Zephyr & LLMs

The Good, the Bad, ...and the Hallucinated

Benjamin Cabé
benjamin@zephyrproject.org

Zephyr Meetup Brussels
Jan 30, 2026

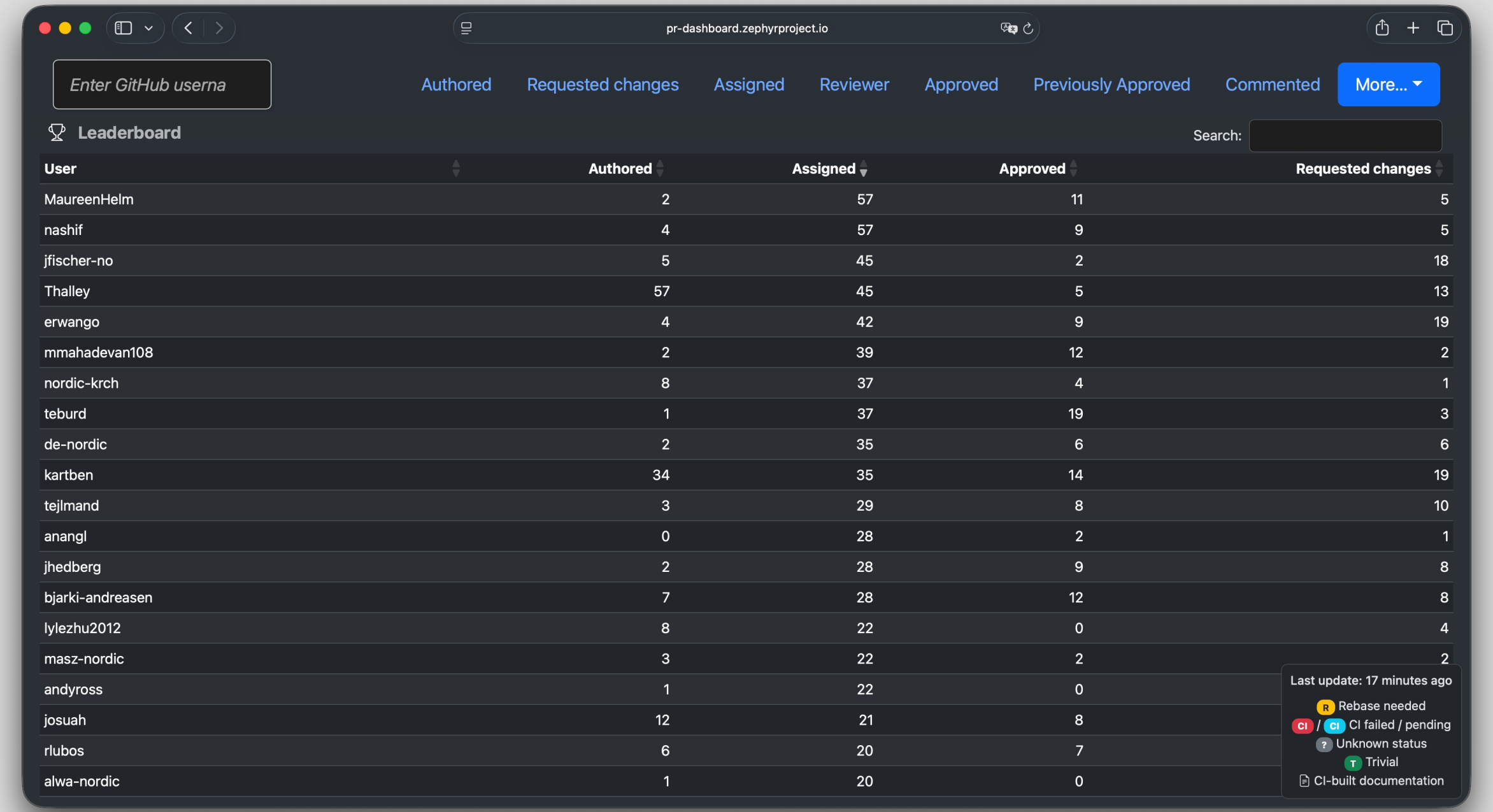
AGENDA

- LLMs what?
- **Reviewing** code
- **Writing** code
  - Drivers
  - Boards
  - Tests
- Writing **documentation**
- Did you say “the ugly”?

