Unitree
宇树科技

Unitree Robotics

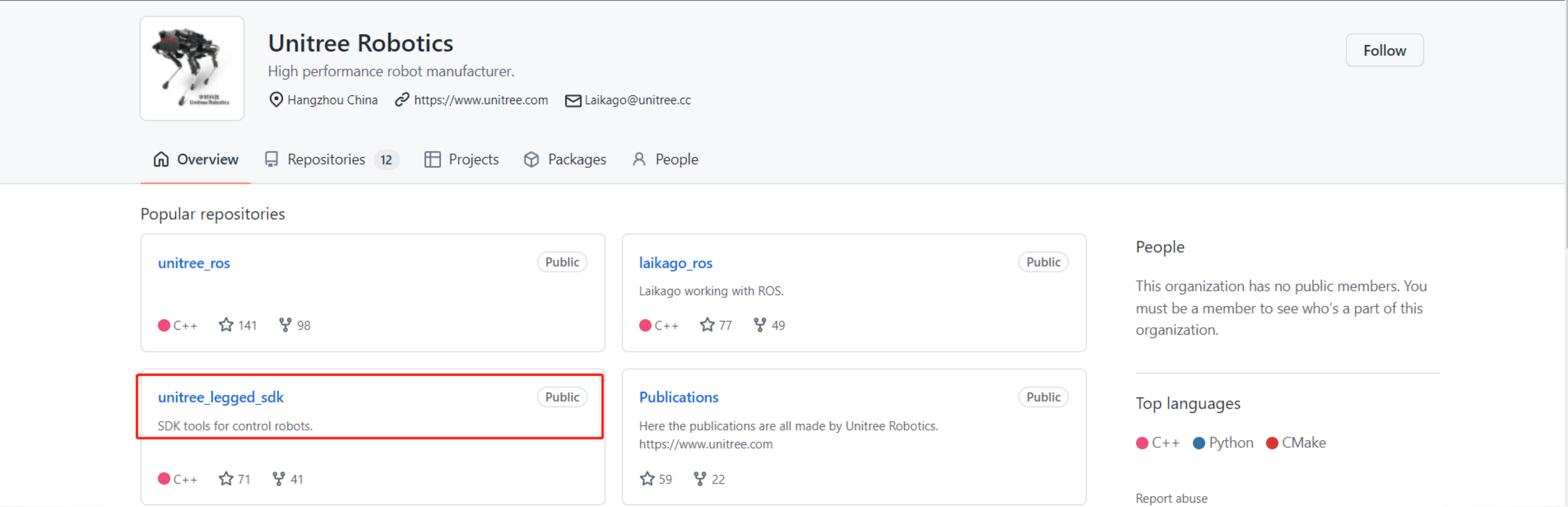
Controlling robots with Unitree_legged_sdk

CONTENTS

1

PART ONE

Introduction



- **Unitree encapsulates the methods to control Unitreerobotics in unitree_legged_sdk, in order to better use it to control Unitreerobotics, we need to understand the basic framework of Unitreerobotics before using it**

- **The Unitree family of robots is shown above. The basic framework of the control routines we provide are all named unitree_legged_sdk.**

v3.5.1 Latest

support robot: Go1
not support robot: Laikago, Aliengo, A1

Changelog

1. use std::array at file comm.h.
2. robot command and state package add head.
3. rearrange HIGHLEVEL flag.
4. rearrange SN and version.

Dependencies

```
g++               >= v8.3.0
Legged_sport      >= v1.36.0
firmware H0.1.7 >= v0.1.35
         H0.1.9 >= v0.1.35
```

v3.4.2

support robot: Go1
not support robot: Laikago, Aliengo, A1

Changelog

First add MotorState to HighState

Sport Mode

Legged_sport >= v1.32

v3.3.4

support robot: A1
not support robot: Laikago, Aliengo, Go1.

Notice

This version is only for quick development which needs motor state under high-level control.
Some other functions like vision following, slam will not take effect in this version.
Please reconfirm your needs before update.

Changelog

High-level first add motorState feedback.
Can change lcm recv block time.

Sport Mode

A1_sport >= v1.20

v3.3.1

support robot: Aliengo, A1
not support robot: Laikago, Go1.

