Do You Need a Virtual Machine? ROS Installation Challenges and Concerns

Isabella Samuelsson
Department Of Computer Science at North Carolina State University
Raleigh, NC, USA
insamuel@ncsu.edu

Kelly Wang
Department of Computer Science at North Carolina State University
Raleigh, NC, USA
kawang@ncsu.edu

Sana Mahmoud
Department of Computer Science at North Carolina State University
Raleigh, NC, USA
simhmou@ncsu.edu

Abstract— **With the increasing adoption of robotics systems in many domains, it is important to enable efficient building and simulation of these systems through the use of software that is compatible with different operating systems. To do this, clear instructions should outline the installation process of the software while also addressing differences between operating systems and remaining updated as the software changes. This paper aims to identify some of the challenges faced during the installation of ROS, a set of libraries and tools for building robot applications. We examine some of the challenges faced during the installation of ROS from the official ROS.org website. We categorize the findings into seven distinct categories using an approach called Open Coding. We aim to identify and prioritize some of the barriers to access ROS for users to enable makers of the guides and of the ROS distributions to improve access to these tools.**

Index Terms—ROS Installation, ROS documentation

I. INTRODUCTION

The first step to developing robotics systems involves installing the software required to create these systems as well as the software required to simulate them. ROS was originally built to run on Linux based systems and support for macOS and Windows came later. The main ROS client libraries are geared toward a Unix-like system due to their dependence on large sets of open-source software dependencies. Ubuntu Linux is listed as "supported" for these client libraries while other OSs like Fedora Linux, macOS, and Windows are listed as "experimental". Currently, Ubuntu versions are used in coordination with specific ROS distributions and makers of the distributions pick particular versions of Ubuntu to target for support for the lifetime of the ROS or the Ubuntu distribution. The same is not true for Windows and macOS as users need to have exact versions of the packages at the time that the developers built them on macOS or Windows to ensure the binary distributions will work. This creates more challenges for users installing and setting up the software to develop and simulate these robotics systems.

We aim to outline some of the challenges faced during the installation process on different operating systems to identify what the main categories for those challenges are. We use open coding, an inductive and analytical approach that sorts through data, categorizing it by properties into categories that portray the meaning of the data as well as its relation to other data [5], [13]. It is important to note that our team consisted of Computer Science scholars with no prior experience in the ROS ecosystem and had not previously worked on any robotics projects.

Our contributions include:

- An examination of the issues that arise during installation of ROS on macOS and Windows, resulting in seven categories of issues.
- Identifying some shortcomings in official ROS installation documentation that hinder support for installation on non-Linux machines.
- An artifact of our original pictures and terminal output available on github [12].

II. BACKGROUND

A. What is ROS?

The Robot Operating System (ROS) is a free and open source set of libraries and tools that can be used to build software systems for robots. A robot has many parts that must work together efficiently to perform a task. Three of the main parts in a robotic system are the sensors, actuators and control systems. Sensors are things that observe the world, actuators are things that move and take action on the world and control systems act as the robot's brain. ROS helps developers efficiently build these sensor, actuator and control system components and easily connect them using ROS tools like nodes, topics and messages. A ROS node is a process that performs a computation. A robot control system will usually be made up of many nodes. ROS nodes use topics in order to communicate with one another. ROS

topics are named buses over which nodes exchange messages [1]. Nodes that are interested in a certain kind of data subscribe to a relevant topic and nodes that generate a certain type of data publish to a relevant topic. These messages can be recorded using ROS bag files or logs in order to test a robotic system.

B. What is a ROS Distro?

A ROS distribution or distro is a versioned set of ROS packages. The purpose of ROS distributions is to allow developers to work with a codebase that isn't constantly changing. Once a distribution is released, the ROS developers try to limit changes to bug fixes and non-breaking improvements for the core packages [6]. The ROS distributions alternate major and minor distributions. Major ROS distributions will have long term support and minor ROS distributions will not. The last five ROS distributions released are shown in Figure 1 [6]. We decided to investigate the installation of the Melodic, Noetic and Kinetic ROS distributions for Windows and macOS.

