4. Robot calibration

Note: The parameters of the product have been calibrated before leaving the factory, so generally no calibration is required. If you feel that the control deviation of the robot is large, you need to calibrate [imu], [linear speed], and [angular speed]; when calibrating, put the robot in place in advance and do not move the robot.

4.1. imu calibration

NAVROBO has already calibrated the imu before leaving the factory, so users do not need to calibrate. The following tutorial refers to the program that should be started if imu calibration is required.

4.1.1. Calibration steps

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

1) Startup

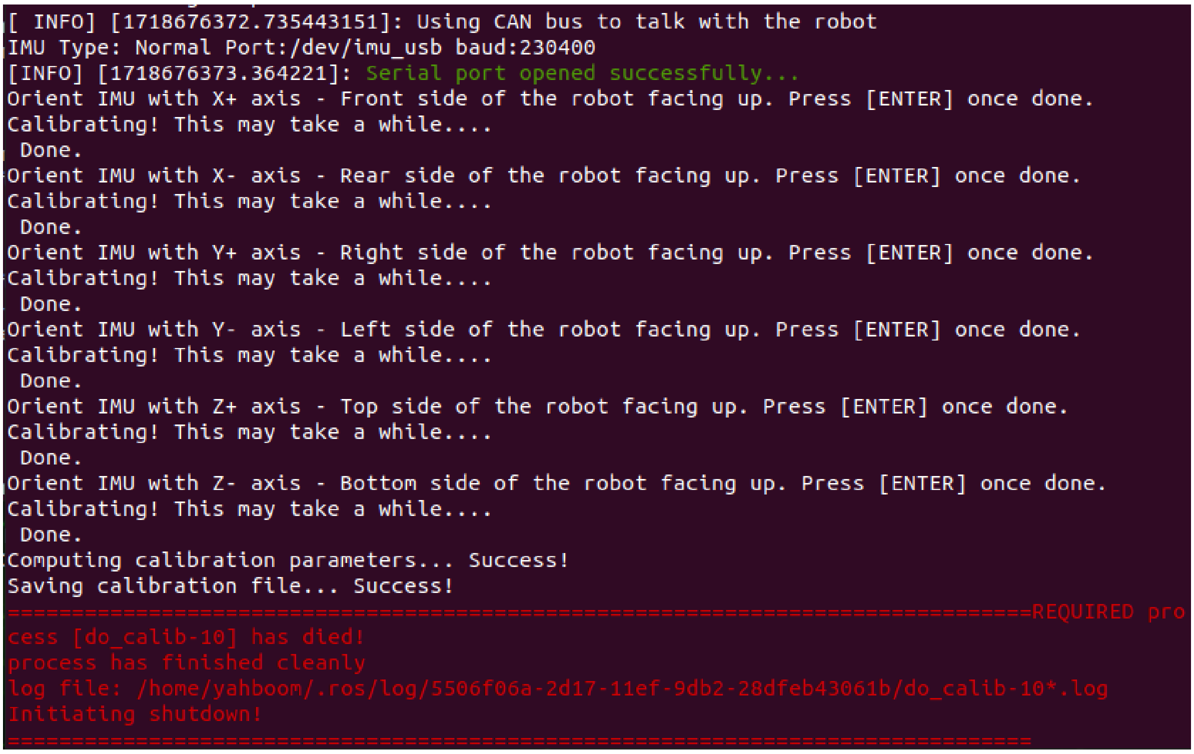
Stop the automatic chassis service

```
sudo supervisorctl stop ChassisServer
```

Startup

```
roslaunch yahboom_navrobo_bringup calibrate_imu.launch
```

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



```
[ INFO] [1718676372.735443151]: Using CAN bus to talk with the robot
IMU Type: Normal Port:/dev/imu_usb baud:230400
[INFO] [1718676373.364221]: Serial port opened successfully...
Orient IMU with X+ axis - Front side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
 Done.
Orient IMU with X- axis - Rear side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
 Done.
Orient IMU with Y+ axis - Right side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
 Done.
Orient IMU with Y- axis - Left side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
 Done.
Orient IMU with Z+ axis - Top side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
 Done.
Orient IMU with Z- axis - Bottom side of the robot facing up. Press [ENTER] once done.
Calibrating! This may take a while....
 Done.
Computing calibration parameters... Success!
Saving calibration file... Success!
================================================================================REQUIRED pro
cess [do_calib-10] has died!
process has finished cleanly
log file: /home/yahboom/.ros/log/5506f06a-2d17-11ef-9db2-28dfeb43061b/do_calib-10*.log
Initiating shutdown!
================================================================================
```

When leaving the factory, the calibration data is saved in,

```
/home/yahboom/YBAMR-COBOT-EDU-
00001/src/yahboom_navrobo_bringup/yahboom_navrobo_bringup/param/imu_calib.yaml
```

4.1.2, use the calibrated imu data

```
roslaunch yahboom_navrobo_bringup bringup_calib.launch
```

The above command is to use the calibrated imu data when starting the chassis driver.

