

# Eclipse OpenBSW

## Eclipse SDV Korea

# A fresh view on Microcontroller Basic Software

Tom Fleischmann

[Thomas.Fleischmann@accenture.com](mailto:Thomas.Fleischmann@accenture.com)

4<sup>th</sup> December 2025

**Accenture GmbH**



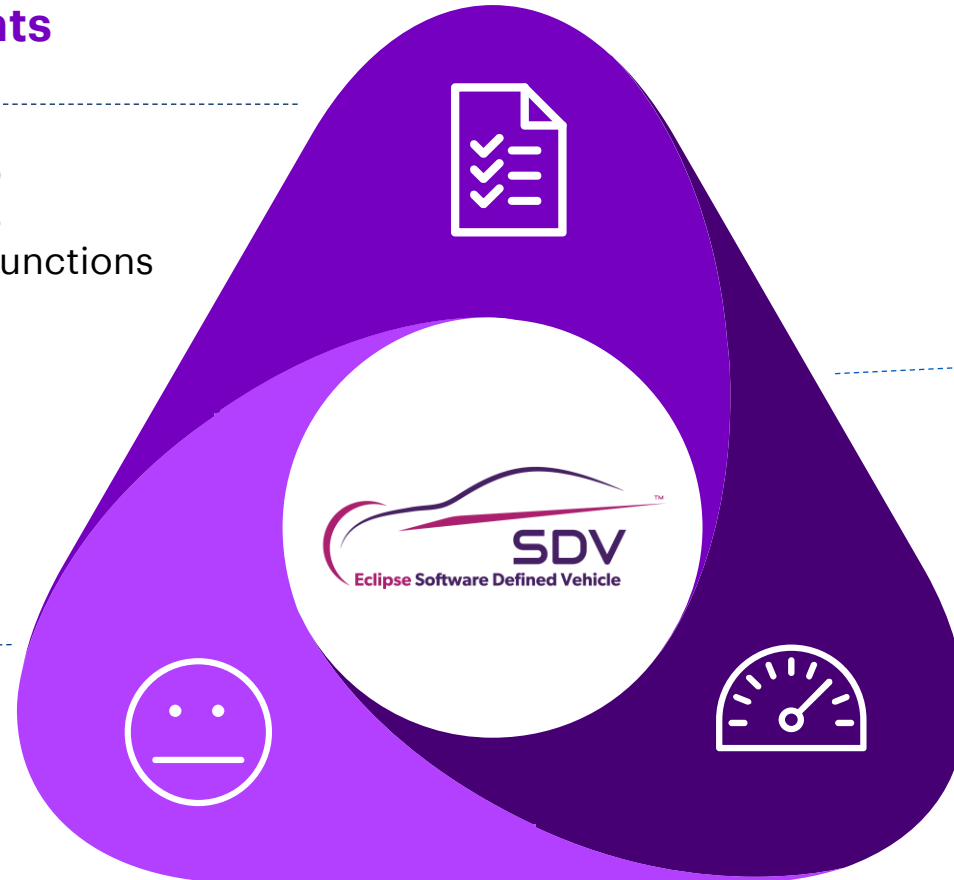
# Automotive MCU SW-Development

## 1 - Basic requirements

- Real-Time Behavior
- Safety & reliability
- Low power, fast wakeup
- Vehicle & Bus interfaces
- Growing integration of functions

## 2 - The struggle in projects

- Integration is a **slow, labor-intensive** process which leads to **long round-trip times** of changes
- Developers struggle with the **complexity of the models** in complex & GUI tools
- **Difficult to flexibly exploit MCU capabilities** due to strict architecture restrictions
- **Hard to apply modern code-workflows** on modelling (compare, merge, management, code-review)



## 3 - A way forward

- Configuration as code  
**“Terraform” the SDV**
- Code generation based on textual models
- Allows for using well-established tools
- A change is a single git pull request
- Enable CI/CD workflows
- Modern codebase



# Eclipse OpenBSW

We are introducing **Eclipse OpenBSW**:

an **open source, developer friendly runtime** for complex embedded software for Automotive embedded MCUs - distilled from 15 years of SOP projects.

Link to GitHub:



Foundation is on the road on several MCUs with German OEMs

1

Written in native C++, work with/without AUTOSAR \*

2

Uses FreeRTOS in production

3

\*Additional add-Ons make it AUTOSAR ICC1 compliant: SWCs on RTE, work with ARXML, Diagnostics, ...

