ROS2 Workshop

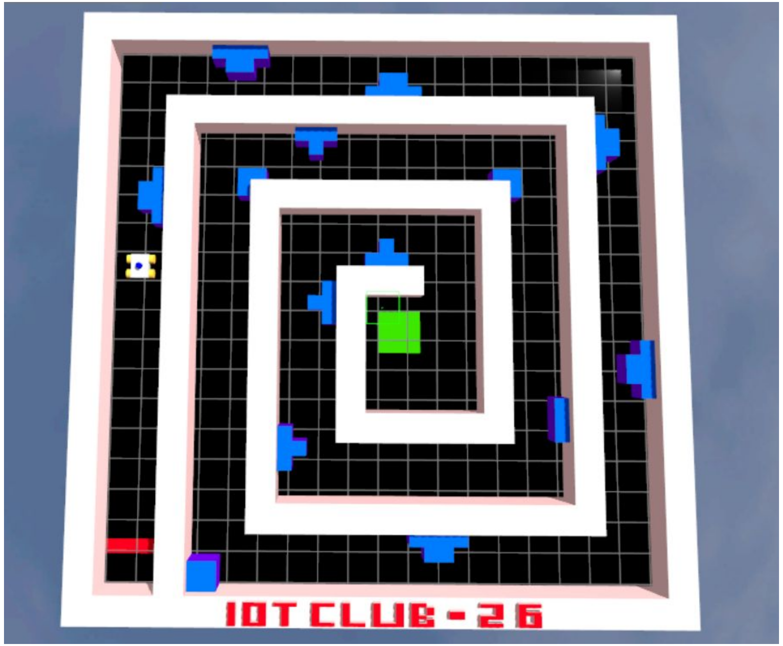
Basics to Advance

Build, connect, and control robots using ROS2!

- Learn the modern robotics middleware
- Understand communication patterns
- Work with real robots & simulation

By the End...

You'll be able to simulate your own Autonomous vehicle



Content

01. Introduction to Robotics
02. History of ROS and ROS2 and Architecture of ROS
03. Components - Nodes, Topics, Services, Actions, Parameters, Workspace and Packages (Turtlesim, rqt and rviz)
04. Custom Publisher & Subscriber
05. Custom Server & Client
06. Custom launch file in python (XML as well)
07. Transform
08. Gazebo
09. URDF and CAD files
10. FINAL Simulation (Basics, Controllers and PFN algo)!!!

Introduction to Robotics

Introduction

What Is Robotics?
Robotics is the science and engineering of creating machines that can perform tasks autonomously or semi-autonomously. It integrates hardware, software, and algorithms to give machines the ability to sense, plan, and act.

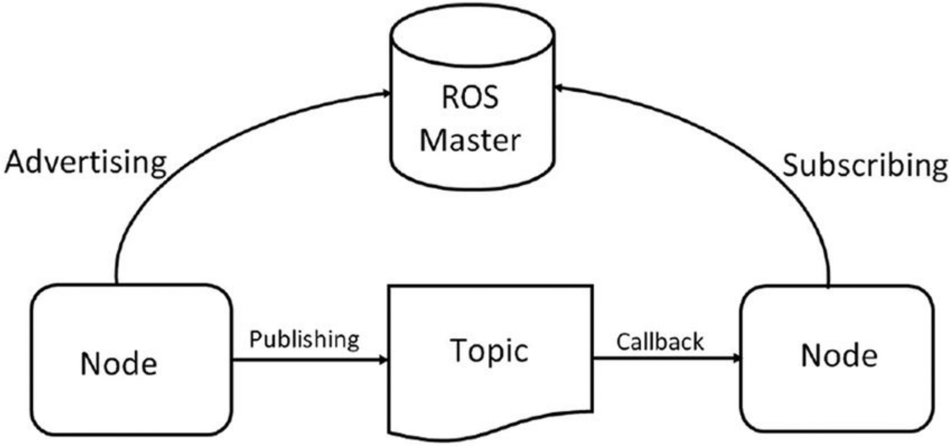
Why Learn Robotics?
Robotics allows us to solve real-world problems: automate repetitive work, explore dangerous environments, assist humans, and build future technologies like self-driving cars and service robots.

History and Architecture of ROS and ROS2

History of ROS and ROS2

ROS began in 2007 as an open-source robotics framework built for research and rapid prototyping.

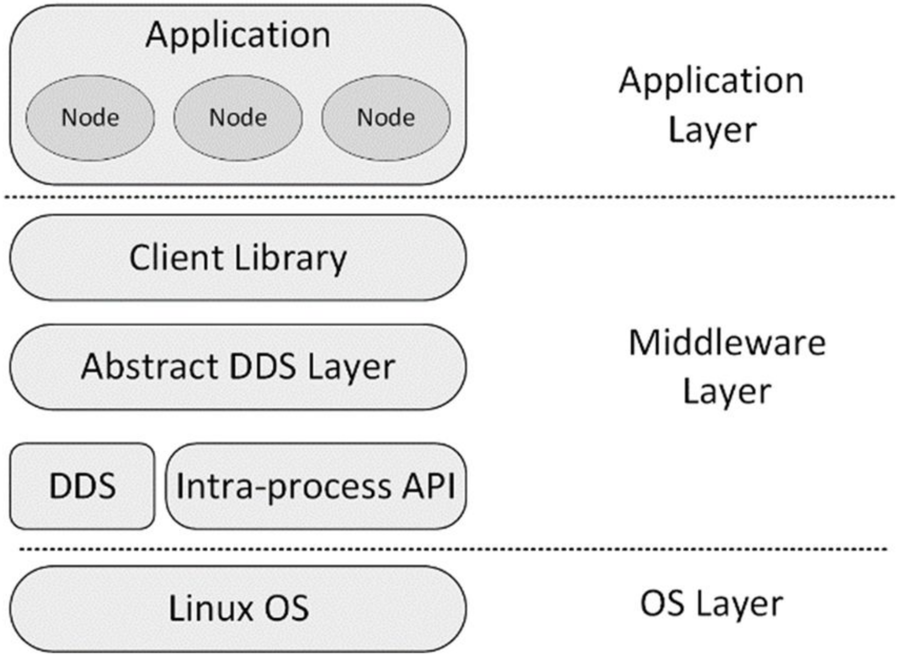
ROS1 provided a powerful foundation for modular robot software, but lacked real-time and multi-robot capabilities.



