Line speed calibration

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be the same. You can check [Must-read before use] to set the IP and ROS_DOMAIN_ID on the board.

1. Program function description

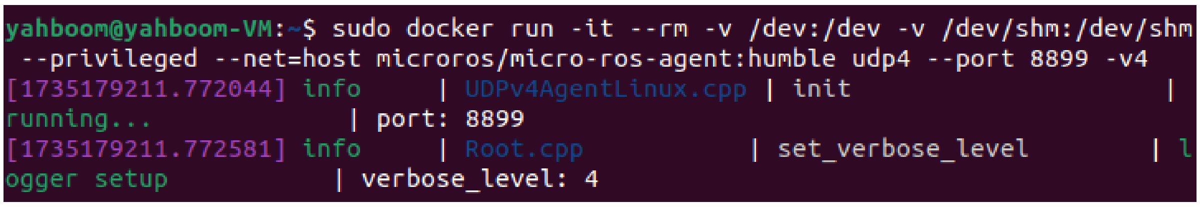
The car connects to the proxy, runs the program, and adjusts the parameters here to calibrate the car's line speed through the dynamic parameter regulator. The intuitive manifestation of calibrating the line speed is to let the car go straight forward for 1 meter to see how far it actually runs and whether it is within the error range.

```
Note: Before running the program, the car needs to be restarted in a stable
standing position to ensure that all sensors are reset
```

2. Start and connect the agent

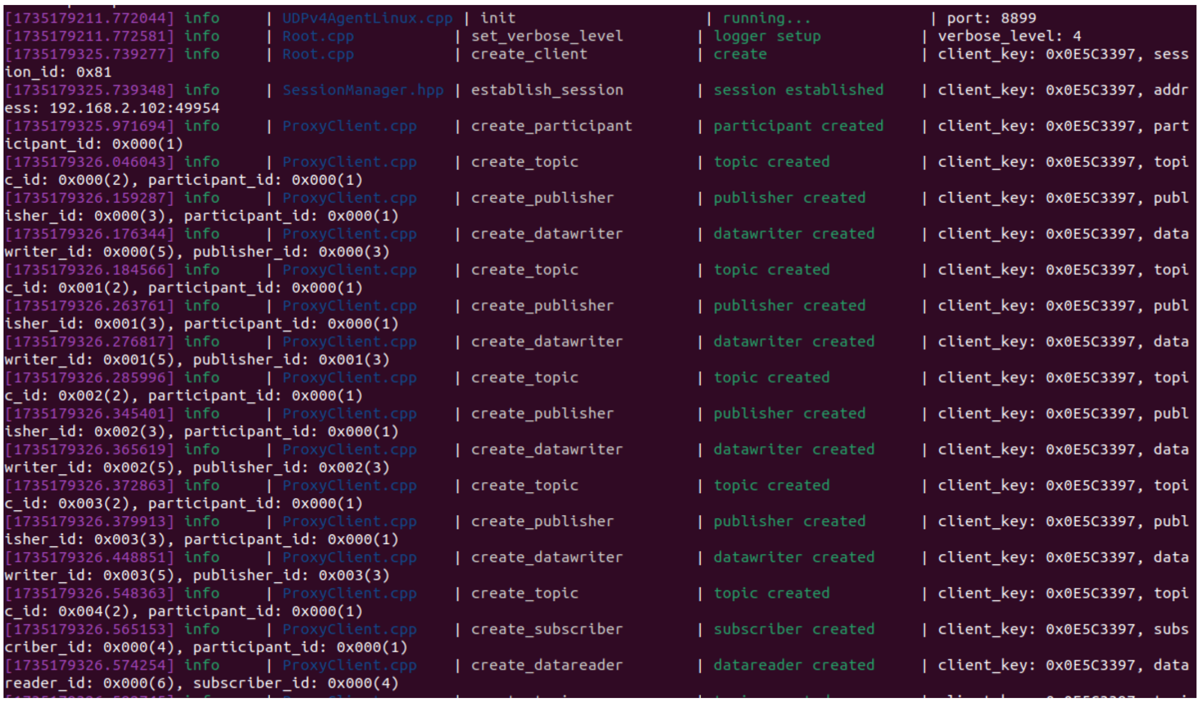
Take the supporting virtual machine as an example, enter the following command to start the agent,

```
sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm --privileged --
net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
```



```
yahboom@yahboom-VM:~$ sudo docker run -it --rm -v /dev:/dev -v /dev/shm:/dev/shm
 --privileged --net=host microros/micro-ros-agent:humble udp4 --port 8899 -v4
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                   |
running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level      | l
ogger setup            | verbose_level: 4
```

