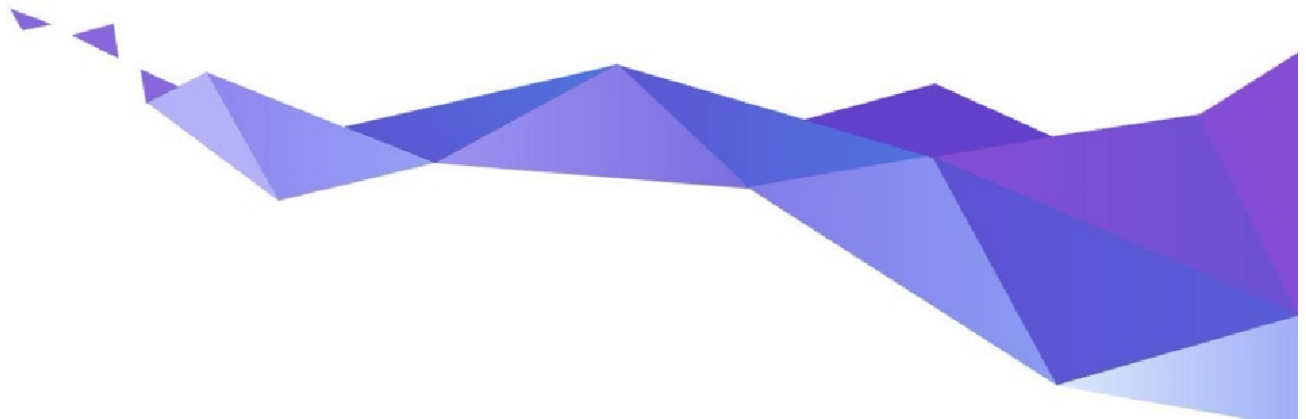


Zephyr Project Overview



What is Zephyr Project?

- Small RTOS for connected, resource-constrained and embedded devices
- Open source (Apache 2.0)
- Linux Foundation supported
- More than RTOS, set to be primary IoT ecosystem, if resource constraints do not allow to run Linux
- Vendor neutral governance



Brief history

- **Origins**

- Originally developed by Wind River Systems as Rocket, a small RTOS.
- Open-sourced in 2016 by Wind River Systems under the Apache 2.0 license.

- **Adoption by Linux Foundation**

- The project is hosted as a vendor-neutral initiative.

- **Community Growth**

- Backed by major technology companies: Intel, Nordic, Qualcomm, NXP, ...
- Continues to grow with contributions from embedded developers and companies worldwide.

Features overview

- **Lightweight kernel & supporting drivers and services**
- **Portable, secure, power-efficient**
- **Highly connected**
 - Bluetooth 5.0 & BLE
 - Wi-Fi, Ethernet, CANbus, ...
 - IoT protocols: CoAP, LwM2M, MQTT, OpenThread, ...
 - USB & USB-C
- **Complete developer environment**
 - Toolchain and HAL management
 - Logging, tracing, debugging,
 - Emulation/Simulation
 - Testing framework



Use cases



Products Running Zephyr Today



Oticon More
Hearing Aid



Lildog & Lilcat
Pet Tracker



Livestock Tracker



Moto Watch 100



Samsung Galaxy
Ring



Progllove



Adhoc Smart Waste



Google
Chromebook



Framework laptop



Keeb.io BDN9



Hati-ACE



Safety Pod



BLiXT solid state
circuit breaker



Aethero Deimos
Satellite



PHYTEC Distancer



Laird Connectivity
sensors & gateways



BeST pump
monitoring



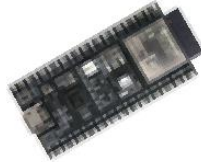
Vestas Wind
Turbines

 zephyrproject.org/products-running-zephyr

750+ Boards Supported



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

+220 Sensors Already Integrated

adt7420
adxl345
adxl362
adxl372
ak8975
amg88xx
ams_as5600
ams_iAQcore
apds9960
bma280
bmc150_magn
bme280
bme680
bmg160
bmi160
bmi270
bmm150
bmp388
bq274xx
ccs811

dht
dps310
ds18b20
ens
esp8266
fda
fx
fx
grove
grow_r502a
hmc5883l
hp206c
ht221
img4250d
im42605
im42670
im42683
icp1125
iis2dh
iis2dlpc

```

lis2iclx
iis2mdc
iis3dhho
ina219
ina23
isl2035
ism332dxx
ite_tac_ite_xxx2
ite_vcmp_it8xxx2
lis2dh
lis2ds12
lis2dw12
lis2m
lis3m
lm75
lm77
lps22
lps22hm
lps25hb
lsm303dlhc_magn

