WASMCON NA 2024 Retrospect

- Video Playlist
- Event Schedule

fujitatomoya@github / tomoyafujita@linkedin

The Wasm Wilderness

Wasm on the Edge

- Conserving energy, Edge devices are more resource and power constrained
- Scale to zero with Wasm, Wasm has instant `<1ms startup` times
- Wasm multi-tenant, secure, serverless on the edge
- Vastly heterogenous ecosystem, Wasm enables secure, portable, polyglot embedded development

Secure and Efficient Sensing Applications with Wasm

- IoT fragmentation generates non-portable applications
- Lack of isolation a real concern in IoT security
- Embedded development dominated by C/C++, but AI & Data frameworks are mostly Python
- Lightweight Execution, Environment Write Once, Run Anywhere, Secure by Default, Polyglot Programming

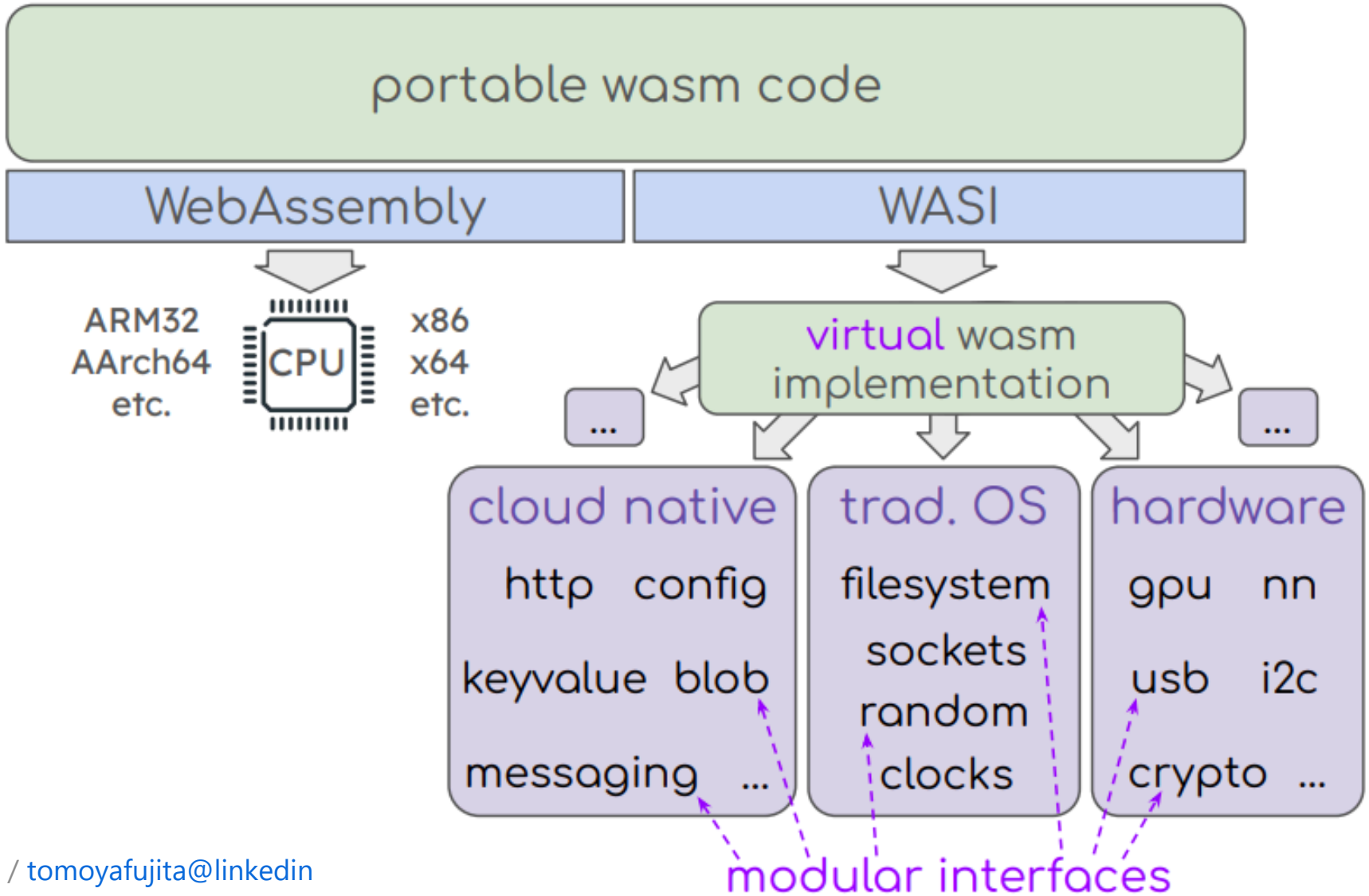
Model	Processor	Flash	RAM
STM32 (STM32F469)	ARM® Cortex® M-4 180 MHz	Up to 2048 KB	384 KB
ESP32 (ESP32-WROVER-E-N8R2)	Xtensa 32-bit LX6 240 MHz	Up to 8 MB	520 KB
Arduino Due (AT91SAM3X8E)	ARM® Cortex® M-3 84 MHz	512 KB	96 KB
NXP i.MX RT700 Cross over MCU	ARM® Cortex® M-33 325 MHz	SD and eMMC interfaces	7.5 MB
Raspberry Pi 5	ARM® Cortex-A76 2.4 GHz	microSD card slot	Up to 8GB