Then, turn on the car switch and wait for the car to connect to the proxy. The connection is successful as shown in the figure below.



```
[1735179211.772044] info     | UDPv4AgentLinux.cpp | init                   | running...             | port: 8899
[1735179211.772581] info     | Root.cpp            | set_verbose_level      | logger setup           | verbose_level: 4
[1735179325.739277] info     | Root.cpp            | create_client          | create                 | client_key: 0x0E5C3397, sess
ion_id: 0x81
[1735179325.739348] info     | SessionManager.hpp | establish_session       | session established    | client_key: 0x0E5C3397, addr
ess: 192.168.2.102:49954
[1735179325.971694] info     | ProxyClient.cpp     | create_participant     | participant created    | client_key: 0x0E5C3397, part
icipant_id: 0x000(1)
[1735179326.046043] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1735179326.159287] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1735179326.176344] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1735179326.184566] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1735179326.263761] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1735179326.276817] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1735179326.285996] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1735179326.345401] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1735179326.365619] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1735179326.372863] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1735179326.379913] info     | ProxyClient.cpp     | create_publisher       | publisher created      | client_key: 0x0E5C3397, publ
isher_id: 0x003(3), participant_id: 0x000(1)
[1735179326.448851] info     | ProxyClient.cpp     | create_datawriter      | datawriter created     | client_key: 0x0E5C3397, data
writer_id: 0x003(5), publisher_id: 0x003(3)
[1735179326.548363] info     | ProxyClient.cpp     | create_topic           | topic created          | client_key: 0x0E5C3397, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1735179326.565153] info     | ProxyClient.cpp     | create_subscriber      | subscriber created     | client_key: 0x0E5C3397, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1735179326.574254] info     | ProxyClient.cpp     | create_datareader      | datareader created     | client_key: 0x0E5C3397, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
```