Changelog

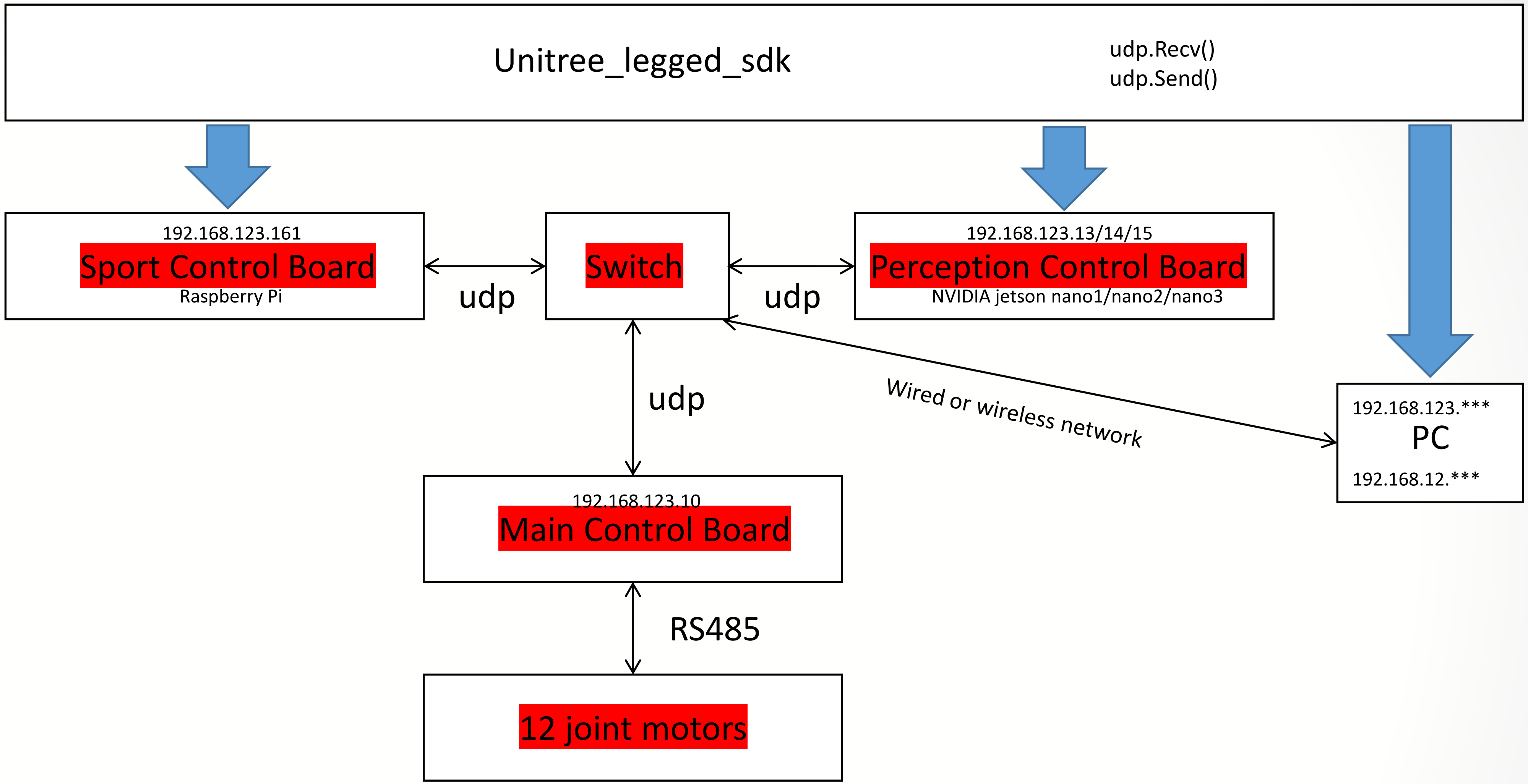
Unified HIGHLEVEL interface

Dependencies

A1 *sport_mode* >= v1.19

- **But not a single routine can control all robots, you can see in github that different versions of unitree_legged_sdk control different robots.And you need to pay attention to whether the software version meets the requirements of the routine**

- **Whether it is Laikago, Aliengo, A1, Go1, B1, the type of computer on them can be divided into three categories, the following is an example of go1:**



- Main control board,sport control board and perception control board are the robot's three computer controllers.
- Main control board controls the motor via RS485,each board is connected to the switch through udp.Each of these boards is assigned an ip address such as the 123 network segment above.
- Unitree has wrapped the corresponding udp send and receive functions in unitree_legged_sdk, so we can put the routine on any board to run (the main control board is not open).
- You can connect to the robot system either wired or wirelessly and then use the routine.

TrivasZhang example use array c449afe on 1 Mar History

..			
example_position.cpp	version 3.4	low level-position control	8 months ago
example_torque.cpp	version 3.4	low level-torque control	8 months ago
example_velocity.cpp	version 3.4	low level-velocity control	8 months ago
example_walk.cpp	HighState add MotorState	high level control	5 months ago
example_wirelessHandle.cpp	example use array	Get remote control key data	2 months ago
lcm_server.cpp	fixed lcm_server		8 months ago
multi_pc_type.h	version 3.4		8 months ago
multi_pc_udp_recv.cpp	version 3.4		8 months ago
multi_pc_udp_send.cpp	version 3.4		8 months ago

- We provide the above routines for high-level and low-level control of the robot and for obtaining remote control key messages

