Eclipse OpenBSW

Eclipse SDV Korea

A fresh view on Microcontroller Basic Software

Tom Fleischmann

Thomas.Fleischmann@accenture.com

4th December 2025

Accenture GmbH

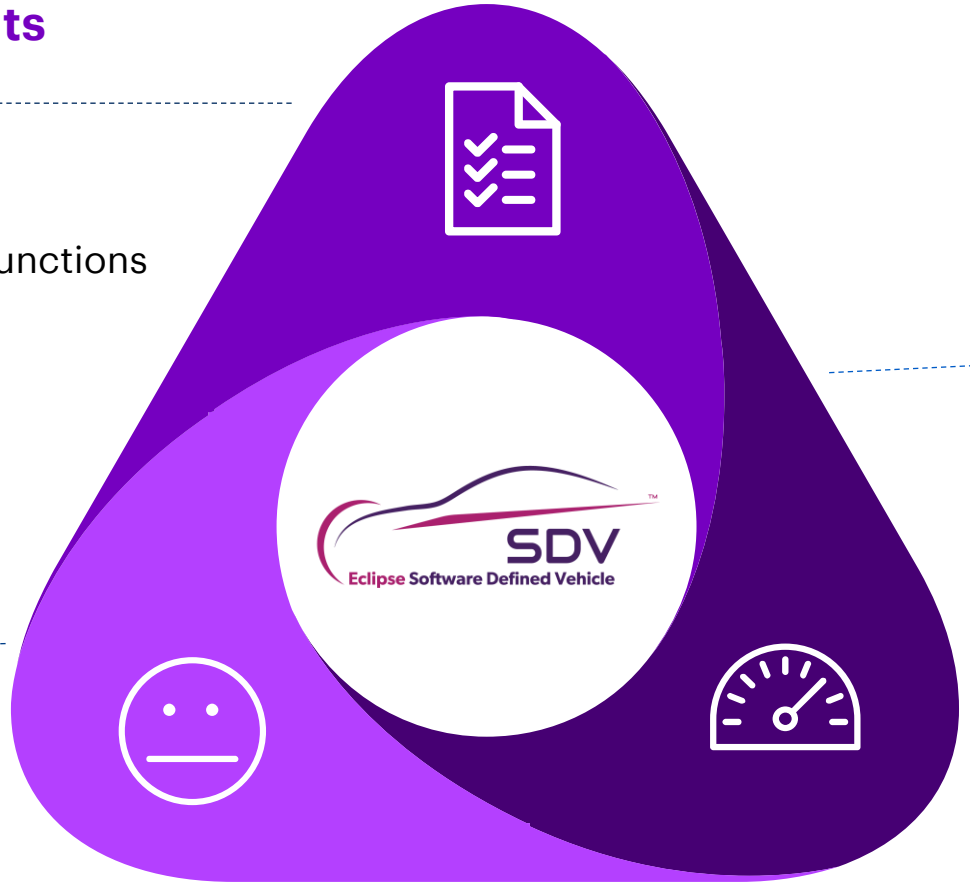
Automotive MCU SW-Development

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

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

Uses FreeRTOS in production

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

accenture

Design Decisions

Embedded C++

Language Choice

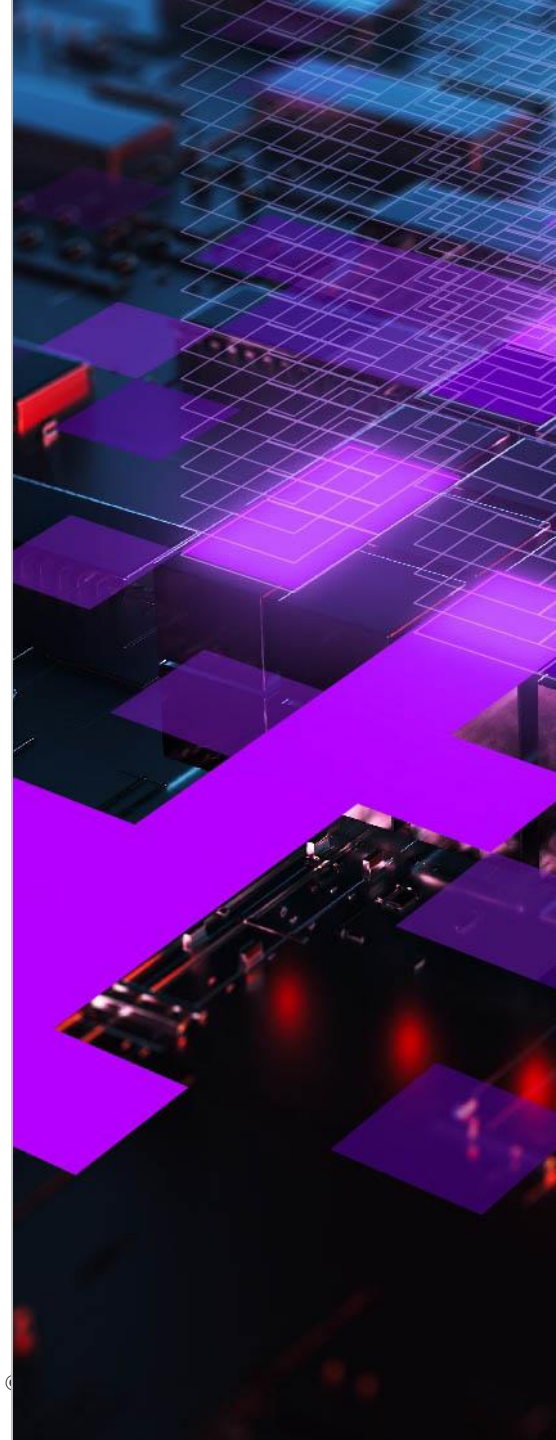
Clear advantages over C, e.g.:

