Modernizing the Development of AI-Based IoT Devices with Wedge

Dan Mihai Dumitriu

CTO

Midokura (Sony Group)

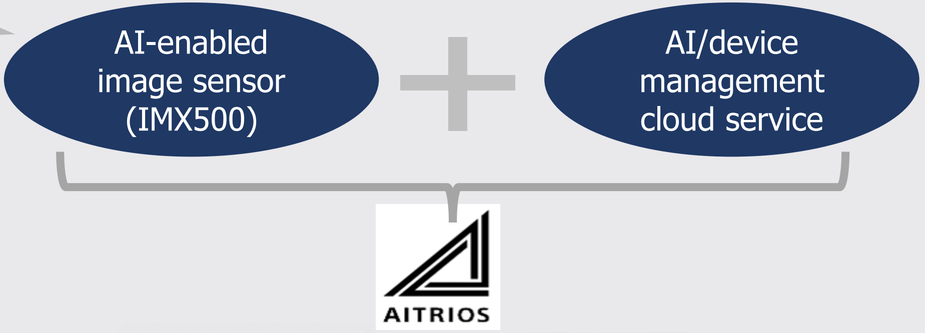
midokura

A Sony Group Company

Introduction

- Sony Semiconductor Solutions is the largest vendor of CMOS image sensors
- AITRIOS is a new business to develop *sensing* solutions based on the sensor portfolio
- Embedded Edge AI and IoT are key ingredients
- Midokura is a subsidiary of Sony Semicon focused on software infrastructure

Sony AITRIOS


AI-enabled image sensor (IMX500)
AI/device management cloud service
AITRIOS

Faster and optimized on-sensor AI processing

Less bandwidth needed and very low latency

Low power and cost friendly

Address privacy concerns by elaborating metadata-only

An Accessible Platform that Enables Intelligent Solutions Based on Distributed Visual Sensing

Targeting solution developers for various vertical applications.

Bring to market a plethora of low cost, easy-to-use sensing devices

Low barrier of entry for solution developers

Agile development of sensing applications

Low operational cost of vision sensing apps

Polyglot Development

Marketplace to connect AI Developers & Solution Developers

Problems of Development on Tiny IoT Devices

midokura ミドクラ
A Sony Group Company

Development is Not Agile

- Operating system (RTOS) is effectively like a library
  - Application and OS must be tested together
- Waterfall development
  - Heavy testing process late in development cycle
- Cannot continuously deploy applications

No Safety and Isolation

- Tiny IoT devices are based on MCUs (microcontroller units)
  - MCU typically cannot do virtual memory
- Without memory isolation, cannot do agile development
  - Dynamic deployment of applications is unsafe

Application Development Is Difficult

- High barrier of entry for application developers
  - Apps are typically written in C
- Poor code reuse across device types
  - HW specific interfaces and drivers
- AI developers typically don't know C well
  - Develop mostly in Python

Introducing WEdge

midokura ミドクラ
A Sony Group Company

Intro to *WEdge*

- Like Kubernetes, but for tiny IoT devices
  - Automated lifecycle management of workloads
  - Strong isolation of modules
- Leverages WebAssembly Micro Runtime (WAMR)
- WEdge cloud does automatic optimization for target device
- Polyglot SDK enables developer productivity

