JSON to {filepath}")
        except Exception as e:
            self.get_logger().error(f"Failed to write JSON file:
{e}")

    def try_parse_and_print(self, json_data):
        if not json_data:
            return
        if "content" not in json_data or "result" not in
json_data.get("content"):
            return

        result = json_data.get("content").get("result")
        if not result:
            return

        if "cbm_meta" not in result:
            return



        if not result.get("cbm_meta"):
            return
        cbm_meta = result.get("cbm_meta")
        if "text" not in cbm_meta:
            return

        text_data = json.loads(cbm_meta.get("text"))
        if not text_data:
            return
        key = next(iter(text_data))
        if key not in result:
            return
        result_text = result.get(key).get("text")
        try:
            res_data = json.loads(result_text)
            print(json.dumps(res_data, indent=2,
ensure_ascii=False))

        except json.JSONDecodeError:
            self.get_logger().info(f"content.result.{key}.text: ")
            print(result_text)
```

```
        self.try_save_json_file(json_data)

        if result_text:
            print("-" * 20 + "\r\n")

    def cleanup_old_files(self):
        files = sorted(
            glob.glob(os.path.join(SAVE_DIR, '*.json')),
            key=os.path.getmtime
        )
        if len(files) > MAX_FILES:
            to_delete = files[:len(files) - MAX_FILES]
            for f in to_delete:
                try:
                    os.remove(f)
                    self.get_logger().info(f"Deleted old file:
{f}")
                except Exception as e:
                    self.get_logger().warn(f"Failed to delete {f}:
{e}")

def main(args=None):
    rclpy.init(args=args)
    node = ASRMonitor()
    try:
        rclpy.spin(node)
    except KeyboardInterrupt:
        pass
    node.destroy_node()
    rclpy.shutdown()

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

9.9.2 **Voice playback**

- Description: Voice playback interface, suitable for Walker Tienkung -Embodied intelligence/Walker Tienkung - Voice & Vision.
- Receiving method：topic
- Topic name：`/xunfei/tts_play`
- Data types：`std_msgs::msg::String`

- Data format：JSON

1. Start playing.

```JSON
{
  "file": "path/to/audio/file.mp3"
}
```

1. Stop playing.

```JSON
{
  "cmd": "stop"
}
```

3. Continuous playback

```JSON
{
  "cmd": "append",
  "file": "path/to/audio/file.mp3"
}
```

```Bash
. ~/voice_ws/install/setup.bash
ros2 launch xunfei_dev_socket xunfei_dev_all.launch.py
# You can try playing the following .mp3 audio files. Make sure
these files exist before playing. You can also obtain audio files
by yourself for playback.
ros2 topic pub /xunfei/tts_play std_msgs/msg/String "{data:
'{\"file\": \"/home/nvidia/data/speech/chenggong.mp3\"}'}"
ros2 topic pub /xunfei/tts_play std_msgs/msg/String "{data:
'{\"file\": \"/home/nvidia/data/speech/guzhang.mp3\"}'}"
ros2 topic pub /xunfei/tts_play std_msgs/msg/String "{data:
'{\"file\": \"/home/nvidia/data/speech/didianliang.wav\"}'}"
ros2 topic pub /xunfei/tts_play std_msgs/msg/String "{data:
'{\"file\": \"/home/nvidia/data/speech/kaishichongdian.mp3\"}'}"
ros2 topic pub /xunfei/tts_play std_msgs/msg/String "{data:
'{\"file\": \"/home/nvidia/data/speech/anjianyin.mp3\"}'}"
```

9.10 Depth Camera

Method for starting the Orbbec camera - ROS2

- Walker Tienkung - Voice & Vision，Orin1（192.168.41.2）：

```
Bash
cd /home/nvidia/orbbec_camera_ros2
source install/setup.bash
ros2 launch orbbec_camera slam_330.launch.py
```

Walker Tienkung -Embodied intelligence，Orin1(192.168.41.2)：

```
Bash
cd /home/nvidia/orbbec_camera_ros2
source install/setup.bash
ros2 launch orbbec_camera gemini_330_series.launch.py
```

Walker Tienkung -Embodied intelligence, Orin2 ([192.168.41.3](192.168.41.3))：

```
Bash
cd /home/nvidia/orbbec_camera_ros2
source install/setup.bash
ros2 launch orbbec_camera slam_330.launch.py
```

After starting, use `ros2 node list` to view the nodes. You can clearly see that there is an additional `/camera/camera` node. Use the `ros2 node info /camera/camera` command to view the node information.

The topics to be published by default are as follows:

```
Bash
/camera/color/camera_info: sensor_msgs/msg/CameraInfo
/camera/color/image_raw: sensor_msgs/msg/Image
/camera/color/metadata: orbbec_camera_msgs/msg/Metadata
/camera/depth/camera_info: sensor_msgs/msg/CameraInfo
/camera/depth/image_raw: sensor_msgs/msg/Image
/camera/depth/metadata: orbbec_camera_msgs/msg/Metadata
/camera/depth/points: sensor_msgs/msg/PointCloud2
/camera/depth_filter_status: std_msgs/msg/String
/camera/depth_to_color: orbbec_camera_msgs/msg/Extrinsics
/diagnostics: diagnostic_msgs/msg/DiagnosticArray
/parameter_events: rcl_interfaces/msg/ParameterEvent
/rosout: rcl_interfaces/msg/Log
/tf: tf2_msgs/msg/TFMessage
/tf_static: tf2_msgs/msg/TFMessage
```

