2.Handheld lidar mapping

This function needs to start the program in the slam_gmapping function package. The source code is located in the [ydlidar_ws] source code file.

Here we use the supporting virtual machine to explain how to start the program. If you want to put it on your own motherboard, you need to put [slam_gmapping] and [openslam_gmapping] in the src directory of the workspace to compile.

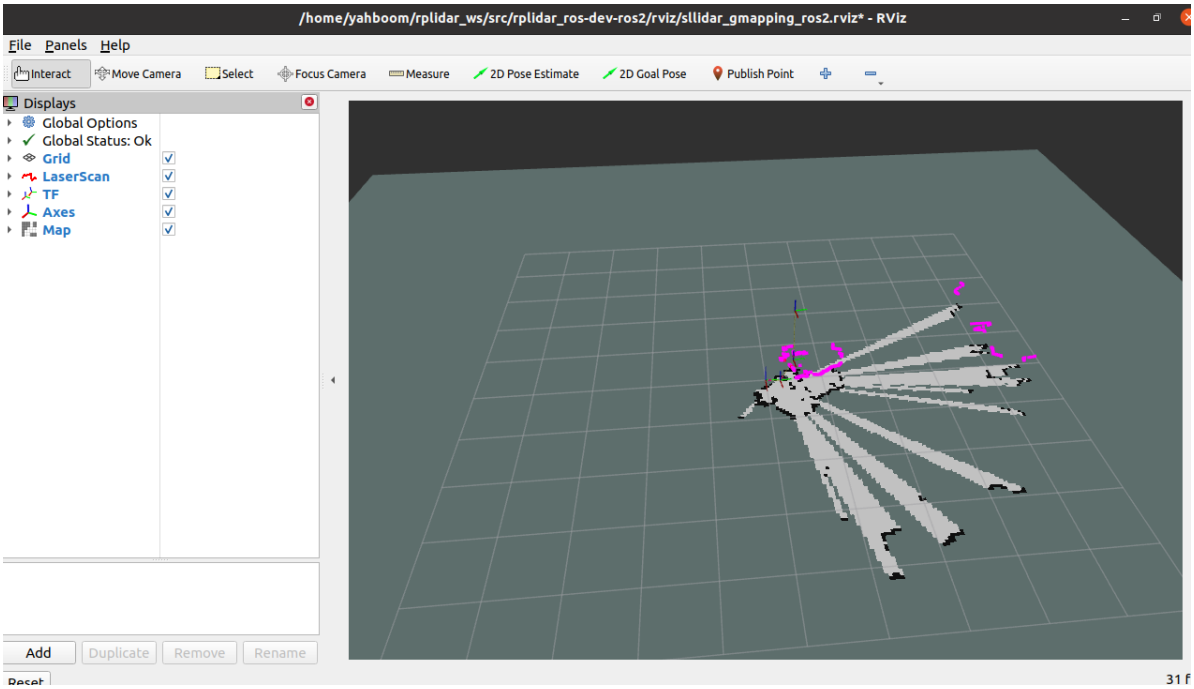
2.1 Start gmapping mapping

Input following command:

```
ros2 launch ydlidar_ros2_driver test_gmapping_launch.py
#x3 lidar
ros2 launch slam_gmapping gmapping_x3_launch.py
#4ros lidar
ros2 launch slam_gmapping gmapping_4ros_launch.py
```

```
[slam_gmapping-2] m_count 0
[slam_gmapping-2] Registering First Scan
[slam_gmapping-2] [INFO] [1699613716.598416796] [slam_gmapping]: Initialization
complete
[slam_gmapping-2] Laser Pose= 0.1 0.2 0
[rviz2-5] [INFO] [1699613716.648205807] [rviz2]: Trying to create a map of size
384 x 384 using 1 swatches
[rviz2-5] [ERROR] [1699613716.652811756] [rviz2]: Vertex Program:rviz/glsl120/in
dexed_8bit_image.vert Fragment Program:rviz/glsl120/indexed_8bit_image.frag GLSL
 link result :
[rviz2-5] active samplers with a different type refer to the same texture image
unit
[slam_gmapping-2] update frame 15
[slam_gmapping-2] update ld=9.3095e-17 ad=0
[slam_gmapping-2] m_count 1
[slam_gmapping-2] Laser Pose= 0.1 0.2 0
[slam_gmapping-2] Average Scan Matching Score=984.978
[slam_gmapping-2] neff= 30
[slam_gmapping-2] Registering Scans:Done
[slam_gmapping-2] update frame 35
[slam_gmapping-2] update ld=0 ad=0
[slam_gmapping-2] m_count 2
[slam_gmapping-2] Laser Pose= 0.1 0.2 0
```