LLMS != AGENTS

LLM	AI Agent
Generates text/code from a prompt and context	LLM + **tools**
No direct access to "live" source code, CLI, hardware, …	Human-in-the-loop – you can validate/approve actions
Can sound very confident …and be very wrong!	Stateful

REVIEWING ZEPHYR CODE?


pr-dashboard.zephyrproject.io
Enter GitHub userna
Authored
Requested changes
Assigned
Reviewer
Approved
Previously Approved
Commented
More...
Leaderboard
Search:
User
Authored
Assigned
Approved
Requested changes
MaureenHelm 2 57 11 5
nashif 4 57 9 5
jfischer-no 5 45 2 18
Thalley 57 45 5 13
erwango 4 42 9 19
mmahadevan108 2 39 12 2
nordic-krch 8 37 4 1
teburd 1 37 19 3
de-nordic 2 35 6 6
kartben 34 35 14 19
tejlmand 3 29 8 10
anangl 0 28 2 1
jhedberg 2 28 9 8
bjarki-andreasen 7 28 12 8
lylezhu2012 8 22 0 4
masz-nordic 3 22 2 2
andyross 1 22 0
josuah 12 21 8
rlubos 6 20 7
alwa-nordic 1 20 0
Last update: 17 minutes ago
R Rebase needed
CI / CI CI failed / pending
? Unknown status
T Trivial
CI-built documentation

CODE REVIEWS

- LOTS of pull requests (1,200+ currently opened)
- Context switching / cognitive load is real
- Stuff slips through
  - Typos, ”nits”, documentation gaps
  - Corner cases, off-by-one, …

AUTOMATED CHECKS IN ZEPHYR CI

- Compliance checks
- SonarQube
- Basic spellcheck
- Linters
- …

They still don’t catch everything!

OPTIONS AVAILABLE FOR CODE REVIEWS

- Just paste the diff into ChatGPT/Gemini/Claude and ask for a review – pretty generic but still catches a lot
- Get an "agent" to review from your IDE
- Copilot review right from GitHub UI

NO NEED TO OVERTHINK YOUR PROMPT!

"Spot nasty bugs (incl. spelling typos) in this patch.
Think about stuff that SCA tools would not catch.
When you're done, please provide a concise table of your findings, most critical first

<< PASTE DIFF/PATCH >>"

NO NEED TO OVERTHINK YOUR PROMPT!

"Spot **nasty bugs** (incl. spelling typos) in this patch.
Think about **stuff that SCA tools would not catch**.
When you're done, please provide a **concise** table of your findings, **most critical first**

<< PASTE DIFF/PATCH >>"

https://gemini.google.com/share/a46e41a2d6df

This driver for the **MAX2253x** is a solid start, but there are several structural and logic issues that standard Static Code Analysis (SCA) would likely miss—specifically regarding thread lifecycle, interrupt management, and Zephyr's asynchronous patterns.

The most critical finding is a logic error that could permanently disable the driver upon a single SPI communication failure.

