Lab 1 – RIDE Product Description

Daniel Koontz

Old Dominion University

CS 410, Fall 2022

Profesor J. Brunelle

December 14, 2022

First Draft

Table of Contents

List of Figures

1. Introduction

Automation and robotics are increasingly becoming the new norm in many facets of life. Semi-autonomous vehicles are on roads and highways, robotic arms appear in factories and warehouses, and vacuum cleaners can clean without human intervention. Robots and autonomous machines are here to stay within society.

As an industry, robots and autonomous machines are expected to grow substantially. The mobile robots market is expected to grow at 24% per year from $19 billion in 2018 to $23 billion by 2021 and further to $54 billion in 2023 (Cardona et al., 2020).

Robot Operating System (ROS) is an open-source set of tools and software libraries to help build robotics applications. ROS is made up of topics and messages and facilitates communication between components. These topics and messages can be recorded into a ROSbag file that can be played back or logged for future use. Messages can also be sent to visualization tools and can allow developers to work with an entirely simulated robot. ROS supports common components such as cameras, LIDAR, and motor controllers. These components are identified within ROS as nodes and can be visually displayed as a node graph. Most applications require many nodes.

ROS has an estimated market value of $270 million as of 2020 (*Market Reports*, 2022), with projected growth to $280 million in 2028 and $460 million in 2032 (*Future Market Insights*, 2022). ROS has been utilized in industries such as autonomous vehicles, aerospace, healthcare, and agriculture. ROS has been utilized by organizations such as NASA ("Lunar Rover") (Wessling, 2022), Microsoft (Ackerman, 2021), and Sony (Fujita, 2018).

While ROS has seen extensive use, development with ROS remains a challenge for many. In particular, the user interface and experience for ROS can be a significant hurdle for newcomers. Since most ROS projects involve many ROS nodes and each node is controlled through a separate terminal, this can quickly clutter screens, and can be difficult to differentiate one process from another. Similarly, it can be difficult to identify when nodes interact with each other. When users search for documentation to solve these issues, they come upon articles that can be over 10 years old, along with unhelpful descriptions and sometimes even missing pages entirely. All of this contributes to the difficulties in building and maintaining a ROS project.

RIDE is an extension pack designed to improve the user experience when developing ROS applications. Hosted on the VSCode extension marketplace, RIDE provides tools and functionalities that simplify common tasks and commands, enable visualization tools, and organize the user's workspace. By taking advantage of RIDE, developers can quickly build simple projects and have access to a variety of template files to implement. RIDE users can easily navigate source code and data to build, fix, and test ROS projects.

2. Product Description

RIDE is an extension pack for VScode designed to simplify and speed up the development lifecycle and learning process for building ROS applications. It utilizes wizards, templates, IntelliSense, and snippets to help developers build products faster while including visualization features and ROSBag interfaces to help test and validate their systems.

2.1. Key Product and Features

The key features of RIDE are its simplified user interface and quick setup, along with the addition of visualization features embedded directly into the IDE. The primary tools involve code manipulation and completion. RIDE uses wizards to create new files and update the relevant compilation and package files. RIDE also takes advantage of tools natively in VSCode to allow users to autocomplete common code patterns. Most common tasks and actions are simplified to a button click or menu. This simplified user interface reduces the need for overreliance on outdated documentation.

Visualization tools allow developers to manage and conceptualize the reality of what the components of their project currently accomplish and can hint towards improvements. Developers can publish messages to topics to test how their system behaves on a set of known data points and build models and algorithms to handle any case.

2.2. Major Components (Hardware / Software)

RIDE consists mainly of the developer environment within the VSCode IDE. It is comprised of multiple modular extensions packaged together into a single extension pack. RIDE takes advantage of the tools that exist in VSCode, such as Microsoft's ROS extension to provide features such as syntax highlighting and commands to control ROS software inside the IDE.

