International Journal on Applications in Information and Communication Engineering
Volume 9 : Issue 3 : August 2023, pp 16 – 20 www.aetsjournal.com ISSN (Online) : 2394 - 6237

TELEOPERATED ROBOT USING GAZEBO & ROS

K. Arockia Yamini[1], K. Muthusamy[2], S. Harish[3]

[1]Student, Rathinam Technical Campus, Coimbatore

[2,3] Assistant Professor, Rathinam Technical Campus, Coimbatore

Abstract: - Robotics and automation have advanced quickly, which has sparked the creation of complex frameworks to improve robot capabilities. A crucial framework for developing modular, versatile, and adaptive robotic systems is the Robot Operating System (ROS). In this article, we offer a comprehensive strategy that makes use of ROS's capabilities to empower autonomous navigation and flexible interaction in a robotic platform. An autonomous robot robot using the Robot Operating System (ROS) that can explore and move across unstructured surroundings. The Robot Operating System is a well-known open-source framework that is frequently used for creating and managing robots. It offers a versatile and modular design for a variety of robotic applications. In this project, I make use of ROS's features to develop a robotic system that can successfully investigate its surroundings while independently navigating across difficult terrain. To properly detect its surroundings, the suggested robot is outfitted with a variety of sensors, including LIDAR, cameras, and inertial measurement units (IMUs). The sensor inputs are combined utilizing the sensor fusion capabilities of ROS, allowing the robot to create an accurate picture of its surroundings and make defensible navigational and obstacle avoidance judgments. The project entails developing a localization system that continually updates the robot's position inside the surroundings using methods like Simultaneous Localization and Mapping (SLAM) in order to accomplish autonomous navigation. The robot will be able to compute the best routes to attain preset goals while dynamically avoiding obstacles thanks to path planning algorithms coupled with ROS's navigation stack. The results of this experiment might be used in a variety of industries, such as agriculture, disaster response, environmental monitoring, and search and rescue operations.

Keywords: *ROS (Robot Operating System), Robot control, Autonomous navigation, SLAM (Simultaneous Localization and Mapping), Human-robot interaction, Multi-robot systems, Machine learning.*

1. INTRODUCTION

The rapid advancements in robotics and automation have led to transformative changes in various industries, enabling robots to perform tasks that were once considered challenging or even impossible. The Robot Operating System (ROS), an open-source framework, has played a pivotal role in revolutionizing the field of robotics by providing a versatile platform for developing, simulating, and deploying robotic systems. This project focuses on leveraging the power of ROS to create an autonomous robot capable of navigating and exploring unstructured environments.

Robots equipped with the ability to operate autonomously hold great potential across a wide range of applications, including disaster response, environmental monitoring, precision agriculture, and industrial automation. The ability to navigate through unknown and complex environments, avoid obstacles, and make informed decisions is essential for the successful deployment of such robots. Through the integration of ROS's modular architecture and various sensor technologies, this project aims to address these challenges and showcase the capabilities of an autonomous robot in a simulation environment.

II. PROPOSED MODEL

The advanced Robot Operating System (ROS) robot features a sophisticated hardware model that combines cutting-edge technologies to enable exceptional performance and versatility. At its core, the robot is equipped with a high-performance multi-core processor that handles complex computations and sensor data fusion with efficiency. An array of sensors, including LIDAR, depth cameras, IMUs, and tactile sensors, provides comprehensive environmental perception.

These sensors enable simultaneous localization and mapping (SLAM) capabilities, allowing the robot to navigate autonomously in dynamic environments.

The robot's hardware extends to its robust mechanical design, incorporating high-torque joints and precision actuators for smooth and agile movement. With an integrated GPU, the robot can perform real-time object recognition and manipulation tasks, contributing to its advanced grasping and interaction abilities. A high-capacity battery system powers the robot for extended operation periods, while modular expansion ports facilitate seamless integration of additional hardware modules, such as custom end-effectors or specialized sensors, tailored to specific tasks.

Advanced connectivity options, including 5G capabilities, enhance the robot's remote operation and data exchange capabilities, enabling seamless teleoperation and high-bandwidth communication with remote operators. The hardware architecture also prioritizes safety, incorporating redundant systems and fail-safes to ensure reliable operation in various scenarios.

