UNITREE

Software Training

Secondary Development Training for Unitree Partner

Software Training

Module 01
Platform Building

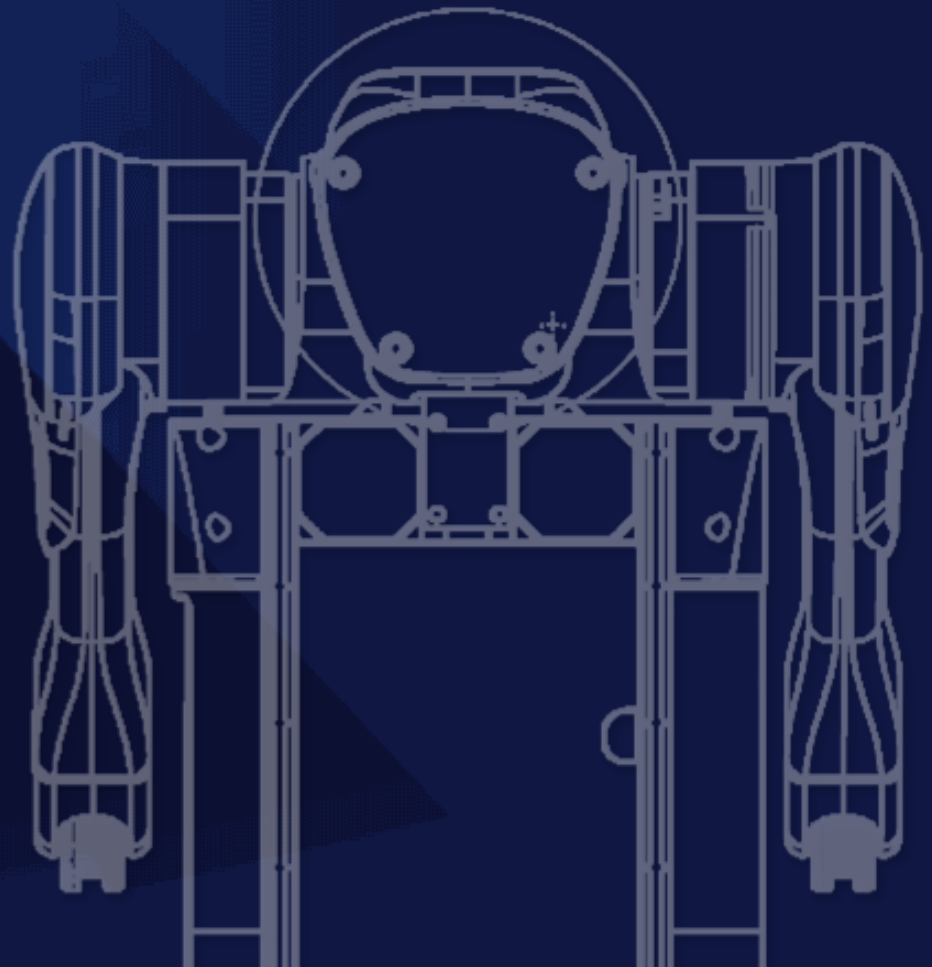
- 100 Hardware Platform
- 101 Connection Platform
- 102 Software Platform

Module 02
SDK Development

- 200 Overview SDK2
- 201 Conception Introduction
- 202H High Level Basic
- 202L Low Level Basic

Module 03
Advanced Development

- 300 Example Overview
- 301 SLAM Interface
- 302 Policy & Mimic
- 303 Embodied Agent



Module 01
Platform Building

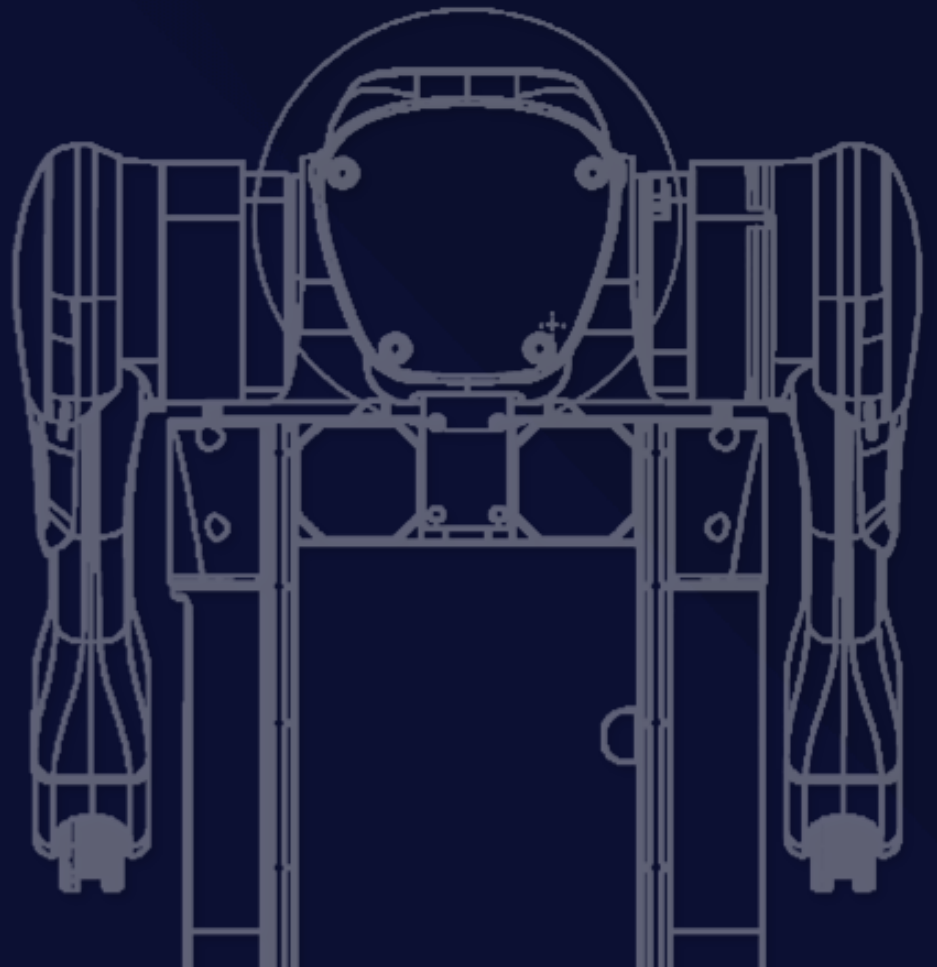
100
Hardware
Platform

101
Connection
Platform

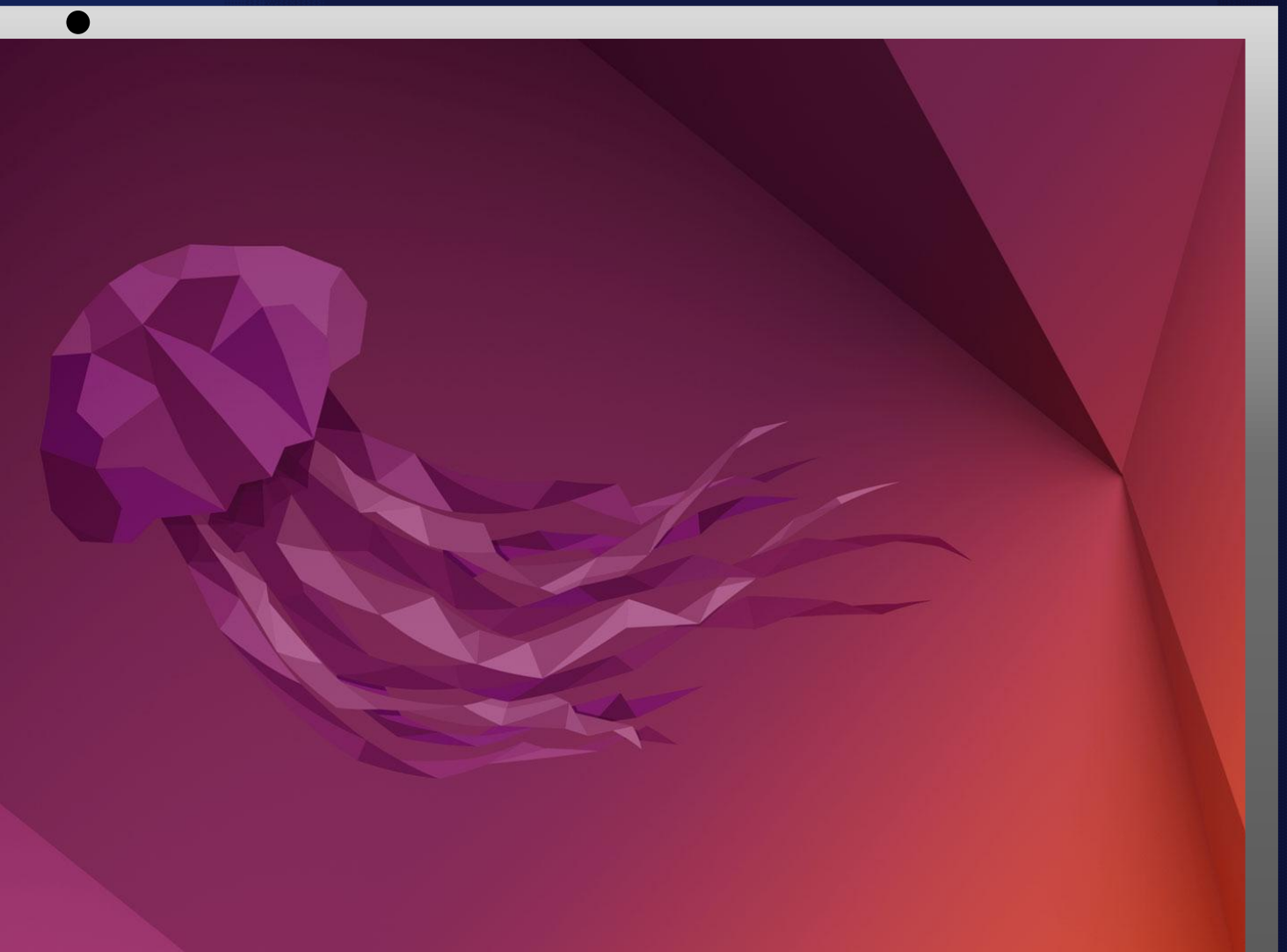
102
Software
Platform

Software Training

100 Hardware Platform



100 Hardware Platform

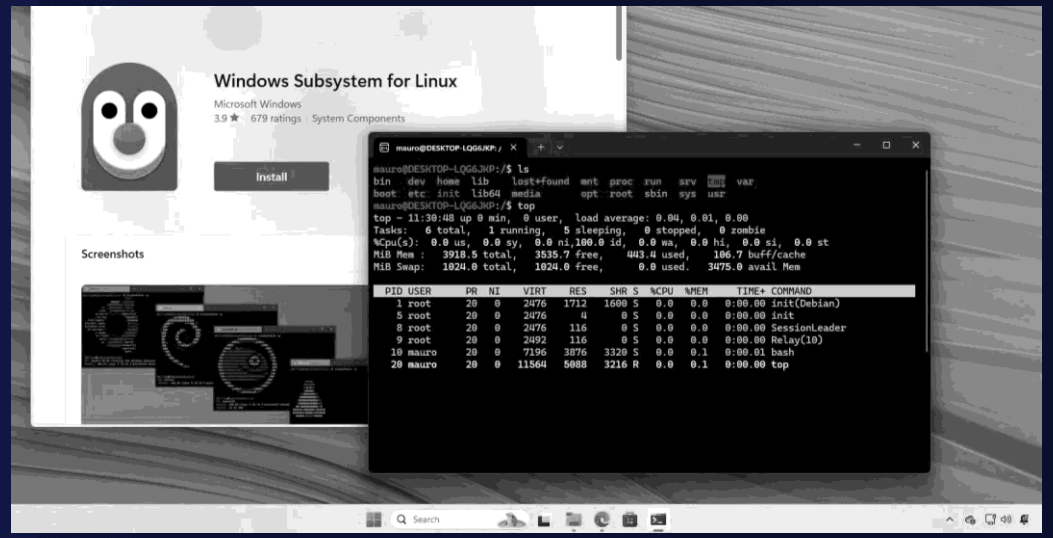


Type
Real Machine

Operating System
Ubuntu 20.04/22.04

Memory
More than 8GB

Interfaces
RJ45*1 (USB HUB v)

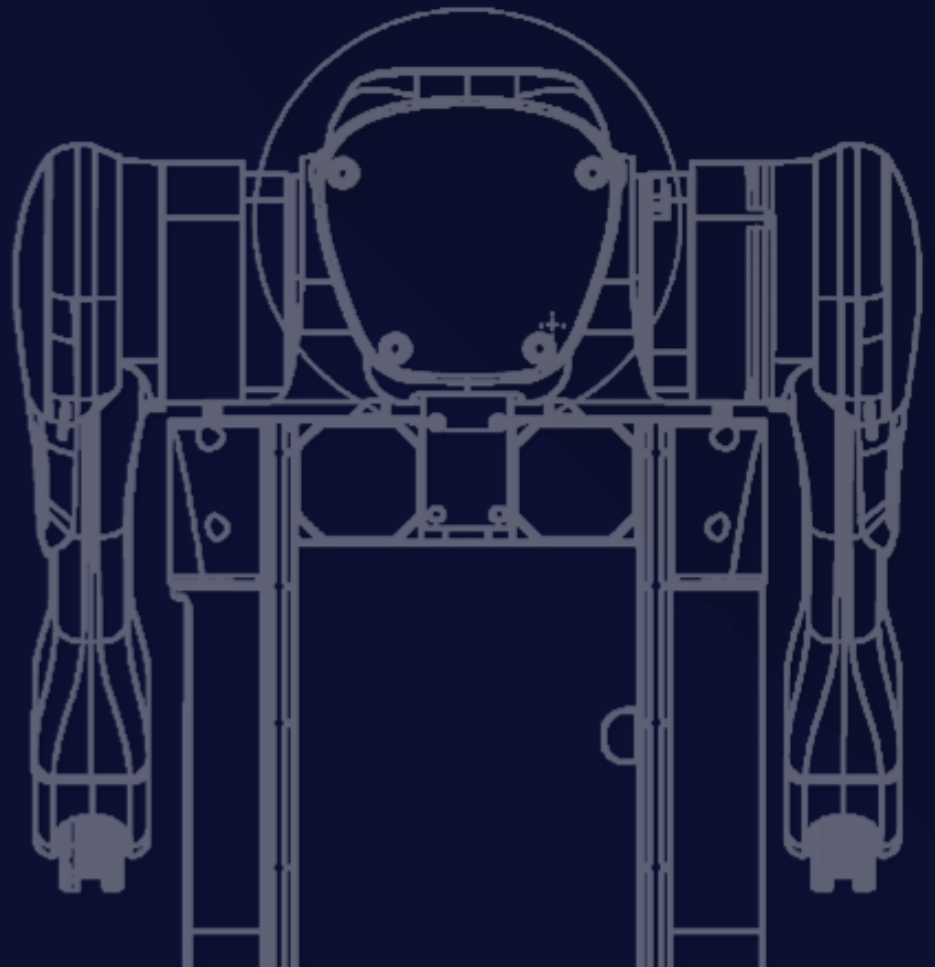


Why not VMs or WSL?

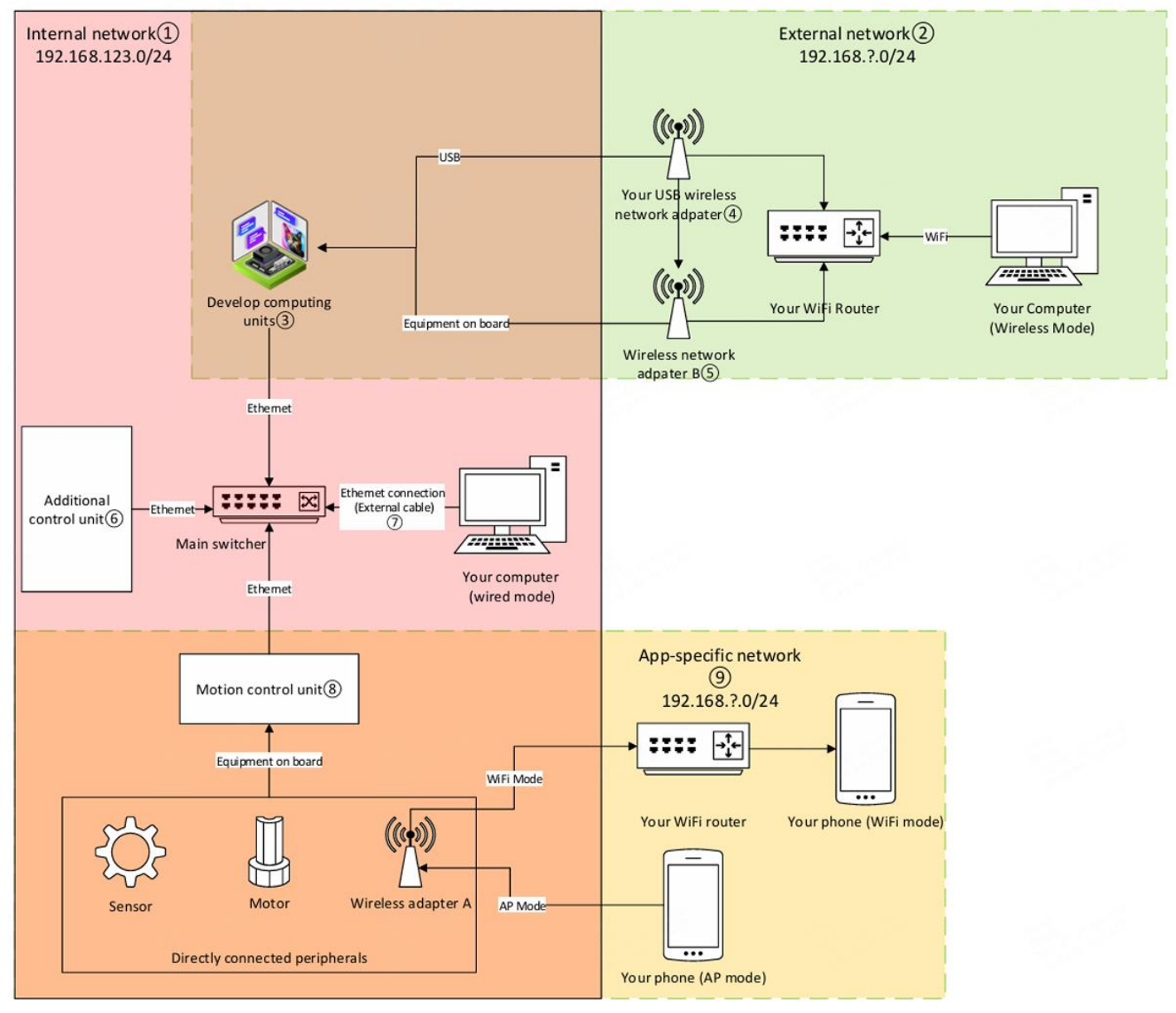
The network configuration of virtual machines is complex, and the intermediate virtual bridge can lead to **poor multicast/broadcast penetration**, especially with WSL often reporting issues like **ROS2 (DDS) topic loss and packet drops**. Running virtual machines on macOS/Windows can also introduce the host machine's own network environment as an additional source of interference.

Software Training

101 Connection Platform



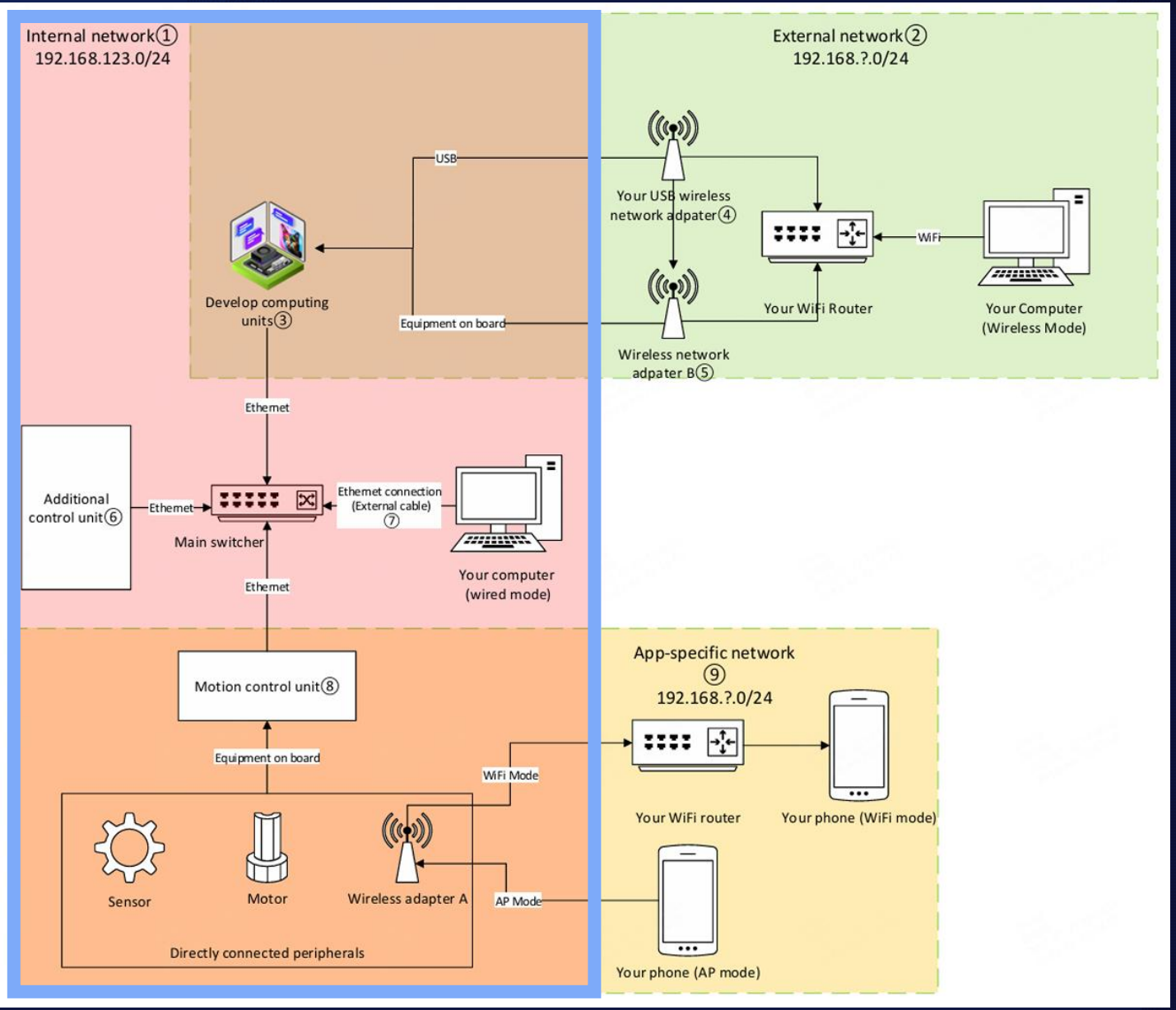
101 Connection Platform



Wired Connection

- To understand the networks inside the robot, we made a machine network topology diagram.
- Hard to view the whole diagram? Scan the QR code to view on your device.

