# Service Robot URDF Documentation

Generated by Grok 3

May 13, 2025

## Contents

## Abstract

This document provides detailed documentation for the Universal Robot Description Format (URDF) files `service_robot.urdf.xml` and `service_robot_real.urdf.xml`, which define a differential drive service robot for simulation and real-world deployment. It includes an in-depth overview, component descriptions, input/output specifications, dependencies, configuration parameters, error handling, and a system component relationship graph. The documentation is designed for developers to understand, maintain, and extend the robot's configuration in ROS 2 and Gazebo environments.


# 1 Introduction

The `service_robot.urdf.xml` and `service_robot_real.urdf.xml` files describe a differential drive service robot equipped with four wheels, a manipulator arm, and sensors (LIDAR, camera, IMU). The simulation URDF (`service_robot.urdf.xml`) is tailored for Gazebo, while the real-world URDF (`service_robot_real.urdf.xml`) interfaces with physical hardware. This documentation provides a comprehensive guide for developers, covering the robot's structure, configuration, and usage.

## 1.1 Purpose

The URDF files serve the following purposes:

- **Simulation**: Enable testing of navigation, manipulation, and sensor processing in Gazebo.
- **Real-World Deployment**: Facilitate control of physical hardware via serial communication.
- **Extensibility**: Provide a modular structure for adding new components or modifying existing ones.

## 1.2 Scope

This document covers:

- Detailed component descriptions (links, joints, sensors, plugins).
- Input/output specifications, including topic names and message types.
- Software and hardware dependencies.
- Configuration parameters and tuning guidelines.
- Error handling and troubleshooting.
- A system component relationship graph.

# 2 System Overview

The service robot is a mobile platform with a differential drive system, a manipulator arm, and multiple sensors. The URDF files define its kinematic and dynamic properties, enabling simulation and hardware control. Key components include:

- **Base Link**: The robot's chassis (`base_link`), anchoring all components.
- **Wheels**: Four wheels driven by continuous joints for mobility.

- **Sensors**: LIDAR, camera, and IMU for environmental perception.
- **Manipulator Arm**: A multi-joint arm for object manipulation.
- **Plugins**: Gazebo plugins for simulation and hardware interfaces for real-world control.

The `service_robot.urdf.xml` uses the `gazebo_ros2_control` plugin for simulation, while `service_robot_real.urdf.xml` uses a `SimpleHardwareInterface` for serial communication with physical motors.

# 3 Component Descriptions

This section provides detailed descriptions of the robot's components, including their physical properties, kinematic constraints, and configuration parameters.

## 3.1 Links

Links represent the robot's physical components, each with inertial, visual, and collision properties.

| Link Name | Properties | Description |
|---|---|---|
| `base_link` | Mass: 10kg, Inertia: 0.5, Size: 0.5x0.5x0.2m | Main chassis, central anchor for wheels, sensors, and arm. |
| `front_left_wheel` | Mass: 1kg, Inertia: 0.2, Radius: 0.07m | Front left wheel, driven by a continuous joint. |
| `front_right_wheel` | Mass: 1kg, Inertia: 0.2, Radius: 0.07m | Front right wheel, driven by a continuous joint. |
| `rear_left_wheel` | Mass: 1kg, Inertia: 0.2, Radius: 0.07m | Rear left wheel, driven by a continuous joint. |
| `rear_right_wheel` | Mass: 1kg, Inertia: 0.2, Radius: 0.07m | Rear right wheel, driven by a continuous joint. |
| `camera_link` | Mass: 0.1kg, Size: 0.05x0.05x0.05m | Mount for the camera sensor, fixed to `base_link`. |
| `lidar_link` | Mass: 0.2kg, Size: 0.1x0.1x0.05m | Mount for the LIDAR sensor, fixed to `base_link`. |
| `imu_link` | Mass: 0.05kg, Size: 0.03x0.03x0.03m | Mount for the IMU sensor, fixed to `base_link`. |
| `arm_base_link` | Mass: 2kg, Size: 0.1x0.1x0.2m | Base of the manipulator arm, fixed to `base_link`. |
| `arm_link1`–`arm_link4` | Mass: 1kg each, Size: 0.3x0.1x0.1m | Segments of the manipulator arm, connected by revolute joints. |

Table 1: Detailed Link Descriptions

```
<link name="base_link">
    <inertial>
```

```
        <mass value="10.0"/>
        <inertia ixx="0.5" ixy="0.0" ixz="0.0" iyy="0.5" iyz="0.0" izz="0.5"/>
    </inertial>
    <visual>
        <geometry>
            <box size="0.5 0.5 0.2"/>
        </geometry>
        <material name="Gazebo/Black"/>
    </visual>
    <collision>
        <geometry>
            <box size="0.5 0.5 0.2"/>
        </geometry>
    </collision>
</link>
```