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys **(Recommended)**	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017

Fig. 1. Most Recent ROS Distributions

C. How do you install ROS?

There are more than one ROS distributions supported at one time, as shown in Figure 1. Some are older releases with long term support, making them more stable, while others are newer with shorter support lifetimes, but with binaries for more recent platforms and versions of ROS packages [2]. Currently the ROS developers recommend ROS Noetic. The most straightforward approach to installation is to view the existing distribution on the ROS documentation website and select the installation instructions for the specific distro you would like to use and operating system you have. Currently only some distributions of ROS are tested and ensured compatible with certain operating systems. As you can see in Figure 2 and Figure 3 [6] not many are supportive of Mac and Windows and many are supportive of Ubuntu. Many developers' main machines are Windows and Mac and the inability to support these popular operating machines lead to many users having to use a virtual Ubuntu machine in order to use ROS [14], [15].

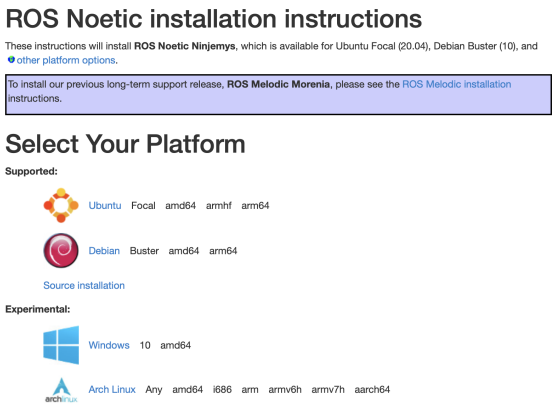

Fig. 2. Noetic Supported Platforms

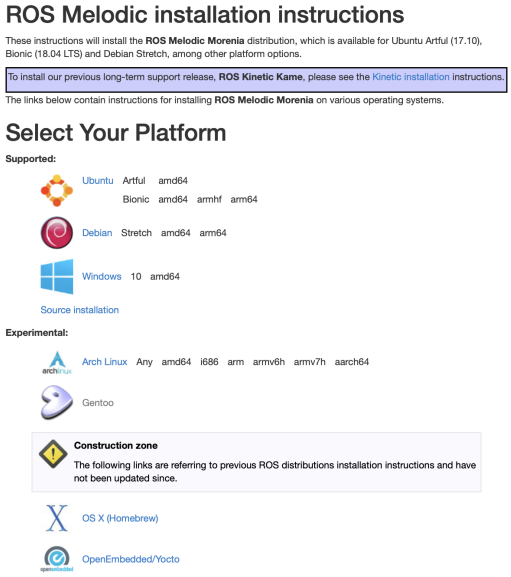

Fig. 3. Melodic Supported Platforms

III. RELATED WORK

White and Christensen [10] discuss the use of ROS with Docker to simplify the set up of ROS environments. The authors discuss how issues with repeatable and reproducible environment setups affect collaboration on robotics software. The paper guides users through a setup for ROS that utilizes Docker as a tool for continuous integration and collaboration between maintainers as well

as enabling maintainers to review pull requests and build patches without contaminating their development environment. Similarly, [11] discusses some of the current problems facing software building and installation caused by “Dependency Hell”, a term used to define problems faced by users when software packages are dependent on other software. Boettiger discusses how solutions like workflow software, virtual machines, and continuous integration among other practices might help solve some of these problems but that researchers still face barriers to entry in learning these tools due to lack of approaches to teach these tools correctly. Further research needs to be done exploring how ROS and development using ROS can be made easier to adopt for different operating systems. There is little work done exploring the challenges faced for developing with ROS on different systems. There are many discussions of the challenges faced installing and developing with ROS on forums like ROS Discourse, StackOverflow, Github, askubuntu, and Reddit. Most existing work discusses reproducibility in software but these works address the challenge of reproducibility from installation to building, running, and developing software instead of delving deeper into identifying the challenges of installation alone and proposing solutions to those.

