EE478
Introduction to robot system integration using Robot Operating System(ROS)

(TA) Seongwoo Moon, Sungjae Min, Donghun Han
School of Electrical Engineering
KAIST

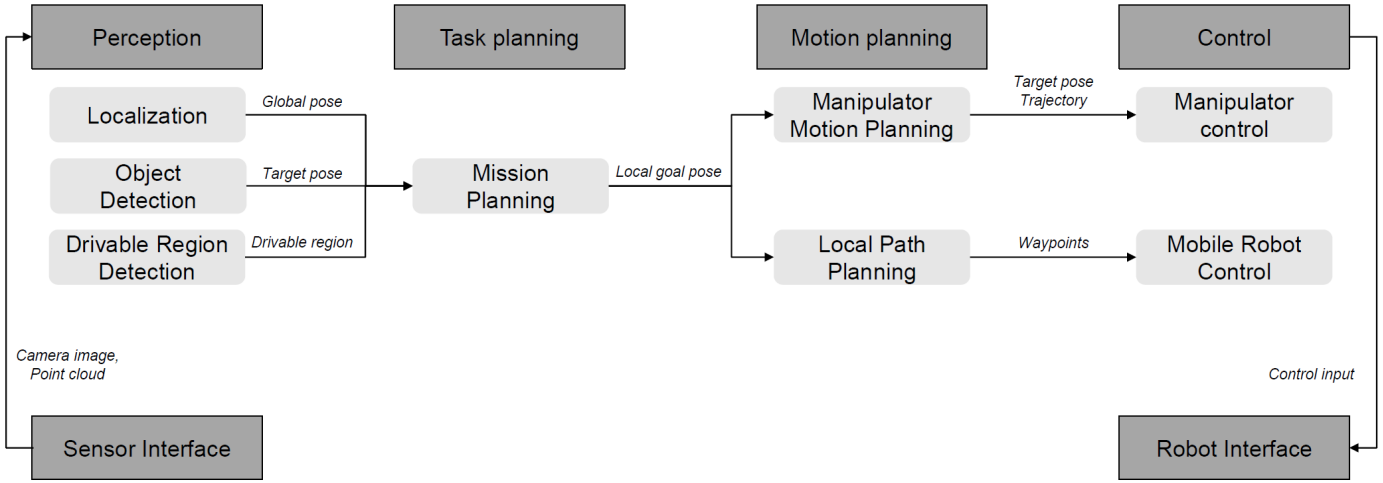
Contents

- Complexity of Robot Systems
- ROS Overview and Modularized Architecture
- ROS Tools
- Case Study: Robot system integration with ROS

Complexity of Robot Systems

- Components of modern robot systems
  - Sensors (encoders, IMUs, cameras, LiDAR, ultrasonic, etc.)
  - Actuators
  - Computing Systems
  - Power Systems
  - Mechanical structure
  - Communication systems

  - and software to drive and interconnect all components above



Complexity of Robot Systems

- Challenges in robot system integration
  - Hardware-Software Integration
    - Interfacing diverse hardware components
    - Driver development and compatibility
  - Data fusion and Synchronization
    - Combining data from multiple sensors
    - Dealing with different sampling rates and latencies
  - Real-time Constraints
    - Balancing computational load
    - Time response in critical systems
  - Scalability and Modularity
    - Managing complexity as system size
    - Code re-useability and Utilizing open-source codes
  - Human-Robot Interaction
    - Intuitive graphic user interface

Complexity of Robot Systems

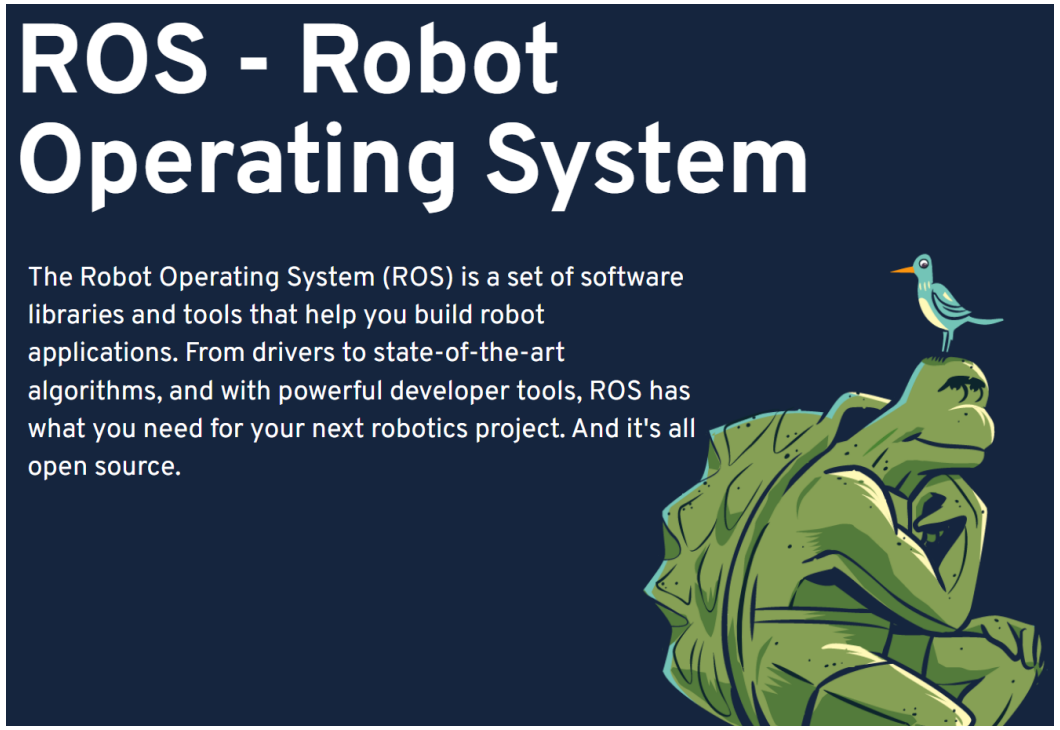
- Need for standardized middleware
  - Interoperability
    - Enabling communication between diverse components
  - Reusability
    - Promoting code and component reuse across projects
    - Reducing development time and costs
  - Community and Ecosystem
    - Fostering collaboration and knowledge sharing
    - Develop specific part of software rather implementing whole architecture
  - Abstraction of Low-level Complexities
    - Hiding hardware-specific details
    - Allowing focus on high-level robotic problems
  - Standardized Communication Protocols
    - Ensuring consistent and easy data exchange
    - Simplifying system integration

ROS Overview and Modularized Architecture

❖ Definition and Purpose of ROS

❑ What is ROS?

