

# Color Tracking

## 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.

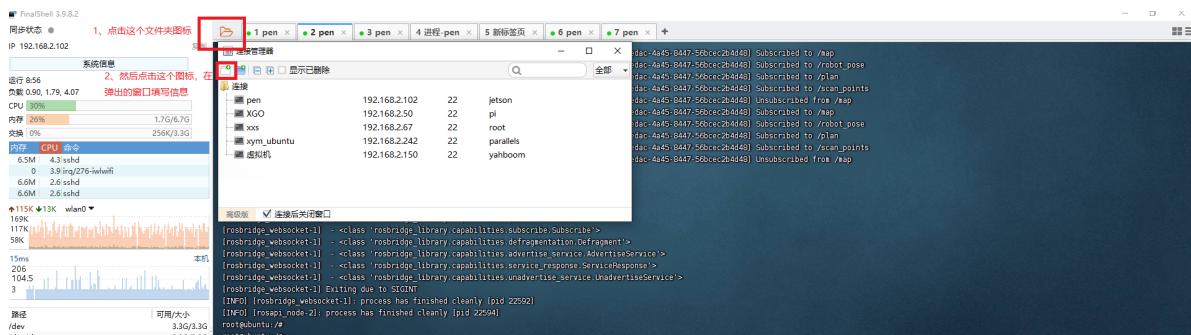
### 2. Start DOGZILLA chassis

#### PI4 version steps:

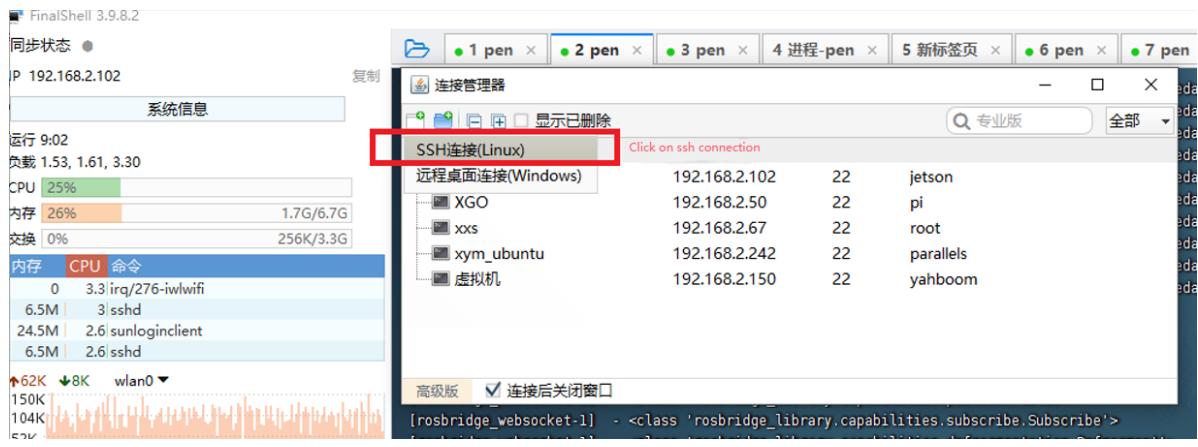
Then use the ssh terminal to connect to the robot dog.

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.

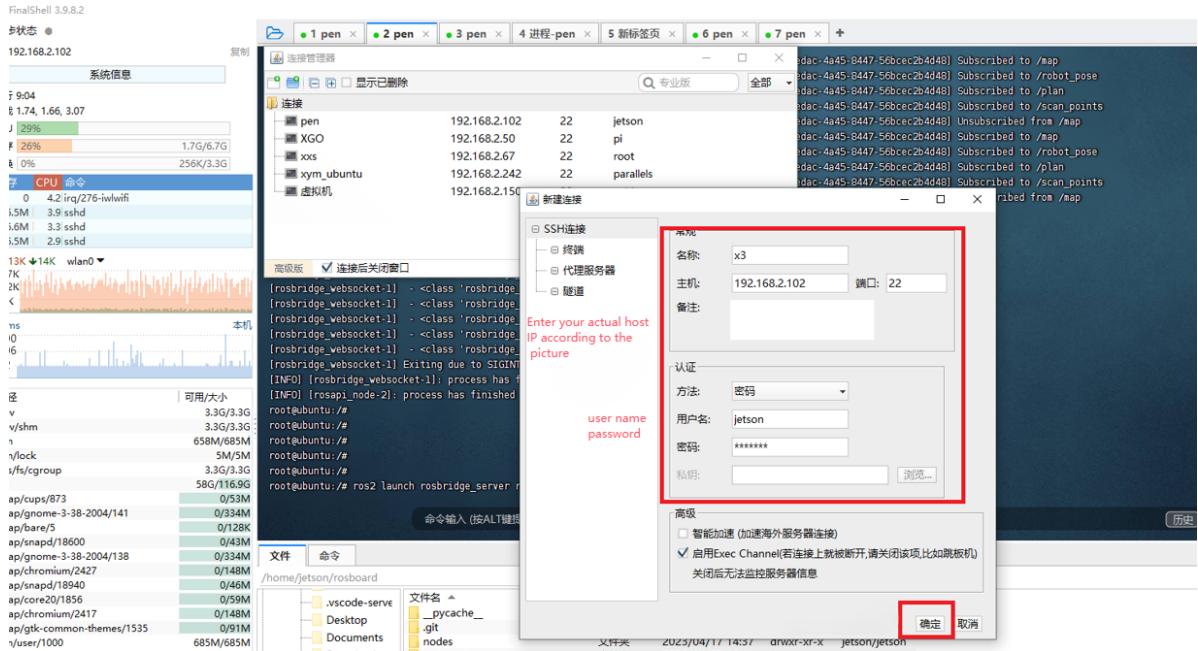
Open the shell tool. The shell tool I use here is FinalShell. Enter username, password, port, connection name and other information.