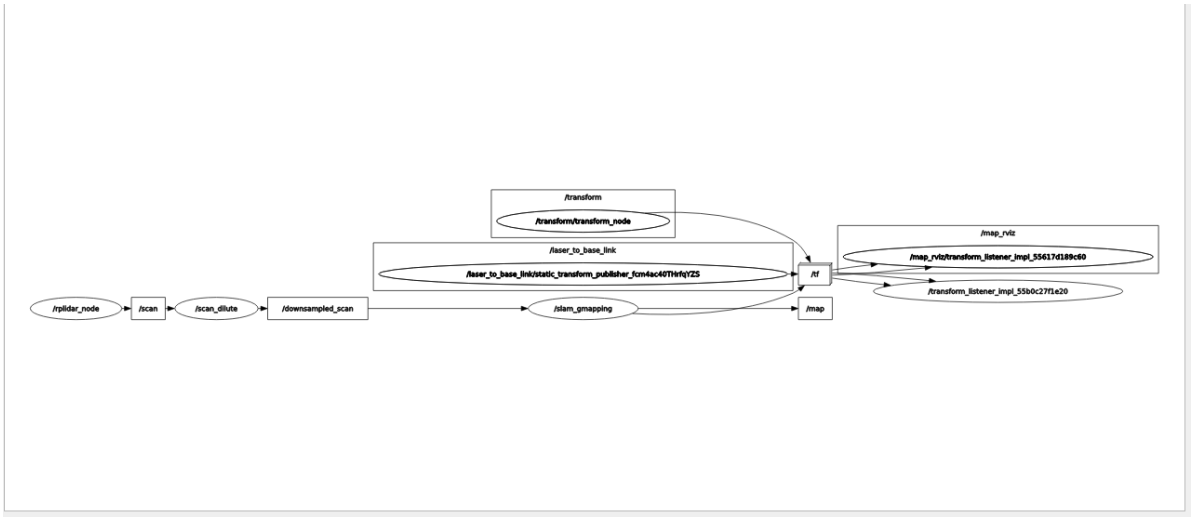
rviz displays as follows:



2.2 View node communication

Input following command:

```
ros2 run rqt_graph rqt_graph
```



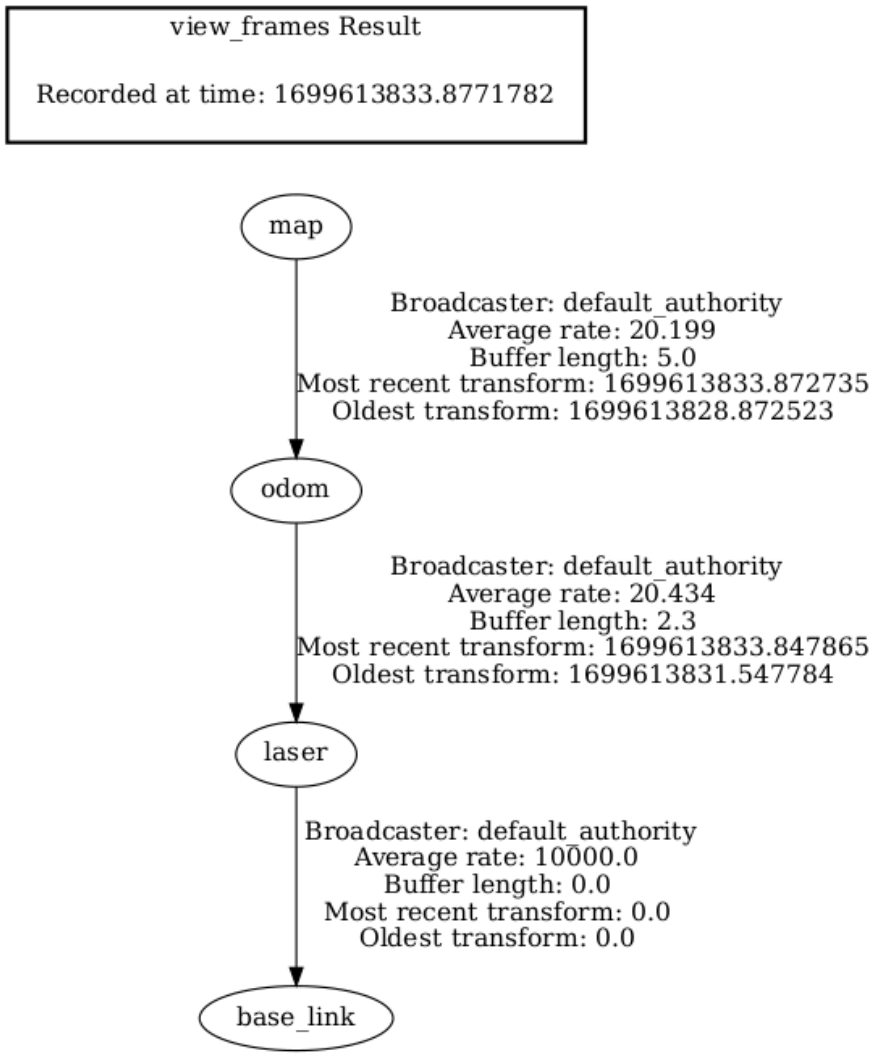
2.3 View TF tree

Input following command:

```
ros2 run tf2_tools view_frames.py
```

```
yahboom@VM:~$ ros2 run tf2_tools view_frames.py
[INFO] [1699500741.610795137] [view_frames]: Listening to tf data during 5 seconds...
[INFO] [1699500746.652813123] [view_frames]: Generating graph in frames.pdf file...
[INFO] [1699500746.654506796] [view_frames]: Result:tf2_msgs.srv.FrameGraph_Response(frame_yaml="odom: \n  parent: 'map'\n  broadcaster: 'default_authority'\n  rate: 20.195\n  most_recent_transform: 1699500746.651993\n  oldest_transform: 1699500741.601316\n  buffer_length: 5.051\nlidar: \n  parent: 'base_link'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_recent_transform: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0.000\nbase_link: \n  parent: 'laser'\n  broadcaster: 'default_authority'\n  rate: 10000.000\n  most_recent_transform: 0.000000\n  oldest_transform: 0.000000\n  buffer_length: 0.000\nlaser: \n  parent: 'odom'\n  broadcaster: 'default_authority'\n  rate: 20.200\n  most_recent_transform: 1699500746.604301\n  oldest_transform: 1699500741.604318\n  buffer_length: 5.000\n")
yahboom@VM:~$
```

In the directory where the command terminal is started, the system will generate a frames.pdf file, which is the generated TF tree.



2.4 launch file analysis

File path (combined with virtual machine):

```
/home/yahboom/ydlidar_ws/src/rplidar_ros-dev-ros2/launch/test_gmapping.launch.py
```

test_gmapping.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node
import os
from launch.actions import IncludeLaunchDescription
from launch.conditions import
LaunchConfigurationEquals,LaunchConfigurationNotEquals
from launch.launch_description_sources import
PythonLaunchDescriptionSource,AnyLaunchDescriptionSource
from ament_index_python.packages import get_package_share_directory
from launch.actions import DeclareLaunchArgument
import os

def generate_launch_description():
    LIDAR_TYPE = os.getenv('LIDAR_TYPE')
```

```
    lidar_type_arg = DeclareLaunchArgument(name='lidar_type',
default_value=LIDAR_TYPE, description='The type of lidar')

