

tag identification

Quick use

1. Power on DOGZILLA

First, we turn on the switching power supply of the mechanical dog and start the mechanical dog



After starting, we can view the IP address on the small screen of the robot dog.

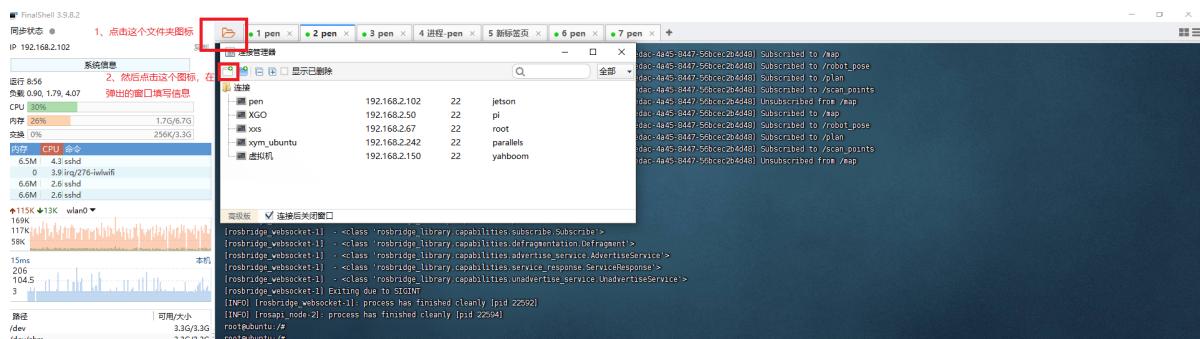
2. Start DOGZILLA chassis

PI4 version steps:

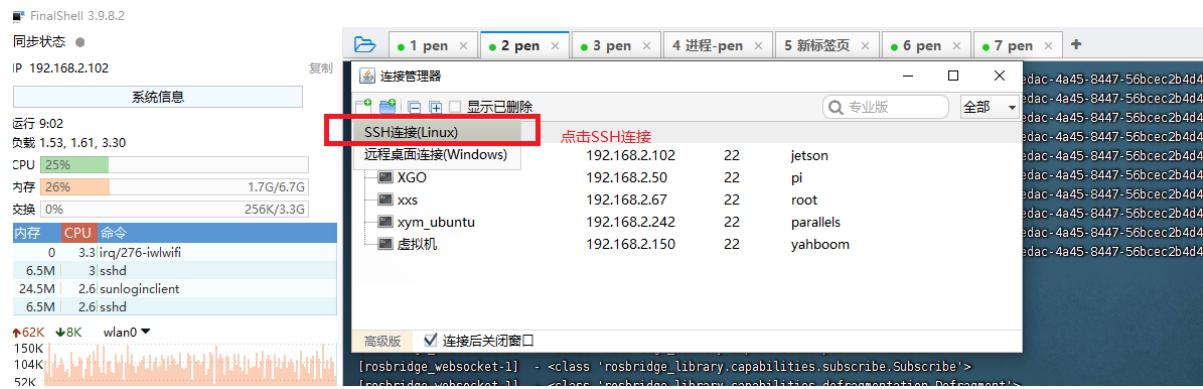
Then use the ssh terminal to connect to the robot dog.

Note: The IP address used when writing this tutorial: 192.168.2.102 User name: pi Password: yahboom The actual IP address shall prevail when used.

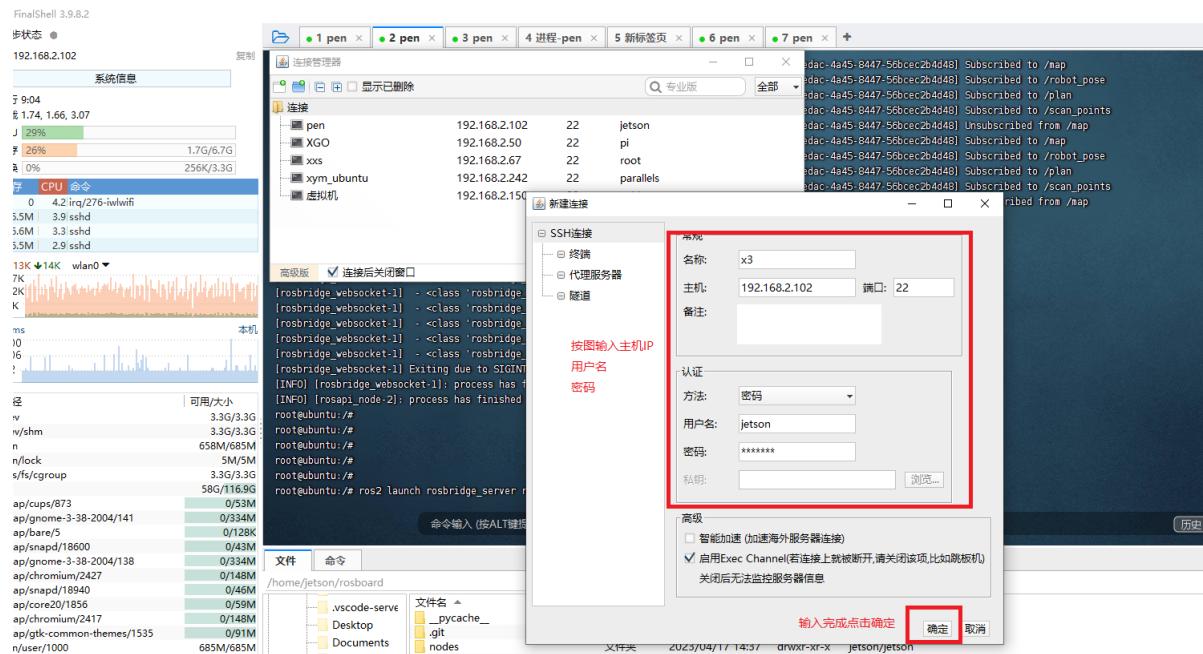
Open the shell tool. The shell tool I use here is FinalShell. Enter username, password, port, connection name and other information.



