

Linear speed calibration

Note: The ROS_DOMAIN_ID of the Raspberry Pi and the microROS control board need to be consistent. You can check **[MicroROS Control Board Parameter Configuration]** to set the microROS control board ROS_DOMAIN_ID. Check the tutorial **[Connect MicroROS Agent]** to determine whether the IDs are consistent.

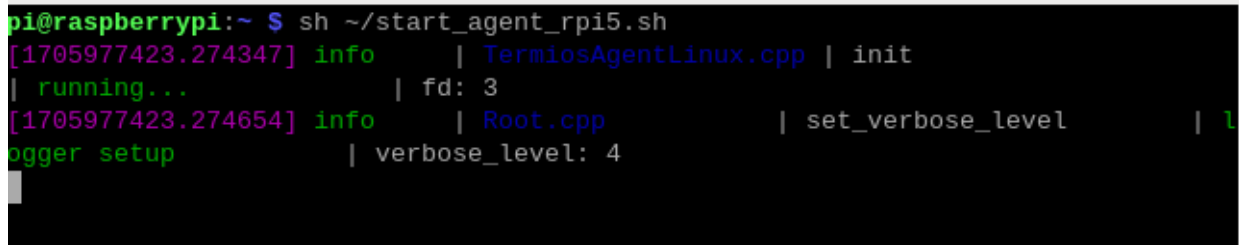
1. Program function description

The car connects to the agent, runs the program, and adjusts the parameters here through the dynamic parameter regulator to calibrate the linear speed of the car. The intuitive reflection of the calibrated linear speed is to give the car an instruction to go straight forward for 1 meter to see how far it actually ran and whether it is within the error range.

2. Start and connect to the agent

After successfully starting the Raspberry Pi, enter the following command in the terminal to start the agent,

```
sh ~/start_agent_rpi5.sh
```



```
pi@raspberrypi:~ $ sh ~/start_agent_rpi5.sh
[1705977423.274347] info      | TermiosAgentLinux.cpp | init
| running...              | fd: 3
[1705977423.274654] info      | Root.cpp               | set_verbose_level      | 1
logger setup              | verbose_level: 4
```

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

```

subscriber created      | client_key: 0x57E5DE1D, subscriber_id: 0x001(4), partici
pant_id: 0x000(1)
[1705977460.524081] info    | ProxyClient.cpp      | create_datareader      | d
atareader created      | client_key: 0x57E5DE1D, datareader_id: 0x001(6), subscri
ber_id: 0x001(4)
[1705977460.530288] info    | ProxyClient.cpp      | create_topic           | t
opic created           | client_key: 0x57E5DE1D, topic_id: 0x005(2), participant_
id: 0x000(1)
[1705977460.533849] info    | ProxyClient.cpp      | create_subscriber      | s
ubscriber created      | client_key: 0x57E5DE1D, subscriber_id: 0x002(4), partici
pant_id: 0x000(1)
[1705977460.540811] info    | ProxyClient.cpp      | create_datareader      | d
atareader created      | client_key: 0x57E5DE1D, datareader_id: 0x002(6), subscri
ber_id: 0x002(4)
[1705977460.548331] info    | ProxyClient.cpp      | create_topic           | t
opic created           | client_key: 0x57E5DE1D, topic_id: 0x006(2), participant_
id: 0x000(1)
[1705977460.553389] info    | ProxyClient.cpp      | create_subscriber      | s
ubscriber created      | client_key: 0x57E5DE1D, subscriber_id: 0x003(4), partici
pant_id: 0x000(1)
[1705977460.556688] info    | ProxyClient.cpp      | create_datareader      | d
atareader created      | client_key: 0x57E5DE1D, datareader_id: 0x003(6), subscri
ber_id: 0x003(4)

```

3. Enter the car docker

Open another terminal and enter the following command to enter docker.

```
sh ros2_humble.sh
```

When the following interface appears, you have successfully entered docker. Now you can control the car through commands.

```

pi@raspberrypi:~ $ ./ros2_humble.sh
access control disabled, clients can connect from any host
Successful
MY_DOMAIN_ID: 20
root@raspberrypi:/#

```

4. Start the program

First, start the car's underlying data processing program. This program will release the TF transformation of odom->base_footprint. With this TF change, you can calculate "how far the car has gone" and input it at the terminal.

