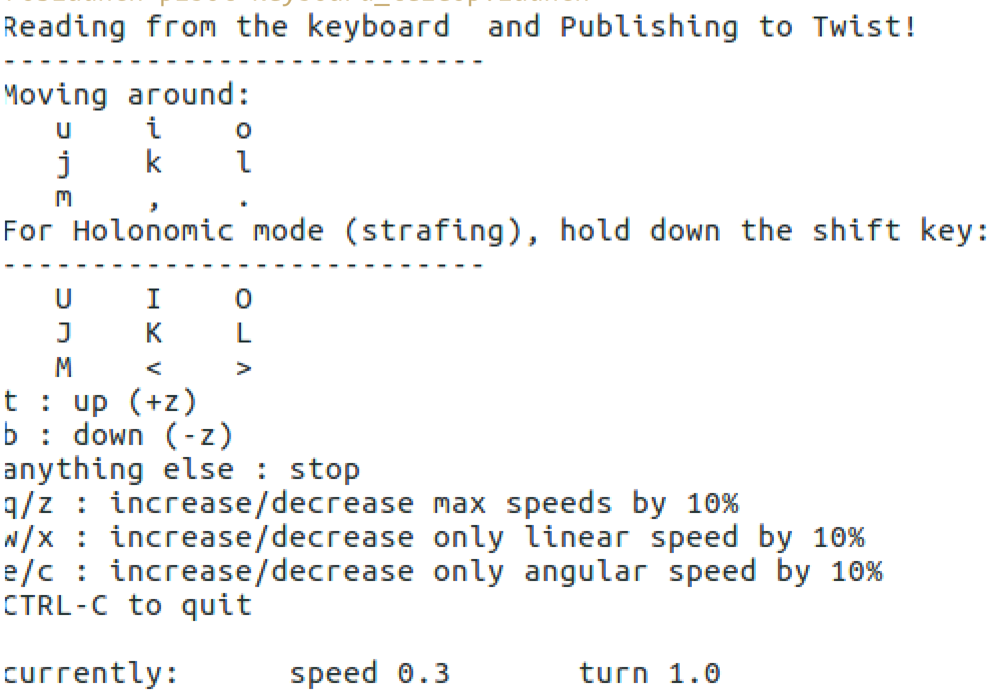
- 1. 运动控制
  - 1.1键盘控制
  - 1.2手柄控制
    - 连接手柄
    - 测试手柄
    - 手柄控制
  - 1.3App控制
- 2. 里程校准
  - 2.1linear_calibrate
    - 启动校准
    - 调整参数
  - 2.2angular_calibrate
    - 启动校准
    - 调整参数
- 3备注

1. 运动控制

1.1键盘控制

> 启动驱动`roslaunch pibot_bringup bringup.launch`,下同

```
roslaunch pibot keyboard_teleop.launch
```



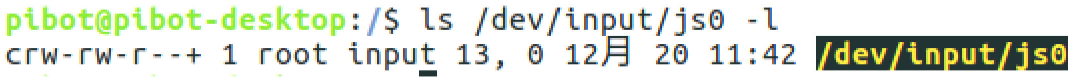
```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .
For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >
t : up (+z)
b : down (-z)
anything else : stop
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
CTRL-C to quit

currently:      speed 0.3       turn 1.0
```

