Zephyr Project: Results from Applying Open Source Best Practices in Embedded

Kate Stewart, VP Dependable Embedded Systems

#OSSummit  @_kate_stewart
Zephyr Project

- **Open source** real time operating system
- **Developer friendly** with vibrant community participation
- Built with **safety and security** in mind
- **Broad SoC, board and sensor support.**
- **Vendor Neutral** governance
- **Permissively licensed** - Apache 2.0
- **Complete**, fully integrated, highly configurable, **modular** for **flexibility**
- **Product** development ready using LTS includes **security updates**
- **Certification** ready with Zephyr Auditable

Open Source, RTOS, Connected, Embedded
Fits where Linux is too big

Zephyr OS

- 3rd Party Libraries
- Application Services
- OS Services
- Kernel
- HAL
<table>
<thead>
<tr>
<th>Operating System</th>
<th>First Commit</th>
<th>Controls</th>
<th>Declared License</th>
<th>Total Contributors</th>
<th>Contributors in last month</th>
<th>Total Commits</th>
<th>Commits in last month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zephyr</td>
<td>2014/11</td>
<td>community</td>
<td>Apache-2.0</td>
<td>1863</td>
<td>269</td>
<td>88,107</td>
<td>1,777</td>
</tr>
<tr>
<td>nuttX</td>
<td>2007/?</td>
<td>community</td>
<td>BSD-variant → Apache-2.0</td>
<td>492</td>
<td>49</td>
<td>51,863</td>
<td>248</td>
</tr>
<tr>
<td>RT-Thread</td>
<td>2009/06</td>
<td>community</td>
<td>GPL-2.0 → Apache-2.0</td>
<td>661</td>
<td>30</td>
<td>15,524</td>
<td>77</td>
</tr>
<tr>
<td>RIOT</td>
<td>2010/09</td>
<td>community</td>
<td>LGPL-2.1</td>
<td>342</td>
<td>17</td>
<td>44,624</td>
<td>137</td>
</tr>
<tr>
<td>Tizen RT</td>
<td>2015/04</td>
<td>Samsung</td>
<td>BSD-variant → Apache-2.0</td>
<td>183</td>
<td>14</td>
<td>10,796</td>
<td>41</td>
</tr>
<tr>
<td>FreeRTOS</td>
<td>2004/07</td>
<td>Richard Barry</td>
<td>GPL-2.0 w/ FreeRTOS → MIT</td>
<td>146</td>
<td>12</td>
<td>3,320</td>
<td>38</td>
</tr>
<tr>
<td>SeL4</td>
<td>2014/07</td>
<td>community</td>
<td>GPLv2 AND BSD-2-Clause</td>
<td>100</td>
<td>8</td>
<td>4,423</td>
<td>27</td>
</tr>
<tr>
<td>Contiki-NG</td>
<td>2017/10</td>
<td>community</td>
<td>BSD-3-Clause</td>
<td>217</td>
<td>6</td>
<td>17,656</td>
<td>35</td>
</tr>
<tr>
<td>myNewt</td>
<td>2015/06</td>
<td>community</td>
<td>Apache-2.0</td>
<td>134</td>
<td>4</td>
<td>10,785</td>
<td>6</td>
</tr>
<tr>
<td>ThreadX</td>
<td>2020/05</td>
<td>MSFT → community</td>
<td>MSL → MIT</td>
<td>15</td>
<td>4</td>
<td>148</td>
<td>12</td>
</tr>
<tr>
<td>mbed OS</td>
<td>2013/02</td>
<td>ARM</td>
<td>Apache-2.0 or BSD-3-Clause</td>
<td>691</td>
<td>2</td>
<td>34,558</td>
<td>2</td>
</tr>
</tbody>
</table>

Data extracted on 2023-12-01 from github
Methodology - with data extracted 2023-10-19