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

```

-----robot_type = x3-----
[INFO] [imu_filter_madgwick_node-1]: process started with pid [467]
[INFO] [ekf_node-2]: process started with pid [469]
[INFO] [static_transform_publisher-3]: process started with pid [471]
[INFO] [joint_state_publisher-4]: process started with pid [473]
[INFO] [robot_state_publisher-5]: process started with pid [475]
[INFO] [static_transform_publisher-6]: process started with pid [477]
[static_transform_publisher-6] [WARN] [1705983153.436730358] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [WARN] [1705983153.440907069] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-6] [INFO] [1705983153.493758940] [static_transform_publisher_BSLtkER9Heqn3Vhd]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.050000')
[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] [1705983153.493758791] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('-0.002999', '-0.003000', '0.031701')
[static_transform_publisher-3] rotation: ('0.000000', '0.000000', '0.000000', '1.000000')
[static_transform_publisher-3] from 'base_link' to 'imu_frame'
[robot_state_publisher-5] [WARN] [1705983153.549719441] [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] [1705983153.550449464] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1705983153.550870115] [robot_state_publisher]: got segment imu_Link
[robot_state_publisher-5] [INFO] [1705983153.551059023] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1705983153.551073171] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1705983153.551084338] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1705983153.551111375] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1705983153.551122042] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1705983153.551132209] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1705983153.551142320] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1705983153.923360093] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1705983153.925493903] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1705983153.925539088] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1705983153.926906838] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1705983153.926966838] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1705983153.926988672] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[joint_state_publisher-4] [INFO] [1705983154.028776814] [joint_state_publisher]: Waiting for robot description to be published on the robot_description topic...
[imu_filter_madgwick_node-1] [INFO] [1705983156.027033468] [imu_filter]: First IMU message received.

```

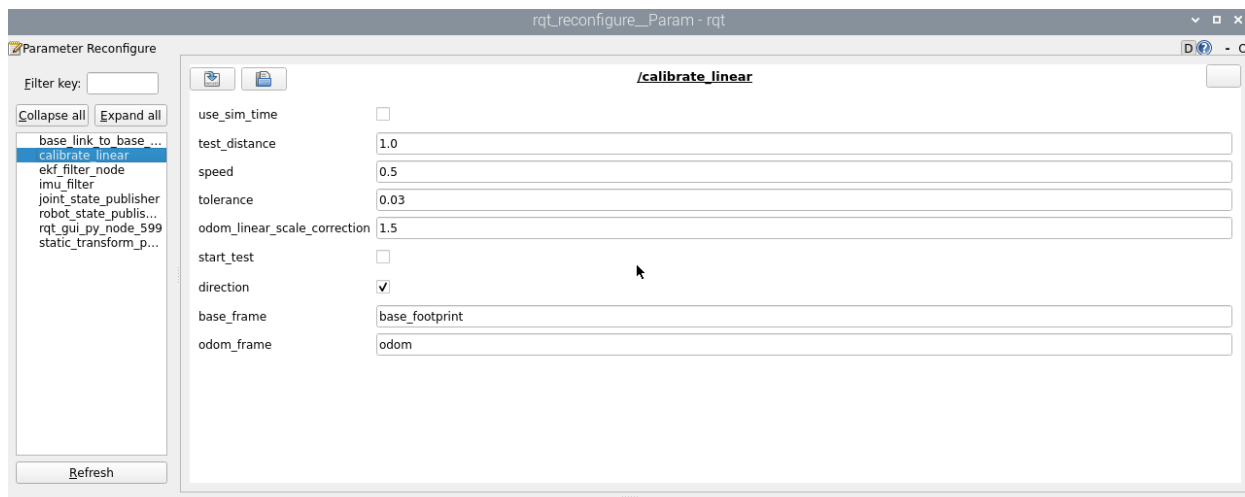
Then start the car linear speed calibration program and enter the following command on the terminal.

```
ros2 run yahboomcar_bringup calibrate_linear
```

```
root@raspberrypi:~# 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!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
Please change the state!
```