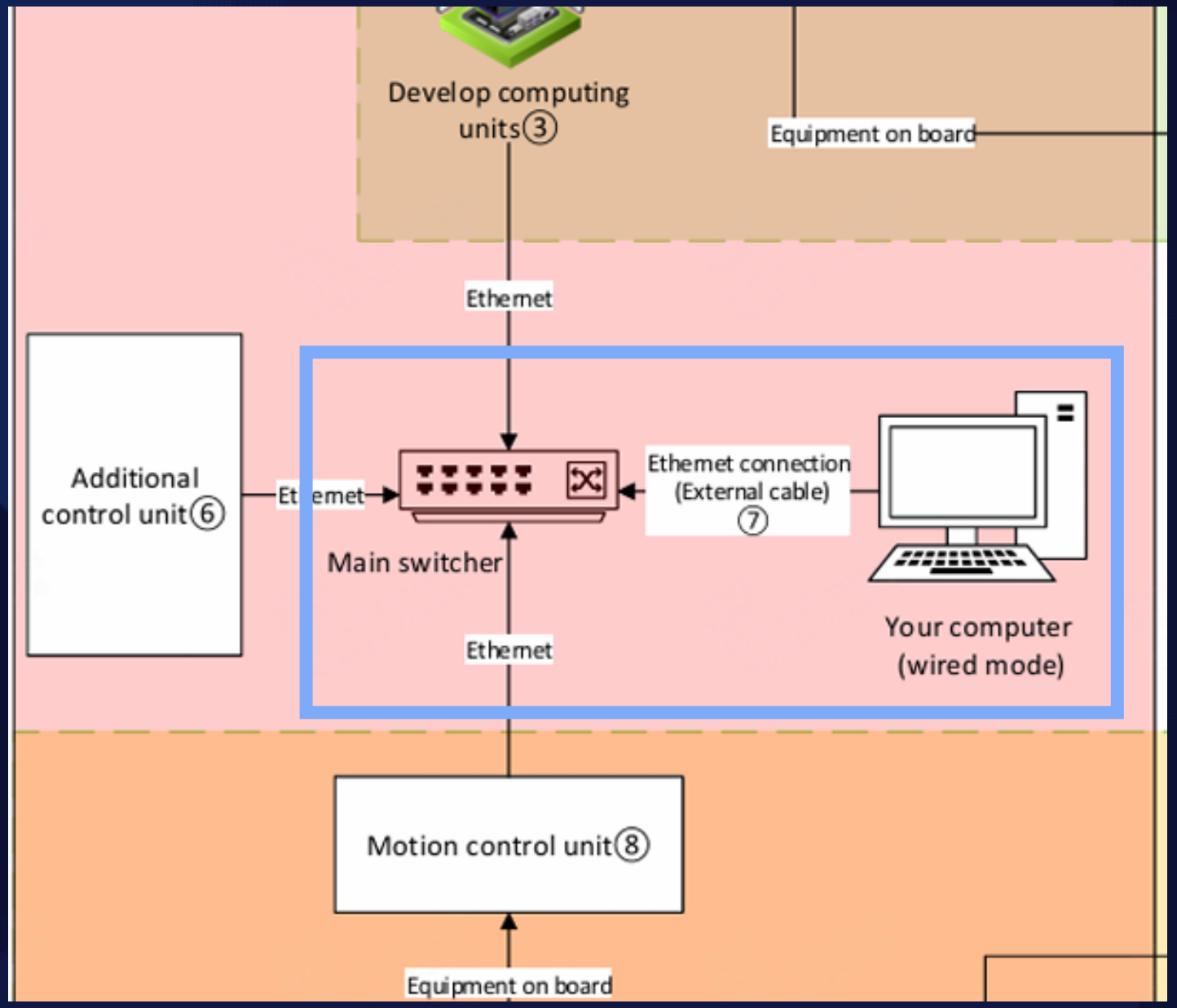
101 Connection Platform



Wired Connection

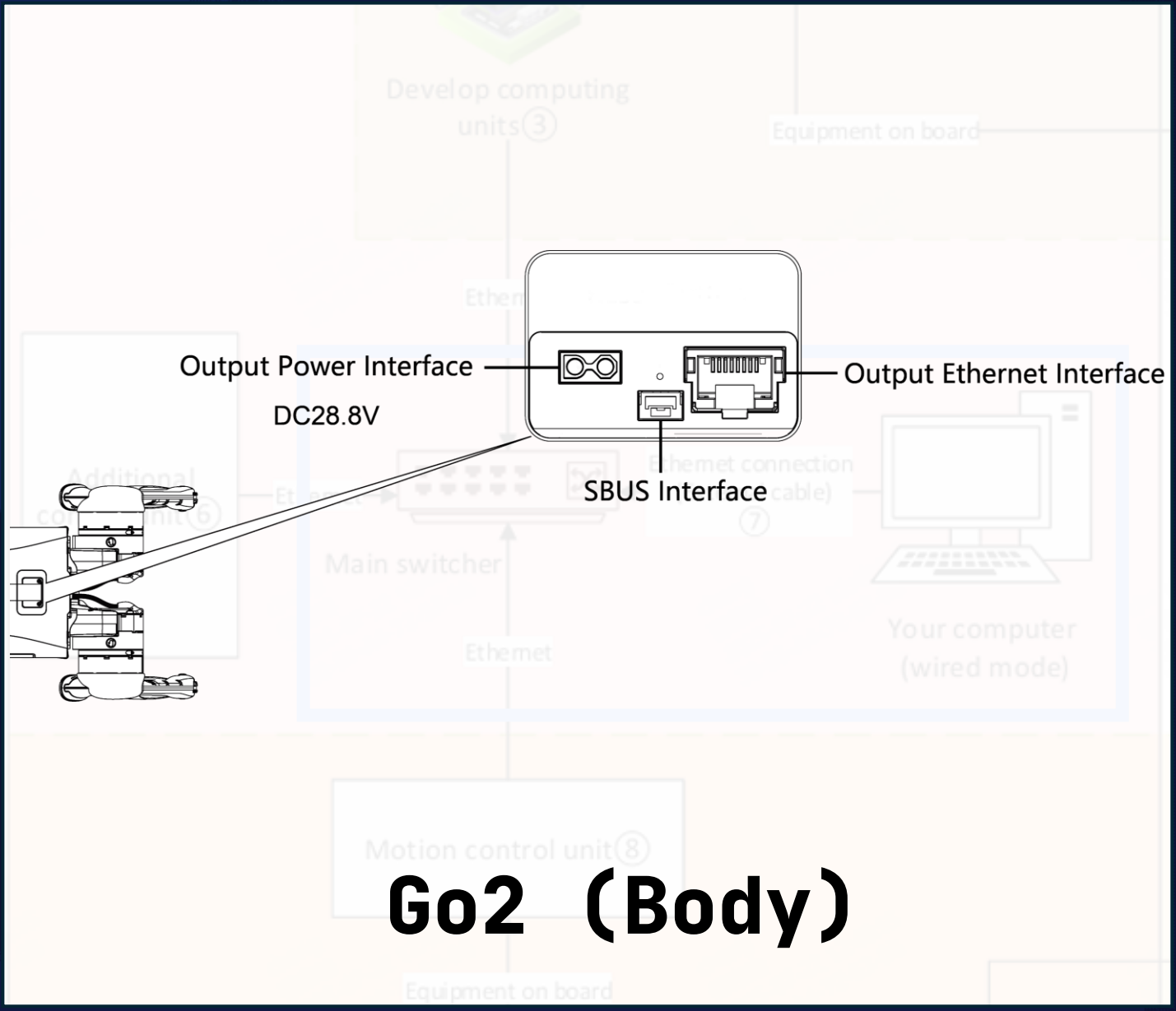
- This part is the internal network inside the robot.
- The network segment of this network is 192.168.123.0/24
- Only members in this subnetwork allows to subscribe and publish topics of robots.

101 Connection Platform



Wired Connection

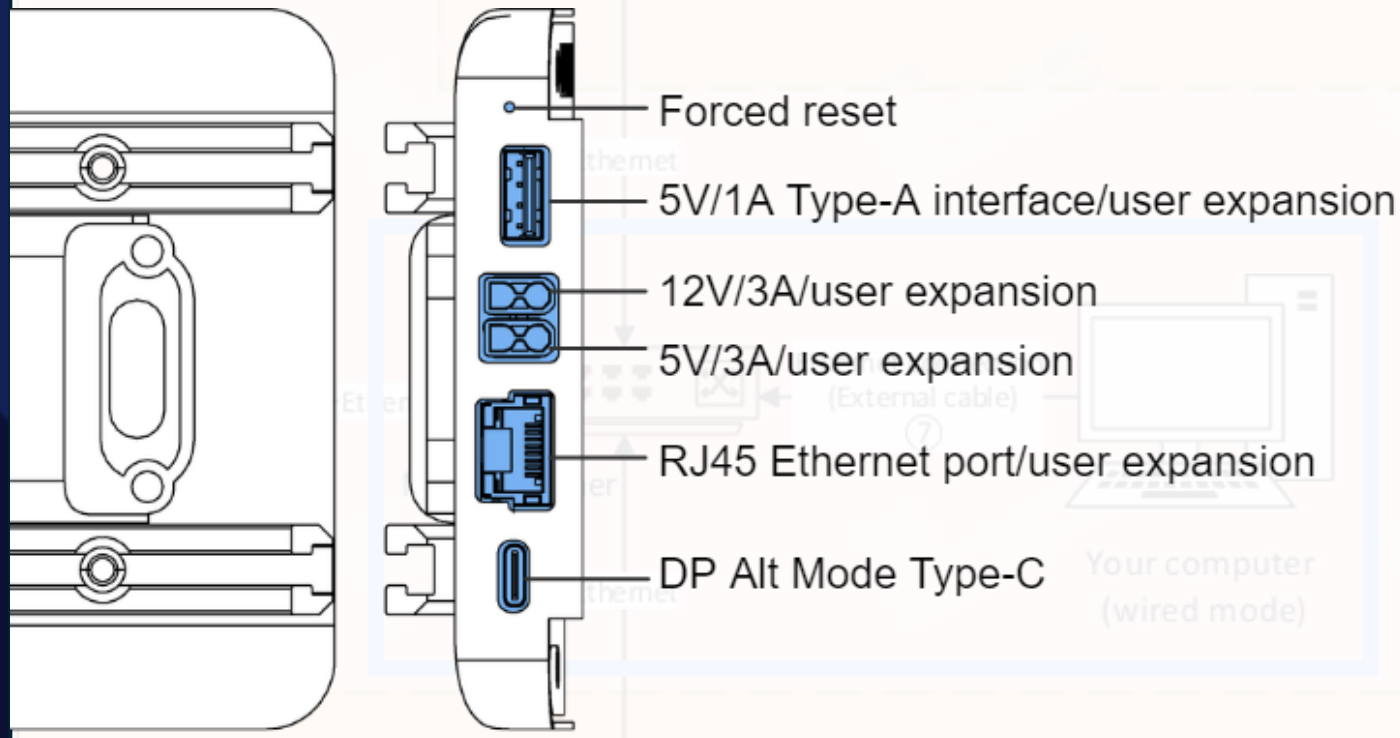
- The most convenient way to join this subnetwork is directly plug your device into it.
- It’s clear that you need to connect to the robot via the LAN interface, signed as “Switch” on the robot.



Go2 (Body)

Wired Connection

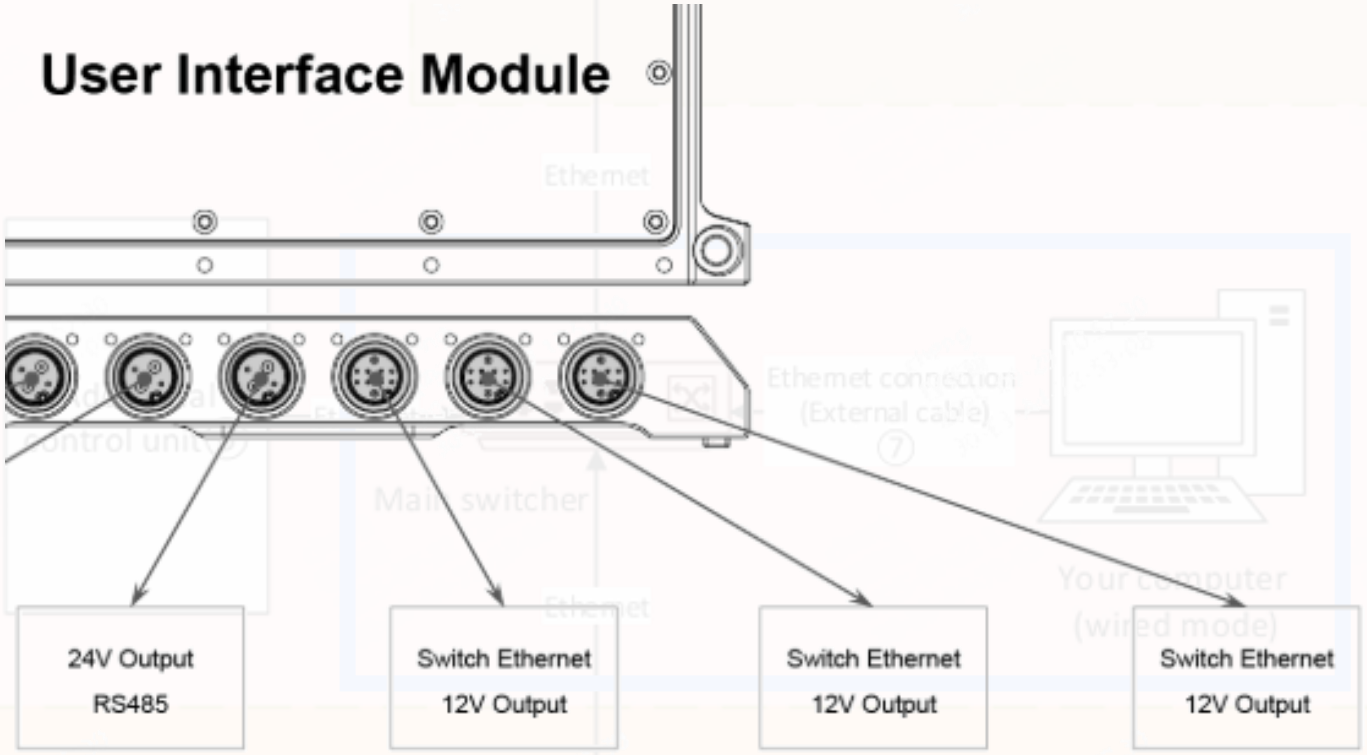
- The most convenient way to join this subnetwork is directly plug your device into it.
- It's clear that you need to connect to the robot via the LAN interface, signed a[illegible]itch" on the robot.



Docking Station

Wired Connection

- The most convenient way to join this subnetwork is directly plug your device into it.
- It’s clear that you need to connect to the robot via the LAN interface, signed as “Switch” on the robot.

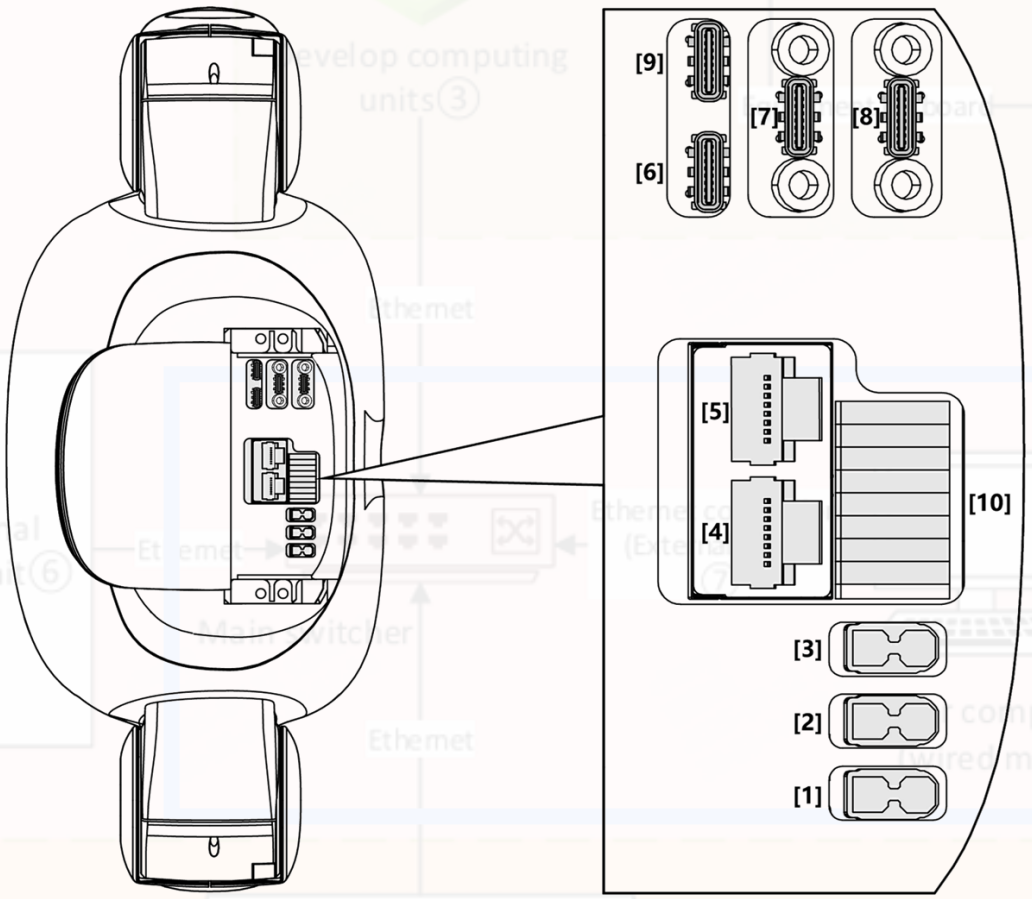


B2 User Interface

Wired Connection

- The most convenient way to join this subnetwork is directly plug your device into it.
- It’s clear that you need to connect to the robot via the LAN interface, signed as “Switch” on the robot.

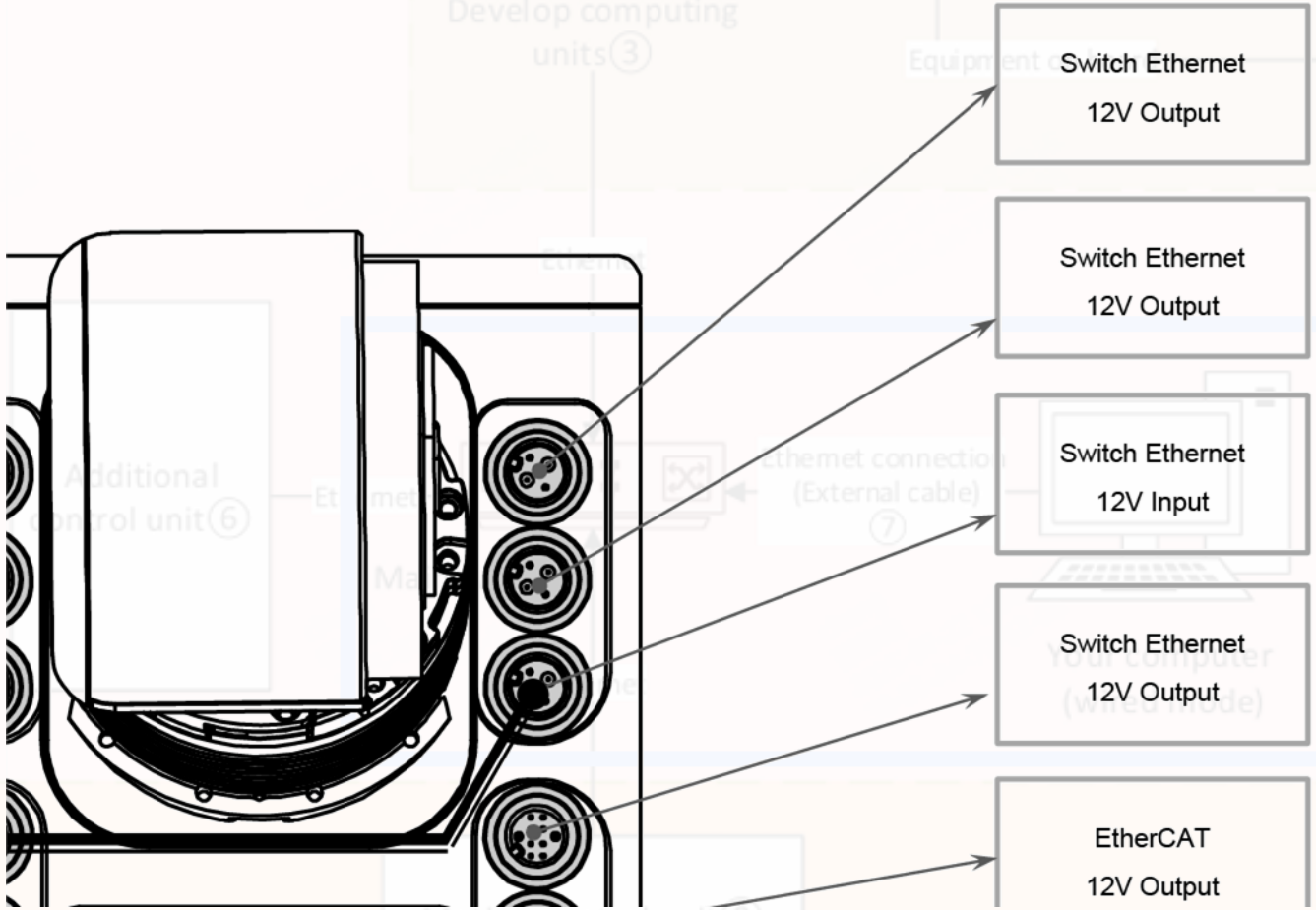
101 Connection Platform



G1

Wired Connection

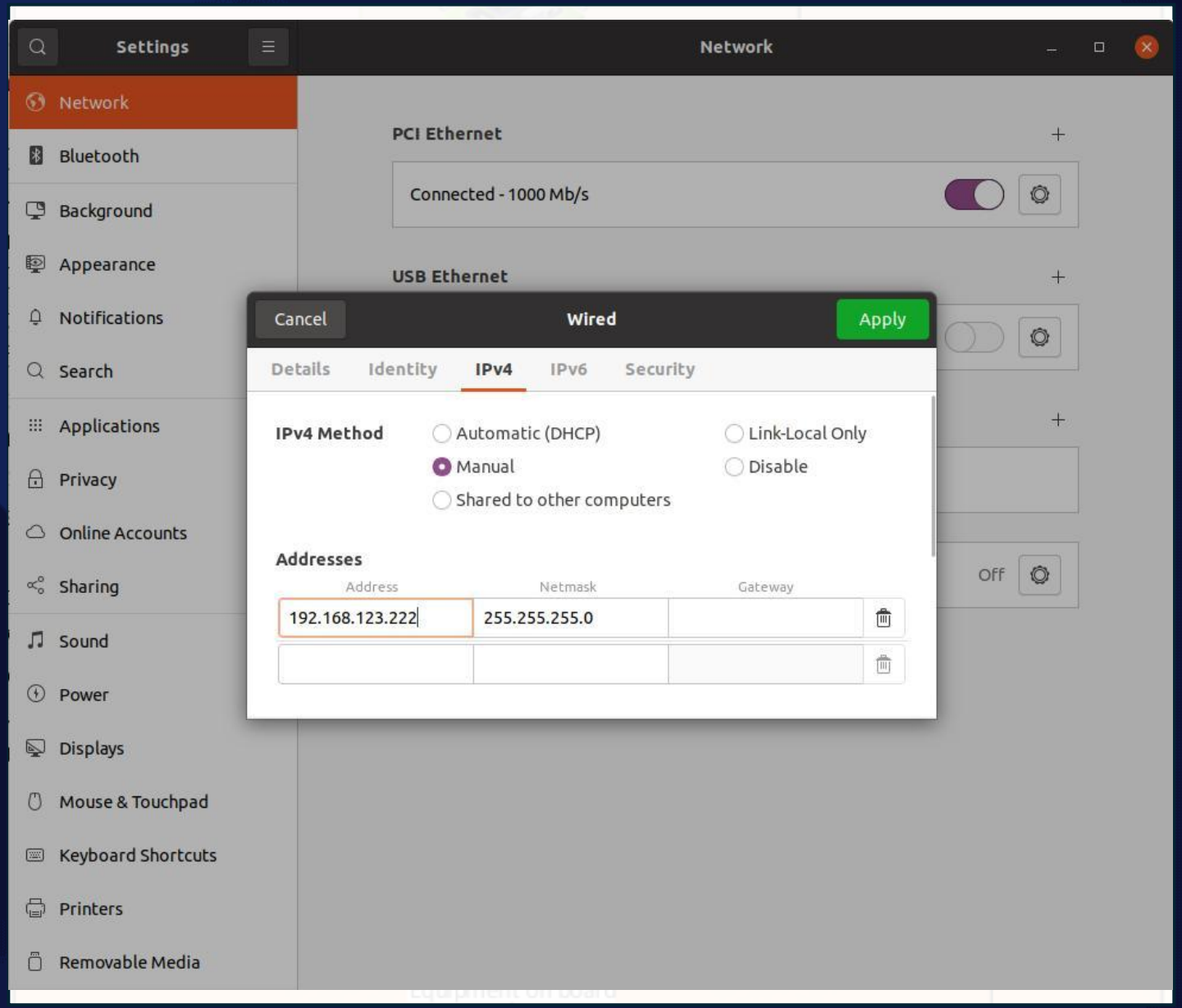
- The most convenient way to join this subnetwork is directly plug your device into it.
- It’s clear that you need to connect to the robot via the LAN interface, signed as “Switch” on the robot.



