ROS 2 Mapping, Teleop & Hardware Hardware

by Nathan Kingori

Understanding Key Concepts in ROS 2 Robotics

What is ROS 2?

ROS 2 (Robot Operating System 2) is an an open-source set of software libraries and libraries and tools that help you build robot robot applications. It provides a flexible flexible framework for developing robotics robotics software, from drivers to state-of-state-of-the-art algorithms, and makes it makes it easy to integrate different components and systems.

Why Mapping & Teleoperation are Essential

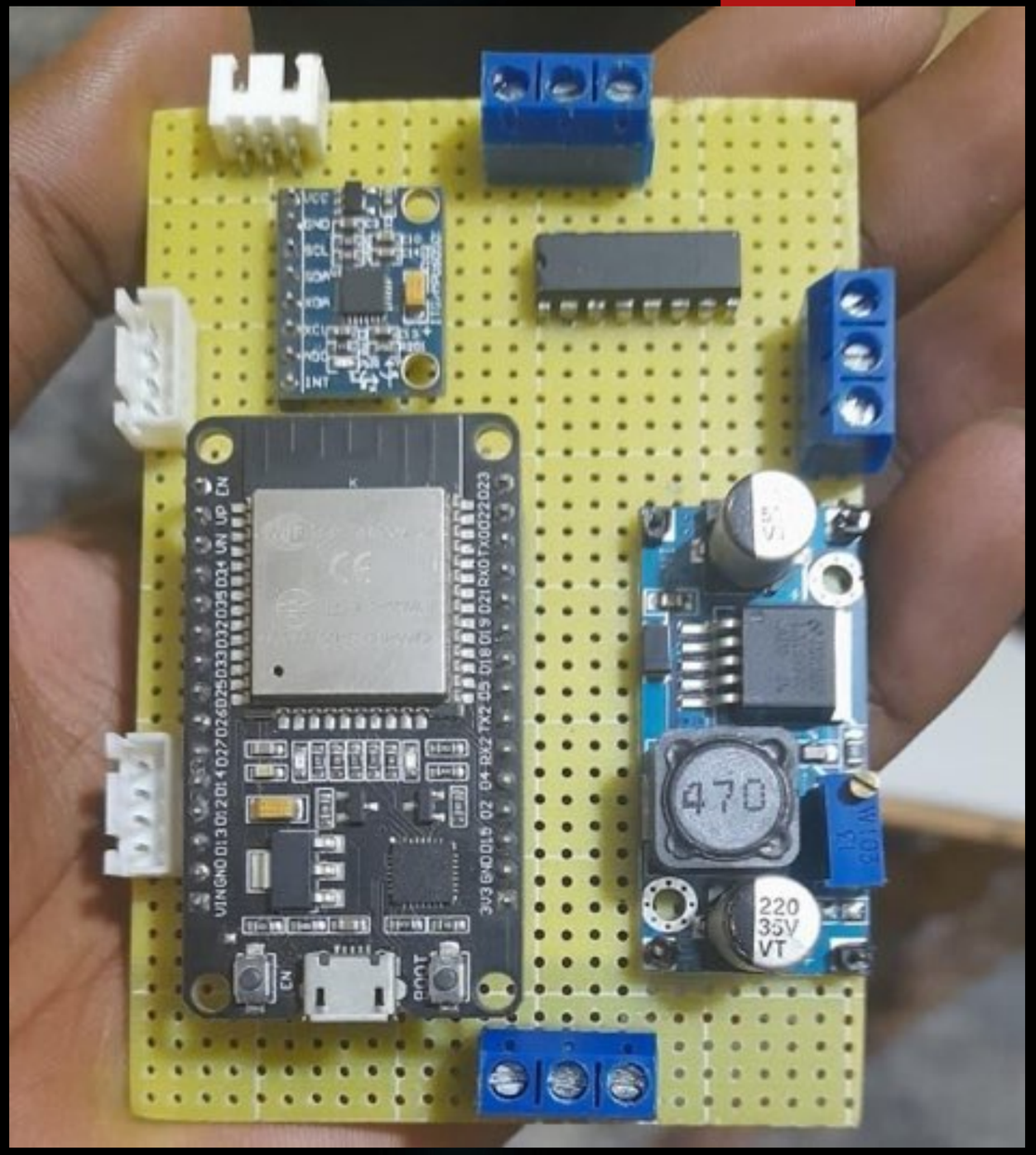
Mapping allows robots to build a representation of their environment, crucial crucial for navigation, localization, and path path planning. **Teleoperation** provides remote control capabilities, enabling human human operators to guide robots, especially especially in complex or hazardous situations, ensuring flexibility and safety. safety.