History of ROS and ROS2

ROS2, launched in 2014, evolved it into an industry-ready platform with real-time, secure, and scalable communication.

ROS2 was redesigned using DDS to deliver reliability, security, and cross-platform support.



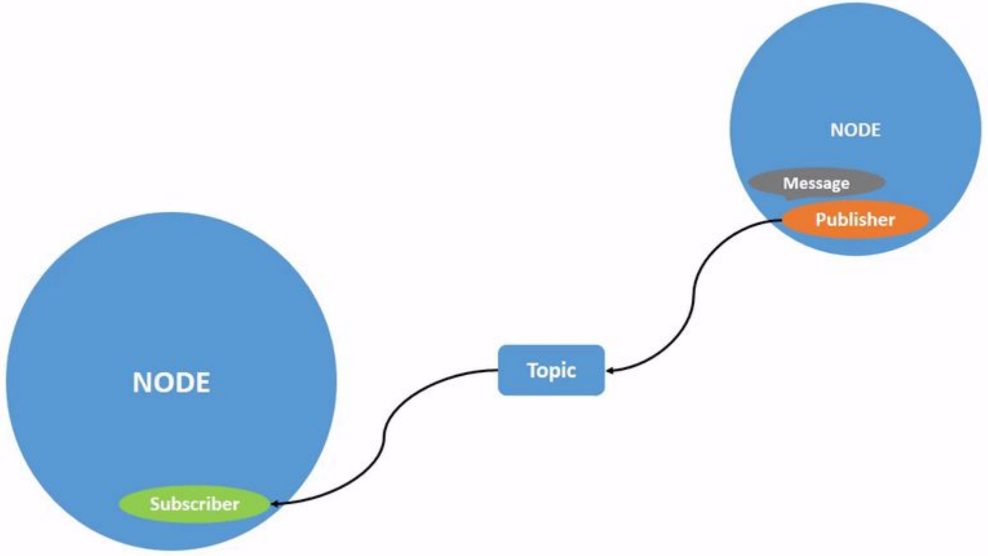
Different versions of ROS2

Components

Nodes and Topics

Nodes: Small, executable processes that perform specific tasks within a robot system.

Topics: Publish–subscribe communication channels used for sending continuous data between nodes.



Installation - Utilities & TurtleSim

Command

sudo apt update
sudo apt install terminator
echo ‘export ROS_LOCALHOST_ONLY=1’
sudo apt install ros-humble-turtlesim
sudo apt install '~nros-humble-rqt*'

Terminator
Cheat Sheet
LINUXSIMPLY

Window Management Shortcuts	
CTRL + SHIFT + T	Opens a new Terminal tab
CTRL + SHIFT + I	Opens a new Terminal window
CTRL + SHIFT + O	Split terminals Horizontally
CTRL + SHIFT + E	Split terminals Vertically
SUPER KEY + I	Spawns a new Terminator process
ALT + L	Opens layout launcher
CTRL + SHIFT + W	Closes the current terminal
CTRL+ SHIFT + Q	Closes the current window
CTRL + SHIFT + N	Minimizes the current terminal
CTRL + SHIFT + X	Maximizes the current terminal
CTRL + SHIFT + R	Resizes the current terminal
F11	Toggles window to fullscreen
CTRL + SHIFT + Z	Zooms current terminal

Navigation Shortcuts	
ALT + UP ARROW	Moves to the terminal above
ALT + DOWN ARROW	Moves to the terminal below
ALT + LEFT ARROW	Moves to the terminal left
ALT + RIGHT ARROW	Moves to the terminal right
CTRL + PAGE DOWN	Moves to next Tab
CTRL + PAGE UP	Moves to previous Tab
CTRL + TAB	Moves to next terminal
CTRL + SHIFT + TAB	Moves to the previous terminal
CTRL + SHIFT + F2	Toggles the terminal session between horizontal and vertical orientation
CTRL + SHIFT + L	Centers the viewport on the current line
SHIFT + HOME	Scrolls the viewport to the top of the terminal session
SHIFT + END	Scrolls the viewport to the bottom of the terminal session
CTRL + XX	Switches between the start of the line and the current cursor position

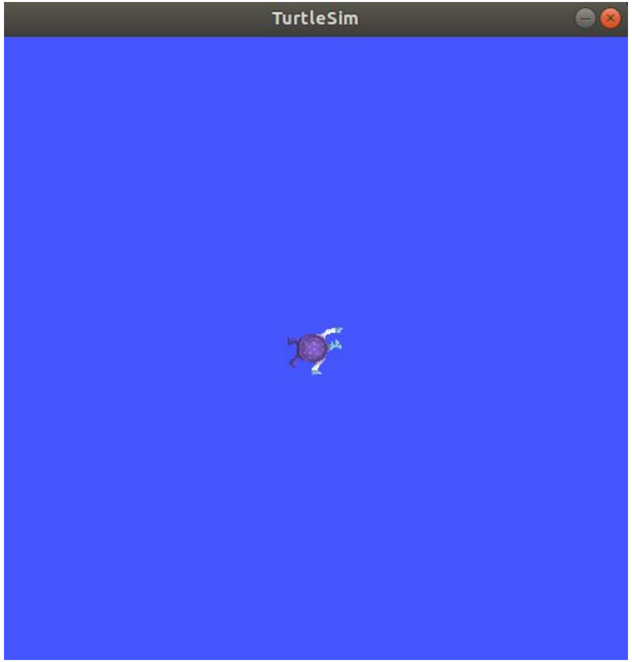
Turtlesim & CLI

Turtlesim

- Beginner-friendly ROS 2 simulator, uses a turtle to learn ROS concepts
- Helps practice topics, services, and actions and Simple way to understand robot movement, testing commands and control logic

