Lab 1 – R-IDE Product Description

Gavin St. Clair

Old Dominion University

CS 411 Spring 2023

Professor J. Brunelle

January 30, 2023

Version 1

Table of Contents

List of Figures

List of Tables

1 Introduction

Robots and autonomous machines are becoming more common and widespread as time progresses. They are adopting many routine tasks across various industries. Reports suggest that the mobile robot market is projected to grow at a rate of 24% per year. The market value was estimated at $19 billion in 2018 and is expected to reach $54 billion by 2023 (COVID-19 Pandemic Impact on Mobile Robotics Market). Data published by the International Federation of Robotics suggests that the annual number of industrial robots will also increase by around 15% by the year 2024 (IFR International Federation of Robotics). These would include manufacturing or assembly line robots.

Robot Operating System (ROS) is an open-source set of software libraries and tools that help developers build robot applications (Robot Operating System). ROS is popular and widely used. It has a current market value estimated at $270 million, and projected to reach $380 million by 2028, and 460 million by 2032 (Global Robot Operating System Market Research Report 2022 (Status and Outlook)). ROS is utilized by many industries ranging from autonomous vehicles to healthcare, and even agriculture. There are some major companies that use ROS based systems such as The National Aeronautics and Space Administration in their Mars Curiosity Rover, or in their humanoid robot on the international space station.

ROS workflows leverage several non-attributional windows that elevate the difficulty of debugging, monitoring & understanding. Many parts of the documentation are not up to date and require significant technical knowledge to understand and begin ROS development. Environment setup can be difficult in any project, but especially in one that can use multiple versions and distributed dependencies. ROS development contains high barriers of entry for new developers and environments. This is where R-IDE provides assistance with an extension pack to simplify and speed up the development lifecycle and learning process.

2 R-IDE Product Description

R-IDE is an extension pack to simplify and speed up the development lifecycle and learning process for ROS. The focus of R-IDE is on simplifying the process and creating systems that encourage best practices for a developer. R-IDE utilizes an extension pack to develop modular systems so that developers can make selections for programming language and ROS version.

2.1 Key Product Features and Capabilities

The solution involves developers working inside VSCode, with R-IDE as a product extension. Users interact with one of the R-IDE code generation tools to quickly build common patterns and quickly generate new files. A user can feed data into their ROS environment from a ROSbag where they can select options from our interface on filtering data or selecting a timestamp. 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. Nodes are a computer file in a tree data structure. While the nodes and functions are running, they can be monitored with data visualization and manipulation tools such as RQT and ROS visualization. RQT is a software framework of ROS that implements the various GUI tools in the form of plugins. When a developer is ready, they can deploy their code in whatever way is best suited for them to their robot where they can record data into a ROSbag to feed back into RIDE at a later point in time.

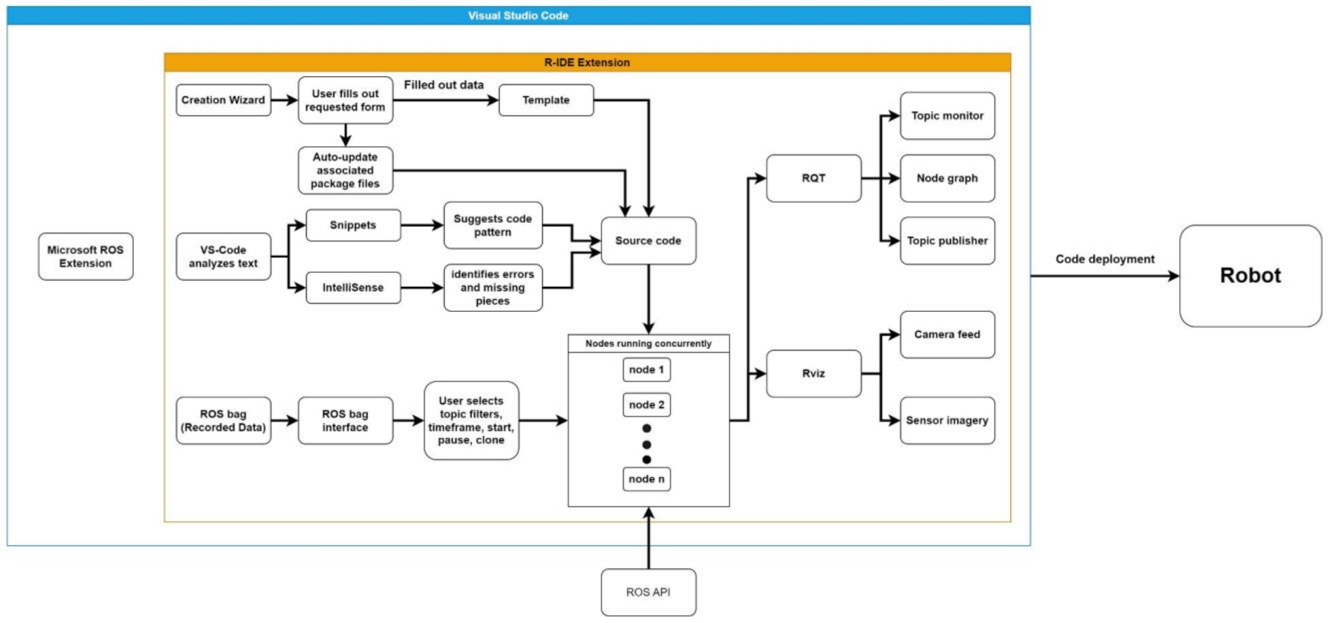
2.2 Major Components (Hardware/Software)