Importance of Hardware-Software Software Integration

Seamless integration between a robot's robot's physical components (hardware) (hardware) and its controlling programs programs (software) is fundamental. This This synergy ensures the robot can accurately perceive its surroundings, process information, and execute actions actions reliably, leading to optimal performance, safety, and efficiency in real-real-world applications.

Hardware: The Backbone of a Functional Robot

- **Sensors** – Detect the environment (e.g., LiDAR, ultrasonic sensors, cameras).
- **Actuators / Motors** – Enable movement of wheels, arms, or joints.
- **Controllers / Microcontrollers** – The “brain” that executes commands (e.g., ESP32, Raspberry Pi, Arduino).
- **Power Supply** – Batteries or adapters that provide energy.
- **Chassis / Frame** – The structural body that holds everything together.
- **Communication Modules** – Allow data transfer between components or with external devices (e.g., Wi-Fi, Bluetooth, serial).
- **Peripheral Devices** – LEDs, speakers, robotic arms, grippers, etc.



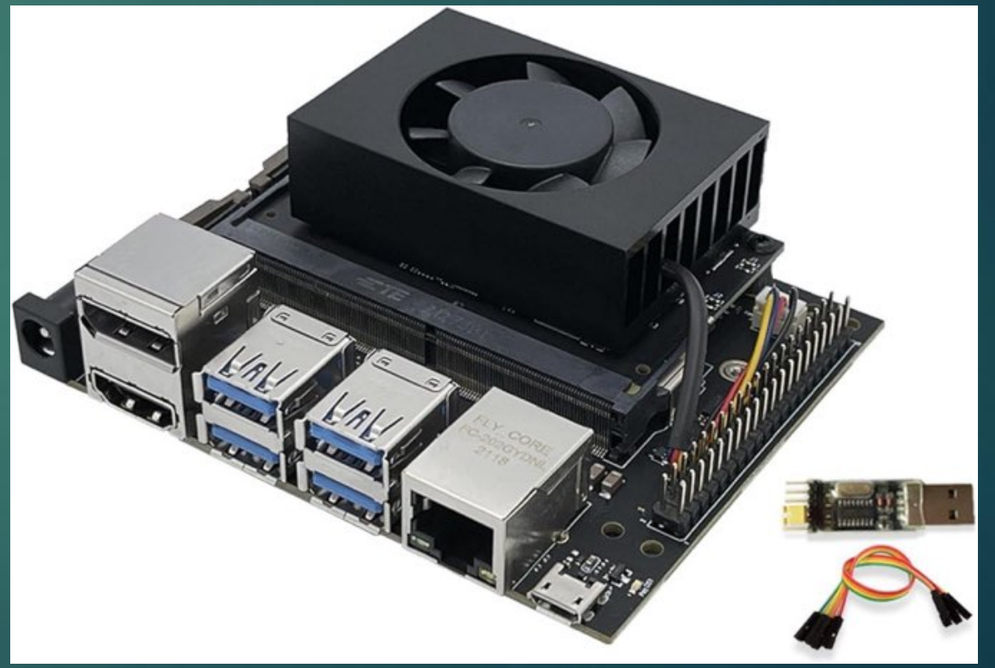
Hardware Considerations

Motors

The driving force for movement. Options Options range from simple **DC motors** for **motors** for basic tasks, **stepper motors** for **motors** for precise positioning, to high-high-efficiency **BLDC motors** for demanding applications.