IV. METHODOLOGY

To identify the challenges faced with ROS installation, we follow the ROS installation documentation for the Noetic, Melodic, and Kinetic distributions found in [2] for installing ROS on Mac and Windows. For all installations we created new user accounts to have a fresh installation of the software to ensure no prior configurations or installations affect our installation of ROS. For the Windows installation we used a Windows 10 Base 64 bit virtual machine. The Mac installation was done on a Macbook pro running macOS 11.7.4. These installations yielded seven issues and three sub-issues. We then followed the procedure for Open Coding discussed in [3], [4], [5] to assign the issues categories without determining the categories beforehand to enable the categories to emerge from the data. To ensure familiarity with the types of issues we encountered, we searched for the errors we encountered on StackOverflow and other forums to ensure we were understanding the reasoning behind the errors and categorizing them accordingly. After all of us completed a pass over the issues and assigned categories we reevaluated the categories that we assigned the issues to and reached a consensus on the set of categories that the issues were assigned to.

V. RESULTS

Our analysis yields 7 distinct categories from the 14 issues discovered across the macOS and Windows operating systems and Kinetic, Melodic and Noetic Distros. We will first describe each category and include images for the issues for that category and then display charts on the aggregated counts of each type of issue for each distribution.

Category 1: Dependency Issues

This category includes issues with installing packages that have dependencies that fail to install. There are three instances of these types of issues spanning the macOS Melodic and Kinetic installation guides. The first issue is in the Melodic installation guide at Step 1.3: Install wxPython. During the wxPython installation command an import error for the attrdict package occurs. This issue can be resolved by first installing attrdict individually and then running the initial wxPython installation command again. While this fixes the attrdict import error the Step 1.3: Install wxPython command still does not complete. An additional syntax error is encountered which is discussed further in the subsequent Category 2: Syntax Errors results section.

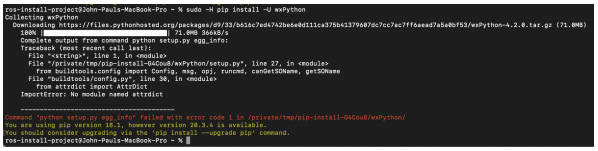

Fig. 7. Attrdict Install Error

The second Category 1 issue was found when trying to resolve the Figure 11 Category 2 syntax error issue with the Step 1.3: install wxPython command in the macOS Melodic installation. To address the Category 2 error we tried the same wxPython installation command but with python 3 instead of python 2. In this case the wxPython installation did not error out with a syntax error but with a building wheel failure for the Pillow library. This wheel failure for the Pillow package is shown in Figure 8.

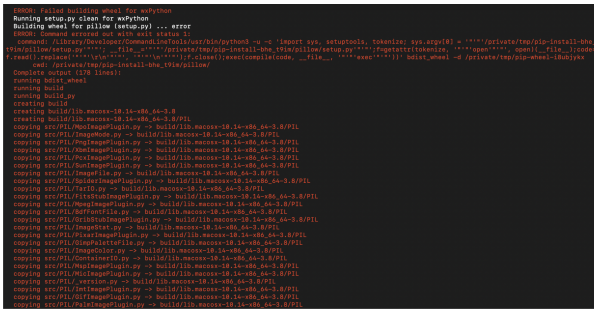

Fig. 8. Pillow Wheel Failure

The third Category 1 issue addresses the Pillow wheel failure discussed above. In order to resolve the previous Pillow wheel failure we wanted to try to install the Pillow library individually before again trying to complete Step 1.3: Install wxPython. The individual Pillow installation failed as well with a building wheel failure that can be seen in Figure 9.

