Lab 1 – R-IDE Product Description

Gavin St. Clair

Department of Computer Science Old Dominion University

CS 411 Professional Workforce Development Spring 2023

Professor J. Brunelle

March 13, 2023

Final Version

Table of Contents

List of Figures

List of Tables

1 Introduction

The proliferation and ubiquity of robots and autonomous machines are progressively increasing over time. They are adopting many routine tasks across various industries. According to technical reports, the mobile robot market is anticipated to exhibit a compound annual growth rate of 24% with the market size estimated at $19 billion in 2018 and forecasted to escalate to $54 billion by the end of 2023, considering the impact of the COVID-19 pandemic (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 is 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. Several prominent enterprises have adopted ROS-based systems, including The National Aeronautics and Space Administration, which employs such technology in its Mars Curiosity Rover and humanoid robot deployed on the International Space Station.

ROS workflows incorporate various non-attributional windows that exacerbate the challenges associated with debugging, monitoring, and comprehending the system. Additionally, several sections of the documentation are not current and necessitate a considerable degree of technical proficiency to grasp and initiate ROS development. The setup of the environment can pose a formidable challenge in any project, particularly in one that employs distributed dependencies and multiple versions of ROS. As such, ROS development presents significant hurdles for new developers and environments. This is where R-IDE offers support with an extension pack that streamlines and accelerates the development lifecycle and learning curve.

2 R-IDE Product Description

R-IDE is a software extension pack that streamlines and accelerates the development lifecycle and learning process for ROS. The primary objective of R-IDE is to simplify the development process and foster the adoption of best practices for developers. To achieve this goal, R-IDE employs an extension pack that facilitates the development of modular systems, allowing developers to choose the programming language and ROS version that best suits their needs.

2.1 Key Product Features and Capabilities

The proposed solution entails the utilization of Visual Studio Code (VSCode) as the primary development environment, augmented with the integration of R-IDE as a supplementary product extension. End-users are facilitated with a set of specialized code generation tools offered by R-IDE, which enables the rapid construction of commonly occurring design patterns and swift file generation. Further, data ingestion into the ROS environment is facilitated using a ROSbag, where the user interface permits flexible filtering and timestamp selection options. The bag file format in ROS is the standard mechanism for storing ROS message data, offering a multitude of tools to process, analyze, and visualize the data offline. Nodes, on the other hand, form a tree data structure within the system and can be monitored during runtime through 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. Once a developer has finalized their code, it can be deployed to the robot in a manner best suited for their use case, where the resultant data can be logged into a ROSbag and fed back into R-IDE at a later point for further analysis and refinement.

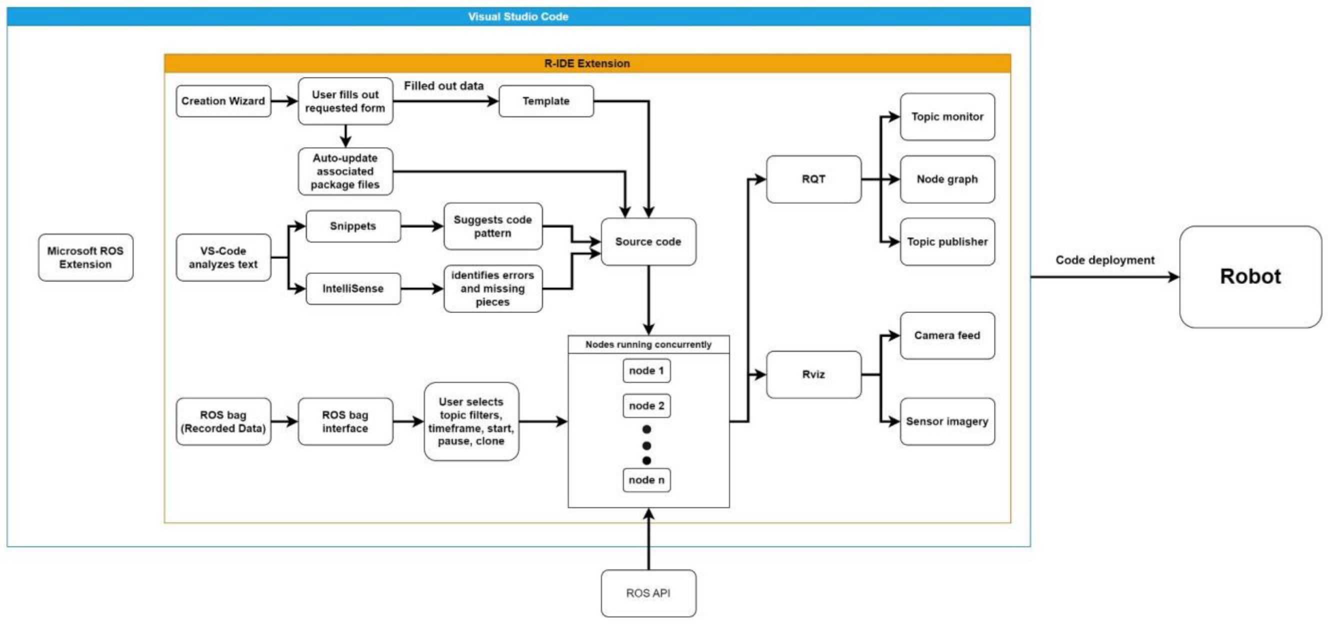
2.2 Major Components (Hardware/Software)