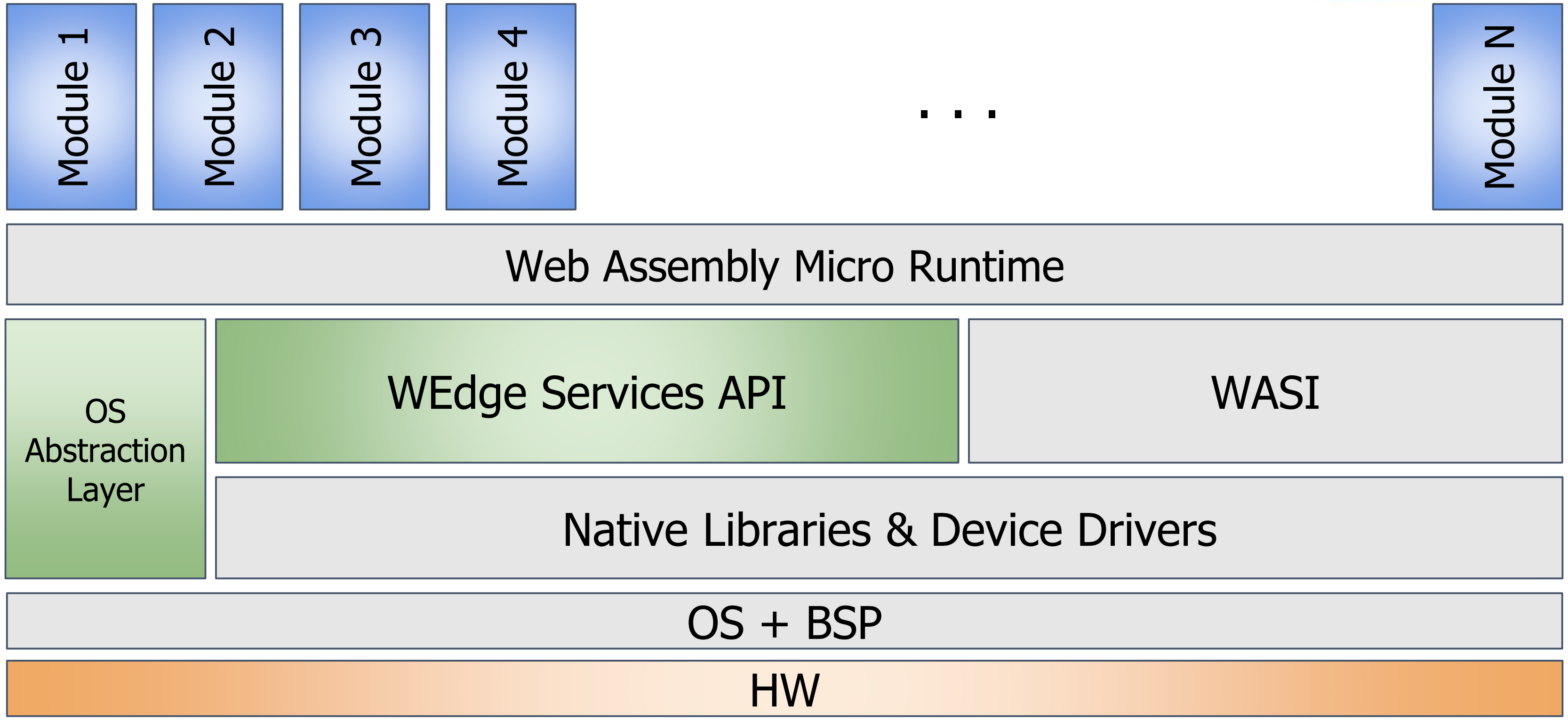
Intro to WebAssembly

- WebAssembly (Wasm) is a low-level bytecode format that runs in a sandboxed environment
- Designed as a portable target for compiling high-level languages like C, C++, and Rust
- Runs on various platforms, including browsers, servers, and now even on tiny IoT devices
- Compact size, fast execution, and high security compared to other runtime environments
- AoT (Ahead of Time) compilation supports multiple target ISAs via LLVM backends
  - High runtime performance - within 2x of native

WebAssembly for Sandboxing

- Combination of language-level and runtime-level protections
- Linear memory model
  - All memory accesses are bounds-checked to prevent buffer overflows or underflows
- Enforces type safety at both compile-time and runtime
- Control Flow Integrity (CFI) prevents control flow hijacking attacks
- WebAssembly System Interface (WASI) for standardized and secure way to access system resources
  - Fine grained control

