Zephyr® Project

Developer Summit

Zephyr® Project
Developer Summit

How Eclipse Oniro Uses Zephyr – Lessons Learned

Stefan Schmidt, Principal Solution Architect, Huawei OSTC

EOSS 2023

#EMBEDDEDOSSUMMIT

Agenda

- Oniro Overview
- OpenHarmony Overview
- 5 Use cases and Lessons Learned
- Future Roadmap Discussion

Scope

- Where Zephyr is used in Eclipse Oniro and how
- The perspective will be as a user and consumer of Zephyr
- Needs for integration, stability and maintenance
- Also new features and bug fixes

Oniro Overview

The Eclipse Oniro project is centered around the Oniro Working Group at the Eclipse Foundation https://oniroproject.org

"Oniro is an Eclipse Foundation project focused on the development of a **distributed open source operating system** platform that enables interoperability of consumer devices, regardless of brand, make, or model.The platform is designed to be compatible with a broad range of embedded operating system environments, including OpenHarmony, an open source operating system specified and hosted by the OpenAtom Foundation."

Oniro Overview

- The promise to support devices big and small does mean it supports different kernels, Linux as well as RTOS systems.
  - Linux build with Yocto
  - Zephyr: meta-zephyr
  - LiteOS: meta-liteosm

- Oniro 2.0 end of last year as horizontal platform

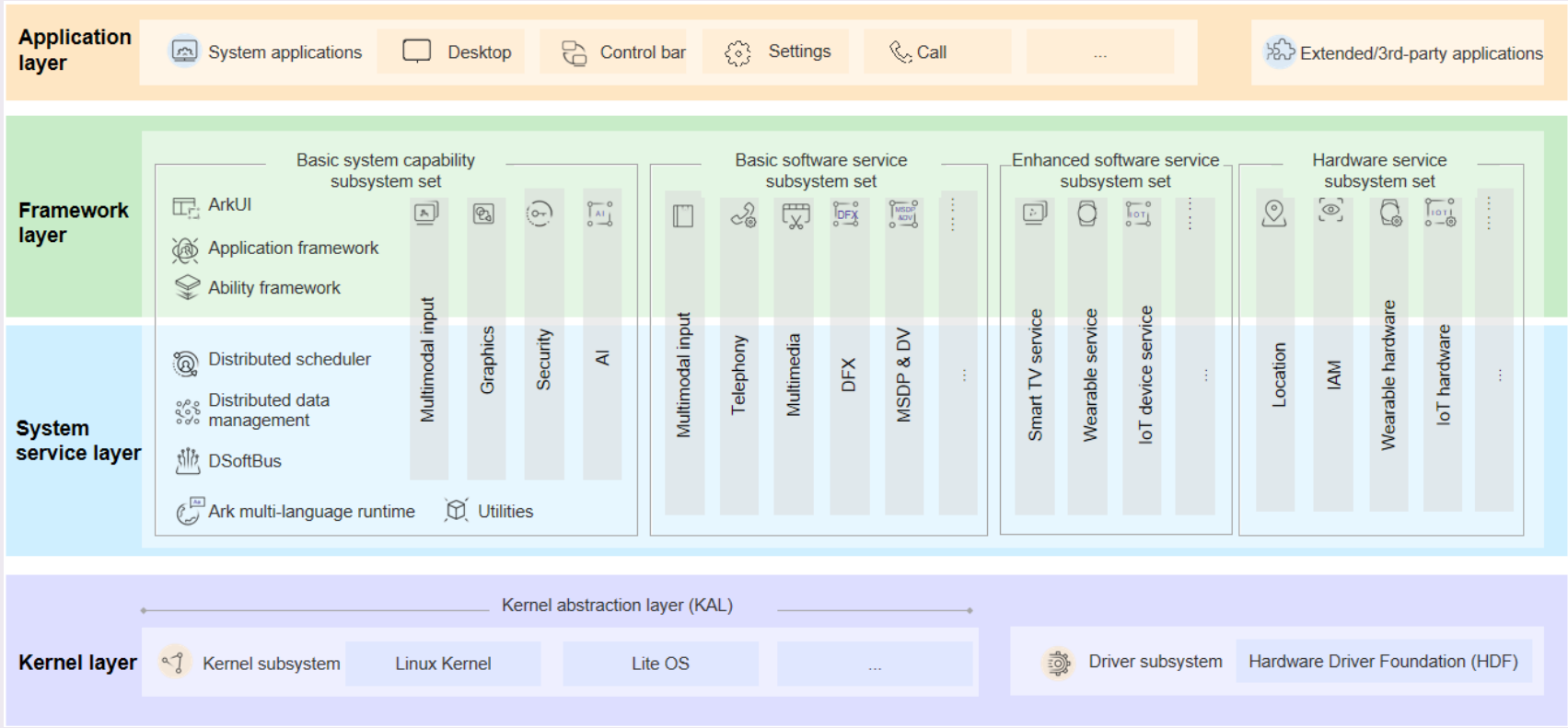
2023 focus on vertical solution: OpenHarmony