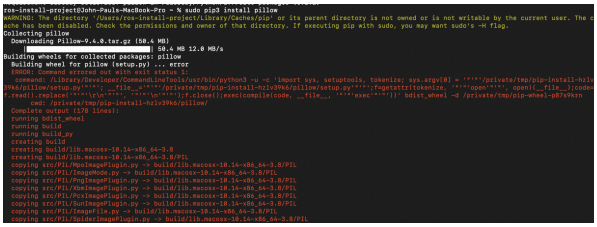

Fig. 9. Pillow Wheel Individual Install Failure

Category 2: Syntax Errors

This category involves issues in which a syntax error hindered the installation of a package. We found two instances of this type of error both of which were for macOS installations of ROS. The same syntax error was found in both the Melodic and Kinetic macOS ROS installations on Step 1.3: installation of the wxPython package. The error can be seen in Figure 10 and Figure 11.

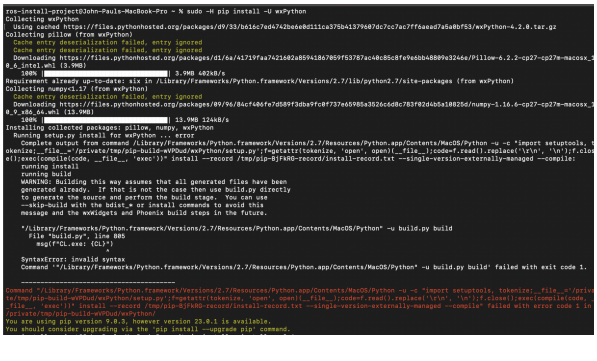

Fig. 10. Melodic Syntax Issue

Fig. 11. Noetic Syntax Issue

Category 3: Deprecated Packages

This category includes issues where packages have been moved from their original location or do not exist anymore. We found one issue that falls under this category. This issue is for the macOS Kinetic installation on Step 1.1: Homebrew, installing the Homebrew science tap. This step allows the user to use non-standard formulae. Figure 12 shows that the homebrew/science tap is deprecated and empty. The error message states that all the tap's contents were either deleted or migrated. On stack overflow other developers have mentioned that the formulae were migrated to the core repository and are available without taping homebrew/science [9].

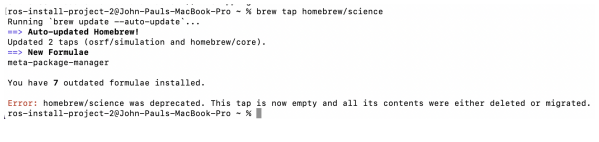

Fig. 12. Deprecated Package Issue

Category 4: SSL Library Issues

Category 4, SSL Library Issues, is a type of issue involving library inconsistencies. This was particularly observed as an issue in the Mac OS setup where the SSL module could not be configured because it could not be found. Figure 13 below shows the issue in detail and the error message associated with it.

Fig. 13. SSLError in the Mac terminal

Category 5: DNE ROS Documentation Guide

Category 5, DNE ROS Documentation Guide, is a type of issue involving nonexistent parts of the installation process. This was observed in both the Noetic setup for Mac OS and the Kinetic setup for Windows. In both operating systems, there was no documentation or installation, rendering setup impossible.

Category 6: Software Installation Issues

Category 6, Software Installation Issues, involves issues pertaining to the installation of the actual software. This was the case for both the Noetic and Melodic setups for Windows. Specifically, the issue was found first when installing Visual Studios and later solidified when a future step could not proceed as intended.

The first problem with installing Visual Studios was the required version. The installation instructions specified to use the 2019 version of Visual Studios. Although the link provided led to the correct installation with older versions available underneath, clicking on the download button for one of the older versions leads to a sign-in page requiring users to sign in to be able to install older versions. However, after signing in and attempting to install the 2019 version again, the only allowed version to install was the 2022 version. Whether installation of older versions is limited to account subscription type is unknown. Regardless, this in itself is a limitation due to the fact that it is not friendly for users who are only looking to install the software for ROS.