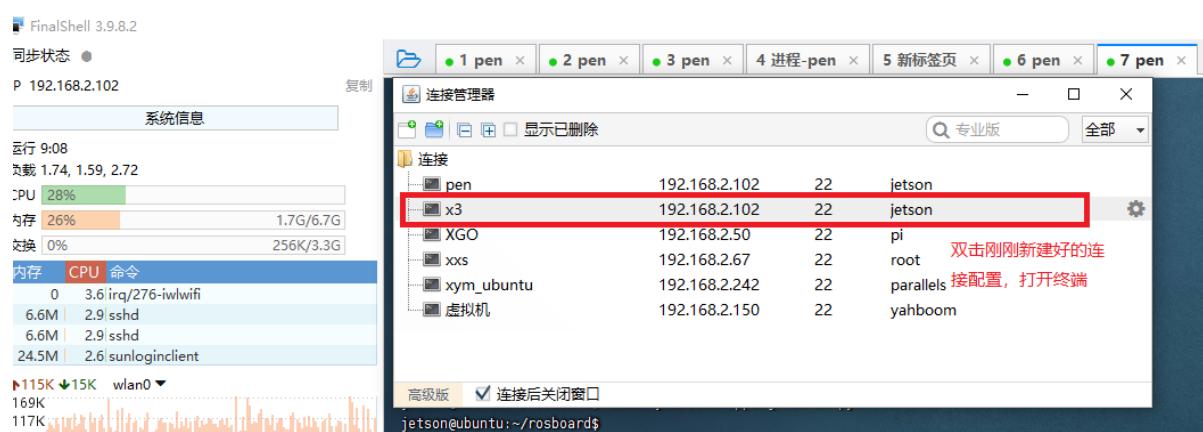
Select ssh connection to create a new ssh connection



Here the username is pi, the password is yahboom, and the ip address is the IP address of the real robot dog.



Select the ssh connection you just created here.



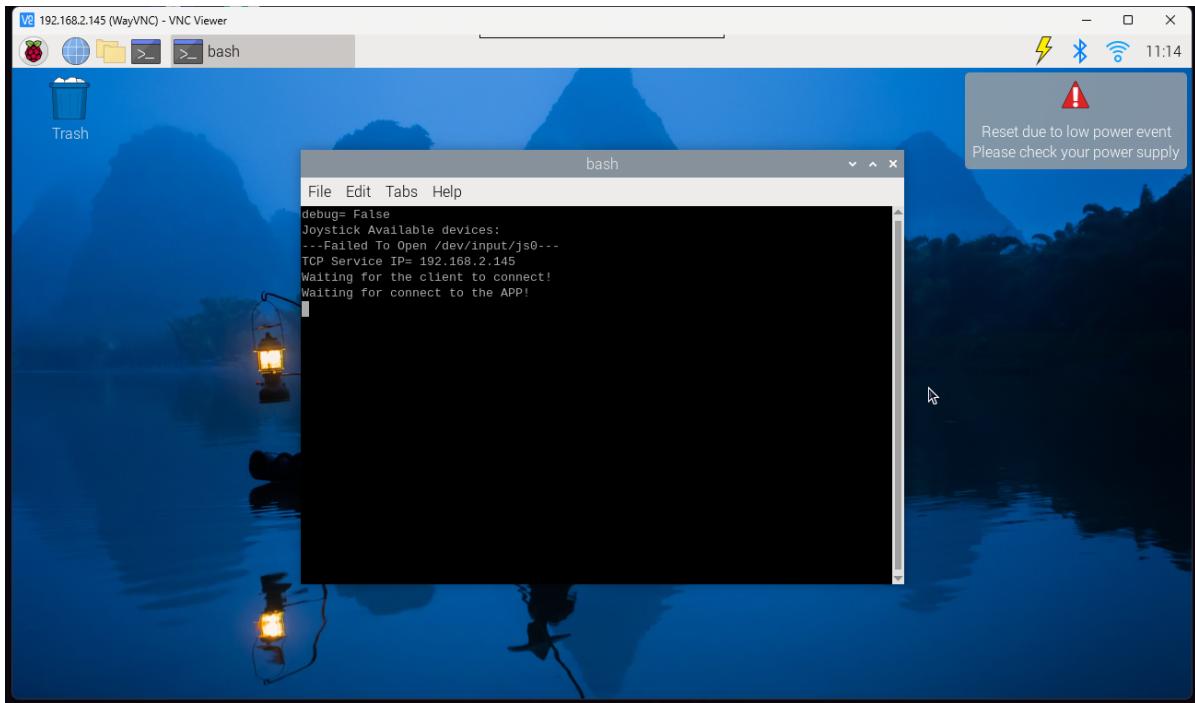
Enter the command in the terminal to start the chassis task.

```
sudo systemctl restart YahboomStart.service
```

```
pi@yahboom:~$ sudo systemctl restart YahboomStart.service
```

Pi5 version steps:

After the mechanical dog is started, use the vnc software to remotely connect to the mechanical dog through the IP address on the OLED (For specific steps, please see "Remote Login Operation").



Then `ctrl+c` closes the large program and enter the following command to enter docker:

```
./run_humble.sh
```

```
TCP Service IP= 192.168.2.145
waiting for the client to connect!
Waiting for connect to the APP!
^CKeyboardInterrupt
2024-04-28T10:17:27Z
-----program end-----
pi@raspberrypi:~ $ ./run_humble.sh
access control disabled, clients can connect from any host
root@raspberrypi:/#
```

Then enter the following commands in the docker terminal to start the car radar, imu, and mechanical dog joint status nodes.

```
ros2 launch bringup Navigation_bringup.launch.py
```

```
root@raspberrypi: /  
File Edit Tabs Help  
at 0x7fff363522f0>  
[yahboomcar_joint_state-3] [13.16, 45.61, 1.34, 10.1, 44.36, -1.09, 10.1, 51.85,  
2.55, 6.53, 51.22, -0.36]  
[yahboomcar_joint_state-3] &&&&&&&&&&&&&&&&&& 0.10927200317382812  
[yahboomcar_joint_state-3] #####  
[yahboomcar_joint_state-3] [-0.17585449218750002, -0.13996582031250002, -9.72702  
63671875, -1.0365853658536586, -0.426829268292683, -0.6097560975609757, 0.010487  
360583411322, -0.02726797640323639, 5.983139933268229]  
[yahboomcar_joint_state-3] ***** <rclpy.timer.Timer object  
at 0x7fff363522f0>  
[yahboomcar_joint_state-3] [13.16, 45.61, 1.34, 10.1, 44.36, -1.09, 10.1, 51.85,  
2.55, 6.53, 51.22, -0.36]  
[yahboomcar_joint_state-3] &&&&&&&&&&&&&&&&& 0.10969948768615723  
[yahboomcar_joint_state-3] #####  
[yahboomcar_joint_state-3] [-0.14475097656250002, -0.131591796875, -9.7401855468  
75, -1.0975609756097562, -0.3658536585365854, -0.6097560975609757, 0.01022947788  
9007993, -0.02749979310565525, 5.983139933268229]  
[yahboomcar_joint_state-3] ***** <rclpy.timer.Timer object  
at 0x7fff363522f0>  
[yahboomcar_joint_state-3] [13.16, 45.61, 1.34, 10.1, 44.36, -1.09, 10.1, 51.85,  
2.55, 6.53, 51.22, -0.36]  
[yahboomcar_joint_state-3] &&&&&&&&&&&&&&&&& 0.10920882225036621  
[yahboomcar_joint_state-3] #####
```