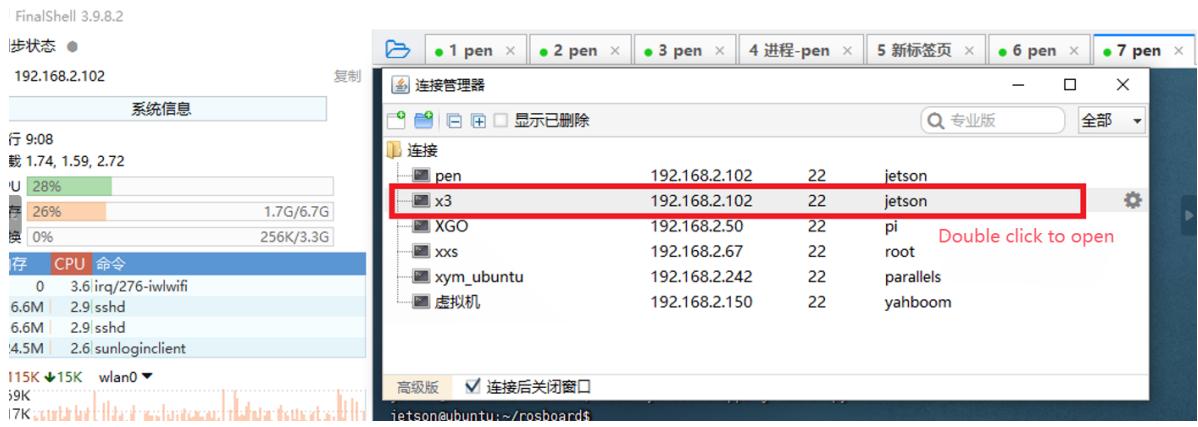
Select ssh connection to create a new ssh connection



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.



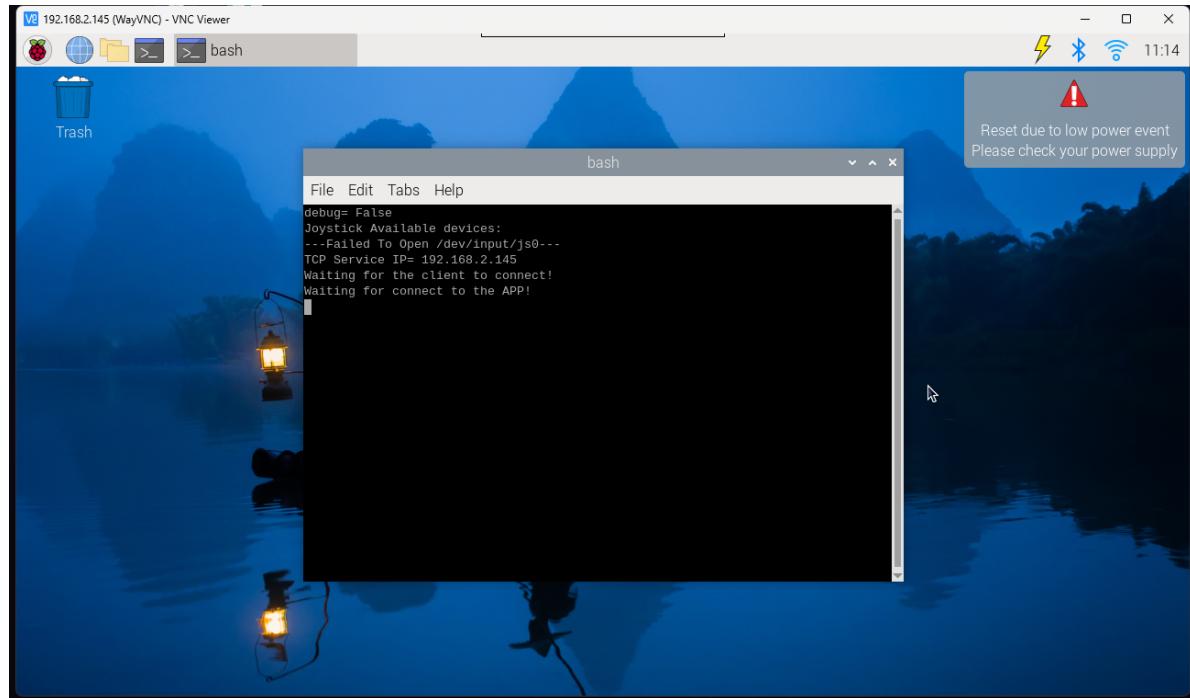
Enter the command in the terminal to start the chassis task.

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

```
pi@yahboom:~$  
pi@yahboom:~$  
pi@yahboom:~$  
pi@yahboom:~$  
pi@yahboom:~$ sudo systemctl restart XgoStart.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:

Enter the following 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_image_publisher_c yahboom_image_publish_c
```

```

pi@yahboom:~/cartographer_ws2$ source install/setup.bash
pi@yahboom:~/cartographer_ws2$ 
pi@yahboom:~/cartographer_ws2$ 
pi@yahboom:~/cartographer_ws2$ 
pi@yahboom:~/cartographer_ws2$ ros2 run xgo_image_publisher_c xgo_image_publish_c
[ WARN:0] global ../modules/videio/src/cap_gstreamer.cpp (1758) handleMessage OpenCV | GStreamer warning: Embedded video playback halted; mod
[ WARN:0] global ../modules/videio/src/cap_gstreamer.cpp (888) open OpenCV | GStreamer warning: unable to start pipeline
[ WARN:0] global ../modules/videoio/src/cap_gstreamer.cpp (480) isPipelinePlaying OpenCV | GStreamer warning: GStreamer: pipeline have not bee

```

Restart a terminal in the same way as in item 2.

```
连接主机...
连接主机成功
Last login: Fri Jun 16 10:18:28 2023 from 192.168.2.64
pi@yahboom:~$
```

Enter the following command in a new terminal

```
cd cartographer_ws2/
```

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

```
ros2 run yahboom_color_tracking yahboom_color_tracking
```

```
pi@yahboom:~$ cd cartographer_ws2/
pi@yahboom:~/cartographer_ws2$ source install/setup.bash
pi@yahboom:~/cartographer_ws2$ ros2 run xgo_color_tracking xgo_color_tracking
ddddddddd
```

