

# Zephyr RTOS

Linux kernel little brother

[vasili\\_slapik@epam.com](mailto:vasili_slapik@epam.com)

# Real-time OS vs General Purpose OS:

- “deterministic” timing behavior in the real-time operating systems; key factors are minimal interrupt latency and minimal thread switching latency; the “ready” task with highest priority runs immediately and monopolize the CPU
- “fair” slicing of CPU time in general purpose OS; no particular thread can monopolize CPU all the time, no matter what its priority.

# Hardware specifics

Typical for RTOSs:

- Memory constrained (usually up to megabytes, quite often tens of kilobytes)
- Single-core CPU (multi-core is possible but not common)
- MPU

Not-typical for RTOSs:

- Out of order execution
- Speculative execution
- Superscalar architecture
- MMU (virtual memory, CPU caches, TLB, ...)

# List of existing RTOSs

~200 RTOSs mentioned on Wikipedia comparison page, many more are not covered but still in use.

RTOSs differ by:

- License (free or proprietary, mixed, ...)
- Supported architectures
- Supported peripherals (from just pure kernels to full-fledged BSP)
- Development status (active, dead, dormant)

# Zephyr RTOS

- Originally developed as Rocket kernel by Wind River Systems for Internet of things (IoT) devices
- Initial release: 1.0.0 (February 2016)
- Latest release: 1.14.0 (April 2019)
- Work continues on release 2.0.0
- Still in its early stage (bugs, no-buildable configurations, lack of support for platform drivers on many boards)

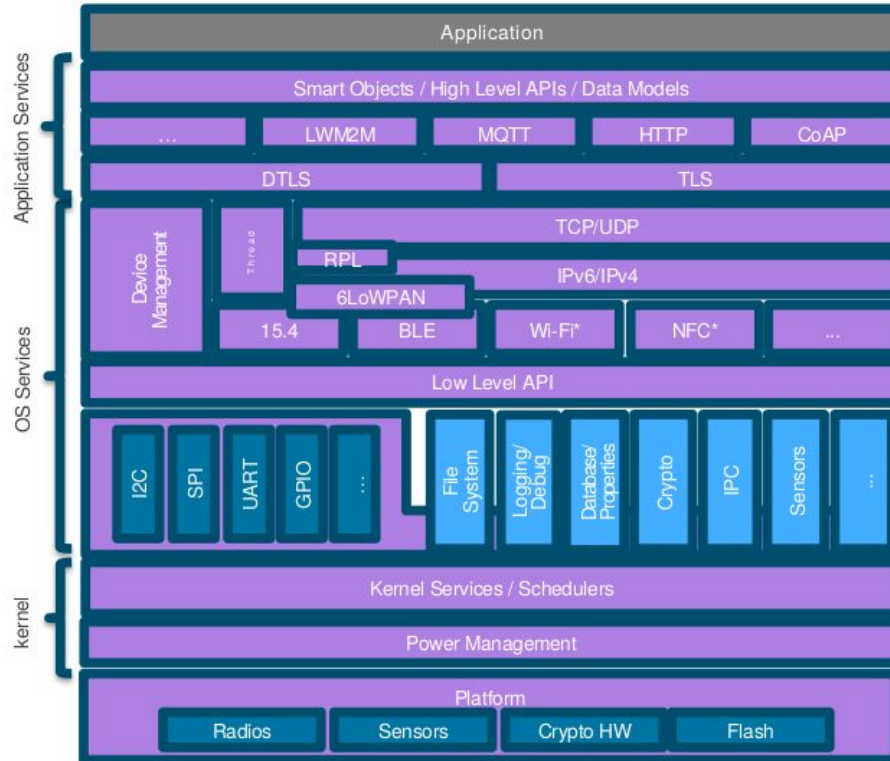
# Zephyr RTOS

- Apache 2.0 (you can use Zephyr in commercial product and you don't have to provide your sources or changes made to the kernel)
- Open-source model, a project of Linux Foundation
- Driven by Technical Steering Committee (TSC), members from Intel, Linaro, Nordic, ST Micro, Synopsis, NXP, Texas instruments, etc.
- Extremely active development, contributors from huge silicon makers companies (as above) and from enthusiasts all over the world

