

AGIQUAD D1 Edu & Ultra Quadruped Robot

SDK Development Guide



Version: V0.2.7
Date: July 2026

Contents

1. Product Introduction.....	5
1.1 Product Overview.....	5
1.2 Product Checklist.....	5
1.3 Product Specifications.....	5
1.4 Rear-Mount Payload Installation.....	8
1.5 Electrical Expansion Interfaces.....	8
2. System Architecture.....	11
2.1 AGIBOT System Architecture.....	11
2.2 SDK Control Block Diagram.....	12
2.3 SDK Interface Block Diagram.....	12
2.4 Software Service Interface List.....	13
3. SDK Development Guide.....	14
3.1 SDK Introduction.....	14
3.2 Environment Dependencies.....	14
3.3 Network Connection.....	14
3.4 Device Login.....	15
3.5 Firmware Version Query.....	17
3.6 Obtaining the SDK.....	18
3.7 SDK File Configuration.....	18
3.7.1 Wireless AP Connection Mode.....	18
3.7.2 Wired or External Wi-Fi Connection Mode.....	18
3.8 Modifying SDK Demo Configuration.....	22
3.8.1 Configuring Python IP and Port.....	22
3.8.2 Configuring C++ IP and Port.....	22
3.9 Demo Execution Steps.....	22
3.9.1 Compiling the Demo.....	22
3.9.2 Running the SDK Demo.....	23
3.9.3 Calling the SDK.....	23
3.10 Secondary Development Control Instructions.....	26
3.10.1 Joint Control Command Instructions.....	27
3.10.2 Joint Direction Definitions.....	27
4. API Function Introduction.....	28
4.1 High-Level Functions Introduction.....	28
4.1.1 Initialization (initRobot).....	28
4.1.2 Check Connection Status (checkConnection).....	28
4.1.3 Stand Up (standUp).....	28
4.1.4 Lie Down (lieDown).....	29
4.1.5 Damping (passive).....	29
4.1.6 Move (move).....	29
4.1.7 In-Place Jumping (jump).....	29
4.1.8 Forward Jumping (frontJump).....	30
4.1.9 Backflip (backflip).....	30
4.1.10 Greet (shakeHand).....	30
4.1.11 Two-Leg Stand (twoLegStand).....	31
4.1.12 Cancel Two-Leg Stand (cancelTwoLegStand).....	31
4.1.13 Attitude Control (attitudeControl).....	31

4.1.14	Get Attitude Quaternion (getQuaternion)	32
4.1.15	Get Attitude Euler Angles (getRPY)	32
4.1.16	Get Body Acceleration (getBodyAcc)	32
4.1.17	Get Angular Velocity (getBodyGyro)	32
4.1.18	Get Position in Origin Coordinate Frame (getPosition)	32
4.1.19	Get Velocity in Origin Coordinate Frame (getWorldVelocity)	32
4.1.20	Get Body Velocity (getBodyVelocity)	33
4.1.21	Get Battery Level (getBatteryPower)	33
4.1.22	Get Control Mode (getCurrentCtrlMode)	33
4.1.23	Get Abad Joint Angles (getLegAbadJoint)	33
4.1.24	Get Hip Joint Angles (getLegHipJoint)	33
4.1.25	Get Knee Joint Angles (getLegKneeJoint)	34
4.1.26	Get Abad Joint Angular Velocities (getLegAbadJointVel)	34
4.1.27	Get Hip Joint Angular Velocities (getLegHipJointVel)	35
4.1.28	Get Knee Joint Angular Velocities (getLegKneeJointVel)	35
4.1.29	Get Abad Joint Torques (getLegAbadJointTorque)	35
4.1.30	Get Hip Joint Torques (getLegHipJointTorque)	35
4.1.31	Get Knee Joint Torques (getLegKneeJointTorque)	36
4.2	Low-Level Function Introduction	36
4.2.1	Create Connection (initRobot)	36
4.2.2	Check Connection Status (checkConnection)	36
4.2.3	Send Motor Command (sendMotorCmd)	36
4.2.4	Get Attitude Quaternion (getQuaternion)	37
4.2.5	Get Attitude Euler Angles (getRPY)	38
4.2.6	Get Body Acceleration (getBodyAcc)	38
4.2.7	Get Angular Velocity (getBodyGyro)	38
4.2.8	Get Motor Data (getMotorState)	38
4.2.9	Check Motor Data Availability (haveMotorData)	39
5.	Debugging Guide	40
5.1	Video Stream Push Service	40
5.2	Indicator Light Description	41
5.2.1	Quadruped Robot Tail Light Description	41
5.2.2	Battery Indicator Light Description	42
5.3	Log Query	42
5.4	Control Authority Switching Function	42
5.4.1	Version Requirements	42
5.4.2	Function Description	43
5.4.3	Control Block Diagram	44
5.5	Modifying the Wired Ethernet Port IP Address	44
5.6	Running the SDK on the Robot Body	45
5.7	Running the SDK on an External LAN	47
5.7.1	Control PC Configuration	47
5.7.2	Quadruped Robot Configuration	47
5.7.3	Running the SDK	49
6.	FAQ	51
6.1	Connect Timeout	51
6.2	Wired Connection IP Binding Failure	51
6.3	Using wlan0 to Connect to an External Wi-Fi Network; Device Does	

Not Auto-Connect After Reboot.....	52
6.4: Python Import Failure.....	52
6.5: C++ Linking Failure?.....	53
6.6 What is the Supported Frequency for the Device Status Retrieval Interface in the Motion Control System?.....	53
6.7: What is the Supported Frequency for the SDK Control Interface?....	53
6.8: Is the SDK Program Thread-Safe?.....	53
6.9: Does the SDK Support Communication Monitoring?.....	53
6.10: When Starting the Motion Control Program via Service on the Robot, How Can I Ensure the Robot's Own Programs Are Started?.....	53
6.11: Can Gamepad Remote Control and SDK Control Be Switched Simultaneously?.....	53
6.12: Is the Retrieved IMU Data Raw Data?.....	54
6.13: The Retrieved IMU Data Seems to Have Errors?.....	54
6.14: After Using SDK Control, When Using the Gamepad Remote Controller to Send a Stand Command, Why Does the Device Perform "Push-Ups" In-Place?.....	54
6.15: What is the Rotation Order When Converting Quaternion to Euler Angles?.....	54
6.16: Can High-Level and Low-Level Be Used Simultaneously?.....	54
6.17: Why Can't the SDK Control the Device After the Motion Control Program Is Updated?.....	54
6.18: Is There a ROS Interface?.....	54
6.19: When Using the SDK, the Program Stalls at the initRobot Function and Does Not Proceed?.....	55
6.20 Does It Support Deploying Control Logic Directly on the Quadruped Robot Body?.....	55

Document Revision History

Revision Date	Revision Description	SDK Version
Jul 16, 2025	● Initial SDK release	v2.0.5
Jul 19, 2025	● Added high-level SDK interfaces	v2.0.6
Sep 15, 2025	● Fixed abnormal status issue with the "CheckConnect" interface	v0.2.6-patch1
Nov 4, 2025	● Added low-level SDK interfaces ● Added high-level SDK interfaces for D1 Ultra-W	v2.0.7
May 18, 2026	● Added Sections 5.5, 5.6, and 5.7	v2.0.7
Jul 2026	● Errata/corrections	v2.0.7

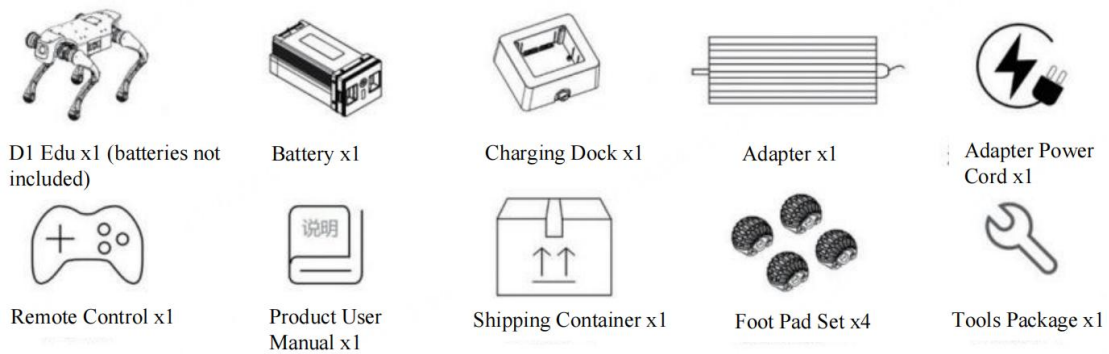
1. Product Introduction

1.1 Product Overview

The AGIBOT D1 Edu & Ultra quadruped robot features 12 degrees of freedom and supports a wide variety of locomotion gaits and stunt maneuvers. The product also provides an open motion-control algorithm development SDK and communication protocol, enabling users to perform secondary development as needed.

Note: The secondary development guide applies to both the D1 Edu and D1 Ultra, with motion-control firmware version 0.2.6 or higher.

1.2 Product Checklist



1.3 Product Specifications

Category	Specification	Description
Basic Information	Material	Aluminum alloy + high-strength engineering plastic (phenolic resin) D1 Edu: PANTONE Warm Gray 5U D1 Ultra: PANTONE Warm Gray 3U
	Standing Dimensions (L * W * H)	D1 Edu: 635mm * 360mm * 420mm D1 Ultra: 630mm * 360mm * 420mm
	Prone Dimensions (L * W * H)	D1 Edu: 675mm * 435mm * 145mm D1 Ultra: 670mm * 435mm * 145mm
	Total Weight (incl. battery)	D1 Edu: 15.5kg D1 Ultra: 16kg

	Operating Temperature	0° C ~ 40° C
	Ingress Protection Rating	IP54
	Charging Time	Approx. 1 hour (10% - 90% charge)
	Battery Life	1 - 2 hours
	Operating Range	6 ± 1km
Performance Parameters	Base Compute	8-core high-performance CPU
	Maximum Operating Power	3500W
	Maximum Speed	D1 Edu: 3.5m/s D1 Ultra: 3.7m/s
	Rated Payload	5kg
	Continuous Stair-Climbing Height	16cm
	Maximum Climbing Angle	30° under standard conditions, 40° under extreme conditions
	Jump Clearance Height	35cm
Joint Parameters	Actuator Type	High power-density motion actuator
	Maximum Joint Torque	48N • m
	Joint Range of Motion (Body)	- 28° ~ 28°
	Joint Range of Motion (Thigh)	- 170° ~ 66°
	Joint Range of Motion (Calf)	35° ~ 156°

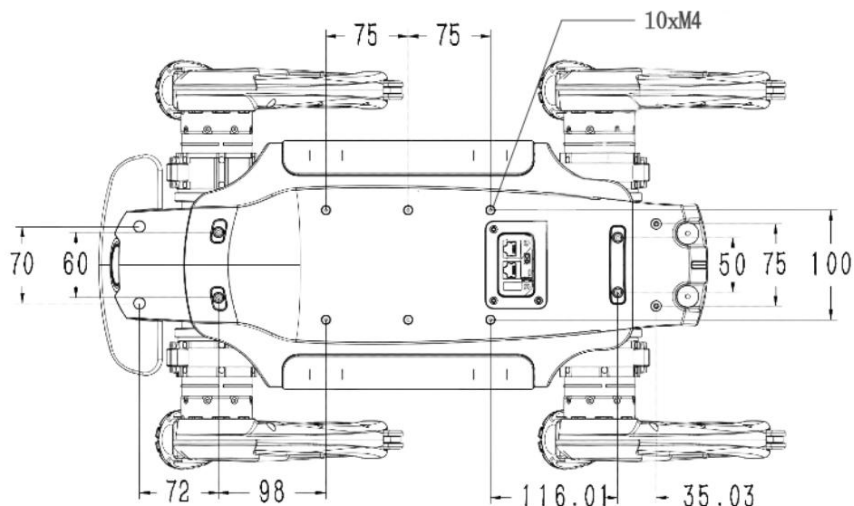
Sensors	Camera FOV	DFOV 122° , HFOV 122° , VFOV 70°
	Inertial Measurement Unit (IMU)	Standard
Function List	Standard Actions	Stand, prone, damping mode, pitch adjustment, height adjustment, in-place turn
	Stunt Maneuvers	Forward jump, backward jump, two-legged stand, backflip, waving, etc.
	Real-Time Image Transmission	Supported
	Over-The-Air (OTA) Updates	Supported
	Secondary Development	Supported
	Functional Expansion Ports	Ethernet port, USB port, power ports, SBUS interface, UART interface
Accessories	Power Adapter	Standard
	Charging Dock	Standard
	Handheld Remote Controller	Standard for D1 Edu
	Remote Controller with Integrated Display	Standard for D1 Ultra
Other	Warranty Period	1 year

Charging time data is measured at a standard ambient temperature of 25° C.
 Operating range data is measured at a speed of 1m/s under no-load conditions.

