3. Robot state estimation

3.1. Startup

3.1.1. Code reference path

```
/home/yahboom/YBAMR-COBOT-EDU-
00001/src/yahboom_navrobo_bringup/yahboom_navrobo_bringup/launch
```

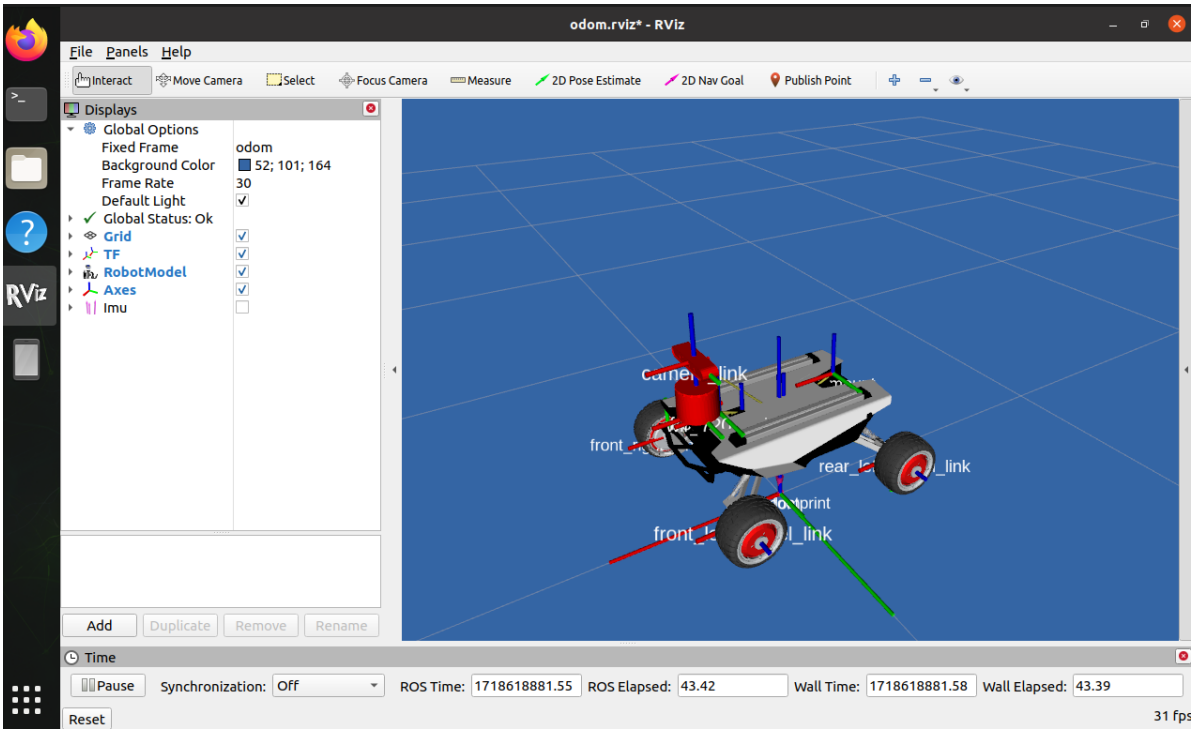
3.1.1. Startup

Stop the self-starting chassis service

```
sudo supervisorctl stop ChassisServer
```

Start the chassis driver

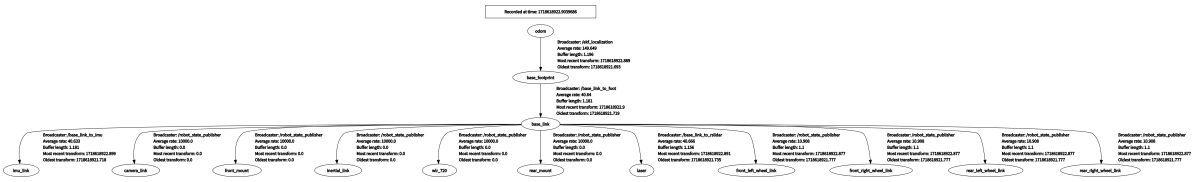
```
roslaunch yahboom_navrobo_bringup bringup.launch use_rviz:=true pub_tf:=true
```