- Total commits: 85,565
- Total contributors: 1,800

Monthly contributors: 216
Monthly commits: 1,473 → 2 commits/hour
<table>
<thead>
<tr>
<th>Operating System</th>
<th>First Commit</th>
<th>Controls</th>
<th>declared License</th>
<th>Total Contributors</th>
<th>Contributors in last month</th>
<th>Total Commits</th>
<th>Commits in last month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zephyr</td>
<td>2014/11</td>
<td>community</td>
<td>Apache-2.0</td>
<td>1863</td>
<td>269</td>
<td>88,107</td>
<td>1,777</td>
</tr>
<tr>
<td>nuttX</td>
<td>2007/?</td>
<td>community</td>
<td>BSD-variant → Apache-2.0</td>
<td>492</td>
<td>49</td>
<td>51,863</td>
<td>248</td>
</tr>
<tr>
<td>RT-Thread</td>
<td>2009/06</td>
<td>community</td>
<td>GPL-2.0 → Apache-2.0</td>
<td>661</td>
<td>30</td>
<td>51,863</td>
<td>77</td>
</tr>
<tr>
<td>RIOT</td>
<td>2010/09</td>
<td>community</td>
<td>LGPL-2.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tizen RT</td>
<td>2015/04</td>
<td>Samsung</td>
<td>BSD-variant → Apache-2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FreeRTOS</td>
<td>2004/07</td>
<td>Richard Barry</td>
<td>GPL-2.0 w/ FreeRTOS → MIT</td>
<td>100</td>
<td>8</td>
<td>4,423</td>
<td>27</td>
</tr>
<tr>
<td>SeL4</td>
<td>2014/07</td>
<td>community</td>
<td>GPLv2 AND BSD-2-Clause</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contiki-NG</td>
<td>2017/10</td>
<td>community</td>
<td>BSD-3-Clause</td>
<td>217</td>
<td>6</td>
<td>17,656</td>
<td>35</td>
</tr>
<tr>
<td>myNewt</td>
<td>2015/06</td>
<td>community</td>
<td>Apache-2.0</td>
<td>134</td>
<td>4</td>
<td>10,785</td>
<td>6</td>
</tr>
<tr>
<td>ThreadX</td>
<td>2020/05</td>
<td>MSFT → community</td>
<td>MSL → MIT</td>
<td>15</td>
<td>4</td>
<td>148</td>
<td>12</td>
</tr>
<tr>
<td>mbed OS</td>
<td>2013/02</td>
<td>ARM</td>
<td>Apache-2.0 or BSD-3-Clause</td>
<td>691</td>
<td>2</td>
<td>34,558</td>
<td>2</td>
</tr>
</tbody>
</table>

2.45 commits/hour

Data extracted on 2023-12-01 from github
Average Number of Unique Contributors per Month

- Amazon FreeRTOS
- Apache Mynewt
- Apache NuttX
- Arm Mbed OS
- Azure RTOS ThreadX
- Contiki-NG
- FreeRTOS
- RIOT OS
- RT-Thread
- TizenRT
- Zephyr


Contributors:
- Amazon FreeRTOS: 3, 3, 30, 24, 13, 3, 1
- Apache Mynewt: 18, 18, 15, 11, 8, 5, 4
- Apache NuttX: 22, 22, 26, 36, 41, 45, 62
- Arm Mbed OS: 44, 73, 69, 41, 26, 7, 7
- Azure RTOS ThreadX: 2, 2, 2, 2
- Contiki-NG: 17, 9, 10, 8, 6, 8, 7
- FreeRTOS: 2, 2, 2, 3, 11, 8, 6
- RIOT OS: 29, 37, 42, 41, 31, 29, 20
- RT-Thread: 5, 11, 24, 28, 36, 36, 30
- TizenRT: 2, 31, 35, 35, 16, 17, 11
- Zephyr: 47, 55, 82, 104, 125, 154, 178, 212
GitHub Clones & Unique Visitors

2023-10-01 → 2023-10-14

~883 unique clones per day
~1212 unique visitors per day
Contributors Growth per Release

Contributors per release

- Total contributors to date
- Contributors to the release

Release:
- v2.6.0
- v2.7.0
- v3.0.0
- v3.1.0
- v3.2.0
- v3.3.0
- v3.4.0
- v3.5.0
New Contributors per Release

% of first-time contributors

Release

v2.6.0  v2.7.0  v3.0.0  v3.1.0  v3.2.0  v3.3.0  v3.4.0  v3.5.0
How does this compare to the Linux Kernel?
How does this compare to Linux?

