ROS2 Multi-Sensor Robot Setup

Technical Documentation

Unified Workspace Configuration for
Jetson Orin Nano with LD-19 LiDAR and RealSense D435

Version 1.0

Author: Srinjay

December 17, 2025

Platform: ROS2 Humble
Hardware: Jetson Orin Nano, LD-19 LiDAR, Intel RealSense D435

Contents

1 Introduction

This document provides a comprehensive guide to setting up a unified ROS2 workspace for a multi-sensor robot system on Jetson Orin Nano. The system integrates:

- **LD-19 LiDAR** - 360° 2D laser scanner from Waveshare
- **Intel RealSense D435** - RGB-D depth camera
- **Robot Description (URDF)** - Unified robot model with TF frames
- **Centralized Bringup System** - Single launch file for all sensors

1.1 System Architecture Overview

Information

The architecture follows ROS2 best practices with separate packages for:

- **robot_description** - URDF and robot model
- **robot_bringup** - Launch files and configurations
- **Sensor drivers** - LD-19 and RealSense packages

1.2 Prerequisites

Before starting, ensure you have:

- Jetson Orin Nano with Ubuntu 20.04/22.04
- ROS2 Humble installed
- LD-19 LiDAR connected via USB (appears as `/dev/ttyUSB0`)
- Intel RealSense D435 connected via USB 3.0
- Basic knowledge of ROS2 concepts (nodes, topics, TF)

2 Workspace Creation

2.1 Step 1: Create the Unified Workspace

First, create a ROS2 workspace that will house all robot packages:

Listing 1: Create workspace directory

```
mkdir -p ~/robot_ws/src
cd ~/robot_ws
colcon build
```

2.2 Step 2: Source the Workspace

Add the workspace to your environment:

Listing 2: Source workspace

```
source ~/robot_ws/install/setup.bash
```

Information

To automatically source the workspace on every terminal session, add the following line to your `~/.bashrc`:
`source ~/robot_ws/install/setup.bash`

2.3 Step 3: Verify Workspace Setup

Listing 3: Verify workspace

```
echo $ROS_PACKAGE_PATH
```

You should see your workspace path in the output.

3 LiDAR Integration

3.1 Step 1: Clone LD-19 Driver

Navigate to the workspace source directory and clone the Waveshare LD-19 driver:

Listing 4: Clone LD-19 LiDAR driver

```
cd ~/robot_ws/src
git clone https://github.com/waveshare/ld19_lidar_ros2.git
```

Information

The actual repository name may be `ldlidar_stl_ros2` depending on the version. Use the repository provided in the Waveshare tutorial.

3.2 Step 2: Install Dependencies

Install ROS2 dependencies for the LiDAR package:

Listing 5: Install dependencies

```
cd ~/robot_ws
rosdep install --from-paths src --ignore-src -r -y
```

3.3 Step 3: Build the LiDAR Package

Listing 6: Build workspace

```
colcon build
source install/setup.bash
```

3.4 Step 4: Test LiDAR Manually

Before integrating into the bringup system, verify the LiDAR works:

Listing 7: Test LiDAR node

```
ros2 run ldlidar_stl_ros2 ldlidar_stl_ros2_node \
  --ros-args \
  -p product_name:=LDLiDAR_LD19 \
  -p topic_name:=LiDAR/LD19 \
  -p frame_id:=base_laser \
  -p port_name:=/dev/ttyUSB0 \
  -p port_baudrate:=230400
```

Success

If successful, you should see:

```
[INFO] [ldlidar_published]: ldlidar node start is success
[INFO] [ldlidar_published]: ldlidar communication is normal.
```

3.5 Step 5: Verify LiDAR Topic

In a new terminal:

Listing 8: Check LiDAR topic

```
source ~/robot_ws/install/setup.bash
ros2 topic list | grep LiDAR
ros2 topic echo /LiDAR/LD19
```

4 RealSense Camera Integration

4.1 Step 1: Install RealSense ROS2 Package

The easiest method is to install the pre-built package:

Listing 9: Install RealSense ROS2 package

```
sudo apt install ros-humble-realsense2-camera
```

Information

Alternative: Build from Source
If you need the latest features or custom modifications:

```
cd ~/robot_ws/src
git clone https://github.com/IntelRealSense/realsense-ros.git \
  -b ros2-development
cd ~/robot_ws
colcon build
source install/setup.bash
```

4.2 Step 2: Test RealSense Camera

Listing 10: Test RealSense node

```
ros2 launch realsense2_camera rs_launch.py
```

4.3 Step 3: Verify Camera Topics

In a new terminal:

Listing 11: Check camera topics

```
source ~/robot_ws/install/setup.bash
ros2 topic list | grep camera
```

Expected topics include:

- `/camera/color/image_raw`
- `/camera/depth/image_rect_raw`
- `/camera/depth/color/points`

5 Robot Description (URDF)

5.1 Step 1: Create robot_description Package

Listing 12: Create robot_description package

```
cd ~/robot_ws/src
ros2 pkg create robot_description --build-type ament_cmake
cd robot_description
mkdir urdf
```

5.2 Step 2: Create URDF File

Create the robot URDF file:

Listing 13: Create URDF file

```
nano urdf/robot.urdf
```

Add the following URDF content:

Listing 14: robot.urdf - Robot description

```
<?xml version="1.0"?>
<robot name="my_robot">

  <!-- Base Link -->
  <link name="base_link"/>

  <!-- LiDAR Link -->
  <link name="base_laser"/>
  <joint name="laser_joint" type="fixed">
    <parent link="base_link"/>
    <child link="base_laser"/>
    <!-- Adjust xyz values to match your robot's geometry -->
    <origin xyz="0.0 0.0 0.2" rpy="0 0 0"/>
  </joint>

  <!-- Camera Link -->
  <link name="camera_link"/>
  <joint name="camera_joint" type="fixed">
    <parent link="base_link"/>
    <child link="camera_link"/>
    <!-- Adjust xyz values to match your robot's geometry -->
    <origin xyz="0.15 0.0 0.25" rpy="0 0 0"/>
  </joint>

</robot>
```

Warning

Important Frame Names:

- `base_laser` - Must match LiDAR's frame_id
- `camera_link` - Must match RealSense's frame_id
- `base_link` - Common parent frame for all sensors

5.3 Step 3: Configure CMakeLists.txt

Edit `CMakeLists.txt` to install the URDF:

Listing 15: Edit CMakeLists.txt

```
nano CMakeLists.txt
```

Add before `ament_package()`:

Listing 16: CMakeLists.txt installation rule

```
install(
  DIRECTORY urdf
  DESTINATION share/${PROJECT_NAME}
)
```

5.4 Step 4: Update package.xml

Add the following dependencies to `package.xml`:

Listing 17: package.xml dependencies

```
<exec_depend>robot_state_publisher</exec_depend>
<exec_depend>urdf</exec_depend>
```

5.5 Step 5: Build robot_description

Listing 18: Build robot_description package

```
cd ~/robot_ws
colcon build --packages-select robot_description
source install/setup.bash
```

5.6 Step 6: Test Robot State Publisher

Listing 19: Test robot_state_publisher

```
ros2 run robot_state_publisher robot_state_publisher \
  --ros-args -p robot_description:="$(cat ~/robot_ws/src/
      robot_description/urdf/robot.urdf)"
```

5.7 Step 7: Verify TF Tree

In a new terminal:

Listing 20: Generate TF tree visualization

```
ros2 run tf2_tools view_frames
```

This generates `frames.pdf` showing the TF tree. You should see:

```
base_link
  base_laser
  camera_link
       camera_depth_frame
       camera_color_frame
```

6 Robot Bringup Package

6.1 Step 1: Create robot_bringup Package

Listing 21: Create robot_bringup package

```
cd ~/robot_ws/src
ros2 pkg create robot_bringup --build-type ament_python
cd robot_bringup
mkdir launch config
```

