Zephyr RTOS

Linux kernel little brother

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)

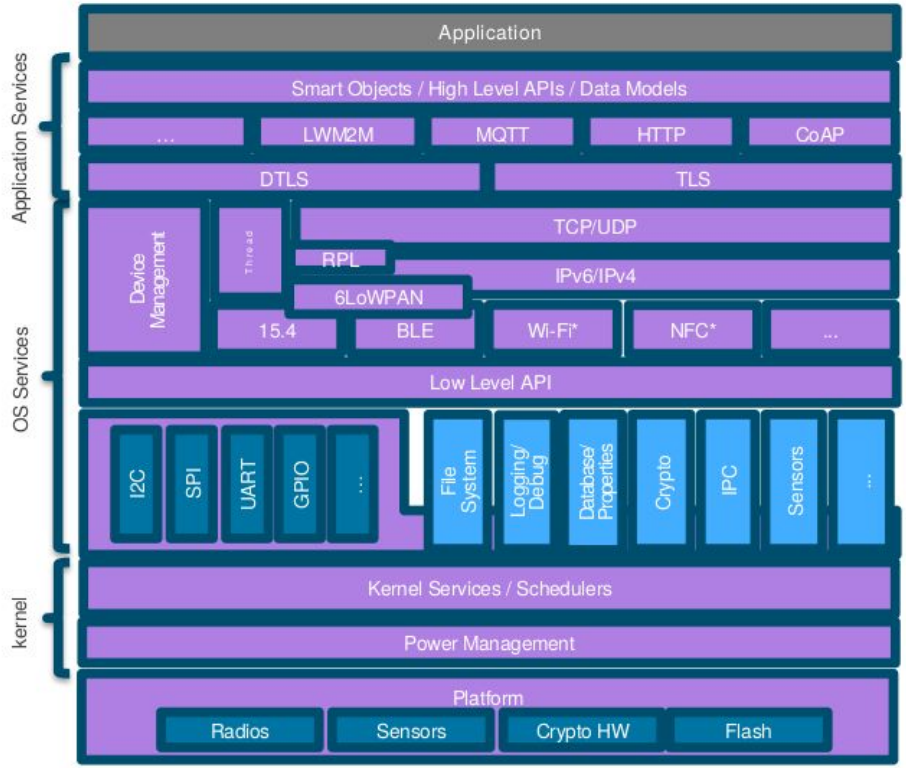
Zephyr RTOS

- 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

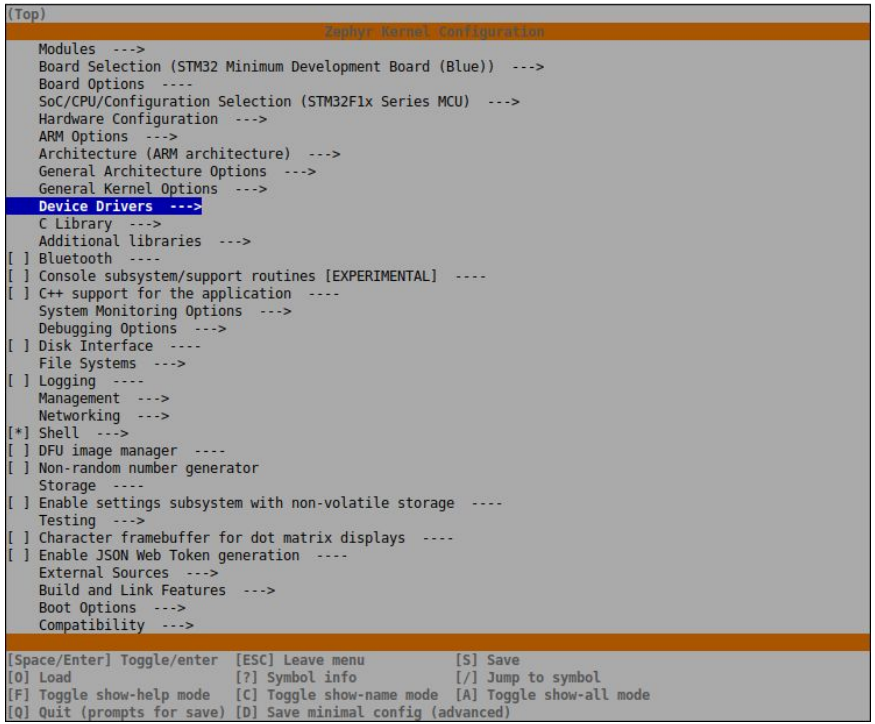
Zephyr uses Kconfig and Device tree (DT) as its configuration systems, inherited from the Linux kernel. HW configuration is stored in DT (cpu, address space, peripheral registers, pinout, ..), OS software configuration is stored in .config file generated by menuconfig utility. Unlike Linux kernel DT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