Finally, enter the following command in the terminal to open the dynamic parameter adjuster.

```
ros2 run rqt_reconfigure rqt_reconfigure
```



(System message might be shown here when necessary)

Note: There may not be the above nodes when you first open it. You can see all the nodes after clicking Refresh. The displayed **calibrate_linear** node is the node for calibrating linear speed.

5. Start calibration

In the rqt_reconfigure interface, select the calibrate_linear node. There is **start_test** below and click the box to the right to start calibration. Other parameters in the rqt interface are explained as follows:

- test_distance: Calibrate the test distance, here the test goes forward 1 meter;
- speed: Linear speed;
- tolerance: Allowable error tolerance;
- odom_linear_scale_correction: Linear speed proportional coefficient. If the test result is not ideal, just modify this value.;
- start_test: test switch;
- direction: It can be ignored. This value is used for the wheat wheel structure car. After modification, the linear speed 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 monitor the TF transformation of base_footprint and odom, calculate the theoretical distance traveled by the car, wait until the error is less than the tolerance, and issue a parking instruction.

```

distance: 0.0
error: -1.0
distance: 0.0
error: -1.0
distance: 0.0
error: -1.0
distance: 0.0
error: -1.0
distance: 0.8126761633541635
error: -0.18732383664583652
distance: 0.8126761633541635
error: -0.18732383664583652
distance: 0.8126761633541635
error: -0.18732383664583652
distance: 0.8126761633541635
error: -0.18732383664583652
distance: 0.9451114396499452
error: -0.05488856035005485
distance: 0.9451114396499452
error: -0.05488856035005485
distance: 1.0437490477028715
error: 0.04374904770287147
distance: 1.0437490477028715
error: 0.04374904770287147
distance: 1.0437490477028715
error: 0.04374904770287147
distance: 1.0437490477028715
error: 0.04374904770287147
distance: 1.1698576521830526
error: 0.1698576521830526
distance: 1.1698576521830526
error: 0.1698576521830526
distance: 1.1698576521830526
error: 0.1698576521830526
distance: 1.3016118703796185
error: 0.30161187037961845
distance: 1.3016118703796185
error: 0.30161187037961845
distance: 1.3833684675517492
error: 0.38336846755174925
distance: 1.3833684675517492
error: 0.38336846755174925
distance: 1.3833684675517492
error: 0.38336846755174925
distance: 1.495507356756789
error: 0.4955073567567889
distance: 1.495507356756789
error: 0.4955073567567889
distance: 1.5819954799687612
error: 0.5819954799687612
distance: 1.5819954799687612
error: 0.5819954799687612
distance: 1.70041785808715
error: 0.7004178580871501
distance: 1.70041785808715
error: 0.7004178580871501
distance: 1.791632660393782
error: 0.791632660393782

```

If the actual distance the car runs 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 goes for modifying other parameters. You need to click on the blank space to write the modified parameters.

6. Code analysis

Source code reference path.

```
/root/yahboomcar_ws/src/yahboomcar_bringup/yahboomcar_bringup
```

calibrate_linear.py, The core code is as follows,

```
#Monitor the 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

#Get the current coordinates
self.position.x = self.get_position().transform.translation.x
self.position.y = self.get_position().transform.translation.y
#Calculate how far distance and error are
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
```