

MicroROS-VM: First Trial

MicroROS-VM: First Trial

- 1、Burn MicroROS control board firmware
 - 1.1、Serial device identification
 - Device connection
 - Install serial device driver
 - View device
 - 1.2、Write factory firmware
 - Factory firmware
 - Flash download tool
 - 1.3、Burn factory firmware
 - Open the Flash download tool
 - Software download mode
 - Download factory firmware
 - Verify factory firmware
- 2、VMware installation and use
 - 2.1、VMware installation
 - VMware Workstation Player
 - VMware Workstation Pro
 - 2.2、VMware uses
- 3、Configure MicroROS control board
 - 3.1、Parameter introduction
 - 2、Configuration parameters
 3. Start/connect the agent
 - Start agent
 - connection broker
- 4、test
 - View ROS nodes
 - Keyboard control car

This tutorial is just a simplified version, mainly paired with quick-start videos!

For detailed tutorial steps, please see the [Preparation before development] tutorial!
Software locations involved in getting started quickly:
Factory firmware: program source code summary
virtual machine: virtual machine system

1、Burn MicroROS control board firmware

Note: For users who have just received the product, the MicroROS control board does not need to burn the factory firmware.

Please check directly below for the tutorial on configuring the MicroROS control board

Our products will have firmware burned before leaving the factory. This video is mainly provided to users who need to burn firmware!

1.1、Serial device identification

Device connection

Use Type-C data cable to connect the computer and MicroROS control board.

Note: Burning firmware requires turning on the power switch of the MicroROS control board for power supply!

When connecting to the MicroROS control board for the first time, the system may not be able to correctly identify the serial device. We need to install the driver of the CP210x series.

Install serial device driver

Open system device manager

If the device manager does not correctly identify the serial device, we need to install the CP210x series driver:

<https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers>

View device

Check the corresponding serial device in the device manager port option.

1.2、Write factory firmware

Factory firmware

The firmware we demonstrate here: microROS_Robot_V0.0.4

The firmware may be upgraded later and the name may be slightly different, but the steps are the same.

Flash download tool

We need to prepare the Flash download tool in advance:

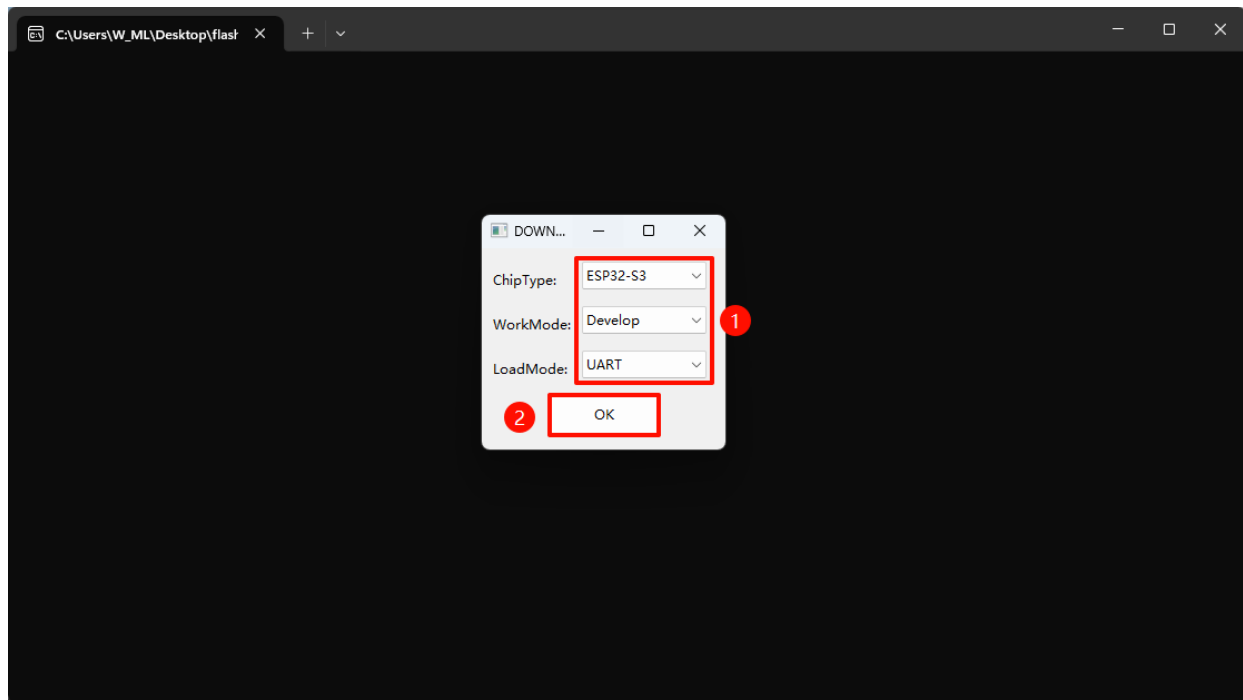
<https://www.espressif.com.cn/zh-hans/support/download/other-tools>