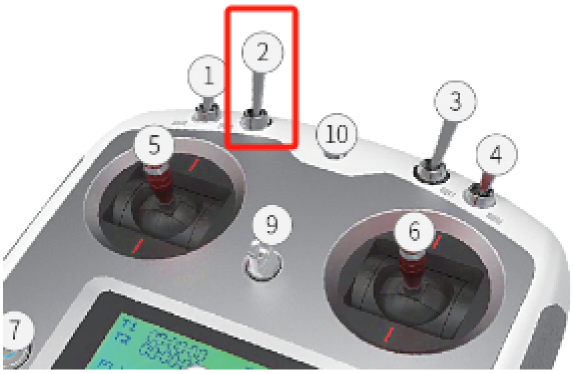
4.2, Linear speed calibration

4.2.1, Preparation

1), Use a ruler to measure a distance of 1 meter and mark it.

2), Put the car at the starting point.

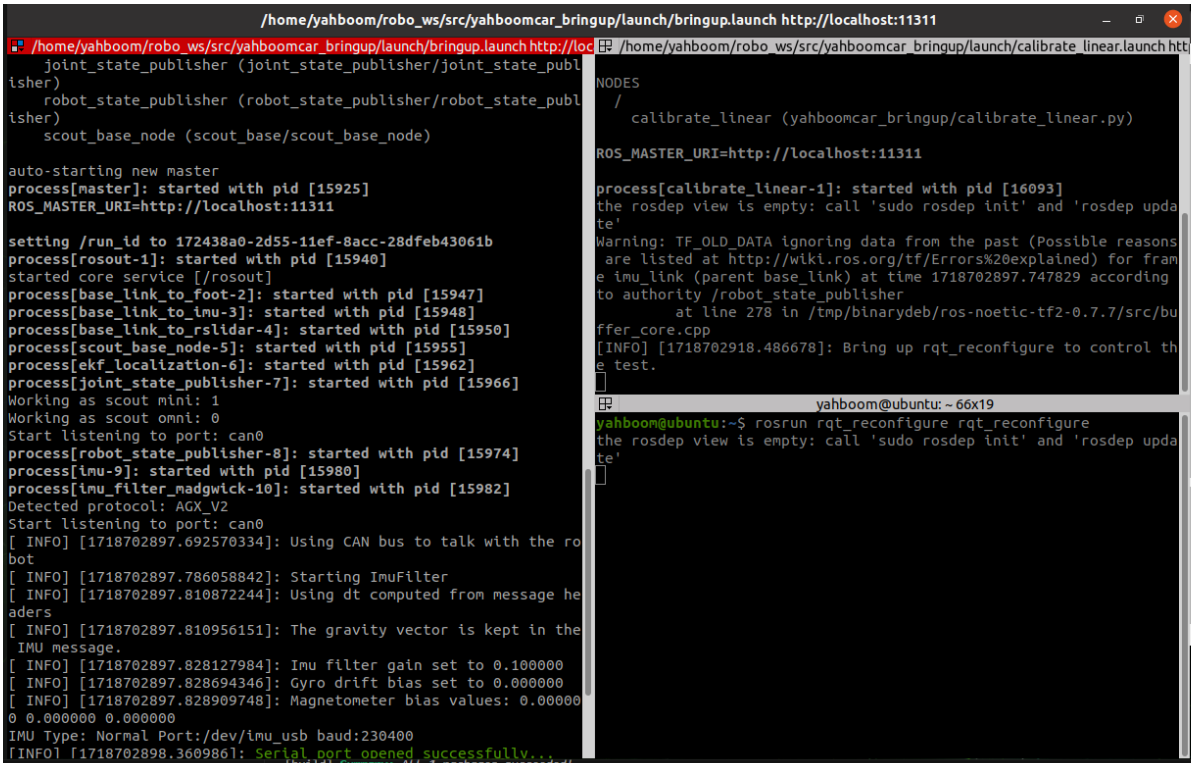
Note: To start control, you need to first turn the SWB button to the upper gear position (control command mode) to release the remote control



