Retrospective Sprint II

Team 4

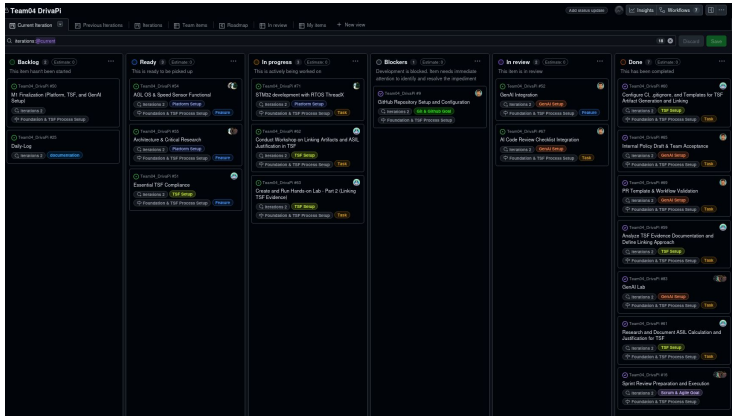
sea|me
software engineering in automotive and mobility ecosystems

Milestone one finalization

- Platform
- TSF
- Gen AI Setup

sea|me
software engineering in automotive and mobility ecosystems

Project

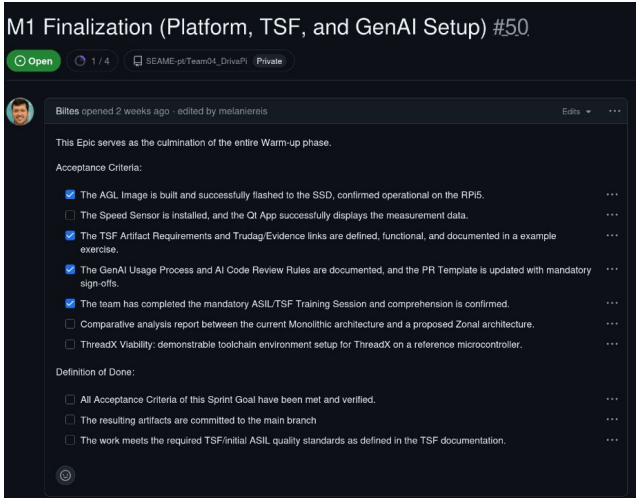


- 2 issues ‘In progress’
- 1 ‘blockers’
- 10 ‘Done’

