古月居
GYH.AI

ROS
暑期学校

从零开始OriginBot

主讲人：李乔龙

目录

1. 课程安排

2. OriginBot概论

3. OriginBot组装

4. OriginBot基础及使用

1.课程安排

课程安排

古月居 GYH.AI

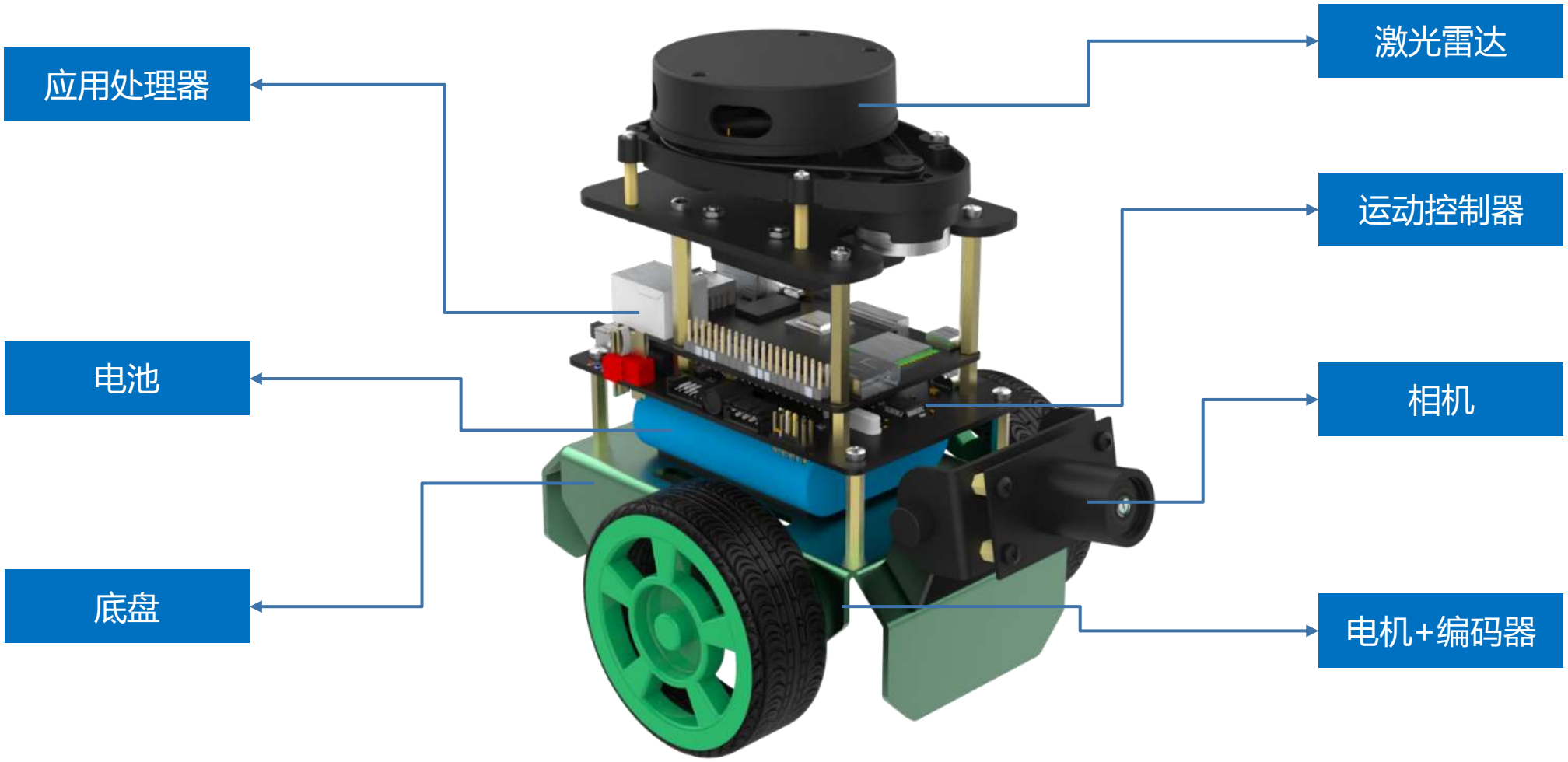
时间	课次	章节主题	授课老师	授课时间	内容要点
第一天	1	OriginBot机器人概述	李乔龙	上午	• OriginBot套件组装 • OriginBot机器人基础及使用
	2	OriginBot功能体验	李乔龙	下午	• 键盘控制 • SLAM建图实现 • 机器人自主导航 • 手势识别与姿态识别 • 人体跟踪
第二天	1	OriginBot深度学习开发	李乔龙	上午	• 深度学习基础知识 • 智能机器人深度学习环境部署
	2	智能小车巡线比赛	李乔龙	下午	• 机器学习视觉巡线比赛