2

PART TWO

Download

Download URL: https://github.com/unitreerobotics/unitree_legged_sdk

v3.5.1 Latest

support robot: Go1
not support robot: Laikago, Aliengo, A1

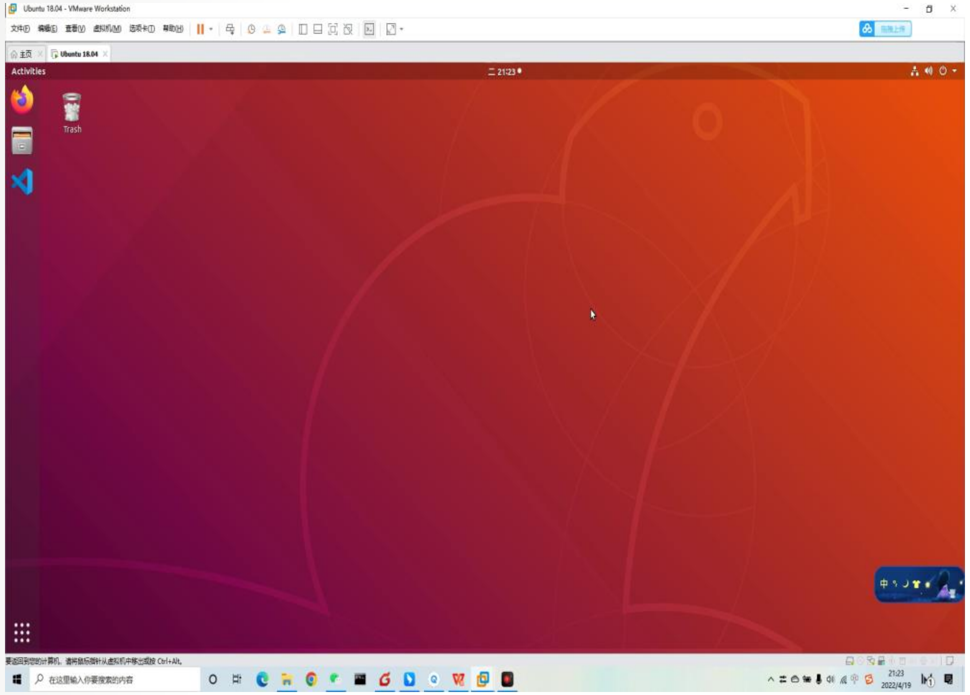
Changelog

1. use std::array at file comm.h.
2. robot command and state package add head.
3. rearrange HIGHLEVEL flag.
4. rearrange SN and version.

Dependencies

```
g++                 >= v8.3.0
Legged_sport        >= v1.36.0
firmware H0.1.7 >= v0.1.35
         H0.1.9 >= v0.1.35
```

- **Take the current latest version of the Go1 robot routine as an example, first confirm the version, as well as the sport mode and firmware version (you can also look for the current version of a more appropriate routine to use)**
- **If a different version is used, the bot will not work even if the compilation passes.**
- **Versions Legged_sport and firmware are viewed as follows:**



- Legged_sport:
- Use your pc
- Connect the wifi of robot(Goxxxxxxx, password 0000000000)
- ssh pi@192.168.12.1
- 123
- cd Unitree/autostart/02sportMode
- ./bin/Legged_sport -v

- firmware:
- Use your phone
- Connect the wifi of robot(Goxxxxxxx, password 0000000000)
- Open the Go1 app
- Click the Settings button
- Click the Status button
- If no hardware is shown here, it means the version is v0.1.35

v3.5.1

The unitree_legged_sdk is mainly used for communication between PC and Controller board. It also can be used in other PCs with UDP.

Notice

support robot: Go1

not support robot: Laikago, Aliengo, A1. (Check release v3.3.1 for support)

Sport Mode

```
Legged_sport    >= v1.36.0
firmware H0.1.7 >= v0.1.35
         H0.1.9 >= v0.1.35
```

Dependencies

- Boost (version 1.5.4 or higher)
- CMake (version 2.8.3 or higher)
- LCM (version 1.4.0 or higher)
- g++ (version 8.3.0 or higher)

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

Build

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

Usage

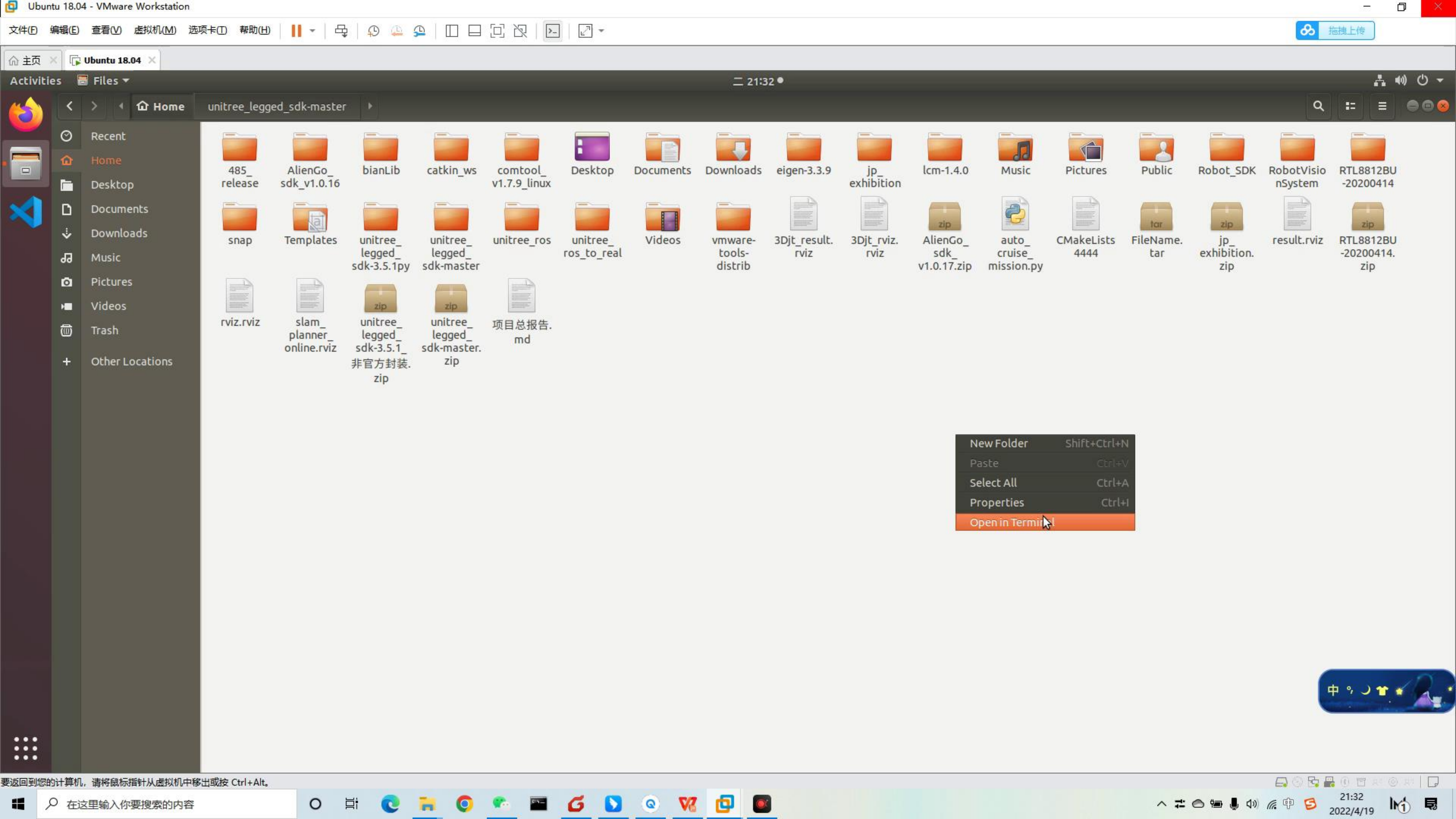
Run examples with 'sudo' for memory locking.