The parameters provided are based on laboratory test data. Actual performance may vary depending on the operating environment and usage

methods. For precise specifications, refer to the actual product.

1.4 Rear-Mount Payload Installation



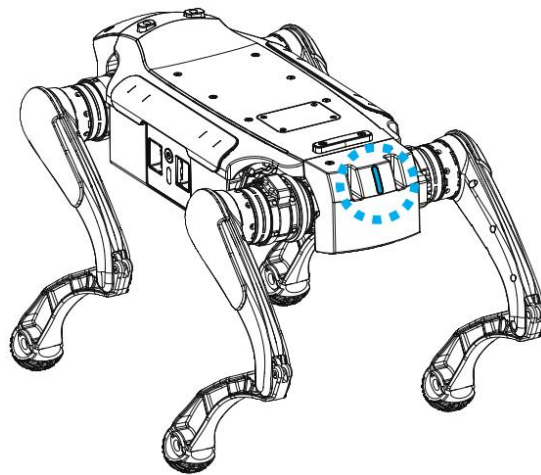
Note: The mounting holes for rear-mounted payloads use M4 threads. Before installation, remove the dust plugs from the threaded holes.

1.5 Electrical Expansion Interfaces

No.	Interface Name	Quantity	Description	Remarks
1	Ethernet Port	1	RJ45 interface, supports 1000Mbps transmission rate for high-speed data transfer; supports network connectivity, remote monitoring, data transmission, and other applications	
2	USB Port	2	Type-A interface, for connecting external storage devices, cameras, sensors, and other USB devices;	

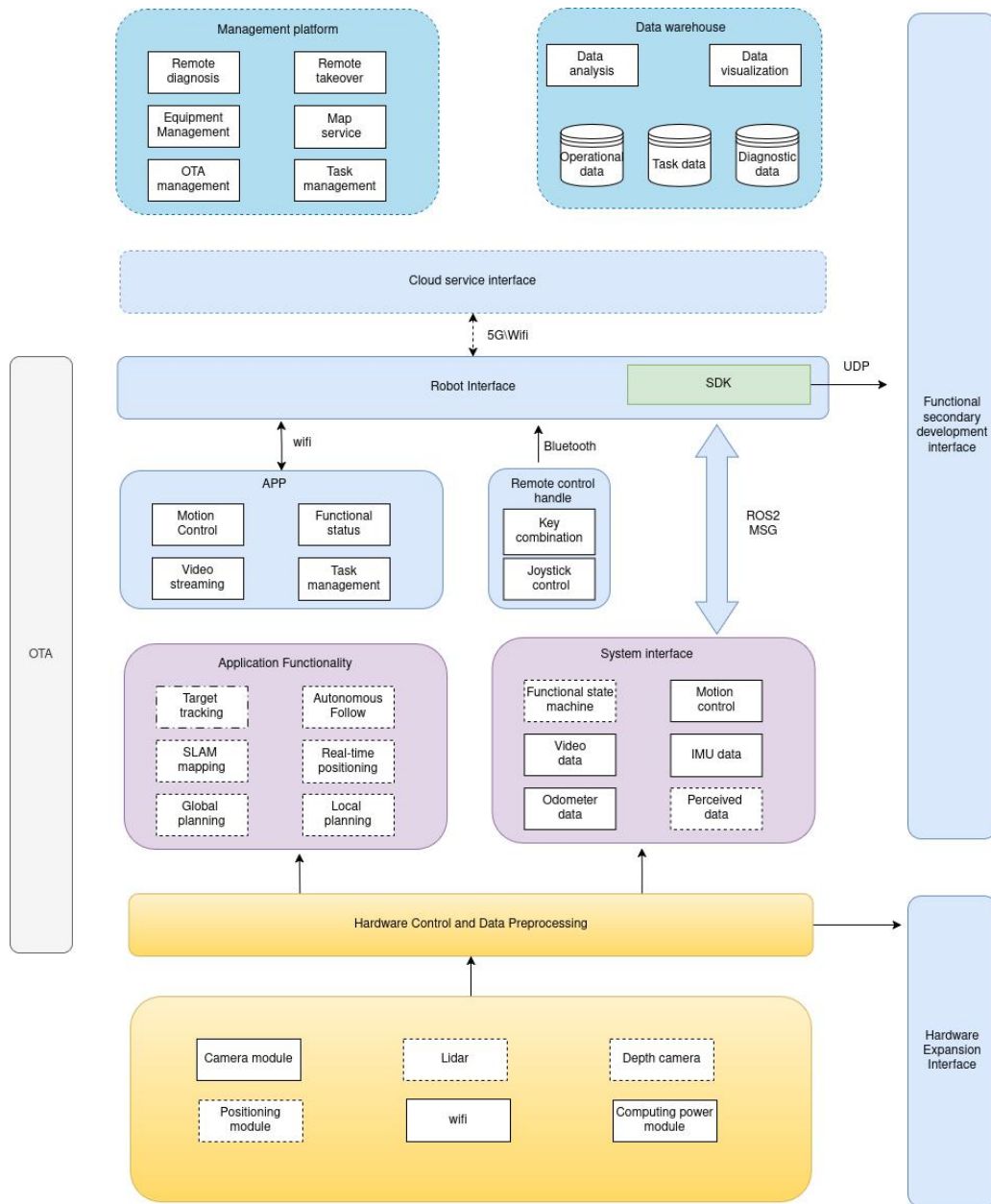
			provides high-speed data transfer and power supply	
3	Power Port	2	Standard DC power interface, provides stable power supply with 12V and BAT outputs. Total output power for 12V shall not exceed 36W; total output power for BAT shall not exceed 100W, satisfying the power requirements of the robot and top-mounted payloads. Mating connector model: XT30U-M	
4	SBUS Port	1	Standard SBUS interface, for connecting a remote controller receiver or other communication devices supporting the SBUS protocol, enabling precise remote command transmission	
5	UART Port	1	Standard UART interface, for connecting embedded systems, communication devices, or other devices supporting the UART protocol, enabling data exchange and control between	

devices via
asynchronous
serial
communication



2. System Architecture

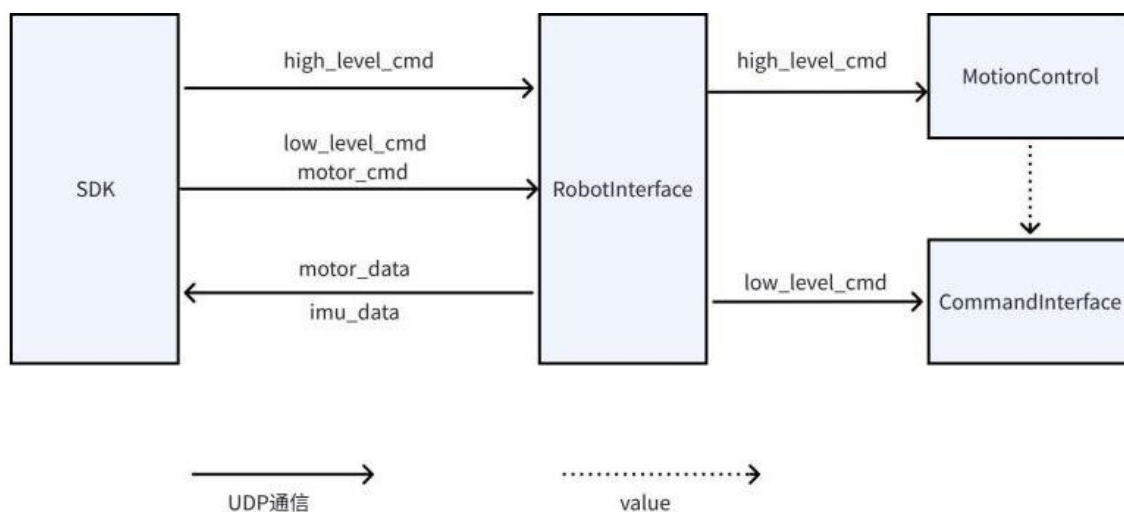
2.1 AGIBOT System Architecture



Note: The solid line diagram represents the functionality that has been implemented
The dashed diagram represents functionality that can be achieved through secondary development

Note: Functions shown in solid-line boxes are already implemented; functions shown in dashed-line boxes can be implemented via secondary development.

2.2 SDK Control Block Diagram

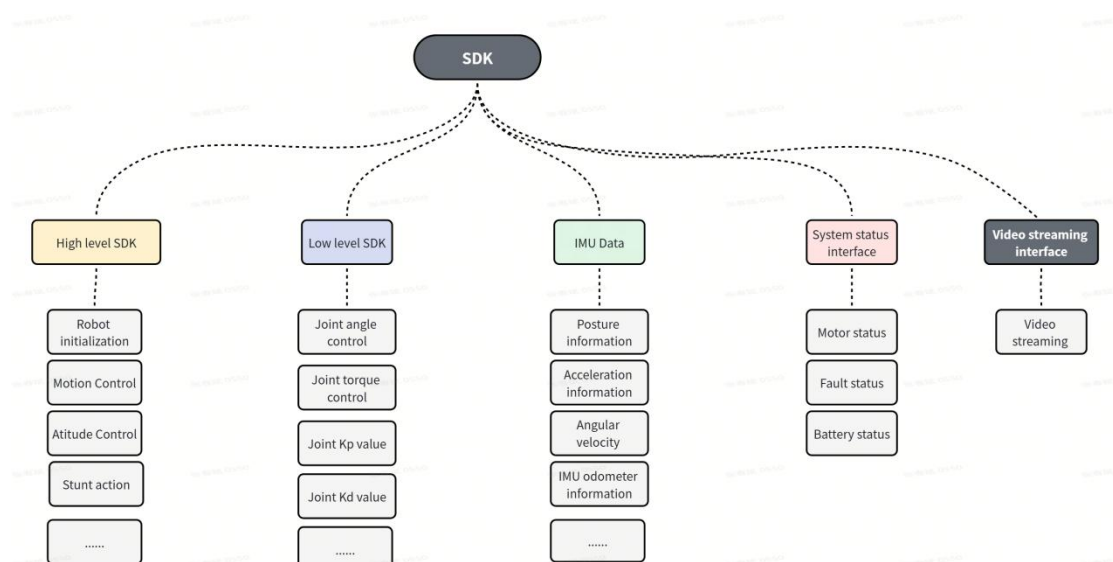


High-level command set interfaces operate through the robot's body motion control system, while low-level motor control interfaces act directly on the joint motors.

