Robot state estimation

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be consistent. You can check [Must-read before use] to set the IP and ROS_DOMAIN_ID on the board.

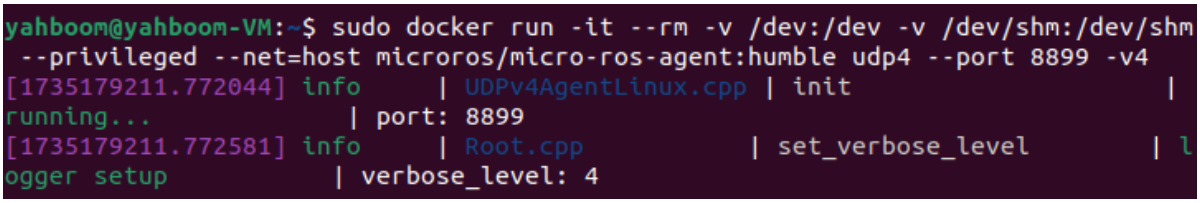
1. Program function description

After the program is started, it will subscribe to imu and odom data, filter out part of the imu data, and then merge it with the odom data, and finally output a fused odom data to estimate the robot's state. This data is mostly used in mapping and navigation.

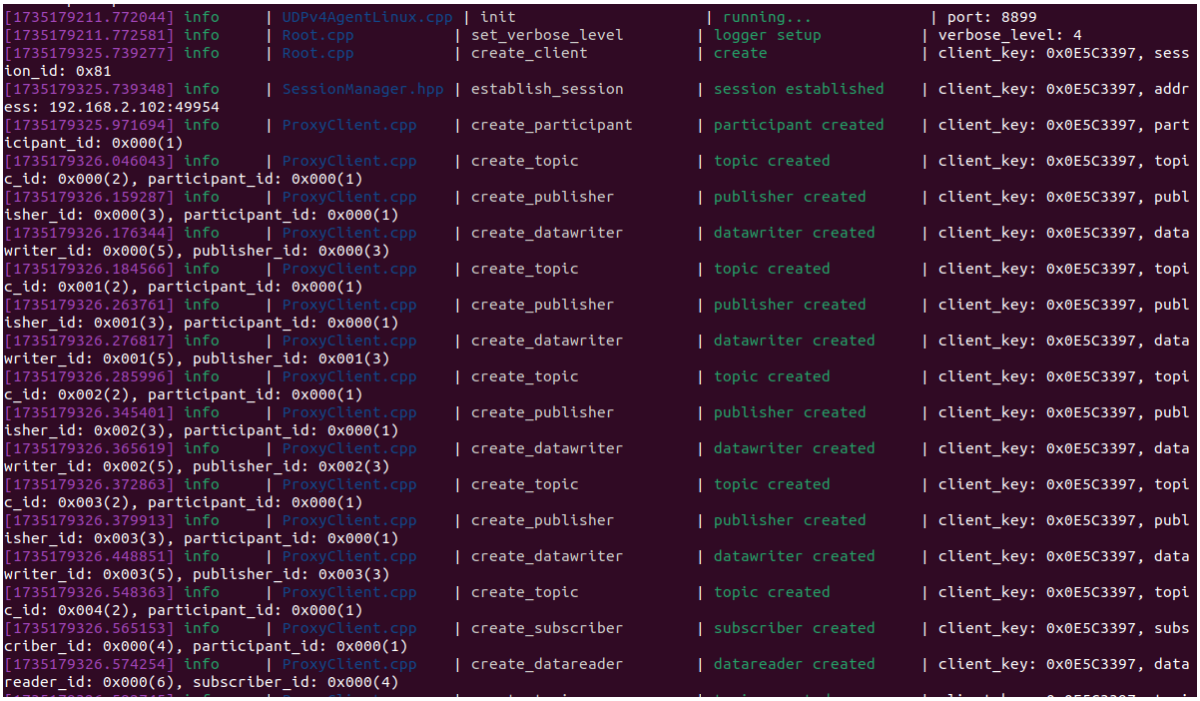
2. Start and connect the agent

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
```



Then, turn on the car switch and wait for the car to connect to the agent. The connection is successful as shown in the figure below,

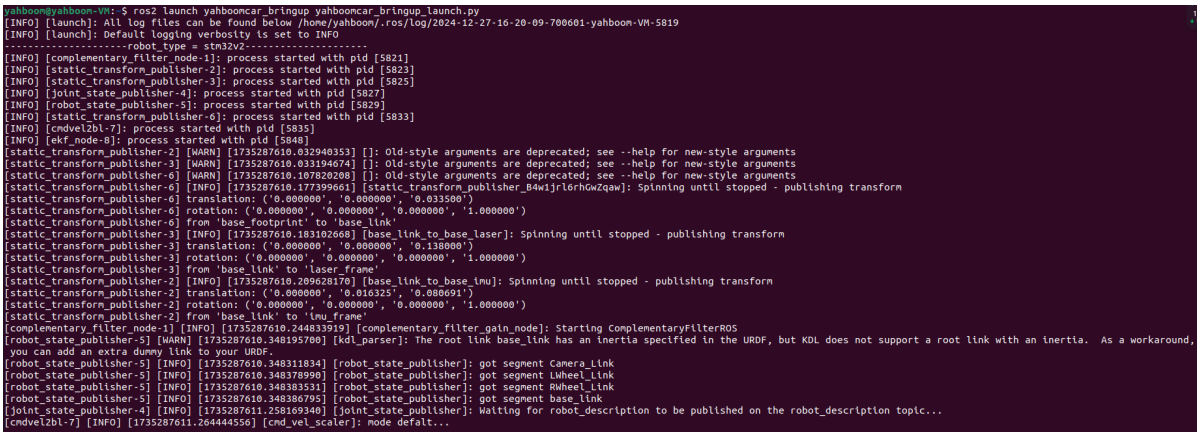


3. Start the program

3.1 Run command

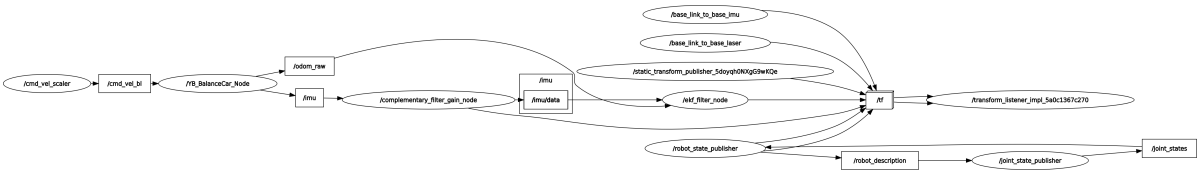
Take the matching virtual machine as an example, input in the terminal,

```
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```



Enter the following command to view the communication graph between nodes,

```
ros2 run rqt_graph rqt_graph
```



If it is not displayed at the beginning, select [Nodes/Topics(all)] and click the refresh button in the upper left corner.