6.5 Linux Kernel Statistics*

- **1,921** Contributors From 218 Organizations
- **1,945** Lines of Code Modified Daily
- **1,281** Lines of Code Removed Daily
- **9** Changes Per Hour
- **7,064** Lines of Code Added Daily

* Source: [https://lwn.net/Articles/948970/](https://lwn.net/Articles/948970/)  
Time period for 6.5: 2023/6/26-2023/8/27=63 days

Also data from: [https://github.com/philipjohnson/kernel-history/blob/master/kernel_stats.ods](https://github.com/philipjohnson/kernel-history/blob/master/kernel_stats.ods)
So what was it like when Linux started?

UNIX Source Available:
SVR4, MINIX 1.5, 4.3BSD

Commercial Distributions:
A/UX, IBM AIX, Dec Ultrix,
HP-UX, IRIX, SunOS, MIPS
RISC/os, Xenix …
What is Linux like Today?

Every market Linux has entered it eventually dominates

- 100% Supercomputer Market
- 82% Smartphone Market Share
- 2nd To Windows in Enterprise
- 90% Mainframe Customers
- 90% Public Cloud Workload
- 69% Embedded Systems Market
- #1 Internet Client
Lessons Learned by Linux Community circa 2016/2017

- Short release cycles are important.
- Process scalability requires a distributed, hierarchical development model.
- Tools matter.
- The kernel’s strongly consensus-oriented model is important.
- A related factor is the kernel’s strong “no regressions” rule.
- Corporate participation in the process is crucial.
- There should be no internal boundaries within the project

Source:

More recent stats can be found at:
++ Lessons Learned

• **Vendor-neutral environment for technical decision making**
• Mix of companies and individuals participating – “scratching their itches”
• **Streamline upstreaming process** – DCO - “signed-off-by:”
• **Public code reviews** – “reviewed-by:”
• Consensus-oriented decision model – email, in-person summits
• Hierarchical development model (**maintainer model**) – “signed-off-by”
• No internal boundaries – developer can contribute anywhere
• **Tools matter** - git enabled distributed version control - push/pull
• Short predictable release cycles and **with fixed merge windows**
• **Stable & LTS**: stable and long term support releases support product development

KEY: Developer frustration with status quo inspires creative solutions.
So what lessons did Zephyr apply from the Linux Kernel Community?
Zephyr’s Vision

The Zephyr Project strives to deliver the best-in-class RTOS for connected resource-constrained devices, built to be secure and safe.
Zephyr Developers Decide Technical Directions

- **Configuration:** kconfig & kbuild added in 2015 prior to launch
- **Unified kernel:** nano + microkernels → **unified kernel** in 2016
- **Infrastructure:** Gerrit/JIRA → **GitHub/Issues** in 2017
- **Build system:** kbuild → **cmake** in 2018
- **Other areas:**
  - APIs & HALs - reworked
  - Modularization & Device Tree support
  - Release & LTS processes refined
<table>
<thead>
<tr>
<th>Linux Best Practice</th>
<th>Zephyr Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Neutral Decision Making</td>
<td>Yes, Project support from multiple companies.</td>
</tr>
<tr>
<td>Companies and Individuals Participate</td>
<td>Yes, TSC has companies &amp; community participation.</td>
</tr>
<tr>
<td>Streamline upstreaming process</td>
<td>Yes, see /CONTRIBUTING.rst, DCO used</td>
</tr>
<tr>
<td>Public code reviews?</td>
<td>Yes, issues &amp; pull requests reviewed on <a href="https://github.com/zephyrproject-rtos/zephyr">https://github.com/zephyrproject-rtos/zephyr</a></td>
</tr>
<tr>
<td>Consensus Oriented Decision Models</td>
<td>Yes, TSC votes on features &amp; release readiness.</td>
</tr>
<tr>
<td>Hierarchical development (Maintainers)</td>
<td>Yes, see /MAINTAINERS.yml</td>
</tr>
<tr>
<td>No Internal Boundaries</td>
<td>Yes, anyone can make pull request for any area</td>
</tr>
<tr>
<td>Distributed version control</td>
<td>Yes, see /CONTRIBUTING.rst</td>
</tr>
<tr>
<td>Short Release Cycle (w/ Merge Window)</td>
<td>Yes, 10 week merge, 2-4 week stabilize</td>
</tr>
<tr>
<td>Long Term Support Releases</td>
<td>Yes, LTS 1 had 4 update release, LTS 2 active maintain</td>
</tr>
</tbody>
</table>
So what does Zephyr support today?
Supported Hardware Architectures