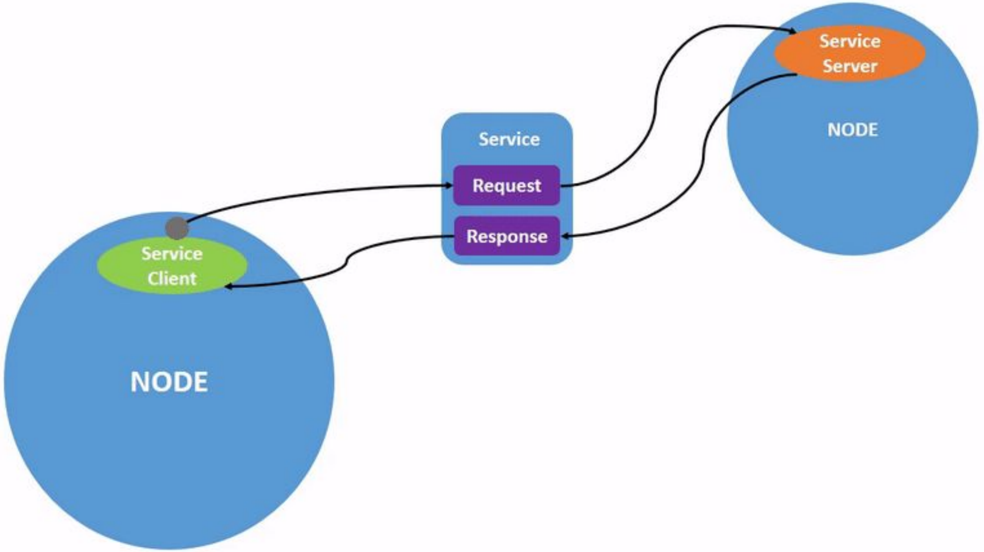
Command

ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key



Services

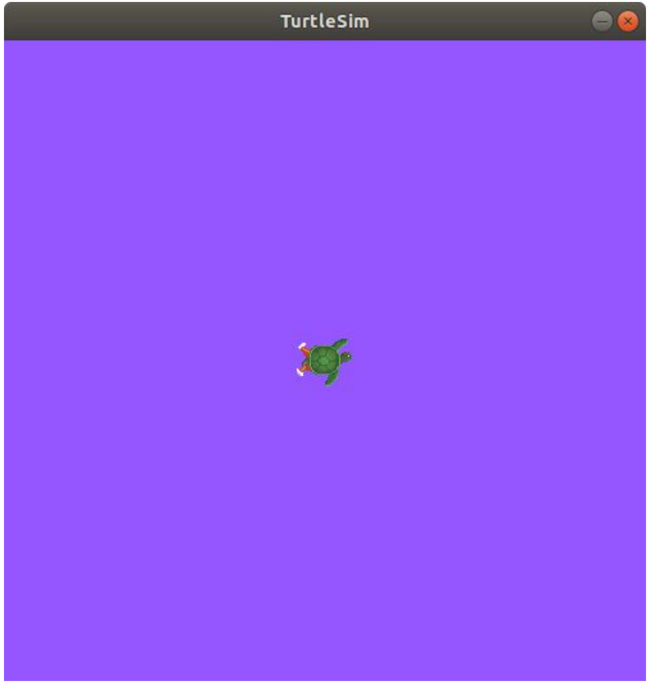
Services: Request–response communication used when a node needs to send a question and receive a direct answer.



Parameters

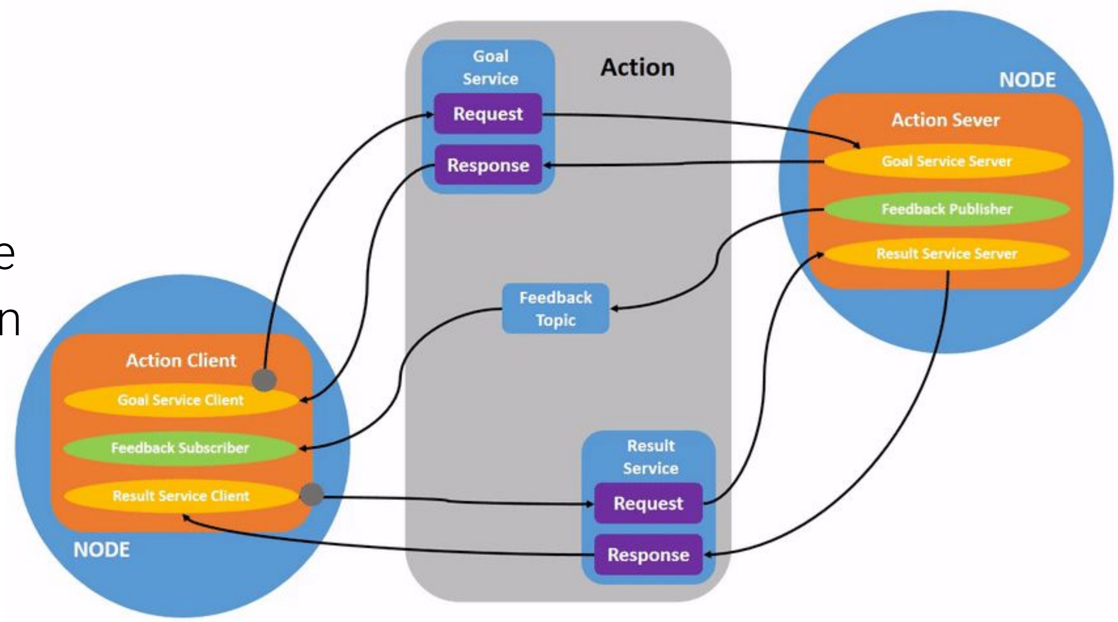
Parameters: Configurable values stored inside nodes that can be updated without changing the code.