- 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

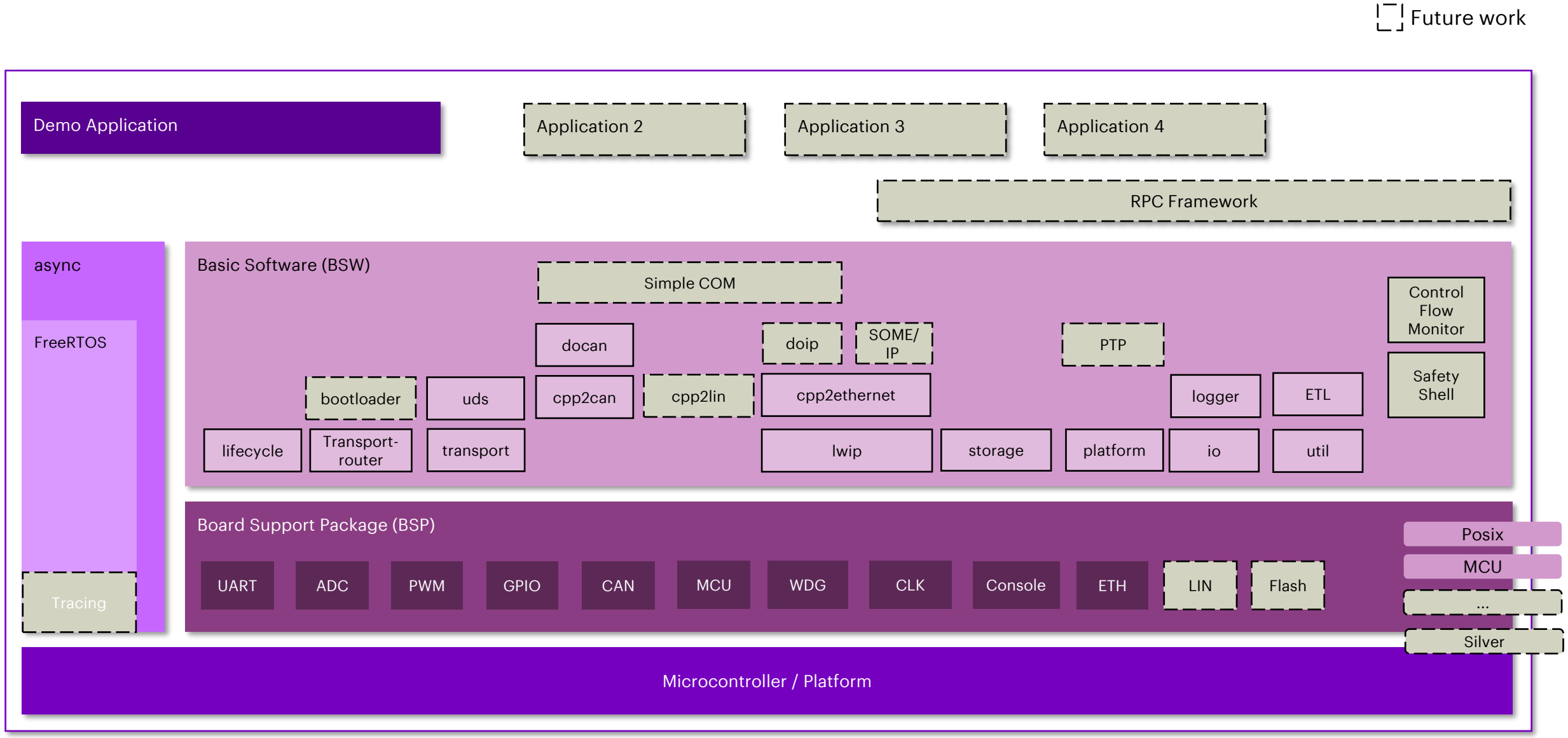
Abstraction Layers hide implementations

Safety Shell Approach

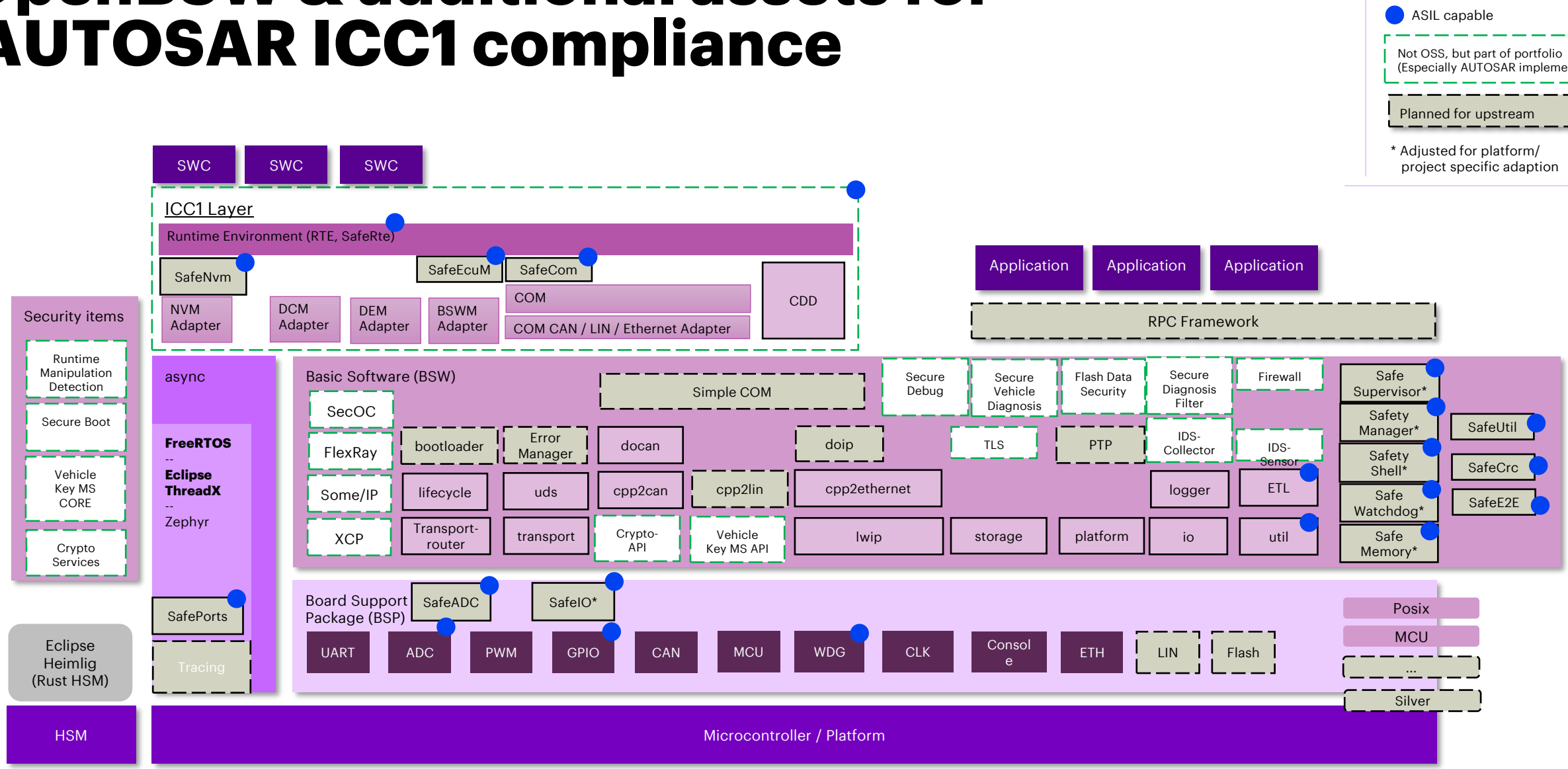
Safe on a non-safe OS