The second problem involves the optional configuration of a ROS terminal window. Despite being optional, it would be a hindrance to users who do want to do the optional installation. In this substep, the instructions indicate to add a new profile block for ROS, and a code segment was included without further detail on what to do with it in the newly created block. If someone were attempting to follow this step-by-step, they would not know to manually input the different components into the profile block's attributes. This combined with the Visual Studios version discrepancy later became an issue when trying to open a new window of the custom-made ROS profile block. As seen in Figure 16, the expected file path does not match the installed version of Visual Studios. Because of this, the ROS profile block was unable to be configured properly.

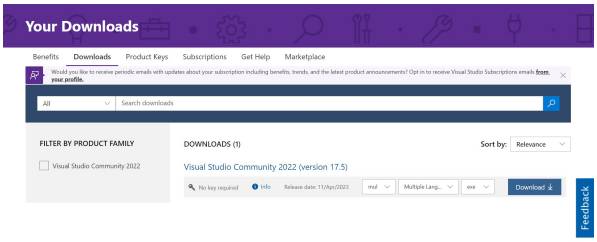

Fig. 14. Downloads tab on the Visual Studios account page

```
    "profiles" :
    {
        list:
        [
            ...
            {
                "commandline" : "C:\\Windows\\System32\\cmd.exe /k \"C:\\Program Files (x86)\\Mi
crosoft Visual Studio\\2019\\Community\\Common7\\Tools\\VsDevCmd.bat\" -arch=amd64 -host_arch=am
d64 && set ChocolateyInstall=c:\\opt\\chocolatey&& c:\\opt\\ros\\noetic\\x64\\setup.bat",
                "guid" : "{xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx}",
                "icon" : "ms-appx:///ProfileIcons/{0caa0dad-35be-5f56-a8ff-afceeeaa6101}.png",
                "name" : "ROS Noetic",
                "startingDirectory" : "c:\\ws"
        },
```

Fig. 15. Block of optional code to configure in the custom ROS terminal

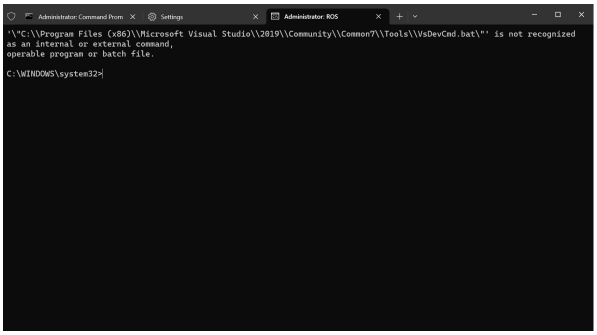

Fig. 16. Visual Studios version discrepancy after opening custom-made ROS terminal

Category 7: Custom Configuration Issues

Category 7, Custom Configuration Issues, is a subset of issues to Category 6, Software Installation Issues. This category of issues focuses on troubles encountered when configuring and adjusting custom settings. This was observed when creating the optional ROS profile block for both Noetic and Melodic installations on Windows. Part of the substep instructs the user to generate a GUID from the Visual Studios terminal and paste it into the user-created ROS profile. There was no place to paste in the GUID since the input box did not exist in the configuration settings.