Eg: `ros2 param set /turtlesim background_r 150`

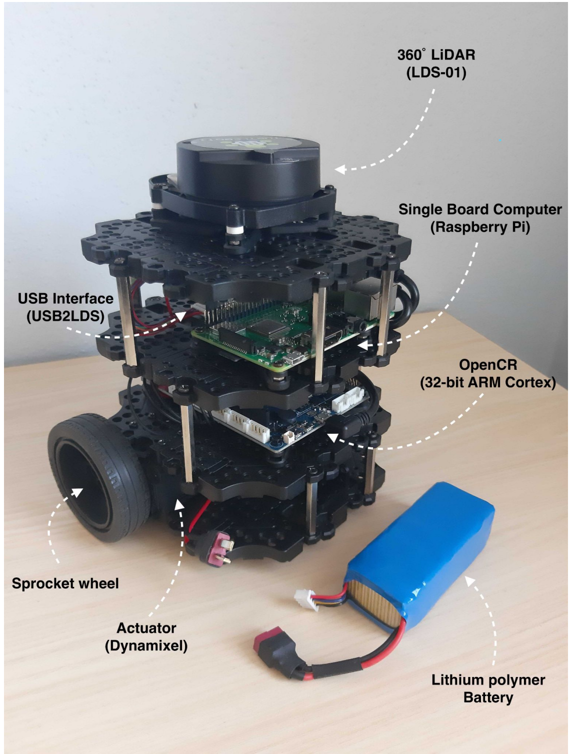


Actions

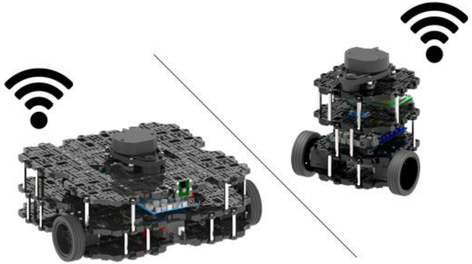
Actions: Asynchronous, long-duration tasks that provide feedback and allow cancellation (e.g., navigation goals).



Example



TurtleBot



ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
ROS_HOSTNAME = IP_OF_TURTLEBOT

Remote PC

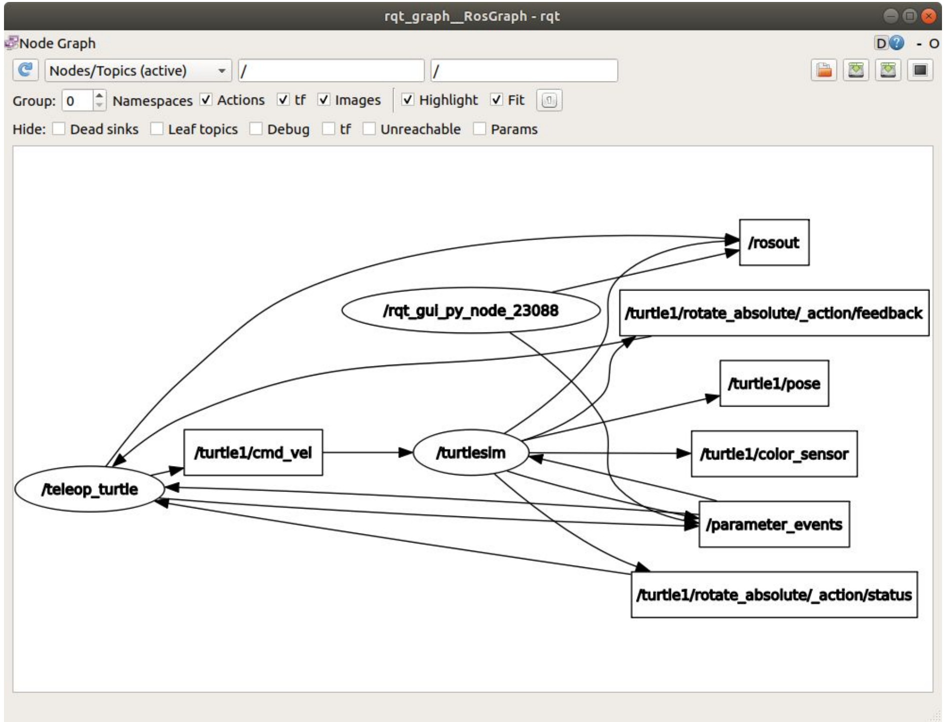
ROS_MASTER_URI = http://IP_OF_REMOTE_PC:11311
ROS_HOSTNAME = IP_OF_REMOTE_PC

* Example when ROS Master is running on the Remote PC

RQT

- GUI tool for monitoring ROS 2 systems
- Visualizes nodes, topics, and connections
- Supports plugins like rqt_graph and rqt_plot

Command - rqt



Custom Publisher & Subscriber

Custom Server & Client

Custom launch file in python (XML as well)

Workspace and Packages

Workspace: A directory structure where ROS2 packages are developed, built, and organized.

Eg:

```
mkdir -p ~/ros2_ws/src
cd ~/ros2_ws/src
```

Packages: The basic building blocks of ROS2—containing nodes, code, configuration, and launch files.

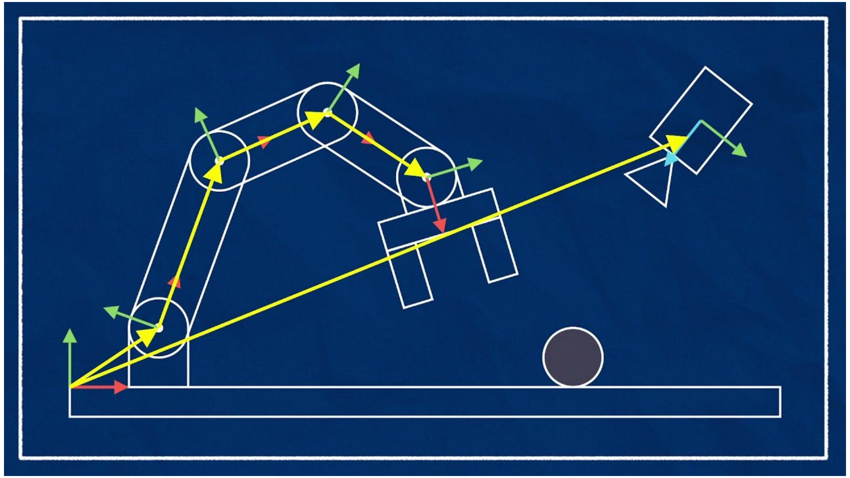
Eg:

```
ros2 pkg create --build-type ament_python <package_name>
```

Transform

Transform

- TF (Transform) manages coordinate frames in a robot system
- It tracks position & orientation between links (base, sensors, end-effector)
- Helps robots understand spatial relationships in real time
- We have TF2 library in ROS 2 for accuracy and speed



```
sudo apt install ros-humble-xacro ros-humble-joint-state-publisher-gui
```

```
ros2 run robot_state_publisher robot_state_publisher --ros-args -p robot_description:="$(xacro path/to/my/xacro/file.urdf.xacro)"
```

```
ros2 run joint_state_publisher_gui joint_state_publisher_gui
```

