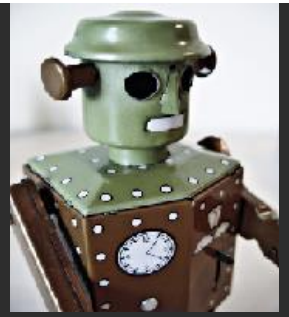
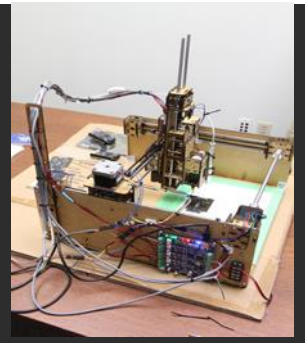
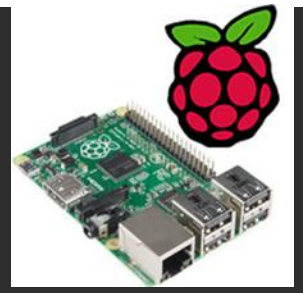



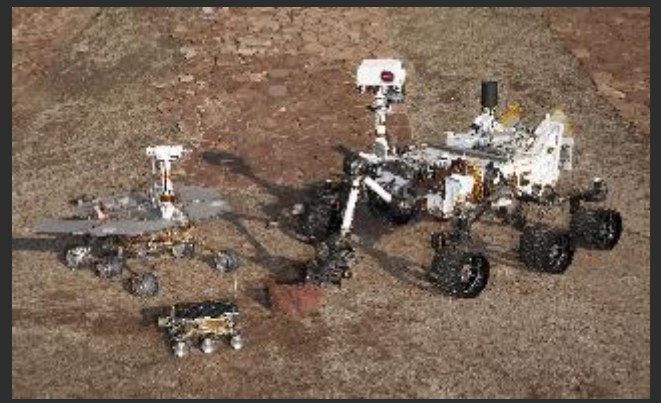



OVERVIEW OF ROS NAVIGATION WITH NEATO TURTLE

ROSS LUNAN

AUGUST 12, 2023





What is ROS & OSRF ?

Reference: https://www.ros.org/blog/why-ros/)

- **ROS** = **Robot Operating System**, an open source software development kit for robotics applications. **ROS** offers a standard software platform to developers across industries that will carry them from research and prototyping all the way through to deployment and production
- OSRF: **Open Source Robot Foundation**, founded in 2012 in Menlo Park, Ca,
- (https://www.osrfoundation.org/about/). In 2016, OSRF created the for-profit **Open Source Robotics Corporation**. In Dec/22 **Google Intrinsic** acquired **OSRC**. **Open Robotics** (not-for-profit) holds the **ROS**, **Gazebo**, and **Open-RMF IP** , and is their continuing support.

  https://discourse.ros.org/t/the-osrc-team-is-joining-intrinsic-and-what-it-means-for-the-ros-community/28764
- Mission: To support the development, distribution, and adoption of open software and hardware for use in robotics research, education, and product development.
- **What is ROS** ?: An open-source, meta-operating system for your robot.
- It provides the services you would expect from an **operating system**, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.
- Robots: 100's around the world….
- Installations, Tutorials, Blog, Repositories: www.ros.org : ROS primarily runs on Unix-based platforms. Others, are available but not to the same support extent. Currently Long-Term Support Versions are now the ROS 2 platform Software primarily tested on Ubuntu. e.g. ROS 2 Humble uses Ubuntu 22.04

Thar be Robots...(a small sample)

Reference: http://robots.ros.org/all

Agenda

1. How to Build a Neato Turtle (Reference “HomeBrewed Robots!” Camp Peavy)
   a. Parts
   b. Installing Ubuntu and ROS 2
   c. Installing & Configuring the Neato Turtle code
2. What is Navigation 2 – Nav2 ?
   a. Overview in 2 pages
   b. IROS 2020 Talk
   c. Github Repository
3. A Running Neato Turtle
   a. Nodes, Topics, rosgraph
   b. SLAM & Navigation

Another handy book is “Linux Pocket Guide”, Daniel Barrett, O'Reilly

Agenda

1. How to Build a Neato Turtle (Reference “HomeBrewed Robots!” Camp Peavy)
   a. Parts
   b. Installing Ubuntu and ROS 2
   c. Installing & Configuring the Neato Turtle code
2. What is Navigation 2 – Nav2 ?
   a. Overview in 2 pages
   b. IROS 2020 Talk & Nav2 SLAM Toolbox ROSCON 2019
   c. Github Repository
3. A Running Neato Turtle
   a. Nodes, Topics, rosgraph
   b. SLAM & Navigation

Neato Turtle Parts List

- Single Board Computer (SBC): Raspberry Zero 2 W, Pi3 or Pi4 2/4GB
- Power Raspberry Pi with 10,000 MaH or so Battery Bank
- USB-A to Micro USB 12“ cable, USB-A to mini USB (RPi3) or USB-c (RPi4)
- Micro SD Card 16 or 32 GB. Select fast 4K or better
- Neato Robot Vacuum: Most models use a similar API from the original 2010 BV 70 & 80 series, newer D-series models D70, D75, D80, D85, and current models D3, D4, D6, D7. Latest D6 & D7 do not.

- Remote Desktop: Ubuntu Machine, such as modest Windows PC (i5) HD reformatted to “Ubuntu ext”, 2nd Partition with ext/Ubuntu 22.04 Desktop, Windows WSL2 (messy)
- HDMI Monitor, USB Keyboard & Mouse to configure the Pi. Be aware that RPi4 uses USB-C power and micro HDMI connectors, RasPi Zero has 1 USB requiring an OTG cable & Hub
- Optional Teleop: Game Controller, Logitech F710 or Xbox PS4 work great
- Optional Camera: USB Webcam (Logitech C270)

Raspberry Pi Configuration

Configuring & Installing ROS 2 on a Linux Remote Desktop

- Select and configure your desired Remote Desktop Machine, such as:
  - a dedicated modest I3/I5 Laptop or Desktop Intel x86-64 PC HD reformatted for Ubuntu 22.04,
  - Separate partition (30-40 GB) Partition on your Windows Desktop,
  - WSL2 on Windows 11 which provides a traditional Ubuntu Command line and a GUI -a bit tricky to configure because of the Command Line interface & ROS Networking requirements

- From another Windows PC, download the Ubuntu 22.04 Desktop .iso Image
- Boot the machine and configure the Ubuntu Parameters
  - Update packages & add your desired Apps & Utilities (e.g. synapatic, top, tree, gedit, etc)
- Open browser & navigate to ROS 2 Install procedure, selecting Desktop

  https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html
  - Configure the "**Environment Setup**" and "source" from your /Home directory

    :~/ source /opt/ros/humble/setup.bash
  - Follow "**Using colcon to build packages**" https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html . Select your preferred workspace, e.g. botvac_ws. Note that later, after **compiling** any code, you must "**Source** " every Terminal instance from the colcon root, e.g.