4

# Design Decisions

# Embedded C++

## Language Choice

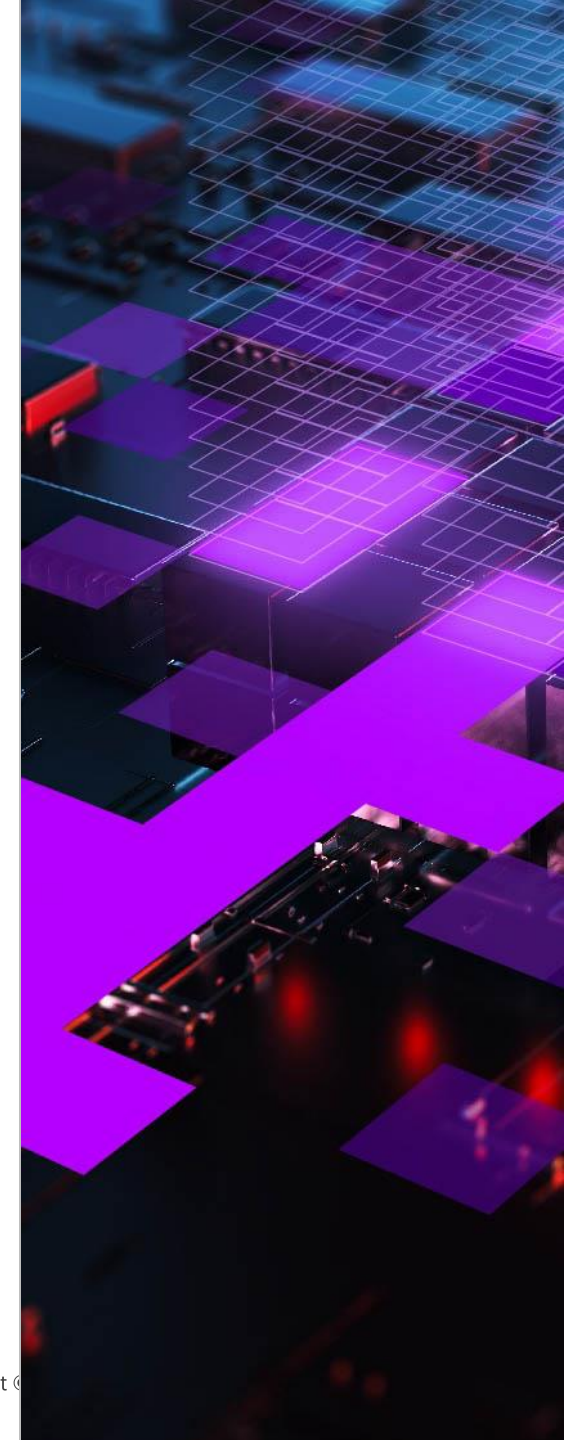
Clear advantages over C, e.g.:

- Namespaces
- Encapsulation
- Stronger Typing
- Standard Library

Can be used in Embedded Domain with only a few restrictions:

- No exceptions
- No dynamic memory allocation

**Next Stop: Rust**



# Open Architecture

## Modules – Ready to be Composed

C++ Objects composed into “Systems”