2.OriginBot概述

开源共建　　全栈开发

智能应用　　源于热爱

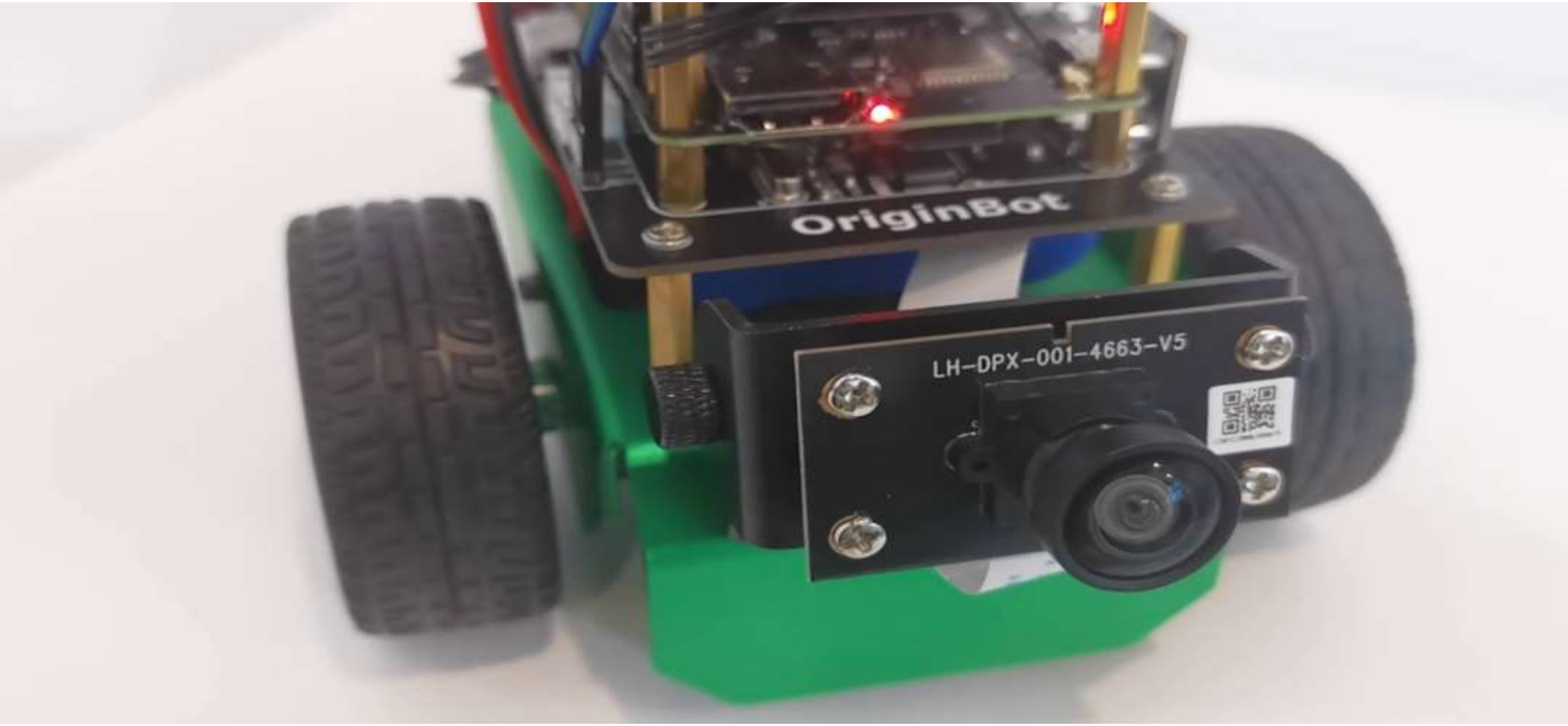
OriginBot概述



OriginBot

* 开源项目主站：https://www.originbot.org

• OriginBot概述



OriginBot

* 开源项目主站：https://www.originbot.org

3.OriginBot组装

OriginBot组装

• OriginBot组装

古月居 GYH.AI

OriginBot智能机器人开源套件

项目主页

使用指引

套件资料 >

硬件组装

软件配置 >

快速上手

基础使用 >

应用功能 >

常见问题

教学课程

参考资料

更多套件

更多内容 >

社区交流

关于我们

套件组装

Hint

安装步骤预计需要30~60分钟，可以参考以下步骤进行，或者参考套件中附带的拼装说明书。

Attention

安装过程中，请注意：

1. 固定螺丝时，请勿大力拧紧螺丝，避免滑丝；
2. 固定板卡时，请在四个铜柱或螺丝都固定之后，再分别拧紧；

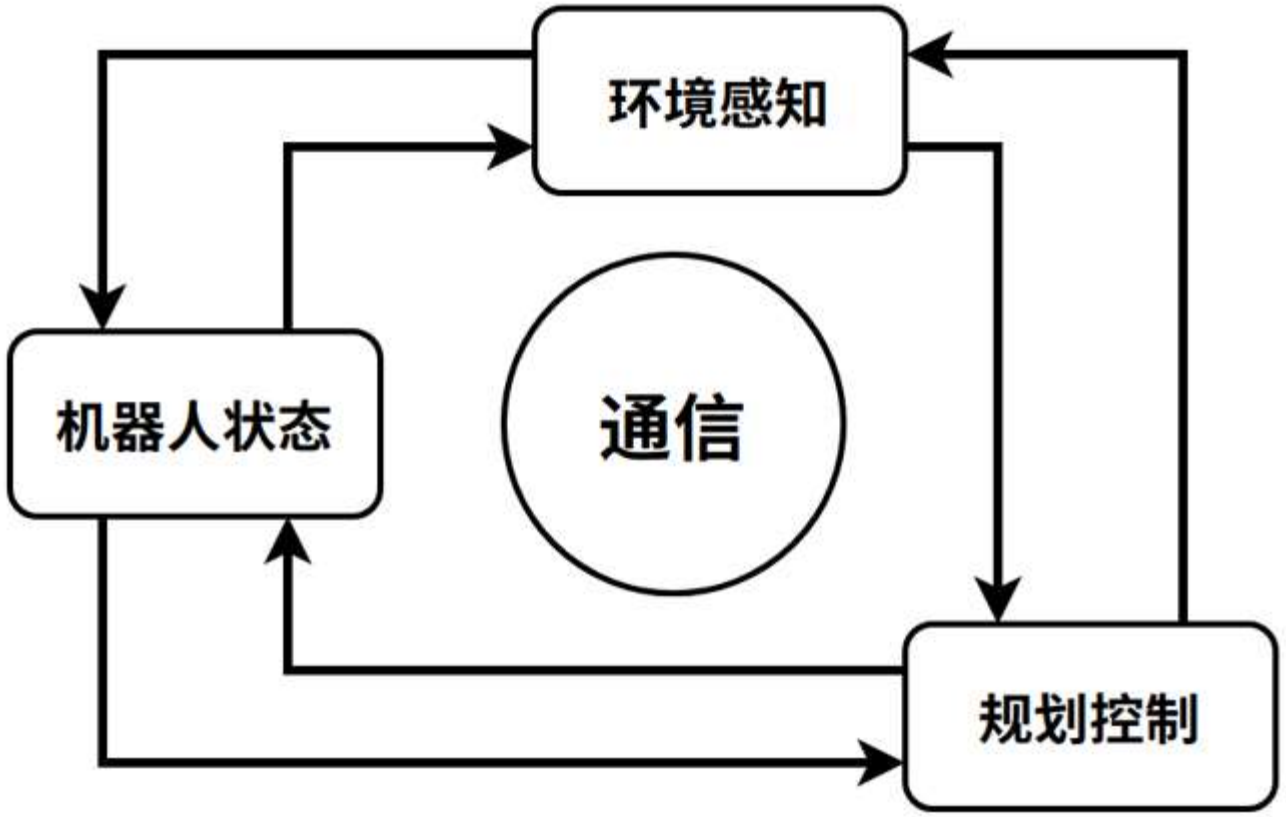
目录

1. 安装机器人底盘
   1.1 安装万向轮
   1.2 安装控制器支撑铜柱
   1.3 安装动力电池
   1.4 安装车轮
2. 安装控制器板卡
   2.1 安装控制器上层支撑铜柱
   2.2 安装控制器板卡
   2.3 连接电机线
3. 安装处理器板卡
   3.1 安装处理器散热片与天线
   3.2 安装处理器板卡
4. 安装相机
   4.1 安装相机模块
   4.2 安装相机
5. 安装激光雷达
   5.1 安装雷达串口模块
   5.2 安装雷达支架
   5.3 安装雷达亚克力板
   5.4 安装雷达
6. 连接线缆
7. 安装车牌和雷达贴纸（可选）
   7.1 粘贴车牌
   7.2 粘贴雷达贴纸

http://originbot.org/guide/hardware_setup/

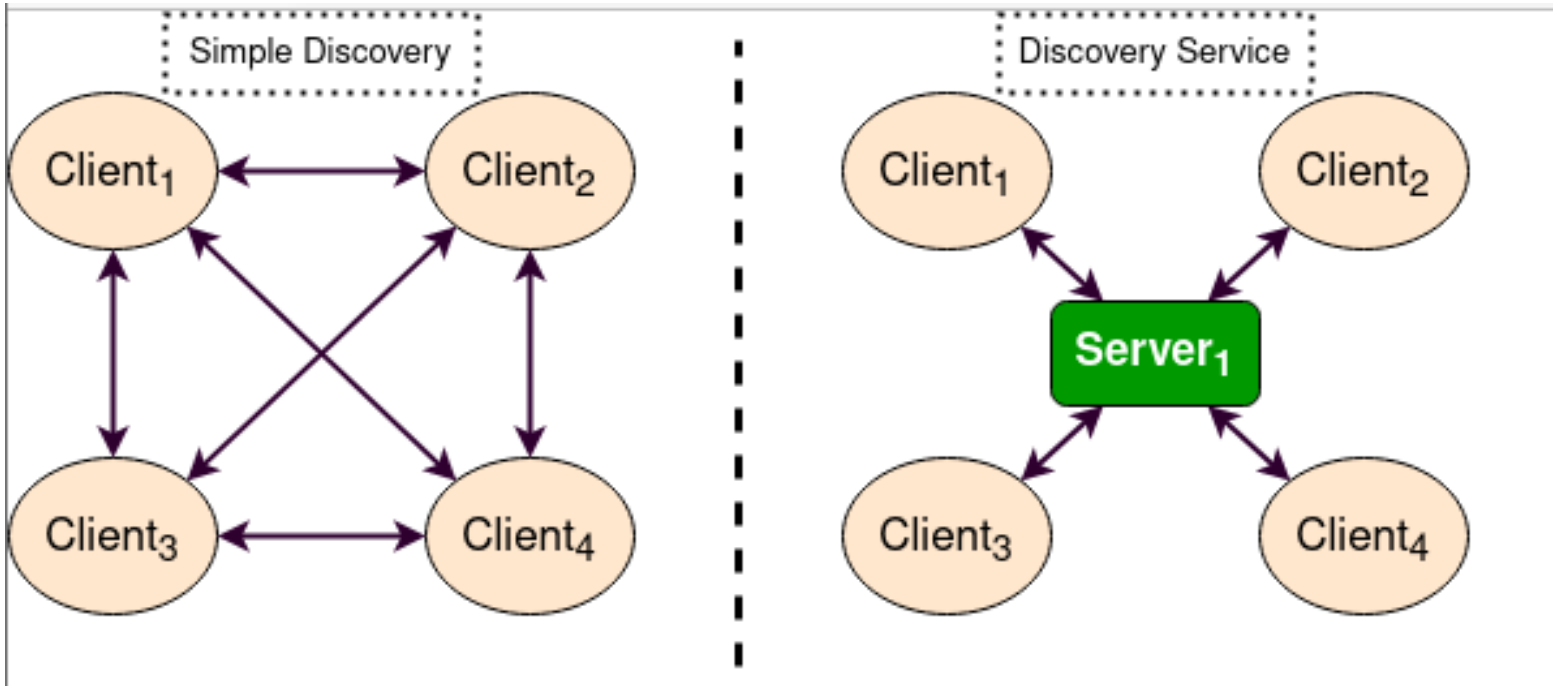
4.OriginBot基础与使用

通信是机器人软件的基础核心功能



ROS2 通信

ROS2是基于**DDS（Data Distribution Service ）**的**去中心化**通信系统



ROS2　　ROS

ROS2从架构上看具有更好的稳定性

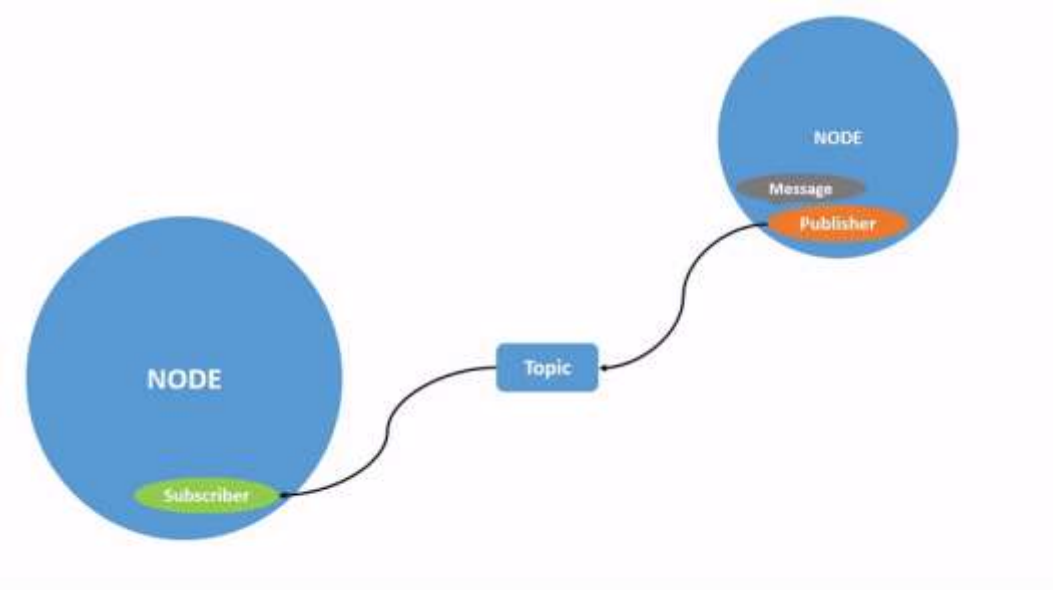
ROS2 通信

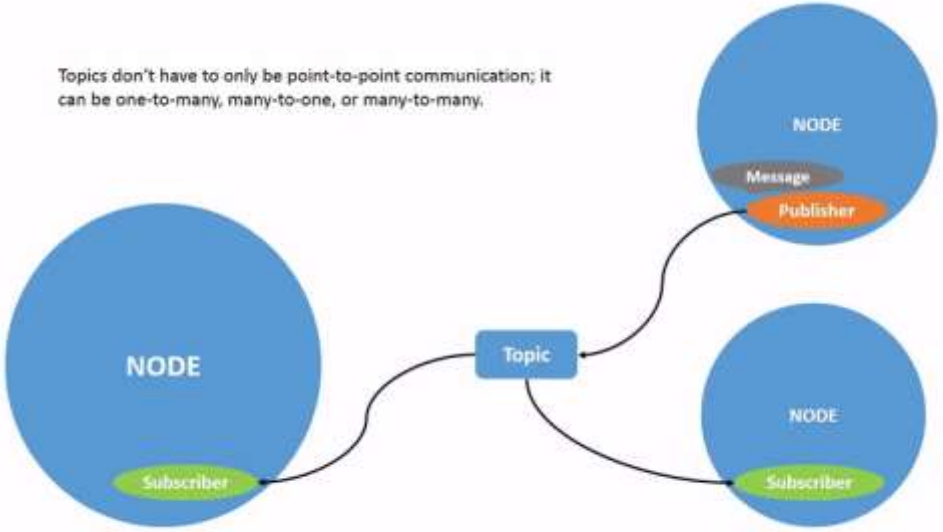
ROS2常见的通信方式主要有三种：Topic、Service、Action

Topic是最为基础和常用的通信方式

1 pub & 1 sub

M pus & N sub





创建cpp_pubsub node

ros2 pkg create --build-type ament_cmake cpp_pubsub

注意：这条创建指令需要的dev_ws/src目录下执行

```
liuxijun@lxj:~/classTest/dev_ws/src$ ros2 pkg create --build-type ament_cmake cp
p_pubsub
going to create a new package
package name: cpp_pubsub
destination directory: /home/liuxijun/classTest/dev_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['liuxijun <liuxijun@todo.todo>']
licenses: ['TODO: License declaration']
build type: ament_cmake
dependencies: []
creating folder ./cpp_pubsub
creating ./cpp_pubsub/package.xml
creating source and include folder
creating folder ./cpp_pubsub/src
creating folder ./cpp_pubsub/include/cpp_pubsub
creating ./cpp_pubsub/CMakeLists.txt
liuxijun@lxj:~/classTest/dev_ws/src$
```

```
.
├── CMakeLists.txt
├── include
├── package.xml
├── src
```