Therefore, high-level command set interfaces and low-level motor control interfaces cannot run simultaneously.

2.3 SDK Interface Block Diagram

The currently available motion control SDK interfaces include high-level command set interfaces, low-level motor control interfaces, IMU inertial navigation data interfaces, motor status data interfaces, and video stream data interfaces.



2.4 Software Service Interface List

No.	Interface Name	Function	SDK Version
1	High-Level Command Set Interface (App Function Service Interface)	Implements high-level command set of the robot, including forward/backward and turning speed control, prone posture, rotation, and standing on all fours	V0.2.6
2	Low-Level Motor Control Interface (Motor Control Service Interface)	Controls the motor low-level interfaces to set target motion states, including position, velocity, torque, etc. By configuring different parameters, users can command motors to perform specific actions in various modes	V0.2.6
3	Device State Service Interface	Queries and retrieves current device status and system resource usage information	V0.2.6
4	Inertial Navigation Status Data Interface (IMU Service Interface)	Retrieves raw IMU data	V0.2.6
5	Video Stream Push Data Interface	Streams the robot body camera's video feed to the client	V0.2.6
6	Control Authority Switching SDK Interface	Provides control authority switching functionality between the remote controller and the SDK	v2.0.7

3. SDK Development Guide

3.1 SDK Introduction

- This document describes the SDK for the D1 Edu & Ultra, and provides detailed information on its core service interfaces, including high-level command set interfaces, low-level motor control interfaces, IMU inertial navigation data interfaces, and motor status data interfaces.
- Each chapter describes the invocation methods for different service interfaces, the standard format definitions for data input/output, and provides functional development examples.
- Interface definitions are subject to change based on production development status. Please refer to the latest version of the interface definition document.
- The SDK communicates with the robot body software via a specific protocol. Therefore, different SDK versions may have compatibility differences with the robot body software. Therefore, the SDK version that corresponds to the device version should be selected. If conditions permit, it is recommended to update the robot body software to the latest version for better compatibility.
- When controlling via the SDK interfaces, all states are designed and specified by the secondary developer, while the robot body motion controller does not perform additional intervention or state transitions.
- When running the SDK, ensure that the system has sufficient resources; otherwise, the motion control module may fail.

3.2 Environment Dependencies

- Ubuntu22.04
- CMake 3.8+
- GCC 11+
- Eigen3
- Boost 1.74
- Python3.10

It is recommended to use a physical machine with Ubuntu 22.04 for secondary development. Secondary development cannot be performed using WSL virtual machines on Windows systems.

3.3 Network Connection

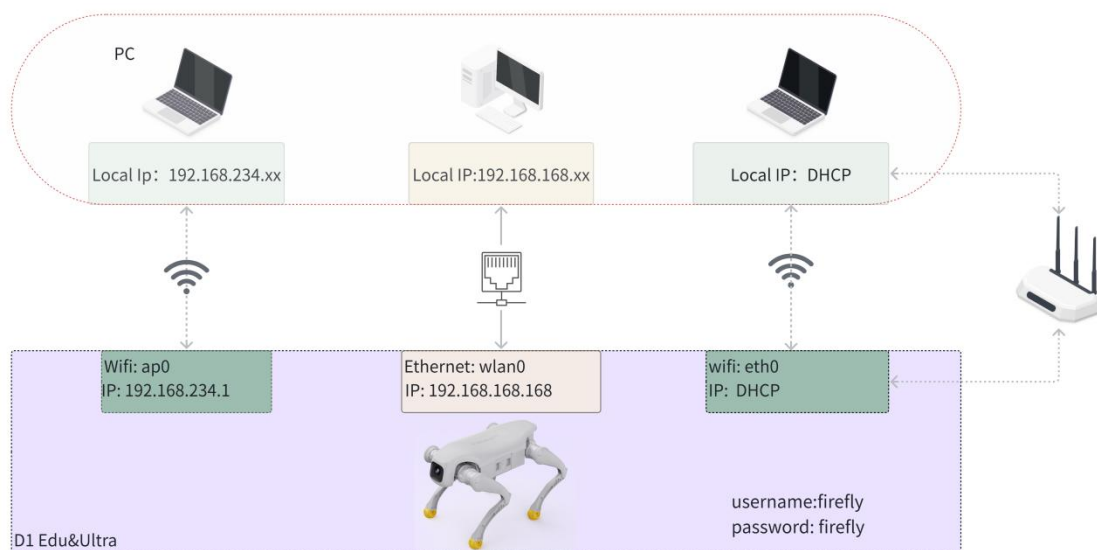
The device is equipped with both wireless and wired network interfaces. Wireless network information is located on a label on the right side or

underside of the device, which contains the SSID and password. The robot's body default IP addresses are as follows:

Network Interface	IP Address	DHCP
Wireless Network	192.168.234.1	Yes
Wired Network	192.168.168.168	None

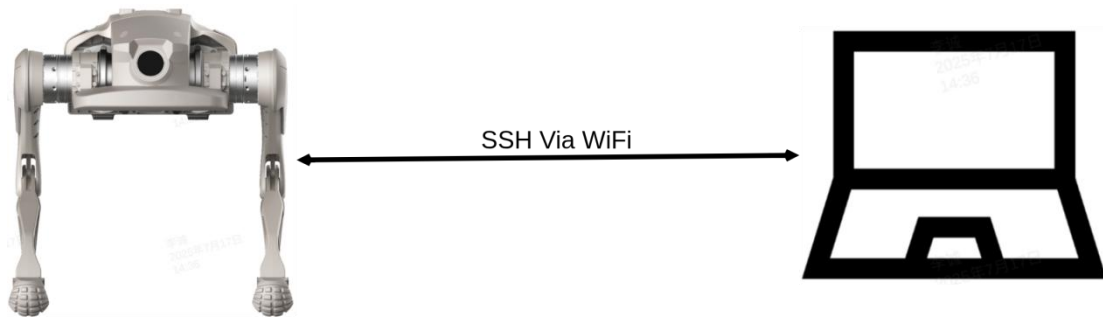
The wireless network provides a DHCP service. After connecting to the wireless network, ensure that the operating device is not configured with a static IP. Communication with the robot can then be established directly via 192.168.234.1.

The wired network does not provide a DHCP service. After establishing a wired connection, configure a static IP address for the operating device's wired network. The IP address must be in the 168 subnet to establish communication with the robot.



3.4 Device Login

The device supports SSH login. After establishing a connection via wireless or wired network, the device can be accessed via SSH.



- Wireless Connection

```
ssh firefly@192.168.234.1 #password:firefly
```

- Wired Connection

```
ssh firefly@192.168.168.168 #password:firefly
```

3.5 Firmware Version Query

After logging into the device, execute the following command to check the current device version:

```
grep -oP 'motion-control_\K[^\_]+' /etc/release/*[^rootfs]*.yaml
```



```
firefly@firefly:~$ grep -oP 'motion-control_\K[^\_]+' /etc/release/*[^rootfs]*.yaml
/etc/release/62200265BQBA.yaml:0.3.9-xgw
firefly@firefly:~$ |
```

Example output: 62200265BQBA.yaml:0.3.9-xgw

Firmware Version: Bytes 5 - 6 - 7, which is: 0.2.6

Motion Control Version: The output string, which is: 0.3.9-xgw

If, after running this command, the motion control version is lower than 0.2.6, the device firmware is outdated and requires an update. Please contact after-sales support.

This document applies to motion control version 0.2.6 and later. If the version is lower, please contact after-sales technical support for the latest firmware update package

Robots with versions below 0.3.4: Remote controller cannot be used during SDK control

Robots with version 0.3.4 and later: Control authority switching is supported, enabling toggling between remote controller and SDK control

3.6 Obtaining the SDK

Method 1: Contact after-sales technical support to obtain the SDK

Method 2: Obtain via the following GitHub repository

```
git clone https://github.com/AgibotTech/agibot_D1_Edu-Ultra.git
```

3.7 SDK File Configuration

3.7.1 Wireless AP Connection Mode

Step 1: Modify the SDK communication configuration file

Use the vim command to modify the `/opt/export/config/sdk_config.yaml` file on the robot body:

```
target_ip: "192.168.234.xx" #local_ip
target_port: 43988
```

`target_ip`: Robot controller IP address. Query it via ifconfig on the control PC

`target_port`: Default: 43988. Do not modify

Note: When controlling via the robot's Wi-Fi AP, no other configuration files need to be modified. However, ensure that `/opt/app_launch/start_motion_control.sh` and `/etc/systemd/system/robot-launch.service` files remain in their default states. Refer to [6.1 "Connect Timeout" for details](#)

Step 2: After modifying the configuration parameters, restart the quadruped robot for the changes to take effect

3.7.2 Wired or External Wi-Fi Connection Mode

Step 1: Modify the SDK communication configuration file

Use the vim command to modify the `/opt/export/config/sdk_config.yaml` file on the quadruped robot body:

Plain Text

```
target_ip: "192.168.168.xxx" #local_ip
target_port: 43988
```

`target_ip`: IP address of the control-side of the quadruped robot. Avoid IP conflicts (do not use 192.168.168.168)

`target_port`: Default: 43988. Do not modify

Step 2: Configure SDK_CLIENT_IP

If using a wired connection or an IP outside the 192.168.234.X subnet to control the device, use the `vim` command to modify the `/opt/app_launch/start_motion_control.sh` file:

```
Bash

#!/bin/bash

echo "start motion control"

# Shared memory file path
SHM_FILE="/dev/shm/spline_shm"

# Loop to check if the device exists
while true; do
    if [-e "$SHM_FILE"]; then
        echo "Shared memory file $SHM_FILE already exists."
        break
    else
        echo "Shared memory file $SHM_FILE does not exist, waiting 1
second before retry..."
        sleep 1
    fi
done

# Commands to execute after shared memory file is ready
echo "Shared memory file is ready, proceeding with subsequent
operations."

sudo ifconfig lo multicast
sudo route add -net 224.0.0.0 netmask 240.0.0.0 dev lo

export LD_LIBRARY_PATH=/opt/export/mc/bin
export ROBOT_TYPE=P2
export SDK_CLIENT_IP="IP address of the network interface connected to
the robot"

cd /opt/export/mc/bin && taskset -c 7 ./mc_ctrl r
```

If using a wired network connection to the quadruped robot's expansion Ethernet port, add the corresponding configuration at line 27 of the above code:

```
export SDK_CLIENT_IP="192.168.168.168"
```

For a wired connection with the default wired network interface IP:
`export SDK_CLIENT_IP="192.168.168.168"`

For a wired connection with a modified wired network interface IP: `export SDK_CLIENT_IP="<modified IP address of the wired interface wlan0>"`

For an external LAN connection via a wireless router: `export SDK_CLIENT_IP="<modified IP address of the wireless interface eth0>"`

Step 3: Modify the network interface name

Modify the `/etc/systemd/system/robot-launch.service` file on the quadruped robot body

Add the following line to the Service section:

`Environment="ROBOT_NET_INTERFACES=wlan0"`, where `wlan0` is the corresponding wireless network interface name. For wired network connections, this can be changed to `eth0`

Complete configuration file example:

```
JSON
robot-launch.service

[Unit]
Description=Run robot-launch at startup
After=network.target

[Service]
ExecStart=robot-launch server
ExecStop=/usr/local/bin/robot_launch_stop.sh
TimeoutStopSec=5s
Restart=always
User=root
Environment="ROBOT_HOME=/home/firefly"
Environment="ROBOT_LOG_DIR=/userdata/log"
Environment="ROBOT_NET_INTERFACES=wlan0" #For wired connections, change
wlan0 to eth0; for external Wi-Fi connections, select the corresponding
wireless interface name

[Install]
WantedBy=multi-user.target
```

For wired connections: **`Environment="ROBOT_NET_INTERFACES=wlan0"`**

For external LAN connections via a wireless router:

`Environment="ROBOT_NET_INTERFACES=eth0"`

After configuring `SDK_CLIENT_IP`, if the actual network connection status upon device boot does not match the network interface name set in `robot-launch.service`, the device's **blue LED will flash**, the corresponding network interface will not be assigned an IP address, and the motion control program will fail to bind to the IP address, rendering the remote controller unusable.