- 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



OpenBSW & additional assets for AUTOSAR ICC1 compliance



accenture

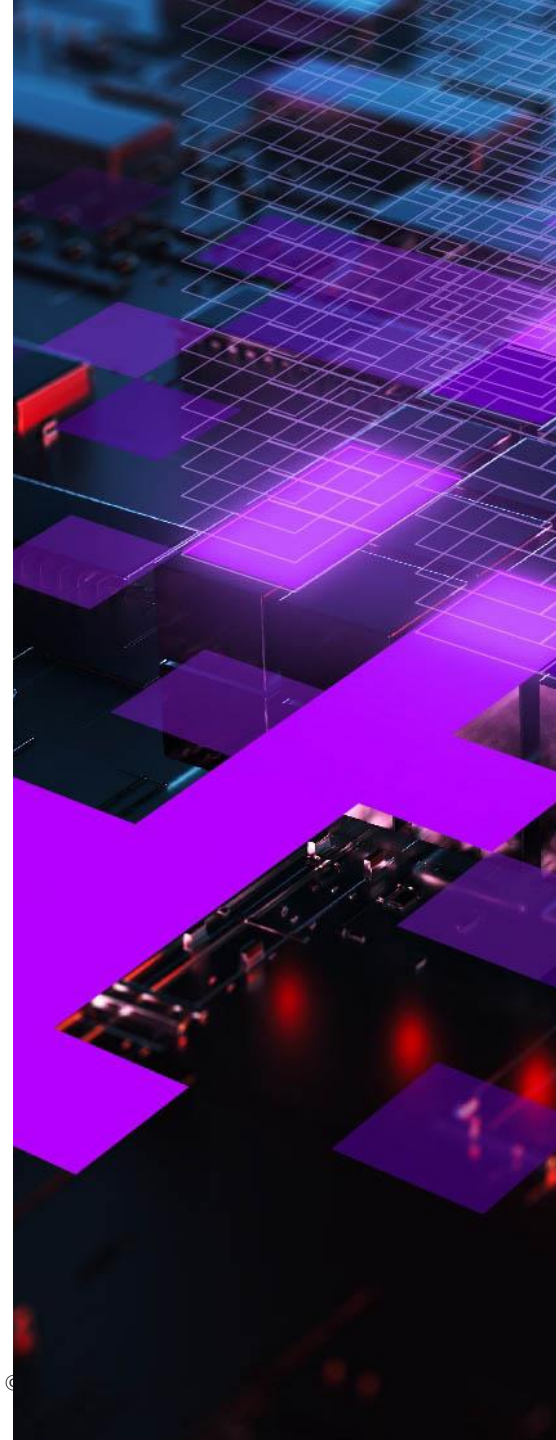
Examples

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())