R-IDE, as a multifaceted development tool, encompasses several critical components that operate within the VSCode IDE environment. The tool is built around a suite of plugins or extensions, working in concert to enhance developer efficiency and efficacy. Figure one represents the visual depiction of the R-IDE Major Functional Components Diagram, which serves as a comprehensive overview of the various functional components and tools that are incorporated into the R-IDE extension for the VSCode IDE. The system's functionality is anchored in the ROS framework, but it also includes the Microsoft ROS Extension, which affords developers the ability to commence and terminate the ROScore while providing them with syntax highlighting and automatic import and include paths specific to ROS. These components synergize to enable a seamless development experience that is optimized for ROS-based development.

When working within the R-IDE development environment, a suite of powerful tools and components are readily available to assist developers in writing clean and efficient code. These include, but are not limited to, a code creation wizard for generating initial code, boilerplate code for building new nodes or components, and code snippets that offer intelligent predictions and suggestions for next steps or corrections. Upon successful compilation and launching of nodes, R-IDE facilitates data ingestion into the ROS framework from a ROSbag, where the interface allows for seamless interaction with the timestamped media. Moreover, users can filter out topics and clone the code repository, allowing for greater flexibility in data manipulation. As data flows through the nodes, developers can visualize and manipulate the data, either by publishing or subscribing to the topics, monitoring the node graph of their project, or viewing visual data feeds from cameras or other sources. When ready to publish their software contribution, developers can seamlessly integrate their code into their project outside of the scope of R-IDE.

Figure 1

R-IDE Major Functional Components Diagram



3 Identification of Case Study

ODU's autonomous vehicles leverage the ROS framework to preprocess sensor data, perform advanced computations, and make informed decisions on the most optimal route for the vehicle. Through collaborative efforts between the engineering and computer science departments and their respective mentors, ODU aims to participate in the Self-drive challenge at the Intelligent Ground Vehicle Competition (IGVC). R-IDE, which employs Dr. Belfore's Monarch 1 autonomous vehicle as a model ROS project, has been specifically tailored to meet the needs of Dr. Belfore and his students. However, the goal is to create a tool that is broadly applicable to any ROS developer seeking to enhance their development capabilities.

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

R-IDE is implemented as a highly integrated and modular extension for the widely used VSCode IDE. This extension seamlessly integrates with the VSCode environment, providing users with a diverse array of features and functionalities that streamline the development of complex, ROS-based applications. The primary focus of R-IDE is on simplifying the development process and promoting the adoption of best practices among developers. To achieve this goal, R-IDE leverages an extension pack that allows for the creation of highly modular systems, enabling developers to select the programming language and ROS version that best suits their needs. The development of the prototype is informed by customer feedback gleaned from regular, weekly dialogues with the customer, Dr. Belfore, as well as feedback from a cohort of engineering students who have been granted early access to the extension within the VS-code environment. This approach allows for ongoing refinement and optimization of the tool, with a focus on meeting the specific needs and requirements of the end-users.