H1/H1-2

Wired Connection

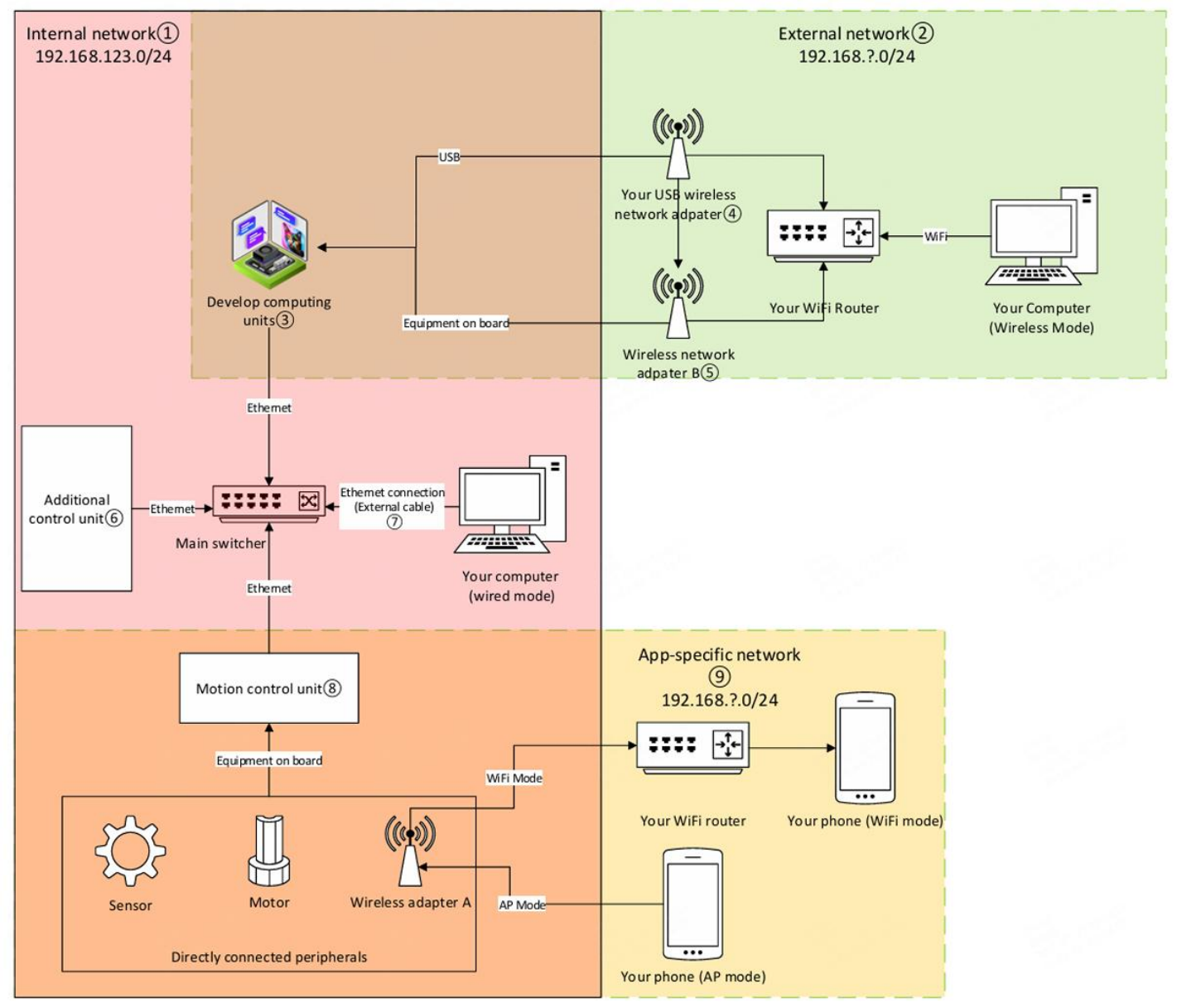
- The most convenient way to join this subnetwork is directly plug your device into it.
- It’s clear that you need to connect to the robot via the LAN interface, signed as “Switch” on the robot.



Wired Connection

- Then you need to set the IPv4 address to an address in 192.168.123.0/24.

101 Connection Platform



Wired Connection

- Then you need to set the IPv4 address to an address in 192.168.123.0/24.
- But avoid to use:

161
Motion Control Unite

20/120
LiDAR Sensors

18/162~166
Develop Computing Units

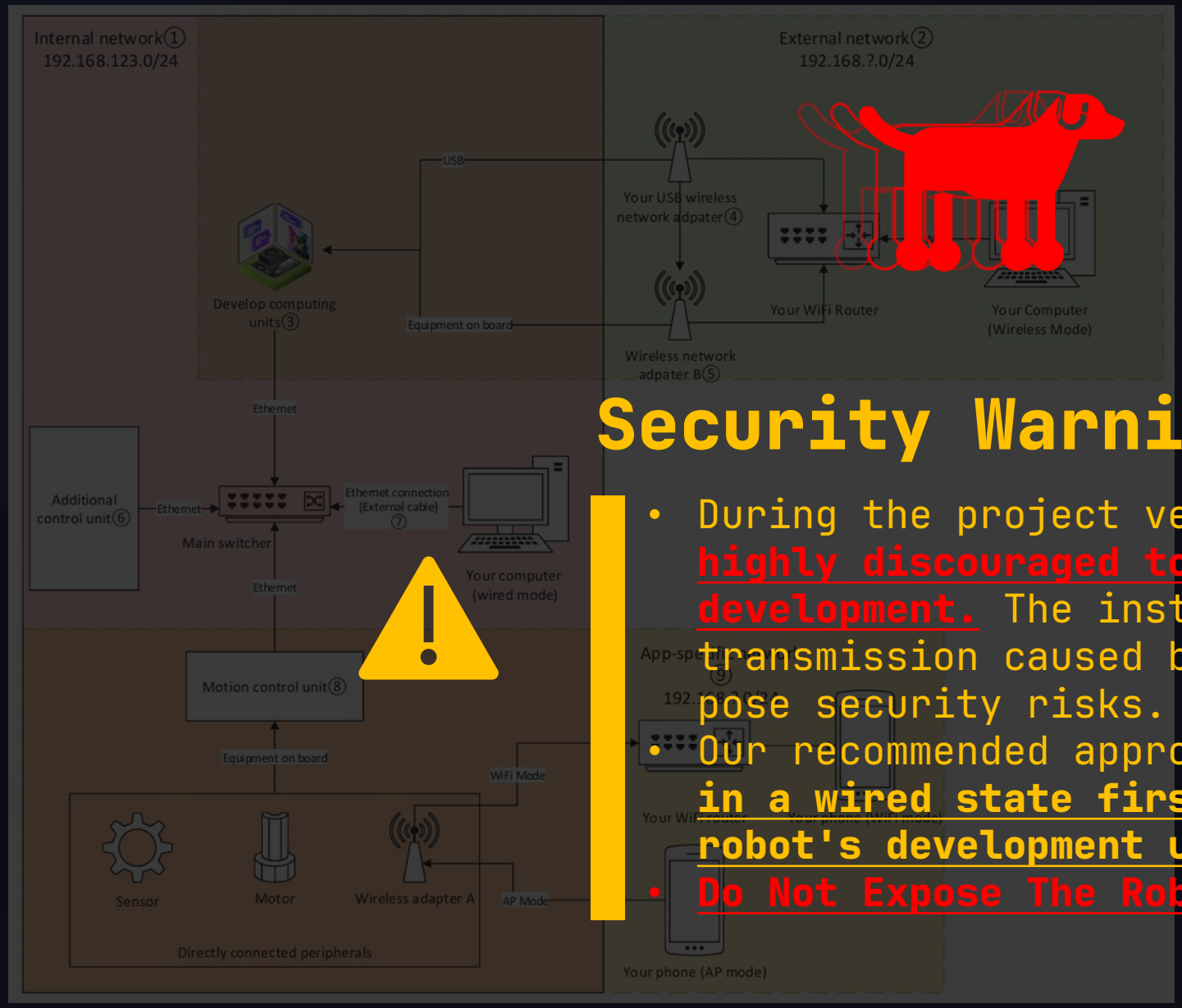
- If you take over the other devices' IP, you or the device will be set to 165.*.*.* address
- 192.168.123.222 always works.

LiDAR on Robots

Robot	LiDAR	IP Address	Note
Go2/Go2W	Mid 360 & HESAI XT 16	192.168.123.20	/
	Unitree L2	None	Not Connect To Ethernet Use Topic to Access
B2/B2W	RoboSense Helios 32	None	Not Connect To Ethernet Use Topic to Access
G1	Mid 360	192.168.123.120	/
H1		192.168.123.120	For old versions without PC2 (192.168.123.162)
		192.168.124.20	For new versions without PC2 (192.168.123.162)
H1-2		192.168.123.120	For old versions with 12th Intel CPU PC4
		192.168.124.20	For new versions with 13th Intel CPU PC4

PCs on Robots

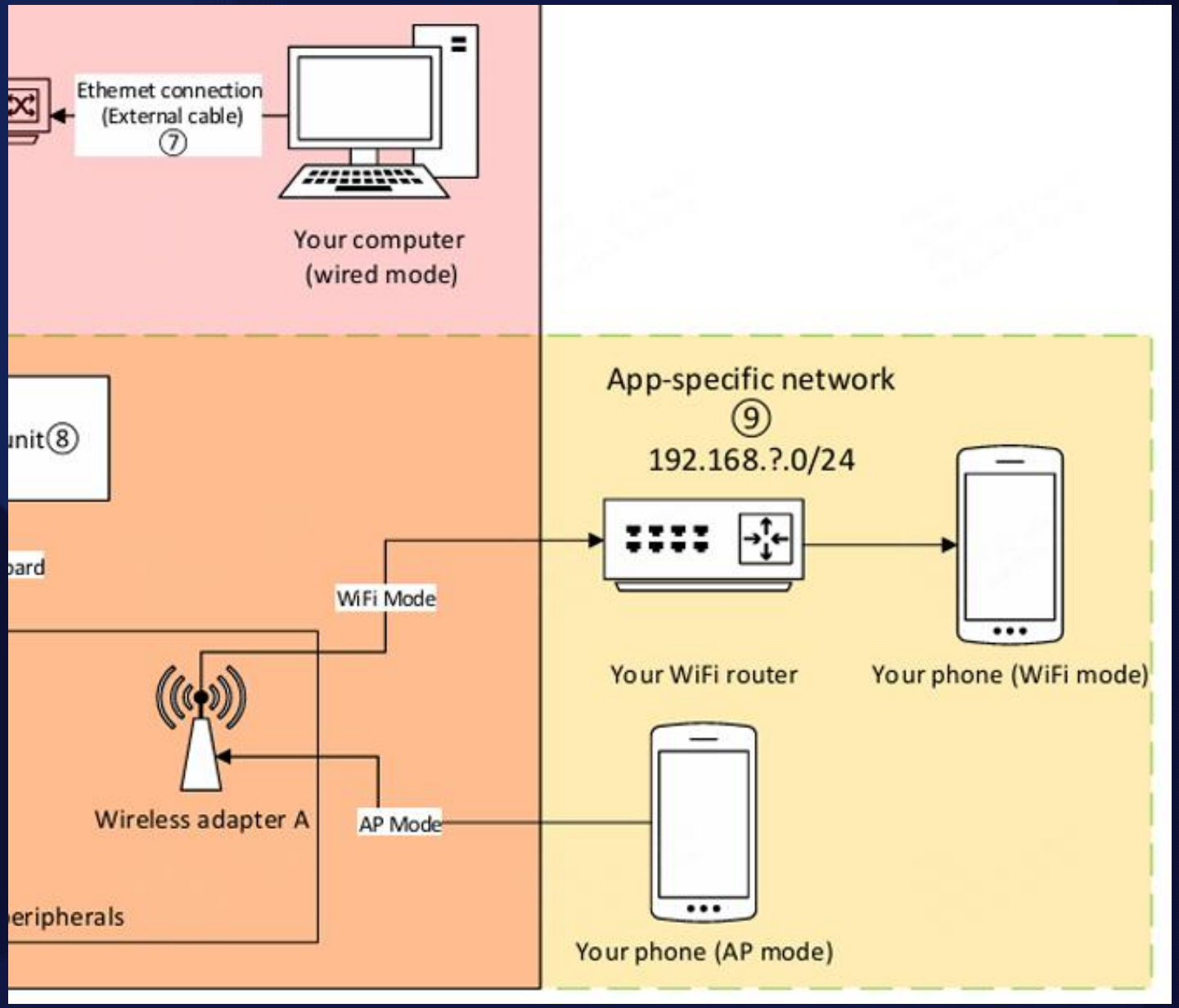
Robot	PC/Alias	IP Address	Model	Note
General	Motion Controller/PC1	192.168.123.161	/	
Go2&Go2W	Docking Station/PC2	192.168.123.18	Jetson Orin NX/Nano	Not available for Go2X
G1	Developing Unit/PC4	192.168.123.164	Jetson Orin NX	
B2/B2W/H1/H1-2	Developing Unit/PC4	192.168.123.164	13th i7	Some old version is 12th i7
	SLAM Unit/PC2	192.168.123.162	13th i7	Included in SLAM kit Not for H1-2
	Extra Develop Unite/PC3~6	192.168.123.163~166	13th i7 or Jetson Orin NX/Nano	Optional



Security Warning

- During the project verification phase, **it is highly discouraged to use wireless mode for development.** The instability of information transmission caused by network fluctuations can pose security risks.
- Our recommended approach is to **verify the program in a wired state first, then deploy it to the robot's development unit.**
- **Do Not Expose The Robot To The Public Internet.**

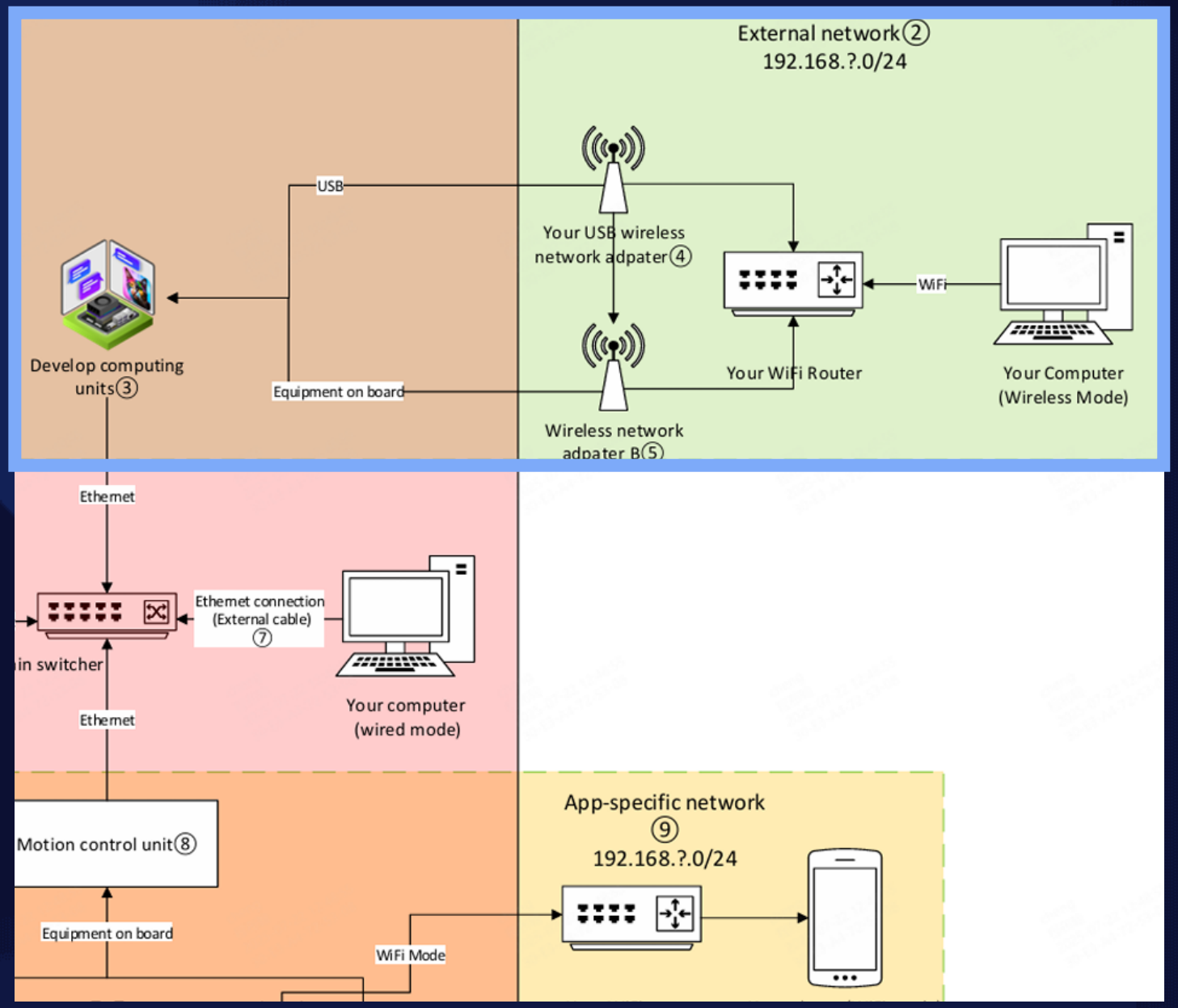
101 Connection Platform



Wireless Connection

- First, we should focus on a wrong method, which confused many users why it won't work.
- It's the App-specific network.
- Basically, when you set the robot as AP mode, the robot create a AP node. If you set it as WiFi mode, it joins the AP from router. But both are not enabled to control the robot, different from old versions like Go1.
- For safety reasons, we have implemented physical isolation.

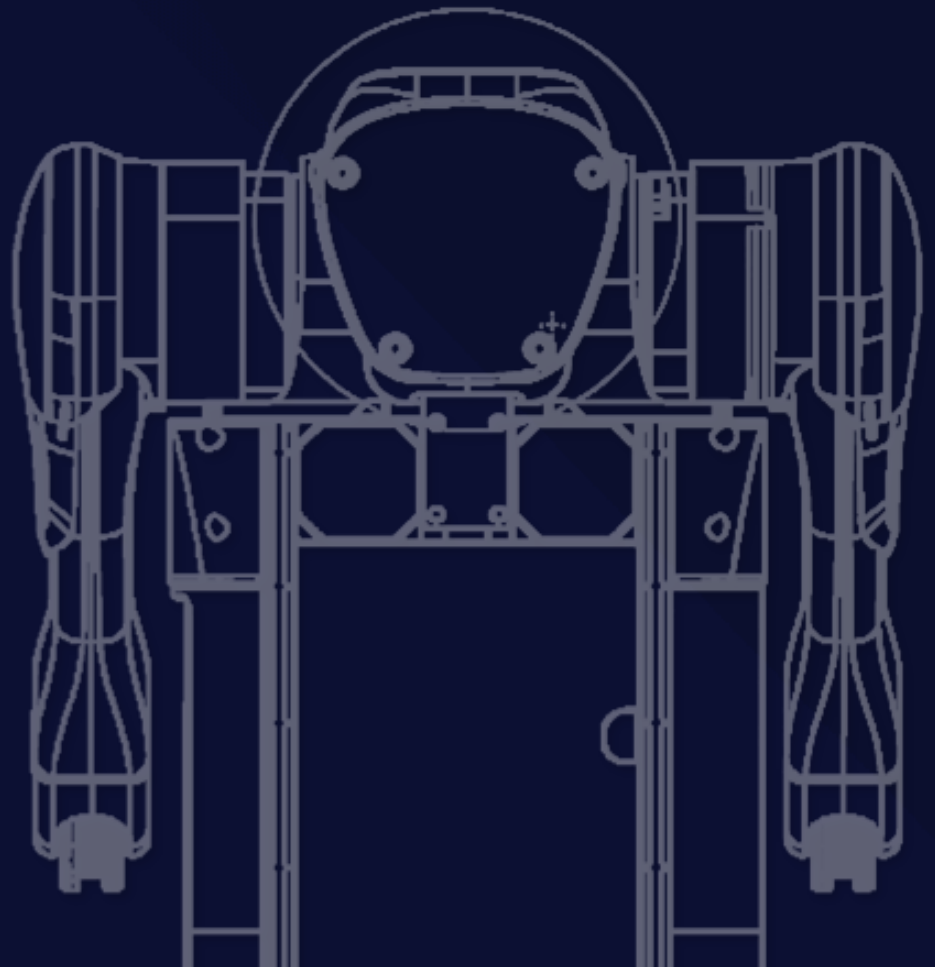
101 Connection Platform



Wireless Connection

- You can connect to the develop computing units wirelessly.
- With the built-in wireless adapter (G1) or a plug-and-play USB wireless adapter, the develop computing unit will be able to connect to the network wirelessly with a new IP address and external network access.
- Join the develop computing unit and your device together to an external network, you're able to access the develop computing unit remotely via SSH.

