Work plan master's thesis

Title	WebAssembly beyond the browser: capability-based sandboxing in the edge
Name student	…
Email	…
Company/ Research group	…
Promoters	…
Supervisors	…

Breakdown per semester:

	Semester 1	**Semester 2**
# Credits courses	…	…
# Credits thesis	…	…

Existing situation and problem definition

Originally WebAssembly was made to execute binary code alongside JavaScript inside browsers. It allowed developers to compile codebases written in languages like C++ and Rust to a single wasm binary, and then have that file run at near-native speed in the browser inside a secure and isolated sandbox. This was great option to run more demanding operations the JavaScript engines from that time could not deliver.

Because of WebAssembly's features, developers who worked completely outside of the browser started bringing WebAssembly to the server and other places. They were giving these WebAssembly binaries full access to the operating system's system call library, which compromised portability and security.

WebAssembly needed a way to talk to the underlying system – a system interface. Not just a system interface, but an abstract interface, which allowed it to run binaries on every operating system. An effort to create such a standardized system interface has been made, called WASI. Although WASI has grown a lot over the last few years it is by no means complete.

One of the missing APIs is an agnostic GPIO/IC2/SPI API. This API would among other things allow embedded devices to access underlying hardware and communicate with connected peripherals.

During my thesis I want to contribute to WASI by proposing an implementation that supports these use cases. This includes code contributions, attending design discussions, writing a specification, related documentation and more.

Objective of the master's thesis

1. WebAssembly is still very new and rapidly changing. Documentation and examples are often hard to find, incomplete or out of date. The thesis should provide an overview of the essential components of WebAssembly including the new WebAssembly component model and the WebAssembly System Interface.

2. The thesis is mainly focused on WASI. It should provide an overview of the moving parts of WASI and provide at least an answer to the following questions: What are the steps to standardize an API and it to become part of WASI? What is the Wit format? How is Wit used within WASI APIs? What are World Profiles?

3. Contribute to WASI to standardize a platform-agnostic serial interface. This means working together with the community to design a new API and create a wit specification. This might include world profiles depending on the implementation of these features.

4. Wit specifications describe interfaces, eventually, they will need to be implemented across multiple languages. Contribute an implementation in one of these languages.

5. Depending on the time required to contribute a wit specification and develop a technical implementation, additional work can be done. One of the obvious options is to work out an example project where underlying sensor data is accessed from within a native WebAssembly module running on a (range of) embedded device(s). The performance of the WebAssembly module can be compared with other container/VM implementations.

Note: The WASI standardization process may take months or even years. Proposals consist of 5 phases. Currently, there only exist proposals in phase 1 and phase 2. There are other projects, such as WasmEdge, that adopt new proposals faster. This enables the community to use and test them, often these eventually become part of the WebAssembly standard. If the standardization process goes very slow, it might be an option to shift the implementation to another runtime.

Planning and milestones

Task 1	3 weeks	*Deadline:* 29 October 2022	**Literature study and technology exploration**
Content - Search and summarize scientific articles about WebAssembly, WASI, HAL, the WebAssembly component model, and meta computing. - Setting up basic applications in rust.			
Most important results, deliverables, or insights after this phase: - Defined list of research questions I want to answer - Get a better understanding of the WebAssembly jargon - Get in touch with people within the WebAssembly/WASI community - Workplan - Automatic week summary based on commit messages			

Task 2	3 weeks	*Deadline:* 4 November 2022	**Implement baseline**
Content - More literature studies - Demystify the high-level workings of WebAssembly - Demystify The flow of creating a standardized WASI interface			
Most important results, deliverables, or insights after this phase: - Chapter about WebAssembly, including a schema of all parts - Chapter about WASI, including a schema / flow chart			

Task 3	1 week	*Deadline:* 18 December 2022	**Prepare midterm presentation.**
Content - Prepare midterm presentation			
Most important results, deliverables, or insights after this phase: - Give midterm presentation for supervisors and promoters in week of 19/12			

Task 4	8 weeks	*Deadline:* 19 December 2022	**Theoretical implementation**
Content - Conduct more research regarding the actual implementation. Look into existing non-cross platform implementation of accessing GPIO/I2C. The Hardware Abstraction Layer (HAL) - Investigate existing proposals - Create a theoretical implementation of the API and present in the community. Implement feedback.			
Most important results, deliverables, or insights after this phase: - A theoretical implementation of the API. - Chapter explaining the theoretical implementation - Get a better insight into wit format.			