4.2.2, Start

1), Terminal input

```
roslaunch yahboom_navrobo_bringup bringup.launch #Start chassis control
roslaunch yahboom_navrobo_bringup calibrate_linear.launch #Start linear speed
calibration
rosrun rqt_reconfigure rqt_reconfigure # Enable dynamic parameter adjustment
```

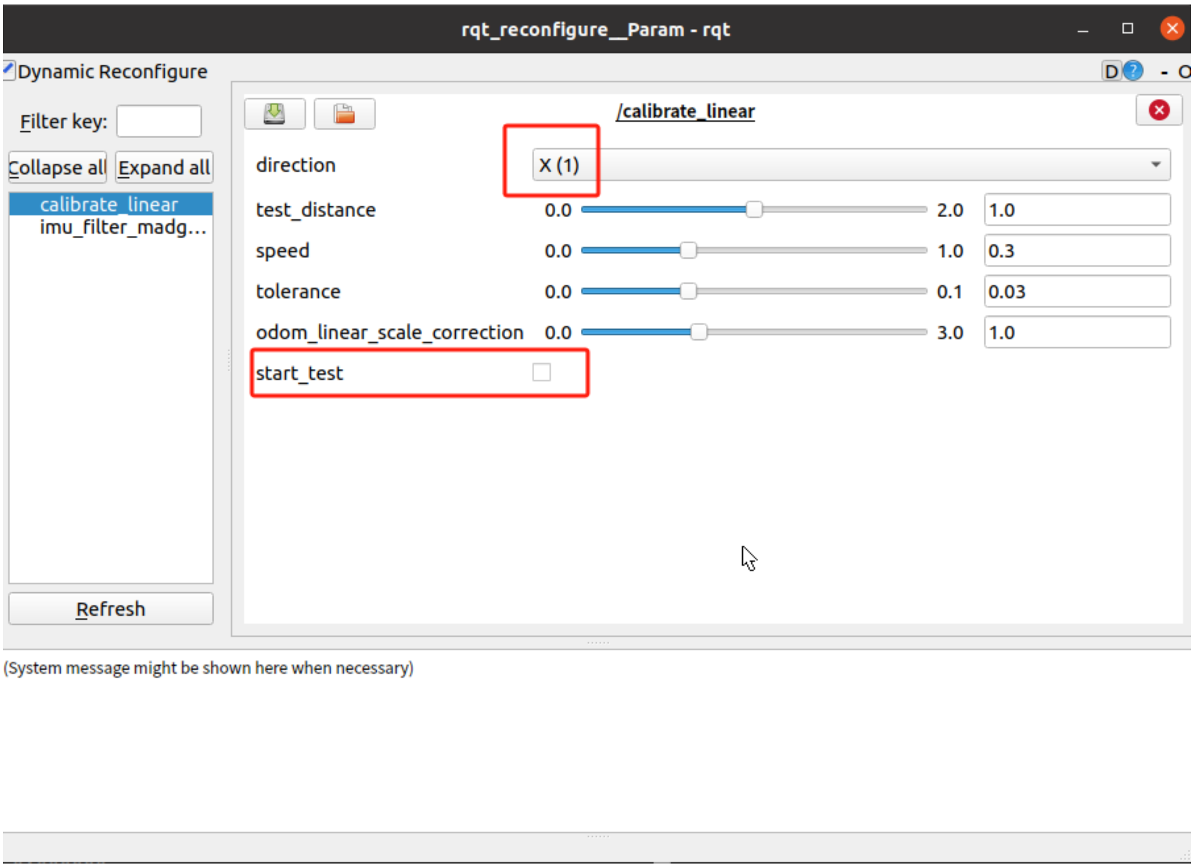

/home/yahboom/robo_ws/src/yahboomcar_bringup/launch/bringup.launch http://localhost:11311
/home/yahboom/robo_ws/src/yahboomcar_bringup/launch/bringup.launch http://loc
/home/yahboom/robo_ws/src/yahboomcar_bringup/launch/calibrate_linear.launch htt
yahboom@ubuntu: ~ 66x19

```
    joint_state_publisher (joint_state_publisher/joint_state_publ
isher)
    robot_state_publisher (robot_state_publisher/robot_state_publ
isher)
    scout_base_node (scout_base/scout_base_node)

auto-starting new master
process[master]: started with pid [15925]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 172438a0-2d55-11ef-8acc-28dfeb43061b
process[rosout-1]: started with pid [15940]
started core service [/rosout]
process[base_link_to_foot-2]: started with pid [15947]
process[base_link_to_imu-3]: started with pid [15948]
process[base_link_to_rslidar-4]: started with pid [15950]
process[scout_base_node-5]: started with pid [15955]
process[ekf_localization-6]: started with pid [15962]
process[joint_state_publisher-7]: started with pid [15966]
Working as scout mini: 1
Working as scout omni: 0
Start listening to port: can0
process[robot_state_publisher-8]: started with pid [15974]
process[imu-9]: started with pid [15980]
process[imu_filter_madgwick-10]: started with pid [15982]
Detected protocol: AGX_V2
Start listening to port: can0
[ INFO] [1718702897.692570334]: Using CAN bus to talk with the ro
