LAB 1 - R-IDE PRODUCT DESCRIPTION

Justin Tymkin

Old Dominion University

CS411

Professor J. Brunelle

January 30, 2023

Final Version

Table of Contents

Figures

Tables

1. Introduction

The Robotics Operating System (ROS) is a software program created in 2007 by Eric Berger and Keenan Wyrobek, with the goal of assisting in building robots faster. They believed too little time was being spent building intelligent robots because of the time required to reimplement the programs for their robotics applications. Since its creation prestigious companies such as the National Aeronautics and Space Administration have used ROS to build robots to send to space such as the Astrobee, and are currently using the newest version of ROS, ROS 2, to build VIPER a mobile robot to explore the south pole of the moon. With ROS being the software of choice the past ten years, the robotics industry has grown to be worth upwards of $249.2 million and estimates to be worth a little under double that mark in ten years from now.

The R-IDE product is intended for a professor from Old Dominion University, Dr. Belfore, who has a college budget and two semesters to teach new students how to use and implement ROS to compete in the Intelligent Ground Vehicle Competition (IGVC) once a year. This upcoming IGVC marks 30 years of competition for students in concentrations of electrical engineering, computer engineering, and mechanical engineering, from 33 different universities across North America. Dr. Belfore is focused on competing in the autonomous vehicle competition, combining aspects of all fields, to get his own vehicle to drive itself through a course designed by the IGVC. When Dr. BeIfore expresses frustrations with ROS, the question arises, if ROS is good enough for NASA to create robots used in space, why is it not good enough for Dr. Belfore and his students competition in IGVC.

ROS is the best tool for building robotics, but it has a steep learning curve to master in order to be productive. For students with multiple obligations and limited time with ROS, many of them do not often begin to fully grasp ROS until a month before the IGVC. The timeline

consists of, fall semester Dr. Belfore goes through ROS tutorials with students, then spring semester students get their hands on attempting to program with ROS on the Monarch 1, Old Dominion Universities IGVC autonomous vehicle. This cycle has led Dr. Belfore to seek ideas from outside his department.

ROS works in a multitude of developer environments, but often these environments are overwhelming and frustrating for new users. Programming packages for robotics using ROS on a Linux system can require multiple terminal windows, leading to a messy User Interface/User Experience. ROS-Integrated Development Environment will create an IDE for ROS users by developing an extension accessible through VSCode marketplace to lower the learning curve of ROS and empower new ROS developers by offering a friendly UI/UX.

2. R-IDE Product Description

R-IDE provides a wizard in VSCode for the user to create important components, such as nodes, messages, and services, within a ROS package, along with visual tools like rviz, and stat monitoring tools from rqt. A unique aspect of VSCode, compared to other IDEs, is their user marketplace which offers users flexible options to create your own IDE as VSCode does not advertise itself as an IDE, but as a text editor.

2.1 Key Product Features

Providing an extension within VSCode means the user will have everything they need in an organized environment, making for a friendly UI/UX. VSCode offers the option to have buttons on the side of the window to easily navigate between R-IDE, the text editor, and the debugger. VSCode also offers support in opening new windows for the user, allowing ROS developers to program for multiple nodes, or debug multiple packages at the same time.

Being able to create a ROS node, msg, or srv through the R-IDE wizard offers a unique tool to new ROS developers, instead of bumbling through multiple terminal windows attempting to track what nodes are communicating, R-IDE has everything in one place allowing users to quickly program their robotics. These are three things VSCodes ROS extension does not currently offer to ROS developers. The wizard walks through the steps of creation, then send generic templates to the text editor for the developer to begin creating their robotics application.

Allowing the user to record and play back a ROS bag, a file for storing ROS message data, at the touch of a button offers more flexibility for learning what is happening within their robot. A ROS bag records the raw data collected by the robot using the program by the user. With the option to play it back, this offers a seamless experience for new users to understand any mistakes they may have made.