WEdge Device Stack



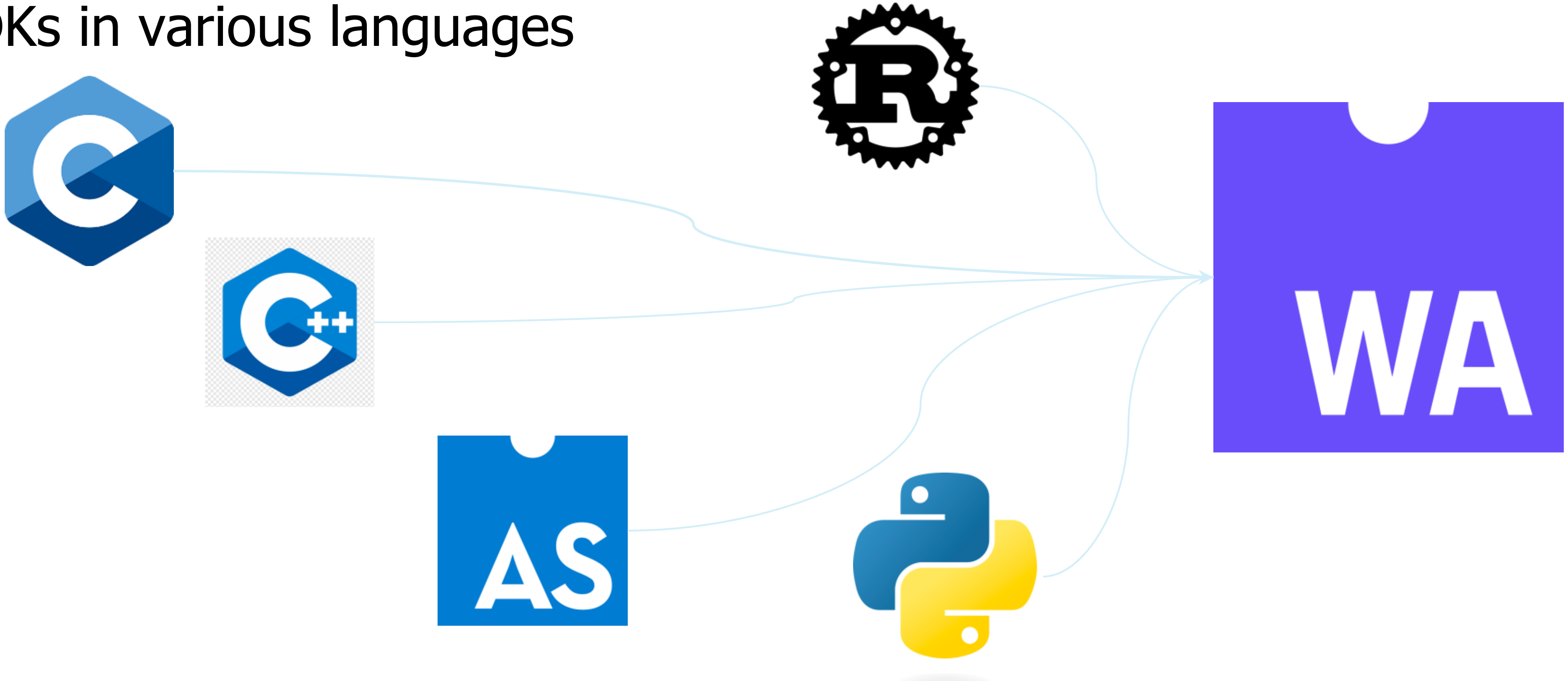
WEdge Services API

- Sensors
  - Read image
  - Configure (e.g. frame rate)
- Communication
  - Send telemetry to cloud
  - HTTP PUT/GET/POST
  - Other protocols (e.g. CAN)