Using Dependency Injection

Lower modules don't know higher level modules

New modules can be added

Abstraction Layers hide implementations



# Safety Shell Approach

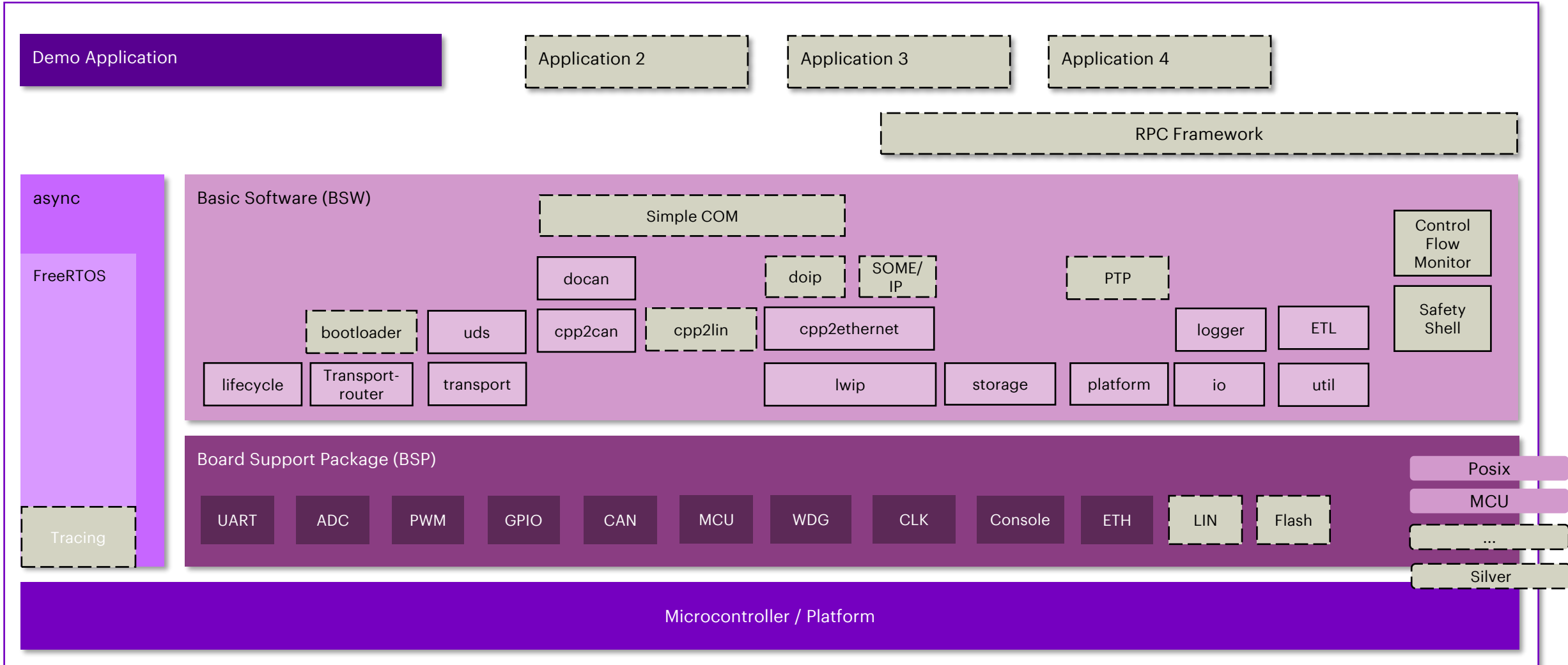
## Safe on a non-safe OS

- Safety relevant (ASIL) Code gets separated
- Make safety related code as small as possible
- Freedom From Interference
- Established on top of a non-ASIL OS
- Safety Shell consists of the building blocks
  - MPU Controller
  - Program Flow Monitor
  - Watchdog



# Eclipse OpenBSW Architecture

 Future work



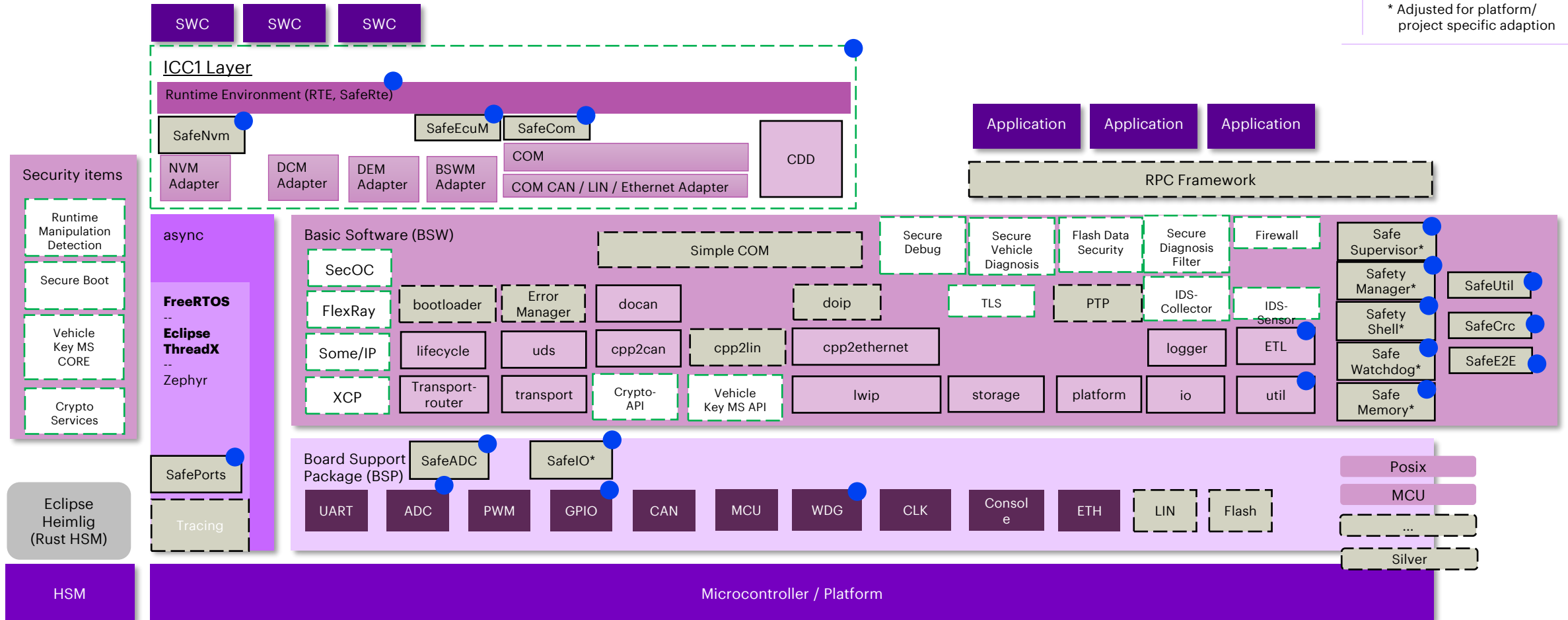
# OpenBSW & additional assets for AUTOSAR ICC1 compliance

● ASIL capable

Not OSS, but part of portfolio (Especially AUTOSAR implementations)