bot
[ INFO] [1718702897.786058842]: Starting ImuFilter
[ INFO] [1718702897.810872244]: Using dt computed from message he
aders
[ INFO] [1718702897.810956151]: The gravity vector is kept in the
 IMU message.
[ INFO] [1718702897.828127984]: Imu filter gain set to 0.100000
[ INFO] [1718702897.828694346]: Gyro drift bias set to 0.000000
[ INFO] [1718702897.828909748]: Magnetometer bias values: 0.00000
0 0.000000 0.000000
IMU Type: Normal Port:/dev/imu_usb baud:230400
[INFO] [1718702898.360986]: Serial port opened successfully...
```

```
NODES
  /
    calibrate_linear (yahboomcar_bringup/calibrate_linear.py)

ROS_MASTER_URI=http://localhost:11311

process[calibrate_linear-1]: started with pid [16093]
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep upda
te'
Warning: TF_OLD_DATA ignoring data from the past (Possible reasons
 are listed at http://wiki.ros.org/tf/Errors%20explained) for fram
e imu_link (parent base_link) at time 1718702897.747829 according
to authority /robot_state_publisher
         at line 278 in /tmp/binarydeb/ros-noetic-tf2-0.7.7/src/bu
ffer_core.cpp
[INFO] [1718702918.486678]: Bring up rqt_reconfigure to control th
e test.
```

```
yahboom@ubuntu:~$ rosrun rqt_reconfigure rqt_reconfigure
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep upda
te'
```

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



direction X (1)

test_distance 0.0 2.0 1.0

speed 0.0 1.0 0.3

tolerance 0.0 0.1 0.03

odom_linear_scale_correction 0.0 3.0 1.0

start_test □

Note: The car starts by calibrating the linear speed in the X-axis direction by default. After switching the direction in [direction], click [start_test] to start calibrating the linear speed in the Y-axis.