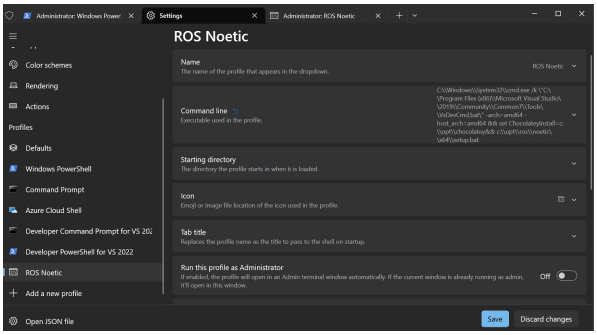

Fig. 18. Settings page for Windows Terminal

Results Summary

Figure 4 shows the counts of questions by category. As shown in Figure 4, Category 1: Dependency Issues and Category 6: Software Installation Issue tie for the most common type of issue.

TABLE I.

Number of Issues Per Category						
Category 1	*Category 2*	*Category 3*	*Category 4*	*Category 5*	*Category 6*	*Category 7*
3	2	1	1	2	3	2

Fig. 4. Issues Per Category

Figure 5 and Figure 6 show a count of the ROS installation issues for the overarching OS and Distro for all the issues. Figure 5 also shows how issue categories 2, 5 and 6 cut across different ROS distributions and issue categories 1, 2, 4 and 7 stay within one distribution. Figure 6 shows that issue category 5 cuts across both operating systems while categories 1, 2, 3 and 4 pertain to macOS and categories 6 and 7 pertain to Windows. The distribution of issues in Figure 6 shows how the majority of ros installation challenges are OS specific.

TABLE II.

Number of Issues Per Distribution		
Melodic: 5 Issues	*Kinetic: 4 Issues*	*Noetic: 5 Issues*
C1, C2, C6	C2, C3, C4, C5	C5, C6, C7

Fig. 5. Issues Per Category

TABLE III.

Category Types Per OS	
macOS: 8 Issues	*Windows: 6 Issues*
C1, C2, C3, C4, C5	C5, C6, C7

Fig. 6. Issues Per Category

VI. DISCUSSION

Our process of coming up with the categories began with individual categorization. Each person recorded their way of grouping the issues together before coming together and discussing the reasonings, as well as coming to a consensus of a general grouping convention. At the end of discussion, we ended up with a total of seven categories. Here, we will discuss a few key observations noted while identifying and categorizing the issues.

All the dependency errors from Category 1 and the syntax errors from Category 2 were both from the same operating system, macOS, and resulted from the same step in the installation instructions, Step 1.3 Install wxPython. The category 1 and 2 issues for this step pertain to both the Melodic and Kinetic installation guides. The installation guides both advise Python 2.7 which is the Python version that was used in our experiments. The syntax issues were encountered first. These issues allude to possibly a Python versioning mismatch in the packages which may mean Python 3 is required. When Step 1.3: Install wxPython was executed with python 3 the syntax errors were not encountered, instead category 1 package dependency errors were displayed. Some deeper investigation is needed to figure out the root cause of these issues if it is the wxPython library itself, its dependencies or a conflict with both depending on different versions of Python.

The Category 3 issue has instructions on stack overflow on how to get the non-standard formulae regardless of the homebrew/science tap deprecation. Open source contributors to the ROS installation guides should update the macOS Kinetic installation guide to reflect the new solution and steps that need to be taken for installing the

non-standard formulae packages.

Something to note is that for Windows, both the Noetic and Melodic versions require Visual Studios 2019. However, the link redirects to the most updated 2022 version with a few links below for the older versions. Upon clicking on the installation link for the 2019 version, it prompts a sign-in. After signing in and clicking the 2019 installation link again, it redirects to a downloads page which only contains Visual Studios 2022 and not any of the previous versions.

Overall, we were not able to complete any installation for any of the distributions on Mac or Windows. In scenarios like this where a user encounters many issues like the ones described in our categories, the best option is to use a virtual Ubuntu machine. Suggestions to combat these issues could include better documentation of the ROS installation and providing alternatives in case something does not work (i.e. version discrepancies). Since ROS is open-source software, improvements could be made in regards to version and OS compatibility. Based on the current state, ROS would only run completely ideally on a Linux machine with the least troubles encountered during installation and setup.

