PROGRAMMING FOR AUTONOMOUS SYSTEMS

Fred Livingston, PhD

Work Shop 002

CONTACT INFO

Fred Livingston, PhD

Email: fjliving@ncsu.edu

Mobile: 919.795.4710

Web: https://livingston.wordpress.ncsu.edu/

Bitbucket: https://bitbucket.org/livingston_ai/

SPRING 2023 WORKSHOP SERIES

- WS 001 – Introduction to Robot Programming using ROS2 [Feb 17th, 2023]
- WS 002 – Navigation [March 10th, 2023]
- WS 003 – Autonomous Navigation [TBD]

ROBOT NAVIGATION

- Review of ROS2
- F1TENTH Gym Setup
- Developing ROS Packages and Programs
- Mapping and Localization

ROS 2 Cheats Sheet

Command Line Interface

All ROS 2 CLI tools start with the prefix 'ros2' followed by a command, a verb and (possibly) positional/optional arguments.

For any tool, the documentation is accessible with,

$ ros2 **command** --help

and similarly for verb documentation,

$ ros2 **command** verb -h

Similarly, auto-completion is available for all commands/verbs and most positional/optional arguments. E.g.,

$ ros2 **command** [tab][tab]

Some of the examples below rely on:

ROS 2 demos package.

action Allows to manually send a goal and displays debugging information about actions.

Verbs:

info	Output information about an action.
list	Output a list of action names.
send_goal	Send an action goal.
show	Output the action definition.

Examples:

```
$ ros2 action info /fibonacci
$ ros2 action list
$ ros2 action send_goal /fibonacci \
  action_tutorials/action/Fibonacci "order: 5"
$ ros2 action show action_tutorials/action/Fibonacci
```

bag Allows to record/play topics to/from a rosbag.

Verbs:

info	Output information of a bag.
play	Play a bag.
record	Record a bag.

list	Output a list of running containers and components.
load	Load a component into a container node.
standalone	Run a component into its own standalone container node.
types	Output a list of components registered in the ament index.
unload	Unload a component from a container node.

Examples:

```
$ ros2 component list
$ ros2 component load /ComponentManager \
 composition composition::Talker
$ ros2 component types
$ ros2 component unload /ComponentManager 1
```

daemon Various daemon related verbs.

Verbs:

start	Start the daemon if it isn't running.
status	Output the status of the daemon.
stop	Stop the daemon if it is running

doctor A tool to check ROS setup and other potential issues such as network, package versions, rmw middleware etc.

Alias: **wtf** (where's the fire).

Arguments:

--report/-r	Output report of all checks.
--report-fail/-rf	Output report of failed checks only.
--include-warning/-iw	Include warnings as failed checks.

Examples:

```
$ ros2 doctor
$ ros2 doctor --report
$ ros2 doctor --report-fail
$ ros2 doctor --include-warning
```

interface Various related verbs. Inte the following optio srvs'.

Verbs:

list	Li
package	Ou wi
packages	Ou te
proto	Pr fa
show	Ou

Examples:

```
$ ros2 interface
$ ros2 interface
$ ros2 interface
$ ros2 interface
$ ros2 interface
```

launch Allows to without to 'cd' the

Usage:

$ ros2 launch <

Example:

$ ros2 launch de

lifecycle Various

Verbs:

get	Get li
list	Outp
nodes	Outp
set	Trigg

msg (deprecated messages.

Verbs:

$ ros2 msg list
$ ros2 msg package std_msgs
$ ros2 msg packages
$ ros2 msg show geometry_msgs/msg/Pose

multicast Various multicast related verbs.
Verbs:

receive	Receive a single UDP multicast packet.
send	Send a single UDP multicast packet.

node Displays debugging information about nodes.
Verbs:

info	Output information about a node.
list	Output a list of available nodes.

Examples:
$ ros2 node info /talker
$ ros2 node list

param Allows to manipulate parameters.
Verbs:

delete	Delete parameter.
describe	Show descriptive information about declared parameters.
dump	Dump the parameters of a given node in yaml format, either in terminal or in a file.
get	Get parameter.
list	Output a list of available parameters.
set	Set parameter

Examples:
$ ros2 param delete /talker /use_sim_time
$ ros2 param get /talker /use_sim_time
$ ros2 param list
$ ros2 param set /talker /use_sim_time false

pkg Create a ros2 package or output package(s)-related information.
Verbs:

create	Create a new ROS2 package.

$ ros2 pkg executables demo_nodes_cpp
$ ros2 pkg list
$ ros2 pkg prefix std_msgs
$ ros2 pkg xml -t version

run Allows to run an executable in an arbitrary package without having to 'cd' there first.
Usage:
$ ros2 run <package> <executable>
Example:
$ ros2 run demo_node_cpp talker

security Various security related verbs.
Verbs:

create_key	Create key.
create_permission	Create keystore.
generate_artifacts	Create permission.
list_keys	Distribute key.
create_keystore	Generate keys and permission files from a list of identities and policy files.
distribute_key	Generate XML policy file from ROS graph data.
generate_policy	List keys.

Examples (see sros2 package):
$ ros2 security create_key demo_keys /talker
$ ros2 security create_permission demo_keys /talker \
policies/sample_policy.xml
$ ros2 security generate_artifacts
$ ros2 security create_keystore demo_keys

service Allows to manually call a service and displays debugging information about services.
Verbs:

call	Call a service.
find	Output a list of services of a given type.
list	Output a list of service names.

srv (deprecated)
Verbs:

list	Ou
package	Ou wi
packages	Ou se
show	Ou

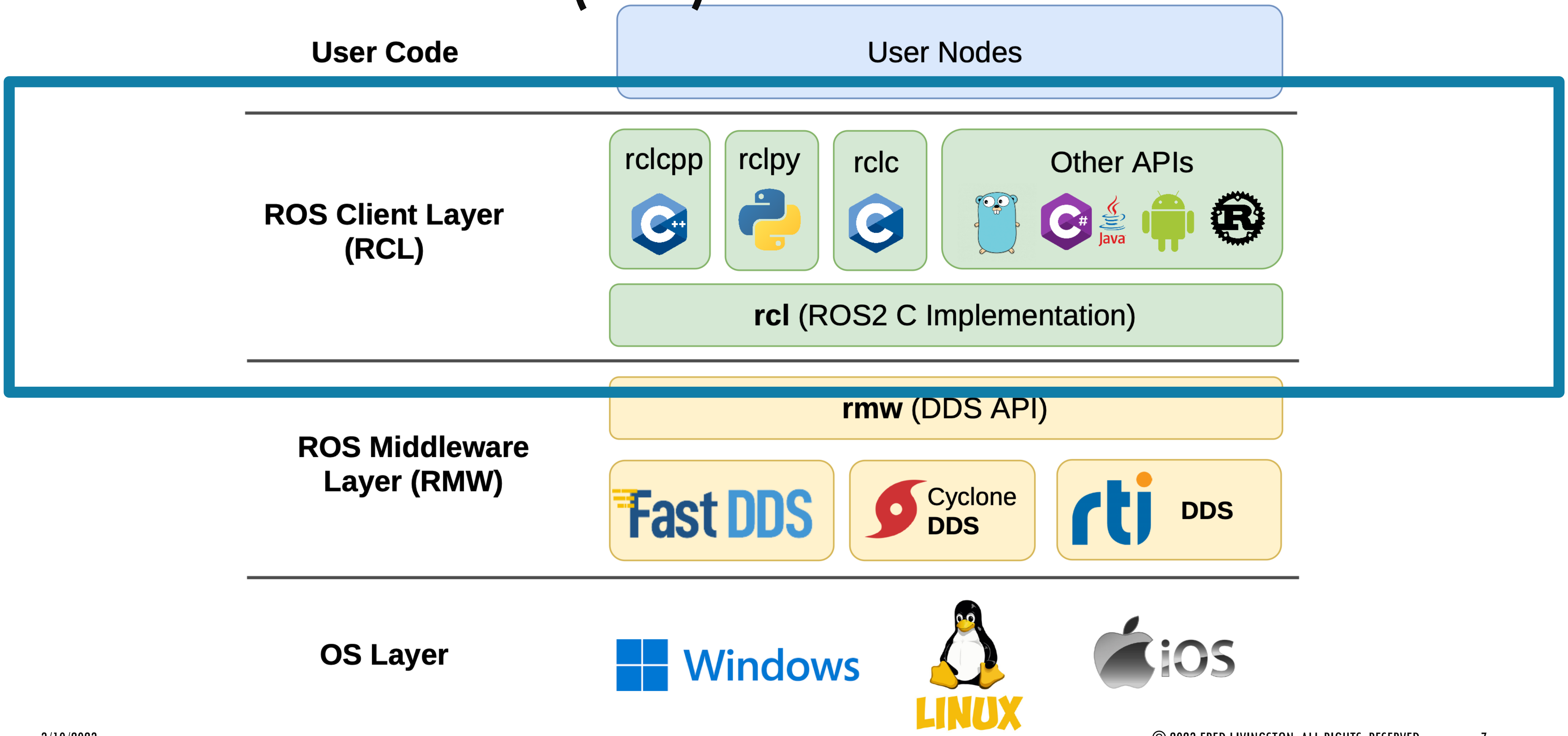
test Run a ROS2

topic A tool for d topics, including p and messages.
Verbs:

bw	Displa
delay	Displa header
echo	Outpu
find	Find t
hz	Displa
info	Outpu
list	Outpu
pub	Publis
type	Outpu

Examples:
$ ros2 topic bw
$ ros2 topic ech
$ ros2 topic finc
$ ros2 topic hz
$ ros2 topic infc
$ ros2 topic list
$ ros2 topic pub
'data: Hello RC
$ ros2 topic typ

ROS CLIENT LAYER (RCL)



F1TENTH GYM

https://github.com/f1tenth/f1tenth_gym_ros

Native on Ubuntu 20.04

Install the following dependencies:

- **ROS 2** Follow the instructions here to install ROS 2 Foxy.
- **F1TENTH Gym**

```
git clone https://github.com/f1tenth/f1tenth_gym
cd f1tenth_gym && pip3 install -e .
```

Installing the simulation:

- Create a workspace: `cd $HOME && mkdir -p sim_ws/src`
- Clone the repo into the workspace:

```
cd $HOME/sim_ws/src
git clone https://github.com/f1tenth/f1tenth_gym_ros
```

- Update correct parameter for path to map file: Go to `sim.yaml` https://github.com/f1tenth/f1tenth_gym_ros/blob/main/config/sim.yaml in your cloned repo, change the `map_path` parameter to point to the correct location. It should be `'<your_home_dir>/sim_ws/src/f1tenth_gym_ros/maps/levine'`
- Install dependencies with rosdep:

```
source /opt/ros/foxy/setup.bash
cd ..
rosdep install -i --from-path src --rosdistro foxy -y
```

- Build the workspace: `colcon build`

F1TENTH GYM (CLONE REPO)

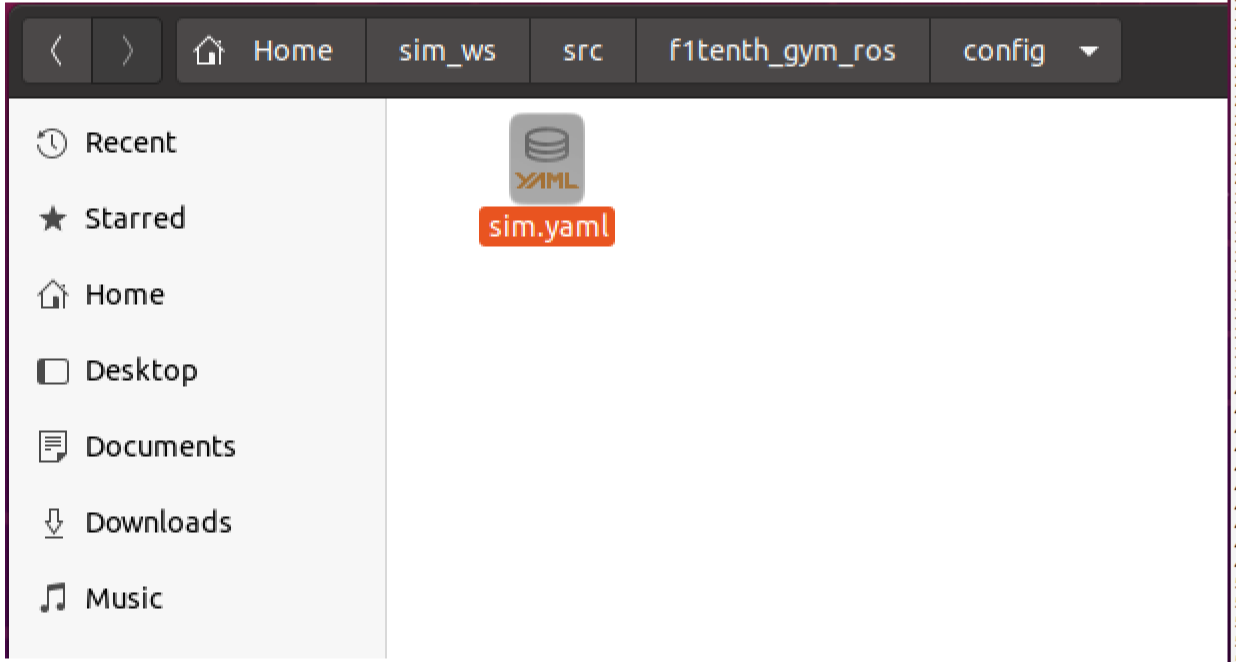
$ git clone https://github.com/f1tenth/f1tenth_gym_ros

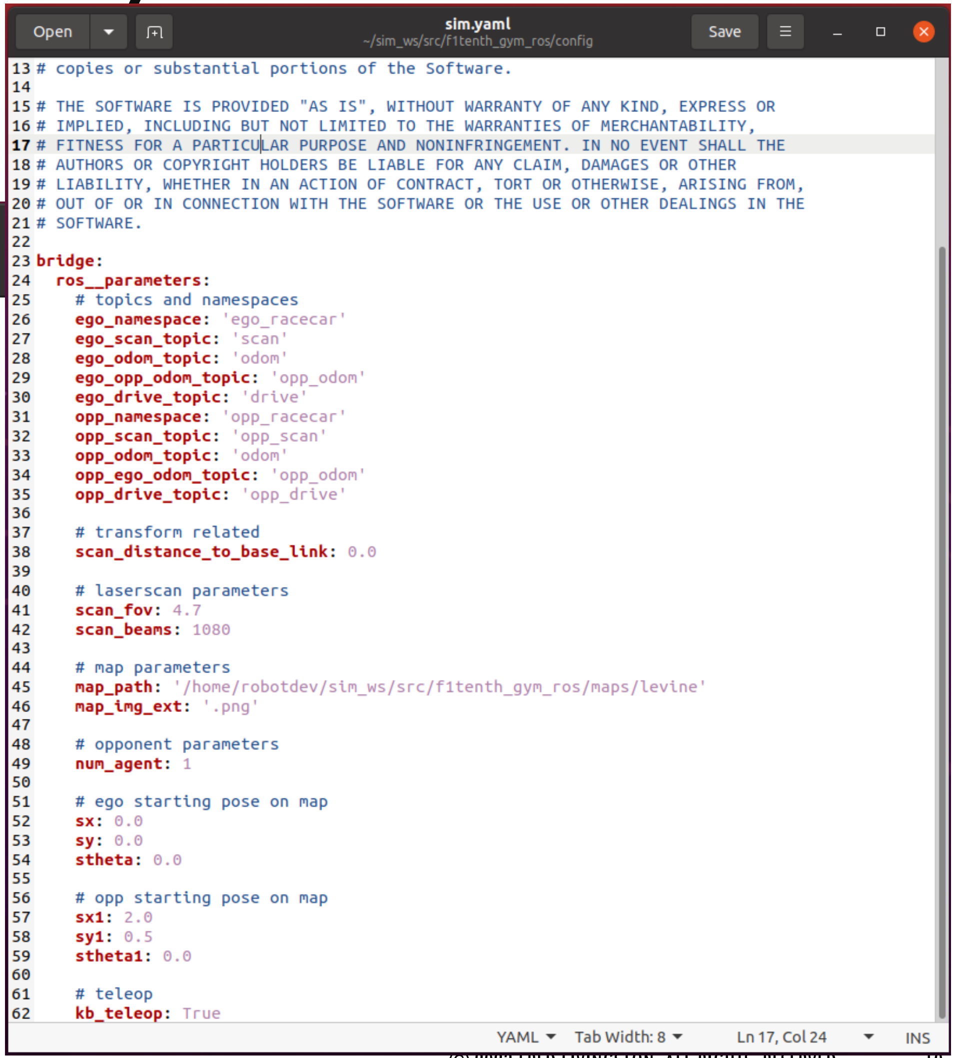
robotdev@ubuntu: ~/sim_ws/src

```
robotdev@ubuntu:~/sim_ws/src$ git clone https://github.com/f1tenth/f1tenth_gym_ros
Cloning into 'f1tenth_gym_ros'...
remote: Enumerating objects: 544, done.
remote: Counting objects: 100% (282/282), done.
remote: Compressing objects: 100% (71/71), done.
remote: Total 544 (delta 241), reused 221 (delta 211), pack-reused 262
Receiving objects: 100% (544/544), 501.06 KiB | 3.60 MiB/s, done.
Resolving deltas: 100% (303/303), done.
robotdev@ubuntu:~/sim_ws/src$ ls
f1tenth_gym_ros
robotdev@ubuntu:~/sim_ws/src$
```

F1TENTH GYM (CONFIGURATION)

Map_path: must contain full path name
Num_agent: 1 or 2





F1TENTH GYM (INSTALL DEPENDENCIES)

$ cd ..