Planned for upstream

\* Adjusted for platform/project specific adaption



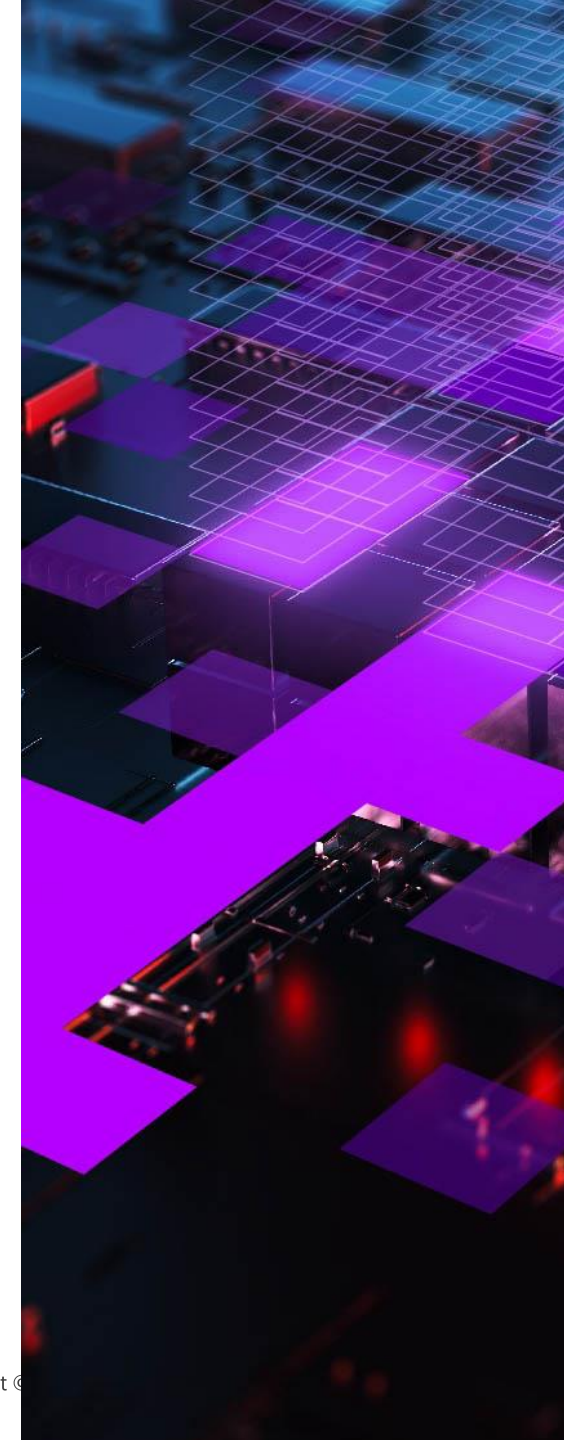
# Examples

# Example - Configuration as Code

## CAN Configuration: the CanSystem

```
::systems::CanSystem::CanSystem (::async::ContextType context,  
                                   StaticBsp& staticBsp)  
  
: : lifecycle::SingleContextLifecycleComponent (context)  
, _context (context)  
, _transceiver0 (  
    context,  
    ::busid::CAN_0,  
    Can0Config,  
    staticBsp.getCanPhy (),  
    staticBsp.getPowerStateController ())  
  
{ }
```

bios::CanFlex2Transceiver



# Example - Textual Modelling

## COM Configuration: via a Textual DSL

```
ComInstance Tester {  
    ComChannel SampleChannel, cycleTime: 10 {  
        Message SeatControl, id: 0x234, length: 4, direction: out {  
            Signal SeatVPos, length: 12, offset: 0  
            Signal SeatHPos, length: 12, offset: 12  
            PeriodicTiming interval: 100  
            ...  
        }  
        ...  
    }  
}
```



A photograph of a complex industrial machine room. The room is filled with various pieces of machinery, including large pipes, valves, and control panels. The lighting is warm and focused on the equipment. The floor is made of metal grating. The overall scene is one of a well-maintained industrial facility.

