Institute of Mechanism Theory, Machine Dynamics and Robotics

RWTH AACHEN UNIVERSITY

This thesis was submitted to the Institute of Mechanism Theory, Machine Dynamics and Robotics

Cross-Compiling ROS2 Humble to WebAssembly

Master Thesis

by:

Isabel Paredes B.Sc.

Student number: 415723

supervised by:

Dipl.-Ing. Martin Mustermann

Examiner:

Univ.-Prof. Dr.-Ing. Dr. h. c. Burkhard Corves

Prof. Dr.-Ing. Mathias Hüsing

Aachen, 31 March 2023

Master Thesis

by Isabel Paredes B.Sc.

Student number: 415723

Cross-Compiling ROS2 Humble to WebAssembly

The issue will be inserted here after being drafted and provided by the supervisor beforehand. The issue should contain a detailed list of all work packages. It should not exceed one page and the version handed to the students has to be signed by the professor.

Supervisor: Dipl.-Ing. Martin Mustermann

Eidesstattliche Versicherung

Isabel Paredes Matrikel-Nummer: 415723

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Master Thesis mit dem Titel

Cross-Compiling ROS2 Humble to WebAssembly

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 31 March 2023

Isabel Paredes

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vor-schriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 31 March 2023

Isabel Paredes

The present translation is for your convenience only.
Only the German version is legally binding.

Statutory Declaration in Lieu of an Oath

Isabel Paredes Student number: 415723

I hereby declare in lieu of an oath that I have completed the present Master Thesis titled

Cross-Compiling ROS2 Humble to WebAssembly

independently and without illegitimate assistance from third parties. I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 31 March 2023 Isabel Paredes

Official Notification:
Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whosoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1)If a person commits one of the offences listed in sections 154 to 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly. I have read and understood the above official notification:

Aachen, 31 March 2023 Isabel Paredes

Contents

List of abbreviations

General abbreviations

GUI	Graphical User Interface
LIFO	Last In, First Out
ROS	Robot Operating System
UI	User Interface
URDF	Universal Robotic Description Format
WASM	Web Assembly

1. Introduction

1.1. Robot Operating System 2

Robot Operating System (ROS) 2

1.2. Motivation

Web Assembly (WASM)

