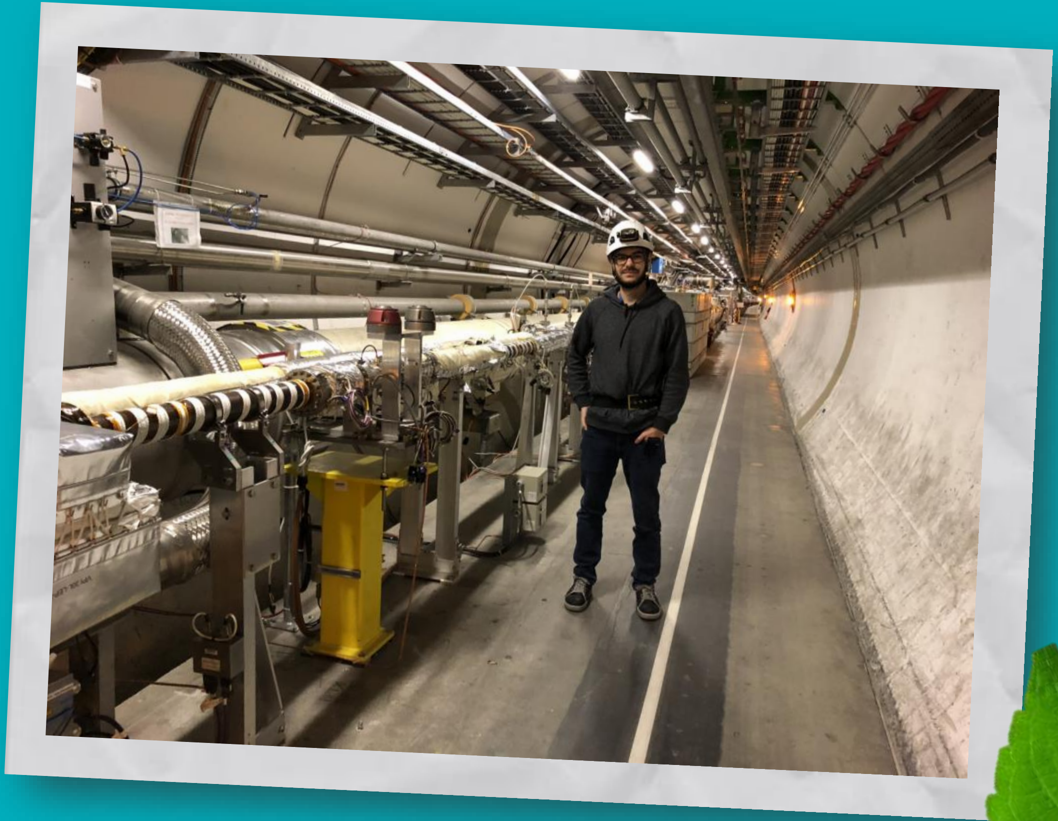
GARDENA®

Growing IoT with Open Source: Zephyr OS at Gardena

Zephyr Meetup | Genoble, 26.03.2025

- Studied electronic engineering in Karlsruhe
- at GARDENA since 5 years
- Interested in
  - Linux, BSD
  - Self-hosting
  - Networks, IPv6
  - IoT and Open Source
- GitHub **mlasch**