    {}
```

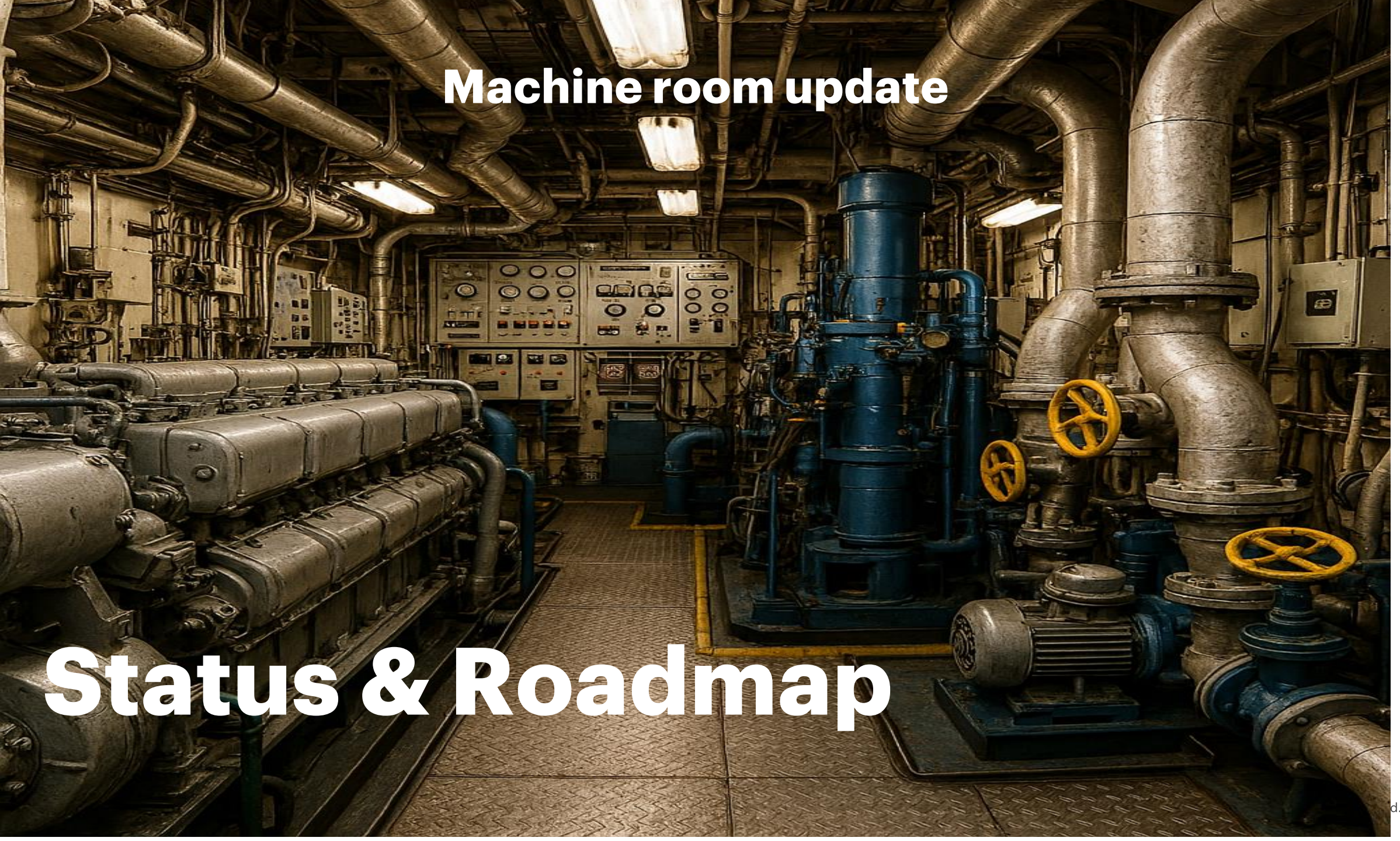
bios::CanFlex2Transceiver



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
            ...
        }
        ...
    }
}
```