根据提示可以控制全向的Zues和差分的Apollo小车

1.2手柄控制

连接手柄

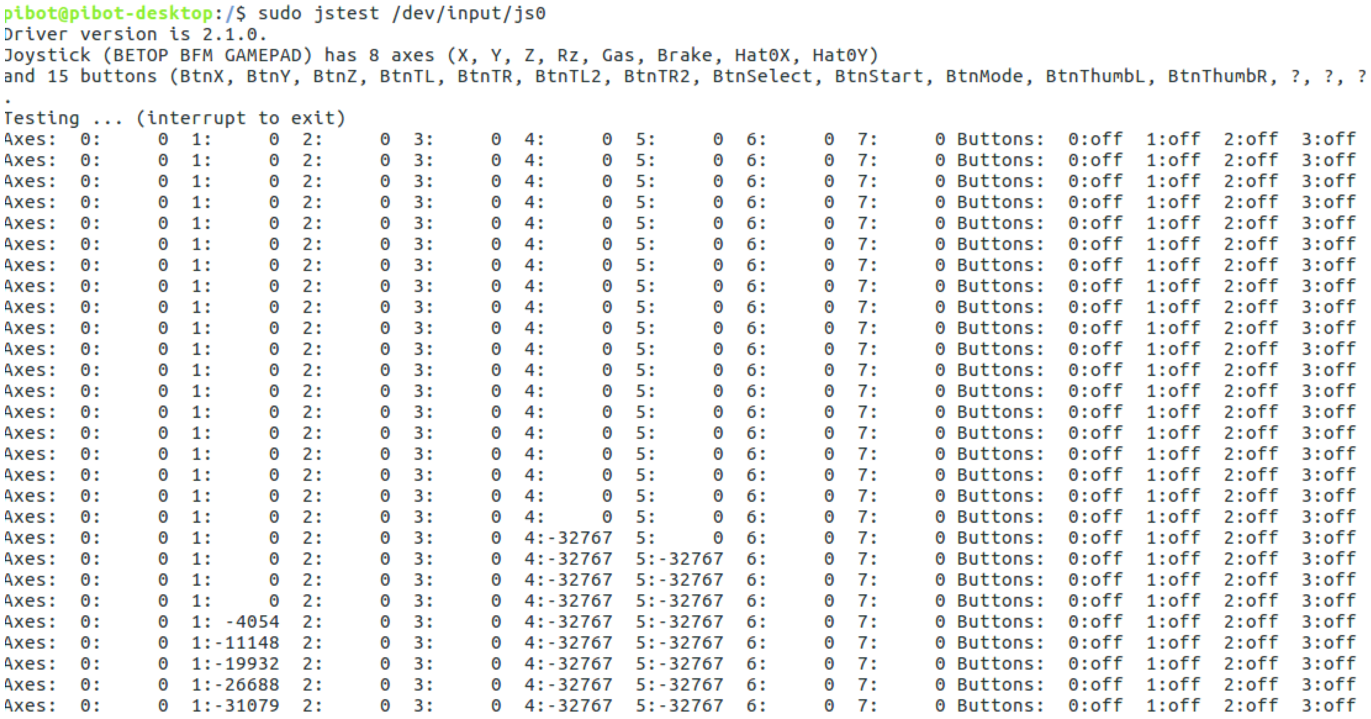
连接手柄至电脑或者`Raspberry Pi 3` `ls /dev/input/js* -l`



```
pibot@pibot-desktop:/$ ls /dev/input/js0 -l
crw-rw-r--+ 1 root input 13, 0 12月 20 11:42 /dev/input/js0
```

