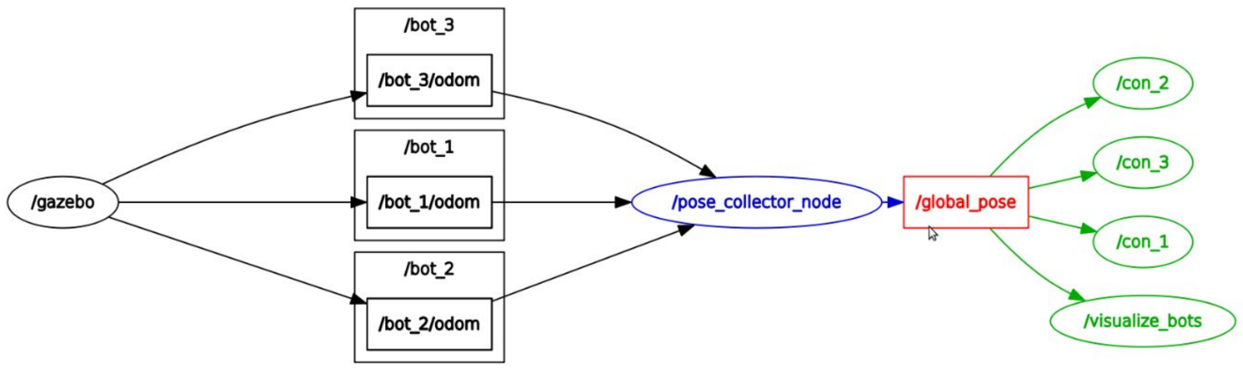
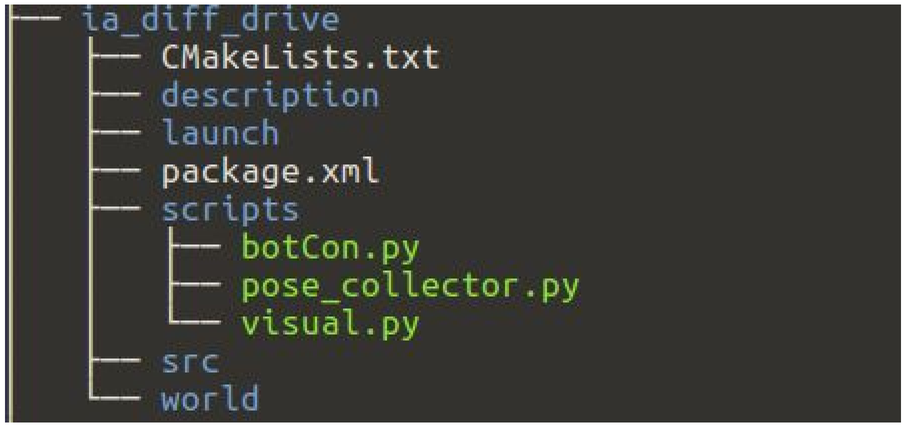
The ros-graph for the project looks like the following:



The project directory structure is as following :



```
├── ia_diff_drive
    ├── CMakeLists.txt
    ├── description
    ├── launch
    ├── package.xml
    ├── scripts
    │   ├── botCon.py
    │   ├── pose_collector.py
    │   └── visual.py
    ├── src
    └── world
```

Description of scripts:

Pose_collector.py

Subscribes to all visible ‘Odometry’ topics and creates a dictionary of these poses. This dictionary is published as a string via topic ‘/global_pose’.

This global pose is provided to all the robots. The controller uses the location information of all robots to perform collision avoidance.

botCon.py

Implements the collision avoidance behavior. Each robot is running the same behavior. Global positioning is available via topic ‘/global_pose’.
In the next page, we will go through the member functions of this script.

visual.py

Uses global positioning information to display robot locations and intervals. This was done to help in debugging. Stop all robots and manually move one robot and observe the visualization to see intersection and free sectors etc..

Member functions of botCon.py **script**:

controller

This is the main function. It performs the following tasks:

- Initializes the controller node
- Subscribes to '/global_pose' topic
- Continuously checks for collision and publishes 'Twist' message to '/cmd_vel' topic. This twist message will steer the robot, thus our control law has to compute linear and angular velocity (components of the 'Twist' message).

collision_detection

It accepts two intervals and returns a boolean value indicating whether a collision was observed between those intervals

collision_avoidance

Given two intersecting intervals returns free interval (if any)

inclusion_interval

Checks if one interval is included within the other interval

arc_arc

Helper function: Detects intersection of an arc with another arc

arc_line

Helper function: Detects intersection of a line with an arc

line_line

Helper function: Detects intersection of a line with another line

bisect

Helper function to split a range into two

<u>Exercise:</u>

Manually actuate one robot. Keep other robots at rest. Observe the plot generated by visual.py