6.2 Step 2: Create LiDAR Configuration

Create the LiDAR parameter file:

Listing 22: Create ld19.yaml

```
nano config/ld19.yaml
```

Add the following configuration:

Listing 23: ld19.yaml - LiDAR configuration

```
ld19_config:
  product_name: LDLiDAR_LD19
  topic_name: LiDAR/LD19
  frame_id: base_laser
  port_name: /dev/ttyUSB0
  port_baudrate: 230400
  laser_scan_dir: false
  enable_angle_crop_func: false
  angle_crop_min: 0.0
  angle_crop_max: 0.0
```

Warning

Critical Parameter Notes:

- `product_name` must be exactly `LDLiDAR_LD19`
- `laser_scan_dir` is boolean: `false` = counterclockwise
- `port_name` must match your actual device (check with `ls /dev/ttyUSB*`)

6.3 Step 3: Create Bringup Launch File

Create the main launch file:

Listing 24: Create bringup.launch.py

```
nano launch/bringup.launch.py
```

Add the following Python launch code:

Listing 25: bringup.launch.py - Main robot launch file

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

```
from launch.substitutions import Command
import os
import yaml

def generate_launch_description():

    # Robot URDF path
    robot_description_path = os.path.join(
        get_package_share_directory('robot_description'),
        'urdf',
        'robot.urdf'
    )

    # Load LiDAR parameters from YAML
    ld19_config_file = os.path.join(
        get_package_share_directory('robot_bringup'),
        'config',
        'ld19.yaml'
    )
    with open(ld19_config_file, 'r') as f:
        ld19_params = yaml.safe_load(f)['ld19_config']

    return LaunchDescription([

        # Robot State Publisher (publishes TF from URDF)
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            parameters=[{
                'robot_description': Command(['cat ',
                   robot_description_path])
            }]
        ),

        # LD-19 LiDAR Node
        Node(
            package='ldlidar_stl_ros2',
            executable='ldlidar_stl_ros2_node',
            name='lidar',
            output='screen',
            parameters=[ld19_params]
        ),

        # RealSense D435 Camera Node
        Node(
            package='realsense2_camera',
            executable='realsense2_camera_node',
            name='camera',
            output='screen',
            parameters=[{
                'enable_depth': True,
                'enable_color': True,
                'pointcloud.enable': False
            }]
        ),
    ])
```

6.4 Step 4: Configure setup.py

Edit `setup.py` to install launch files and configs:

Listing 26: Edit setup.py

```
nano setup.py
```

Modify the `data_files` section:

Listing 27: setup.py - Data files configuration

```
from setuptools import setup
from glob import glob
import os

package_name = 'robot_bringup'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        ('share/' + package_name + '/launch',
            glob('launch/*.launch.py')),
        ('share/' + package_name + '/config',
            glob('config/*.yaml')),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='your_name',
    maintainer_email='your_email@example.com',
    description='Robot bringup package',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [],
    },
)
```

Warning

The `data_files` section is critical. Without it, launch files and config files won't be installed and the system will fail with "file not found" errors.

6.5 Step 5: Build robot_bringup

Listing 28: Build robot_bringup package

```
cd ~/robot_ws
colcon build --packages-select robot_bringup
source install/setup.bash
```

6.6 Step 6: Verify Installation

Check that files are installed correctly:

Listing 29: Verify installed files

```
ls install/robot_bringup/share/robot_bringup/launch/
ls install/robot_bringup/share/robot_bringup/config/
```

You should see:

- `launch/bringup.launch.py`
- `config/ld19.yaml`

7 System Launch and Verification

7.1 Step 1: Launch the Complete System

Start all sensors with a single command:

Listing 30: Launch robot bringup

```
ros2 launch robot_bringup bringup.launch.py
```