#### 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
```

```
pi@raspberrypi:~ $ docker ps
CONTAINER ID   IMAGE   COMMAND      CREATED
STATUS          PORTS     NAMES
c06cd712e14e   yahboomtechnology/ros-humble:3.1   "/bin/bash"   24 minutes ago
Up 24 minutes           nice_keldysh
pi@raspberrypi:~ $ docker exec -it c06c /bin/bash
root@raspberrypi:/#
```

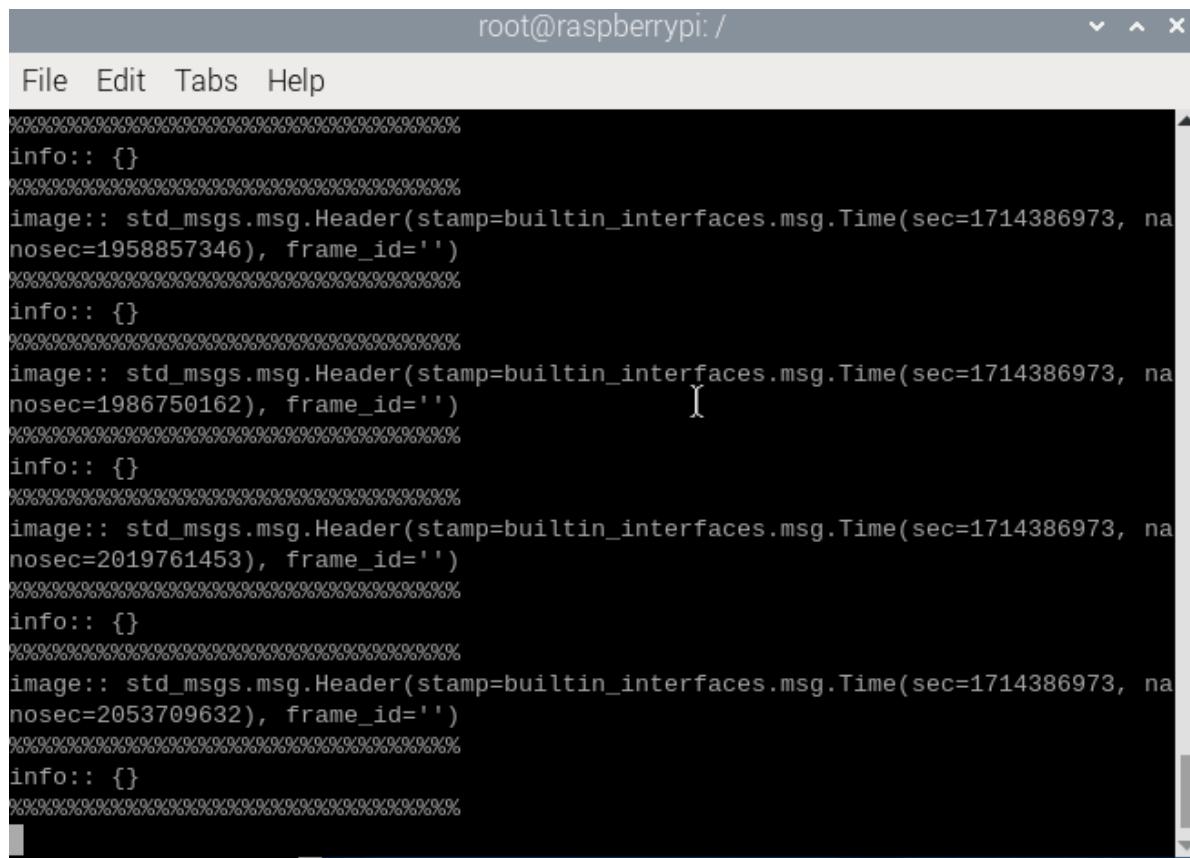
Then enter the following command

```
ros2 run yahboom_publish pub_color
```

```
[ WARN:0] global ./modules/videoio/src/cap_gstremer.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_gstremer.cpp (1053) open OpenCV | GS treamer warning: unable to start pipeline
[ WARN:0] global ./modules/videoio/src/cap_gstremer.cpp (616) isPipelinePlaying OpenCV | GStreamer warning: GStreamer: pipeline have not been created
```

Reopen the same docker terminal and enter the command:

```
ros2 run yahboom_qrcode_tracking yahboom_qrcode_tracking
```

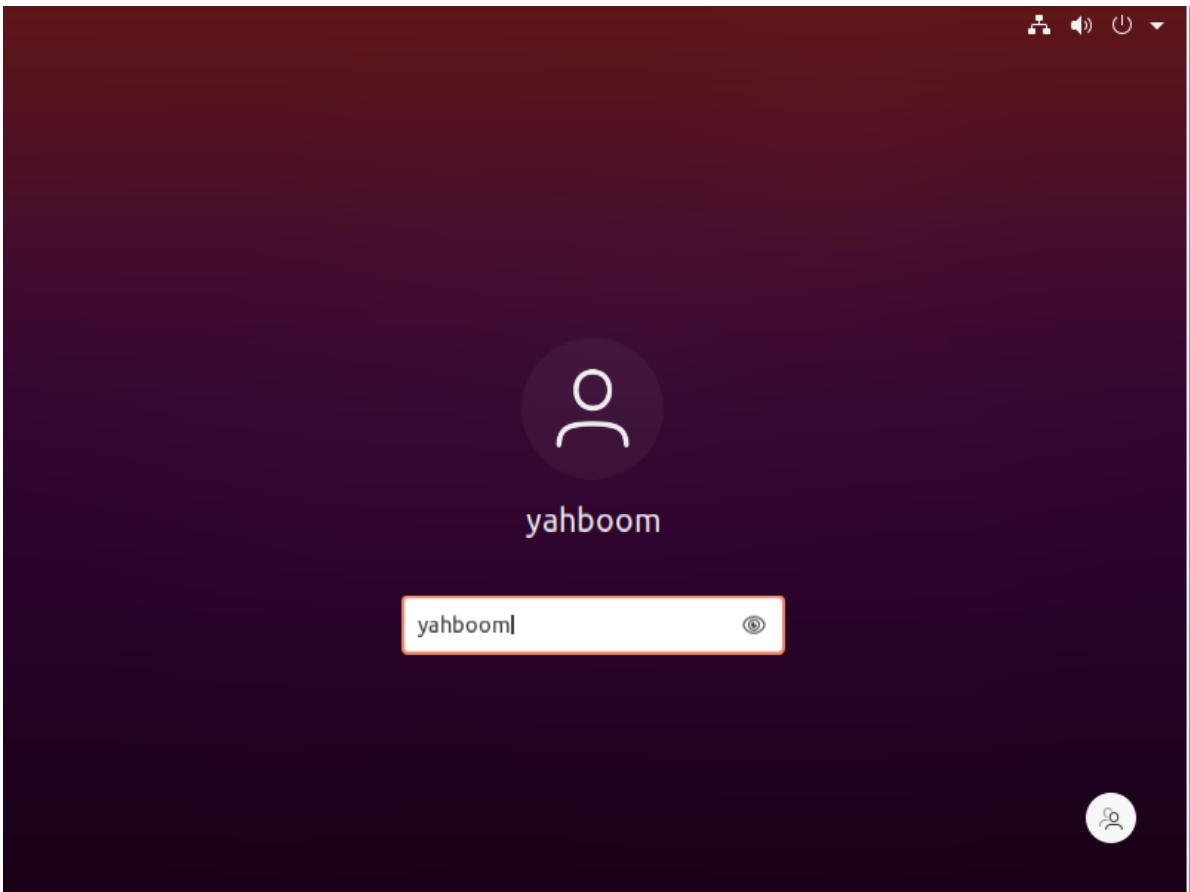


The screenshot shows a terminal window titled "root@raspberrypi: /". The window contains several lines of text output from a ROS 2 node named "yahboom\_qrcode\_tracking". The output consists of multiple "info:: {}" messages, each preceded by a timestamp and a header message. The timestamps are approximately 1714386973, 1958857346, 1986750162, 2019761453, and 