- ➢ Robot Operating System(ROS): A software framework for writing robot software

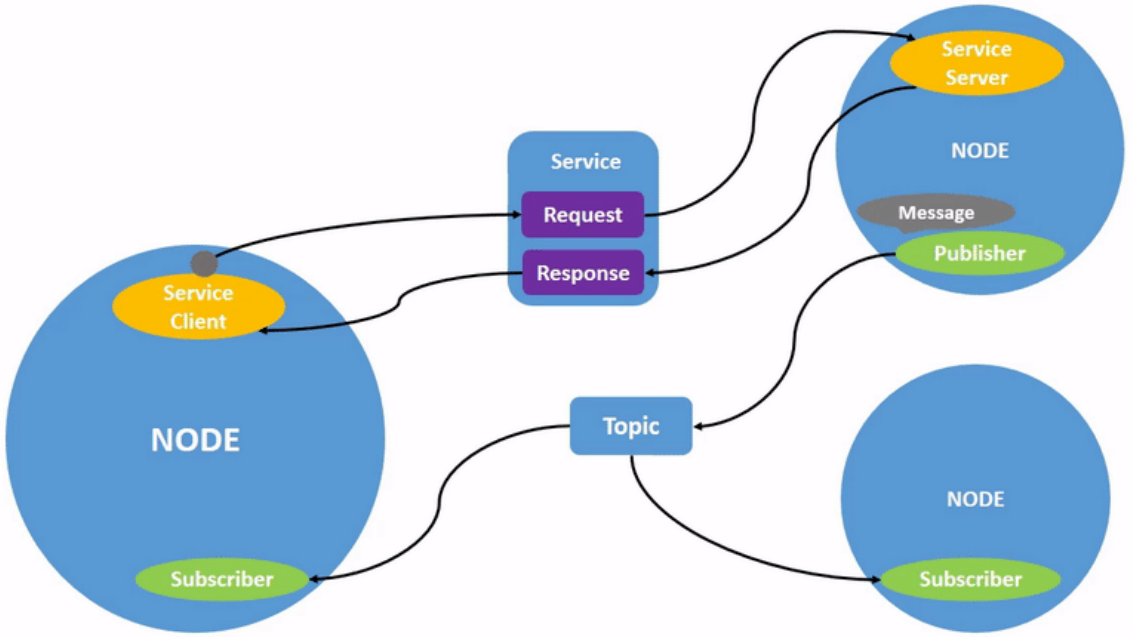


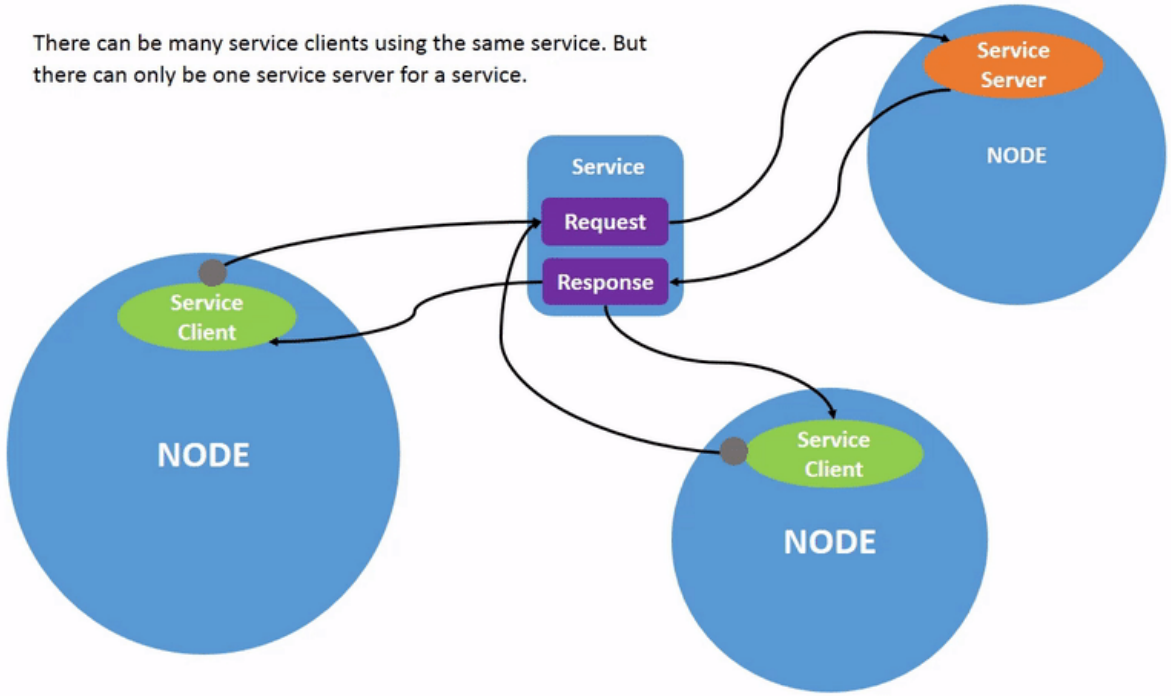
https://ros.org/

- ➢ Not an actual operating system, but a middleware providing OS-like functionality
  - ▪ OS: Windows, Ubuntu, Android, MacOS, IOS, …)

ROS Overview and Modularized Architecture

- Definition and Purpose of ROS
  - Key Features
    - Simplify the task of creating complex robot task
      - Modularized approach in ROS using Nodes and Messages





ROS Overview and Modularized Architecture

- Modularized Architecture of ROS
  - Concept and Advantages of Modularized Systems
    - Definition: A system composed of separate, interchangeable components
    - Advantages:
      - Flexibility: Easy to modify or replace individual components
      - Reusability: Components can be used in different systems or projects
      - Maintainability: Easier to update or fix specific parts of the system
      - Scalability: New functionalities can be added by introducing new modules
      - Parallel development: Different teams can work on different modules simultaneously

ROS Overview and Modularized Architecture

https://docs.ros.org/en/melodic/api/std_msgs/html/index-msg.html

❖ Modularized Architecture of ROS

❑ Messages

- ➢ Definition: Typed data structures for communication in ROS
- ➢ Usage:
  - ▪ Serve as data types for Topics, Services, and Actions
  - ▪ Enable standardized data exchange between nodes
- ➢ Characteristics:
  - ▪ Standardized: ROS provides common message types for typical robot data
  - ▪ Customizable: Users can define custom message types for specific needs
- ➢ Common Message Types:
  - ▪ std_msgs: For primitive data types (e.g., Int32, String)
  - ▪ sensor_msgs: For sensor data (e.g., Image, LaserScan)
  - ▪ geometry_msgs: For spatial information (e.g., Pose, Transform)
- ➢ Custom Messages:
  - ▪ Defined using simple text files with a .msg extension
  - ▪ Can combine existing message types and primitive data types