2. Literature Review

2.1. State of the Art

2.1.1. ROS on Web

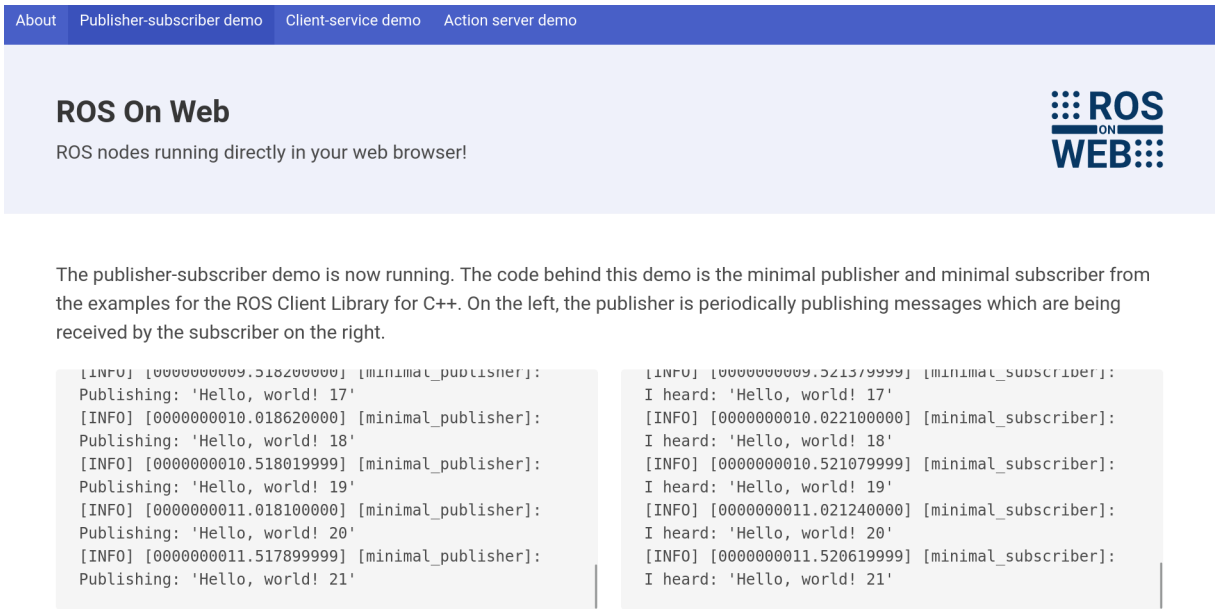


Figure 2.1. *ROS on Web* publisher and subscriber demo

Advantages and disadvantages

Not open source

ROS1 or ROS2

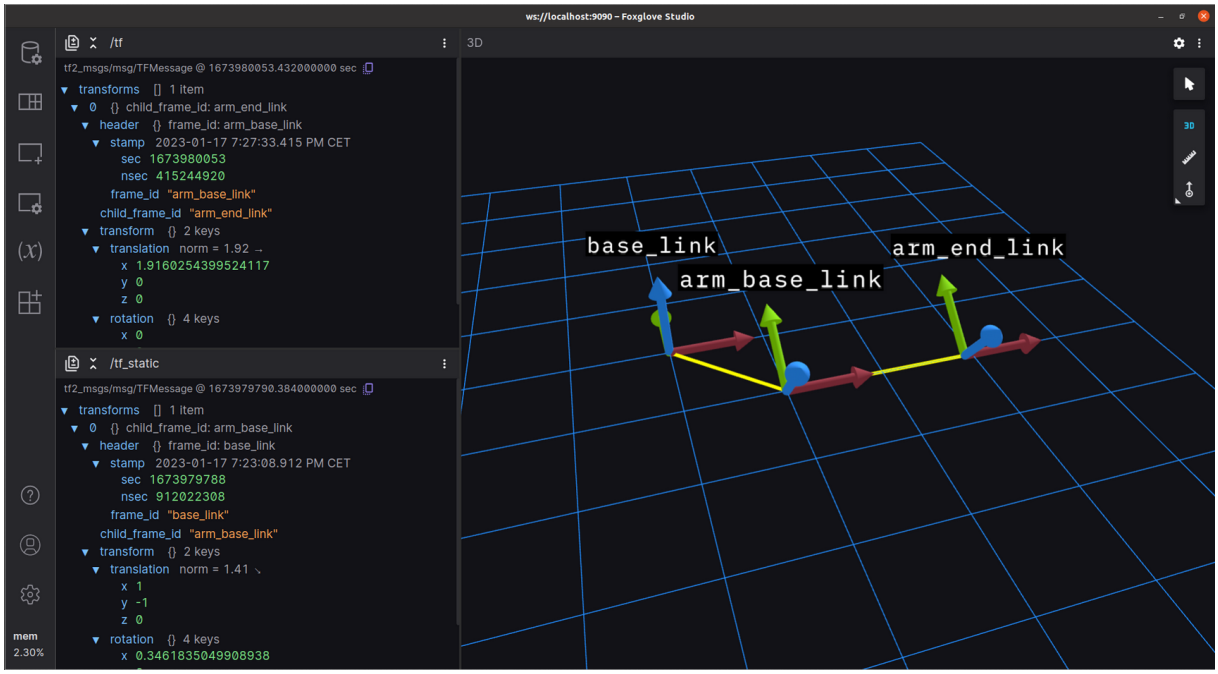


Figure 2.2. Visualizing ROS 2 Transforms with Foxglove Studio

2.2. Relevant Works

2.2.1. ROSbridge

2.2.2. ROS Control Center

2.2.3. ROSboard

2.2.4. ROSlink

2.2.5. Foxglove Studio

2.3. State of WASM

2.3.1. Unity in WebAssembly

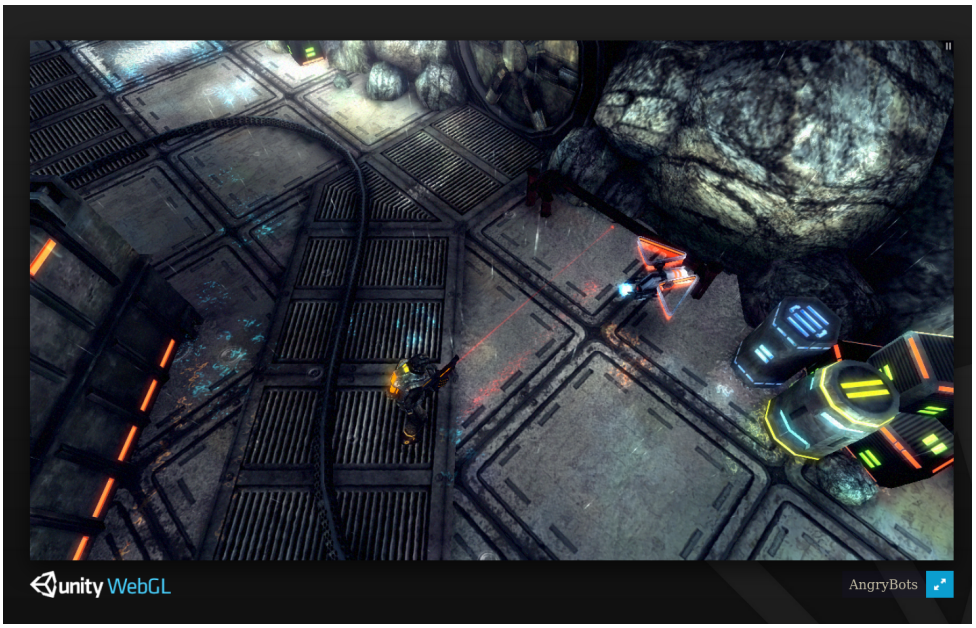


Figure 2.3. Demo of Angry Bots in Unity WebGL

3. Concept Realization

This section provides the major milestones from the project beginning with a brief description of the overall concept solution to the challenges presented in the Introduction, followed by the layers of implementation accomplished during the development phase.

3.1. Target Scenario

To introduce the concept, a "target scenario" is first considered. In this scenario, an intermediate ROS user should be able to reach a high level of usability with the tools developed in this project. First, an intermediate user is described as an individual who is familiar with the ROS ecosystem but does not have the need to maintain or test ROS packages across different platforms. In the target scenario, this intermediate user will be capable of performing the following tasks:

- install pre-compiled ROS 2 packages in the browser
- launch nodes including publishers, subscribers, servers, and clients
- interact with the environment to obtain information about running nodes, this would include echoing topics, listing parameters, reviewing log files, etc.
- visualize Universal Robotic Description Format (URDF) files, transforms, point clouds, markers, etc.
- play and record bag files
- connect with robots via bluetooth

Outside of this scenario, another goal for this project includes making the developed tools available to the general public by distributing them as open-source software. This will allow other roboticists to compile their own packages and share them on the web.

3.2. Implementation Layers

The development of this project is subdivided into multiple levels for the users, the interactions that the users have with the tools developed, and the technical difficulty of developing the tools. These subdivisions are beneficial in providing the reader with an illustration of the progressing stages of development of this project.

Note

If the reader would like to follow along with the demonstrations provided in the following pages, it is recommended to visit ros2wasm.dev. Throughout the text, links will be provided to redirect the reader to specific examples.

3.2.1. User Levels

For the purpose of establishing target users for the developed tools, potential users were categorized based on expertise level with ROS and programming in general. A summary of these levels can be observed in Table 3.1.

Table 3.1. Target users categorized by expertise level.