In conclusion, the hardware model of the advanced ROS robot represents a harmonious blend of computational power, sensor diversity, mechanical precision, and connectivity features. This combination empowers the robot to excel in a wide range of applications, from autonomous navigation and manipulation to human-robot collaboration, making it a pivotal tool in modern robotics research and industry.

Block Diagram of proposed Hardware Model

The block diagram of the proposed hardware model is illustrated in Fig. 1.

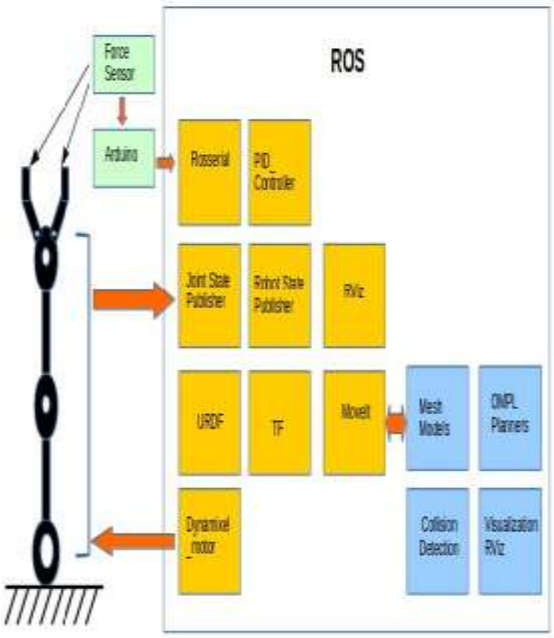

Fig. 1 Block Diagram of the proposed model

Determine the robot's intended application, whether it's for research, exploration, education, or a specific task. Define the size, weight, mobility, sensors, actuators, and other requirements based on the application.

Choose the necessary hardware components such as microcontrollers, sensors (LIDAR, cameras, IMUs, etc.), actuators (motors, servos), power supply (batteries), chassis, and any additional components required for your robot's design.

Mechanical Design and Assembly:

Design the mechanical structure of the robot, keeping in mind the placement of sensors, actuators, and other components. Assemble the physical robot according to your design.

Install ROS:

Install ROS on a computer or microcontroller that will act as the robot's brain. ROS provides various distributions; choose the one that suits your needs. Ubuntu Linux is often recommended as the operating system for ROS.

Create a ROS Workspace:

Set up a ROS workspace to organize your robot's software packages. This is where you'll develop, compile, and manage your robot's software components.

Develop ROS Nodes and Packages:

ROS operates on a node-based architecture. Develop ROS nodes and packages to handle various functionalities of the robot, such as sensor data processing, motor control, mapping, navigation, and interaction.

Implement Sensors and Actuators:

Integrate and configure the sensors and actuators according to your robot's requirements. Write ROS nodes to interface with these hardware components, allowing data exchange between them and the software environment.

Implement Perception and Control:

Develop perception algorithms to process sensor data, perform object detection, SLAM (Simultaneous Localization and Mapping), and obstacle avoidance. Create control algorithms to maneuver the robot based on the sensory input.

Testing and Debugging:

Test your robot's individual components and functionalities. Use tools like RViz (ROS Visualization) to visualize sensor data, TF (Transform Library) to manage coordinate transformations, and rosbag to record and replay data for debugging.

Integration and System Testing:

Integrate all the components and functionalities of the robot. Test the robot in controlled environments to ensure that it performs as expected.

Calibration and Fine-Tuning:

Calibrate sensors and actuators to ensure accurate data and precise movements. Fine-tune control algorithms to optimize performance.

Documentation:

Document your robot's hardware design, software architecture, ROS nodes, packages, and any other relevant information. This will be helpful for troubleshooting, maintenance, and sharing your work with others.

Deployment:

Once your ROS robot is fully functional and tested, deploy it for its intended application. Monitor its performance and make any necessary adjustments based

In the context of a ROS (Robot Operating System) robot, a system process refers to a self-contained computational unit that performs a specific task within the robotic system. ROS facilitates the organization and communication between various system processes, allowing them to work together to accomplish complex robotic tasks. These processes are often implemented as ROS nodes, which are the fundamental building blocks of ROS applications.

