Lab 1 - R-IDE Product Description

Dominik Soós

Old Dominion University, Computer Science Dept.

CS 411 – Professional Workforce Dev. II

Professor J. Brunelle

March 13th, 2022

Final Version

Table of Contents

List of Figures

List of Tables

1. Introduction

ROS (Robot Operating System) is a popular open-source framework that allows developers to easily create and integrate applications for robotics. It is designed to simplify the process of creating complex and sophisticated robot behavior by allowing developers to focus on the core functionalities of their applications (Kerr J. & Nickels K., 2012). It also has a large and growing research community of developers and users, which provides a wealth of resources and support for working with ROS.

The market for robotics is rapidly growing and is expected to continue to expand in the future. As a result, the demand for ROS and other robotics frameworks is also expected to increase. This can be attributed to the growing use of robotics in various industries, such as manufacturing, logistics and healthcare. ROS is well-positioned to capitalize on this growing market and continue to be a major contributor in the robotics industry (Garber L., 2013). According to market research, the current market value of ROS was valued around $274.2 million in 2021 (Global Robot Operating System Market Research Report, 2022). The market for ROS is more likely to grow even higher than the predicted market value and expected to reach $460 million by 2030 (Global Robot Operating System Market Research Report, 2022). ROS is being used by some major companies including the National Aeronautics and Space Administration (NASA) in the Mars Curiosity Rover and at the International Space Station (Gerkey B., 2014).

NASA uses ROS for a variety of applications in its robotics research and space exploration missions. One example of how NASA uses ROS is in the development of autonomous robots that can operate in challenging environments, such as the surface of Mars

(Neel V. Patel, 2021). ROS offers a set of powerful tools and libraries that allow NASA engineers to program and visually control these robots using Gazebo, enabling them to perform complex tasks and gather valuable data (Cardoso et al., 2020). Additionally, ROS is also used in the development of robotic spacecraft and rovers, which are used to explore the surface of other planets and gather scientific data (Wessling B., 2022). Overall, ROS plays a critical role in NASA's efforts to advance robotics and space exploration.

Despite its many advantages, ROS also has some significant challenges that can make it difficult to use and maintain. One of the main challenges of using ROS is the need to set up and configure a ROS environment, which can be time-consuming and require specialized knowledge. This process involves installing the ROS software, creating a workspace, and setting up the environment variables. It can be especially difficult for beginners who are not familiar with the ROS architecture and conventions. Since it requires specialized knowledge, there is a steep learning curve. For new users, getting started with ROS can be overwhelming due to the large number of libraries, tools and conventions that need to be understood in order to effectively use the framework. This can lead to long development times and frustration for those who are trying to build their first ROS applications.

Another challenge of using ROS is the time-consuming debugging process. Since ROS is a complex system with many interconnected components, tracking down and fixing errors can be a daunting task. This is compounded by the fact that ROS applications need to run on multiple terminal windows, requiring developers to switch between them frequently in order to manage and debug their code. If the nodes are interconnected and rely on each other to function properly, this can be particularly inconvenient and confusing.

There are tools available that can help alleviate these challenges and make ROS more accessible and user-friendly. One such tool is Robotics Integration & Development Environment or R-IDE, a Visual Studio Code extension that offers a simplified user interface, quick environment setup, and wizards for common tasks. In addition, R-IDE provides debugging capabilities for ROS nodes and ROS bag recording, making it easier to troubleshoot and optimize ROS applications. This report explores how R-IDE helps overcome these challenges. It looks at the features of R-IDE and how they improve the development experience for ROS users, as well as the benefits of using this tool in the context of robotics development.

2. R-IDE Product Description

R-IDE is a powerful Visual Studio Code extension that is designed for developers working with ROS. Users can use its tools and features to quickly set up a ROS environment and develop ROS applications. The main features, as shown in Figure 1, include wizards for common tasks, code autocompletion, and tools for working with ROS bag recording and visualization tools. With R-IDE, users are able to use the Creation Wizard to perform common tasks like creating new ROS nodes, creating messages and services, and updating the CMake and package.xml files.