$ source /opt/ros/foxy/setup.bash

$ rosdep install -i –from-path src –rosdistro foxy -y

```
robotdev@ubuntu: ~/sim_ws
robotdev@ubuntu:~/sim_ws/src$ cd ..
robotdev@ubuntu:~/sim_ws$ source /opt/ros/foxy/setup.bash
ROS_DISTRO was set to 'noetic' before. Please make sure that the environment does not mix paths from different distributi
ons.
robotdev@ubuntu:~/sim_ws$ source /opt/ros/foxy/setup.bash
robotdev@ubuntu:~/sim_ws$ rosdep install -i --from-path src --rosdistro foxy -y
executing command [sudo -H pip3 install -U transforms3d]
[sudo] password for robotdev:
Collecting transforms3d
  Downloading transforms3d-0.4.1.tar.gz (1.4 MB)
     |                                | 1.4 MB 2.6 MB/s
Building wheels for collected packages: transforms3d
  Building wheel for transforms3d (setup.py) ... done
  Created wheel for transforms3d: filename=transforms3d-0.4.1-py3-none-any.whl size=1376754 sha256=ec54f8bb45f48d20031031
3e7cb96ec64b8e7a6e8067c3d5603bacfef3c12731
  Stored in directory: /root/.cache/pip/wheels/7b/f0/88/0198ac6eca77ffa36e7247595585e9ec34e1dcc327847ebe73
Successfully built transforms3d
Installing collected packages: transforms3d
Successfully installed transforms3d-0.4.1
executing command [sudo -H apt-get install -y ros-foxy-ackermann-msgs]
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

F1TENTH GYM (COMPILE SRC)

$ colcon build

```
robotdev@ubuntu: ~/sim_ws

robotdev@ubuntu:~/sim_ws$ ls
f1tenth_gym  src
robotdev@ubuntu:~/sim_ws$ colcon build
Starting >>> f110_gym
Starting >>> f1tenth_gym_ros
Finished <<< f110_gym [1.20s]
Finished <<< f1tenth_gym_ros [1.72s]

Summary: 2 packages finished [2.36s]
robotdev@ubuntu:~/sim_ws$
```

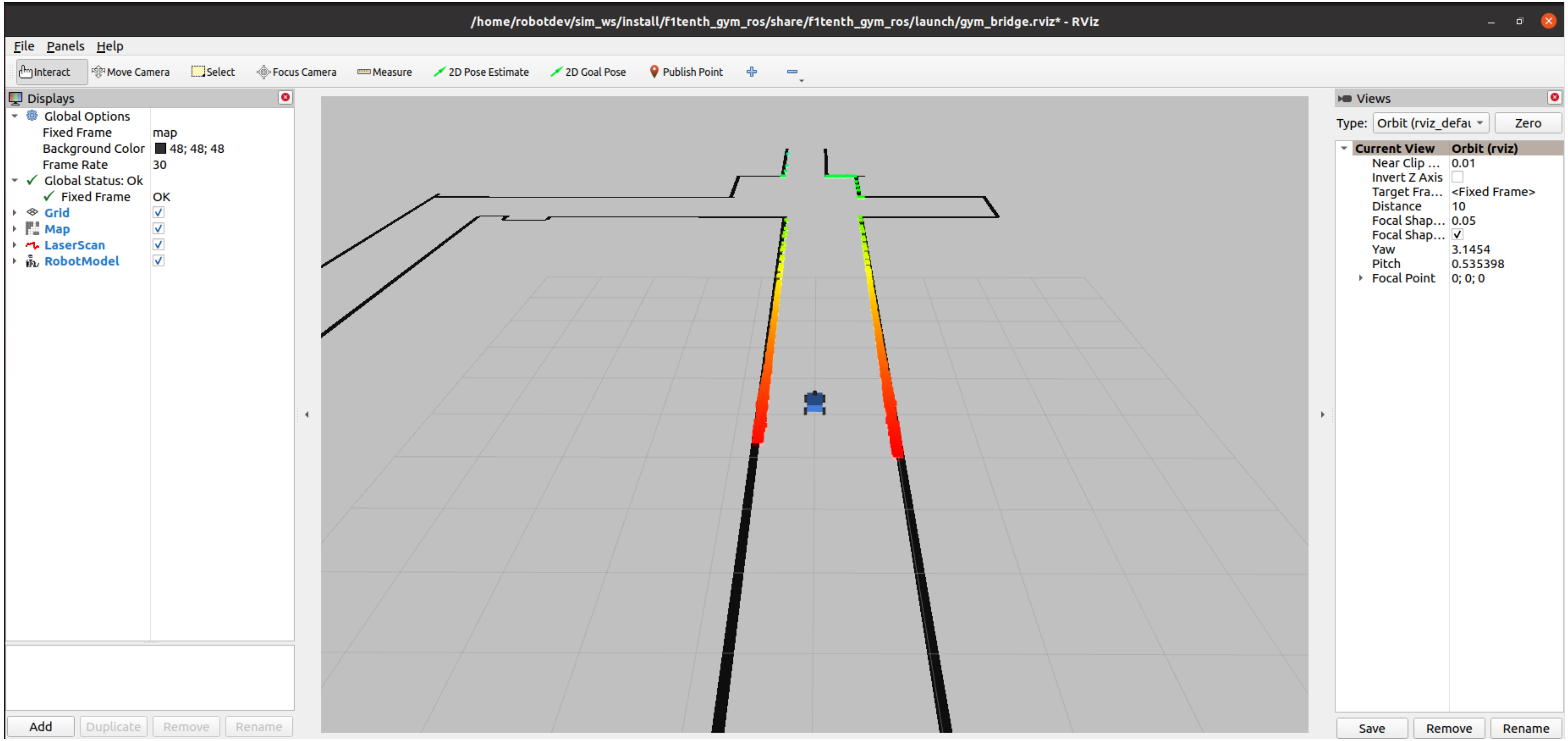
F1TENTH GYM (RUN SIMULATOR)

$ ros2 launch f1tenth_gym_ros gym_bridge_launch.py

robotdev@ubuntu: ~/sim_ws

```
robotdev@ubuntu:~/sim_ws$ ls
f1tenth_gym  src
robotdev@ubuntu:~/sim_ws$ colcon build
Starting >>> f110_gym
Starting >>> f1tenth_gym_ros
Finished <<< f110_gym [1.20s]
Finished <<< f1tenth_gym_ros [1.72s]

Summary: 2 packages finished [2.36s]
robotdev@ubuntu:~/sim_ws$ ros2 launch f1tenth_gym_ros gym_bridge_launch.py
```

F1TENTH GYM (RUN SIMULATOR)



F1TENTH GYM (TOPICS)

$ ros2 topic list -t

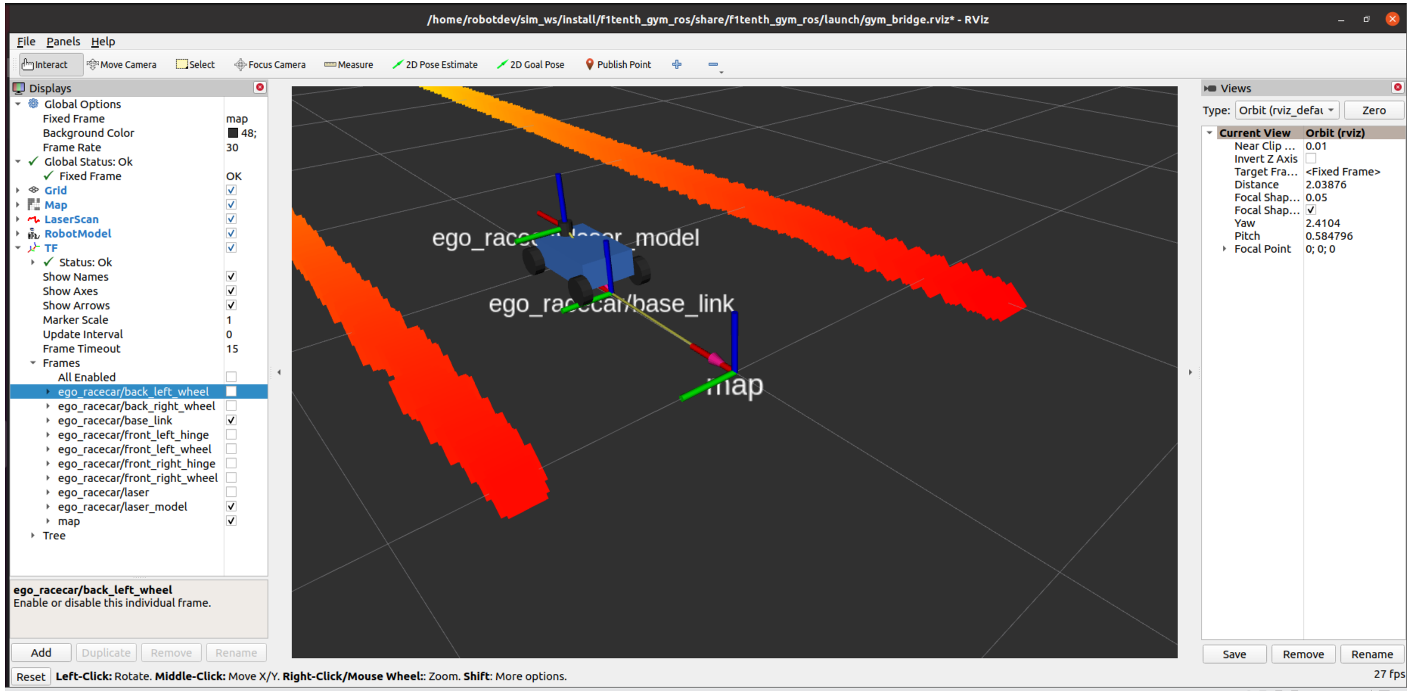
robotdev@ubuntu: ~/sim_ws

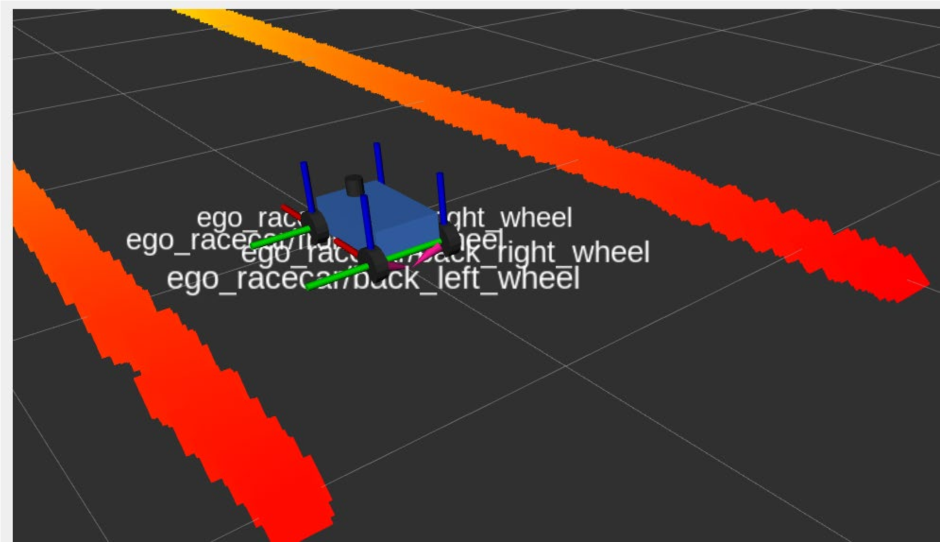
robotdev@ubuntu: ~/sim_ws | robotdev@ubuntu: ~/sim_ws

```
rorobotdev@ubuntu:~/sim_ws$ ros2 topic list -t
/clicked_point [geometry_msgs/msg/PointStamped]
/clock [rosgraph_msgs/msg/Clock]
/cmd_vel [geometry_msgs/msg/Twist]
/drive [ackermann_msgs/msg/AckermannDriveStamped]
/ego_racecar/odom [nav_msgs/msg/Odometry]
/ego_robot_description [std_msgs/msg/String]
/goal_pose [geometry_msgs/msg/PoseStamped]
/initialpose [geometry_msgs/msg/PoseWithCovarianceStamped]
/joint_states [sensor_msgs/msg/JointState]
/map [nav_msgs/msg/OccupancyGrid]
/map_server/transition_event [lifecycle_msgs/msg/TransitionEvent]
/map_updates [map_msgs/msg/OccupancyGridUpdate]
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/scan [sensor_msgs/msg/LaserScan]
/tf [tf2_msgs/msg/TFMessage]
/tf_static [tf2_msgs/msg/TFMessage]
robotdev@ubuntu:~/sim_ws$
```

TOPICS /TF

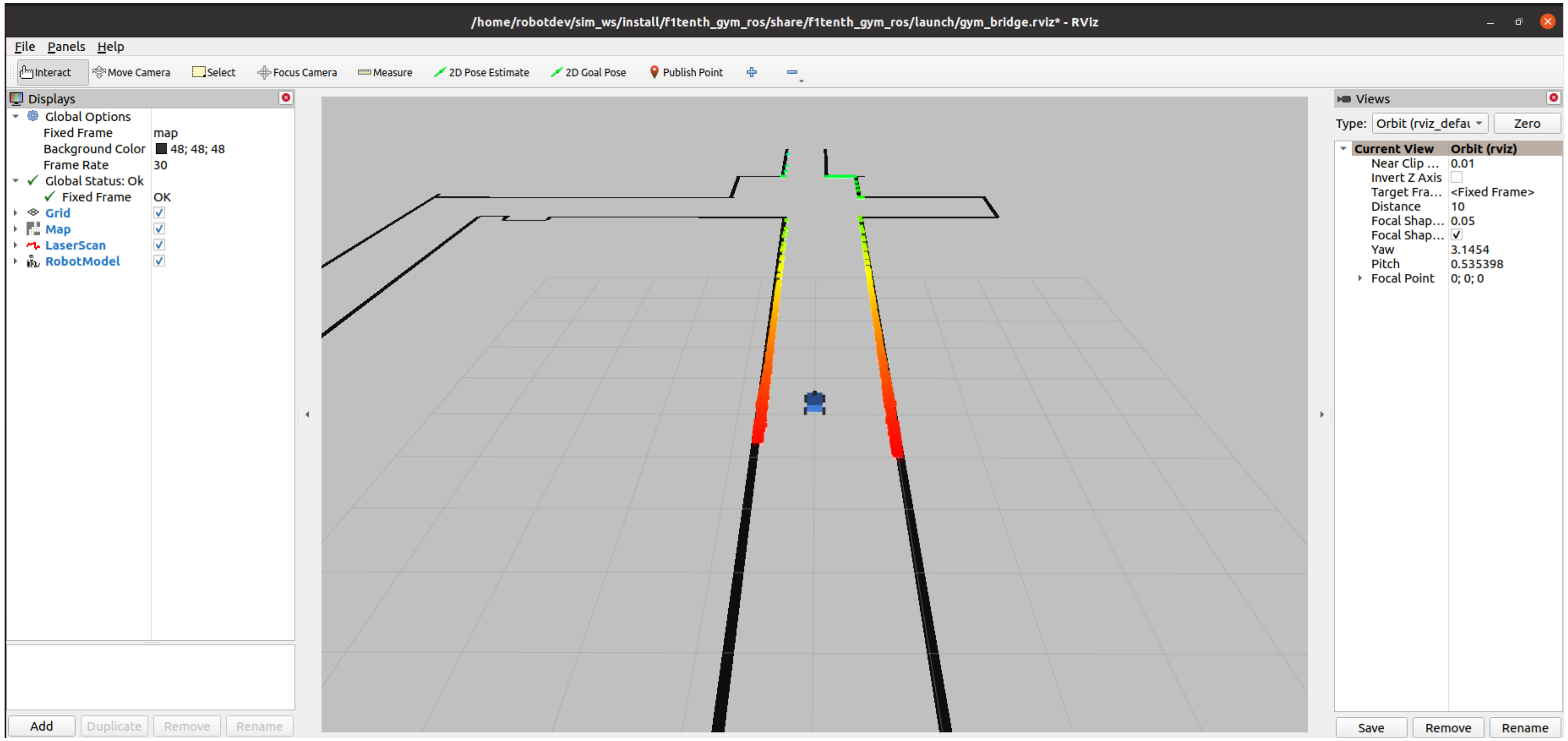
NEU (North, East, Up) Coordinates Systems





TOPIC /INITIALPOSE

A useful function of the simulator is that you can instantly move the car without driving it to its new location. To do this, click the 2D Pose Estimate pose button at the top of the rViz window, and then click the desired location on the track to move the car there.



TOPIC /COMAND_VEL

http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Twist.html

geometry_msgs/Twist Message

File: geometry_msgs/Twist.msg

Raw Message Definition