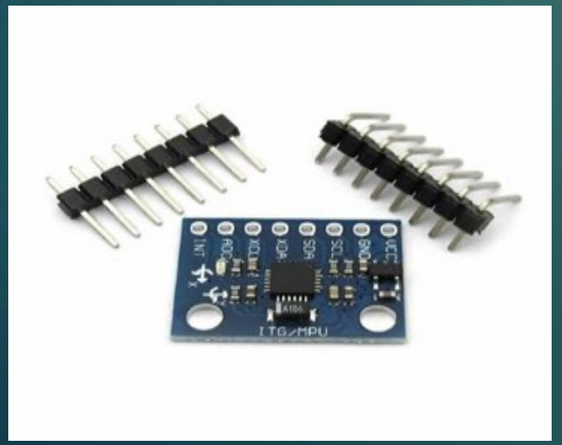
Compute

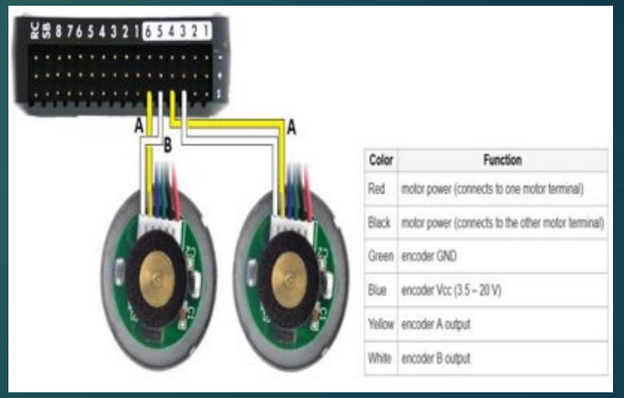
The "brain" running ROS 2. Options include compact boards like **Raspberry Pi Raspberry Pi** and **Jetson Nano** for embedded systems, or a **laptop** for more more powerful processing and development flexibility.



Sensors

Perceive the environment. **LiDAR** creates detailed maps, maps, **IMUs** track orientation, **wheel encoders** measure measure distance, and **cameras** provide visual information information for navigation and object detection.