	User	Description
1	Beginner	Complete beginners who have never used ROS or programmed in any language.
2	Student	University students with basic programming experience.
3	ROS User	Students and researchers who actively use ROS for projects.
4	Roboticist	Robotics software developers including contributors to the ROS ecosystem.

Commencing with Level 1, the *Beginner* category is reserved for students in secondary education who have had little to no experience with programming, and therefore are not familiar with ROS. The tools developed in this project would serve as an initial introduction to robotics for this category of users.

Level 2 consists of university students who have completed elementary programming courses but have not yet been introduced to ROS. For this type of user, this project will provide essential tutorials to become acquainted with the inner workings of ROS.

With a slightly higher level of expertise, Level 3 comprises students or other enthusiasts who are already familiar with ROS and have collaborated in projects which use ROS as the main system to handle communications of multiple robotics elements. This ROS user is equivalent to the intermediate user described in the target scenario (Section 3.1).

Lastly, the highest level of experience is dedicated to roboticists who actively use ROS and contribute to its development. For this category of users, the intention of this project will be to involve more contributors in order to more promptly meet the needs of most ROS users.

3.2.2. Interaction Levels

The Graphical User Interface (GUI) is an essential element in the development of this project because it determines the benefits the users will receive by utilizing these tools. Similarly, the interface the user experiences with the tools has been categorized in increasing levels of interaction. These categories are summarized in Table 3.2.

Table 3.2. User Interface (UI) segmented based on the level of interaction.

	Interface	**Description**
1	Non-interactive	A publisher runs automatically as soon as the site is loaded.
2	Minimal	User can start/stop a publisher by pressing a button.
3	Basic	User can select and run publisher and subscriber nodes simultaneously.
4	Intermediate	Publishers, subscribers, and services are available, and the user can request basic information about the environment.
5	Advanced	A complete GUI where the user has full control of the environment, can start/stop nodes, modify parameters, manage bag files, and visualize robots.
6	Complete	All ROS 2 features are available and packages can be built on the browser, plus the user can directly connect and interact with external robots.

As the name implies, the *non-interactive* Level 1 does not offer the user any sort of interaction with the ROS environment. With this non-interactive interface, the user can do nothing more than load and reload the page. As soon as the user loads the page, a node which has been pre-compiled will automatically start running. In the simplest case scenario, a publisher node would run and the published messages would be displayed on the window. Because the user has no ability to interact with the environment, this publisher node will continue to run uninterrupted. The scenario described is illustrated in Figure 3.1.

Figure 3.1. TODO: Level 1 image

Example 1

A demonstration of a *non-interactive* user interface (Level 1) can be found at ros2wasm.dev/pages/demo01
NOTE: The page must be reloaded to restart the node.

By marginally expanding the interface, the *minimal* Level 2 provides the user with the ability to start and stop a pre-compiled node. This stage is accomplished by the addition of buttons to the website. Continuing with the example previously described, in this level the user can press a button to start a publisher node, view the output of any published messages, and stop the node at any point. Level 2 is exhibited in Figure 3.2

Figure 3.2. TODO: Level 2 image

Example 2

A demonstration of a *minimal* user interface (Level 2) can be found at ros2wasm.dev/pages/demo02

The *basic* Level 3 offers the user increasingly more control over the environment. In this level, the user has the ability to run more than one node simultaneously. This makes it possible to have publishers sending messages to a particular topic and subscribers retrieving the published messages accordingly. Nonetheless, the nodes are still pre-compiled and thus the user does not have the ability to change the topic names or any other parameters of the nodes. Figure 3.3 demonstrates a snapshot of Level 3.

Figure 3.3. TODO: Level 3 image

Example 3

A demonstration of a *basic* user interface (Level 3) can be found at ros2wasm.dev/pages/demo03

Stepping further into the list, the *intermediate* Level 4 introduces services, these include both the servers and the clients. Additionally, with this level the users now possess the ability to request information from the environment such as the type of nodes which are running at a given time, or which topics are available. A depiction of Level 4 is shown in Figure 3.4

Figure 3.4. TODO: Level 4 image

Example 4

A demonstration of an *intermediate* user interface (Level 4) can be found at ros2wasm.dev/pages/demo04

Finally arriving at the *advanced* Level 5, this level is more prominent because it introduces the integration of JupyterLite with the ROS environment. With JupyterLite, the user can directly interact with the ROS environment by using the ROS client libraries such as `rclpy`. A typical workspace in JupyterLite is pictured in Figure 3.5.