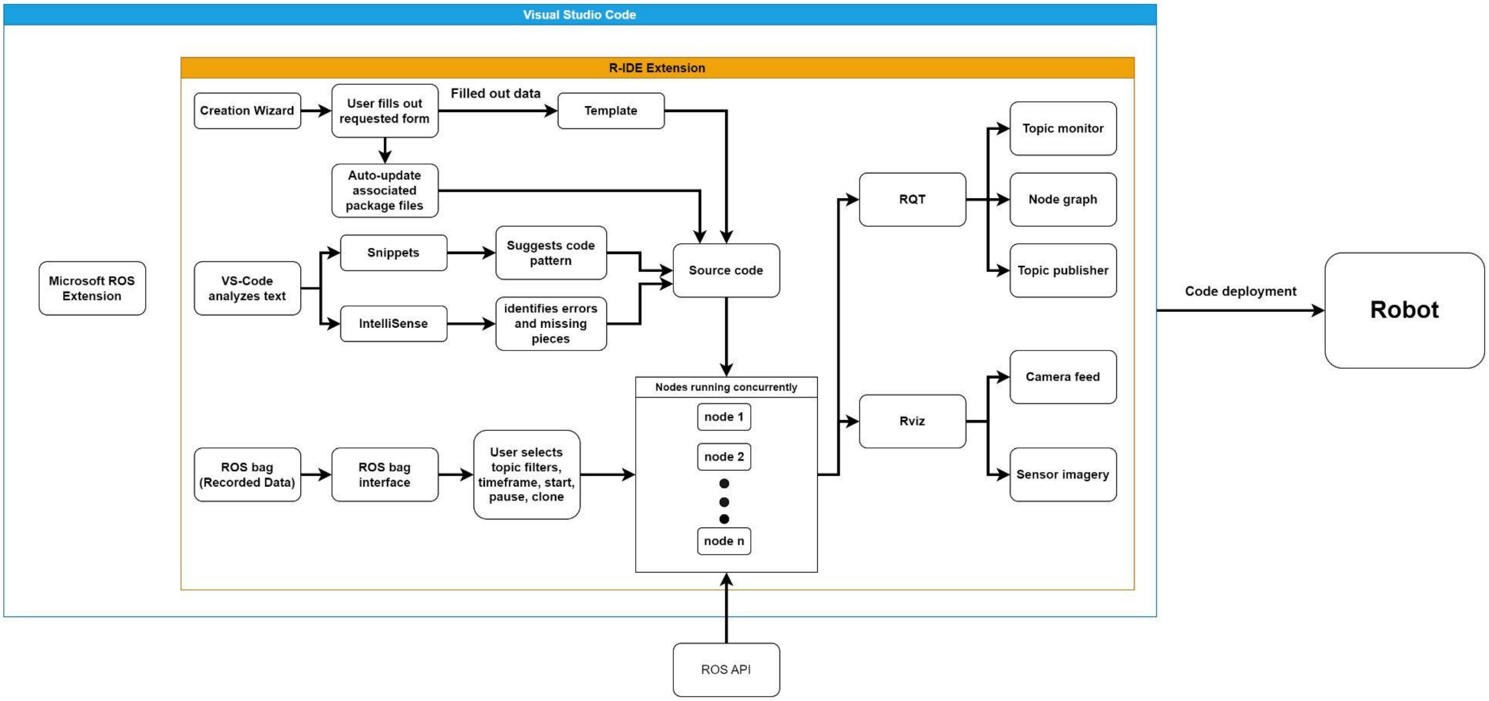
While working with code, developers have access to a variety of tools to speed up development and encourage clean readable, testable code. Users are able to access the various creation wizards to build components of their project, such as creating ROS nodes, topics, and messages. In addition, these wizards make sure to keep the necessary compilation and package files updated as the user makes changes to their source code. While the user is typing, VSCode allows RIDE to make suggestions to build common code patterns and can implement them for the user.

RIDE allows users to interact with ROSBags through a separate interface. Users can record, playback, clone, and filter ROSBags entirely with the interface. This allows users to have powerful control over ROSbags without leaving the source code and data stored within the IDE.

A variety of visualization tools inspired by what GUI tools exist are also available to the user. Users can view and publish to ROS topics while the ROS master is running while also viewing what nodes are active and connected nodes in the graph view. The major components of RIDE according to developers are shown in Figure 1.

Figure 1.

Major Functional Component Diagram



3. Identification of Case Study

The main users of RIDE are ROS developers. ROS developers are often members of the robotics industry, universities, and hobbyists. RIDE will provide an intuitive interface for ROS development along with the tools to enable new and existing developers.

The purpose of RIDE is to improve the user experience when developing and interacting with ROS. A case study is being performed at Old Dominion University with members of the Electrical and Computer Engineering department who are currently learning and building with ROS under the direction of Dr. Lee Belfore. Students and professors will use RIDE within VSCode as their primary development environment. Users will be encouraged to use the templates and wizards in RIDE to build initial ROS nodes, services, and messages. The developers and maintainers of RIDE can access usage data so that fixes can be made as quickly as possible and such that it can ascertain how often different services are used and make an estimation of the approximate time to complete a wizard. More data will be collected by providing surveys to the member of the case study and analyzing the time it takes for members to build simple applications.

As more developers migrate to use RIDE, feedback will improve and expand the capabilities of RIDE. New services and synergies can be identified that improve upon the development process.

4. RIDE Product Prototype Description

Placeholder for future use

4.1. Prototype Architecture (Hardware/Software)

Placeholder for future use

4.2. Prototype Features and Capabilities

Placeholder for future use

4.3. Prototype Development Challenges

Placeholder for future use

5. Glossary

Autonomous Machine: A machine capable of sensing its environment, carrying out computations to make decisions, and performing actions in the real world.