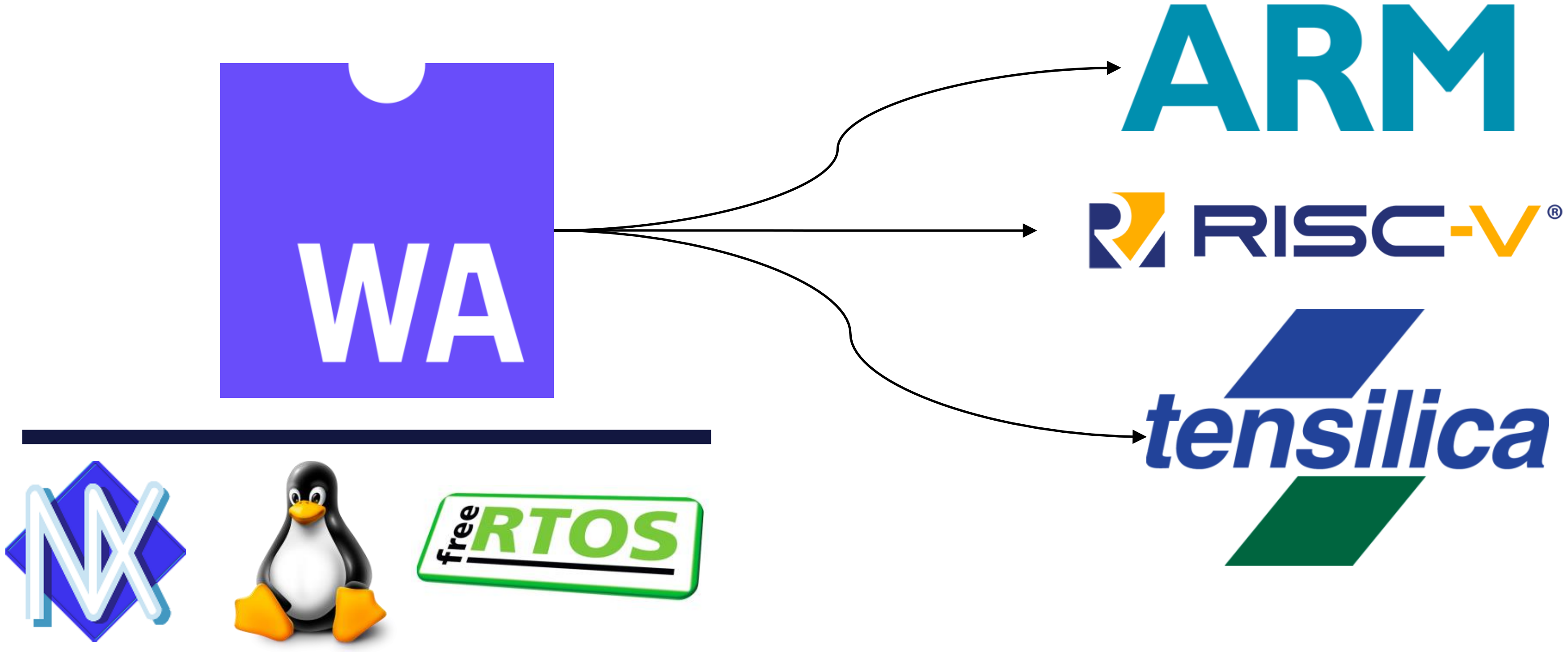
- AI/ML
  - Load model
  - Run inference
- Data storage
  - Local DB
  - Distributed DB cache

Polyglot Development

SDKs in various languages



Decouple from Target Architecture & OS



Agile Development Enabler

- Enables more flexible and dynamic development cycles
- Devs can break down complex software into smaller, more manageable modules that can be developed, tested, and deployed independently
- Polyglot development helps devs code in their language of choice
  - For AI developers, that is Python

Vision Sensing Applications

midokura ミドクラ
A Sony Group Company

Vision Sensing Applications (VSA)

- Programming sensing applications is complex
- Code reuse is typically poor and applications are monolithic

- VSA models a complex application as a series of simple nodes
- Promotes code reuse
- Enables low-code development via API or GUI
- VSA nodes are deployed as Wasm modules