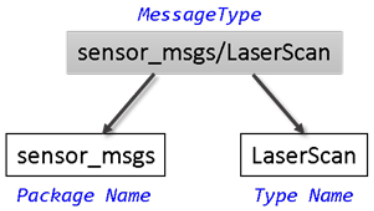
std_msgs Msg/Srv Documentation

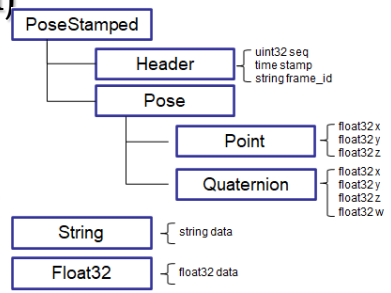
See also:

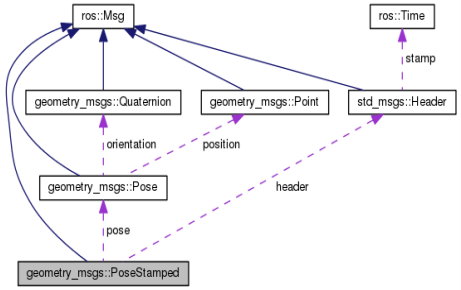
- Website
- Code API Documentation

Message types

- Bool
- Byte
- ByteMultiArray
- Char
- ColorRGBA
- Duration
- Empty
- Float32
- Float32MultiArray
- Float64
- Float64MultiArray
- Header
- Int16
- Int16MultiArray
- Int32
- Int32MultiArray
- Int64
- Int64MultiArray
- Int8





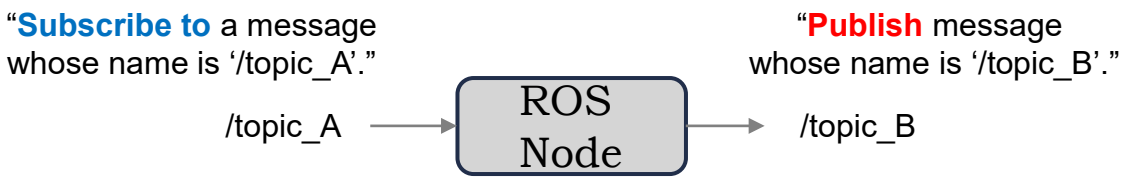


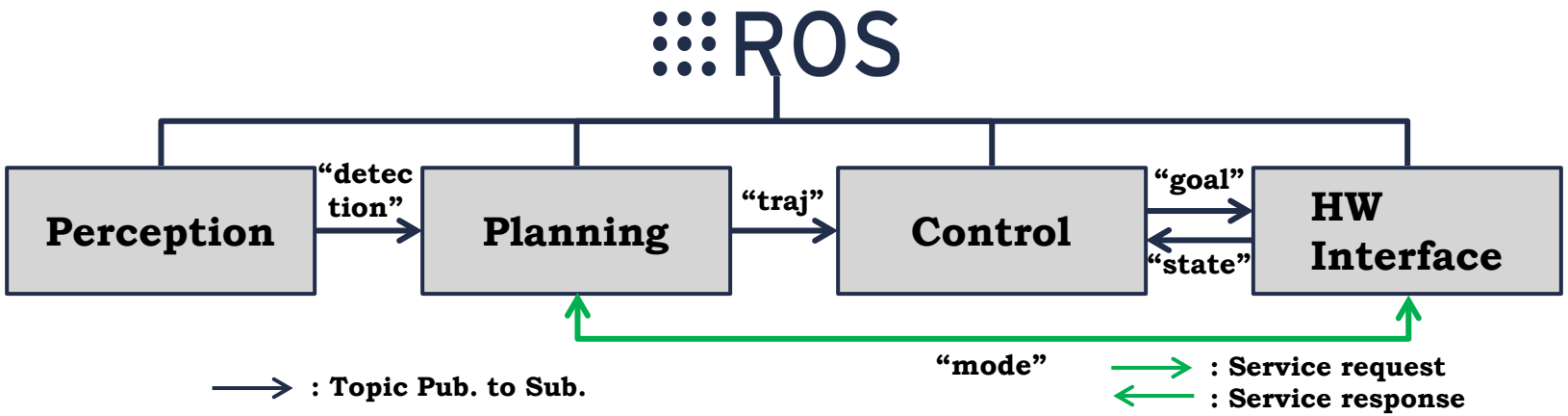
ROS Overview and Modularized Architecture

❖ Modularized Architecture of ROS

❑ Introduction to ROS Nodes

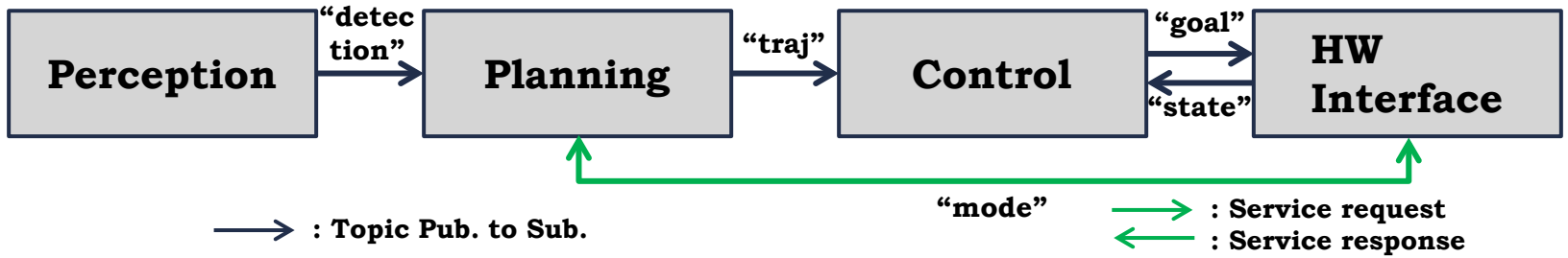
- ➢ Definition: Basic modular units in ROS
- ➢ Characteristics:
  - ▪ Encapsulation: Each node contains specific functionality
  - ▪ Loose coupling: Nodes interact through well-defined interfaces
  - ▪ Single responsibility: Each node typically handles one aspect of the robot's functionality