102 Software Platform



Unitree SDKs are published on GitHub

Unitree SDK2

- Language: CPP
- Highest Performance
- All functions support
- Update with firmware
- Recommend for high level application.

Unitree SDK2 Python

- Language: Python
- Academic research applicable
- Most functions support
- Update after SDK2
- Recommend for fast prototype verification.

Unitree ROS2

- Raw support for topics
- For ecosystem combination
- Handle Program Updates slow. Need manual adaptation.

Unitree SDKs Support Statues

* Data Update at 2025/10/11

	Humanoid						Quadruped							
Model	G1		H1		H1-2		Go2		Go2-W		B2		B2-W	
Level	HL	LL	HL	LL	HL	LL	HL	LL	HL	LL	HL	LL	HL	LL
SDK2	√	√	√	√	√	√	√	√	√	√	√	√	√	√
SDK2 Python	√	√	√	√	×	√	√	√	√	√	√	√	√	√
ROS2	√	√	×	√	×	√	√	√	×	√	×	√	√	√

After the version and interface updates, we will immediately release the adaptation update for Unitree SDK2, followed by engineers responsible for adaptation to migrate the topic data processing program from the CPP side to Python and ROS2. The update priority for SDK2 is the highest, followed by Python, and then ROS2.

Module 02
SDK Development

200 Overview SDK2

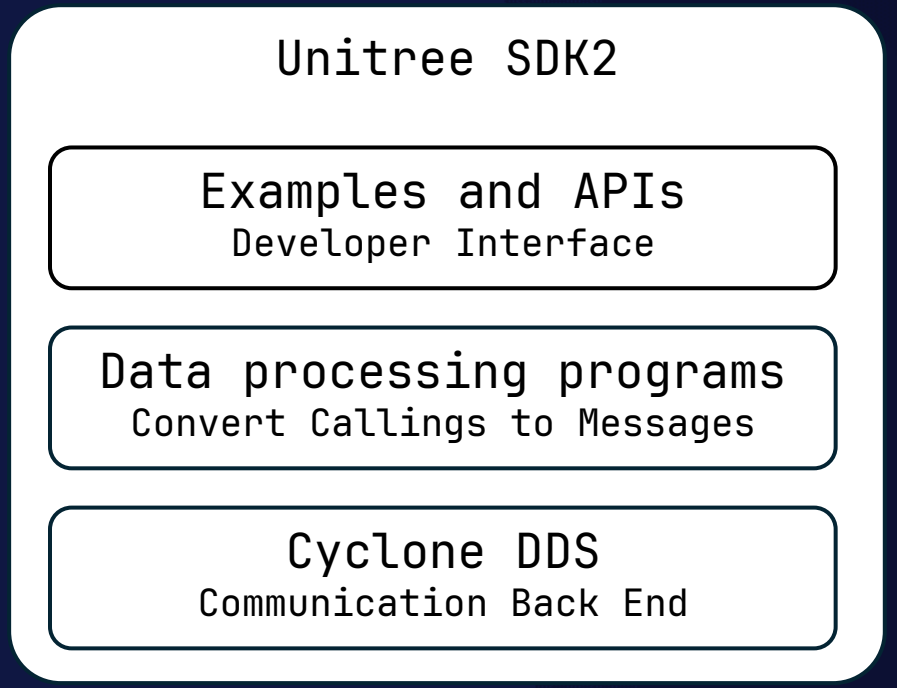
201 Conception Introduction

202H High Level Basic

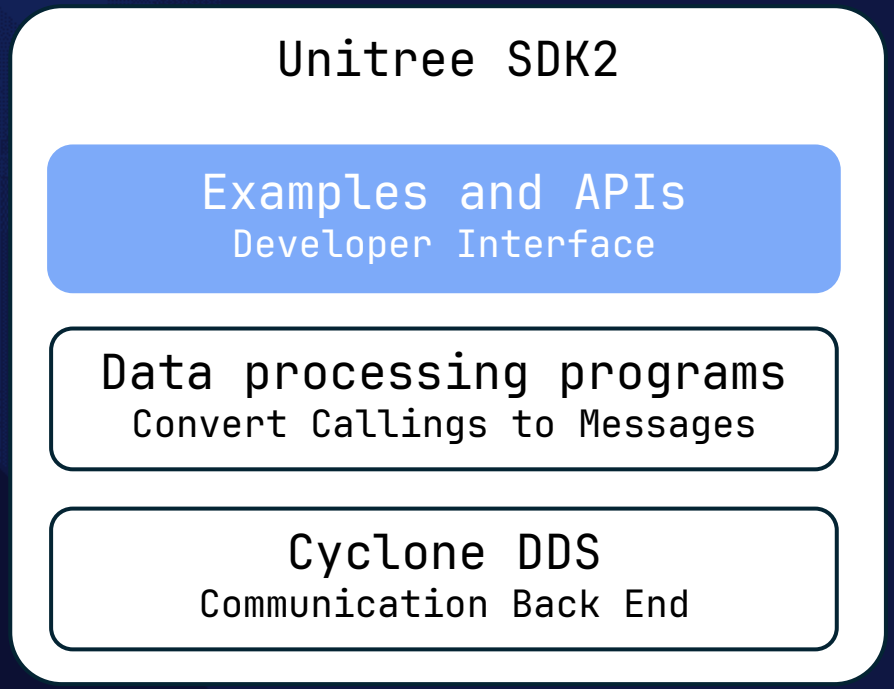
202L Low Level Basic

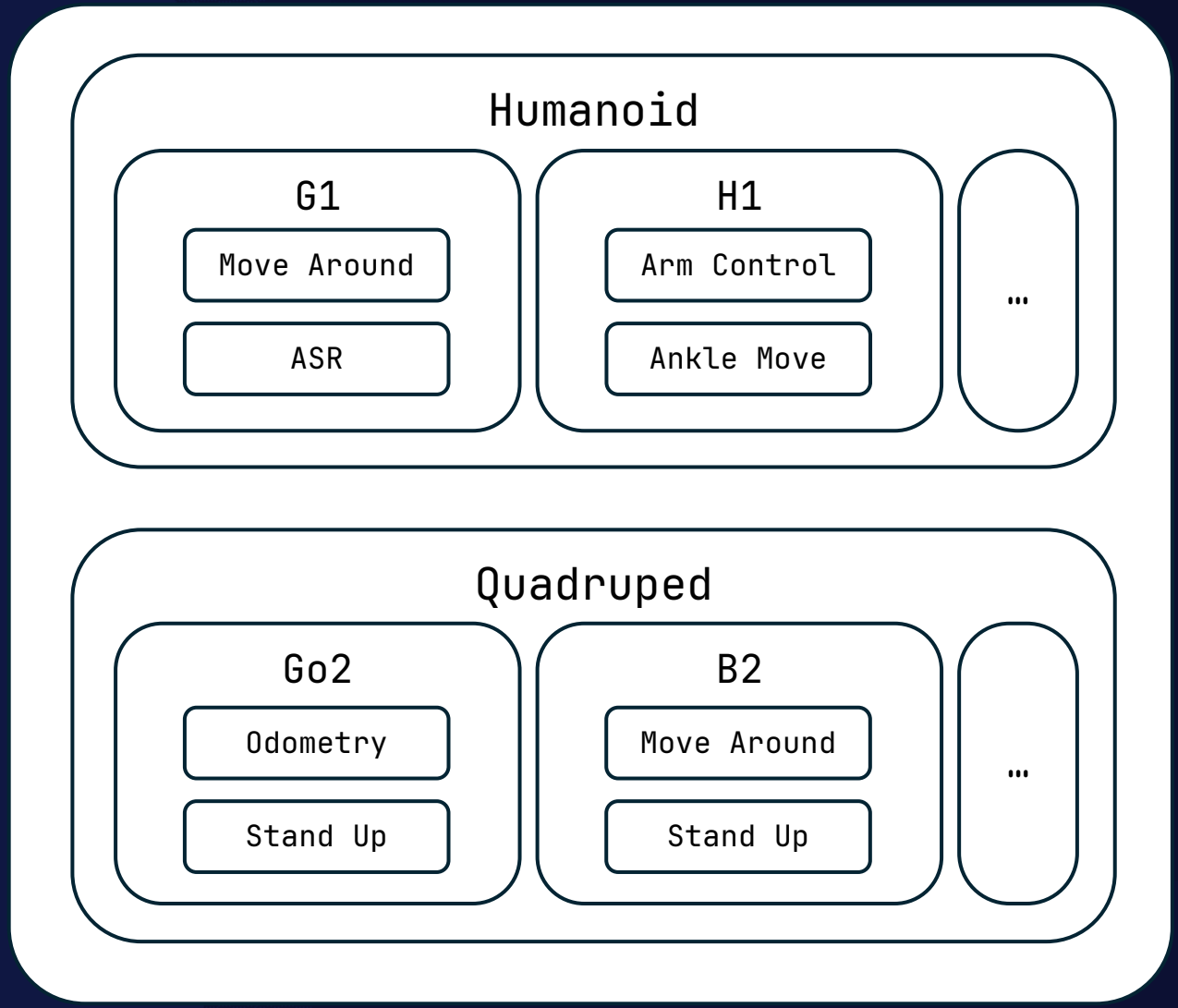
Software Training

Overview

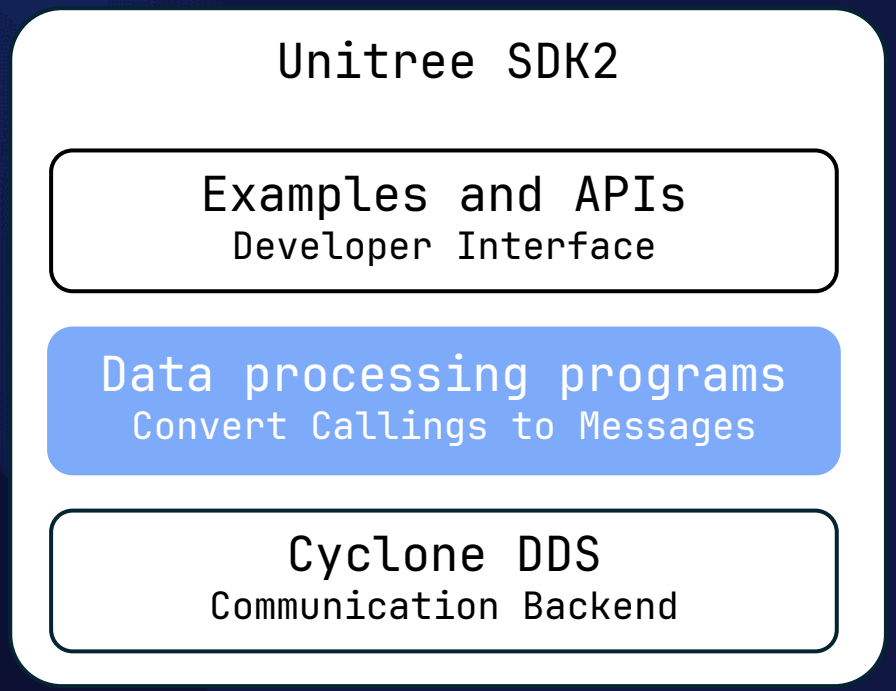

Unitree SDK2
Examples and APIs
Developer Interface
Data processing programs
Convert Callings to Messages
Cyclone DDS
Communication Back End

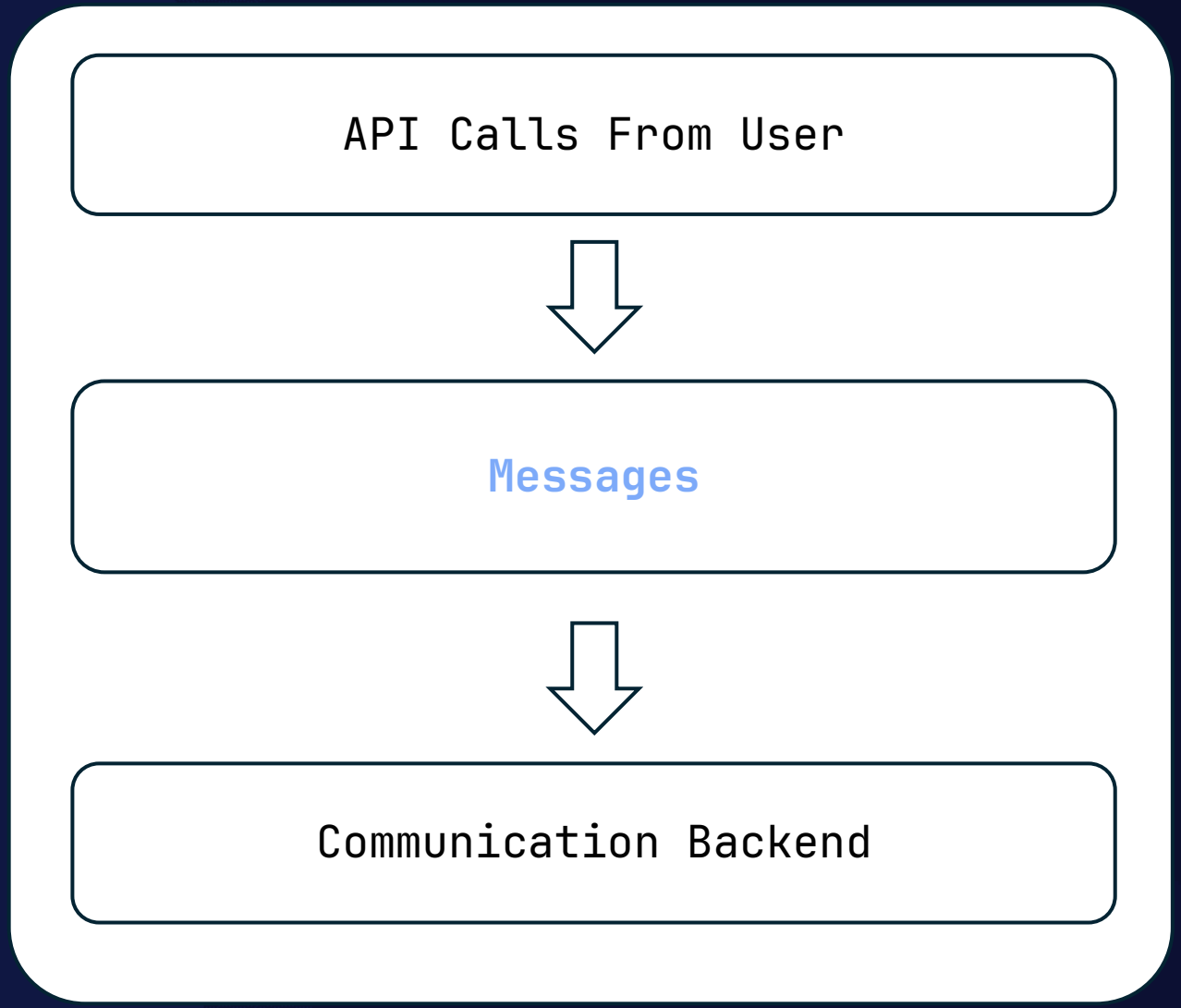
Overview


Unitree SDK2
Examples and APIs
Developer Interface
Data processing programs
Convert Callings to Messages
Cyclone DDS
Communication Back End


Humanoid
G1
Move Around
ASR
H1
Arm Control
Ankle Move
...
Quadruped
Go2
Odometry
Stand Up
B2
Move Around
Stand Up
...

Overview


Unitree SDK2
Examples and APIs
Developer Interface
Data processing programs
Convert Callings to Messages
Cyclone DDS
Communication Backend


API Calls From User
Messages
Communication Backend

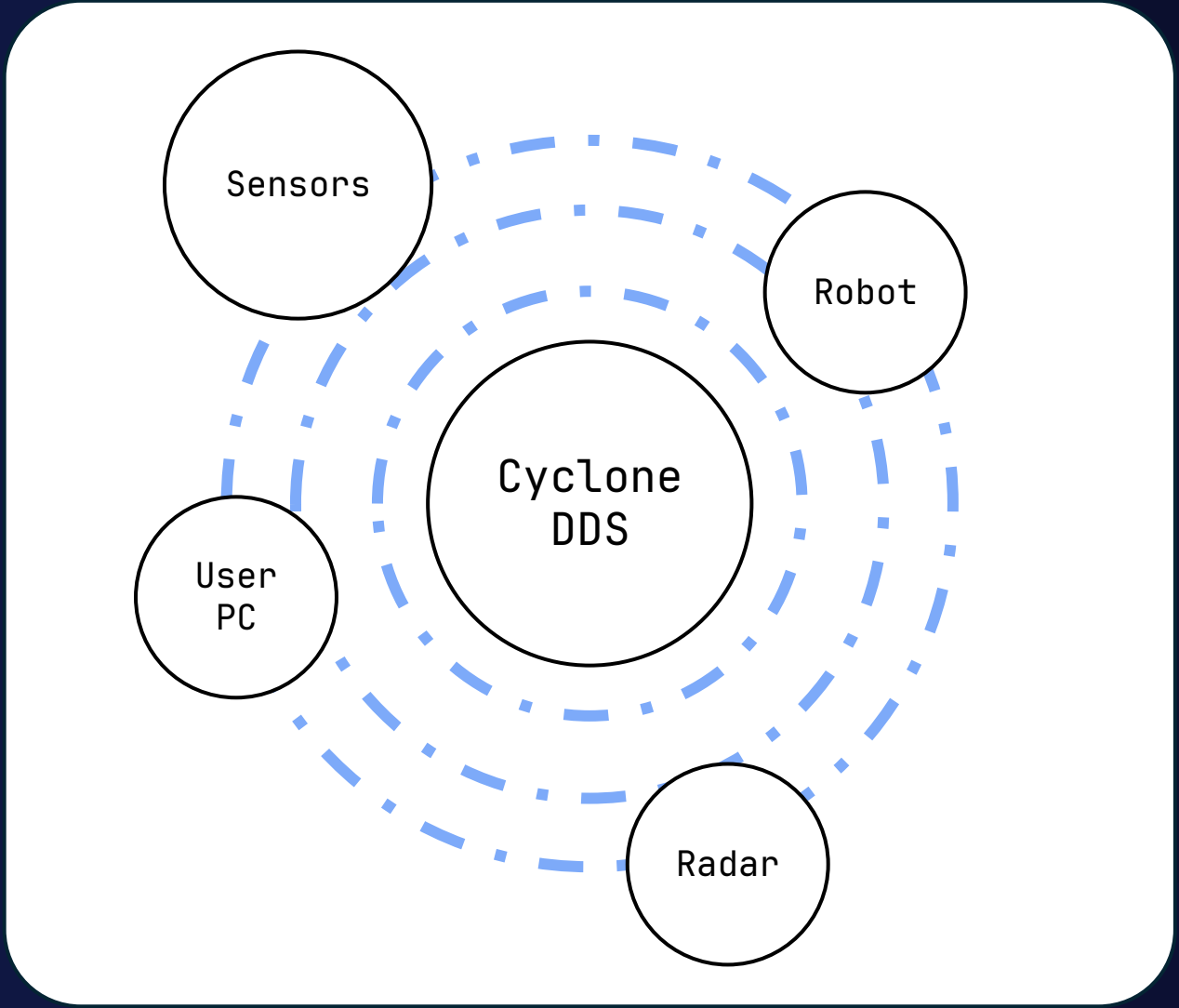
Overview

Unitree SDK2

Examples and APIs
Developer Interface

Data processing programs
Convert Callings to Messages

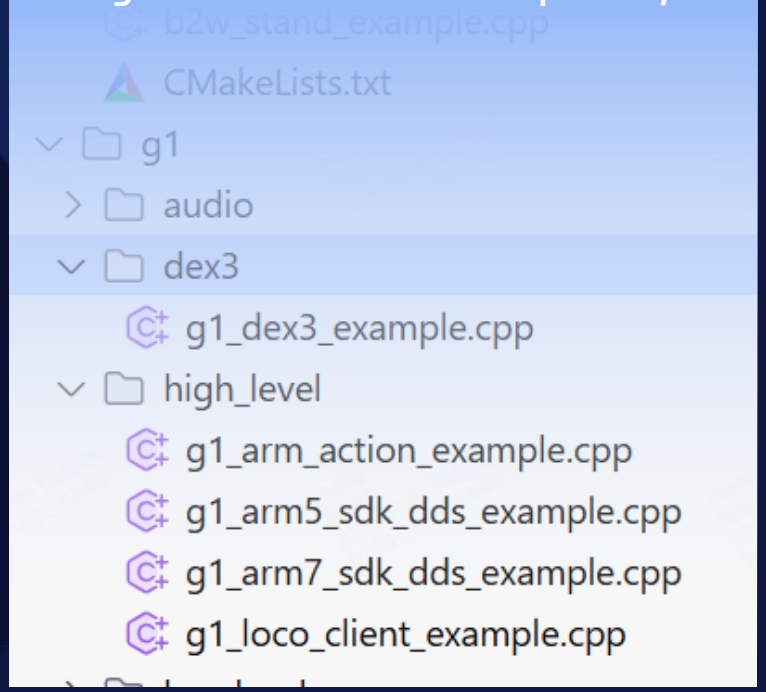
Cyclone DDS
Communication Backend



Learning Steps

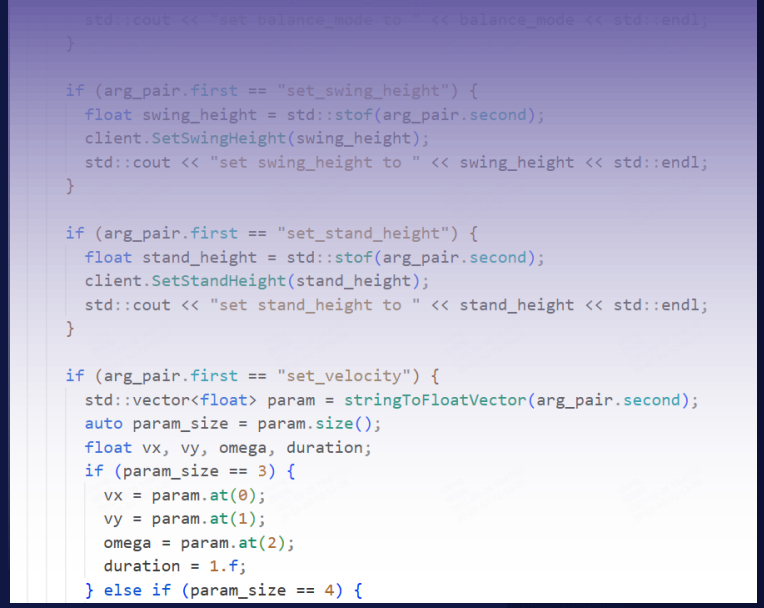
Run Examples

Understand the functional execution logic of the examples;



