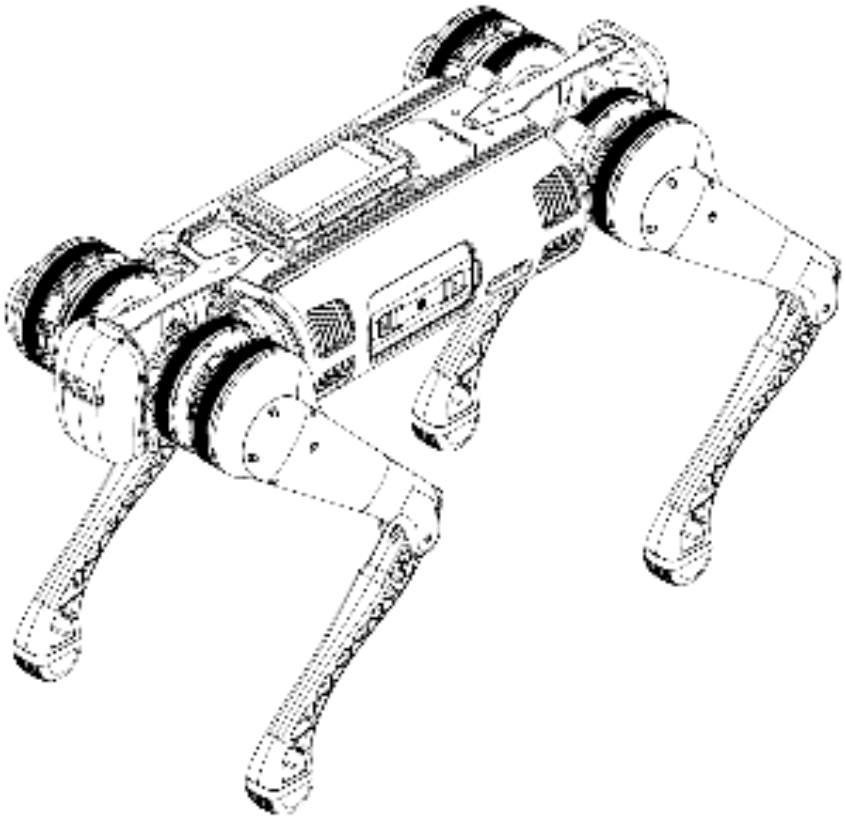
B1

Software Manual V1.0



Unitree

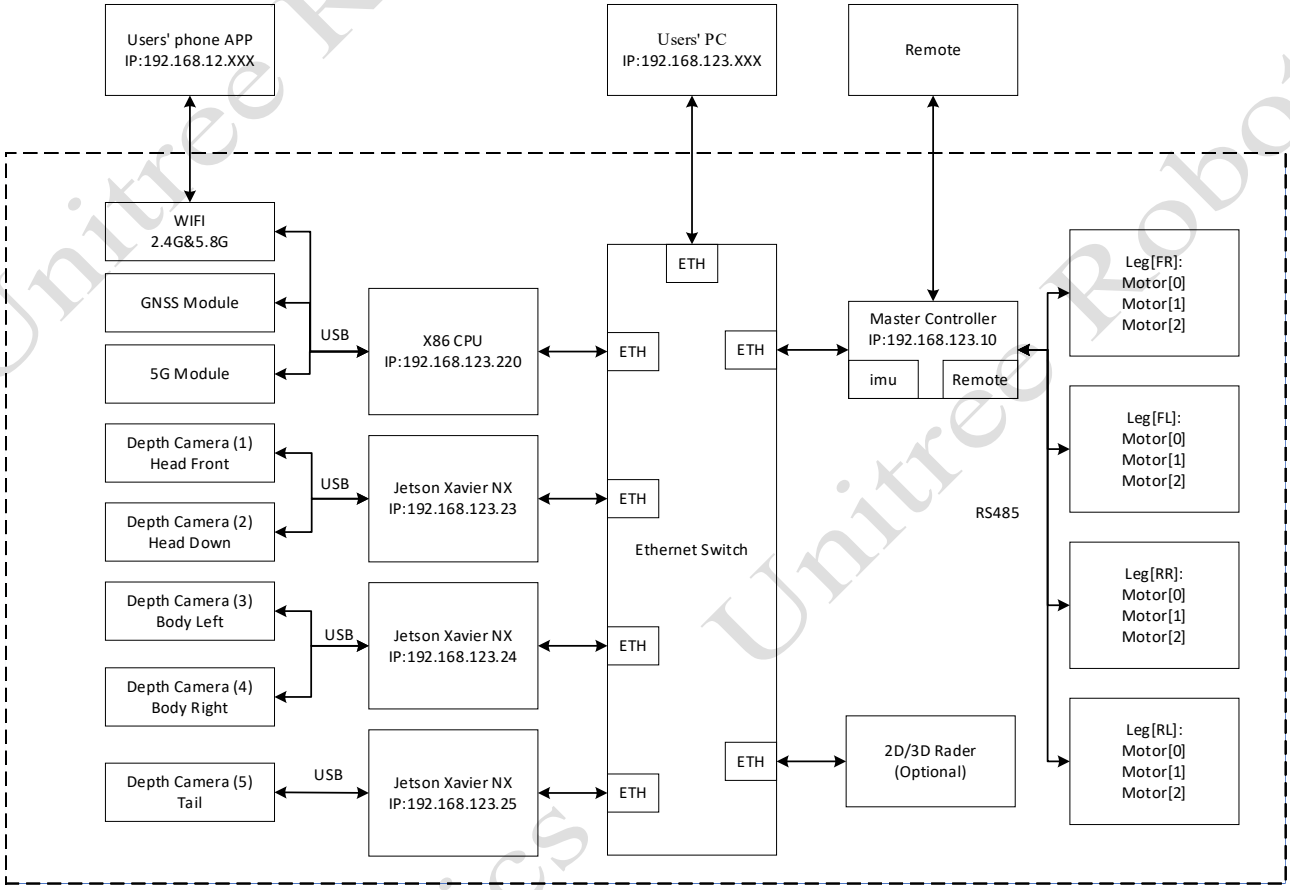
www.unitree.com

Catalogue

System

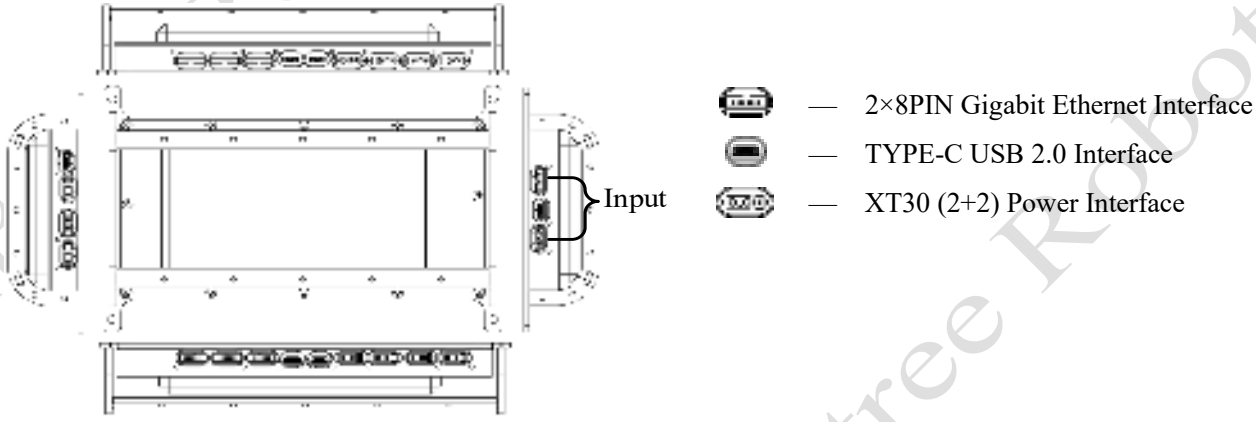
Robot system structure

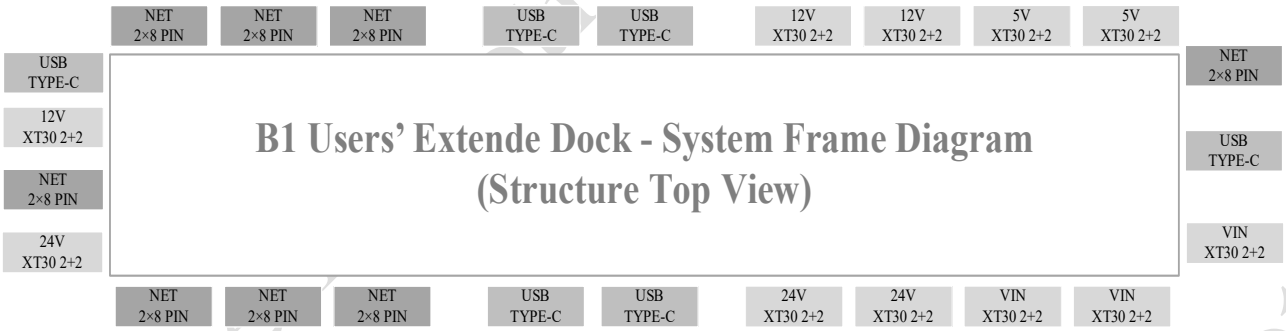
The following diagram shows the system architecture of B1.



The onboard PC has a real-time Linux (Ubuntu) operating system and X86 architecture with real-time communication frequency up to 1 KHz. 3 NVIDIA NX controllers in the body, 1 in the head and 2 in the body, provide powerful computing capabilities.

The following is a diagram of the top view interface of the B1 user docking station:





The back of the B1 provides sdk expansion interfaces to develop more extended functions for users. It includes 3 input interfaces (Gigabit Ethernet interface, Type-C interface, power interface), 7 Gigabit Ethernet extension interfaces, 5 Type-C extension interfaces, and 10 power extension interfaces. The sdk expansion interfaces are as shown in the figure above.

- **2×8PIN Gigabit Ethernet Interface+12V Power Supply+485/CAN (Pass-through) Interface:**

  1)Total number of Interfaces: 8 interfaces (2×8PIN waterproof interfaces)