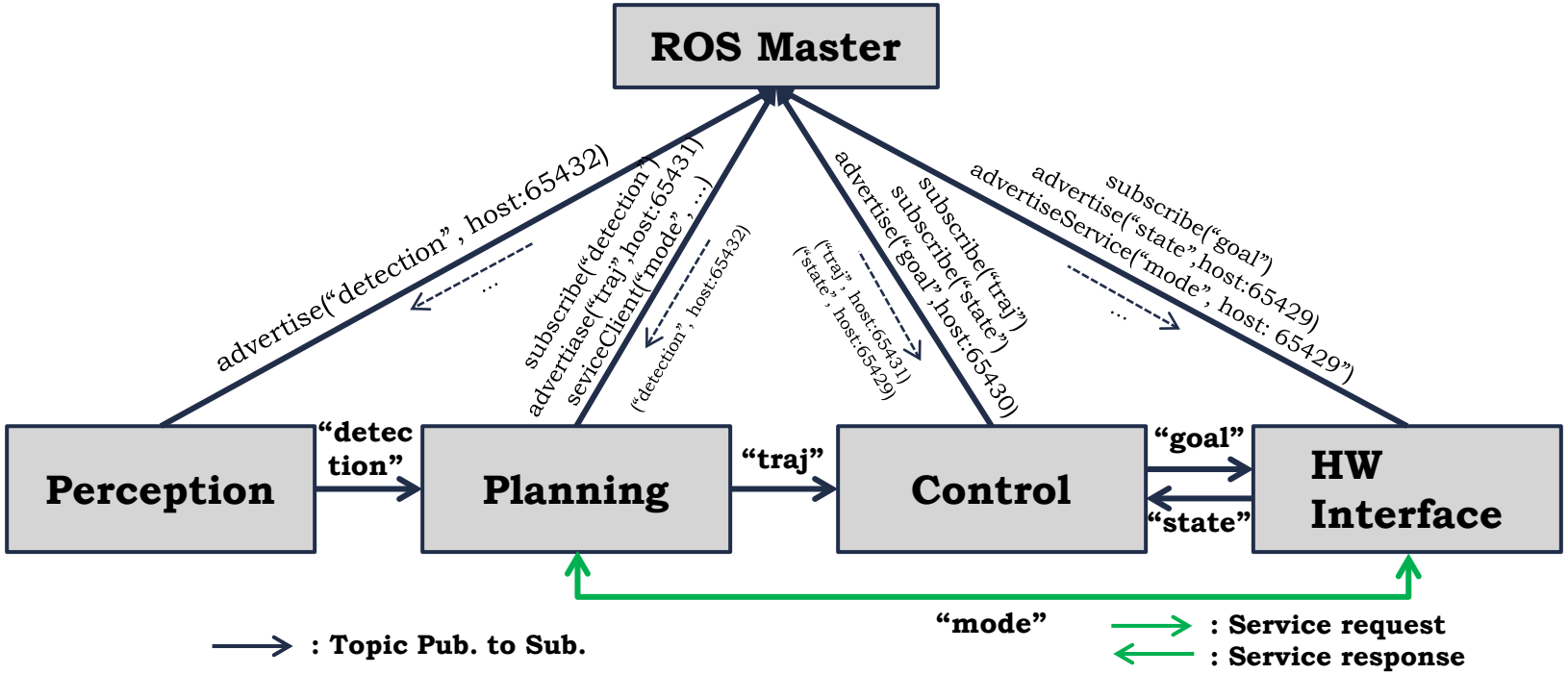
ROS Overview and Modularized Architecture

- Modularized Architecture of ROS
  - Topics and Services system
    - Topics:
      - Publish-subscribe model for inter-node communication
      - Enable asynchronous, many-to-many data streaming between nodes
    - Services:
      - Request-response model for inter-node communication
      - Enable synchronous, one-to-one interactions between nodes



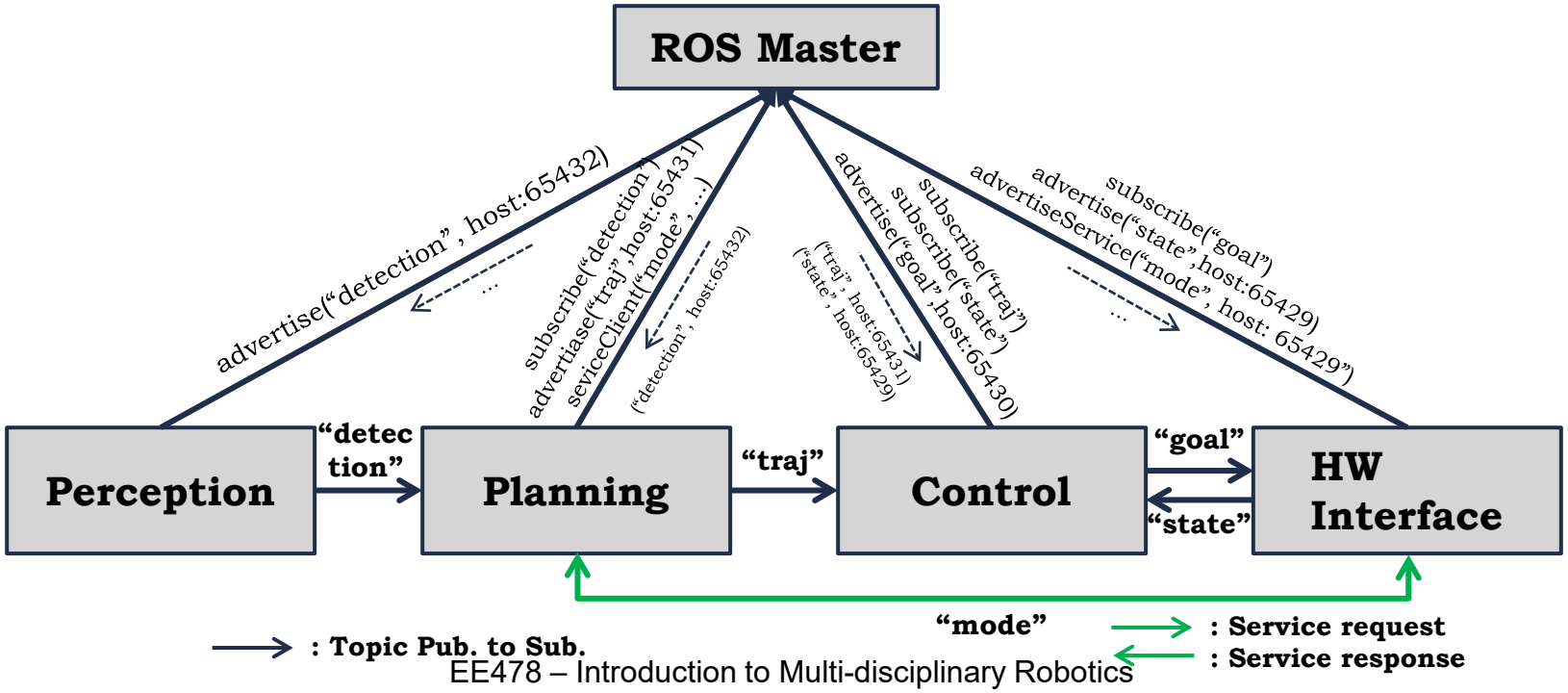
ROS Overview and Modularized Architecture