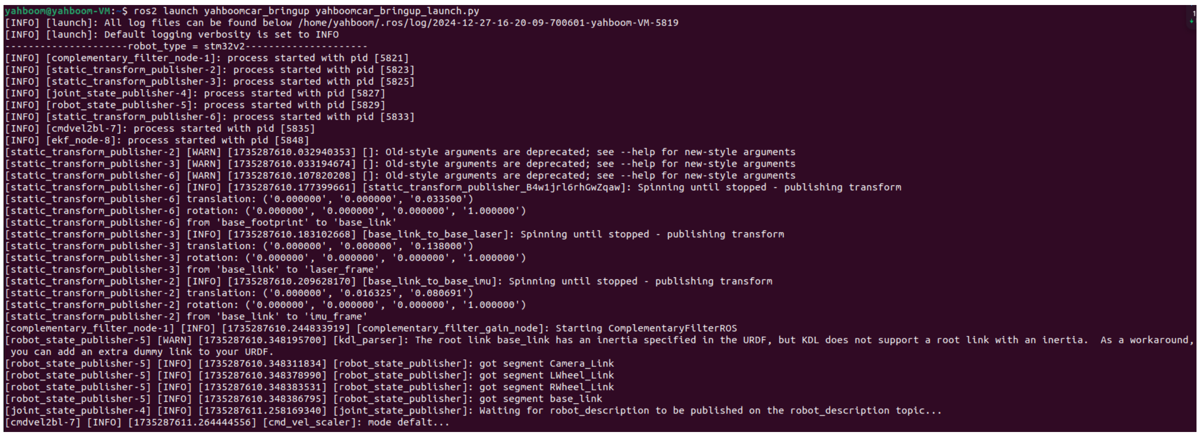
3. Start the program

3.1 Run the command

Take the matching virtual machine as an example, input in the terminal,

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

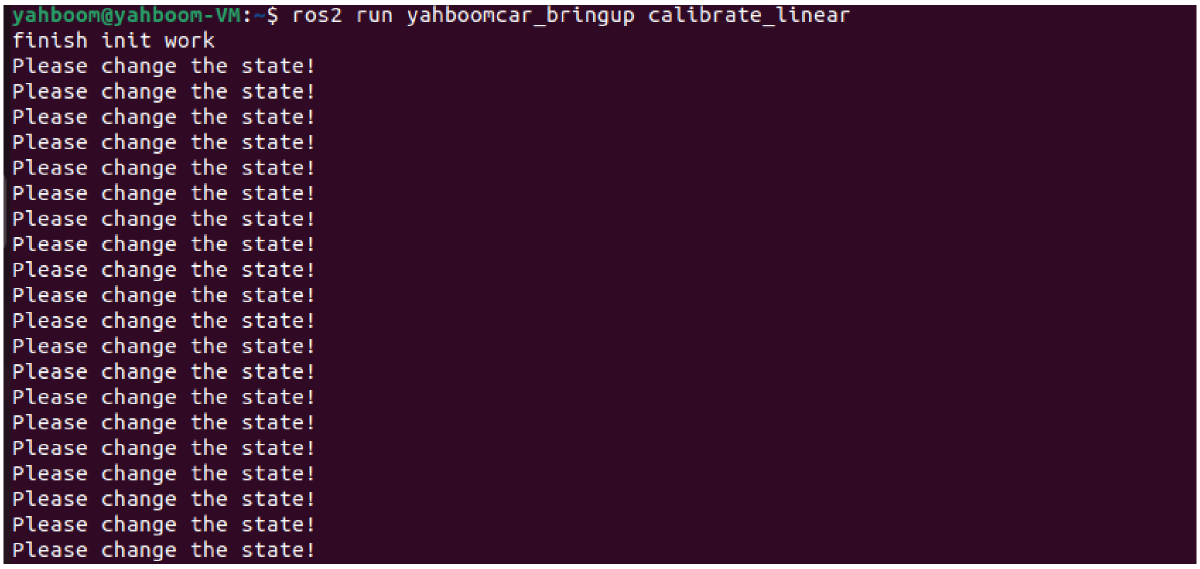
First, start the car to process the underlying data program. The program will publish the TF transformation of odom->base_footprint. With this TF transformation, we can calculate "how far the car has traveled".



```
yahboom@yahboom-VM:~$ ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-12-27-16-20-09-700601-yahboom-VM-5819
[INFO] [launch]: Default logging verbosity is set to INFO
---------------------robot_type = stm32v2--------------------
[INFO] [complementary_filter_node-1]: process started with pid [5821]
[INFO] [static_transform_publisher-2]: process started with pid [5823]
[INFO] [static_transform_publisher-3]: process started with pid [5825]
[INFO] [joint_state_publisher-4]: process started with pid [5827]
[INFO] [robot_state_publisher-5]: process started with pid [5829]
[INFO] [static_transform_publisher-6]: process started with pid [5833]
[INFO] [cmdvel2bl-7]: process started with pid [5835]
[INFO] [ekf_node-8]: process started with pid [5848]
[static_transform_publisher-2] [WARN] [1735287610.032940353] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [WARN] [1735287610.033194674] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-6] [WARN] [1735287610.107820208] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-6] [INFO] [1735287610.177399661] [static_transform_publisher_B4w1jrl6rhGwZqaw]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.033500')
[static_transform_publisher-6] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-6] from 'base_footprint' to 'base_link'
[static_transform_publisher-3] [INFO] [1735287610.183102668] [base_link_to_base_laser]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('0.000000', '0.000000', '0.138000')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'laser_frame'
[static_transform_publisher-2] [INFO] [1735287610.209628170] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-2] translation: ('0.000000', '0.016325', '0.080691')
[static_transform_publisher-2] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-2] from 'base_link' to 'imu_frame'
[complementary_filter_node-1] [INFO] [1735287610.244833919] [complementary_filter_gain_node]: Starting ComplementaryFilterROS
[robot_state_publisher-5] [WARN] [1735287610.348195700] [kdl_parser]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an inertia.  As a workaround,
 you can add an extra dummy link to your URDF.
[robot_state_publisher-5] [INFO] [1735287610.348311834] [robot_state_publisher]: got segment Camera_Link
[robot_state_publisher-5] [INFO] [1735287610.348378990] [robot_state_publisher]: got segment LWheel_Link
[robot_state_publisher-5] [INFO] [1735287610.348383531] [robot_state_publisher]: got segment RWheel_Link
[robot_state_publisher-5] [INFO] [1735287610.348386795] [robot_state_publisher]: got segment base_link
[joint_state_publisher-4] [INFO] [1735287611.258169340] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
[cmdvel2bl-7] [INFO] [1735287611.264444556] [cmd_vel_scaler]: mode defalt...
```