  2)Gigabit Ethernet: 1 channel is connected to the robot dog Mini PC, 7 channels are connected to external

  3)Power Output: 12V × 7 channels, 10A electric current in total (same path as external power supply)

  4)485/CAN (Pass-through): 3-Interface CHA channel 485/CAN, 3-interface CHB channel 485/CAN

- **Type-C USB2.0 Interface:**

  1) Total number of interfaces: 6 interfaces (Type-C waterproof interface)

  2) 7-channel USB-HUB: The generatrix is connected to the robot dog MINIPC. 5 channels are connected to the external; 2 channels are connected to 2 groups of USB, and then transfer to 485 (or CAN) Pass-through modules: CHA/CHB

  3) Power output: 5V/1A×5 channels (USB power supply is independent of external power supply)

- **XT30(2+2) power supply +485/CAN (Pass-through) interface:**

  1) 36-58V input/output: 3 channels 10A electric current in total, CHA channel 2 interfaces 485/CAN + CHB channel 1interfacet CAN/485

  2) 24V output: 3 channels 10A electric current in total, CHA channel 2 interfaces 485/CAN + CHB channel 1 interface CAN/485

  3) 12V output: 3 channels 10A electric current in total, CHB channel 2 interfaces 485/CAN + CHA channel 1 interface CAN/485

  4) 5V output: 2 channels 5A electric current in total, CHB channel 1 interface 485/CAN + CHA channel 1 interface CAN/485

Network configuration

- **Wifi settings**

The robot transmits a wireless network segment of 192.168.123.xxx, so users can connect to this wifi with their PC and remotely connect to the robot's operating system via SSH:

```
ssh unitree@192.168.123.220
Password:123
```

To access the rest of the NX controllers, you need to ssh remote via udp on the internal network segment 192.168.123.xxx, using the head NX controller as an example.

```
ssh unitree@192.168.123.23
Password:123
```

- **Wired settings**

The user docking station on top of the robot opens the udp Ethernet interface, which is directly connected to the internal switch, so after connecting the user's personal PC to the robot via a network cable, any of the controllers can be accessed, taking the head NX controller as an example.

```
ssh unitree@192.168.123.23
Password:123
```

> ⚠ Note: Since the internal wired network of the robot are 192.168.123.xxx network segment, so when the personal PC is directly connected to the robot through the network cable, you need to set this network of the personal PC as static 192.168.123.xxx network segment, this static IP should avoid the conflict with the above board IP.