    :~/dev_ws/ source install/setup.bash .

  Note that patience is required for compiling: Zero: 65 min/3: 10 mins/4 2 mins

Installing ROS 2 on Raspberry Pi (or equivalent)

- **ROBOT SBC**: Using the **Remote Desktop** machine, download 64-bit Ubuntu desired .iso image: either Ubuntu Desktop of the Raspberry Pi 4 or Server on the Raspberry Pi 3 & Zero 2 W. https://ubuntu.com/download/raspberry-pi
- **Format** Micro SD card with "RaspberryPi Imager" which enables configuring username, machine name, WiFi. https://www.raspberrypi.com/software/
- With Keyboard, Mouse, Power, Monitor connected: **Boo**t the RPi, and **configure** the Ubuntu User, Machine, WiFi Parameters,
  - Update packages (For all Raspberry Pi models, Update OS: ($: sudo apt update && sudo apt upgrade) &, install openssh-server, avahi-daemon, top. For Raspberry Pi4 Desktop: Add your desired Apps & Utilities (e.g. chromium-browser, synapatic, top, tree, gedit, etc) .
- Open browser & navigate to ROS 2 Install procedure, selecting Desktop for RPi 4. Base on RPi 3 or Zero W 2
  https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html
  - Configure the "**Environment Setup**" and "source" from your /Home directory
    :~/ source /opt/ros/humble/setup.bash
  - Follow "**Using colcon to build packages**" https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html . Configure your preferred workspace in home directory, e.g. ~/dev_ws/src. Note that patience is required for compiling: Zero: 65 min/3: RPI 3: 10 mins/4, RPi4: 2 mins
- Note that later, after compiling any code, you must "**Source**" every Terminal instance from the colcon root, e.g. :~/botvac_ws/ source install/setup.bash . OR edit the Home directory ~/.bashrc to include the **/opt/ros** and **colcon** source script

Neato Turtle Installation from Github Repository

After installing & configuring Ubuntu 22.04 (Jammy) and ROS 2 Humble , Neato Turtle code is installed by compiling with “colcon development” on Remote Desktop and SBC Raspberry Pi (https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html)

Prerequisites: From ROS 2 Debian and https://github.com/cpeavy2/ Repositories

sudo apt install build-essential python3-colcon-common-extensions

sudo apt install ros-humble-xacro python3-rosdep

Be sure and create a **workspace** and “source” its root directory for your ROS 2 from source builds, on both **Desktop** and **Raspberry Pi Robot**.

Check ($ sudo clone <repo name>) these following repositories into that **workspace** / source directory as follows:

Workspace on home directory, e.g. ~/botvac_ws

From home directory $: ~/ $ mkdir –p botvac_ws/src . $: cd botvac_ws/src

git clone https://github.com/cpeavy2/botvac_node.git

git clone https://github.com/cpeavy2/neato_robot.git

git clone https://github.com/kobuki-base/cmd_vel_mux.git

git clone https://github.com/kobuki-base/kobuki_velocity_smoother

git clone https://github.com/stonier/ecl_tools

Install Navigation 2 ONLY on Ubuntu Remote Desktop workstation (not necessary to have on Pi).

sudo apt install ros-humble-navigation2

sudo apt install ros-humble-nav2-bringup

By Camp Peavy

Over-the-shoulder instructions on how to build your own homebrewed robot!

Robot Workspace Configuration

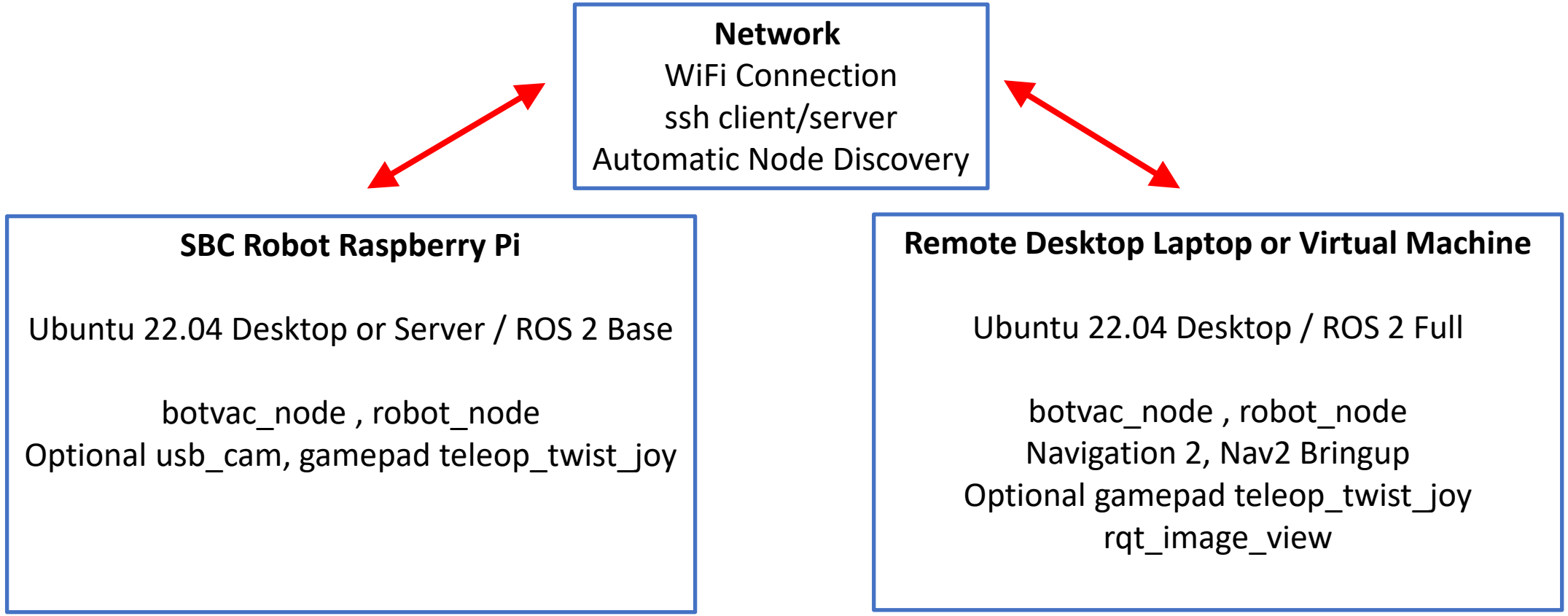
```
.
├── CMakeLists.txt
├── README.md
├── config
│   ├── cmd_vel_mux_params.yaml
│   └── velocity_smoother_params.yaml
├── launch
│   ├── botvac_base_map.launch.py
│   ├── botvac_base_nav.launch.py
│   ├── botvac_base_only.launch.py
│   ├── botvac_gui_only.launch.py
│   ├── botvac_map_gui.launch.py
│   ├── botvac_nav_gui.launch.py
│   └── include
│       ├── botvac_base.launch.py
│       ├── bv80bot_base_cam.launch.py
│       ├── bv80bot_gui.launch.py
│       ├── bv80bot_map.launch.py
│       ├── bv80bot_nav.launch.py
│       ├── keyboard_teleop.launch.py
│       ├── logitech_teleop.launch.py
│       ├── ps3_teleop.launch.py
│       ├── velocity_smoother.launch.py
│       └── xbox360_teleop.launch.py
├── meshes
│   ├── neato.dae
│   └── neato.skp
├── package.xml
└── urdf
    └── neato.urdf.xacro

6 directories, 24 files
```

```
.
├── README.md
├── neato_2dnav
│   ├── CMakeLists.txt
│   ├── COLCON_IGNORE
│   ├── launch
│   │   ├── amcl.launch
│   │   └── move_base.launch
│   ├── maps
│   │   ├── neato-ils.pgm
│   │   └── neato-ils.yaml
│   ├── package.xml
│   └── params
│       ├── base_local_planner_params.yaml
│       ├── costmap_common_params.yaml
│       ├── global_costmap_params.yaml
│       └── local_costmap_params.yaml
├── neato_driver
│   ├── neato_driver
│   │   ├── __init__.py
│   │   └── neato_driver.py
│   ├── package.xml
│   ├── resource
│   │   └── neato_driver
│   ├── setup.cfg
│   └── setup.py
├── neato_msgs
│   ├── CMakeLists.txt
│   ├── msg
│   │   ├── Button.msg
│   │   └── Sensor.msg
│   └── package.xml
├── neato_node
│   ├── launch
│   │   └── neato_node_launch.py
│   ├── neato_node
│   │   ├── __init__.py
│   │   └── neato_node.py
│   ├── package.xml
│   ├── resource
│   │   └── neato_node
│   ├── setup.cfg
│   └── setup.py
└── neato_robot
    ├── CMakeLists.txt
    └── package.xml

15 directories, 31 files
```