The services provided by default are as follows:

```Bash
/camera/camera/describe_parameters:
rcl_interfaces/srv/DescribeParameters
/camera/camera/get_parameter_types:
rcl_interfaces/srv/GetParameterTypes
/camera/camera/get_parameters: rcl_interfaces/srv/GetParameters
/camera/camera/list_parameters: rcl_interfaces/srv/ListParameters
/camera/camera/set_parameters: rcl_interfaces/srv/SetParameters
/camera/camera/set_parameters_atomically:
rcl_interfaces/srv/SetParametersAtomically
/camera/get_auto_white_balance: orbbec_camera_msgs/srv/GetInt32
/camera/get_color_exposure: orbbec_camera_msgs/srv/GetInt32
/camera/get_color_gain: orbbec_camera_msgs/srv/GetInt32
/camera/get_depth_exposure: orbbec_camera_msgs/srv/GetInt32
/camera/get_depth_gain: orbbec_camera_msgs/srv/GetInt32
/camera/get_device_info: orbbec_camera_msgs/srv/GetDeviceInfo
/camera/get_ldp_status: orbbec_camera_msgs/srv/GetBool
/camera/get_lrm_measure_distance: orbbec_camera_msgs/srv/GetInt32
/camera/get_sdk_version: orbbec_camera_msgs/srv/GetString
/camera/get_white_balance: orbbec_camera_msgs/srv/GetInt32
/camera/reboot_device: std_srvs/srv/Empty
/camera/save_images: std_srvs/srv/Empty
/camera/save_point_cloud: std_srvs/srv/Empty
/camera/set_auto_white_balance: std_srvs/srv/SetBool
/camera/set_color_ae_roi: orbbec_camera_msgs/srv/SetArrays
/camera/set_color_auto_exposure: std_srvs/srv/SetBool
/camera/set_color_exposure: orbbec_camera_msgs/srv/SetInt32
/camera/set_color_flip: std_srvs/srv/SetBool
/camera/set_color_gain: orbbec_camera_msgs/srv/SetInt32
/camera/set_color_mirror: std_srvs/srv/SetBool
/camera/set_color_rotation: orbbec_camera_msgs/srv/SetInt32
/camera/set_depth_ae_roi: orbbec_camera_msgs/srv/SetArrays
/camera/set_depth_auto_exposure: std_srvs/srv/SetBool
/camera/set_depth_exposure: orbbec_camera_msgs/srv/SetInt32
/camera/set_depth_flip: std_srvs/srv/SetBool
/camera/set_depth_gain: orbbec_camera_msgs/srv/SetInt32
/camera/set_depth_mirror: std_srvs/srv/SetBool
/camera/set_depth_rotation: orbbec_camera_msgs/srv/SetInt32
/camera/set_fan_work_mode: orbbec_camera_msgs/srv/SetInt32
/camera/set_filter: orbbec_camera_msgs/srv/SetFilter
/camera/set_floor_enable: std_srvs/srv/SetBool
/camera/set_ir_long_exposure: std_srvs/srv/SetBool
/camera/set_laser_enable: std_srvs/srv/SetBool
/camera/set_ldp_enable: std_srvs/srv/SetBool
```

```
/camera/set_read_customer_data: orbbec_camera_msgs/srv/SetString
/camera/set_reset_timestamp: std_srvs/srv/SetBool
/camera/set_sync_hosttime: std_srvs/srv/SetBool
/camera/set_sync_interleaverlaser: orbbec_camera_msgs/srv/SetInt32
/camera/set_white_balance: orbbec_camera_msgs/srv/SetInt32
/camera/set_write_customer_data: orbbec_camera_msgs/srv/SetString
/camera/switch_ir: orbbec_camera_msgs/srv/SetString
/camera/toggle_color: std_srvs/srv/SetBool
/camera/toggle_depth: std_srvs/srv/SetBool
```

- Orin2 ([192.168.41.3](192.168.41.3)), suitable for Walker Tienkung -Embodied intelligence:

```
Bash
cd /home/nvidia/orbbec_camera_ros2
source install/setup.bash
ros2 launch orbbec_camera slam_330.launch.py
```

9.11 Hand joints

9.11.1 Global value description

- Message name：`MotorName`
- Data definition location：
- Note: The definitions of the IDs of each joint of the fingers are as follows, which are applicable to the Walker Tienkung -Embodied intelligence.：

```
Plain Text
uint16 MOTOR_FINGER_LITTLE = 1,        //little finger
uint16 MOTOR_FINGER_RING = 2,          //ring finger
uint16 MOTOR_FINGER_MIDDLE = 3,        //middle finger
uint16 MOTOR_FINGER_FORE = 4,          //Fore finger
uint16 MOTOR_FINGER_THUMB_BEND = 5,    //thumb bend
uint16 MOTOR_FINGER_THUMB_ROTATION = 6,//thumb rotation
```

9.11.2 Status Acquisition Interface

- Description: Obtain the status information of each finger joint, including the current position, speed, and current of the motor. It is applicable to Walker Tienkung -Embodied intelligence.
- Control method：topic
- Topic name：`/inspire_hand/state/left_hand` and `/inspire_hand/state/right_hand`

- Data definition location：sensor_msgs::msg::JointState

Note: The data here (position, speed, current) refers to percentages.

```
Bash
std_msgs/Header header
string[] name    #Note: id = 1 represents the little finger, and
id = 6 represents the rotation of the thumb.
float64[] position  #%
float64[] velocity  #%
float64[] effort  #%
```