3.1.1、View tf tree and node graph

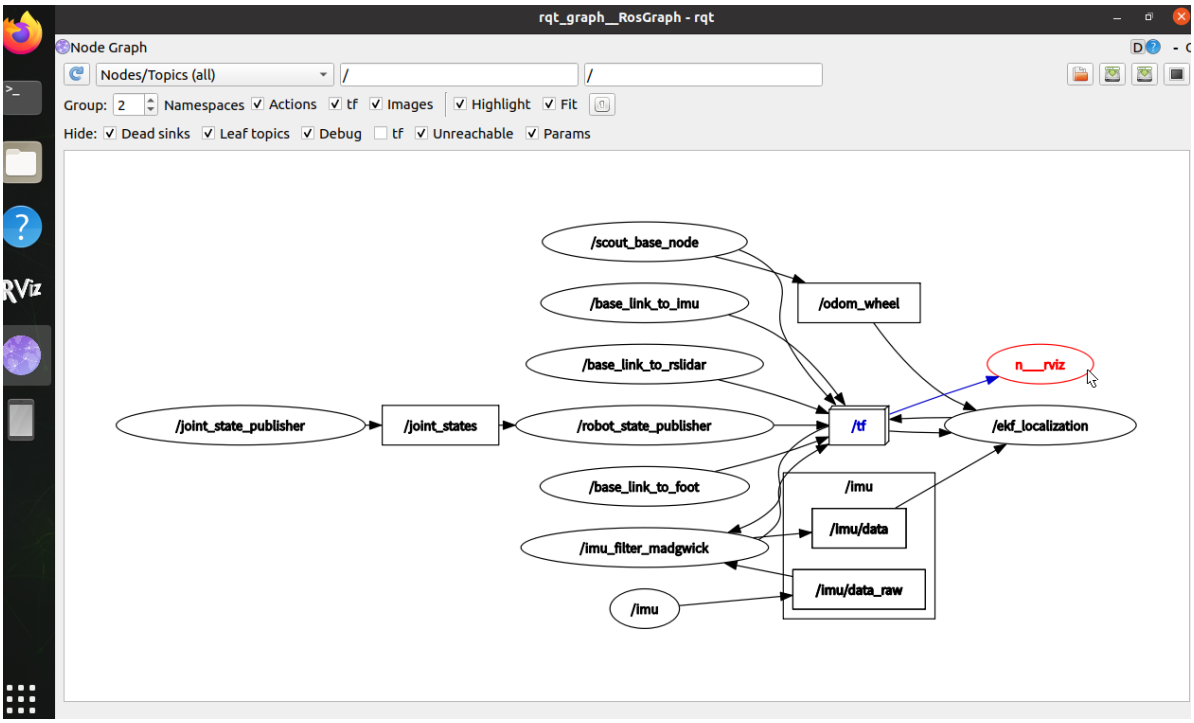
1）、View TF tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