- ARC
- ARM Cortex-M, Cortex-R & Cortex-A
- Intel x86 & x86_64
- MIPS
- Nios II Processor
- RISC-V 32 & 64 bit
- SPARC
- Tensilica

docs.zephyrproject.org/latest/hardware/index.html#hardware-support
What about Zephyr security?
Code Repositories

- Development
  - Community Contributions via DCO
  - Releases
- Long Term Support “Stable”
  - Forward ports & Keeping Configurations in Sync
  - LTS Releases
- Auditable
  - Safety & Security Processes
  - Certifiable Releases
Long Term Support (Zephyr 2.7.x)

- **Product Focused**
- **Current with latest** Security Updates
- **Compatible with new hardware**
  - Functional support for new hardware is regularly backported
- **Tested:** Shorten the development window and extend the Beta cycle to allow for more testing and bug fixing
- **Supported for 2+ years**
- ⚠ **Doesn’t include cutting-edge functionality**

[github.com/zephyrproject-rtos/zephyr/releases/tag/zephyr-v2.7.0](https://github.com/zephyrproject-rtos/zephyr/releases/tag/zephyr-v2.7.0)
Long Term Support (LTS - 1.14)

Delivered bug fixes and latest security updates for 2 years!
Auditable

- An **auditable code base** will be established from a **subset** of the Zephyr OS LTS
- Code bases will be kept in sync
- More rigorous processes (necessary for certification) will be applied to the auditable code base.
- Processes to achieve selected certification to be:
  - Determined by Safety Committee and Security Committee
  - Coordinated with Technical Steering Committee
Project Security Documentation

- **Project Security Overview**
- Started with documents from other projects
- Built around Secure Development, Secure Design, and Security Certification
- Ongoing process, rather than something to just be accomplished
Software Supply Chain Support

- Zephyr ships an **SBOM** (Software Bill of Materials) with each release
- Downstream consumers can leverage built-in tools to, in turn, generate source & build SBOMs for their deliverables
 CVE Numbering Authority

- **Registered with MITRE in 2017**
  - We issue our own CVEs

- **Zephyr Project Security Incident Response Team (PSIRT)**
  - Volunteers from the Security Subcommittee led by the Zephyr Security Architect.

---

### Zephyr Project

The majority of the links on this page redirect to external websites; these links will open a new window or tab depending on the web browser used.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Zephyr project components, and vulnerabilities that are not in another CNA's scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>MITRE Corporation</td>
</tr>
<tr>
<td>Security Advisories</td>
<td>View Advisories</td>
</tr>
<tr>
<td>Program Role</td>
<td>CNA</td>
</tr>
<tr>
<td>Organization Type</td>
<td>Vendors and Projects</td>
</tr>
<tr>
<td>Country</td>
<td>USA</td>
</tr>
</tbody>
</table>
OpenSSF Gold Badge

- **Core Infrastructure Initiative**
  Best Practices Program

- Awards badges based on “project commitment to security”

- Mostly about project infrastructure: is project hosting, etc following security practices

- Gold status since Feb, 2019
Vulnerability Alert Registry

- For an **embargo** to be effective, product makers need to be **notified early** so they can **remediate**

- **Goal**: Zephyr to **fix issues within 30 days** to give vendors 60 days before publication of vulnerability

- Product makers can register to receive these alerts for free by signing up at Vulnerability Alert Registry
Advisory Issued by project on 20201208:

- Zephyr current release (2.4) does not use Fnet or other stacks.
- The Zephyr LTS release 1.14 contains an implementation of the TCP stack from Fnet.

Of the vulnerabilities reported in Fnet, 2, CVE-2020-17468, and CVE-2020-17469, are in the IPv6 Fnet code, one, CVE-2020-17467, affects Link-local Multicast Name Resolution LLMNR), and 2, CVE-2020-24383, and CVE-2020-17470 affect DNS functionality.