3. Start the image publishing node

PI4 version steps:

First enter the following command in the terminal

```
export PATH=/home/pi/opencv_install/bin:$PATH  
export LD_LIBRARY_PATH=/home/pi/opencv_install/lib:$LD_LIBRARY_PATH
```

Then enter the command in the terminal

```
cd cartographer_ws2/
```

```
source install/setup.bash
```

```
pi@yahboom:~$ cd cartographer_ws2/  
pi@yahboom:~/cartographer_ws2$  
pi@yahboom:~/cartographer_ws2$  
pi@yahboom:~/cartographer_ws2$ source install/setup.bash  
pi@yahboom:~/cartographer_ws2$
```

Then enter the following command

```
ros2 run yahboom_qrcode yahboom_qrcode_node
```

```

pi@yahboom:~/cartographer_ws2$ ros2 run xgo_qrcode xgo_qrcode_node
[ WARN:0@0.334] global cap_gstremer.cpp:2784 handleMessage OpenCV | GStreamer warning: Embedded video playback halted; module source reported: Could not read from resource.
[ WARN:0@0.335] global cap_gstremer.cpp:1679 open OpenCV | GStreamer warning: unable to start pipeline
[ WARN:0@0.336] global cap_gstremer.cpp:1164 isPipelinePlaying OpenCV | GStreamer warning: GStreamer: pipeline have not been created
A demo program of Wechat QRCode Detector:

-----
11111.here.
-----
init.fov.0.997839
init111.fov. 0.707185

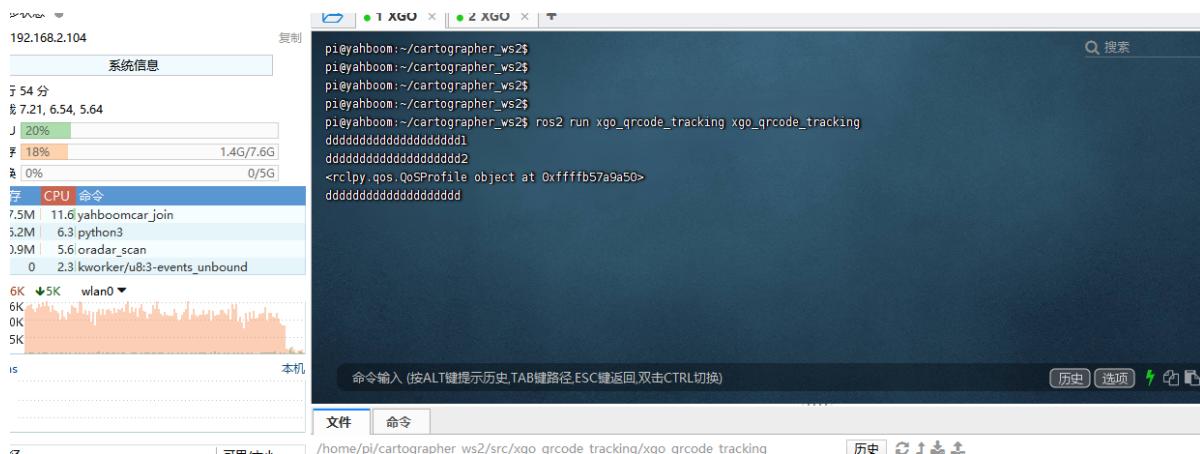
```

Reopen a terminal and enter the command:

```

cd cartographer_ws2/
source install/setup.bash
ros2 run yahboom_qrcode_tracking yahboom_qrcode_tracking

```



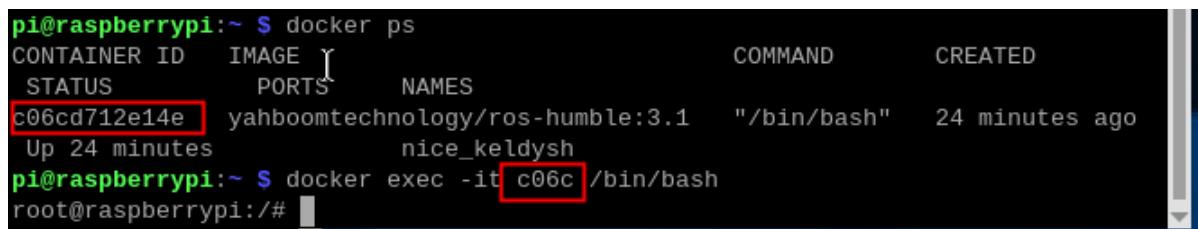
PI5 version steps:

Enter in the root directory of the Raspberry Pi and enter the same terminal

```

docker ps
docker exec -it id /bin/bash