URDF and CAD files

URDF and CAD

URDF:-

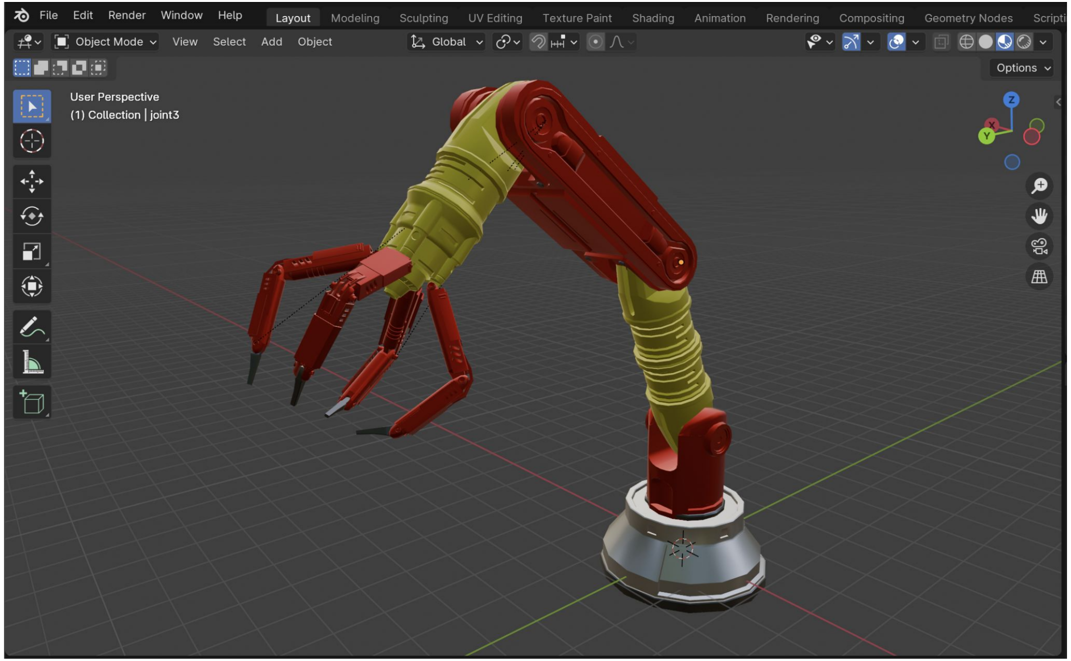
- Defines links, joints, sensors, and visuals
- Helps visualize robots in RViz and simulators
- Essential for robot modeling in ROS 2

CAD (Computer-Aided Design)

- Used to design robot parts and assemblies
- Helps ensure accurate dimensions before building
- CAD models can be converted for URDF use

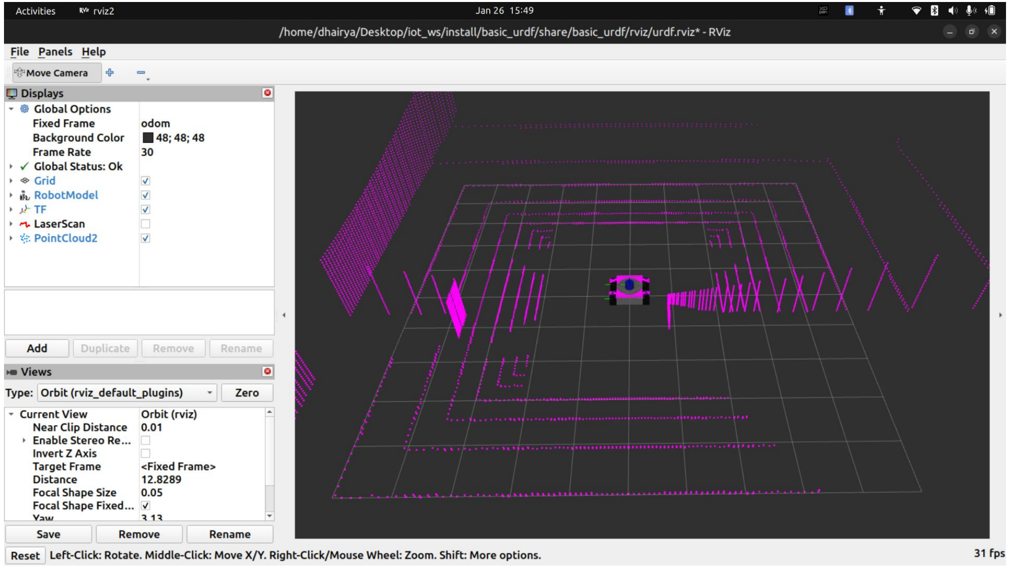
CAD softwares

Blender Example



RVIZ

- 3D visualization tool in ROS 2
- Displays robot models and sensor data
- Shows maps, paths, and TF frames
- Useful for navigation and perception