3.8 Modifying SDK Demo Configuration

3.8.1 Configuring Python IP and Port

Navigate to `{user directory}/agibot_D1_Edu-Ultra-main/demo/zsl-1/python/examples/` and modify as needed

Python

```
app.initRobot("192.168.234.xx", 43988, "192.168.234.1")    #local_ip,  
local_port, dog_ip
```

3.8.2 Configuring C++ IP and Port

Navigate to `{user directory}/agibot_D1_Edu-Ultra-main/demo/zsl-1/cpp/` and modify as needed

C++

```
constexpr int CLIENT_PORT = 43988;    // local port  
std::string CLIENT_IP = "192.168.234.xx"; // local IP address  
std::string DOG_IP = "192.168.234.1"; // dog ip
```

The robot IP and local port in the configuration parameters must match the `target_ip` and `target_port` parameters in [sdk_config](#).

Please ensure the robot IP and port number match; otherwise, communication cannot be established!

If the IP in `sdk_config.yaml` is deleted or commented out, the motion control service on the device will fail to start automatically!

When the device program or motion control program is updated, configuration files will be reset, and all related configurations must be reconfigured!

3.9 Demo Execution Steps

3.9.1 Compiling the Demo

On the local PC, navigate to the SDK directory

Bash

```
cd demo/zsl-1/cpp  
mkdir build  
cd build
```

```
cmake ..  
make -j6
```

3.9.2 Running the SDK Demo

Running Python Demo

Bash

```
cd demo/zsl-1/python/examples  
python highlevel_demo.py  
or  
python lowlevel_demo.py
```

Running C++ Demo

Bash

```
cd demo/zsl-1/cpp/build  
./highlevel_demo  
or  
./lowlevel_demo
```

3.9.3 Calling the SDK

- Calling the SDK via Python

Python

```
import os  
import platform  
import sys  
arch = platform.machine().replace(' amd64', ' x86_64').replace(' arm64',  
' aarch64')  
lib_path =  
os.path.abspath(f' {os.path.dirname(__file__)} /../.. /../lib/zsl-  
1/{arch}')  
sys.path.insert(0, lib_path)  
  
import mc_sdk_zsl_1_py  
app = mc_sdk_zsl_1_py.HighLevel()  
app.initRobot("{local PC IP in same subnet as  
robot}", 43988, "192.168.234.1")  
app.standUp()  
app.lieDown()  
...
```

- Using the SDK via CMake

```
CMake

cmake_minimum_required(VERSION 3.16)

project(test)

set(CMAKE_CXX_STANDARD_REQUIRED ON)
set(CMAKE_CXX_STANDARD 17)

# Define architecture variable (default unknown)
set(ARCH "unknown")

# Identify the x86_64 architecture (compatible with return values from
different systems)
if(CMAKE_HOST_SYSTEM_PROCESSOR MATCHES "x86_64|amd64")
    set(ARCH "x86_64")
# Identify arm64/aarch64 architecture
elseif(CMAKE_HOST_SYSTEM_PROCESSOR MATCHES "aarch64|arm64")
    set(ARCH "aarch64")
else()
    message(FATAL_ERROR "Unsupported architecture:
${CMAKE_HOST_SYSTEM_PROCESSOR}, Please check if the corresponding
architecture library exists in the lib directory")
endif()

message(STATUS "Detected system architecture: ${ARCH}")

# Library path: parent directory -> lib/zsl-1/architecture
set(LIB_PATH "${CMAKE_SOURCE_DIR}/../../lib/zsl-1/${ARCH}")
message(STATUS "Library path to link: ${LIB_PATH}")

# Find the library for the corresponding architecture (Library file name:
libmc_sdk_zsl_1_${ARCH}.so)
find_library(MC_SDK_LIB
    NAMES "mc_sdk_zsl_1_${ARCH}" # Library name (without the lib prefix
and .so suffix)
    PATHS ${LIB_PATH}
    NO_DEFAULT_PATH # Search only in the specified path to avoid finding
other versions
    REQUIRED
)

# Verify whether the library was found
if(NOT MC_SDK_LIB)
    message(FATAL_ERROR "Library file mc_sdk_zsl_1_${ARCH}.so not found
in ${LIB_PATH}. Please check if the library exists")
```

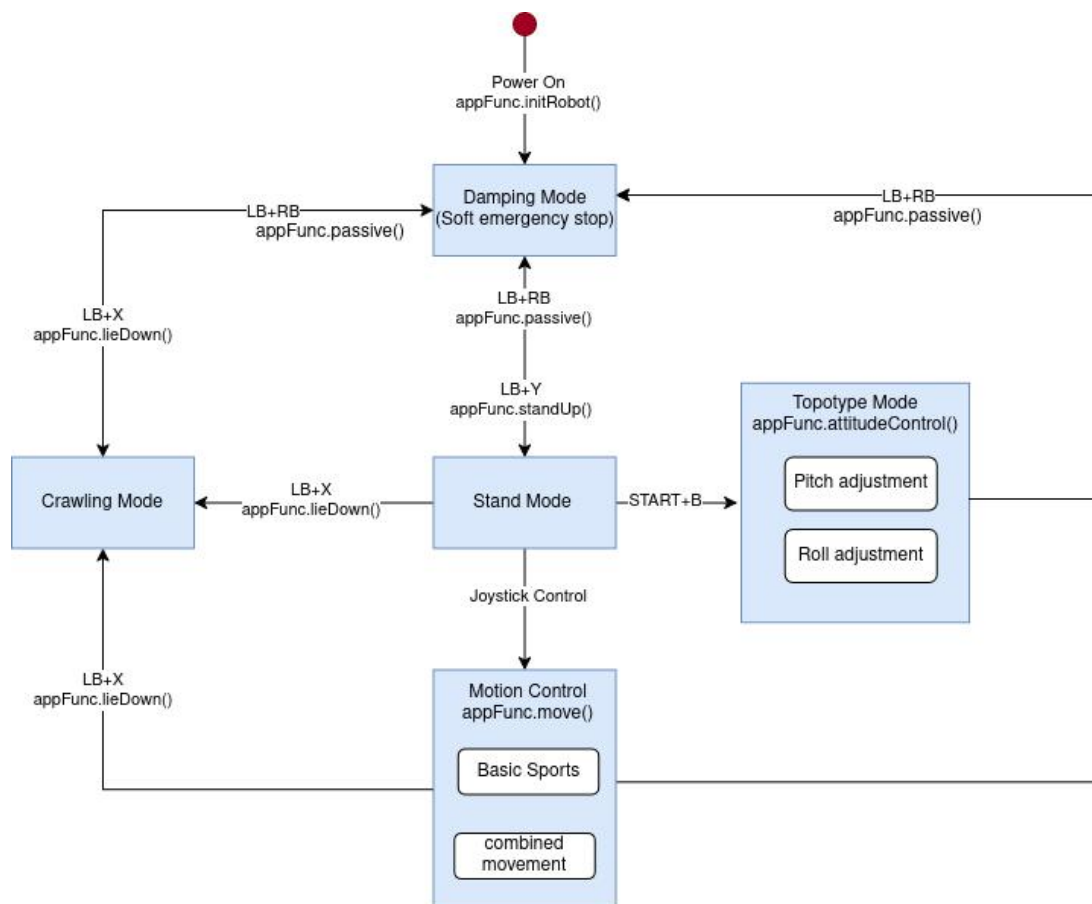
```
else()
    message(STATUS "Found library: ${MC_SDK_LIB}")
endif()

# High-level example
add_executable(highlevel_demo highlevel_demo.cpp)
target_link_libraries(highlevel_demo ${MC_SDK_LIB})
target_include_directories(highlevel_demo PRIVATE
    ${CMAKE_SOURCE_DIR}/../../../include # Common header files
    ${CMAKE_SOURCE_DIR}/../../../include/zsl-1 # ZSL-1 specific header
files
)

# Low-level example
add_executable(lowlevel_demo lowlevel_demo.cpp)
target_link_libraries(lowlevel_demo ${MC_SDK_LIB})
target_include_directories(lowlevel_demo PRIVATE
    ${CMAKE_SOURCE_DIR}/../../../include
    ${CMAKE_SOURCE_DIR}/../../../include/zsl-1
```

3.10 Secondary Development Control Instructions

Commands must be issued according to the following state transition logic. Otherwise, abnormal conditions such as the robot falling, faults, or no response may occur.



3.10.1 Joint Control Command Instructions

Command Sequence

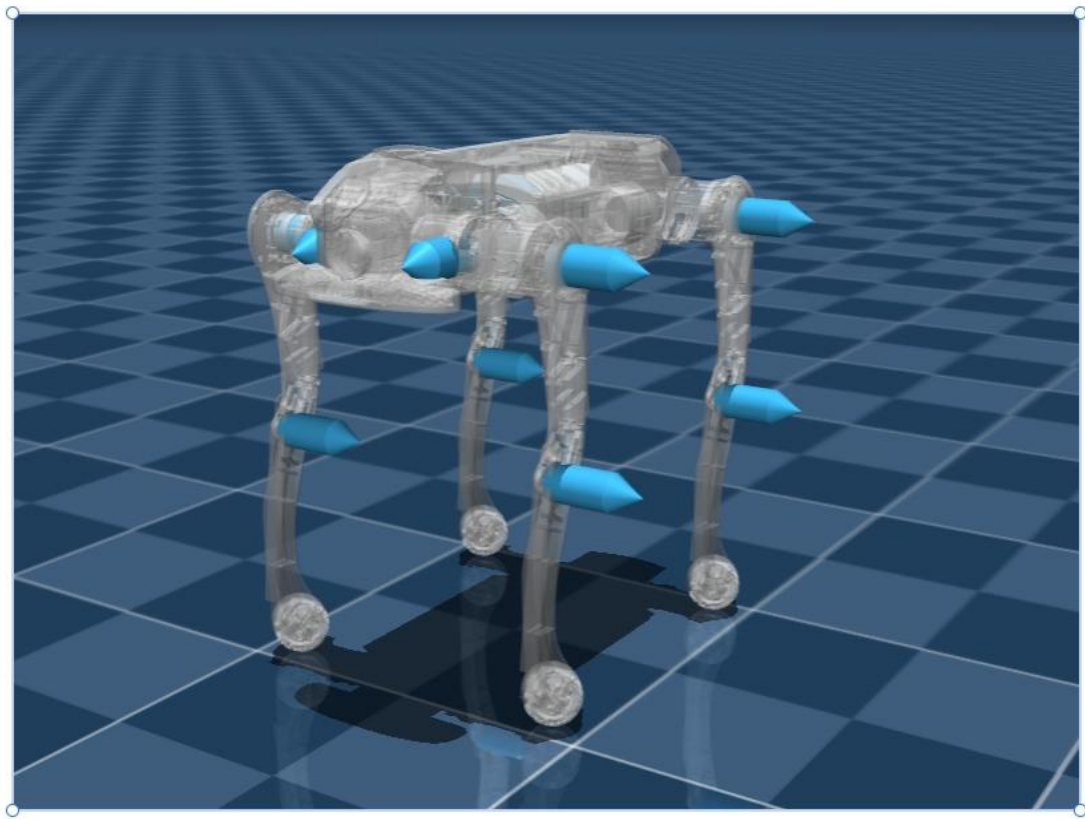
- FR (Front Right)
- FL (Front Left)
- RR (Rear Right)
- RL (Rear Left)

