7 Robot calibration

Note: The parameters have been calibrated before the product leaves the factory, and generally do not need to be calibrated. If you feel that the control robot has a large deviation, you need to calibrate [imu], [linear velocity], and [angular velocity]; when calibrating, place the robot in advance and do not move the robot.

It should be noted that this lesson is applicable to the ROSMASTER X3 and X3Plus for the Mecanum wheel models. X1 is an ordinary four-wheel drive model, and R2 is an Ackerman model. Due to the difference in the principle of motion direction control, this lesson is not applicable.

7.1 imu school associate

ROSMASTER has already calibrated the imu before leaving the factory, and the user does not need to calibrate. The following tutorial refers to the procedure that should be started if the imu needs to be calibrated.

7.1.1 Calibration steps

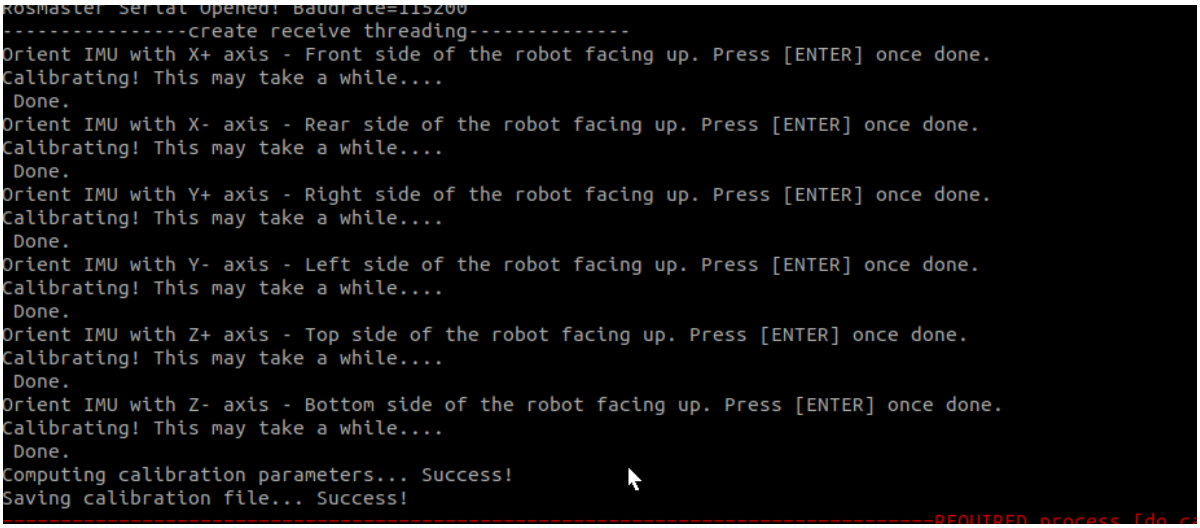
Note: When calibrating, make sure the robot is still.

1. start

```
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to Jetson Orin-
Docker/05, Enter the robot's docker container
~/run_docker.sh
#Multiple ros commands require multiple terminals to enter the same docker
container for execution, please refer to Jetson Orin-Docker/05, Section 5.8
tutorial
```

```
roslaunch yahboomcar_bringup calibrate_imu.launch
```

As shown in the figure below, press Enter to calibrate the data in the X+, X-, Y+, Y-, Z+, Z- directions in turn. After the calibration, it will be automatically saved to the specified folder.



When shipped from the factory, the calibration data is stored in,

```
~/yahboomcar_ws/src/yahboomcar_bringup/param/imu_calib.yaml
```

7.1.2 Use the calibrated imu data

```
roslaunch yahboomcar_bringup bringup_calib.launch
```

The above command is to use the self-calibrated imu data when starting the chassis drive.

7.2 Linear speed calibration

7.2.1 Preparation

1. Measure the distance of 1 meter with a meter ruler and make a mark.
2. put the trolley at the starting point.
3. Modify the parameters **linear_scale_x** and **linear_scale_y** to 1.0.

