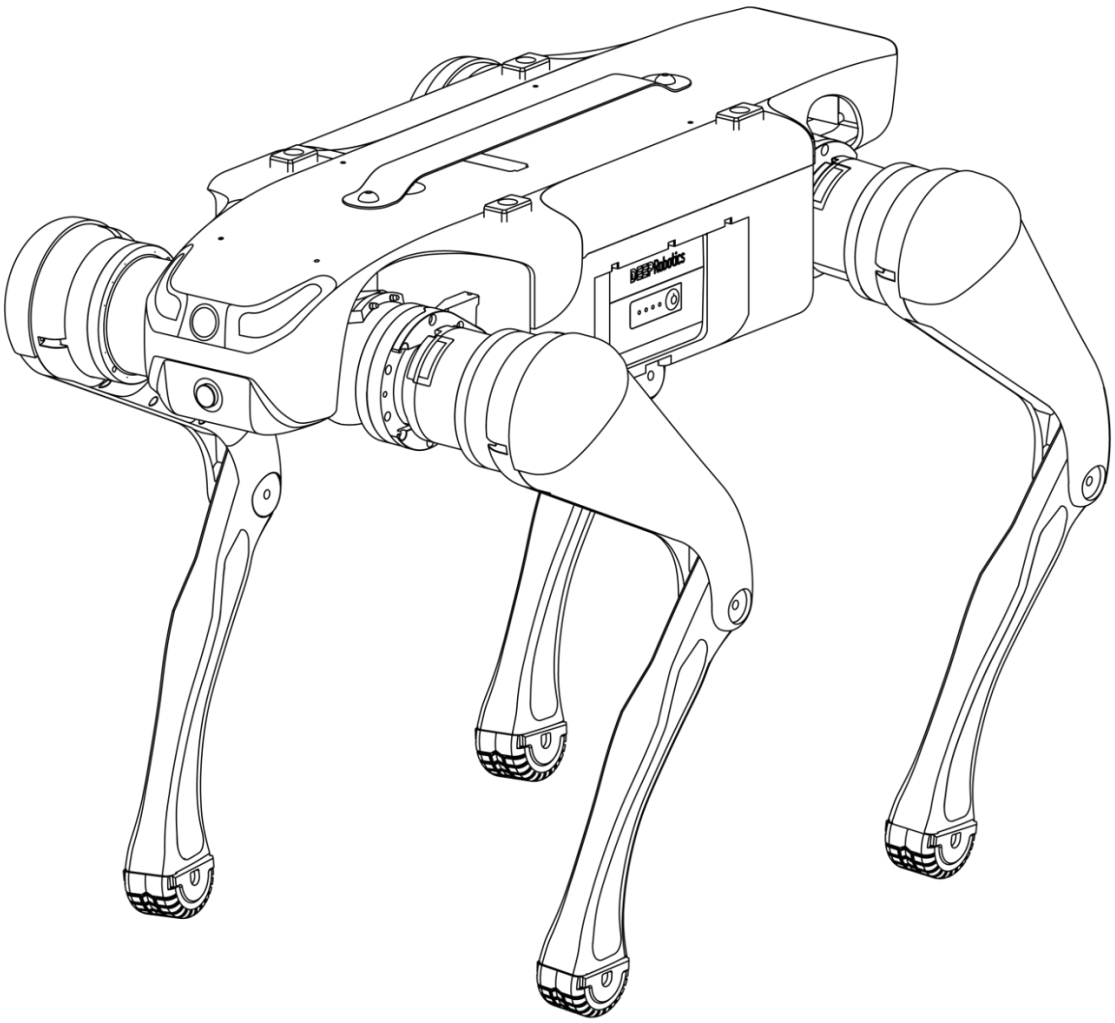
DEEPRobotics

Jueying Lite3

Motion Development Manual (beta)