- Modularized Architecture of ROS
  - ROS Master: Central Coordinator
    - Purpose: Facilitates node discovery and communication setup
    - Role in modular architecture:
      - Enables dynamic configuration of the robot system
      - Allows nodes to be added or removed at runtime



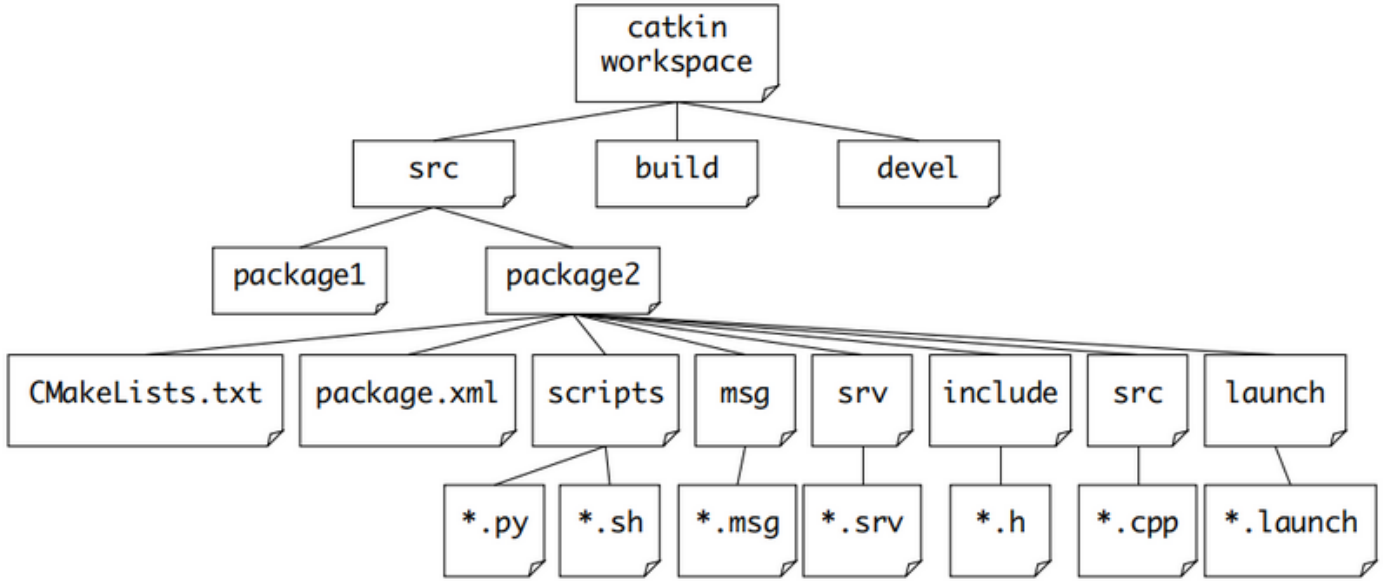
ROS Overview and Modularized Architecture

- Modularized Architecture of ROS
  - ROS Communication Flow in Modular Architecture
    - Nodes register with the ROS Master
    - Nodes advertise their capabilities (topics, services, actions)
    - ROS Master facilitates connections between nodes based on their advertised capabilities
    - Nodes communicate directly using the established connections
    - System can be reconfigured by adding, removing, or modifying nodes



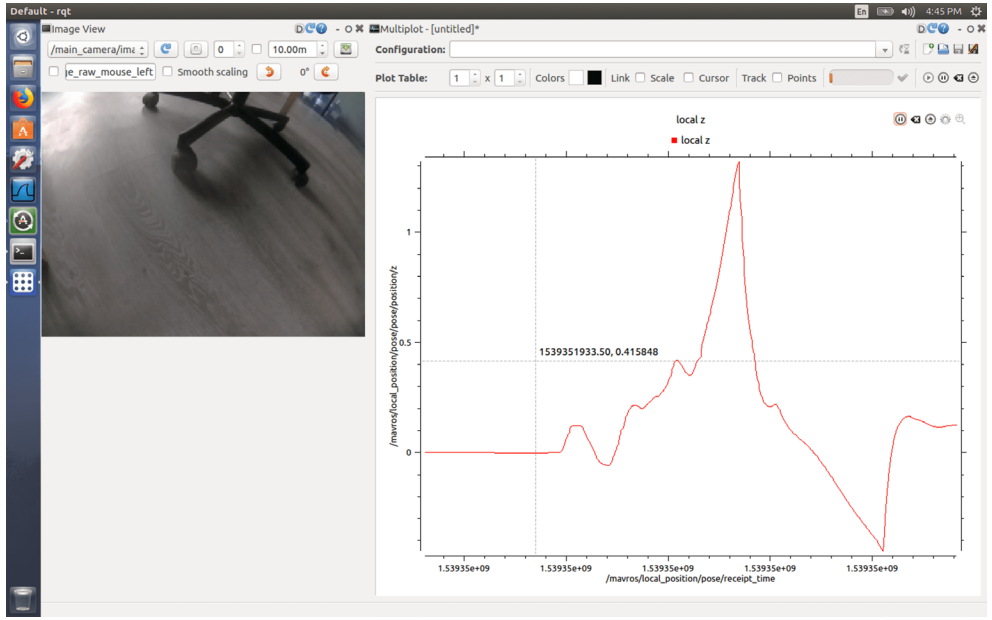
ROS Overview and Modularized Architecture

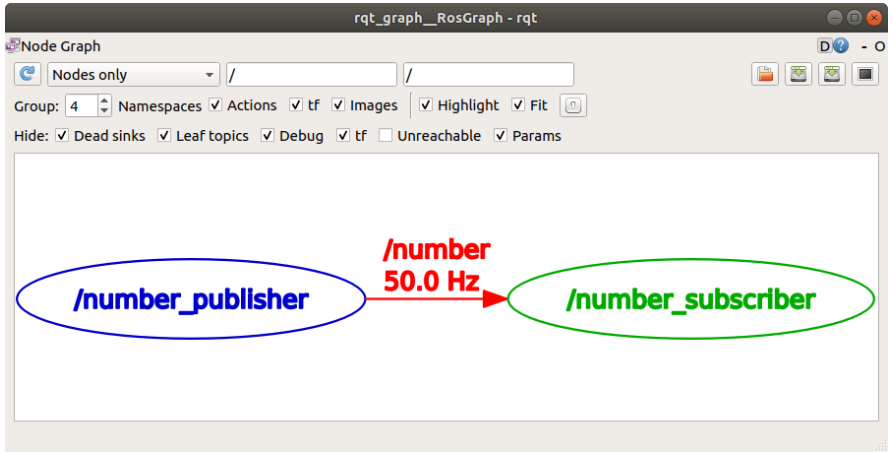
- Modularized Architecture of ROS
  - ROS Packages
    - Definition: Basic unit of software organization in ROS
    - Contents: Nodes, libraries, datasets, configuration files, etc.
    - Benefits:
      - Promotes modularity and code reuse
      - Simplifies distribution and installation of ROS software