sea|me

software engineering in automotive and mobility ecosystems

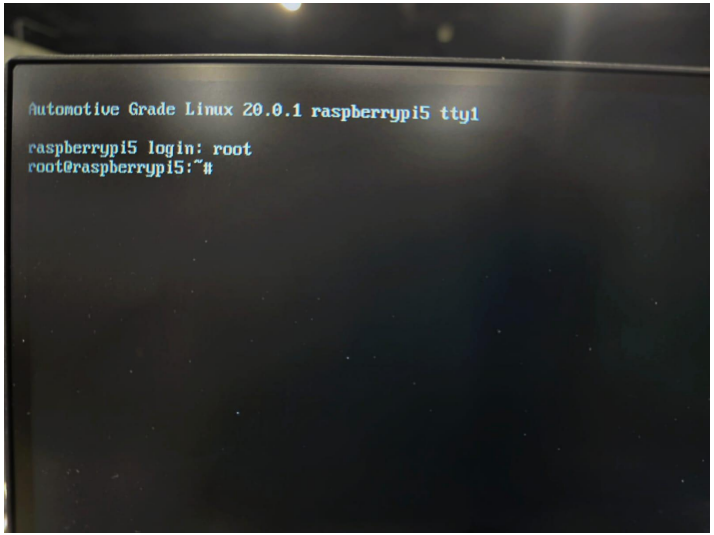
Sprint II Goals



Automotive grade linux

sea:me
software engineering in automotive and mobility ecosystems

AGL on Raspberry pi



- Fully functional OS
- Wifi available
- Only TUI

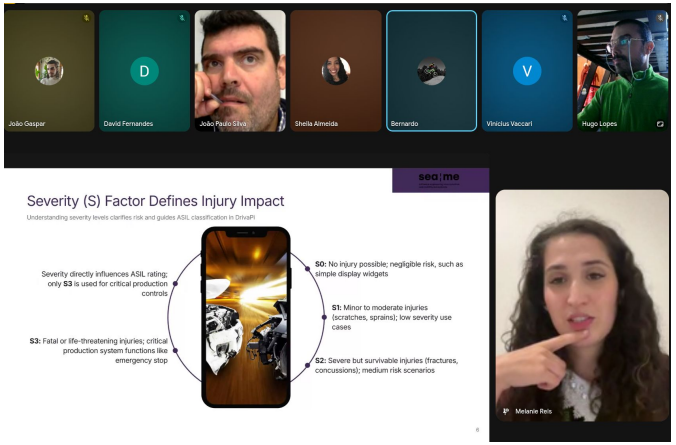
sea|me
software engineering in automotive and mobility ecosystems

Trustable software framework

sea|me

software engineering in automotive and mobility ecosystems

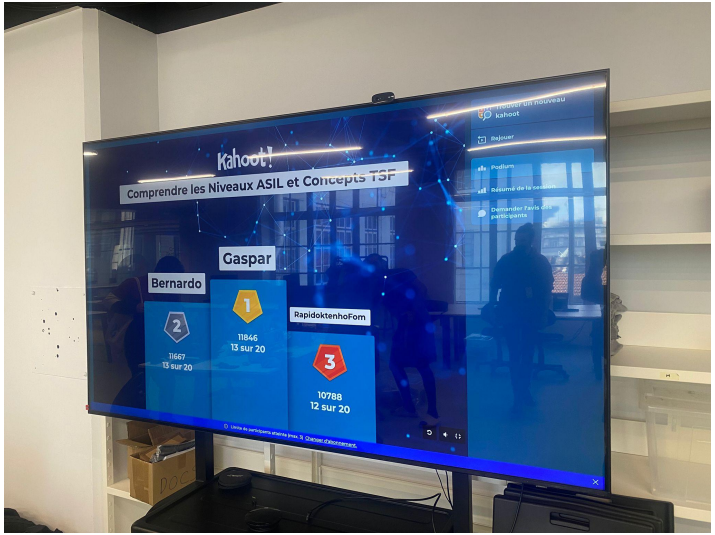
ASIL/TSF Training Session



- 1st Lab Recap (URD linking to SRD)
- Linking evidences with references
- Manual Score
- ASIL (Automotive Safety integrity level)
- HARA (Hazard Analysis and Risk Assessment) Exposure, severity and control

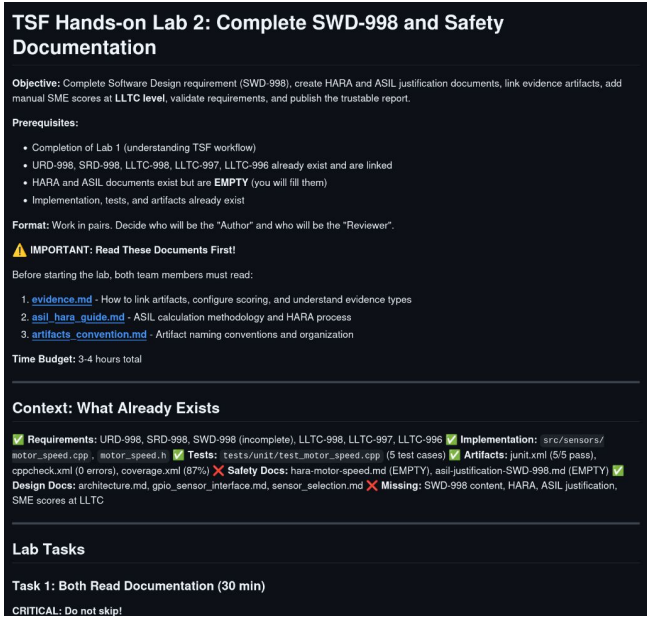
sea|me
software engineering in automotive and mobility ecosystems

Assess learning



- A Kahoot quiz to assess learning

2nd lab



TSF Hands-on Lab 2: Complete SWD-998 and Safety Documentation

Objective: Complete Software Design requirement (SWD-998), create HARA and ASIL justification documents, link evidence artifacts, add manual SME scores at **LLTC level**, validate requirements, and publish the trustable report.

Prerequisites:

- Completion of Lab 1 (understanding TSF workflow)
- URD-998, SRD-998, LLTC-998, LLTC-997, LLTC-996 already exist and are linked
- HARA and ASIL documents exist but are **EMPTY** (you will fill them)
- Implementation, tests, and artifacts already exist

Format: Work in pairs. Decide who will be the "Author" and who will be the "Reviewer".