Units

In development, unspecified units are unified according to international standard units:

```
Length unit: meter (m)
Angle: radian (rad)
Angular velocity: radians per second (rad/s)
Torque: Nm (N.m)
Mass unit: kilograms (kg)
Inertial tensor unit: (kg·m2)
```


Coordinate, kinematics and dynamics

- **Joint number and joint limits**

Like an animal, the body (Trunk) and limbs (Leg) of a quadruped robot are symmetrical from left to right. The four legs are divided into two groups according to the front and back, and the two groups have the same coordinate system and joint range of motion, except for the front and back.

Number of legs and joints:

Leg0 FR = right front leg

Leg1 FL = left front leg

Leg2 RR = right rear leg

Leg3 RL = left rear leg

Joint 0:Hip, Hip joint

Joint 1:Thigh, Thigh joint

Joint 2:Calf, Calf joint

e.g. FR_thigh: right front leg thigh joint

The joint limits of leg is the same. Joint limits:

Body joint: -60°~60°

Thigh joint: -38°~170°

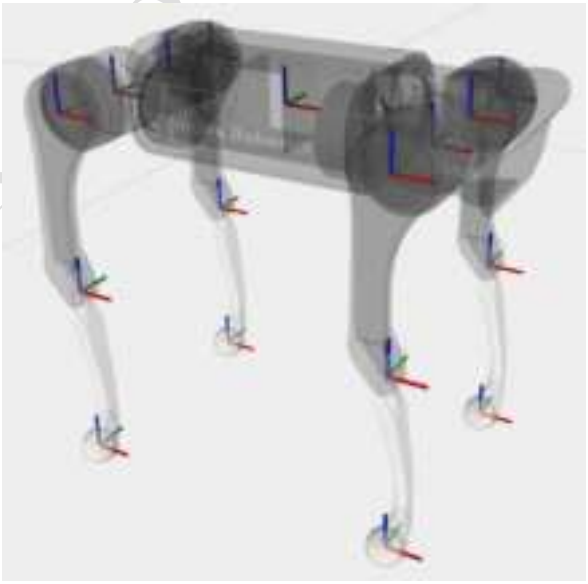
Calf joint: -156°~-48°

Hip lateral swing joint: -45~+45°

Hip front swing joint: -220~+50°

Knee joint: +24~+138°

- **Coordinate, joint axis and zero point**



All coordinates under ROS

The rotation axis of the hip joint is the x-axis, and the rotation axis of the thigh and calf joints is the y-axis, and the positive rotation direction conforms to the right-hand rule.

The zero points of each coordinate are shown above. Red is the x-axis, green is the y-axis, and blue is the z-axis. Due to the limit of the calf joint, this position cannot actually be reached. It can be seen that the initial posture of each joint coordinate system is the same, but the position and rotation axis are different.

- **Kinematic parameters**

Can be measured by the three-dimensional model we provide

(unitree_ros/const.xacro at master · unitreerobotics/unitree_ros (github.com))

- **Dynamics parameters**

Considering the symmetry, we only provide parameters of necessary modules. You can find the reference coordinate of each module in our 3D models. You can also find those parameters in our ROS package.

Each module contains three key elements: mass, position of the center of mass(CoM) and inertial tensor.

For other details about dynamics, please refer to

(unitree_ros/robots/b1_description at master · unitreerobotics/unitree_ros (github.com))

Unitree_legged_sdk

Unitree has opened up unitree_legged_sdk for users to control the robot by themselves, which is mainly used for communication between PC (linux system) and Controller board, but also for other PC with UDP. After booting, the user can only be in one of these two control modes and cannot switch between them.