2.1. Key Product Features

R-IDE is equipped with a comprehensive environment to speed up the development cycle for ROS applications. One of the key features is its ability to quickly set up a ROS environment within VSCode. By utilizing the provided user interface, it is possible to quickly initiate ROS projects with minimal effort, thereby optimizing time efficiency.

R-IDE includes Creation Wizard for common tasks like creating new ROS nodes, creating messages and services, and updating CMake and package.xml files. This makes it more

time-efficient for developers to focus on the core functionalities of their projects, rather than wasting time on writing boilerplate code.

A distinct aspect of R-IDE is its support for advanced features like code auto-completion and usage analytics. With these tools, users can write code more efficiently and even track their wizard usage over time. R-IDE also includes tools for starting, stopping and playing back ROS bag recordings, allowing users to easily test and debug their applications.

Other functionalities of R-IDE offer debugging capabilities for ROS nodes. This allows the user to identify and fix errors quickly and efficiently, saving them time and frustration. R-IDE's debugging tools are user-friendly and intuitive, making them easy to use even for developers who are new to ROS.

R-IDE also provides convenient visualizations for the ROS environment using ROS Quick Toolkit or (RQT) which is a graphical user interface for ROS, ROS Visualization or Rviz, and Node Graph. RQT is a graphical user interface for ROS while Rviz is a 3D visualization tool for ROS. Node Graph is a visual representation of the nodes in a ROS system and how they are connected to one another. These tools allow users to see the size of their ROS nodes and topics, and to quickly identify any potential issues. With R-IDE, users are able to easily monitor and publish messages on ROS topics, giving them complete control over their ROS environment.

Another key feature of R-IDE is its versatility. It can be used in real-time, allowing the user to test their applications as they develop them. Alternatively, users can also use R-IDE with previously recorded data, allowing developers to test their applications using previously recorded data sets. This flexibility makes R-IDE a valuable tool for developers working on a wide range of ROS applications.

R-IDE is an essential extension for any developer looking to streamline their workflow and improve their productivity. Its user-friendly environment and powerful debugging capabilities make it an invaluable asset for speeding up the development process and ensuring the quality of ROS applications. Whether the user is new to ROS or an experienced developer, R-IDE is a valuable tool that can help build and test ROS applications more efficiently and effectively.

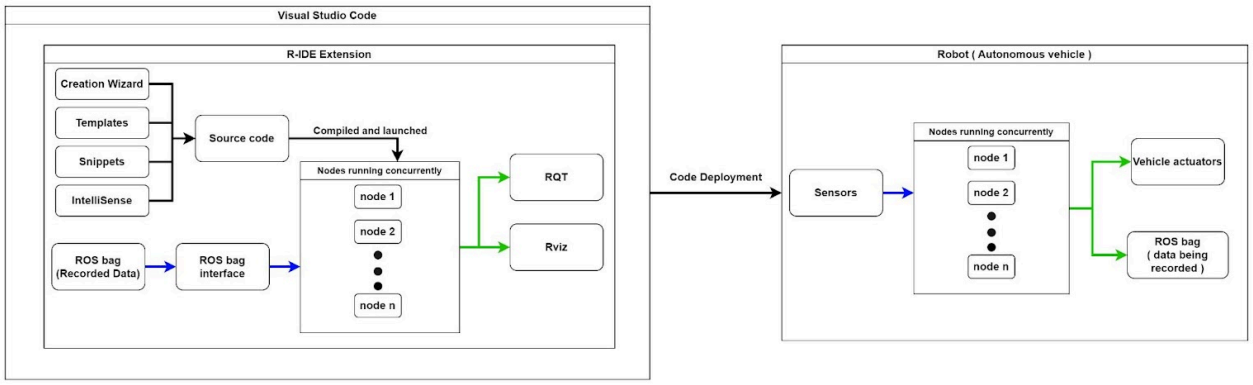

Figure 1 Solution Process Flow

2.2. Major Components

R-IDE consist of several core components. These include a project creation wizard, VSCode analyzer, a ROS bag recorder, and the RQT and Rviz visualizers. The creation wizard facilitates the rapid establishment of new ROS projects. The VSCode analyzer assists in identifying and rectifying code errors. The ROS bag recorder enables the recording and playback of data from ROS systems, aiding in testing and debugging. The RQT and Rviz visualizers provide visualizations of ROS systems in real-time, enabling it to monitor the functionalities of the system. These components enable R-IDE to serve as a comprehensive tool for ROS application development.