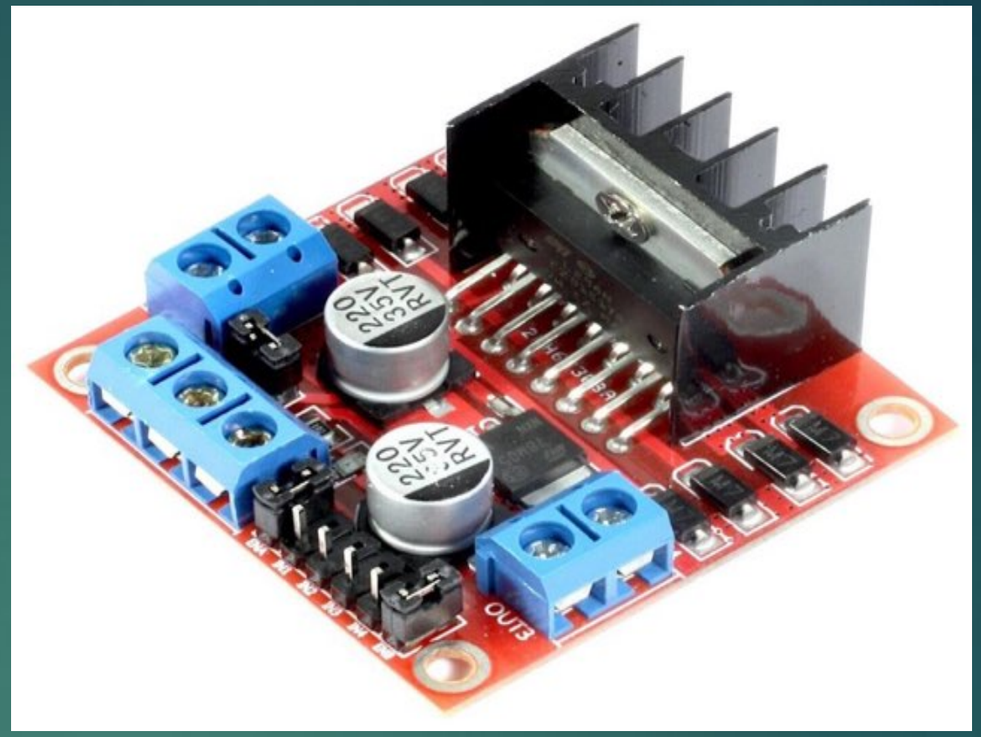


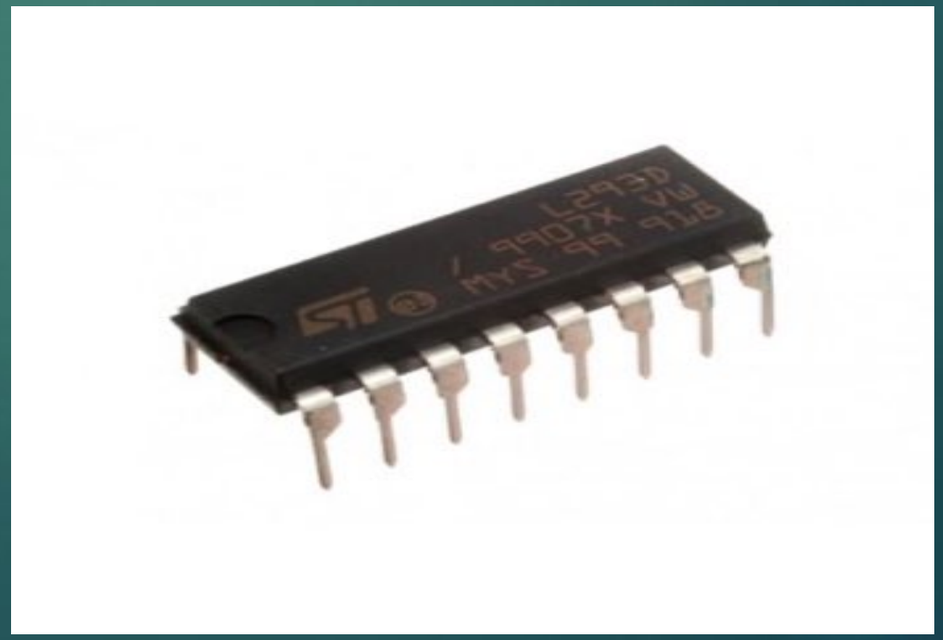


Color	Function
Red	motor power (connects to one motor terminal)
Black	motor power (connects to the other motor terminal)
Green	encoder GND
Blue	encoder Vcc (3.5 – 20 V)
Yellow	encoder A output
White	encoder B output

Motor Drivers

These electronic circuits control the speed and direction of motors. Common Common choices include **L298N** for basic control, **TB6612FNG** for low-power power applications, and **ODrive**/**Roboclaw** for advanced, high-power needs. needs.





Core Components of ROS 2

Data Distribution Service (DDS) (DDS)