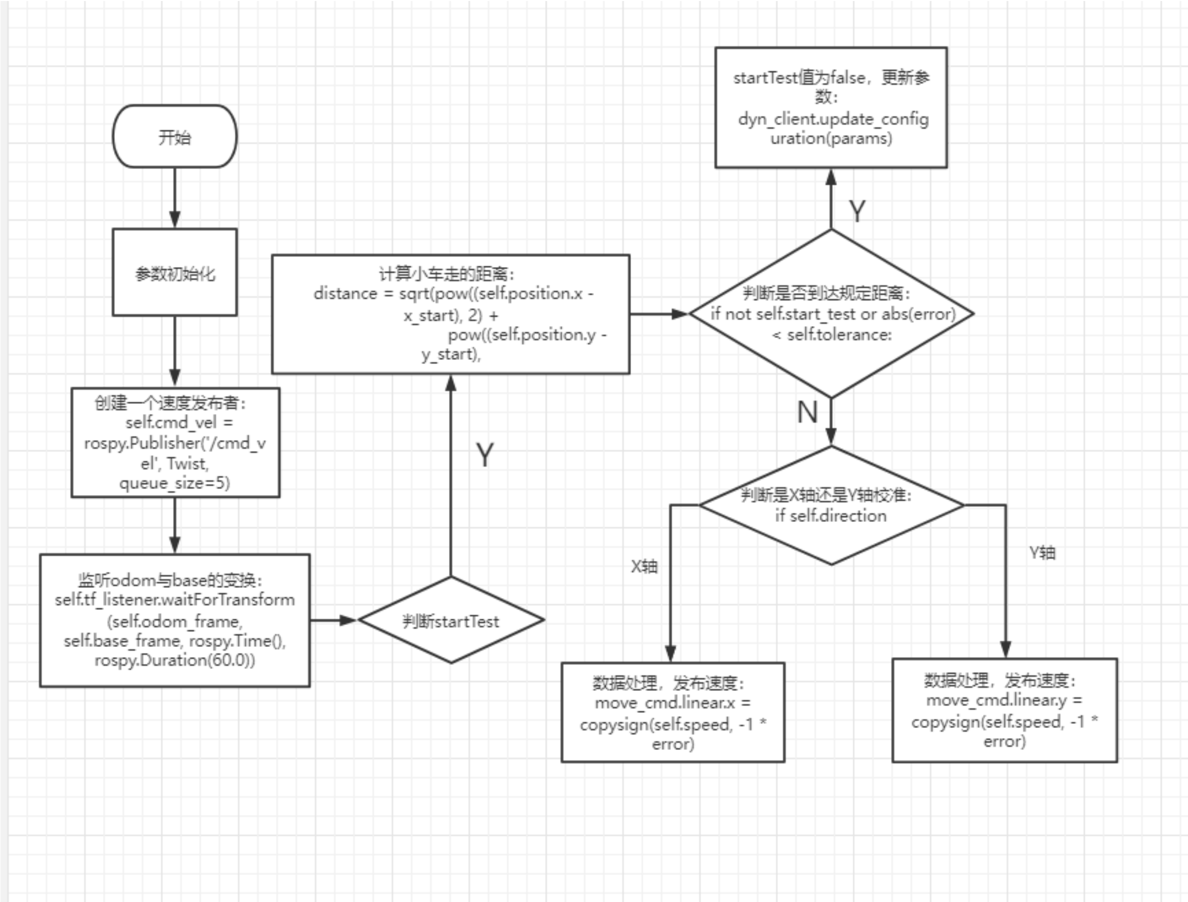
This type of chassis car does not need to calibrate the Y-axis direction! !

- test_distance: test distance. It should not be too large, the default is one meter.
- speed: Test linear speed. If the speed is too high, the inertia will be large.

- tolerance: The error of reaching the target. If the error is too small, it will jitter at the target position. On the contrary, the error of reaching the target point will be large.
- odom_linear_scale_correction: Odometer scaling ratio.
- start_test: Start the test.
- direction: Linear speed direction, the default is X axis.

Remember the value of [odom_linear_scale_correction] after the test.

2), calibrate_linear.py program flow chart



开始
参数初始化
创建一个速度发布者：
self.cmd_vel = rospy.Publisher('/cmd_vel', Twist, queue_size=5)
监听odom与base的变换：
self.tf_listener.waitForTransform (self.odom_frame, self.base_frame, rospy.Time(), rospy.Duration(60.0))
判断startTest
Y
计算小车走的距离：
distance = sqrt(pow((self.position.x - x_start), 2) + pow((self.position.y - y_start),
判断是否到达规定距离：
if not self.start_test or abs(error) < self.tolerance:
Y
startTest值为false，更新参数：
dyn_client.update_configuration(params)
N
判断是X轴还是Y轴校准：
if self.direction
X轴
Y轴
数据处理，发布速度：
move_cmd.linear.x = copysign(self.speed, -1 * error)
数据处理，发布速度：
move_cmd.linear.y = copysign(self.speed, -1 * error)