1.3、Burn factory firmware

Open the Flash download tool

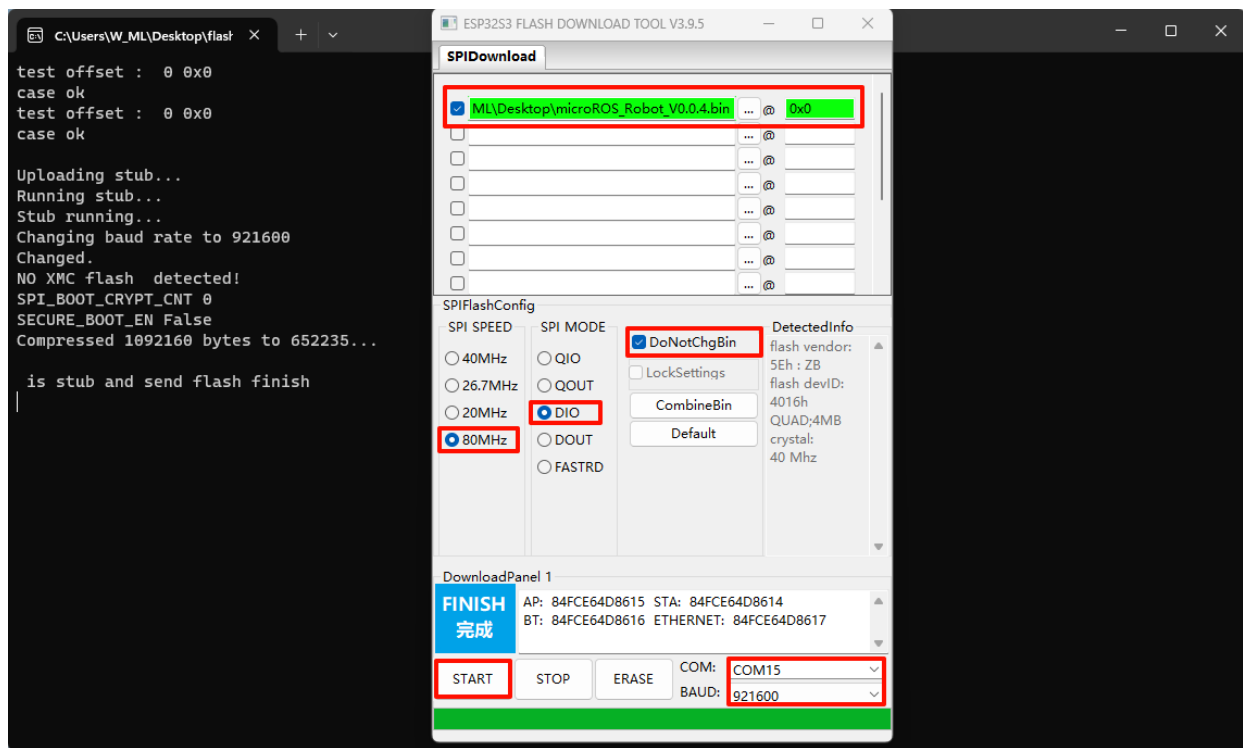
Software download mode

Check the corresponding download tool mode option



The terminal can see the corresponding download information!