DT example

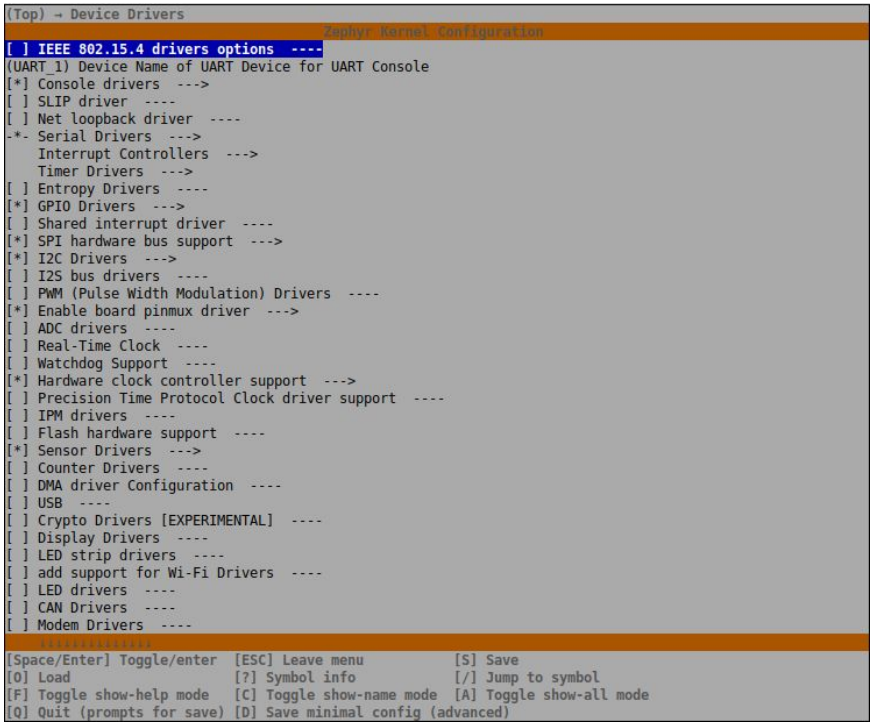
```
#include <st/f1/stm32f103Xb.dtsi>

/ {
        flash0: flash@8000000 {
                reg = <0x08000000 DT_SIZE_K(512)>;
        };

        sram0: memory@20000000 {
                reg = <0x20000000 DT_SIZE_K(64)>;
        };
...
```

```
uart4: serial@40004c00 {
                compatible = "st,stm32-uart";
                reg = <0x40004c00 0x400>;
                clocks = <&rcc STM32_CLOCK_BUS_APB1 0x00080000>;
                interrupts = <52 0>;
                status = "disabled";
                label = "UART_4";
        };
...
```