Listing 1: Sample Link Definition: base_link

### 3.2 Joints

Joints define kinematic relationships, with specific limits and dynamics.

| Joint Name | Properties | Description |
|---|---|---|
| `front_left_wheel_joint` | Type: Continuous, Limits: [-0.44, 0.44] rad/s | Connects `front_left_wheel` to `base_link`. |
| `front_right_wheel_joint` | Type: Continuous, Limits: [-0.44, 0.44] rad/s | Connects `front_right_wheel` to `base_link`. |
| `rear_left_wheel_joint` | Type: Continuous, Limits: [-0.44, 0.44] rad/s | Connects `rear_left_wheel` to `base_link`. |
| `rear_right_wheel_joint` | Type: Continuous, Limits: [-0.44, 0.44] rad/s | Connects `rear_right_wheel` to `base_link`. |
| `arm_base_to_link1` | Type: Revolute, Limits: [-1.57, 1.57] rad | Connects `arm_base_link` to `arm_link1`. |
| `arm_link1_to_link2` | Type: Revolute, Limits: [-1.57, 1.57] rad | Connects `arm_link1` to `arm_link2`. |

Table 2: Detailed Joint Descriptions

```
<joint name="front_left_wheel_joint" type="continuous">
    <parent link="base_link"/>
    <child link="front_left_wheel"/>
    <origin xyz="0.2 0.15 -0.05" rpy="0 0 0"/>
    <axis xyz="0 1 0"/>
    <limit lower="-0.44" upper="0.44" effort="100.0" velocity="0.44"/>
</joint>
```

Listing 2: Sample Joint Definition: front_left_wheel_joint

### 3.3 Sensors

Sensors provide environmental data, configured with Gazebo plugins in `service_robot.urdf.xml`.

- **LIDAR**:
  - Topic: `scan`
  - Message Type: `sensor_msgs/LaserScan`
  - Field of View: 360 degrees
  - Range: 0.12m to 12m
  - Update Rate: 10Hz
  - Noise: Gaussian, mean 0.0, stddev 0.01
- **Camera**:
  - Topics: `pi_camera/image_raw`, `pi_camera/camera_info`
  - Message Types: `sensor_msgs/Image`, `sensor_msgs/CameraInfo`
  - Resolution: 1920x1080
  - Update Rate: 30Hz
  - Noise: Gaussian, mean 0.0, stddev 0.007
- **IMU**:
  - Topic: `imu/data`
  - Message Type: `sensor_msgs/Imu`
  - Update Rate: 100Hz
  - Noise: Angular velocity (0.00033 rad/s), Linear acceleration (0.005 m/sš)

```
<gazebo reference="lidar_link">
    <sensor name="lidar" type="ray">
        <pose>0 0 0 0 0 0</pose>
        <ray>
            <scan>
                <horizontal>
                    <samples>360</samples>
                    <resolution>1.0</resolution>
                    <min_angle>0.0</min_angle>
                    <max_angle>6.28318530</max_angle>
                </horizontal>
            </scan>
            <range>
                <min>0.120000</min>
                <max>12.0</max>
                <resolution>0.015000</resolution>
            </range>
            <noise>
                <type>gaussian</type>
                <mean>0.0</mean>
                <stddev>0.01</stddev>
            </noise>
        </ray>
```

```
        <plugin name="lidar_plugin" filename="libgazebo_ros_ray_sensor.so">
            <ros>
                <remapping>~/out:=scan</remapping>
            </ros>
            <output_type>sensor_msgs/LaserScan</output_type>
            <frame_name>lidar_link</frame_name>
        </plugin>
    </sensor>
</gazebo>
```

Listing 3: Sample Sensor Plugin: LIDAR

### 3.4 Plugins

Plugins enable simulation and hardware control.

| Plugin Name | File | Description |
|---|---|---|
| gazebo_ros2_control/GazeboSystem | service_robot.urdf.xml | Simulates motor control for wheels and arm joints. |
| robot_control/SimpleHardwareInterface | service_robot_real.urdf.xml | Interfaces with physical motors via /dev/ttyUSB0 at 9600 baud. |
| gazebo_ros_camera | service_robot.urdf.xml | Simulates the camera sensor. |
| gazebo_ros_ray_sensor | service_robot.urdf.xml | Simulates the LIDAR sensor. |
| gazebo_ros_imu_sensor | service_robot.urdf.xml | Simulates the IMU sensor. |

Table 3: Detailed Plugin Descriptions

## 4 Input/Output Specifications

This section details the input and output interfaces for the URDF files.