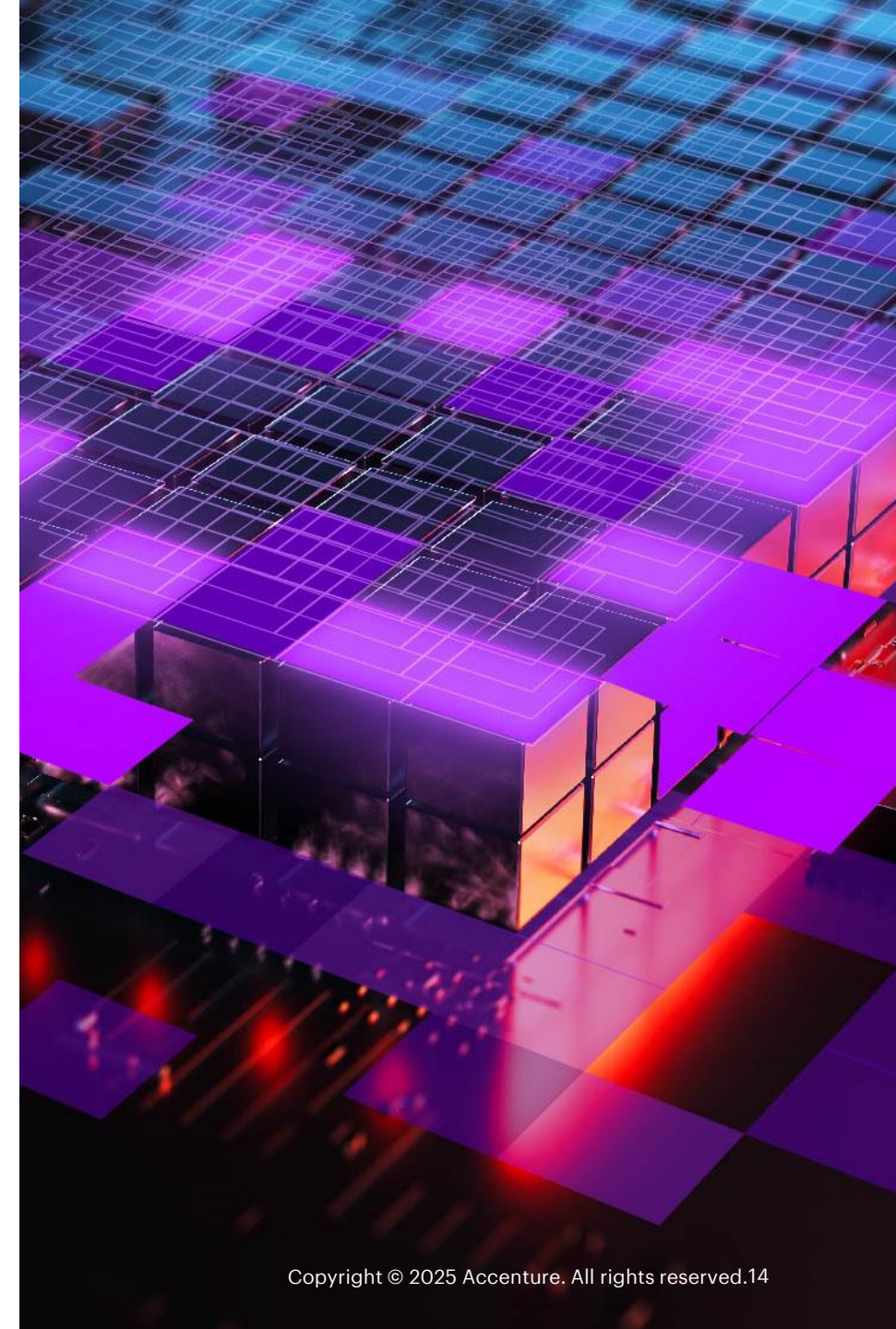
# Machine room update

# Status & Roadmap

# Current Status

## What's already there

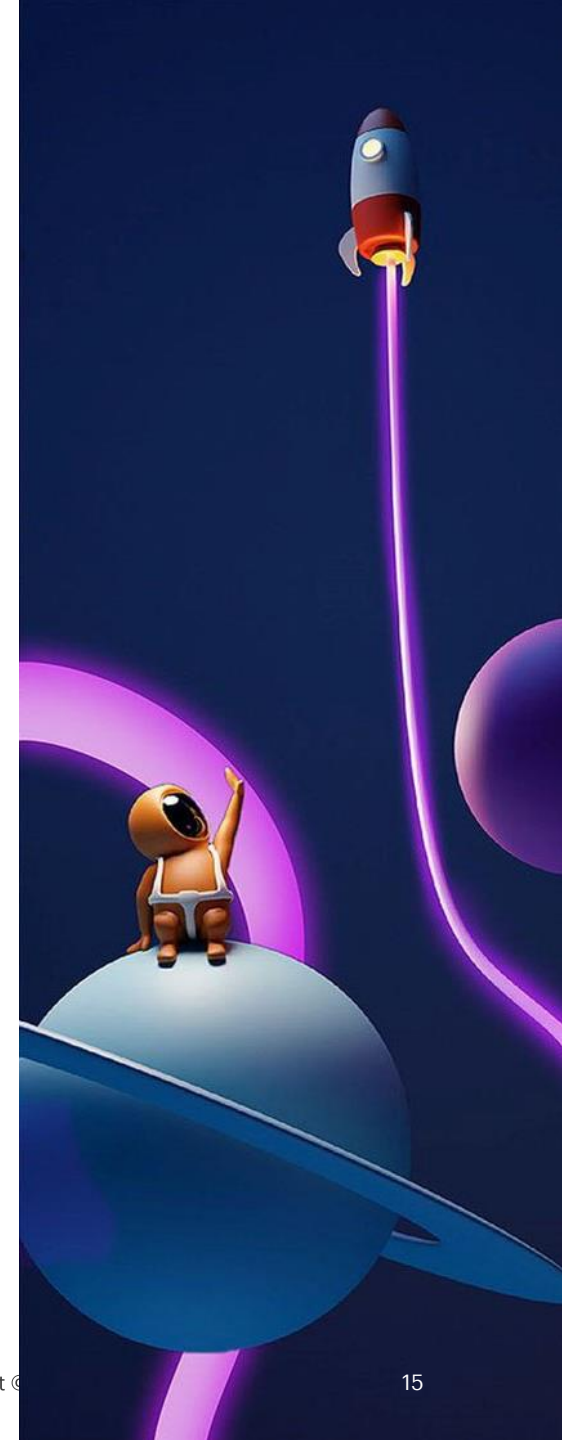
- Lifecycle (startup/shutdown) system
- Communication (CAN, DoCAN, UDS)
- **async** OS abstraction layer
- **estd** STL-like container
- Basic I/O, ADC, PWM
- Development tools (serial console, logging framework)
- S32K1 and Posix Build
- NVM - Non-volatile memory – Done
- Ethernet Support – upstream since August