Then, start the car linear speed calibration program, input in the terminal,

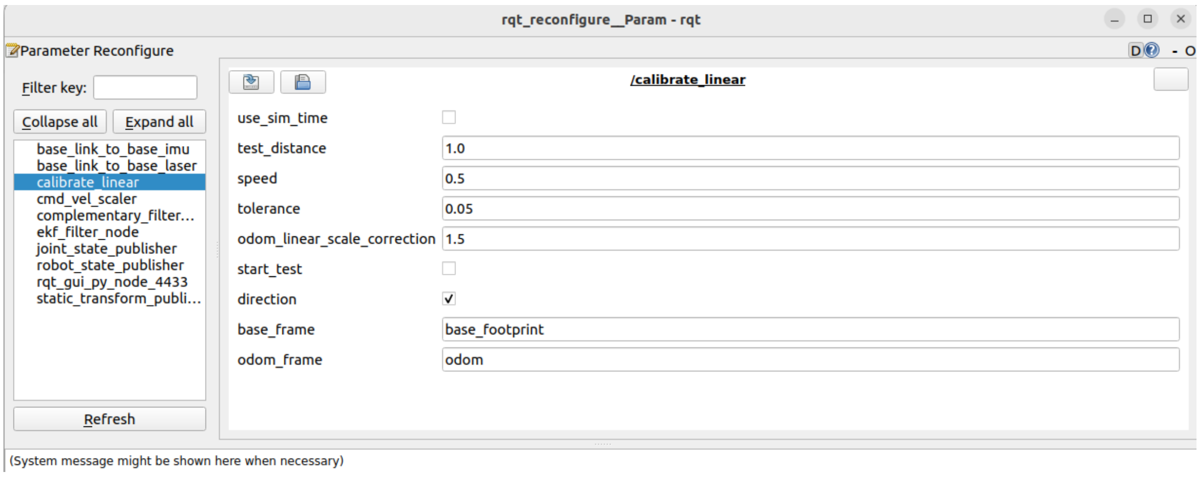
```
ros2 run yahboomcar_bringup calibrate_linear
```



```
yahboom@yahboom-VM:~$ ros2 run yahboomcar_bringup calibrate_linear
finish init work
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
```

Finally, open the dynamic parameter regulator, input in the terminal,

```
ros2 run rqt_reconfigure rqt_reconfigure
```


rqt_reconfigure__Param - rqt
Parameter Reconfigure
Filter key:
Collapse all
Expand all
Refresh
(System message might be shown here when necessary)

```
base_link_to_base_imu
base_link_to_base_laser
calibrate_linear
cmd_vel_scaler
complementary_filter...
ekf_filter_node
joint_state_publisher
robot_state_publisher
rqt_gui_py_node_4433
static_transform_publi...
```

/calibrate_linear

```
use_sim_time                     [ ]
test_distance                    1.0
speed                            0.5
tolerance                        0.05
odom_linear_scale_correction     1.5
start_test                       [ ]
direction                        [✓]
base_frame                       base_footprint
odom_frame                       odom
```