Vision Sensing Application (VSA)

Goal: Make an intelligent sensor perform a complex task

How: Model this complex task as a sequence of individual small tasks

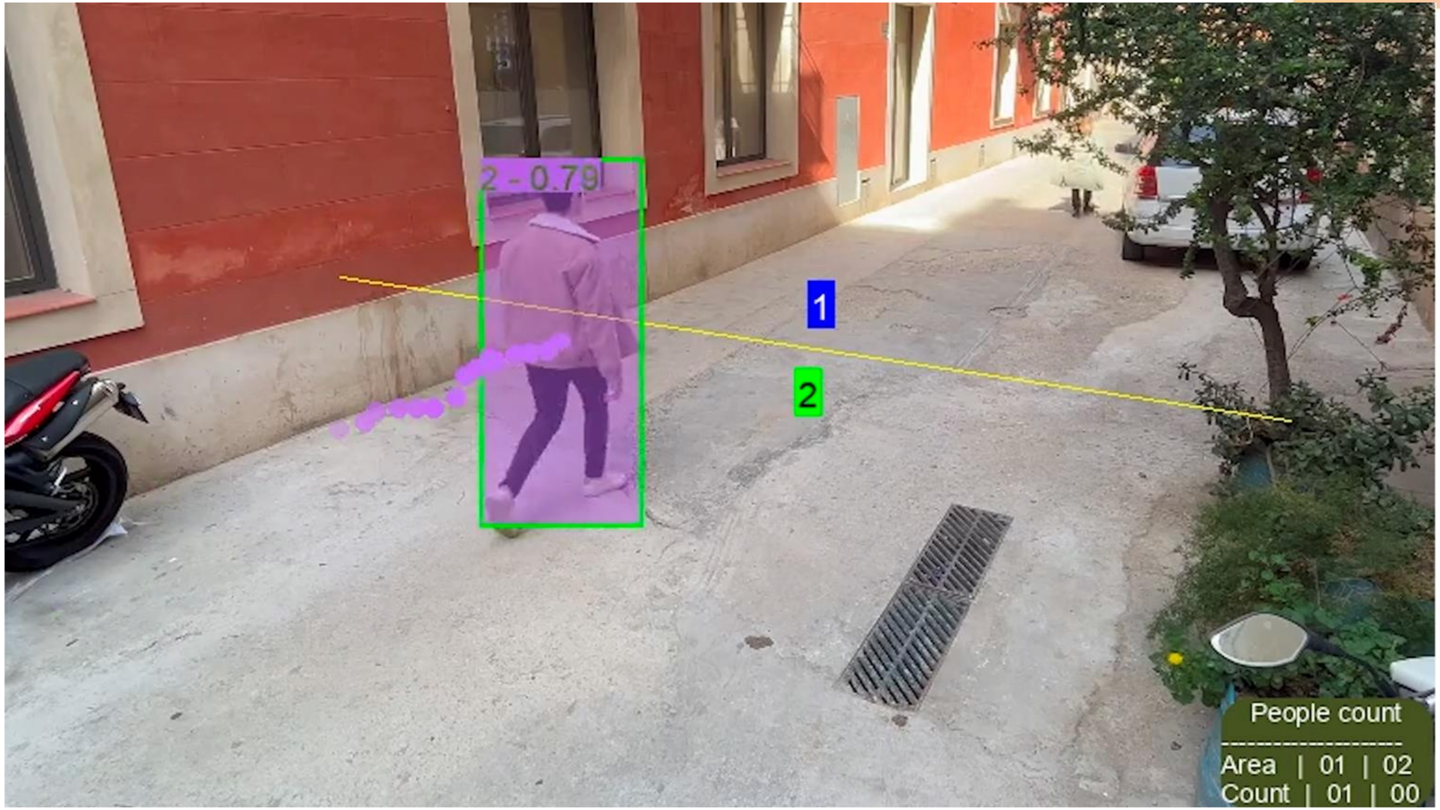
Complex task	=>	**VSA**
Small tasks (partial)	=>	**Nodes** of the VSA

Why?