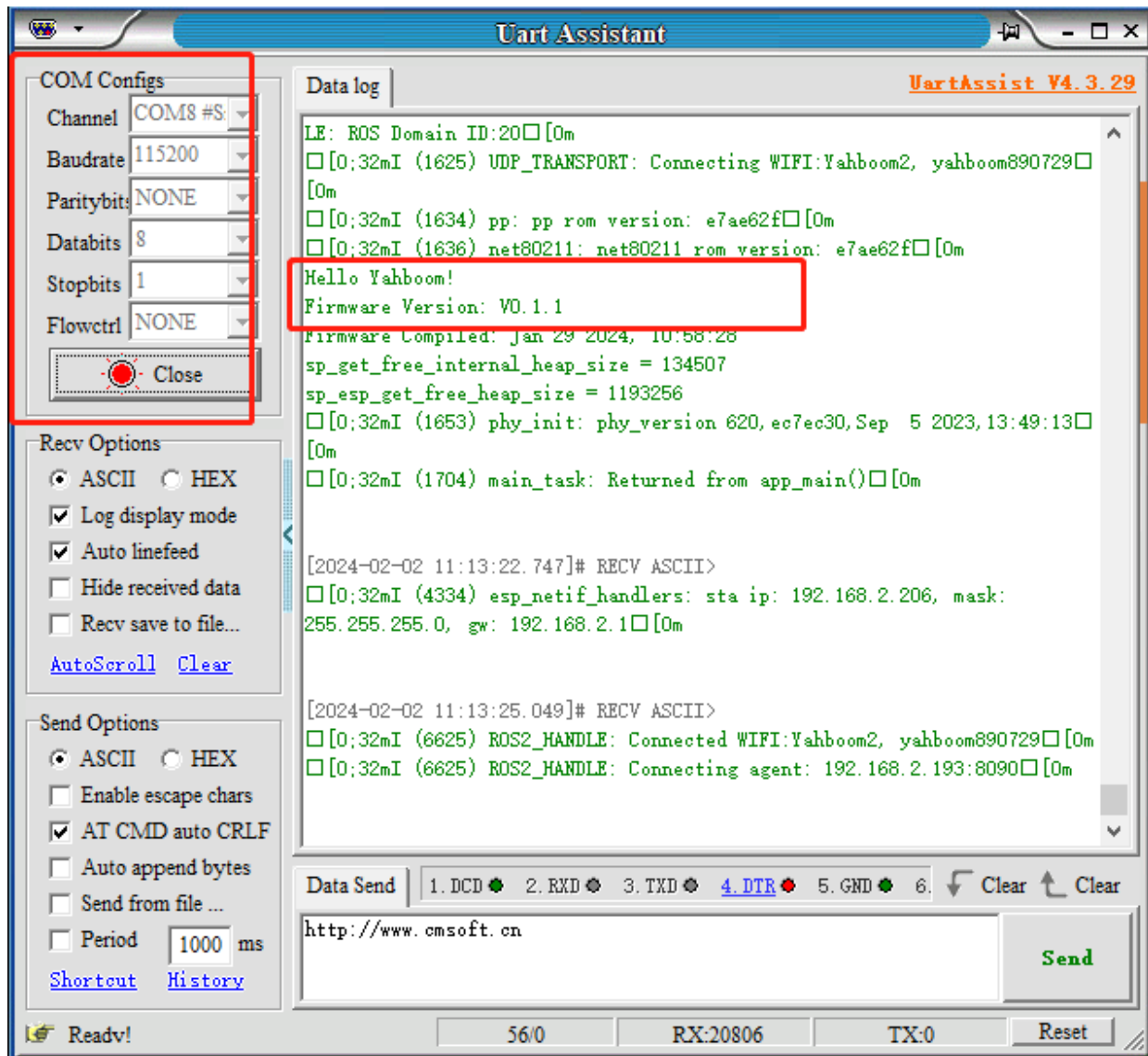
Download factory firmware



Verify factory firmware

Check the information printed by the MicroROS control board through the UartAssist software: After connecting the serial port, press the Reset button of the development board!

The serial port will print firmware version information!



2、VMware installation and use

2.1、VMware installation

VMware Workstation Player

This version is free for non-commercial use.