- Here are the libraries that the SDK depends on,
- Here shows how to install the LCM
- The steps to compile the SDK
- And the need to add sudo privileges before using it

3

PART THREE

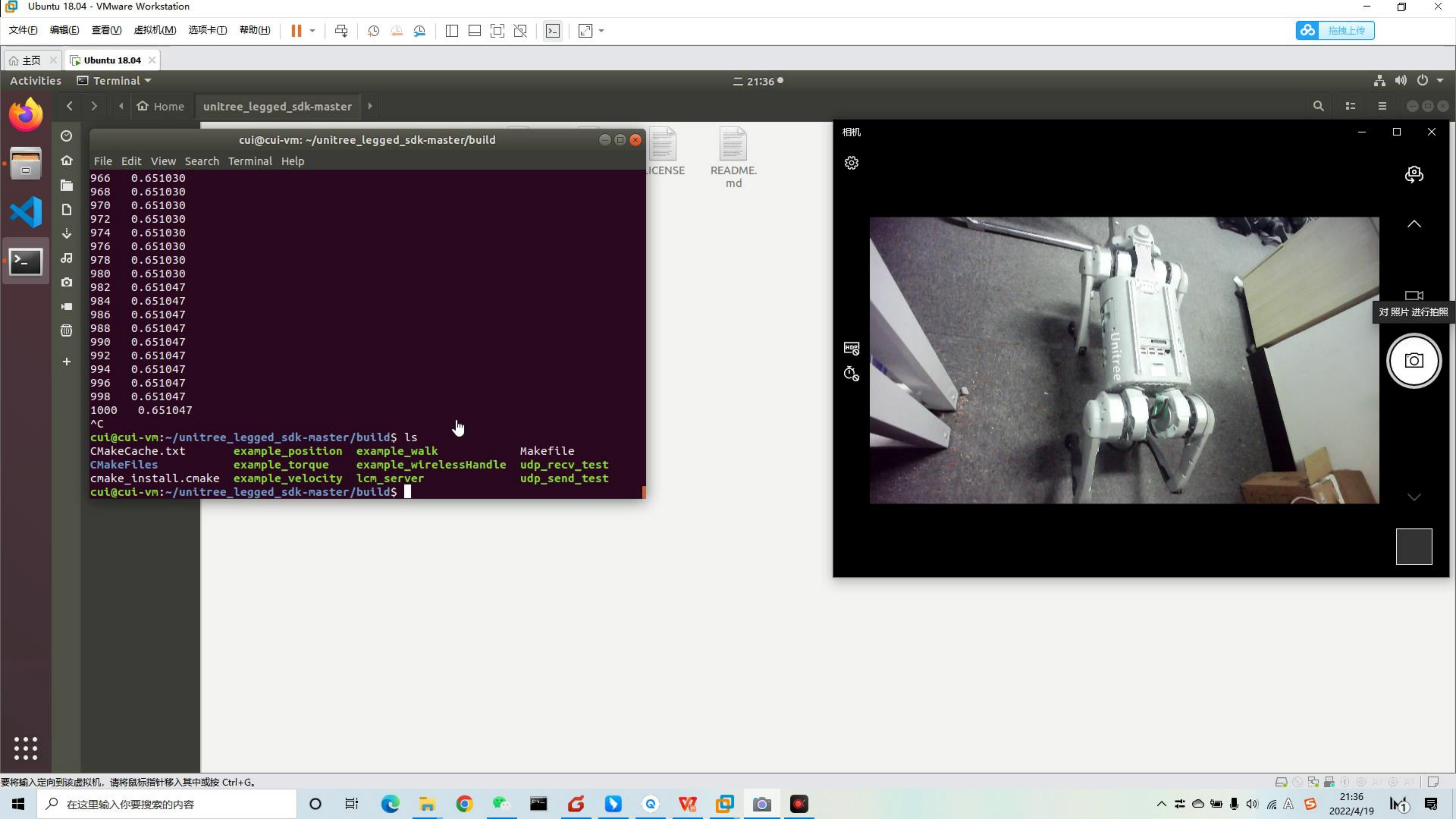
Compilation


Ubuntu 18.04 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
主页
Ubuntu 18.04
Activities
Files
21:32
Home
unitree_legged_sdk-master
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
485_release
AlienGo_sdk_v1.0.16
bianLib
catkin_ws
comtool_v1.7.9_linux
Desktop
Documents
Downloads
eigen-3.3.9
jp_exhibition
lcm-1.4.0
Music
Pictures
Public
Robot_SDK
RobotVisionSystem
RTL8812BU-20200414
snap
Templates
unitree_legged_sdk-3.5.1py
unitree_legged_sdk-master
unitree_ros
unitree_ros_to_real
Videos
vmware-tools-distrib
3Djt_result.rviz
3Djt_rviz.rviz
AlienGo_sdk_v1.0.17.zip
auto_cruise_mission.py
CMakeLists4444
FileName.tar
jp_exhibition.zip
result.rviz
RTL8812BU-20200414.zip
rviz.rviz
slam_planner_online.rviz
unitree_legged_sdk-3.5.1_非官方封装.zip
unitree_legged_sdk-master.zip
项目总报告.md
New Folder Shift+Ctrl+N
Paste Ctrl+V
Select All Ctrl+A
Properties Ctrl+I
Open in Terminal
要返回到您的计算机，请将鼠标指针从虚拟机中移出或按 Ctrl+Alt。
在这里输入你要搜索的内容
21:32
2022/4/19

4

PART FOUR

Running

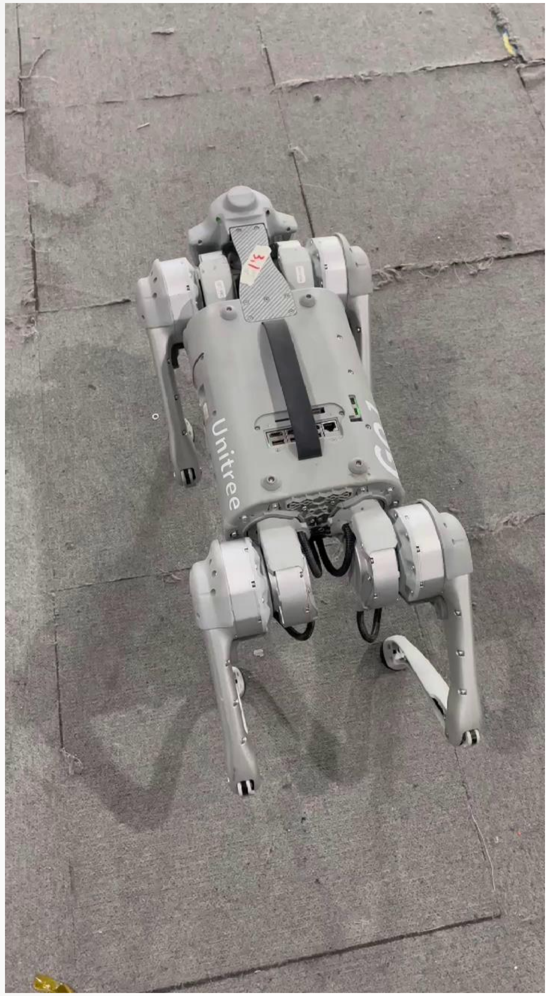

