Aliengo Software Developer Guide

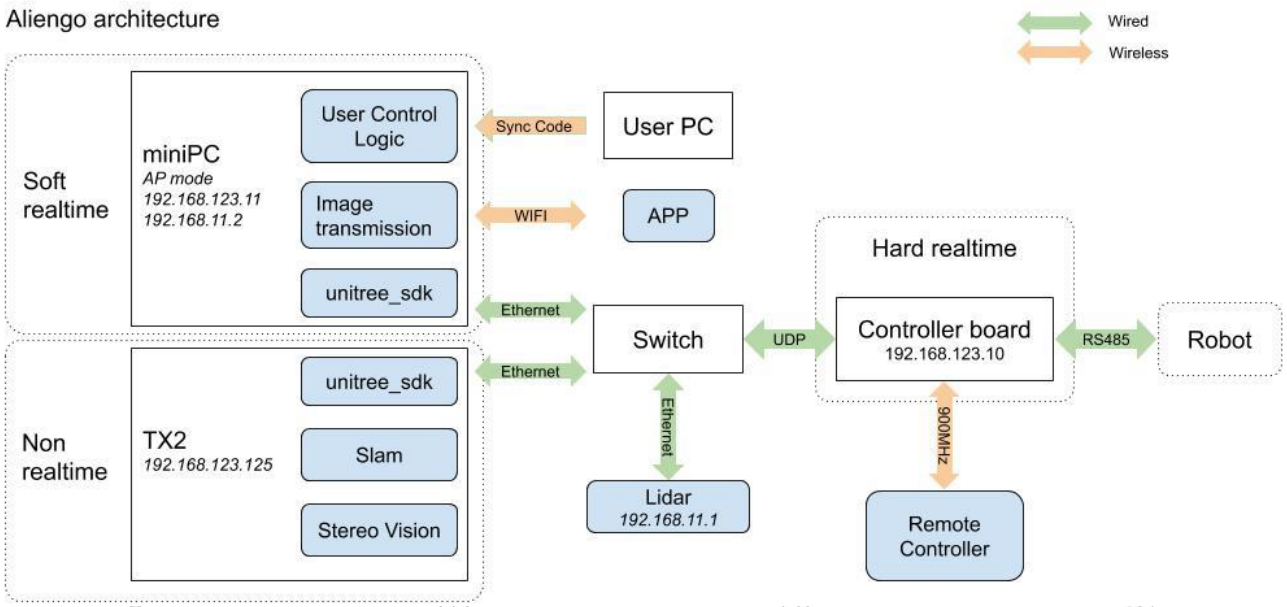
Version 1.0
3.5.2020

Catalogue

1 Getting started

1.1 Aliengo system structure

The basic architecture of the system schematic diagram shown below.



Aliengo system schematic

The operation system of Onboard PC is real time Linux(Ubuntu), which maximum real-time communication bandwidth is 1KHz.

1.2 Setting up the network

For the first time, you need to use HDMI cable and keyboard to connect the On board PC to find out its IP address. Be sure your own PC and On board PC is under the same local area network. Once you get the IP address (e.g., 192.168.1.100), you can remove the HDMI cable and connect it wireless.

Check the connection by running the "ping" test:

```
ping ${Laikago_1}
(equals to "ping 192.168.1.100")
```

User code synchronization function depends on "scp". Remote login function depends on "ssh". Make sure your own PC has installed:

sudo apt-get install ssh

ssh-keygen -t rsa

(then press three enter)

scp ~/.ssh/id_rsa.pub unitree@${Laikago_1}:~/.ssh/authorized_keys

(the remote login password: Mini PC is "123", TX2 is "nvidia", and the host name should be replaced to "nvidia" too。

If meet error about public key, run the following:

mv ~/.ssh/known_hosts known_hosts.bak

scp -o StrictHostKeyChecking=no ~/.ssh/id_rsa.pub unitree@${Laikago_1}:~/.ssh/authorized_keys

)

The On board PC is directly connected to the motion controller without any intermediate switches to decrease delay, jitter or package loss. The LAN port is initialized with default IP address and Port.

The LAN port is connected to motion controller to control the robot. The WLAN port is connected to user PC to exchange data and sync the user control codes.