4.3, Angular velocity calibration

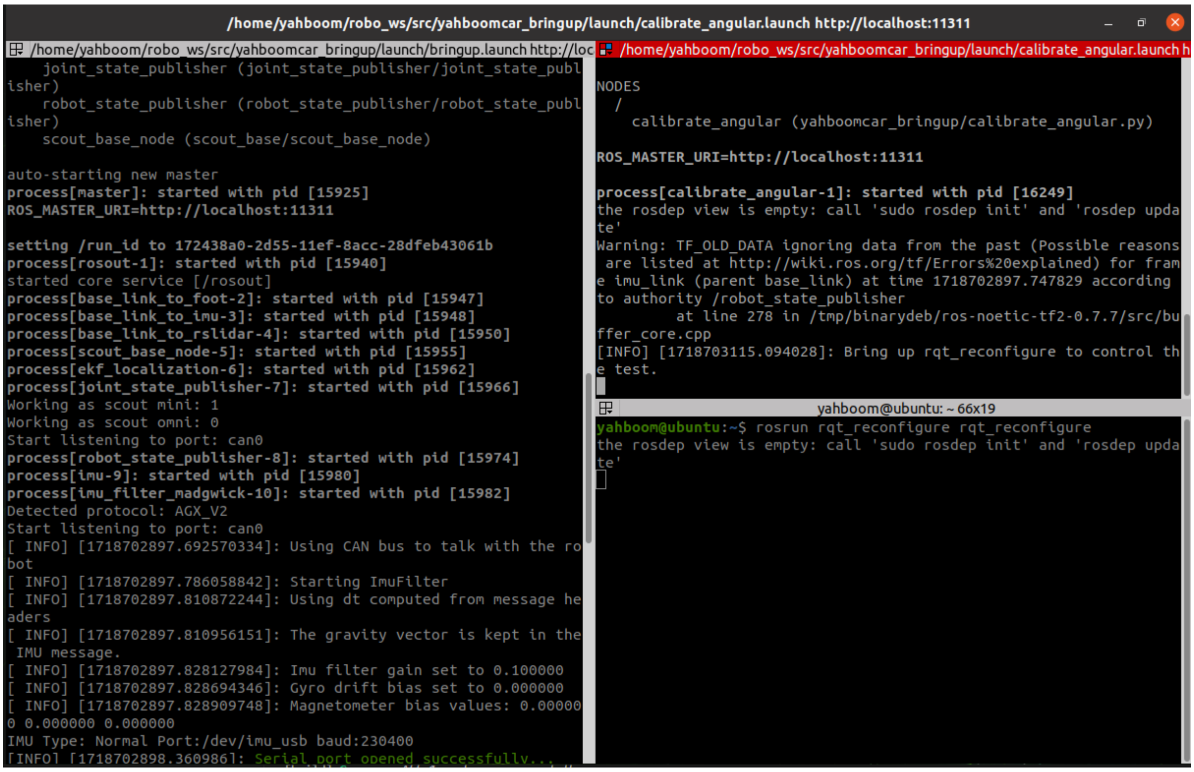
4.3.1, Preparation

1), Put the car in a position where it is easy to rotate the angles.

4.3.2, Startup

1), Terminal input

```
roslaunch yahboom_navrobo_bringup bringup.launch #Open chassis control
roslaunch yahboom_navrobo_bringup calibrate_angular.launch #Open angular velocity calibration
rosrun rqt_reconfigure rqt_reconfigure #Open dynamic parameter adjustment
```



