# Field Session Summer 2023: WebAssembly Interface for GPIO

Proposed by Andrew Gracey, Lead Product Manager for Cloud Native Edge at SUSE

## About SUSE

SUSE provides services, support, and maintenance for open source platforms. We were the first enterprise Linux and is now a leader in the Kubernetes/Cloud Native ecosystem.

Open Source is core to our company ethos and all our work is done in upstream projects! This project is primarily benefiting the Bytecode Alliance: https://bytecodealliance.org/

## The Project

One of the current research topics for our team is providing support for WebAssembly (WASM) as a first-class workload type.

WebAssembly is quickly gaining popularity in the ecosystem by allowing for a reduction in developer friction when building components for microservice applications (while also providing a very secure baseline platform)

Building on top of a project called SpiderLightning, we would like to provide an WebAssembly System Interface (WASI) for working with the General Purpose Input/Output (GPIO) pins on a variety of devices. The base functionality would be:

- Query pin state
- Function trigger on high
- Function trigger on low
- Configurable pin state at platform initialization
- Configurable debounce time

There are existing libraries to do most of the heavy lifting so much of the challenge (and fun) in this project is learning the existing stack and toolchain.

### Example Use Case

An example use case for this proposed functionality would be to attach a message bus as well and publish events when the pin state changes. In this example, a developer would be able to write a small middleware function then configure that function to trigger on state change.

Similarly, you could reverse the direction and pull a pin high or low based on an incoming message (or only pass through messages between a sink and source based on pin state)

### Potential Milestones:

- Learn toolchain, build project, run on Raspberry Pi
- Explore potential libraries for implementation
  - https://github.com/golemparts/rppal ?
- Specify reference workload for testing against
- Build WIT specification(s)
  - Examples at https://github.com/deislabs/spiderlightning/tree/main/wit
  - Will need to be generic enough to be appropriate across
- Write reference implementation
  - Docs here: https://github.com/deislabs/spiderlightning/blob/main/docs/service-implementation-101.md
- Record demo & open PR for upstream consideration

### Stretch Goals:

- Also write types and wire in
  - I2C (1-wire)
  - RS-232
  - PWM

## Technologies

- Rust
- WebAssembly
- Linux (Specifically SUSE/openSUSE)
- ARM64 / Raspberry Pi
- https://github.com/deislabs/spiderlightning

Optionally (but recommended):

- Containers
  - Via: https://github.com/containerd/runwasi
- Kubernetes

## Team size

I would recommend a team of 3-4 but scope can be expanded for larger teams. Work is fully remote.