```

lsm6ds0
lsm6dsl
lsm6ds0
lsm9ds0
lsm9ds0_mfd
max17055
max17262
max30101
max31875
max44009
max6675
mchp_tach_xec
mcp9808
mcu_lacmp
mhz19
mpr
mpu6050
mpu9250
ms5607
ms5837

nrf5
 nuvoton_adc_cmnpnpcx
 nuvoton_tn_npcx
 nxp_kin
 opt300
 pcnt_es3
 pms7003
 qdec_m
 qdec_nrfx
 qdec_sam
 qdec_stm32
 rpi_pico_temp
 sbs_gaug
 sgp40
 sht3xd
 sht4x
 shtcx
 si7006
 si7055
 si7060

si7210
sm3511lt
stm32_temp
stm32_vbat
stmemsc
stts751
sx9500
th02
ti_hdc
ti_hdc20xx
tmp007
tmp108
tmp112
tmp116
vcnl4040
vl53l0x
wsen_hids
wsen_itds

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

Supported Hardware Architectures



Cortex-M, Cortex-R
& Cortex-A



x86 & x86_64



32 & 64 bit

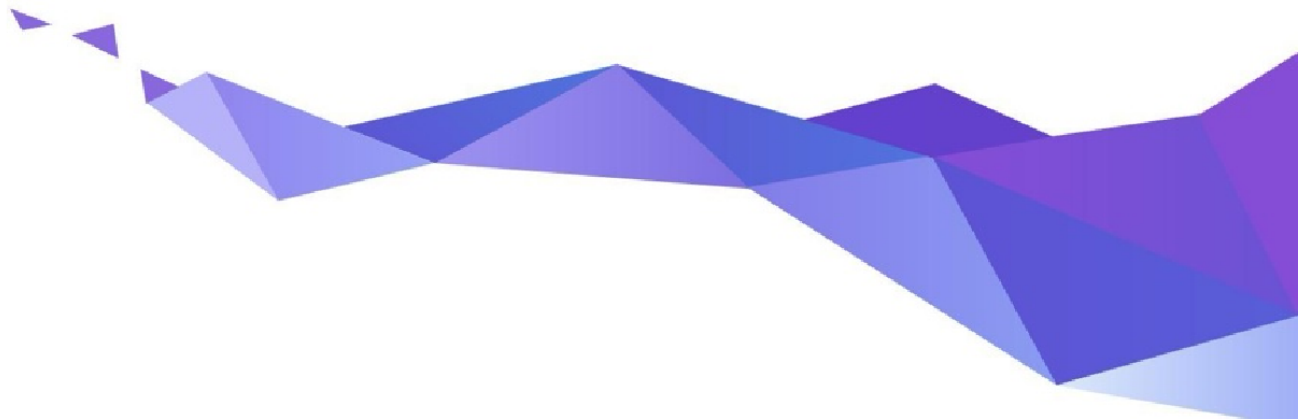


Xtensa



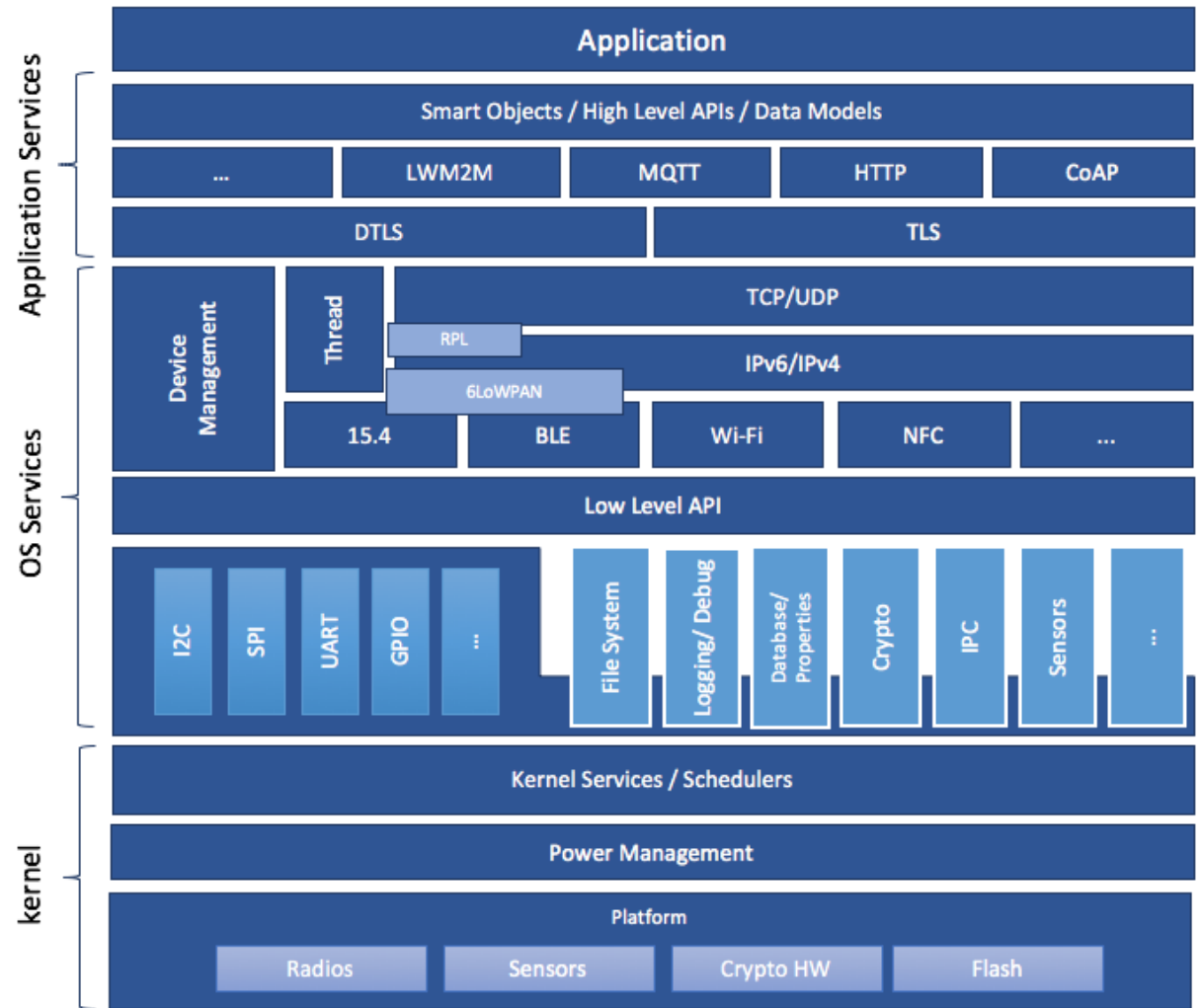
docs.zephyrproject.org/latest/hardware/index.html#hardware-support

Diving into Zephyr's features



Architecture

- Zephyr's architecture is divided into **OS part** (kernel + services) and **Application Services**.
- The **OS part** includes platform-specific drivers, I/O APIs, file systems, and a cryptographic library
- **Application Services** handle user-defined functionality.