Review Findings

Severity	Location	Finding
Critical	`adc_max2253x.c` (Thread)	**Permanent Thread Exit:** In `max2253x_acquisition_thread`, if `max2253x_read_raw_adc` returns an error (e.g., transient SPI glitch), the loop hits `break`. This terminates the thread, rendering the ADC driver non-functional until a system reboot.
High	`adc_max2253x.c` (Stream)	**IRQ Stalling:** In `max2253x_stream_irq_handler`, the GPIO interrupt is disabled via `gpio_pin_interrupt_configure_dt`. However, it is only re-enabled in `adc_max2253x_submit_stream`. In a continuous streaming scenario, the next interrupt will never fire because the handler itself doesn't re-enable it after the RTIO submission.
Medium	`adc_max2253x.c` (SPI)	**Confusing Write Logic:** In `max2253x_write_reg`, `value` is updated with `sys_cpu_to_be16` **after** the `tx_buf` array (which holds `&value`) is initialized. While technically valid in C as the pointer remains the same, it is highly unconventional and error-prone for future maintenance.
Medium	`adc_max2253x.c` (Math)	**Precision Loss:** The sensitivity calculation `(vref_mv * (scale - 1)) / scale * 1000 / scale` uses multiple integer divisions. This can lead to significant rounding errors before the final value is used in $Q31$ conversion.
Low	`Kconfig.max2253x`	**Dead Config:** `ADC_MAX2253X_ASYNC_THREAD_INIT_PRIORITY` is defined in Kconfig but never referenced in the C code or build system.
Minor	`Kconfig.max2253x`	**Typo:** In `ADC_MAX2253X_STREAM` help text, "**Inplies** the use of interrupts" should be "**Implies**".
Minor	`adc_max2253x.c`	**Missing Check:** `max2253x_read_raw_adc` is called in the thread with `ARRAY_SIZE(adc_raw)`, but the function doesn't verify if `data->channels` actually fits within that size (though protected by constant defines, it's "blind" coding).

This driver for the **MAX2253x** is a solid start, but there are several structural and logic issues that standard Static Code Analysis (SCA) would likely miss—specifically regarding thread lifecycle, interrupt management, and Zephyr's asynchronous patterns.

The most critical finding is a logic error that could permanently disable the driver upon a single SPI communication failure.

Review Findings

Severity	Location	Finding
Critical	`adc_max2253x.c` (Thread)	**Permanent Thread Exit:** In `max2253x_acquisition_thread`, if `max2253x_read_raw_adc` returns an error (e.g., transient SPI glitch), the loop hits `break`. This terminates the thread, rendering the ADC driver non-functional until a system reboot.
High	`adc_max2253x.c` (Stream)	**IRQ Stalling:** In `max2253x_stream_irq_handler`, the GPIO interrupt is disabled via `gpio_pin_interrupt_configure_dt`. However, it is only re-enabled in `adc_max2253x_submit_stream`. In a continuous streaming scenario, the next interrupt will never fire because the handler itself doesn't re-enable it after the RTIO submission.
Medium	`adc_max2253x.c` (SPI)	**Confusing Write Logic:** In `max2253x_write_reg`, `value` is updated with `sys_cpu_to_be16` **after** the `tx_buf` array (which holds `&value`) is initialized. While technically valid in C as the pointer remains the same, it is highly unconventional and error-prone for future maintenance.
Medium	`adc_max2253x.c` (Math)	**Precision Loss:** The sensitivity calculation `(vref_mv * (scale - 1)) / scale * 1000 / scale` uses multiple integer divisions. This can lead to significant rounding errors before the final value is used in $Q31$ conversion.
Low	`Kconfig.max2253x`	**Dead Config:** `ADC_MAX2253X_ASYNC_THREAD_INIT_PRIORITY` is defined in Kconfig but never referenced in the C code or build system.
Minor	`Kconfig.max2253x`	**Typo:** In `ADC_MAX2253X_STREAM` help text, "**Inplies** the use of interrupts" should be "**Implies**".
Minor	`adc_max2253x.c`	**Missing Check:** `max2253x_read_raw_adc` is called in the thread with `ARRAY_SIZE(adc_raw)`, but the function doesn't verify if `data->channels` actually fits within that size (though protected by constant defines, it's "blind" coding).

This driver for the **MAX2253x** is a solid start, but there are several structural and logic issues that standard Static Code Analysis (SCA) would likely miss—specifically regarding thread lifecycle, interrupt management, and Zephyr's asynchronous patterns.

The most critical finding is a logic error that could permanently disable the driver upon a single SPI communication failure.