7.2.2 Start

1. terminal input

```
roslaunch yahboomcar_bringup bringup.launch             # chassis control
```

1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9   "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$
```

2. Now enter the docker container in the newly opened terminal:

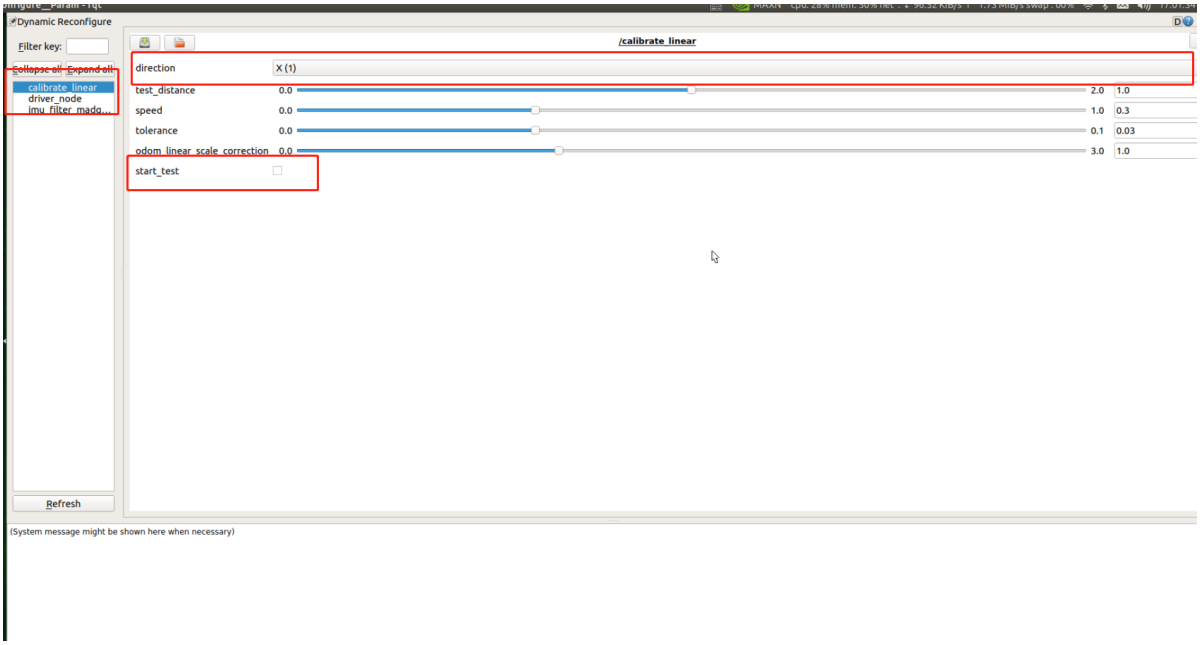
```
docker exec -it 5b698ea10535 /bin/bash
```

```
jetson@ubuntu:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND       CREATED      STATUS       PORTS     NAMES
5b698ea10535   yahboomtechnology/ros-foxy:3.3.9   "/bin/bash"   3 days ago   Up 9 hours             ecstatic_lewin
jetson@ubuntu:~$ docker exec -it 5b698ea10535 /bin/bash
-------------------------------------------------------
my_robot_type: x3 | my_lidar: a1 | my_camera: astrapro
-------------------------------------------------------
root@ubuntu:/#
```

After successfully entering the container, you can open countless terminals to enter the container.