Marc Lasch

Embedded Developer

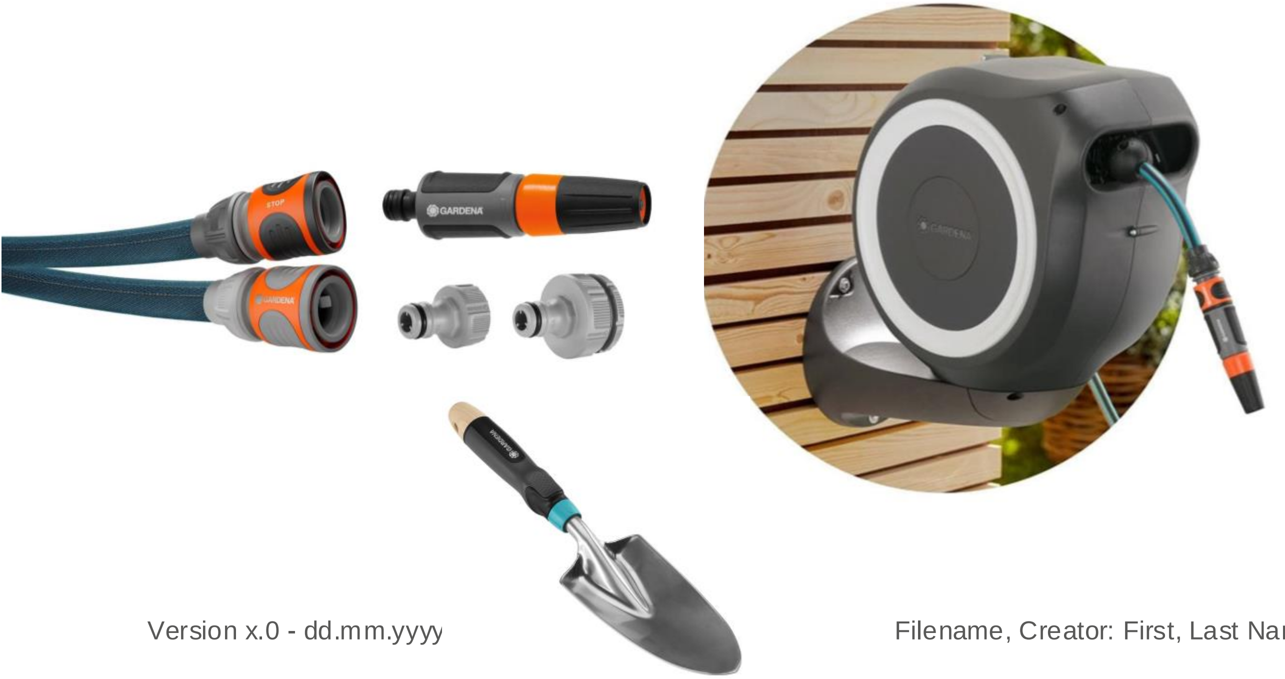
Outline

- The GARDENA smart system
- The journey to Zephyr
- How we use Zephyr in a multi-team setup
  - Requirements and goals
  - Zephyr Releases
  - Kconfig setup
  - Testing

GARDENA and the smart system

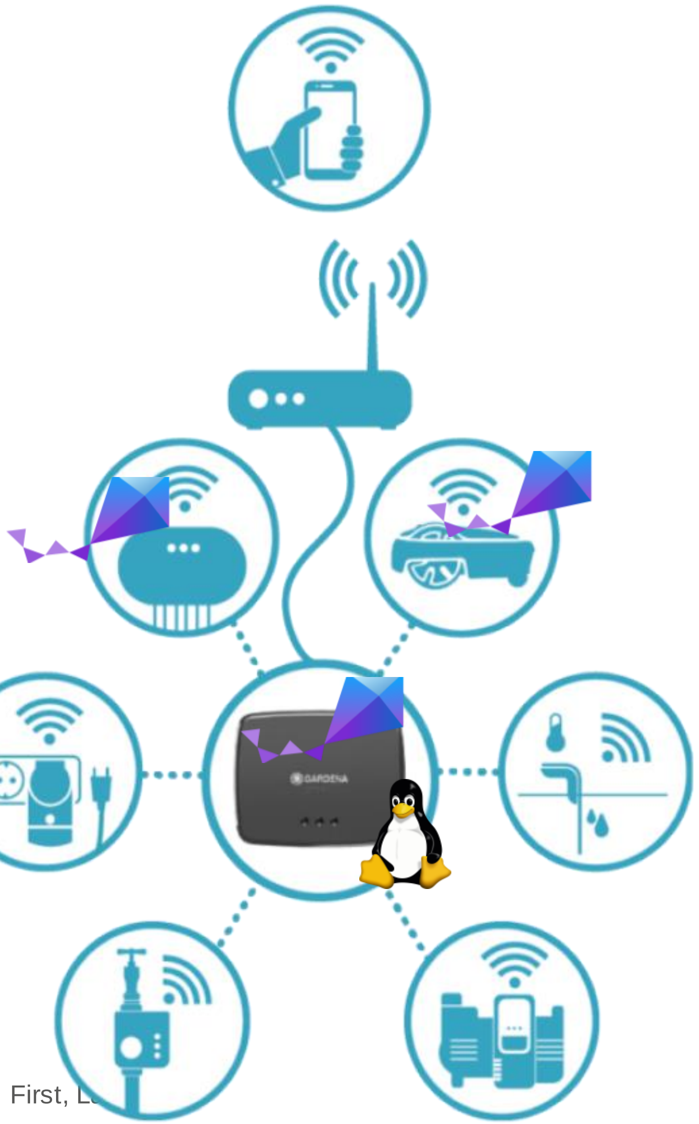
- GARDENA is a division of the Husqvarna Group