VII. CONCLUSION

In this paper, we ran installations of ROS on three different OSes and documented issues encountered through the installation process. Since there were three OSes, each person was responsible for documenting installation issues on their chosen OS. The documentation took place through a shared document. Once the documentation for each OS was complete, we took part in open coding, which involved individually categorizing the issues and then discussing as a group to arrive at a categorization consensus. Through our methodology, we found that Mac had the most issues, followed by Windows. Linux had the least number of issues as it is the most compatible OS for operating ROS on. As such, Linux is the recommended OS for setting up and installing ROS.

However, we have only tested one distribution of Linux, which happened to be an Ubuntu distribution. Linux distributions that do not use Ubuntu have not been tested, so there is the possibility that similar issues may arise in those distributions. This can be a potential next step for the project, and other operating systems, such as older versions of Windows, may also be considered for testing.

Future research for addressing these setup and configuration issues could involve looking into improving installation instructions for each OS and explicitly specifying whether something is compatible or not. Additionally, research into whether there are better alternatives to using virtual machines to use ROS could be done, as virtual machines can sometimes be slow and inefficient for running a development environment.

REFERENCES

[1] “Robot Operating System.” *ROS*, https://www.ros.org/.

[2] “Wiki.” *Ros.org*, http://wiki.ros.org/ROS/Installation.

[3] D. Ezzy, Qualitative analysis. Routledge, 2013.

[4] J. W. Creswell and C. N. Poth, Qualitative inquiry and research design: Choosing among five approaches. Sage publications, 2016

[5]“Open Coding Analysis.” *Events | University of Phoenix Research Hub*, https://research.phoenix.edu/content/research-methodology-group/open-coding-analysis#:~:text=Open%20coding%20is%20the%20systematic,to%20see%20how%20they%20interplay.

[6] “Wiki.” *Ros.org*, http://wiki.ros.org/Distributions.

[7] Wikipedia contributors. (2023, April 9). Robot Operating System. In Wikipedia, The Free Encyclopedia. Retrieved 20:11, April 16, 2023, from https://en.wikipedia.org/w/index.php?title=Robot_Operating_System&oldid=1148987876

[8] A. Bala, “Why does the ROS2 distro need to be pinned to a specific Ubuntu LTS?”. ROS Discourse. https://discourse.ros.org/t/why-does-the-ros2-distro-need-to-be-pinned-to-a-specific-ubuntu-lts/17435 (accessed April 16, 2023).

[9] “Error: Homebrew/Science Was Deprecated. What Should I Do?” *Stack Overflow*, https://stackoverflow.com/questions/49104856/error-homebrew-science-was-deprecated-what-should-i-do .

[10] White, R., Christensen, H. (2017). ROS and Docker. In: Koubaa, A. (eds) Robot Operating System (ROS). Studies in Computational Intelligence, vol 707. Springer, Cham. https://doi.org/10.1007/978-3-319-54927-9_9

[11] C. Boettiger, An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 1 (January 2015), 71–79. https://doi.org/10.1145/2723872.2723882

[12] Insamuel. *Insamuel/ROS-install-project*. GitHub. Retrieved April 22, 2023, from

https://github.com/insamuel/ROS-install-project
[13] Wikipedia contributors. Open coding. Wikipedia, The Free Encyclopedia. January 9, 2023, 10:03 UTC. https://en.wikipedia.org/w/index.php?title=Open_coding&oldid=1132538880.
[14] Wikipedia contributors. Usage share of operating systems. Wikipedia, The Free Encyclopedia. April 21, 2023, 13:47 UTC. https://en.wikipedia.org/w/index.php?title=Usage_share_of_operating_systems&oldid=1151031702.
[15] P. Taylor, Global market share held by operating systems for desktop PCs, from January 2013 to January 2023. Statista. Feb 27, 2023. https://www.statista.com/statistics/218089/global-market-share-of-windows-7/