2）、View node graph

```
rosrun rqt_graph rqt_graph
```



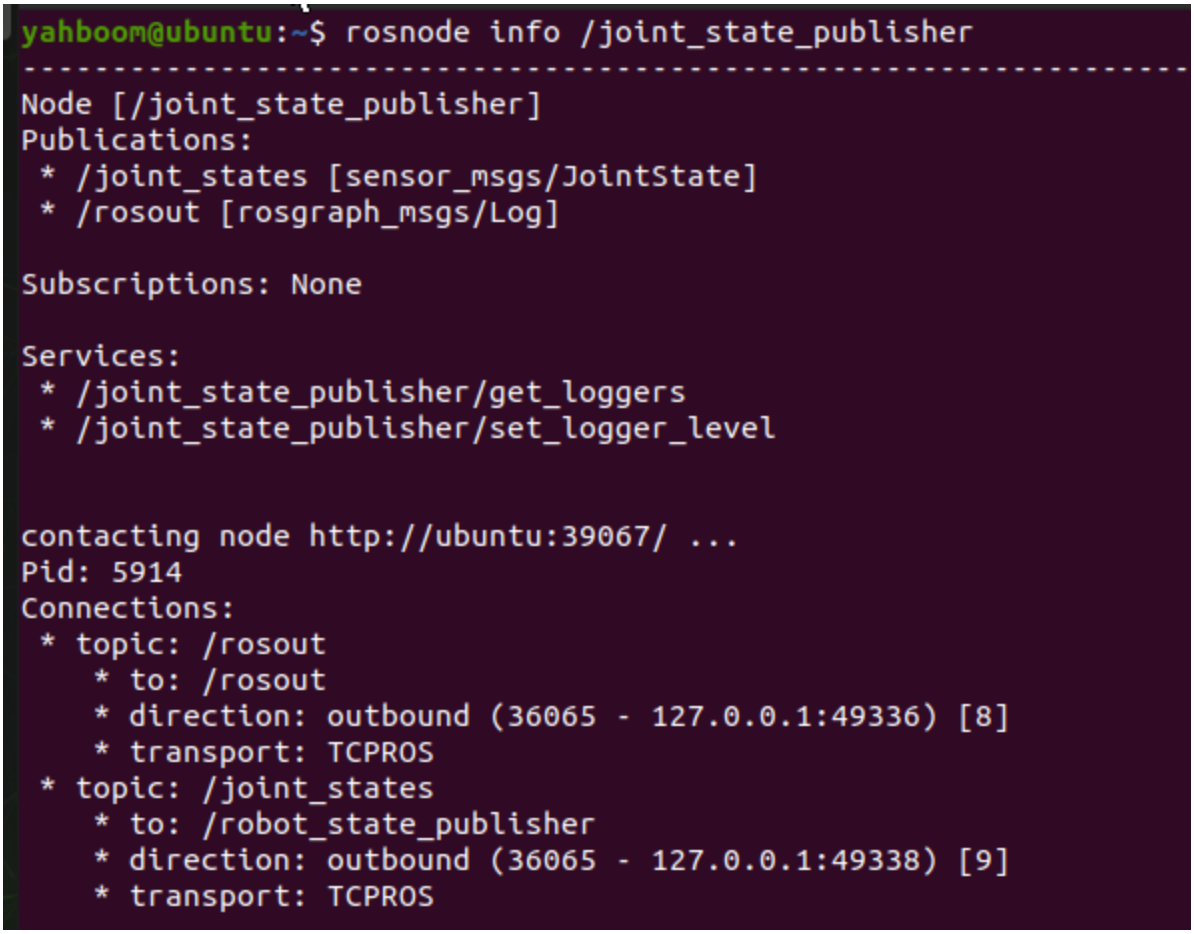
3.1.2, launch file analysis

In the bringup.launch file, there are several important nodes

1), /joint_state_publisher

joint_state_publisher is mainly used to publish joint state information, including joint angle, velocity, acceleration and other information. It receives data from the robot hardware and converts it into JointState messages in ROS, and then publishes it to the ROS system. Terminal input,

```
rosnode info /joint_state_publisher
```



2), /robot_state_publisher

robot_state_publisherr is based on joint_state_publisher, and further publishes the complete state information of the robot. In addition to joint state information, it also publishes the robot's TF transformation information, including the transformation relationship between the robot coordinate system and the joint coordinate system. Terminal input,

```
rosnode info /robot_state_publisher
```

```
yahboom@ubuntu:~$ rosnode info /robot_state_publisher
--------------------------------------------------------------------------------
Node [/robot_state_publisher]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /tf [tf2_msgs/TFMessage]
 * /tf_static [tf2_msgs/TFMessage]

Subscriptions:
 * /joint_states [sensor_msgs/JointState]

Services:
 * /robot_state_publisher/get_loggers
 * /robot_state_publisher/set_logger_level


contacting node http://ubuntu:42475/ ...
Pid: 5919
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound (35227 - 127.0.0.1:40924) [13]
    * transport: TCPROS
 * topic: /tf
    * to: /ekf_localization
    * direction: outbound (35227 - 127.0.0.1:40936) [10]
    * transport: TCPROS
 * topic: /tf
    * to: /rviz
    * direction: outbound (35227 - 127.0.0.1:40984) [17]
    * transport: TCPROS
 * topic: /tf_static
    * to: /ekf_localization
    * direction: outbound (35227 - 127.0.0.1:40938) [14]
    * transport: TCPROS
 * topic: /tf_static
    * to: /rviz
    * direction: outbound (35227 - 127.0.0.1:40992) [18]
    * transport: TCPROS
 * topic: /joint_states
```