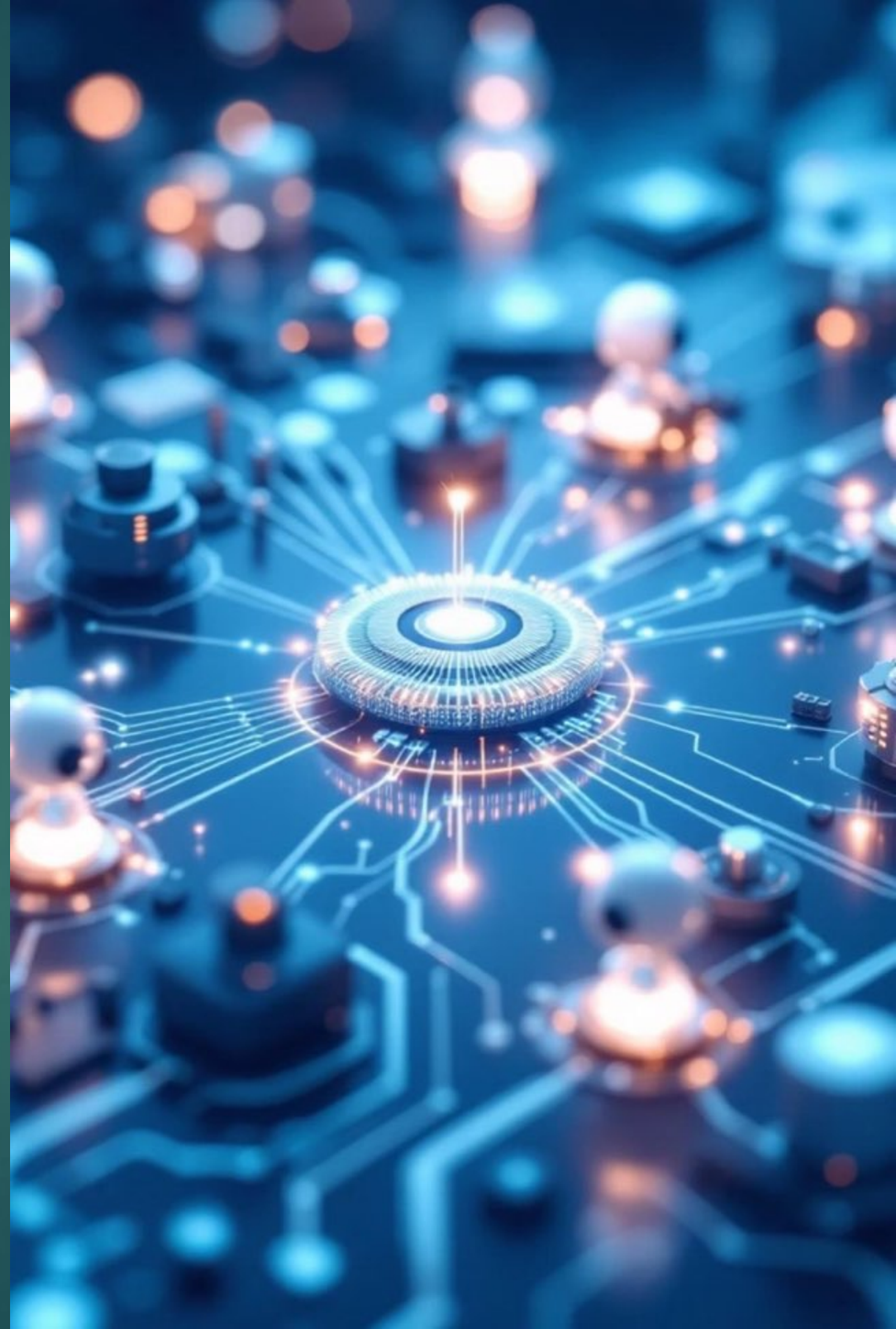
ROS 2 leverages DDS as its primary communication middleware. This allows allows for robust, real-time data exchange across distributed systems, systems, ensuring high performance performance and reliability for robot robot applications.

Modular Nodes & Topics

The architecture is built around modular "nodes" (individual executable executable processes) that communicate via "topics" (named buses buses for messages). This design promotes reusability, fault tolerance, tolerance, and scalable development. development.

Multi-Platform Compatibility

ROS 2 is designed for broad compatibility, running natively on Linux and via Docker or Docker or WSL2 for Windows environments. This flexibility supports diverse development and deployment scenarios across various operating systems.



Teleoperation: Direct Robot Control Control

1

Keyboard Teleop

Utilizes keyboard inputs (e.g., `teleop_twist_keyboard`)

2

Joystick/Gamepad Teleop

Provides intuitive, continuous control with joysticks or gamepads. Offers a more natural interface for complex maneuvers and dynamic environments.

Setting Up Your Teleoperation Node

Install Package

Begin by installing the `teleop_twist_keyboard` ROS 2 package, which provides the essential executable for keyboard-based robot control.