```
# This expresses velocity in free space broken into its linear and angular parts.
Vector3  linear
Vector3  angular
```

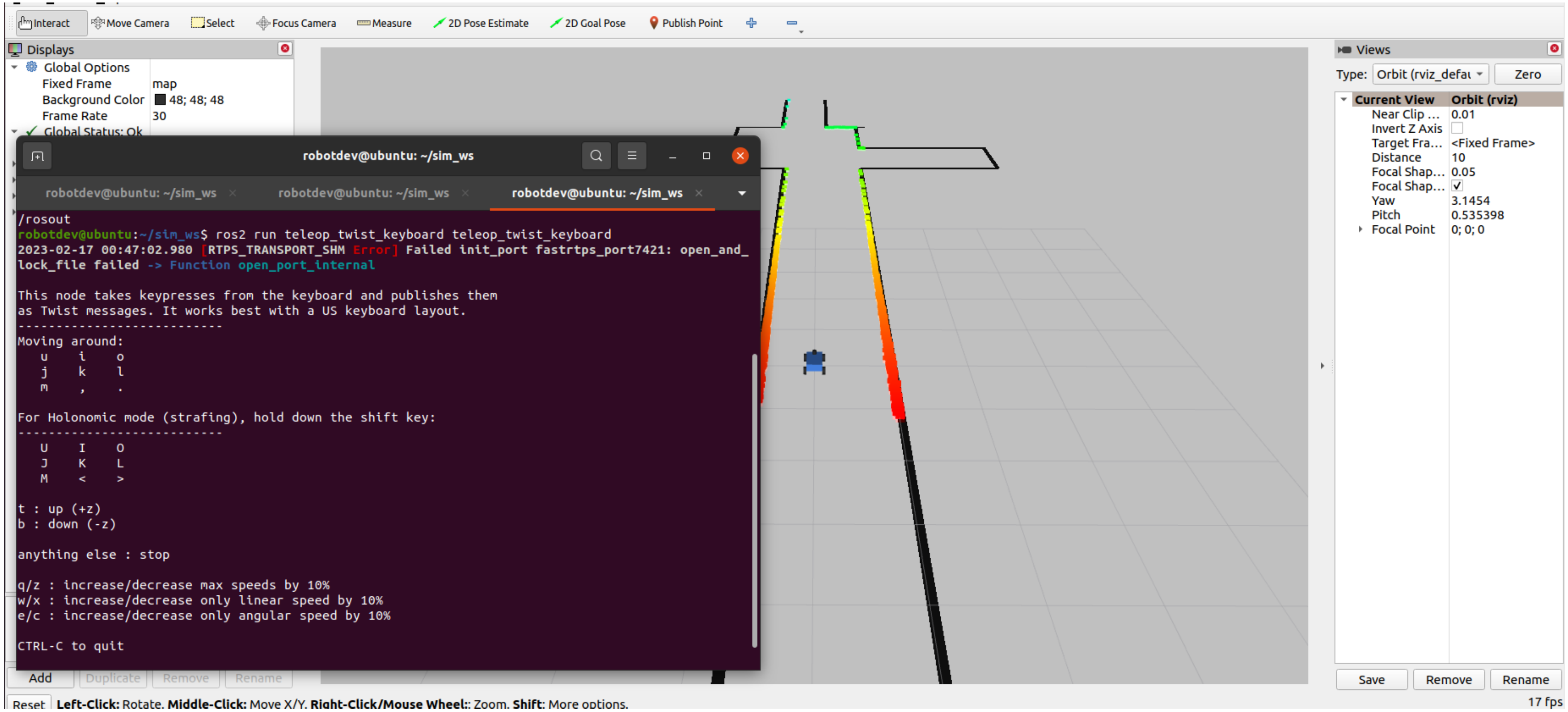
Compact Message Definition

```
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

autogenerated on Wed, 02 Mar 2022 00:06:53

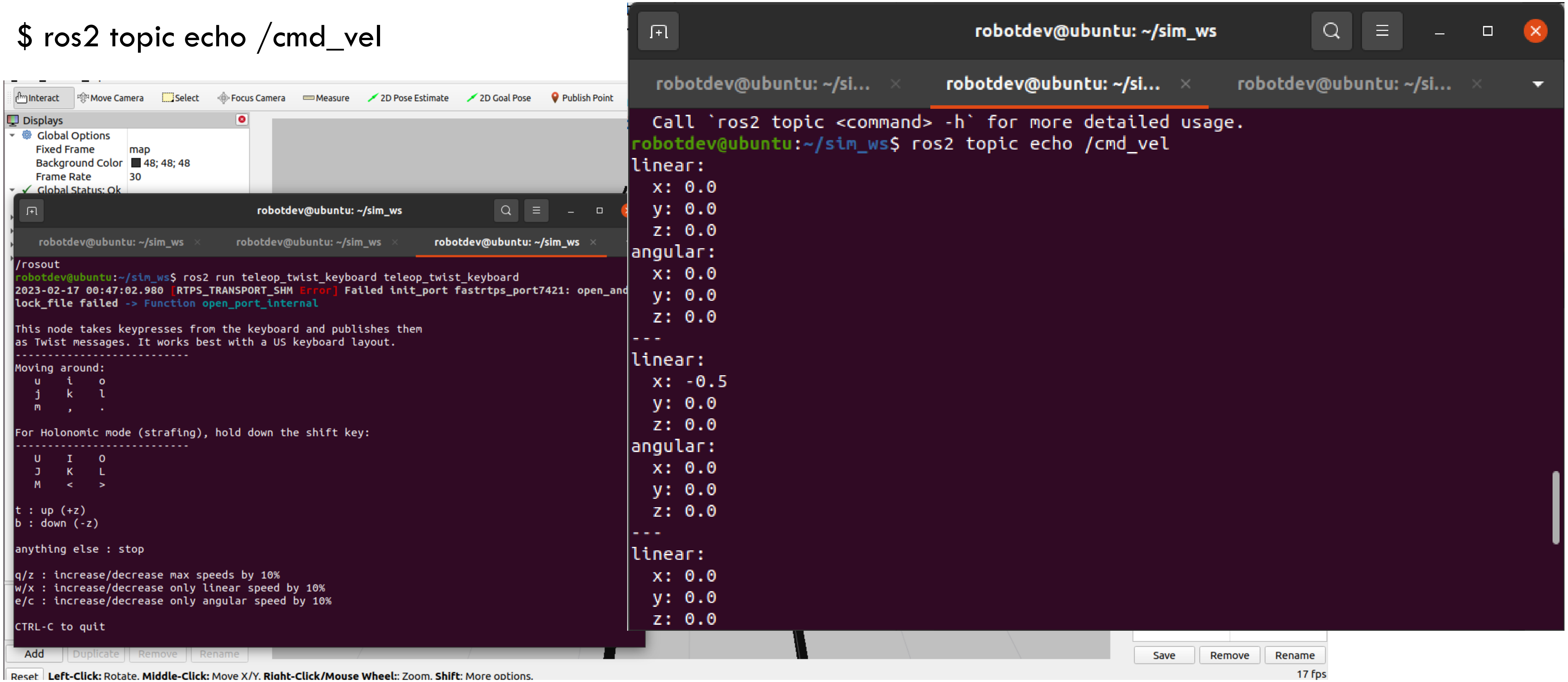
F1TENTH GYM (TELEOP)

$ ros2 run teleop_twist_keyboard teleop_twist_keyboard



F1TENTH GYM (TELEOP)

$ ros2 topic echo /cmd_vel



TOPIC /DRIVE

http://docs.ros.org/en/melodic/api/ackermann_msgs/html/msg/AckermannDriveStamped.html

ackermann_msgs/AckermannDriveStamped Message

File: ackermann_msgs/AckermannDriveStamped.msg

Raw Message Definition

```
## Time stamped drive command for robots with Ackermann steering.
#  $Id$

Header          header
AckermannDrive  drive
```

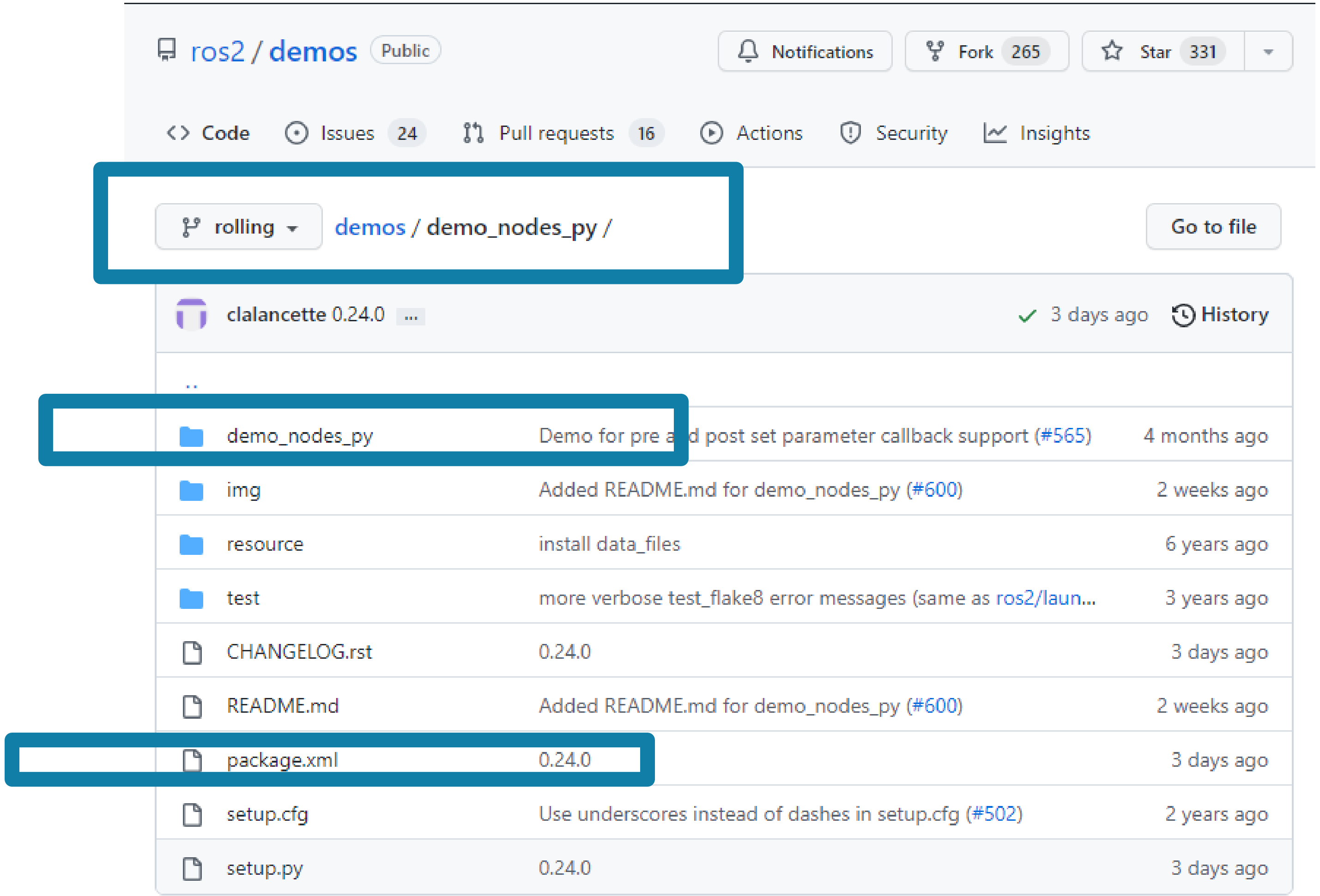
Compact Message Definition

```
std_msgs/Header header
ackermann_msgs/AckermannDrive drive
```