Devicetree

Describe & configure the available hardware on the target system

Decouple the application from the hardware

+ **Kconfig** for all things configuration

 docs.zephyrproject.org/latest/build/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 = <0x5f >;  
    };  
  
    // ...  
};
```

.dts file example

West meta-tool

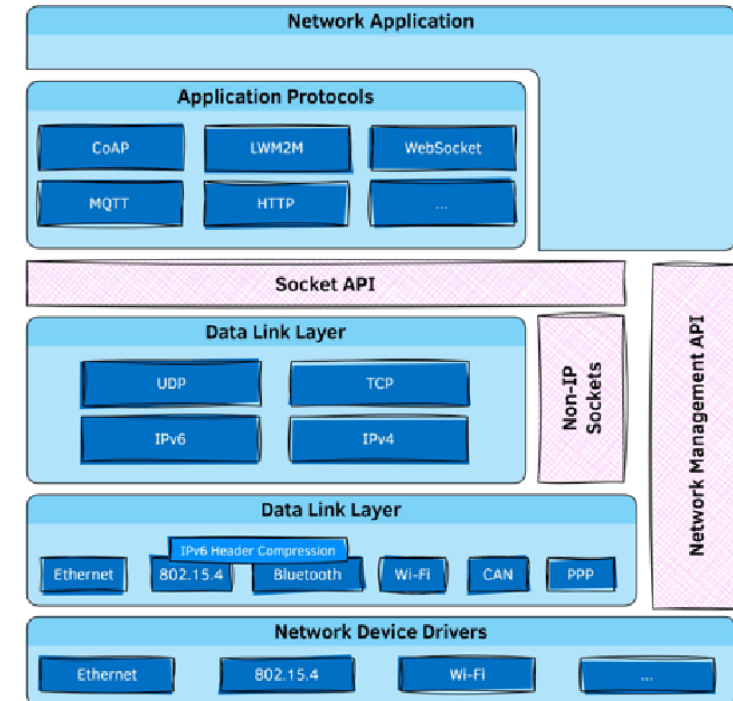
- **Module Management**
 - Simplifies Versioning and integration of various modules/libraries in the build system
- **Build**
- Extensible **command-line interface**
 - e.g. custom commands for specific board
 - Static code analysis, RAM/ROM reports

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



Power Management

- **Goal**: reduce power consumption while preserving responsiveness
- **Key concepts**
 - **Tickless kernel**
