Angular velocity calibration

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

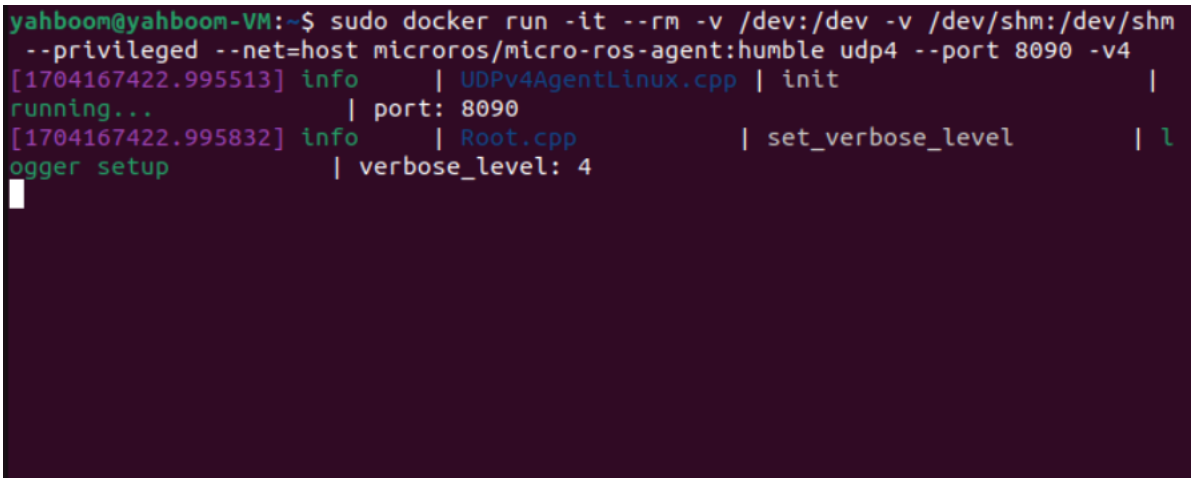
1、Program function description

The car connects to the agent, runs the program, and adjusts the parameters here through the dynamic parameter adjuster to calibrate the car's angular velocity. The intuitive reflection of the calibrated angular speed is to give the car a command to rotate 360 degrees (one revolution) to see how many degrees it actually rotates and whether it is within the error range.

2、Start and connect to the agent

Taking 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 8090 -v4
```



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.

```
[1702630014.015846] info     | ProxyClient.cpp    | create_participant       | participant created    | client_key: 0x0B62A009, part
icipant_id: 0x000(1)
[1702630014.135363] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0B62A009, topi
c_id: 0x000(2), participant_id: 0x000(1)
[1702630014.223689] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0B62A009, publ
isher_id: 0x000(3), participant_id: 0x000(1)
[1702630014.415510] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0B62A009, data
writer_id: 0x000(5), publisher_id: 0x000(3)
[1702630014.428530] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0B62A009, topi
c_id: 0x001(2), participant_id: 0x000(1)
[1702630014.527190] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0B62A009, publ
isher_id: 0x001(3), participant_id: 0x000(1)
[1702630014.543889] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0B62A009, data
writer_id: 0x001(5), publisher_id: 0x001(3)
[1702630014.554490] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0B62A009, topi
c_id: 0x002(2), participant_id: 0x000(1)
[1702630014.737059] info     | ProxyClient.cpp    | create_publisher         | publisher created      | client_key: 0x0B62A009, publ
isher_id: 0x002(3), participant_id: 0x000(1)
[1702630014.755072] info     | ProxyClient.cpp    | create_datawriter        | datawriter created     | client_key: 0x0B62A009, data
writer_id: 0x002(5), publisher_id: 0x002(3)
[1702630014.818985] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0B62A009, topi
c_id: 0x003(2), participant_id: 0x000(1)
[1702630014.840001] info     | ProxyClient.cpp    | create_subscriber        | subscriber created     | client_key: 0x0B62A009, subs
criber_id: 0x000(4), participant_id: 0x000(1)
[1702630014.864010] info     | ProxyClient.cpp    | create_datareader        | datareader created     | client_key: 0x0B62A009, data
reader_id: 0x000(6), subscriber_id: 0x000(4)
[1702630014.959908] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0B62A009, topi
c_id: 0x004(2), participant_id: 0x000(1)
[1702630015.033537] info     | ProxyClient.cpp    | create_subscriber        | subscriber created     | client_key: 0x0B62A009, subs
criber_id: 0x001(4), participant_id: 0x000(1)
[1702630015.140350] info     | ProxyClient.cpp    | create_datareader        | datareader created     | client_key: 0x0B62A009, data
reader_id: 0x001(6), subscriber_id: 0x001(4)
[1702630015.158510] info     | ProxyClient.cpp    | create_topic             | topic created          | client_key: 0x0B62A009, topi
c_id: 0x005(2), participant_id: 0x000(1)
[1702630015.241039] info     | ProxyClient.cpp    | create_subscriber        | subscriber created     | client_key: 0x0B62A009, subs
criber_id: 0x002(4), participant_id: 0x000(1)
[1702630015.347393] info     | ProxyClient.cpp    | create_datareader        | datareader created     | client_key: 0x0B62A009, data
reader_id: 0x002(6), subscriber_id: 0x002(4)
```

3、starting 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 many degrees the car has turned" and and enter the following command in the terminal

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