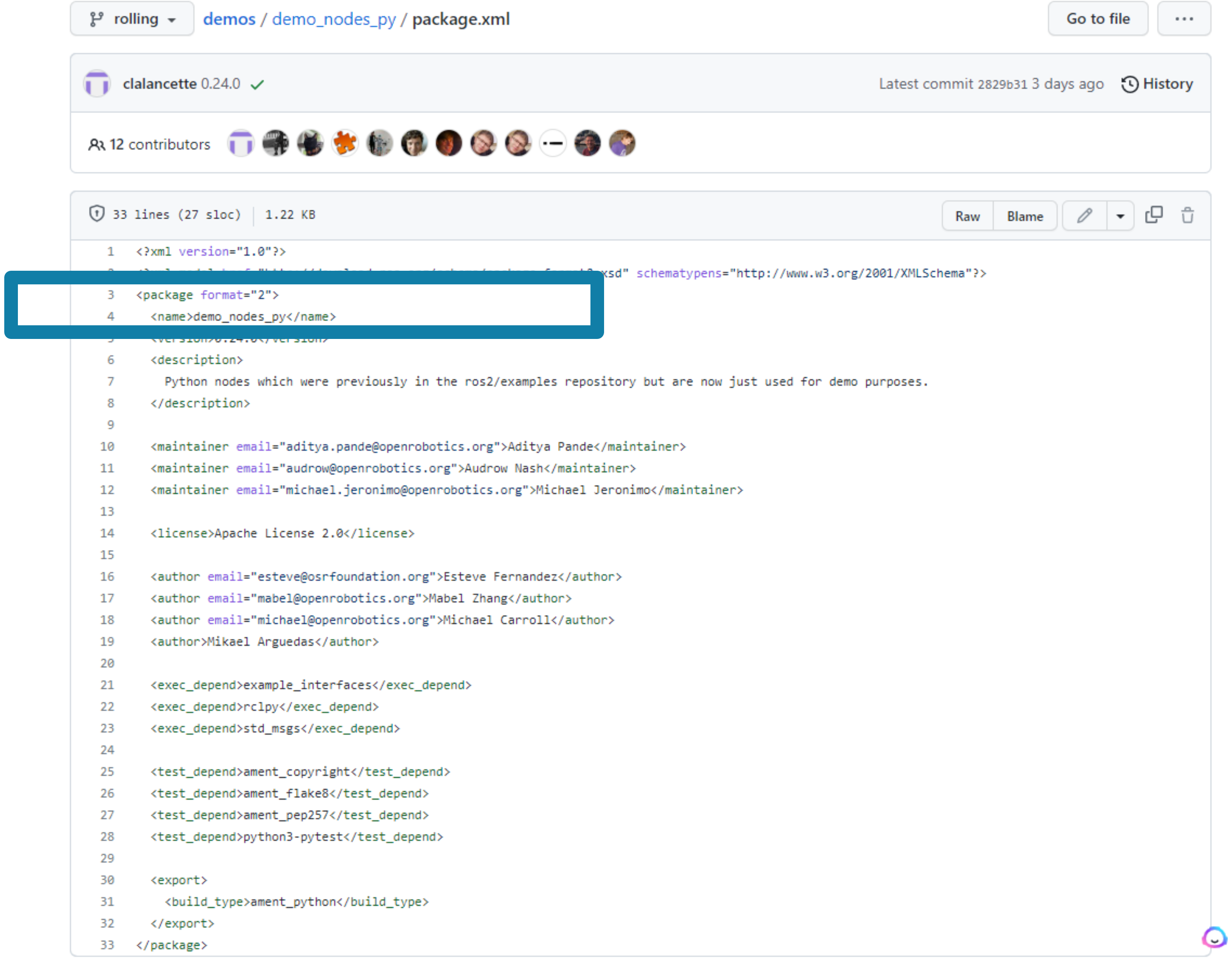
autogenerated on Mon, 28 Feb 2022 21:32:24

DEVELOPING PROGRAMS

ROS2 Demos
https://github.com/ros2/demos



PACKAGE.XML




```xml
<?xml version="1.0"?>
xsd" schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="2">
  <name>demo_nodes_py</name>

  <description>
    Python nodes which were previously in the ros2/examples repository but are now just used for demo purposes.
  </description>

  <maintainer email="aditya.pande@openrobotics.org">Aditya Pande</maintainer>
  <maintainer email="audrow@openrobotics.org">Audrow Nash</maintainer>
  <maintainer email="michael.jeronimo@openrobotics.org">Michael Jeronimo</maintainer>

  <license>Apache License 2.0</license>

  <author email="esteve@osrfoundation.org">Esteve Fernandez</author>
  <author email="mabel@openrobotics.org">Mabel Zhang</author>
  <author email="michael@openrobotics.org">Michael Carroll</author>
  <author>Mikael Arguedas</author>

  <exec_depend>example_interfaces</exec_depend>
  <exec_depend>rclpy</exec_depend>
  <exec_depend>std_msgs</exec_depend>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

CREATE A NEW PACKAGE

$ cd ~/sim_ws/src

$ ros2 pkg create my_robot_controller --build-type ament_python

```
robotdev@ubuntu:~/sim_ws/src$ ls
f1tenth_gym_ros
robotdev@ubuntu:~/sim_ws/src$ ros2 pkg create my_robot_controller --build-type ament_python
going to create a new package
package name: my_robot_controller
destination directory: /home/robotdev/sim_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['robotdev <Fred.Livingston@gmail.com>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: []
creating folder ./my_robot_controller
creating ./my_robot_controller/package.xml
creating source folder
creating folder ./my_robot_controller/my_robot_controller
creating ./my_robot_controller/setup.py
creating ./my_robot_controller/setup.cfg
creating folder ./my_robot_controller/resource
creating ./my_robot_controller/resource/my_robot_controller
creating ./my_robot_controller/my_robot_controller/__init__.py
creating folder ./my_robot_controller/test
creating ./my_robot_controller/test/test_copyright.py
creating ./my_robot_controller/test/test_flake8.py
creating ./my_robot_controller/test/test_pep257.py
robotdev@ubuntu:~/sim_ws/src$
```

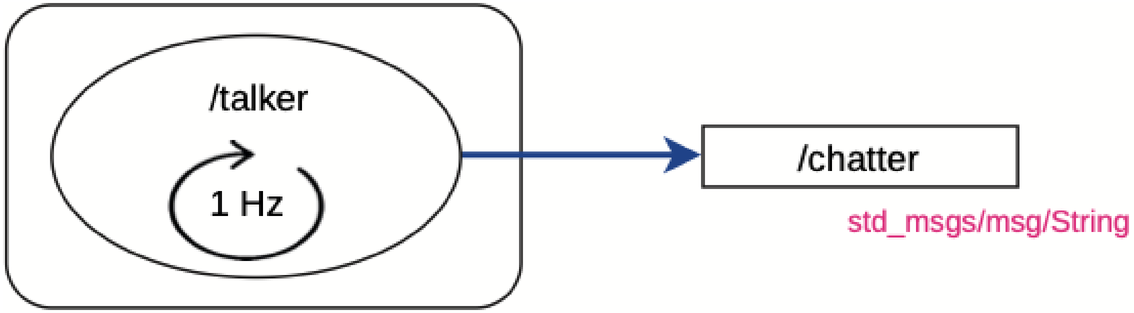
SIMPLE PUBLISHER (TALKER.PY)

https://github.com/ros2/demos/blob/rolling/demo_nodes_py/demo_nodes_py/topics/talker.py

Line 25:
super().__init__('NAME_OF_PROCESS')

Line 27:
self.create_publisher(String, 'chatter', 10)

Line 29:
self.create_timer(1, self.time_call_back)




```
14
15  import rclpy
16  from rclpy.executors import ExternalShutdownException
17  from rclpy.node import Node
18
19  from std_msgs.msg import String
20
21
22  class Talker(Node):
23
24      def __init__(self):
25          super().__init__('talker')
26          self.i = 0
27          self.pub = self.create_publisher(String, 'chatter', 10)
28          timer_period = 1.0
29          self.tmr = self.create_timer(timer_period, self.timer_callback)
30
31      def timer_callback(self):
32          msg = String()
33          msg.data = 'Hello World: {0}'.format(self.i)
34          self.i += 1
35          self.get_logger().info('Publishing: "{0}"'.format(msg.data))
36          self.pub.publish(msg)
37
```

CREATE A PYTHON PROGRAM

$ cd my_robot_controller/my_robot_controller/
$ touch move_robot.py
$ gedit move_robot.py

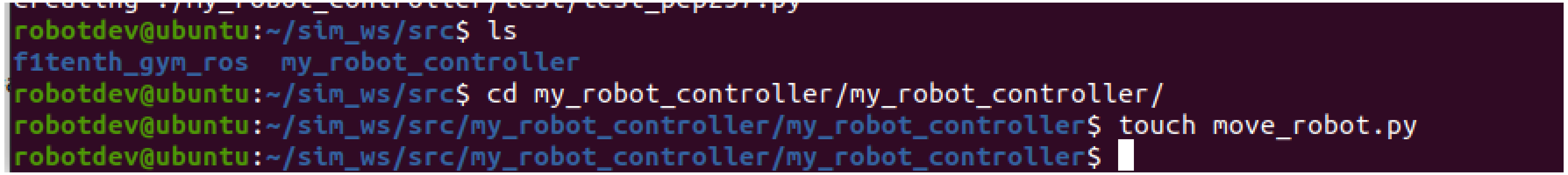
```
robotdev@ubuntu:~/sim_ws/src$ ls
f1tenth_gym_ros  my_robot_controller
robotdev@ubuntu:~/sim_ws/src$ cd my_robot_controller/my_robot_controller/
robotdev@ubuntu:~/sim_ws/src/my_robot_controller/my_robot_controller$ touch move_robot.py
robotdev@ubuntu:~/sim_ws/src/my_robot_controller/my_robot_controller$
```

MOVE_ROBOT.PY

$ cd my_robot_controller/my_robot_controller/

$ touch move_robot.py

$ gedit move_robot.py



MOVE_ROBOT.PY

move_robot.py ×

home > robotdev > sim_ws > src > my_robot_controller > my_robot_controller > move_robot.py > ...

```python
# move_robot.py
# Fred Livingston (fjliving@ncsu.edu) 2-17-2023

import rclpy
from rclpy.executors import ExternalShutdownException
from rclpy.node import Node

from geometry_msgs.msg import Twist


class Controller(Node):

    def __init__(self):
        super().__init__('move_robot')
        self.pub = self.create_publisher(Twist, 'cmd_vel', 10)

        # move robot fwd
        msg = Twist()
        msg.linear.x = 0.5
        msg.linear.y = 0.0
        msg.linear.z = 0.0
        msg.angular.x = 0.0
        msg.angular.y = 0.0
        msg.angular.z = 0.0
        self.pub.publish(msg)

        timer_period = 10.0
        self.tmr = self.create_timer(timer_period, self.timer_callback)

    def timer_callback(self):
        # stop robot
        msg = Twist()
        msg.linear.x = 0.0
        msg.linear.y = 0.0
        msg.linear.z = 0.0
        msg.angular.x = 0.0
        msg.angular.y = 0.0
        msg.angular.z = 0.0
        self.pub.publish(msg)


def main(args=None):
    rclpy.init(args=args)

    node = Controller()

    try:
        rclpy.spin(node)
    except (KeyboardInterrupt, ExternalShutdownException):
        pass
    finally:
        node.destroy_node()
        rclpy.try_shutdown()


if __name__ == '__main__':
    main()
```

SETUP.PY

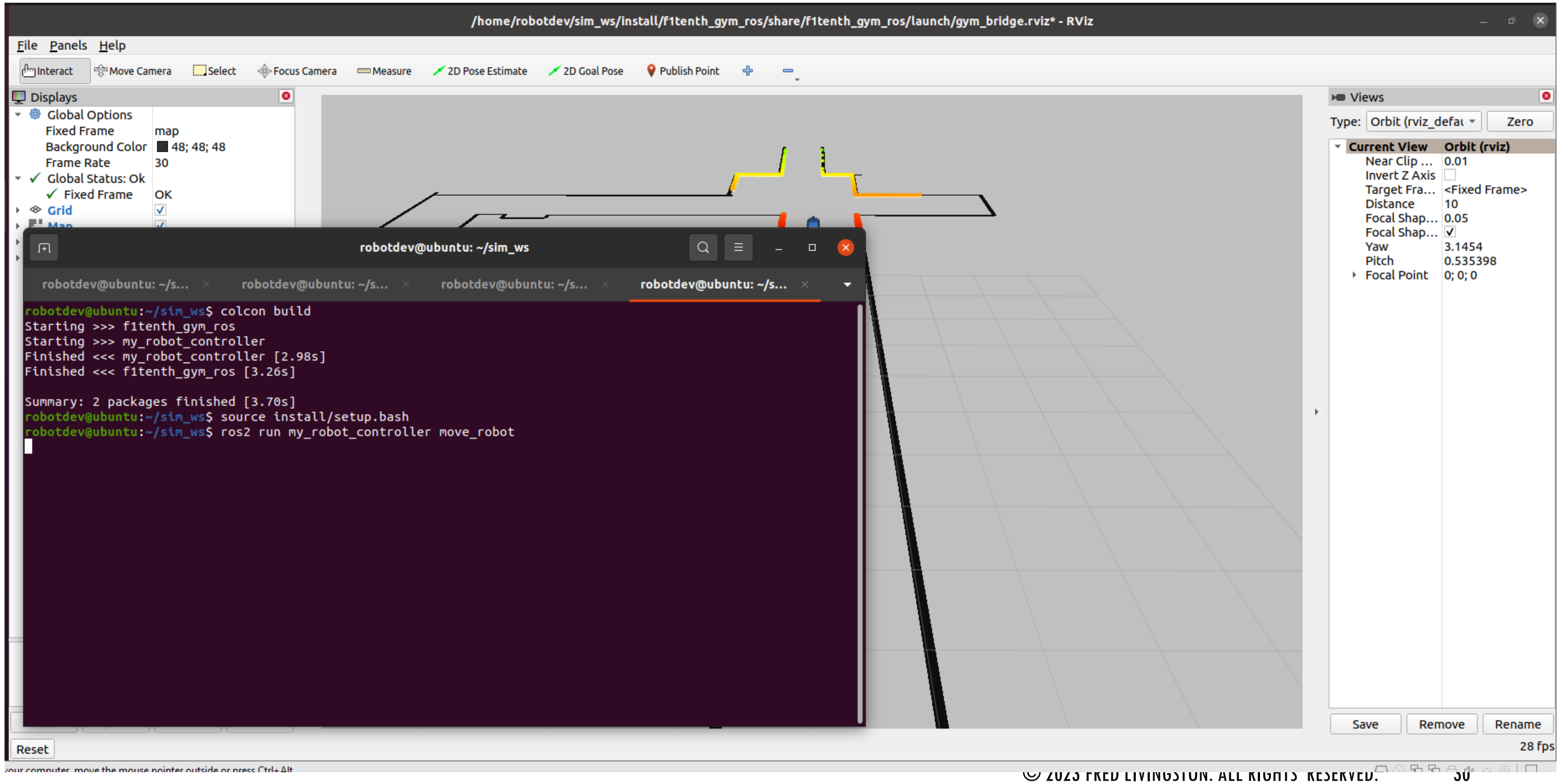
setup.py
~/sim_ws/src/my_robot_controller

```python
from setuptools import setup

package_name = 'my_robot_controller'