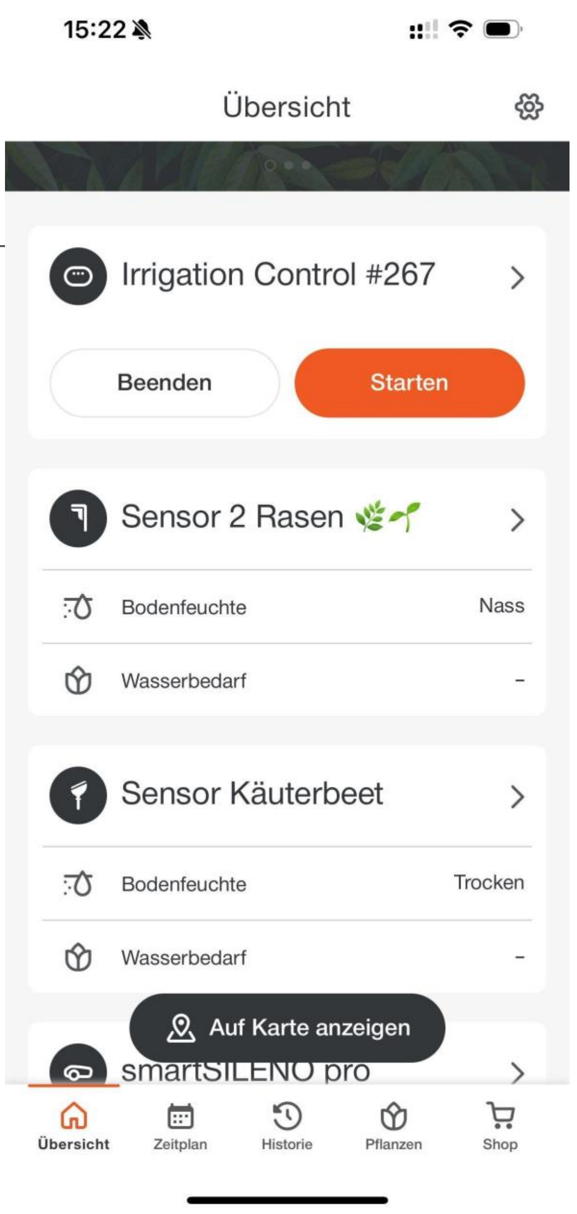
- some well-known Gardena products
  - watering hoses and connectors
  - mowers
  - shovels and other garden tools
  - irrigation



GARDENA and the smart system

- **Automated yard care**
- Plant watering and environment sensing
- Lawn mowing robots




