

MicroROS Balance Car: Quick Start

MicroROS Balance Car: Quick Start

1. Burn ESP32 communication board firmware
2. Burn STM32 control board firmware
3. APP Bluetooth remote control
 1. BalanceBot
 2. Bluetooth remote control
 - Bluetooth connection
 - Direction control
4. VMware installation and use
 1. VMware installation
 - VMware Workstation Pro
 2. VMware usage
5. Configure ESP32 communication board
 1. Parameter introduction
 2. Configuration parameters
 3. Start/connect the agent
 - Start the agent
 - Connect to the agent
 4. Test
 - View ROS nodes
 - Keyboard control of the car
- VI. View the camera screen (optional)
 1. Hardware connection
 2. Connection agent
 3. Test
 - View ROS nodes
 - View the camera screen
 - Image correction

This tutorial is just a simplified version, mainly with a quick start video!

For detailed tutorial steps, please see the [Pre-development Preparation] tutorial!

Software location involved in the quick start:

Factory firmware: Source code summary

Virtual machine: Virtual machine system

1. Burn ESP32 communication board firmware

Note: For users who have just received the product, the ESP32 communication board does not need to be burned with factory firmware

Our products will be burned with firmware before leaving the factory. If the firmware has not been modified, there is no need to burn it. If the firmware has been modified, you can refer to the next tutorial to burn the factory firmware.

2. Burn STM32 control board firmware

Note: For users who have just received the product, the ESP32 communication board does not need to be burned with factory firmware

Our products will be burned with firmware before leaving the factory. If the firmware has not been modified, there is no need to burn it. If the firmware has been modified, you can refer to the next tutorial to burn the factory firmware.

3. APP Bluetooth remote control

1. BalanceBot

Use the mobile browser to scan the QR code to download the BalanceBot APP (only supports Android)



2. Bluetooth remote control

Before Bluetooth remote control, we need to install the BalanceBot APP.

Bluetooth connection

When you open the Bluetooth remote control APP for the first time, it will automatically search and connect to nearby Bluetooth devices. We need to bring the phone close to the car, and the Bluetooth remote control APP will automatically connect to the Esp32 expansion board; if the connection is successful, a "Connection successful" prompt will appear.

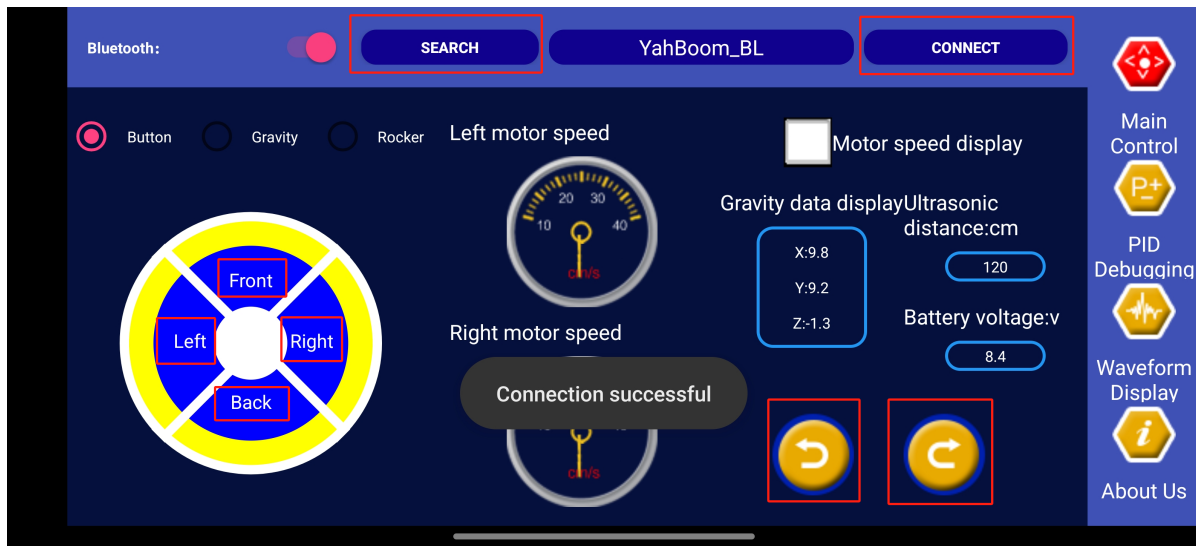
If no search is found, you can click the Bluetooth search and connection options in turn.