2053709632. The header message for each timestamp is "image:: std\_msgs.msg.Header(stamp= builtin\_interfaces.msg.Time(sec=1714386973, nanosec=1958857346), frame\_id='')". The terminal has a dark background with light-colored text and includes standard window controls (minimize, maximize, close) at the top.

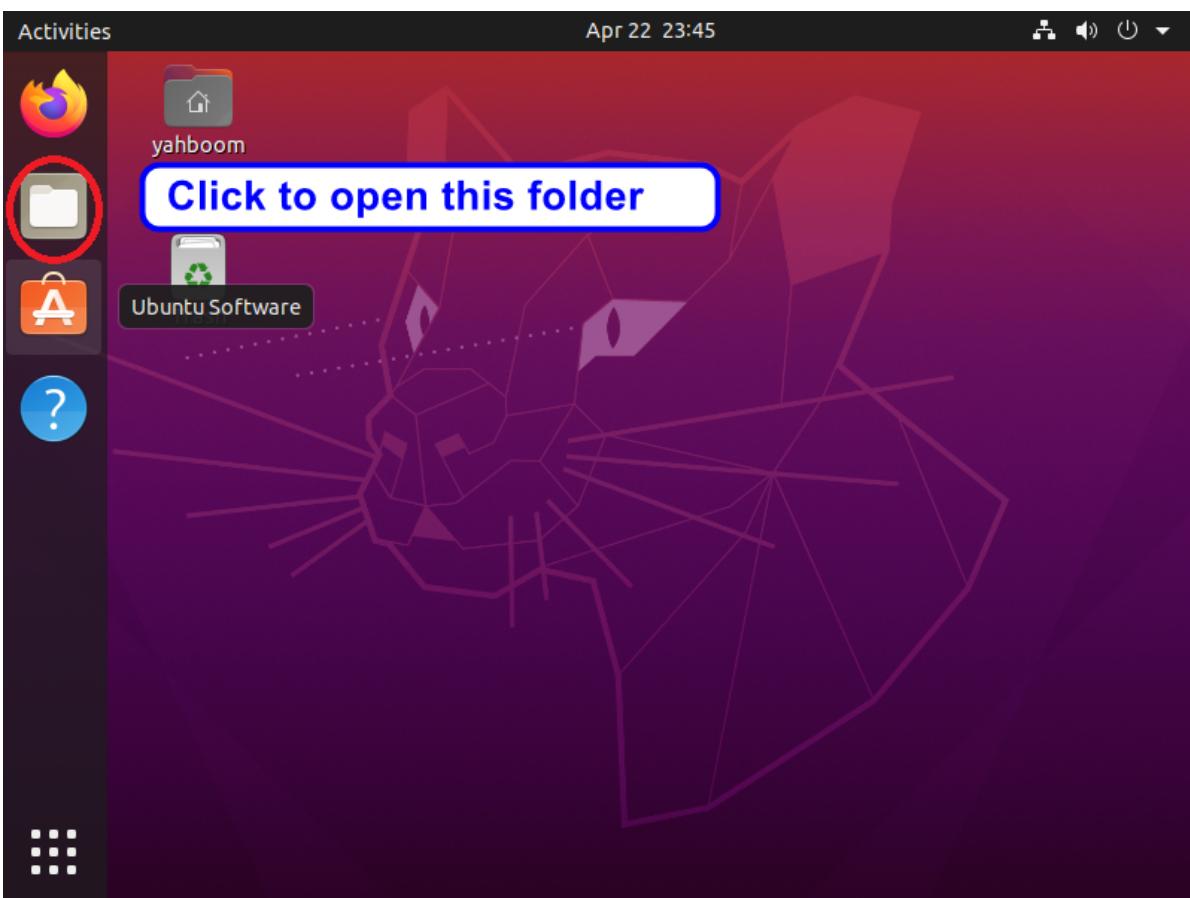
## 5. Set the recognition color through the web interface

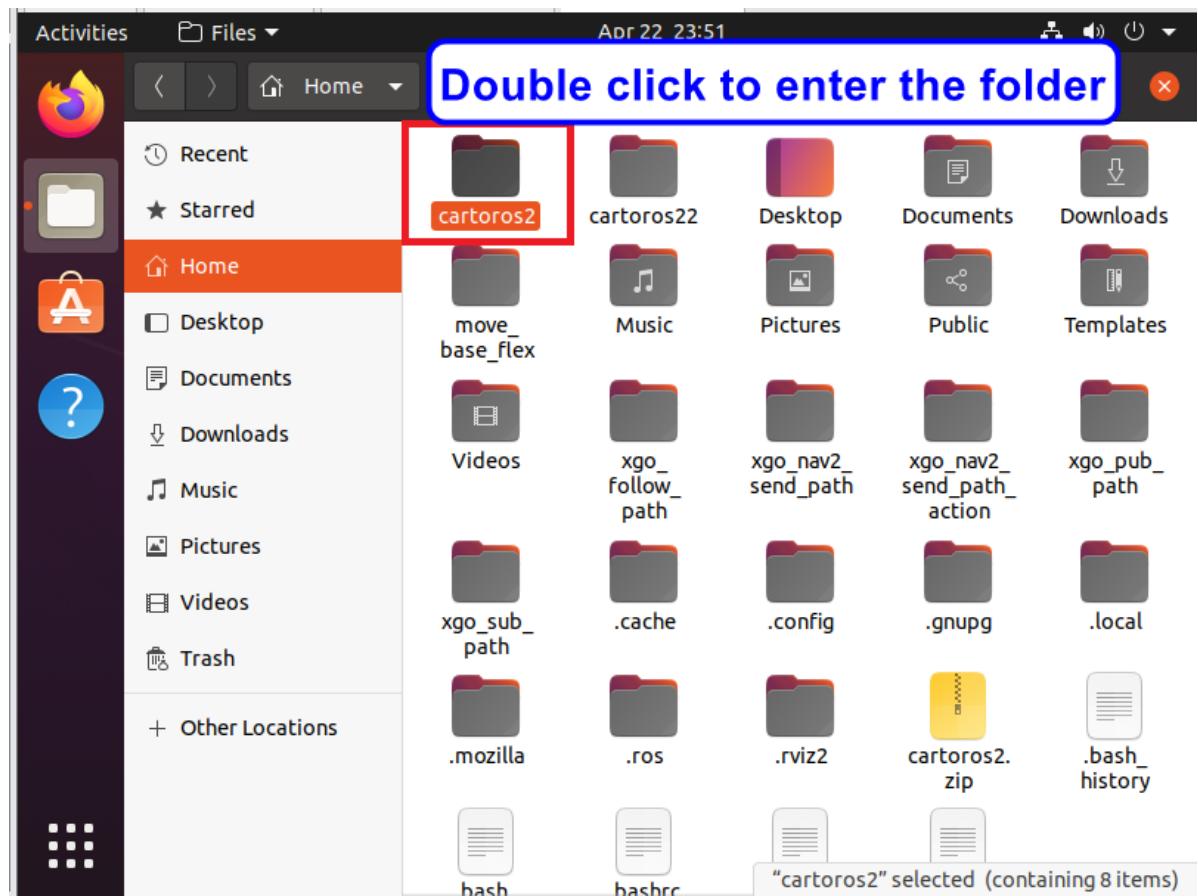