Success

Expected output:

```
[INFO] [robot_state_publisher]: got segment base_link
[INFO] [robot_state_publisher]: got segment base_laser
[INFO] [robot_state_publisher]: got segment camera_link
[INFO] [ldlidar_published]: ldlidar node start is success
[INFO] [camera.camera]: RealSense Node Is Up!
```

7.2 Step 2: Verify Topics

In a new terminal, check all active topics:

Listing 31: List all topics

```
source ~/robot_ws/install/setup.bash
ros2 topic list
```

Expected topics:

- `/LiDAR/LD19` - LiDAR scan data
- `/camera/color/image_raw` - RGB image
- `/camera/depth/image_rect_raw` - Depth image
- `/tf` - Dynamic transforms
- `/tf_static` - Static transforms

7.3 Step 3: Verify TF Frames

Listing 32: Check TF tree

```
ros2 run tf2_tools view_frames
```

The generated PDF should show a unified TF tree with `base_link` as the root.

7.4 Step 4: Visualize in RViz2

Launch RViz2:

Listing 33: Launch RViz2

```
rviz2
```

Configure RViz2:

1. Set **Fixed Frame** to `base_link`

2. Add **TF** display
3. Add **LaserScan** display → Set topic to `/LiDAR/LD19`
4. Add **Camera** display → Set image topic to `/camera/color/image_raw`

Success

Both LiDAR and camera should now be visible in RViz2 simultaneously, properly aligned in 3D space.

8 Troubleshooting

8.1 LiDAR Issues

8.1.1 Error: "product_name is illegal"

Error

Error Message:

```
[ERROR] [lidar]: Error, input <product_name> is illegal.
```

Cause: Incorrect or missing `product_name` parameter.
Solution: Verify `ld19.yaml` has:

```
product_name: LDLiDAR_LD19
```

8.1.2 Error: "No such file or directory: /dev/ttyUSB0"

Cause: LiDAR not connected or appears on different port.
Solution:

```
ls /dev/ttyUSB*
```

Update `port_name` in `ld19.yaml` to match the actual device.

8.1.3 Error: "parameter has invalid type"

Error

Error Message:

```
parameter 'laser_scan_dir' has invalid type:
Wrong parameter type, parameter {laser_scan_dir}
is of type {bool}, setting it to {string} is not allowed.
```

Cause: `laser_scan_dir` set as string instead of boolean.
Solution: Change to boolean in `ld19.yaml`:

```
laser_scan_dir: false  # not "Counterclockwise"
```

8.2 RealSense Camera Issues

8.2.1 Warning: "USB 2.1 port detected"

Warning

Warning Message:

```
Device is connected using a 2.1 port.
Reduced performance is expected.
```

Cause: Camera connected to USB 2.0 port instead of USB 3.0.
Solution: Connect RealSense D435 to a USB 3.0 port (blue port) for full performance.

8.3 TF and RViz Issues

8.3.1 Problem: Only LiDAR or camera visible, not both

Cause: No TF connection between `base_link` and sensor frames.
Solution:

1. Verify robot_state_publisher is running
2. Check TF tree: `ros2 run tf2_tools view_frames`
3. Ensure URDF has correct frame names
4. Set RViz Fixed Frame to `base_link`

8.3.2 Problem: "Transform timeout" in RViz

Solution: Increase TF timeout in RViz TF display settings to 15 seconds.

8.4 Build Issues

8.4.1 Error: "file 'bringup.launch.py' was not found"

Cause: Launch file not installed properly.
Solution:

1. Verify `setup.py` has correct `data_files`
2. Clean build:

```
cd ~/robot_ws
rm -rf build install log
colcon build
source install/setup.bash
```

9 System Architecture Diagram

9.1 Package Structure

```
robot_ws/
 src/
    ldlidar_stl_ros2/          # LiDAR driver
    robot_description/         # URDF and robot model
       urdf/
           robot.urdf
    robot_bringup/             # Launch and config files
        launch/
           bringup.launch.py
        config/
            ld19.yaml
 build/                         # Build artifacts
 install/                       # Installed packages
 log/                           # Build logs
```