V2.0.1-0 2024.7.8



Content

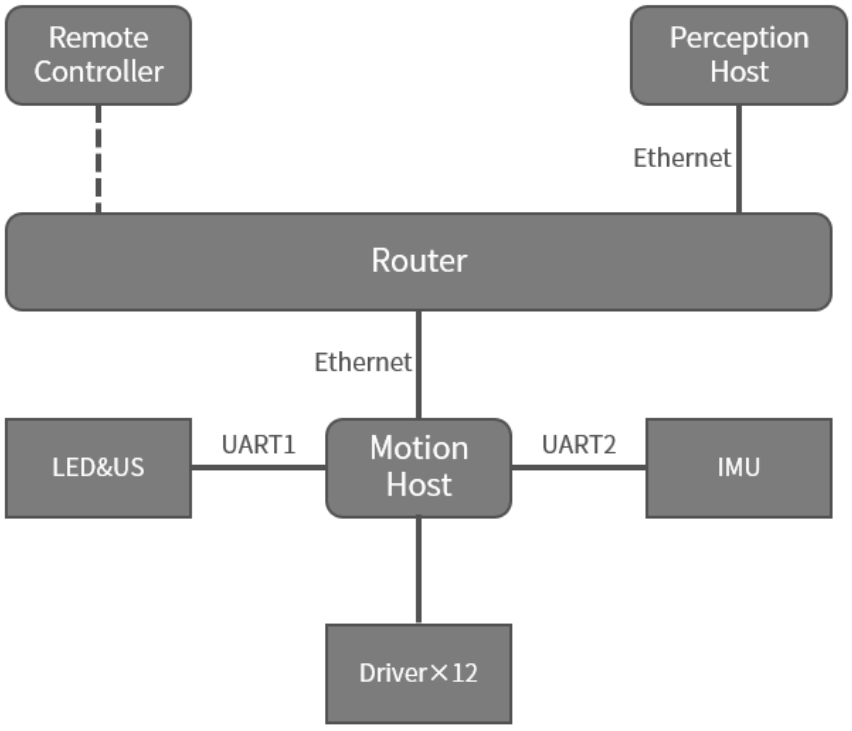
Document Description

This manual is for users who have some expertise and need to explore, develop or validate motion control algorithms with Jueying Lite3.

Manual Version	Update Description	Release Date
V1.0.3-0	First Release	2023/6/16
V2.0.0-0	Modify development steps	2024/5/10
V2.0.1-0	Modify joint rotation range	2024/7/8

1 Motion System

Jueying Lite3's motion host uses an ARM-architecture CPU processor, RK3588, with commonly used software, dependencies, and libraries in it, to meet the needs of highly-dynamic motion control and gait planning development. Jueying Lite3 is composed of Front Left Leg (FL), Front Right Leg (FR), Hind Left Leg (HL), Hind Right Leg (HR) and the body. Each leg consists of 3 joints, including HipX (the hip joint for abduction and adduction), HipY (the hip joint for flexion and extension), and Knee. A joint consists of a high-power DC motor, a precise reducer and an absolute encoder.

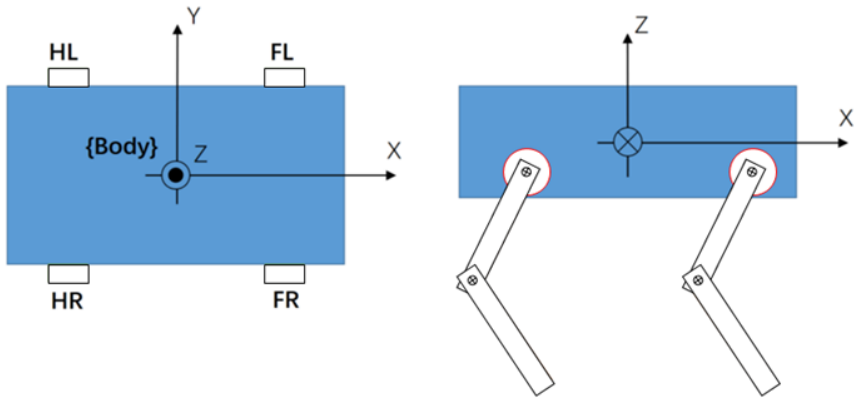


1.1 Coordinate System

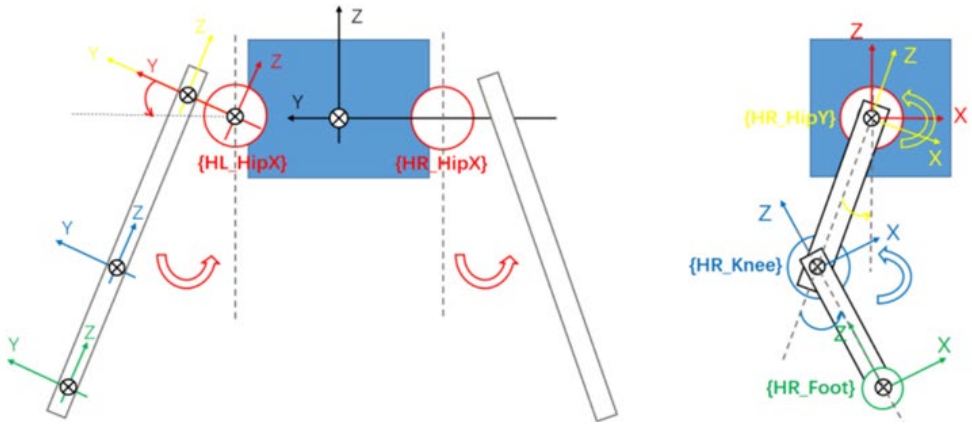
Jueying Lite3 uses 3D coordinate systems to describe the relationship between the body, sensors, joints, and soles of the feet.