9.11.3 Control interface-topic

1. Control the angles of each joint of the fingers.

- Note: The position control interface of the hand motor requires the provision of the desired position, desired speed, and maximum current. It is applicable to Walker Tienkung -Embodied intelligence.
- Control method：topic
- Topic name：/inspire_hand/ctrl/left_hand 和/inspire_hand/ctrl/right_hand
- Data definition location：sensor_msgs::msg::JointState.msg

Note: The "position" here still refers to the percentage of finger extension.

```
Plain Text
std_msgs/Header header
string[] name
float64[] position  #%
```

9.11.4 Control interface-service

1. Control the angles of each joint of the fingers.

- Description: Position control interface for each joint of the finger, in percentage, suitable for Walker Tienkung -Embodied intelligence.
- Control method：service
- Topic name：/inspire_hand/set_angle_flexible/left_hand 和 /inspire_hand/set_angle_flexible/right_hand
- Data definition location：bodyctrl_msgs::srv::set_angle_flexible.srv

```
Plain Text
string[]name
```

```
float32[]angle_ratio
bool angle_accepted
```

9.11.5 Control the torque of each joint of the finger

- Note: The position torques of each joint of the fingers are presented in percentage, applicable to Walker Tienkung -Embodied intelligence.
- Control mode：service
- Topic name：/inspire_hand/set_force/left_hand and /inspire_hand/set_force/right_hand
- Data definition location：bodyctrl_msgs::srv::set_force.srv
- request：

```
Plain Text
float32 force0_Ratio #Little finger - Percentage (0 represents 0g, 1 represents 1000g)
float32 force1_Ratio #Ring finger - Percentage
float32 force2_Ratio #Middle finger - Percentage
float32 force3_Ratio #Index finger - Percentage
float32 force4_Ratio #Thumb bending - Percentage
float32 force5_Ratio #Thumb rotation - Percentage
```

- response：

```
Plain Text
bool force_accepted #Is the setting successful?
```

9.11.6 Control the speed of each joint of the finger

- Description: The position torques of each joint of the fingers are presented in percentage and are applicable to Walker Tienkung -Embodied intelligence.
- Control method：service
- Topic name：/inspire_hand/set_speed/left_hand 和 /inspire_hand/set_speed/right_hand
- Data definition location：bodyctrl_msgs::srv::set_speed.srv
- request：

```
Plain Text
float32 speed0_Ratio #Little finger - Percentage (1 represents 800ms from the maximum angle to the minimum angle, 0.5 represents
```

```
1600ms, and 0.25 represents 3200ms)
float32 speed1_Ratio #Ring finger - Percentage
float32 speed2_Ratio #Middle finger - Percentage
float32 speed3_Ratio #Index finger - Percentage
float32 speed4_Ratio #Thumb bend - Percentage
float32 speed5_Ratio #Thumb rotation - Percentage
```

response

```
Plain Text
bool speed_accepted #Is the setting successful?
```

9.11.7 Clear errors in each finger joint

- Instruction: Clear the incorrect interfaces of each finger joint. Applicable to Walker Tienkung -Embodied intelligence.
- Control mode：service
- Topic name：/inspire_hand/set_clear_error/left_hand 和 /inspire_hand/set_clear_error/right_hand
- Data definition location：bodyctrl_msgs::srv::set_clear_error.srv
- Request

```
Plain Text
null
```

Response

```
Plain Text
boll setclear_error_accepted #Is the setting successful?
```

9.12 Six-dimensional force

1. Six-dimensional force status acquisition interface

- Description: Obtain the current information of the six-axis force sensor. The reporting frequency is 1000Hz, which is applicable to Walker Tienkung -Embodied intelligence.
- Control method：topic
- Topic name：/arm_6dof_left and /arm_6dof_right
- Type：geometry_msgs::msg::WrenchStamped.msg
- Data format：

```
Plain Text
Header header uint32 seq #Serial number
Header header time stamp #Timestamp
Header header string frame_id #Frame ID

Wrench wrench Vector3 force #Forces in three directions
Wrench wrench Vector3 torque #Moments in three directions
```

9.13 Battery Status and Power Level Acquisition

1. Battery status

- Description: Obtain battery status information. The reporting frequency is 1 Hz, which is applicable to all series of Walker Tienkung.
- Charging status: The `master_battery_current` field. A negative value indicates discharging, while a positive value indicates charging.
- Control method：topic
- Topic name：/power/battery/status
- Type：bodyctrl_msgs::msg::PowerBatteryStatus.msg
- Data format：

```
C++
# @brief Power board voltage, current, battery level, temperature
and other information

std_msgs/Header header

# battery status information
int32 battery_installed  # 0x00 means neither battery exists, 0x01
means the small battery exists, 0x02 means the large battery
exists, and 0x03 means both the small and large batteries exist.
int32 battery_working    # 0x01large battery, 0x10 smalle battery

float32 master_battery_voltage # Battery voltage information
float32 master_battery_current # Battery current information
float32 master_battery_power   # Battery level

float32 little_battery_voltage # Voltage information of the small
battery
float32 little_battery_current # Small battery current information
float32 little_battery_power   # Small battery charge
```

```
int8 pg12a # 00 low, 01 high.
int8 pg12b # 00 low，01 high
int8 pg12c # 00 low，01 high
int8 pg12d # 00 low，01 high
int8 pg5cd # 00 low，01 high
int8 pg5ab # 00 low，01 high

int8 pgrdc2    # 00 low，01 high
int8 pgrdc1    # 00 low，01 high
int8 pgheader   # 00 low，01 high
int8 pgbutton2 # 00 low，01 high
```