测试手柄

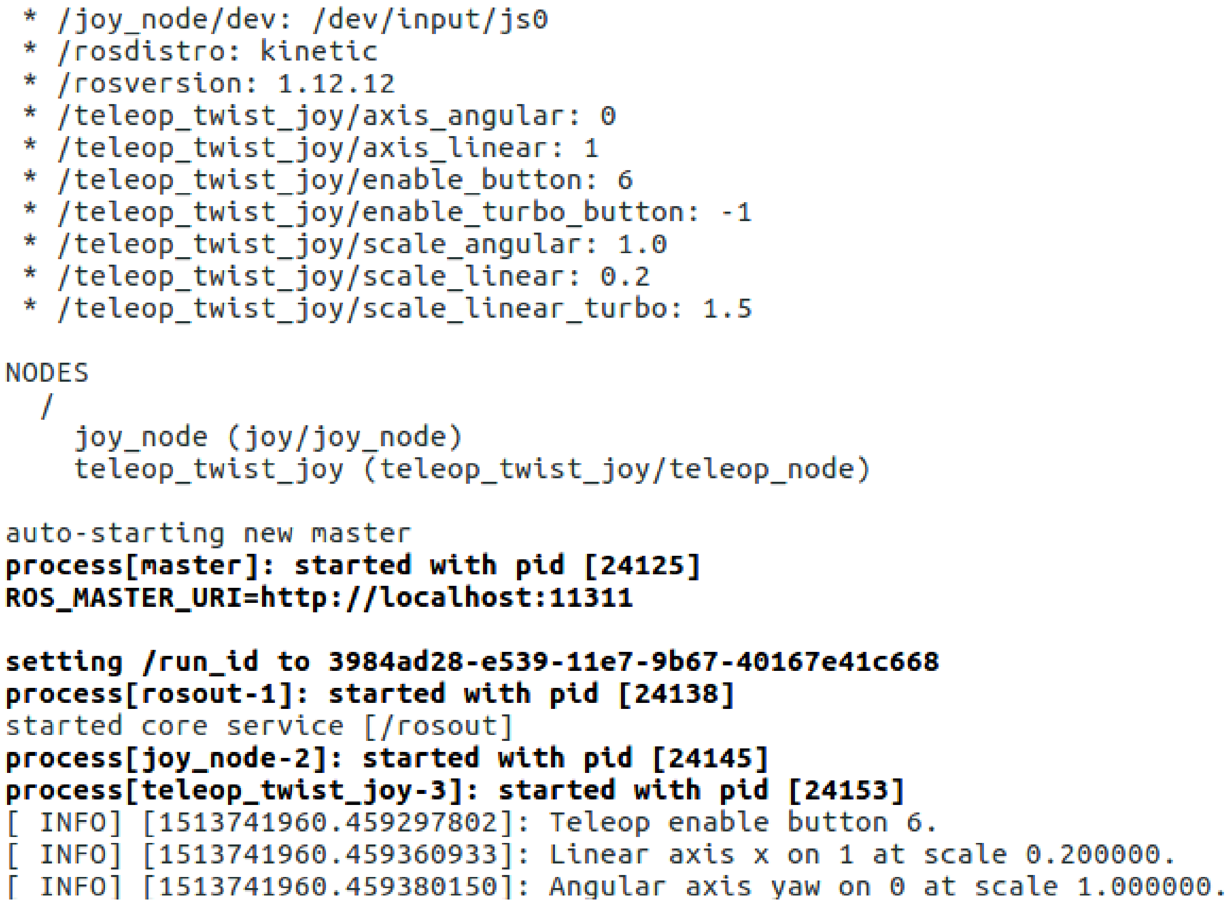
`sudo jstest /dev/input/js0`



```
pibot@pibot-desktop:/$ sudo jstest /dev/input/js0
Driver version is 2.1.0.
Joystick (BETOP BFM GAMEPAD) has 8 axes (X, Y, Z, Rz, Gas, Brake, Hat0X, Hat0Y)
and 15 buttons (BtnX, BtnY, BtnZ, BtnTL, BtnTR, BtnTL2, BtnTR2, BtnSelect, BtnStart, BtnMode, BtnThumbL, BtnThumbR, ?, ?, ?
.
Testing ... (interrupt to exit)
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:     0  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:-32767  5:     0  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:     0  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1: -4054  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:-11148  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:-19932  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:-26688  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
Axes:  0:     0  1:-31079  2:     0  3:     0  4:-32767  5:-32767  6:     0  7:     0 Buttons:  0:off  1:off  2:off  3:off
```