**The steps are the same for PI4 and PI5 versions:**

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

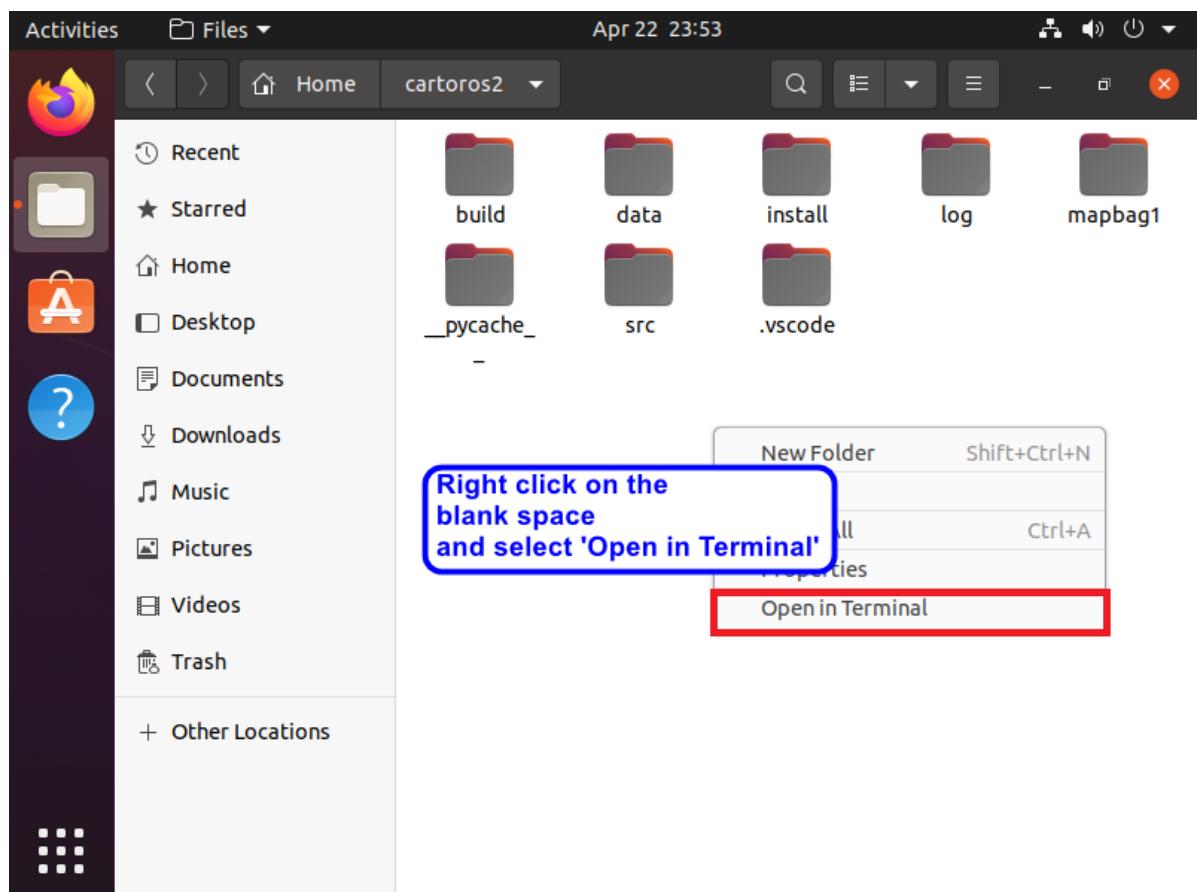


Click on the folder to open the cartoros2 folder.



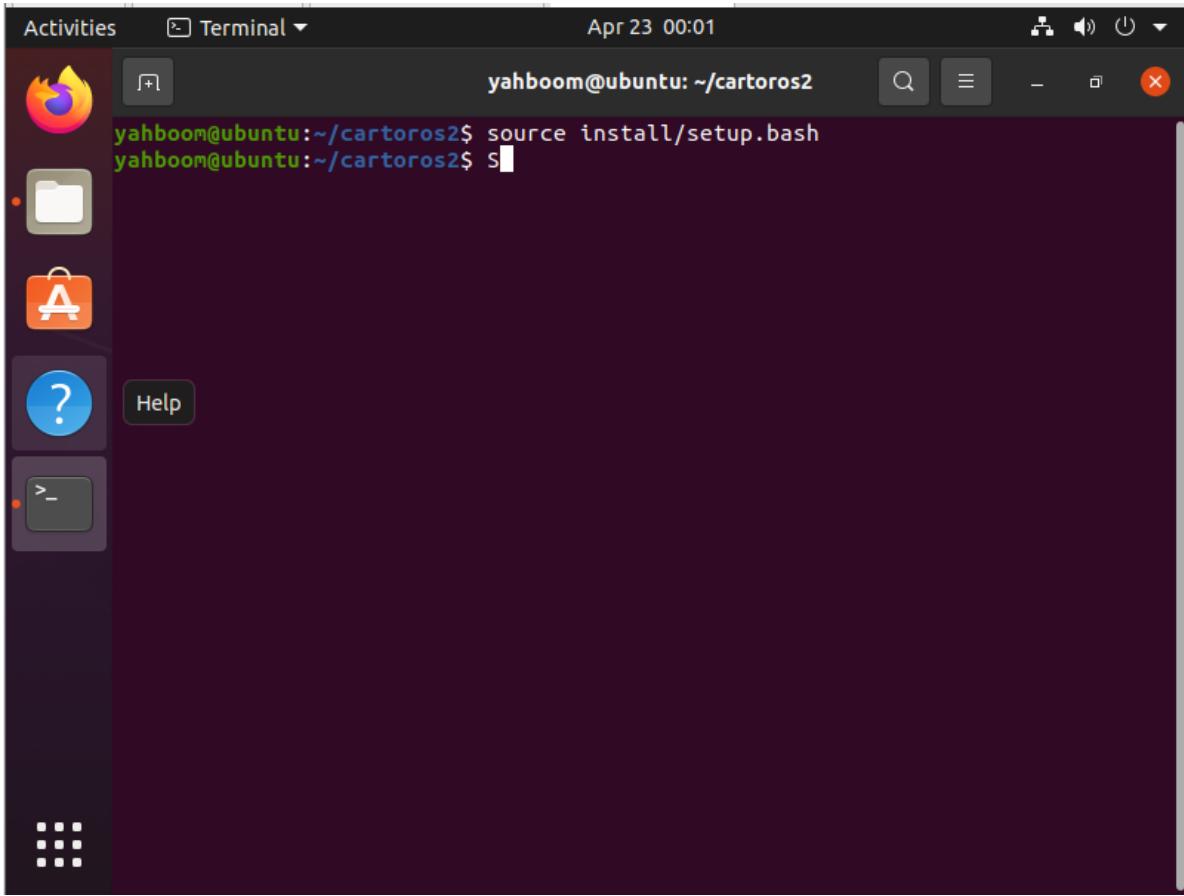


Open the terminal under the folder



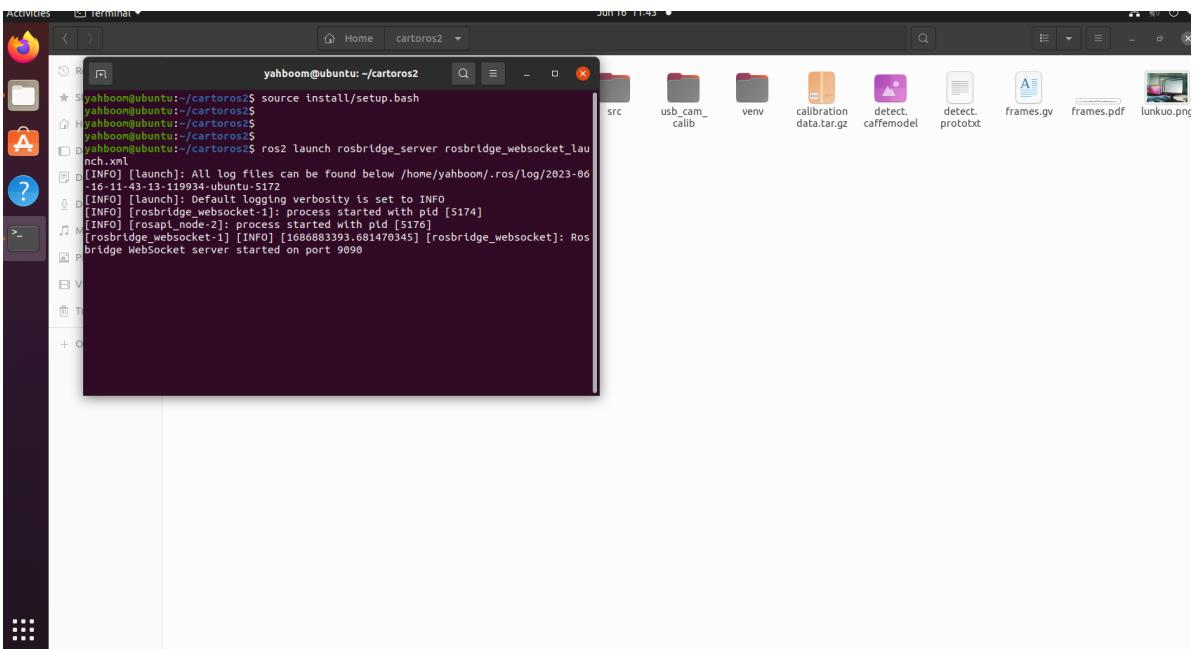
Then enter the following command

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



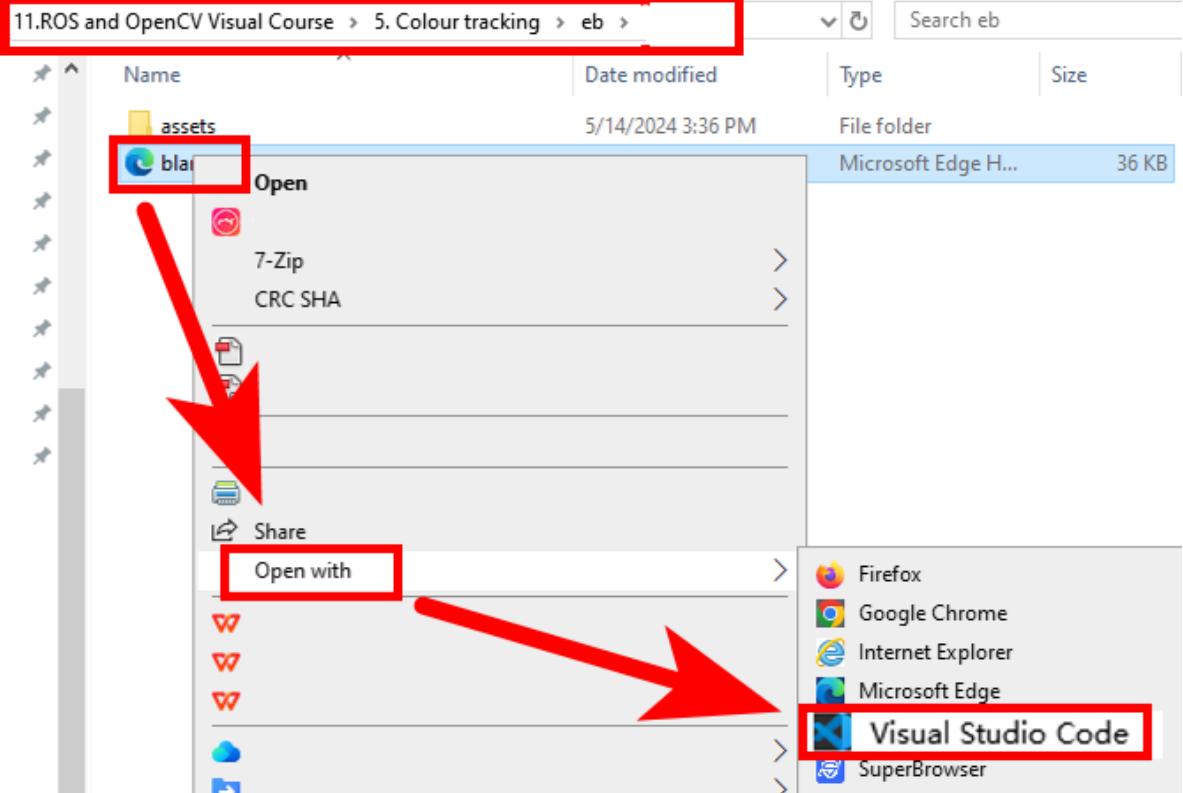
Then start rosbridge and enter the following command