```



Then enter the following command

```

ros2 run yahboom_qrcode yahboom_qrcode_node

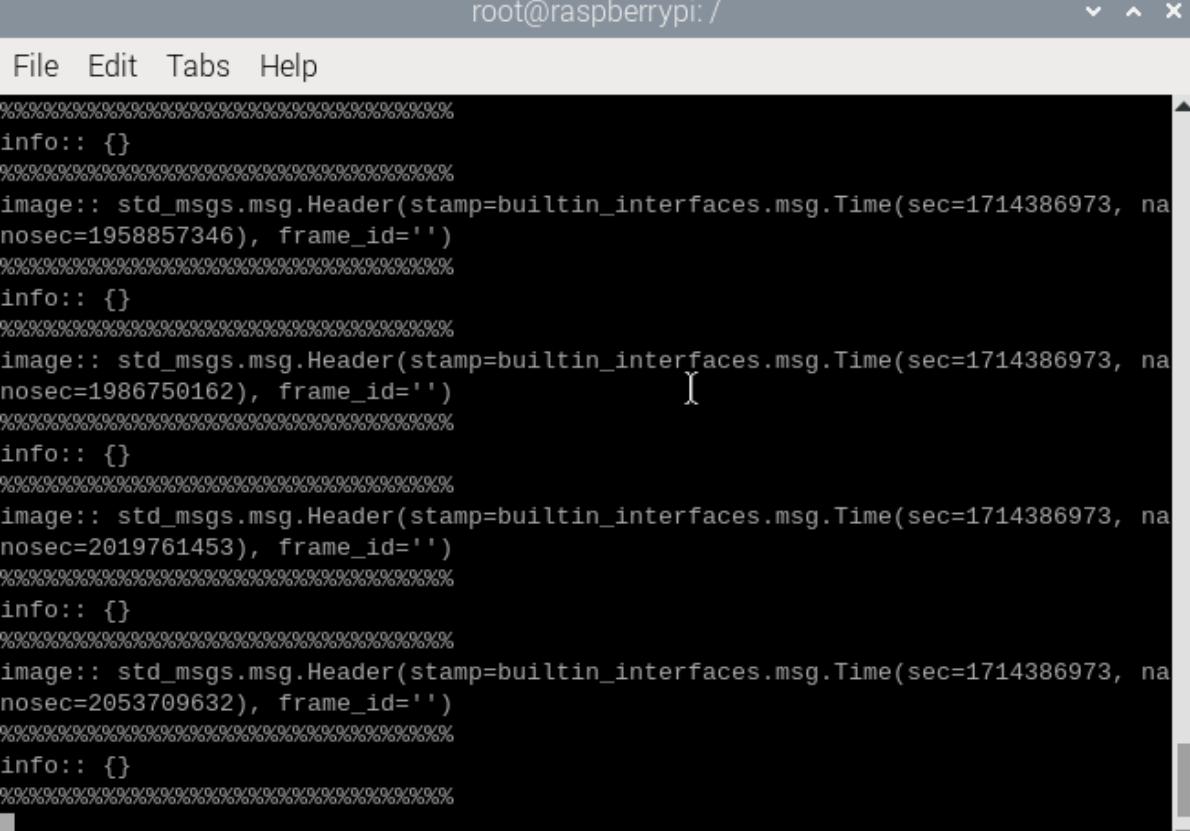
```

```
root@raspberrypi:/# ros2 run yahboom_qrcode yahboom_qrcode_node
[ WARN:0] global ./modules/videoio/src/cap_gststreamer.cpp (2075) handleMessage OpenCV | GStreamer warning: Embedded video playback halted; module source reported : Could not read from resource.
[ WARN:0] global ./modules/videoio/src/cap_gststreamer.cpp (1053) open OpenCV | GStreamer warning: unable to start pipeline
[ WARN:0] global ./modules/videoio/src/cap_gststreamer.cpp (616) isPipelinePlaying OpenCV | GStreamer warning: GStreamer: pipeline have not been created
A demo program of WeChat QRCode Detector:

-----
11111,here.
-----
init,fov.0.997839
init111,fov. 0.707185
```

Reopen the same docker terminal and enter the command:

```
ros2 run yahboom_qrcode_tracking yahboom_qrcode_tracking
```



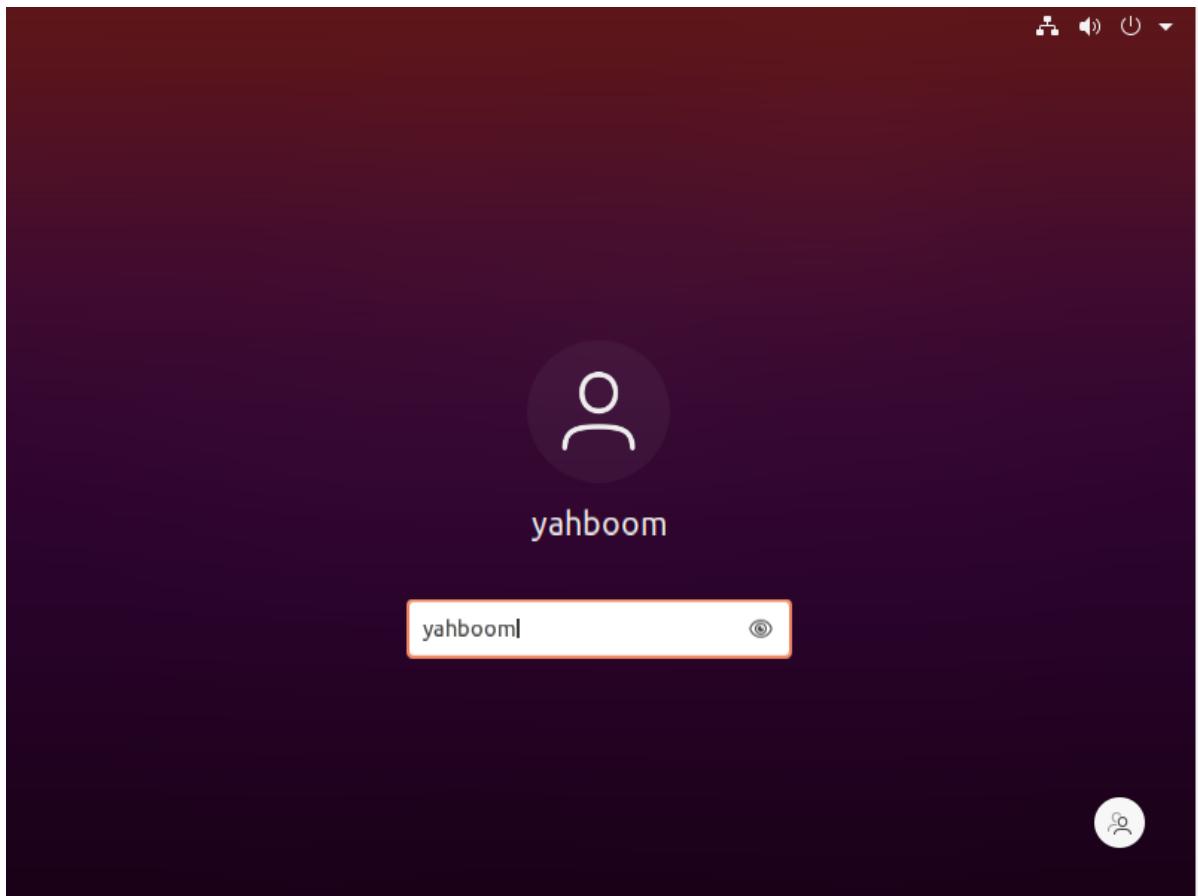
The screenshot shows a terminal window with the title "root@raspberrypi: /". The window contains a menu bar with "File", "Edit", "Tabs", and "Help". The main area of the terminal displays the following ROS 2 log output:

```
info:: {}
image:: std_msgs.msg.Header(stamp= builtin_interfaces.msg.Time(sec=1714386973, nanosec=1958857346), frame_id='')
info:: {}
image:: std_msgs.msg.Header(stamp= builtin_interfaces.msg.Time(sec=1714386973, nanosec=1986750162), frame_id='')
info:: {}
image:: std_msgs.msg.Header(stamp= builtin_interfaces.msg.Time(sec=1714386973, nanosec=2019761453), frame_id='')
info:: {}
image:: std_msgs.msg.Header(stamp= builtin_interfaces.msg.Time(sec=1714386973, nanosec=2053709632), frame_id='')
```