2. Power board status

- Description: Obtain the status information of the power supply board. The reporting frequency is 1 Hz, applicable to all series of Walker Tienkung.
- Control method：topic
- Topic name：/power/board/status
- Type：bodyctrl_msgs::msg::PowerStatus.msg
- Data format：

```
C++
# @brief Information such as voltage, current, power, and
temperature of the power supply board

std_msgs/Header header

# temperature information
float32 waist_temp # Waist MOS temperature
float32 arm_a_temp # Arm A MOS temperature
float32 arm_b_temp # Arm B MOS temperature
float32 leg_a_temp # Leg A MOS temperature
float32 leg_b_temp # Leg B MOS temperature

float32 waist_temp_max # Maximum temperature of the MOS at the
waist
float32 arm_a_temp_max # Maximum temperature of Arm A MOS
float32 arm_b_temp_max # Maximum temperature of Arm B MOS
float32 leg_a_temp_max # Maximum temperature of leg A MOS
float32 leg_b_temp_max # Maximum temperature of leg B MOS

float32 waist_temp_min # Minimum value of MOS temperature at the
```

```
waist
float32 arm_a_temp_min # Minimum MOS temperature of Arm A
float32 arm_b_temp_min # Minimum temperature of Arm B MOS
float32 leg_a_temp_min # Minimum MOS temperature of leg A
float32 leg_b_temp_min # Minimum temperature of Leg B MOS

# current information
float32 arm_a_curr # Arm A current
float32 arm_b_curr # Arm B current
float32 leg_a_curr # Leg A current
float32 leg_b_curr # Leg B current
float32 waist_curr # Lumbar current
float32 head_curr  # Head current

float32 arm_a_curr_max # Maximum current of Arm A
float32 arm_b_curr_max # Maximum current of Arm B
float32 leg_a_curr_max # Maximum current of leg A
float32 leg_b_curr_max # Maximum current of leg B
float32 waist_curr_max # Maximum current in the waist
float32 head_curr_max  # Maximum value of head current

float32 arm_a_curr_min # Minimum current of Arm A
float32 arm_b_curr_min # Minimum current of Arm B
float32 leg_a_curr_min # Minimum current of leg A
float32 leg_b_curr_min # Minimum current of leg B
float32 waist_curr_min # Minimum current for the waist
float32 head_curr_min  # Minimum value of head current

float32 arm_a_volt #Arm A voltage
float32 arm_b_volt #Arm B voltage
float32 leg_a_volt #Leg A voltage
float32 leg_b_volt #Leg B voltage
float32 waist_volt #Lumbar voltage
float32 bus_volt   #Bus voltage

float32 arm_a_volt_max #Maximum voltage of arm A
float32 arm_b_volt_max #Maximum voltage of Arm B
float32 leg_a_volt_max #Maximum voltage of leg A
float32 leg_b_volt_max #Maximum voltage of Leg B
float32 waist_volt_max #Maximum voltage at the waist
float32 bus_volt_max #Maximum value of bus voltage

float32 arm_a_volt_min #Minimum voltage of Arm A
float32 arm_b_volt_min #Minimum voltage of Arm B
```

```
float32 leg_a_volt_min #Minimum voltage of leg A
float32 leg_b_volt_min #Minimum voltage of leg B
float32 waist_volt_min #Minimum waist voltage
float32 bus_volt_min #Minimum value of bus voltage

# version information
string software_version # Software version
string hardware_version # Hardware version

# battery status information
float32 battery_voltage # Battery voltage information
float32 battery_current # Battery current information
float32 battery_power   # Battery level
```

3. Emergency stop button status

- Description: Obtain the status information of the emergency stop button. The reporting frequency is 1 Hz, which is applicable to all series of Walker Tienkung.
- Control method：topic
- Topic name：/power/board/key_status
- Type：bodyctrl_msgs::msg::PowerBoardKeyStatus.msg
- Data format：

```
C++
# @brief Power board button status

std_msgs/Header header

# power board status
uint32 work_time               # Working hours
std_msgs/Bool is_estop         # Is the emergency stop button
pressed?
std_msgs/Bool is_remote_estop  # Is the soft emergency stop button
pressed?
std_msgs/Bool is_power_on      # Is the power supply working
properly?
```

9.14 Remote control interface

1. Remote control event data

- Description: Obtain the original data of the remote control, applicable to all series

of Walker Tienkung.