ROS Tools

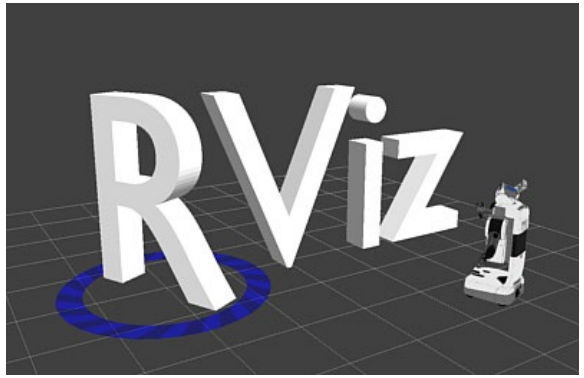
- rqt toolsuite
  - rqt is a Qt-based framework for GUI development for ROS.
    It provides various plugins for visualizing and introspecting ROS systems.
  - Key features:
    - Customizable dashboard for robot monitoring
    - Tools for debugging and visualizing ROS graph
    - Plugins for various tasks like plotting, parameter editing, and service calling

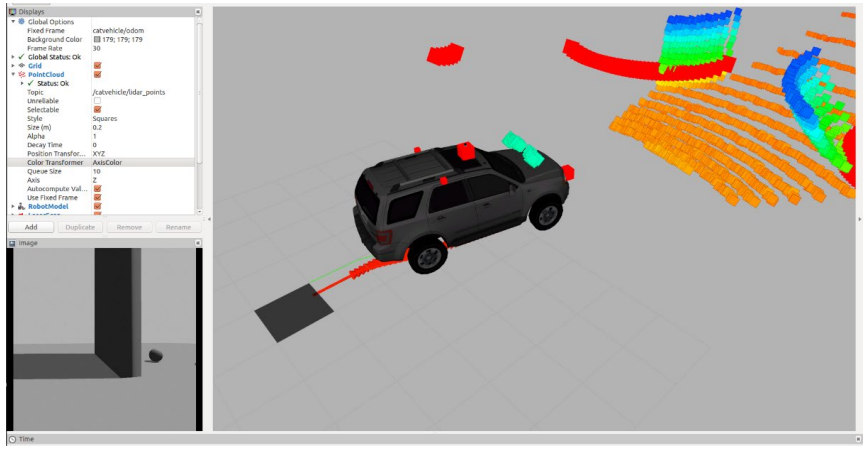


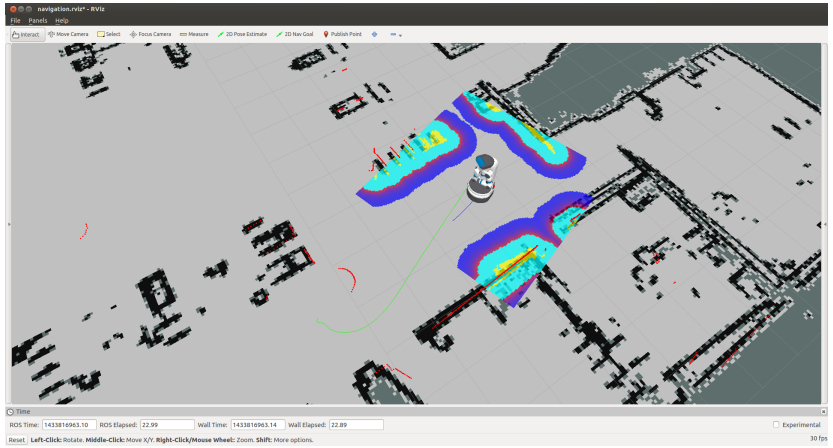


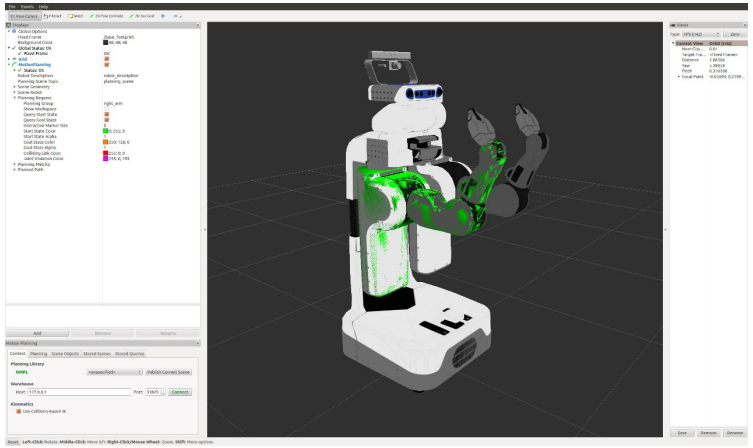
ROS Tools

- Rviz (ROS Visualization)
  - Rviz is a 3D visualization tool for ROS. It allows users to visualize sensor data and state information from ROS.
  - Main capabilities:
    - Displays live sensor data in 3D environment
    - Visualizes robot models and transformations
    - Customizable views and plugins for specific visualization needs