Try modifying routines

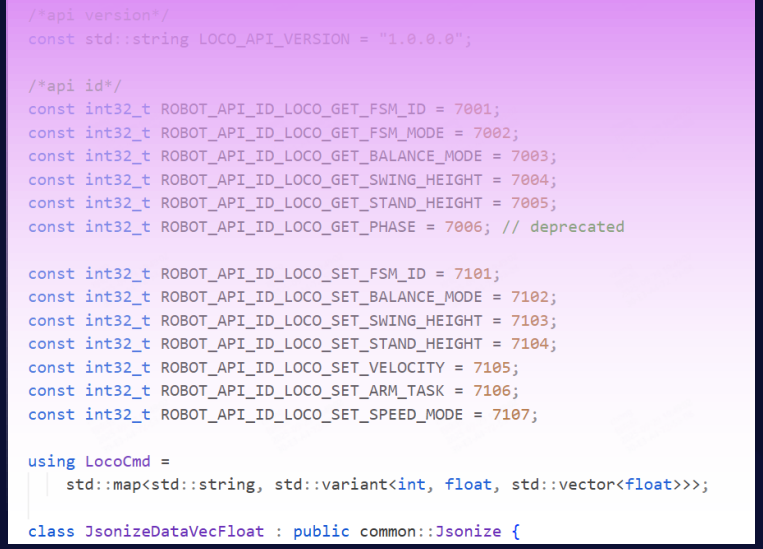
Comprehend the program logic of the routines;



```
    std::cout << "set balance_mode to " << balance_mode << std::endl;
  }

  if (arg_pair.first == "set_swing_height") {
    float swing_height = std::stof(arg_pair.second);
    client.SetSwingHeight(swing_height);
    std::cout << "set swing_height to " << swing_height << std::endl;
  }

  if (arg_pair.first == "set_stand_height") {
    float stand_height = std::stof(arg_pair.second);
    client.SetStandHeight(stand_height);
    std::cout << "set stand_height to " << stand_height << std::endl;
  }

  if (arg_pair.first == "set_velocity") {
    std::vector<float> param = stringToFloatVector(arg_pair.second);
    auto param_size = param.size();
    float vx, vy, omega, duration;
    if (param_size == 3) {
      vx = param.at(0);
      vy = param.at(1);
      omega = param.at(2);
      duration = 1.f;
    } else if (param_size == 4) {
```

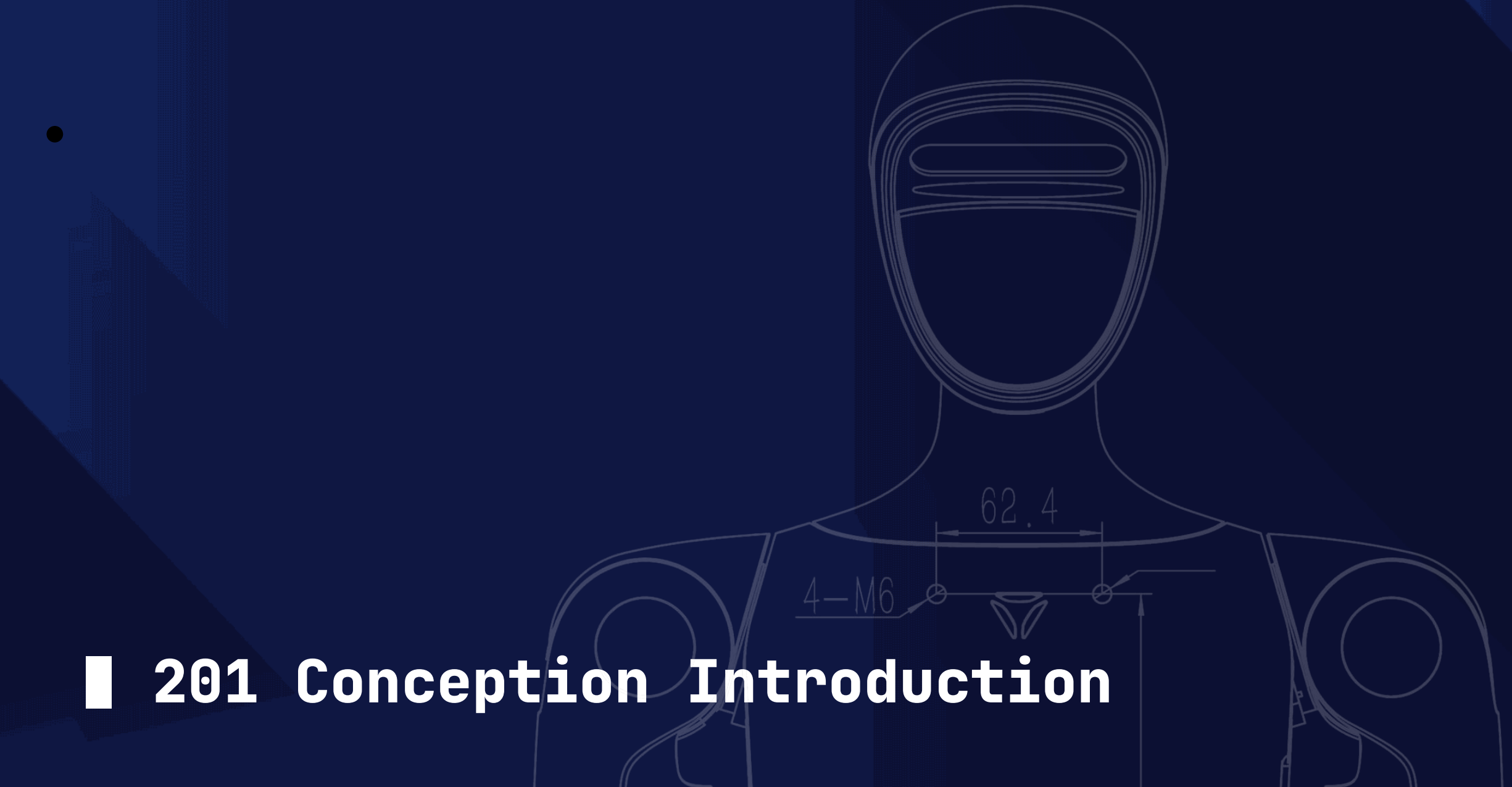
Clarify resources and usages

Master the application methods of the sections covered by the routines.



```
/*api version*/
const std::string LOCO_API_VERSION = "1.0.0.0";

/*api id*/
const int32_t ROBOT_API_ID_LOCO_GET_FSM_ID = 7001;
const int32_t ROBOT_API_ID_LOCO_GET_FSM_MODE = 7002;
const int32_t ROBOT_API_ID_LOCO_GET_BALANCE_MODE = 7003;
const int32_t ROBOT_API_ID_LOCO_GET_SWING_HEIGHT = 7004;
const int32_t ROBOT_API_ID_LOCO_GET_STAND_HEIGHT = 7005;
const int32_t ROBOT_API_ID_LOCO_GET_PHASE = 7006; // deprecated

const int32_t ROBOT_API_ID_LOCO_SET_FSM_ID = 7101;
const int32_t ROBOT_API_ID_LOCO_SET_BALANCE_MODE = 7102;
const int32_t ROBOT_API_ID_LOCO_SET_SWING_HEIGHT = 7103;
const int32_t ROBOT_API_ID_LOCO_SET_STAND_HEIGHT = 7104;
const int32_t ROBOT_API_ID_LOCO_SET_VELOCITY = 7105;
const int32_t ROBOT_API_ID_LOCO_SET_ARM_TASK = 7106;
const int32_t ROBOT_API_ID_LOCO_SET_SPEED_MODE = 7107;

using LocoCmd =
    std::map<std::string, std::variant<int, float, std::vector<float>>>;

class JsonizeDataVecFloat : public common::Jsonize {
```

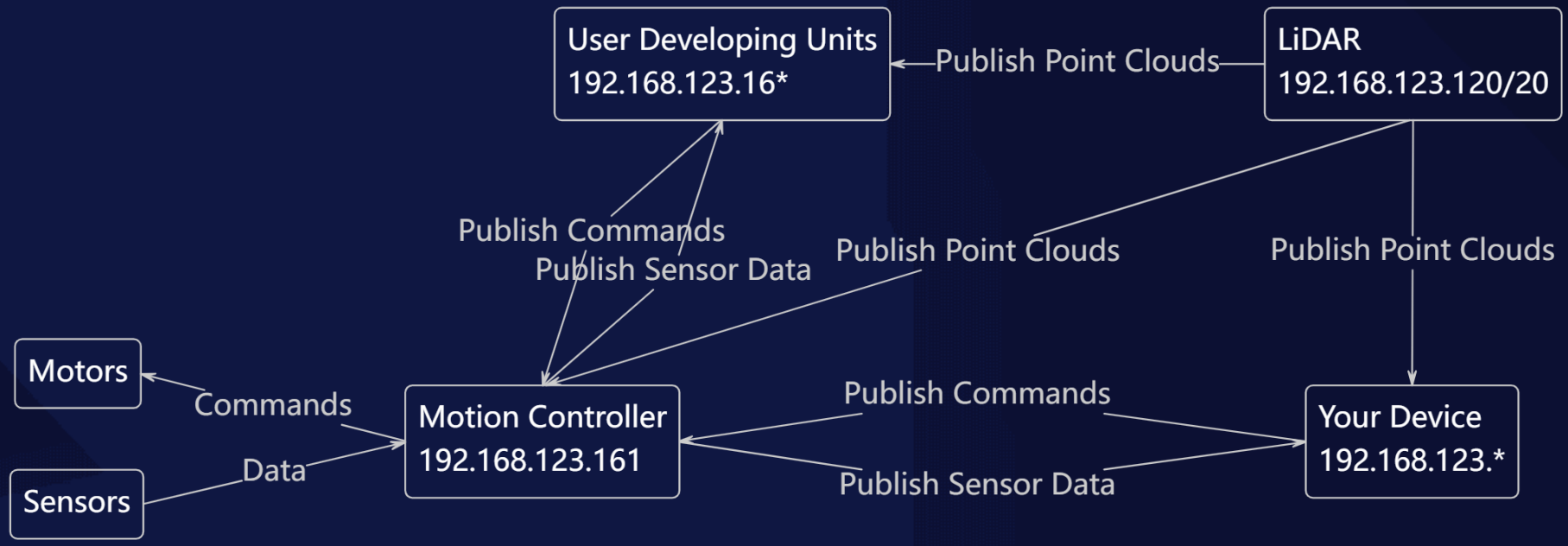
Software Training



201 Conception Introduction

Concept 01:DDS and Cyclone DDS

- Unitree SDKs are a second encapsulation on Cyclone DDS. We have designed some proprietary data types and API conventions within it.
- DDS works as a multicast network. Publishers publishing their commands or information signed with a special topic, and subscribers subscribe topics they interested in. ROS2 use DDS as back-end communication realization too.



Concept 02

High Level and Low Level

Develop using Unitree motion control capabilities.

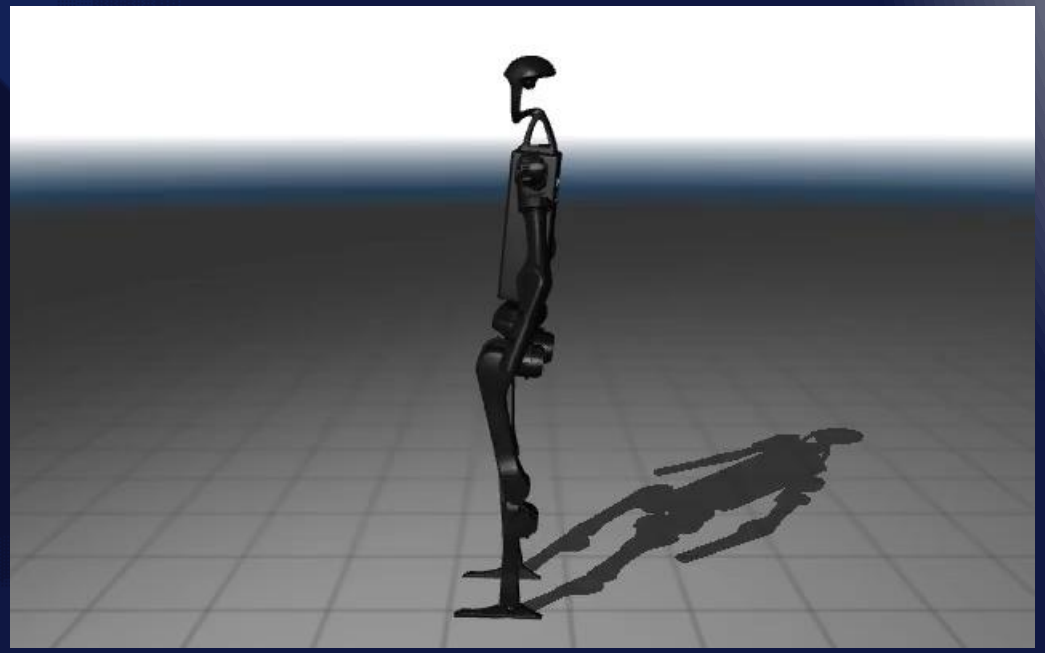
Complete tasks by orchestrating various interfaces provided by Unitree's motion control.

Example From State Grid

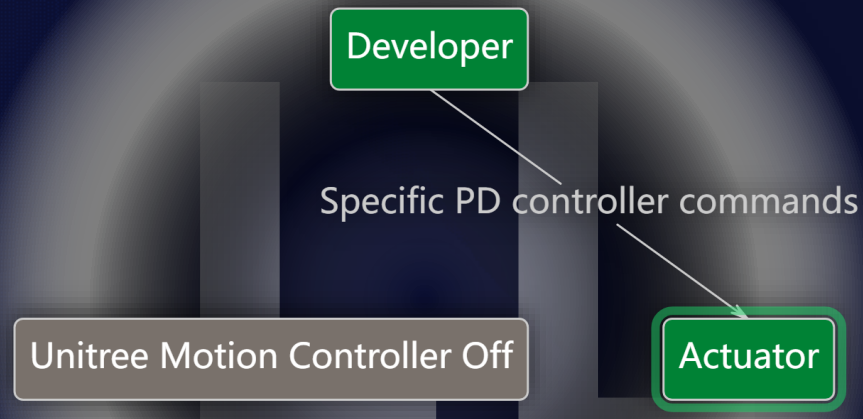
Develop without Unitree motion control capabilities

Design a robot's motion control system to achieve complex lower limb or full-body coordination tasks.

Example From Yanjie Ze

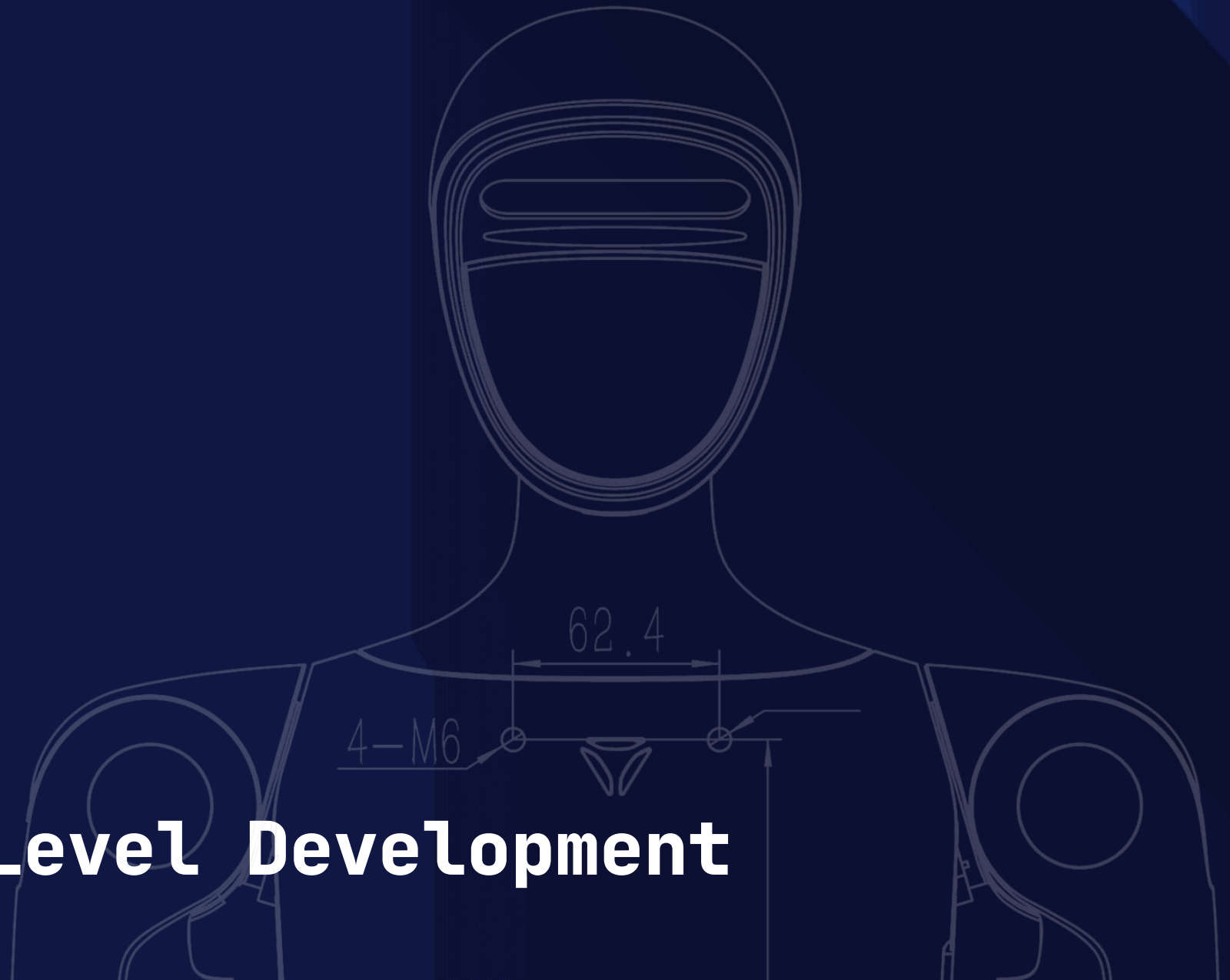


201 Conception Introduction



- The SDK is conceptually divided into two sections: High-Level (HL) and Low-Level (LL). The key difference lies in whether the Unitree motion control capability is utilized, which also corresponds to different application scenarios.

Software Training



202H High Level Development

Motion Controller

Finite-State Machine (FSM) Mode

FSM Modes

Humanoid Quadruped

General Statements

- Damping
- Zero Torque
- Debug

Special Statements

Humanoid	Quadruped
Lock Standing	Stand Down
Sit-Stand	Recovery Stand
Square-Stand	Balance Stand
…	…

Gait

Humanoid	Quadruped
Classic	
Walk/Run	Free Walk
Dance	Hand Stand
…	…

Motion Controller

Finite-State Machine (FSM) Mode

FSM Modes

Humanoid Quadruped

General Statements

- Damping
- Zero Torque
- Debug

Special Statements

- Lock Standing
- Stand Down
- Sit Stan
- Squar Stan
- …
- …

Gait

- Classic
- …
- …

Although the debugging mode conceptually belongs to a type of state machine, but in program implementation, it directly disables the robot's motion control system for low-level development.

Motion Controller

RPC Client

RPC Clients

Humanoid Quadruped

Loco Client Motion Controller	**Vui Client** Multimedia and Light	**Sport Client** Motion Controller
Audio Client TTS, ASR and Speaker	**Motion Switcher Client** Motion Controller Manager	**Obstacles Avoid Client** Obstacles Avoid
Arm Action Client Arm Actions	**Robot State Client** Robot Services Manager	**Video Client** Camera Service

Motion Controller

RPC Client

Call the Client of the specified function in the SDK

Post-processing program fits into DDS message in Json

API ID	Parameters

Motion Controller

RPC Client

API ID
7105
Set Velocity

Parameters
float vx, float vy, float omega, float duration = 1.f

```
(base) supportcbh@supportcbh-MDF-XX:~/Desktop/unitree_sdk2/build/bin$ ./g1_loco_client --network_interface=enx00e04c36021f --move="0 0 1.5"
Processing command: [move] with param: [0 0 1.5] ...
Done!
Processing command: [network_interface] with param: [enx00e04c36021f] ...
(base) supportcbh@supportcbh-MDF-XX:~/Desktop/unitree_sdk2/build/bin$ ./g1_loco_client --network_interface=enx00e04c36021f --move="0 0 -1.5"
Processing command: [move] with param: [0 0 -1.5] ...
Done!
Processing command: [network_interface] with param: [enx00e04c36021f] ...
(base) supportcbh@supportcbh-MDF-XX:~/Desktop/unitree_sdk2/build/bin$ ./g1_loco_client --network_interface=enx00e04c36021f --move="0 0 -1.5"
Processing command: [move] with param: [0 0 -1.5] ...
Done!
Processing command: [network_interface] with param: [enx00e04c36021f] ...
(base) supportcbh@supportcbh-MDF-XX:~/Desktop/unitree_sdk2/build/bin$ ./g1_loco_client --network_interface=enx00e04c36021f --move="0 0 1.5"
Processing command: [move] with param: [0 0 1.5] ...
Done!
Processing command: [network_interface] with param: [enx00e04c36021f] ...
(base) supportcbh@supportcbh-MDF-XX:~/Desktop/unitree_sdk2/build/bin$
```

yecheng.shao (10 个月前)

Motion Controller

RPC Client

include\unitree\robot\[Robot Name]\[Client_Name]\[Client_Name]_client.hpp

```
197
    Trae: 解释代码 | 注释代码 | ×
198 int32_t HighStand() { return SetStandHeight(std::numeric_limits<uint32_t>::max()); }
199
    Trae: 解释代码 | 注释代码 | ×
200 int32_t LowStand() { return SetStandHeight(std::numeric_limits<uint32_t>::min()); }
201
202 int32_t Move(float vx, float vy, float vyaw, bool continous_move) {
203   return SetVelocity(vx, vy, vyaw, continous_move ? 864000.f : 1.f);
204 }
205
```

```
int32_t SetVelocity(float vx, float vy, float omega, float duration = 1.f) {
  std::string parameter, data;

  JsonizeVelocityCommand json;
  std::vector<float> velocity = {vx, vy, omega};
  json.velocity = velocity;
  json.duration = duration;
  parameter = common::ToJsonString(json);

  return Call(ROBOT_API_ID_LOCO_SET_VELOCITY, parameter, data);
}
```

Motion Controller

RPC Client

include\unitree\robot\[Robot Name]\[Client_Name]\[Client_Name]_client.hpp

```cpp
int32_t SetVelocity(float vx, float vy, float omega, float duration = 1.f) {
  std::string parameter, data;

  JsonizeVelocityCommand json;
  std::vector<float> velocity = {vx, vy, omega};
  json.velocity = velocity;
  json.duration = duration;
  parameter = common::ToJsonString(json);

  return Call(ROBOT_API_ID_LOCO_SET_VELOCITY, parameter, data);
}
```

function name	SetVelocity
Function prototype	int32_t SetVelocity(float vx, float vy, float omega, float duration = 1.f)
Function overview	Set the speed.
Parameter	vvx: forward speed; vy: horizontal speed; omega: rotation speed; duration: duration of speed command.
Return value	Returns 0 if the call is successful, otherwise returns the relevant error code.
Remarks	The program will automatically set the cropping to the allowed range.