1.1.1 Body Coordinate System

The origin of the body coordinate system is located at the geometric center of the robot's body.



1.1.2 Joint Coordinate System



[Caution] The arc-shaped arrow indicates the positive direction of rotation for the joint coordinate system with the same color.

Coord	Color	X(m)	Y(m)	Z(m)	Notes
FL_HipX	Red	0.1745	0.062	0	Relative to Body Coordinate System
FL_HipY	Yellow	0	0.0985	0	Relative to FL_HipX Coordinate System
FL_Knee	Blue	0	0	-0.20	Relative to FL_HipY Coordinate System
FL_Foot	Green	0	0	-0.21	Relative to FL_Knee Coordinate System

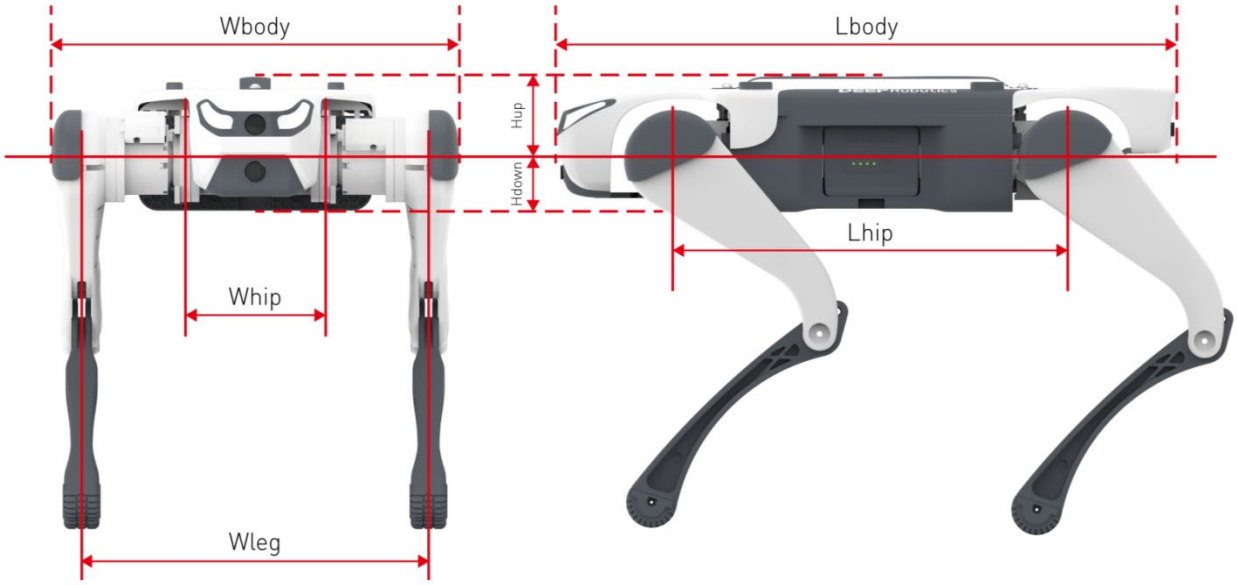
FR_HipX	Red	0.1745	-0.062	0	Relative to Body Coordinate System
FR_HipY	Yellow	0	-0.0985	0	Relative to FR_HipX Coordinate System
FR_Knee	Blue	0	0	-0.20	Relative to FR_HipY Coordinate System
FR_Foot	Green	0	0	-0.21	Relative to FR_Knee Coordinate System
HL_HipX	Red	-0.175	0.062	0	Relative to Body Coordinate System
HL_HipY	Yellow	0	0.0985	0	Relative to HL_HipX Coordinate System
HL_Knee	Blue	0	0	-0.20	Relative to HL_HipY Coordinate System
HL_Foot	Green	0	0	-0.21	Relative to HL_Knee Coordinate System
HR_HipX	Red	-0.1745	-0.062	0	Relative to Body Coordinate System
HR_HipY	Yellow	0	-0.0985	0	Relative to HR_HipX Coordinate System
HR_Knee	Blue	0	0	-0.20	Relative to HR_HipY Coordinate System
HR_Foot	Green	0	0	-0.21	Relative to HR_Knee Coordinate System