None of the affected code has been used in the Zephyr project, while 1.14 does use the Fnet TCP, it does not use the affected IPv6, DNS or LLMNR code.
Zephyr Security Summary

Documented secure coding practices
Vulnerability response criteria publicly documented

Weekly Coverity scans
MISRA scans

SBOM generation
What’s all this about Zephyr safety?
Compliant Development: V-model

It is difficult to map a stereotypical open-source development to the V-model

- Specification of features
- Comprehensive documentation
- Traceability from requirements to source code
- Number of committers and information known about them

⇒ Provide the evidences that open source developers can map to compliance and meet all requirements
Safety Collateral Proposal

- Requirement definition, Source Code & Test linkage are public; and developed in open using strictdoc
- The set of requirements (and associated traceability) are applicable to safety scope is managed by the safety committee.
- Other project artifacts have owners designated.
Initial certification focus

- Start with a limited scope of kernel and interfaces
- Initial target is IEC 61508 SIL 3 / SC 3 (IEC 61508-3, 7.4.2.12, Route 3s)
- Option for 26262 certification has been included in contract with certification authority should there be sufficient member interest

Scope can be **extended** to include **additional components** with associated **requirements** and **traceability** as determined by the safety committee.
Current requirements work

- Used tooling: StrictDoc ([https://github.com/strictdoc-project/strictdoc](https://github.com/strictdoc-project/strictdoc))
- Decision on UIDs for requirements (UID will be generated by StrictDoc)
- Hierarchical structure of requirements that works for the project
- Capturing the requirements in StrictDoc
What’s happening now..

Safety Committee

- Safety Certification Strategy decisions
  - Scope of certification
  - Certification standards
  - Certification timeline
- Assessment and audit specific tasks
- Owner of certification artefacts
- Participation limited to the project’s platinum members, the safety architect and the functional safety manager

Safety Working Group

- Enabling safety qualifications/ certifications in the project
- Working on creating the required documentation and evidences
  - Setting up requirements management tooling
  - creating/deriving and documenting requirements
- Open to everyone to participate, join today: https://lists.zephyrproject.org/g/safety-wg
Results of applying the best practices
Zephyr in the wild... 5.4K Forks!

Primary Git Repository for the Zephyr Project. Zephyr is a new generation, scalable, optimized, secure RTOS for multiple hardware architectures.

Website: https://caninosloucos.org/en/pulgacore-v2-en/
Code: https://github.com/caninos-loucos/pulga-zephyr

Source: https://github.com/zephyrproject-rtos/zephyr

The Caninos Loucos Pulga board is a powerful microcontroller with a large number of sensors, highly secure, and ideal for IoT applications. Completely designed in Brazil.

The low power board is approximately the size of a quarter dollar coin (24.26mm) and supports Bluetooth 5.0, allowing for wireless connection between multiple Pulgas.

Aspects such as modularity, via an autonomous core board and an optional base board, and energy harvesting capability, enable great extensibility for hardware interfaces prototyping.

The base board allows for adding functionality for custom applications. For example, by adding long distance wireless protocols and new sensors, you can meet the demands of different projects.

Finally, the harvesting allows for capturing energy from different environmental sources, ensuring energetic autonomy to the board.

Technological Integrated Systems Laboratory (LSI-TEC) with the support of the Polytechnic School of the University of São Paulo (Poli-USP)
Products Running Zephyr Today

- Proglow
- Ruuvi Tag
- PHYTEC Distancer
- Keeb.io BDN9
- Hati-ACE
- Oticon More
- Adhoc Smart Waste
- GNARBOX 2.0 SSD
- Anicare Reindeer Tracker
- Safety Pod
- BLiXT solid state circuit breaker
- Moto Watch 100
- Lildog & Lilcat pet tracker
- Rigado IoT Gateway
- Livestock Tracker
- Laird Connectivity sensors & gateways
- BeST pump monitoring
- Vestas Wind Turbines

zephyrproject.org/products-running-zephyr
550+ supported boards... and growing

- Arduino Portenta H7
- ESP32
- Sipeed HiFive1
- nRF9160 DK
- STM32F746G Disco
- M5StickC PLUS
- TDK RoboKit 1
- BBC micro:bit v2
- Blue Wireless Swan
- Arduino Nano 33 BLE
- Intel UP Squared
- Dragino LSN50 LoRA Sensor Node
- Microchip SAM E54 Xplained Pro Evaluation Kit
- Raspberry Pi Pico
- Altera MAX10
- NXP i.MX8MP EVK
- Adafruit Feather M0 LoRa
- u-blox EVK-NINA-B3