```
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.



blank.html

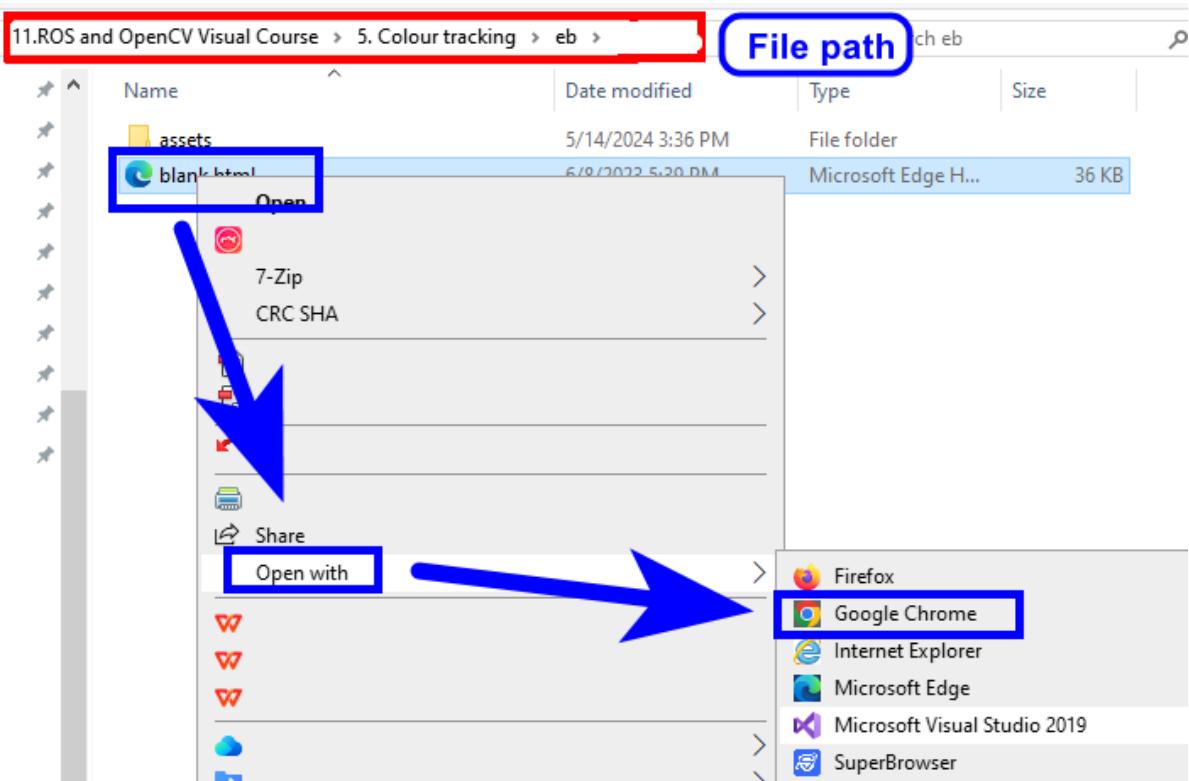
```

C:\> Users > Admin > Desktop > git > XGO > ROS2教程 > ROS+OpenCV视觉课程 > 标签追踪 > eb > blank.html > html > body > script
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363 var ros = new ROSLIB.Ros({
364   url : 'ws://192.168.2.117:9090'
365 });
366
367 ros.on('connection', function() {
368   console.log('Connected to websocket server.');
369 });
370
371 ros.on('error', function(error) {
372   console.log('Error connecting to websocket server: ', error);
373 });
374
375 ros.on('close', function() {
376   console.log('Connection to websocket server closed.');
377 });
378
379 var imagerender_compressed= new ROSLIB.Topic({
380   ROS : ros,
381   name : '/image_raw/compressed',
382   messageType : 'sensor_msgs/msg/CompressedImage',
383   throttle_rate : 100
384 });
385 var that = this;
386 imagerender_compressed.subscribe(function(data) {
387   // Update the robot's position on the map
388 });
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
759
760

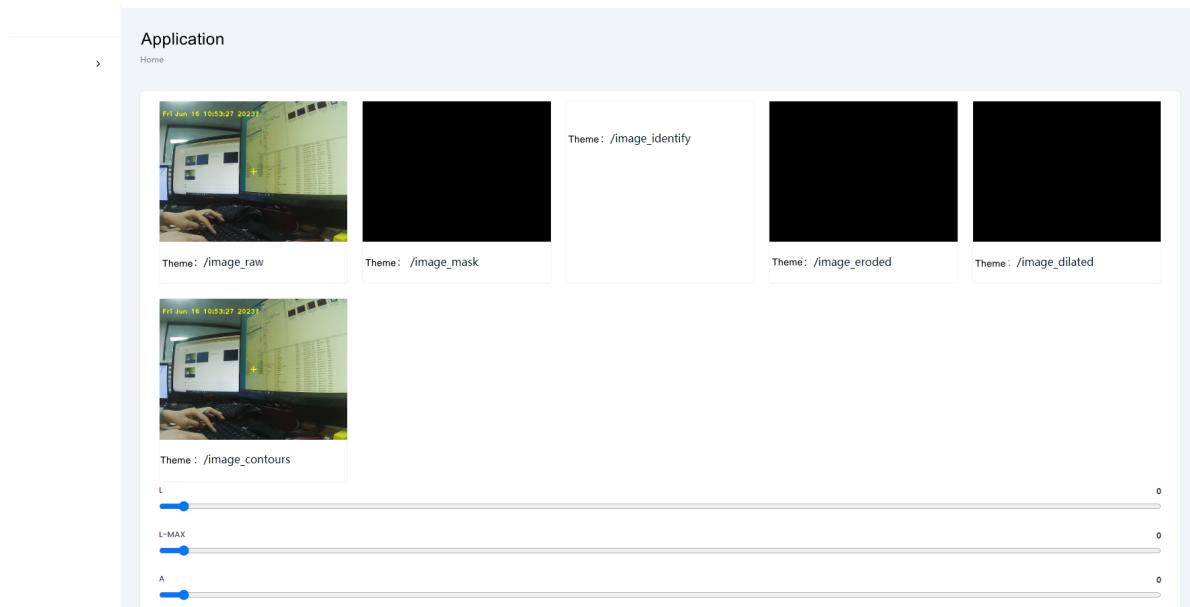
```

**Change the IP address here to the actual IP address of the RoboRidge**

在工作区的文件夹或打开的文件中找到了 git 存储库。是否要打开存储库？  
来源: Git 扩展  
是 始终 从不



As shown in the picture below, you can see the pictures transmitted by the camera.