Note: The mobile phone needs to turn on Bluetooth and positioning functions and allow the Bluetooth remote control APP related permissions.

Direction control

The main control interface can control the movement of the balance car through buttons, gravity and joysticks.

(Only the selected functions are available, and other functions of the interface require additional installation of the Bluetooth module)



Bluetooth module: If the Bluetooth module and Esp32 Bluetooth exist at the same time, if you want to connect the Bluetooth module, you can turn off the power of ESP32, wait for the app to connect, and then power the Esp32 board.

If you want to connect to the Bluetooth of ESP32, you need to unplug the Bluetooth module, otherwise you can only connect to the Bluetooth of ESP32 with probability

4. VMware installation and use

1. VMware installation

VMware Workstation Pro

Customers need to use the following link to register an account and download the virtual machine (refer to the manual for detailed steps), or directly use the virtual machine compressed package in the attachment to decompress and use.

For non-commercial use only

<https://support.broadcom.com/group/ecx/productdownloads?subfamily=VMware%20workstation%20Pro>

2. VMware usage

You need to download and decompress the virtual machine image provided in our materials in advance, open it and enter the virtual machine.

5. Configure ESP32 communication board

1. Parameter introduction

Edit the config_Balance_Car.py file according to your own configuration

```

if __name__ == '__main__':
    robot = MicroROS_Robot(port='/dev/ttyUSB0', debug=False)
    print("Rebooting Device, Please wait.")
    robot.reboot_device()

    robot.set_wifi_config("Yahboom2", "yahboom890729")
    robot.set_udp_config([192, 168, 2, 108], 8899)
    robot.set_car_type(robot.CAR_TYPE_COMPUTER)
    #robot.set_car_type(robot.CAR_TYPE_UASRT_CAR)

    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 ESP32 communication board
- set_udp_config: Virtual machine IP address (displayed at the top of the virtual machine terminal or viewed using ifconfig)
- set_car_type: Set the car type (currently only the virtual machine version)
- set_ros_serial_baudrate: Set the baud rate of ROS serial communication
- set_ros_namespace: Set the name of the multi-machine communication car

2. Configuration parameters

Put the microROS balance car in a balanced position and turn it on. After balancing, connect the Type-C data cable to the ESP32 communication board on the upper layer of the car and choose to connect to the virtual machine (the CP210X driver must be installed on the computer in advance).

Short press the reset button of the ESP32 communication board (reset within 5 seconds is in the configuration state), enter the following command to configure the robot, if the data returned by the terminal is consistent with your settings, it means that the configuration parameters are successful!

```

sudo python3 ~/config_Balance_Car.py

```

After successful configuration, you can unplug the data cable connecting the virtual machine and the ESP32 communication board, turn the car off and on, or press the reset button of the ESP32 expansion board!

```

-----
MY_DOMAIN_ID: 20
MY_IP: 192.168.2.108
-----

yahboom@yahboom-VM:~$ python3 config_Balance_Car.py
Rebooting Device, Please wait.
version: 2.8.0
ssid: Yahboom2
passwd: yahboom890729
ip_addr: 192.168.2.108
ip_port: 8899
car_type: CAR_TYPE_COMPUTER
domain_id: 20
ros_serial_baudrate: 921600
ros_namespace:
Please reboot the device to take effect, if you change some device config.