The Creation Wizard is a user-friendly tool that allows developers to easily create new ROS nodes, messages and services. It does this by asking the user to fill out a request form, where the user have the ability to specify the file type, node name, path to the file and other features. The submitted form is then used to automatically update the associated package files. This makes it easy for developers to set up a new ROS project without having to manually create and configure all of the necessary files.

The VSCode analyzer is a helpful tool that allows developers to easily write code for ROS applications. It uses advanced machine learning techniques to analyze the text of the code and give helpful suggestions in the form of code snippets and Intellisense. Intellisense is designed to provide programmers with context-aware suggestions while writing code, thereby reducing the need to manually type out complete code snippets. This makes it convenient for developers to write correct and efficient code, and it also helps identify any errors or missing pieces in the code.

Another key component of R-IDE is the ROS bag interface. This allows users to connect their VSCode environment to a ROS bag, which is a file format used to store data from a variety of sensors and other sources. By connecting to a ROS bag, users can access and visualize the data contained within it using tools like RQT and Rviz.

The RQT and Rviz visualizers are practical tools that allow developers to see what is happening in their ROS application. The RQT visualizer supplies a node graph that shows the connections between nodes, as well as a topic monitor that allows the user to see the data being published on different topics in real time. The Rviz visualizer provides a live camera feed and sensory imagery, allowing developers to see the environment in which their ROS application is running. This is particularly useful for debugging and understanding the behavior of the nodes.

Together, these tools provide users with a comprehensive understanding of the data being collected by their sensors and other devices.

3. Identification of Case Study

The collaboration between computer science and the electrical engineering department has been instrumental in the development of R-IDE. The computer science students have provided expertise in software development and user experience, while the engineering students gave their insight into the needs and challenges of ROS developers. Together, the team has designed a useful tool that is tailored to the needs of ROS developers. Working closely with Dr. Belfore and his engineering students, who are the intended users, sought to create a solution that would help improve their productivity, simplify common tasks as well as attacking the steep learning curve of ROS. The intended use of R-IDE is to offer an intuitive yet powerful development environment for building ROS-based applications, and to supply tools that can help reduce the initial overhead of learning ROS and enable new developers to quickly become proficient in using it. R-IDE serves as a valuable resource for ROS developers, and it is expected that it will be utilized in a variety of applications.

4. Product Prototype Description

The R-IDE is an advanced extension for VSCode, designed to optimize the functionality of the widely-used development environment for users working with ROS. The extension offers a proof of concept for creating new functional nodes, srv, msg, enabling users to conveniently create and manipulate these elements within VSCode. R-IDE allows users to play previously recorded ROS bags from Dr. Belfore, providing a streamlined approach to reviewing and analyzing data from previous projects. The extension includes risk mitigation features, leveraging the capabilities of VSCode extensions to provide users with the necessary tools to

work with ROS in an efficient manner. To ensure that R-IDE meets the requirements of its users, the development team will engage in a weekly discussion with the initial customer Dr. Belfore and his engineering students to gather feedback and make ongoing adjustments to the product. The ultimate goal of R-IDE is to furnish users with a powerful and user-friendly tool for working with ROS inside VSCode.