3), /ekf_localization

This node is mainly used to integrate imu data and odom data and publish tf data. Terminal input,

```
rosnode info /ekf_localization
```

```
yahboom@ubuntu:~/robo_ws/src/yahboomcar_bringup/launch$ rosnode info /ekf_localization
--------------------------------------------------------------------------------
Node [/ekf_localization]
Publications:
 * /diagnostics [diagnostic_msgs/DiagnosticArray]
 * /odometry/filtered [nav_msgs/Odometry]
 * /rosout [rosgraph_msgs/Log]
 * /tf [tf2_msgs/TFMessage]

Subscriptions:
 * /imu/data [sensor_msgs/Imu]
 * /odom_wheel [nav_msgs/Odometry]
 * /set_pose [unknown type]
 * /tf [tf2_msgs/TFMessage]
 * /tf_static [tf2_msgs/TFMessage]

Services:
 * /ekf_localization/enable
 * /ekf_localization/get_loggers
 * /ekf_localization/set_logger_level
 * /ekf_localization/toggle
 * /set_pose


contacting node http://ubuntu:43339/ ...
Pid: 351993
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound (41805 - 127.0.0.1:39970) [19]
    * transport: TCPROS
 * topic: /tf
    * to: /ekf_localization
    * direction: outbound
    * transport: INTRAPROCESS
 * topic: /tf
    * to: /rviz
```

The following nodes are also started,

- base_link_to_foot: Publish a static change, mainly publish the position transformation of the robot chassis and the car.
- base_link_to_imu: Publish a static change, mainly publish the position transformation of the imu module and the car.
- base_link_to_rslidar: Publishes a static change, mainly publishing the position transformation between the radar and the car.

3.1.3, imu_filter_madgwick

1), Introduction

Madgwick is an Orientation Filter used to filter and fuse raw data from IMU devices. It fuses angular velocity, acceleration, and (optional) magnetometer readings from generic IMU devices into orientation quaternions and publishes the fused data on the IMU topic, without considering the integration process of the entire IMU.

2), Topics

/	Topic Name	Type	Parsing
Subscribed	/imu/data_raw	sensor_msgs/Imu	Messages of calibrated IMU data, including angular velocity and linear acceleration
Subscribed	/mag/mag_raw	sensor_msgs/MagneticField	[Optional] Magnetometer, affected by magnetic field
Published	/imu/data	sensor_msgs/Imu	Fused Imu information.

3) Parameters

Parameter name	Type	Default value	Explanation
~gain	double	0.1	Gain of the filter. Higher values lead to faster convergence but more noise. Lower values lead to slower convergence but smoother signals. Range: 0.0 to 1.0
~zeta	double	0.0	Gyro drift gain (approximately rad/s). Range: -1.0 to 1.0
~mag_bias_x	double	0.0	Magnetometer bias (hard iron correction), x component. Range: -10.0 to 10.0
~mag_bias_y	double	0.0	Magnetometer bias (hard iron correction), y component. Range: -10.0 to 10.0
~mag_bias_z	double	0.0	Magnetometer bias (hard iron correction), z-component. Range: -10.0 to 10.0
~orientation_stddev	double	0.0	Standard deviation of the orientation estimate. Range: 0.0 to 1.0
~world_frame	string	"nwu"	World frame indicating orientation (see REP-145). The old default was "nwu" (northwest up). New deployments should use "enu". Valid values: "nwu", "enu", "ned".
~use_mag	bool	true	Whether to use magnetic field data in data fusion.
~use_magnetic_field_msg	bool	false	If set to true, subscribe to /imu, /mag topics as sensor_msgs/MagneticField; if set to false (not recommended), use geometry_msgs/Vector3Stamped
~fixed_frame	string	odom	Parent frame to use in publishing
~publish_tf	bool	false	Whether to publish the TF transform representing the IMU orientation as the IMU pose; use the fixed frame as the parent frame, and the input imu information as the child frame
~reverse_tf	bool	false	If set to true, publish the transform from the imu frame to the fixed frame, not the other way around.