 - System PM: idle thread, interruptions only for registered events
 - Device PM: device drivers can react to PM state changes
- Handled by the kernel / Customizable by the user

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

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

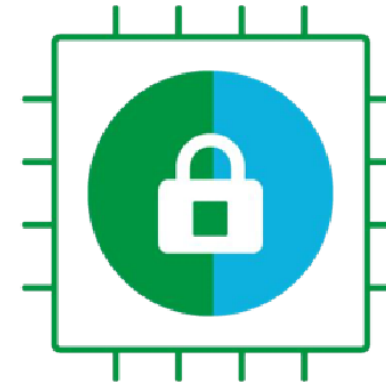
Hardware security

- **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

 docs.zephyrproject.org/latest/guides/portability/posix.html

A real-time OS

Benchmark on Arm Cortex-M4F running at 120 MHz

Operation	Time
Thread create	2.5 μ s
Thread start	3.6 μ s
Thread suspend	3.3 μ s
Thread resume	3.8 μ s
Context switch (yield)	2.2 μ s
Get semaphore	0.6 μ s
Put semaphore	1.1 μ s

 github.com/zephyrproject-rtos/zephyr/tree/main/tests/benchmarks

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

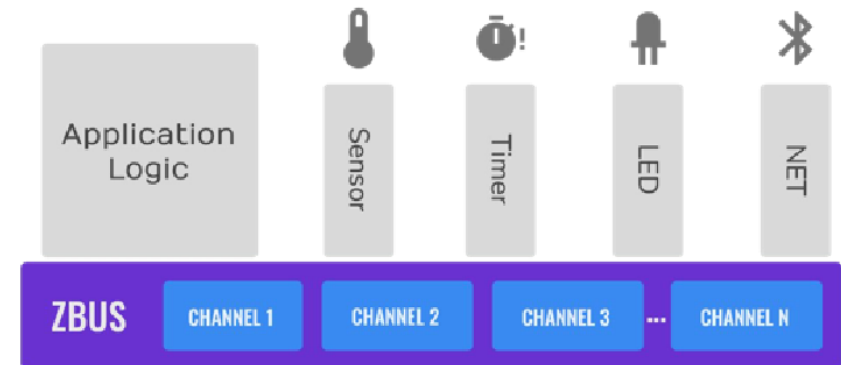