# Zephyr RTOS

- Support ARC, ARM (Cortex M0/M3/M4/M23/M33, Cortex R4/R5), Nios II, RISC-V, Tensilica, x86 (both 32 and 64 bits) and RISC-V
- Over 150 board configurations are supported
- Cooperative and preemptive threading
- Memory and resources are typically statically allocated
- Integrated device driver interface
- Stack overflow protection, kernel object and device driver permission tracking, thread isolation
- Native, fully featured and optimized networking stack

# Zephyr RTOS architecture





# Peripherals API

- ADC
- CAN
- Counter
- DMA
- Entropy
- Flash
- GPIO
- I2C EEPROM Slave
- I2C
- I2S
- Pinmux
- PWM
- Sensors
- SPI
- UART
- Watchdog

# Kernel Services API

- Threads
- Scheduling
- Interrupts
- Semaphores
- Mutexes
- FIFOs
- Stacks
- Message queues
- Pipes
- Timers
- Memory slabs
- Memory pools
- Polling API
- Ring buffers

# Networking support

- core IP stack (IPv4, IPv6, UDP, TCP, ICMPv4 and ICMPv6) implementation
- BSD Sockets compatible API
- DNS resolver API
- DHCP client API
- simple NTP client library
- PPP support
- CoAP, LwM2M, MQTT

# Miscellaneous API

- File system abstraction (VFS)
- Display interface
- Logging
- Shell
- USB device stack
- Bluetooth stack (including BLE and Mesh)

# Configuration

Zephyr uses Kconfig and Device tree (DT) as its configuration systems, inherited from the Linux kernel. HW configuration is stored in DT (cpu, address space, peripheral registers, pinout, ..), OS software configuration is stored in .config file generated by menuconfig utility. Unlike Linux kernel DT files are used only at compile time.



# Zephyr configuration example

```
(Top)
Zephyr Kernel Configuration
Modules --->
Board Selection (STM32 Minimum Development Board (Blue)) --->
Board Options ----
SoC/CPU/Configuration Selection (STM32F1x Series MCU) --->
Hardware Configuration --->
ARM Options --->
Architecture (ARM architecture) --->
General Architecture Options --->
General Kernel Options --->
Device Drivers --->
C Library --->
Additional libraries --->
[ ] Bluetooth ----
[ ] Console subsystem/support routines [EXPERIMENTAL] ----
[ ] C++ support for the application ----
System Monitoring Options --->
Debugging Options --->
[ ] Disk Interface ----
File Systems --->
[ ] Logging ----
Management --->
Networking --->
[*] Shell --->
[ ] DFU image manager ----
[ ] Non-random number generator
Storage ----
[ ] Enable settings subsystem with non-volatile storage ----
Testing --->
[ ] Character framebuffer for dot matrix displays ----
[ ] Enable JSON Web Token generation ----
External Sources --->
Build and Link Features --->
Boot Options --->
Compatibility --->

[Space/Enter] Toggle/enter [ESC] Leave menu [S] Save
[O] Load [?] Symbol info [/] Jump to symbol
[F] Toggle show-help mode [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

# Zephyr configuration example