```
[INFO] [imu_filter_madgwick_node-1]: process started with pid [6638]
[INFO] [ekf_node-2]: process started with pid [6640]
[INFO] [static_transform_publisher-3]: process started with pid [6642]
[INFO] [joint_state_publisher-4]: process started with pid [6644]
[INFO] [robot_state_publisher-5]: process started with pid [6646]
[INFO] [static_transform_publisher-6]: process started with pid [6658]
[static_transform_publisher-3] [WARN] [1702865272.944043208] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-6] [WARN] [1702865272.984740987] []: Old-style arguments are deprecated; see --help for new-style argume
nts
[static_transform_publisher-3] [INFO] [1702865272.991057276] [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'
[static_transform_publisher-6] [INFO] [1702865273.005707993] [static_transform_publisher_JH06Gexf4GRodmgs]: 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'
[robot_state_publisher-5] [WARN] [1702865273.013202438] [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] [1702865273.013312806] [robot_state_publisher]: got segment base_link
[robot_state_publisher-5] [INFO] [1702865273.013516195] [robot_state_publisher]: got segment imu_Link
[robot_state_publisher-5] [INFO] [1702865273.013524175] [robot_state_publisher]: got segment jq1_Link
[robot_state_publisher-5] [INFO] [1702865273.013528144] [robot_state_publisher]: got segment jq2_Link
[robot_state_publisher-5] [INFO] [1702865273.013531665] [robot_state_publisher]: got segment radar_Link
[robot_state_publisher-5] [INFO] [1702865273.013535185] [robot_state_publisher]: got segment yh_Link
[robot_state_publisher-5] [INFO] [1702865273.013538763] [robot_state_publisher]: got segment yq_Link
[robot_state_publisher-5] [INFO] [1702865273.013542135] [robot_state_publisher]: got segment zh_Link
[robot_state_publisher-5] [INFO] [1702865273.013545612] [robot_state_publisher]: got segment zq_Link
[imu_filter_madgwick_node-1] [INFO] [1702865273.030399479] [imu_filter]: Starting ImuFilter
[imu_filter_madgwick_node-1] [INFO] [1702865273.031826501] [imu_filter]: Using dt computed from message headers
[imu_filter_madgwick_node-1] [INFO] [1702865273.031858361] [imu_filter]: The gravity vector is kept in the IMU message.
[imu_filter_madgwick_node-1] [INFO] [1702865273.032488302] [imu_filter]: Imu filter gain set to 0.100000
[imu_filter_madgwick_node-1] [INFO] [1702865273.032525566] [imu_filter]: Gyro drift bias set to 0.000000
[imu_filter_madgwick_node-1] [INFO] [1702865273.032531441] [imu_filter]: Magnetometer bias values: 0.000000 0.000000 0.000000
[imu_filter_madgwick_node-1] [INFO] [1702865273.053298796] [imu_filter]: First IMU message received.
[joint_state_publisher-4] [INFO] [1702865273.282975810] [joint_state_publisher]: Waiting for robot_description to be published on th
e robot_description topic...
```

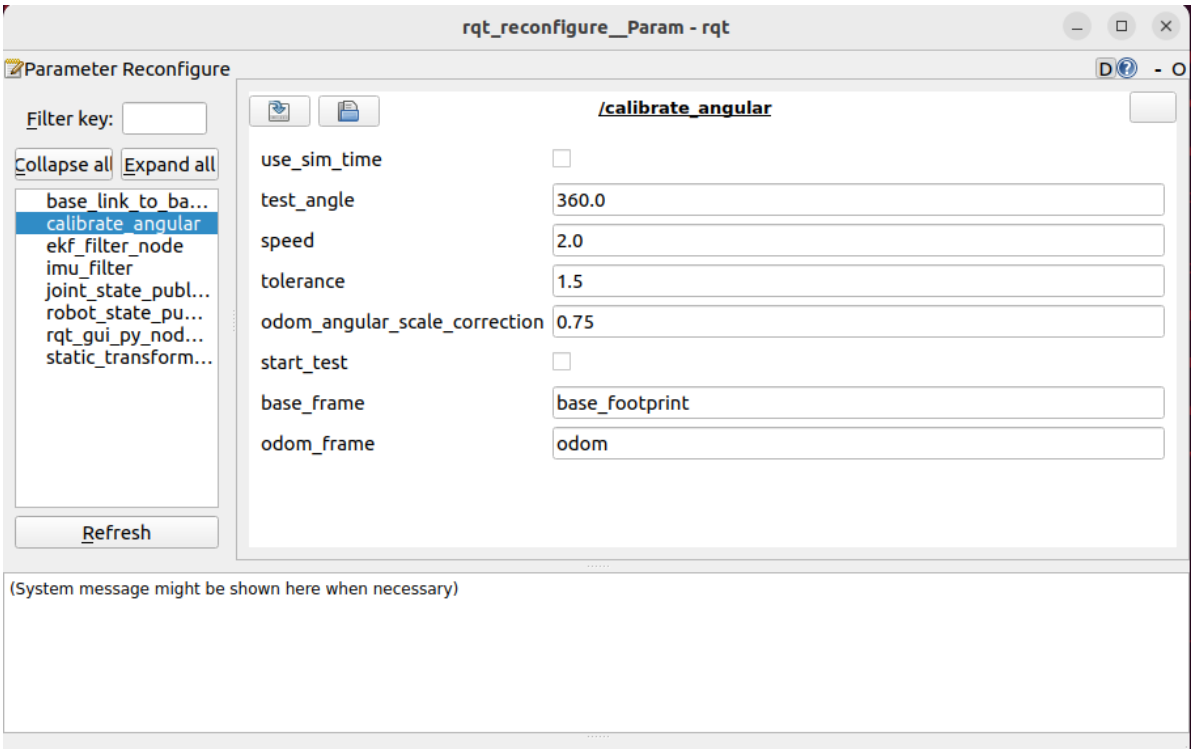
Then, start the car angular velocity calibration program and enter the following command in the terminal