2 Machine Configuration: Remote Desktop & SBC Robot



Neato Turtle Extras

1. Game Controller Logitech F710 connected to Desktop USB.
   a. Uses ros2 binary installed package: ros-humble-teleop-twist-joy
   b. Run with customized config yaml file: : $ ros2 launch teleop_twist_joy teleop_launch.py config_filepath:='/home/ubuntu/Desktop/config/f710.yaml'

2. WebCam: Uses usb_cam package installed on SBC RasPi
   a. Install from binary package : $ sudo apt install ros-humble-usb-cam
   b. Run with WebCam plugged into Raspberry Pi, by ssh from Desktop:

      : $ ros2 run usb_cam usb_cam_exe.py with full resolution, or

      : $ ros2 run usb_cam usb_cam_exe.py –ros-args –params-file ~/Desktop/usb_cam/config/params10.yaml, where a revised "params10.yaml" config file with image_height=240, image_width=320 framerate=10.

Agenda

1. How to Build a Neato Turtle
   a. Parts
   b. Installing Ubuntu and ROS 2
   c. Installing & Configuring the Neato Turtle code
2. What is Navigation 2 – Nav2 ?
   a. Overview in 2 pages
   b. IROS 2020 Talk & Nav2 SLAM Toolbox ROSCON 2019
   c. Github Repository
3. A Running Neato Turtle
   a. Nodes, Topics, rosgraph
   b. SLAM & Navigation

Navigation 2 Docs, Github, Tutorials

- NAV 2 https://navigation.ros.org
- Github https://github.com/ros-planning/navigation2

  The required packages can be installed from Binary on your Laptop Desktop. e.g.

  :$ sudo apt install ros-humble-navigation2

  :$ sudo apt install ros-humble-nav2-bringup
- Getting Started https://navigation.ros.org/getting_started/index.html

  (Don't necessarily do the Simulation – unless you are curious)
- SLAM Toolbox

  A set of tools and capabilities for 2D SLAM built by Steve Macenski .

  This project contains the ability to do most everything any other available SLAM library, both free and paid, and more.