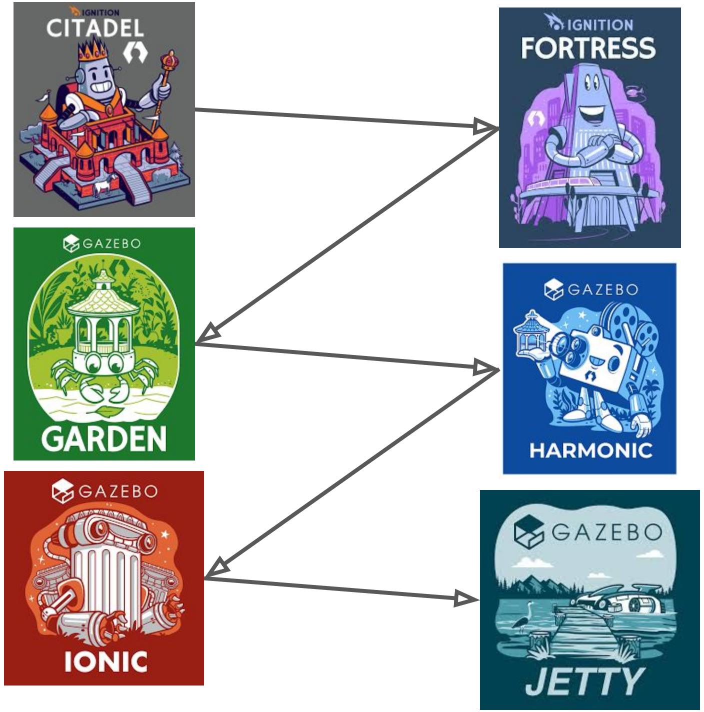


Command - rviz2

Gazebo

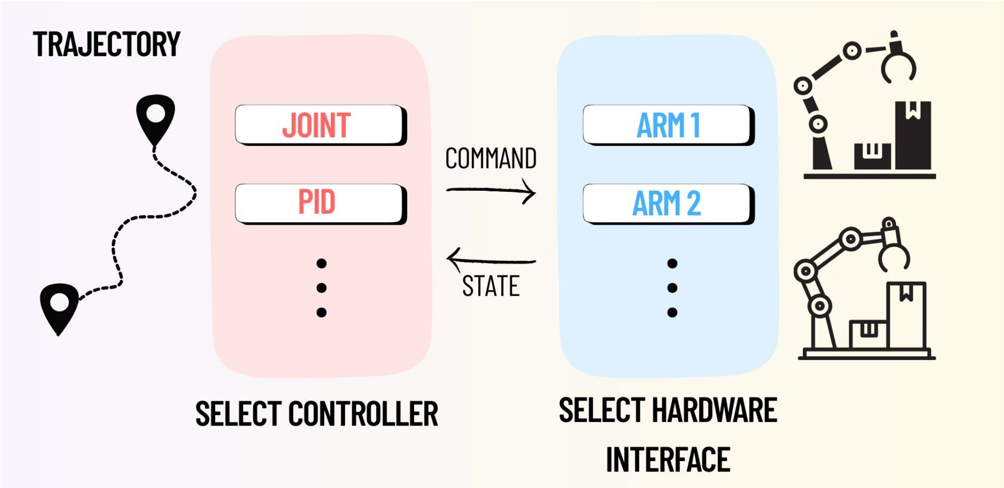
Gazebo

- 3D robot simulation environment
- Used to test robots without real hardware
- Provides realistic physics, sensors, and environments
- Saves cost and improves safety during development
- Integrates smoothly with ROS 2



Controllers

- Controllers manage robot motion and behavior
- Convert commands into motor/joint actions
- Work with hardware or simulation
- Examples: position, velocity, effort controllers



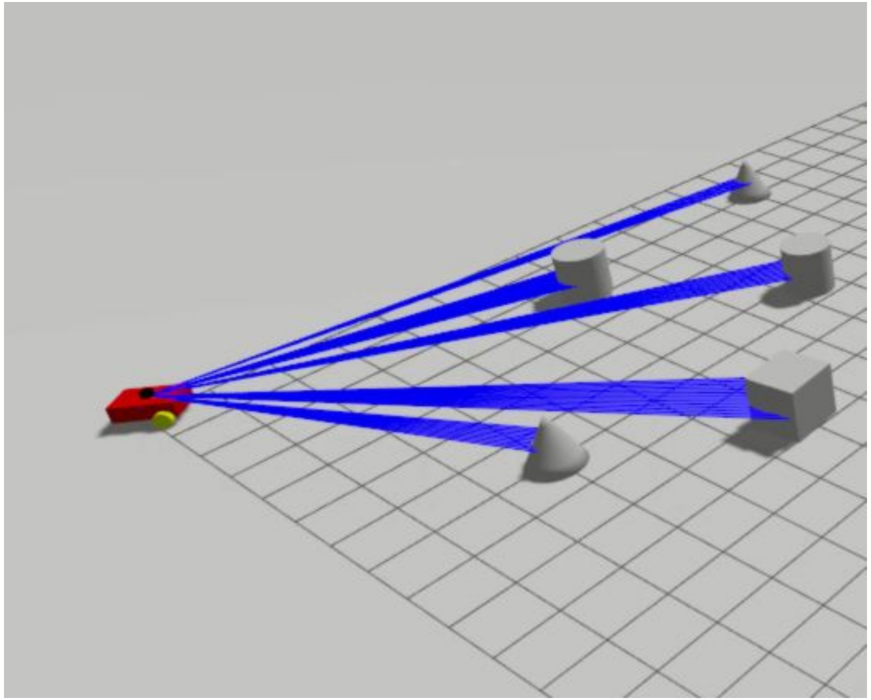
FINAL Simulation

(Basics, Controllers and Potential field navigation algo)!!!

Potential Field Algorithm

Potential Field Navigation is a reactive path-planning algorithm used in mobile robotics where:

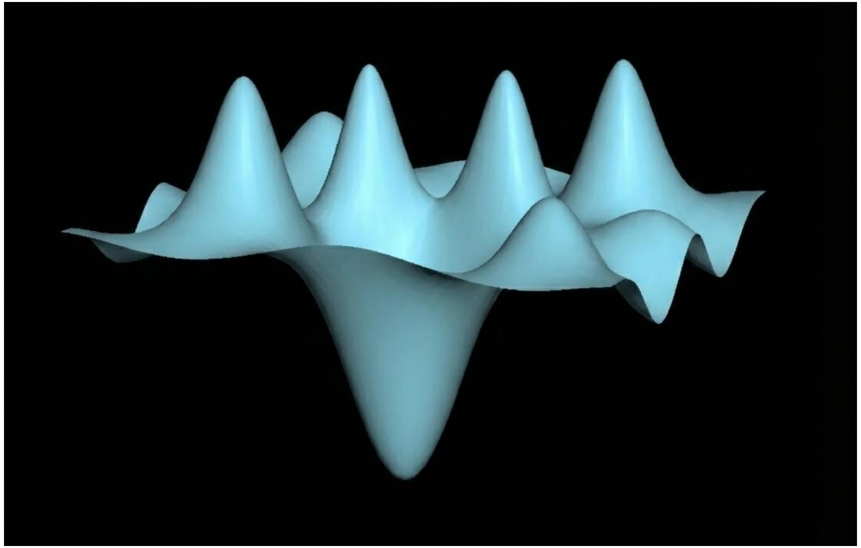
- The goal exerts an attractive force
- Obstacles exert repulsive forces
- The robot moves as if it is a particle in an artificial force field