WASI 0.2 to 0.3 and beyond

a virtualized collection of modular interfaces between wasm and the outside world that are in the process of being standardized

- WASI every 2 months development cycle
- WASI 0.3: native async support !!!
- Wasm OCI Artifact layout: representation of a component or module as an OCI Artifact to take advantage of massive existing OCI tooling and cloud infrastructure

WebAssembly System Interface



WASI to Go

- Go 1.21: wasm/wasip1(WASI Preview 1)
  - Single-threaded: calling a go:wasmimport function blocks all goroutines
  - No network or sockets support in wasip1
- wit-bindgen-go: Command-line tool to generate Go from WIT
- TinyGo 0.33.0 released with native support for WASI 0.2
- Other things: wasi-http-go

Wasm and OCI spec

WASM OCI Image Specification defines how to bundle WASM modules as OCI images. WASM OCI Images consist of a WASM binary file, configuration file, and metadata for the target WASM runtime.

- WIT (Wasm Interface Types): text format that is the IDL for Wasm Component.
- Component: Wasm binary that can communicate and be composed with other components.
- WIT Package: Wit text files encoded as a component, primary way for sharing interfaces.

A Walking-Tour of wasm-tools

- CLI and Rust libraries for low-level manipulation of WebAssembly modules
- Open source repository bytecodealliance/wasm-tools
  - There seems to be a couple of similar projects but this is mainline under Byte Code Alliance.

WebAssembly at Google

- can't provide you with the complete view of WebAssembly at Google.
- Google contributes to WebAssembly's standardization.
- Google Wasm projects
  - Emscripten: C / C++ / LLVM to WebAssembly
  - Binaryen: Wasm tools used by Emscripten and many other toolchains (Dart (Flutter), Java (J2CL), Kotlin, and Rust)
- Products
  - Google Earth: ported to Web with WebAssembly.
  - Google Sheets, Google Photos, Google Meet, Skia ...

Google bets heavily on Wasm—visibly (what we shared today) and hidden behind the curtains.

Distributing and Running Containers for Wasm-Enabled Environments

- Leveraging existing apps on browser (dev environment, playground, building block, etc)
- Leveraging Wasm features for running apps outside of browser
- porting apps to Wasm is hard...
- `container2wasm` enables to run containers on Wasm with CPU emulation
- Experimental support of QEMU on browser

Exploring the Landscape for Open Telemetry for Wasm

- Challenges in WASM observability is that WASM does not really know about the host, nor access... like timestamps for clocks and IPC calls...
- Most of OTEL libraries are developing to WASM.
- WASI logging demo
- ObserveSDK adapter exports the telemetry data to OpenTelemetry.