ROS2 通信-c++

古月居 GYH.AI

基础Publisher实现

引入需要使用的rclcpp模块和msg模块

```
5
6   #include "rclcpp/rclcpp.hpp"
7   #include "std_msgs/msg/string.hpp"
8
```

src > publish.cpp > main(int, char * [])

```
1   #include <chrono>
2   #include <functional>
3   #include <memory>
4   #include <string>
5
6   #include "rclcpp/rclcpp.hpp"
7   #include "std_msgs/msg/string.hpp"
8
9   using namespace std::chrono_literals;
10
11  class MinimalPublisher : public rclcpp::Node
12  {
13    public:
14      MinimalPublisher()
15      : Node("minimal_publisher"), count_(0)
16      {
17        publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
18        timer_ = this->create_wall_timer(
19        500ms, std::bind(&MinimalPublisher::timer_callback, this));
20      }
21
22    private:
23      void timer_callback()
24      {
25        auto message = std_msgs::msg::String();
26        message.data = "Hello, world! " + std::to_string(count_++);
27        RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
28        publisher_->publish(message);
29      }
30      rclcpp::TimerBase::SharedPtr timer_;
31      rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
32      size_t count_;
33  };
34
35  int main(int argc, char * argv[])
36  {
37    rclcpp::init(argc, argv);
38    rclcpp::spin(std::make_shared<MinimalPublisher>());
39    rclcpp::shutdown();
40    return 0;
41  }
```

ROS2 通信-c++

通过继承创建MinimalPublisher

```
class MinimalPublisher : public rclcpp::Node
```

MinimalPublisher构造函数

```
MinimalPublisher()
: Node("minimal_publisher"), count_(0)
{
  publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
  timer_ = this->create_wall_timer(
  500ms, std::bind(&MinimalPublisher::timer_callback, this));
}
```

src > publish.cpp > main(int, char * [])

```
1  #include <chrono>
2  #include <functional>
3  #include <memory>
4  #include <string>
5
6  #include "rclcpp/rclcpp.hpp"
7  #include "std_msgs/msg/string.hpp"
8
9  using namespace std::chrono_literals;
10
11 class MinimalPublisher : public rclcpp::Node
12 {
13   public:
14     MinimalPublisher()
15     : Node("minimal_publisher"), count_(0)
16     {
17       publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
18       timer_ = this->create_wall_timer(
19       500ms, std::bind(&MinimalPublisher::timer_callback, this));
20     }
21
22   private:
23     void timer_callback()
24     {
25       auto message = std_msgs::msg::String();
26       message.data = "Hello, world! " + std::to_string(count_++);
27       RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
28       publisher_->publish(message);
29     }
30     rclcpp::TimerBase::SharedPtr timer_;
31     rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
32     size_t count_;
33 };
34
35 int main(int argc, char * argv[])
36 {
37   rclcpp::init(argc, argv);
38   rclcpp::spin(std::make_shared<MinimalPublisher>());
39   rclcpp::shutdown();
40   return 0;
41 }
```

ROS2 通信-c++

定义timer回调函数

在回调函数中实现message的填充和pub

```cpp
void timer_callback()
{
  auto message = std_msgs::msg::String();
  message.data = "Hello, world! " + std::to_string(count_++);
  RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
  publisher_->publish(message);
}
```

src > publish.cpp > main(int, char * [])

```cpp
#include <chrono>
#include <functional>
#include <memory>
#include <string>

#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

using namespace std::chrono_literals;

class MinimalPublisher : public rclcpp::Node
{
  public:
    MinimalPublisher()
    : Node("minimal_publisher"), count_(0)
    {
      publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
      timer_ = this->create_wall_timer(
      500ms, std::bind(&MinimalPublisher::timer_callback, this));
    }

  private:
    void timer_callback()
    {
      auto message = std_msgs::msg::String();
      message.data = "Hello, world! " + std::to_string(count_++);
      RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
      publisher_->publish(message);
    }
    rclcpp::TimerBase::SharedPtr timer_;
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
    size_t count_;
};

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalPublisher>());
  rclcpp::shutdown();
  return 0;
}
```

ROS2 通信-c++

main函数

初始化rclcpp
创建minimal_publisher
调用spin

```cpp
int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalPublisher>());
  rclcpp::shutdown();
  return 0;
}
```

src > publish.cpp > main(int, char * [])

```cpp
#include <chrono>
#include <functional>
#include <memory>
#include <string>

#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"

using namespace std::chrono_literals;

class MinimalPublisher : public rclcpp::Node
{
  public:
    MinimalPublisher()
    : Node("minimal_publisher"), count_(0)
    {
      publisher_ = this->create_publisher<std_msgs::msg::String>("topic", 10);
      timer_ = this->create_wall_timer(
      500ms, std::bind(&MinimalPublisher::timer_callback, this));
    }

  private:
    void timer_callback()
    {
      auto message = std_msgs::msg::String();
      message.data = "Hello, world! " + std::to_string(count_++);
      RCLCPP_INFO(this->get_logger(), "Publishing: '%s'", message.data.c_str());
      publisher_->publish(message);
    }
    rclcpp::TimerBase::SharedPtr timer_;
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publisher_;
    size_t count_;
};

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalPublisher>());
  rclcpp::shutdown();
  return 0;
}
```

ROS2 通信-c++

在package.xml文件中添加依赖信息

```
<depend>rclcpp</depend>
<depend>std_msgs</depend>
```

package.xml

```xml
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>cpp_pubsub</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="liuxijun@todo.todo">liuxijun</maintainer>
  <license>TODO: License declaration</license>

  <buildtool_depend>ament_cmake</buildtool_depend>

  <test_depend>ament_lint_auto</test_depend>
  <test_depend>ament_lint_common</test_depend>

  <depend>rclcpp</depend>
  <depend>std_msgs</depend>

  <export>
    <build_type>ament_cmake</build_type>
  </export>
</package>
```

ROS2 通信-c++

在CmakeLists.txt中配置信息

```
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
# uncomment the following section i
```

```
add_executable(talker src/publish.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

install(TARGETS
  talker
  DESTINATION lib/${PROJECT_NAME})
```

M CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(cpp_pubsub)

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

add_executable(talker src/publish.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

add_executable(listener src/subscription.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})

ament_package()
```

ROS2 通信-c++

基础Subscriber实现

创建subscriber

- 订阅的 topic
- msg
- callback函数

```
subscription_ = this->create_subscription<std_msgs::msg::String>(
"topic", 10, std::bind(&MinimalSubscriber::topic_callback, this, _1));
```

src > subscription.cpp > main(int, char * [])

```
1  #include <memory>
2
3  #include "rclcpp/rclcpp.hpp"
4  #include "std_msgs/msg/string.hpp"
5  using std::placeholders::_1;
6
7  class MinimalSubscriber : public rclcpp::Node
8  {
9    public:
10     MinimalSubscriber()
11     : Node("minimal_subscriber")
12     {
13       subscription_ = this->create_subscription<std_msgs::msg::String>(
14       "topic", 10, std::bind(&MinimalSubscriber::topic_callback, this, _1));
15     }
16
17   private:
18     void topic_callback(const std_msgs::msg::String::SharedPtr msg) const
19     {
20       RCLCPP_INFO(this->get_logger(), "I heard: '%s'", msg->data.c_str());
21     }
22     rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
23 };
24
25 int main(int argc, char * argv[])
26 {
27   rclcpp::init(argc, argv);
28   rclcpp::spin(std::make_shared<MinimalSubscriber>());
29   rclcpp::shutdown();
30   return 0;
31 }
```

ROS2 通信-c++

基础Subscriber实现

callback函数实现

```cpp
void topic_callback(const std_msgs::msg::String::SharedPtr msg) const
{
  RCLCPP_INFO(this->get_logger(), "I heard: '%s'", msg->data.c_str());
}
```

src > subscription.cpp > main(int, char * [])

```cpp
#include <memory>

#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
using std::placeholders::_1;

class MinimalSubscriber : public rclcpp::Node
{
  public:
    MinimalSubscriber()
    : Node("minimal_subscriber")
    {
      subscription_ = this->create_subscription<std_msgs::msg::String>(
      "topic", 10, std::bind(&MinimalSubscriber::topic_callback, this, _1));
    }

  private:
    void topic_callback(const std_msgs::msg::String::SharedPtr msg) const
    {
      RCLCPP_INFO(this->get_logger(), "I heard: '%s'", msg->data.c_str());
    }
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr subscription_;
};

int main(int argc, char * argv[])
{
  rclcpp::init(argc, argv);
  rclcpp::spin(std::make_shared<MinimalSubscriber>());
  rclcpp::shutdown();
  return 0;
}
```

ROS2 通信-c++

在CMakeLists.txt中配置

```
add_executable(listener src/subscription.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})

```

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(cpp_pubsub)

# Default to C++14
if(NOT CMAKE_CXX_STANDARD)
  set(CMAKE_CXX_STANDARD 14)
endif()

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# find dependencies
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)

add_executable(talker src/publish.cpp)
ament_target_dependencies(talker rclcpp std_msgs)

add_executable(listener src/subscription.cpp)
ament_target_dependencies(listener rclcpp std_msgs)

install(TARGETS
  talker
  listener
  DESTINATION lib/${PROJECT_NAME})

ament_package()
```

ROS2 通信-c++

编译sample code

colcon build --packages-select cpp_pubsub

运行sample code

. install/setup.bash(或者是. install/setup.sh)

ros2 run cpp_pubsub talker

ros2 run cpp_pubsub listener

```
[INFO] [minimal_publisher]: Publishing: "Hello World: 0"
[INFO] [minimal_publisher]: Publishing: "Hello World: 1"
[INFO] [minimal_publisher]: Publishing: "Hello World: 2"
[INFO] [minimal_publisher]: Publishing: "Hello World: 3"
[INFO] [minimal_publisher]: Publishing: "Hello World: 4"
...
```