Launch the Node

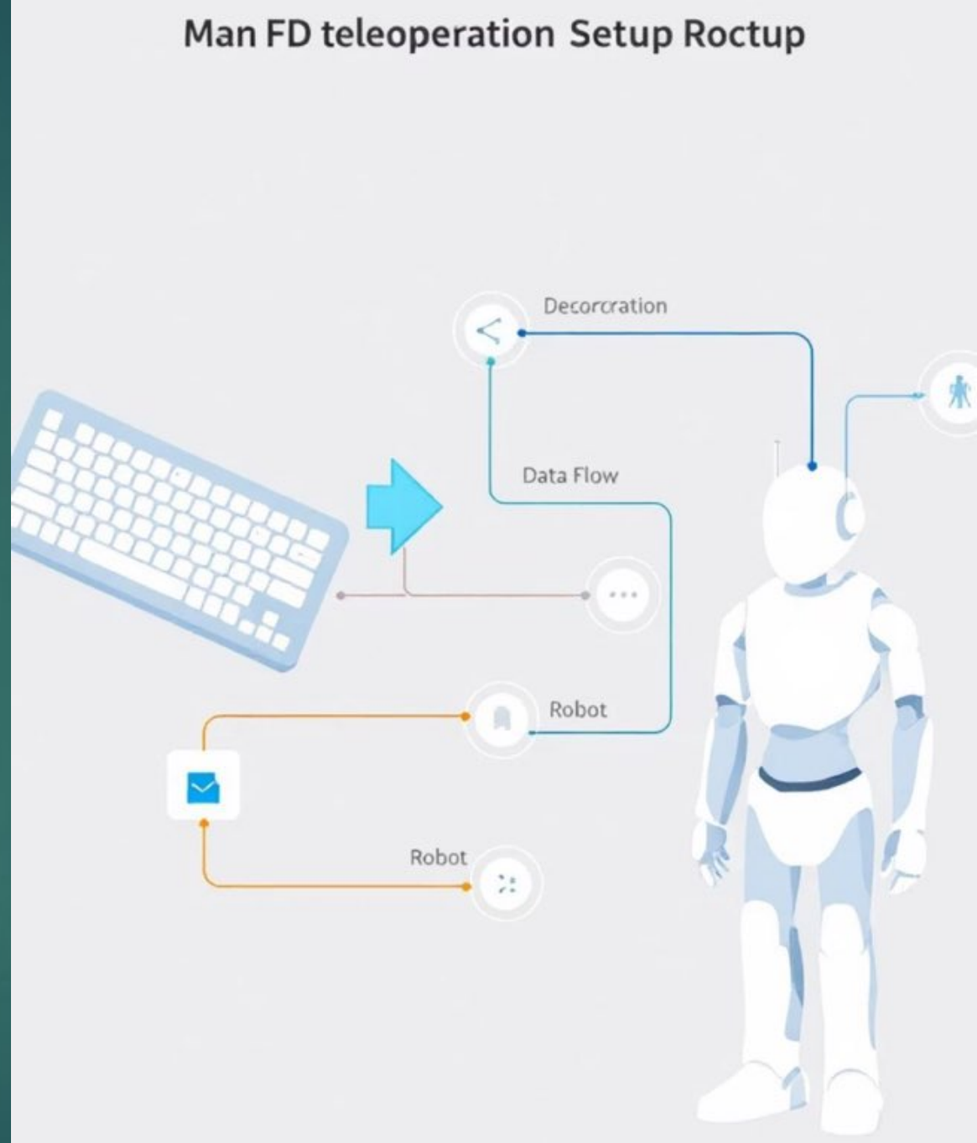
Execute `ros2 run teleop_twist_keyboard teleop_twist_keyboard` in your terminal to start the teleoperation node and enable input.

Robot Configuration

Ensure the teleoperation node is configured correctly for your robot's specific locomotion system, whether it's differential drive, skid steer, or another type.

Understand Data Flow

The command sequence typically follows: `Keyboard → ROS 2 Node → /cmd_vel topic → Motor Driver`, translating your inputs into robot movements.