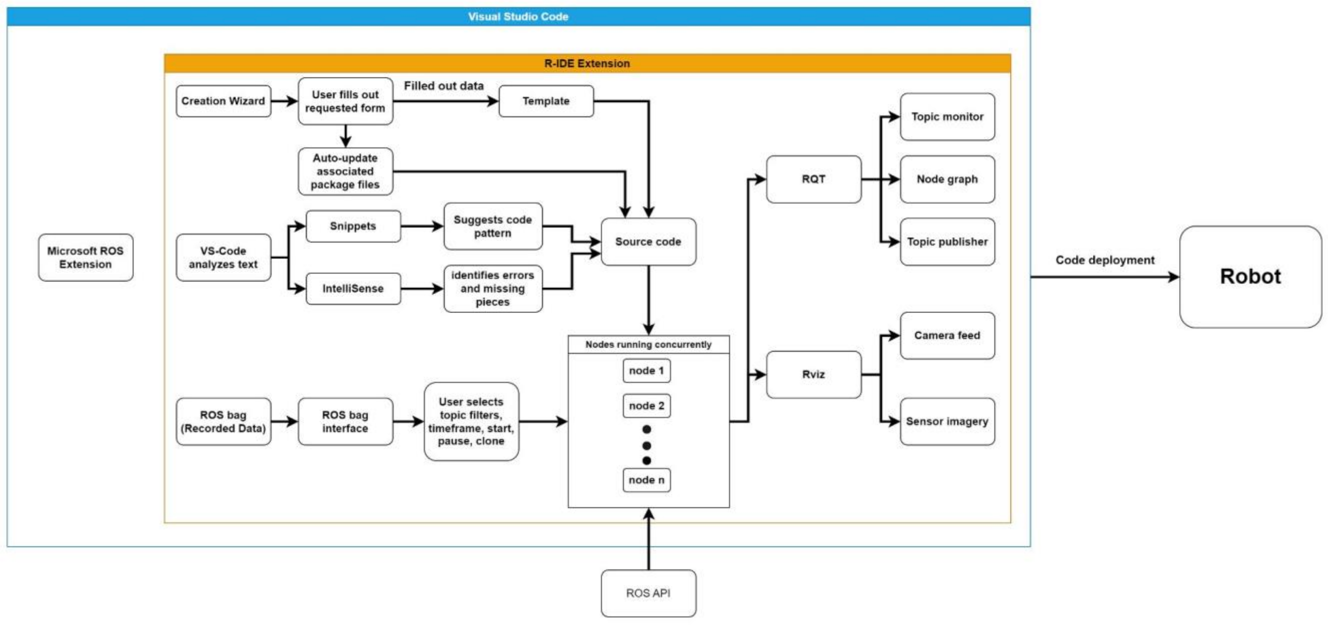
4.1. Prototype Architecture (Hardware/Software)

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.

The R-IDE Major Functional Components Diagram and the prototype MFCD share a high degree of technical and functional similarity. The difference is that the real world prototype shall not incorporate a database. 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).