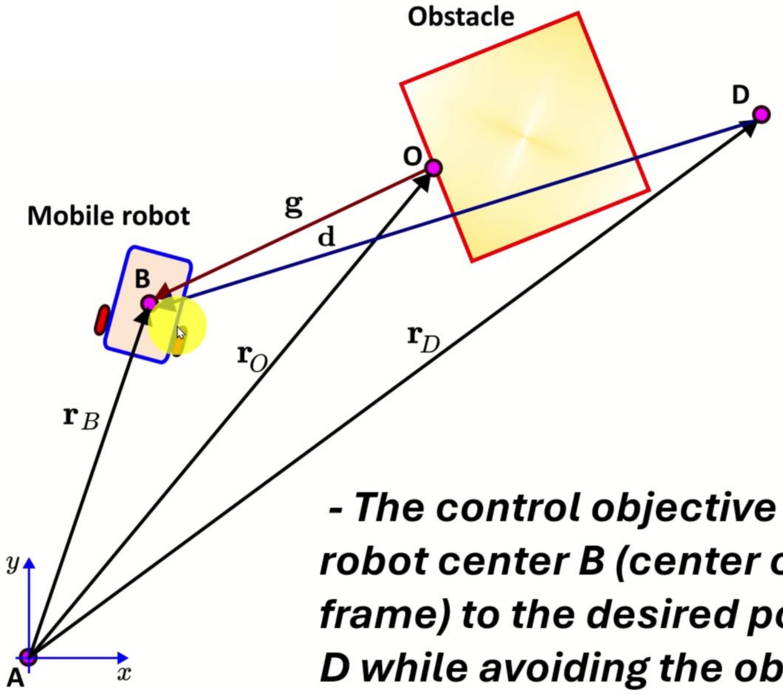
Potential Field Algorithm

Analogy:-

- Imagine the robot as a ball rolling on a surface
- The goal is a deep valley
- Obstacles are high hills
- The robot naturally rolls downhill toward the goal, avoiding hills

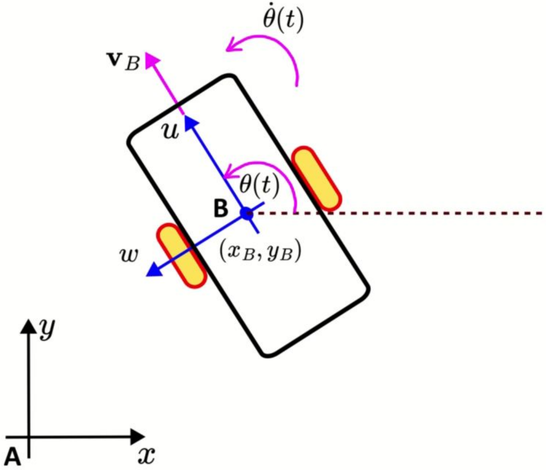


Potential Field Algorithm



- The control objective is to move the robot center B (center of the body frame) to the desired point (goal point) D while avoiding the obstacle.

Potential Field Algorithm

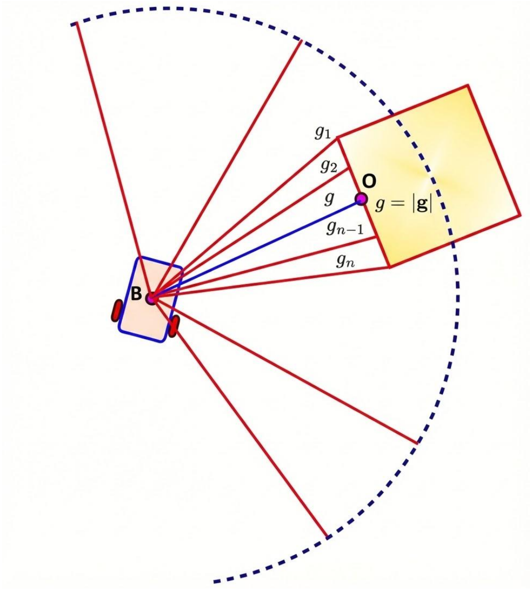


- **Position:** (x_B, y_B)
- **Orientation:** $\theta(t)$
- **Angular velocity:** $\dot{\theta}(t)$
- **Linear velocity:** $\mathbf{v}_B$
- **Body frame:** B_{uw}

- ***We assume that position*** (x_B, y_B) ***and robot orientation*** $\theta(t)$ ***are measured.***

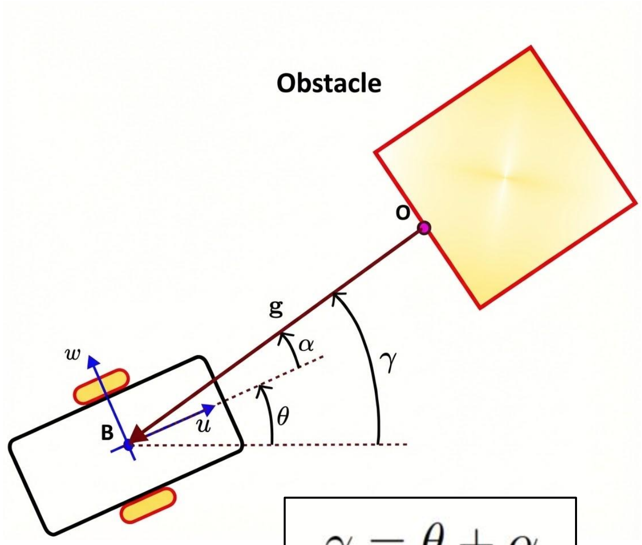
Potential Field Algorithm

- Using LiDAR, we obtain distance measurements to the obstacle from multiple scan points.