The fused node is /ekf_filter_node. You can query the relevant information of this node by inputting in the terminal,

```
ros2 node info /ekf_filter_node
```

```
yahboom@yahboom-VM:~$ ros2 node info /ekf_filter_node
/ekf_filter_node
  Subscribers:
    /imu/data: sensor_msgs/msg/Imu
    /odom_raw: nav_msgs/msg/Odometry
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /set_pose: geometry_msgs/msg/PoseWithCovarianceStamped
  Publishers:
    /diagnostics: diagnostic_msgs/msg/DiagnosticArray
    /odom: nav_msgs/msg/Odometry
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /tf: tf2_msgs/msg/TFMessage
  Service Servers:
    /ekf_filter_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /ekf_filter_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /ekf_filter_node/get_parameters: rcl_interfaces/srv/GetParameters
    /ekf_filter_node/list_parameters: rcl_interfaces/srv/ListParameters
    /ekf_filter_node/set_parameters: rcl_interfaces/srv/SetParameters
    /ekf_filter_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
    /enable: std_srvs/srv/Empty
    /reset: std_srvs/srv/Empty
    /set_pose: robot_localization/srv/SetPose
    /toggle: robot_localization/srv/ToggleFilterProcessing
  Service Clients:

  Action Servers:

  Action Clients:
```