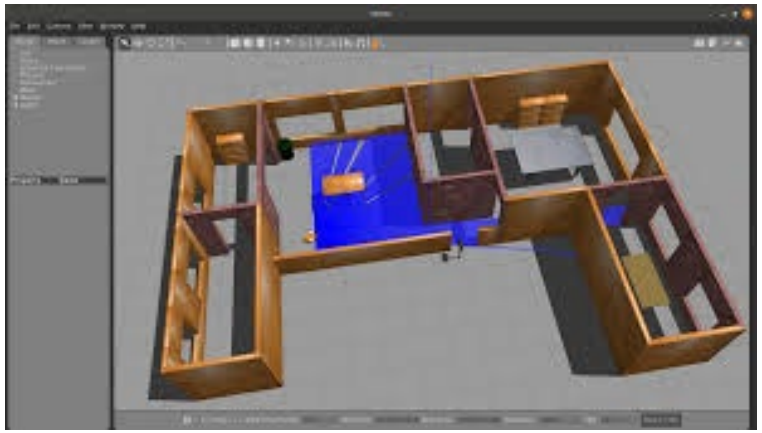


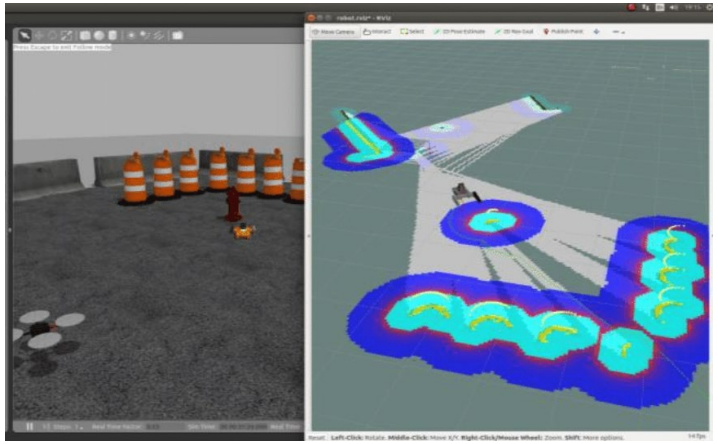




ROS Tools

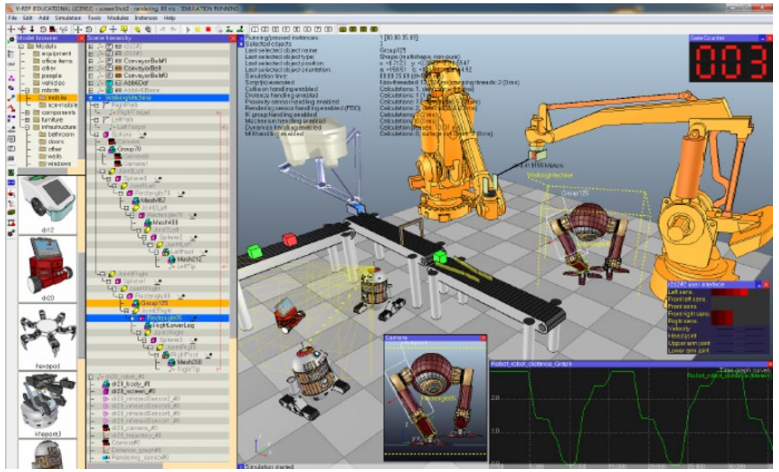
- Gazebo
  - Gazebo is the default 3D robot simulation tool provided with ROS, offering robust capabilities for simulating robots in various environments.
  - Key features:
    - Physics simulation for realistic robot behavior
    - Sensor simulation (cameras, LiDAR, IMU, etc.)
    - Library of robot models and environments
    - Custom robot and world creation tools
  - Main uses:
    - Testing robot designs and control algorithms
    - Developing perception and navigation systems
    - Simulating single and multi-robot systems
    - Training and validating AI models for robotics





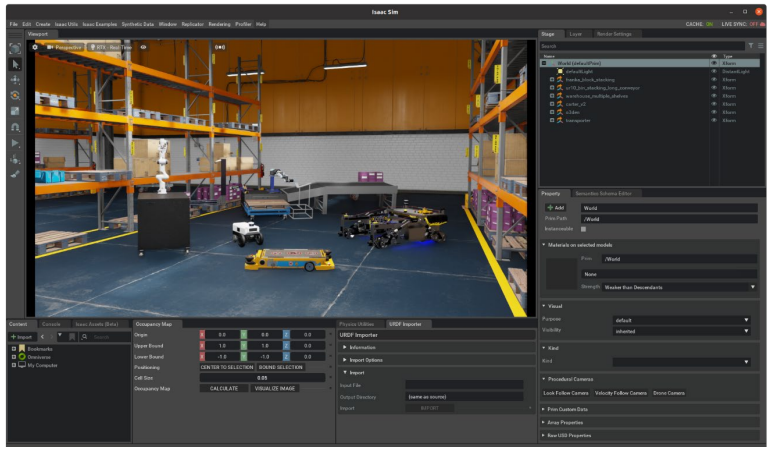
ROS Tools

- Simulators commonly used with ROS?
  - V-REP (CoppeliaSim): Versatile simulation not just robots but also sensors, mechanism, and even human-robot interactions with various scripting language.



CoppeliaSim
from the creators of V-REP

  - Isaac Sim: Ultra realistic simulation provides advanced tools for training and testing AI models, including AI for perception tasks.



NVIDIA
ISAAC

V-REP example image: https://www.researchgate.net/figure/Screen-shot-of-V-REPs-application-main-window-robot-models-are-courtesy-of-Lyall_fig6_220850179
Isaac Sim example image: https://docs.omniverse.nvidia.com/isaacsim/latest/index.html