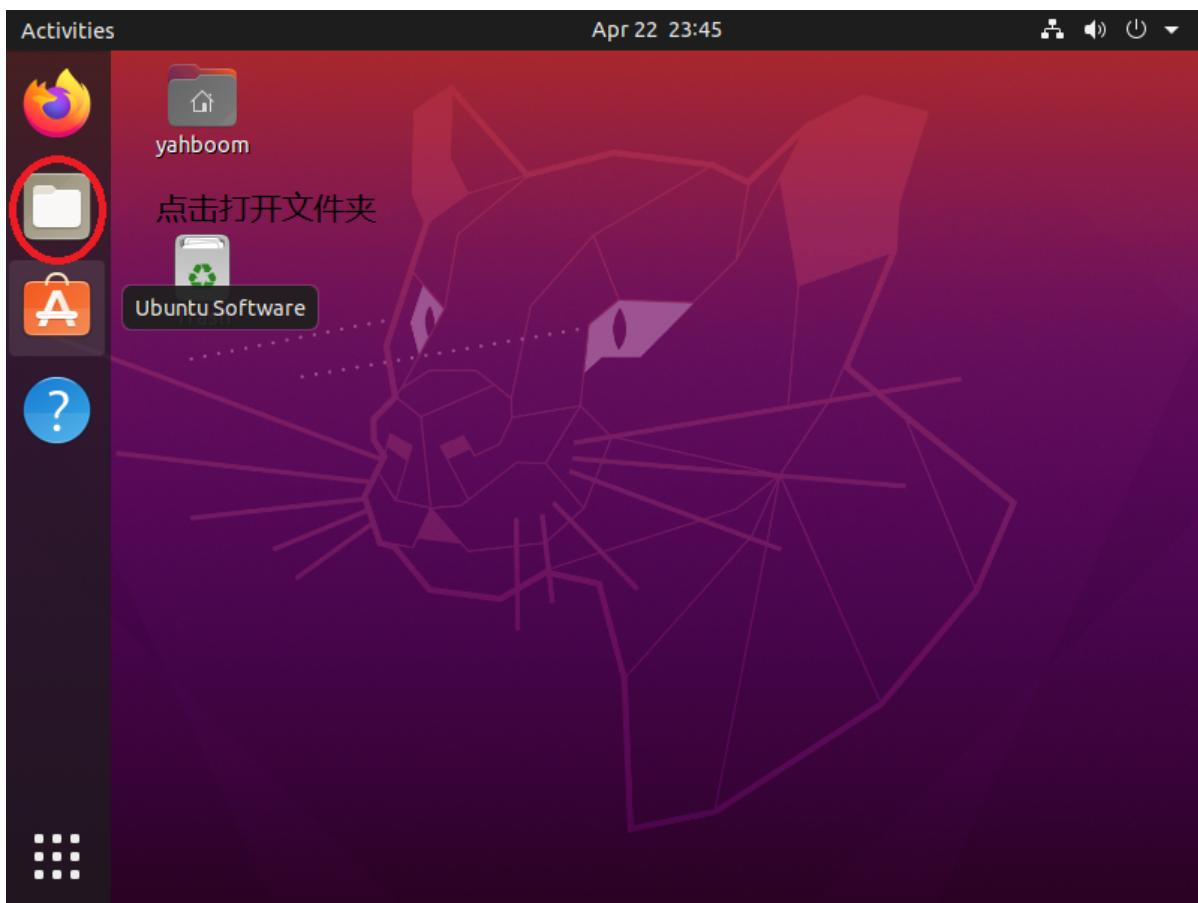
5. Set the recognition color through the web interface

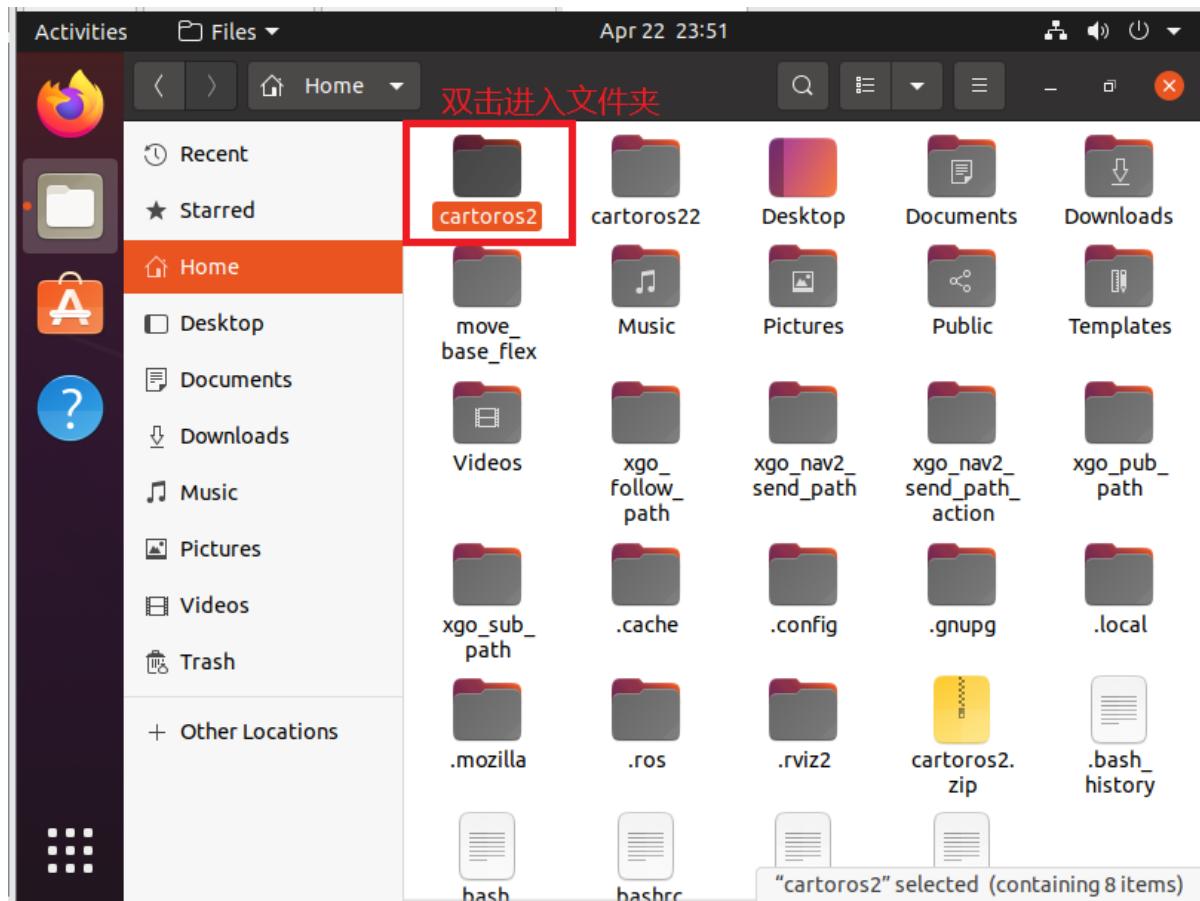
The steps are the same for PI4 and PI5 versions:

Open the virtual machine and enter the user name yahboom and the password yahboom.

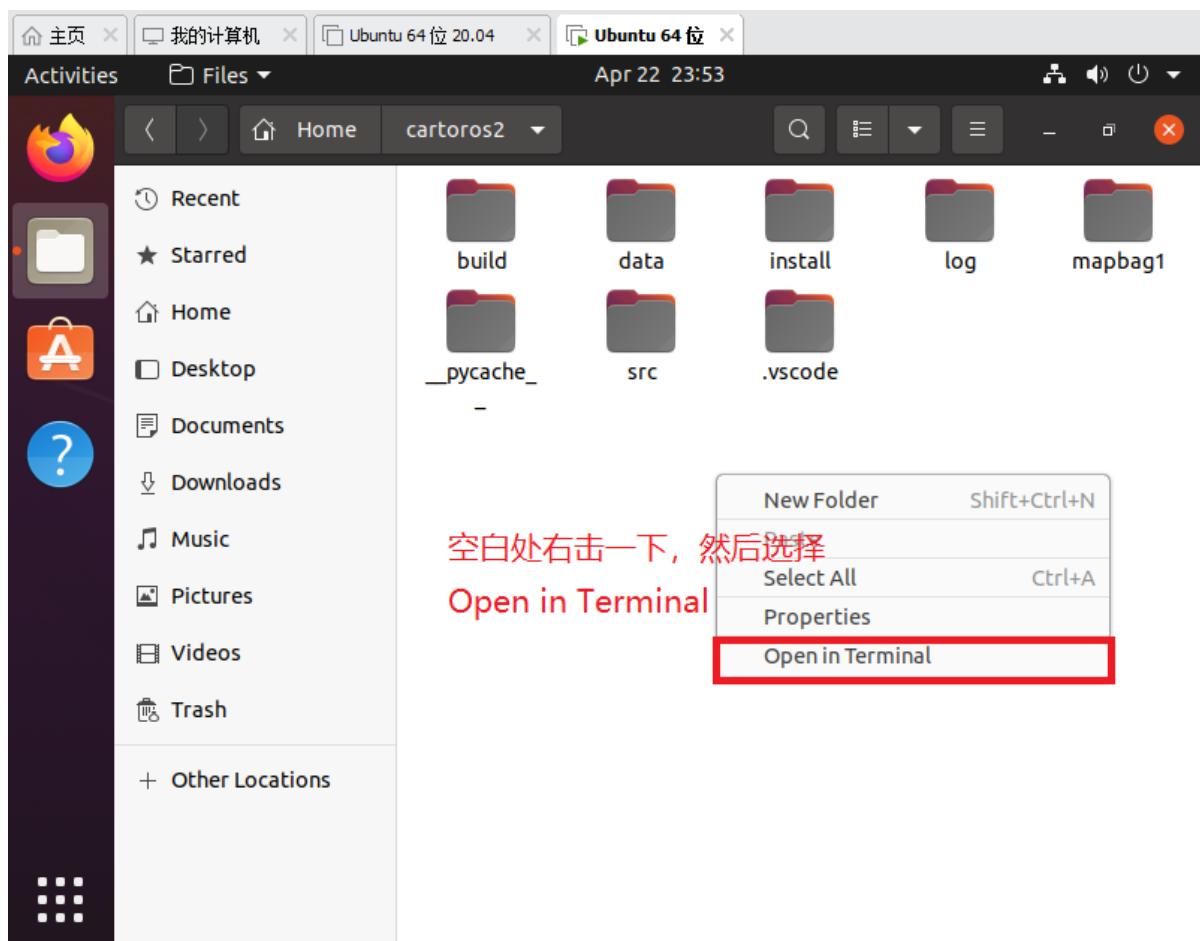


Click on the folder to open the cartoros2 folder.