4.1. Prototype Architecture (Hardware/Software)

The R-IDE extension for VSCode is a valuable tool for any developer working with ROS and developing source code for robotics applications. The extension includes several functional components that work together to provide an effective and intuitive experience for users. The Creation Wizard, ROS bag interface, and code analysis capabilities make it a valuable tool for any ROS developer. Figure 2 illustrates several major functional components of the extension including the Creation Wizard and ROS bag interface. The Creation Wizard allows users to create template source code, which can then be customized to fit their specific needs. The ROS

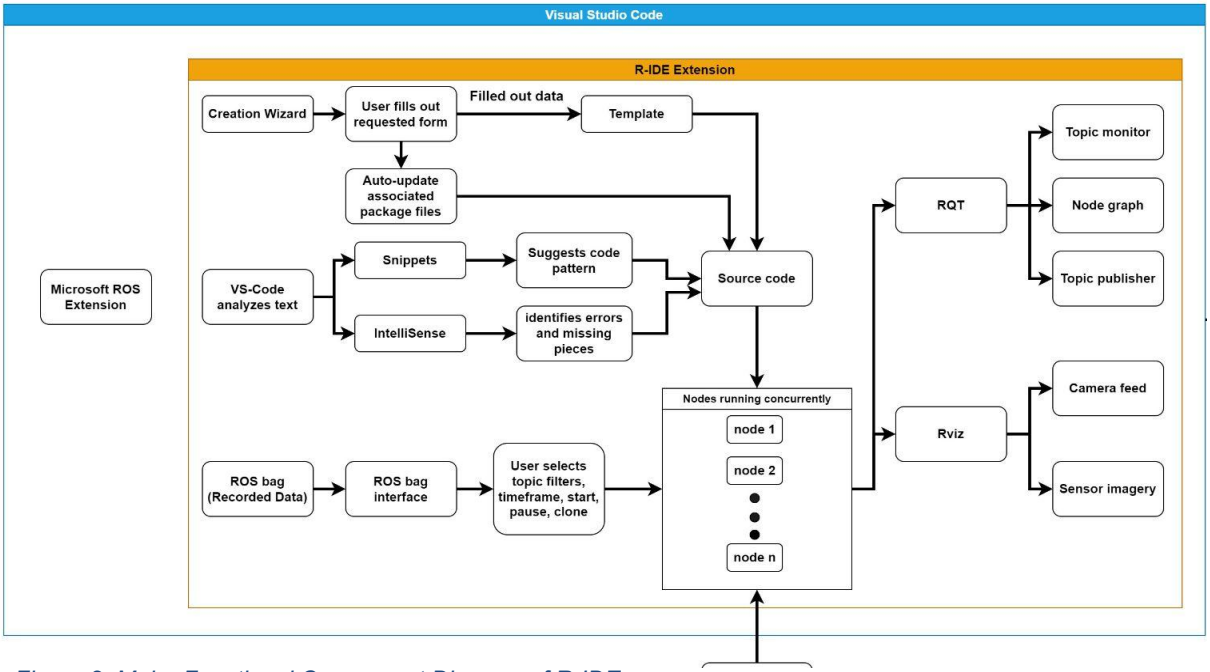

Figure 2 Major Functional Component Diagram of R-IDE

bag interface connects to ROS bags and allows users to access and visualize the data contained within them. Additionally, the extension has the ability to analyze text and include code pattern suggestions, as well as identify errors and missing pieces in the source code.

To use the R-IDE extension, a computer that meets the hardware requirements for VSCode, ROS and Rviz is required. These requirements may include a sufficient amount of memory and storage, as well as compatible operating system such as Ubuntu. The software requirements for the extension include ROS1 or ROS2 and VSCode. ROS 1 requires a Linux environment, while ROS 2 can run on Linux, Mac, or Windows. A basic understanding of ROS and VSCode is necessary to use the extension effectively.

The extension works seamlessly, with the existing setup of VSCode and ROS, making it a powerful tool for ROS developers. The extension exists withing VSCode alongside a complementary extension Microsoft ROS extension, and it includes a Creation Wizard that requests the user to fill out a form to generate code templates for common ROS components like Node, Service, and ROS messages. VSCode suggests and fixes code using customized snippets and IntelliSense, and the other ROS bag interface manipulates and manages ROS bag recordings. Output data is displayed through various visualization tools embedded within Microsoft ROS extension such as Rviz, RQT and Node Graph. The software is deployed to the car but deployment is not handled by R-IDE. With this architecture, the R-IDE extension works seamlessly with the existing setup of VSCode and ROS, making it a powerful tool for ROS developers.