PS:

1.The IP Adressof Motion Controller is static, like 192.168.123.10

2. The two modules that need to communicate should be on the same subnet

3. The miniPC wired network is currently a dual sub-network segment.

4. *wired & wireless networks should be on different subnets*

1.3 Unit

Before going down, units need to be unified：

Length unit: meter (m)

Angle unit: radian(rad)

Angular velocity unit: radian per second(rad/s)

Moment: newton metre(N.m)

Mass unit: kilogram (kg)

Inertial tensor unit: (kg·m2)

1.4 Coordinate system, kinematics and dynamics

1.4.1 Joint axis and zero point

Aliengo is like animal， the body and legs are symmertrical. The legs are divided into two groups according to the front and back. The coordinate and the range of motion of the joints are the same except for the front and back between the two groups.

Leg and Joint serial:

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 thigh joint

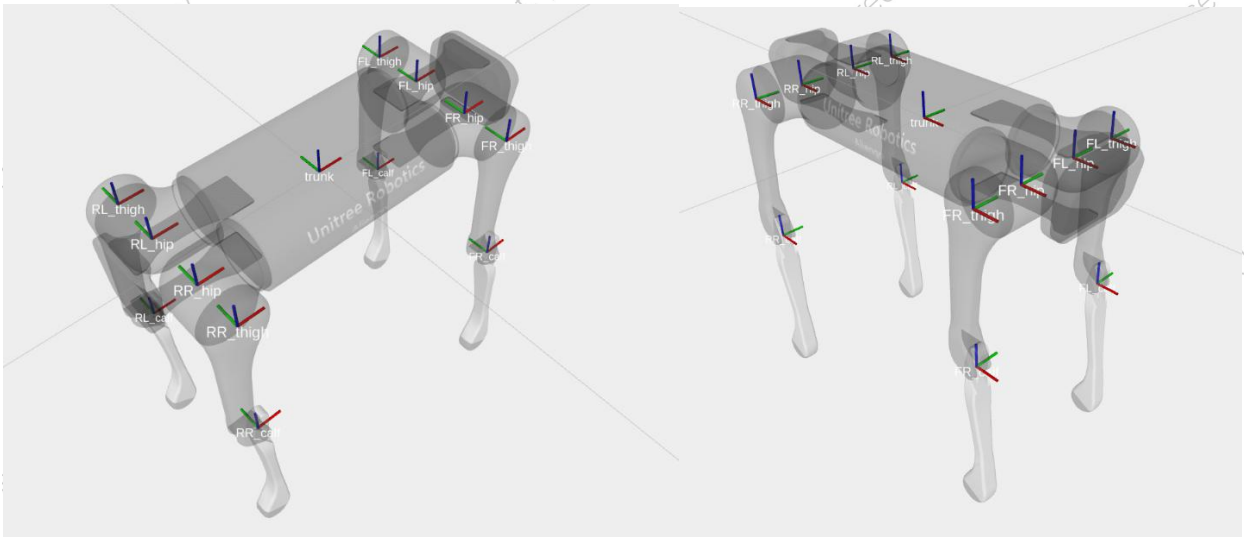
Joint limitations:

Hip joint: -70°~70°

Thigh joint: -360°~360°

Calf joint: -159°~-37°

1.4.2 Coordinate system, Joint axis and zero point



ROS each coordinate system of the whole body

The axis of rotation of the body joint is the x-axis, and the axis of rotation of the thigh and

calf joints is the y-axis. The positive rotation direction conforms to the right-hand rule.

When each joint is at zero degrees, each coordinate system is as shown above. Red is the x-axis, green is the y-axis, and blue is the z-axis. Due to the limitation of the calf joint, the position can not be reached. It can be seen that the initial pose of each joint coordinate system is the same, but the position and rotation axis are different.

1.4.3 Kinematic parameters

Kinematic parameters:

```
Hip link length: 0.083
Thigh link length: 0.25
Calf link length: 0.25
Trunk length = 0.2399 * 2
Trunk width = 0.051 * 2
```

Please refer to Appendix 1 for other configuration information of Aliengo. Other dimensions can be obtained

by measuring the 3D model .

1.4.4 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.

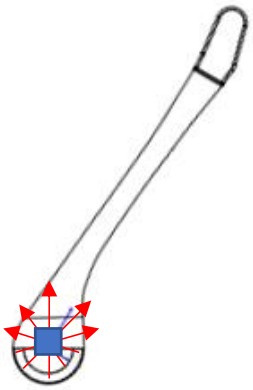
Each module contains three key elements: mass, position of the center of mass(CoM) and inertial tensor.

For other details about dynamics, please refer to

https://github.com/unitreerobotics/aliengo_ros/blob/master/aliengo_description/xacro/const.xacro.