<https://www.vmware.com/cn/products/workstation-player.html>

VMware Workstation Pro

Commercial use requires payment, and you can experience the 30-day trial version.

<https://www.vmware.com/cn/products/workstation-pro/workstation-pro-evaluation.html>

2.2、VMware uses

You need to download and decompress the virtual machine image provided in our information in advance

3、Configure MicroROS control board

3.1、Parameter introduction

Edit the config_robot.py file according to your own configuration

```
if __name__ == '__main__':
    robot = MicroROS_Robot(port='/dev/ttyUSB0', debug=False)

    robot.set_wifi_config("Yahboom2", "yahboom890729")
    robot.set_udp_config([192, 168, 2, 238], 8090)
    robot.set_car_type(robot.CAR_TYPE_COMPUTER)
    # robot.set_car_type(robot.CAR_TYPE_RPI5)
    robot.set_ros_domain_id(20)
    # robot.set_ros_serial_baudrate(921600)
    robot.set_ros_namespace("")
```

- set_wifi_config: Configure the network information connected to the MicroROS control board
- set_udp_config: Virtual machine IP address
- set_car_type: Set the car type (currently there are virtual machine version and Raspberry Pi version)
- set_ros_domain_id: Set the environment variable of ROS2 communication domain ID
- set_ros_serial_baudrate: Set the baud rate for ROS serial communication
- set_ros_namespace: Set the name of the multi-machine communication car

```

MY_DOMAIN_ID: 20
yahboom@yahboom-VM:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
yahboom@yahboom-VM:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
    ether 02:42:9f:16:da:0b txqueuelen 0 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.2.238 netmask 255.255.255.0 broadcast 192.168.2.255
    inet6 fe80::3560:d93c:6a75:fc7c prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:24:c4:55 txqueuelen 1000 (Ethernet)
    RX packets 6217 bytes 8609346 (8.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1640 bytes 319493 (319.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 195 bytes 23918 (23.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 195 bytes 23918 (23.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

2、 Configuration parameters

Shortly press the reset button of the MicroROS control board (it is in the configuration state within 5 seconds of reset), and enter the following command to configure the robot. If the data returned by the terminal is consistent with what you set, it means that the configuration parameters are successful!

```
sudo python3 config_robot.py
```

After the configuration is successful, you can unplug the data cable connecting the virtual machine and the MicroROS control board!

```

yahboom@yahboom-VM:~$ sudo python3 config_robot.py
version: 0.0.4
ssid: Yahboom2
passwd: yahboom890729
ip_addr: 192.168.2.238
ip_port: 8090
car_type: CAR_TYPE_COMPUTER
domain_id: 20
ros_serial_baudrate: 921600
ros_namespace:
servo_offset: 0, 0
motor_pid_parm: 1.00, 0.20, 0.20
imu_yaw_pid_parm: 1.00, 0.00, 0.20
Please reboot the device to take effect, if you change some device config.

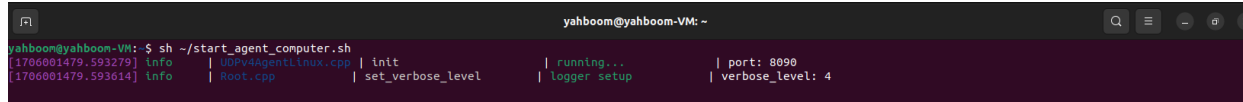
```

3. Start/connect the agent

Start agent

Using the virtual machine provided in our materials, run the following command:

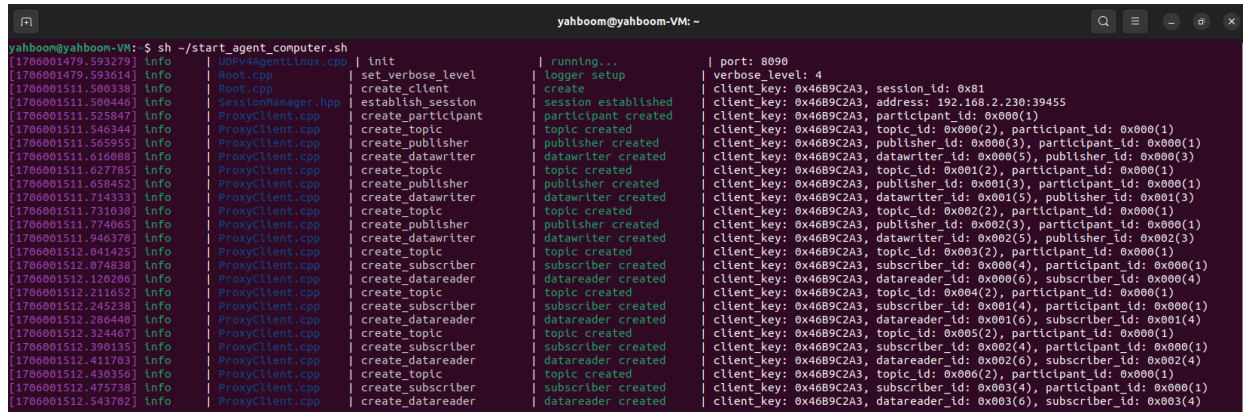
```
sh ~/start_agent_computer.sh
```



