Project: WebAssembly for ROS Packages

|  |  |
| --- | --- |
| **Date:** | 02/04/2023 |
| **Student Engineer:** | Michael Batchelor |
| **Student ID:** | N9991000 |
| **Supervisor:** | Dr. Tobias Fischer |

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Changes/Comments |
| 1 | 02/04/2023 | Michael Batchelor | Initial work on document |

General Objective

This project will build on top of previous work done in the RoboStack project. RoboStack is a cross-platform bundling of Robot Operating System (ROS) which uses the Conda package manager. It implements a continuous integration/continuous deployment pipeline (CI/CD) which aims to produce consistent, reproducible builds of ROS packages for Linux, Mac and Windows. The goal of this project is to add a WebAssembly (WASM) as a compile target to RoboStack to produce WASM binaries for ROS packages. To that end we will investigate how can we modify the existing RoboStack framework to cross compile ROS packages to *Web Assembly*.

Key Finding from the Literature - Impact on Your Project Design

*[Here you should succinctly state a key takeaway / most relevant learning from your review of literature (which will be included as an appendix)].*

* Emscripten-forge (similar project aimed at a different set of packages).
  + Does not use robostack but does use some of the same tools as robostack

Stakeholders & Resources

The end users of the project will be individuals or organisations using ROS to develop robotics software. Specifically, those interested in interacting with ROS packages/applications in a browser.

The major stakeholder in this project is Dr. Tobias Fischer. He is the project supervisor and a major contributor to the RoboStack project and so will be informed of progress and consulted when necessary. There are also numerous contributors to RoboStack, who may have a vested interested in the outcome of this project, and how it may impact the existing systems.

The resources required to complete the project are,

* Access to RoboStack and its source code.
* Access to ROS packages and their source code.
* A workstation for development/testing. The preferred OS would be Linux, but the project can be completed on any platform due to the cross-platform nature of WASM.
* WASM build tools (e.g. Emscripten).
* Anaconda package repository
* Git repository with CI/CD capabilities. This can be either local or remote and is needed to ee

Project Methodology

*[Describe what methodology and methods you will use to conduct your capstone research project. When considering your approach to project methodology, you may wish to revisit your learning from EGH404 Research in Engineering Practice. Consider what data, information, or measurements you may collect and how you will analyse and synthesise that into a final outcome.]*

Deliverables

|  |  |  |
| --- | --- | --- |
| Deliverable | Description | Date |
| Progress Report | A progress report delivered midway through the project. This will inform stakeholders of progress made, any challenges, next steps, as well as any changes in scope that differ from what is detailed in this proposal. |  |
| Progress Presentation | A short presentation summarising the details outlined in the Progress Report for the benefit of the stakeholders. |  |
| Setup and Usage Guide | This document will detail the setup process, any requirements/dependencies, and the intended usage of the final deliverables. |  |
| Design Document | This document will detail design decisions made during the project. This is to aid future maintainers in understanding the project. |  |
| Modified Vinca build tool | A modified Vinca tool will be delivered that can produce the build scripts necessary for cross-compiling ROS packages to web-assembly. |  |
| Modified RoboStack | A modified version of RoboStack will be delivered that uses the modified Vinca tool to run builds that cross-compile ROS packages to Web Assembly. |  |

Risks, Requirements & Constraints

*[Identify any existing systems that must be interfaced with. Are there any regulatory constraints? What safety and ethical concerns will have to be considered? Are there limits to the scope of your project because of dependencies?]*

*[Identify any risks including the likelihood and consequence and specify how you will mitigate or monitor that risk]*

**Requirements**

* Integrate into existing build infrastructure by making use of RoboStack build processes and pipelines.
* Produce WebAssembly binaries from ROS packages.
* Deploy binaries by uploading built binaries to Anaconda repository.

**Risks**

* Damaging existing packages in the Anaconda repository.
  + Will interrupt user’s workflow e.g., they may need to revert to earlier versions or wait for a fix to be published. This results in a time, and potentially monetary cost.
  + Can be mitigated by working in a fork of the RoboStack repository and only using a local Conda channel. Any catastrophic events will be restricted to the dev machine.