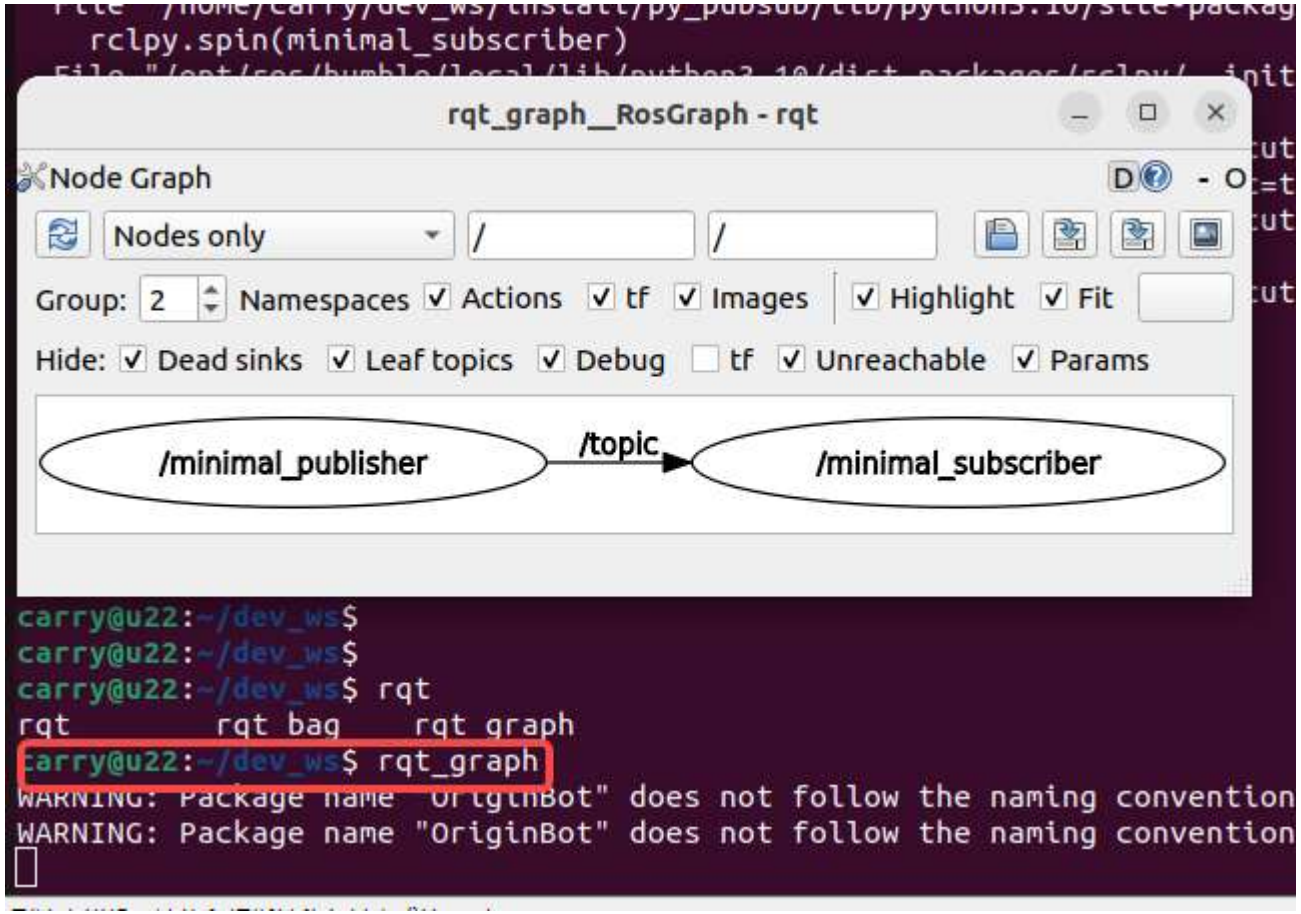
talker

```
[INFO] [minimal_subscriber]: I heard: "Hello World: 10"
[INFO] [minimal_subscriber]: I heard: "Hello World: 11"
[INFO] [minimal_subscriber]: I heard: "Hello World: 12"
[INFO] [minimal_subscriber]: I heard: "Hello World: 13"
[INFO] [minimal_subscriber]: I heard: "Hello World: 14"
```

listener

ROS2 通信

PC Ubuntu上运行 rqt_graph



ROS2 通信

PC Ubuntu上运行

ros2 topic list ##查看话题列表

```
carry@u22:~/dev_ws$
carry@u22:~/dev_ws$ ros2 topic list
/parameter_events
/rosout
/topic
carry@u22:~/dev_ws$
```

ros2 topic echo /topic

```
carry@u22:~/dev_ws$ ros2 topic echo /topic
data: 'Hello World: 458'
---
data: 'Hello World: 459'
---
data: 'Hello World: 460'
---
data: 'Hello World: 461'
---
data: 'Hello World: 462'
---
data: 'Hello World: 463'
---
data: 'Hello World: 464'
---
```

ros2 topic pub -r 1 /topic std_msgs/msg/String "{data: 'ni hao a'}"

```
carry@u22:~/dev_ws$ ros2 topic pub -r 1 /topic std_msgs/msg/String "{data: 'ni hao a'}"
publisher: beginning loop
publishing #1: std_msgs.msg.String(data='ni hao a')

publishing #2: std_msgs.msg.String(data='ni hao a')

publishing #3: std_msgs.msg.String(data='ni hao a')
```