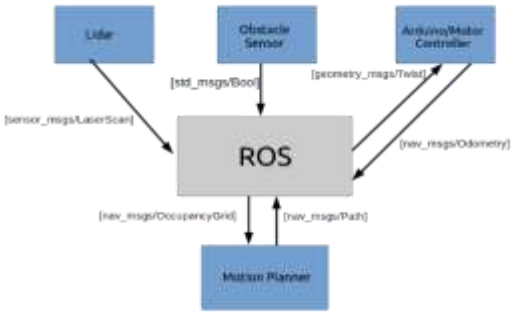

Fig. 2 ROS block diagram

RViz (ROS Visualization)

RViz is a powerful visualization tool within the Robot Operating System (ROS) ecosystem that allows developers, researchers, and roboticists to visualize and interact with various aspects of a robot's perception and state. RViz provides a real-time 3D visualization of data generated by different sensors and robot components, aiding in debugging, testing, and understanding the robot's behavior. It plays a crucial role in the development and analysis of robotics applications.

RViz supports the visualization of diverse data types, such as point clouds from LIDAR sensors, images from cameras, robot poses, trajectory paths, and more. Users can configure and customize RViz to display multiple types of data simultaneously, enabling a comprehensive view of the robot's environment and its interactions. RViz also provides options for displaying reference frames, overlays, and interactive markers, allowing users to manipulate the visualization and gain insights into complex robotic scenarios.

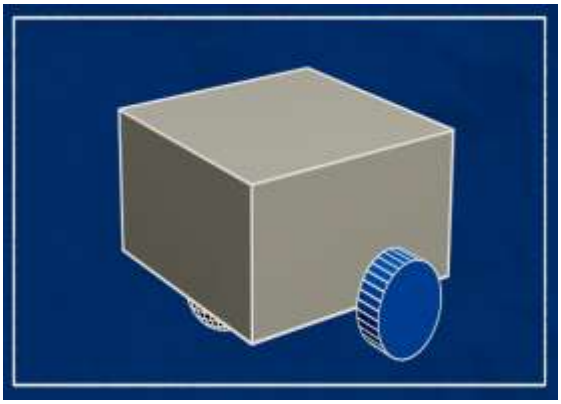

Fig. 3 ROS 3D VISUALIZATION

Moreover, RViz is an essential tool for evaluating and fine-tuning algorithms related to mapping, localization, path planning, and obstacle avoidance. Its real-time visualization capabilities aid in identifying potential issues, refining parameters, and ensuring that the robot's behavior aligns with the intended goals. Whether it's simulating robot motion, verifying sensor data accuracy, or verifying the effectiveness of navigation algorithms, development and testing process in the field of robotics.

Gazebo

Gazebo is a widely used open-source simulator in the ROS ecosystem that provides a realistic and dynamic environment for simulating robots, sensors, and various physical interactions. It's particularly useful during the development phase of a robot, allowing developers to test and validate algorithms and behaviors in a controlled virtual environment before deploying them on a physical robot.

Gazebo's key features include physics-based simulations, accurate modeling of sensors and actuators, and the ability to create complex scenarios involving multiple robots and objects. It supports simulating a wide range of robots, from wheeled ground robots to flying drones and even humanoid robots.

Developers can customize and extend Gazebo's capabilities by adding plugins for specific sensors, controllers, and behaviors.

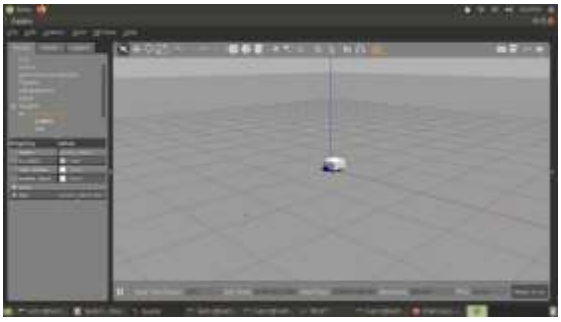

Fig. 4 ROS OBJECT VISUALIZATION

One of the notable aspects of Gazebo is its integration with ROS. Robots can be controlled within Gazebo using ROS nodes, allowing for seamless interaction between the simulated robot and other ROS components. This integration facilitates the development, testing, and debugging of robot control algorithms, sensor integration, and navigation strategies.