3.10.2 Joint Direction Definitions

A, H, K Joint Coordinate Frame: Forward = X, Left = Y, Up = Z

The coordinate directions follow the right-hand rule: X positive points forward, Y positive points left, Z positive points upward

Leg Order: Front-Right → Front-Left → Rear-Right → Rear-Left



Note: When using the SDK for control, low-battery warning logic must be implemented. When the quadruped robot's battery is low, the SDK connection will automatically disconnect. Therefore, when implementing high-level SDK control, corresponding handling logic must be incorporated

4. API Function Introduction

4.1 High-Level Functions Introduction

This class contains methods for controlling the quadruped robot's application-level functions. Users can utilize these methods for high-level control and to retrieve data.

Supported frequency for the device status retrieval interface in the motion control system: High-Level at 50 Hz

Supported frequency for the request interface in the motion control system: request data must be sent at 500 Hz to ensure stable device operation

4.1.1 Initialization (initRobot)

Function Name	<code>initRobot(std::string & local_ip, int local_port, string& dog_ip = "192.168.234.1")</code>
Function Overview	Establishes communication with the quadruped robot. The robot's default IP is 192.168.234.1; if the robot IP is changed, the new IP must be passed via this interface
Parameters	local_ip: User host IP address (required) local_port: User host port (required) dog_ip: Quadruped robot IP address (optional)
Return Value	None
Remarks	When communication setup fails, a failure log is output to the terminal

4.1.2 Check Connection Status (checkConnection)

Function Name	<code>checkConnection()</code>
Function Overview	Checks the robot's communication status
Parameters	None
Return Value	Type: bool true: Connection normal false: Connection failed
Remarks	Can be used for heartbeat monitoring

4.1.3 Stand Up (standUp)

Function Name	<code>standUP()</code>
Function Overview	Commands the quadruped robot to stand up
Parameters	None
Return Value	Type: uint32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	The quadruped robot stands up normally. This state cannot be entered directly while the robot is in motion

4.1.4 Lie Down (lieDown)

Function Name	lieDown()
Function Overview	Commands the quadruped robot to lie down
Parameters	None
Return Value	Type: int32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	The quadruped robot lies down with its joints locked. This state cannot be entered directly while the robot is in motion.

4.1.5 Damping (passive)

Function Name	passive()
Function Overview	Commands the quadruped robot to emergency stop and enter damping mode
Parameters	None
Return Value	Type: uint32t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	The quadruped robot enters damping mode and lies down

4.1.6 Move (move)

Function Name	move(float vx, float vy, float yaw_rate)
Function Overview	Commands the quadruped robot to move
Parameters	vx: Forward velocity (required; pass 0 for no movement) vy: Lateral velocity (required; pass 0 for no movement) yaw_rate: Angular velocity around the Z-axis (required; pass 0 for no movement)
Return Value	Type: uint32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Only valid when the robot is in the standing state. Controls the device to move at the specified velocities. vx range: -3m/s to -0.05m/s; 0.05m/s to 3m/s vy range: -1.0m/s to -0.1m/s; 0.1m/s to 1.0m/s yaw_rate range: -3.0rad/s to -0.02rad/s; 0.02rad/s to 3.0rad/s

4.1.7 In-Place Jumping (jump)

Function Name	jump()
Function Overview	Commands the quadruped robot to perform a vertical jump
Parameters	None

Return Value	Type: uint32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Only valid when the robot is in the standing state. Cannot transition while the robot is in motion.

4.1.8 Forward Jumping (frontJump)

Function Name	frontJump()
Function Overview	Commands the quadruped robot to jump forward
Parameters	None
Return Value	Type: uint32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Only valid when the robot is in the standing state. Cannot transition while the robot is in motion.

4.1.9 Backflip (backflip)

Function Name	backflip()
Function Overview	Commands the quadruped robot to perform a backflip
Parameters	None
Return Value	Type: int32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Only valid when the robot is in the standing state. Cannot transition while the robot is in motion.

4.1.10 Greet (shakeHand)

Function Name	shakeHand()
Function Overview	Commands the quadruped robot to perform a greeting gesture
Parameters	None
Return Value	Type: uint32t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Only valid when the robot is in the standing state. Cannot transition while the robot is in motion.

4.1.11 Two-Leg Stand (twoLegStand)

Function Name	twoLegStand(float vx, float yaw_rate)
Function Overview	Commands the quadruped robot to stand on two legs
Parameters	vx: Forward velocity (required; pass 0 for no movement) yaw_rate: Angular velocity around the Z-axis (required; pass 0 for no movement)
Return Value	Type: uint32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Only valid when the robot is in the standing state. Cannot enter this state while the robot is in motion To exit the two-leg stand state, call cancelTwoLegStand(); other interfaces will not respond. This parameter controls the device to move at the specified velocities. vx range: -0.5m/s to -0.2m/s; 0.2m/s to 0.5m/s yaw_rate range: -1.0rad/s to 0.2rad/s; 0.2rad/s to 1.0rad/s

4.1.12 Cancel Two-Leg Stand (cancelTwoLegStand)

Function Name	cancelTwoLegStand()
Function Overview	Commands the quadruped robot to exit the two-leg standing state
Parameters	None
Return Value	None
Remarks	Exits the two-leg stand state. Valid only when the robot is currently in the two-leg stand state

4.1.13 Attitude Control (attitudeControl)

Function Name	attitudeControl(float roll_vel, float pitch_vel, float yaw_vel, float height_vel)
Function Overview	Controls the quadruped robot's in-place twisting motion and body height variation
Parameters	roll_vel: Angular velocity around the X-axis (required; pass 0 for no control) pitch_vel: Angular velocity around the Y-axis (required; pass 0 for no control) yaw_vel: Angular velocity around the Z-axis (required; pass 0 for no control) height_vel: Vertical height velocity (required; pass 0 for no control)
Return Value	Type: uint32_t 0: State transition successful 0x3012: Motor data lost 0x3010: Motor disabled 0x3011: Motor fault 0x3009: Motor angle limit exceeded 0x3007: State machine transition failed 0x3013: Velocity command too large
Remarks	Valid only when the robot is in standing state. Controls the device to twist in-place and change body height at the specified velocities. yaw_vel, pitch_vel, roll_vel range: -0.5rad/s to 0.5rad/s, height_vel range: -0.5m/s to 0.5m/s

4.1.14 Get Attitude Quaternion (getQuaternion)

Function Name	getQuaternion()
Function Overview	Retrieves the quadruped robot's current attitude quaternion
Parameters	None
Return Value	A vector containing 4 elements [w, x, y, z], corresponding to the four components of the quaternion, respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.15 Get Attitude Euler Angles (getRPY)

Function Name	getRPY()
Function Overview	Retrieves the quadruped robot's current attitude Euler angles (roll, pitch, yaw).
Parameters	None
Return Value	A vector containing 3 elements [roll, pitch, yaw] in radians, corresponding to rotations around the X, Y, and Z axes respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.16 Get Body Acceleration (getBodyAcc)

Function Name	getBodyAcc()
Function Overview	Retrieves the quadruped robot body's acceleration data in the body coordinate frame.
Parameters	None
Return Value	A vector containing 3 elements [ax, ay, az] in m/s^2 , corresponding to accelerations along the X, Y, and Z axes respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.17 Get Angular Velocity (getBodyGyro)

Function Name	getBodyGyro()
Function Overview	Retrieves the quadruped robot's body angular velocity data (body coordinate frame).
Parameters	None
Return Value	A vector containing 3 elements [gx, gy, gz] in rad/s, corresponding to angular velocities around the X, Y, and Z axes respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.18 Get Position in Origin Coordinate Frame (getPosition)

Function Name	getPosition()
Function Overview	Retrieves the quadruped robot's position in the power-on origin coordinate frame.
Parameters	None
Return Value	A vector containing 3 elements [x, y, z] in meters, representing the current position.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.19 Get Velocity in Origin Coordinate Frame (getWorldVelocity)

Function Name	getWorldVelocity()
----------------------	---------------------------

Function Overview	Retrieves the quadruped robot's movement velocity in the power-on origin coordinate frame.
Parameters	None
Return Value	A vector containing 3 elements [vx, vy, vz] in m/s, corresponding to velocities along the X, Y, and Z axes of the origin coordinate frame respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.20 Get Body Velocity (getBodyVelocity)

Function Name	getBodyVelocity()
Function Overview	Retrieves the quadruped robot's movement velocity within the body coordinate frame.
Parameters	None
Return Value	A vector containing 3 elements [vx, vy, vz] in m/s, corresponding to velocities along the X, Y, and Z axes of the body coordinate frame respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.1.21 Get Battery Level (getBatteryPower)

Function Name	getBatteryPower()
Function Overview	Retrieves the quadruped robot's battery level
Parameters	None
Return Value	Type: uint32_t Range: 0-100
Remarks	Return value range: 0-100, 0 = no power, 100 = fully charged

4.1.22 Get Control Mode (getCurrentCtrlMode)

Function Name	getCurrentCtrlMode()
Function Overview	Retrieves the quadruped robot's current control mode
Parameters	None
Return Value	Type: uint32_t 0: Damping mode (device not enabled) 1: Standing / greeting state 10: Motor free state when device is lying down 18: Moving state 21: Action state (attitude mode, jump mode, two-leg stand, etc.) 51: Lying-down state.
Remarks	None

4.1.23 Get Abad Joint Angles (getLegAbadJoint)

Function Name	getLegAbadJoint()
Function Overview	Retrieves the current angles of the hip abduction/adduction (Abad) joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements in radians, corresponding to the Abad joint angles of the front-right, front-left, rear-right, and rear-left legs, respectively.
Remarks	Unit: rad. Robot forward = X, left = Y, vertical up = Z

4.1.24 Get Hip Joint Angles (getLegHipJoint)

Function Name	getLegHipJoint()
Function Overview	Retrieves the current angles of the hip pitch joints for each leg.