Robot Operating System (ROS): ROS is a set of software libraries that helps to build robot applications. Ranging from drivers to algorithms, and powerful developer tools, ROS is the preferred tool for robotics projects.

ROS Bag: A bag is a file format in ROS for storing ROS message data. These bags have an important role in ROS, and a variety of tools have been written to allow you to store, process, analyze, and visualize them. Bags are the primary mechanism in ROS for data logging, which means that they have a variety of offline uses.

ROS Master: The ROS Master provides naming and registration services to the rest of the nodes in the ROS system. It tracks publishers and subscribers to topics as well as services. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer. The Master also provides the Parameter Server and is most commonly run using the roscore command, which loads the ROS Master along with other essential components.

ROS Messages: Nodes communicate with each other by publishing messages to topics. A message is a simple data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays. Nodes can also exchange a request and response message as part of a ROS service call.

ROS Node: A node is a process that performs computation. Nodes are combined together and communicate with one another using streaming topics, RPC services, and the Parameter Server. These nodes are meant to operate at a fine-grained scale; a robot control system will usually comprise many nodes. For example, one node controls a laser range-finder, one Node controls the robot's wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on.

ROS Parameter Server: A parameter server is a shared, multi-variate dictionary that is accessible via network APIs. Nodes use this server to store and retrieve parameters at runtime. As it is not designed for high performance, it is best used for static, non-binary data such as configuration parameters. It is meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify if necessary.

ROS Services: Request/reply is done via a Service, which is defined by a pair of messages: one for the request and one for the reply. A providing ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply. Client libraries usually present this interaction to the programmer as if it were a remote procedure call.

ROS Topics: Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with. Instead, nodes that are interested in data subscribe to the relevant topic; nodes that generate data publish to the relevant topic. There can be multiple publishers and subscribers to a topic. Topics are intended for unidirectional, streaming communication. Nodes that need to perform remote procedure calls, i.e. receive a response to a request, should use services instead. There is also the Parameter Server for maintaining small amounts of state.

roswtf: A tool for diagnosing issues with a running ROS system.

rqt: A QT-based framework for GUI development for ROS. It contains tools that support ROS topics, bags, node graphs, and many other tools for visualization and manipulation of ROS nodes

Rviz (Ros Visualization): A 3D visualizer for displaying sensor data and state information from ROS

Template: An editable text or code snippet that can be filled with given values from another tool like a wizard.

Tutorial: A method of transferring knowledge that teach via example and supplies the information to complete a given task.

VSCode Extension: A tool designed to add additional features and capabilities to VSCode. It can create new dialogues, add functions, or change the appearance of VSCode

Wizard: A user interface that presents dialog to lead a user through a sequence of steps. Often used to configure a service for the first time or to simplify a complex or unfamiliar process.

6. References

Ackerman, E. (2021, June 24). Microsoft announces experimental release of ROS for windows 10. Retrieved November 3, 2022, from https://spectrum.ieee.org/microsoft-announces-experimental-release-of-ros-for-windows-10

Belfore, L. (Jul 8, 2022). Software Design Report for the ODU Monarch I (Vol. 1.3, p. 3, Tech.).

Cardona, M., & Manzanares, J. (2020). COVID-19 Pandemic Impact on Mobile Robotics Market (pp. 1-4) (A. Palma, Ed.). *IEEE*. doi:10.1109/ANDESCON50619.2020.9272052

Fujita, T. (2018). Tomoya Fujita R&D center Sony Corporation - roscon.ros.org. Retrieved November 3, 2022, from https://roscon.ros.org/2018/presentations/ROSCon2018_Aibo.pdf

Gerkey, B. (2014, September 1). Ros running on ISS. Retrieved November 3, 2022, from https://www.ros.org/news/2014/09/ros-running-on-iss.html

Global Robot Operating System Market Research Report 2022(status and outlook). (2022, June 29). Retrieved November 3, 2022, from https://www.marketreportsworld.com/global-robot-operating-system-market-21185690

Guerry, M., Müller, C., Kraus, W., & Bieller, S. (2021, October 28). *IFR International Federation of Robotics*. Retrieved November 3, 2022, from https://ifr.org/downloads/press2018/2021_10_28_WR_PK_Presentation_long_version.pdf

Lunar Rover. (2021). Retrieved November 3, 2022, from https://www.openrobotics.org/customer-stories/lunar-rover

Robot Operating System Market. (2022, August). Retrieved November 3, 2022, from https://www.futuremarketinsights.com/reports/robot-operating-system-market

Robot operating system. (n.d.). Retrieved November 3, 2022, from https://www.ros.org/

Wessling, B. (2022, February 11). Open robotics developing space ROS with Blue Origin, NASA. Retrieved November 3, 2022, from https://www.therobotreport.com/open-robotics-developing-space-ros/