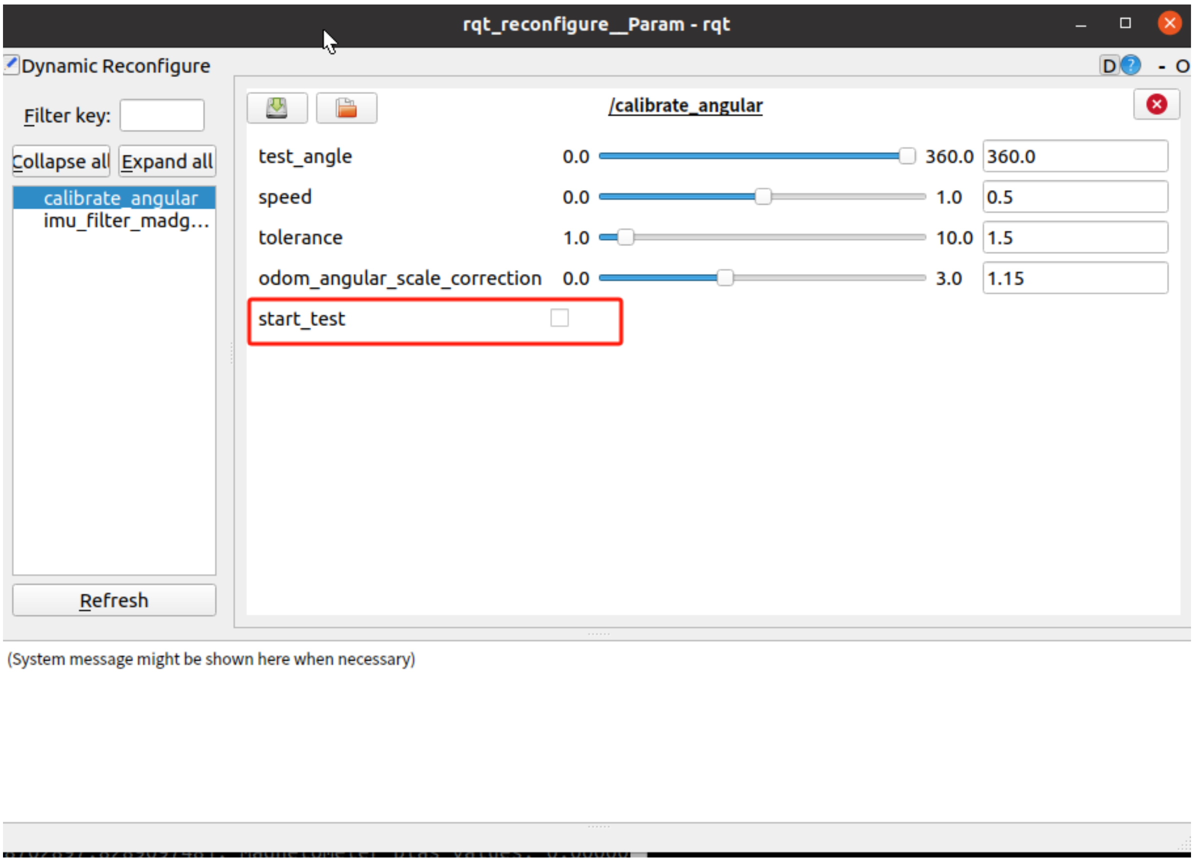
/home/yahboom/robo_ws/src/yahboomcar_bringup/launch/calibrate_angular.launch http://localhost:11311
/home/yahboom/robo_ws/src/yahboomcar_bringup/launch/bringup.launch http://loc
/home/yahboom/robo_ws/src/yahboomcar_bringup/launch/calibrate_angular.launch h
yahboom@ubuntu: ~ 66x19

```
    joint_state_publisher (joint_state_publisher/joint_state_publ
isher)
    robot_state_publisher (robot_state_publisher/robot_state_publ
isher)
    scout_base_node (scout_base/scout_base_node)

auto-starting new master
process[master]: started with pid [15925]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 172438a0-2d55-11ef-8acc-28dfeb43061b
process[rosout-1]: started with pid [15940]
started core service [/rosout]
process[base_link_to_foot-2]: started with pid [15947]
process[base_link_to_imu-3]: started with pid [15948]
process[base_link_to_rslidar-4]: started with pid [15950]
process[scout_base_node-5]: started with pid [15955]
process[ekf_localization-6]: started with pid [15962]
process[joint_state_publisher-7]: started with pid [15966]
Working as scout mini: 1
Working as scout omni: 0
Start listening to port: can0
process[robot_state_publisher-8]: started with pid [15974]
process[imu-9]: started with pid [15980]
process[imu_filter_madgwick-10]: started with pid [15982]
Detected protocol: AGX_V2
Start listening to port: can0
[ INFO] [1718702897.692570334]: Using CAN bus to talk with the ro
bot
[ INFO] [1718702897.786058842]: Starting ImuFilter
[ INFO] [1718702897.810872244]: Using dt computed from message he
aders
[ INFO] [1718702897.810956151]: The gravity vector is kept in the
 IMU message.
[ INFO] [1718702897.828127984]: Imu filter gain set to 0.100000
[ INFO] [1718702897.828694346]: Gyro drift bias set to 0.000000
[ INFO] [1718702897.828909748]: Magnetometer bias values: 0.00000
0 0.000000 0.000000
IMU Type: Normal Port:/dev/imu_usb baud:230400
[INFO] [1718702898.360986]: Serial port opened successfully...
```

```

NODES
  /
    calibrate_angular (yahboomcar_bringup/calibrate_angular.py)

ROS_MASTER_URI=http://localhost:11311

process[calibrate_angular-1]: started with pid [16249]
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep upda
te'
Warning: TF_OLD_DATA ignoring data from the past (Possible reasons
 are listed at http://wiki.ros.org/tf/Errors%20explained) for fram
e imu_link (parent base_link) at time 1718702897.747829 according
to authority /robot_state_publisher
         at line 278 in /tmp/binarydeb/ros-noetic-tf2-0.7.7/src/bu
ffer_core.cpp
[INFO] [1718703115.094028]: Bring up rqt_reconfigure to control th
e test.
```

```
yahboom@ubuntu:~$ rosrun rqt_reconfigure rqt_reconfigure
the rosdep view is empty: call 'sudo rosdep init' and 'rosdep upda
te'
```