机器人

控制系统

- 运动控制
- 人机交互
- 系统监控
- 图像处理
- 自主导航
- 机器学习
- 仿真系统
- 数据可视化

驱动系统

- 电机驱动
- 伺服驱动
- 气压驱动
- 传感驱动
- 外设驱动

执行机构

- 电机
- 伺服
- 传动
- 吸盘
- 关节

传感系统

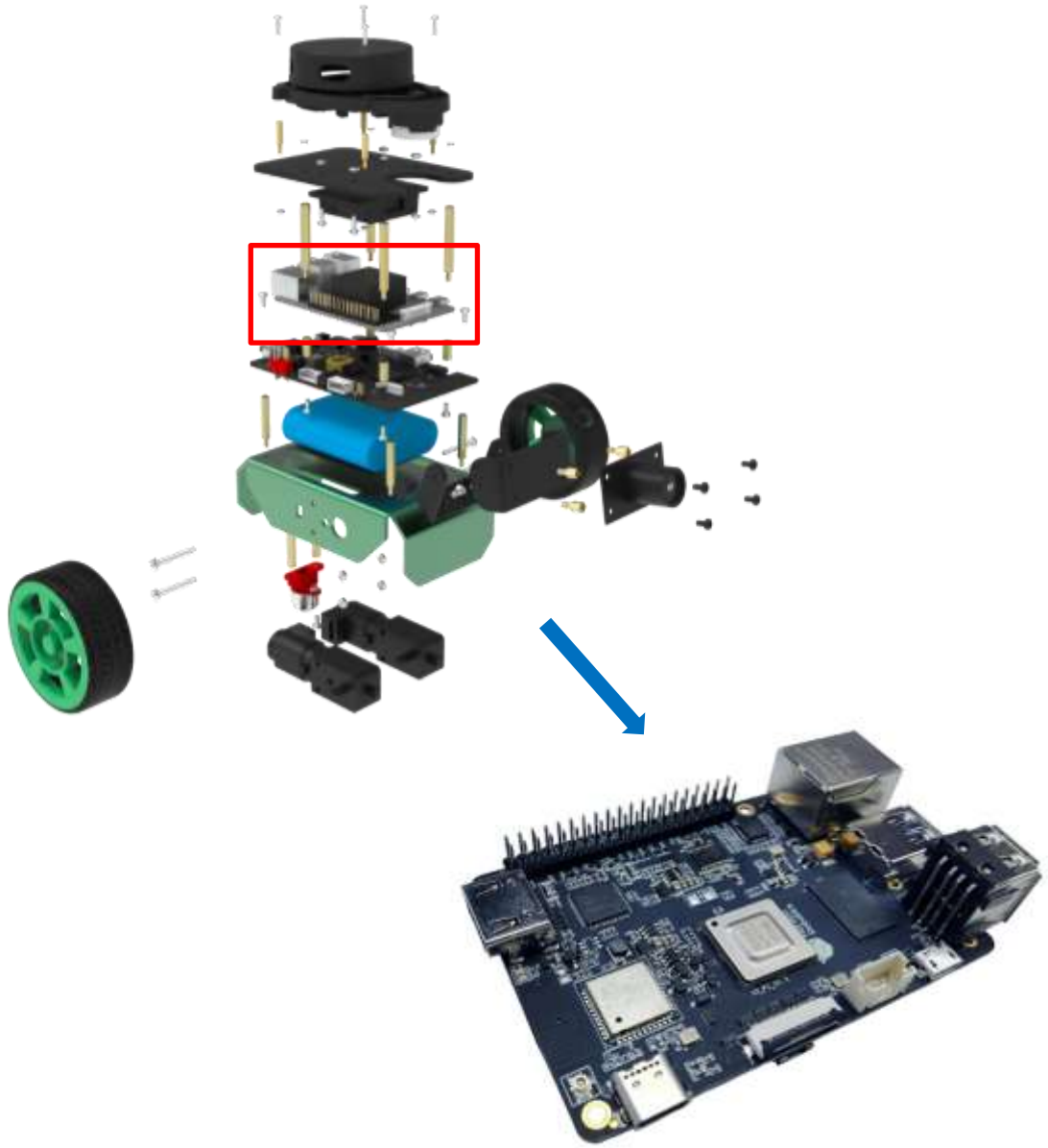
内部

- 里程计
- 陀螺仪
- 加速度计
- 力传感器

外部

- 摄像头
- 红外
- 距离感应
- 麦克风

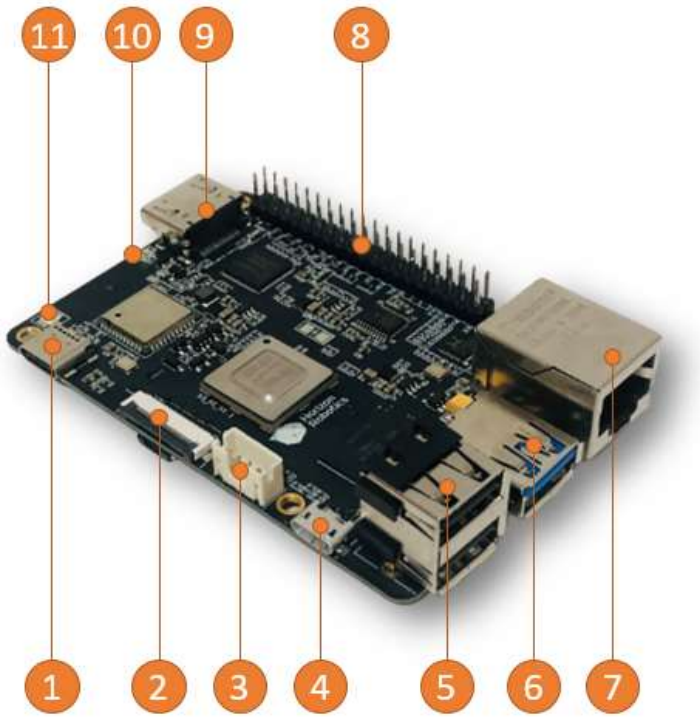
控制系统设计



CPU	ARM Cortex-A53，4*Core，1.2GHZ	
BPU	Bernoulli Arch, 2*Core, up to1.0G，~5Tops	
内存	2GB/4GB LPDDR4 RAM	
存储	SD Card Support	
多媒体	H.265(HEVC) Encoding and Decoding Main Profile, L5.1 4K@60fps	
	H.264 Encoding and Decoding Baseline/Main/High Profile, L5.1 4K@60fps	
	JPEG Encoding and Decoding 16Mpixels	
	CBR, VBR, AVBR, FixQp and QpMap bitrate control	
接口	Sensor接口	1*MIPS-CSI (2*2lane)
	USB	Host: 1*USB3.0 + 2*USB2.0 Device: 1*MicroUSB2.0
	串口	UART 0 Debug
	显示接口	1*HDMI1.4 1*4 lane MIPI-DSI(up to 1920*1080)
	麦克风	音频配件
	Micro SD	系统盘，Micro SD接口（SDSC/SDHC/SDXC）
	无线网络接口	WiFi/Bluetooth二合一模块 WiFi 2.4GHZ/Bluetooth (working distance: 5m)
	有线网络接口	RJ45 (100/1000M) * 1
	40PIN接口	GPIO&I2C&UART&SPI&I2S&PWM
电源	电源输入	TYPE C供电5V-3A (支持QC&PD适配器5V输入)
	电源输出	5V 1A & 3.3V 1A
系统支持	OS	Ubuntu 20.04
	固件升级	本地SD卡升级

* 参考链接：https://hhp.guyuehome.com/

• OriginBot基础及使用



序号	功能	序号	功能	序号	功能
1	USB Type C 供电接口	6	USB 3.0 Type A接口	11	Wi-Fi天线接口
2	MIPI CSI 摄像头接口	7	千兆以太网口	12	TF卡接口（底面）
3	调试串口	8	40PIN接口	13	MIPI 接口的LCD屏接口
4	Micro USB 2.0 接口	9	HDMI接口	14	触摸屏接口
5	USB 2.0 Type A接口两路	10	电源和状态LED指示灯		

旭日X3派 40Pin 引脚对照表

功能说明	X3管脚号	BCM编码	CVM功能名	物理引脚BOARD编码		CVM功能名	BCM编码	X3管脚号	功能说明
3.3V电源信号			VDD_3V3	1	2	VDD_5V			5V电源信号
I2C0数据信号	9	2	I2C0_SDA	3	4	VDD_5V			5V电源信号
I2C0时钟信号	8	3	I2C0_SCL	5	6	GND			地信号
I2S0 MCLK时钟信号	101	4	I2S0_MCLK	7	8	UART_TXD	14	111	UART3发送信号
地信号			GND	9	10	UART_RXD	15	112	UART3接收信号
GPIO6信号	6	17	GPIO6	11	12	I2S0_BCLK	18	102	I2S0 BCLK时钟信号
GPIO5信号	5	27	GPIO5	13	14	GND			地信号
GPIO30信号	30	22	GPIO30	15	16	GPIO27	23	27	GPIO27信号
3.3V电源信号			VDD_3V3	17	18	GPIO7	24	7	GPIO7信号
SPI2 MOSI信号	12	10	SPI2_MOSI	19	20	GND			地信号
SPI2 MISO信号	13	9	SPI2_MISO	21	22	GPIO29	25	29	GPIO29信号
SPI2 CLK信号	14	11	SPI2_SCLK	23	24	SPI2_CSN	8	15	SPI2 CS信号
地信号			GND	25	26	GPIO28	7	28	GPIO28信号
I2S1 BCLK时钟信号	106	0	I2S1_BCLK	27	28	I2S1_LRCK	1	107	I2S1 LRCLK时钟信号
GPIO119信号	119	5	GPIO119	29	30	GND			地信号
GPIO118信号	118	6	GPIO118	31	32	PWM4	12	25	PWM4信号
PWM0信号	4	13	PWM0	33	34	GND			地信号
I2S0 LRCK信号	103	19	I2S0_LRCK	35	36	GPIO3	16	3	GPIO3信号
GPIO105信号	105	26	GPIO105	37	38	I2S0_SDIO	20	104	I2S0_SDIO信号
地信号			GND	39	40	I2S1_SDIO	21	108	I2S1_SDIO信号

常用开发方法介绍—镜像烧录

1. 下载最新的旭日X3派系统镜像（http://originbot.org/guide/image_install/）

2. 使用读卡器将SD卡插入计算机，SD卡容量建议≥16GB；

3. 启动镜像烧写软件（以Rufus为例），配置关键参数。

4. 等待烧录完成。

获取必须软件

软件获取地址：http://originbot.org/material/common_software/

originbot.org/material/common_software/

OriginBot智能机器人开源套件　搜索

OriginBot智能机器人开源套件
项目主页
使用指引
套件资料
套件清单
资料链接
接口说明
常用软件
硬件组装
软件配置
快速上手
基础使用
应用功能
常见问题
教学课程
参考资料
更多内容
社区交流
关于我们

常用软件下载

软件名称	下载链接	说明
rufus	点击下载	SD卡镜像烧写工具
balenaEtcher	点击下载	SD卡镜像烧写工具
MobaXterm	点击下载	远程访问工具箱
串口驱动	点击下载	串口模块的windows驱动， CP210x_USB2UART_Driver
FlyMcu	点击下载	控制器固件下载软件