# Roadmap

## Ongoing Feature Development

- COM Module (Signal based communication) – contribution by another OEM
- Moving away from in-House estd (embedded standard library) towards ETL & future upstream work towards ETL for embedded usage
- **Eclipse ThreadX** Support – [PR](#) is out now
- **Zephyr Integration** – Expected to be public soon
- Not yet open sourced:
  - DoIP (Diagnostic over IP)
  - UDS Software Updates (Flashing via bootloader)
  - Independent DSL to define signals



# Roadmap

## Future Work

Multi-core platform support

SOME/IP support, to be discussed ...

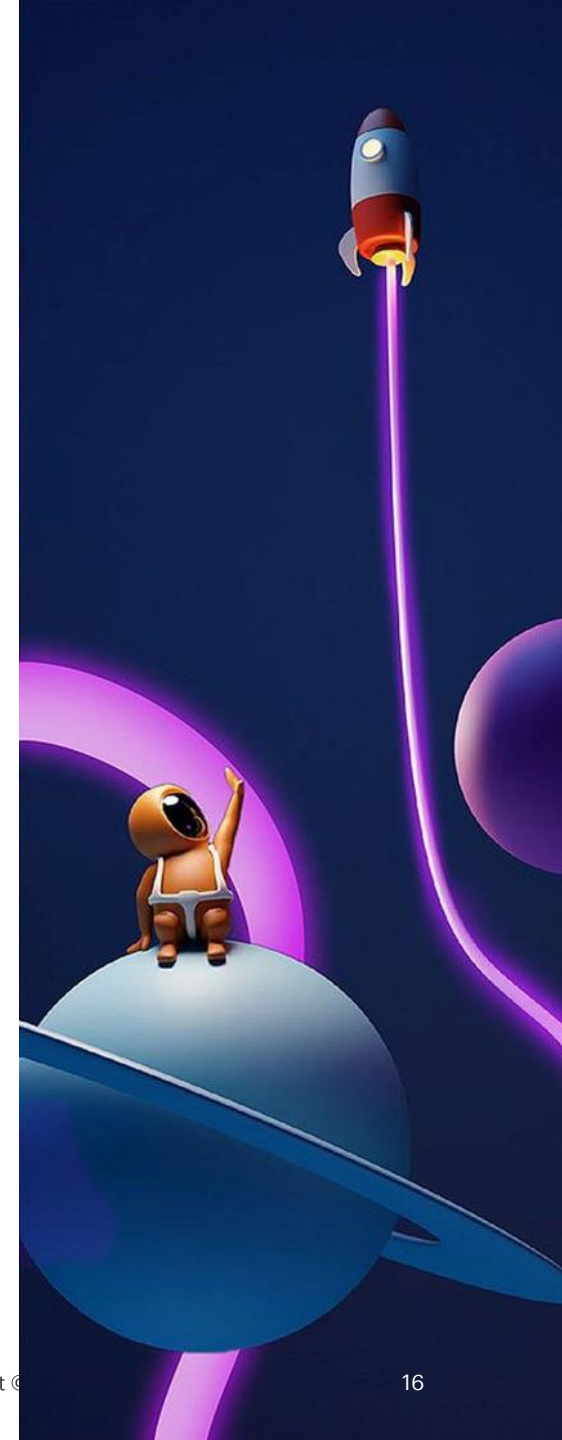
Improve BSP abstraction layer, more BSPs

Security modules (works with **Eclipse Heimlig**, a Rust based HSM)

Improved Safety support targeting ASIL-D

Tailoring of governance model

**All future features will be subject to feedback, prioritization and engagement by you, the SDV community!**



# Governance & how to get involved

- GitHub with issues & discussions
- Monthly meeting in the Eclipse calendar
- Check the open issues

## **Contributions are welcome, especially for**

- New supported hardware, ..
- New features, ...
- Bugs reports, ...
- Ideas and requirements are highly welcome ...

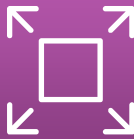


# SDV Future ...



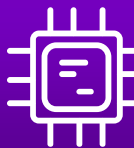
## Enable new BSW collaboration

Open-Source automotive Basic Software enables new industry collaborations between our customers.



## Scaling SOP-proven SW

The origins of OpenBSW are 15+ years on the road with German Premium & Volume OEMs.