⚠️ **IMPORTANT: Read These Documents First!**

Before starting the lab, both team members must read:

1. evidence.md - How to link artifacts, configure scoring, and understand evidence types
2. asil_hara_guide.md - ASIL calculation methodology and HARA process
3. artifacts_convention.md - Artifact naming conventions and organization

Time Budget: 3-4 hours total

Context: What Already Exists

✅ **Requirements:** URD-998, SRD-998, SWD-998 (incomplete), LLTC-998, LLTC-997, LLTC-996 ✅ **Implementation:** `src/sensors/motor_speed.cpp`, `motor_speed.h` ✅ **Tests:** `tests/unit/test_motor_speed.cpp` (5 test cases) ✅ **Artifacts:** junit.xml (5/5 pass), cppcheck.xml (0 errors), coverage.xml (87%) ❌ **Safety Docs:** hara-motor-speed.md (EMPTY), asil-justification-SWD-998.md (EMPTY) ✅ **Design Docs:** architecture.md, gpio_sensor_interface.md, sensor_selection.md ❌ **Missing:** SWD-998 content, HARA, ASIL justification, SME scores at LLTC

Lab Tasks

Task 1: Both Read Documentation (30 min)

CRITICAL: Do not skip!

- Real-case scenario HARA and ASIL implementation
- In progress

sea|me

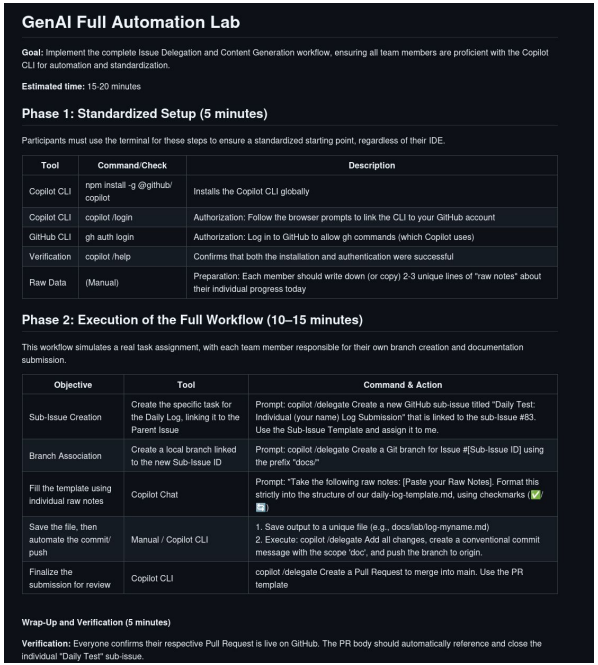
software engineering in automotive and mobility ecosystems

Generative AI

- Automation for GitHub
- GenAI full automation lab

sea|me
software engineering in automotive and mobility ecosystems

Gen AI lab



GenAI Full Automation Lab

Goal: Implement the complete Issue Delegation and Content Generation workflow, ensuring all team members are proficient with the Copilot CLI for automation and standardization.

Estimated time: 15-20 minutes

Phase 1: Standardized Setup (5 minutes)

Participants must use the terminal for these steps to ensure a standardized starting point, regardless of their IDE.

Tool	Command/Check	Description
Copilot CLI	npm install -g @github/copilot	Installs the Copilot CLI globally
Copilot CLI	copilot /login	Authorization: Follow the browser prompts to link the CLI to your GitHub account
GitHub CLI	gh auth login	Authorization: Log in to GitHub to allow gh commands (which Copilot uses)
Verification	copilot /help	Confirms that both the installation and authentication were successful
Raw Data	(Manual)	Preparation: Each member should write down (or copy) 2-3 unique lines of "raw notes" about their individual progress today

Phase 2: Execution of the Full Workflow (10–15 minutes)

This workflow simulates a real task assignment, with each team member responsible for their own branch creation and documentation submission.

