**Nodes**

**Package: ramp\_planner**

ramp\_planner:

* Subscribed topics:
  + “update”
    - ramp\_msgs/MotionState
    - Get the latest pose information
  + “obstacles”
    - ramp\_msgs/ObstacleList
    - Get the latest obstacle information
* Published topics:
  + “bestTrajec”
    - ramp\_msgs/RampTrajectory
    - Best trajectory in the population at the current time
  + “/visualization\_marker\_array”
    - visualization\_msgs/MarkerArray
    - Circles to show the start and goal positions
  + “imminent\_collision”
    - std\_msgs/Bool
    - Boolean to toggle imminent collision for the ramp\_control

pub\_map\_odom: This node publishes a transform with child=odom and parent=map at a fixed rate.

**Package trajectory\_generator**

trajectory\_generator:

* Services:
  + “trajectory\_generator”
    - ramp\_msgs/TrajectorySrv

**Package trajectory\_evaluation**

trajectory\_evaluation:

* Services:
  + “trajectory\_evaluation”
    - ramp\_msgs/EvaluationSrv

**Package path\_modification**

path\_modification:

* Services:
  + “path\_modification”
    - ramp\_msgs/ModificationRequest

**Package ramp\_sensing**

ramp\_sensing:

* Subscribed topics:
  + “/costmap\_node/costmap/costmap”
    - nav\_msgs/OccupancyGrid
    - Costmap based on laser information
* Published topics:
  + “/obstacles”
    - ramp\_msgs/ObstacleList
    - Array of obstacle information
  + “/visualization\_marker\_array”
    - visualization\_msgs/MarkerArray
    - Various visualization information

pub\_\*: These nodes publish various transforms that may be useful.

**Package ramp\_control**

ramp\_control:

* Subscribed topics:
  + “/bestTrajec”
    - ramp\_msgs/BestTrajectory
    - Trajectory to move the robot on
* Published topics:
  + “/update”
    - ramp\_msgs/MotionState
    - Current pose of the robot based on odometry

reset\_odom.py: Resets the odometry so that the robot nodes don’t need to be restarted after each run

* Published topics:
  + “mobile\_base/commands/reset\_odometry”
    - std\_msgs/Empty
    - Empty message to trigger resetting odometry

**Important Launch Files:**

All launch files can be found in the ramp\_launch package.

planner\_full\_costmap: This is the launch file used to start the planner when using a real robot.

* Nodes:
  + ramp\_planner
  + ramp\_control
  + ramp\_sensing
  + trajectory\_generator
  + trajectory\_evaluation
  + path\_modification
  + reset\_odom.py
  + costmap
  + pub\_map\_odom
  + laser\_filter (range filter)
* Parameters:
  + ramp\_planner/config.yaml
  + ramp\_launch/launch/config/robot\_0.yaml
  + laser\_filters/examples/range\_filter.yaml

planner\_full\_costmap\_simulation: This is the launch file used to start the planner when using the Gazebo simulator.

This file needs the gazebo\_costmap.launch file to be ran separately.

* Nodes:
  + ramp\_planner
  + ramp\_control
  + ramp\_sensing
  + trajectory\_generator
  + trajectory\_evaluation
  + path\_modification
  + reset\_odom.py
  + move\_gazebo\_obs
* Parameters:
  + ramp\_planner/config.yaml
  + ramp\_launch/launch/config/robot\_0.yaml

gazebo\_costmap.launch: This launch file runs Gazebo and the costmap nodes

* Nodes:
  + Gazebo
  + costmap
  + pub\_map\_odom
  + laser\_filter (range filter)
* Parameters:
  + laser\_filters/examples/range\_filter.yaml