The major functional components for R-IDE consist mainly of the developer environment inside the VSCode IDE. R-IDE consists of multiple plugins or extensions packaged together. It relies on ROS, but the Microsoft ROS Extension is also included. This extension provides functionality to start and stop the ROScore, syntax highlighting, and automatic import and include paths for ROS.

When inside RIDE and working with a project’s source code, several tools and components are available to assist developers in writing clean working code. This consists of tools like a creation wizard for the initial code, boilerplate of building a new node or component of a project, and snippets which make simple prediction about the code and identify next steps and corrections. After compilation and launching the nodes, R-IDE can feed data into the functions that makes up ROS from a ROSbag, where the interface allows users to interact with the ROSbag like any other recorded, timestamped media. In addition, it allows users to filter out topics and clone the repository as it exists. As data leaves the nodes a developer can visualize and manipulate the data. Users can manipulate the topics, either publishing data or subscribing to the feed, monitor the node graph of their project, or see a feed from a camera or other visual data feed. When the developer decides to publish their software contribution it will get fed into their project, which is outside of the scope of R-IDE.

Figure 1

R-IDE Major Functional Components Diagram



3 Identification of Case Study

ODU's autonomous vehicles utilize ROS to convert raw sensor data, perform calculations, and make intelligent decisions on the vehicles upcoming route. With the combined efforts of the following departments and their respective mentors, ODU hopes to place in the top 6 for the Self-drive challenge at the Intelligent Ground Vehicle Competition (IGVC). R-IDE uses Dr. Belfore's autonomous vehicle as an example ROS project, R-IDE is designed to be used primarily by Dr. Belfore and his students, but the goal is to create a product that can be used by any ROS developer.

The intended use of R-IDE is to provide an intuitive yet powerful development environment for ROS based applications. The intended user for R-IDE is Dr. Belfore and his students. R-IDE may also be used by other universities, robotics professionals or hobbyists. R-IDE provides tools that reduce initial overhead and enable new ROS developers. R-IDE will be available for download in the Virtual Studio Code extension marketplace.

4. Product Prototype Description

The final product for R-IDE will be a full extension for VS-code not a prototype. The focus of R-IDE is on simplifying the process and creating systems that encourage best practices for a developer. R-IDE utilizes an extension pack to develop modular systems so that developers can make selections for programming language and ROS version. The prototype largely relies on customer feedback from weekly dialogues with the initial customer, Dr. Belfore as well as users inside VS-code.

4.1. Prototype Architecture (Hardware/Software)

The hardware requirements for R-IDE consist of a computer that meets inherited hardware requirements to install VS code, ROS, and ROS visualization. The software requirements for R-IDE consist of installing ROS 1 or ROS 2 and VS-Code. ROS 1 requires a Linux environment (can use WSL or virtual machine). ROS 2 can run on Linux, Mac, or Windows.