Nav2 Overview

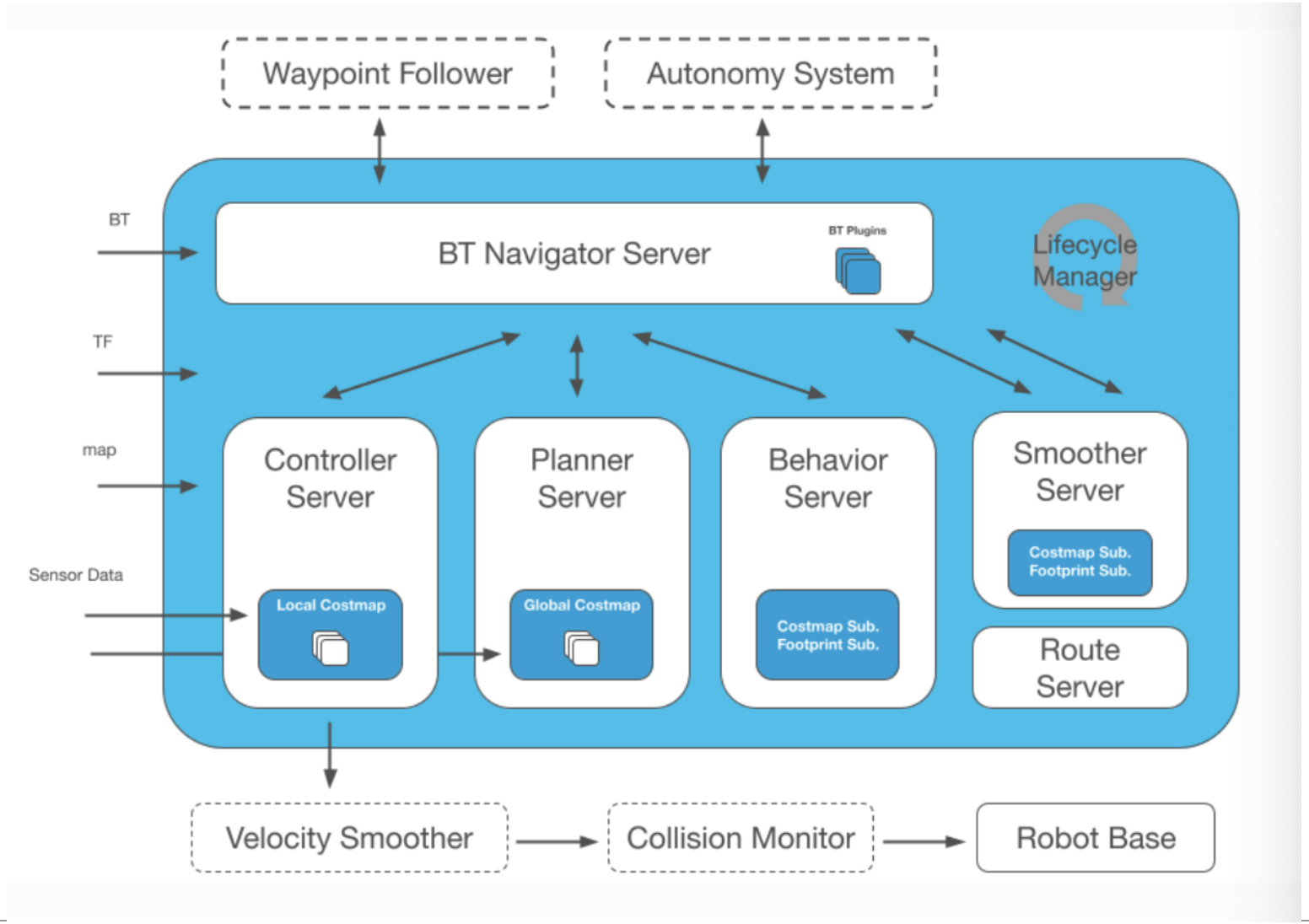
https://navigation.ros.org

- Project seeks to find a safe way to have a mobile robot move to complete complex tasks through many types of environments and classes of robot kinematics.
- Not only can it move from Point A to Point B, but it can have intermediary poses, and represent other types of tasks like object following and more.
- Nav2 is a production-grade and high-quality navigation framework trusted by 50+ companies worldwide.
- It provides perception, planning, control, localization, visualization, and much more to build highly reliable autonomous systems.
- This will complete environmental modeling from sensor data, dynamic path planning, compute velocities for motors, avoid obstacles, represent semantic regions and objects, and structure higher-level robot behaviors.

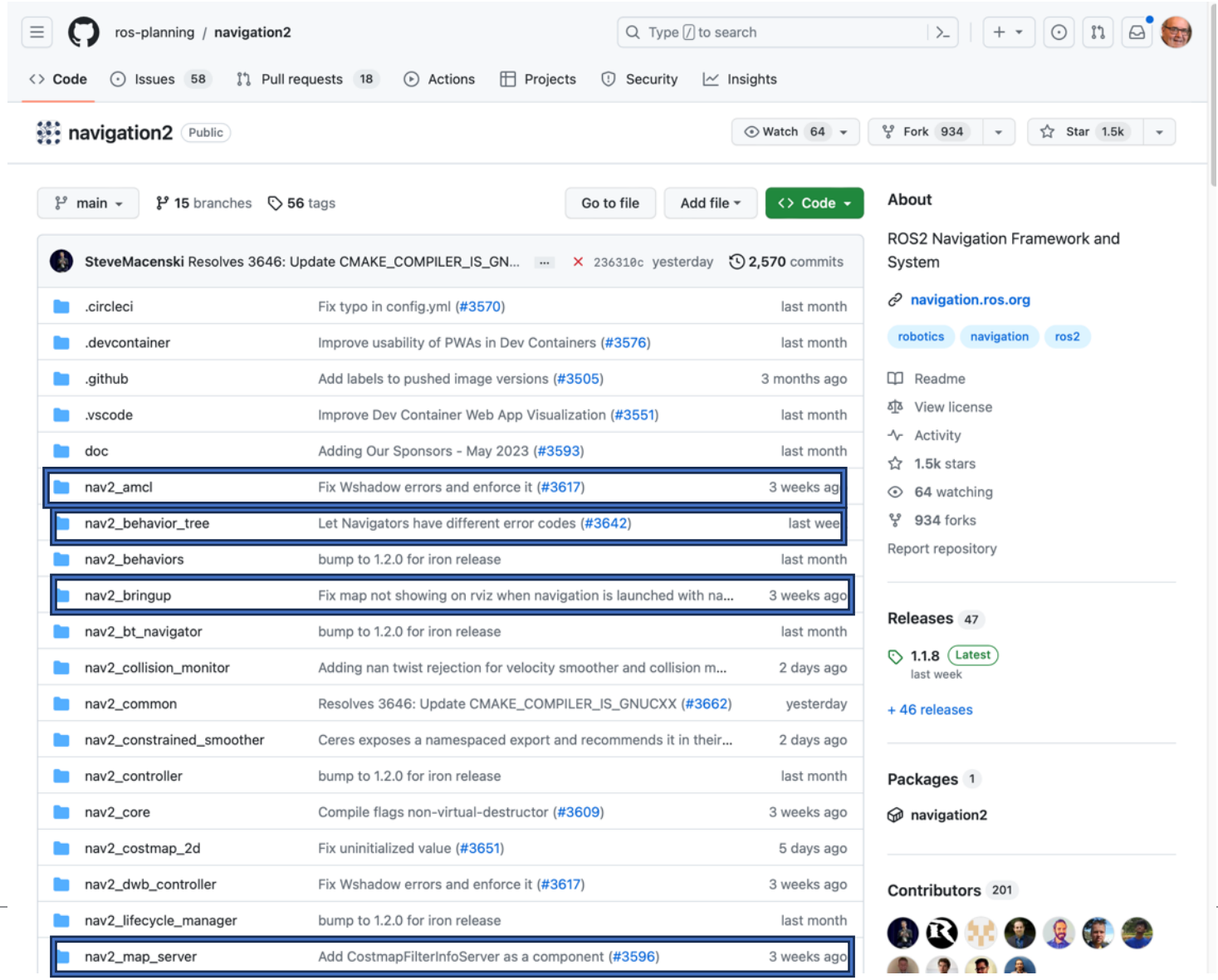
It has tools to:

- 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)