```
yahboom@yahboom-VM: ~  
yahboom@yahboom-VM: $ sh ~/start_agent_computer.sh  
[1706001479.593279] info | UDPv4AgentLinux.cpp | init | running... | port: 8090  
[1706001479.593614] info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4
```

connection broker

Turn on the power switch and the car will automatically connect to the agent.



```
yahboom@yahboom-VM: ~  
yahboom@yahboom-VM: $ sh ~/start_agent_computer.sh  
[1706001479.593279] info | UDPv4AgentLinux.cpp | init | running... | port: 8090  
[1706001479.593614] info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4  
[1706001511.500338] info | Root.cpp | create_client | create | client_key: 0x4689C2A3, session_id: 0x81  
[1706001511.500446] info | SessionManager.hpp | establish_session | session established | client_key: 0x4689C2A3, address: 192.168.2.230:39455  
[1706001511.502847] info | ProxyClient.cpp | create_participant | participant created | client_key: 0x4689C2A3, participant_id: 0x000(1)  
[1706001511.546344] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x000(2), participant_id: 0x000(1)  
[1706001511.565955] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x4689C2A3, publisher_id: 0x000(3), participant_id: 0x000(1)  
[1706001511.616088] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x4689C2A3, datawriter_id: 0x000(5), publisher_id: 0x000(3)  
[1706001511.627785] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x001(2), participant_id: 0x000(1)  
[1706001511.658452] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x4689C2A3, publisher_id: 0x001(3), participant_id: 0x000(1)  
[1706001511.714333] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x4689C2A3, datawriter_id: 0x001(5), publisher_id: 0x001(3)  
[1706001511.731030] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x002(2), participant_id: 0x000(1)  
[1706001511.774065] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x4689C2A3, publisher_id: 0x002(3), participant_id: 0x000(1)  
[1706001511.946370] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x4689C2A3, datawriter_id: 0x002(5), publisher_id: 0x002(3)  
[1706001512.041425] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x003(2), participant_id: 0x000(1)  
[1706001512.074838] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x4689C2A3, subscriber_id: 0x000(4), participant_id: 0x000(1)  
[1706001512.120206] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x4689C2A3, datareader_id: 0x000(6), subscriber_id: 0x000(4)  
[1706001512.211652] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x004(2), participant_id: 0x000(1)  
[1706001512.245338] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x4689C2A3, subscriber_id: 0x001(4), participant_id: 0x000(1)  
[1706001512.286440] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x4689C2A3, datareader_id: 0x001(6), subscriber_id: 0x001(4)  
[1706001512.324467] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x005(2), participant_id: 0x000(1)  
[1706001512.390135] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x4689C2A3, subscriber_id: 0x002(4), participant_id: 0x000(1)  
[1706001512.411703] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x4689C2A3, datareader_id: 0x002(6), subscriber_id: 0x002(4)  
[1706001512.430356] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x006(2), participant_id: 0x000(1)  
[1706001512.475738] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x4689C2A3, subscriber_id: 0x003(4), participant_id: 0x000(1)  
[1706001512.543702] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x4689C2A3, datareader_id: 0x003(6), subscriber_id: 0x003(4)
```

4. test

View ROS nodes

Reopen a terminal and enter the following command:

```
ros2 node list
```

Keyboard control car

Enter the following command in the terminal and keep the mouse on the terminal: Control the car according to the terminal key prompts!