Then we set the LAB value of the color through the slider.

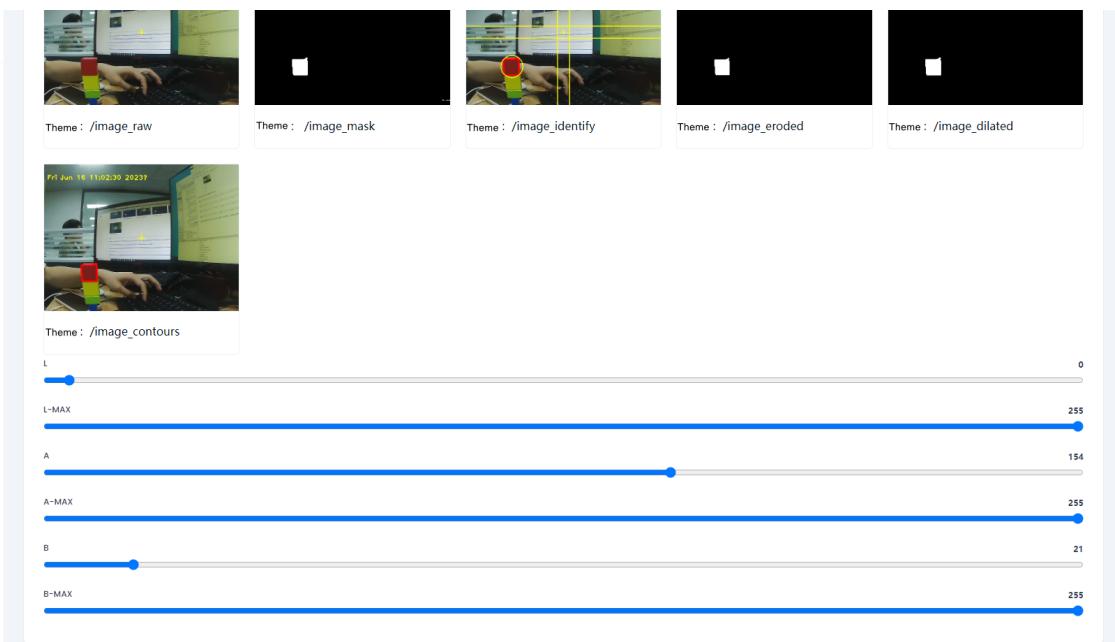
```
Yellow: {"l":96, "a": 55, "b":188, "l_max": 252 , "a_max": 141, "b_max": 255}
```

```
Red: {"l":0, "a": 155, "b":21, "l_max": 255 , "a_max": 255, "b_max": 255}
```

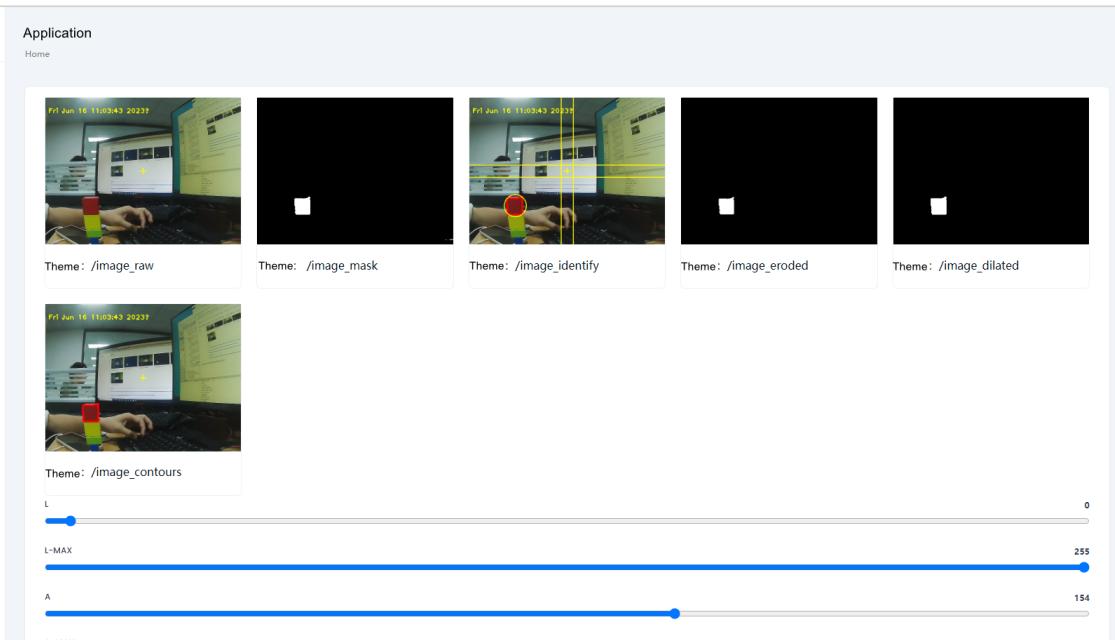
```
Green: {"l":26, "a": 7, "b":170, "l_max": 143 , "a_max": 110, "b_max": 255}
```

```
Blue: {"l":0, "a": 0, "b":0, "l_max": 255 , "a_max": 255, "b_max": 102}
```

Above are the LAB values of several colors. We can choose one to set. For example, we set red, as shown in the figure below, and move the slider.



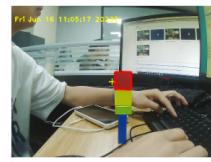
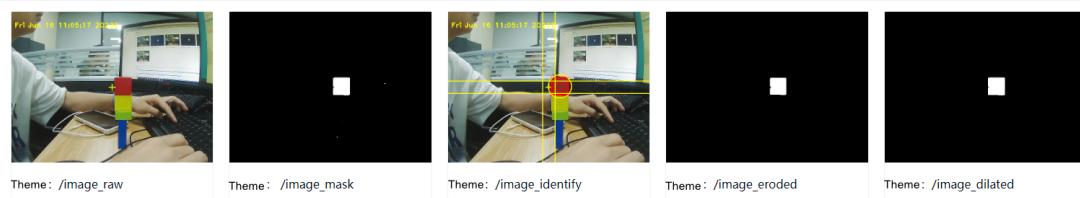
In the picture we can see that the red square has been identified.



Then the mechanical dog will adjust its attitude so that the red block is near the center of the screen.

## Application

Home



Theme: /image\_contours

L

L-MAX

A

A-MAX

0

255

154

255