### 4.1 Inputs

| Topic | Message Type | Description |
|---|---|---|
| /diff_controller/cmd_vel | geometry_msgs/Twist | Linear and angular velocity commands for differential drive. |
| /arm_controller/joint_trajectory | trajectory_msgs/JointTrajectory | Position/velocity commands for manipulator arm joints. |
| /dev/ttyUSB0 | Serial (custom) | Motor control commands for real hardware (9600 baud). |

Table 4: Input Specifications

### 4.2 Outputs

| Topic | Message Type | Description |
|---|---|---|
| scan | sensor_msgs/LaserScan | LIDAR data with range measurements. |
| pi_camera/image_raw | sensor_msgs/Image | Raw camera images (1920x1080). |
| pi_camera/camera_info | sensor_msgs/CameraInfo | Camera calibration and metadata. |
| imu/data | sensor_msgs/Imu | IMU data with orientation and acceleration. |

| | | |
|---|---|---|
| `/joint_states` | `sensor_msgs/JointState` | Wheel and arm joint positions/velocities. |

Table 5: Output Specifications

# 5 Dependencies

The URDF files require the following dependencies:

- **ROS 2 Humble**:
  - Version: Humble Hawksbill
  - Packages: `ros2_control`, `ros2_controllers`, `robot_state_publisher`
  - Installation: `sudo apt install ros-humble-ros2-control ros-humble-robot-state-publishe`
- **Gazebo**:
  - Version: 11 or later
  - Packages: `gazebo_ros_pkgs`
  - Installation: `sudo apt install ros-humble-gazebo-ros-pkgs`
- **Hardware Drivers**:
  - Driver: `SimpleHardwareInterface`
  - Port: `/dev/ttyUSB0`
  - Baud Rate: 9600
  - Installation: Custom package (`robot_control`)
- **Python Packages**:
  - Packages: `numpy`, `opencv-python`
  - Installation: `pip install numpy opencv-python`

# 6 Configuration Parameters

This section lists key configuration parameters for tuning the URDF files.

| Parameter | Location | Description |
|---|---|---|
| `wheel_velocity_limits` | `gazebo_ros2_control` | Velocity limits for wheels ([-0.44, 0.44] rad/s in simulation, [-0.5, 0.5] rad/s in real). |
| `serial_baudrate` | `SimpleHardwareInterface` | Baud rate for serial communication (9600). |
| `lidar_range` | `gazebo_ros_ray_sensor` | LiDAR range (0.12m to 12m). |
| `camera_resolution` | `gazebo_ros_camera` | Camera resolution (1920x1080). |
| `imu_noise` | `gazebo_ros_imu_sensor` | Noise parameters for IMU data. |

Table 6: Key Configuration Parameters

# 7 System Component Relationships

The following diagram illustrates the relationships between components, including data flow through plugins.