15:22

Übersicht

Irrigation Control #267

Beenden | Starten

Sensor 2 Rasen 🌿🌱

Bodenfeuchte	Nass
Wasserbedarf	-

Sensor Käuterbeet

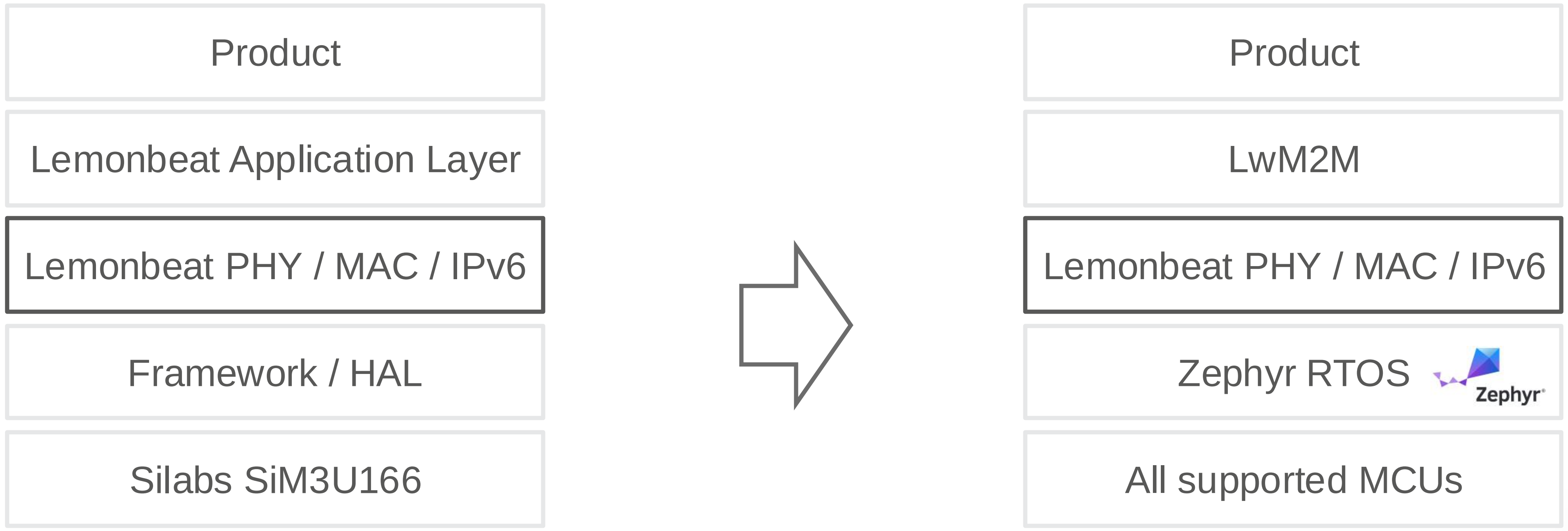
Bodenfeuchte	Trocken
Wasserbedarf	-

Auf Karte anzeigen

smartSILENO pro

The journey to Zephyr RTOS

- Smart system started with proprietary framework
- Look out to replace the stack with **open standards**