Review Findings

Severity	Location	Finding
Critical	`adc_max2253x.c` (Thread)	**Permanent Thread Exit:** In `max2253x_acquisition_thread`, if `max2253x_read_raw_adc` returns an error (e.g., transient SPI glitch), the loop hits `break`. This terminates the thread, rendering the ADC driver non-functional until a system reboot.
High	`adc_max2253x.c` (Stream)	**IRQ Stalling:** In `max2253x_stream_irq_handler`, the GPIO interrupt is disabled via `gpio_pin_interrupt_configure_dt`. However, it is only re-

```
+static void max2253x_acquisition_thread(void *p1, void *p2, void *p3)
+{
+    ARG_UNUSED(p2);
+    ARG_UNUSED(p3);
+
+    struct max2253x_data *data = p1;
+    uint16_t adc_raw[MAX2253X_CH_COUNT];
+    int ret;
+
+    while (true) {
+        k_sem_take(&data->acq_sem, K_FOREVER);
+
+        ret = max2253x_read_raw_adc(data->dev, adc_raw, ARRAY_SIZE(adc_raw));
+        if (ret) {
+            LOG_ERR("Failed to read raw samples (err %d)", ret);
+            adc_context_complete(&data->ctx, ret);
+            break;
+        }
+
+        for (int i = 0; i < ARRAY_SIZE(adc_raw); i++) {
+            if (data->channels & BIT(i)) {
+                *data->buffer++ = adc_raw[i];
+            }
+        }
+
+        adc_context_on_sampling_done(&data->ctx, data->dev);
+    }
+}
```

```
+static void max2253x_acquisition_thread(void *p1, void *p2, void *p3)
+{
+    ARG_UNUSED(p2);
+    ARG_UNUSED(p3);
+
+    struct max2253x_data *data = p1;
+    uint16_t adc_raw[MAX2253X_CH_COUNT];
+    int ret;
+
+    while (true) {
+        k_sem_take(&data->acq_sem, K_FOREVER);
+
+        ret = max2253x_read_raw_adc(data->dev, adc_raw, ARRAY_SIZE(adc_raw));
+        if (ret) {
+            LOG_ERR("Failed to read raw samples (err %d)", ret);
+            adc_context_complete(&data->ctx, ret);
+            break;
+        }
+
+        for (int i = 0; i < ARRAY_SIZE(adc_raw); i++) {
+            if (data->channels & BIT(i)) {
+                *data->buffer++ = adc_raw[i];
+            }
+        }
+
+        adc_context_on_sampling_done(&data->ctx, data->dev);
+    }
+}
```

AGENTS.MD (AND THE LIKE)

- Providing the same basic instructions over and over again feels wrong?
- Multiple standards exist (ex. agents.md, copilot-instructions.md, “skills”, …) to provide them directly in the repo
- Still a bit of a hit-or-miss: many LLMs/agents do not systematically use the provided instructions

WRITING ZEPHYR DRIVERS?