OpenHarmony Overview

- Hosted and developed at the OpenAtom Foundation
- Open Source foundation of HarmonyOS
- Apache 2.0 licensed

- Mini, small and standard systems
- Linux and Lite OS

- 274 certified products from 108 manufacturers
- Chinese market
- Expanding markets



OpenHarmony Overview

Three different sytem types

- Mini (Cortex-M, >= 128 KiB, RTOS) system
- Small (Cortex-A, >= 1 MiB, RTOS or Linux) system
- Standard (Cortex-A, >= 128 MiB, Linux) system
- Linux and LiteOS

- DSoftBus, distributed scheduler and data management
- Kernel Abstraction Layer (KAL)
- Hardware Driver Foundation (HDF)

Use Cases & Lessons Learned

Use Case 1: meta-zephyr

- Use of meta-zephyr to build with bitbake through Yocto/OE
- Ensure unified developer workflow for building

Problems:

- Layer was mixing machine support, distro policies and more in one
- Only a handfull of machines supported in comparison to what Zephyr actually supports

Solution:

- Split meta-zepyhr into meta-zephyr-bsp and meta-zephyr-core layers
- $ bitbake generate-zephyr-machines
- Automatically generate machines from west boards with some Cmake hacks
- Machines need to be manually verified, tested and added to CI afterwards

Use Case 1: meta-zephyr

- Using Zephyr through meta-zephyr has imposed limits in available machines
- For us the extra steps to unify our developer flow was worth the work

- Splitting the layers, generating machines, adding application and sample recipes
- ~68 patches upstreamed into meta-zephyr

Use Case 2: Connectivity and Graphics

- Hardware support for Arduino Nano 33 BLE, Nitrogen 96 and others
- Subsystems used besides core: OpenThread, CoAP and LVGL

Problems:

- Version mismatches in OpenThread spinnel protocol (with Linux host)
- LVGL version to old to work as intended for us (LVGL used in Linux as well as Zephyr)

Solution:

- Switching to a newer Zephyr version solved some OpenThread problems
- LVGL version bump and further revamp in upstream LVGL module

Use Case 2: Connectivity and Graphics

- LVGL subsystem update resulted in revamp upstream
- Smaller fixes where needed (persistend storage in partion table)
- Matching protocol versions for OpenThread needed to be coordinated for updates

- The core functionality and bringup worked smoothly for us

- ~20 patches upstreamed into Zephyr

Use Case 3: Zephyr and Linux Code Sharing

- Shared code base between Linux and Zephyr
- EDDIE project runs on top of both
- C++ code base, using CoAP, JSON and OpenThread
- Clearly not a Zephyr problem, but a niche use case

Problems:

- Linux and Zephyr offer different APIs for CoAP (libcoap vs native)
- OpenThread configuration interface

Solution:

- Duplicated code, abstrated where possible

Use Case 3: Zephyr and Linux Code Sharing

- Is this a common request?
- Do others split projects between Linux and Zephyr from the start?