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']),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='robotdev',
    maintainer_email='Fred.Livingston@gmail.com',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        'move_robot = my_robot_controller.move_robot:main'
        ],
    },
)
```

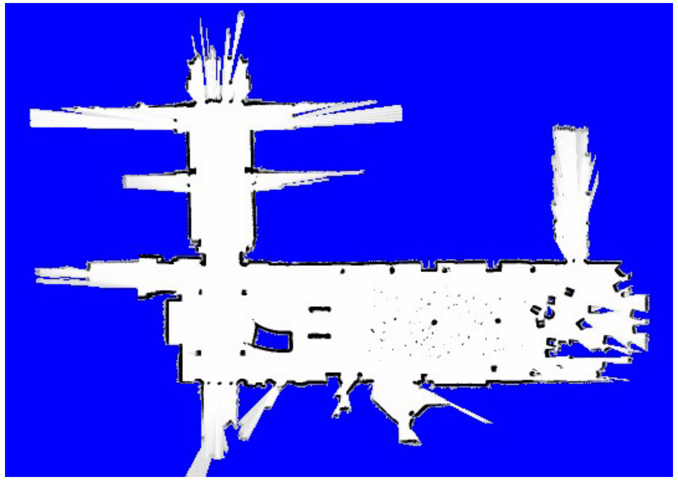
Python Tab Width: 8 Ln 23, Col 20

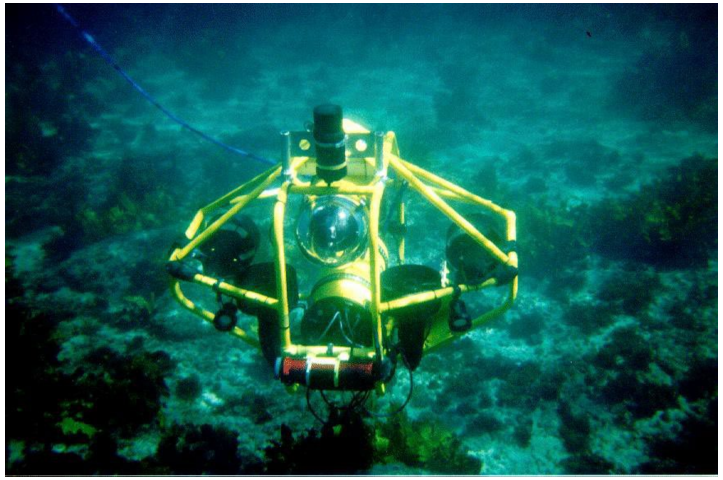
BUILD AND EXCUTE ROBOT CONTROLLER

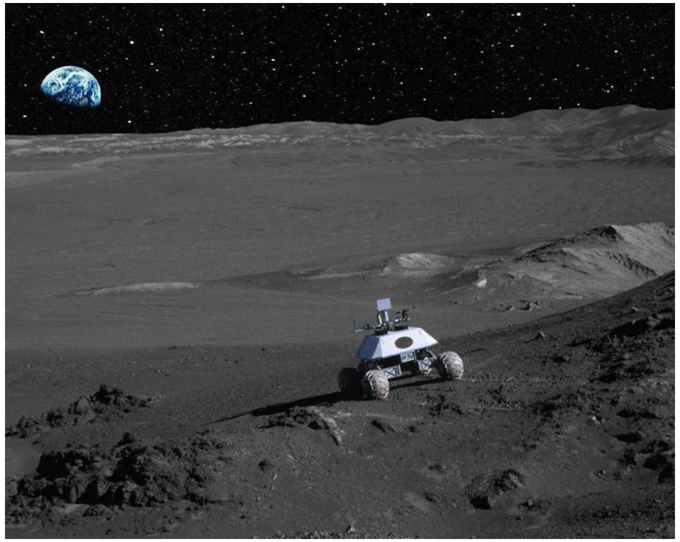
$ colcon build
$ source install/setup.bash
$ ros2 run my_robot_controller move_robot



SLAM – SIMULTANEOUS LOCALIZATION & MAPPING







SLAM is a technique used to build up a map within an unknown environment or a known environment while at the same time keeping track of the current location.

WHAT IS SLAM

- The problem has 2 stages
  - Mapping
  - Localization
- The paradox:
  - In order to build a map, we must know our position
  - To determine our position, we need a map!
- SLAM is like the chicken-egg problem
- Solution is to alternate between the two steps.

SLALM – MULTIPLE PARTS

- Landmark extraction
- data association
- State estimation
- state update
- landmark update

There are many ways to solve each of the smaller parts

THE GOAL OF THE PROCESS

The SLAM process consists of number of steps.

- Use environment to update the position of the robot. Since the odometry of the robot is often erroneous we cannot rely directly on the odometry.
- We can use laser scans of the environment to correct the position of the robot.
- This is accomplished by extracting features from the environment and re observing when the robot moves around.

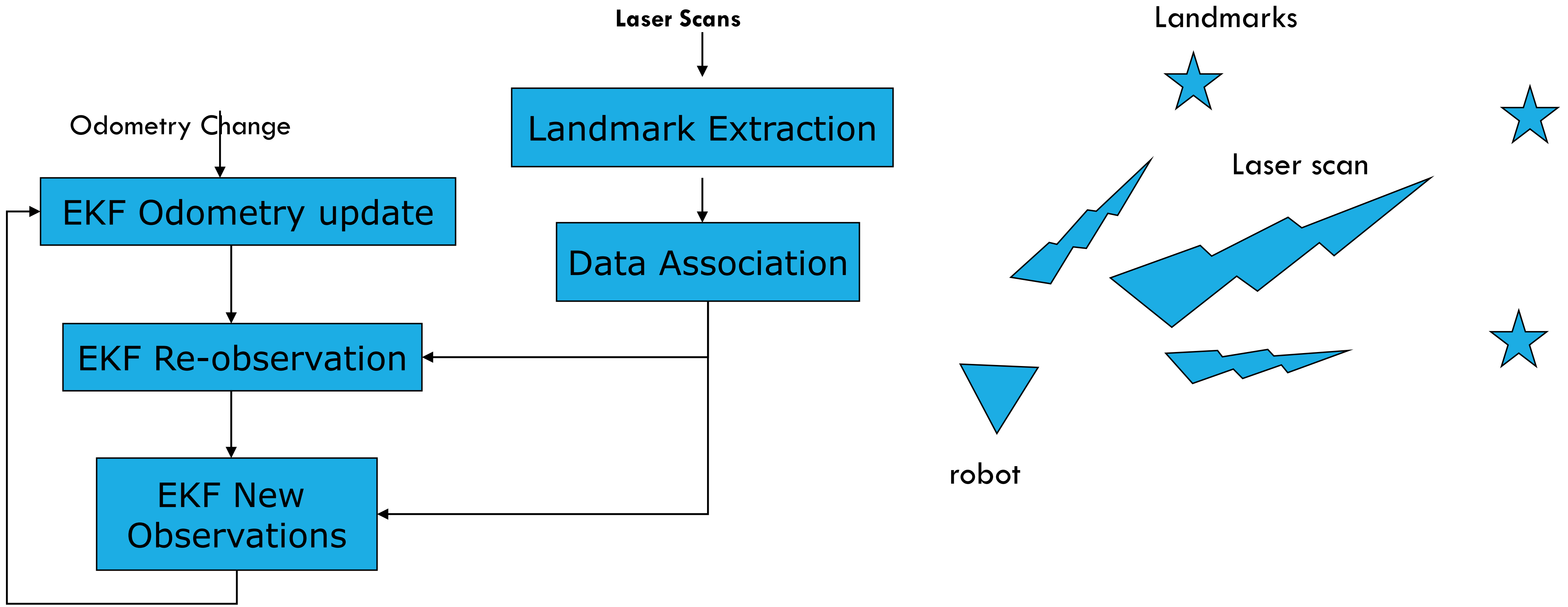
EXTENDED KALMAN FILTER

An EKF (Extended Kalman Filter) is the heart of the SLAM process.

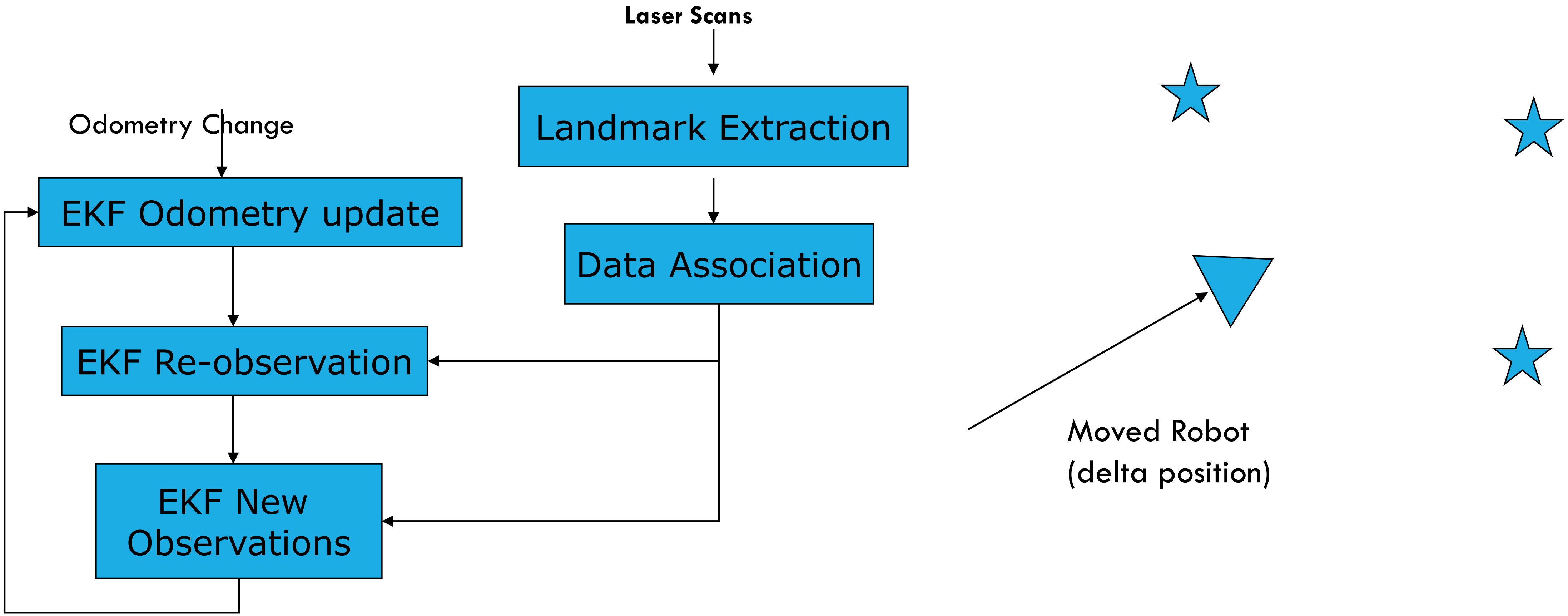
- It is responsible for updating where the robot thinks it is based on the Landmarks (features).