Nav2 Lifecycle Manager



Navigation 2 github.com

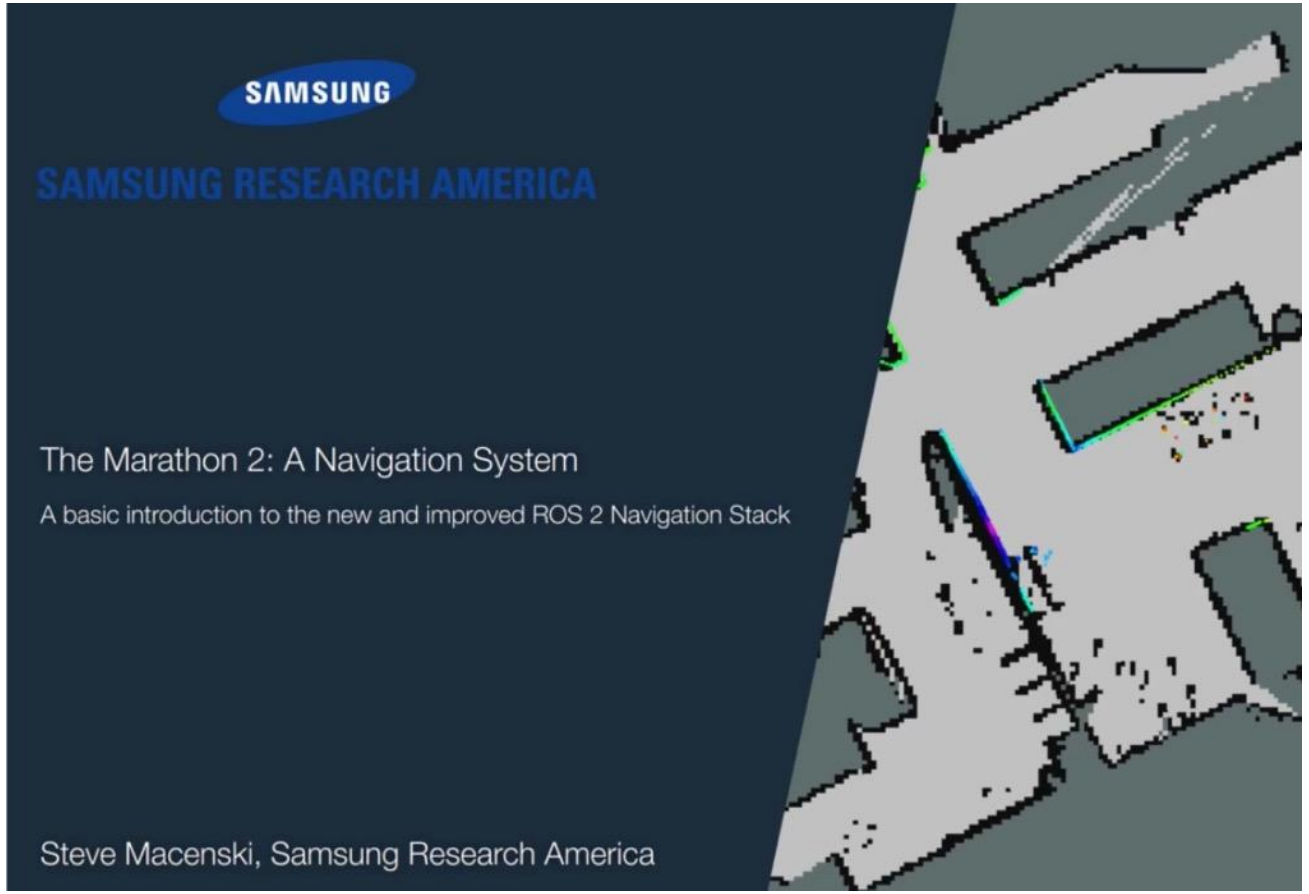


N A V 2

[ros-planning/navigation2]

IROS 2020 Talk on Nav 2
https://youtu.be/QB7lOKp3ZDQ

- S. Macenski, F. Martín, R. White, J. Clavero. The Marathon 2: A Navigation System. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.



Nav2 SLAM Toolbox ROSCon 2019

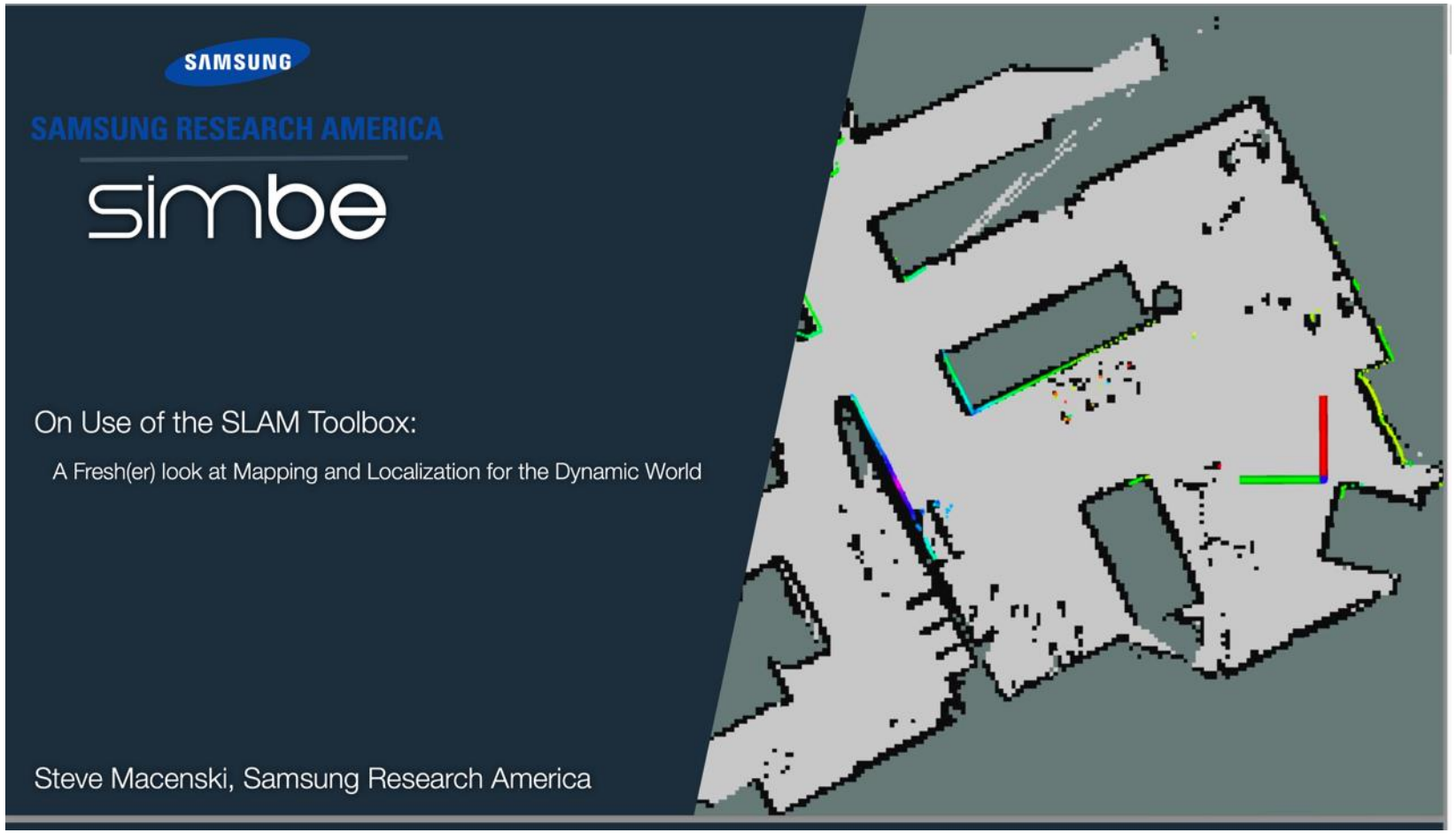
https://roscon.ros.org/2019/talks/roscon2019_slamtoolbox.pdf

We introduce the SLAM Toolbox. It implements synchronous and asynchronous SLAM for massive indoor and changing environments as well as life-long mapping and localization modes.

SLAM Toolbox brings several improvements over the existing solutions.

This includes plugin optimizers with default Ceres, speed-ups in Karto's scan matcher, pose-graph manipulation tools, serialization, continued mapping on serialized SLAM graphs, pose-graph localization rolling window technique as a replacement for AMCL, and enables fully distributed mapping without the use of derived 2D occupancy image maps.