1.1.3 IMU Coordinate System

- IMU Coordinate (Relative to Body Coordinate System): (0.05,0,0)
- IMU Quaternion (Relative to Body Coordinate System): (1,0,0,0)

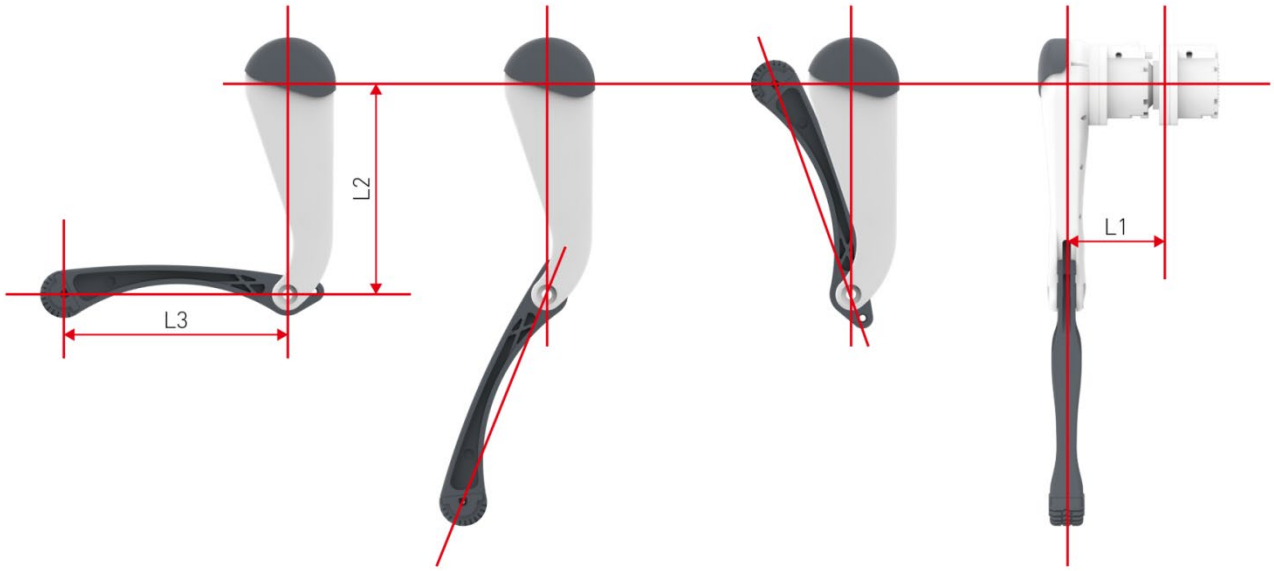
1.2 Specifications

1.2.1 Body Dimensions



Parameter	Value	Description
mbody	11kg	mass of body(with battery and HipX joints)
Lbody	548mm	length of body
Wbody	370mm	width of body
Hup	74mm	distance from the top of body to Body Coordinate Origin
Hdown	44mm	distance from the bottom of body to Body Coordinate Origin
Whip	124mm	distance between the centers of left hip and right hip
Lhip	349mm	distance between the centers of front hip and hind hip
Wleg	328mm	distance between the centers of left leg and right leg

1.2.2 Leg Dimensions



Parameter	Value	Description
mleg	1.12kg	weight of a single leg
L1	102mm	distance between the centers of leg and HipX
L2	200mm	length of upper leg
L3	210mm	length of lower leg
Rfoot	20mm	foot radius

1.2.3 Joint Dimensions

Joint	Weight	Principal Moment of Inertia		
		Px	Py	Pz
HipX	0.435kg	220kg・mm^2	220kg・mm^2	254kg・mm^2
HipY	0.501kg	267kg・mm^2	316kg・mm^2	369kg・mm^2
Knee & upper leg	0.96kg	945kg・mm^2	4440kg・mm^2	4692kg・mm^2
lower leg	0.16kg	32kg・mm^2	1294kg・mm^2	1310kg・mm^2

1.2.4 Joint Performance

Joint	Range	Nominal Torque	Max Torque	Max Velocity
HipX	-24°~24°	6.48N・m	24N・m	26.16rad/s
HipY	-200°~20°	6.48N・m	24N・m	26.1rad/s
Knee	34.5°~156°	9.72N・m	36N・m	17.1rad/s

1.3 Log

The system will generate three kinds of log files, which will be saved in the /jy_exe/data directory in the motion host.