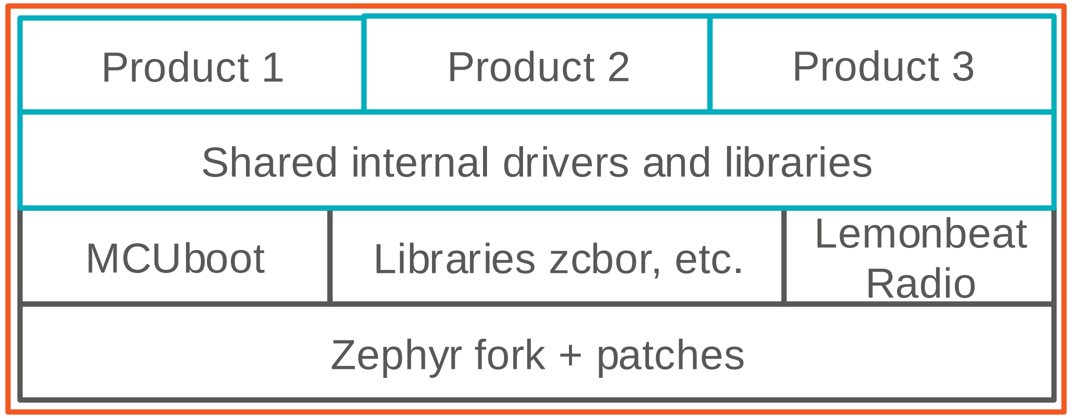


The journey to Zephyr RTOS

- Requirements and goals
  - Long term support and maintainability
  - Vendor independent (Open Source)
  - No binary blobs (if possible)
  - Not just an RTOS, provides lots of libraries
  - Supports Linux as primary development platform
- Initial commit and first examples with Zephyr 2.4
- First release on Zephyr 3.2
- Keep close to upstream, follow every release (if possible)
- Based on Zephyr project topology T2
  - https://github.com/zephyrproject-rtos/example-application