古月居 GYH.AI

怕什么真理无穷，进一寸有一寸的欢喜

常用开发方法介绍-串口通信

串口通信：**按位（bit）**发送和接收字节

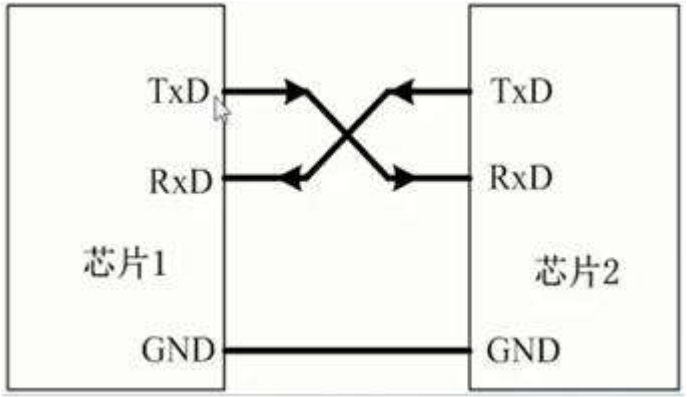
关键参数

波特率：单位时间内载波参数变化的次数，代表传输速率

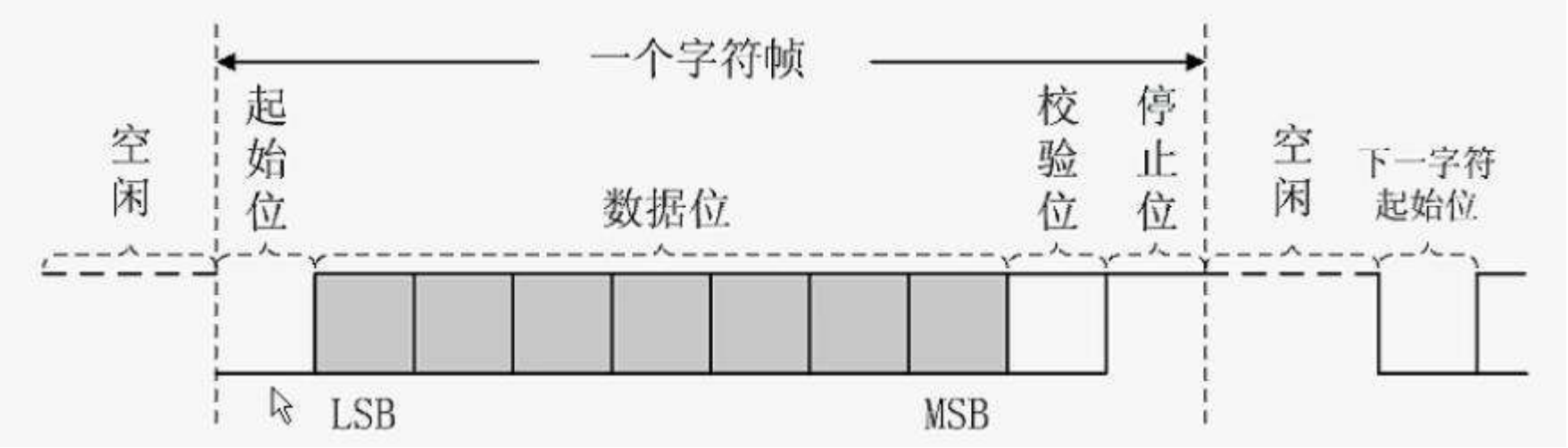
数据位：通信中实际数据位的个数，一般为6、7或8bit

停止位：用于表示单个包的最后一位。典型的值为1，1.5和2位。

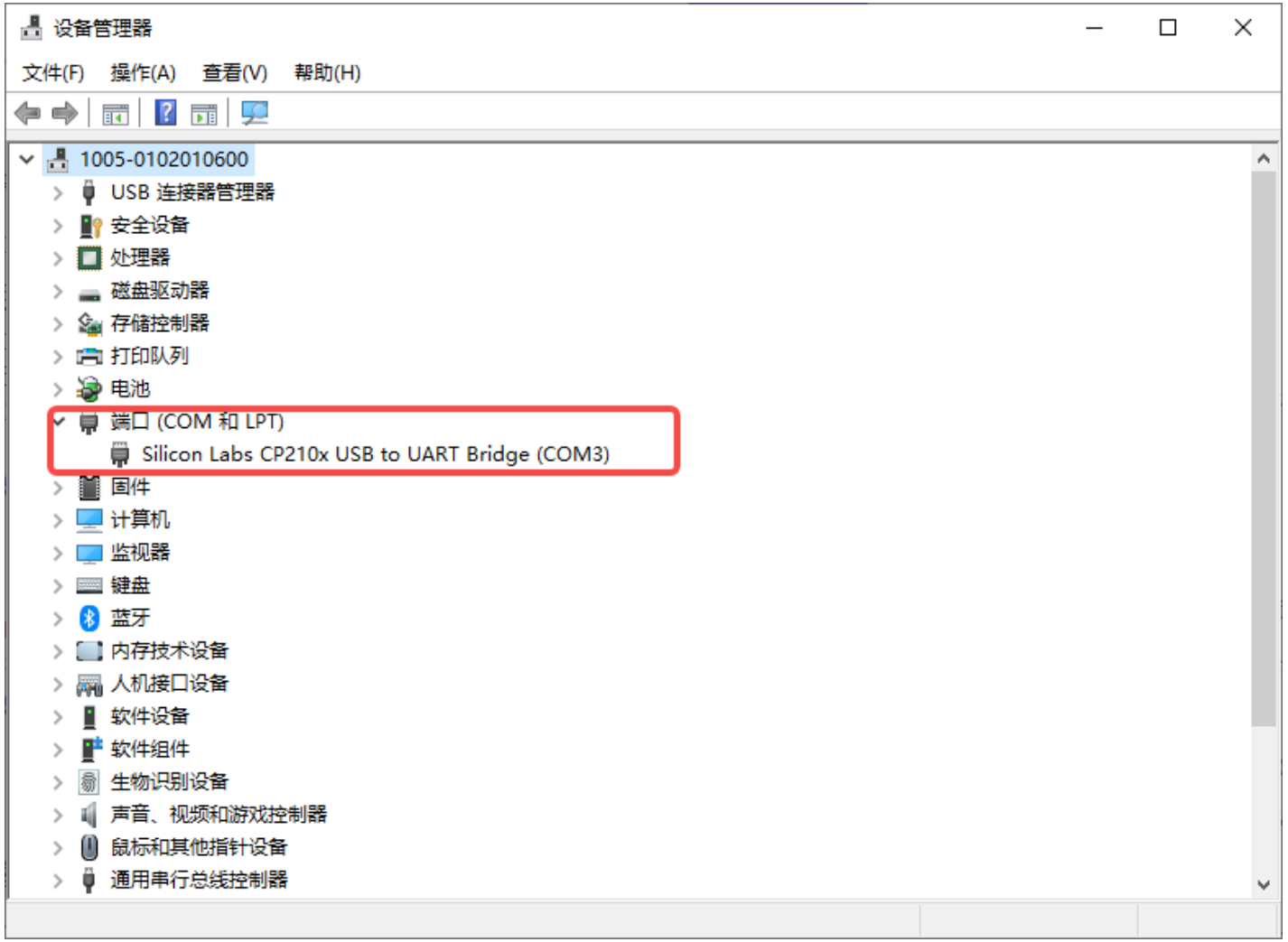
奇偶校验位：在串口通信中一种简单的检错方式。对于偶和奇校验的情况，串口会设置校验位（数据位后面的一位），用一个值确保传输的数据有偶个或者奇个逻辑高位。



常用开发方法介绍-串口通信



安装串口驱动（CP210x_USB2UART_Driver）



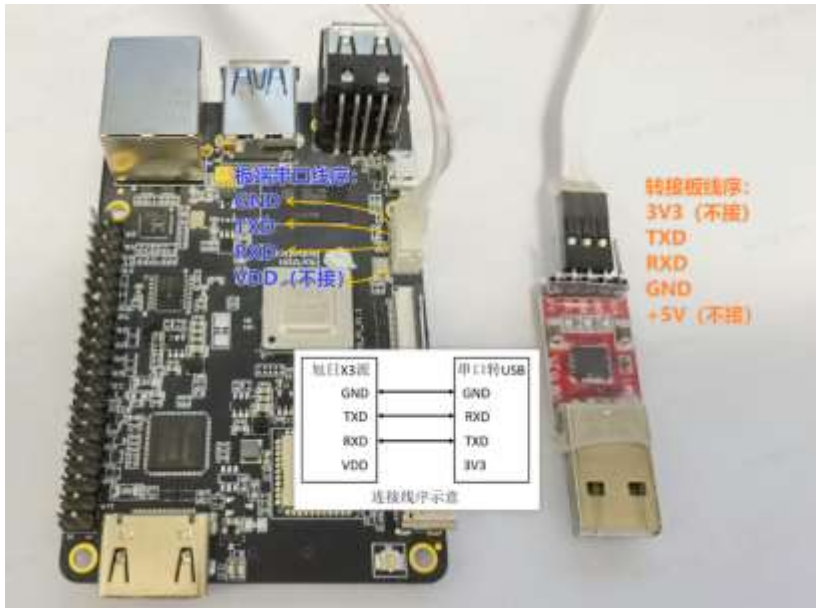
旭日X3派系统配置

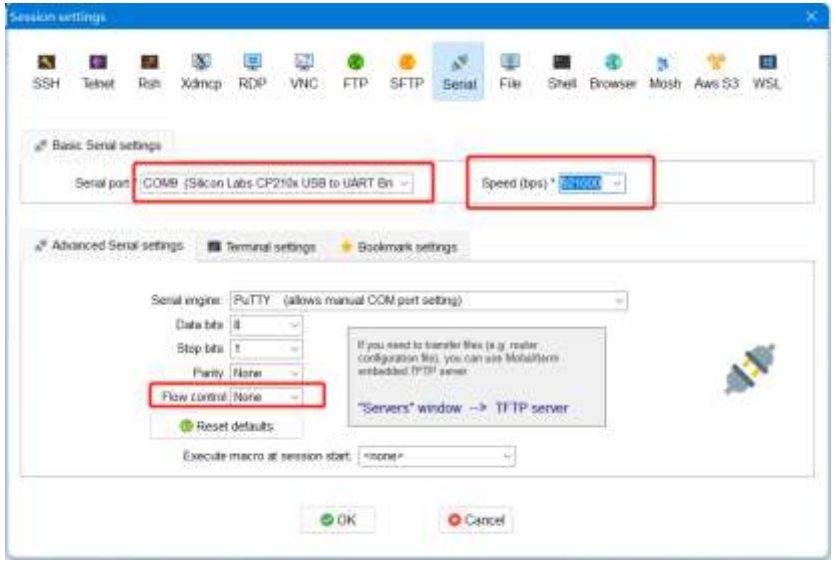
1. 安装 MobaXterm

2. 将烧写好的SD卡插入旭日X3派，将串口转接板与旭日X3派正确连接。**注意RX、TX连接方式。**

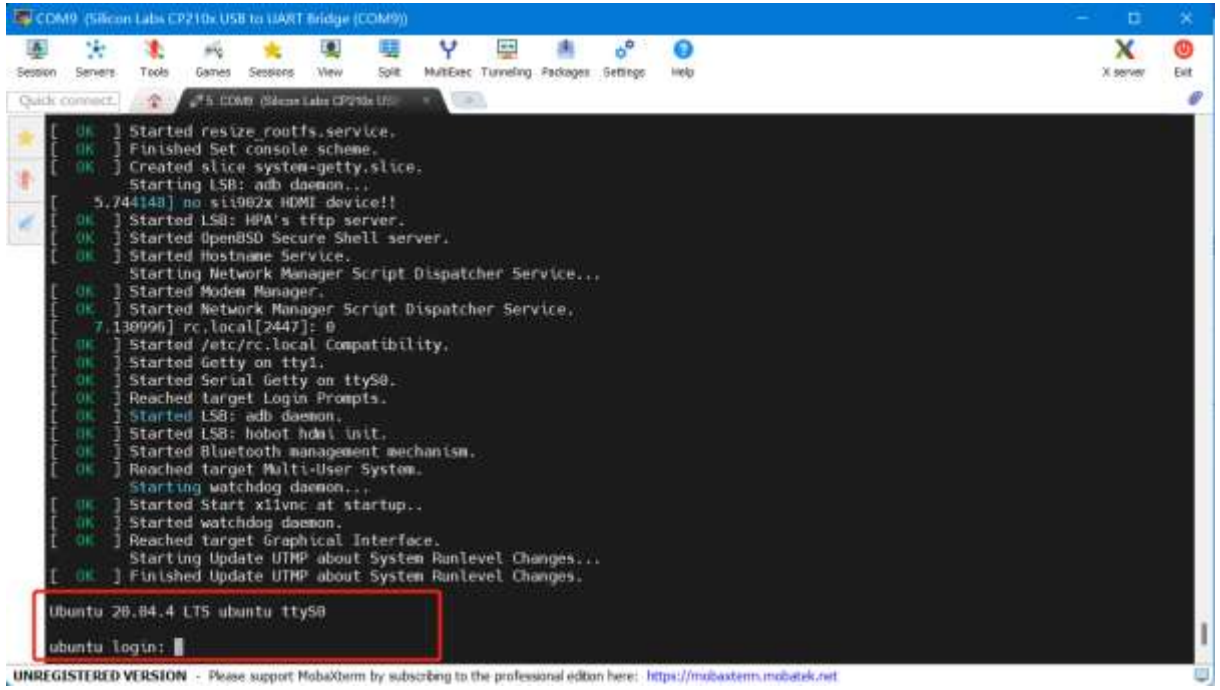
3. Session->new session。设置MobaXterm串口通信参数