This package was built for mapping of massive retail and warehouse applications, though likely effective out of those scopes. This talk will go over key points of SLAM Toolbox, demonstrations in production environments, and how to enable it in your application.



Behavior Navigator (bt_navigator)

- A behavior tree (BT) is a mathematical model of plan execution used in computer science, robotics, control systems and video games. They describe switchings between a finite set of tasks in a modular fashion

- The BT Navigator (Behavior Tree Navigator) module implements the NavigateToPose and NavigateThroughPoses task interfaces.

- It is a Behavior Tree-based implementation of navigation that is intended to allow for flexibility in the navigation task and provide a way to easily specify complex robot behaviors.

Agenda

1. How to Build a Neato Turtle
   a. Parts
   b. Installing Ubuntu and ROS 2
   c. Installing & Configuring the Neato Turtle code
2. What is Navigation 2 – Nav2 ?
   a. Overview in 2 pages
   b. IROS 2020 Talk & Nav2 SLAM Toolbox ROSCON 2019
   c. Github Repository
3. A Running Neato Turtle
   a. Nodes, Topics, rosgraph
   b. SLAM & Navigation

Neato Turtle Base & usb_cam (only) Launch

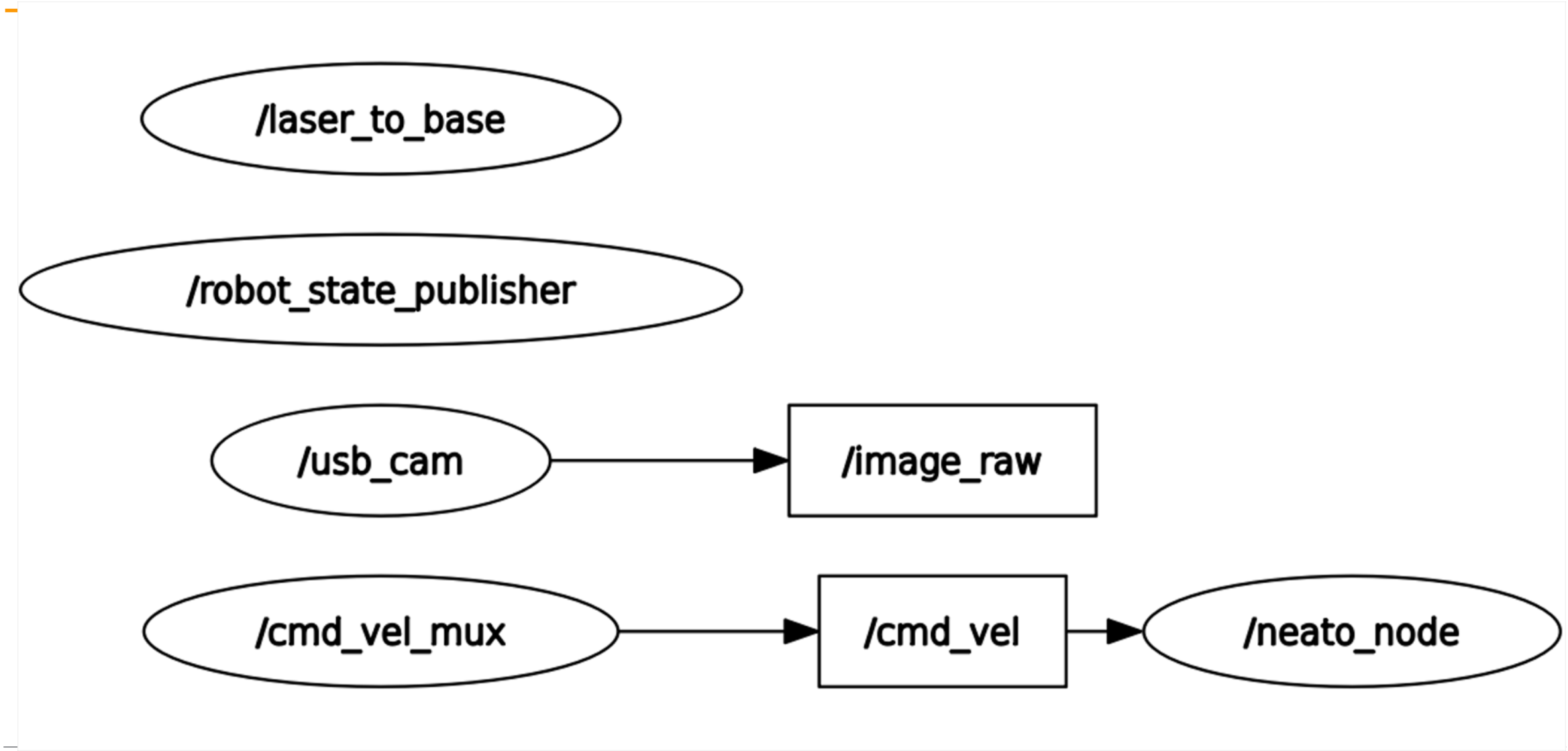
Nodes

ubuntu@macvm-ub22h-r2:~$ ros2 node list
/cmd_vel_mux
/laser_to_base
/neato_node
/robot_state_publisher
/rqt_gui_cpp_node_111891
/usb_cam

Topics

ubuntu@macvm-ub22h-r2:~$ ros2 topic list
/active
/button
/camera_info
/cmd_vel
/image_raw
/image_raw/compressed
/image_raw/compressedDepth
/image_raw/theora
/input/nav
/input/teleop
/joint_states
/odom
/parameter_events
/robot_description
/rosout
/scan
/sensor
/tf
/tf_static

Base & Camera (only) Launch “rqt_graph”