- Full abstraction layer out of scope for Zephyr project

Use Case 4: Blueprints (doorlock & CATS)

- Blueprints involving sensors, actors, wireless and graphics
- Keypad, OpenThread, CoAP, LVGL and graphical assets

Problems:

- Demo hardware bringup time
- SRAM (256 KB) and flash (1 MB) size limitations for featureset

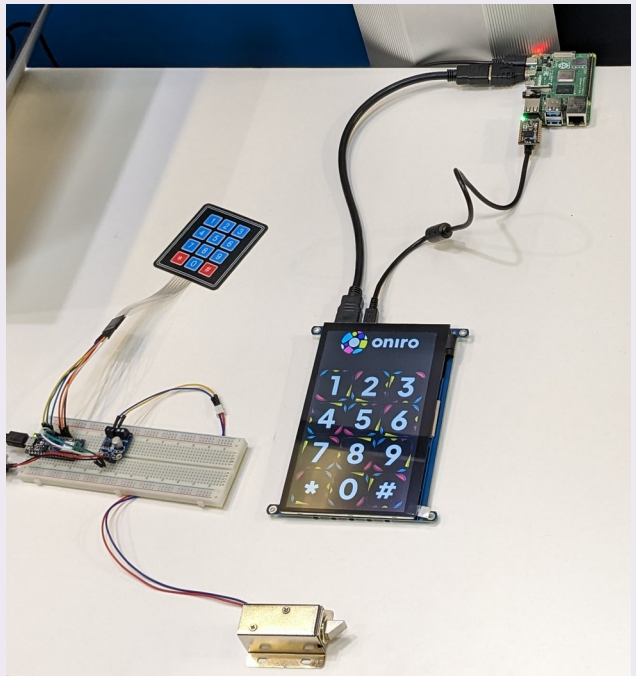
Solution:

- Selecting the right hardware components was easy due to wide range of driver support
- Switched early prototype from Zephyr to Linux

Use Case 4: Blueprints (doorlock & CATS)

- Doorlock still using Zephyr
- Motor driver, keypad, OpenThread, CoAP, application logic

- Context Aware TouchScreen switched from Zephyr to Linux
- Graphical application with different pages and graphical assets
- Running out of flash space
- Even without OpenThread subsystem



Use Case 5: IP Compliance

- Our IP compliance process goes beyond SBOM generation
- Manual audits on source for build artifacts
- Several issues spotted in our Zephyr target builds

Problems:

- Binaries with unclear licence situation (NXP, espressif HAL's)
- Proprietary font in LVGL module
- Unclear license (proprietary as well as open source) statement (mcuboot, nios2f)

Solution:

- IP audit team prepared list of potential issues
- Further analysed in the project and raised as upstream Zephyr tickets

Use Case 5: IP Compliance

Resolved:

- https://github.com/zephyrproject-rtos/zephyr/issues/46954 (hal_nxp)
- https://github.com/zephyrproject-rtos/zephyr/issues/48111 (lvgl module)

Still open (or stale closed):

- https://github.com/mcu-tools/mcuboot/issues/1490 (mcuboot)
- https://github.com/zephyrproject-rtos/zephyr/issues/51317 (nios2f)

Future Roadmap

Current Focus and Roadmap

- 2023 focus on enhancing OpenHarmony as verticial solution
- Application frameworks and ecosystem (e.g. ReactNative, Flutter)
- IDE Tooling
- System profiling and optimization
- Rust in security critical areas (e.g. web engine)

OpenHarmony on Zephyr?

- External Zephyr module to glue both systems together
- OpenHarmony already has RTOS support (LiteOS)
- Could extend platform support
- Porting effort around KAL and HDF
- Depends on interest of partner, customers and the Oniro working group
- Interested in feedback

Thank you!

Join us @
oniroproject.org