- The EKF keeps track of an estimate of the uncertainty in the robots position and also the uncertainty in these landmarks it has seen in the environment.

SLAM OVERVIEW

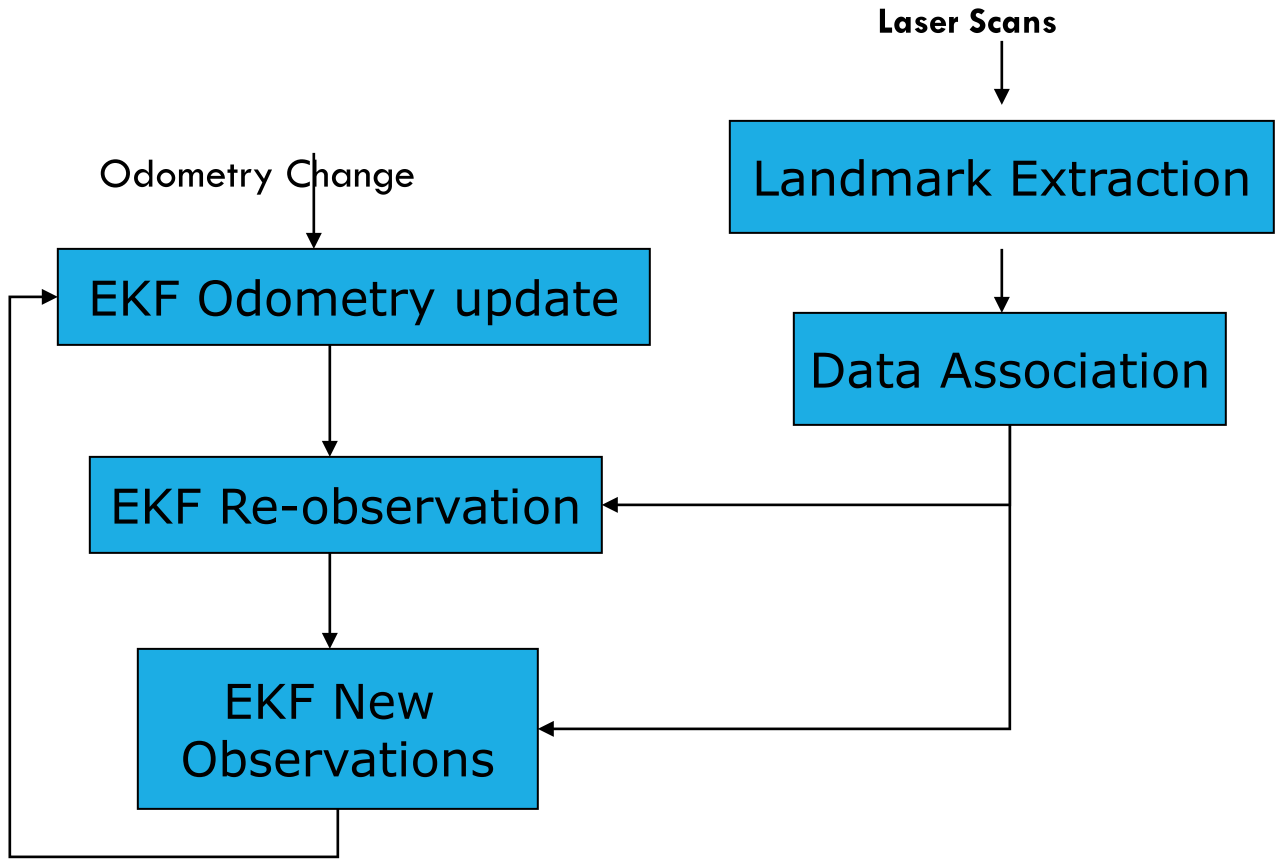


OVERVIEW

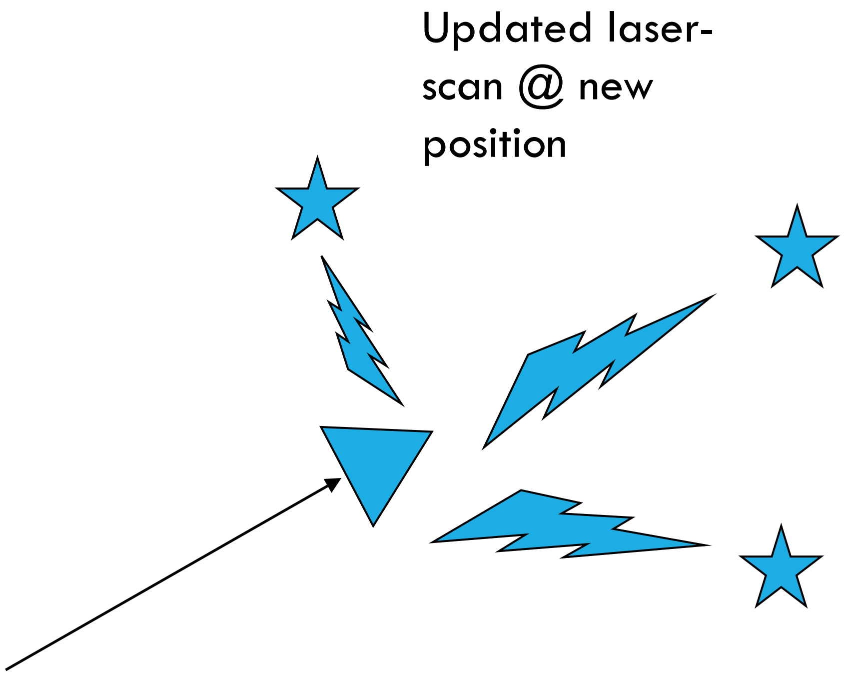


Laser Scans
Odometry Change
Landmark Extraction
EKF Odometry update
Data Association
EKF Re-observation
EKF New Observations
Moved Robot
(delta position)

OVERVIEW

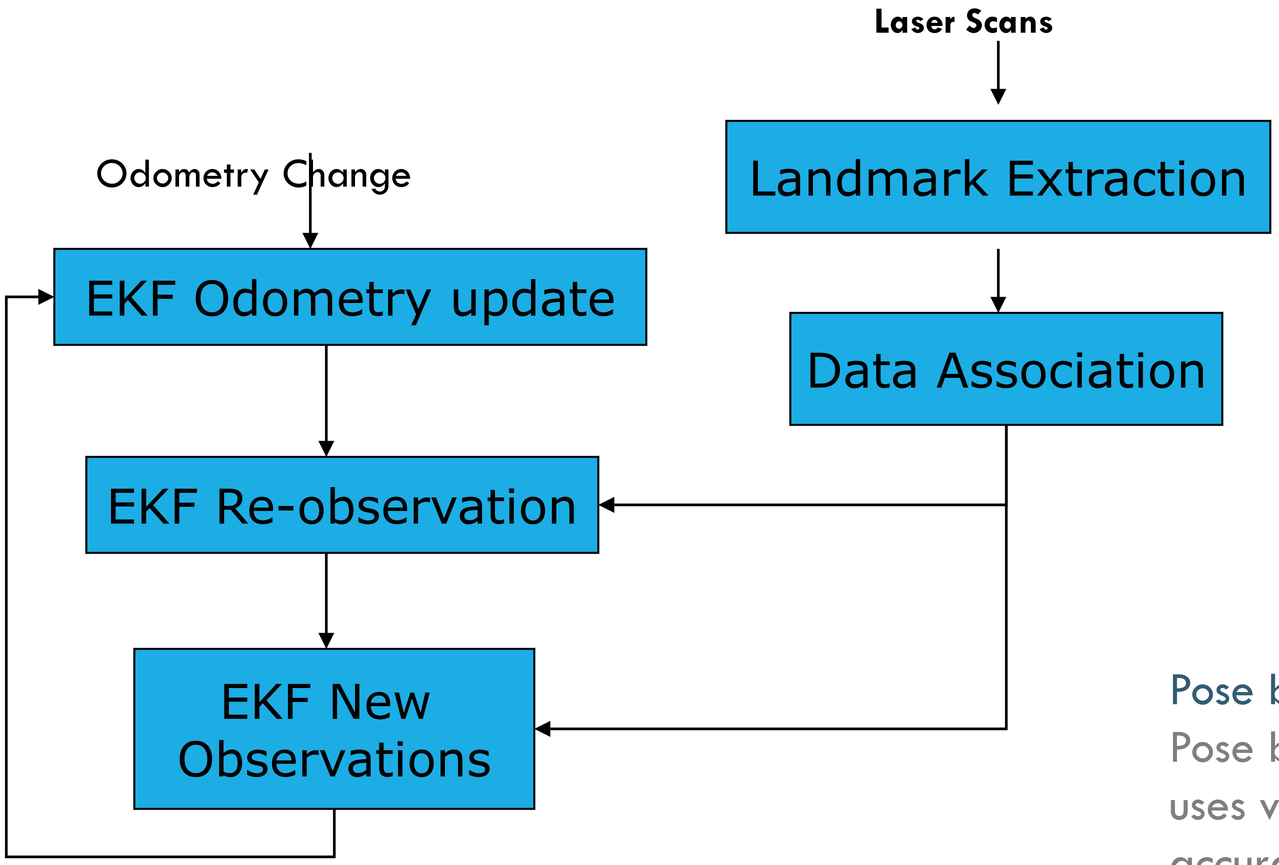


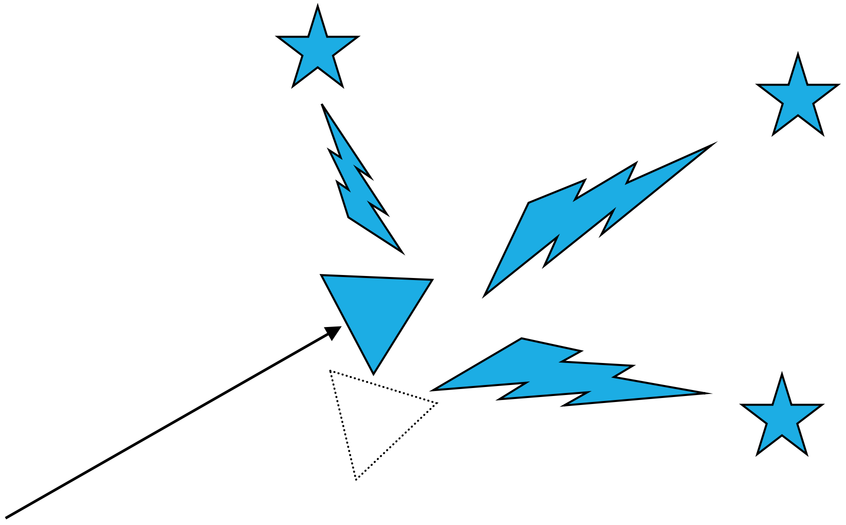
Laser Scans
Odometry Change
Landmark Extraction
EKF Odometry update
Data Association
EKF Re-observation
EKF New Observations



Updated laser-scan @ new position

OVERVIEW

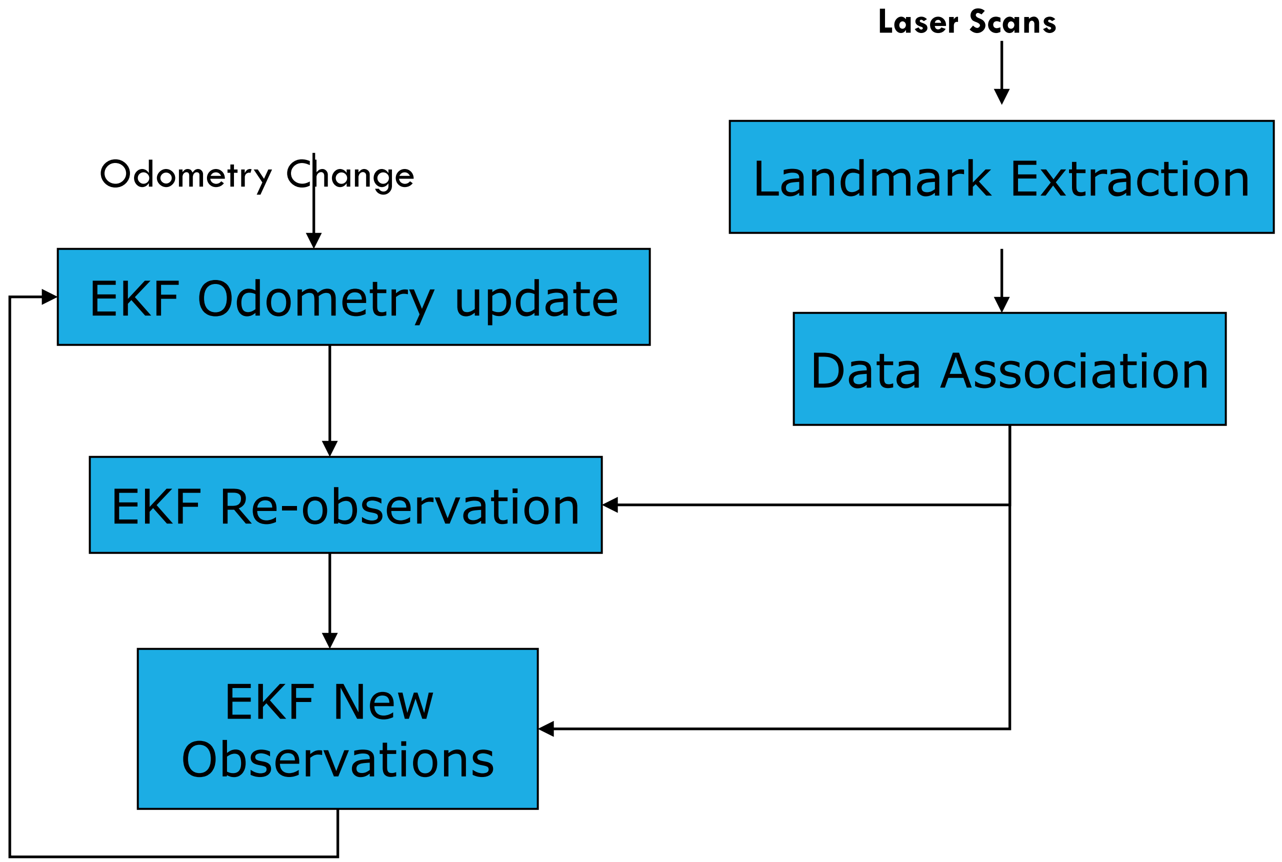




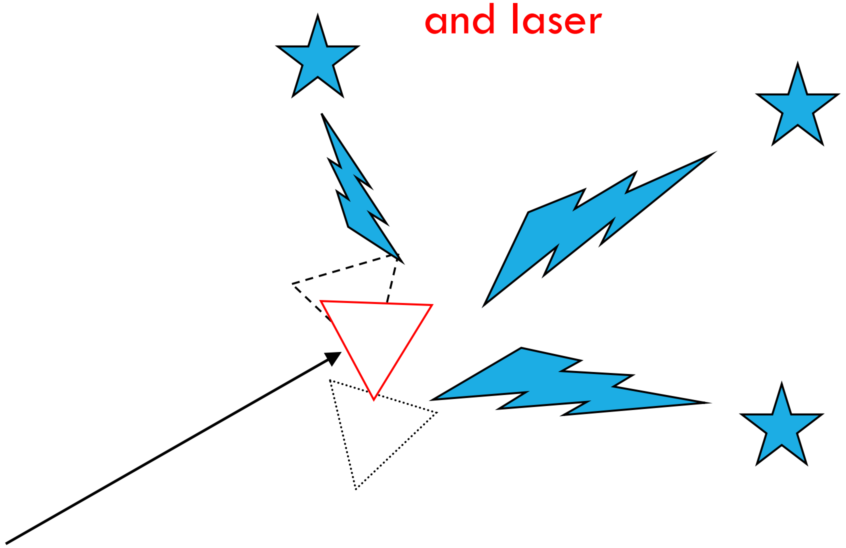
Pose based on laser data

Pose based on using odometry, note that odometry uses velocity to compute delta position and is less accurate than laser

OVERVIEW



EKF to estimate robot position using odometry and laser

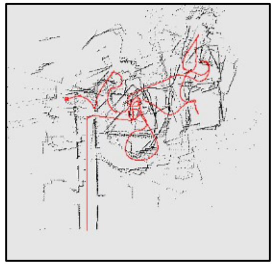


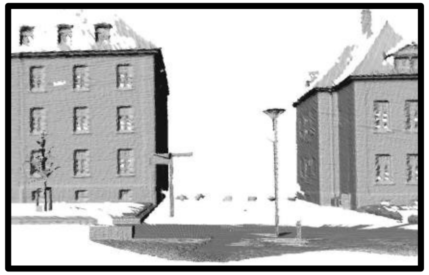
LASER AND ODOMETRY DATA

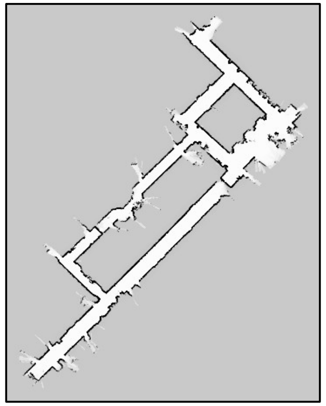
- Laser data is the reading obtained from the scan
- The goal of the odometry data is to provide an approximate position of the robot
- The difficult part about the odometry data and the laser data is to get the timing right.

REPRESENTATION

- Grid maps or scans

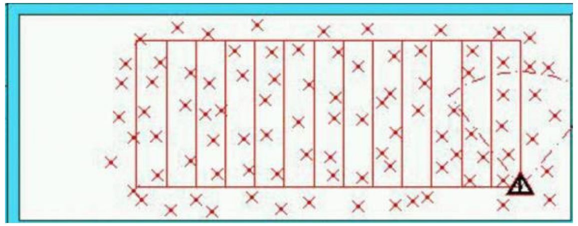


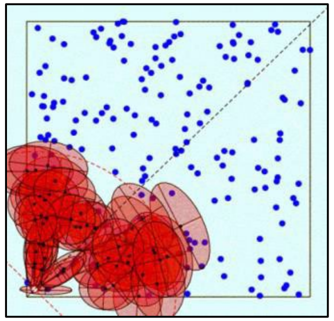


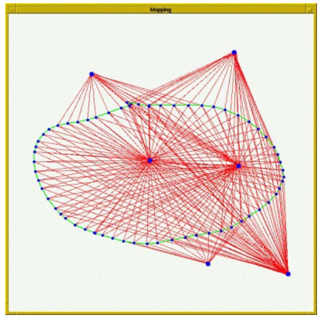


[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

- Landmark-based







[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

LANDMARKS

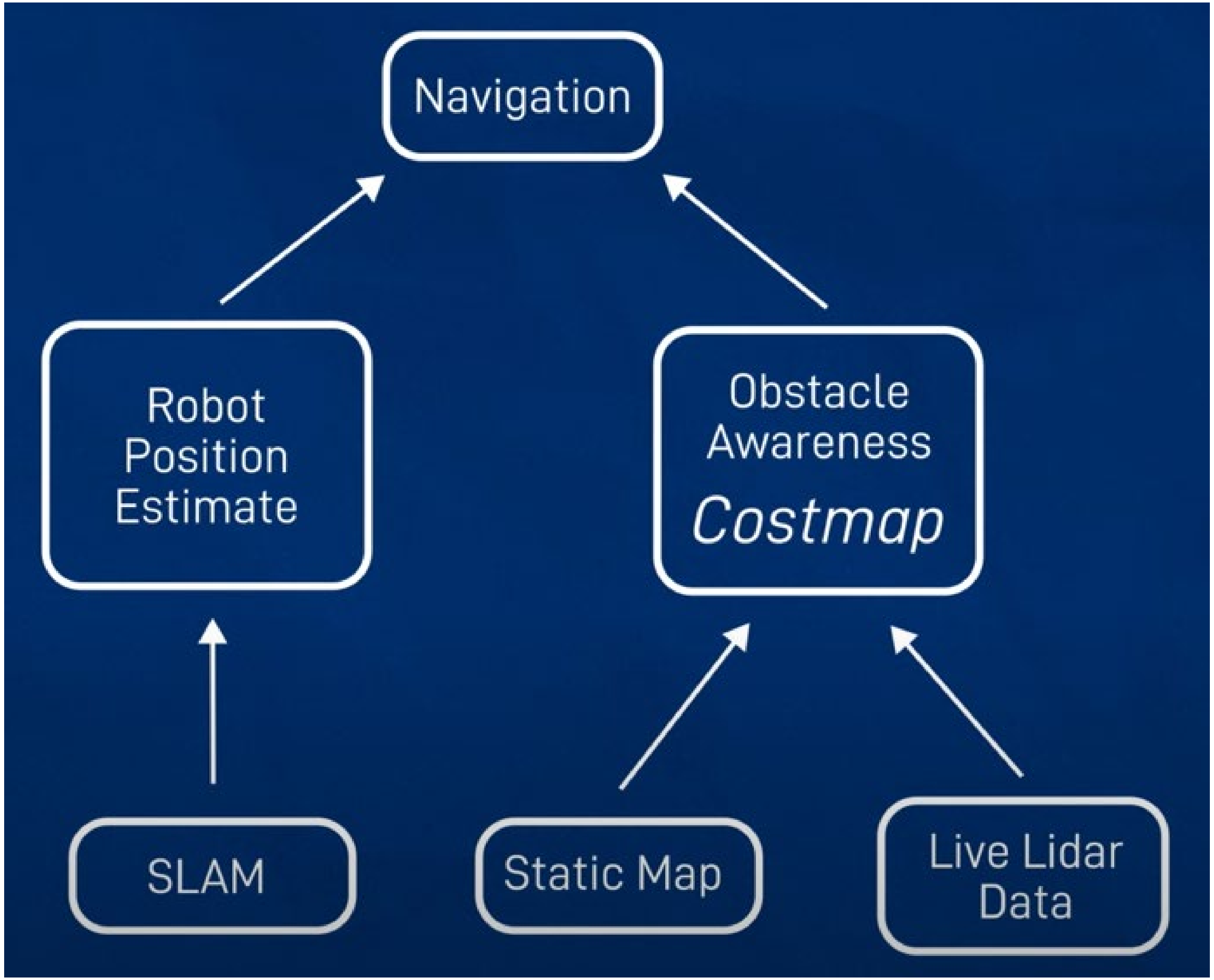
Landmarks are features which can easily be re-observed and distinguished from the environment.

These are used by the robot to find out where it is (to localize itself).

KEY POINTS ABOUT SUITABLE LANDMARKS

- Landmarks should be easily re observable.
- Individual landmarks should be distinguishable from each other.