- Control method：topic
- Topic name：/sbus_data/event
- Type： bodyctrl_msgs::msg::SbusData.msg
- Data format：

```
C++
int32 KEY_NONE =0
int32 KEY_A_UP =1          # Key A release event
int32 KEY_A_DOWN =2        # Key A press event
int32 KEY_B_UP =3          # Key B release event
int32 KEY_B_DOWN =4        # Key B press event
int32 KEY_C_UP =5          # Key C release event
int32 KEY_C_DOWN =6        # Key C press event
int32 KEY_D_UP =7          # Key D release event
int32 KEY_D_DOWN =8        # Key D press event
int32 KEY_E_UP =9          # Key E up event
int32 KEY_E_MID =10        # Key E mid event
int32 KEY_E_DOWN =11       # Key E down event
int32 KEY_F_UP =12         # Key F up event
int32 KEY_F_MID =13        # Key F mid event
int32 KEY_F_DOWN =14       # Key F down event
int32 KEY_G_LEFT =15       # Key G left event
int32 KEY_G_MID =16        # Key G mid event
int32 KEY_G_RIGHT =17      # Key G right event
int32 KEY_H_LEFT =18       # Key H left event
int32 KEY_H_MID =19        # Key H mid event
int32 KEY_H_RIGHT =20      # Key H right event

std_msgs/Header header
int32 key_event_new        # New value of key event
int32 key_event_old        # Old value of key event
                           # For example, if the G key is moved
from the right position to the middle position, then key_event_old
=KEY_G_RIGHT , key_event_new =KEY_G_MID

int8 button_a              # Key value of key A  ,  -1 : release ,
1 : press
int8 button_b              # Key value of key B  ,  -1 : release ,
1 : press
int8 button_c              # Key value of key C  ,  -1 : release ,
1 : press
```

```
int8 button_d              # Key value of key D  ,  -1 : release ,
1 : press
int8 button_e              # Key value of key E  ,  <-0.5 : up ,  -
0.5~0.5:  mid ,  >0.5 : down
int8 button_f              # Key value of key F  ,  <-0.5 : up ,  -
0.5~0.5:  mid ,  >0.5 : down
int8 button_g              # Key value of key G  ,  <-0.5 : left ,
-0.5~0.5:  mid ,  >0.5 : right
int8 button_h              # Key value of key H  ,  <-0.5 :
right ,  -0.5~0.5:  mid ,  >0.5 : left

float32 x1                 # Left joystick X-direction value
(left-right direction), range -1.0 to 1.0
float32 y1                 # Left joystick Y-direction value (up
and down direction), range -1.0 to 1.0
float32 x2                 # Right joystick X-direction value
(left-right direction), range: -1.0 to 1.0
float32 y2                 # The Y-axis value (up and down
direction) of the right joystick, ranging from -1.0 to 1.0
```

9.15 Software Version Information Acquisition

The complete version information file (including the complete version information description files for x86, Orin, firmware, etc.) is defined in the installation directory of the main software package and is applicable to all series of Walker Tienkung.

The path of ROS2 is：`/home/ubuntu/ros2ws/version_info.json` The format is as follows.：

```JSON
{
    "product": "TG2.0-Pro",
    "version": "v2.0.0 ",
    "commit":
"bu/integration/tiangong2.0/pro,82d2486835ba3436e43b8d5cee31329713
67a5 0a",
    "release_time": "2025-06-11 16:47:23",
    "x86 ": [
        {
            "module": "benti",
            "status": "enable",
            "product": "TG2.0-Pro",
            "platform": "linux-x86",
```

```
            "version": "v1.0.12",
            "file_name": "TG2.0-Pro_linux-
x86_general_v1.0.12_20250611_163512.tar.gz",
            "commit":
"bu/project/tiangong2.0/pro,ee03d5862208501074a0f3e766354667595b54
87 ",
            "install_dir_name": "ros2ws",
            "install_address": "192.168.41.1 ",
            "release_time": "2025-06-11 16:44:44"
        },
        {
            "module": "ota",
            "status": "disable",
            "product": "ota",
            "platform": "linux-x86",
            "version": "1.1.4 ",
            "file_name": "ota_linux-x86_general_ 1.1.4_20250526_
135636.tar.gz",
            "commit":
"bu/base/ota,ee523b29c66c26694a9c42ecda9d7e27faac8f06 ",
            "install_dir_name": "ota",
            "install_address": "192.168.41.1 ",
            "release_time": "2025-05-26 13:56:36"
        }
    ],
    "orin_slam": [
        {
            "module": "orbbec_camera ",
            "status": "enable",
            "product": "OrbbecCamera-Orin",
            "platform": "ros2",
            "version": "v2.0.0 ",
            "file_name": "OrbbecCamera-
Orin_ros2_general_v2.0.0_20250424_193011.tar.gz",
            "commit":
"bu/project/camera/orbbec_camera,b7e38e3a66c05784fa6cc1fe2e7a20012
5a3a 52b",
            "install_dir_name": "orbbec_camera_ros2",
            "install_address": "192.168.41.3 ",
            "release_time": "2025-04-24 19:31:27"
        }
    ],
    "orin_llm": [
        {
```

```
            "module": "orbbec_camera ",
            "status": "enable",
            "product": "OrbbecCamera-Orin",
            "platform": "ros2",
            "version": "v2.0.0 ",
            "file_name": "OrbbecCamera-
Orin_ros2_general_v2.0.0_20250424_193011.tar.gz",
            "commit":
"bu/project/camera/orbbec_camera,b7e38e3a66c05784fa6cc1fe2e7a20012
5a3a 52b",
            "install_dir_name": "orbbec_camera_ros2",
            "install_address": "192.168.41.2 ",
            "release_time": "2025-04-24 19:31:27"
        },
        {
            "module": "voice",
            "status": "enable",
            "product": "Voice-Orin",
            "platform": "ros2",
            "version": "v2.0.1 ",
            "file_name": "Voice-Orin_ros2_general_v2.0.1_20250515_
111213.tar.gz",
            "commit":
"bu/project/common/xunfei_dev_socket,ffb48eb38fa2755da99d4ce290535
82c1 a970039 ",
            "install_dir_name": "voice_ros2",
            "install_address": "192.168.41.2 ",
            "release_time": "2025-05-15 11:12:13"
        }
    ],
    "firmware": [
        {
            "module": "joint-e2c",
            "status": "disable",
            "product": "TG2.0-SelfJoint-e2c",
            "platform": "",
            "version": "v1.0.0 ",
            "file_name": "TG2.0-SelfJoint-
e2c_general_v1.0.0_20250418_203500.tar.gz",
            "commit":
"Universal
HumanoidPrototypeGroup/OldProjects/mcu_ota,a0035672ca25147b7
a6fdf0fa3bf621d9c7bef85 ","install_dir_name": "joint-e2c",
            "install_address": "",
```

```
            "release_time": "2025-04-28_10:25:23"
        },
        {
            "module": "joint-u2c",
            "status": "disable",
            "product": "TG2.0-SelfJoint-u2c",
            "platform": "",
            "version": "v1.0.0 ",
            "file_name": "TG2.0-SelfJoint-
u2c_general_v1.0.0_20250418_214000.tar.gz",
            "commit":
"Universal
HumanoidPrototypeGroup/OldProjects/mcu_ota,bae67e7278ea0c67a485d8e
4245d0a23345b1185","install_dir_name": "joint-u2c",
            "install_address": "",
            "release_time": "2025-04-28_14:10:23"
        }
    ]
}
```