Potential Field Algorithm

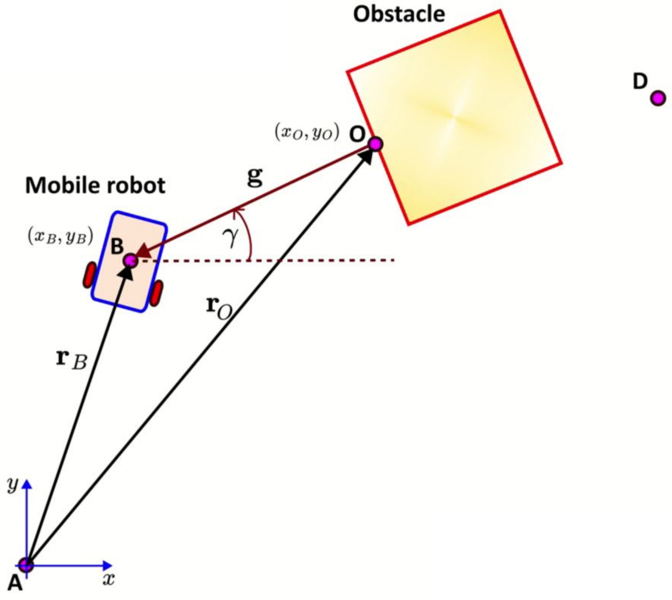
- For each scan point, the corresponding angle is computed with respect to the robot's body frame.
- These angles are then transformed into the global (inertial) frame by adding the relative orientation between the body frame and the global frame.



$$\gamma = \theta + \alpha$$

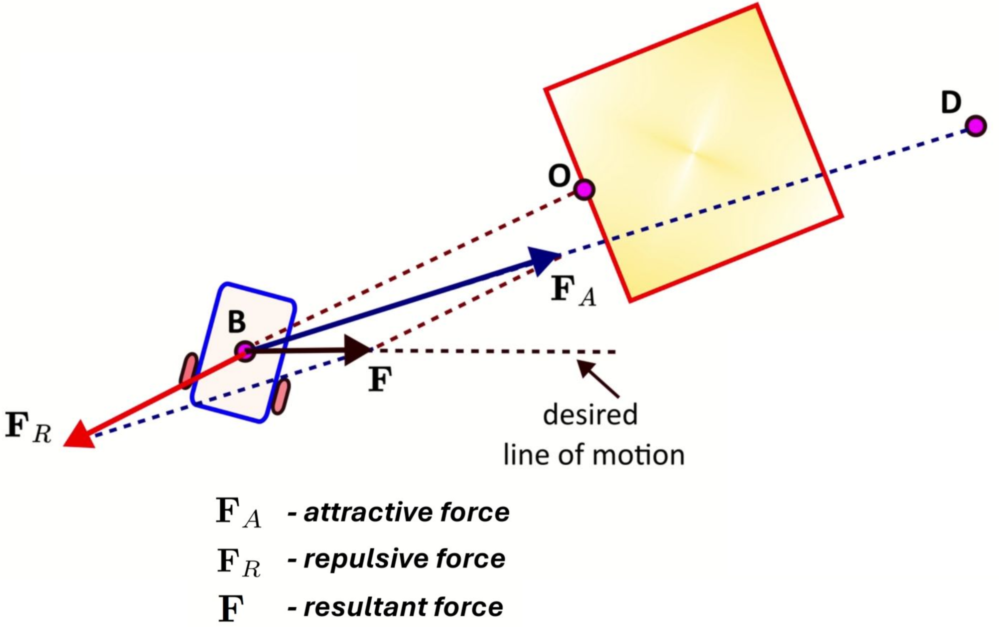
Potential Field Algorithm

- Using these values we can estimate the points on the obstacle w.r.t. global frame.
- And later we can calculate the resultant artificial force exerted by the surrounding on our robot.



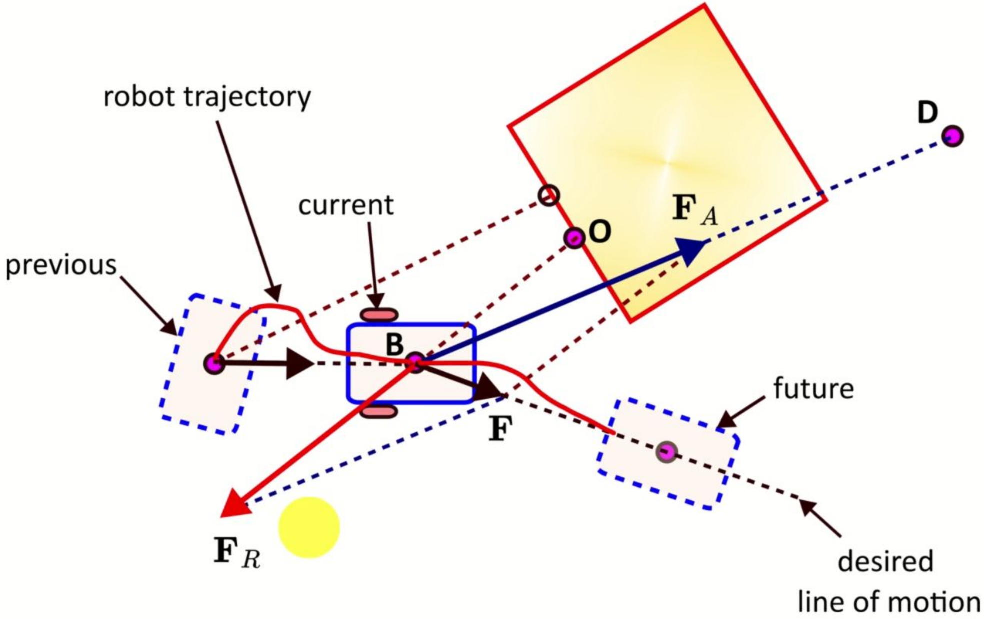
Potential Field Algorithm

- Using these values we can estimate the points on the obstacle w.r.t. global frame.
- And later we can calculate the resultant artificial force exerted by the surrounding on our robot.



Potential Field Algorithm

- We repeat these steps again and again to get desired path which avoids the obstacle.



Potential Field Algorithm

Key Notes:

- The Attractive force is applicable throughout the navigation.

$$U_{att}(q) = \frac{1}{2} k_{att} \| q - q_{goal} \|^2$$

- The Repulsive force is only exerted when the robot is within certain distance of the obstacle.

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{d(q)} - \frac{1}{d_0} \right)^2 & d(q) \leq d_0 \\ 0 & d(q) > d_0 \end{cases}$$

Let’s move onto the code!!!

Thank U
for your time!!!