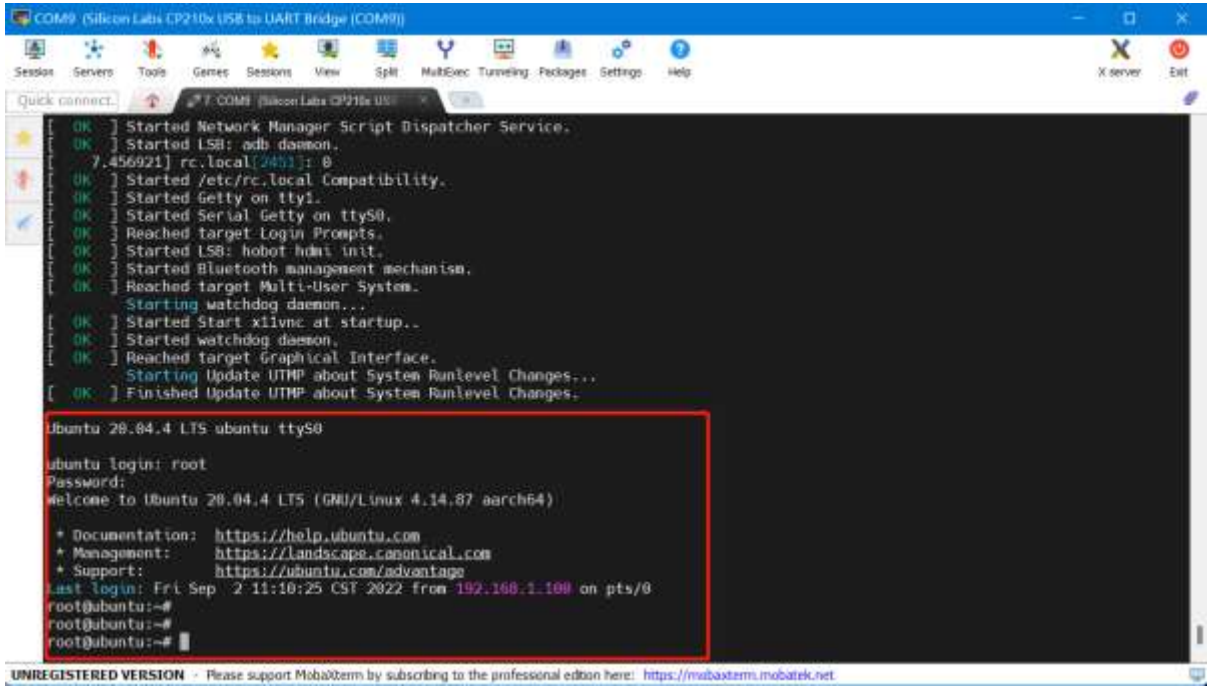
4. 打开OriginBot电源，查看串口信息





运行环境配置





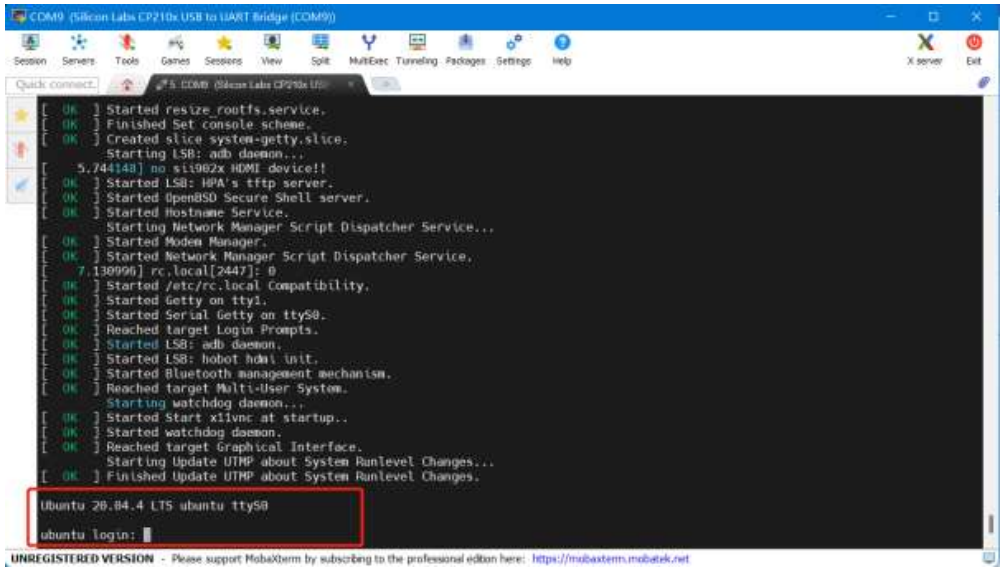
1. 输入用户名root
2. 输入密码root

运行环境配置

1. 扩展SD卡空间

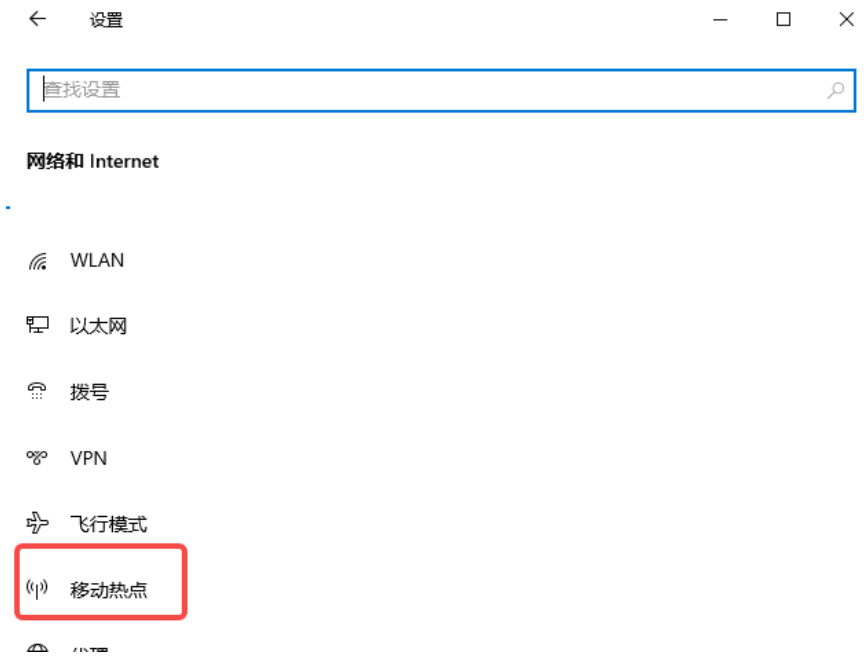
```
$ sudo growpart /dev/mmcblk2 1
$ sudo resize2fs /dev/mmcblk2p1
```

2. 输入 df –h 查看文件系统




```
root@ubuntu:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        30G  8.9G   21G  31% /
devtmpfs        1.6G     0  1.6G   0% /dev
tmpfs           2.0G     0  2.0G   0% /dev/shm
tmpfs           394M  1.2M  393M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
tmpfs           394M     0  394M   0% /run/user/0
root@ubuntu:~#
```

利用笔记本创建wifi热点



运行环境配置

1. 进行wifi连接

```
$ sudo nmcli device wifi rescan        # 扫描wifi网络
$ sudo nmcli device wifi list          # 列出找到的wifi网络
$ sudo wifi_connect "SSID" "PASSWD"    # 连接某指定的wifi网络
```

2. 输入 ifcong 查看IP地址

```
root@ubuntu:~# sudo nmcli device wifi rescan
root@ubuntu:~# sudo nmcli device wifi list
IN-USE  BSSID              SSID           MODE   CHAN  RATE        SIGNAL  BARS  SECURITY
        A2:9D:7E:55:0A:AA  --             Infra  2     130 Mbit/s  94            --
        50:2D:BB:D0:0B:7A  midea_ca_0019  Infra  2     65 Mbit/s   82            WPA2
        34:FC:A1:9C:A7:AB  602            Infra  1     130 Mbit/s  79            WPA1 WPA2
*       9C:9D:7E:55:0A:AA  XH-Home        Infra  2     130 Mbit/s  72            WPA1 WPA2
        74:05:A5:93:24:2B  D2-501         Infra  11    270 Mbit/s  65            WPA1 WPA2
        9C:D8:63:DA:4C:22  HF-LPT130      Infra  6     135 Mbit/s  49            --
        DC:FE:18:88:30:1B  THINK-Network  Infra  11    405 Mbit/s  37            WPA1 WPA2
        FC:7C:02:40:FD:B7  quer770503     Infra  3     270 Mbit/s  29            WPA1 WPA2
        C8:8F:26:19:DC:4F  Topway_19DC4F  Infra  1     130 Mbit/s  22            WPA2
root@ubuntu:~# sudo wifi_connect "[illegible]" "[illegible]"
Device 'wlan0' successfully activated with '4ea86192-91fa-4cd0-bdd7-ae08ff69c1d7'.
```

