Zephyr LTS Release
What to expect and why are we doing this

Anas Nashif, Intel Corp.
Zephyr: a modular RTOS and a complete solution stack

Challenge
Many companies paying for and heavily investing in different real-time OS solutions, for small connected devices and embedded controllers. This lead to costly, time consuming and divergent solutions across the industry.

Solution, Zephyr
- A small, modular, open source, real-time operating system (RTOS) for use on connected resource-constrained and embedded controllers
- Supports diverse use cases and architectures
- Focused on safety, security, connections with Bluetooth support, and a full native networking stack
- Apache 2.0 license, hosted at Linux Foundation

Ecosystem Support

Stack
- Zephyr OS
- 3rd Party Libraries
- Application Services
- Middleware/Networking
- OS Services
- Kernel
- HAL
Zephyr – A fully featured RTOS

Zephyr is a small, modular, open-source real-time operating system (RTOS) for use on resource-constrained systems covering diverse use cases and supporting multiple architectures.

Configurable & Modular
- Zephyr Kernel can be configured to run in as little as 8k RAM
- Enables application code to scale
- Configurable and Modular

Cross Platform
- Support for multiple architectures
- Native Port
- Developed on Linux, Windows and MacOS

Open Source
- Licensed under Apache II License
- Managed by the Linux Foundation*
- Transparent development
- Fork it on Github!

Focus on Safety & Security
- Thread Isolation
- User-space Support
- Stack Protection (HW/SW)
- Crypto Support
- DFU (IP+BLE)

Connected
- Full Bluetooth 5.0 Support
- Bluetooth Controller
- BLE Mesh
- Thread Support
- Full featured native networking stack

Zephyr is not an ingredient, Zephyr provides a complete solution.
Why Zephyr?

The Zephyr OS addresses broad set of embedded use cases across a broad set of platforms and architectures using a modular and configurable infrastructure.

**Address Fragmentation**
- No single RTOS addresses broad set of embedded use cases across a broad set of platforms and architectures
- Disjoint use cases have led to fragmentation in RTOS space
- Existing commercial solutions force "roll your own" solutions and duplication of software components

**Modular Infrastructure**
- Modular and configurable infrastructure allows creation of highly compact and optimal solutions for different products from a common origin
- Reuse allows NRE costs to be amortized across multiple products and solutions
- Multi-architecture support reduces platform switching costs and vendor lock-in concerns

**Open-Source**
- Roll your own is expensive & difficult to develop & maintain
- Permissively licensed corresponds to ease of adoption
- Corporate sponsorship assures long term commitment and longevity
- Community innovation has proven faster for progression and project development is a collaboration of industry experts

**Feature Richness**
- Need for a solution or semi-complete solution rather than just an ingredient.
- Lowers entry level barrier for new products and speeds up software delivery using existing feature and hardware support
- Encourages adherence to standards and promotes collaboration on complex features inside the organization
- Developers focus on the end-user facing interfaces instead of re-inventing low level interfaces

Reduce costs and improve efficiency through reuse
Project Members

Platinum Members

Silver Members

[Logos of Platinum and Silver Members]
# Steady Growth

<table>
<thead>
<tr>
<th>Release</th>
<th>Architectures</th>
<th>Boards</th>
<th>Drivers</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>3</td>
<td>15</td>
<td>50</td>
<td>39</td>
</tr>
<tr>
<td>1.1</td>
<td>3</td>
<td>14</td>
<td>51</td>
<td>39</td>
</tr>
<tr>
<td>1.2</td>
<td>3</td>
<td>16</td>
<td>89</td>
<td>43</td>
</tr>
<tr>
<td>1.3</td>
<td>3</td>
<td>17</td>
<td>96</td>
<td>40</td>
</tr>
<tr>
<td>1.4</td>
<td>4</td>
<td>20</td>
<td>116</td>
<td>50</td>
</tr>
<tr>
<td>1.5</td>
<td>4</td>
<td>22</td>
<td>129</td>
<td>64</td>
</tr>
<tr>
<td>1.6</td>
<td>4</td>
<td>30</td>
<td>145</td>
<td>87</td>
</tr>
<tr>
<td>1.7</td>
<td>6</td>
<td>52</td>
<td>182</td>
<td>95</td>
</tr>
<tr>
<td>1.8</td>
<td>6</td>
<td>60</td>
<td>217</td>
<td>103</td>
</tr>
<tr>
<td>1.9</td>
<td>6</td>
<td>70</td>
<td>227</td>
<td>85</td>
</tr>
<tr>
<td>1.10</td>
<td>6</td>
<td>82</td>
<td>223</td>
<td>90</td>
</tr>
<tr>
<td>1.11</td>
<td>7</td>
<td>90</td>
<td>247</td>
<td>95</td>
</tr>
</tbody>
</table>