Motion Controller

RPC Client

include\unitree\robot\[Robot Name]\[Client_Name]\[Client_Name]_api.hpp

```cpp
int32_t SetVelocity(float vx, float vy, float omega, float duration = 1.f) {
  std::string parameter, data;

  JsonizeVelocityCommand json;
  std::vector<float> velocity = {vx, vy, omega};
  json.velocity = velocity;
  json.duration = duration;
  parameter = common::ToJsonString(json);

  return Call(ROBOT_API_ID_LOCO_SET_VELOCITY, parameter, data
}
```

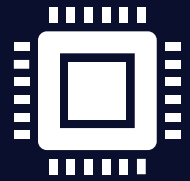
```cpp
/*api id*/
const int32_t ROBOT_API_ID_LOCO_GET_FSM_ID = 7001;
const int32_t ROBOT_API_ID_LOCO_GET_FSM_MODE = 7002;
const int32_t ROBOT_API_ID_LOCO_GET_BALANCE_MODE = 7003;
const int32_t ROBOT_API_ID_LOCO_GET_SWING_HEIGHT = 7004;
const int32_t ROBOT_API_ID_LOCO_GET_STAND_HEIGHT = 7005;
const int32_t ROBOT_API_ID_LOCO_GET_PHASE = 7006; // deprecated

const int32_t ROBOT_API_ID_LOCO_SET_FSM_ID = 7101;
const int32_t ROBOT_API_ID_LOCO_SET_BALANCE_MODE = 7102;
const int32_t ROBOT_API_ID_LOCO_SET_SWING_HEIGHT = 7103;
const int32_t ROBOT_API_ID_LOCO_SET_STAND_HEIGHT = 7104;
const int32_t ROBOT_API_ID_LOCO_SET_VELOCITY = 7105;
const int32_t ROBOT_API_ID_LOCO_SET_ARM_TASK = 7106;
const int32_t ROBOT_API_ID_LOCO_SET_SPEED_MODE = 7107;
```

Motion Controller

RPC Client

API Request send to DDS topic /api/[Client Name]/request
*Frequency in 40~60Hz, 50Hz Recommend.

API Responding formats in [API ID]+[Respond]
send to DDS topic /api/[Client Name]/response



Subscribe /api/[Client Name]/response, listen to [API ID]
In 50Hz

Motion Controller

RPC Client

API Request send to DDS topic /api/[Client Name]
*Frequency in 40~60Hz, 50Hz Recommend.

Execute Result

0	-1
Success	Failed

General Code

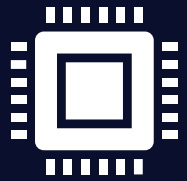
3001	3102	3103
Unknown	Request Sending	API not registered

...

Specific Code

7301	7401	4205
G1 Loco	G1 Arm	B2 Sport
Invalid fsm ID	Arm is holding	Server function not init

...



Motion Controller

RPC Client

include\unitree\robot\[Robot Name]\[Client_Name]\[Client_Name]_error.hpp

Motion Controller

RPC Client

G1

Client Name	Code	Remarks
Arm Action	7400	The topic rt/armsdk is occupied
	7401	The arm is holding. Expecting release action(99) or the same last action
		d
		ter
		available

ode	Remarks
me as B2	Same as B2
	/
me as B2	Same as B2
me as B2	Same as B2
0	Invalid parameter

	Remarks
	LocoState not available

RPC Error Code Lists

General Codes

Number	Error description	Remarks
3001	Unknown error	Client/server returned
3102	Request sending error	Client returned
3103	API not registered	Client returned
3104	Request timeout	Client returned
3105	Request and response data do not match	Client returned
3106	Invalid response data	Client returned
3107	Invalid lease	Client returned
3201	Response sending error	Occurred on the server and will not be returned to the client
3202	Internal server error	Server returned
3203	API not implemented on the server	Server returned
3204	API parameter error	Server returned
3205	Request rejected	Server returned
3206	Invalid lease	Server returned
3207	Lease already exists	Server returned
0*	Execute Successful	Server returned

For specific RPC Client error codes and their corresponding explanations, please refer to this document.

Robot States

DDS Topics

DDS Topics

Humanoid Quadruped

Audio Message ASR Recognition	Low State IMUs and Motors	Sport Mode State Odometer, Gait, etc.…
Dexterous Hand Dex1, Dex3 and Dex5	Unitree SLAM Slam and Navigation	Unitree LiDAR Front Lidar Point Cloud
GPT Flow Feedback Text Respond from OpenAI	BMS State Battery Management	UWB State Following Module
…	…	…

Robot States

DDS Topics


```idl
// generated from rosidl_generator_dds_idl/resource/idl.idl.em
// with input from unitree_arm:msg/ArmString.idl
// generated code does not contain a copyright notice

#ifndef __unitree_arm__msg__arm_string__idl__
#define __unitree_arm__msg__arm_string__idl__


module unitree_arm {

module msg {

module dds_ {


struct ArmString_ {
string data_;

};

}; // module dds_

}; // module msg

}; // module unitree_arm

#endif // __unitree_arm__msg__arm_string__idl__
```

Message Type Definition

.idl File

Robot States

DDS Topics

Message Type Definition

.idl File

Struct

Type	Parameter
Type	Parameter
…	…

Robot States

DDS Topics

Message Type Definition

.idl File

Cyclone DDS

Idlc compiler

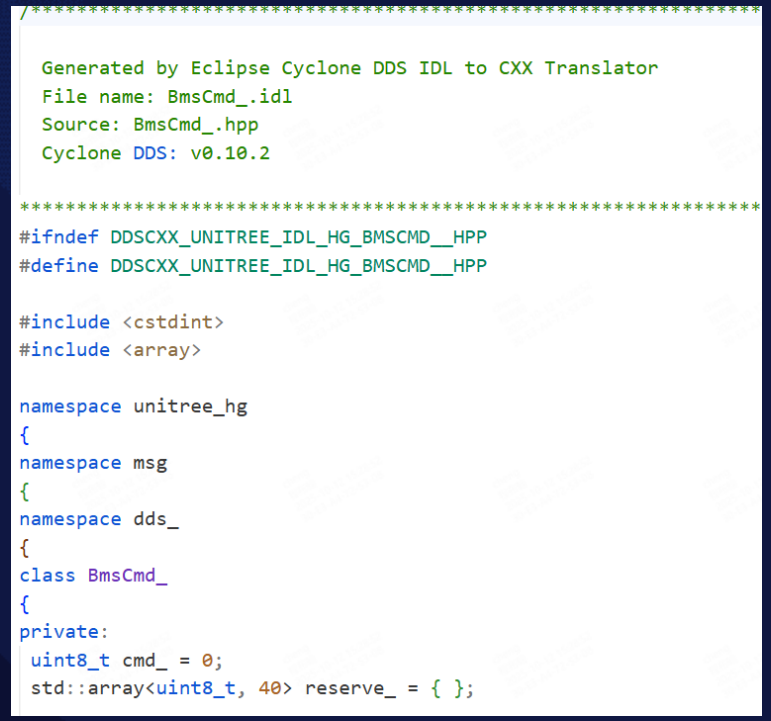
Library File

CPP
.hpp

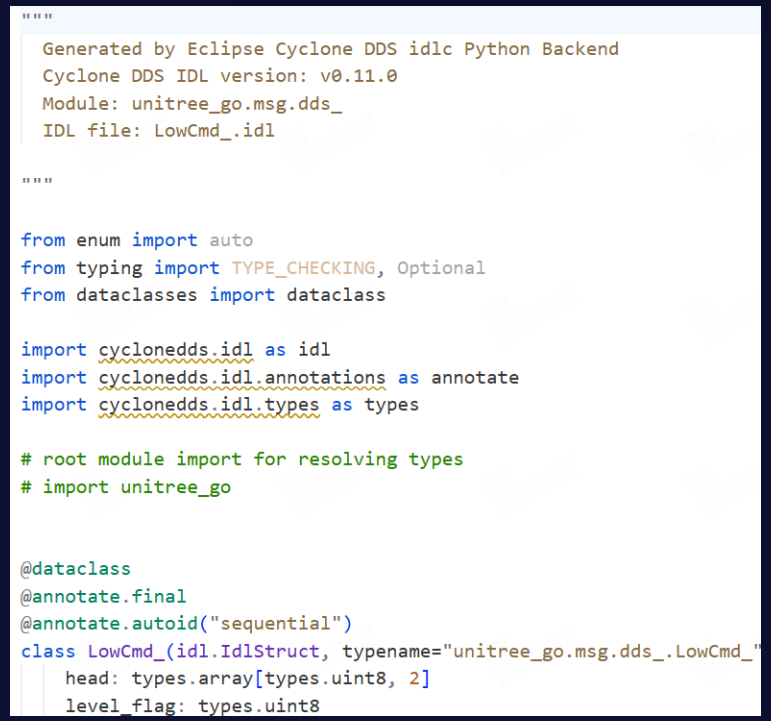
Python
.py

Robot States

DDS Topics



```
/******************************************************************

  Generated by Eclipse Cyclone DDS IDL to CXX Translator
  File name: BmsCmd_.idl
  Source: BmsCmd_.hpp
  Cyclone DDS: v0.10.2

******************************************************************/
#ifndef DDSCXX_UNITREE_IDL_HG_BMSCMD__HPP
#define DDSCXX_UNITREE_IDL_HG_BMSCMD__HPP

#include <cstdint>
#include <array>

namespace unitree_hg
{
namespace msg
{
namespace dds_
{
class BmsCmd_
{
private:
 uint8_t cmd_ = 0;
 std::array<uint8_t, 40> reserve_ = { };
```



```
"""
  Generated by Eclipse Cyclone DDS idlc Python Backend
  Cyclone DDS IDL version: v0.11.0
  Module: unitree_go.msg.dds_
  IDL file: LowCmd_.idl

"""

from enum import auto
from typing import TYPE_CHECKING, Optional
from dataclasses import dataclass

import cyclonedds.idl as idl
import cyclonedds.idl.annotations as annotate
import cyclonedds.idl.types as types

# root module import for resolving types
# import unitree_go


@dataclass
@annotate.final
@annotate.autoid("sequential")
class LowCmd_(idl.IdlStruct, typename="unitree_go.msg.dds_.LowCmd_"
    head: types.array[types.uint8, 2]
    level_flag: types.uint8
```

unitree_sdk2\include\unitree\idl\[Robot Type]\[Topic Name].hpp

unitree_sdk2py\idl\[Robot Type]\msg\dds_\[Topic Name].py

Robot States

DDS Topics

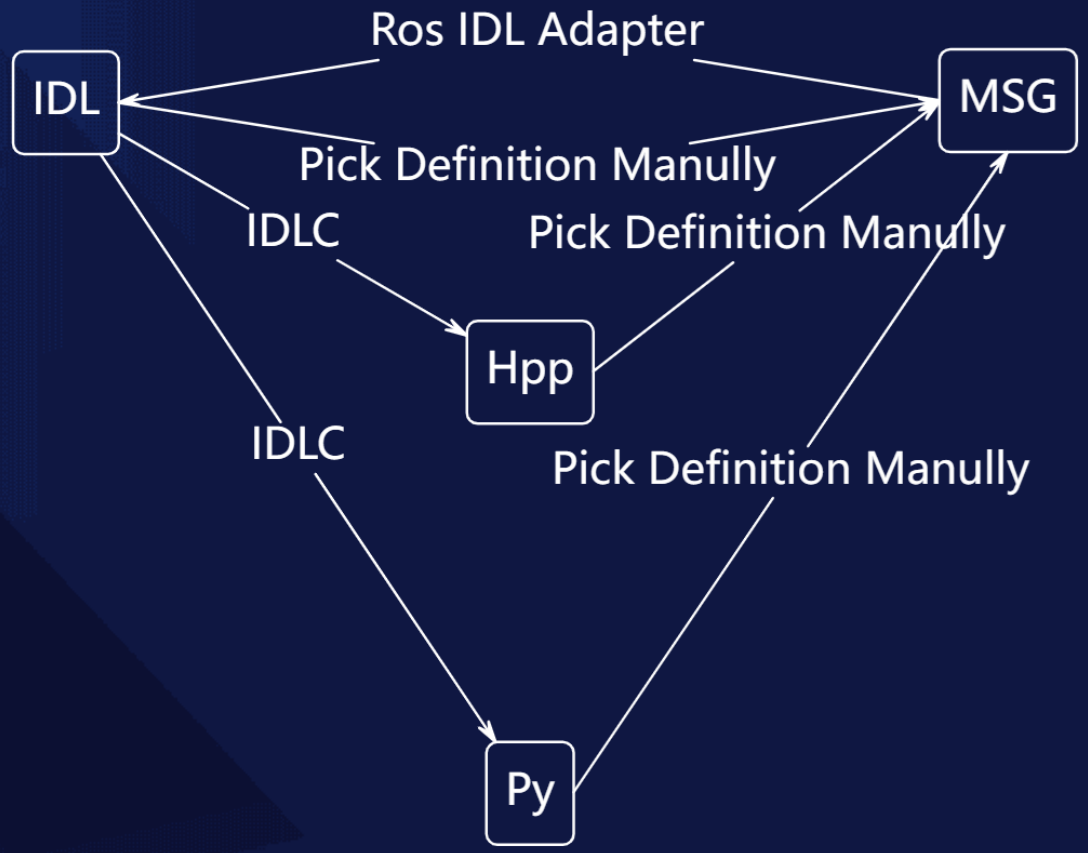
cyclonedds_ws > src > unitree > unitree_hg > msg > MotorCmd.msg

```
1   uint8 mode
2   float32 q
3   float32 dq
4   float32 tau
5   float32 kp
6   float32 kd
7   uint32 reserve
```

unitree_ros2\cyclonedds_ws\src\unitree\[Robot Type]\msg\[Topic Name].msg

Robot States

DDS Topics



IDL Type	MSG Type
boolean	bool
octet	byte
char	char
int8	int8
uint8	uint8
int16	int16
uint16	uint16
int32	int32
uint32	uint32
int64	int64
uint64	uint64
float	float32
double	float64
string	string
sequence<T>	T[]

Robot States

DDS Topics

```cpp
#include <unitree/robot/channel/channel_publisher.hpp>
#include <unitree/common/time/time_tool.hpp>
#include "HelloWorldData.hpp"

#define TOPIC "TopicHelloWorld"

using namespace unitree::robot;
using namespace unitree::common;

Trae: 解释代码 | 注释代码 |
int main()
{
    ChannelFactory::Instance()->Init(0);
    ChannelPublisher<HelloWorldData::Msg> publisher(TOPIC);

    publisher.InitChannel();

    while (true)
    {
        HelloWorldData::Msg msg(unitree::common::GetCurrentTimeMillisecond(), "HelloWorld.");
        publisher.Write(msg);
        sleep(1);
    }

    return 0;
}
```

```python
import time

from unitree_sdk2py.core.channel import ChannelPublisher, ChannelFactoryInitialize
from user_data import *


if __name__ == "__main__":
    ChannelFactoryInitialize()

    # Create a publisher to publish the data defined in UserData class
    pub = ChannelPublisher("topic", UserData)
    pub.Init()

    for i in range(30):
        # Create a Userdata message
        msg = UserData(" ", 0)
        msg.string_data = "Hello world"
        msg.float_data = time.time()

        # Publish message
        if pub.Write(msg, 0.5):
            print("Publish success. msg:", msg)
```

Hello World Example

Publish Topic

CPP Python

Robot States

DDS Topics

```cpp
#include <unitree/robot/channel/channel_publish
#include <unitree/common/time/time_tool.hpp>
#include "HelloWorldData.hpp"

#define TOPIC "TopicHelloWorld"

using namespace unitree::robot;
using namespace unitree::common;

Trae: 解释代码 | 注释代码 | ×
int main()
{
    ChannelFactory::Instance()->Init(0);
```

```python
import time

from unitree_sdk2py.core.channel im
from user_data import *


if __name__ == "__main__":
    ChannelFactoryInitialize()

    # Create a publisher to publish
```

Step 00

Import Message Type Definition

Robot States

DDS Topics

```
/*****************************************************************

  Generated by Eclipse Cyclone DDS IDL to CXX Translator
  File name: HelloWorldData.idl
  Source: HelloWorldData.hpp
  Cyclone DDS: v0.10.2

*****************************************************************/
#ifndef DDSCXX_HELLOWORLDDATA_HPP
#define DDSCXX_HELLOWORLDDATA_HPP

#include <cstdint>
#include <string>

namespace HelloWorldData
{
class Msg
{
private:
```

```
from dataclasses import dataclass
from cyclonedds.idl import IdlStr


# This class defines user data cc
@dataclass
class UserData(IdlStruct, typenam
    string_data: str
    float_data: float
```

Step 00

Import Message Type Definition

Robot States

DDS Topics

```
sing namespace unitree::robot;
sing namespace unitree::common;

Trae: 解释代码 | 注释代码 | ×
nt main()

    ChannelFactory::Instance()->Init(0);
    ChannelPublisher<HelloWorldData::Msg> publis

    publisher.InitChannel();

    while (true)
```

```
om user_data import *


__name__ == "__main__":
    ChannelFactoryInitialize()

    # Create a publisher to publish th
    pub = ChannelPublisher("topic", Us
    pub.Init()
```

Step 01

Create Channel Factory

Robot States

DDS Topics

```cpp
sing namespace unitree::robot;
sing
Trae:
nt m
C
C
p
W
r
```

```cpp
G1Example(std::string networkInterface)
    : time_(0.0),
      control_dt_(0.002),
      duration_(3.0),
      counter_(0),
      mode_pr_(Mode::PR),
      mode_machine_(0) {
  ChannelFactory::Instance()->Init(0, networkInterface);

  // try to shutdown motion control-related service
  msc_ = std::make_shared<unitree::robot::b2::MotionSwitcher
  msc_->SetTimeout(5.0f);
  msc_->Init();
  std::string form, name;
  while (msc_->CheckMode(form, name), !name.empty()) {
    if (msc_->ReleaseMode())
      std::cout << "Failed to switch to Release Mode\n";
    sleep(5);
```

```python
om user data import *
if __name__ == '__main__':

    print("WARNING: Please ensure there are no obstacles ar
    input("Press Enter to continue...")

    if len(sys.argv)>1:
        ChannelFactoryInitialize(0, sys.argv[1])
    else:
        ChannelFactoryInitialize(0)

    custom = Custom()
    custom.Init()
    custom.Start()

    while True:
        time.sleep(1)
```

Step 01

Create Channel Factory

Robot States

DDS Topics

```
ng namespace unitree::robot;
ng namespace unitree::common;

e: 解释代码 | 注释代码 | ×
main()

ChannelFactory::Instance()->Init(0);
ChannelPublisher<HelloWorldData::Msg> publisher(TOPIC);

publisher.InitChannel();

while (true)
{
    HelloWorldData::Msg msg(unitree::common::GetCurrentT
    publisher.Write(msg);
```

```
_name__ == "__main__":
ChannelFactoryInitialize()

# Create a publisher to publish the data de
pub = ChannelPublisher("topic", UserData)
pub.Init()

for i in range(30):
    # Create a Userdata message
    msg = UserData(" ", 0)
```

Step 02

Init Channel Publisher

Robot States

DDS Topics

Hand Joint Index in URDF	Hand Joint Index in IDL	Hand Joint Name
right_hand_zero	0	thumb_0
right_hand_one	1	thumb_1
right_hand_two	2	thumb_2
right_hand_three	3	middle_0
right_hand_four	4	middle_1
right_hand_five	5	index_0
right_hand_six	6	index_1

Interface Description

Users can control the dexterous hand by sending unitree_hg::msg::dds_::HandCmd_ messages to the topic "rt/dex3/(left or right)/cmd". To receive the dexterous hand status, subscribe to the topic "rt/dex3/(left or right)/state" for unitree_hg::msg::dds_::HandState_ messages.

```
ng namespace unitree
ng namespace unitree

e: 解释代码 | 注释代码 | ×
main()

ChannelFactory::Inst
ChannelPublisher<Hel

publisher.InitChanne

while (true)
{
    HelloWorldData::
    publisher.Write(
```

```
)

ublish the data de
opic", UserData

essage
)
```

Step 02

Init Channel Publisher

Robot States

DDS Topics

```
ChannelFactory::Instance()->Init(0);
ChannelPublisher<HelloWorldData::Msg> publisher(T

publisher.InitChannel();

while (true)
{
    HelloWorldData::Msg msg(unitree::common::Ge
    publisher.Write(msg);
    sleep(1);
}

return 0;
```

```
r i in range(30):
  # Create a Userdata message
  msg = UserData(" ", 0)
  msg.string_data = "Hello world"
  msg.float_data = time.time()

  # Publish message
  if pub.Write(msg, 0.5):
      print("Publish success. msg:", msg)
  else:
      print("Waitting for subscriber.")
```

Step 03

Fill and Send Message

Robot States

DDS Topics

```cpp
#include <unitree/robot/channel/channel_subscriber.hpp>
#include <unitree/common/time/time_tool.hpp>
#include "HelloWorldData.hpp"

#define TOPIC "TopicHelloWorld"

using namespace unitree::robot;
using namespace unitree::common;

Trae: 解释代码 | 注释代码 | ×
void Handler(const void* msg)
{
    const HelloWorldData::Msg* pm = (const HelloWorldData::Msg*)msg;

    std::cout << "userID:" << pm->userID() << ", message:" << pm->message() << std::endl;
}

Trae: 解释代码 | 注释代码 | ×
int main()
{
    ChannelFactory::Instance()->Init(0);
    ChannelSubscriber<HelloWorldData::Msg> subscriber(TOPIC);
    subscriber.InitChannel(Handler);
```

```python
import time

from unitree_sdk2py.core.channel import ChannelSubscriber, ChannelFactoryInitialize
from user_data import *


if __name__ == "__main__":
    ChannelFactoryInitialize()
    # Create a subscriber to subscribe the data defined in UserData class
    sub = ChannelSubscriber("topic", UserData)
    sub.Init()

    while True:
        msg = sub.Read()
        if msg is not None:
            print("Subscribe success. msg:", msg)
        else:
            print("No data subscribed.")
            break
    sub.Close()
```

Hello World Example

Subscribe Topic

Robot States

DDS Topics

```cpp
Trae: 解释代码 | 注释代码 |
int main()
{
    ChannelFactory::Instance()->Init(0);
    ChannelSubscriber<HelloWorldData::Msg> subscriber(TOPIC);

    sleep(5);
    subscriber.CloseChannel();

    std::cout << "reseted. sleep 3" << std::endl;

    sleep(3);
    subscriber.InitChannel();

    sleep(5);
    subscriber.CloseChannel();
```

```python
import time

from unitree_sdk2py.core.channel import ChannelSubscriber, Cha
from user_data import *


if __name__ == "__main__":
    ChannelFactoryInitialize()
    # Create a subscriber to subscribe the data defined in User
    sub = ChannelSubscriber("topic", UserData)
    sub.Init()

    while True:
        msg = sub.Read()
        if msg is not None:
```