9.2 TF Frame Hierarchy

```
map (future SLAM)
  odom (future odometry)
      base_link (robot center)
          base_laser (LiDAR frame)
          camera_link (RealSense frame)
               camera_depth_frame
                   camera_depth_optical_frame
               camera_color_frame
                    camera_color_optical_frame
```

9.3 Node Communication Diagram

Information

Data Flow:

- **ldlidar_stl_ros2_node** → publishes `/LiDAR/LD19` (sensor_msgs/LaserScan)
- **realsense2_camera_node** → publishes camera topics and TF
- **robot_state_publisher** → publishes static TF from URDF
- All nodes → subscribe/publish to `/tf` and `/tf_static`

10 Parameter Reference

10.1 LD-19 LiDAR Parameters

Parameter	Description	Value
product_name	LiDAR model identifier	LDLiDAR_LD19
topic_name	ROS topic for scan data	LiDAR/LD19
frame_id	TF frame for LiDAR	base_laser
port_name	Serial device path	/dev/ttyUSB0
port_baudrate	Serial communication speed	230400
laser_scan_dir	Scan direction (bool)	false (CCW)
enable_angle_crop_func	Enable angle filtering	false
angle_crop_min	Minimum angle (degrees)	0.0
angle_crop_max	Maximum angle (degrees)	0.0

Table 1: LD-19 LiDAR Configuration Parameters

10.2 RealSense Camera Parameters

Parameter	Description	Value
enable_depth	Enable depth stream	true
enable_color	Enable RGB stream	true
pointcloud.enable	Enable point cloud	false
depth_module.depth_profile	Depth resolution	640x480x15
rgb_camera.color_profile	RGB resolution	640x480x15
align_depth.enable	Align depth to color	true

Table 2: RealSense D435 Configuration Parameters

11 Future Extensions

11.1 Adding IMU (BNO055 on Raspberry Pi)

To add an IMU sensor on a separate computer:

1. Set `ROS_DOMAIN_ID` on both machines
2. Install BNO055 driver on Raspberry Pi
3. Configure network discovery
4. Add IMU frame to URDF

11.2 SLAM Integration

Next steps for mapping:

Listing 34: Install SLAM Toolbox

```
sudo apt install ros-humble-slam-toolbox
```

11.3 Robot Localization (EKF)

For sensor fusion:

Listing 35: Install robot_localization

```
sudo apt install ros-humble-robot-localization
```

11.4 Navigation (Nav2)

For autonomous navigation:

Listing 36: Install Nav2

```
sudo apt install ros-humble-navigation2
```

12 Best Practices

12.1 Workspace Management

- Always source workspace after building: `source install/setup.bash`
- Use `colcon build --symlink-install` for faster Python development
- Clean build when changing CMakeLists.txt or package.xml

12.2 Parameter Management

- Store all parameters in YAML files
- Never hardcode parameters in launch files
- Use descriptive parameter names
- Document default values

12.3 TF Best Practices

- Use `base_link` as the robot's center
- Define all static transforms in URDF
- Keep frame names consistent across packages
- Visualize TF tree regularly with `view_frames`

12.4 Debugging Tips

1. Check logs: `ros2 topic echo /rosout`
2. Monitor TF: `ros2 run tf2_ros tf2_monitor`
3. Verify topics: `ros2 topic hz /LiDAR/LD19`
4. Check node status: `ros2 node list`

13 Quick Reference Commands

13.1 Workspace Commands

Listing 37: Common workspace operations

```
# Build entire workspace
cd ~/robot_ws && colcon build

# Build specific package
colcon build --packages-select robot_bringup

# Clean build
rm -rf build install log && colcon build

# Source workspace
source install/setup.bash
```

13.2 Launch Commands

Listing 38: Launch system components

```
# Launch complete system
ros2 launch robot_bringup bringup.launch.py

# Launch RViz
rviz2

# Launch with specific config
ros2 launch robot_bringup bringup.launch.py config:=custom.yaml
```