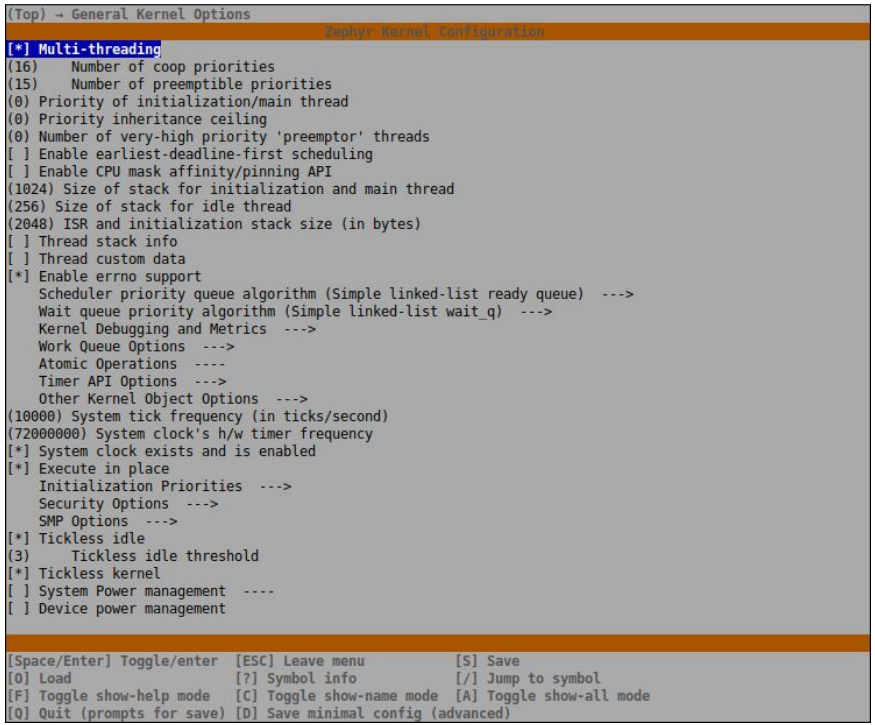
Zephyr configuration example



Zephyr configuration example



Zephyr configuration example



Build, flash and debug

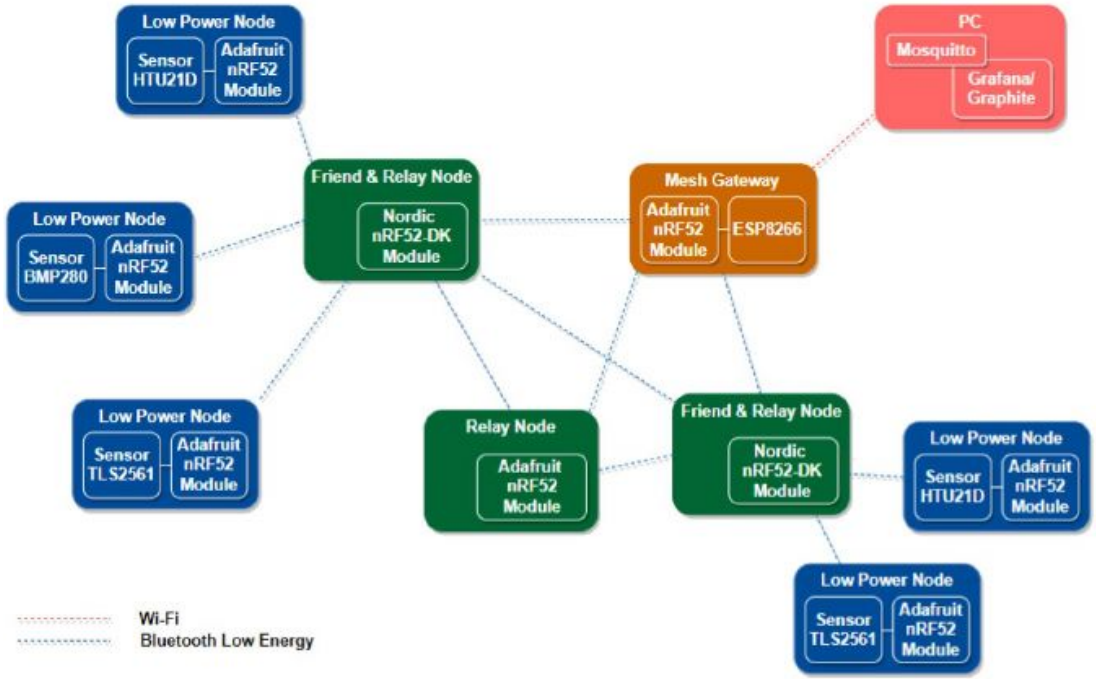
- a separate repository with up-to-date toolchains for all supported platforms
- **cmake** is used for generating either Makefile or Ninja build files
- all application components are compiled and linked into a single application image
- ***flash*** target used for uploading firmware image to the MCU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

Flashing and debugging on many platforms implemented with OpenOCD support.