手柄控制

`Zeus`小车 `roslaunch pibot joystick.launch` `Apollo`小车 `roslaunch pibot joystick-holonomic.launch`



```
 * /joy_node/dev: /dev/input/js0
 * /rosdistro: kinetic
 * /rosversion: 1.12.12
 * /teleop_twist_joy/axis_angular: 0
 * /teleop_twist_joy/axis_linear: 1
 * /teleop_twist_joy/enable_button: 6
 * /teleop_twist_joy/enable_turbo_button: -1
 * /teleop_twist_joy/scale_angular: 1.0
 * /teleop_twist_joy/scale_linear: 0.2
 * /teleop_twist_joy/scale_linear_turbo: 1.5

NODES
  /
    joy_node (joy/joy_node)
    teleop_twist_joy (teleop_twist_joy/teleop_node)

auto-starting new master
process[master]: started with pid [24125]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 3984ad28-e539-11e7-9b67-40167e41c668
process[rosout-1]: started with pid [24138]
started core service [/rosout]
process[joy_node-2]: started with pid [24145]
process[teleop_twist_joy-3]: started with pid [24153]
[ INFO] [1513741960.459297802]: Teleop enable button 6.
[ INFO] [1513741960.459360933]: Linear axis x on 1 at scale 0.200000.
[ INFO] [1513741960.459380150]: Angular axis yaw on 0 at scale 1.000000.
```

`joystick.launch`

```
<launch>
  <arg name="joy_config" default="joystick" />
  <arg name="joy_dev" default="/dev/input/js0" />
  <arg name="config_filepath" default="$(find pibot)/config/$(arg
joy_config).config.yaml" />

  <node pkg="joy" type="joy_node" name="joy_node">
    <param name="dev" value="$(arg joy_dev)" />
    <param name="deadzone" value="0.3" />
    <param name="autorepeat_rate" value="20" />
  </node>

  <node pkg="teleop_twist_joy" name="teleop_twist_joy" type="teleop_node"
output="screen">
    <rosparam command="load" file="$(arg config_filepath)" />
  </node>
</launch>
```

`joystick.config.yaml`

```
axis_linear: 1  # Left thumb stick vertical
scale_linear: 0.2
scale_linear_turbo: 1.5
```

```
  axis_angular: 0  # Left thumb stick horizontal
  scale_angular: 1.0

  enable_button: 6  # Left trigger button
  enable_turbo_button: -1
```