Under high-level control, the target ip and port of udp are initialized to ip:192.168.123.161, port:8082

Under bottom-level control, the target ip and port for initializing udp are ip:192.168.123.10, port:8007

Under the underlying control, the motor has three control modes: torque mode, speed mode, and position mode.

Unlike those APIs that are called by functions, our API is implemented by sending and receiving encapsulated structures, which is more conducive to development. The default workspace for user control is "~/unitree_legged_sdk". The communication library libunitree_legged_sdk.so and the header file "unitree_legged_sdk.h" are used to control the commands sent and the status received by the robot. This library is the most common library used by developers and contains the communication interface needed for control.

Communocation

Establish communication between the PC and the controller board. If you are using a PC (X86-CPU in the figure below) on the B1, you can skip this section, but it is not recommended.

- **Use your own PC (Ubuntu system) USB port to connect to B1's network port (use a 16-pole junction box to adapt)**

Then open a terminal and execute the following command:

```
# Run this command after plugging in the USB hub.
# You can find an additional device ID. e.g., enpxxx
ifconfig
```

```
sudo ifconfig enpxxx dowm # enpxxx is your PC usb port
sudo ifconfig enpxxx 192.168.123.162/64
sudo ifconfig enpxxx up
ping 192.168.123.220
```

If you can receive a message like "64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=xxx ms", then you are connected.

- **Use your own PC (Ubuntu system) USB port to connect to B1's network port (use a 16-pole junction box to adapt)**

Then open a terminal and execute the following command.

```
sudo ifconfig eth0 down # eth0your own computer
sudo ifconfig eth0 192.168.123.162/24
sudo ifconfig eth0 up
ping 192.168.123.161
```

If you can receive "64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=xxx ms", then you have connected successfully.

Dependencies

If you are using it on the B1's onboard computer, you can skip this step.

Boost (version 1.5.4 or higher)

CMake (version 2.8.3 or higher)

```
cd lcm-x.x.x
mkdir build
cd build
cmake ../
make
sudo make install
```

Build

Open a terminal in the unitree_legged_sdk folder and execute the following command.

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

Run

Open a terminal and run the binary in the "build/bin" folder

High-level routines can be used directly, e.g.

```
#Add "sudo" before running to increase privileges
Sudo ./example_walk
```

Before running the underlying routines, use the handle to switch the B1 control mode to basic mode(L2+B-----L1+L2+START)

```
# Warning: Always suspend the robot before running.
sudo ./example_postion
```


File System

The files in *"include/unitree_legged_sdk/"* define constants, variables, classes, API functions, etc.

include/unitree_legged_sdk/	**description**
a1_const.h aliengo_const.h go1_const.h b1_const.h	Define the robot's 12-joint limit
quadruped.h	Define robot type, control level type, number of legs and joint number
unitree_joystick.h	Defining key and joystick variable data types
comm.h	Define common classes, such as LowState, LowCmd, HighState, and HighCmd classes, for storing user commands or robot state data
safety.h	Define safety levels to keep robots in a safe environment
udp.h	Define UDP class, user can communicate between PC and Controller board through UDP object, also can be used on other PC
loop.h	Define LoopFunc class, user can generate different threads for different objects through LoopFunc object
unitree_legged_sdk.h	Contains all the above header files

After the "Build" section a "build/bin/" folder will be added to the file system, which contains the "example/" folder showing

Example of the generated executable.

examples/	**Descryption**	**notics**
example_position.cpp	Robot bottom position control example, right front leg calf will tap	Normall mode
example_torque.cpp	Robot bottom torque control example, the right front leg thigh joint will move to zero position and be in impedance control	Normall mode
example_velocity.cpp	Robot bottom speed control example, the right front leg calf will slap	Normall mode
example_walk.cpp	Example of high-level robot control, where the robot does a series of actions	Sport mode
example_wirelessHandle.cpp	Example of wireless joystick communication, when you press the A button, the joystick data will be printed on the screen	Normall mode