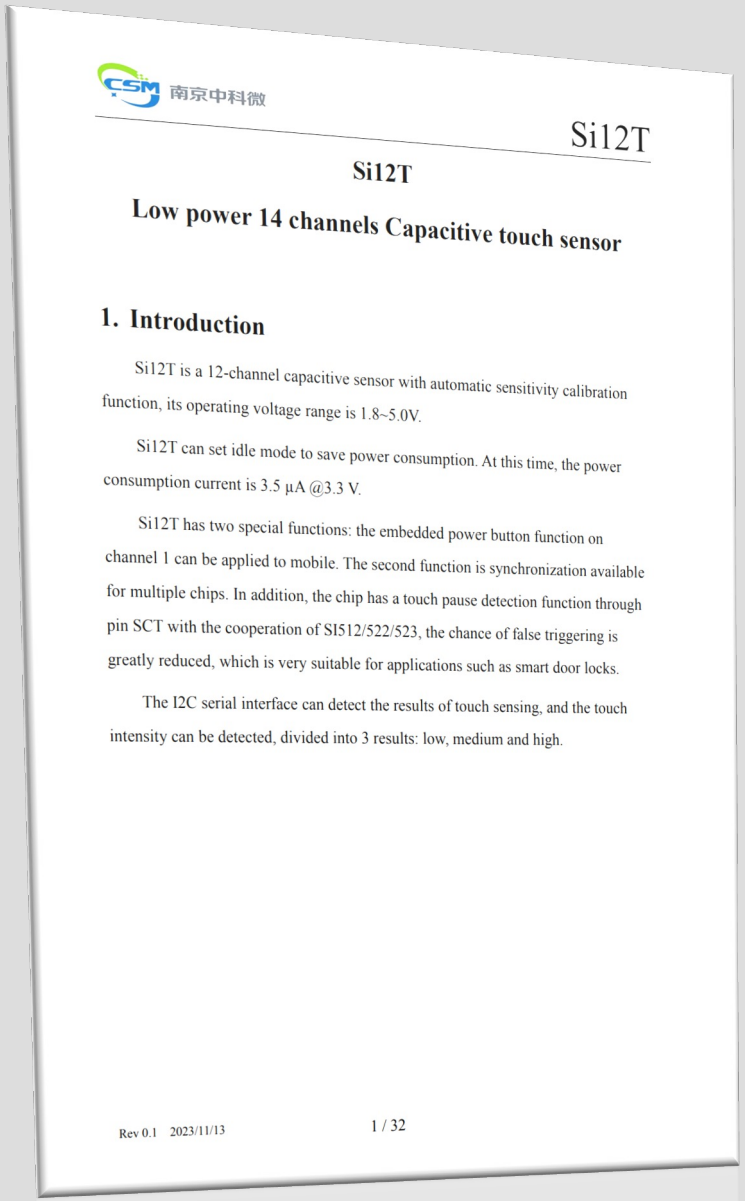
ZEPHYR DRIVERS

Zephyr drivers involve a fair amount of boilerplate code (and many opportunities for silly copy-paste errors!)

- Devicetree binding
- Kconfig
- Macros for hardware registers
- Actual driver code: init macro, logging, …

BOOTSTRAPPING/WRITING A DRIVER

- Provide datasheet
- Point to existing drivers in-tree that are “close enough”
- Explicitly ask for code reuse from other existing drivers, macros etc.



Si12T

Low power 14 channels Capacitive touch sensor

1. Introduction

Si12T is a 12-channel capacitive sensor with automatic sensitivity calibration function, its operating voltage range is 1.8~5.0V.

Si12T can set idle mode to save power consumption. At this time, the power consumption current is 3.5 μA @3.3 V.

Si12T has two special functions: the embedded power button function on channel 1 can be applied to mobile. The second function is synchronization available for multiple chips. In addition, the chip has a touch pause detection function through pin SCT with the cooperation of SI512/522/523, the chance of false triggering is greatly reduced, which is very suitable for applications such as smart door locks.

The I2C serial interface can detect the results of touch sensing, and the touch intensity can be detected, divided into 3 results: low, medium and high.