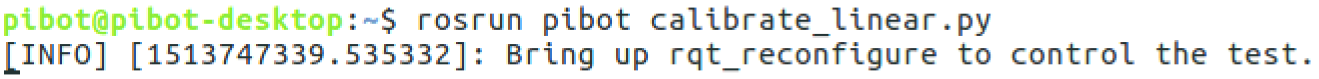
1.3App控制

2. 里程校准

2.1linear_calibrate

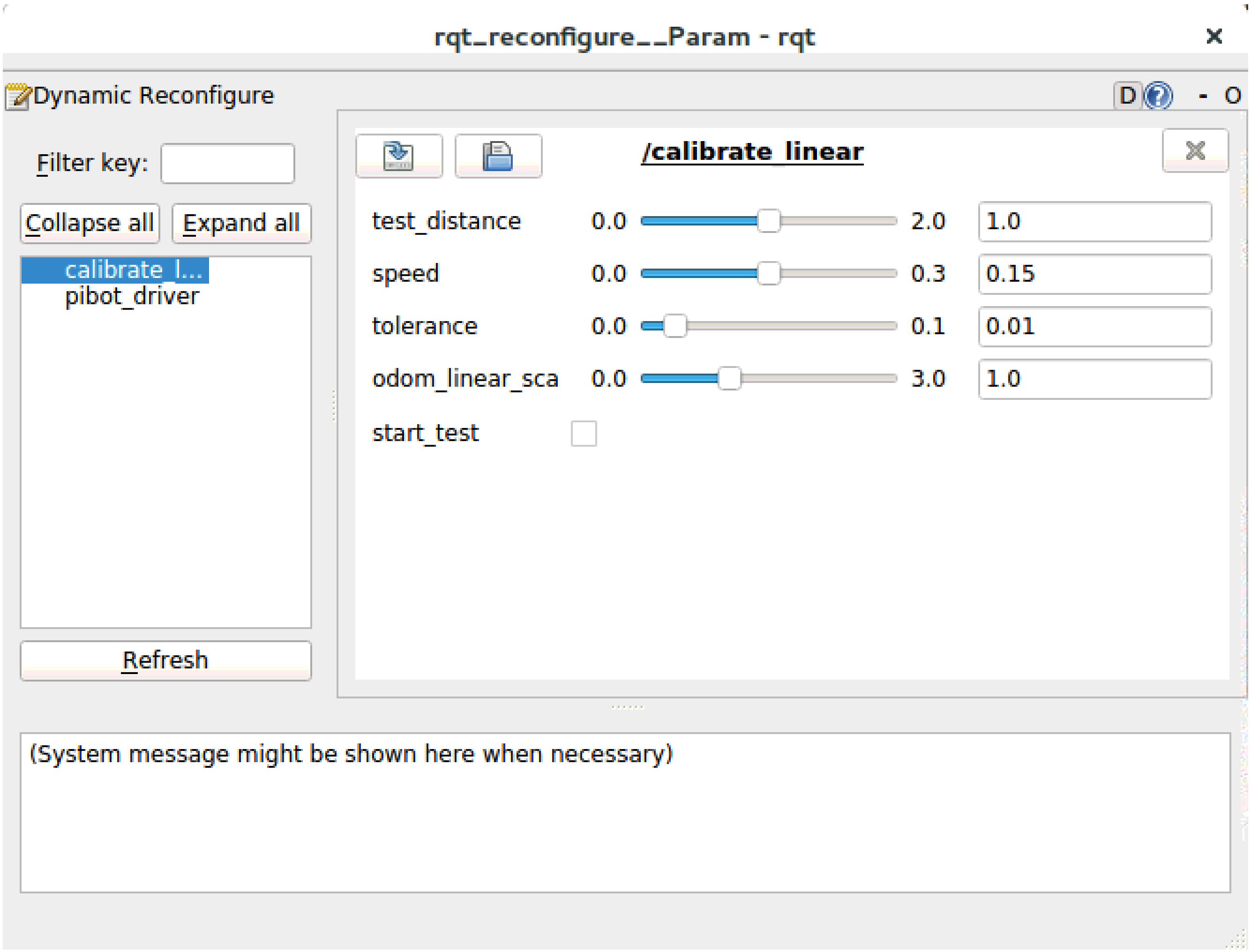
启动校准

`rosrun pibot calibrate_linear.py`



```
pibot@pibot-desktop:~$ rosrun pibot calibrate_linear.py
[INFO] [1513747339.535332]: Bring up rqt_reconfigure to control the test.
```

按照提示启动`rqt_reconfigure`



rqt_reconfigure__Param - rqt

Dynamic Reconfigure

/calibrate_linear

```
test_distance     0.0  2.0  1.0
speed             0.0  0.3  0.15
tolerance         0.0  0.1  0.01
odom_linear_sca   0.0  3.0  1.0
start_test        □
```

(System message might be shown here when necessary)

切换到`calibrate_linear`选项， 勾选`start_test`即可开始测试。小车按照设置的速度(`speed`),向前运动设定的距离(`test_distance`),误差不超过设定值(`tolerance`)

> `odom_linear_scale_correction`为比例参数，设为默认1即可

调整参数

用尺子测量小车实际行径距离，如果与`test_distance`相差较大，则需要调整相关参数 对于2轮差分`Apollo differential.h`

```
        void get_odom(struct Odom* odom, float* motor_dis, unsigned long interval)
        {
                float dxy_ave = (-motor_dis[0] + motor_dis[1]) / 2.0;
                float dth = (motor_dis[0] + motor_dis[1]) / (2* body_radius);
                float vxy = 1000 * dxy_ave / interval;
                float vth = 1000 * dth / interval;

                odom->vel_x = vxy;
                odom->vel_y = 0;
                odom->vel_z = vth;
                float dx = 0, dy = 0;
                if (motor_dis[0] != motor_dis[1])
                {
                        dx = cos(dth) * dxy_ave;
                        dy = -sin(dth) * dxy_ave;
                        odom->x += (cos(odom->z) * dx - sin(odom->z) * dy);
                        odom->y += (sin(odom->z) * dx + cos(odom->z) * dy);;
                }

                if (motor_dis[0] + motor_dis[1] != 0)
                        odom->z += dth;
        }
```