Ubuntu 18.04 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
拖拽上传
主页
Ubuntu 18.04
Activities
Terminal
21:36
Home
unitree_legged_sdk-master
cui@cui-vm: ~/unitree_legged_sdk-master/build
File Edit View Search Terminal Help
966 0.651030
968 0.651030
970 0.651030
972 0.651030
974 0.651030
976 0.651030
978 0.651030
980 0.651030
982 0.651047
984 0.651047
986 0.651047
988 0.651047
990 0.651047
992 0.651047
994 0.651047
996 0.651047
998 0.651047
1000 0.651047
^C
cui@cui-vm:~/unitree_legged_sdk-master/build$ ls
CMakeCache.txt example_position example_walk Makefile
CMakeFiles example_torque example_wirelessHandle udp_recv_test
cmake_install.cmake example_velocity lcm_server udp_send_test
cui@cui-vm:~/unitree_legged_sdk-master/build$
LICENSE
README.md
相机
对 照片 进行拍照
Unitree
要将输入定向到该虚拟机，请将鼠标指针移入其中或按 Ctrl+G。
在这里输入你要搜索的内容
21:36
2022/4/19

5

PART FIVE

Routine Analysis

HIGHLEVEL



LOWLEVEL

- Unitree provides high-level as well as bottom-level control routines as shown in the video above.
- The high-level is the robot as the control object, giving instructions related to travel speed, etc. No need to build complex dynamics controllers
- The low-level is the motors as the control object, which controls the robot through three loops: speed, position, and torque.