9.16 Bag recording interface

You can directly use the launch file to start recording ROS2 topics and store them in the default path (/home/ubuntu/bags). The recording is segmented into 100M bags. The maximum storage capacity of the recording directory is 2G. Once the limit is exceeded, the earliest folder and bag files will be automatically deleted. This applies to all models of the Walker Tienkung series.

```Bash
#Run the recording for the specified topic. The recording list is
stored in
/home/ubuntu/ros2ws/install/utils/lib/utils/topics.config #By
default, the list stores the key topics controlled by the
ontology.
ros2 launch utils record_config_topic.py
#Record full topic packages
ros2 launch utils record_all_topic.py
```

9.17 Radar

Method for replacing the radar head:

If it is a radar head or you need to replace it with a radar head, you need to modify the configuration parameters in the device's software. The modification method is as follows:

Configuration files that need to be changed:

/home/ubuntu/ros2ws/install/body_control/share/body_control/param/config_ethercat.yaml

```YAML
auto_init: true
enable_check_ready: true
net_card_name: "enp4s0"
enable_arms: true
enable_legs: true
enable_imu: true
enable_hands: true
enable_6dof: true
enable_head: false                #  Set this to false
enable_waist: true
enable_waist_home: true
enable_head_home: false
enable_power: false
selfMotor_idle_current: 0.3
# 0 - standard, 1 - extended, 2 - time_sharing_mixed
slave_mode:
- 0     # left_leg
- 0     # right_leg
- 0     # xsens_imu
```

Method to start the radar - ROS2:

Walker Tienkung - Voice & Vision： Orin（192.168.41.2）

Walker Tienkung -Embodied intelligence： Orin2（192.168.41.3）

```Bash
cd livox_ws
source install/setup.bash
ros2 launch livox_ros_driver2 msg_MID360_launch.py
```

1. Radar data

Description: Obtain raw radar data.

Control mode：topic

Topic name：`/livox/lidar`

Type： `sensor_msgs::msg::PointCloud2`

Data format：

C++

```
# This message holds a collection of N-dimensional points, which
may
# contain additional information such as normals, intensity, etc.
The
# point data is stored as a binary blob, its layout described by
the
# contents of t he "fields" array.
#
# The point cloud data may be organized 2d (image-like) or 1d
(unordered).
# Point clouds organized as 2d images may be produced by camera
depth
sensors
# such as stereo or time-of-flight.
# Time of sensor data acquisition, and the coordinate frame ID
(for 3d points).
std_msgs/Header header
    builtin_interfaces/Time stamp
        int32 sec
        uint32 nanosec
    string frame_id
# 2D structure of the point cloud. If the cloud is unordered,
height is
# 1 and width is the length of the point cloud.
uint32 height
uint32 width
# Describes the channels and their layout in the binary data blob.
PointField[] fields
    uint8 INT8       = 1
    uint8 UINT8     = 2
    uint8 INT16     = 3
    uint8 UINT16    = 4
    uint8 INT32     = 5
    uint8 UINT32    = 6
    uint8 FLOAT32 = 7
    uint8 FLOAT64 = 8
    string name          #
    uint32 offset       #
    uint8 datatype   #
    uint32 count        #

bool is_bigendian # Is this data bigendian?
uint32 point_step     # Length of a point in bytes
```

```
uint32 row_step        # Length of a row in bytes
uint8[] data                # Actual point data, size is
(row_step*height)
bool is_dense              # True if there are no invalid points
```

```Python
# Point cloud data doesn't seem very intuitive. You can use the
following code to subscribe to the topic and save the point cloud
data for viewing.
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import PointCloud2
import sensor_msgs_py.point_cloud2 as pc2

import numpy as np
import os
import time
import glob
import matplotlib.pyplot as plt

class PointCloudXZPlotter(Node):
    def __init__(self):
        super().__init__('livox_pointcloud_xz_plotter')

        self.save_dir = 'pointcloud_imgs'
        self.prepare_directory()

        self.subscription = self.create_subscription(
            PointCloud2,
            '/livox/lidar',
            self.pointcloud_callback,
            10
        )

        self.image_count = 0
        self.max_images = 100
        self.last_save_time = time.time()

    def prepare_directory(self):
        if not os.path.exists(self.save_dir):
            os.makedirs(self.save_dir)
        else:
            png_files = glob.glob(os.path.join(self.save_dir,
```

```
'*.png'))
            for f in png_files:
                os.remove(f)
            self.get_logger().info(f'Clear the old pictures and
prepare to save them to the directory.: {self.save_dir}')

    def pointcloud_callback(self, msg):
        now = time.time()
        if now - self.last_save_time < 0.2:  # Save every 0.2
seconds.
            return
        self.last_save_time = now

        if self.image_count >= self.max_images:
            self.get_logger().info('100 pictures have been saved.
Exiting the program.')
            rclpy.shutdown()
            return

        points = np.array([
            [x, y, z]
            for x, y, z in pc2.read_points(msg, field_names=("x",
"y", "z"), skip_nans=True)
        ])

        if len(points) == 0:
            self.get_logger().warn('The received point cloud is
empty. Skip this save.')
            return

        # Extract X and Z, and use Z as the color map.
        x = points[:, 0]
        z = points[:, 2]
        c = z  # The Z value is used for color mapping.

        # Draw an X-Z diagram
        plt.figure(figsize=(8, 6))
        scatter = plt.scatter(x, z, c=c, cmap='viridis', s=0.5)
#Viridis is a blue-green-yellow gradient.
        plt.colorbar(scatter, label='Z height [m]')
        plt.title(f'PointCloud X-Z View #{self.image_count}')
        plt.xlabel('X [m]')
        plt.ylabel('Z [m]')
        plt.axis('equal')
```

```
        plt.grid(True)

        # Save the image.
        image_path = os.path.join(self.save_dir,
f'pointcloud_{self.image_count}.png')
        plt.savefig(image_path, dpi=150)
        plt.close()
        self.get_logger().info(f'Save the image: {image_path}')
        self.image_count += 1

def main(args=None):
    rclpy.init(args=args)
    node = PointCloudXZPlotter()
    rclpy.spin(node)
    node.destroy_node()
    rclpy.shutdown()

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

1. Radar IMU data

- Description: Obtain the raw IMU data of the radar.
- Control method ：topic
- Topic name：/livox/imu
- Type： sensor_msgs::msg::Imu
- Data format：