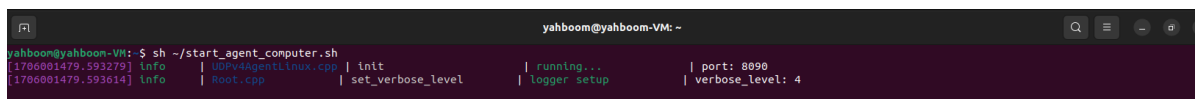
```

3. Start/connect the agent

Start the agent

Use the virtual machine provided in our materials and run the following command:

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



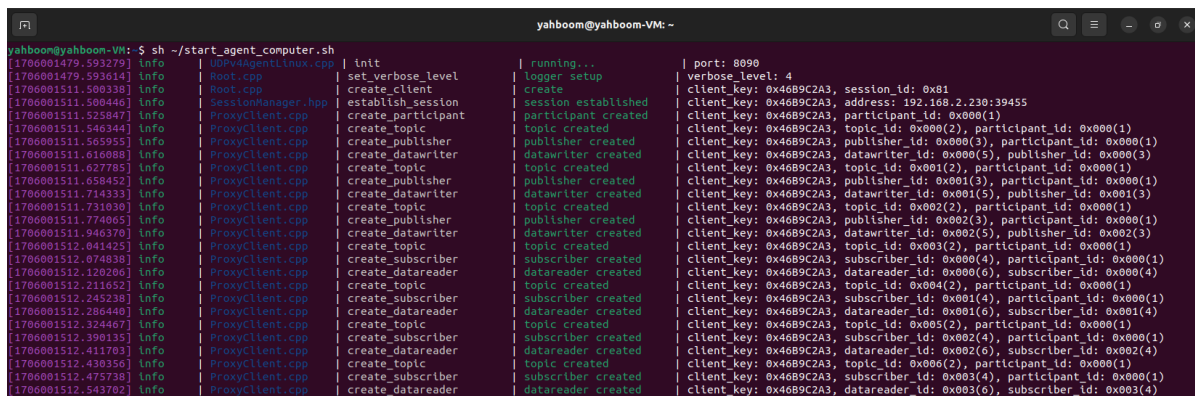
```

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

```

Connect to the agent

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



```

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: 0x46B9C2A3, session_id: 0x81
[1706001511.500446] info | SessionManager.hpp | establish_session | session established | client_key: 0x46B9C2A3, address: 192.168.2.230:39455
[1706001511.525847] info | ProxyClient.cpp | create_participant | participant created | client_key: 0x46B9C2A3, participant_id: 0x000(1)
[1706001511.546344] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x000(2), participant_id: 0x000(1)
[1706001511.565955] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x46B9C2A3, publisher_id: 0x000(3), participant_id: 0x000(1)
[1706001511.616088] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x46B9C2A3, datawriter_id: 0x000(5), publisher_id: 0x000(3)
[1706001511.627785] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x001(2), participant_id: 0x000(1)
[1706001511.658452] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x46B9C2A3, publisher_id: 0x001(3), participant_id: 0x000(1)
[1706001511.714333] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x46B9C2A3, datawriter_id: 0x001(5), publisher_id: 0x001(3)
[1706001511.731030] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x002(2), participant_id: 0x000(1)
[1706001511.774065] info | ProxyClient.cpp | create_publisher | publisher created | client_key: 0x46B9C2A3, publisher_id: 0x002(3), participant_id: 0x000(1)
[1706001511.946370] info | ProxyClient.cpp | create_datawriter | datawriter created | client_key: 0x46B9C2A3, datawriter_id: 0x002(5), publisher_id: 0x002(3)
[1706001512.041425] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x003(2), participant_id: 0x000(1)
[1706001512.074838] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x46B9C2A3, subscriber_id: 0x000(4), participant_id: 0x000(1)
[1706001512.102060] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x46B9C2A3, datareader_id: 0x000(6), subscriber_id: 0x000(4)
[1706001512.211652] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x004(2), participant_id: 0x000(1)
[1706001512.245238] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x46B9C2A3, subscriber_id: 0x001(4), participant_id: 0x000(1)
[1706001512.286440] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x46B9C2A3, datareader_id: 0x001(6), subscriber_id: 0x001(4)
[1706001512.324467] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x005(2), participant_id: 0x000(1)
[1706001512.390135] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x46B9C2A3, subscriber_id: 0x002(4), participant_id: 0x000(1)
[1706001512.411703] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x46B9C2A3, datareader_id: 0x002(6), subscriber_id: 0x002(4)
[1706001512.430356] info | ProxyClient.cpp | create_topic | topic created | client_key: 0x46B9C2A3, topic_id: 0x006(2), participant_id: 0x000(1)
[1706001512.475738] info | ProxyClient.cpp | create_subscriber | subscriber created | client_key: 0x46B9C2A3, subscriber_id: 0x003(4), participant_id: 0x000(1)
[1706001512.543702] info | ProxyClient.cpp | create_datareader | datareader created | client_key: 0x46B9C2A3, datareader_id: 0x003(6), subscriber_id: 0x003(4)