- Landmarks should be plentiful in the environment.
- Landmarks should be stationary.

AUTONOMOUS NAVIGATION



WORK IN-PROGRESS

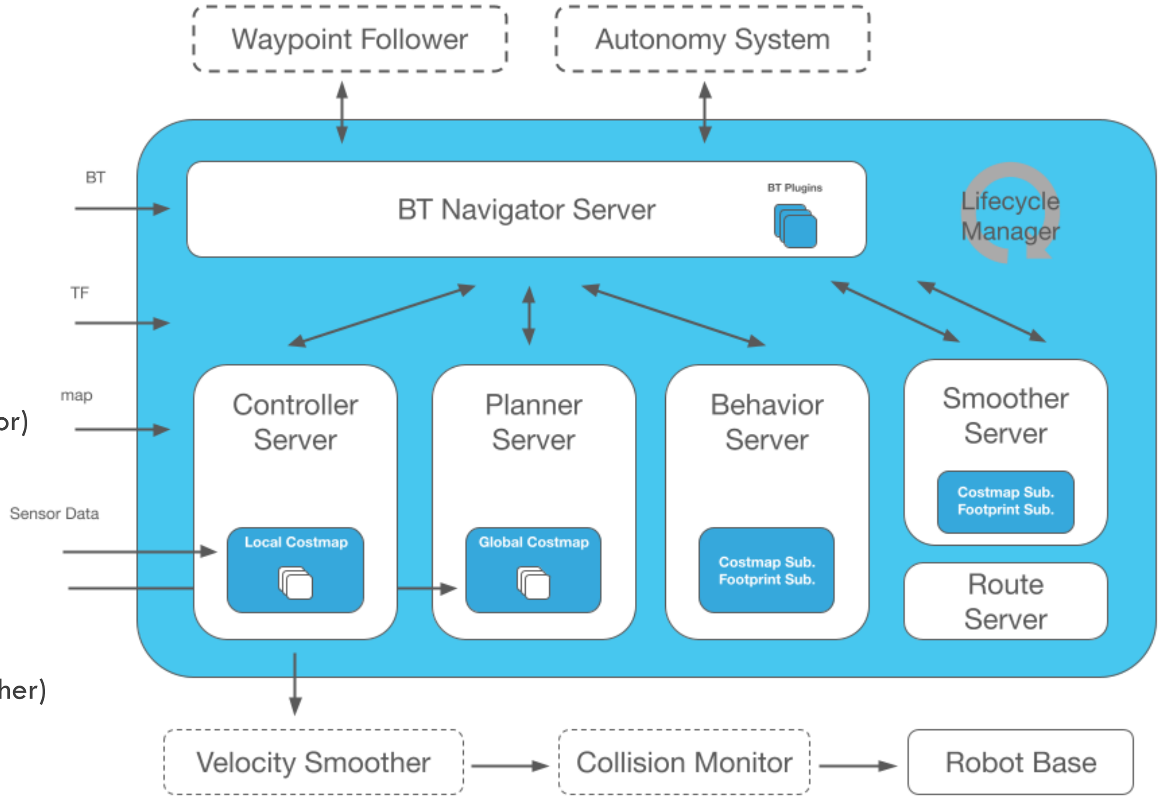
- WS 003 – Autonomous Navigation [TBD]

ROS NAV2

https://navigation.ros.org/

It has tools to:

•***Load, serve, and store maps (Map Server)***
•***Localize the robot on the map (AMCL)***
•***Plan a path from A to B around obstacles (Nav2 Planner)***
•Control the robot as it follows the path (Nav2 Controller)
•Smooth path plans to be more continuous and feasible (Nav2 Smoother)
•***Convert sensor data into a costmap representation of the world (Nav2 Costmap 2D)***
•Build complicated robot behaviors using behavior trees (Nav2 Behavior Trees and BT Navigator)
•Compute recovery behaviors in case of failure (Nav2 Recoveries)
•***Follow sequential waypoints (Nav2 Waypoint Follower)***
•Manage the lifecycle and watchdog for the servers (Nav2 Lifecycle Manager)
•Plugins to enable your own custom algorithms and behaviors (Nav2 Core)
•Monitor raw sensor data for imminent collision or dangerous situation (Collision Monitor)
•Python3 API to interact with Nav2 in a pythonic manner (Simple Commander)
•A smoother on output velocities to guarantee dynamic feasibility of commands (Velocity Smoother)



INSTALLATION

SLAM Toolbox

$ sudo apt install ros-foxy-slam-toolbox

ROS NAV2

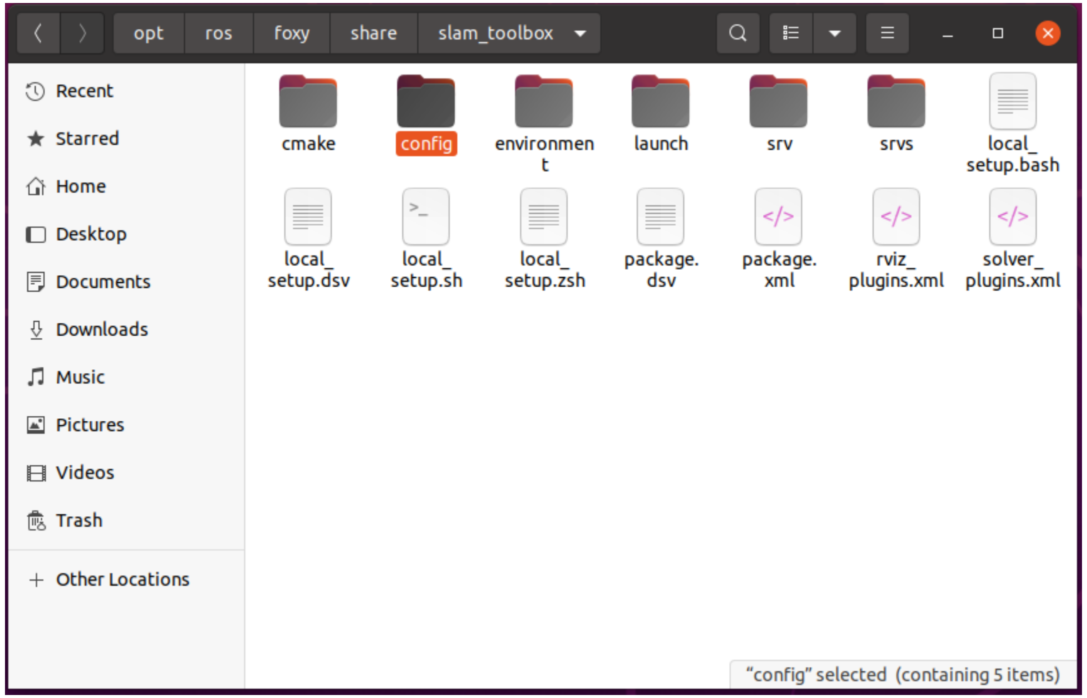
$ sudo apt install ros-foxy-navigation2

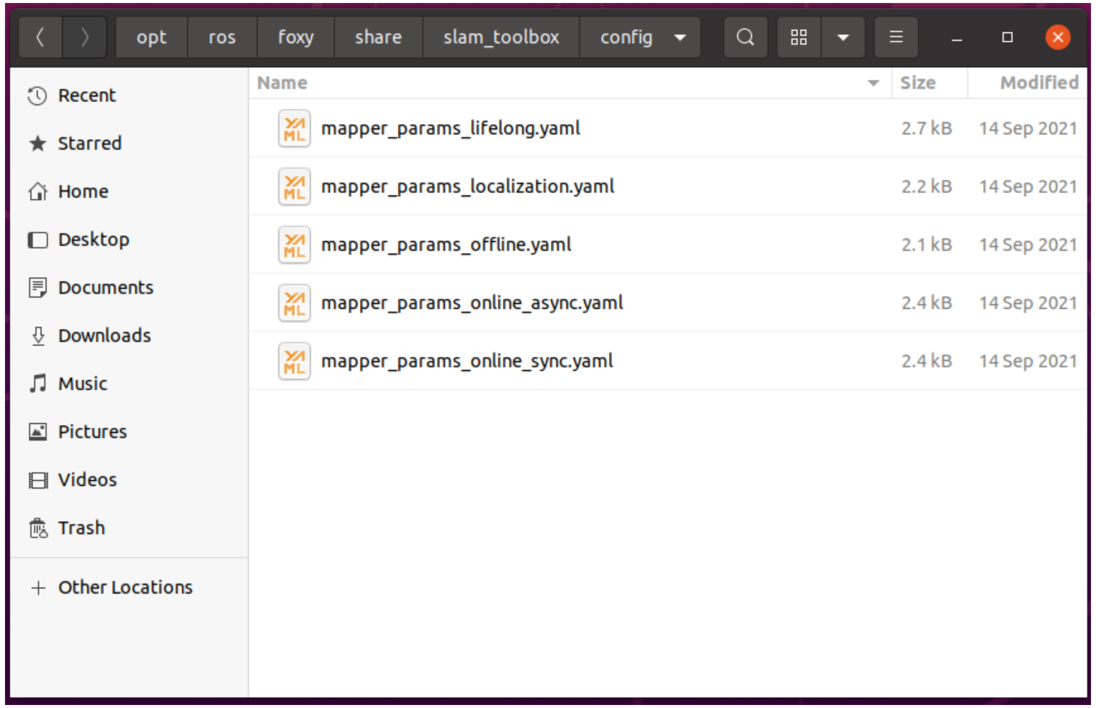
$ sudo apt install ros-foxy-nav2-bringup

$ sudo apt install ros-foxy-twist-mux

CONFIGURING SLAM TOOLBOX

/opt/ros/foxy/share/slam_toolbox





CONFIGURING SLAM TOOLBOX [MAPPING]

$ sudo nano /opt/ros/foxy/share/slam_toolbox/config/mapper_params_online_async.yaml

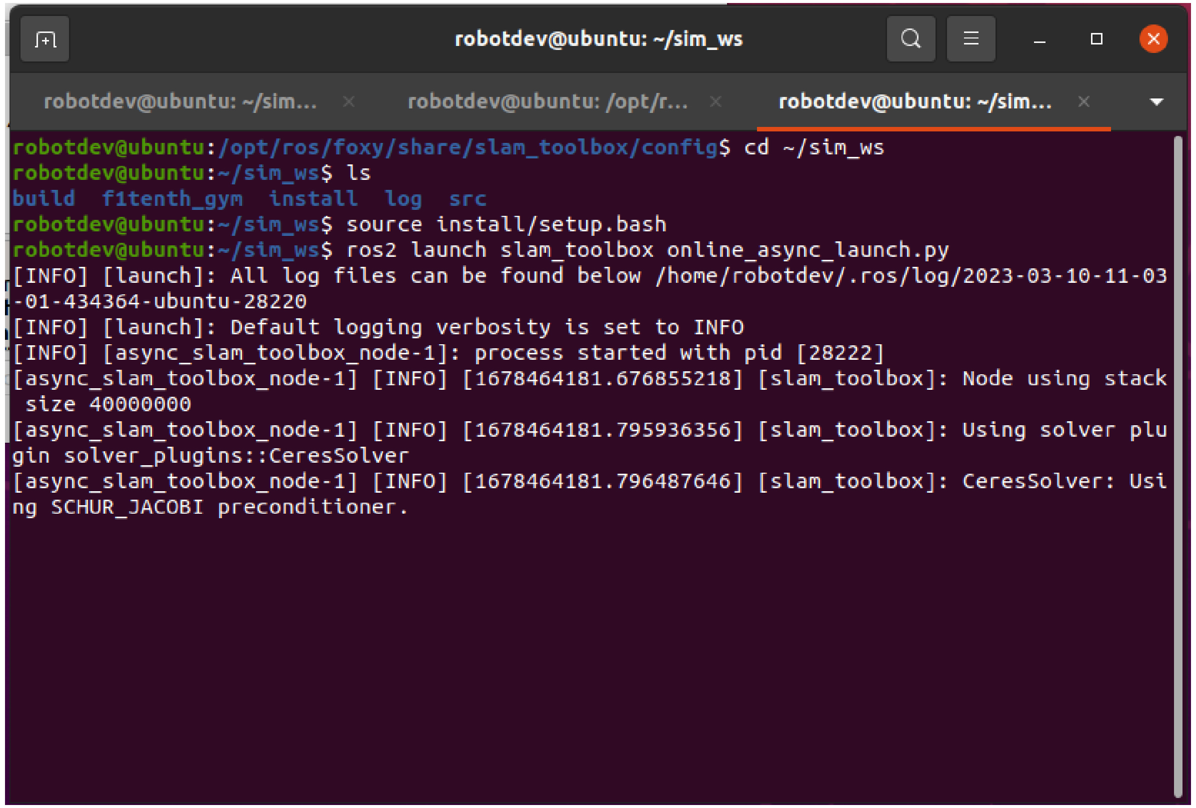
```
robotdev@ubuntu: ~/sim_ws
robotdev@ubuntu: ~/sim_ws    robotdev@ubuntu: ~/sim_ws    robotdev@ubuntu: ~/sim_ws
      /opt/ros/foxy/share/slam_toolbox/config/mapper_params_online_async.yaml      Modified
slam_toolbox:
  ros__parameters:

    # Plugin params
    solver_plugin: solver_plugins::CeresSolver
    ceres_linear_solver: SPARSE_NORMAL_CHOLESKY
    ceres_preconditioner: SCHUR_JACOBI
    ceres_trust_strategy: LEVENBERG_MARQUARDT
    ceres_dogleg_type: TRADITIONAL_DOGLEG
    ceres_loss_function: None

    # ROS Parameters
    odom_frame: ego_racecar/odom
    map_frame: map
    base_frame: base_link
    scan_topic: /scan
    mode: mapping #localization

    # if you'd like to immediately start continuing a map at a given pose
    # or at the dock, but they are mutually exclusive, if pose is given

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```

RUNING SLAM TOOLBOX [MAPPING]

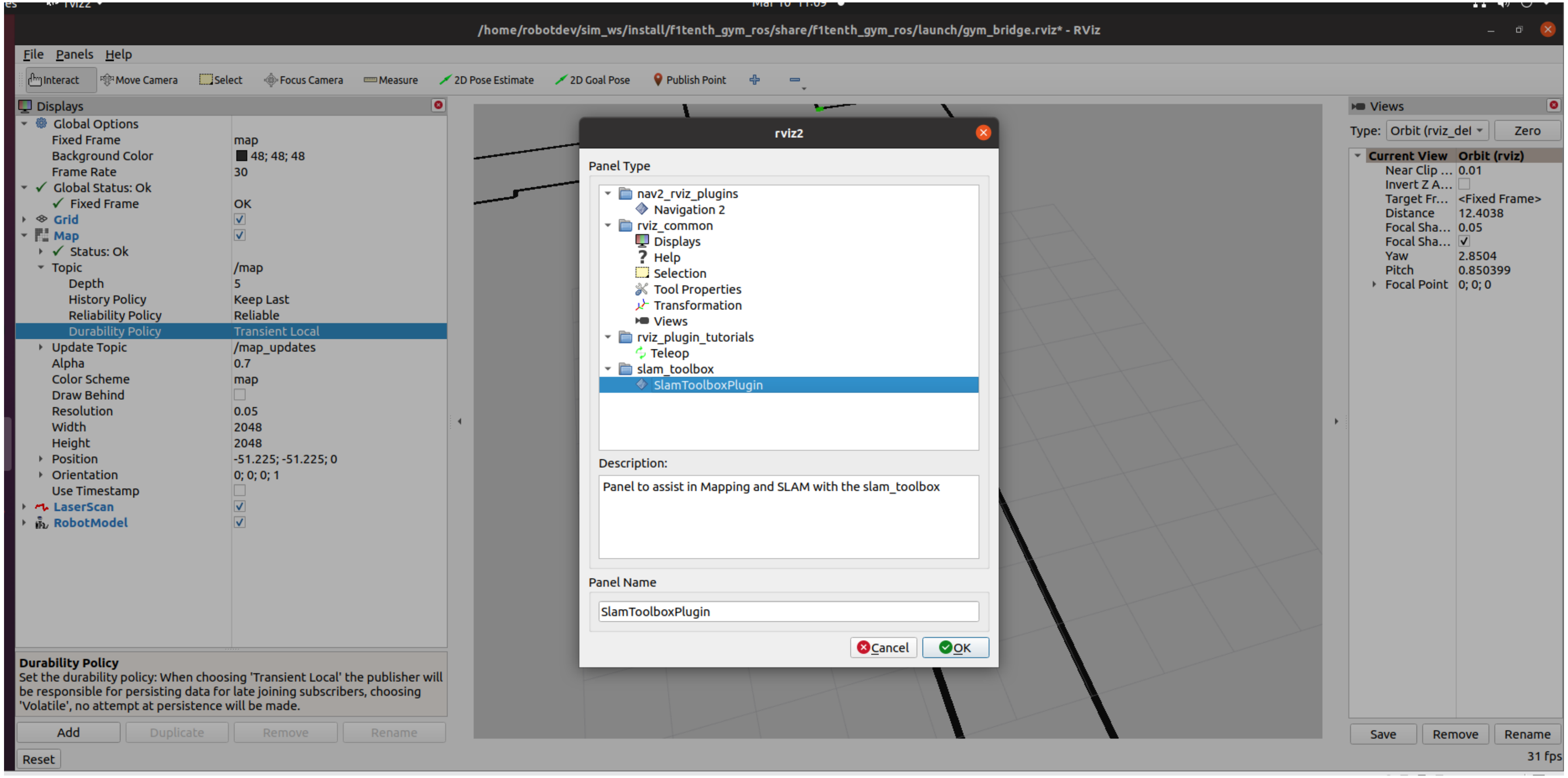
$ cd ~/sim_ws

$ source install/setup.bash

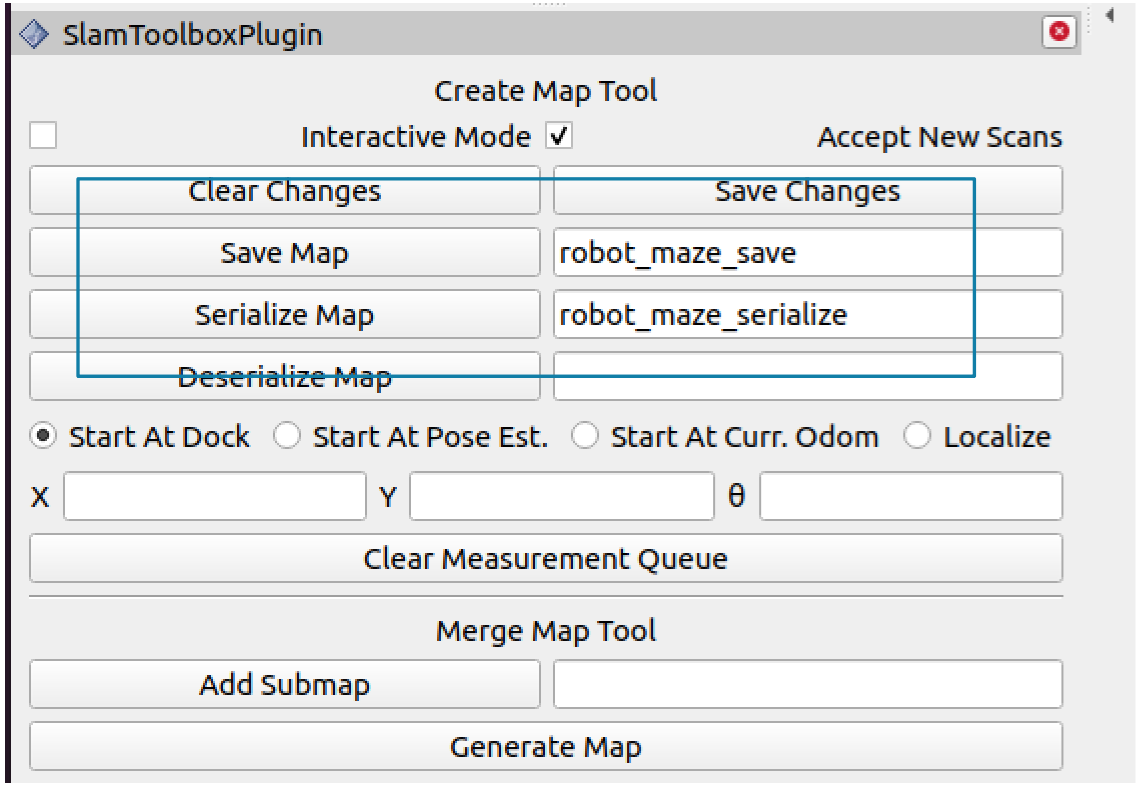
$ ros2 launch slam_toolbox online_async_launch.py

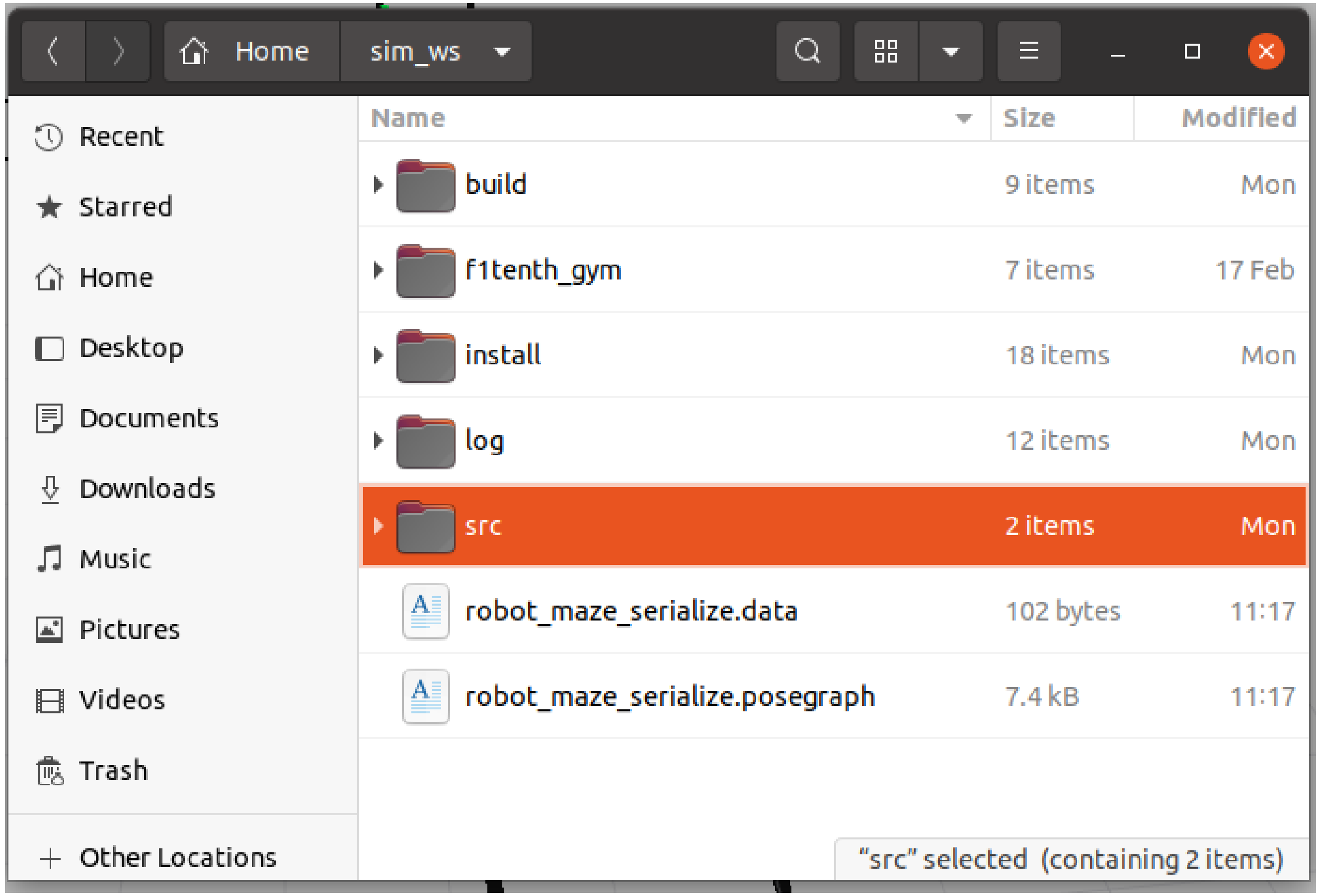


SLAM TOOLBOX PLUGIN



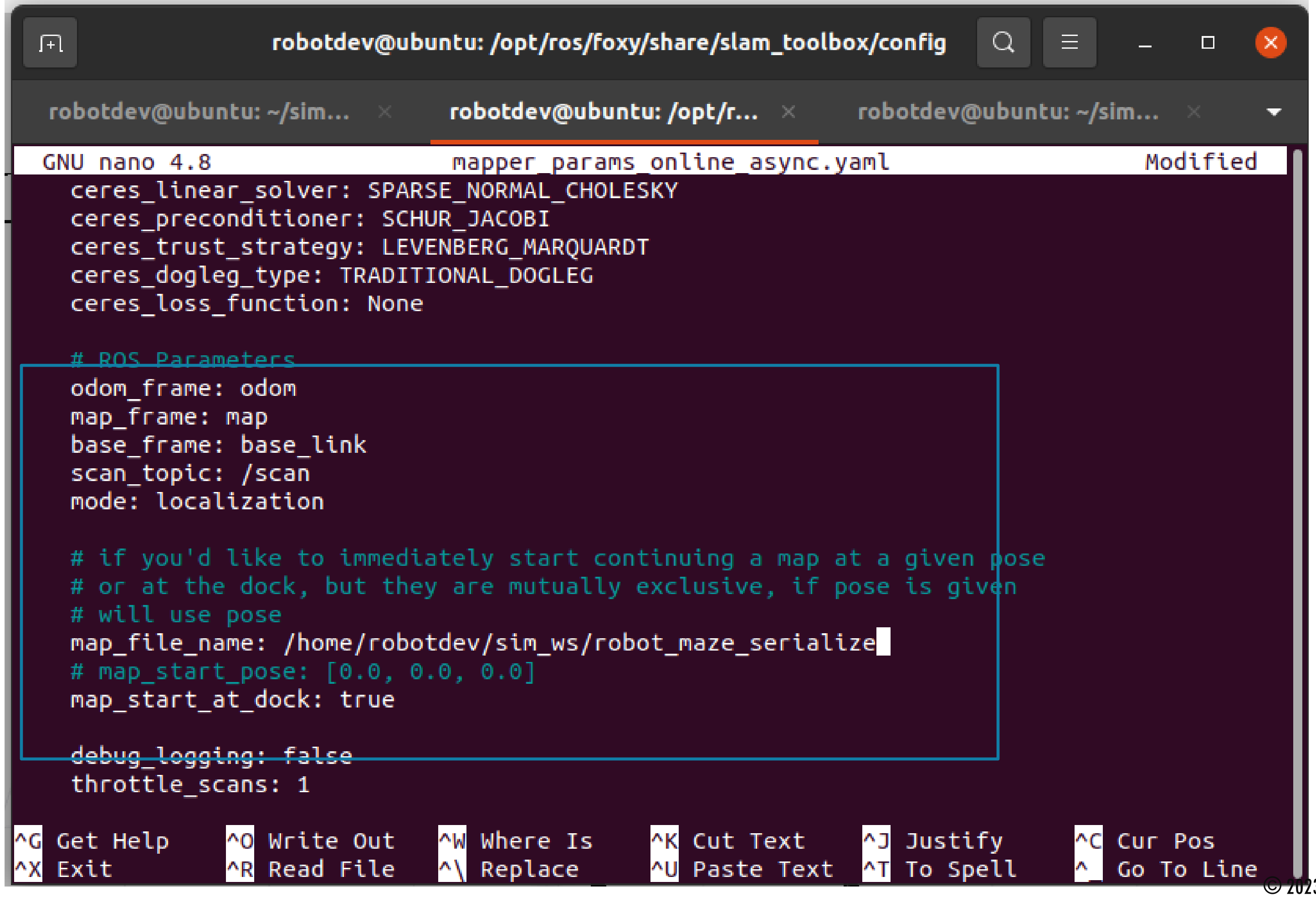
SAVE MAP


SlamToolboxPlugin
Create Map Tool
Interactive Mode
Accept New Scans
Clear Changes
Save Changes
Save Map
robot_maze_save
Serialize Map
robot_maze_serialize
Deserialize Map
Start At Dock
Start At Pose Est.
Start At Curr. Odom
Localize
X
Y
θ
Clear Measurement Queue
Merge Map Tool
Add Submap
Generate Map


Home
sim_ws
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
Name
Size
Modified
build 9 items Mon
f1tenth_gym 7 items 17 Feb
install 18 items Mon
log 12 items Mon
src 2 items Mon
robot_maze_serialize.data 102 bytes 11:17
robot_maze_serialize.posegraph 7.4 kB 11:17
"src" selected (containing 2 items)

CONFIGURING SLAM TOOLBOX [LOCALIZATION]

$ sudo nano /opt/ros/foxy/share/slam_toolbox/config/mapper_params_online_async.yaml

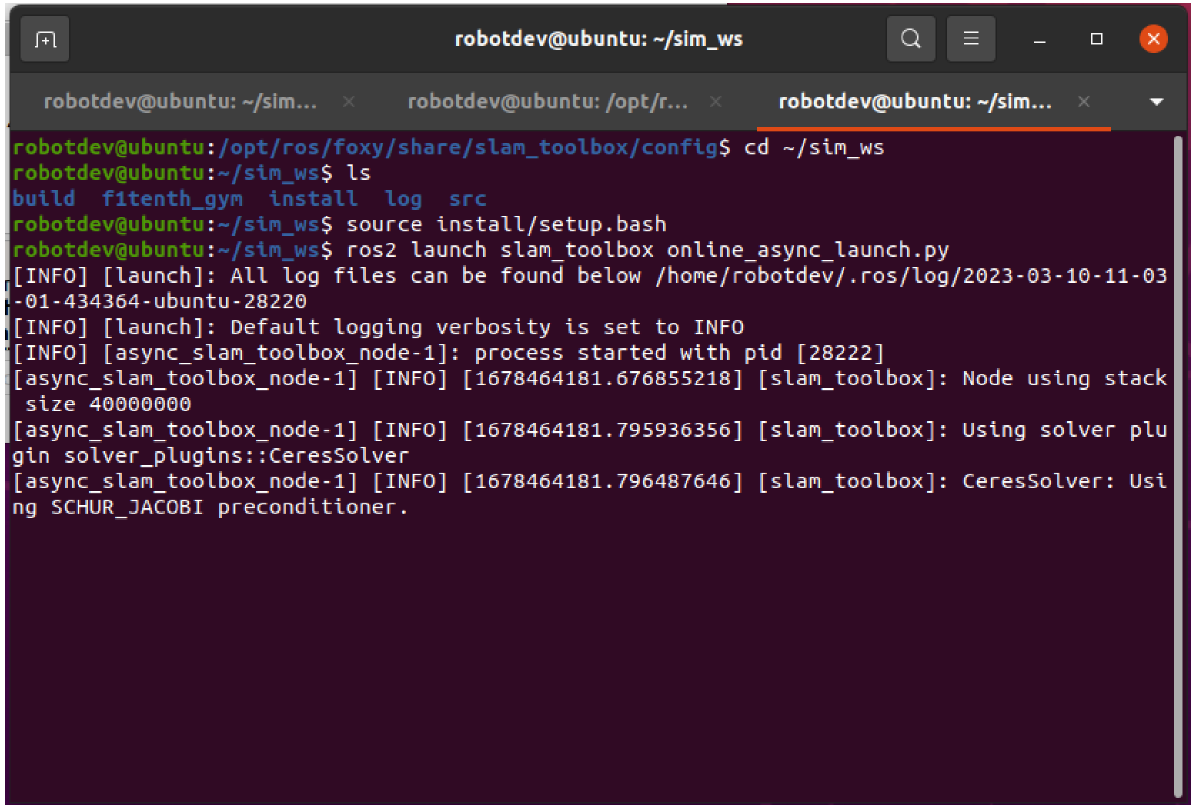


RUNING NAVIGATION

$ cd ~/sim_ws

$ source install/setup.bash

$ ros2 launch nav2_bringup navigation_launch.py



END OF WORKSHOP

Fred Livingston (fjliving@ncsu.edu)