13.3 Debugging Commands

Listing 39: Debugging and inspection

```
# List all topics
ros2 topic list

# Echo topic data
ros2 topic echo /LiDAR/LD19

# Check topic frequency
ros2 topic hz /LiDAR/LD19

# List all nodes
ros2 node list

# Get node info
ros2 node info /lidar

# Visualize TF tree
ros2 run tf2_tools view_frames

# Monitor specific transform
ros2 run tf2_ros tf2_echo base_link camera_link
```

14 Appendix A: Complete File Listings

14.1 robot.urdf

Listing 40: Complete URDF file

```
<?xml version="1.0"?>
<robot name="my_robot">

  <!-- Base Link -->
  <link name="base_link"/>

  <!-- LiDAR Link -->
  <link name="base_laser"/>
  <joint name="laser_joint" type="fixed">
    <parent link="base_link"/>
    <child link="base_laser"/>
    <origin xyz="0.0 0.0 0.2" rpy="0 0 0"/>
  </joint>

  <!-- Camera Link -->
  <link name="camera_link"/>
  <joint name="camera_joint" type="fixed">
    <parent link="base_link"/>
    <child link="camera_link"/>
    <origin xyz="0.15 0.0 0.25" rpy="0 0 0"/>
  </joint>

</robot>
```

14.2 ld19.yaml

Listing 41: Complete LiDAR configuration

```
ld19_config:
  product_name: LDLiDAR_LD19
  topic_name: LiDAR/LD19
  frame_id: base_laser
  port_name: /dev/ttyUSB0
  port_baudrate: 230400
  laser_scan_dir: false
  enable_angle_crop_func: false
  angle_crop_min: 0.0
  angle_crop_max: 0.0
```

14.3 bringup.launch.py

Listing 42: Complete launch file

```
from launch import LaunchDescription
from launch_ros.actions import Node
from ament_index_python.packages import get_package_share_directory
from launch.substitutions import Command
import os
import yaml

```

```
def generate_launch_description():

    robot_description_path = os.path.join(
        get_package_share_directory('robot_description'),
        'urdf',
        'robot.urdf'
    )

    ld19_config_file = os.path.join(
        get_package_share_directory('robot_bringup'),
        'config',
        'ld19.yaml'
    )
    with open(ld19_config_file, 'r') as f:
        ld19_params = yaml.safe_load(f)['ld19_config']

    return LaunchDescription([

        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            parameters=[{
                'robot_description': Command(['cat ',
                   robot_description_path])
            }]
        ),

        Node(
            package='ldlidar_stl_ros2',
            executable='ldlidar_stl_ros2_node',
            name='lidar',
            output='screen',
            parameters=[ld19_params]
        ),

        Node(
            package='realsense2_camera',
            executable='realsense2_camera_node',
            name='camera',
            output='screen',
            parameters=[{
                'enable_depth': True,
                'enable_color': True,
                'pointcloud.enable': False
            }]
        ),
    ])
```

15 Appendix B: Glossary

Colcon Build tool for ROS 2 workspaces

Frame Coordinate system in 3D space

Launch File Python script to start multiple nodes

Node Executable ROS 2 process

Package Collection of ROS 2 code and resources

TF Transform library for managing coordinate frames

Topic Named bus for message passing between nodes

URDF Unified Robot Description Format (XML)

Workspace Directory containing ROS 2 packages

16 Appendix C: References

- ROS 2 Humble Documentation: https://docs.ros.org/en/humble/
- Waveshare LD-19 Tutorial: https://www.waveshare.com/wiki/Tutorial_IX_Lidar
- RealSense ROS Wrapper: https://github.com/IntelRealSense/realsense-ros
- TF2 Tutorials: https://docs.ros.org/en/humble/Tutorials/Intermediate/Tf2/Tf2-Main.html

17 Revision History

Version	Date	Changes
1.0	2024-12-18	Initial documentation release

Table 3: Document Revision History

This documentation was created for educational and development purposes.
For questions or improvements, please contact the maintainer.