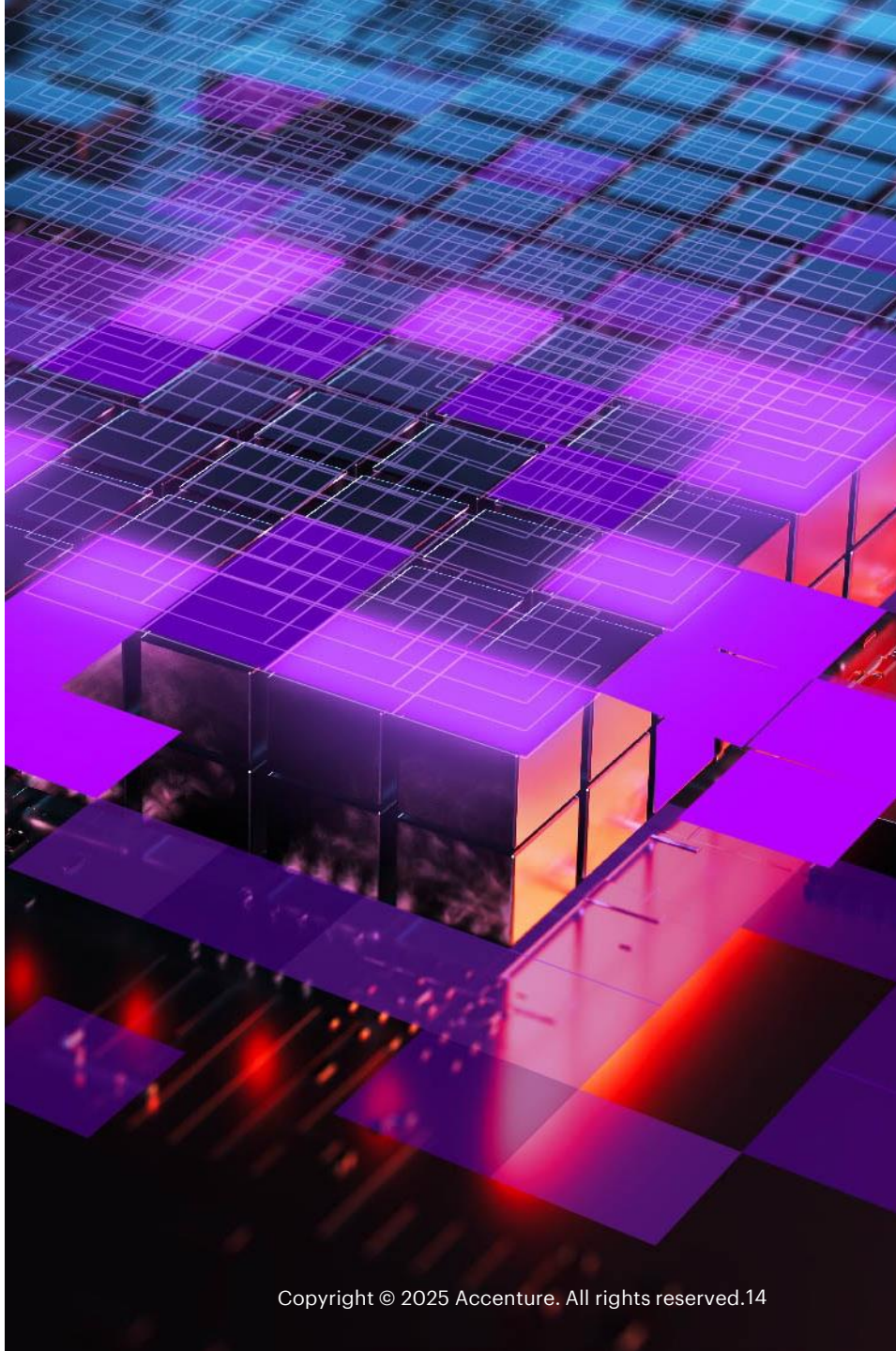
Machine room update

Status & Roadmap

Current Status

What’s already there

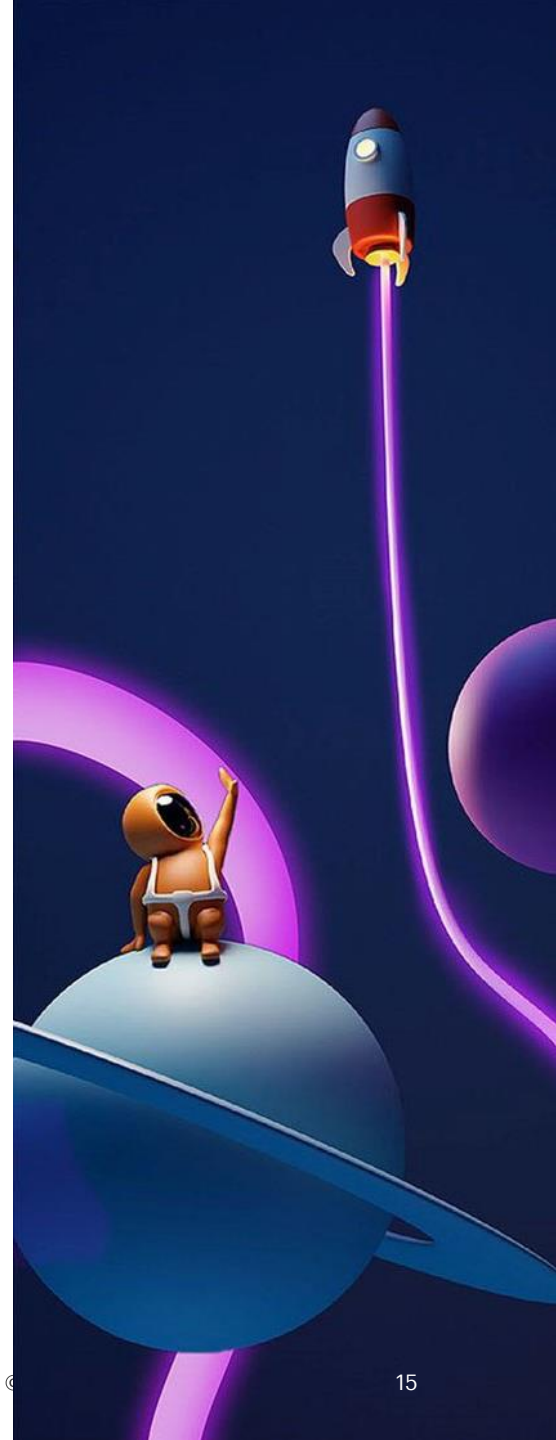
- Lifecycle (startup/shutdown) system
- Communication (CAN, DoCAN, UDS)
- **`async`** OS abstraction layer
- **`estd`** STL-like container
- Basic I/O, ADC, PWM
- Development tools
  (serial console, logging framework)
- S32K1 and Posix Build
- NVM - Non-volatile memory – Done
- Ethernet Support – upstream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