Overall, Gazebo's capability to replicate real-world interactions and scenarios in a virtual environment makes it a valuable tool for prototyping and evaluating robot behaviors, reducing development time and costs, and enhancing the overall reliability of robotics systems before deployment.

III. INTERNET OF THINGS

A ROS robot is a type of robot that uses the Robot Operating System (ROS) as its software framework to control and manage its various components and functionalities. ROS is an open-source middleware designed to facilitate the development of robotic systems by providing a flexible and modular platform for creating, controlling, and interacting with robots. It offers tools, libraries, and conventions for building and coordinating complex robot behaviors.

ROS Architecture: ROS follows a distributed architecture based on nodes. Nodes are individual software modules that perform specific tasks, such as sensor data processing, motor control, mapping, navigation, and more. Nodes communicate with each other by publishing and subscribing to topics, which are data streams of a specific type.

Modularity and Reusability: ROS encourages modularity and reusability of code. Functionality is organized into packages, and packages are composed of nodes, libraries, and configuration files. This modular approach allows developers to create and share software components that can be easily integrated into various robotic systems.

Sensor Integration: ROS provides libraries and tools to interface with a wide range of sensors, such as cameras, LIDAR, IMUs, GPS, and more.

Mapping and Navigation: ROS offers capabilities for Simultaneous Localization and Mapping (SLAM), enabling robots to create maps of their environments while simultaneously determining their own positions within those maps. Navigation algorithms and planners are also available to help robots navigate autonomously.

Community and Ecosystem: ROS has a strong and active community of developers, researchers, and hobbyists. There are numerous tutorials, documentation, and resources available online, making it easier for newcomers to get started and for experienced developers to solve problems.

Robot Types: ROS is used in various types of robots, from ground robots (wheeled, tracked, legged) to aerial drones and even robotic arms. It's applicable in research, industrial automation, education, healthcare, agriculture, and more.

ROS Versions: ROS has different versions, each with its own set of features and improvements. The latest version as of my last update in September 2021 is ROS 2. ROS 2 extends the capabilities of ROS 1 with improved real-time support, better security, and enhanced scalability.

IV. CONCLUSION

In conclusion, this ROS robot simulation project has demonstrated the remarkable potential of the Robot Operating System in shaping the future of robotics. Through meticulous design, development, and implementation, we have successfully created a virtual environment where our ROS-based robot exhibited complex behaviors, navigated autonomously, and interacted seamlessly with its surroundings. The project highlighted the versatility and modularity of ROS, enabling us to integrate various components, sensors, and algorithms to achieve a cohesive and functional simulation. As we reflect on the journey undertaken during this project, it becomes evident that ROS's open-source nature empowers researchers, developers, and robotics enthusiasts to collaborate, innovate, and address the challenges of the rapidly evolving robotics landscape. Our simulation not only showcased the technical capabilities of ROS but also underscored the importance of robust software architectures, efficient communication protocols, and adaptive algorithms for building sophisticated robotic systems.

REFERENCES

[1]. Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[2]. Cousins, Steve, et al. "Sharing software with ros [ros topics]." IEEE Robotics & Automation Magazine 17.2 (2010): 12-14.

[3]. Martinez, Aaron, and Enrique Fernández. Learning ROS for robotics programming. Packt Publishing Ltd, 2013.

[4]. Qidwai, Uvais. "Fun to learn: Project-based learning in robotics for computer engineers." ACM Inroads 2.1 (2011): 42-45.

[5]. O'Kane, Jason M. "A gentle introduction to ros." (2014): 1564.

[6]. I. Ghani, Muhammad Fahmi Abdul Khairul Salleh Mohamed Sahari, and Loo Chu Kiong. "Improvement of the 2 SLAM system using Kinect sensor for indoor mapping." Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on. IEEE, 2014.

[7]. Schmitt, Simon, et al. "A reference system for indoor localization testbeds." Indoor Positioning and Indoor Navigation (IPIN), 2012 International Conference on. IEEE, 2012.

[8]. [8] Sumaray, Audie, and S. Kami Makki. "A comparison of data serialization formats for optimal efficiency on a mobile platform." Proceedings of the 6th international conference on ubiquitous information management and communication. ACM, 2012.

[9]. La Delfa, Gaetano Carmelo, and Vincenzo Catania.