ROS 2 Nodes from Neato Turtle Launch w/Nav2

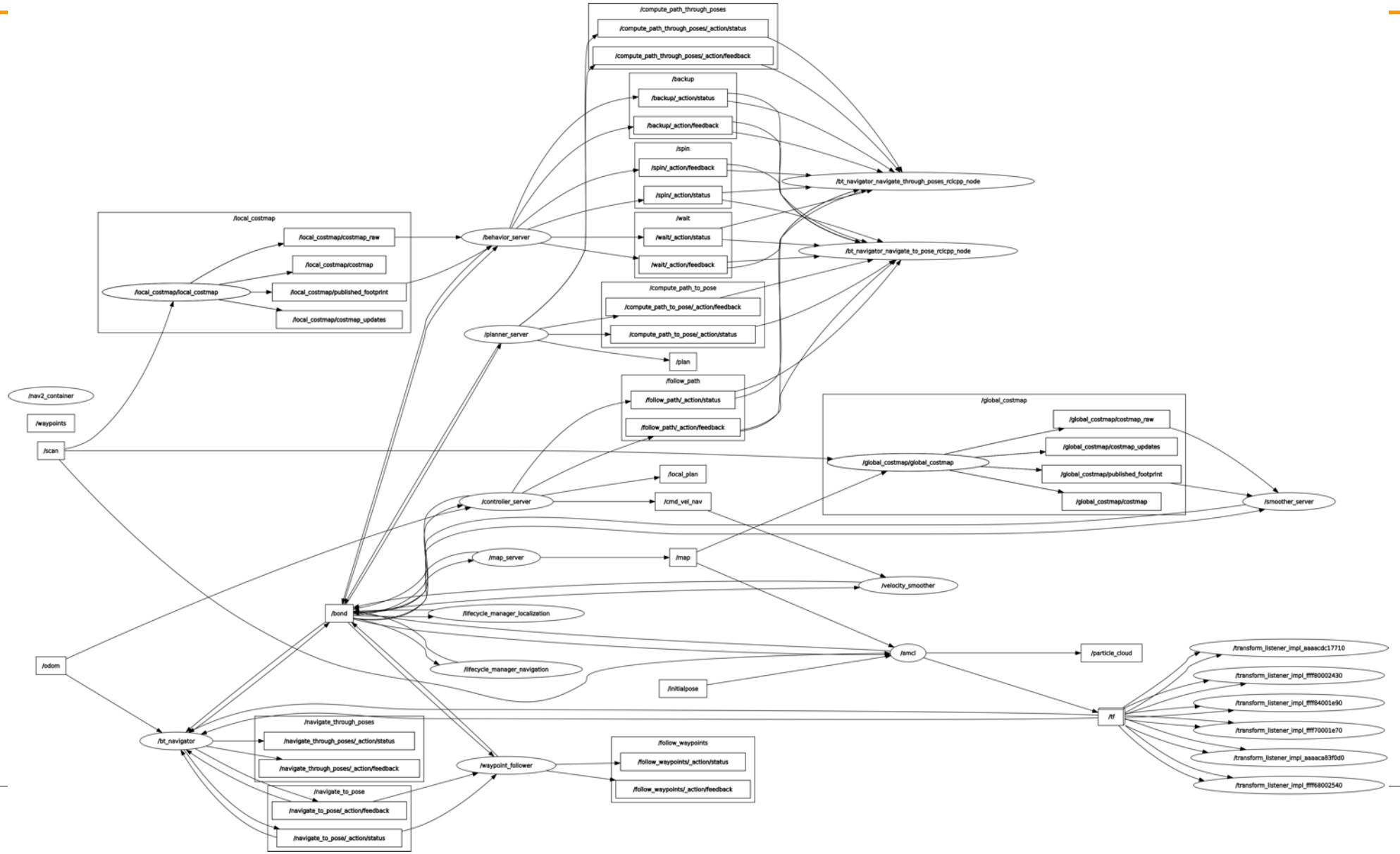
```
ubuntu@rpiz2w-ub22h-bv:~$ ros2 node list
/amcl
/behavior_server
/bt_navigator
/bt_navigator_navigate_through_poses_rclcpp_
node
/bt_navigator_navigate_to_pose_rclcpp_node
/cmd_vel_mux
/controller_server
/global_costmap/global_costmap
/joy_node
/laser_to_base
/launch_ros_4492
/lifecycle_manager_localization
/lifecycle_manager_navigation
/local_costmap/local_costmap
/map_server
/nav2_container
/neato_node
/planner_server
robot_state_publisher
/rqt_gui_cpp_node_4362
/rviz
/smoother_server
/teleop_twist_joy_node
/transform_listener_impl_aaaaffaa0b40
/transform_listener_impl_aaab0f690d90
/transform_listener_impl_aaab0fd05ea0
/transform_listener_impl_ffff4c00c2e0
/transform_listener_impl_ffff54006170
/transform_listener_impl_ffff600024b0
/v4l2_camera
/velocity_smoother
/waypoint_follower
```

ROS 2 Topics w/Neato and Nav2 launched

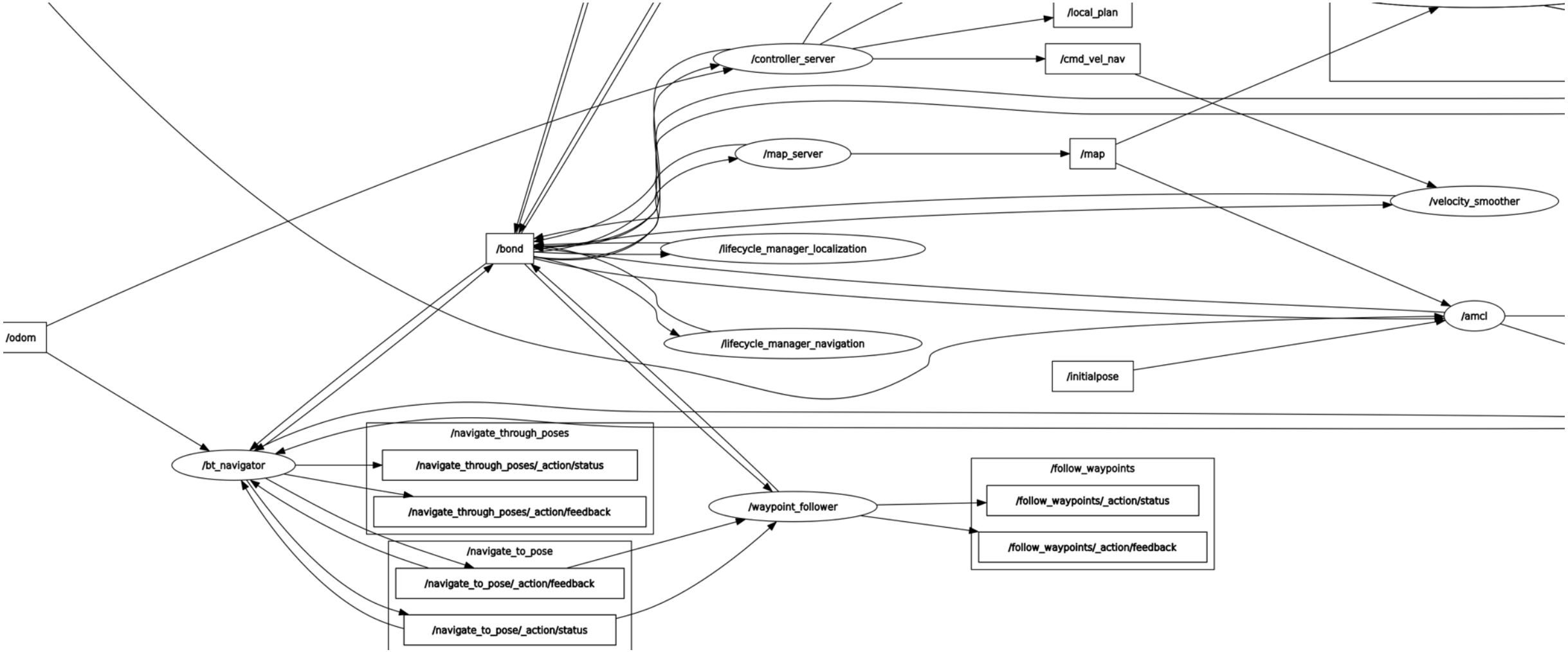
ubuntu@rpiz2w-ub22h-bv:~$ ros2 topic list