Objective	Tool	Command & Action
Sub-Issue Creation	Create the specific task for the Daily Log, linking it to the Parent Issue	Prompt: copilot /delegate Create a new GitHub sub-issue titled "Daily Test: Individual (your name) Log Submission" that is linked to the sub-Issue #83. Use the Sub-Issue Template and assign it to me.
Branch Association	Create a local branch linked to the new Sub-Issue ID	Prompt: copilot /delegate Create a Git branch for Issue #[Sub-Issue ID] using the prefix "docs/"
Fill the template using individual raw notes	Copilot Chat	Prompt: "Take the following raw notes: [Paste your Raw Notes]. Format this strictly into the structure of our daily-log-template.md, using checkmarks (✅/➡️).
Save the file, then automate the commit/push	Manual / Copilot CLI	1. Save output to a unique file (e.g., docs/lab/log-myname.md) 2. Execute: copilot /delegate Add all changes, create a conventional commit message with the scope 'doc', and push the branch to origin.
Finalize the submission for review	Copilot CLI	copilot /delegate Create a Pull Request to merge into main. Use the PR template

Wrap-Up and Verification (5 minutes)

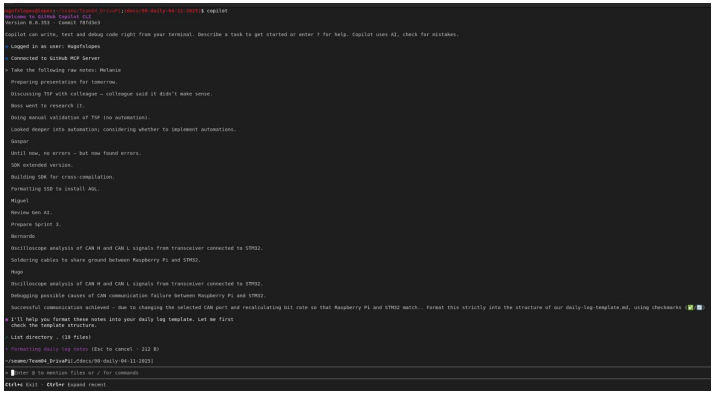
Verification: Everyone confirms their respective Pull Request is live on GitHub. The PR body should automatically reference and close the individual "Daily Test" sub-issue.