单独向前是`motor_dis[0] + motor_dis[1]`应该为0

> 左轮向后motor_dis[0]为正，右轮向前为正

容易得到`odom->x`因为`(-motor_dis[0] + motor_dis[1]) / 2.0`，而`motor_dis[0]`,`motor_dis[1]`跟一周编码器个数和轮子的直接相关，在假定一周编码器个数恒定情况下，即只与轮子直接相关

> **这也是为什么先进行linear_calibrate的原因**

如果实际测量值<`test_distance`，应该如何调整轮子直径，调大？调小？

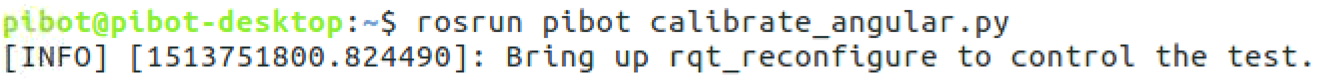
> 即例如实际行走了0.8m，计算出来的为1m，`odom->x`大了，即用来计算直径的参数大了，应该减小直径。

2.2angular_calibrate

启动校准

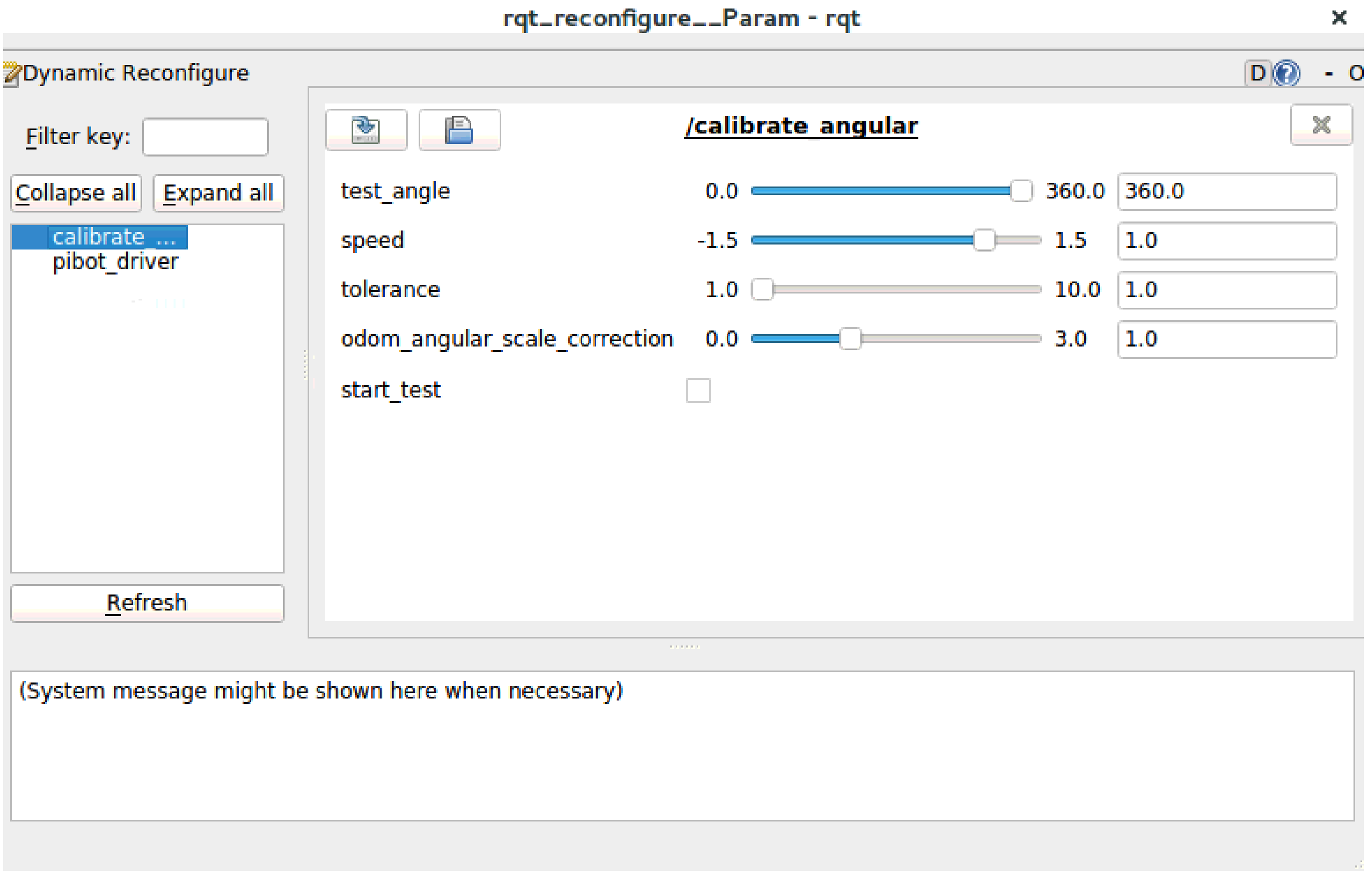
`rosrun pibot calibrate_angular.py`

> 出现`ImportError: No module named PyKDL`错误需要`sudo apt-get install ros-kinetic-kdl-parser-py`



```
pibot@pibot-desktop:~$ rosrun pibot calibrate_angular.py
[INFO] [1513751800.824490]: Bring up rqt_reconfigure to control the test.
```

按照提示启动rqt_reconfigure



test_angle 0.0 — 360.0 360.0
speed -1.5 — 1.5 1.0
tolerance 1.0 — 10.0 1.0
odom_angular_scale_correction 0.0 — 3.0 1.0
start_test □

(System message might be shown here when necessary)

切换到calibrate_angular选项， 勾选start_test即可开始测试。小车按照设置的速度(speed),旋转设定的角度(test_angle),误差不超过设定值(tolerance)

> odom_linear_scale_correction为比例参数，设为默认1即可

调整参数

观察设计旋转的角度，如果与test_angle相差较大，则需要调整相关参数 对于2轮差分Apollo differential.h

```
        void get_odom(struct Odom* odom, float* motor_dis, unsigned long interval)
        {