1.5 Foot force sensor

There are four force sensors at the same position of each foot. The position is as follows:



The blue part in the figure is the sensor. The red arrow represents the direction of the force sensor. The direction of the force is determined by the contact between the foot and the ground, which is perpendicular to the contact surface. If the contact between the feet is at multiple points, the magnitude of the force can be equivalent to the combined force of these forces. This sensor is sensitive to the direction of the vertical line at any point on the spherical surface. In addition, the drift of this sensor is serious, and it needs to calibrate the zero point intermittently (such as when the end of the foot is off the ground).

2 Aliengo API

User control to robot is divided into two types: high level control mode and low level control mode. User can only be in one of the two modes at the same time, and switch the mode will take some times. The operation of switching the mode is not recommended.

In low level mode, the motor also has three modes: torque mode, speed mode, and position mode. For details, refer to Part 3.4.3.

Unlike other API through function calls, our API is all through ethernet communication interface that packaged into “struct”, which is more conducive to development. The default workspace is “~/laikago_sdk”. Currently, the tutorials and interfaces given are based on C++.

Before control the robot, the library file “liblaikago_comm.so” and header file ”laikago_sdk.hpp” should be included. It contains the high-level and low-level control API.

2.1 High level control mode

Some instructions, specifically refer to the “comm.hpp” in the tutorial codes.

HighStatus	Significance
levelFlag	the flag of control level, high level:0x00, low level:0xff
mode	the running mode: standing:1, walking:2 . It takes about a second to switch this mode.
IMU	include gyroscope, accelerometer, magnetometer, thermometer and solved euler angle and quaternion
forwardSpeed	the speed of forward walking about the body
sideSpeed	the speed of sideward walking about the body
rotateSpeed	the speed of self rotating about the body
bodyHeight	the current height about the body
updownSpeed	the speed of standing or squating
forwardPosition	the forward position from odometry
sidePosition	the sideward position from odometry
footPosition2Body	the foot position about the body
footSpeed2Body	the foot speed about the body
footForce	the foot force
tick	reference time since robot boot
crc	check code
HighCmd	
levelFlag	ditto
mode	ditto
forwardSpeed	Move backward/frontward command, value range (-1~1), corresponding to the piecewise linear proportional value of (-0.5~0.8m/s) (0 is taken as the dividing point), the maximum forward speed is 0.8m/s, and the maximum backward speed is 0.5m/s
sideSpeed	Move rightward/leftward command, value range (-1~1), corresponding to the linear proportional value of (-0.3 ~ 0.3 m/s)
rotateSpeed	Turn right/left command, value range (-1~1), corresponds to a linear proportional value of (-50 ~ 50 degrees per second)
bodyHeight	Adjust body height command, value range (-1~1), corresponding to the piecewise linear proportional value of (0.3~0.45m) (take 0.41m as the dividing point, as the default height)
yaw	Yaw command, value range (-1~1), corresponding to the linear

	proportional value of (-28~28 degrees)
pitch	Pitch command, value range (-1~1), corresponding to the linear proportional value of (-20~20 degrees)
roll	Roll command, value range (-1~1), corresponding to the linear proportional value of (-20~20 degrees)
led	the color and brightness at the foot
crc	ditto

2.2 Low level control mode

Some instructions, specifically refer to the “comm.hpp” in the tutorial codes.

LowState	
levelFlag	ditto
IMU	ditto
motorState	include the position, velocity, torque, temperature about the target motor
footForce	ditto
tick	ditto
wirelessRemote	ditto
crc	ditto
LowCmd	
levelFlag	ditto
motorCmd	include the torque, position, velocity, stiffness of position and stiffness of velocity about the target motor, and also the working mode
led	ditto
wirelessRemote	ditto
crc	ditto

2.3 Protection mode

2.3.1 Fall protection

When the robot is under high level control and encounters falling down, or hung up while the robot is standing or walking, it will switch to protection mode: All the joints switch to pure damping

mode, and light up red led.

2.3.2 Disconnection protection

Disconnection protection is also called consecutive lose package protection, as a response to poor communication. Possible reasons are: unstable network, USB mass data transfer, interruptions by GPU, and so on.

If packet is lost but does not last for 30 milliseconds, such as one or more packets, the robot will continue to run according to the last received command. If no commands are received for 30 milliseconds in a row, then it will switch to disconnection protection mode, until new command arrives.

1. High level disconnection: Robot will switch to stand mode and all of the high level command lose efficacy
2. Low level disconnection: Joints will switch to electronic braking mode.