```
roslaunch yahboomcar_bringup calibrate_linear.launch    # linear speed calibration
rosrun rqt_reconfigure rqt_reconfigure                  # Enable  dynamic parameter adjustment
```

2. Click the square on the right of [start_test] to start moving the distance of [test_distance]. At this time, observe whether the car really moves [test_distance]. If not adjust the parameter [odom_linear_scale_correction], put the car back to the starting point to continue the test.



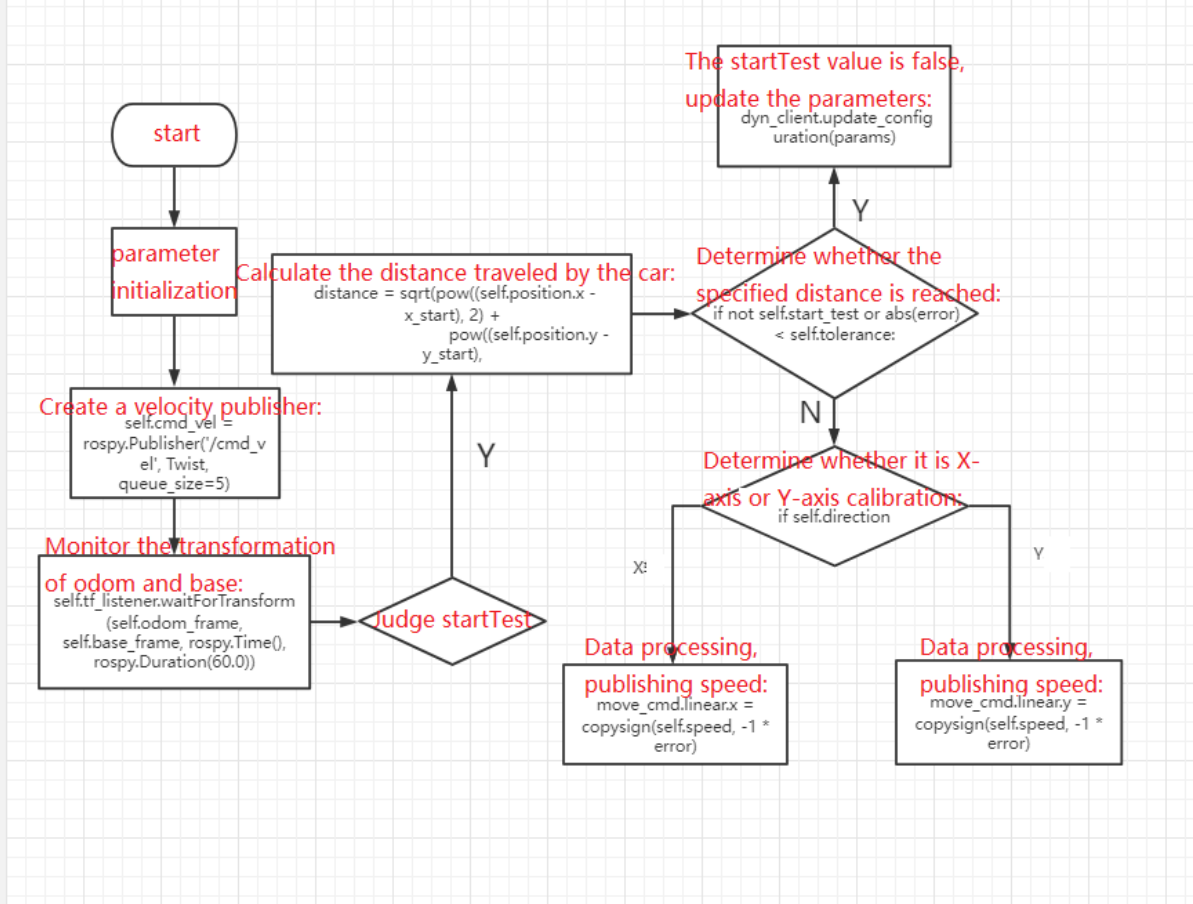
Note: The car defaults to calibrate the linear speed of the X-axis direction at the beginning. If you want to calibrate the Y-axis, click [direction] to switch the direction and then click [start_test] to start calibrating the linear speed of the Y-axis.

- test_distance: test distance. It should not be too large, the default is one meter.
- speed: Test line speed. The higher the speed, the greater the inertia.

- tolerance: The error in reaching the target. If the error is too small, it will shake at the target position, otherwise, the error of reaching the target point will be very large.
- odom_linear_scale_correction: odometer scaling.
- start_test: start the test.
- direction: the direction of the linear velocity, the default is the X axis.

After the test, remember the value of [odom_linear_scale_correction] and modify it to the values of the parameters linear_scale_x and linear_scale_y in bringup.launch.

2. calibrate_linear.py program flow chart



7.3. Angular velocity calibration

7.3.1. Preparation

1. Put the trolley in a position where it is easy to rotate the angle.
2. Modify the parameter **angular_scale** to 1.0.

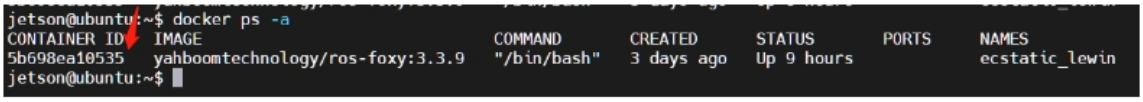
7.3.2 Start

1. terminal input