Figure 3.5. TODO: Level 5 image

Example 5

A demonstration of an *advanced* user interface (Level 5) can be found at ros2wasm.dev/pages/demo05

Lastly, Level 6 consists of a *complete* ROS environment also made available to the user through JupyterLite. The user would be able to install any additional ROS packages from `emscripten-forge`. An extension for JupyterLite could enable communications with external robots by using the Web Bluetooth API. And a web compiler could be implemented to directly build packages within JupyterLite.

3.2.3. Technical Levels

Table 3.3. Implementation categories with increasing technical complexity.

Level	Description
1	A publisher is displayed.
2	A publisher and subscriber can communicate with each other.
3	Multiple nodes and distinct topics can run simultaneously.
4	Graphical display and interaction with a ROS client library.
5	Manipulation of a physical robot via bluetooth or wifi.
6	Visualization of a robot with Zethus.
7	Simulation of a robotics scenario with Gazebo.
8	Development workspace for creating and debugging ROS packages.

4. Methodology

- Development environment - Building tools - Testing tools (chrome, firefox)

4.1. Development Environment

4.2. Cross-Compilation Tools

4.3. Testing Environment

5. Middleware Implementation

- What does the middleware do? - ROS supported middleware implementations - Why it needs to be replaced - Minimal implementation (minimal set of functions) - Design of middleware packages (tree diagram or something)

5.1. DDS Middleware

5.1.1. FastDDS

default

5.1.2. Eclypse

5.1.3. Gurum

5.2. Custom Middleware

5.2.1. Email

5.2.2. Zenoh

5.3. Substituting ROS 2 Middleware

At run time

At build time

5.4. Custom Middleware Design

6. Package Building Process

- Emscripten - Colcon - Toolchains

6.1. Environment

6.2. Tools

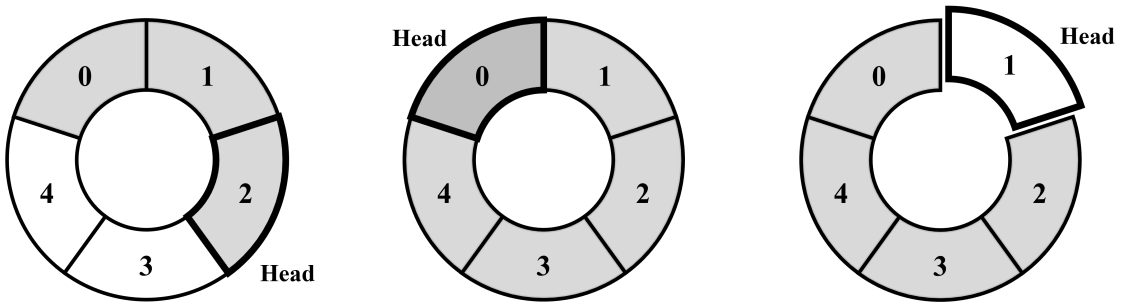
6.3. Post Processing

7. Design of Web Elements

7.1. Web Workers

7.2. Message Stacks

Circular Stack Last In, First Out (LIFO)



(a) Filling stack **(b)** Overwriting stack **(c)** Reading stack

Figure 7.1. TODO:

- Web workers, what are they? why are they needed? - Communication channels - Registry of topics/subs/pubs - Message handling width

8. Package Management and Distribution

- Automating package building - robostack?

9. Concept Assessment

- Survey - Performance measures - Limitations

10. Summary

11. Outlook

- Compiling on the browser - Packaging Gazebo - WASI

List of Tables

List of Figures

A. Illustrations

B. Tables