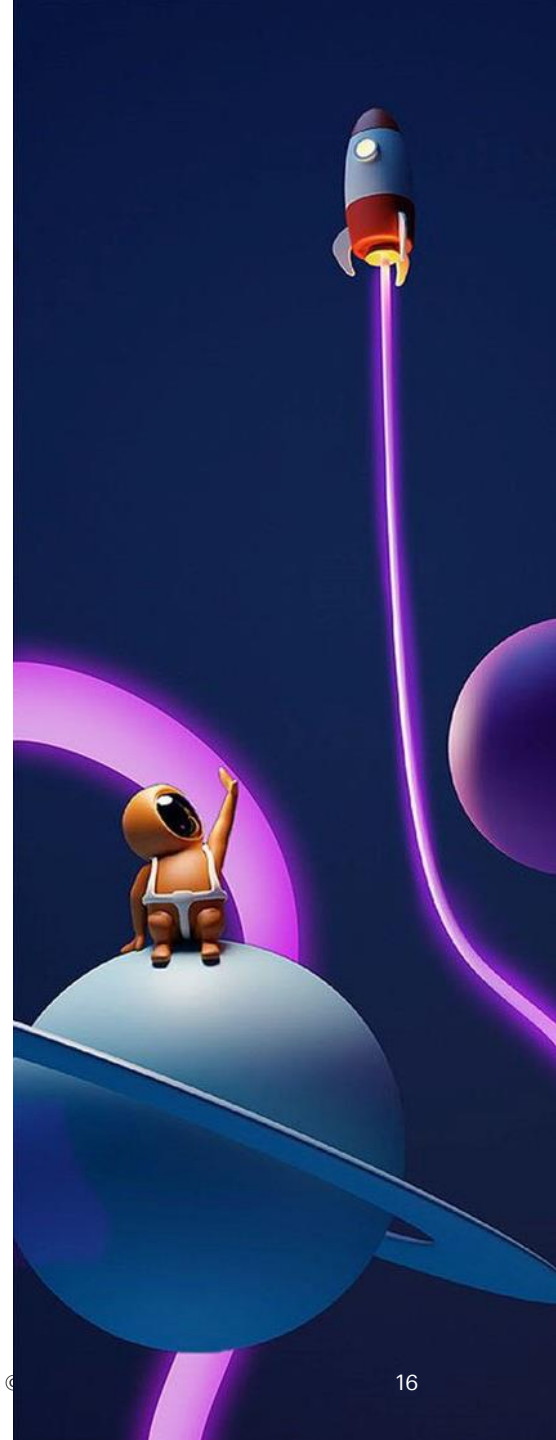
Improve BSP abstraction layer, more BSPs

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

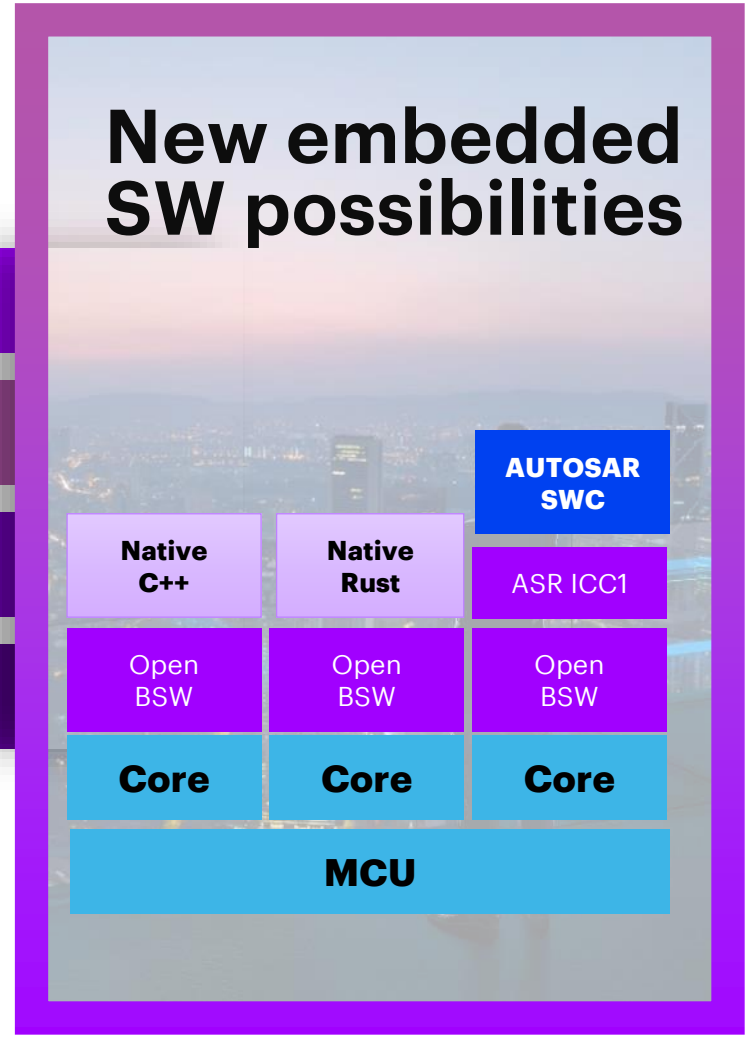
The origins of OpenBSW are 15+ years on the road with German Premium & Volume OEMs.