    lidar_launch =
IncludeLaunchDescription(PythonLaunchDescriptionSource([os.path.join(get_package
_share_directory('rplidar_ros'), 'launch'),'/lidar.launch.py']))

    s2_gmapping_launch =
IncludeLaunchDescription(PythonLaunchDescriptionSource([os.path.join(get_package
_share_directory('slam_gmapping'),
'launch'),'/s2_gmapping_launch.py']),condition=LaunchConfigurationEquals('lidar_
type', 's2'))

    gmapping_launch =
IncludeLaunchDescription(PythonLaunchDescriptionSource([os.path.join(get_package
_share_directory('slam_gmapping'),
'launch'),'/gmapping_launch.py']),condition=LaunchConfigurationNotEquals('lidar_
type', 's2'))

    s2_slam_gmapping_launch =
IncludeLaunchDescription(PythonLaunchDescriptionSource([os.path.join(get_package
_share_directory('slam_gmapping'),
'launch'),'/s2_slam_gmapping.launch.py']),condition=LaunchConfigurationEquals('l
idar_type', 's2'))

    slam_gmapping_launch =
IncludeLaunchDescription(PythonLaunchDescriptionSource([os.path.join(get_package
_share_directory('slam_gmapping'),
'launch'),'/slam_gmapping.launch.py']),condition=LaunchConfigurationNotEquals('l
idar_type', 's2'))


    return LaunchDescription([
        lidar_type_arg,
        lidar_launch,
        s2_gmapping_launch,
        s2_slam_gmapping_launch,
        gmapping_launch,
        slam_gmapping_launch
    ])
```

[lidar_type_arg] Gets the value of [LIDAR_TYPE] set in the environment variable, and starts the corresponding radar and corresponding mapping instructions based on this value.

[lidar_launch] launch radar launch file, the file is located at `/home/yahboom/ydlidar_ws/src/rplidar_ros-dev-ros2/launch/lidar.launch.py`

[slam_4ros_gmapping_launch] Release some necessary static TF transformations for other lidar mapping and start rviz. Here you need to remap the radar topic into a filtered radar topic.

The file is located

```
/home/yahboom/ydlidar_ws/src/slam_gmapping/launch/slam_4ros_gmapping.launch.py
```

[slam_x3_gmapping_launch] Release some necessary static TF transformations and start rviz for other lidar mapping, the file is located

`/home/yahboom/ydlidar_ws/src/slam_gmapping/launch/slam_x3_gmapping.launch.py`

gmapping_x3_launch.py

```
#!/usr/bin/env python3

import os
import launch
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch_ros.actions import Node


def generate_launch_description():
    use_sim_time = launch.substitutions.LaunchConfiguration('use_sim_time',
default='False')
    return LaunchDescription([

        Node(
            package='slam_gmapping',
            executable='slam_gmapping',
            output='screen',
            parameters=
[os.path.join(get_package_share_directory("slam_gmapping"), "params",
"slam_gmapping.yaml")]
        ),
    ])
```

Start the mapping node, the parameter file slam_gmapping.yaml path is `/home/yahboom/ydlidar_ws/src/slam_gmapping/params/slam_gmapping.yaml`

gmapping_4ros_launch.py

```
#!/usr/bin/env python3

import os
import launch
from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch_ros.actions import Node


def generate_launch_description():
    use_sim_time = launch.substitutions.LaunchConfiguration('use_sim_time',
default='False')
    return LaunchDescription([

        Node(
            package='slam_gmapping',
            executable='slam_gmapping',
            output='screen',
```

```
            parameters=
[os.path.join(get_package_share_directory("slam_gmapping"), "params",
"slam_gmapping.yaml")],
            remappings = [("/scan","/downsampled_scan")],
        ),
    ])
```

Start the mapping node. The path of the parameter file slam_gmapping.yaml is `/home/yahboom/ydlidar_ws/src/slam_gmapping/params/slam_gmapping.yaml`.

Here you also need to map the lidar topic to the filtered lidar topic/downsampled_scan.