```
roslaunch yahboomcar_bringup bringup.launch             # chassis control
```

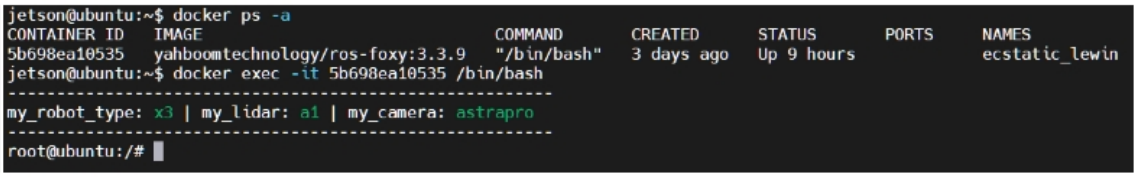
1. In the above steps, a docker container has been opened. You can open another terminal on the host (car) to view:

```
docker ps -a
```



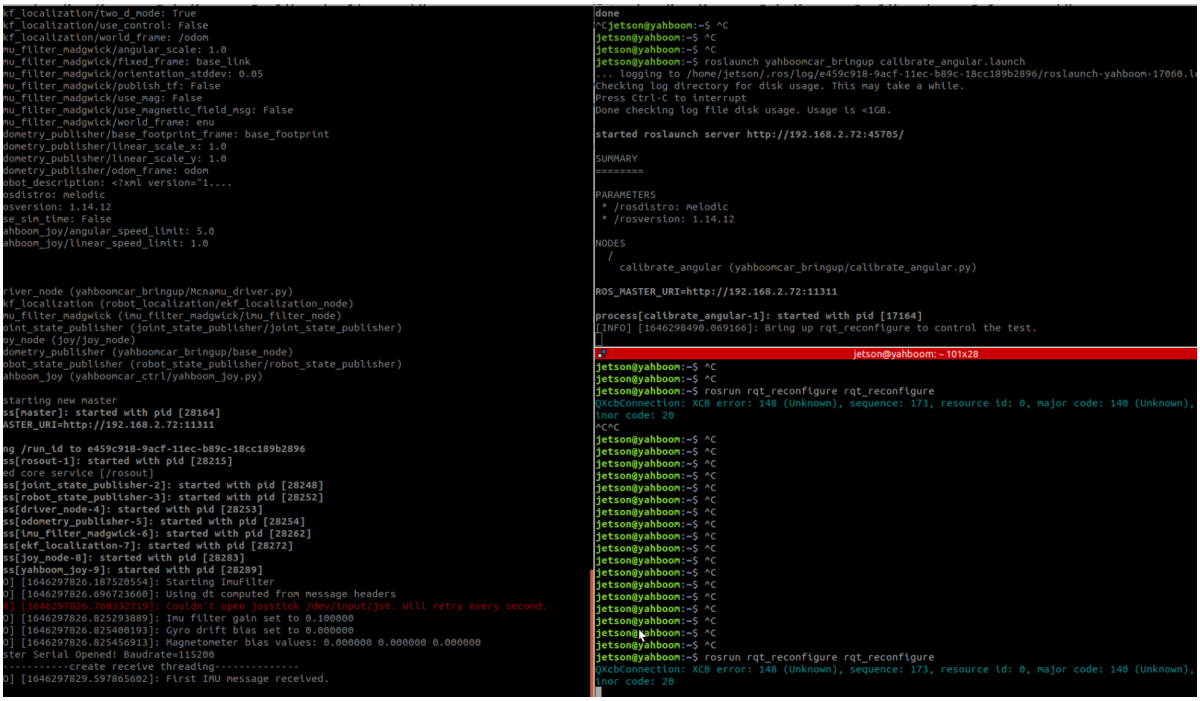
2. Now enter the docker container in the newly opened terminal:

```
docker exec -it 5b698ea10535 /bin/bash
```

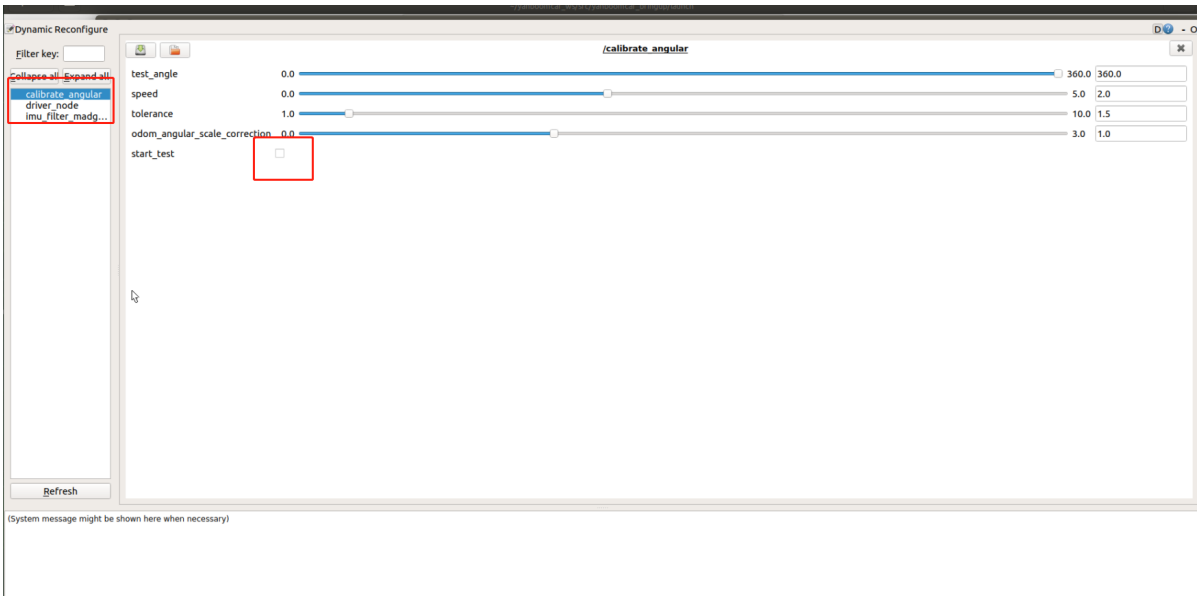


After successfully entering the container, you can open countless terminals to enter the container.

```
roslaunch yahboomcar_bringup calibrate_angular.launch   # angular velocity calibration
rosrun rqt_reconfigure rqt_reconfigure                  # Enable dynamic parameter adjustment
```



2. Click the square on the right side of [start_test] to start moving the distance of [test_angle]. At this time, observe whether the car really turns [test_angle]. If not adjust the parameter [odom_angule_scale_correction], put the car back to the starting point to continue the test.



- test_angle: test distance. It should not be too large, the default is 360°.
- speed: Test angular velocity. The higher the speed, the greater the inertia.
- tolerance: The error in reaching the target. If the error is too small, it will shake at the target position, otherwise, the error of reaching the target point will be very large.
- odom_angule_scale_correction: odometer scaling.
- start_test: start the test.

After the test, remember the value of [odom_angule_scale_correction] and modify it to the value of the parameter [angular_scale] in bringup.launch.

3. calibrate_angular.py program flow chart