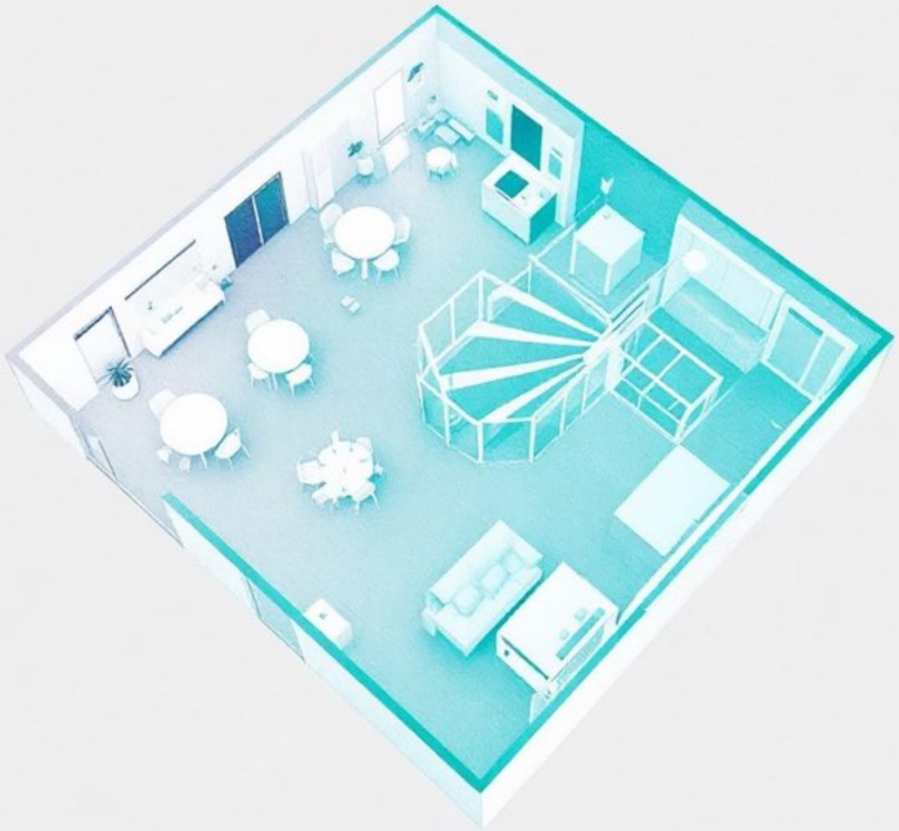


source /opt/ros/humble/setup.bash

sudo apt update
sudo apt install ros-humble-teleop-twist-keyboard

ros2 run teleop_twist_keyboard teleop_twist_keyboard

ros2 launch my_launch_file.py



Mapping: Building a Robot's World View

What is Mapping?

Mapping in robotics is the process of creating a representation of the robot's environment, often in the form of a 2D or 3D map. This map is crucial for the robot to understand its surroundings and navigate effectively.

Understanding SLAM

SLAM stands for **S**imultaneous **L**ocalization **a**nd **M**apping. It's a fundamental challenge where a robot builds a map of an unknown environment while simultaneously localizing itself within that map.