3 Control tutorial

3.1 Emergency braking

If any accident occurs, pressing the OFF key (1.5 seconds) through the emergency remote control , then the robot will stop all actions and power off. For more details, please refer to the user manual.

3.2 Motor

Let's take the first motor on the left front leg as an example and assume that motorCmd is an array of MotorCmd types.

3.2.1 Cautions for motor

User should pay attention to the phenomenon of heating while using motor. Two thermal sensors are built into the motor to monitor the temperature in real time. According to the following two formulas:

1. According to Joule Law: Q = I2Rt (Q: heat, I: current, R: resistance, t: time)
2. Torque versus current: T = K*I (T: torque, I: current, K: torque coefficient)

It is concluded that when the motor has a square relationship with the output torque, so when

output a bigger torque, the heat of the motor will be larger. In a short time, due to the specific heat capacity of the motor itself, the instantaneous high torque output will not significantly increase the temperature of the motor. However, due to the slow heat conduction speed (thermal resistance), the temperature changes detected by the sensor will have a more obvious lag. When the temperature of any thermal sensor detected by the motor drive board is over 60°C, the motor will be forced to shut down until the temperature drops to a certain extent before opening it again.

3.2.2 Motor operation mode

The low level control can operate the motor directly. The first step is to set up the operation mode of the motor:

motorCmd[xx].mode	Explaination
0x00	Electronic brake mode
0x0A	Servo (PMSM) mode

3.2.3 Control Mode: Position, Speed and Torque

The position, speed and torque modes of the motor control are set by numerical parameters.

Torque mode: output the desired torque T, which need to disable the position and speed control loop. There are two ways, the first is to set the desired position and speed as the forbidden flag value, PosStopF = 2.146E + 9f, VelStopF = 16000.0f; the second is to set the position stiffness and velocity stiffness to zero. The first is a little faster than the second, and they can be used compound:

```
motorCmd[FL_1].position = PosStopF;
motorCmd[FL_1].positionStiffness = 0;
motorCmd[FL_1].velocity = VelStopF;
motorCmd[FL_1].velocityStiffness = 0;
motorCmd[FL_1].torque = T;
```

Speed mode: output expected speed V, which speed stiffness can not be set to zero and the position loop should be set to forbidden flag. The expected torque should be set to zero too:

```
motorCmd[FL_1].position = PosStopF;
motorCmd[FL_1].positionStiffness = 0;
motorCmd[FL_1].velocity = V;
motorCmd[FL_1].velocityStiffness = 0.02;    // just for reference
motorCmd[FL_1].torque = 0;
```

Position mode: output desired position P, which position and velocity stiffness can not be set to zero. At this time, the expected velocity should be set to zero as the pure damping term, and the

expected torque should be set to zero:

```
motorCmd[FL_1].position = P;
motorCmd[FL_1].positionStiffness = 0.2;    // just for reference
motorCmd[FL_1].velocity = 0;
motorCmd[FL_1].velocityStiffness = 0.02;    // just for reference
motorCmd[FL_1].torque = 0;
```

Compand mode: three modes can be combined as needed that two or three loops can run at the same time. For example, position loop and torque loop are running together:

```
motorCmd[FL_1].position = P;
motorCmd[FL_1].positionStiffness = 0.2;    // just for reference
motorCmd[FL_1].velocity = 0;
motorCmd[FL_1].velocityStiffness = 0.02;    // just for reference
motorCmd[FL_1].torque = T;
```

3.3 example

Refer to https://github.com/unitreerobotics/unitree_sdk/examples，Note that during execution , You need to add the sudo permissions

The walk_example is the high-level control, position_example, velocity_example, torque_example are the low-level controls.

4 Aliengo_ros

https://github.com/unitreerobotics/aliengo_ros

We provide ROS interface to control virtual robots. Place the "laikago_ros" file under catkin workspace(normally path: ~/catkin_ws/src/), then compile:

```
cd ~/catkin_ws
catkin_make
```

The environment we use: Ubuntu16.04 + ROS Kinetic or Ubuntu18.04 + ROS Melodic .

4.1 ROS Depends

4.1.1 build message Aliengo_msgs

In ROS, the node itself contains many basic data types for communication. It is necessary to define data sets needed by robots, like structures in C language. The msgs we use are in the laikago_msgs folder, and the content is consistent with the communication architecture used in the Laikago API. You need to compile laikago_msgs with catkin_make first, otherwise you may have dependency problems (such as that you can't find the header file).

4.1.2 build controller Aliengo_controller

ROS itself provides a variety of controllers (such as position_controllers). In order to achieve good integration with the robot, we don’t use its own controller here, but build our own controller. This controller inherits from “hardware::EffortJointInterface”.