Step 00~01

Message Type Definition and Initiation

Robot States

DDS Topics

```
Trae: 解释代码 | 注释代码 | ×
int main()
{
    ChannelFactory::Instance()->Init(0);
    ChannelSubscriber<HelloWorldData::Msg> subscriber(TOPIC);
    subscriber.InitChannel(Handler);

    sleep(5);
    subscriber.CloseChannel();

    std::cout << "reseted. sleep 3" << std::endl;

    sleep(3);
    subscriber.InitChannel();
```

```
using namespace unitree::robot;
using namespace unitree::common;

void Handler(const void* msg)
{
    const HelloWorldData::Msg* pm = (const HelloWorldData::Msg*)msg;

    std::cout << "userID:" << pm->userID() << ", message:" << pm->message(
}
```

```
Trae: 解释代码 | 注释代码 | ×
int main()
{
    ChannelFactory::Instance()->Init(0);
    ChannelSubscriber<HelloWorldData::Msg> subscriber(TOPIC);
    subscriber.InitChannel(Handler);
```

Step 02

Read the message

Robot States

DDS Topics

```cpp
Trae: 解释代码 | 注释代码 |
int main()
{
    ChannelFactory::Instance()->Init(0);
    ChannelSubscriber<HelloWorldData::Msg> subscriber(TOP
    subscriber.InitChannel(Handler);

    sleep(5);
    subscriber.CloseChannel();

    std::cout << "reseted. sleep 3" << std::endl;

    sleep(3);
    subscriber.InitChannel();
```

```python
import time

from unitree_sdk2py.core.channel import ChannelSubscriber, ChannelFactoryInitialize
from user_data import *


if __name__ == "__main__":
    ChannelFactoryInitialize()
    # Create a subscriber to subscribe the data defined in UserData class
    sub = ChannelSubscriber("topic", UserData)
    sub.Init()

    while True:
        msg = sub.Read()
        if msg is not None:
            print("Subscribe success. msg:", msg)
        else:
            print("No data subscribed.")
            break
    sub.Close()
```

Step 02

Read the message

Robot States

DDS Topics

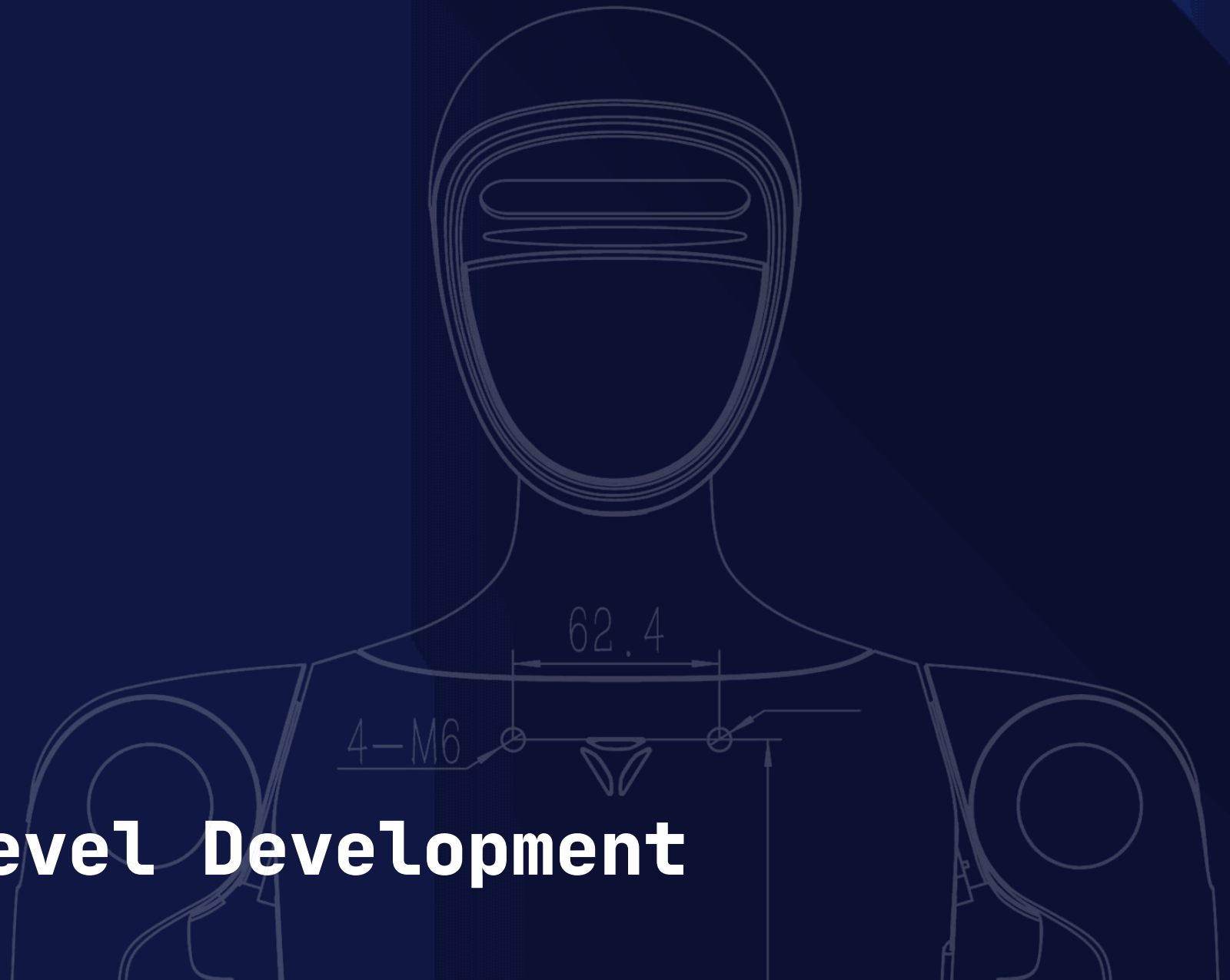
```
    # Publish message
    if pub.Write(msg, 0.5):
        print("Publish success. m
    else:
        print("Waitting for subsc

    time.sleep(1)

pub.Close()
```

```
    while True:
        msg = sub.Read()
        if msg is not None:
            print("Subscribe success. ms
        else:
            print("No data subscribed.")
            break
    sub.Close()
```

Step 02

Read the message

Software Training



202L Low Level Development

Motor Control

Joint Motor Sequence

Joint Motor Sequence

Last Updated On: 2024-01-16 17:37:09

Sorting by Message Structure

Joint Index	Joint Name
0	right hip roll
1	right hip pitch
2	right knee
3	left hip roll
4	left hip pitch
5	left knee
6	waist yaw
7	left hip yaw
8	right hip yaw
9	-
10	left ankle
11	right ankle

Left hand

Hand Joint Index in URDF	Hand Joint Index in IDL	Hand Joi
left_hand_zero	0	thum
left_hand_one	1	thum
left_hand_two	2	thum
left_hand_five	3	midd
left_hand_six	4	middle_1
left_hand_three	5	index_0
left_hand_four	6	index_1

```
module unitree_go {

module msg {

module dds_ {


struct LowCmd_ {
octet head[2];              // Frame header, used for data verifi

octet level_flag;           //Reserved, currently not used.
octet frame_reserve;        //Reserved, currently not used.
unsigned long sn[2];        //Reserved, currently not used.
unsigned long version[2];   //Reserved, currently not used.
unsigned short bandwidth;   //Reserved, currently not used.

// FR_0 -> 0 , FR_1 -> 1  , FR_2 -> 2   The motor control sequer
// FL_0 -> 3 , FL_1 -> 4  , FL_2 -> 5
// RR_0 -> 6 , RR_1 -> 7  , RR_2 -> 8
// RL_0 -> 9 , RL_1 -> 10 , RL_2 -> 11
unitree_go::msg::dds_::MotorCmd_ motor_cmd[20];    //Motor contro
unitree_go::msg::dds_::BmsCmd_ bms_cmd;            // Battery cor

octet wireless_remote[40];  //Reserved, currently not used.
octet led[12];              //It has been changed to internal co
octet fan[2];               //It has been changed to internal co
```

Motor Control

Motor Commands

```
low_cmd.motor_cmd()[i].mode() = (0x01);
low_cmd.motor_cmd()[i].q() = (PosStopF);
low_cmd.motor_cmd()[i].kp() = (0);
low_cmd.motor_cmd()[i].dq() = (VelStopF);
low_cmd.motor_cmd()[i].kd() = (0);
low_cmd.motor_cmd()[i].tau() = (0);
```

Motor Mode

PD Controller Parameters

Motor Control

Motor Commands

Motor Mode

PD Controller Parameters

0x01	0x00
FOC	Shutdown

Motor Control

Motor Commands

Motor Mode

PD Controller Parameters

$$\tau = tau + kp * (q_{des} - q) + kd * (dq_{des} - dq)$$

Motor Control

Motor Commands

$$\tau = tau + kp * (q_{des} - q) + kd * (dq_{des} - dq)$$

- τ , The expected output torque of the motor rotor, in units of `N.m`
- tau is the feedforward torque of the motor rotor, measured in `N.m`
- q_{des} is the expected angular position of the motor rotor, in `rad`
- q is the current angle position of the motor rotor, in `rad`
- dq_{des} is the expected rotor angular velocity of the motor rotor. Unit: `rad/s`
- dq is the current rotor angular velocity of the motor rotor. Unit: `rad/s`
- kp is the proportional coefficient of motor position error, which can also be regarded as the `stiffness coefficient`
- kd is the proportional coefficient of motor speed error, which can also be regarded as the `damping coefficient`

Motor Control

Joint and Torque Limitation

unitree_ros/robots/b2_description
/urdf/

```
<joint name="left_hip_pitch_joint" type="revolute">
  <origin xyz="0 0.064452 -0.1027" rpy="0 0 0"/>
  <parent link="pelvis"/>
  <child link="left_hip_pitch_link"/>
  <axis xyz="0 1 0"/>
  <limit lower="-2.5307" upper="2.8798" effort="88" velocity="32"/>
</joint>
```

Out of
Joint Limitation

Break the stop
block
Cannot
calibration

Out of
Torque
Limitation

Burn the Motor
and Circuit

Out Of Warranty

Motor Control

Example

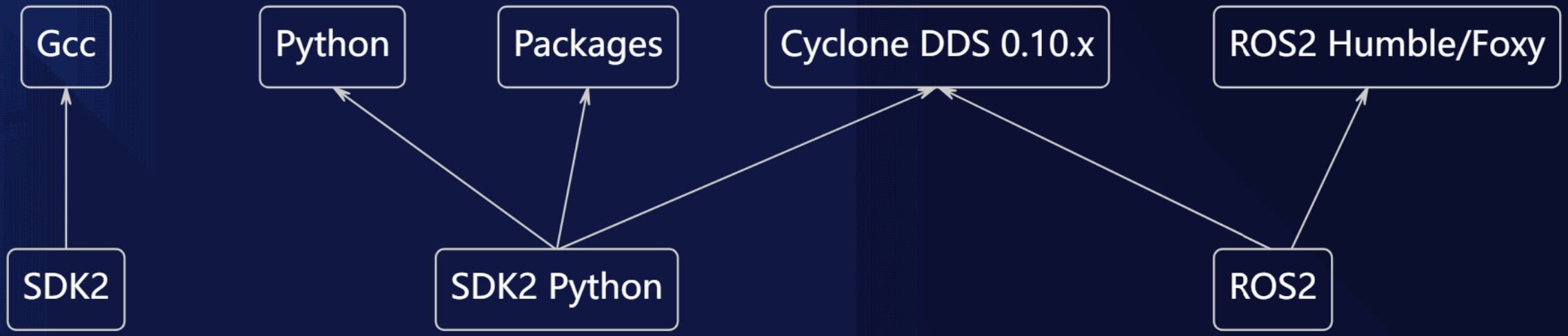
```
// Stiffness for all G1 Joints
std::array<float, G1_NUM_MOTOR> Kp{
    60, 60, 60, 100, 40, 40,      // legs
    60, 60, 60, 100, 40, 40,      // legs
    60, 40, 40,                   // waist
    40, 40, 40, 40,  40, 40, 40,  // arms
    40, 40, 40, 40,  40, 40, 40   // arms
};
```

```
// Damping for all G1 Joints
std::array<float, G1_NUM_MOTOR> Kd{
    1, 1, 1, 2, 1, 1,     // legs
    1, 1, 1, 2, 1, 1,     // legs
    1, 1, 1,              // waist
    1, 1, 1, 1, 1, 1, 1,  // arms
    1, 1, 1, 1, 1, 1, 1   // arms
};
```

```
for (int i = 0; i < G1_NUM_MOTOR; ++i) {
  double ratio = std::clamp(time_ / duration_, 0.0, 1.0);
  motor_command_tmp.q_target.at(i) = (1.0 - ratio) * ms->q.at(i);
}
```

```
const std::shared_ptr<const MotorCommand> mc = motor_command_buffer_.GetData();
if (mc) {
  for (size_t i = 0; i < G1_NUM_MOTOR; i++) {
    dds_low_command.motor_cmd().at(i).mode() = 1;  // 1:Enable, 0:Disable
    dds_low_command.motor_cmd().at(i).tau() = mc->tau_ff.at(i);
    dds_low_command.motor_cmd().at(i).q() = mc->q_target.at(i);
    dds_low_command.motor_cmd().at(i).dq() = mc->dq_target.at(i);
    dds_low_command.motor_cmd().at(i).kp() = mc->kp.at(i);
    dds_low_command.motor_cmd().at(i).kd() = mc->kd.at(i);
  }
```

Unitree SDKs Requirements



- Since both SDK2Python and ROS2 require cloning and compiling Cyclone DDS, please remember the location where you compiled Cyclone DDS to avoid duplicate installations.
- Unitree ROS2 will occupy Cyclone DDS after initialization. Attempting to run SDK2 after initialization may result in a **Core Dumped** error due to binary conflicts.

Software Training

Trouble Shooting for SDK

SDK Cannot Connect/Control The Robot

Module 03
Advanced Development

300 Example Overview

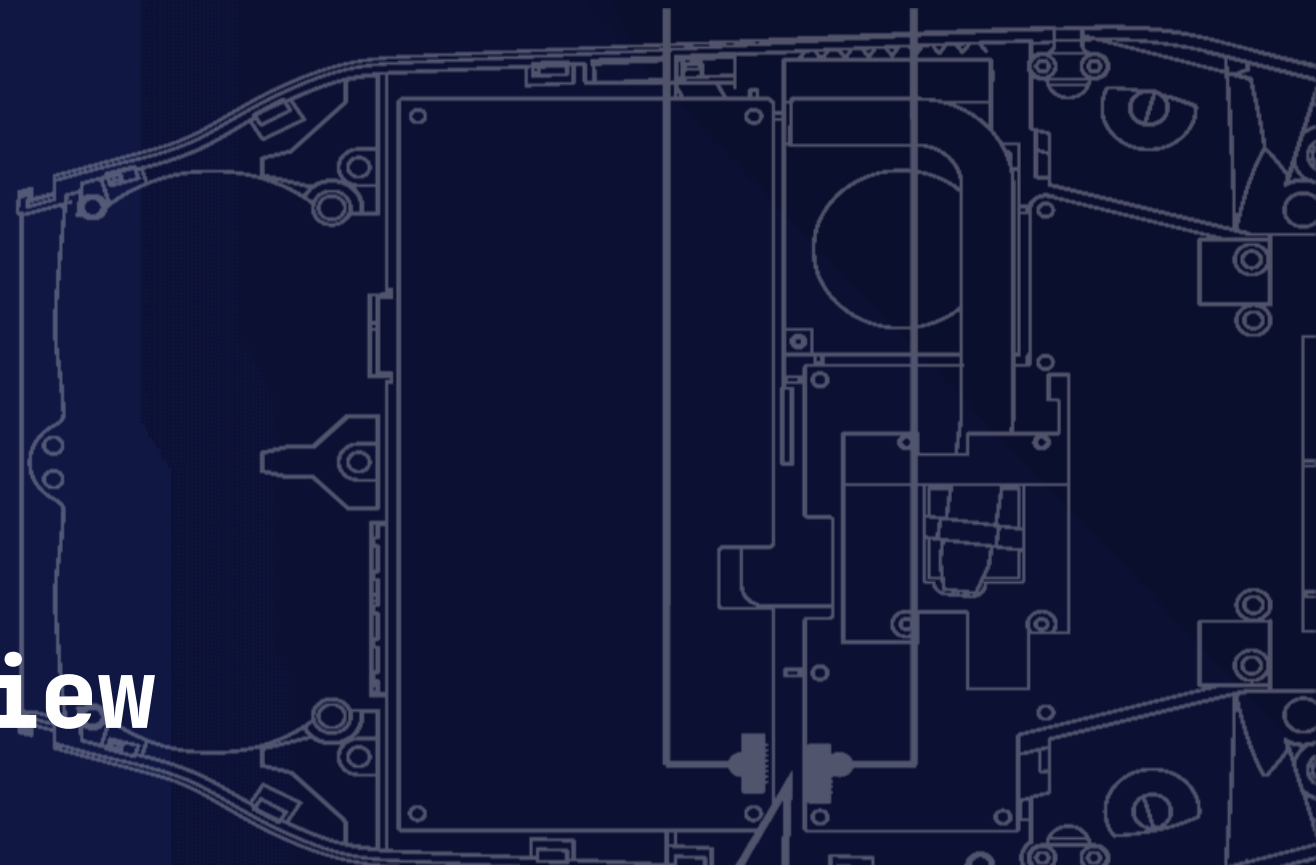
301 Unitree Interfaces

302 Policy & Mimic

303 Embodied Agent

Software Training

Software Training



300 Example Overview

Example Overview

Development Application Examples

High Level

- Equipment patrol inspection
- Emergency search and secure
- Electric-Risk Operation
- “Last Kilometer” Delivery
- Tour Guide

Low Level

- Motion Mimic
- Whole body Teleoperation
- Sport Competition
- End-to-end full-body robot model
- Exploring Extreme Performance

Example Overview

Development Application Examples

High Level

Where To Go

What To Do

Low Level

How To Move

Example Overview

Development Application Examples

High Level

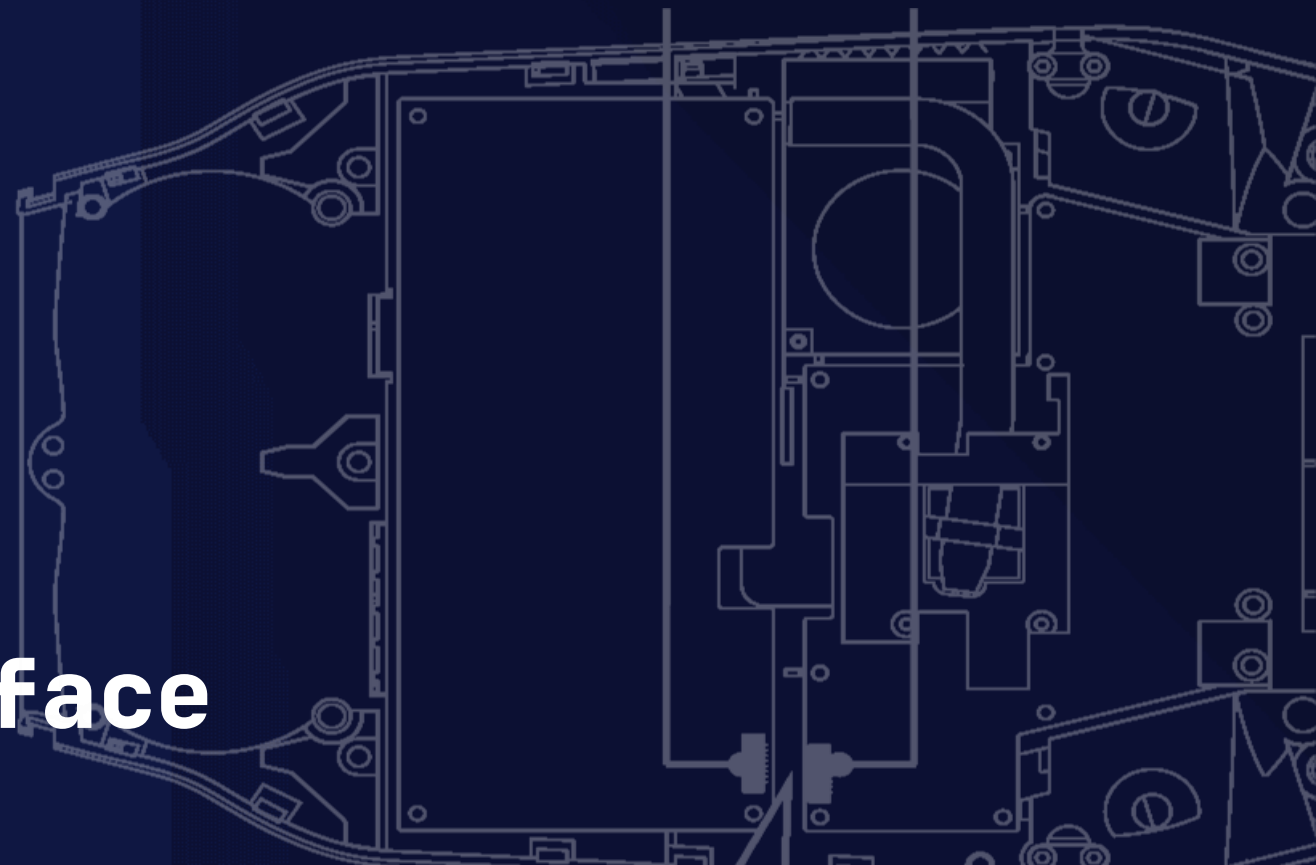
SLAM Navigation
Where To Go

Operation
What To Do

Low Level

Neural Network Policy
How To Move

Software Training



301 Unitree Interface

SLAM Interface

Unitree SLAM Interface

Type	Quadruped				Humanoid	
Robot	Go2	Go2W	B2	B2W*	G1	H1
Requirements (At least EDU)	LIVOX Mid360/HESAI XT-16 External Lidar		PC2 Robo Sense Helios 32 (SLAM Kit)	13 Gen Intel PC2 Robo Sense Helios 32 (SLAM Kit)	LIVOX Mid360 (Default Installed)	PC2
Version	2.0		1.0	1.0	2.0	1.0
Demo	√		×	×	√	×

*For B2W, please contact us to upgrade a package to enable the SLAM interface.

Unitree Interface (unitree slam) is the build-in SLAM and navigation interface provided by Unitree for the education and scientific research industries.