Parameter name	Type	Default value	Explanation
~constant_dt	double	0.0	dt to use; if 0.0 (default), the dt dynamic value is calculated from the message start position.
~publish_debug_topics	bool	false	If set to true, publish two debug topics.
~stateless	bool	false	If set to true, do not publish the filtered orientation. Instead, publish only a stateless estimate of the orientation based on the latest accelerometer (and optionally magnetometer) readings. Useful for debugging.
~remove_gravity_vector	bool	false	If set to true, subtract the gravity vector from the acceleration field in the published IMU message.

3.1.4, robot_localization

1), Introduction

`robot_localization` is a collection of state estimation nodes, each of which is an implementation of a nonlinear state estimator for robots moving in 3D space. It includes two state estimation nodes `ekf_localization_node` and `ukf_localization_node`. In addition, `robot_localization` provides `navsat_transform_node`, which helps integrate GPS data.

`ekf_localization_node` is an implementation of the Extended Kalman Filter. It uses an omnidirectional motion model to predict the state in time, and uses the sensed sensor data to correct the predicted estimate.

`ukf_localization_node` is an implementation of the Unscented Kalman Filter. It uses a set of carefully chosen sigma points to project the state through the same motion model used in the EKF, and then uses these projected sigma points to recover the state estimate and covariance. This eliminates the use of the Jacobian matrix and makes the filter more stable. However, it is also more computationally heavy than the `ekf_localization_node`.

2), Topic

/	Topic name	Type	Parsing
Subscribed	/imu/data	sensor_msgs/Imu	Filtered imu information
Subscribed	/odom_wheel	nav_msgs/Odometry	Odometry information
Published	/odom	nav_msgs/Odometry	Fused odometry information
Published	/tf	tf2_msgs/TFMessage	Coordinate system information

3), Parameters

- frequency: The actual frequency (in Hz) at which the filter generates state estimates. **Note: The filter will only start calculating after receiving at least one message from one of**

the inputs.

- [sensor]: For each sensor, the user needs to define this parameter according to the message type. Each parameter name is indexed starting at 0 (e.g. odom0, odom1, etc.) and must be defined in order (e.g. do not use pose0 and pose2 if pose1 has not been defined yet). The value of each parameter is the topic name for that sensor.

```
odom0: /odom_wheel
imu0: /imu/data
```

- [sensor]_differential: For each sensor message defined above containing pose information, the user can specify whether the pose variable should be differentially integrated. If the given value is set to true, for a measurement taken at time t from the associated sensor, we will first subtract the measurement at time t-1 and then convert the resulting value to velocity.

```
~odomN_differential
~imuN_differential
~poseN_differential
```

- [sensor]_relative: If this parameter is set to true, any measurements from this sensor will be fused relative to the first measurement received from this sensor. This is useful, for example, if you want your state estimate to always start at (0,0,0) with roll, pitch, and yaw values of (0,0,0).

```
~odomN_relative
~imuN_relative
~poseN_relative
```

- two_d_mode: Set this to true if your robot is operating in a planar environment and can ignore slight variations in the ground surface (as reported by the IMU). It fuses all 3D variables (Z, roll, pitch, and their respective velocities and accelerations) to a value of 0. This ensures that the covariance of these values does not explode, while ensuring that your robot's state estimate remains fixed to the X-Y plane.
- odom0_config: [false, false, false, false, false, false, true, true, false, false, false, true, false, false, false]

The order of boolean values is: [[X], [Y], [Z], [roll], [pitch], [yaw], [X'], [Y'], [Z'], [roll'], [pitch'], [yaw'], [X''], [Y''], [Z'']]. The user must specify which variables of these messages should be fused into the final state estimate.

4) Published transformations

If the user's `world_frame` parameter is set to the value of `odom_frame`, the transformation is published from the coordinate system given by the `odom_frame` parameter to the coordinate system given by the `base_link_frame` parameter. If the user's `world_frame` parameter is set to the value of `map_frame`, the transformation is published from the coordinate system given by the `map_frame` parameter to the coordinate system given by the `odom_frame` parameter.

For example, we set the transformation to be published from the coordinate system given by the [odom_frame] parameter to the coordinate system given by the [base_link_frame] parameter.

```
odom_frame: odom
base_link_frame: base_footprint
world_frame: odom
```