2), Click the right square of [start_test] and start moving [test_angle] distance. At this time, observe whether the car actually rotates [test_angle]. If not, adjust the parameter [odom_angule_scale_correction] and put the car back to the starting point to continue testing.



test_angle 0.0 360.0 360.0

speed 0.0 1.0 0.5

tolerance 1.0 10.0 1.5

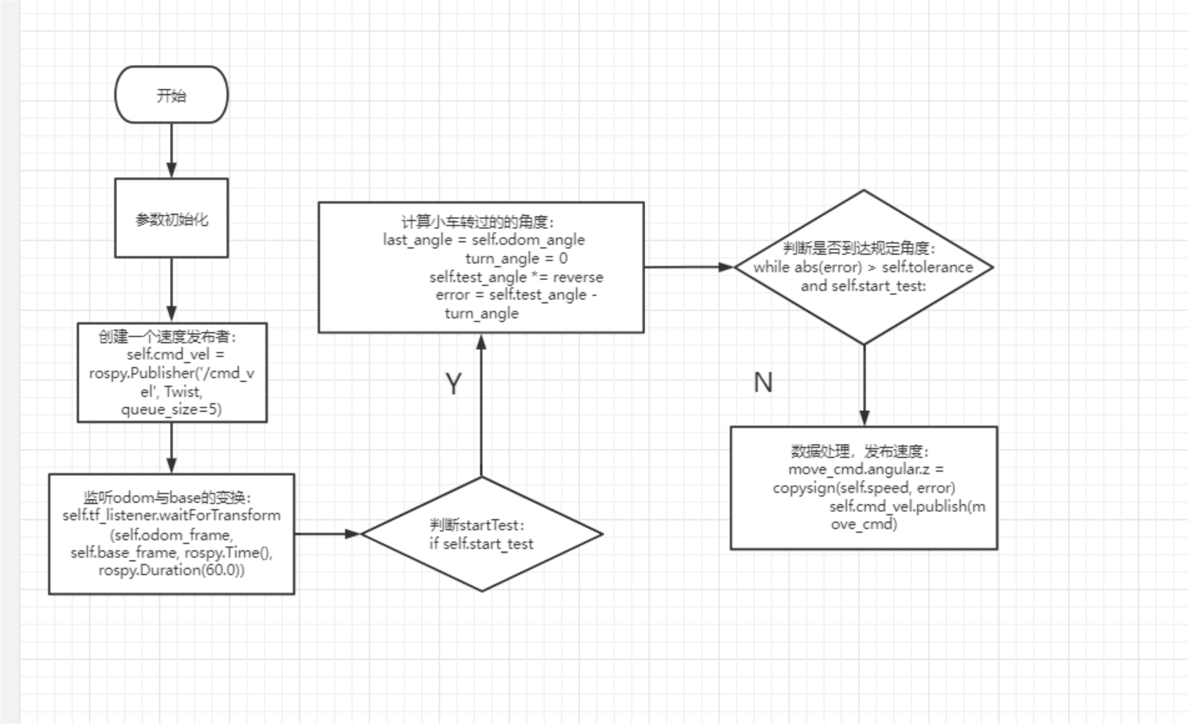
odom_angular_scale_correction 0.0 3.0 1.15

start_test □

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

After the test, remember the value of [odom_angule_scale_correction].

3), calibrate_angular.py program flow chart


开始
参数初始化
创建一个速度发布者：self.cmd_vel = rospy.Publisher('/cmd_vel', Twist, queue_size=5)
监听odom与base的变换：self.tf_listener.waitForTransform (self.odom_frame, self.base_frame, rospy.Time(), rospy.Duration(60.0))
判断startTest：if self.start_test
Y
计算小车转过的的角度：last_angle = self.odom_angle turn_angle = 0 self.test_angle *= reverse error = self.test_angle - turn_angle
判断是否到达规定角度：while abs(error) > self.tolerance and self.start_test:
N
数据处理，发布速度：move_cmd.angular.z = copysign(self.speed, error) self.cmd_vel.publish(move_cmd)