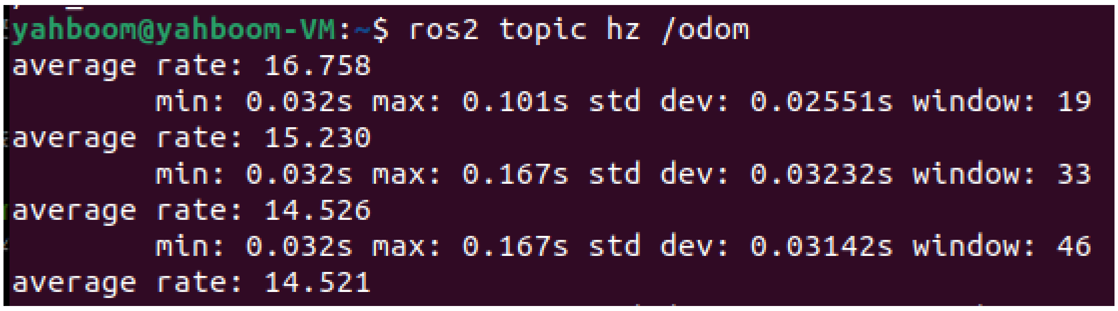
Note: The above nodes may not be available when you first open it. Click Refresh to see all the nodes. The **calibrate_linear** node displayed is the node for calibrating linear speed.

If conversion or other errors occur, this is due to network delay. The coordinate conversion of odom is not successful. You need to restart the car and connect to the agent again.

```
[INFO] [1735358962.891464389] [YahboomCarPatrol]: transform not ready
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))

publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0))
```

When the network is normal, the frequency of the odometer



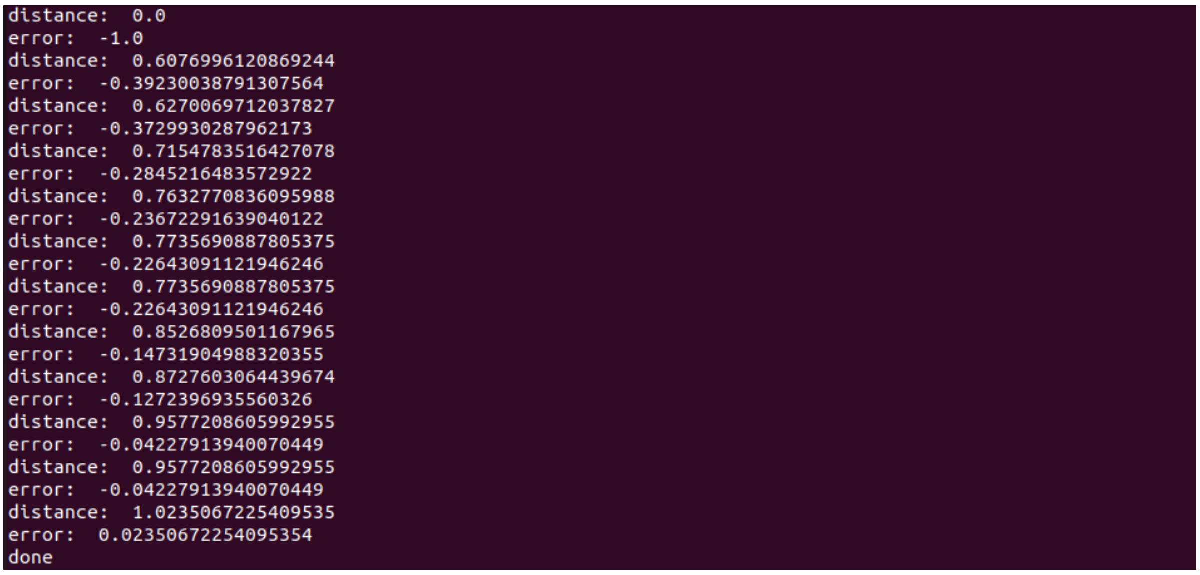
```
yahboom@yahboom-VM:~$ ros2 topic hz /odom
average rate: 16.758
        min: 0.032s max: 0.101s std dev: 0.02551s window: 19
average rate: 15.230
        min: 0.032s max: 0.167s std dev: 0.03232s window: 33
average rate: 14.526
        min: 0.032s max: 0.167s std dev: 0.03142s window: 46
average rate: 14.521
```