```
ros2 run yahboomcar_bringup calibrate_angular
```

```
yahboom@yahboom-VM:~/yahboomcar_ws$ ros2 run yahboomcar_bringup calibrate_angular
finish init work
```

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

```
ros2 run rqt_reconfigure rqt_reconfigure
```



Note: There may not be the above nodes when you first open it. You can see all nodes after clicking Refresh. The **calibrate_angular** node shown is the node for calibrating angular velocity.

4、Start calibration

In the rqt_reconfigure interface, select the calibrate_angular node. There is **start_test** below. Click the box to the right of it to start calibration. Other parameters in the rqt interface are described as follows.

- test_angle：Calibrate the test angle, here the test rotates 360 degrees
- speed：Angular velocity magnitude;
- tolerance：tolerance for error;
- odom_angular_scale_correction：Linear speed proportional coefficient, if the test result is not ideal, just modify this value;
- start_test：test switch;
- 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.

```
turn_angle:  4.494604809173198
error:  1.7885804980063886
turn_angle:  4.494604809173198
error:  1.7885804980063886
turn_angle:  4.771883271266408
error:  1.511302035913178
turn_angle:  4.771883271266408
error:  1.511302035913178
turn_angle:  4.771883271266408
error:  1.511302035913178
turn_angle:  5.065386774659367
error:  1.2177985325202192
done
```

The turn_angle here is in radians. If the actual angle of the car's rotation is not 360 degrees, then modify the odom_angular_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. Calibration. The same goes for modifying other parameters. You need to click on the blank space to write the modified parameters.

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_angular.py, the core code is as follows,

```
#Monitor the TF transformation of base_footprint and odom
def get_odom_angle(self):
    try:
        now = rclpy.time.Time()
        rot =
self.tf_buffer.lookup_transform(self.odom_frame,self.base_frame,now)
        #print("oring_rot: ",rot.transform.rotation)
        cacl_rot = PyKDL.Rotation.Quaternion(rot.transform.rotation.x,
rot.transform.rotation.y, rot.transform.rotation.z, rot.transform.rotation.w)
        #print("cacl_rot: ",cacl_rot)
        angle_rot = cacl_rot.GetRPY()[2]
        #print("angle_rot: ",angle_rot)
        return angle_rot

    except (LookupException, ConnectivityException, ExtrapolationException):
        self.get_logger().info('transform not ready')
        raise
        return
#Calculate the angle and error of rotation
self.odom_angle = self.get_odom_angle()
self.delta_angle = self.odom_angular_scale_correction *
self.normalize_angle(self.odom_angle - self.first_angle)
#print("delta_angle: ",self.delta_angle)
self.turn_angle += self.delta_angle
print("turn_angle: ",self.turn_angle)
self.error = self.test_angle - self.turn_angle
```