AUTOSAR & non-AUTOSAR

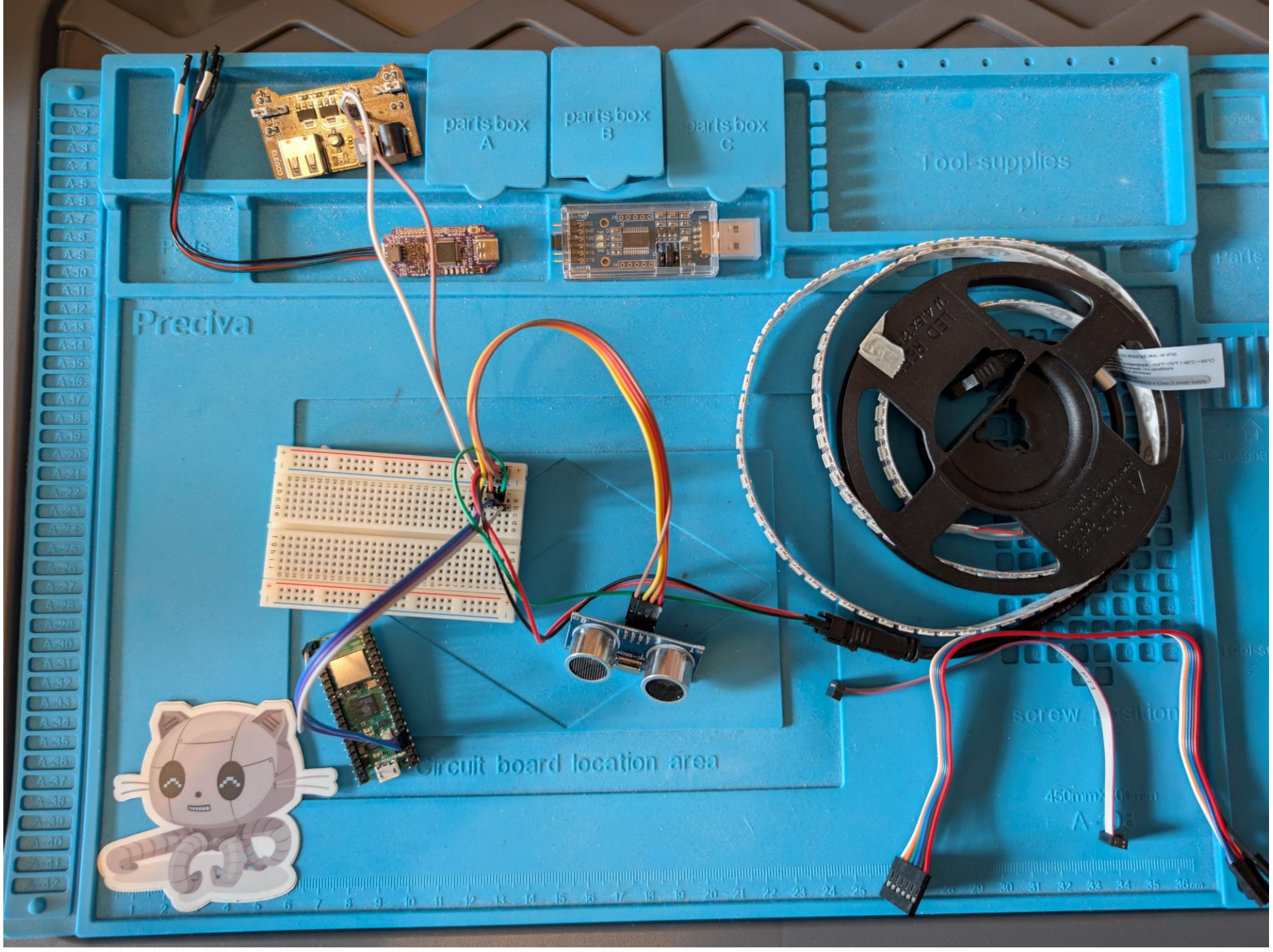
Run legacy SWCs and as well effectively develop w/o AUTOSAR constraints.