HIGHLEVEL

master unitree_legged_sdk / examples / example_walk.cpp

Go to file ...

TrivasZhang HighState add MotorState

Latest commit b5d6db2 on 25 Nov 2021 History

1 contributor

157 lines (141 sloc) | 3.86 KB

Raw Blame

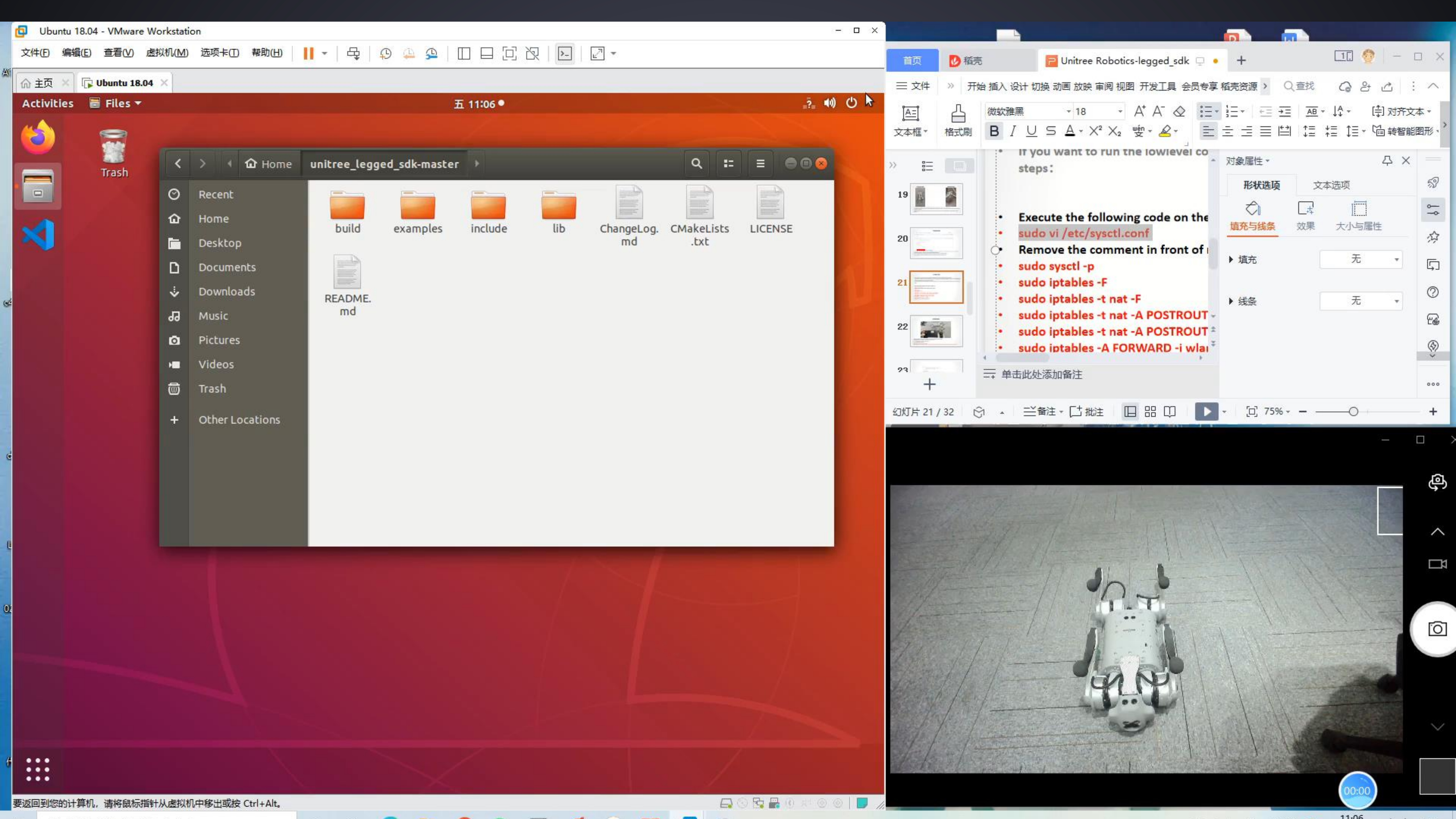
```
/************************************************************************
 Copyright (c) 2020, Unitree Robotics.Co.Ltd. All rights reserved.
************************************************************************/

#include "unitree_legged_sdk/unitree_legged_sdk.h"
#include <math.h>
#include <iostream>
#include <unistd.h>
#include <string.h>

using namespace UNITREE_LEGGED_SDK;

class Custom
{
public:
    Custom(uint8_t level):
      safe(LeggedType::Go1),
      udp(8090, "192.168.123.161", 8082, sizeof(HighCmd), sizeof(HighState))
```

- The target side of the high-level code is the Raspberry Pi, so you need to confirm whether the communication between you and the robot is a wired or wireless connection before using it
- If it is wired you need to change the ip of the location in the picture to 192.168.123.161
- If it is wireless you need to change the ip of the location in the picture to 192.168.12.1