                float dxy_ave = (-motor_dis[0] + motor_dis[1]) / 2.0;
                float dth = (motor_dis[0] + motor_dis[1]) / (2* body_radius);
                float vxy = 1000 * dxy_ave / interval;
                float vth = 1000 * dth / interval;

                odom->vel_x = vxy;
                odom->vel_y = 0;
                odom->vel_z = vth;
                float dx = 0, dy = 0;
                if (motor_dis[0] != motor_dis[1])
                {
                        dx = cos(dth) * dxy_ave;
                        dy = -sin(dth) * dxy_ave;
                        odom->x += (cos(odom->z) * dx - sin(odom->z) * dy);
                        odom->y += (sin(odom->z) * dx + cos(odom->z) * dy);;
```

```
                    }

                    if (motor_dis[0] + motor_dis[1] != 0)
                            odom->z += dth;
            }
```

旋转是odom->z为dth累加即 (motor_dis[0] + motor_dis[1]) / (2* body_radius)

> 先前完成了linear_calibrate，(motor_dis[0] + motor_dis[1])就固定了，现在odom->z就只与body_radius相关，且为反比关系

如果实际观察角度<test_angle，应该如何调整轮子间距，调大？调小？

> 即例如实际行走了345°，计算出来的为360°，odom->z大了即body_radius小了(反比)，应该增加body_radius。

3备注

上述为差分轮apollo的参数调整，zues 、hades和hera也类似

> **如果实在搞不清楚应该调大参数还是调小，那就调整参数直接测试，观察结果，这样直接也同样高效！**