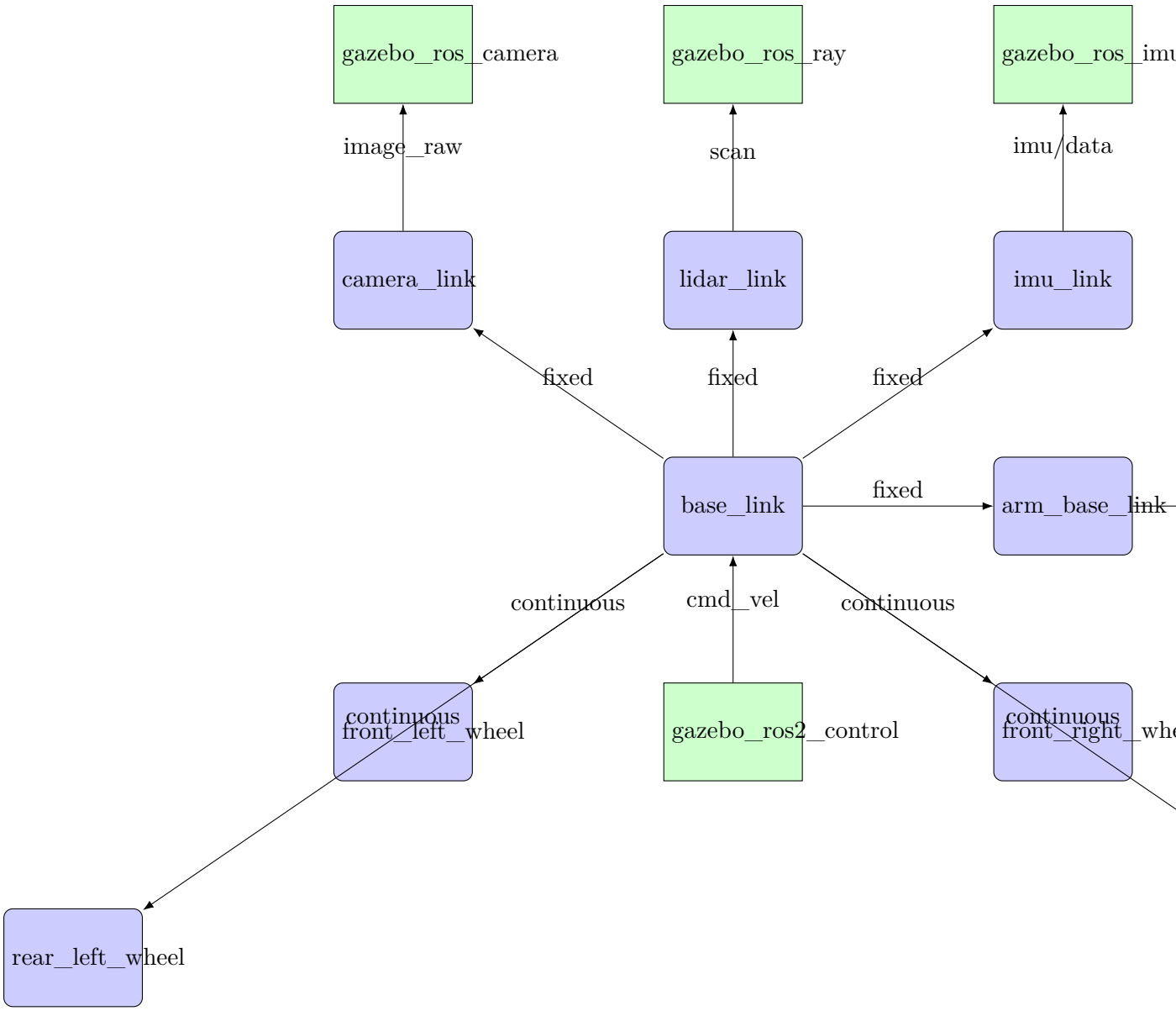


Figure 1: System Component Relationship Graph with Plugin Interactions

# 8 Error Handling

Common issues and their resolutions:

- **URDF Syntax Errors**: Use `check_urdf service_robot.urdf.xml` to validate syntax.
- **Plugin Failures**: Ensure Gazebo plugins are compatible with the installed Gazebo version.
- **Serial Communication Errors**: Verify `/dev/ttyUSB0` permissions and baud rate (9600).
- **Sensor Noise**: Adjust noise parameters in sensor plugins to match real-world conditions.

# 9 Performance Considerations

To optimize performance:

- **Simulation**: Reduce LIDAR and camera update rates for faster simulation (e.g., LIDAR to 5Hz, camera to 15Hz).
- **Real-World**: Minimize serial communication latency by optimizing `SimpleHardwareInterface` code.
- **Resource Usage**: Monitor CPU and memory usage in Gazebo using `htop` or `ros2 topic hz`.

# 10 Usage Guidelines

## 10.1 Simulation Setup

1. Install dependencies:

```
    sudo apt install ros-humble-gazebo-ros-pkgs ros-humble-ros2-
   control
    pip install numpy opencv-python

```

2. Source ROS 2:

```
    source /opt/ros/humble/setup.bash

```

3. Launch simulation:

```
    ros2 launch robot_control gazebo_launch.py

```

## 10.2 Real-World Deployment

1. Connect hardware to `/dev/ttyUSB0`.
2. Install `SimpleHardwareInterface`.
3. Launch hardware interface:

```
    ros2 launch robot_control hardware_launch.py

```

## 10.3 Best Practices

- Use Git for version control.
- Validate URDF with `check_urdf`.
- Document all changes to URDF files.

# 11 Maintenance and Extension

- **Adding Sensors**: Define new `link`, `joint`, and plugin entries.
- **Tuning Joints**: Adjust `limit` and `dynamics` for desired behavior.

- **Updating Plugins**: Verify compatibility with ROS 2 and Gazebo versions.
- **Troubleshooting**: Use `rviz2` to visualize the robot model.

# 12 Conclusion

The `service_robot.urdf.xml` and `service_robot_real.urdf.xml` files provide a robust framework for a differential drive service robot. This documentation equips developers with the knowledge to configure, deploy, and extend the robot's capabilities.

# 13 References

- ROS 2 Documentation: `https://docs.ros.org`
- Gazebo Documentation: `http://gazebosim.org`
- URDF Specification: `http://wiki.ros.org/urdf`

# 14 Appendix

## 14.1 Sample Plugin Configuration

```
<plugin name="gazebo_ros2_control" filename="libgazebo_ros2_control.so" >
    <ros>
        <namespace>/</namespace>
    </ros>
    <controller_manager_prefix>controller_manager</controller_manager_prefix>
    <controller_manager_timeout>1.0</controller_manager_timeout>
    <robot_description>robot_description</robot_description>
    <robot_state_publisher>robot_state_publisher</robot_state_publisher>
    <parameters>$(find robot_control)/config/diff_controller_sim.yaml</parameters>
</plugin>
```