Task 5	3 weeks	*Deadline:* 31 March 2023	**Thesis – First 25 pages**
Content Work on Thesis itself, introduction			
Most important results, deliverables, or insights after this phase: - First 25 pages of thesis - Maybe a chapter about wit (in case it is complex).			

Task 6	9 weeks	*Deadline:* 1 May 2023	**Technical implementation API**
Content The specifics of the technical implementation will depend on the research done in task 4 The specifics of the technical implementation will depend on previous task At this moment in time, I am not sure what this will contains.			
Most important results, deliverables, or insights after this phase: - An accepted phase 1 or phase 2 technical implementation of the API.			

Task 7	2 weeks	*Deadline:* May 2023	**Profiling**
Content - Create native WebAssembly and non-WebAssembly test applications that reads out basic sensor data. Probably a rust application and compiled to WebAssembly version - Profile these applications			

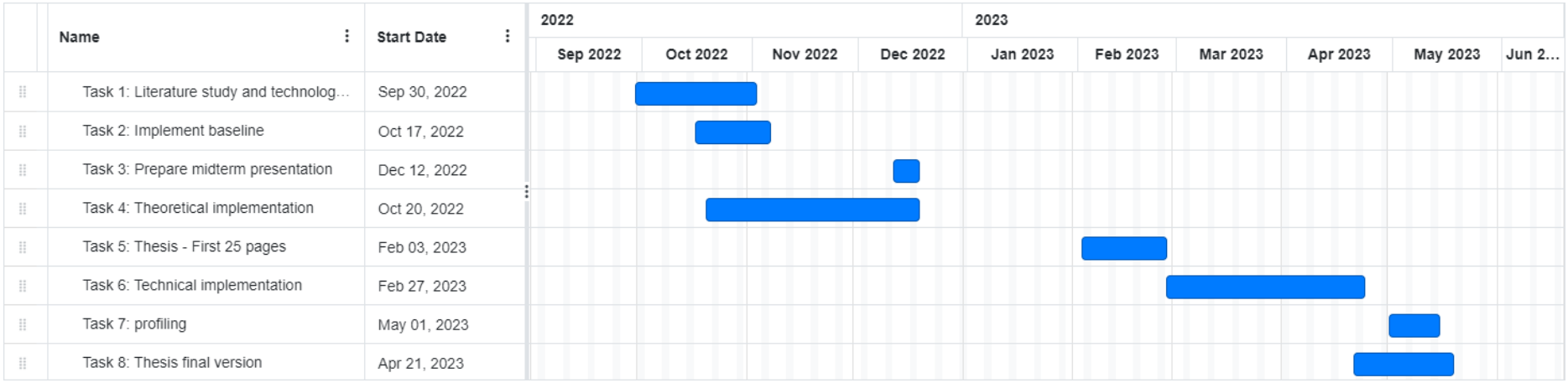
Most important results, deliverables, or insights after this phase:
- Write profile results in thesis

Task 8	2 weeks	*Deadline:* 25 May 2023	**Thesis – final version**
Content Finish final parts thesis, - Make table of all used acronyms			
Most important results, deliverables, or insights after this phase: . - Short abstract describing the master's thesis - One of the final versions (+-95%) of the thesis			

Contact

There is a weekly email reporting the summary of work done over the past week. This contains a list of commit messages, remarks, and non-urgent questions. Other meetings are scheduled as needed. The guideline is to hold a Teams meeting every two weeks to monitor the progression. There is an additional Teams chat that is used as an information hub and to ask more urgent questions.

Gantt chart


Name
Start Date
2022
2023
Sep 2022
Oct 2022
Nov 2022
Dec 2022
Jan 2023
Feb 2023
Mar 2023
Apr 2023
May 2023
Jun 2...
Task 1: Literature study and technolog...
Sep 30, 2022
Task 2: Implement baseline
Oct 17, 2022
Task 3: Prepare midterm presentation
Dec 12, 2022
Task 4: Theoretical implementation
Oct 20, 2022
Task 5: Thesis - First 25 pages
Feb 03, 2023
Task 6: Technical implementation
Feb 27, 2023
Task 7: profiling
May 01, 2023
Task 8: Thesis final version
Apr 21, 2023