Inter-Process Communication

- **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

Object	Bidirectional?	Data structure
FIFO	✗	Queue
LIFO	✗	Queue
Stack	✗	Array
Message queue	✗	Ring buffer
Mailbox	✓	Queue
Pipe	✗	Ring buffer

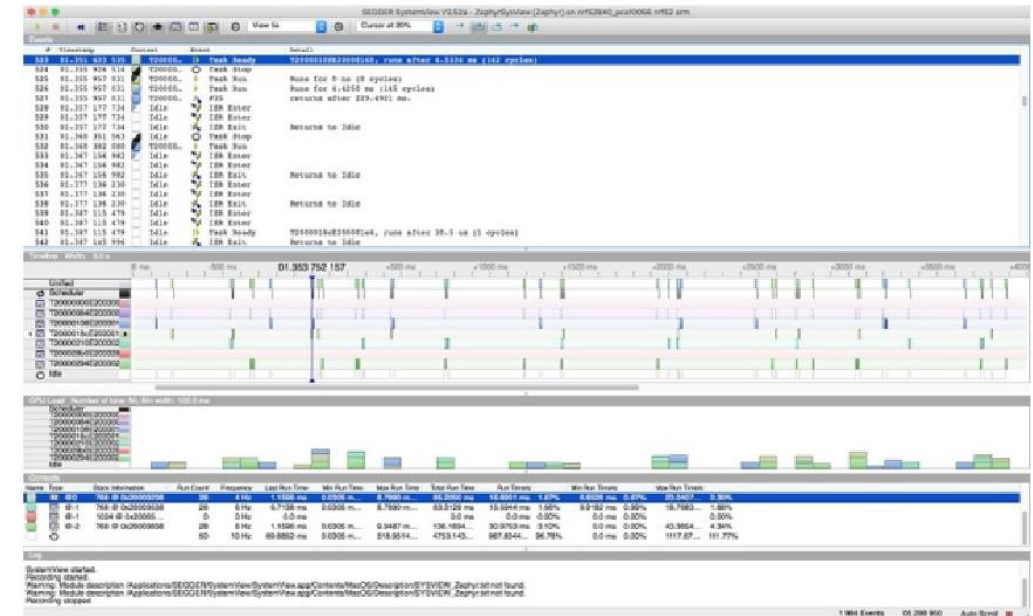
Data passing objects available in Zephyr kernel



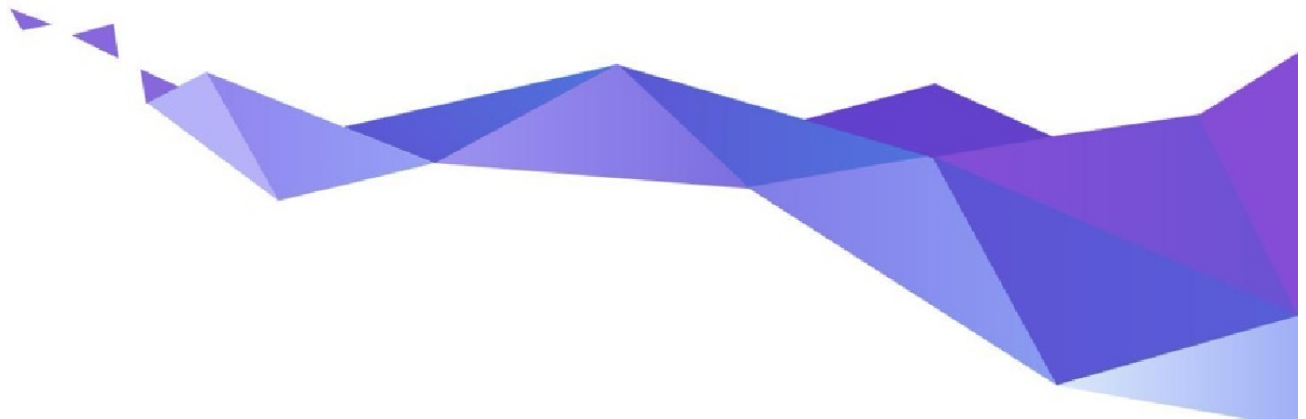
A typical zbus application architecture

Tracing & Debugging

- Advanced **logging** framework
 - Multiple backends (UART, network)
 - Compile-time & runtime filtering
- **Tracing** framework
 - Visualize the inner-working of the kernel and its various subsystems
 - Object tracking (mutexes, timers, etc.)



Ecosystem & Governance



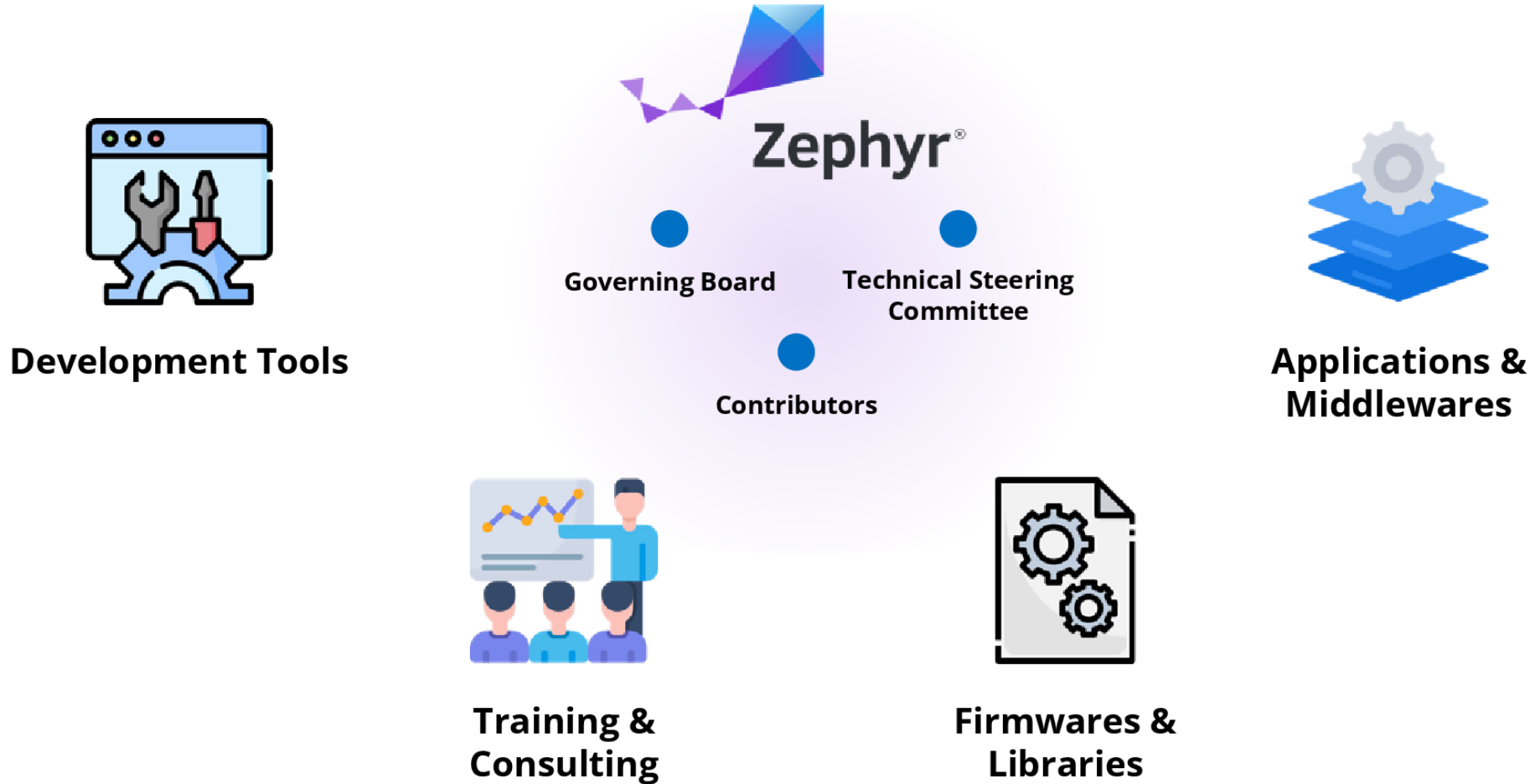
Zephyr Project: Platinum Members



Zephyr Project: Silver Members



Vibrant Ecosystem