Parameters	None
Return Value	A vector containing 4 elements in radians, corresponding to the Hip joint angles of the front-right, front-left, rear-right, and rear-left legs, respectively.
Remarks	Unit: rad/s. Robot forward = X, left = Y, vertical up = Z

4.1.25 Get Knee Joint Angles (getLegKneeJoint)

Function Name	getLegKneeJoint()
Function Overview	Retrieves the current angles of the knee joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements in radians, corresponding to the Knee joint angles of the front-right, front-left, rear-right, and rear-left legs, respectively.
Remarks	Unit: rad/s. Robot forward = X, left = Y, vertical up = Z

4.1.26 Get Abad Joint Angular Velocities (getLegAbadJointVel)

Function Name	getLegAbadJointVel()
Function Overview	Retrieves the current angular velocities of the Abad joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements in rad/s, corresponding to the Abad joint angular velocities of the front-right, front-left, rear-right, and rear-left legs, respectively.
Remarks	Unit: rad/s. Robot forward = X, left = Y, vertical up = Z

4.1.27 Get Hip Joint Angular Velocities (getLegHipJointVel)

Function Name	getLegHipJointVel()
Function Overview	Retrieves the current angular velocities of the Hip joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements in rad/s, corresponding to the Abad joint angular velocities of the front-right, front-left, rear-right, and rear-left legs, respectively.
Remarks	Unit: rad/s. Robot forward = X, left = Y, vertical up = Z

4.1.28 Get Knee Joint Angular Velocities (getLegKneeJointVel)

Function Name	getLegKneeJointVel()
Function Overview	Retrieves the current angular velocities of the Knee joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements in rad/s, corresponding to the Abad joint angular velocities of the front-right, front-left, rear-right, and rear-left legs, respectively.
Remarks	Unit: rad/s. Robot forward = X, left = Y, vertical up = Z

4.1.29 Get Abad Joint Torques (getLegAbadJointTorque)

Function Name	getLegAbadJointTorque()
Function Overview	Retrieves the current output torques of the Abad joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements, typically in N·m, corresponding to the same order as joint torque.
Remarks	Units are typically N·m, corresponding to the Abad joint torques of the front-right, front-left, rear-right, and rear-left legs, respectively.

4.1.30 Get Hip Joint Torques (getLegHipJointTorque)

Function Name	getLegHipJointTorque()
Function Overview	Retrieves the current output torques of the Hip joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements, typically in N·m, corresponding to the same order as joint torque.
Remarks	Units are typically N·m, corresponding to the Hip joint torques of the front-right, front-left, rear-right, and rear-left legs, respectively.

4.1.31 Get Knee Joint Torques (getLegKneeJointTorque)

Function Name	getLegKneeJointTorque()
Function Overview	Retrieves the current output torques of the Knee joints for each leg.
Parameters	None
Return Value	A vector containing 4 elements, typically in N·m, corresponding to the same order as joint torque.
Remarks	Units are typically N·m, corresponding to the Knee joint torques of the front-right, front-left, rear-right, and rear-left legs, respectively.

4.2 Low-Level Function Introduction

Class: Low-Level

- This class is used to interact with motors, including sending motor commands and retrieving motor status.

Supported frequency for the device status retrieval interface in the motion control system: Low-Level at 500 Hz

Supported frequency for the request interface in the motion control system: request data must be sent at 500 Hz to ensure stable device operation

4.2.1 Create Connection (initRobot)

Function Name	initRobot(std::string & local_ip, int local_port, string& dog_ip = "192.168.234.1")
Function Overview	Establishes communication with the quadruped robot. The robot's default IP is 192.168.234.1; if the robot IP is changed, the new IP must be passed via this interface
Parameters	local_ip: User host IP address (required) local_port: User host port (required) dog_ip: Quadruped robot IP address (optional)
Return Value	None
Remarks	When communication setup fails, a failure log is output to the terminal

4.2.2 Check Connection Status (checkConnection)

Function Name	checkConnection()
Function Overview	Checks the robot's communication status
Parameters	None
Return Value	Type: bool true: Connection normal false: Connection failed
Remarks	Can be used for heartbeat monitoring

4.2.3 Send Motor Command (sendMotorCmd)

Function Name	sendMotorCmd(motorCmd& cmd)
Function Overview	Publishes motor control commands
Parameters	motorCmd
Return Value	Returns 0 on success, -1 on failure

Remarks	Leg order: Leg 1 = Front-Right, Leg 2 = Front-Left, Leg 3 = Rear-Right, Leg 4 =Rear-Left. Command coordinate frame: Robot forward = X, left = Y, vertical up = Z
---------	---

● Cmd Data Protocol

```

C++

struct alignas(4) motorCmd {

    float q_des_abad[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint position
    commands
    float q_des_hip[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint position
    commands
    float q_des_knee[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint position
    commands

    float qd_des_abad[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint velocity
    commands
    float qd_des_hip[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint velocity
    commands
    float qd_des_knee[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint velocity
    commands

    float kp_abad[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint Kp gains
    float kp_hip[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint Kp gains
    float kp_knee[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint Kp gains

    float kd_abad[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint Kd gains
    float kd_hip[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint Kd gains
    float kd_knee[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint Kd gains

    float tau_abad_ff[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint torque
    commands
    float tau_hip_ff[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint torque
    commands
    float tau_knee_ff[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint torque
    commands

};
  
```

4.2.4 Get Attitude Quaternion (getQuaternion)

Function Name	getQuaternion()
Function Overview	Retrieves the quadruped robot's current attitude quaternion
Parameters	None
Return Value	A vector containing 4 elements [w, x, y, z], corresponding to the four components of the quaternion, respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.2.5 Get Attitude Euler Angles (getRPY)

Function Name	getRPY()
Function Overview	Retrieves the quadruped robot's current attitude Euler angles (roll, pitch, yaw).
Parameters	None
Return Value	A vector containing 3 elements [roll, pitch, yaw] in radians, corresponding to rotations around the X, Y, and Z axes respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.2.6 Get Body Acceleration (getBodyAcc)

Function Name	getBodyAcc()
Function Overview	Retrieves the quadruped robot body's acceleration data in the body coordinate frame.
Parameters	None
Return Value	A vector containing 3 elements [ax, ay, az] in m/s ² , corresponding to accelerations along the X, Y, and Z axes respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.2.7 Get Angular Velocity (getBodyGyro)

Function Name	getBodyGyro()
Function Overview	Retrieves the quadruped robot's body angular velocity data (body coordinate frame).
Parameters	None
Return Value	A vector containing 3 elements [gx, gy, gz] in rad/s, corresponding to angular velocities around the X, Y, and Z axes respectively.
Remarks	Robot forward = X, left = Y, vertical up = Z

4.2.8 Get Motor Data (getMotorState)

Function Name	getMotorState()
Function Overview	Retrieves the quadruped robot's motor data
Parameters	None
Return Value	Returns motor data for each motor
Remarks	Unit: rad

● MotorState Data Protocol

C++

```
struct alignas(4) motorState {
    // Joint order: FL, FR, RL, RR
    float q_abad[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint angles
    float q_hip[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint angles
    float q_knee[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint angles

    float qd_abad[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint velocities
    float qd_hip[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint velocities
    float qd_knee[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint velocities

    float tau_abad_fb[4] = {0.0, 0.0, 0.0, 0.0}; // Abad joint torque
    feedback
    float tau_hip_fb[4] = {0.0, 0.0, 0.0, 0.0}; // Hip joint torque
    feedback
    float tau_knee_fb[4] = {0.0, 0.0, 0.0, 0.0}; // Knee joint torque
    feedback
};
```

4.2.9 Check Motor Data Availability (haveMotorData)

Function Name	haveMotorData()
Function Overview	Checks whether motor data has been successfully received
Parameters	None
Return Value	Returns true or false
Remarks	Check for motor data availability before sending motor commands

5. Debugging Guide

5.1 Video Stream Push Service

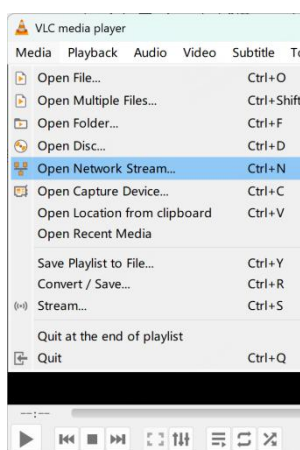
The quadruped robot's wide-angle camera video is pushed via Gstreamer to the following URLs:

```
rtsp://192.168.234.1:8554/test #Wireless connection
rtsp://192.168.168.168:8554/test #Wired connection
rtsp://{Dog_IP}:8554/test      #Other connection methods
```

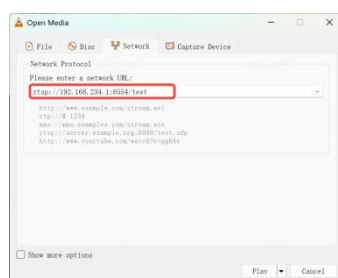
By default, the live video feed is pulled and displayed on the remote-controller. The streams can also be previewed and accessed using VLC or ffmpeg. Examples:

Case 1: On a Windows PC connected to the robot's Wi-Fi, use VLC to view the video stream

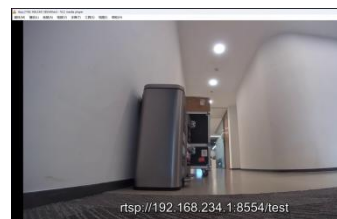
Step 1: Open VLC,
click "Media → Open
Network Stream"



Step 2: Enter the
device stream URL,
e. g.,
rtsp://192.168.234.1:8
554/test



Step 3: Preview the
video



Case 2: On an Ubuntu PC connected to the robot's Wi-Fi, use the following commands in the terminal to retrieve the video stream:

Bash

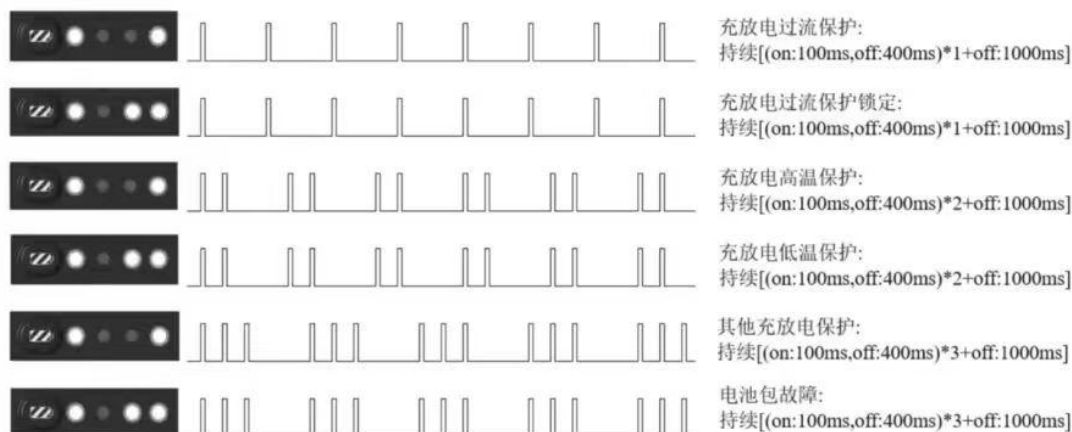
```
ffmpeg rtsp://192.168.234.1:8554/test      #Wireless connection
ffmpeg rtsp://192.168.168.168:8554/test    #Wired connection
```

5.2 Indicator Light Description

5.2.1 Quadruped Robot Tail Light Description

Light Status	Indication	Remarks
Solid Blue	Boot complete	Normal Status
Flashing Blue	Booting in progress	Boot failure commonly occurs when using a wired connection for SDK control without restoring IP configuration
Solid White	Lab mode	Stunt maneuvers are available in this mode
Solid Green	SDK control mode	Control authority switching enabled and the robot is currently in SDK control mode
Solid Yellow	Battery level below 20%	Low battery; cannot enable Turbo mode or perform SDK control
Flashing Yellow	Battery level below 5%	Low battery; recharge immediately
Solid Red	Joint temperature too high, cooling required	
Flashing Red	System anomaly	Contact after-sales support for troubleshooting

5.2.2 Battery Indicator Light Description



Note: If the indicator light flashes in the "Charge/Discharge High-Temperature Protection" pattern during charging, remove the battery and let it sit for 30 minutes before charging again.

