

# 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 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 angular velocity of the car through the dynamic parameter regulator. The intuitive manifestation of angular velocity calibration is to let the car rotate 360 degrees (one circle) to see how many degrees it actually rotates 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 matching 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
logger 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
[1735179211.772581] info      | Root.cpp             | set_verbose_level
[1735179325.739277] info      | Root.cpp             | create_client
ion_id: 0x81
[1735179325.739348] info      | SessionManager.hpp   | establish_session
ess: 192.168.2.102:49954
[1735179325.971694] info      | ProxyClient.cpp      | create_participant
participant_id: 0x000(1)
[1735179326.046043] info      | ProxyClient.cpp      | create_topic
c_id: 0x000(2), participant_id: 0x000(1)
[1735179326.159287] info      | ProxyClient.cpp      | create_publisher
isher_id: 0x000(3), participant_id: 0x000(1)
[1735179326.176344] info      | ProxyClient.cpp      | create_datawriter
writer_id: 0x000(5), publisher_id: 0x000(3)
[1735179326.184566] info      | ProxyClient.cpp      | create_topic
c_id: 0x001(2), participant_id: 0x000(1)
[1735179326.263761] info      | ProxyClient.cpp      | create_publisher
isher_id: 0x001(3), participant_id: 0x000(1)
[1735179326.276817] info      | ProxyClient.cpp      | create_datawriter
writer_id: 0x001(5), publisher_id: 0x001(3)
[1735179326.285996] info      | ProxyClient.cpp      | create_topic
c_id: 0x002(2), participant_id: 0x000(1)
[1735179326.345401] info      | ProxyClient.cpp      | create_publisher
isher_id: 0x002(3), participant_id: 0x000(1)
[1735179326.365619] info      | ProxyClient.cpp      | create_datawriter
writer_id: 0x002(5), publisher_id: 0x002(3)
[1735179326.372863] info      | ProxyClient.cpp      | create_topic
c_id: 0x003(2), participant_id: 0x000(1)
[1735179326.379913] info      | ProxyClient.cpp      | create_publisher
isher_id: 0x003(3), participant_id: 0x000(1)
[1735179326.448851] info      | ProxyClient.cpp      | create_datawriter
writer_id: 0x003(5), publisher_id: 0x003(3)
[1735179326.548363] info      | ProxyClient.cpp      | create_topic
c_id: 0x004(2), participant_id: 0x000(1)
[1735179326.565153] info      | ProxyClient.cpp      | create_subscriber
criber_id: 0x000(4), participant_id: 0x000(1)
[1735179326.574254] info      | ProxyClient.cpp      | create_datareader
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 many degrees the car has turned".

```
yahboom@yahboom-VM: $ ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
[INFO] [launch]: All log files can be found below /home/yahboom/.ros/log/2024-12-27-16-20-09-706601-yahboom-VM-5819
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] -----robot_type = stm32v2-----
[INFO] [complementary_filter_node-1]: process started with pid [5821]
[INFO] [static_transform_publisher-2]: process started with pid [5822]
[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] [cmdvel2b1-7]: process started with pid [5835]
[INFO] [ekf_node-8]: process started with pid [5848]
[static_transform_publisher-2] [WARN] [1735287610.832940353] []: Old-style arguments are deprecated; see --help for new-style arguments
[static_transform_publisher-3] [WARN] [1735287610.833194674] []: 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_B4wJrL6rhGwZqaw]: Spinning until stopped - publishing transform
[static_transform_publisher-6] translation: ('0.000000', '0.000000', '0.033580')
[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] [1735287610.183102608] [base_link_to_base_laser]: Spinning until stopped - publishing transform
[static_transform_publisher-3] translation: ('0.000000', '0.000000', '0.120000')
[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] [1735287610.209028170] [base_link_to_base_imu]: Spinning until stopped - publishing transform
[static_transform_publisher-2] translation: ('0.000000', '0.010325', '0.000091')
[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] [1735287610.348378996] [robot_state_publisher]: got segment LiWheel_Link
[robot_state_publisher-5] [INFO] [1735287610.348383531] [robot_state_publisher]: got segment RiWheel_Link
[robot_state_publisher-5] [INFO] [1735287610.348386795] [robot_state_publisher]: got segment base_link
[joint_state_publisher-4] [INFO] [1735287611.258109340] [joint_state_publisher]: Waiting for robot_description to be published on the robot_description topic...
[cmdvel2b1-7] [INFO] [1735287611.264444556] [cmd_vel_scaler]: mode default...
```

Then, start the car angular velocity calibration program, enter 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 regulator, enter in the terminal,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



Note: The above nodes may not appear when you first open the application. Click Refresh to see all nodes. The **calibrate\_angular** node is the node for calibrating angular velocity.

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))
```

## 4. Start calibration

In the rqt\_reconfigure interface, select the calibrate\_angular 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\_angle: the angle of the calibration test, here the test rotates 360 degrees;
- speed: angular velocity;
- tolerance: tolerance of error;
- odom\_angular\_scale\_correction: linear velocity scale coefficient. If the test result is not ideal, 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, and issue a stop command when the error is less than tolerance.

```
error: 1.7885804980063886
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 rotation angle of the car 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 to calibrate. Modifying other parameters is the same. You need to click on the blank space 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\_angular.py, the core code is as follows,

```
#Monitor 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)
        cac1_rot = PyKDL.Rotation.Quaternion(rot.transform.rotation.x,
rot.transform.rotation.y, rot.transform.rotation.z, rot.transform.rotation.w)
        angle_rot = cac1_rot.GetRPY()[2]
        return angle_rot

#Calculate the rotation angle and error
def on_timer(self):
    self.start_test =
self.get_parameter('start_test').get_parameter_value().bool_value
    self.odom_angular_scale_correction =
self.get_parameter('odom_angular_scale_correction').get_parameter_value().double_
value
    self.test_angle =
self.get_parameter('test_angle').get_parameter_value().double_value
    self.test_angle = radians(self.test_angle)
    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 * 100
    move_cmd = Twist()
    self.test_angle *= self.reverse
    self.error = self.test_angle - self.turn_angle
```