## AUTOSAR & non-AUTOSAR

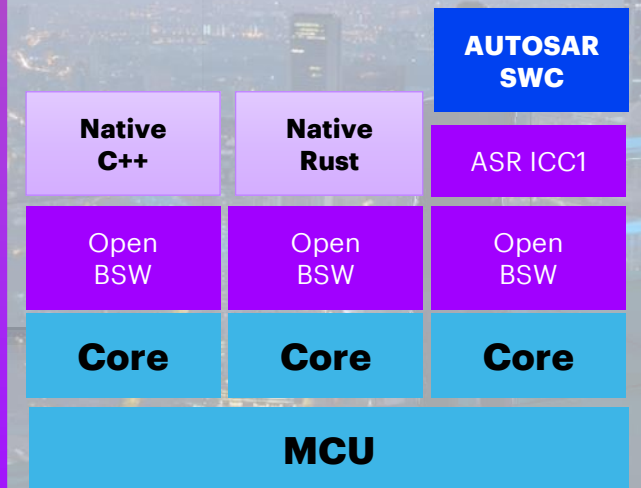
Run legacy SWCs and as well effectively develop w/o AUTOSAR constraints.



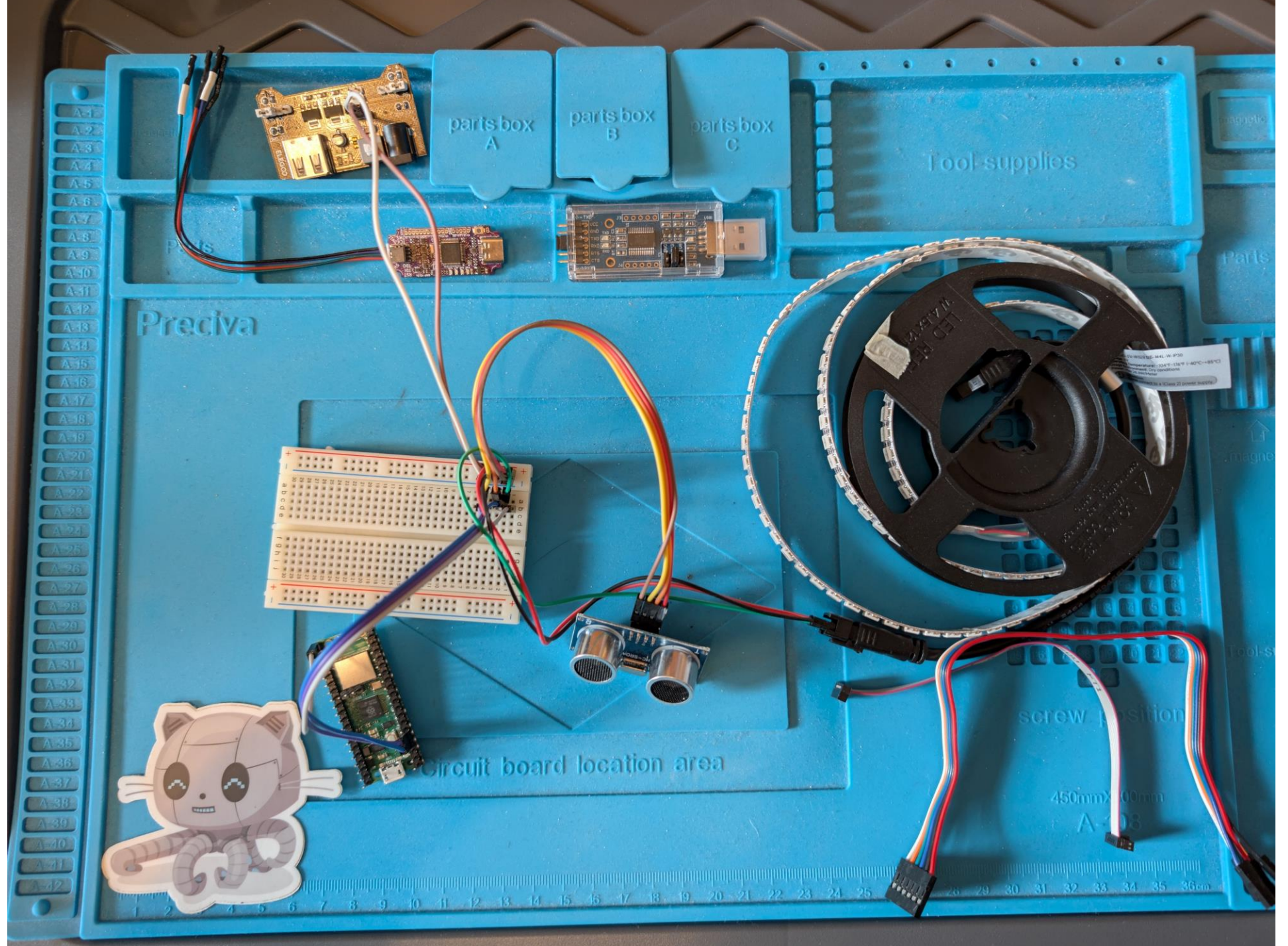
## Outlook – AI in development

The whole OpenBSW is managed in Visual Studio Code as code & text >> This will unlock agentic AI support on BSW development.