5.3 Log Query

Log directory: `/home/firefly/tmp/log`

```
firefly@firefly:~$ cd /tmp/log/
firefly@firefly:/tmp/log$ ls
arc_platform-LOG  ecal2ros-LOG  motor_cmd.2.log  motor_state.4.log
dog_task.1.log    imu_data.1.log  motor_cmd.3.log  motor_state.log
dog_task.2.log    imu_data.2.log  motor_cmd.4.log  robot_forward-LOG
dog_task.3.log    imu_data.3.log  motor_cmd.log     spline.log
dog_task.4.log    imu_data.4.log  motor_state.1.log
dog_task.5.log    imu_data.log    motor_state.2.log
dog_task.log      motor_cmd.1.log  motor_state.3.log
```

Caution: The quadruped robot has a log cleanup service enabled at startup. If an anomaly occurs, do not shut down the robot; retrieve the logs first.

5.4 Control Authority Switching Function

Applicable to D1 Edu and D1 Ultra quadruped robots. This section describes how to switch control authority between the remote controller and the SDK

Simultaneous control via both the SDK and remote controller is not supported

When using SDK control, if the battery level is low, control authority will automatically switch back to the remote controller

5.4.1 Version Requirements

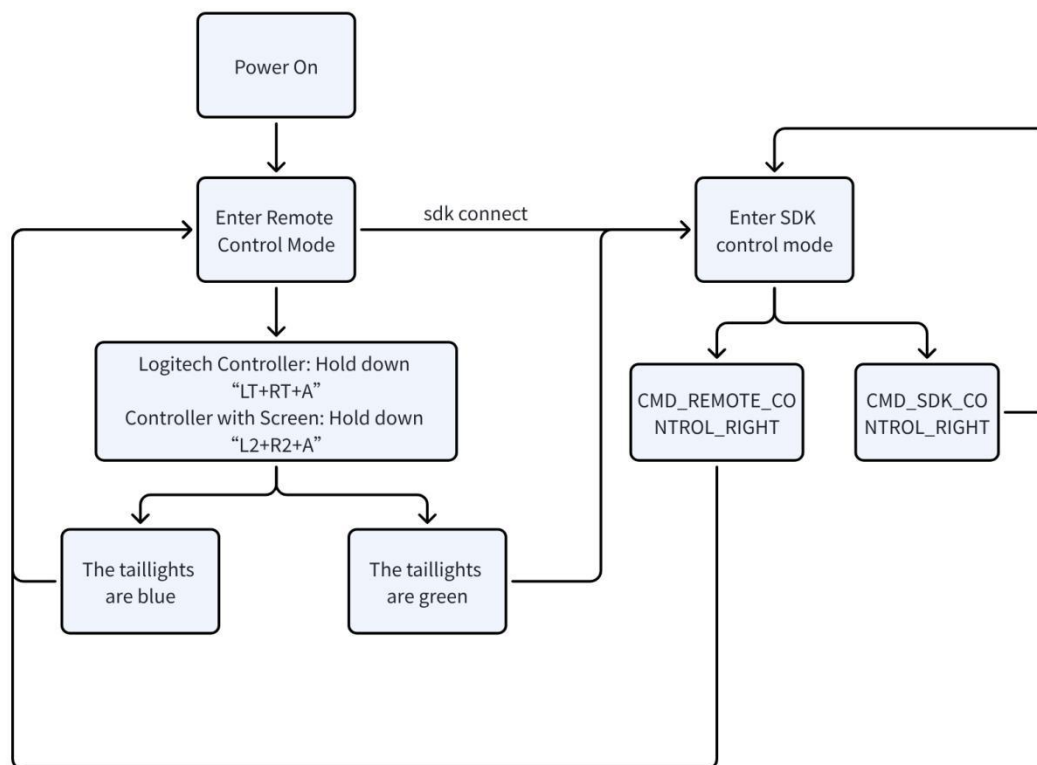
Quadruped Robot Firmware: v0.2.5 or higher

Remote Controller APP: v2.4.5 or higher (D1 Ultra)

5.4.2 Function Description

Control Side		How to Switch	Indicator Light Description
Remote Control Side	Logitech Gamepad Remote Controller	Long press LT+RT+A	Remote controller mode: Tail light solid blue SDK control mode: Tail light solid green
	Remote Controller with Integrated Display	Long press L2+R2+A	
SDK Side	CPP	Switch to SDK mode: <code>zsibot_exec.SetCmd(CmdCode::CMD_SDK_CONTROL_RIGHT);</code> Switch to remote controller mode: <code>zsibot_exec.SetCmd(CmdCode::CMD_REMOTE_CONTROL_RIGHT);</code>	

5.4.3 Control Block Diagram



5.5 Modifying the Wired Ethernet Port IP Address

1. Use a terminal or SSH tool to access the quadruped robot

```
Bash
ssh firefly@192.168.234.1
```

2. Edit the eth0 configuration file

```
Bash
sudo vim /etc/NetworkManager/system-connections/eth0.nmconnection
```

To modify the RJ45 port IP address of the quadruped robot, change `address1=192.168.168.168/24` to the desired IP, e.g., `192.168.90.100/24`

Before modification:

```
[connection]
id=eth0
uuid=da3eaba0-947b-4768-ab9b-264b0a4a5178
type=ethernet
interface-name=eth0

[ethernet]

[ipv4]
address1=192.168.168.168/24
method=manual

[ipv6]
addr-gen-mode=stable-privacy
method=auto

[proxy]
```

After modification:

```
[connection]
id=eth0
uuid=da3eaba0-947b-4768-ab9b-264b0a4a5178
type=ethernet
interface-name=eth0

[ethernet]

[ipv4]
address1=192.168.90.100/24
method=manual

[ipv6]
addr-gen-mode=stable-privacy
method=auto

[proxy]
```

3. Reboot the quadraped robot for the configuration to take effect

5.6 Running the SDK on the Robot Body

Step 1: Copy the SDK into the robot body

```
Bash
sudo scp -r /home/{user directory}/agibot_D1_Edu-Ultra
firefly@192.168.234.1:/home/firefly/Documents
```

Step 2: Log in to the robot body

```
Bash
ssh firefly@192.168.234.1
```

Step 3: Modify the SDK configuration file

```
Bash
sudo vim /opt/export/config/sdk_config.yaml
```

Modify to:

```
firefly@firefly:~$ cat /opt/export/config/sdk_config.yaml
target_ip: "192.168.234.1"
target_port: 43988
```

After modifying, reboot the robot

Step 4: Modify and compile the SDK files

Modify the IP address in the Python or C++ script for robot initialization. Set both `local_ip` and `dog_ip` to `192.168.234.1`:

```
Bash
#python
app.initRobot("192.168.234.1", 43988, "192.168.234.1")
#local_i, local_port, dog_ip

#C++
constexpr int CLIENT_PORT = 43988;    // local port
std::string CLIENT_IP = "192.168.234.1"; // local IP address
std::string DOG_IP = "192.168.234.1";  // dog ip
```

Compile:

```
Bash
cd /home/user/Documents/agibot_D1_Edu-Ultra_v0.2.7/agibot_D1_Edu-
Ultra/demo/zsl-1/cpp
mkdir build
cd build
cmake ..
make -j6
```

Step 5: Run the demo

```
Bash
cd demo/zsl-1/python/examples
python highlevel_demo.py
or
python lowlevel_demo.py
```

5.7 Running the SDK on an External LAN

5.7.1 Control PC Configuration

Connect the control PC (Ubuntu 22.04) to the external LAN and check the current network interface IP. As shown in the figure below, `local_ip = 192.168.43.222`

```
wlp0s20f3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.43.222 netmask 255.255.255.0 broadcast 192.168.43.255
    inet6 fe80::a4c1:30c4:22f4:f07d prefixlen 64 scopeid 0x20<link>
    ether 98:5f:41:4f:e2:78 txqueuelen 1000 (Ethernet)
    RX packets 288012 bytes 205295258 (205.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 392205 bytes 574850559 (574.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

agiuser@PF51WX3A:~$
```

5.7.2 Quadruped Robot Configuration

(1) Use the `nmcli` command to connect the quadruped robot to the external Wi-Fi network (note: network isolation and firewall settings may apply)

Bash

```
ssh firefly@192.168.234.1
sudo nmcli device wifi connect <wifi_name> password <wifi_password>
```

Note: The device has a network cleanup service enabled by default. To persist the network connection after reboot, run the following commands on the robot body:

Bash

```
sudo systemctl stop networkmanager-cleanup.service
sudo systemctl disable networkmanager-cleanup.service
```

2. Check the assigned IP address

Use `ifconfig` to check the IP address of the `wlan0` interface. The `dog_IP` is the IP address of the `wlan0` interface found. As shown in the figure below, `dog_IP = 192.168.43.93`

Bash

```
ifconfig
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.43.93 netmask 255.255.255.0 broadcast 192.168.43.255
    inet6 fe80::d5c7:d3f6:1e5f:fffe prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:72:18:18 txqueuelen 3000 (Ethernet)
    RX packets 26 bytes 11389 (11.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 71 bytes 10985 (10.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

3. Modify the communication configuration files

3.1 Modify sdk_config.yaml

Bash

```
sudo vim /opt/export/config/sdk_config.yaml
```

```
target_ip: "192.168.43.222"
target_port: 43988
~
~
```

3.2 Modify start_motion_control.sh

Bash

```
sudo vim /opt/app_launch/start_motion_control.sh
```

```
#!/bin/bash

echo "start motion control"

# 共享内存文件路径
SHM_FILE="/dev/shm/spline_shm"

# 循环检查设备是否存在
while true; do
    if [ -e "$SHM_FILE" ]; then
        echo "共享内存文件 $SHM_FILE 已存在。"
        break
    else
        echo "共享内存文件 $SHM_FILE 不存在，等待 1 秒后重试..."
        sleep 1
    fi
done

# 共享内存文件存在后执行的命令
echo "共享内存文件已准备好，可以执行后续操作。"

sudo ifconfig lo multicast
sudo route add -net 224.0.0.0 netmask 240.0.0.0 dev lo

export LD_LIBRARY_PATH=/opt/export/mc/bin
export ROBOT_TYPE=XG
export SDK_CLIENT_IP="192.168.43.93"

cd /opt/export/mc/bin && taskset -c 7 ./mc_ctrl r
~
```

3.3 Modify robot-launch.service

Bash

```
sudo vim /etc/systemd/system/robot-launch.service
```

```
[Unit]
Description=Run robot-launch at startup
After=network.target

[Service]
ExecStart=robot-launch server
ExecStop=/usr/local/bin/robot_launch_stop.sh
TimeoutStopSec=5s
Restart=always
User=root
Environment="ROBOT_HOME=/home/firefly"
Environment="ROBOT_LOG_DIR=/userdata/log"
Environment="ROBOT_NET_INTERFACES=wlan0"

[Install]
WantedBy=multi-user.target
~
~
```

3.4 Reboot for changes to take effect

Bash

```
sudo reboot
```

5.7.3 Running the SDK

1. Modify the IP addresses in the SDK demo file

Using Python demo as an example: set `local_ip` to 192.168.43.222 and `dog_ip` to 192.168.43.93

```

agiuser@PF51WX3A: ~/Documents/test/edu/agibot_D1_Edu-...
print("8. Turn Right")
print("9. Jump")
print("10. Front Jump")
print("11. Backflip")
print("12. Shake Hand")
print("13. Attitude Control (Gentle Wobble)")
print("0. Exit")
print("-----")

def main():
    """Main function to run the interactive demo."""
    try:
        app = mc_sdk_zsl_1_py.HighLevel()
        # Use 127.0.0.1 for both local and robot IP for simulation/local testing
        app.initRobot("192.168.43.222", 43988, "192.168.43.93")
        print("Successfully initialized robot connection.")