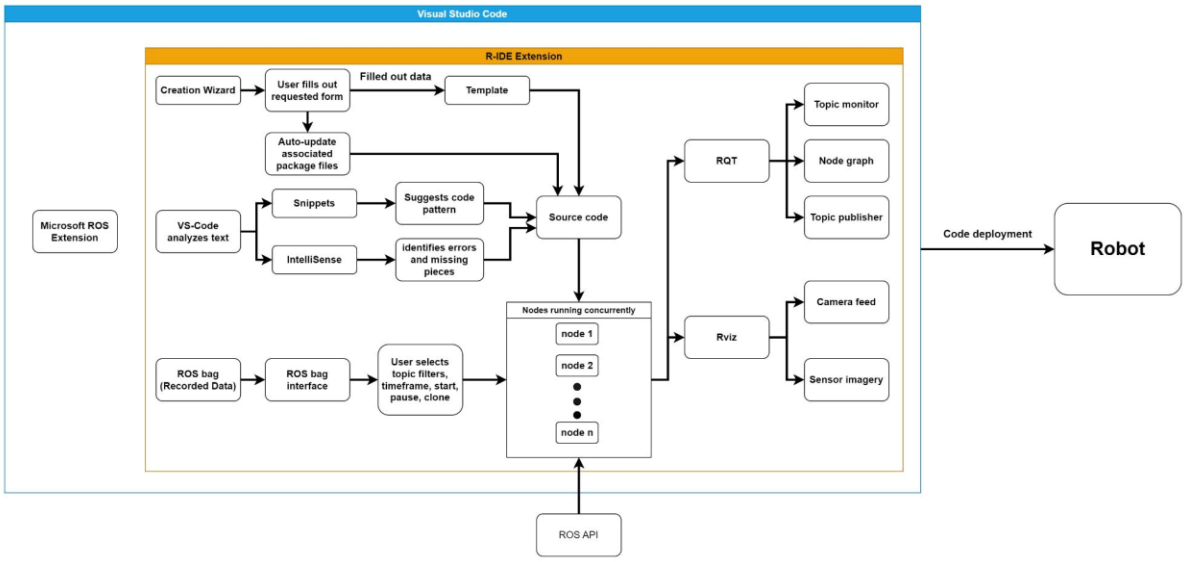
The addition of tools from rviz and rqt offers tools for the user to develop a more defined understanding of what and how they are programming. Rviz is a 3d visual tool that replays what the user has programmed the robot to see. Rqt is an extension for ROS which offers a variety of tools for ROS developers. The first tool from rqt being the node graph which is a graph that shows how the current node the developer is programming is communicating with other nodes. The second tool from rqt is the topic monitor which allows the user to select which ROS topic they would like to look at and shows the types, bandwidth, and Hz of all topics. The last tool from rqt is the message publisher, it tracks the message type to publisher and the frequency for publishers in Hz.

2. 2 Major Components

The major components of R-IDE can be seen in figure 1, the Major Function Component Diagram (MFCD). Inside of VSCode, along with the VSCode ROS extension, the R-IDE extension offers the key features in the chart below. There are two starting points for a ROS developer inside of R-IDE, the first being the creation wizard. The wizard allows the user to create a node, msg, or srv, for a ROS package. The wizard then auto fills the type of file selected from the user with file templated code provided by R-IDE, uses snippets and intellisense to guide the user through programming their node, and auto updates associated cmake and .xml files.

Figure 1

R-IDE Major Functional Component Diagram



Once the node is completed the second place to begin with R-IDE is using the ROS bag to record data being sent between the nodes. R-IDE allows the user to start and stop recording

then playback the data recorded. From the nodes created the user can also utilize tools from rqt and rviz, rviz is a visualization tool which offers the user the ability to see what the robot sees and rqt offers multiple ways to chart and monitor how the nodes are communicating.

3. Identification of Case Study

R-IDE is designed for new ROS developers and is being created inside of VSCode to encourage additional features be added to future iterations. ROS is always changing and adding new features and R-IDE wants to change with it, through the help of the community. The first iteration of R-IDE is focused on Dr. Belfore and the Electronic and Communications Engineering team.

Now all the IGVC use ROS in their competition, so R-IDE could be appealing to the universities competing. And R-IDE is already helping to lower the learning curve for students, so it could also be useful for other universities, or encourage other universities to attempt to develop an extension that adds features that compliment R-IDE.

Old and new robot enthusiasts can also find use from R-IDE, new enthusiasts for the lower learning curve, but new enthusiasts may find the R-IDE UX inside of VSCode more appealing than a Linux environment.

4. Product Prototype Description

R-IDE was initially intended to be a prototype but will be a full product, offering Dr. Belfore to create new nodes, srv, msg and play through his previous recorded ROS bags. Being a full product, R-IDE will receive weekly feedback on the functionality of the software. VSCode will handle all risk mitigations associated with running extensions.