- **Event Log**(.log): Record every event during operation.
- Zipped Package of **Operation Log**(.csv) and **Failure Snapshot**(.snapshot.csv): If the robot fails during operation, please press the button [SAVE DATA] on the app, and then robot will save the fault data and stop the motion program. The default recording frequency of the log file is 1000Hz. Operation Log(.csv) can be viewed directly with Microsoft Excel.

1.4 Protection

1.4.1 Overtemperature Protection

When the robot runs for a long time and cause the motor or driver to overheat, it will automatically turn on overtemperature protection: stop moving and lie down in place. Please do not use the robot until it cools down.

1.4.2 Fall Down Protection

When the IMU module detects that the posture changes too much and judges that the robot will inevitably fall, the damping of the joints will automatically be increased and the velocity of the robot will go to zero.

Please make sure that there are no obstacles around the robot and then let it stand up.

If the robot doesn't react to the command, please press the button [SAVE DATA] and shut down the robot.

2 Motion Control Algorithm Development

2.1 Motion SDK

Jueying Lite3 Motion SDK can get the data of each joint and send joint control command, to control the robot to move.

SDK structure reads as below:

```
.
├── CMakeLists.txt
├── include
│   ├── common
│   ├── dr_timer.h
│   ├── motionexample.h
│   ├── receiver.h
│   ├── robot_types.h
│   └── sender.h
├── lib
│   ├── eigen3
│   ├── libdeeprobotics_legged_sdk_aarch64.so
│   └── libdeeprobotics_legged_sdk_x86_64.so
├── main.cpp
└── src
    └── motionexample.cpp
```

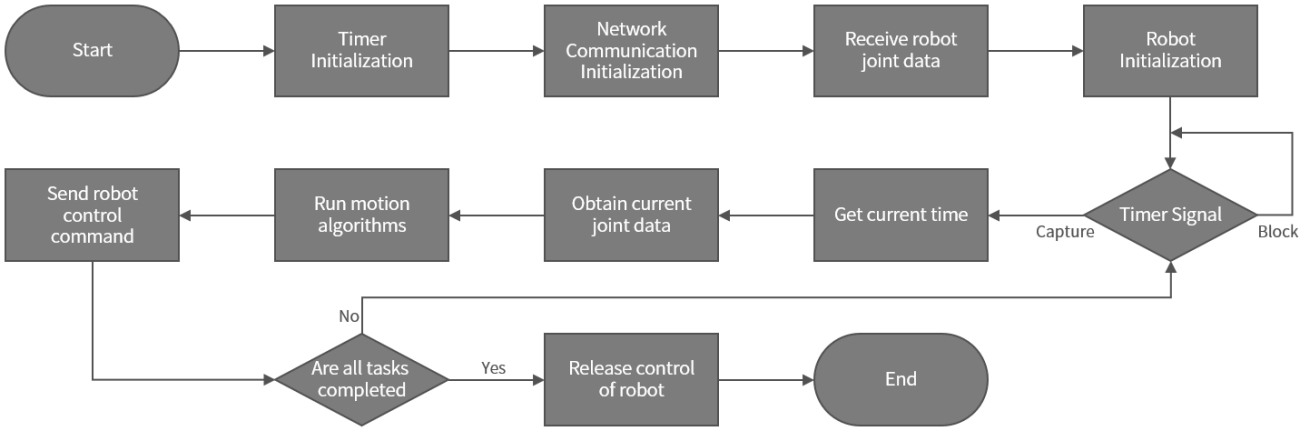
SDK contains:

1. CMakeLists Compiler: needed to be slightly modified according to the two different architectures, ARM and x86;
2. include Directory: about receiving, sending, motion algorithms, timer, and general tools;
3. lib Directory: contains the libraries needed for the motion algorithms, and dynamic libraries for x86 and ARM;

4. **main.cpp**: main function file.
5. **src** Directory: contains the main demos of motion algorithms.

Users can modify the code in main.cpp to control the joints and develop new motions.

2.2 Program Flow



1. Timer Initialization: Create a timer `DRTimer` and set the timer period to 1 (algorithm period);
2. Network Communication Initialization: Create a `Sender` and a `Receiver` for sending and receiving data;
3. Robot Initialization: reset joints to zero (obtaining control);
4. Get the current timestamp (for running motion algorithm);
5. Obtain the data `RobotData` of robot joints (preparation is completed);
6. Enter the main loop:
    - Wait for the timer to trigger;
    - Get the current time;
    - Obtain the joint status and data;
    - Run the motion algorithm `MotionExample` (including example code);

- Send the robot control commands;

(After robot completes all the control commands, release control of robot);

7. End.

2.3 Development Steps

Please refer to Lite3_MotionSDK Repository.