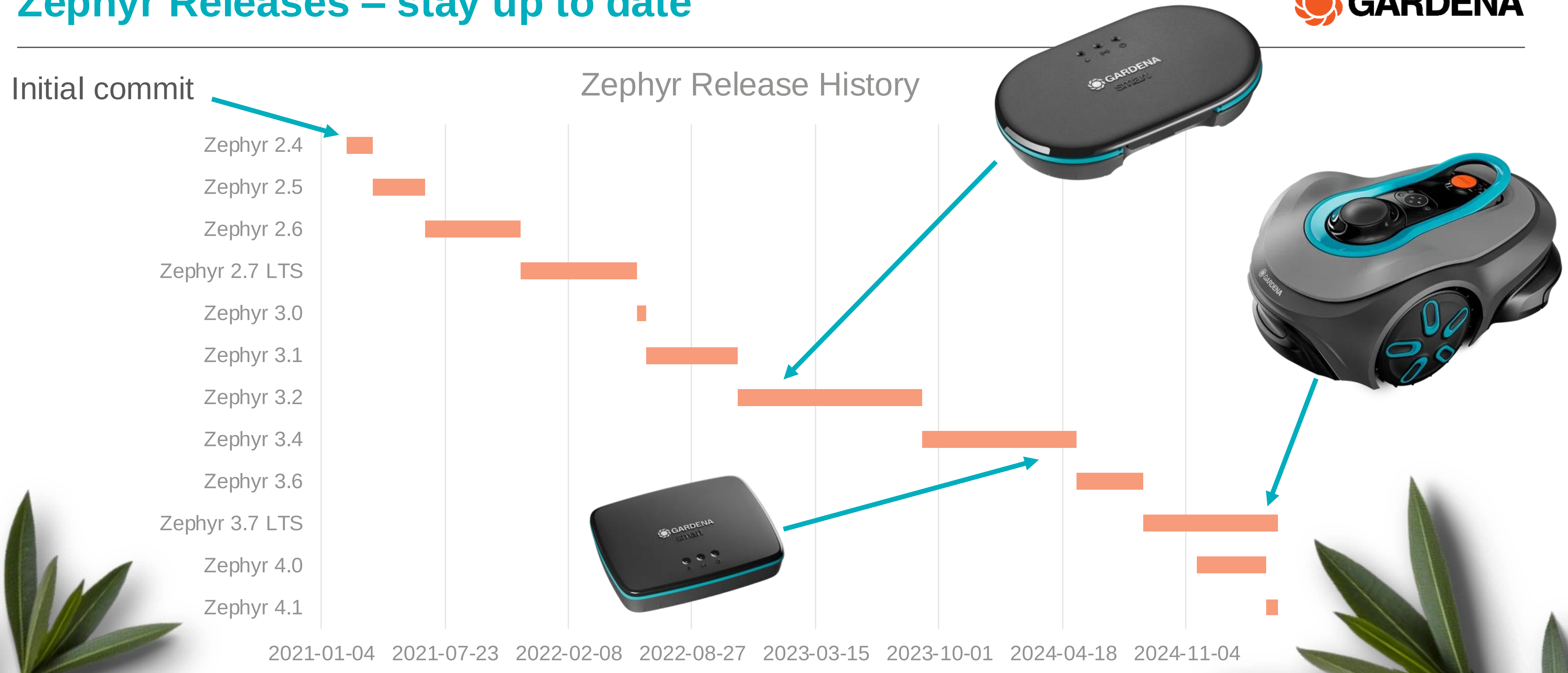
How we use Zephyr at GARDENA

- **Development setup – a platform approach**
  - Multiple applications in a single repo (T2)
  - Out-of-tree shared modules, libraries and drivers
  - Single Zephyr fork with in-tree patches, kept as close as possible to upstream
    - Regular rebase of downstream patches to new releases
  - nRF Connect inspired commit messages [sg toup], [sg fromlist], [sg fromtree], [sg noup]
  - Workflow based on "InnerSource" strategy
    - Code ownership by teams, required reviews enforce by Azure DevOps
    - Accept code from all developers everywhere, all have access
    - Everyone is encouraged to work close to main



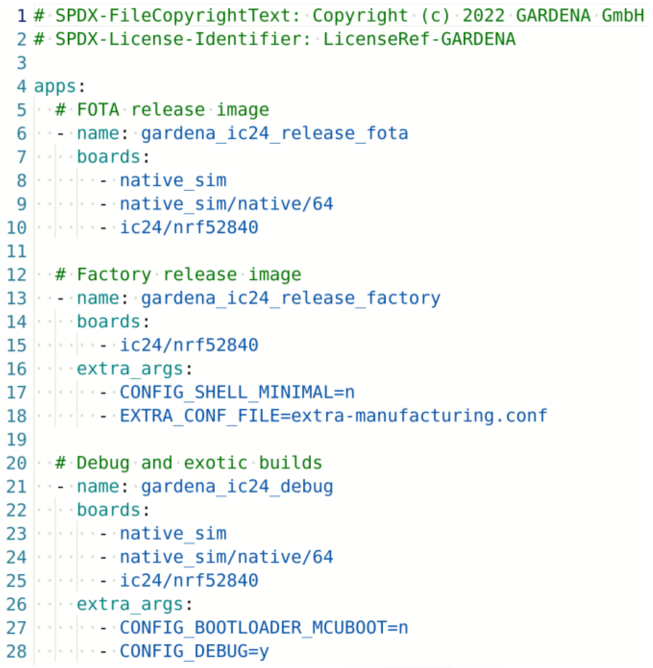
modular approach in a single repository

Zephyr Releases – stay up to date



Kconfig

- Extra / overlay files: Located in the app directory of every application, provides a collection of Kconfig settings for specific usages e.g. manufacturing, ci-tests.
- Kconfig priorities
  - **Kconfig**
  - board specific symbols **nrf52840dk_nrf52840_defconfig**
  - Application specific config **prj.conf**
  - Application board specific overlays **boards/nrf52840dk_nrf52840.conf**
  - snippet extra-conf
  - cmake arguments –**D**
    - **EXTRA_CONF_FILE**
    - **CONFIG_X**
- **configdefault** vs. **select** vs. **imply** vs. set in **X.conf**
- Strategy introduce our own Kconfig or use an existing one from Zephyr?
- Build: Twister vs. custom script



```
# SPDX-FileCopyrightText: Copyright (c) 2022 GARDENA GmbH
# SPDX-License-Identifier: LicenseRef-GARDENA

apps:
  # FOTA release image
  - name: gardena_ic24_release_fota
    boards:
      - native_sim
      - native_sim/native/64
      - ic24/nrf52840

  # Factory release image
  - name: gardena_ic24_release_factory
    boards:
      - ic24/nrf52840
    extra_args:
      - CONFIG_SHELL_MINIMAL=n
      - EXTRA_CONF_FILE=extra-manufacturing.conf

  # Debug and exotic builds
  - name: gardena_ic24_debug
    boards:
      - native_sim
      - native_sim/native/64
      - ic24/nrf52840
    extra_args:
      - CONFIG_BOOTLOADER_MCUBOOT=n
      - CONFIG_DEBUG=y
```