WebAssembly for IoT Devices

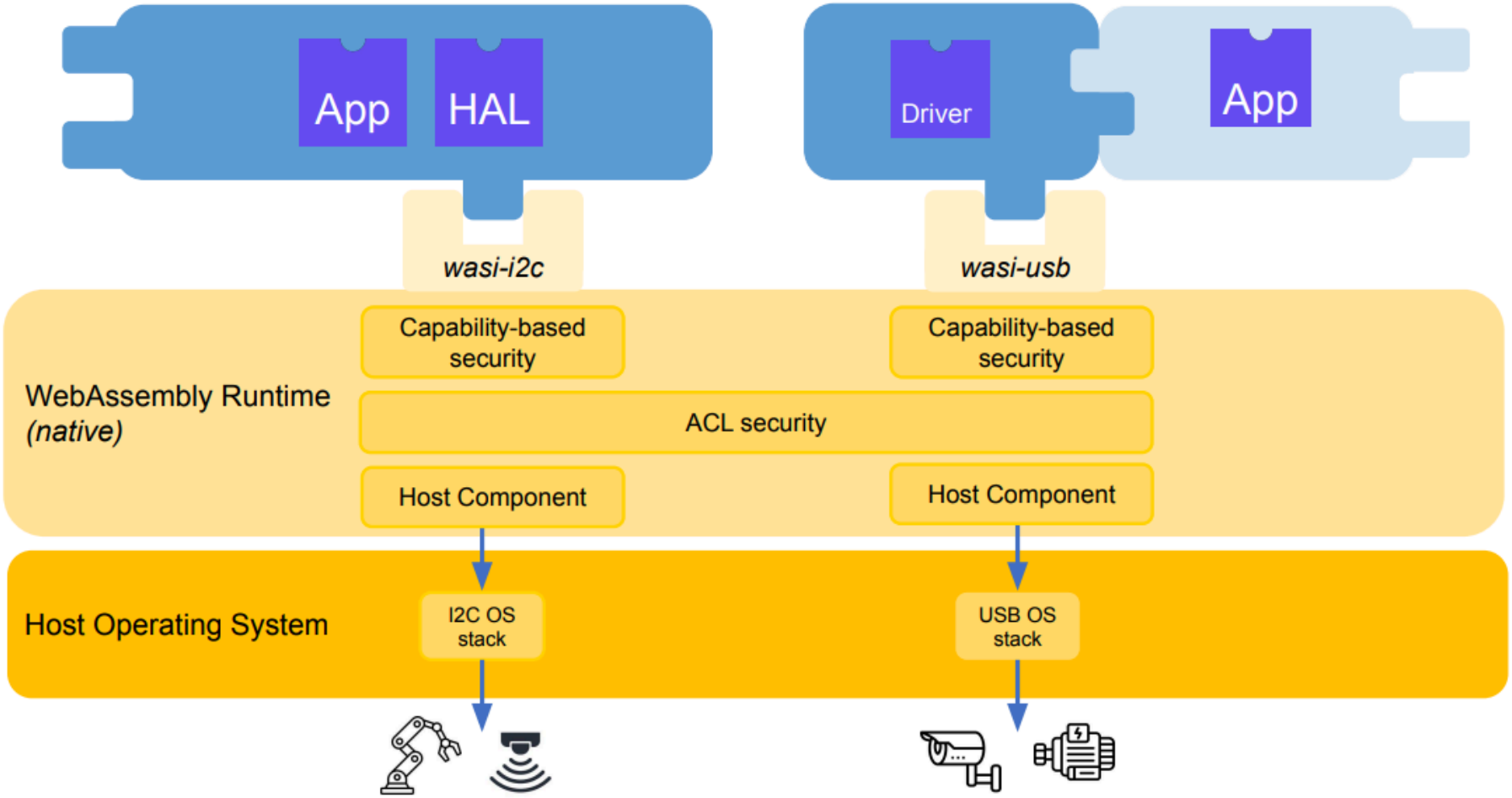
- Interfacing with USB and I2C Hardware
- Average lifespan of cars in Europe is 30 years...

Binary device portability across ISAs (Instruction Set Architectures) and platforms.

Support for more programming languages and language interoperability on embedded devices. Forward compatibility with newer application toolchains over multiple decades

Secure and sandboxed execution of software, where other solutions like containers do not fit.

Hardware WASI interfaces & Componentized drivers



- wasi-usb
- wasi-i2c
- Latency to USB device: +0.007ms
- USB throughput: -0.6%
- W.I.P: GPIO / SPI

COMPONENT MODEL IN SOFTWARE DEFINED VEHICLES

- agnostic from platform, Wasm components fit anywhere including simulation environment.
- Component Model offers graphical composition.
- High level data types Option, Result, String, Vector, Future, Stream for AUTOSAR.
- secured, sandboxed execution runtime.

The WebAssembly Component Model

- warg: WebAssemblyRegistry
- wac: Wasm Composition
- wkg: Wasm package-tools

Work with Components

Authoring

Create Wasm Compoonets.

cargo component

spin new

Composing

Stack Components together by wiring up the export to imports

wac plug

spin

Running

Run or Execute a Component

wasmtime

spin and spinkube

NGINX Unit

Distributing

Share Components

registry OCI / WARG

wa.dev

Object-Storage