```
Python
# This is a message to hold data from an IMU (Inertial Measurement
Unit) #
# Accelerations should be in m/s^2 (not in g's), and rotational
velocity should be in rad/sec
#
# If the covariance of the measurement is known, it should be
filled in (if all you know is the
# variance of each measurement, e.g. from the datasheet, just put
those along the diagonal)
# A covariance matrix of all zeros will be interpreted as
"covariance unknown", and to use the
# data a covariance will have to be assumed or gotten from some
other source #
```

```
# If you have no estimate for one of the data elements (e.g. your
IMU doesn't produce an
# orientation estimate), please set element 0 of the associated
covariance matrix to -1
# If you are interpreting this message, please check for a value
of -1 in the first element of each
# covariance matrix, and disregard the associated estimate.
std_msgs/Header header
    builtin_interfaces/Time stamp
        int32 sec
        uint32 nanosec
    string frame_id

geometry_msgs/Quaternion orientation
    float64 x 0
    float64 y 0
    float64 z 0
    float64 w 1
float64[9] orientation_covariance # Row major about x, y, z axes
geometry_msgs/Vector3 angular_velocity
    float64 x
    float64 y
    float64 z
float64[9] angular_velocity_covariance # Row major about x, y, z
axes

geometry_msgs/Vector3 linear_acceleration
    float64 x
    float64 y
    float64 z
float64[9] linear_acceleration_covariance # Row major x, y z
```

9.18 Reinforcement Learning Control Interface

Please ensure that both the ontology driver and the reinforcement learning motion control have been started. When the E key on the joystick is toggled upwards, the state can be switched through the following ROS service, and speed commands can be sent via the topic. The buttons and joysticks on the remote control are ineffective (only the C button is effective). When the E key is centered, the functions of the remote control are restored.

9.18.1 Motion control mode switching

- **Interface Name：** `/hric/motion/set_motion_mode`

- **Type：** ROS service
- **Description：** Request to switch to sports mode.
- **Release Module：** Motion control
- **Client module：** Robot management
- **Message type：** hric_msgs/SetMotionMode.srv

Service msg	Parameter Name	Parameter Type	Required (Yes/No)	Remarks
Request Fields	walk_mode_request	uint8	yes	#Request to switch the motion mode. 1: stop 2: zero 3: stand 4: walk 5: run
	is_need_swing_arm	bool	yes	Is it necessary to swing the arms when walking? true:Need false:No need. (The current version of RL does not support setting this function yet, so you can set it to either true or false.)
Response Fields	sucess	bool	yes	Indicates whether the service call is successful.
	error_code	uint32	yes	Error code NONE =0 UNKNOWN =400

- **Release Example：**

```bash
cd ros2ws/
source install/setup.bash
```

```
# Stiff stop:
ros2 service call /hric/motion/set_motion_mode
hric_msgs/srv/SetMotionMode "{walk_mode_request: 1,
is_need_swing_arm: false}"

# Return to zero:
ros2 service call /hric/motion/set_motion_mode
hric_msgs/srv/SetMotionMode "{walk_mode_request: 2,
is_need_swing_arm: false}"

# Stand:
ros2 service call /hric/motion/set_motion_mode
hric_msgs/srv/SetMotionMode "{walk_mode_request: 3,
is_need_swing_arm: false}"

# Tread on the spot (If the robot is in the zero-return state and
placed on the ground, directly calling this interface can make it
enter the state of treading on the spot from the zero-return
state. However, this is quite dangerous, so don't do this. First
enter the standing state and then enter the state of treading on
the spot.):
ros2 service call /hric/motion/set_motion_mode
hric_msgs/srv/SetMotionMode "{walk_mode_request: 4,
is_need_swing_arm: false}"