4.2. Prototype Features and Capabilities

The R-IDE extension is a comprehensive tool suite for developers who are engaged in working with ROS. Its features, as presented in Table 1, are geared to streamline the

development process and to minimize the occurrence of errors. The extension boasts an array of essential features such as automated tests and mock data, a Creation Wizard for nodes, messages and services, and code snippets which autocomplete common code patterns. All these features help in accelerating the process of application development and can help developers save considerable amounts of time. With these tools, developers can create better quality applications in less time, making R-IDE extension a valuable addition to any ROS developer's toolbox.

One of the key features of the R-IDE extension is the ability to state, stop and play back ROS bag recordings. This allows developers to easily debug and test their code, making it easier to resolve potential issues arising. The extension includes visualization tools such as Rviz, RQT, and Node Graph to help developers understand and debug their code, along with ROS topic monitor and publisher for debugging purposes. The Rviz feature may only be implemented partially in the prototype as it depends on performance of embedded features.

The R-IDE extension also provides capabilities such as code generation and code suggestions, making it easy to create new functional nodes, services, and message types. It also includes risk mitigation features that ensure compatibility with other extensions in VSCode, making it easy to use alongside other tools. The R-IDE extension is an excellent resource for

	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

Table 1 Real World Product vs Prototype Features Table

developers working with the Robot Operating System. It has many features and capabilities that make the development and debugging process easier and more efficient, making it a highly valuable tool for developers.

4.3. Prototype Development Challenges

The R-IDE extension has been developed with the aim of facilitating wider accessibility to ROS. To achieve this goal, the extension must be published to the VSCode marketplace, which poses a challenge due to the need to comply with the guidelines and standards of the marketplace.

In order to ensure seamless functionality across multiple terminals, the R-IDE extension must maintain consistency in cursor and line position. This requires rigorous testing and debugging to eliminate any potential inconsistencies. Similarly, highlighting words across multiple terminals requires equal amounts of testing and debugging.

A major challenge is the incompatibility of the R-IDE extension with both ROS 1 and ROS 2, which are separate versions of the Robot Operating System. The lack of full compatibility between the version requires separate codebase and testing. ROS can also pose a challenge for new users and for developers who are inexperienced with the platform, despite the R-IDE's extension's aim of simplifying the learning process. Becoming familiar with ROS still requires a great amount of time and effort.

Documentation for ROS may not be comprehensive or up-to-date, which can present difficulties in the development of the extension. This may require additional research and experimentation to resolve any issues. The embedding of visualization tools such as Rviz, RQT and Node Graph into VSCode may also require additional testing.

The R-IDE extension is intended for use by Dr. Belfore and his ECE students, and it is crucial to collaborate with them to ensure the extension meets their needs. This requires communication and feedback to make the extension user-friendly and effective. Making the R-IDE extension work with both ROS 1 and ROS 2 is a challenge because they are not the same and may need additional work and testing.

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 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.

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

Cardoso, R.C., Farrell, M., Luckcuck, M., Ferrando, A., Fisher, M. (2020). Heterogeneous Verification of an Autonomous Curiosity Rover. In: Lee, R., Jha, S., Mavridou, A., Giannakopoulou, D. (eds) NASA Formal Methods. NFM 2020. Lecture Notes in Computer Science(), vol 12229. Springer, Cham. https://doi.org/10.1007/978-3-030-55754-6_20

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

Garber, L. (2013). Robot OS: A new day for robot design. *Computer*, *46*(12), 16-20.

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

Kerr J and Nickels K., "Robot operating systems: Bridging the gap between human and robot," Proceedings of the 2012 44th Southeastern Symposium on System Theory (SSST), 2012, pp. 99-104, doi: 10.1109/SSST.2012.6195127.

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

Neel V. Patel (2021, April 12). NASA's next lunar rover will run open-source software. Retrieved December 14, 2022, from https://www.technologyreview.com/2021/04/12/1022420/nasa-lunar-rover-viper-open-source-software/

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/