**Constraints**

* Minimal impact on the existing build systems.
  + We don’t want to break Linux support to add WebAssembly

**Safety & Ethical Concerns**

* Deploying a faulty package could be dangerous if it is used in a critical system.
* Deploying packages with security vulnerabilities. It could be worth considering the impacts of continuing to host packages with known security vulnerabilities. ROS packages are generally used in tightly controlled systems but there are cases where security can be a big concern. The harm done by these vulnerabilities could be minimised by making the packages less available, or at least making sure the vulnerabilities are acknowledged somewhere.
* Licensing concerns (e.g. building/deploying packages with incorrect licensing practices, could be considered unethical and perhaps introduce legal issues). Software licensing can be complex and it is not feasible to vet every package built and published by RoboStack. Making this issue worse there can also be some ambiguity in how the licenser vs licensee interpret the many different software licenses used.

Quality & Sustainability

* How well the solution can be integrated into RoboStack. Was any existing functionality crippled or removed to achieve the goals.
* This solution will be maintained by another team once this project has concluded and the lifecycle of the project is unbound. If the solution is of poor quality or hard to maintain then it may get replaced and the work will be redone. This will essentially result in a waste of resources.
* Compiling software is quite resource intensive. Centralising builds and distributing prebuilt binaries may result in less overall resource usage, increasing efficiency of software development. This not only saves developers time, but also reduces power wasted on recompiling packages over and over.

*[How will you measure the quality of your outcome or deliverables? How will you test the validity of your solution? What is the scope of sustainability that you will consider, i.e. are you only interested in the short term impact or will you have to consider life cycle analysis?]*

Timeline & Deliverables

*[Outline the broad phases of your project and the deliverables that will be achieved at each stage. Include designs, reports, physical systems, etc. The following table contains some examples of what might appear in this table. NOTE: This plan should extend across both EGH400-1 and EGH400-2]*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Focus | Deliverable | Dependency | Release Date/ Milestone |
| 1 | Literature Review | Literature Review Report |  |  |
| 2 | Conceptual design |  | 1 |  |
| 3 | Sub project 1 |  |  |  |
| 4 | Sub project 2 |  |  |  |
| 5 | Detailed design |  | 2 |  |
| 6 | Modelling/ Rapid prototype & experiment |  | 5 |  |
| 7 | Interim Report |  | 6 |  |
| 8 | Pilot |  |  |  |
| 9 | Build/ Construct |  |  |  |
| 10 | Test & Validate/ Verify |  |  |  |
| 11 | Draft report |  |  |  |
| 12 | Stakeholder consultation |  |  |  |
| 13 | Project Delivery Report & Oral Defence |  |  |  |

Management of Project Changes

*[Outline how you will manage any stakeholder, partner, or supervisor requests for change. Note if there are changes as the project progresses, then you should update this scope and update the version outlining the nature of the change]*

Sign off

|  |  |  |
| --- | --- | --- |
|  | Signature | Date |
| Student Engineer |  |  |

Appendix: Review of Literature

## Introduction

This review of literature is investigating the current state of Web Assembly (WASM) as it relates to robotics and embedded systems. The main points of interest are recent attempts to use WASM in this context, and general advantages/disadvantages of web assembly as a compile target. The focus of this project is compiling ROS packages to Web Assembly, so we will also look at the current state of work relating to this.

## Key themes

**RoboStack**

This paper provides a basis for the work that will be done throughout this project. It introduces RoboStack, which tightly couples ROS with the conda package manager. This is in an effort to enable reproducable builds with robust management of dependencies.

The main advantages gained are; reproducable environments with robust management of versions, better compatibility with different platforms and linux distros, distrubution of pre-built binarys, isolation of environments, and simple creation of packages.

**Ros On Web**

This project hosts demos of ROS nodes running in a web browser and was achieved by cross-compiling ROS nodes to Web Assembly. The demos are based on example code for the C++ ROS client library. This project demonstrates the possibility of having ROS nodes compiled to Web Assembly. This is done with the hope to enable developers to share their ROS systems securely, without the need to install dependencies locally.