4.1 Prototype Architecture (Hardware/Software)

The hardware requirements for R-IDE require the developer to have a computer that can run VSCode and ROS. The software for R-IDE will require the user to have access to VSCode, ROS1 which runs in a Linux environment like Windows Subsystem for Linux using Ubuntu version 20.04 or lower, and ROS2 which can run on Windows, Mac, or Linux. During the development phase simulated data will be used to test the functionality of features created until the prototype is finished. The R-IDE extension will use a MySQL database provided by ODU, but most data will be handled locally by the user.

Figure 2

R-IDE Major Functional Component Diagram

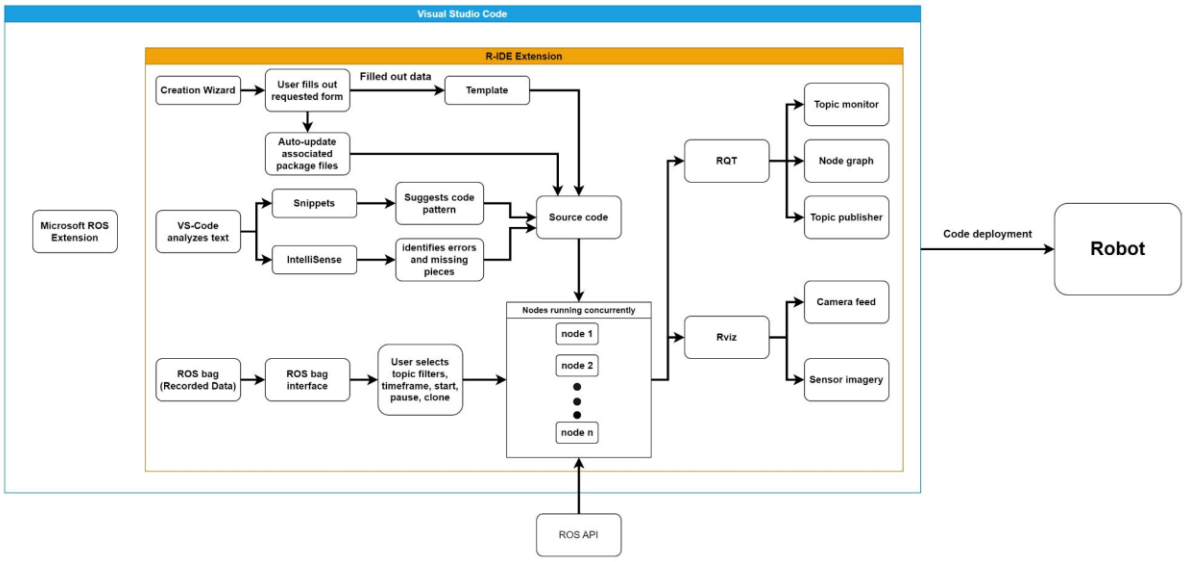


Figure 2 shows the prototype for the MFCD but there is no prototype, so it is the same as the original MFCD. R-IDE extension will work along with the Microsoft version of ROS in VSCode to complete a ROS IDE.

4.2 Prototype Features and Capabilities

All the R-IDE features will be in the prototype and the real world product will exclude the testing features. The testing will used simulated data to test the prototype which will not be necessary for the RWP. Table 1 shows features of R-IDE.

Table 1

R-IDE Real World Product vs. Prototype

	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 wizard will create one of the three options, node, msg, and srv, automatically filled with file snippets. Code snippets will also be used to help program robots by auto completing code in files, and as these programs are being created the cmake file and package .xml file will be automatically updating. The user will have the ability to record and stop recording ROS bags, and then play back the data collected in the bag. The visual tools and ROS topic will be embedded into the extension, rviz is used to visualize sensor data and the Node Graph is used to observe what publisher and subscribers nodes are using to communicate. The ROS topic monitor

and publisher create a chart tracking the name, types and bandwidth of either a topic or publisher. The data management will be tracked by VSCode.

4.3 Prototype Development Challenges

Developing the full product will include many challenges, the prototype will need to meet standards before the Electronic and Communications Engineering team can test the full product. Building the prototype requires an abundance of knowledge of ROS which has a steep learning curve, and then there is the obstacle of compatibility with ROS2. Developing in ROS requires multiple terminal windows and VSCode allows the user to utilize multiple terminal windows when developing, and when developing with R-IDE if there is a syntax error it needs to highlight the error across all open windows. Embedding existing tools in to VSCode, then the tool to work seamless will require the work of an entire team. If all this works, ROS is old enough that finding documentation for ROS can be very difficult.

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 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 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 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.

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

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

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.

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

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

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.

Refences

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/