        # It's often necessary to stand up first
        print("Robot is standing up...")
        app.standUp()
        time.sleep(3)
        print("Robot is ready for commands.")
    except Exception as e:
        print(f"Error: {e}")

-- INSERT --
41,62 22%

```

2. Run the demo program

```

agiuser@PF51WX3A: ~/Documents/test/edu/agibot_D1_Edu-Ultra/demo/zsl-1/python/examples$ python3 highlevel_demo_Interactive.py
/home/agiuser/Documents/test/edu/agibot_D1_Edu-Ultra/lib/zsl-1/x86_64
ROBOT SDK is Running, APP Command or other input is InValid!
mp_recv_cp: 192.168.43.222:43988
client bind success
Successfully initialized robot connection.
Robot is standing up...
connect success!
Robot is ready for commands.

ZsiBot Interactive Demo
-----
1. Stand Up
2. Lie Down
3. Move Forward
4. Move Backward
5. Move Left
6. Move Right
7. Turn Left
8. Turn Right
9. Jump
10. Front Jump
11. Backflip
12. Shake Hand
13. Attitude Control (Gentle Wobble)
0. Exit
-----
Enter command number: 

```

6. FAQ

6.1: Connect Timeout

When running the SDK program, a connection timeout occurs with the following error message:

```
Bash

client bind success
connect time out
error async receive from
```

1. Modify the `target_ip` parameter in the [SDK configuration file](#) according to the guide.
2. If communication with the device does not use a direct WiFi connection, also modify the [SDK_CLIENT_IP](#) parameter according to the guide. When not using Wi-Fi direct connection, the device may fail to bind the SDK for motion control. Follow the guide for [wired connection IP binding failure](#) to adjust the parameters.
3. If communication with the device uses the quadraped robot's AP hotspot, in addition to correctly configuring `target_ip`, ensure that:

In `/opt/app_launch/start_motion_control.sh`, delete or comment out `export SDK_CLIENT_IP="IP address of the network interface connected to the robot"`

In `/etc/systemd/system/robot-launch.service`, delete `Environment="ROBOT_NET_INTERFACES=wlan0"`

6.2 Wired Connection IP Binding Failure

After configuring `SDK_CLIENT_IP`, when using the `wlan0` or `eth0` network interface for SDK control, the network may not have been initialized after the device boots, and the corresponding network interface may not have been assigned an IP address, causing the motion control program to fail to bind to the IP address.

Symptoms: After running the SDK program, a `connect time out` error is displayed. After logging into the device's main controller, execute the command `sudo lsof -i -P -n -c mc_ctrl | grep 43997` to check the output; the motion control port is not bound.

Fix: Modify the `/etc/systemd/system/robot-launch.service` configuration file on the device's main controller. In the Service section, add the following line: `Environment="ROBOT_NET_INTERFACES=wlan0"`, where `wlan0` is the name of the corresponding network interface name. For wired connections, change this to `eth0`.

Complete configuration file example:

```
Bash
```

```
[Unit]
Description=Run robot-launch at startup
After=network.target

[Service]
ExecStart=robot-launch server
ExecStop=/usr/local/bin/robot_launch_stop.sh
TimeoutStopSec=5s
Restart=always
User=root
Environment="ROBOT_HOME=/home/firefly"
Environment="ROBOT_LOG_DIR=/userdata/log"
Environment="ROBOT_NET_INTERFACES=eth0"

[Install]
WantedBy=multi-user.target
```

The device must be rebooted for the configuration to take effect!

This parameter configuration may, in some cases, prevent the quadruped robot program from starting!

After configuring ROBOT_NET_INTERFACES, the program will check whether the specified network interface has been assigned an IP address. If the interface never obtains an IP, the quadruped robot program will fail to start.

6.3 Using wlan0 to Connect to an External Wi-Fi Network; Device Does Not Auto-Connect After Reboot

There is a conflict between the device's AP and the wlan0 network interface. Some versions have an auto-connect cleanup feature enabled. To disable this cleanup feature, execute the following commands:

```
Bash
```

```
sudo systemctl stop networkmanager-cleanup.service //Stop the
network cleanup service
sudo systemctl disable networkmanager-cleanup.service //Disable the
network cleanup service
```

6.4: Python Import Failure

Verify that the script has added `lib/<model>/<arch>` to `sys.path`.

The Python version must match the extension module; the Python version must be 3.10.

```
li@li-desktop:~/agibot_D1_Edu-Ultra-main/demo/zsl-1/python/examples$ python3.10 highlevel_demo.py
ROBOT SDK is Running, APP Command or other input is Invalid!
mp recv cp: 192.168.168.5:43988
client bind success
connect success!
li@li-desktop:~/agibot_D1_Edu-Ultra-main/demo/zsl-1/python/examples$
```

6.5: C++ Linking Failure?

Verify that the corresponding `.so` file exists in `lib/<model>/<arch>`.

Ensure that CMake detects the correct architecture (`x86_64` / `aarch64`).

6.6 What is the Supported Frequency for the Device Status Retrieval Interface in the Motion Control System?

Low-Level can be set to 500Hz. High-Level can be set to 50Hz.

6.7: What is the Supported Frequency for the SDK Control Interface?

The control data can be sent at a maximum of 500Hz. Generally, 20 - 50Hz is sufficient to ensure stable device operation.

6.8: Is the SDK Program Thread-Safe?

Yes, the SDK program is thread-safe.

6.9: Does the SDK Support Communication Monitoring?

The motion control program continuously monitors whether it is receiving data from the SDK. If no data is received within 3 seconds, the SDK communication is considered interrupted, and the device will automatically switch to damping mode and lie down.

6.10: When Starting the Motion Control Program via Service on the Robot, How Can I Ensure the Robot's Own Programs Are Started?

The `After` parameter in the service file can be configured as `robot-launch`; the `robot-launch` service will start the robot's own programs.

6.11: Can Gamepad Remote Control and SDK Control Be Switched Simultaneously?

In the current version, the SDK has higher priority than the gamepad remote

controller. Simultaneous switching is not supported.

6.12: Is the Retrieved IMU Data Raw Data?

Yes, the retrieved IMU data is raw data.

6.13: The Retrieved IMU Data Seems to Have Errors?

The IMU data accuracy on the device is affected by the product model and has inherent precision errors. If higher IMU precision is required, a separate high-precision IMU must be purchased and mounted on the upper body.

6.14: After Using SDK Control, When Using the Gamepad Remote Controller to Send a Stand Command, Why Does the Device Perform "Push-Ups" In-Place?

If the `SDK_CLIENT_IP` parameter has been configured, the motion control program will bind to the corresponding IP at startup. If that IP does not exist, this issue will occur.

6.15: What is the Rotation Order When Converting Quaternion to Euler Angles?

The Euler angle order is ZYX - i.e., rotation around the Z-axis first, then the Y-axis, and finally the X-axis.

6.16: Can High-Level and Low-Level Be Used Simultaneously?

No, High-Level and Low-Level cannot be used simultaneously. When switching between the two interfaces, allow sufficient time for the device ports to be released.

6.17: Why Can't the SDK Control the Device After the Motion Control Program Is Updated?

After a motion control program update, the motion control-related configurations are overwritten. The `SDK_CONFIG` and `SDK_CLIENT_IP` parameters must be reconfigured.

6.18: Is There a ROS Interface?

Currently, there is no ROS interface available.

6.19: When Using the SDK, the Program Stalls at the `initRobot` Function and Does Not Proceed?

The SDK depends on Boost 1.74.0. If an older version is used, compatibility issues may arise due to structural changes.

6.20 Does It Support Deploying Control Logic Directly on the Quadruped Robot Body?

It is not recommended to deploy control logic directly on the quadruped robot body, as the robot has no spare computing capacity internally.

Glossary		
Abbreviation	Full Name	Description / Remarks
AP	Access Point	The Wi-Fi hotspot broadcast by the robot itself (SSID: XG2WIFI_XXXXX).
API	Application Programming Interface	Refers to the high-level and low-level command set interfaces provided by the SDK.
CMake	Cross-platform Make	Cross-platform build tool used to compile the C++ sample programs in the SDK (version 3.8+ required).
CPU	Central Processing Unit	The robot is equipped with an 8-core high-performance CPU.
DFOV	Diagonal Field of View	Camera parameter; diagonal FOV (122° for this model).
DHCP	Dynamic Host Configuration Protocol	DHCP is enabled for wireless networking; disabled for wired networking.
ffplay	FFmpeg player	A media player bundled with FFmpeg, can be used to preview RTSP video streams.
FOV	Field of View	General camera parameter, encompassing DFOV, HFOV, and VFOV.
GCC	GNU Compiler Collection	Dependencies for compiling the SDK (version 11+ required).
GStreamer	GStreamer multimedia framework	Multimedia framework used to push camera video as RTSP streams.
HFOV	Horizontal Field of View	Camera parameter; horizontal FOV (122° for this model).
Hz	Hertz	Unit of frequency (e.g., control interface recommended at 20-50Hz, IMU data at 50Hz, etc.).
IMU	Inertial Measurement Unit	Standard onboard unit providing orientation, acceleration, and angular velocity data.
IP	Internet Protocol	Network address; default wireless IP is 192.168.234.1, wired IP is 192.168.168.168.
IP54	Ingress Protection 54	Protection rating (dust-protected and splash-water-resistant).
OTA	Over-The-Air	Used for firmware updates of the device
RJ45	Registered Jack 45	Standard Ethernet connector interface (one RJ45 port is located on the rear of the robot).
ROS	Robot Operating System	Robot operating system
RPY	Roll, Pitch, Yaw	Euler angles (roll, pitch, yaw), obtained via getRPY(), with rotation order ZYX.
RTSP	Real Time Streaming Protocol	Video streaming address: rtsp://<Dog_IP>:8554/test.
SBUS	Serial Bus	A serial communication protocol for remote controllers; the robot provides one SBUS interface.
SDK	Software Development Kit	Provides C++ and Python interfaces, divided into HighLevel and LowLevel.
SSH	Secure Shell	Secure remote login protocol; default username firefly, default password firefly.
SSID	Service Set Identifier	Wireless network name (printed on the label at the bottom of the robot).
UART	Universal Asynchronous Receiver-Transmitter	The robot provides one UART interface.
USB	Universal Serial Bus	The robot provides two Type-A USB ports.
VFOV	Vertical Field of View	Camera parameter; vertical FOV (70° for this model).
VLC	VideoLAN Client	Media player that can be used to preview RTSP video streams.