4.2 Rviz Visualization

The robot description file is Xacro format, which is simpler than URDF format. It contains details of Aliengo joint limitation, collision space, kinematics and dynamics parameters. The collision space uses simple geometry to improve performance of the physical engine. Rviz is only used for visualization, and the range of motion of each joint can be checked by joint_state_publisher. Usage method: https://github.com/unitreerobotics/aliengo_ros/aliengo_rviz

4.3 Gazebo dynamic simulation

Refer tohttps://github.com/unitreerobotics/aliengo_ros#aliengo_gazebo

4.3.1 build plugins

In Gazebo, if you want to get simulated data (such as joint angle, speed, force), you need to achieve this through plugins. After calling the Gazebo API, the data should be put into the node for communication. Also note that many plugins need to rely on links, joints or modules. For further details, refer to "gazebo. xacro" file. The following shows how to build a plugin that can visualize the foot force.

1. Plugin for obtaining foot force

First, the contact force between the foot and the ground should be obtained. For more information, see "foot_contact_plugin.cc". This plug-in belongs to the sensor plugin.

2. Plugin for force visualization

After obtaining the force value, it needs to be visualized. For more information, see "draw_force_plugin.cc". This plugin belongs to the visual plugin.

4.3.2 Run the simulation

Each node is a process, so we need two terminals here. Firstly, make sure the compilation:

```
cd ~/catkin_ws
catkin_make
```

Terminal-1 will initiate scene, plugins and controllers:

```
roslaunch laikago_gazebo normal.launch
(if can't launch, run:
cd ~/catkin_ws/src/laikago_ros/laikago_gazebo/launch/
roslaunch ./normal.launch
)
```

Terminal-2 will run all the nodes:

```
rosrun laikago_gazebo laikago_servo
```

We also provide a node for imposing an external force to simulation the outside disturbance:

```
rosrun laikago_gazebo laikago_external_force
```

Users can add more nodes to accomplish the target task.

4.4 ROS control robot

4.4.1 LCM Server

The real-time performance of ROS 1 is not guaranteed. Sending UDP instructions in ROS nodes may cause communication abnormalities. It should be noted that in the communication with robot, ROS Subscriber/Publisher is not adopted to transfer data. As an alternative, LCM is used to transfer messages between processes. Non-real-time ROS processes will not affect the real-time process while controlling robot to ensure real-time performance.

In LCM Server, LCM commands will be converted into UDP commands to send down, UDP status will be converted into LCM status to send up. In ROS, a node can be set up to send LCM commands and receive LCM status separately.

4.4.2 control examples

Here take low-level control as an example, and we need three terminals here.

Firstly, copy laikago_real to ~/catkin_ws/src, and compile it:

```
cd ~/catkin_ws
catkin_make
```

Terminal-1, run 'lcm server' in sdk:

```
sudo ~/laikago_sdk/build/sdk_lcm_server_low
```

Terminal-2, run ros master node:

```
roscore
```

Terminal-3, run user logic with rosrun:

```
rosrun laikago_real position_lcm_publisher
```

There are several examples in the program you can try.

5 Aliengo Multi-sensor multi-PC

Binocular and human tracking recognition
SLAM function and related API, reference to

Appendix 1： Aliengo Product Specification

Length, width and height (stand mode)	0.55*0.35*0.6m	Total DOF (total motor count)	12
Length, width and height (transport mode)	0.56*0.35*0.31m	DOF of each leg (motor count)	3
Total weight (with battery)	About 25kg	Weight of single motor	About 840g
Load	≤5Kg	Resolution of motor encoder	0.022°(1/214)
Walk speed	0~0.8m/s	Reduction ratio of thigh and calf motor	~12:1
Climbing	<20°	Maximum instantaneous torque of thigh and calf	40N·m

		motor under no load state	
Traveling endurance (on flat pavement)	About 3~4h	IMU in trunk	1
Walking power range (on flat pavement)	60W ~ 120W	Total number of foot force sensor	4
Battery weight	4.4kg	Dimention of foot force sensor	1
Battery capacity	10500mAH*51.8V	Resolution of foot force sensor	About 30g
Maximum charge current	10A(About 1.5hfull) Recommand 5A charging	Wireless and Control	WIFI or 915MHz
Temperature range	0℃ ~ 35℃	User Control Interface	Ethernet High speed serial port (baud rate≤6MHz)
Power output for user	19V 3.7A		