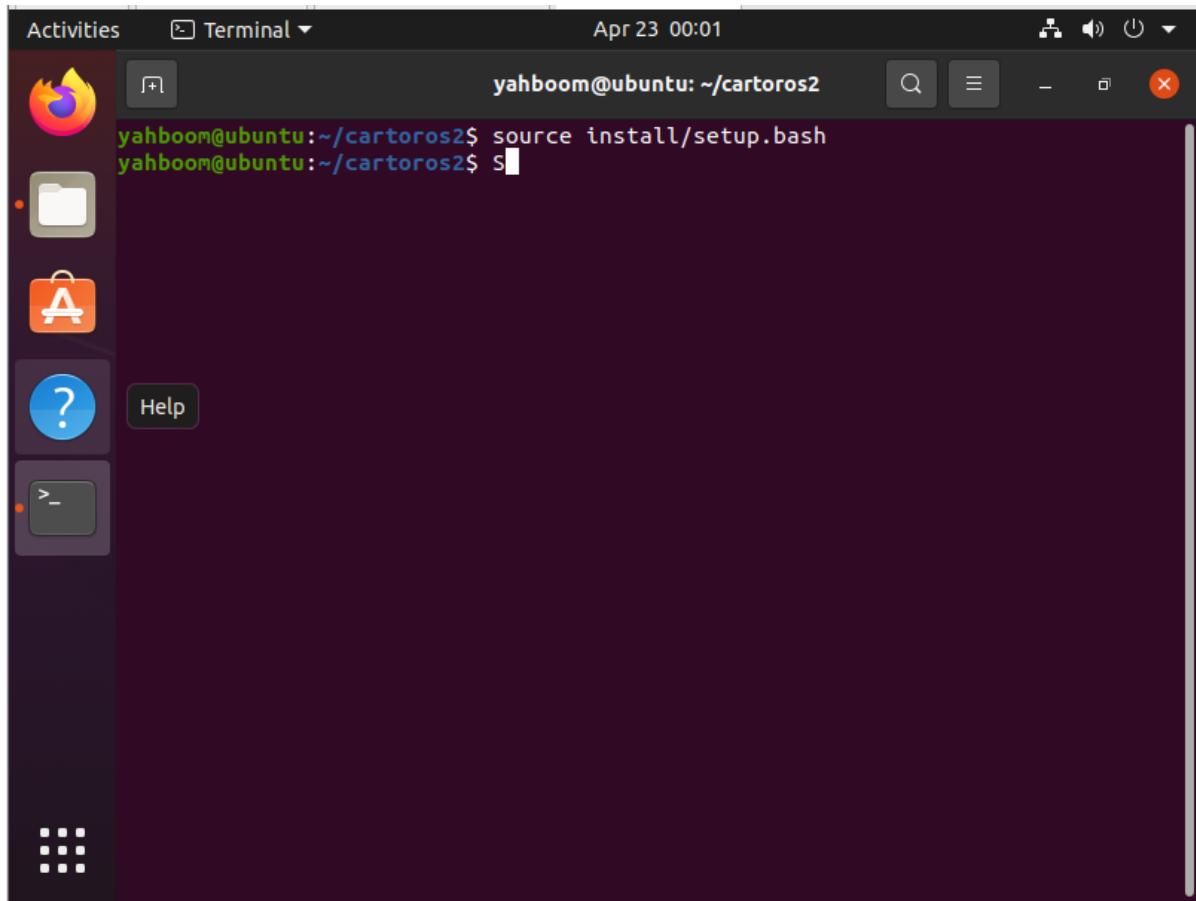


Open the terminal under the folder



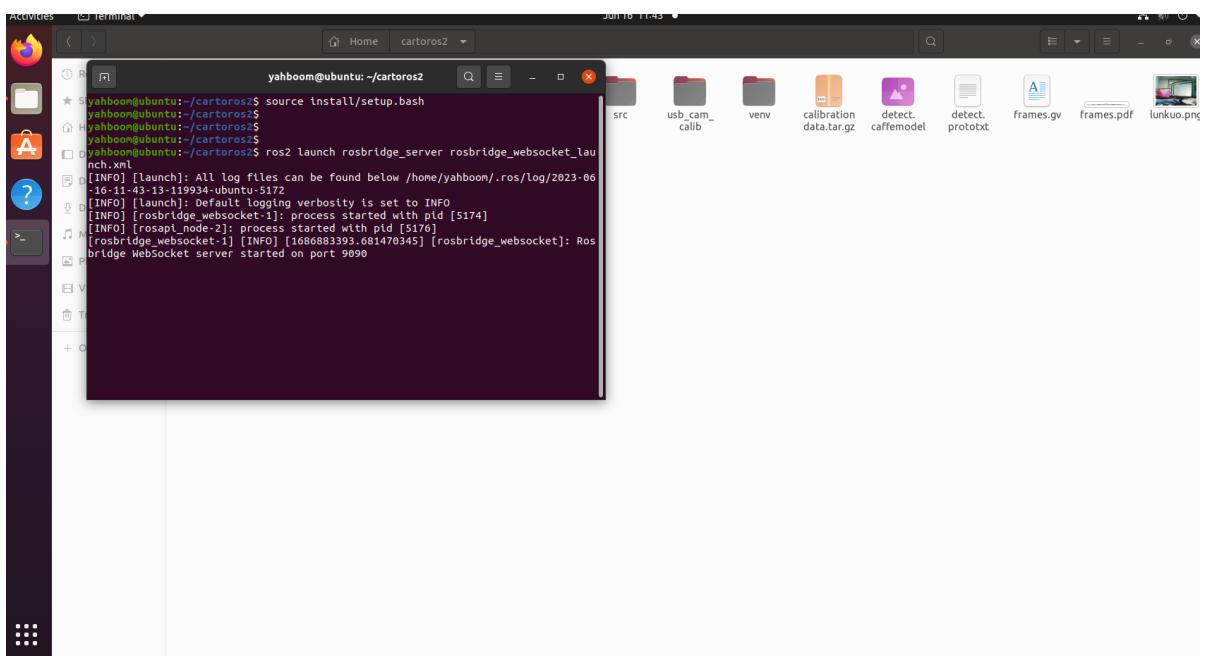
Then enter the following command

```
source install/setup.bash
```