Figure 2

R-IDE Prototype Major Functional Components Diagram

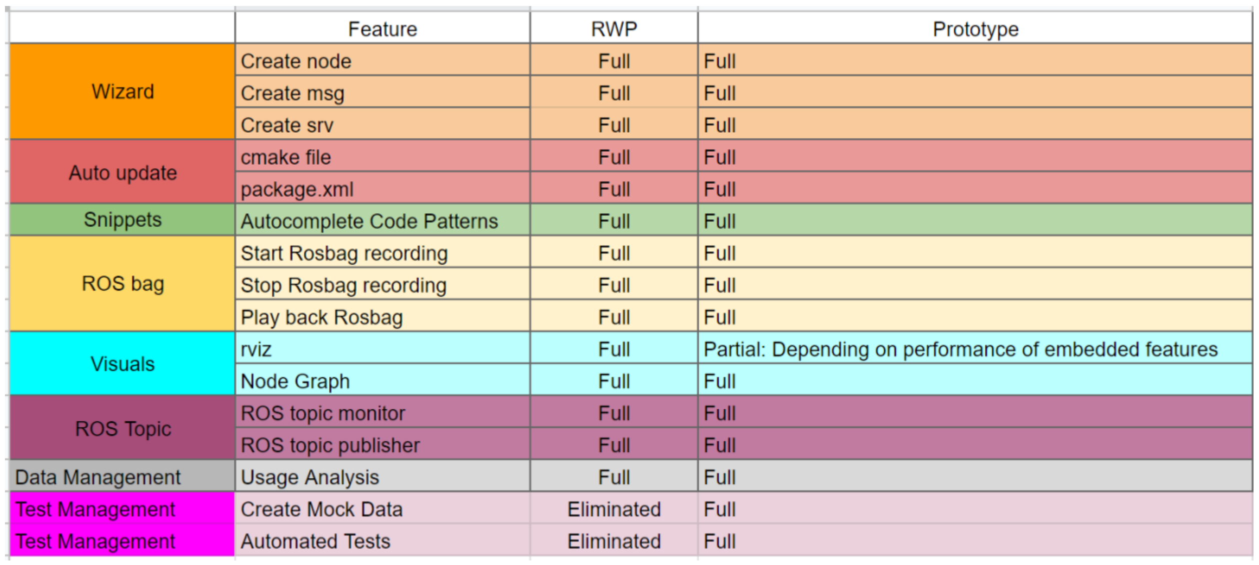


4.2. Prototype Features and Capabilities

The R-IDE real world prototype (RWP) is nearly indistinguishable from the initial prototype, except for some specific features. The prototype alone will be furnished with automated testing and simulated data, whereas these characteristics will be absent in the RWP. The prototype will additionally boast a node creation wizard for nodes, msg, and srv, as well as the ability to automatically update cmake files and package .xml. Code snippets that complete code patterns will be provided by R-IDE to streamline the programming process. The RWP will come equipped with a ROSbag that permits users to initiate, cease, and playback ROSbag recordings. Users will also have access to various visual tools, including ROS visualization, RQT, and a node graph. However, a ROS topic monitor for publishing and debugging will not be included in the RWP, as it is deemed superfluous to its purpose.

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

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

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

Command Palette: refers to the input box in vscode where users can enter commands

Filtering: Create new ROS bag files from older ROS bag files, filtering the new bag based on requested ROS topics from the old one.

Git: Git is a distributed version control system: tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.

Graph View: a graphical representation of the communications between different nodes in a ROS system. It provides a visualization of the ROS topics being published and subscribed to by each node. Graphical View is a useful tool for understanding and debugging complex ROS systems.

GUI (Graphical User Interface): type of user interface that has elements such as buttons, menus to allow the user to interact with the application.

IDE (Integrated Development Environment): application software that provides tools and features for software development. They include features like debugger, source code editor, code completion and even version control in some cases.

IntelliSense: IntelliSense is a general term for various code editing features including: code completion, parameter info, quick info, and member lists.

Monarch 1: two-passenger autonomous vehicle design based on Polaris GEM e2, which is an electric vehicle that is intended to participate in the IGVC's "Self-Drive Challenge".

MongoDB: free non-relational database that stores data in JSON format. The data is stored in collections of documents that can have nested structures for flexible data storage.

Mongoose: library for Node.js that provides a higher level of abstraction on top of MongoDB that simplifies the process of interacting with MongoDB.

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.

ROS Bag: a file format in ROS for storing ROS message data. An important role in ROS, and a variety of tools have been written to allow the user 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: A message is a simple data structure, comprising typed fields. Nodes communicate with each other by publishing messages to topics. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays. As part of a ROS service call, a message can be exchanged between nodes in the form of request and response.

ROS Node: executable programs to perform tasks within the robotic system. 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 Package: software in ROS is organized by packages, a collection of ROS nodes, a ROS library, a dataset, or anything else that requires a useful module for basic functionality.

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 the server is not designed for high-performance, it is best used for static, non-binary data such as configuration parameters. The server is meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify it if necessary.

ROS Services: provide a way for nodes to communicate with each other in a synchronous, request-response fashion . The 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 Subscriber: Node that receives data from a topic published by another node. Nodes use ROS topics to communicate, and a subscriber subscribes to a topic to receive messages that are published on that topic.

ROS Topics: 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.

Robot Operating System (ROS): 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.

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

Snippets: A coding tool to automatically generate repeating code when developing.

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

Topic Monitor: Displays debug information about ROS topics including publishers, subscribers, publishing rate, and ROS messages.

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

Trimming the ROS bag time range: Reducing the duration in time of a requested ROS bag.

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.

Webview: A type of web browser that can be embedded within applications allowing it to display web content.

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/