SLAM Interface

Unitree SLAM Interface

Version	Base	Protocol
1.0	Lio SAM Local Planner	DDS Topic
2.0	Internal Algorithm	Client

Attribute	Content
Map Size	25*25 meter
Multi Floor	Not Support
Obstacles Avoidance	Support

SLAM Interface

Build Your Own SLAM Interface

SLAM

Sensors

- LIVOX Mid-360
- Intel RealSense
- Unitree L1/L2
- HESAI XT-16
- ROBOSENSE Helios 32

Map Building

- Lio SAM — By CMU
- Point LIO — By HKU
- Cartographer — By Google

Navigation

Nav2 (ROS2) as example

Planner Server	Global Planner
Controller Server	Local Planner
Recovery Server	Recovery Behavior
Costmap Server	Manage Costmap
Lifecycle Server	Manage Nodes

Teleoperation

Unitree XR Teleoperate

xr_teleoperate

UNITREE

English | 中文 | 日本語

GitHub Wiki Discord

Video Demo

G1 (29DoF) + Dex3-1

H1_2 (Arm 7DoF)

Discord

Chat directly with Developer

Teleoperation

Unitree XR Teleoperate

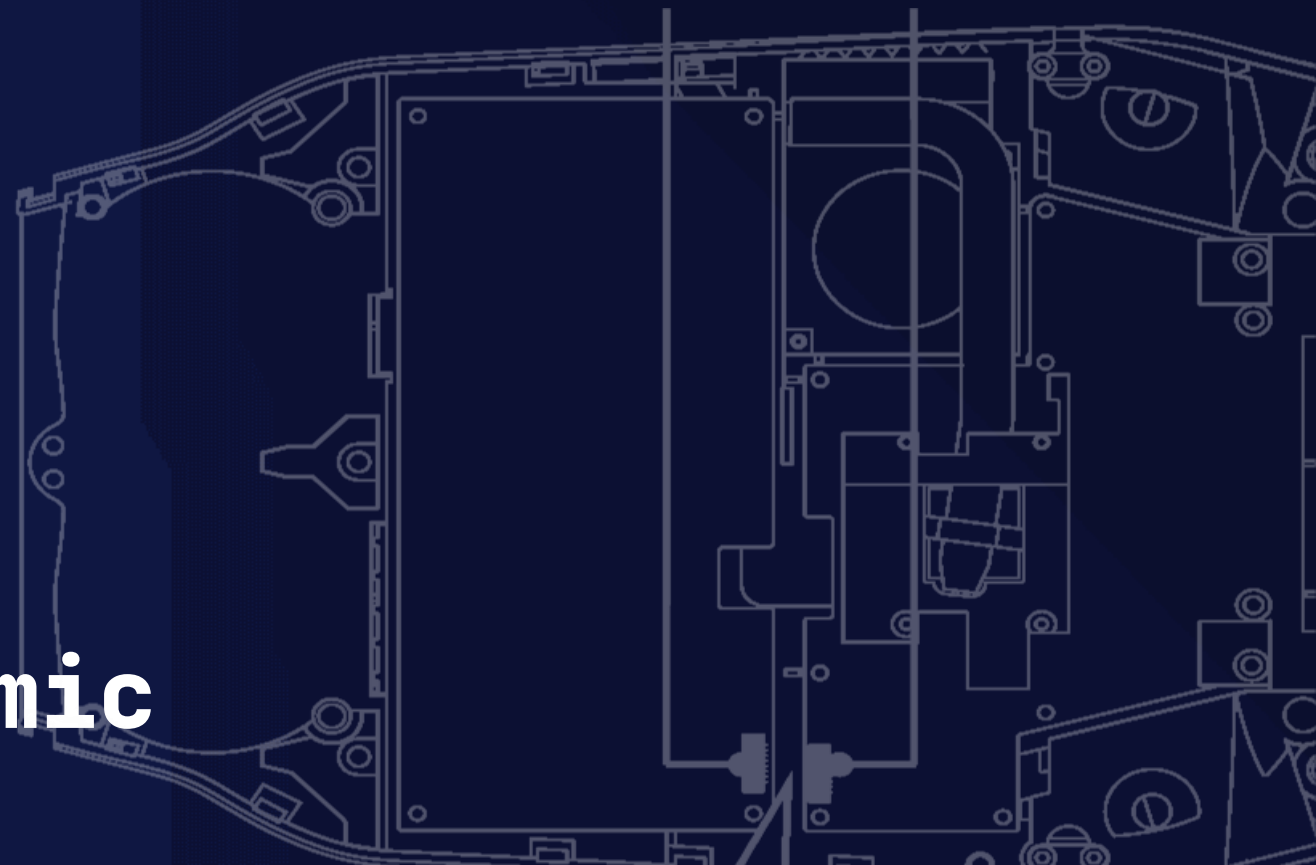
Teleoperation

Unitree XR Teleoperate

Attribute	Content	Note
Support Robot	G1(23/29DOF), H1,H1_2	*Support Controller or Hand Tracking. *We also provide Kinect DK version for H1.
Support Teaching Device	Apple Vision Pro, Pico 4 Ultra, Quest 3, Quest 3S	
Hand	Dex1-1,Dex3-3,Inspire DFX/FTP, Brain Co	
Control Scope	Upper Body	
Support Mode	Debugging Mode and Motion Mode	*In Motion Mode, you can move the robot via SDK or Remote Controller. Please note that without upper body control, balance control will be relatively more fragile.

Software Training

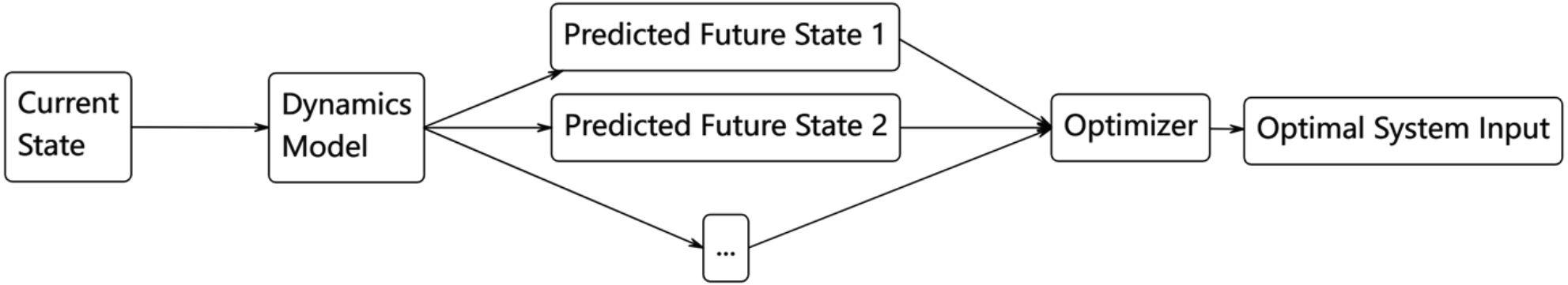
302 Policy and Mimic



Policy and Mimic

PPO Policy

MPC
Motion Predictive Control

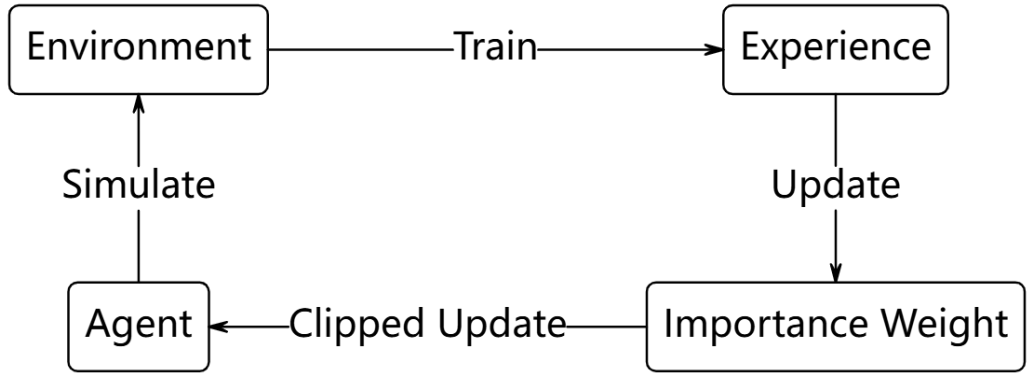


- Relying on accurate dynamic model
- High computational power consumption
- No generalization capability

Policy and Mimic

PPO Policy

PPO
Proximal Policy Optimization



- Simplified models can also be trained
- Low computational power consumption
- Generalization through training

Policy and Mimic

Policy Tasks Example

Locomotion And Controller

- Special Terrain Adaptation
- Gait Style

Motion Migration (Mimic)

- Dancing
- Real Time Whole Body Teleoperation

Multimodal Decision Machine

- Visual Following
- Automation SLAM and Navigation

Language Based Agent

- Whole Body VLM Agent
- Whole Body LLM Agent

Policy and Mimic

Mimic

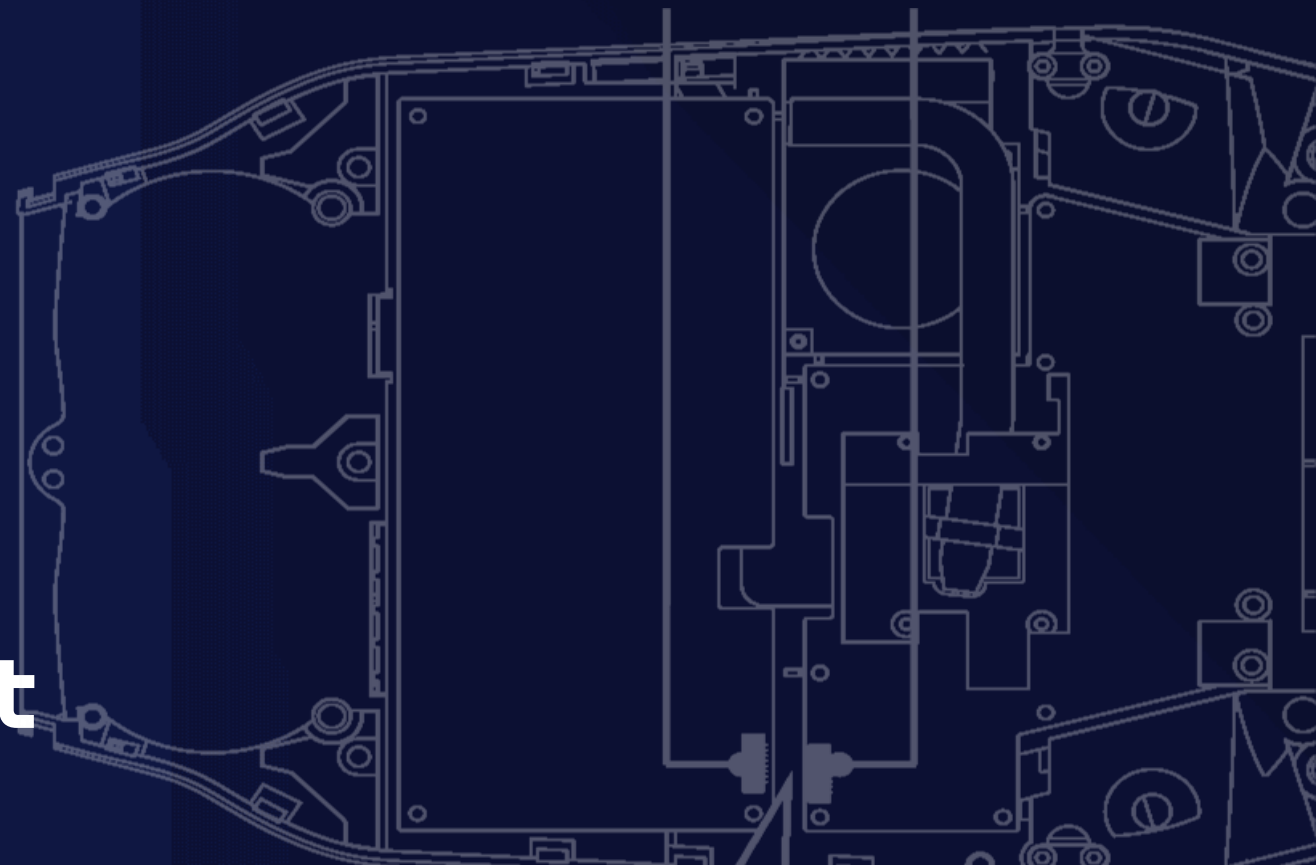
Policy and Mimic

Open Source Project Map

Data Collect	Retarget	Mimic	Deploy
Motion Capture	Loco Mujoco		Unitree Mujoco
GVHMR	GMR	TWIST	RoboMimic Deploy
Kinect	Mocap Retarget	Unitree RL Lab	
Video	Video Mimic		
	ASAP		

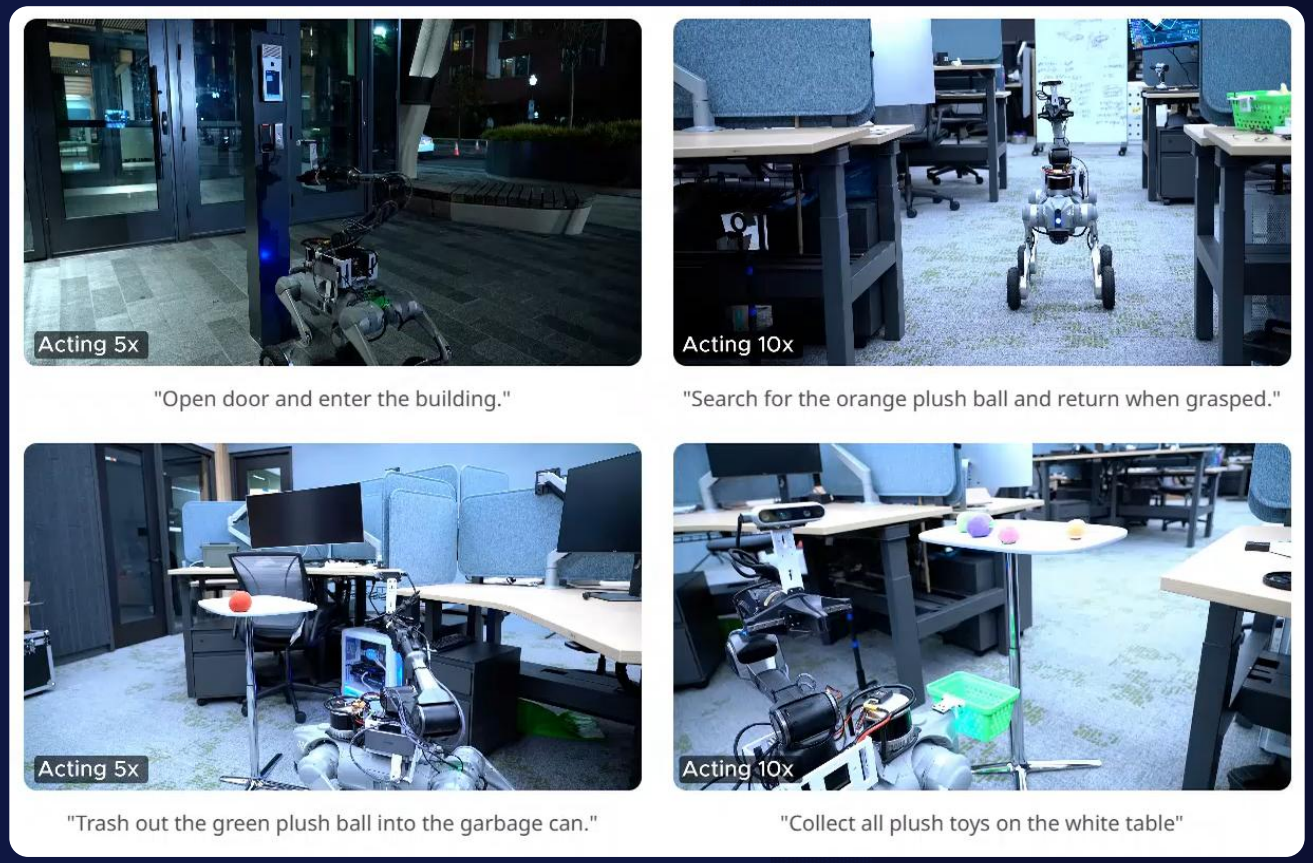
Software Training

303 Embodied Agent



High Level Agent

VLM High Level Controller



Maestro: Orchestrating Robotics Modules with Vision-Language Models for Zero-Shot Generalist Robots

High Level Agent

VLM High Level Controller

Agent System

Prompt

Sensor

Image

Result Feedback

Image Processing

Knowledge Base

Tools

Unitree
Robots

Visual Language Model

Unitree HL SDK

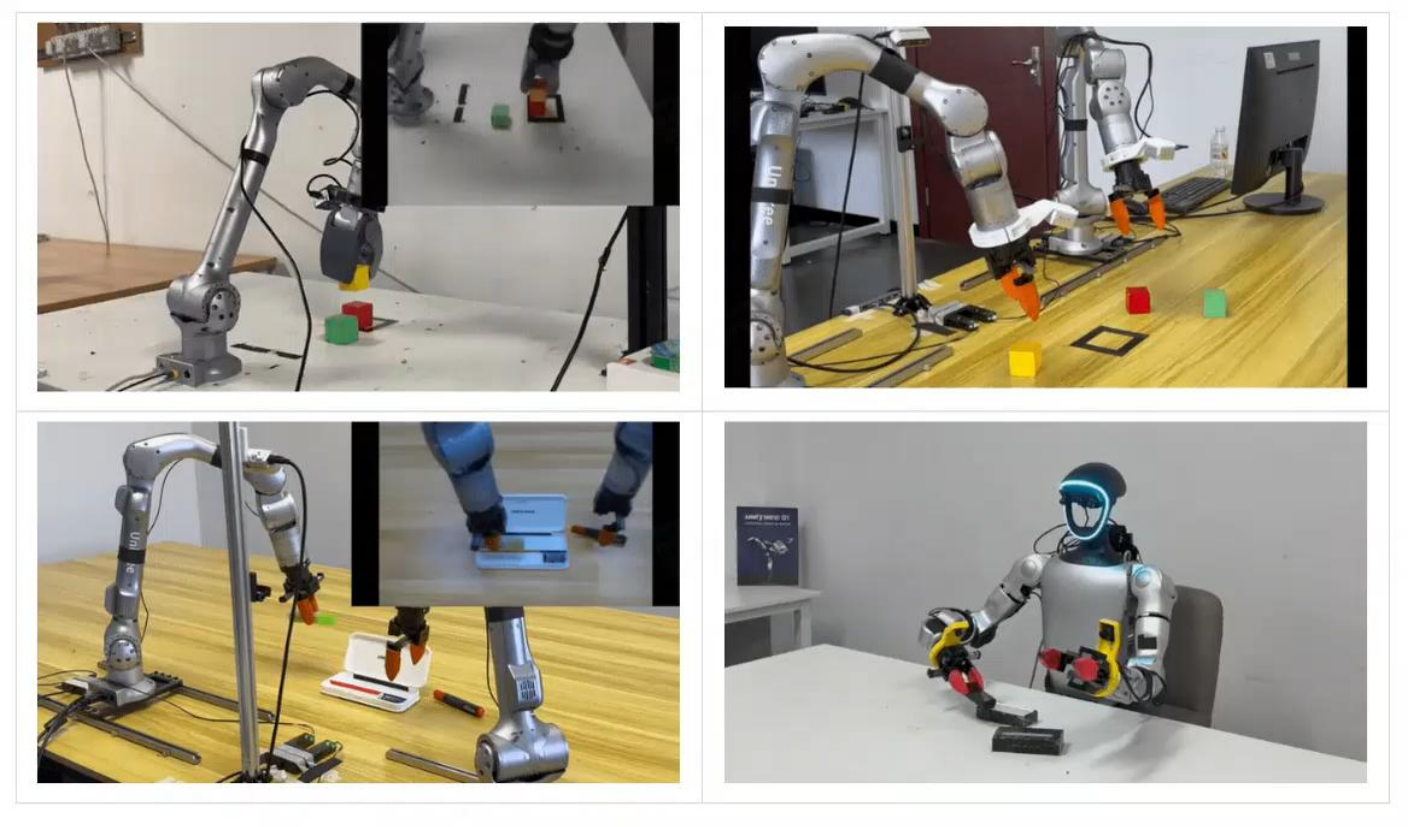
End-To-End Agent

VLM Low Level Controller

HumanoidExo: Scalable Whole-Body Humanoid Manipulation via Wearable Exoskeleton

End-To-End Agent

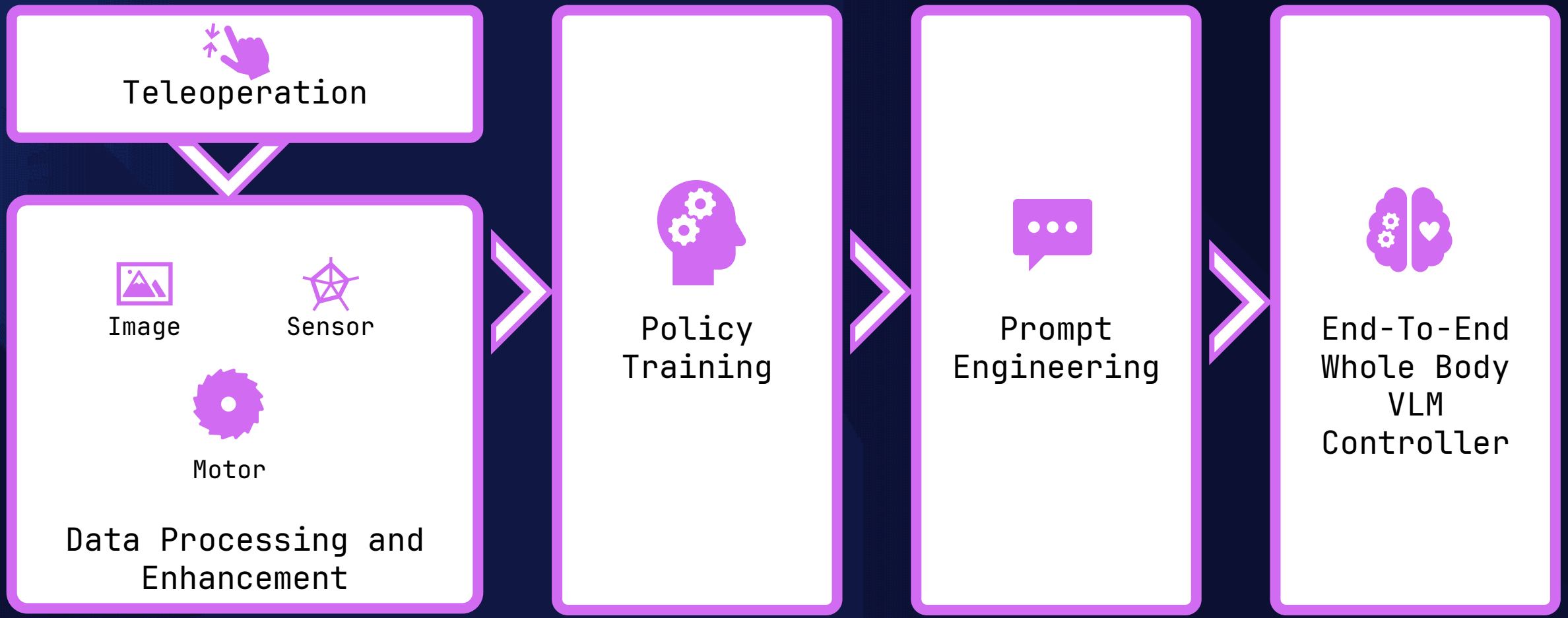
VLM Low Level Controller



UnifoLM-WMA-0: A World-Model-Action (WMA) Framework under UnifoLM Family

End-To-End Agent

VLM Low Level Controller



Software Training

Online Form

Manual Consultation

For more questions about secondary development, please fill out this form to submit, or raise a manual consultation work ticket on our platform.

UNITREE

Software Training

Secondary Development Training for Unitree Partner