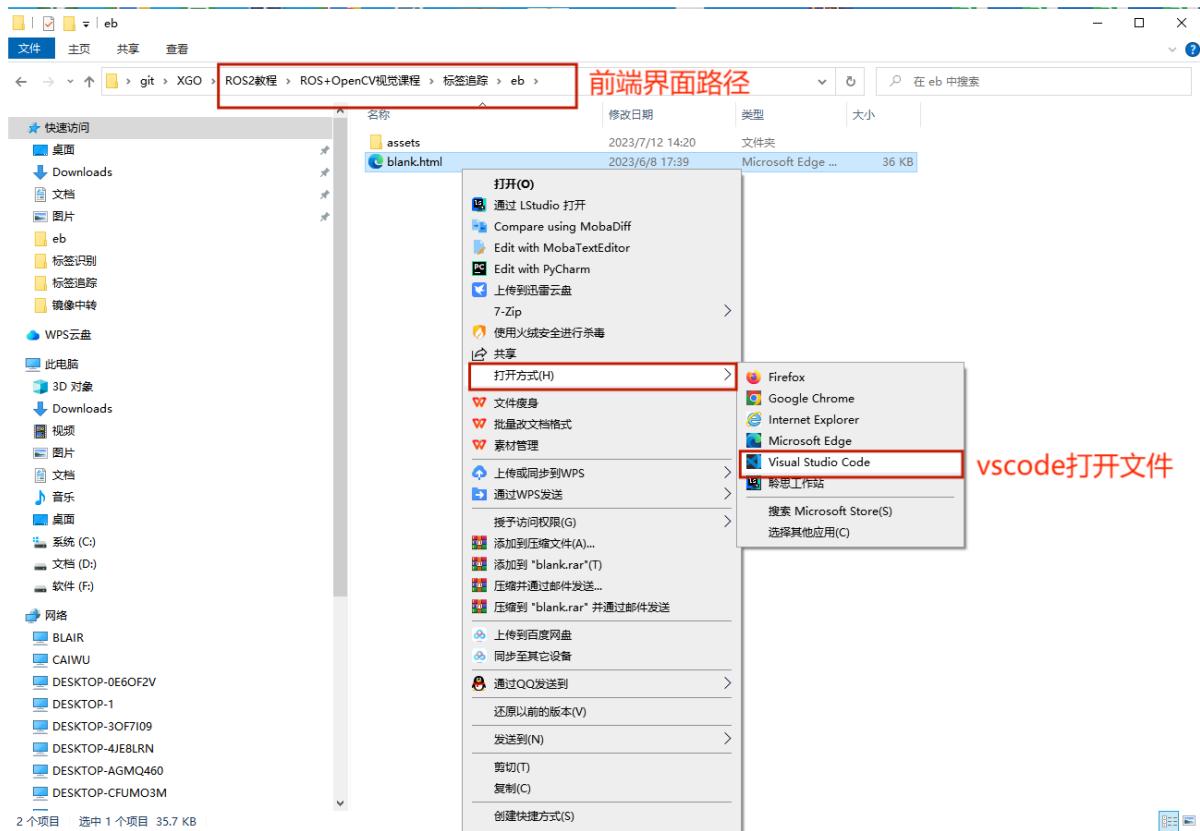
Then start rosbridge and enter the following command

```
ros2 launch rosbridge_server rosbridge_websocket_launch.xml
```



Find the blank.html file in the eb folder in the updated directory of this tutorial and open it with Google Chrome.

Note: The IP address of rosbridge needs to be set here. Obtain the IP address of the virtual machine, then open the blank.html file, modify the IP address of line 363 and save it, as shown in the figure below.



```
var ros = new ROSLIB.Ros({
  url : 'ws://192.168.2.117:9090'
});

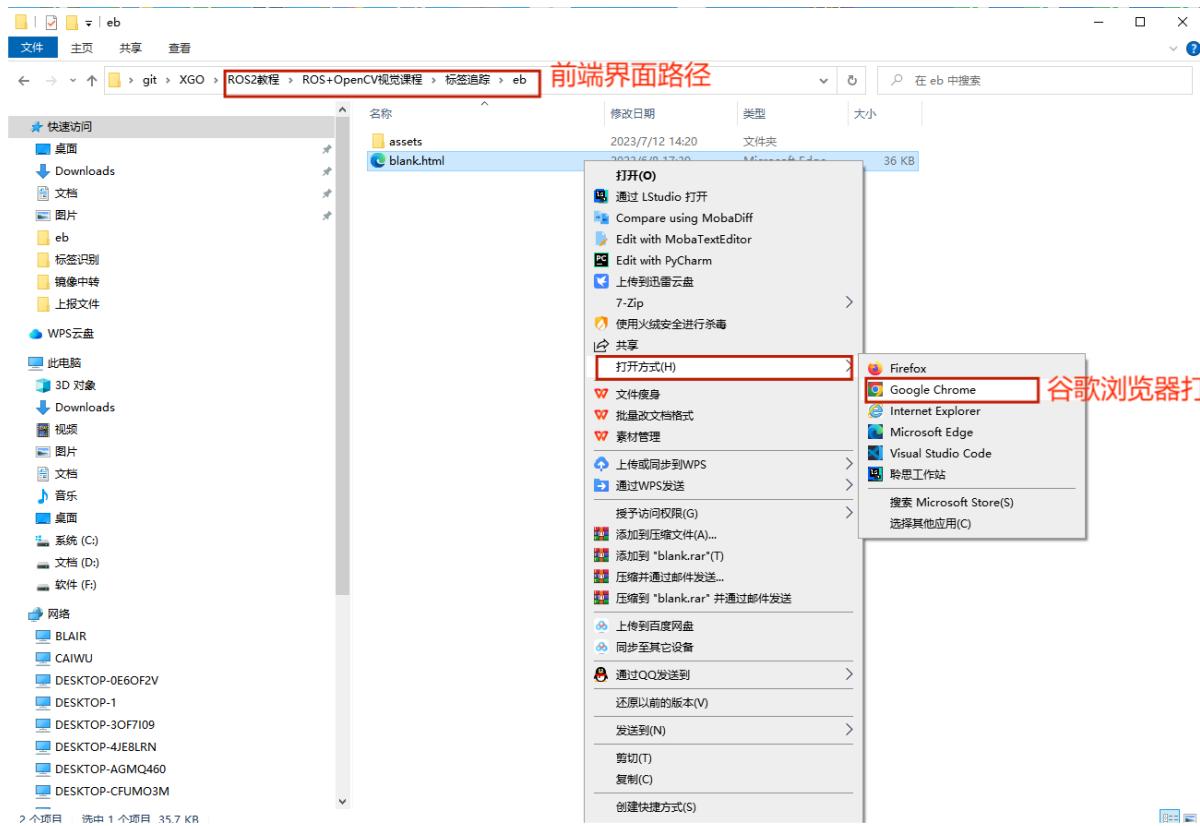
ros.on('connection', function() {
  console.log('Connected to websocket server.');
});

ros.on('error', function(error) {
  console.log('Error connecting to websocket server: ', error);
});

ros.on('close', function() {
  console.log('Connection to websocket server closed.');
});

var imageLisener_compressed= new ROSLIB.Topic({
  ros : ros,
  name : '/image_raw/compressed',
  messageType : 'sensor_msgs/msg/CompressedImage',
  throttle_rate : 100
});
var that = this;
imageLisener_compressed.subscribe(function(data) {
  // update the robot's position on the path
});
```

修改这里的IP为实际rosbridge的IP地址



As shown in the picture below, use your mobile phone to display a QR code. Here is the WeChat QR code. You can see the picture transmitted by the camera and the recognized QR code data. After recognizing the QR code, the robot dog calculates the distance from the QR code to follow the QR code forward and backward.