```
(Top) - Device Drivers
Zephyr Kernel Configuration
[ ] IEEE 802.15.4 drivers options ----
(UART_1) Device Name of UART Device for UART Console
[*] Console drivers --->
[ ] SLIP driver ----
[ ] Net loopback driver ----
-* Serial Drivers --->
  Interrupt Controllers --->
  Timer Drivers --->
[ ] Entropy Drivers ----
[*] GPIO Drivers --->
[ ] Shared interrupt driver ----
[*] SPI hardware bus support --->
[*] I2C Drivers --->
[ ] I2S bus drivers ----
[ ] PWM (Pulse Width Modulation) Drivers ----
[*] Enable board pinmux driver --->
[ ] ADC drivers ----
[ ] Real-Time Clock ----
[ ] Watchdog Support ----
[*] Hardware clock controller support --->
[ ] Precision Time Protocol Clock driver support ----
[ ] IPM drivers ----
[ ] Flash hardware support ----
[*] Sensor Drivers --->
[ ] Counter Drivers ----
[ ] DMA driver Configuration ----
[ ] USB ----
[ ] Crypto Drivers [EXPERIMENTAL] ----
[ ] Display Drivers ----
[ ] LED strip drivers ----
[ ] add support for Wi-Fi Drivers ----
[ ] LED drivers ----
[ ] CAN Drivers ----
[ ] Modem Drivers ----
-----
[Space/Enter] Toggle/enter [ESC] Leave menu [S] Save
[O] Load [?] Symbol info [/] Jump to symbol
[F] Toggle show-help mode [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```