[docs.zephyrproject.org/latest/boards]
170+ Sensors Already Integrated

github.com/zephyrproject-rtos/zephyr/tree/main/drivers/sensor
IoT Connectivity Options

- Wide variety of **communication protocols**
  - Ethernet, 802.15.4, Thread, LoRa, Bluetooth, CAN bus, ...

- **Core network protocols** like IPv6, IPv4, UDP, TCP, ICMPv4, and ICMPv6.

- **Security** (ex. TLS, DTLS, ...)

- **Cloud integration** using MQTT, CoAP and HTTP protocols

- **Over-the-air updates**

- **Device management** using OMA LwM2M 1.1 protocol
Native IP Stack

- Built from scratch, on top of Zephyr native kernel concepts
- Dual mode **IPv4/IPv6 stack**
  - DHCP v4, IPv4 autoconf, IPv6 SLAAC, DNS, SNTP
- Multiple network interfaces support
- Time Sensitive Networking support
- **BSD Sockets**-based API
- Supports IP offloading
- **Compliance and security** tested
Bluetooth Host and Mesh

- Bluetooth 5.3 compliant
- Highly configurable
- Portable to all architectures supported by Zephyr
- Low Energy & experimental Bluetooth Classic
- IPSP/6LoWPAN for IPv6 connectivity over Bluetooth LE
- Multiple HCI transports
Bluetooth Low Energy Controller

- **Bluetooth 5.3 compliant** and qualified (5.1)
- Support for multiple BLE radio hardware architectures
  - Nordic nRF5x on Arm Cortex-M
  - VEGAboard on RISC-V
- Proprietary radios (downstream only)
- Unlimited role and connection count
- Concurrent multi-protocol support ready
- Multiple advertiser and scanner instances
Zephyr USB Device Stack

- **USB 2.0 & USB-C** support
- Supports multiple MCU families (STM32, Kinetis, nRF, SAM,...)
- Supports most common devices classes: CDC, Mass Storage, HID, Bluetooth HCI over USB, DFU, USB Audio, etc.
- Tight integration with the RTOS
- Native execution support for emulated development on Linux
- WebUSB support
Power Management

- Goal: use as little power as possible
- Cross-platform (architecture / SoC agnostic)
- Tickless scheduler
- Handled by the kernel / Customizable by the user
Devicetree

Describe & configure the available hardware on the target system
Decouple the application from the hardware

deficetree

```dts
&i2c1 {
    pinctrl-0 = <&i2c1_scl_pb8 &i2c1_sda_pb9>;
    pinctrl-names = "default";
    clock-frequency = <I2C_BITRATE_FAST>;
    status = "okay";

    lsm6dsl@6a {
        compatible = "st,lsm6dsl";
        reg = <0x06a >;
    };

    hts221@5f {
        compatible = "st,hts221";
        reg = <0x05f >;
    };

    // ...
};
```

docs.zephyrproject.org/latest/build/dts
Secure boot / Device Management

- Leverage **MCUboot** as secure bootloader
- Application binary can be signed/encrypted
  - Can use hardware keys
- But also:
  - Downgrade prevention
  - Dependency checks
  - Reset and failure recovery
- Over-the-air (OTA) upgrades
  - OMA LwM2M, Eclipse hawkBit
  - Vendor offerings
Hardware security

- Cryptography APIs
  - Random Number Generation, ciphering, etc.
  - Supported by crypto HW, or SW implementation (TinyCrypt)

- Trusted Firmware integration
  - Firmware verification/encryption
  - Device attestation
  - Management of device secrets
Building on POSIX

- **Zephyr apps can run as native Linux applications**
  - Easier to debug/profile with native tools
  - Connect to real devices using TCP/IP, Bluetooth, CAN
  - Helps minimize hardware dependencies during the development phase