# Jog in place:
ros2 service call /hric/motion/set_motion_mode
hric_msgs/srv/SetMotionMode "{walk_mode_request: 5,
is_need_swing_arm: false}"
```

9.18.2 Action number calling interface

- **Interface Name：** /hric/motion/set_motion_number
- **Type：** ROS service
- **Description：** Request to call an action
- **Release Module：** Motion control
- **Subscription Module：** Robot management
- **Message type：** h ric_msgs/SetMotionNumber.srv

Service msg	Parameter Name	Parameter Type	Required or Not	Remarks
Request fields	is_motion	bool	Yes	It is only valid in the standing + E upshift state. Setting it to true triggers the action immediately. Setting it to false does not interrupt the action. The action is triggered only once when the state is true.
	motion_number	int32	Yes	1 to 5, where 1 represents waving.
Reply field	sucess	bool	Yes	Indicates whether the service call is successful.

- **Release Example：**

```
Bash
cd ros2ws
source install/setup.bash

ros2 service call /hric/motion/set_motion_number
hric_msgs/srv/SetMotionNumber "{is_motion: true, motion_number:
1}"

ros2 service call /hric/motion/set_motion_number
hric_msgs/srv/SetMotionNumber "{is_motion: true, motion_number:
2}"

ros2 service call /hric/motion/set_motion_number
hric_msgs/srv/SetMotionNumber "{is_motion: true, motion_number:
3}"

ros2 service call /hric/motion/set_motion_number
hric_msgs/srv/SetMotionNumber "{is_motion: true, motion_number:
4}"

ros2 service call /hric/motion/set_motion_number
hric_msgs/srv/SetMotionNumber "{is_motion: true, motion_number:
5}"
```

9.18.3 Vector Velocity Control Interface

- **Interface Name** ：/hric/robot/cmd_vel
- **Type**：topic
- **Description** ：Issue speed commands to control the movement of the robot.
- **Release Module** ：Remote control
- **Subscription Module**：Robot management
- **Message type** ：geometry_msgs/TwistStamped.msg

```
Bash
geometry_msgs/msg/TwistStamped #The structure is as follows.:
std_msgs/Header header
geometry_msgs/Twist twist

geometry_msgs/Twist #The structure is as follows.:
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular

linear #It represents the linear velocity (unit: m/s). x
represents forward and backward movement, y represents left and
right translation, and z represents up and down movement.

angular #$$\omega$$ represents the angular velocity (unit: rad/s),
and z represents the rotation around the z-axis (left turn, right
turn).。
```

- **Message frequency:** 20 Hz
- **Release example:**

```
Bash
cd ros2ws
source install/setup.bash

# The velocity in the x-direction is 0.15, which means moving
forward at a velocity of 0.15.
ros2 topic pub /hric/robot/cmd_vel geometry_msgs/msg/TwistStamped
"{header: {frame_id: 'base_link' }, twist: {linear: {x: 0.15, y:
0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}}"
```

```
# The velocity in the x-direction is 0.20, which means moving
forward at a speed of 0.20.
ros2 topic pub /hric/robot/cmd_vel geometry_msgs/msg/TwistStamped
"{header: {frame_id: 'base_link' }, twist: {linear: {x: 0.20, y:
0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}}"

# All linear velocities and angular velocities are 0.0, which
means staying in place.
ros2 topic pub /hric/robot/cmd_vel geometry_msgs/msg/TwistStamped
"{header: {frame_id: 'base_link' }, twist: {linear: {x: 0.0, y:
0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}}"

# Turn left at an angular velocity of 0.1.
ros2 topic pub /hric/robot/cmd_vel geometry_msgs/msg/TwistStamped
"{header: {frame_id: 'base_link' }, twist: {linear: {x: 0.0, y:
0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.1}}}"

# Turn right at an angular velocity of 0.1.
ros2 topic pub /hric/robot/cmd_vel geometry_msgs/msg/TwistStamped
"{header: {frame_id: 'base_link' }, twist: {linear: {x: 0.0, y:
0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: -0.1}}}"

# Finally, to make the robot enter the state of standing still
again, the interface needs to be called once more, and both the
linear velocity and angular velocity should be set to 0.0.
```

9.18.4 Robot Lower Limb Motion State Publication

- **Interface Name** ：/hric/motion/status
- **Type** ：topic
- **Description**：Publish the status and speed of the motion control module.
- **Release Module** ：Motion control
- **Publication frequency** ：20hz
- **Subscription Module**：Robot management, positioning, and navigation
- **Message type**： hric_msgs/MotionStatus.msg

Topic msgs	Parameter Name	Parameter Type	Required (Yes/	Remarks

			No)	
	header	Header	Yes	Timestamp and coordinate system
	velocity	geometry_msgs/Tw ist.msg	Yes	Velocity of the robot's waist coordinate system
	walk_mode	uint8	Yes	Walking state 0: stop, 1: zero, 3: stand, 5: walk, 7: run
	is_console_control	bool	Yes	Whether it is controlled by the remote control true: Navigation control false: Remote control operation
	is_swing_arm	bool	Yes	Whether it is in the walk arm-swinging state.
	error_code	uint32	Yes	Error code NONE = 0 UNKNOWN = 400

- **Message format example**:

```
JSON
{
  "header": {
    "seq": 1,
    "stamp": {
      "secs": 1622548800,
      "nsecs": 0
    },
    "frame_id": ""
```

```
  },
  "velocity": {
    "linear": {
      "x": 1.0,
      "y": 0.0,
      "z": 0.0
    },
    "angular": {
      "x": 0.0,
      "y": 0.0,
      "z": 0.0
    }
  },
  "walk_mode": 1,
  "error_code": 0,
}
```