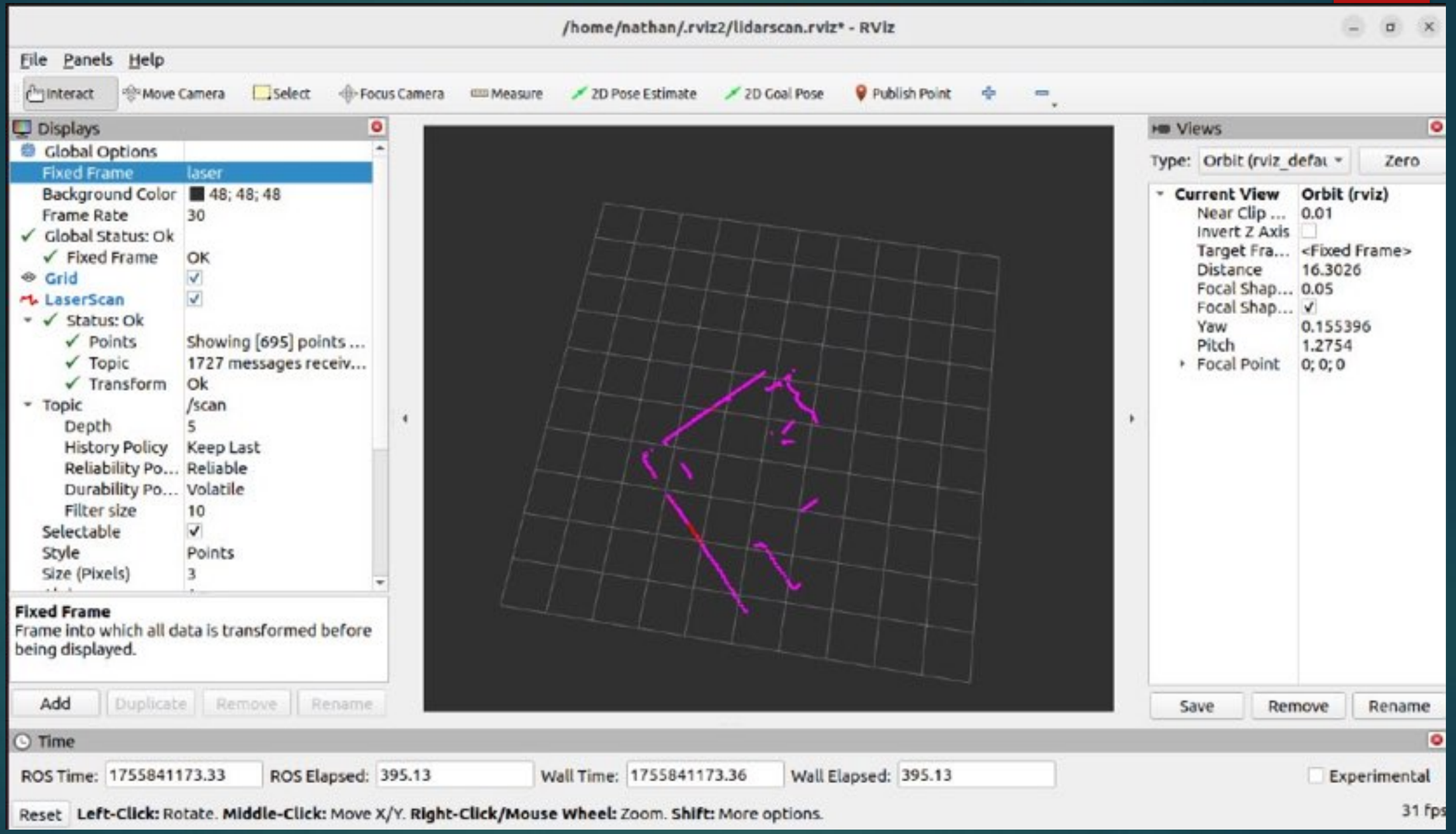
Key Tools for Mapping

ROS 2 offers robust solutions for mapping:

- **SLAM Toolbox**: A highly optimized and flexible ROS 2 package for 2D SLAM.
- **Cartographer**: Google's open-source real-time SLAM library, supporting 2D and 3D mapping.
- **GMapping**: A popular ROS 1 SLAM algorithm, often still used with ROS 2 through bridges for 2D laser-based mapping. based mapping.

Foundation for Autonomy

Accurate maps are indispensable for advanced robotic capabilities like autonomous navigation, path planning, obstacle planning, obstacle avoidance, and precise task execution in complex environments.


/home/nathan/.rviz2/lidarscan.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Displays
Global Options
Fixed Frame laser
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
LaserScan
Status: Ok
Points Showing [695] points ...
Topic 1727 messages receiv...
Transform Ok
Topic /scan
Depth 5
History Policy Keep Last
Reliability Po... Reliable
Durability Po... Volatile
Filter size 10
Selectable
Style Points
Size (Pixels) 3
Fixed Frame
Frame into which all data is transformed before being displayed.
Add
Duplicate
Remove
Rename
Views
Type: Orbit (rviz_defa
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 16.3026
Focal Shap... 0.05
Focal Shap...
Yaw 0.155396
Pitch 1.2754
Focal Point 0; 0; 0
Save
Remove
Rename
Time
ROS Time: 1755841173.33
ROS Elapsed: 395.13
Wall Time: 1755841173.36
Wall Elapsed: 395.13
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
31 fps