常用开发方法介绍-ssh

SSH：安全外壳协议（Secure Shell，简称SSH）是一种在不安全网络上用于安全远程登录和其他安全网络服务的协议。

通过网络的方式访问设备，与串口通信方式相比拥有更大的带宽和自由度

使用方式：

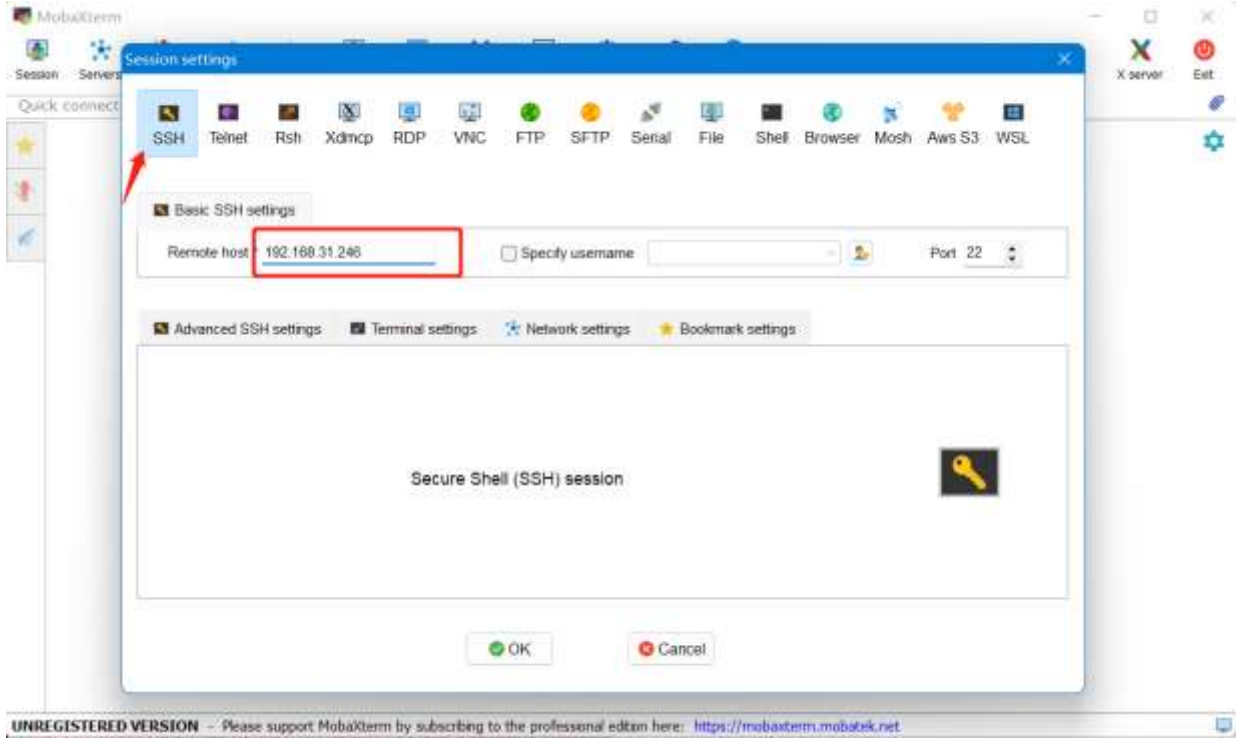
ssh username@IP

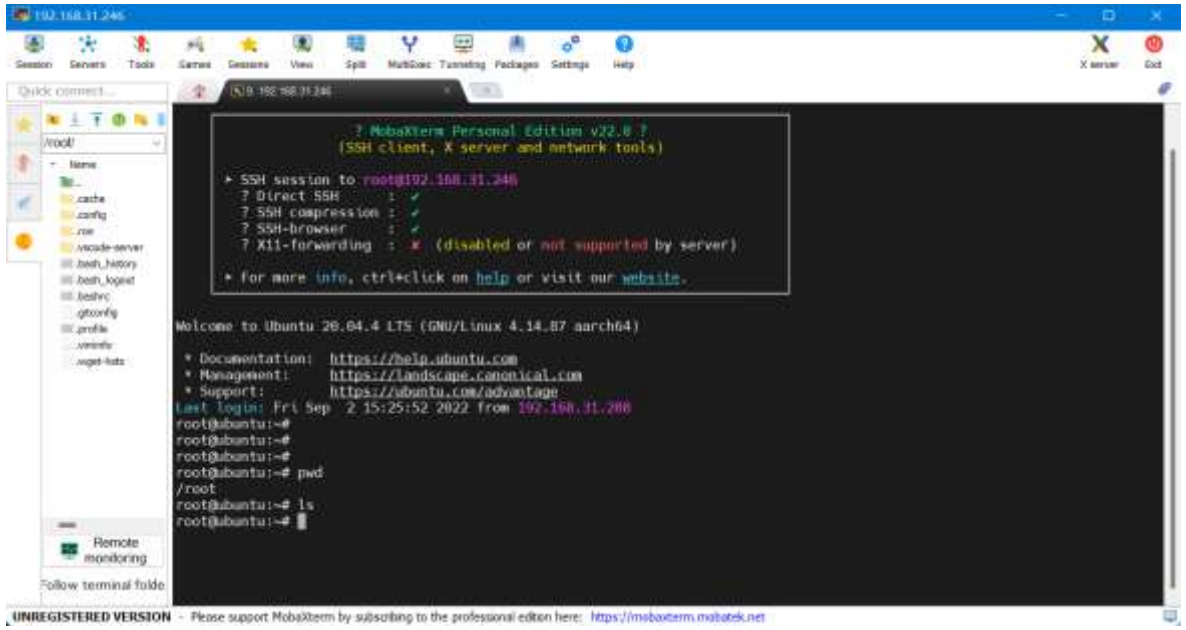
运行环境配置

通过SSH访问旭日X3派

1. 配置mobaxterm，填入X3派IP地址

2. 用户名与密码均输入root





ROS2 DOMAIN ID

需要通信的设备应确保使用相同的DOMAIN ID

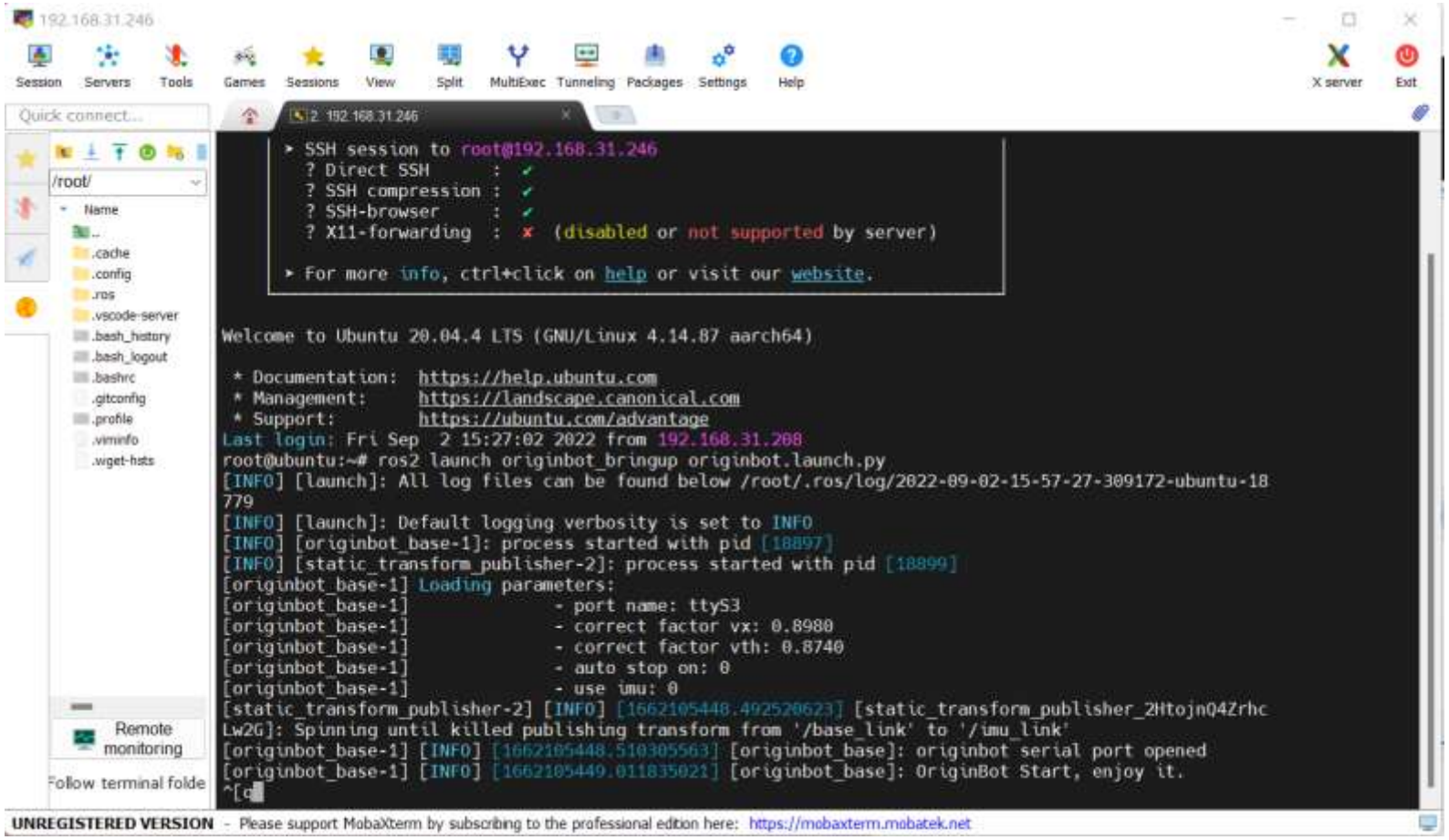
避免同一个局域网里机器人间相互干扰

export ROS_DOMAIN_ID=<your_domain_id> 0~101

键盘控制小车运动

1. SSH登录OriginBot

2. 执行ros2 launch originbot_bringup originbot.launch.py



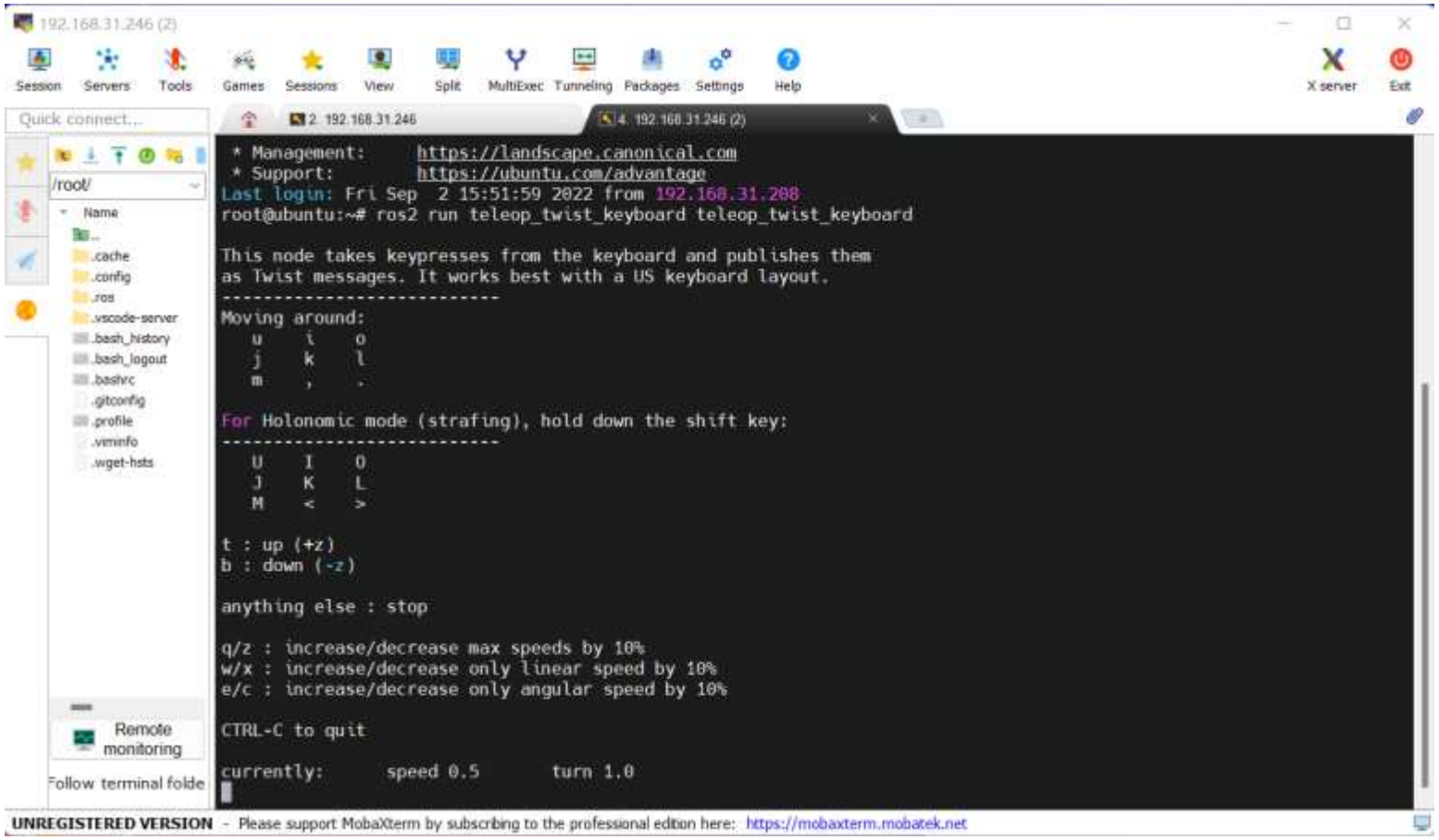
键盘控制小车运动

1. 启动另一个终端SSH连接OriginBot

2. 执行 ros2 run teleop_twist_keyboard teleop_twist_keyboard

3. 根据提示进行小车控制

注意：默认松开按键后，
机器人不会自动停止，
必须要点击“k”才能控制停车



键盘控制小车运动

PC上查看OriginBot发布的信息

sudo apt install python3-argcomplete #tab 自动补全命令

ros2 topic

ros2 topic list

ros2 topic echo /topic_name

安全关闭

1. SSH终端中输入 halt

2. 稍等5秒左右，就可以关闭机器人控制器上的电源开关，机器人关机完成。

注意：直接关闭电源开关也可以断电关机，但有可能损坏软件系统正在读写的文件，导致系统出现意外问题。

感谢观看

怕什么真理无穷，进一寸有一寸的欢喜

更多精彩，欢迎关注

古月居　古月学院

ROS
暑期学校

古月居　搜一下