```
ros2 run yahboomcar_ctr1 yahboom_keyboard
```

```
yahboom@yahboom-VM: ~  
yahboom@yahboom-VM:~$ sh ~/start_agent_computer.sh  
[1706001479.593279] Info | UDPv4AgentLinux.cpp | init | running... | port: 8090  
[1706001479.593614] Info | Root.cpp | set_verbose_level | logger setup | verbose_level: 4  
[1706001511.580338] Info | Root.cpp | create_client | create | client_key: 0x4689C2A3, session_id: 0x81  
[1706001511.580446] Info | SessionManager.hpp | establish_session | session established | client_key: 0x4689C2A3, address: 192.168.2.230:39455  
[1706001511.525847] Info | ProxyClient.cpp | create_participant | participant created | client_key: 0x4689C2A3, participant_id: 0x000(1)  
[1706001511.546344] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x000(2), participant_id: 0x000(1)  
[1706001511.565955] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x4689C2A3, publisher_id: 0x000(3), participant_id: 0x000(1)  
[1706001511.616088] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x4689C2A3, datawriter_id: 0x000(5), publisher_id: 0x000(3)  
[1706001511.627785] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x001(2), participant_id: 0x000(1)  
[1706001511.658452] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x4689C2A3, publisher_id: 0x001(3), participant_id: 0x000(1)  
[1706001511.714333] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x4689C2A3, datawriter_id: 0x001(5), publisher_id: 0x001(3)  
[1706001511.731030] Info | ProxyClient.cpp | create_topic | topic created | client_key: 0x4689C2A3, topic_id: 0x002(2), participant_id: 0x000(1)  
[1706001511.774065] Info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x4689C2A3, publisher_id: 0x002(3), participant_id: 0x000(1)  
[1706001512.074038] Info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x4689C2A3, datawriter_id: 0x002(5), publisher_id: 0x002(3)  
[1706001512.120206] Info | ProxyClient.cpp | MY_DOMAIN_ID: 20 | _id: 0x003(2), participant_id: 0x000(1)  
[1706001512.211652] Info | ProxyClient.cpp | yahboom@yahboom-VM:~$ ros2 node list | riber_id: 0x000(4), participant_id: 0x000(1)  
[1706001512.245238] Info | ProxyClient.cpp | /0 Car_Node | eader_id: 0x000(6), subscriber_id: 0x000(4)  
[1706001512.286440] Info | ProxyClient.cpp | yahboom@yahboom-VM:~$ ros2 run yahboomcar_ctrl yahboom_keyboard | _id: 0x004(2), participant_id: 0x000(1)  
[1706001512.324467] Info | ProxyClient.cpp | Control Your SLAM-Bot! | riber_id: 0x001(4), participant_id: 0x000(1)  
[1706001512.390135] Info | ProxyClient.cpp | ----- | _id: 0x005(2), participant_id: 0x000(1)  
[1706001512.411703] Info | ProxyClient.cpp | Moving around: | riber_id: 0x002(4), participant_id: 0x000(1)  
[1706001512.430356] Info | ProxyClient.cpp | u l o | eader_id: 0x002(6), subscriber_id: 0x002(4)  
[1706001512.475738] Info | ProxyClient.cpp | j k l | _id: 0x006(2), participant_id: 0x000(1)  
[1706001512.543702] Info | ProxyClient.cpp | , . | riber_id: 0x003(4), participant_id: 0x000(1)  
[1706001512.543702] Info | ProxyClient.cpp | | eader_id: 0x003(6), subscriber_id: 0x003(4)  
  
q/z : increase/decrease max speeds by 10%  
w/x : increase/decrease only linear speed by 10%  
e/c : increase/decrease only angular speed by 10%  
t/T : x and y speed switch  
s/S : stop keyboard control  
space key, k : force stop  
anything else : stop smoothly  
  
CTRL-C to quit  
  
currently: speed 0.2 turn 1.0
```