![Graph showing steady growth trends for Architectures, Boards, Drivers, and Contributors](image-url)
Architecture and Key Features

- Highly Configurable, Highly Modular
- Cooperative and Pre-emptive Threading
- Memory and Resources are typically statically allocated
- Integrated device driver interface
- Memory Protection: Stack overflow protection, Kernel object and device driver permission tracking, Thread isolation
- Bluetooth® Low Energy (BLE 4.2, 5.0) with both controller and host, BLE Mesh
- Native, Fully featured and optimized networking stack

Fully featured OS allows developers to focus on the application
Memory Protection (MMU/MPU Support)

Now

- Same kernel & driver APIs for kernel and user mode threads
- System calls for privilege elevation
- Stack overflow protection
- Kernel object and device driver permission tracking
- Simple app vs. kernel memory separation
- Memory domain APIs for fine-tuning memory region permissions
- Stack memory protection from other threads
- Stack Overflow protection
- Initial alpha-quality thread-level memory protection on x86, user-space, memory domains – Zephyr 1.10

Future

- Refined thread-level memory protection, test cases, documentation – Zephyr 1.12 LTS
- Demand paging – TBD (1.12+)
- Dynamically loadable applications / separate app & kernel build – TBD (1.12+)

See dedicated talk by Andrew Boie @ 2pm
Connectivity: Feature-rich Native networking stack

Now
- Dual mode native IP stack
- UDP/TCP
- Elements of RPL, 6LoWPAN
- Protocols: HTTP, CoAP, MQTT, DNS, DHCPv4, LWM2M, DTLS, TLS
- 802.15.4 Restricted Functionality Device support
- BSD Sockets support
- Network Management API
- Minimal Copy Network Buffer Management
- Link Layers: IEEE 802.15.4, Bluetooth, Ethernet, SLIP (IP over serial line).
- OpenThread

Future
- Thread enablers (MLE, IP-IP tunneling, 6LoWPAN and IP routing etc.)
- Optimization and Profiling
- Testing: IPv6 ready logo, TAHI full coverage
- 15.4 IOP testing
- WiFi offload support
- 4G modem offload support
- Native OIC support
- Automated TTCN-3-based testing
- Tentative: 802.15.4e and TSCH

See dedicated talk by Ravi Kumar Veeramally @ 10:50 am
Connectivity: Bluetooth Host and Controller

**Now**
- Bluetooth 5.0 compliant (ESR10)
- BLE Mesh
- Bluetooth Low Energy Controller support (LE Link Layer)
- Generic Access Profile (GAP) with all possible LE roles
- GATT (Generic Attribute Profile)
- Pairing support, including the Secure Connections feature from Bluetooth 4.2
- IPSP/6LoWPAN for IPv6 connectivity over Bluetooth LE
- Basic Bluetooth BR/EDR (Classic) support
- Clean HCI driver abstraction
- Raw HCI interface to run Zephyr as a Controller instead of a full Host stack
- Verified with multiple popular controllers

**Future**
- Vendor HCI commands & events
  - Mesh optimizations
  - Various things that standard HCI doesn't provide, like default static address reading
- Completion of basic BR/EDR protocols (RFCOMM, L2CAP & SDP)

See dedicated talk by Johan Hedberg@ 04:10 pm
Device Firmware Update

MCUboot bootloader
- Cross-RTOS open source bootloader

MCUmgr DFU framework
- Cross-RTOS open source DFU

2x Nordic Semiconductor nRF52
- Zephyr OS BLE Controller
- Zephyr OS Application, full BLE stack

Linux
- BlueZ Bluetooth Host stack
- mcumgr management tool

Diagram:
- BlueZ + MCUmgr
- HCI
- Controller
- BlueZ
- MCUboot
- Application
- MCUmgr
- Full BLE stack
OpenThread and Zephyr