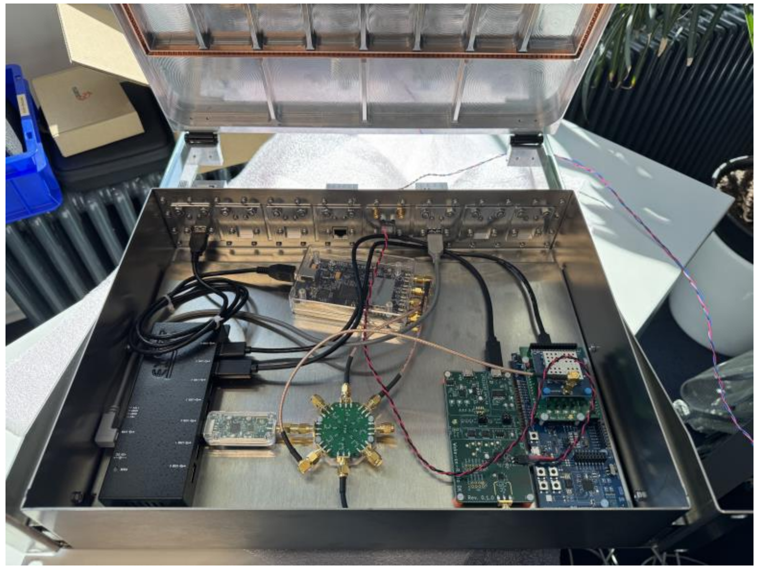
Testing

- **A primary objective is to maintain the codebase in a state that is always ready for release**

- Typical test strategies
  - **Unit tests** (PR, nightly)
  - **Component tests** (PR, nightly)
  - **Integration tests** (nightly)
  - End-to-End tests

- **Challenges**
  - Technical: flakiness
  - Organizational: Maintain and keep up