- **Re-use existing code & libraries by accessing Zephyr services through POSIX API**
  - Easier for non-embedded programmers
  - Implementation is optimized for constrained systems
  - Supported POSIX subsets: PSE51, PSE52, and BSD sockets
## A real-time OS

Benchmark on Arm Cortex-M4F running at 120 MHz

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread create</td>
<td>2.5 µs</td>
</tr>
<tr>
<td>Thread start</td>
<td>3.6 µs</td>
</tr>
<tr>
<td>Thread suspend</td>
<td>3.3 µs</td>
</tr>
<tr>
<td>Thread resume</td>
<td>3.8 µs</td>
</tr>
<tr>
<td>Context switch (yield)</td>
<td>2.2 µs</td>
</tr>
<tr>
<td>Get semaphore</td>
<td>0.6 µs</td>
</tr>
<tr>
<td>Put semaphore</td>
<td>1.1 µs</td>
</tr>
</tbody>
</table>

Graphical User Interfaces

- Drivers available for various types of displays
  - LCD
  - OLED
  - Touch panel displays
  - E-ink
- LVGL integration
- Support for video capture and output
Inter-Process Communication

- **Built-in kernel services** (see table)
  - **IPC service**
    - 1-to-1 or 1-to-many communications
    - No-copy API
  - **zbus** (Zephyr Message Bus)
    - 1-to-1, 1-to-many, or many-to-many channel-based communications
    - Synchronous or asynchronous

<table>
<thead>
<tr>
<th>Object</th>
<th>Bidirectional?</th>
<th>Data structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>✗</td>
<td>Queue</td>
</tr>
<tr>
<td>LIFO</td>
<td>✗</td>
<td>Queue</td>
</tr>
<tr>
<td>Stack</td>
<td>✗</td>
<td>Array</td>
</tr>
<tr>
<td>Message queue</td>
<td>✗</td>
<td>Ring buffer</td>
</tr>
<tr>
<td>Mailbox</td>
<td>✔</td>
<td>Queue</td>
</tr>
<tr>
<td>Pipe</td>
<td>✗</td>
<td>Ring buffer</td>
</tr>
</tbody>
</table>

Data passing objects available in Zephyr kernel
Tracing & Debugging

- **Advanced logging** framework
  - Multiple backends (UART, network, file system, ...)
  - Compile-time & runtime filtering

- **Tracing** framework
  - Visualize the inner-working of the kernel and its various subsystems
  - Object tracking (mutexes, timers, etc.)
Vibrant Ecosystem

Development Tools

Governing Board

Technical Steering Committee

Contributors

Applications & Middlewares

Training & Consulting

Firmwares & Libraries

© 2023 The Zephyr Project — Content made available under CC BY-SA 4.0.
Ecosystem // Training & Consulting

Training

- Nordic Semiconductor
- Goliath
- The Linux Foundation
- Percepio

Services & Consulting

- Baylibre
- NXP
- Laird Connectivity
- Antmicro
- WNDRVR
- AVSystem
Ecosystem // Apps & Middlewares

Remote Management

- HERALD
- Goliath
- AVS

- Memfault
- STERNUM
- blues wireless

Robotics

- ROS
SBOMs at scale - see RENODE Dashboard:
3 SBOMs Included by default on each build.

https://zephyr-dashboard.renode.io/
Interested to Learn More?  Come Join Us!

Zephyr Community Overview:
  • https://www.zephyrproject.org/community/

Code on GitHub:
  • https://github.com/zephyrproject-rtos/zephyr

Mail Lists:
  • https://lists.zephyrproject.org/g/main

Discord (8000+ developers):
  • https://chat.zephyrproject.org/ (https://discord.com/invite/Ck7jw53nU2)
Zephyr Project

- **Open source** real time operating system
- **Vibrant Community** participation
- **Vendor Neutral** governance
- **Permissively** licensed - Apache 2.0
- **Cross-architecture** with broad SoC and development board support.
- **Complete**, fully integrated, highly configurable, **modular** for flexibility
- **Product** development ready using LTS includes security updates
- Built with **safety and security** in mind
Questions?

www.zephyrproject.org
EMBEDDED IOT SUMMIT

@ OPEN SOURCE SUMMIT JAPAN