OpenThread alongside Zephyr

- Integrated at L2 (DLL) level
- Integrated at 15.4 radio driver level

Seamless for developers

- Use Zephyr networking APIs
- Benefit from Zephyr infrastructure

Nordic nRF52840

- Zephyr OS 15.4 driver and IP stack
- OpenThread 6loPAN, Thread stack
Zephyr Roadmap 2018

- Expand use cases and application areas
  - Industrial, safety and security features
  - Deep Embedded usages ✔
  - Advanced Configurations and use cases: SMP, AMP, .. ✔ (1.12)

- Eco System
  - Improve support on Mac* and Windows* ✔ (1.11)
  - IDE integration
  - 3rd Party Tools: Tracing, Profiling, Debugging...
  - LLVM, Commercial compilers, .. ✔ (1.12)

- Introduce and support Zephyr as an E2E platform:
  - Bootloader ✔ (1.11)
  - Device Firmware Updates ✔ (1.11)
  - Cloud Connectivity ✔
  - Development Tools

- Safety and Security
  - Development model and process with security and safety in mind ✔
  - Secure and harden the Kernel ✔ (1.12)
  - MISRA-C 2012 Compliance
  - Standard APIs and Portability: POSIX Layer (PSE54), BSD Socket ✔ (1.12)
What is Zephyr LTS

It is

- **Product Focused**
- **Compatible with New Hardware**: We will make point releases throughout the development cycle to provide functional support for new hardware.
- **More Tested**: Shorten the development window and extend the Beta cycle to allow for more testing and bug fixing
- **Certifiable**: The base for the auditable branch

It is not

- **A Feature-Based Release**: focus on hardening functionality of existing features, versus introducing new ones.
- **Cutting Edge**
- An auditable code base will be established from a subset of Zephyr OS features.
- Both code bases will be kept in sync from that point forward, but more rigorous processes (necessary for certification) will be applied before new features move into the auditable code base.
- Initial and subsequent certification targets to be decided by Zephyr project governing board.
- Processes to achieve selected certification to be determined by Security Working Group and coordinated with the TSC.
Roadmap to FuSA, Security Pre-Cert.

Limit the Scope
- Limit to officially supported and maintained code
- Start of the lowest layers and go up the stack

Compliance with coding and style guidelines, development process
- MISRA-C Compliance (MISRA-C:2012)

Portability
- Support POSIX APIs (PSE52, long term PSE54)

Well defined and Stable APIs

Robustness and operational safety
- MMU and MPU support
- Thread Isolation
- Stack Protection

Enhance and Increase Test Coverage
Zephyr is a one-tree project hosted @ github.com in one GIT repository

Loosely following the Linux kernel development model and using Linux configuration system (Kconfig)

All project features and HW support are part of Zephyr

Modularity is achieved through inclusion/exclusion of features and HW support using configuration options (Kconfig)

Zephyr does not provide modularization on the source code level.

- Limited support for external, remotely hosted modules
- Limited support for out-of-tree boards and HALs (boards, SoCs, drivers, …)

Zephyr includes externally maintained code such as

- Vendor HALs
- Libraries (Crypto, Debug, …)
LTS: Modularity on the source level

- On Demand
  - Users should be able to only “download” and “see” the modules they need for their use case
- Don’t care about XYZ HAL
  - If a user is building for a certain architecture, they should not have to deal with HALs of all other architectures
- Kernel Only mode: Modularity down to the kernel and the barebones.
  - A user should be able to only use the kernel and its basic features excluding everything else in Zephyr.
Candidate Standards for Auditable Code

- Coding for Safety, Security, Portability and Reliability in Embedded Systems:
  - MISRA C:2012, with Amendment 1, following MISRA C Compliance:2016 guidance

- Safety:
  - IEC61508: 2010 (SIL 3, but possibly SIL 4)
    - broadest for robotics and autonomous vehicle engineering companies. Reference for other standards in Robotics domain.
    - Sampled Certifications derived from IEC61508: Medical: IEC 62304; Auto: ISO 26262; Railway: EN 50128

- Aerospace (FAA): DO-178C (DAL A is the highest certification level)
  - Sampled certification derived from DO-178C: Radio: DO-330 through 333; Security: DO-254

- Security:
  - Common Criteria (EAL 4 but possibly higher levels EAL 5,6)