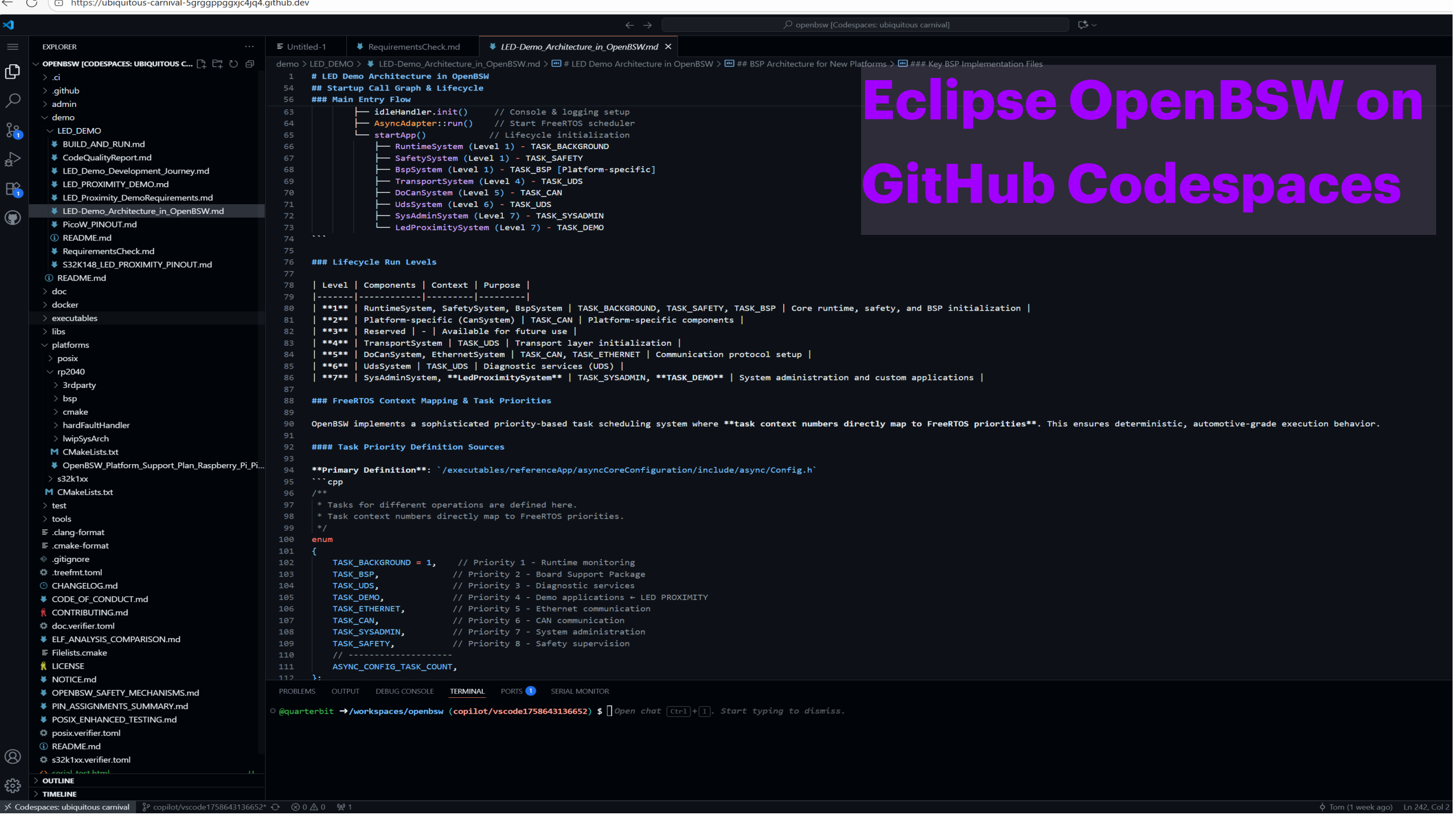
Outlook – AI in development

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

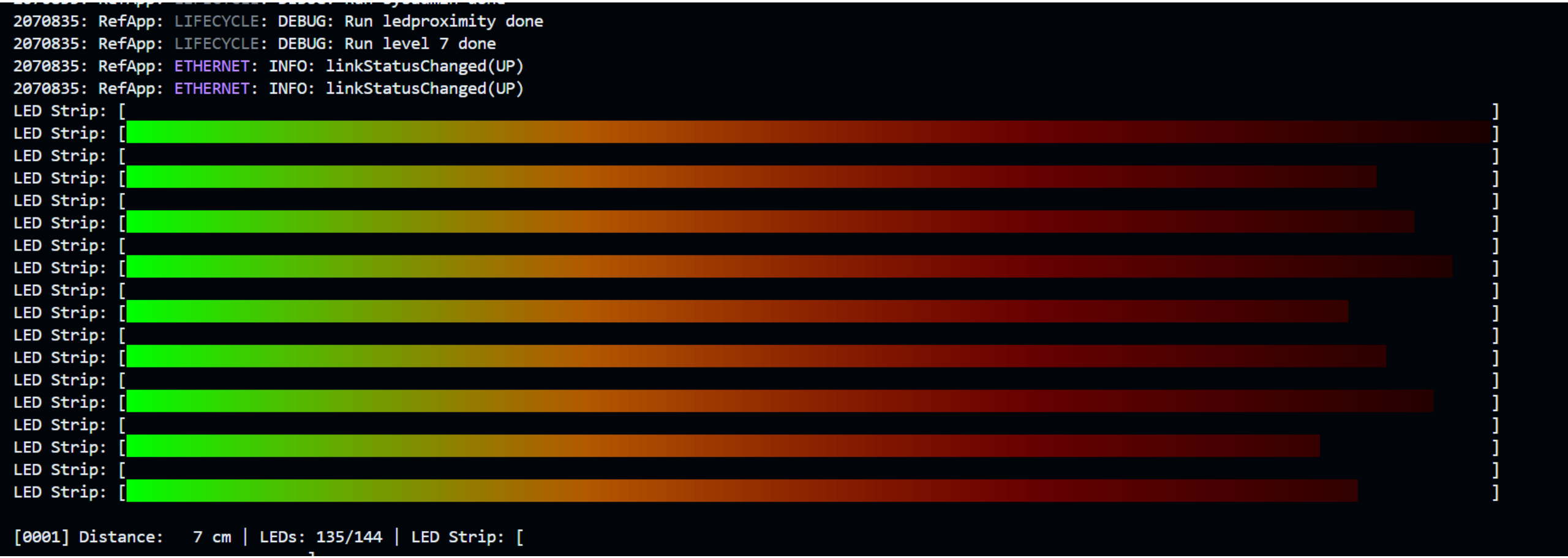


Unlocked AI


parts box A
parts box B
parts box C
Tool supplies
Parts
Preciva
Circuit board location area
screw position


Eclipse OpenBSW on
GitHub Codespaces

Running the Posix simulation directly in GitHub Codespaces



Thank you!
Questions?

Thomas.Fleischmann@accenture.com