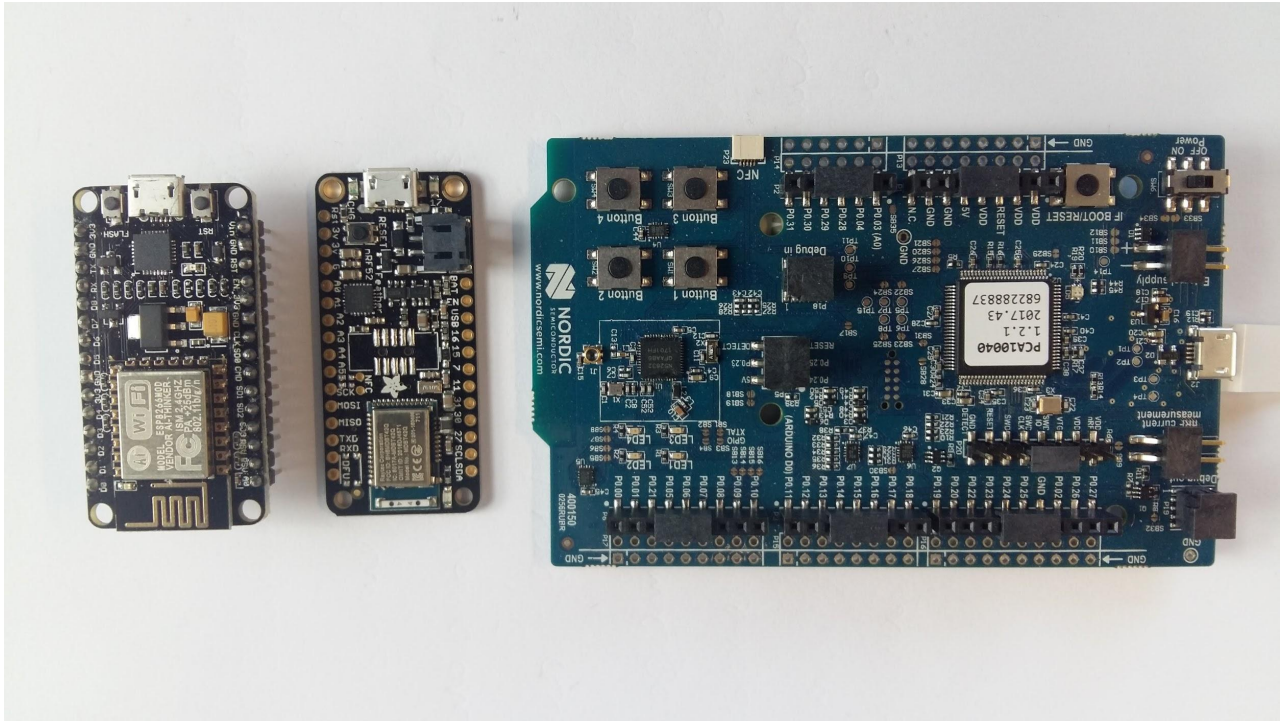
Zephyr RTOS future plans

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

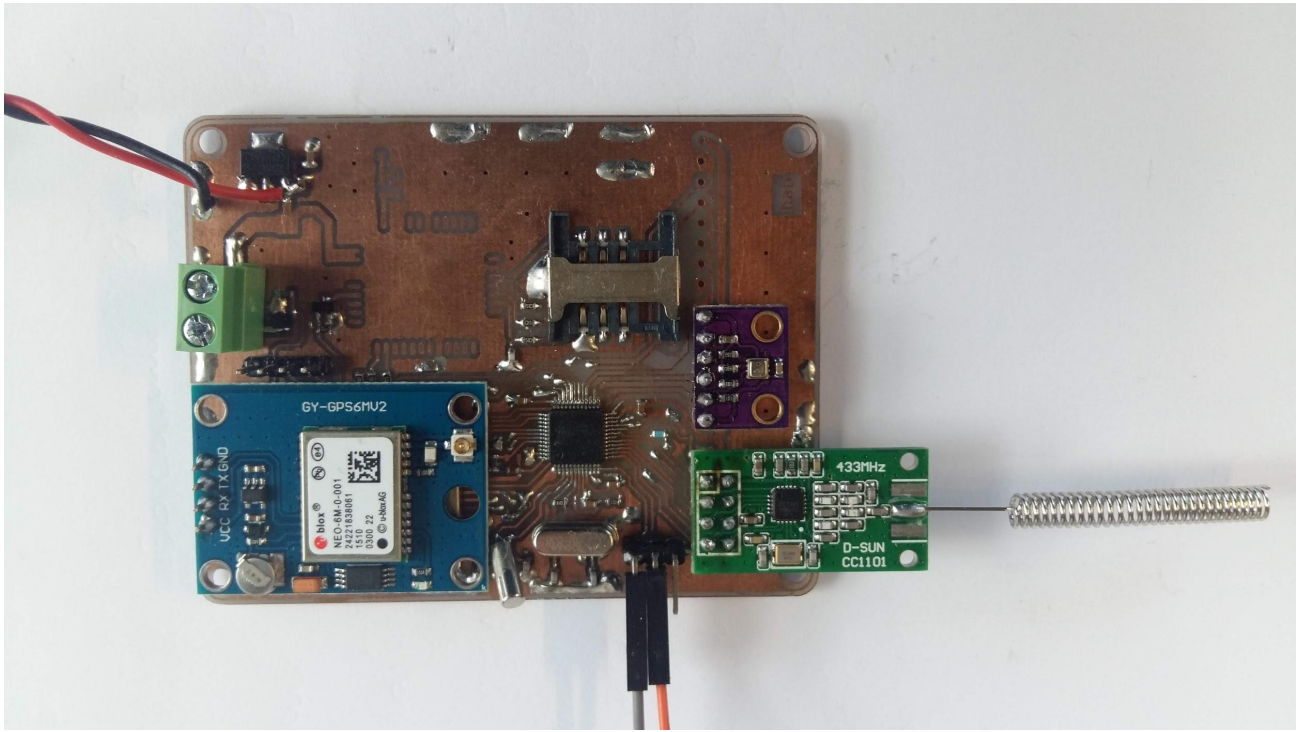
EPAM experience with Zephyr RTOS (BLE mesh)



EPAM experience with Zephyr RTOS (BLE mesh)



EPAM experience with Zephyr RTOS



Q & A