- Installation and Setup of selected AI
- Practical exercise

sea|me

software engineering in automotive and mobility ecosystems

First task

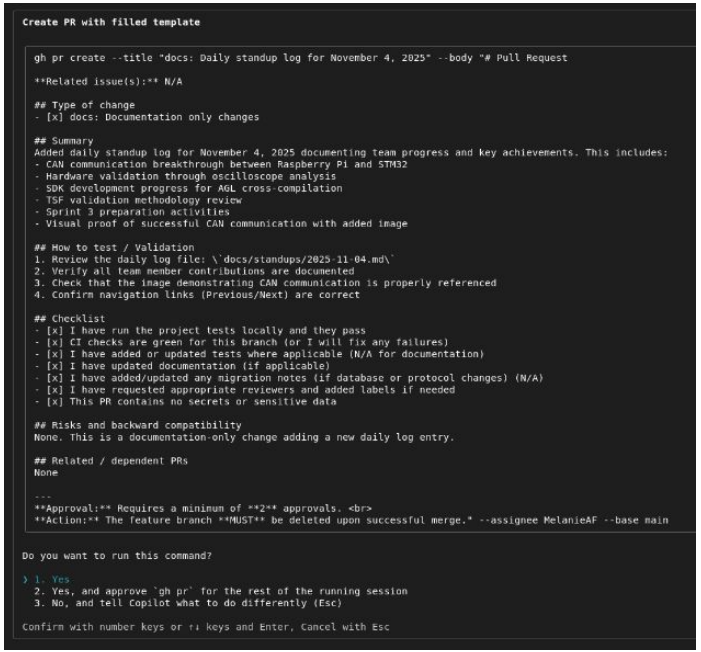


- Create daily log .md based on the template
- Use raw notes as base

sea|me

software engineering in automotive and mobility ecosystems

Pull request



- Creating PR using template
- Attribute it to a reviewer
- Approve the action

sea|me
software engineering in automotive and mobility ecosystems

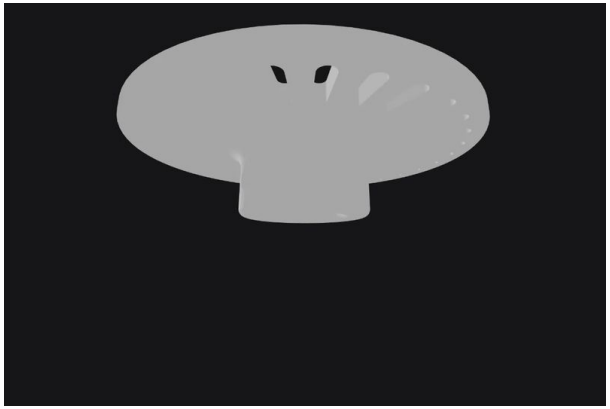
STM32 & Raspberry Pi

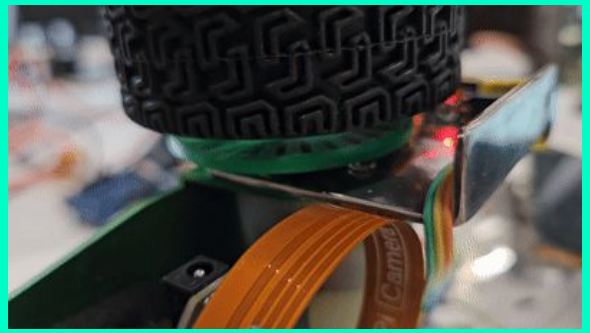
- Receiving data from speed sensor LM393  ✓
- Communicate through CAN between STM32 and Raspberry Pi ✓
- Send car’s speed to Qt app ✗

sea|me

software engineering in automotive and mobility ecosystems

From drawing to reality

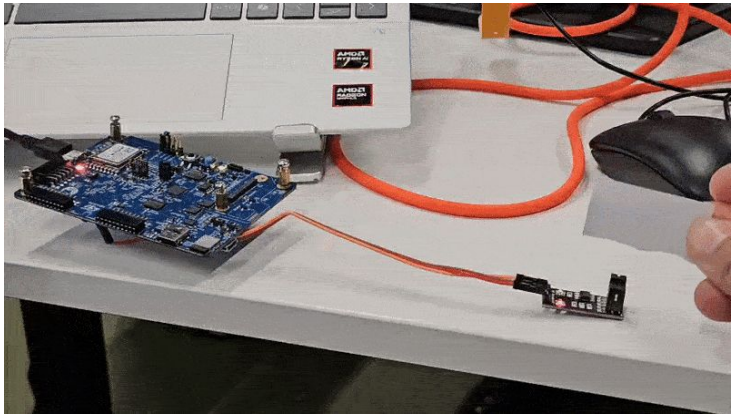




sea|me

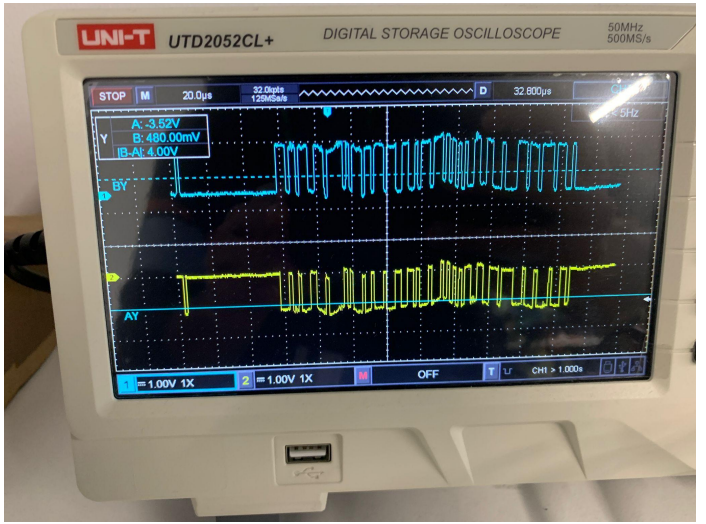
software engineering in automotive and mobility ecosystems