Combined with the node communication diagram above, it can be seen that the node subscribes to /imu/data and /odom_raw data, and then publishes a /odom data.

4. Parse the launch file

Take the matching virtual machine as an example code path:

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_bringup/launch
```

yahboomcar_bringup_launch.py

```
from ament_index_python.packages import get_package_share_path
from launch import LaunchDescription
from launch_ros.actions import Node
import os
from ament_index_python.packages import get_package_share_directory
from launch.actions import IncludeLaunchDescription,DeclareLaunchArgument
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration

print("----------------------robot_type = stm32v2----------------------")

def generate_launch_description():
    imu_filter_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('imu_complementary_filter'), 'launch'),
            '/complementary_filter.launch.py'])
    )

    ekf_params_file = LaunchConfiguration('ekf_params_file')
    ekf_node = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
```

```
            get_package_share_directory('robot_localization'), 'launch'),
            '/ekf.launch.py']),
        launch_arguments={'ekf_params_file': ekf_params_file}.items()
    )

    description_launch = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('yahboomcar_description'), 'launch'),
            '/description_launch.py'])
    )
    base_link_to_imu_tf_node = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        name='base_link_to_base_imu',
        #arguments=['0.0', '0.016325', '0.080691', '0', '0', '0', 'base_link',
'mpu6050_frame']
        arguments=['0.0', '0.016325', '0.080691', '0', '0', '0', 'base_link',
'imu_frame']
    )

    base_link_to_laser_tf_node = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        name='base_link_to_base_laser',
        arguments=['0.0', '0.0', '0.138', '0', '0', '0', 'base_link',
'laser_frame']
    )

    #rf2o_laser_odometry
    rf2o_laser_odometry_launch = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('rf2o_laser_odometry'), 'launch'),
            '/rf2o_laser_odometry.launch.py'])
    )
    odom_to_base_footprint_node = Node(
        package='yahboomcar_bringup',
        executable='base_node',
        name='odom_to_base_footprint',
    )
    cmd_vel_scaler_node = Node(
        package='yahboomcar_bringup',
        executable='cmdvel2bl',
        name='cmd_vel_scaler',
        parameters=[{'mode': LaunchConfiguration('mode')}]
    )

    return LaunchDescription([
        imu_filter_node,
        base_link_to_imu_tf_node,
        base_link_to_laser_tf_node,
        description_launch,
        DeclareLaunchArgument(
        'mode',
        default_value='default',
        description='mode:=default or appslam,nav,slam'
        ),
```

```
        cmd_vel_scaler_node,
        #rf2o_laser_odometry_launch,
        DeclareLaunchArgument(
            'ekf_params_file',

 default_value=os.path.join(get_package_share_directory("robot_localization"),
'params', 'ekf3.yaml'),
        ),
        ekf_node,
    ])
```

The launch file starts the following nodes:

- imu_filter_node: filter imu data node, mainly filter part of imu data;
- ekf_node: fusion node, mainly fuse odom data and filtered imu data.
- base_link_to_imu_tf_node: publish a static change, mainly publish the pose transformation of the imu module and the car.
- base_link_to_laser_tf_node: publish a static change, mainly publish the pose transformation of the radar module and the car.
- description_launch: load URDF model.
- cmd_vel_scaler_node: cmd_vel conversion node, mainly convert the car speed cmd_vel into cmd_vel_bl topic.