LOWLEVEL

- If you use a wired network to use the underlying code, then the robot will be able to be controlled
- The underlying code is sending commands directly to the main control board (IP 192.168.123.10), so if you connect to the robot via wifi and then execute the underlying control code, the robot will not respondIf it is wireless you need to change the ip of the location in the picture to 192.168.12.
- If you want to run the lowlevel code on your own computer via wifi, you need to perform the following network bridging steps：

- Execute the following code on the robot's Raspberry Pi：
- sudo vi /etc/sysctl.conf
- Remove the comment in front of net.ipv4.ip_forward=1
- sudo sysctl -p
- sudo iptables -F
- sudo iptables -t nat -F
- sudo iptables -t nat -A POSTROUTING -o wlan1 -j MASQUERADE
- sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
- sudo iptables -A FORWARD -i wlan1 -o eth0 -j ACCEPT
- sudo iptables -A FORWARD -i eth0 -o wlan1 -j ACCEPT
- Do the following on your own laptop：
- sudo route add default gw 192.168.12.1
- The following is a video demonstration：


Ubuntu 18.04 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
主页
Ubuntu 18.04
Activities
Files
五 11:06
Trash
Home
unitree_legged_sdk-master
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
build
examples
include
lib
ChangeLog.md
CMakeLists.txt
LICENSE
README.md
要返回到您的计算机，请将鼠标指针从虚拟机中移出或按 Ctrl+Alt。
首页
稻壳
Unitree Robotics-legged_sdk
文件
开始 插入 设计 切换 动画 放映 审阅 视图 开发工具 会员专享 稻壳资源
查找
文本框
格式刷
微软雅黑
18
对齐文本
转智能图形
If you want to run the lowlevel co
steps:
Execute the following code on the
sudo vi /etc/sysctl.conf
Remove the comment in front of
sudo sysctl -p
sudo iptables -F
sudo iptables -t nat -F
sudo iptables -t nat -A POSTROUT
sudo iptables -t nat -A POSTROUT
sudo iptables -A FORWARD -i wla
单击此处添加备注
对象属性
形状选项
文本选项
填充与线条
效果
大小与属性
填充
无
线条
无
幻灯片 21 / 32
备注
批注
75%
00:00

LOWLEVEL

- When using the lowlevel it is necessary to switch the robot to normal mode:.
- When the robot is turned on and stands up on its own:
- 1、 L2+B(Together)---robot will get down
- 2、 L1+L2+START(Together)
- At this point you can run the routine
- **Note: Because the robot will fall over when running the low-level routine, please suspend the robot before using it**

```cpp
int main(void)
{
    std::cout << "Communication level is set to HIGH-level." << std::endl
              << "WARNING: Make sure the robot is standing on the ground." << std::endl
              << "Press Enter to continue..." << std::endl;
    std::cin.ignore();

    Custom custom(HIGHLEVEL);
    // InitEnvironment();
    LoopFunc loop_control("control_loop", custom.dt,    boost::bind(&Custom::RobotControl, &custom));
    LoopFunc loop_udpSend("udp_send",     custom.dt, 3, boost::bind(&Custom::UDPSend,      &custom));
    LoopFunc loop_udpRecv("udp_recv",     custom.dt, 3, boost::bind(&Custom::UDPRecv,      &custom));

    loop_udpSend.start();
    loop_udpRecv.start();
    loop_control.start();

    while(1){
        sleep(10);
    };
```

- Let's look directly at the main function.
- Here three threads are created and three functions of the Custom class are bound to each of them
- ‘udpSend’ and ‘udpRecv’ are used for communication
- We just put the control logic in RobotControl()

```cpp
class Custom
{
public:
    Custom(uint8_t level):
      safe(LeggedType::Go1),
      udp(8090, "192.168.123.161", 8082, sizeof(HighCmd), sizeof(HighState))
    {
        udp.InitCmdData(cmd);
    }
    void UDPRecv();
    void UDPSend();
    void RobotControl();

    Safety safe;
    UDP udp;
    HighCmd cmd = {0};
    HighState state = {0};
    int motiontime = 0;
    float dt = 0.002;     // 0.001~0.01
};
```

- We need to focus on the two structure variables in ‘Custom’ as above
- The ‘cmd’ contains commands to control the robot
- The ‘state’ provides feedback on the current status of the robot‘udpSend’ and ‘udpRecv’ are used for communication

```
void Custom::RobotControl()
{
    motiontime += 2;
    udp.GetRecv(state);
    printf("%d   %f\n", motiontime, state.imu.quaternion[2]);

    cmd.mode = 0;      // 0:idle, default stand      1:forced stand     2:walk continuously
    cmd.gaitType = 0;
    cmd.speedLevel = 0;
    cmd.footRaiseHeight = 0;
    cmd.bodyHeight = 0;
    cmd.euler[0]  = 0;
    cmd.euler[1] = 0;
    cmd.euler[2] = 0;
    cmd.velocity[0] = 0.0f;
    cmd.velocity[1] = 0.0f;
    cmd.yawSpeed = 0.0f;
    cmd.reserve = 0;
```

- GetRecv() located in RobotControl()'s top,which puts the robot's status information back into 'state', and then we can get the robot's status by accessing the members of 'state'
- Also you can export the robot's data.It can output data such as imu, foot-end sensors, etc. Details can be found in the comm.h file

```
    }
    if(motiontime > 18000 && motiontime < 20000){
        cmd.mode = 0;
        cmd.velocity[0] = 0;
    }
    if(motiontime > 20000 && motiontime < 24000){
        cmd.mode = 2;
        cmd.gaitType = 1;
        cmd.velocity[0] = 0.2f; // -1  ~ +1
        cmd.bodyHeight = 0.1;
        // printf("walk\n");
    }
    if(motiontime>24000 ){
        cmd.mode = 1;
    }

    udp.SetSend(cmd);
}
```