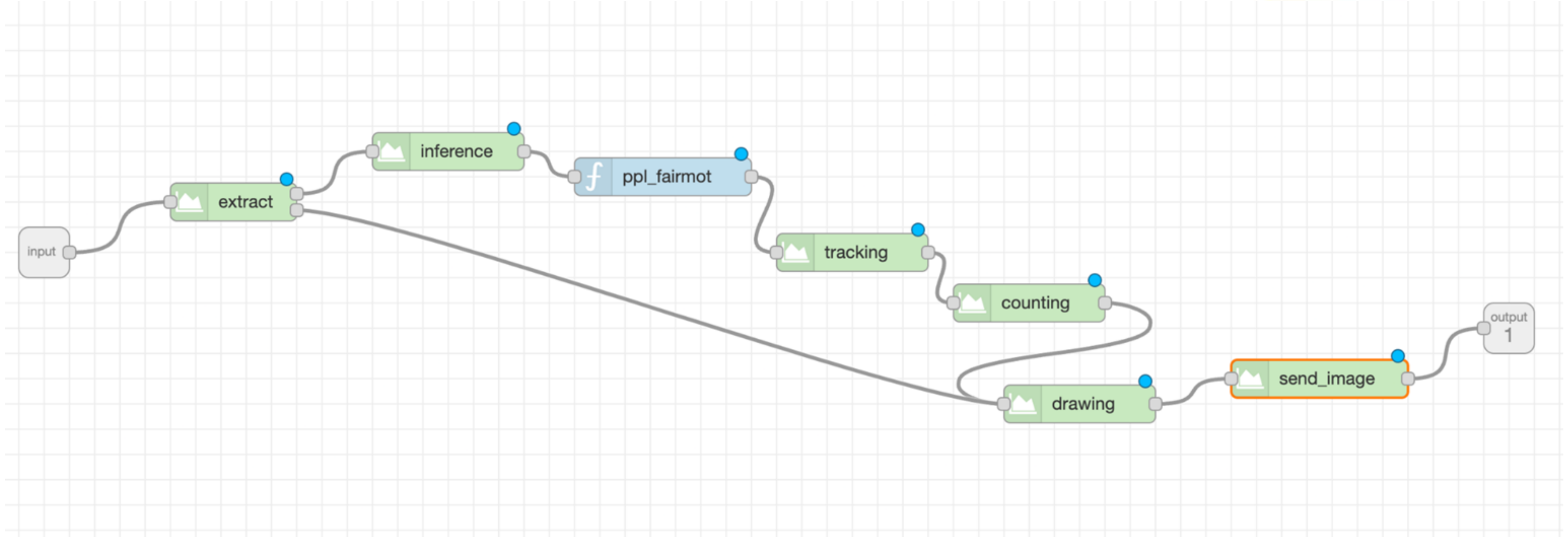
- Reusable nodes - can have different creators
- Allow to customize an existing application to a new use case by modifying some nodes
- Security: Use Webassembly technology with its sandboxing approach to allow custom code without impact on the device security

Example Application: People Counting

People Counting VSA



VSA of People Counting Application



Conclusion

- WebAssembly is a great choice for tiny IoT devices
  - High performance and low overhead
- WEdge enables high development productivity and agility
- Vision Sensing Applications (VSAs) are a convenient paradigm for programming sensing applications
  - Promoting component reuse
  - Enabling low-/no-code development

Further Information

Midokura
https://www.midokura.com

AITRIOS
https://www.aitrios.sony-semicon.com/en/

IMX500
https://developer.sony.com/develop/imx500/

Demo Video
https://bit.ly/mido-vsa-demo

Please visit our booth!

Thank you

midokura ミドクラ

A Sony Group Company