4. Start calibration

In the rqt_reconfigure interface, select the calibrate_linear node. There is a **start_test** below. Click the box on the right to start calibration. The other parameters of the rqt interface are as follows:

- test_distance: the distance of the calibration test. Here, the test is to walk forward 1 meter;
- speed: the linear speed;
- tolerance: the tolerance of the error;
- odom_linear_scale_correction: linear velocity proportional coefficient. If the test result is not ideal, modify this value;
- start_test: test switch;
- direction: can be ignored. This value is used for the McLen structure car. After modification, the linear velocity of left and right movement can be calibrated;
- base_frame: the name of the base coordinate system;
- odom_frame: the name of the odometer coordinate system.

Click start_test to start calibration. The car will listen to the TF transformation of base_footprint and odom, calculate the theoretical distance traveled by the car, and issue a stop command when the error is less than tolerance.


```
distance:  0.0
error:  -1.0
distance:  0.6076996120869244
error:  -0.39230038791307564
distance:  0.6270069712037827
error:  -0.3729930287962173
distance:  0.7154783516427078
error:  -0.2845216483572922
distance:  0.7632770836095988
error:  -0.23672291639040122
distance:  0.7735690887805375
error:  -0.22643091121946246
distance:  0.7735690887805375
error:  -0.22643091121946246
distance:  0.8526809501167965
error:  -0.14731904988320355
distance:  0.8727603064439674
error:  -0.1272396935560326
distance:  0.9577208605992955
error:  -0.04227913940070449
distance:  0.9577208605992955
error:  -0.04227913940070449
distance:  1.0235067225409535
error:  0.02350672254095354
done
```

If the actual distance traveled by the car is not 1m, then modify the odom_linear_scale_correction parameter in rqt. After modification, click on the blank space, click start_test again, reset start_test, and then click start_test again to calibrate. The same is true for modifying other parameters. You need to click the blank space below to write the modified parameters. You can also record the parameters, modify the source code directly, and recompile and update the environment.

5. Code analysis

Source code reference path (taking the supporting virtual machine as an example):

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

calibrate_linear.py，核心代码如下，

```
#Monitor TF transformation of base_footprint and odom
def get_position(self):
     try:
        now = rclpy.time.Time()
        trans = 
self.tf_buffer.lookup_transform(self.odom_frame,self.base_frame,now)
        #print("trans: ",trans)
        return trans
     except (LookupException, ConnectivityException, ExtrapolationException):
        self.get_logger().info('transform not ready')
        raise
        return
#Calculate the straight distance and error
def on_timer(self):
    move_cmd = Twist()
    #self.get_param()
    self.start_test = 
self.get_parameter('start_test').get_parameter_value().bool_value
    self.odom_linear_scale_correction = 
self.get_parameter('odom_linear_scale_correction').get_parameter_value().double_v
alue
    self.direction = 
self.get_parameter('direction').get_parameter_value().bool_value
```

```
    self.test_distance = 
self.get_parameter('test_distance').get_parameter_value().double_value
    self.tolerance = 
self.get_parameter('tolerance').get_parameter_value().double_value
    self.speed = self.get_parameter('speed').get_parameter_value().double_value
    self.speed = self.speed * 45
    if self.speed >45:self.speed = 45
    if self.start_test:
        self.position.x = self.get_position().transform.translation.x
        self.position.y = self.get_position().transform.translation.y
        distance = sqrt(pow((self.position.x - self.x_start), 2) +
                        pow((self.position.y - self.y_start), 2))
        distance *= self.odom_linear_scale_correction
        print("distance: ",distance)
        error = distance - self.test_distance
        print("error: ",error)
```