**Отчет**

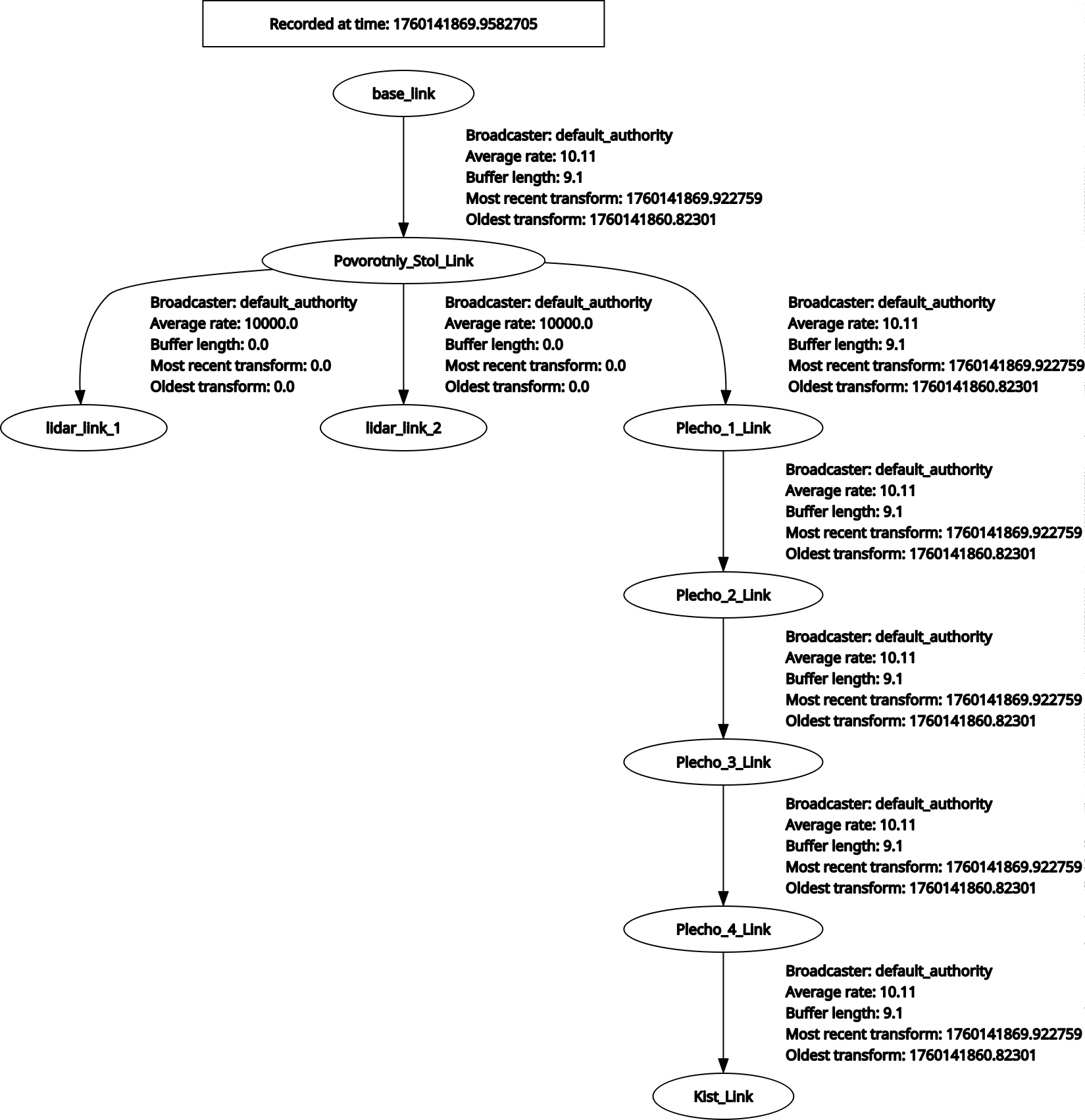
## 1. Цель работы

Создать собственный пакет ROS2, содержащий описание робота в формате URDF, реализовать код управления положением звеньев (joints) и визуализировать движение в среде RViz2.

## 2. Структура проекта

| Элемент | Назначение |
| --- | --- |
| urdf/my\_robot.urdf | Описание структуры робота (звенья и сочленения). |
| launch/visualize.launch.py | Запускает визуализацию в RViz2 с joint\_state\_publisher и robot\_state\_publisher. |
| my\_robot\_vis/move\_demo.py | Код управления положением робота, публикующий состояния суставов. |
| package.xml | Определяет зависимости пакета. |
| setup.py | Сценарий установки Python-ноды. |

#### 3. Структура робота



Робот состоит из **шести основных звеньев (links)** и **восьми сочленений (joints)**, включая базу и два лидара.

**Основные звенья:**

1. Base
2. Stol
3. Plecho\_1
4. Plecho\_2
5. Plecho\_3
6. Plecho\_4
7. Kist
8. Lidar\_1 и Lidar\_2

**Основные сочленения (joints):**

1. J\_1\_Base\_Stol — соединяет базу и стол.
2. lidar\_joint\_1 — поворот первого лидара.
3. lidar\_joint\_2 — поворот второго лидара.
4. J\_2\_Stol\_Plecho\_1 — соединяет стол и первое плечо.
5. J\_3\_Plecho\_1\_and\_2 — соединяет первое и второе плечо.
6. J\_4\_Plecho\_2\_and\_3 — соединяет второе и третье плечо.
7. J\_5\_Plecho\_3\_and\_4 — соединяет третье и четвёртое плечо.
8. J\_6\_Plecho\_4\_Kist — соединяет последнее плечо с кистью.