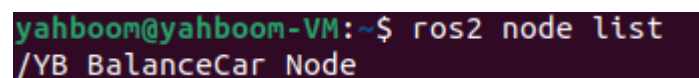
```

4. Test

View ROS nodes

Reopen a terminal and enter the following command:

```
ros2 node list
```



```

yahboom@yahboom-VM:~$ ros2 node list
/YB_BalanceCar_Node

```

Keyboard control of the car

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

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```

The keyboard keys are as follows,

Direction control

【i】 or 【I】	【linear, 0】	【u】 or 【U】	【linear, angular】
【,】	【-linear, 0】	【o】 or 【O】	【linear, - angular】
【j】 or 【J】	【0, angular】	【m】 or 【M】	【- linear, - angular】
【l】 or 【L】	【0, - angular】	【.】	【- linear, angular】

Speed control

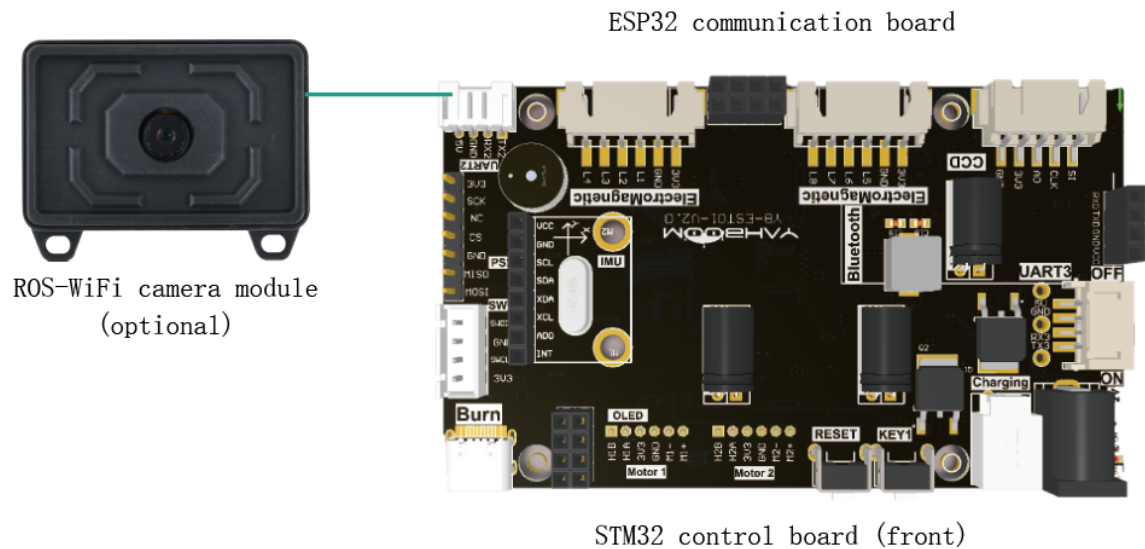
Key	Speed change	Key	Speed change
【q】	Increase both linear and angular speed by 10%	【z】	Linear and angular speeds both decrease by 10%
【w】	Linear speed only increases by 10%	【x】	Linear speed only decreases by 10%
【e】	Angular speed only increases by 10%	【c】	Angular speed only decreases by 10%
【t】	Linear speed X-axis/Y-axis direction switch	【s】	Stop keyboard control

Note: Since the car is a two-wheeled balancing structure and cannot move sideways, the **【t】** key is meaningless.

VI. View the camera screen (optional)

1. Hardware connection

The image transmission module should be connected to the STM32 control board, as shown in the figure.



2. Connection agent

After all the hardware is connected, while configuring the ESP32 communication board parameters, the wifi and ip agents connected to the ROS-WIFI image transmission module will be directly configured.

Use the virtual machine provided in our materials and run the following command:

```
sh ~/start_Camera_computer.sh
```

```
yahboom@yahboom-VM:~$ sh start_Camera_computer.sh
[1715156105.293050] info | UDPv4AgentLinux.cpp | init |
running... | port: 9999 |
[1715156105.293721] info | Root.cpp | set_verbose_level | l
ogger setup | verbose_level: 4 |
[1715156106.106118] info | Root.cpp | create_client | c
reate | client_key: 0x2CC68F30, session_id: 0x81 |
[1715156106.106177] info | SessionManager.hpp | establish_session | s
ession established | client_key: 0x2CC68F30, address: 192.168.2.93:5608 |
[1715156106.126480] info | ProxyClient.cpp | create_participant | p
articipant created | client_key: 0x2CC68F30, participant_id: 0x000(1) |
[1715156106.133785] info | ProxyClient.cpp | create_topic | t
opic created | client_key: 0x2CC68F30, topic_id: 0x000(2), participant_
id: 0x000(1) |
[1715156106.151365] info | ProxyClient.cpp | create_publisher | p
ublisher created | client_key: 0x2CC68F30, publisher_id: 0x000(3), particip
ant_id: 0x000(1) |
[1715156106.173468] info | ProxyClient.cpp | create_datawriter | d
atawriter created | client_key: 0x2CC68F30, datawriter_id: 0x000(5), publish
er_id: 0x000(3) |
```

3. Test

View ROS nodes

Reopen a terminal and enter the following command:

```
ros2 node list
```

```
yahboom@yahboom-VM:~$ ros2 node list
/espRos/Esp32Node
```

View the camera screen

Open the terminal of the factory virtual machine system and enter

```
ros2 run yahboom_esp32_camera sub_img
```



Image correction

If the camera is upside down, open a new terminal and enter

```
python3 ~/SET_Camera.py
```

Then enter the IP address connected to the ROS-wifi image transmission module, and you can check it in the terminal connected to the ROS-wifi image transmission module agent


```

yahboom@yahboom-VM:~$ sh start_Camera_computer.sh
1715156105.293050] info      | UDPv4AgentLinux.cpp | init
unning...
1715156105.293721] info      | Root.cpp             | set_verbose_level
gger setup        | verbose_level: 4
1715156106.106118] info      | Root.cpp             | create_client
eate              | client_key: 0x2CC68F30, session_id: 0x81
1715156106.106177] info      | SessionManager.hpp   | establish_session
ssion established | client_key: 0x2CC68F30, address: 192.168.2.93 5608
1715156106.126480] info      | ProxyClient.cpp      | create_participant
rticipant created | client_key: 0x2CC68F30, participant_id: 0x000(1)
1715156106.133785] info      | ProxyClient.cpp      | create_topic
pic created       | client_key: 0x2CC68F30, topic_id: 0x000(2), participant_
d: 0x000(1)
1715156106.151365] info      | ProxyClient.cpp      | create_publisher
blisher created   | client_key: 0x2CC68F30, publisher_id: 0x000(3), particip
nt_id: 0x000(1)
1715156106.173468] info      | ProxyClient.cpp      | create_datawriter
tawriter created  | client_key: 0x2CC68F30, datawriter_id: 0x000(5), publish
r id: 0x000(3)

```

Enter the IP in the terminal according to the IP found, and then press Enter

```

yahboom@yahboom-VM:~$ python3 SET_Camera.py
please input docket ipv4:
192.168.2.93
Camera is set ok!
yahboom@yahboom-VM:~$

```

The picture is restored