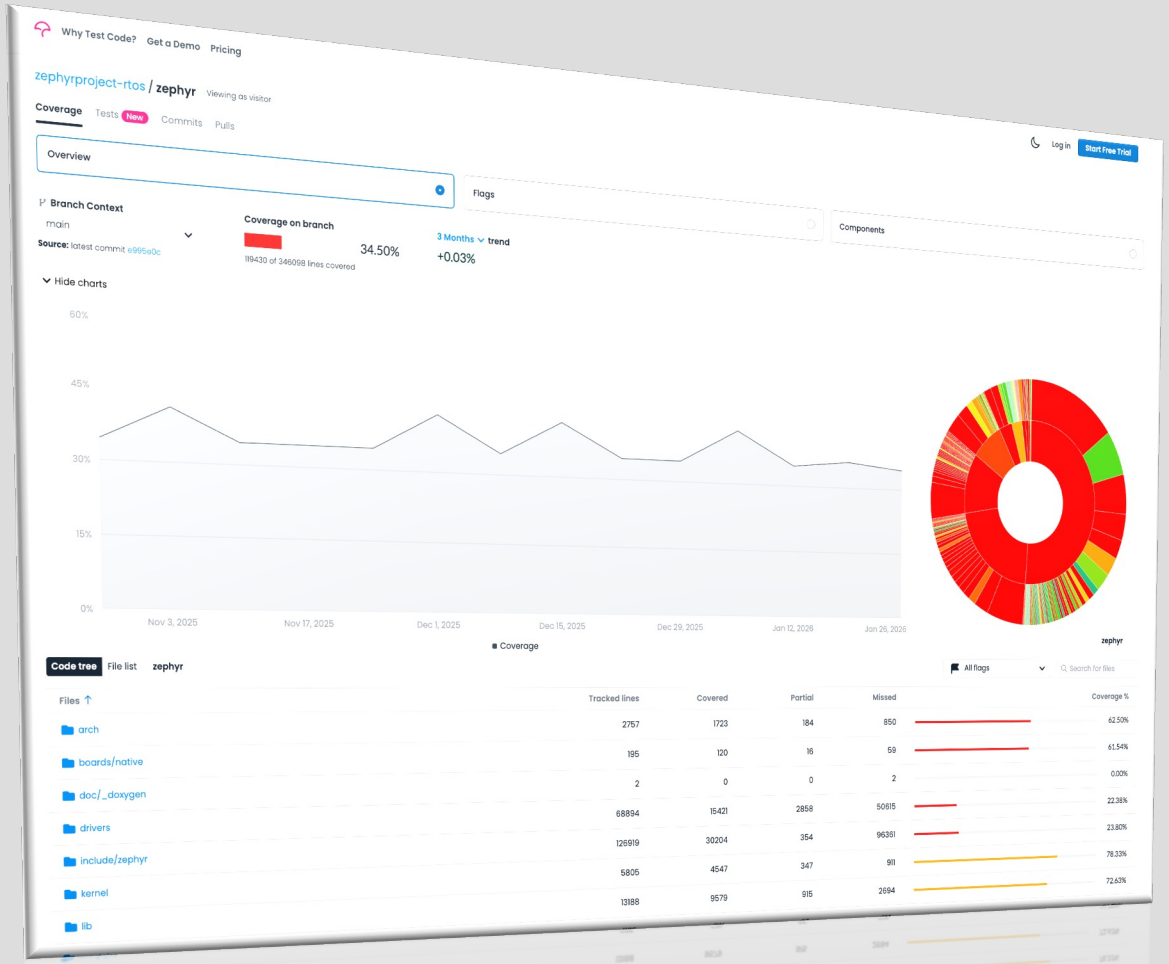
DEMO

WRITING BOARD DEFINITIONS?

TIPS & TRICKS

- Explicitly provide the datasheet
  - “Here’s the PDF, use pdftotext to convert if needed”
- Add schematics (images) to the prompt/context
- Point to similar boards as good reference / starting points
  - That’s we recommend in the Zephyr documentation anyway!

TESTS



DEMO

https://app.codecov.io/gh/zephyrproject-rtos/zephyr/blob/main/subsys%2Finput%2Finput_double_tap.c

YOU CAN EVEN POINT TO THE COVERAGE REPORT DIRECTLY ☺

"Use coverage report in native_sim.json and increase test coverage for subsys/jwt/jwt.c.

Keep tests concise, add comments only when needed and when the test code doesn't speak for itself"

MORE/BETTER TESTS?

- Helps refine the API contract when it is not clearly documented
- Helps spot dead code, refactoring opportunities

DOCUMENTATION

WRITING DOCUMENTATION IS HARD

- Documentation is often "just" another view/representation of the code, and LLMs are very good at this type of "translation"
- They are also really good at following examples, e.g. existing files with good documentation, and/or applying Doxygen guidelines.

THE UGLY?

THE UGLY (?)

- Copyright
- Environmental impact
- Increased “noise level”
- From cognitive load …to metacognitive laziness?

IN A NUTSHELL

- Give it a try and make your own conclusions ☺
- LLMs don’t make engineers dumb(er)… uncritical use does!
- Really just another tool to add to your belt