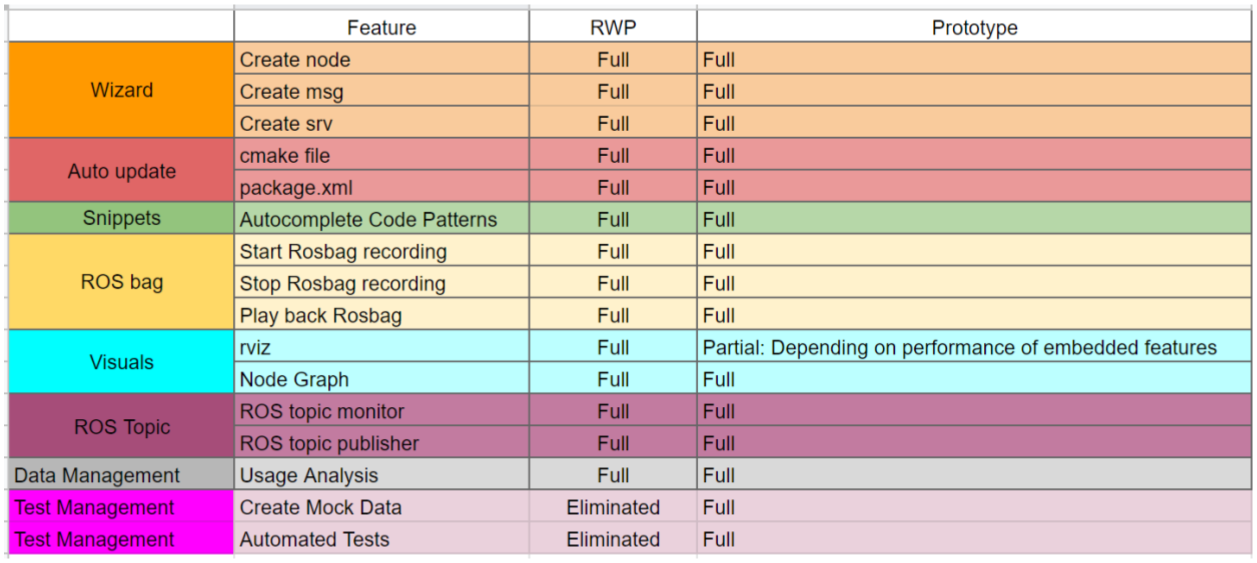
R-IDE will be a full extension thus the MFCD and the prototype MFCD are the same. The R-IDE extension exists within VS-Code alongside a complementary extension

(Microsoft ROS ext.). The creation wizard requests that the user fills out a form to generate code templates for common ROS components. VS-Code suggests and fixes code using customized snippets and IntelliSense. The ROSbag interface manipulates and manages ROSbags. Output data is displayed through various visualization tools that are embedded within VS-code. The code is then deployed to a robot (This is not handled by R-IDE).

4.2. Prototype Features and Capabilities

Table 1

R-IDE RWP v. Prototype Features



	Feature	RWP	Prototype
Wizard	Create node	Full	Full
	Create msg	Full	Full
	Create srv	Full	Full
Auto update	cmake file	Full	Full
	package.xml	Full	Full
Snippets	Autocomplete Code Patterns	Full	Full
ROS bag	Start Rosbag recording	Full	Full
	Stop Rosbag recording	Full	Full
	Play back Rosbag	Full	Full
Visuals	rviz	Full	Partial: Depending on performance of embedded features
	Node Graph	Full	Full
ROS Topic	ROS topic monitor	Full	Full
	ROS topic publisher	Full	Full
Data Management	Usage Analysis	Full	Full
Test Management	Create Mock Data	Eliminated	Full
Test Management	Automated Tests	Eliminated	Full

The R-IDE real world prototype is almost identical to the prototype. Automated tests and mock data will only be in the prototype not the RWP. Other features of the prototype will include the creation wizard for nodes, msg, and srv. The ability to automatically update cmake files and package .xml. R-IDE will have code snippets that autocomplete code patterns. A ROSbag will allow a user to start, stop, and play back ROSbag recording. Visual tools for users including ROS visualization, RQT, and a node graph. A ROS topic monitor for publishing and debugging will not be in the RWP because it is not needed.

4.3. Prototype Development Challenges

R-IDE will have development challenges when creating a prototype. One of the first challenges in development will be publishing the extension package to the VS-Code marketplace. More of VS-Code related development challenges will be getting cursor and line position across multiple terminals and highlighting words across multiple terminals. The next development challenge will be how to make R-IDE compatible for both ROS 1 and ROS 2. There is a steep learning curve for ROS development and the documentation related to ROS is either missing or outdated.

5. Glossary

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 Node: A node is a process that performs computation. Nodes are combined 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 rangefinder, 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 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 run using the ROScore command, which loads the ROS Master along with other essential components.

ROS Parameter Server: A parameter server is a shared, multivariate 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 it if necessary.

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 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.

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

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/