- EOSS Talk about our testing setup https://youtu.be/dKqBrwjR3Lo



Testing – Linting

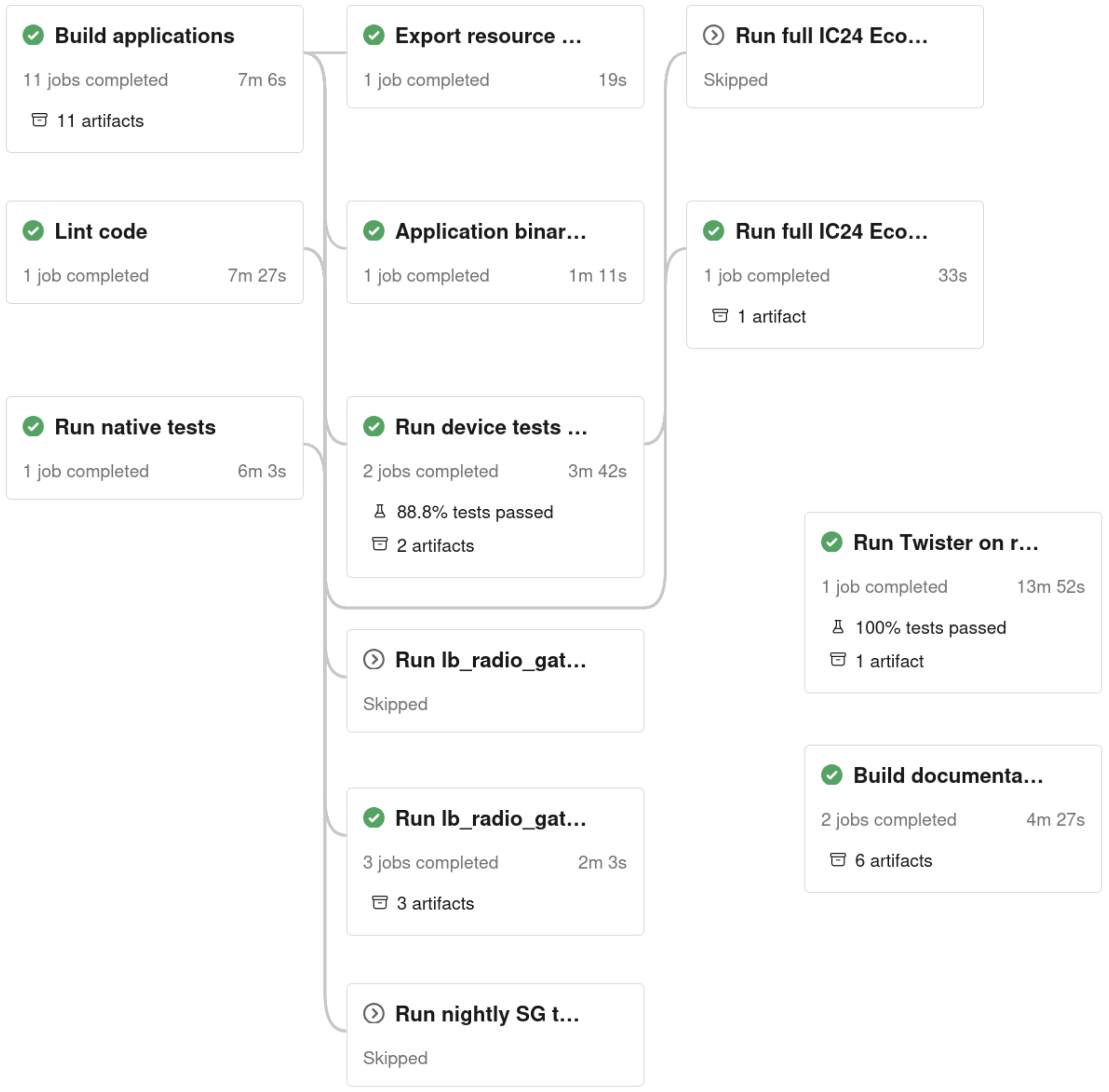
Second most important: **Linting**

- Have tools and formatters run locally (Linux)
  - **clang-format**
  - **checkpatch**
  - **gitlint**
  - **reuse**
  - **markdownlint**
  - **gersemi**
  - **shellcheck**
  - **flake8, isort, ruff**
  - **ansible-lint**
  - **Other scripts: max file size (500KiB), end-of-line white space check, tabs vs. spaces**

- All linters can be ignored if necessary

- Other checks
  - pahole
  - Build statistics: Track memory consumption
  - TODO: Kconfig verification

Testing – CI/CD


Build applications
11 jobs completed 7m 6s
11 artifacts

Lint code
1 job completed 7m 27s

Run native tests
1 job completed 6m 3s

Export resource …
1 job completed 19s

Application binar…
1 job completed 1m 11s

Run device tests …
2 jobs completed 3m 42s
88.8% tests passed
2 artifacts

Run lb_radio_gat…
Skipped

Run lb_radio_gat…
3 jobs completed 2m 3s
3 artifacts

Run nightly SG t…
Skipped

Run full IC24 Eco…
Skipped

Run full IC24 Eco…
1 job completed 33s
1 artifact

Run Twister on r…
1 job completed 13m 52s
100% tests passed
1 artifact

Build documenta…
2 jobs completed 4m 27s
6 artifacts

Thank You!