- Others
  - Medical: FDA 510(K), ISO 14971, IEC 60601; Industrial: UL 1998, ??
Quality Matters

- Quality is a mandatory expectation for software across the industry.
- Assumptions:
  - Software Quality is enforced across Zephyr project members
  - Compliance to internal quality processes is expected.
- Software Quality is not an additional requirement caused by functional safety standards.
- Functional safety considers Quality as an existing pre-condition.
Zephyr LTS Process

- Identify the MUST HAVE features of an LTS
- Based on the above, determine timeline and release date (**Proposal: 1.12**)
- Establish process around LTS
- LTS to be positioned as the long-term target and the recommended Zephyr release for products and certification
- Quarterly releases are short-term, releasable milestones leading to an LTS
  - AKA Engineering, Preview Releases
- Provide updates (bug fixes and LTS to LTS+1)
- Lifetime: 2 years initially
LTS Features and Requirements

- MMU/MPU Support for memory protection, thread isolation
  - On ARC, ARM, X86
- PSE52: Embedded POSIX Profile with File System Support
- BSD Socket APIs
- First class support on Linux, Mac and Windows
- OS Awareness and debugging Support
  - Hooks for tracing kernel objects
  - Memory and Power profiling hooks
LTS Features and Requirements

- Bluetooth 5.0
  - Vendor and mesh extensions - must have
- BLE Mesh Networking
- FOTA Updates / DFU
  - BLE
  - IP networks (LWM2M)
- LWM2M
- Bootloader Support (MCUBoot)
LTS Features and Requirements

- Isolation of kernel and lower part of the stack
  - for audit and certification purposes
  - Perform a review of the kernel code (mini HAZOP)
- Support in 3rd Party debugging tools
  - OpenOCD
  - Systemview
- Enhanced System Logging
  - Support different back-ends
- File System support (beyond FAT)
  - Support multiple filesystems, VFS
LTS Prerequisites

- Networking APIs and consistent API usage in protocols (HTTP, ...)
- Stable driver APIs - must have, needs scoping for 1.12
  - API Versioning and stable vs “experimental”
  - API coverage testing
- Support out of tree SoCs and Boards
- Cleanup public headers
- Real world samples
Nice to Have LTS Features

- Bluetooth 5.0
  - Advertising extensions
- Abstracted Crypto APIs
- WIFI Offloading – Socket Level
  - On socket level or low level APIs! (Need more discussion and a decision for LTS)
- AT command library for LTE based devices (offloading)
- Stack monitoring
  - Monitoring/Statistics “service” Threads
- File System support (beyond FAT)
  - NFFS to support multiple flash hw / multiple flash alignments
- Support in 3rd Party debugging tools
  - Trace Analyzer - WIP / Discussions with percepio
- IDE Integration (Eclipse) - nice to have
- OpenAMP (nice to have)
  - Zephyr to Zephyr
  - Linux to Zephyr
- Multi core + bootloader builds: nice to have
Functional Safety and Security requirements need to coexist with the open-source nature of the project

Quality needs to be driven on the project level
- Need to showcase our quality process and test plans publicly
- Drive adoption through quality managed release process

Manage Developer and Contributor Expectations
- MISRA-C is considered evil and not flexible enough for “open-source” projects.

Continue innovating on main tree while hardening and stabilizing Zephyr LTS
### Get Started

<table>
<thead>
<tr>
<th>Resource</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Website</td>
<td><a href="http://www.zephyrproject.org/">http://www.zephyrproject.org/</a></td>
</tr>
<tr>
<td>Documentation</td>
<td><a href="http://docs.zephyrproject.org/">http://docs.zephyrproject.org/</a></td>
</tr>
<tr>
<td>Git Repository (Code)</td>
<td><a href="https://github.com/zephyrproject-rtos/zephyr">https://github.com/zephyrproject-rtos/zephyr</a></td>
</tr>
<tr>
<td>Issues</td>
<td><a href="https://github.com/zephyrproject-rtos/zephyr/issues">https://github.com/zephyrproject-rtos/zephyr/issues</a></td>
</tr>
<tr>
<td>Mailing lists</td>
<td><a href="https://lists.zephyrproject.org/mailman/listinfo">https://lists.zephyrproject.org/mailman/listinfo</a></td>
</tr>
</tbody>
</table>