- **SetSend() located in RobotControl()'s bottom,which sends the commands used to control the robot out via udp**

```
if(motiontime > 1000 && motiontime < 2000){
    cmd.mode = 1;
    cmd.euler[0] = 0.3;
}
if(motiontime > 2000 && motiontime < 3000){
    cmd.mode = 1;
    cmd.euler[1] = -0.2;
}
if(motiontime > 3000 && motiontime < 4000){
    cmd.mode = 1;
    cmd.euler[1] = 0.2;
}
if(motiontime > 4000 && motiontime < 5000){
    cmd.mode = 1;
    cmd.euler[2] = -0.2;
}
if(motiontime > 5000 && motiontime < 6000){
    cmd.mode = 1;
    cmd.euler[2] = 0.2;
}
if(motiontime > 6000 && motiontime < 7000){
    cmd.mode = 1;
    cmd.bodyHeight = -0.2;
}
```

- **We put the logic to control the robot in the middle of these two functions, and the logic in the course is contained in a series of 'if' statementsp**
- **'motiontime' is a cumulative variable that enters different 'if' statements depending on its value, and then modifies the value of the corresponding cmd member according to the desired command**

6

PART SIX

Interface Analysis

comm.h

File path:unitree_legged_sdk/include/unitree_legged_sdk/comm.h

```
typedef struct
{
	std::array<uint8_t, 2> head;
	uint8_t levelFlag;
	uint8_t frameReserve;

	std::array<uint32_t, 2> SN;
	std::array<uint32_t, 2> version;
	uint16_t bandWidth;
	uint8_t mode;                    // 0. idle, default stand  1. force stand (controlled by dBodyHeight + ypr)
	                                 // 2. target velocity walking (controlled by velocity + yawSpeed)
	                                 // 3. target position walking (controlled by position + ypr[0])
	                                 // 4. path mode walking (reserve for future release)
	                                 // 5. position stand down.
	                                 // 6. position stand up
	                                 // 7. damping mode
	                                 // 8. recovery stand
	                                 // 9. backflip
	                                 // 10. jumpYaw
	                                 // 11. straightHand
	                                 // 12. dance1
	                                 // 13. dance2

	uint8_t gaitType;                // 0.idle  1.trot  2.trot running  3.climb stair  4.trot obstacle
	uint8_t speedLevel;              // 0. default low speed. 1. medium speed 2. high speed. during walking, only respond MODE 3
	float footRaiseHeight;           // (unit: m, default: 0.08m), foot up height while walking, delta value
	float bodyHeight;                // (unit: m, default: 0.28m), delta value
	std::array<float, 2> postion;    // (unit: m), desired position in inertial frame
	std::array<float, 3> euler;      // (unit: rad), roll pitch yaw in stand mode
	std::array<float, 2> velocity;   // (unit: m/s), forwardSpeed, sideSpeed in body frame
	float yawSpeed;                  // (unit: rad/s), rotateSpeed in body frame
	BmsCmd bms;
	std::array<LED, 4> led;
	std::array<uint8_t, 40> wirelessRemote;
	uint32_t reserve;

	uint32_t crc;
```

- **In this file we define each structure, and each definition allows you to understand exactly what interfaces are open to the robot.**
- **And how we assign values to the member variables of the body**

safety.h

File path:unitree_legged_sdk/include/unitree_legged_sdk/comm.h

```
class Safety{
public:
        Safety(LeggedType type);
        ~Safety();
        void PositionLimit(LowCmd&);                        // only effect under Low Level control in Position mode
        int PowerProtect(LowCmd&, LowState&, int);   /* only effect under Low Level control, input factor: 1~10,
                                                                                          means 10%~100% power limit. If you are new, then use 1; if you are familiar,
                                                                                          then can try bigger number or even comment this function. */
        int PositionProtect(LowCmd&, LowState&, double limit = 0.087);  // default limit is 5 degree
private:
        int WattLimit, Wcount;     // Watt. When limit to 100, you can triger it with 4 hands shaking.
        double Hip_max, Hip_min, Thigh_max, Thigh_min, Calf_max, Calf_min;
```

- Safety.h defines the protection function in the bottom mode of unitree_legged_sdk, which mainly contains power protection and position protection. Protection is a software way to reduce the probability of equipment failure caused by abnormal torque, it is recommended that newcomers are added, after the subsequent skilled, you can not use the protection function.
- The function in safety.h is only a software protection means, and does not guarantee that all can be effectively protected. Our bottom control needs to be careful not to output abnormally large torque.
- 'PowerProtect' is mainly power protection, using the principle of p = fv, torque * angular velocity, according to the empirical value is divided into 1-10 files, you can slowly add when using.

For any questions, please contact me at:

office: +86-0571-5671 6562

Mobile/WeChat/ Whatsapp: +86-177-6647-5895

Email:support@unitree.cc

Unitree


Unitree
宇树科技
The future is here
THANKS
It just hasn't caught on yet