# Zephyr configuration example

```
(Top) - General Kernel Options
Zephyr Kernel Configuration
[*] Multi-threading
(16) Number of coop priorities
(15) Number of preemptible priorities
(0) Priority of initialization/main thread
(0) Priority inheritance ceiling
(0) Number of very-high priority 'preemptor' threads
[ ] Enable earliest-deadline-first scheduling
[ ] Enable CPU mask affinity/pinning API
(1024) Size of stack for initialization and main thread
(256) Size of stack for idle thread
(2048) ISR and initialization stack size (in bytes)
[ ] Thread stack info
[ ] Thread custom data
[*] Enable errno support
Scheduler priority queue algorithm (Simple linked-list ready queue) --->
Wait queue priority algorithm (Simple linked-list wait_q) --->
Kernel Debugging and Metrics --->
Work Queue Options --->
Atomic Operations ----
Timer API Options --->
Other Kernel Object Options --->
(10000) System tick frequency (in ticks/second)
(72000000) System clock's h/w timer frequency
[*] System clock exists and is enabled
[*] Execute in place
Initialization Priorities --->
Security Options --->
SMP Options --->
[*] Tickless idle
(3) Tickless idle threshold
[*] Tickless kernel
[ ] System Power management ----
[ ] Device power management

[Space/Enter] Toggle/enter [ESC] Leave menu [S] Save
[O] Load [?] Symbol info [/] Jump to symbol
[F] Toggle show-help mode [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

# Build, flash and debug

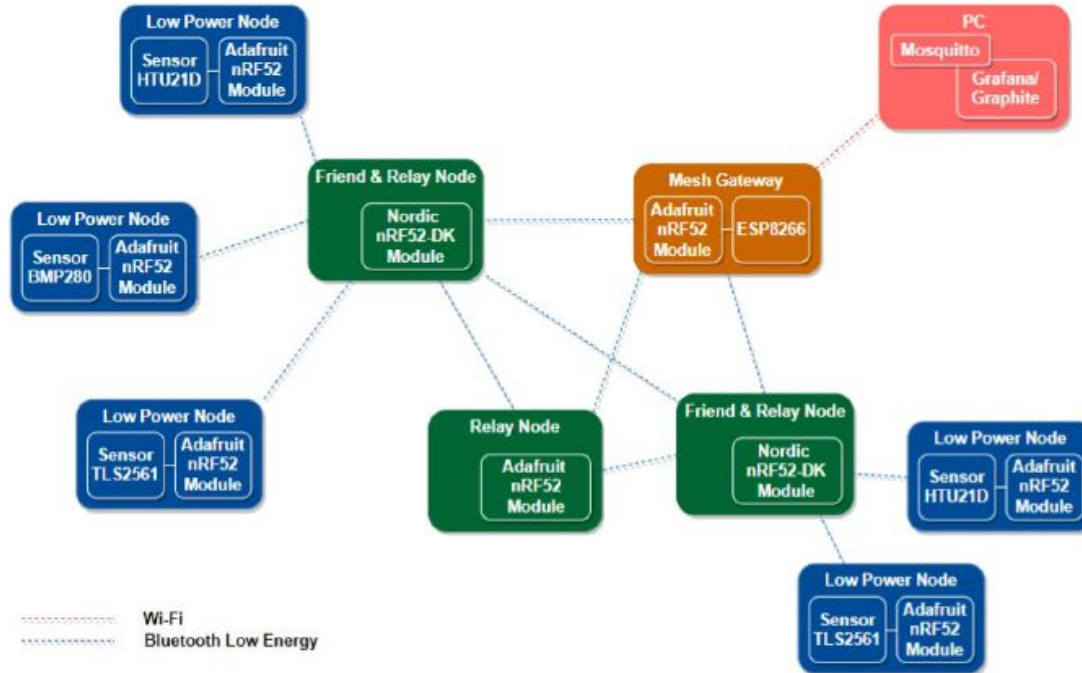
- a separate repository with up-to-date toolchains for all supported platforms
- **cmake** is used for generating either Makefile or Ninja build files
- all application components are compiled and linked into a single application image
- *flash* target used for uploading firmware image to the MCU
- *debug* target starts interactive CLI debug session

Flashing and debugging on many platforms implemented with OpenOCD support.

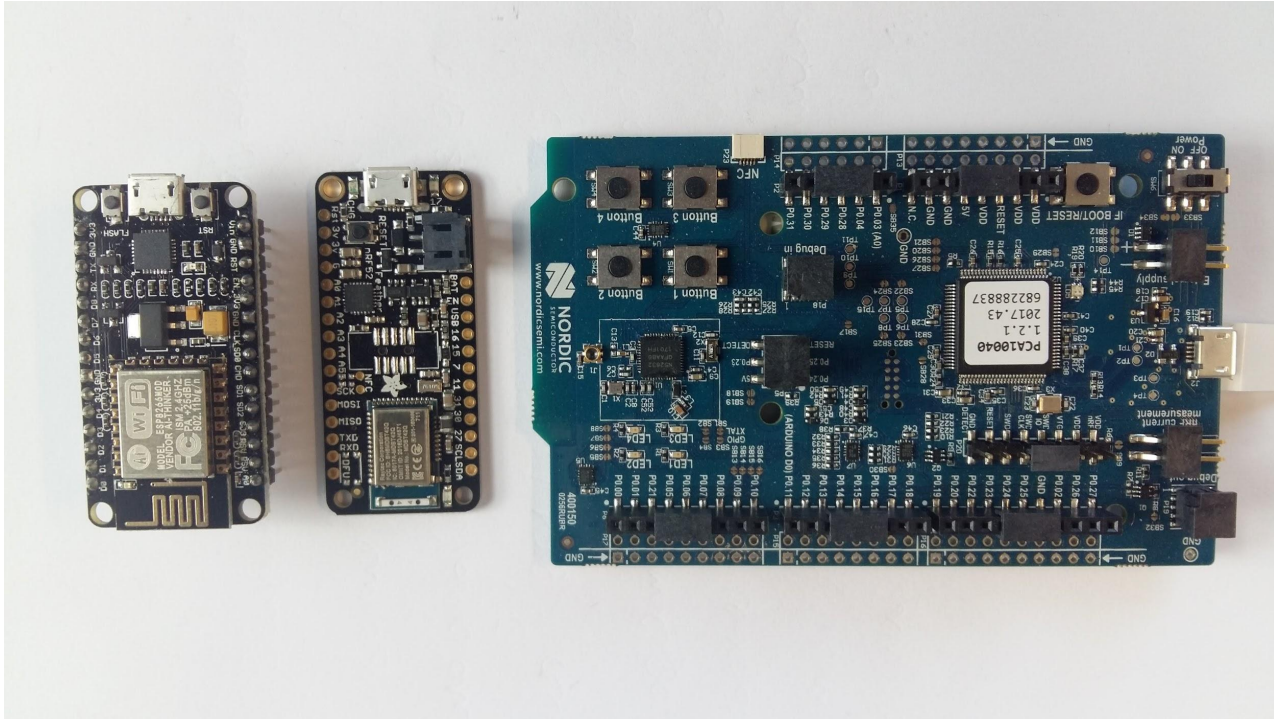
# Zephyr RTOS future plans

- integration with Trusted Execution Environment NFC support
- MIPS architecture support
- dynamic module loading
- toolchain abstraction

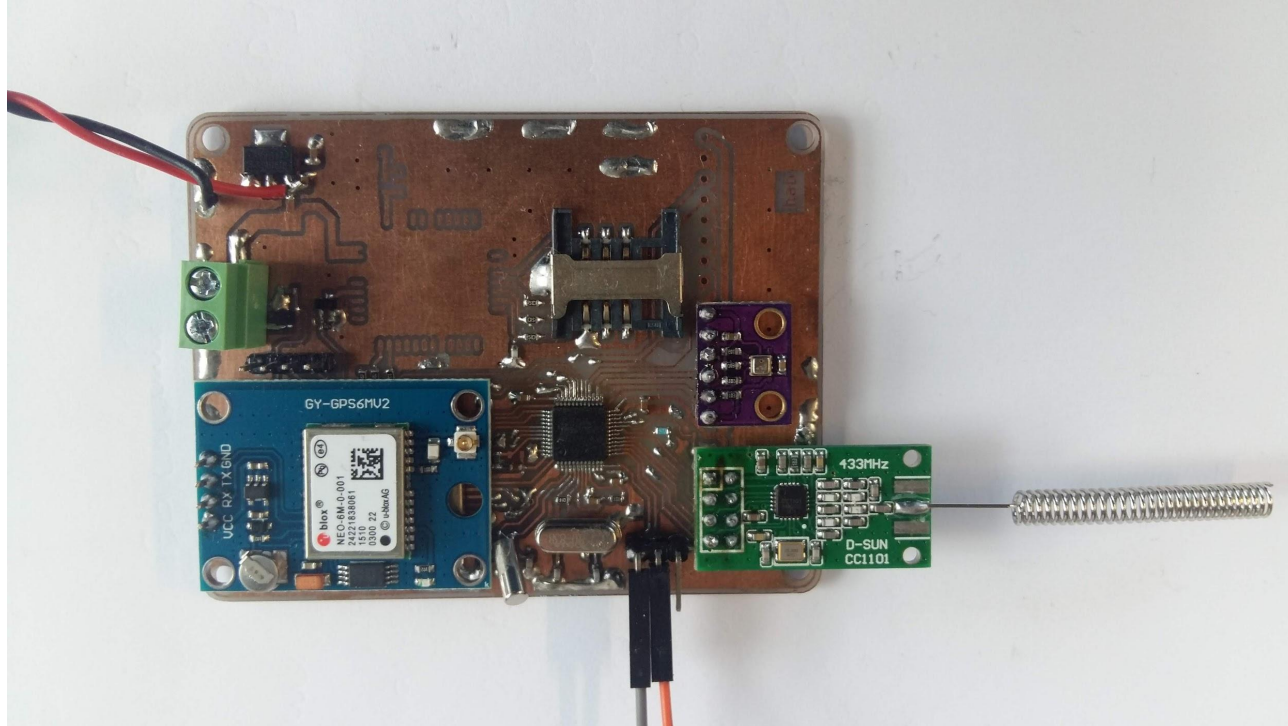
# EPAM experience with Zephyr RTOS (BLE mesh)



# EPAM experience with Zephyr RTOS (BLE mesh)



# EPAM experience with Zephyr RTOS



Q & A