## New embedded SW possibilities



# Unlocked AI



# Eclipse OpenBSW on GitHub Codespaces

```
demo > LED_DEMO > LED-Demo_Architecture_in_OpenBSW.md > # LED Demo Architecture in OpenBSW > ## BSP Architecture for New Platforms > ### Key BSP Implementation Files
1 # LED Demo Architecture in OpenBSW
54 ## Startup Call Graph & Lifecycle
56 ### Main Entry Flow
63 |
64 |   |--- idleHandler.init() // Console & logging setup
65 |   |--- AsyncAdapter::run() // Start FreeRTOS scheduler
66 |   |--- startApp() // Lifecycle initialization
67 |   |--- RuntimeSystem (Level 1) - TASK_BACKGROUND
68 |   |--- SafetySystem (Level 1) - TASK_SAFETY
69 |   |--- BspSystem (Level 1) - TASK_BSP [Platform-specific]
70 |   |--- TransportSystem (Level 4) - TASK_UDS
71 |   |--- DoCanSystem (Level 5) - TASK_CAN
72 |   |--- UdsSystem (Level 6) - TASK_UDS
73 |   |--- SysAdminSystem (Level 7) - TASK_SYSADMIN
74 |   |--- LedProximitySystem (Level 7) - TASK_DEMO
75 |
76 ### Lifecycle Run Levels
77
78 | Level | Components | Context | Purpose |
79 |-----|-----|-----|-----|
80 | **1** | RuntimeSystem, SafetySystem, BspSystem | TASK_BACKGROUND, TASK_SAFETY, TASK_BSP | Core runtime, safety, and BSP initialization |
81 | **2** | Platform-specific (CanSystem) | TASK_CAN | Platform-specific components |
82 | **3** | Reserved | - | Available for future use |
83 | **4** | TransportSystem | TASK_UDS | Transport layer initialization |
84 | **5** | DoCanSystem, EthernetSystem | TASK_CAN, TASK_ETHERNET | Communication protocol setup |
85 | **6** | UdsSystem | TASK_UDS | Diagnostic services (UDS) |
86 | **7** | SysAdminSystem, **LedProximitySystem** | TASK_SYSADMIN, **TASK_DEMO** | System administration and custom applications |
87
88 ### FreeRTOS Context Mapping & Task Priorities
89
90 OpenBSW implements a sophisticated priority-based task scheduling system where task context numbers directly map to FreeRTOS priorities. This ensures deterministic, automotive-grade execution behavior.
91
92 #### Task Priority Definition Sources
93
94 Primary Definition: ~/executables/referenceApp/asyncCoreConfiguration/include/async/Config.h
95
96 ```cpp
97 /**
98  * Tasks for different operations are defined here.
99  * Task context numbers directly map to FreeRTOS priorities.
100  */
101 enum
102 {
103     TASK_BACKGROUND = 1, // Priority 1 - Runtime monitoring
104     TASK_BSP, // Priority 2 - Board Support Package
105     TASK_UDS, // Priority 3 - Diagnostic services
106     TASK_DEMO, // Priority 4 - Demo applications + LED PROXIMITY
107     TASK_ETHERNET, // Priority 5 - Ethernet communication
108     TASK_CAN, // Priority 6 - CAN communication
109     TASK_SYSADMIN, // Priority 7 - System administration
110     TASK_SAFETY, // Priority 8 - Safety supervision
111     // -----
112     ASYNC_CONFIG_TASK_COUNT,
113 };
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SERIAL MONITOR

@quarterbit → /workspaces/openbsw (copilot/vscode1758643136652) \$ Open chat [Ctrl]+[I]. Start typing to dismiss.

EXPLORER

- OPENBSW [CODESPACES: UBIQUITOUS C...
- > .ci
- > .github
- > admin
- > demo
  - LED\_DEMO
    - BUILD\_AND\_RUN.md
    - CodeQualityReport.md
    - LED\_Demo\_Development\_Journey.md
    - LED\_PROXIMITY\_DEMO.md
    - LED\_Proximity\_DemoRequirements.md
    - LED-Demo\_Architecture\_in\_OpenBSW.md
    - PicoW\_PINOUT.md
    - README.md
    - RequirementsCheck.md
    - S32K148\_LED\_PROXIMITY\_PINOUT.md
    - README.md
  - > doc
  - > docker
  - > executables
  - > libs
  - > platforms
    - > posix
    - > rp2040
      - > 3rdparty
      - > bsp
      - > cmake
      - > hardFaultHandler
      - > lwipSysArch
      - M CMakeLists.txt
      - OpenBSW\_Platform\_Support\_Plan\_Raspberry\_Pi...
      - > s32k1xx
        - M CMakeLists.txt
      - > test
      - > tools
    - .clang-format
    - .cmake-format
    - .gitignore
    - .treefmt.toml
    - CHANGELOG.md
    - CODE\_OF\_CONDUCT.md
    - CONTRIBUTING.md
    - doc.verifier.toml
    - ELF\_ANALYSIS\_COMPARISON.md
    - Filelists.cmake
    - LICENSE
    - NOTICE.md
    - OPENBSW\_SAFETY\_MECHANISMS.md
    - PIN\_ASSIGNMENTS\_SUMMARY.md
    - POSIX\_ENHANCED\_TESTING.md
    - posix.verifier.toml
    - README.md
    - s32k1xx.verifier.toml
  - > OUTLINE
  - > TIMELINE





**Thank you!**  
**Questions?**

**[Thomas.Fleischmann@accenture.com](mailto:Thomas.Fleischmann@accenture.com)**