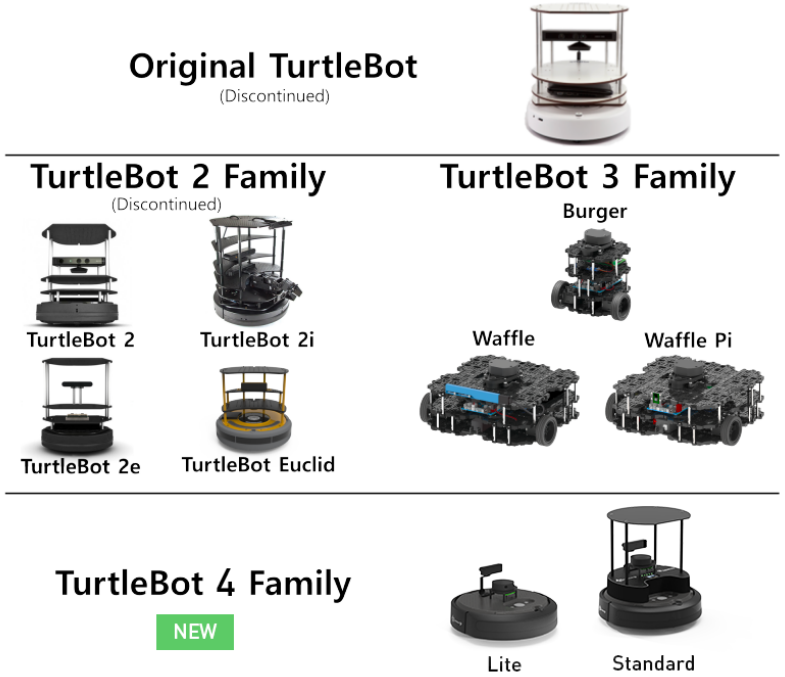
ROS Tools

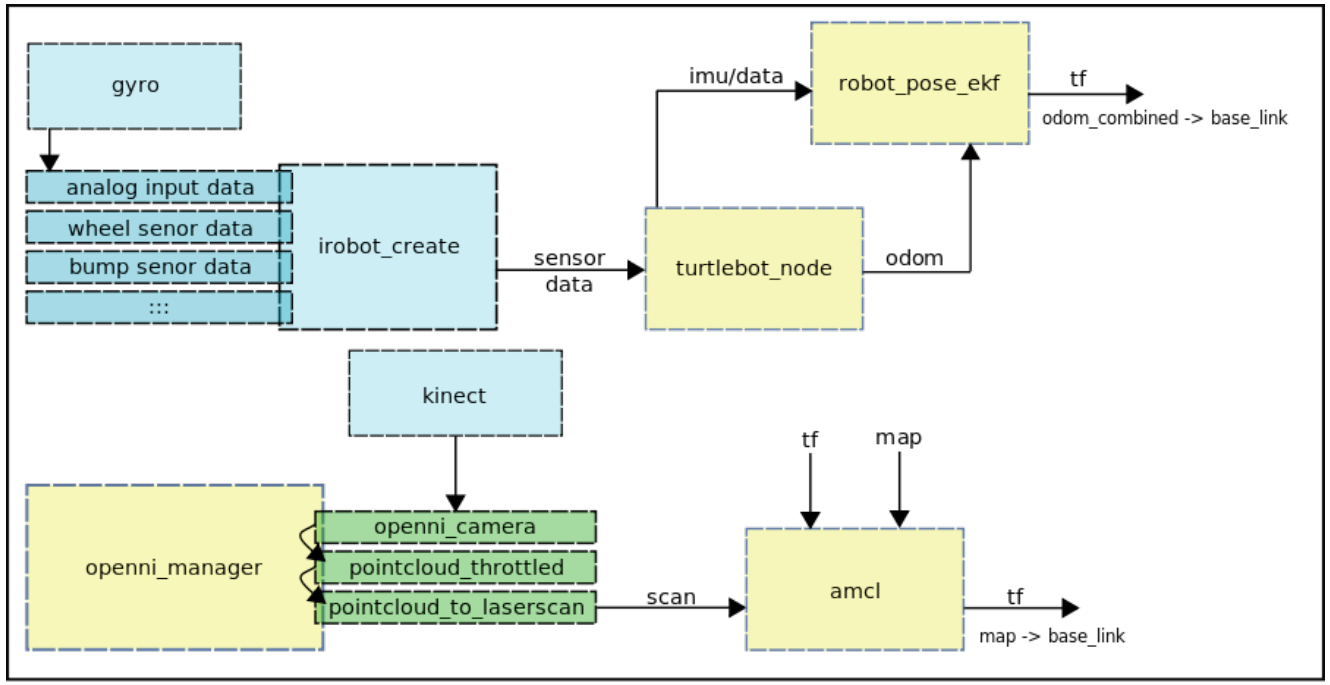
- Essential commands
  - roscore
    - Meaning: Launches the ROS Master, Parameter Server, and rosout logging node
    - Role: Provides core functionality for ROS system
    - Usage: Simply type `roscore` in terminal
  - Rosrun
    - Meaning: Runs a single ROS node
    - Role: Allows running nodes without a launch file
    - Example: `rosrun package_name node_name`
      - package_name: Name of the ROS package containing the node
      - node_name: Name of the node to run
  - roslaunch
    - Meaning: Launches multiple ROS nodes
    - Role: Starts nodes based on a launch file configuration
    - Example: `roslaunch package_name launch_file.launch`
      - package_name: Name of the ROS package containing the launch file
      - launch_file.launch: Name of the launch file to execute

Case Study: Robot System Integration with ROS

❖ TurtleBot Simulation

- ❑ Navigation stack integration:
  - ➢ SLAM for mapping, path planning
  - ➢ Sensor fusion: LiDAR, camera, IMU data integration
  - ➢ Gazebo simulation for algorithm testing
  - ➢ Widely used for educational purpose



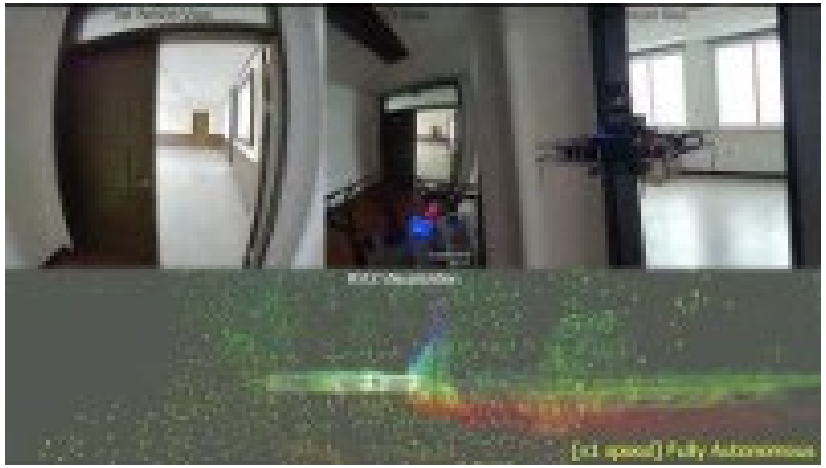


Case Study: Robot System Integration with ROS

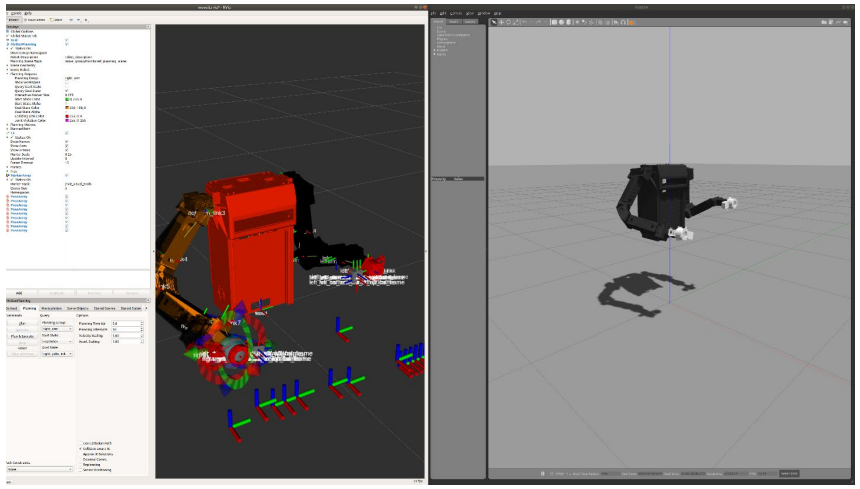
- Autonomous Vehicle
  - Perception system:
    - Uses ROS packages for object detection and tracking
    - Integrates data from cameras, LiDAR, and radar
  - Decision making and path planning:
    - Implements behavior trees for complex scenarios
    - Utilizes ROS navigation stack for route planning
  - Vehicle control:
    - Custom ROS messages for steering, acceleration, and braking

https://youtu.be/2JRb9TluuSE

https://youtu.be/vdEHporru9o



https://youtu.be/SXe_FJpxv94

Case Study: Robot System Integration with ROS

- Humanoid Robots
  - Whole-body control:
    - Uses ROS Control for coordinated joint movements
    - Implements inverse kinematics for complex motions
  - Vision and speech integration:
    - ROS nodes for face recognition and natural language processing
  - Knowledge integration:
    - Integrate Large Language Model node into task planning