The three demos show the publishers and subscribers, clients and services, and action server ROS concepts running in the browser.

**Robots as Web Services**

Describes some interesting use cases that emerge when ROS can run in a browser. These use cases seem to be applicable to our project as with ROS2 cross-compiled to web assembly, we should be able to develop ROS nodes that can run in a web-browser.

The advantages of bring robotics software to the web is allowing researchers and users to access robots via the internet. This would enable a variety of web interfaces for phones and computers, or remote robotics labs for shared experimentation.

This paper also proposes a remote laboratory which could be created using rosjs (and I would also posit with ROS cross-compiled to Web Assembly). This could give researchers access to a common platform for experimentation, and perhaps shared data as well. It may also give some researchers access to expensive, difficult to obtain equipment they may otherwise have been unable to use.

Many of the benefits of the rosjs project are just as applicable to this project as our goal of bringing ROS to the web browser is much the same, just using a more modern technology.

**Emscripten Forge**

Emscripten forge is a project that implements a build pipeline to compile conda forge packages to Web Assembly. It makes use of the same build tools as RoboStack making it a good guide for the kinds of changes that will need to be implemented in RoboStack to cross-compile to WebAssembly.

The point of difference between RoboStack and Emscripten Forge is Vinca. The recipes and build scripts appear to be managed manually by contributors, whereas RoboStack aims to reduce the maintainer burden by automatically generating these files.

**ROSWASM Suite**

This is a set of libraries that can be used to develop ROS nodes that can be compiled to Web Assembly using EmScripten. Using the APIs provided by this project you can write ROS nodes in C++. Once compiled to Web Assembly these nodes are then able to communicate with ROS via a WebSocket opened by `rosbridge\_server`.

This is the same approach taken by to expose ROS to JavaScript via `roslibjs`. It connects to the WebSocket created by `rosbridge\_server` to interface with ROS.

This is an interesting approach to compiling C++ Nodes to Web Assembly, however this project appears to be aimed at developers creating new Nodes, or porting old nodesso they are Web Assembly packages. Our project more general, in that we want to develop a CI/CD pipeline for cross-compiling existing ROS2 packages to Web Assembly.

**Unity on Web Assembly**

This blog post details how the Unity team implemented Web Assembly as a compile target for the Unity game engine. This is quite an interesting endeavour and demonstrates the power of cross-compiling to WASM nicely. They have posted a top down mobile game compiled to Web Assembly that runs in the browser. There are no dependencies to install, it just works.

They briefly describe the toolchain based on IL2CPP, empscripten and binaryen, they use to compile from C# to C/C++ and then to Web Assembly (or asm.js).

Given then complexities involved in building a binary from the unity engine this is promising in terms of our potential success in cross-compiling ROS packages to WebAssembly.

## Conclusion

Write some stuff in conclusion

References

Beier. T., Renou. M., Vollprecht. W., Mabille. J., Tuloup. K., Paredes. I., Brochart. D., Cross. S., Lamotte. J., Traversaro. S. & Hollands. A. (n.d.) *Emscripten forge*. https://github.com/emscripten-forge/recipes

Osentoski. S., Jay. G., Crick. C., Pitzer. B., DuHadway. C. and Jenkins. O. (2011). *Robots as web services: Reproducible experimentation and application development using rosjs.* IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 6078-6083

Fischer. T., Vollprecht. W., Traversaro. S., Yen. S., Herrero. C. & Milford. M. (2022). *A RoboStack Tutorial: Using the Robot Operating System Alongside the Conda and Jupyter Data Science Ecosystems.* IEEE Robotics & Automation Magazine, vol. 29, no. 2, pp. 65-74, June 2022

Allwright. M. (n.d.). *ROS On Web*. https://rosonweb.io

*Ros WASM Suite [Computer Software].* (2023). https://github.com/nilsbore/roswasm\_suite.

Trivellato, M. (2018). *Web Assembly Is here.* https://blog.unity.com/technology/webassembly-is-here