/active
/amcl/transition_event
/amcl_pose
/behavior_server/transition_event
/behavior_tree_log
/bond
/bt_navigator/transition_event
/button
/camera_info
/clicked_point
/cmd_vel
/cmd_vel_nav
/cmd_vel_teleop
/controller_server/transition_event
/cost_cloud
/diagnostics
/downsampled_costmap
/downsampled_costmap_updates
/evaluation
/global_costmap/costmap
global_costmap/costmap_raw
/global_costmap/costmap_updates
/global_costmap/footprint
/local_costmap/voxel_marked_cloud
/local_plan
/map
/map_server/transition_event
/map_updates
/marker
/mobile_base/sensors/bumper_pointclou
d
/odom
/parameter_events
/particle_cloud
/plan
/plan_smoothed
/planner_server/transition_event
/preempt_teleop
/received_global_plan
/robot_description
/rosout
/scan
/sensor
/smoother_server/transition_event
/speed_limit
/map
/map_server/transition_event
/map_updates
/marker
/mobile_base/sensors/bumper_pointclou
d
/odom
/parameter_events
/particle_cloud
/plan
/plan_smoothed
/planner_server/transition_event
/preempt_teleop
/received_global_plan
/robot_description
/rosout
/scan
/sensor
/smoother_server/transition_event
/speed_limit
/tf
/tf_static
/transformed_global_plan
/velocity_smoother/transition_event
/waypoint_follower/transition_event
/waypoints

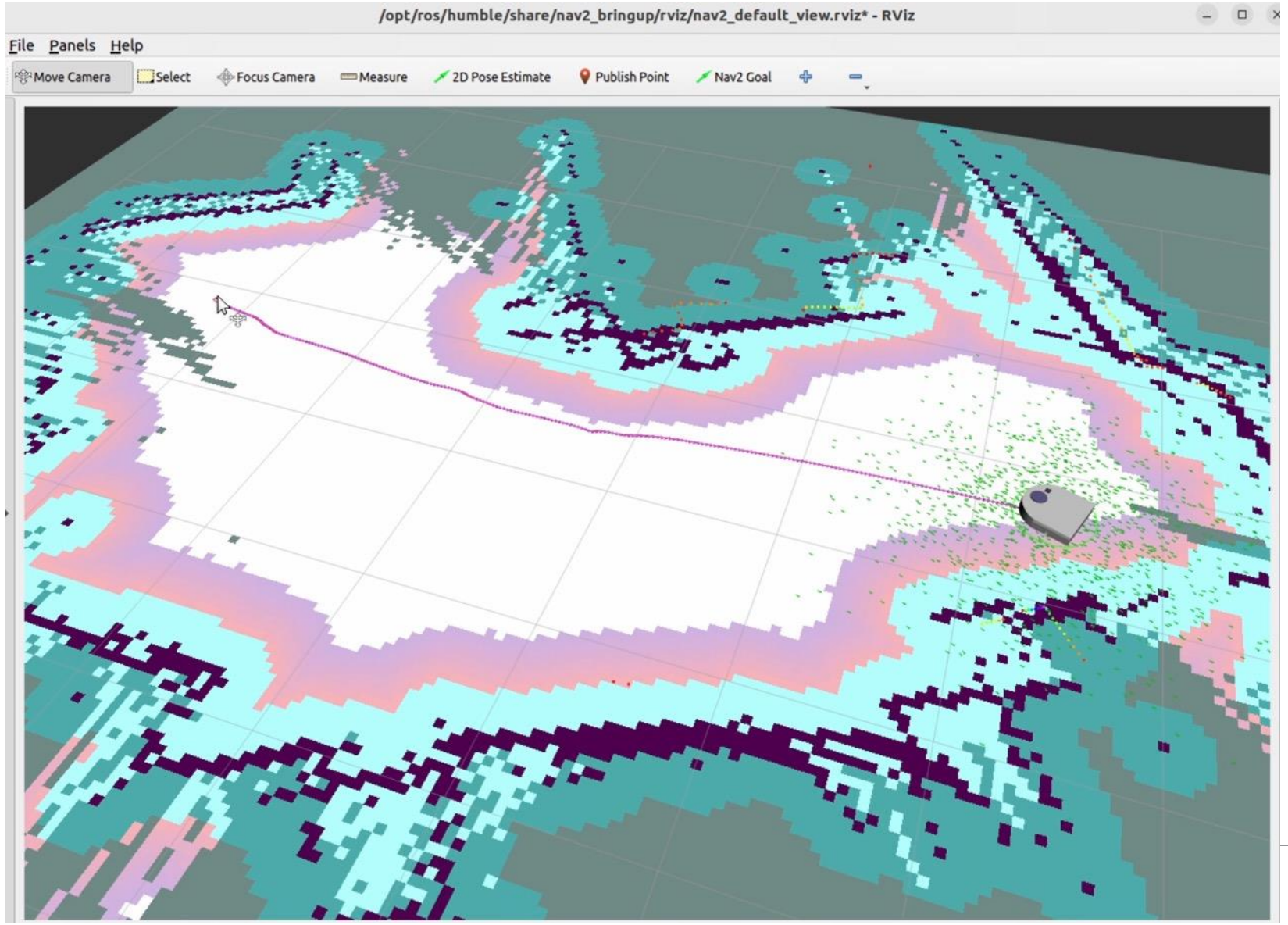
Neato Turtle Nav2 rqt_graph



Neato Turtle ros_graph (/cmd_vel)



SLAM & Navigation on map with SLAM Toolbox



ROS 2 Rviz Display

Global Map: Outside Perimeter of the Scan data

Costmap: Grid maps where each cell is assigned a specific value of cost. It represents the cost (difficulty) of traversing different areas of the map. The different colors represent known spaces, unknown space, obstacles , and inflation layers.

AMCL Package (Adaptive Monte Carlo Localization) which locates the Robot position and orientation on the static map. Path Plan is defined from Pose Estimate to Nav2 Goal

ROS References

- “Home Brewed Robots!”, by Camp Peavy . Chapter 6 describes configuring and running a Neato Turtle Robot System. (from Amazon)
  https://github.com/cpeavy2/botvac_node
- ROS/Introduction
  http://wiki.ros.org/ROS/Introduction
- ROS Users - for general ROS-related discussions
  https://discourse.ros.org
- ROS Developers - for ROS core development
  https://answers.ros.org
- Neato Turtle (Botvac) Developers - for ROS core development
  http://wiki.ros.org/ROS/Introduction
- Navigation 2
  https://navigation.ros.org
  https://github.com/ros-planning/navigation2/tree/humble/nav2_bringup/launch
- SLAM Toolbox
- https://github.com/SteveMacenski/slam_toolbox#

Questions ?