Input from speed sensor



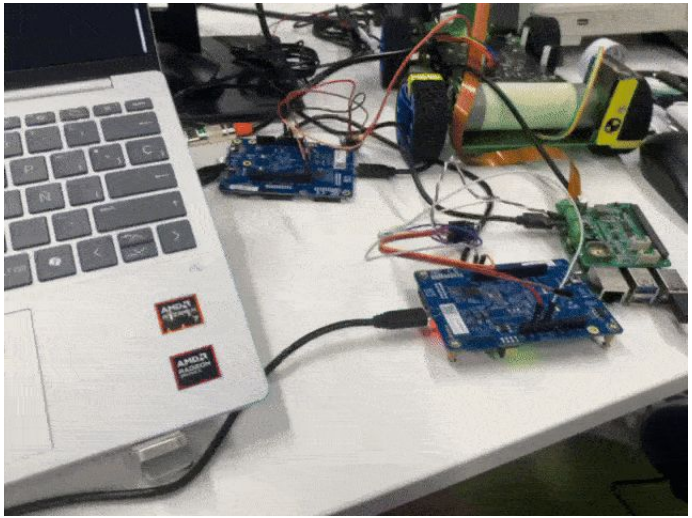
- Lights the green LED when there is no obstacle
- Turns off the green LED and lights the red LED when a block is detected

1st CAN Signal



sea|me
software engineering in automotive and mobility ecosystems

Establish CAN communication



```
// Timing: 36 MHz CAN clock → ~500 kbps nominal bit rate
hfdcan1.Init.NominalPrescaler = 4;
hfdcan1.Init.NominalSyncJumpWidth = 1;
hfdcan1.Init.NominalTimeSeg1 = 15;
hfdcan1.Init.NominalTimeSeg2 = 2;
```

- **Formula for CAN Bit Rate**

$$\text{Bit Rate} = \frac{f_{CAN\_clock}}{\text{Prescaler} \times \text{TimeQuantaPerBit}}$$

where:

$$\text{TimeQuantaPerBit} = 1(\text{SyncSeg}) + \text{TimeSeg1} + \text{TimeSeg2}$$

- **Substitute the values**
  - CAN clock = 36 MHz
  - Prescaler = 4
  - TimeSeg1 = 15
  - TimeSeg2 = 2

$$\text{TimeQuantaPerBit} = 1 + 15 + 2 = 18$$

$$\text{Bit Rate} = \frac{36\,000\,000}{4 \times 18} = 500\,000 \text{ bits/s} = 500kbps$$

sea|me

software engineering in automotive and mobility ecosystems

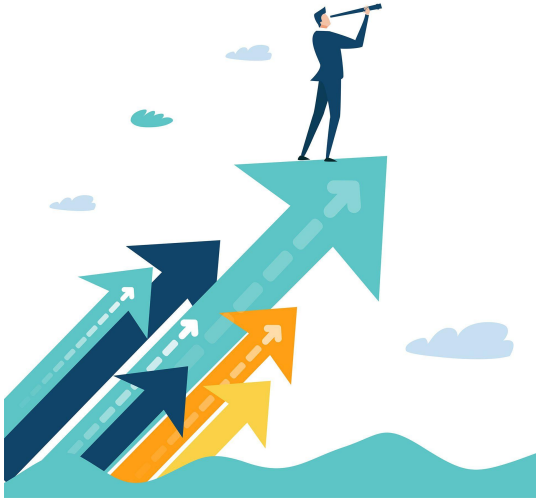
Challenges & Issues Faced

- Poor internet connection
- Complexity of STM32 microcontroller
- Understand CAN communication protocol

Challenges & Issues Faced

- Mastering prompt engineering to guide GenAI in executing complex Git/GitHub automations/completing templates

Next Steps & Improvements

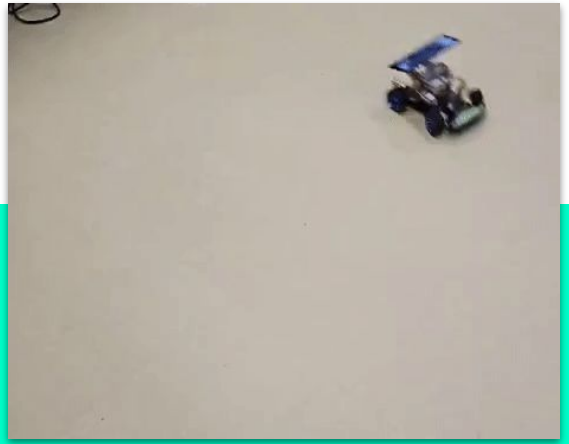


- Migrate from PI OS (debian Trixie) to AGL
- Read, analyze and send data using Threadx

What we learned

- Peer to peer learning is still one of the most valuable ways of learning
- Start simple – focus on understanding first, then scale up
- Being patient and resilient

Thank you



sea|me

software engineering in automotive
and mobility ecosystems