* **Суставы**:
* Дерево TF указывает на наличие не менее 6 суставов, соединяющих звенья, что соответствует 6-DOF управлению в move\_demo.py. Имена суставов из кода (J\_1\_Base\_Stol, J\_2\_Stol\_Plecho\_1, J\_3\_Plecho\_1\_and\_2, J\_4\_Plecho\_2\_and\_3, J\_5\_Plecho\_3\_and\_4, J\_6\_Plecho\_4\_Kist) предполагают:
  + J\_1\_Base\_Stol: Соединяет base\_link с Povorotny\_Stol\_Link.
  + J\_2\_Stol\_Plecho\_1: Соединяет Povorotny\_Stol\_Link с Plecho\_1\_Link.
  + J\_3\_Plecho\_1\_and\_2: Соединяет Plecho\_1\_Link с Plecho\_2\_Link.
  + J\_4\_Plecho\_2\_and\_3: Соединяет Plecho\_2\_Link с Plecho\_3\_Link.
  + J\_5\_Plecho\_3\_and\_4: Соединяет Plecho\_3\_Link с Plecho\_4\_Link.
  + J\_6\_Plecho\_4\_Kist: Соединяет Plecho\_4\_Link с Kist\_Link.
* Все суставы, вероятно, являются вращательными, что позволяет осуществлять движение по нескольким осям.

Дерево TF подтверждает кинематическую цепочку от base\_link через Povorotny\_Stol\_Link, Plecho\_1\_Link до Plecho\_4\_Link и, наконец, до Kist\_Link, с дополнительными ветвями для Lidar\_Jink\_1 и Lidar\_Jink\_2.

#### 4. Описание движения

class MoveDemo(Node):

def \_\_init\_\_(self):

super().\_\_init\_\_('move\_demo')

self.pub = self.create\_publisher(JointState, 'joint\_states', 10)

self.timer = self.create\_timer(0.1, self.timer\_cb)

self.t0 = time.time()

# 6-DOF Antar arm joint names

names = ['J\_1\_Base\_Stol',

'J\_2\_Stol\_Plecho\_1',

'J\_3\_Plecho\_1\_and\_2',

'J\_4\_Plecho\_2\_and\_3',

'J\_5\_Plecho\_3\_and\_4',

'J\_6\_Plecho\_4\_Kist']

self.msg = JointState()

self.msg.name = names

self.msg.position = [0.0] \* 6

def timer\_cb(self):

t = time.time() - self.t0

self.msg.position = [

math.sin(t) \* 0.8, # J1

math.sin(2\*t) \* 0.6, # J2

math.sin(3\*t) \* 0.5, # J3

math.sin(4\*t) \* 0.4, # J4

math.sin(5\*t) \* 0.3, # J5

math.sin(6\*t) \* 0.2 # J6

]

self.msg.header.stamp = self.get\_clock().now().to\_msg()

self.pub.publish(self.msg)

Движение робота управляется скриптом move\_demo.py, узлом ROS 2, который публикует сообщения JointState в топике joint\_states. Движение генерируется следующим образом:

* **Логика управления**: Скрипт использует таймер для обновления положений суставов каждые 0.1 секунды на основе прошедшего времени с момента запуска узла (time.time() - t0).
* **Траектории суставов**: Каждое из 6 звеньев следует синусоидальной функции с увеличивающейся частотой:
  + J\_1: math.sin(t) \* 0.8 (амплитуда 0.8 радиан).
  + J\_2: math.sin(2\*t) \* 0.6 (амплитуда 0.6 радиан, в два раза большая частота).
  + J\_3: math.sin(3\*t) \* 0.5.
  + J\_4: math.sin(4\*t) \* 0.4.
  + J\_5: math.sin(5\*t) \* 0.3.
  + J\_6: math.sin(6\*t) \* 0.2.
* **Поведение**: Это создает плавное, периодическое движение, где каждый сустав колеблется с уникальной амплитудой и частотой, что приводит к волнообразному движению манипулятора. Уменьшающиеся амплитуды (от 0.8 до 0.2) указывают на сужение движения от базы к концевику.
* **Визуализация**: Движение отображается в RViz с помощью robot\_state\_publisher (применяет модель URDF) и joint\_state\_publisher\_gui (для ручного управления при необходимости), с конфигурацией default.rviz, отображающей модель робота и фреймы TF.

#### Детали реализации

* **ROS-пакет**: Пакет my\_robot\_vis содержит URDF, файл запуска и скрипт управления.
* **Файл запуска**: visualize.launch.py инициирует robot\_state\_publisher, joint\_state\_publisher\_gui и rviz2 для визуализации в реальном времени.
* **Визуализация дерева URDF**: Предоставленное дерево TF (записано в 1760141869.952705) показывает иерархию трансформаций, подтверждая структуру звеньев и суставов.

#### 5. Вывод

Робот удовлетворяет требованию наличия не менее 3 звеньев и 3 суставов, имея 6-DOF дизайн. Его движение программно управляется с использованием синусоидальных паттернов, а конфигурация поддерживает визуализацию в RViz. Для улучшения проекта следует полностью определить файл URDF и экспортировать дерево TF как формальную диаграмму (например, с помощью urdf\_to\_graphiz).