Ecosystem: Dev Tools



Development Tools



Training & Consulting



Firmwares & Libraries



Applications & Middlewares

IDE



Compilers



Debuggers / Tracing Tools



Emulation / Simulation



Ecosystem: Training & Consulting



Development Tools



Training & Consulting



Firmwares & Libraries



Applications & Middlewares

Training



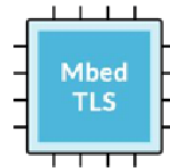
Services & Consulting



Ecosystem: Firmware & Libraries



Security



TinyML



Language runtimes



Others



Ecosystem: Apps & Middlewares



Development Tools



Training & Consulting



Firmwares & Libraries



Applications & Middlewares

Remote Management



HERALD



Golioth



AVSYSTEM



Blynk



Memfault



STERNUM



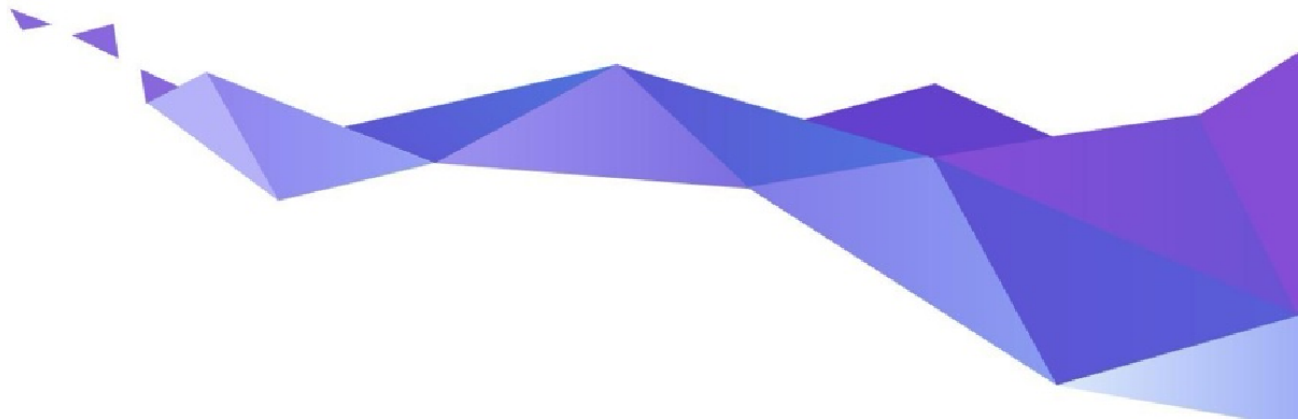
blues

Robotics



ROS

Zephyr in the RTOS landscape

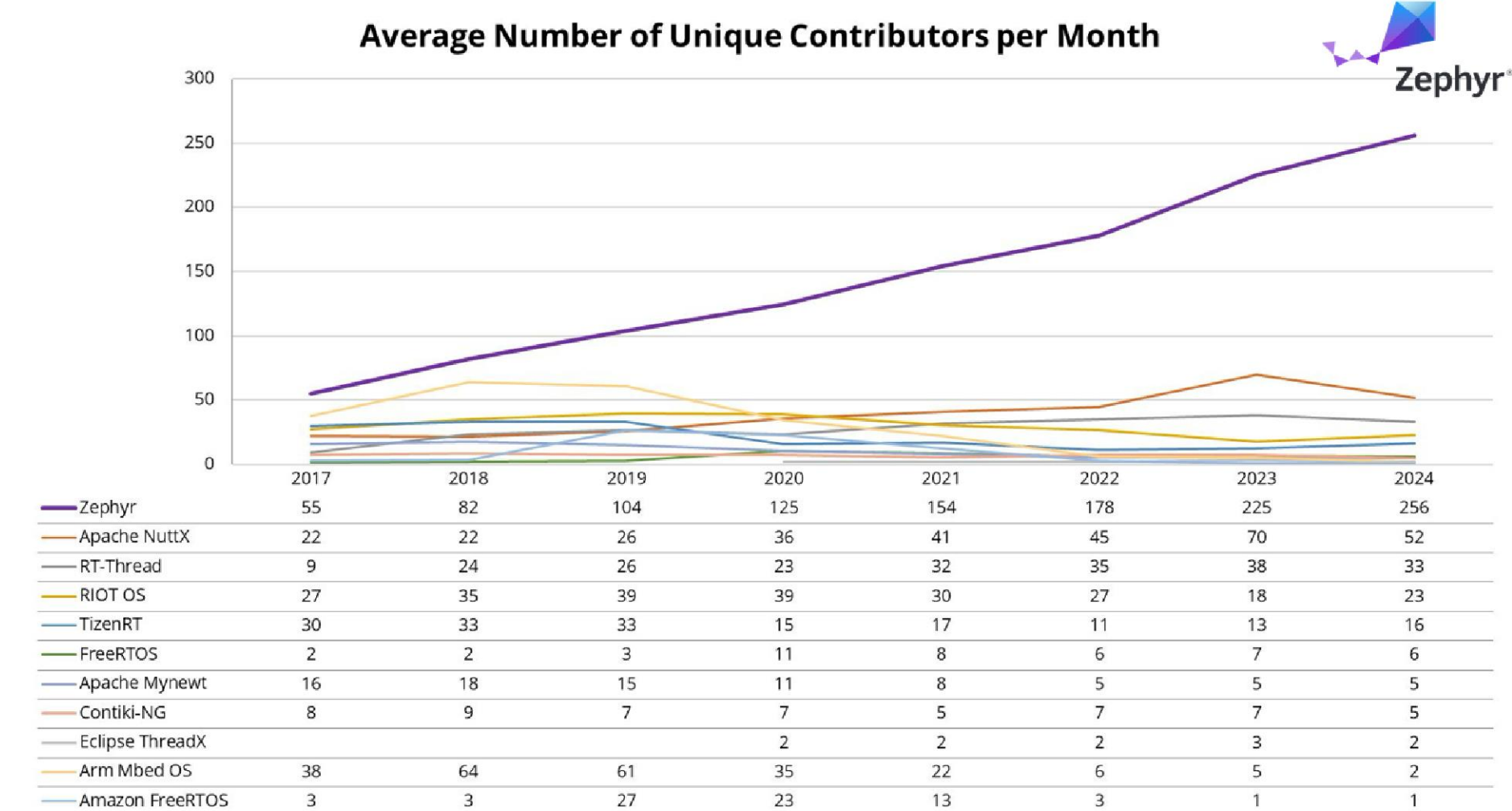


Comparison with Other RTOSs

Feature	Zephyr	FreeRTOS	AWS FreeRTOS	ThreadX	RTEMS
Governance	Linux Foundation	Amazon Web Services	Amazon Web Services	Microsoft	RTEMS Foundation
Licensing	Apache 2.0	MIT	MIT	Azure RTOS License	GPL with linking exception
Modularity	High	Medium	Medium	Medium	High
Networking	Extensive (e.g., Bluetooth, Wi-Fi)	Limited	Broad (IoT protocols and stacks)	Moderate	Limited
Ecosystem	Broad (IoT and edge-focused)	IoT-oriented	Tight AWS IoT and cloud integrations	Enterprise IoT (Azure integrations)	Aerospace and mission-critical systems
Security	Advanced (secure boot, TEE, etc.)	Basic	Advanced (secure AWS IoT integrations)	Advanced (Microsoft IoT integration)	Advanced
Industry Focus	IoT, industrial, consumer devices	IoT	IoT and cloud-connected devices	IoT and industrial systems	Aerospace and scientific systems

.

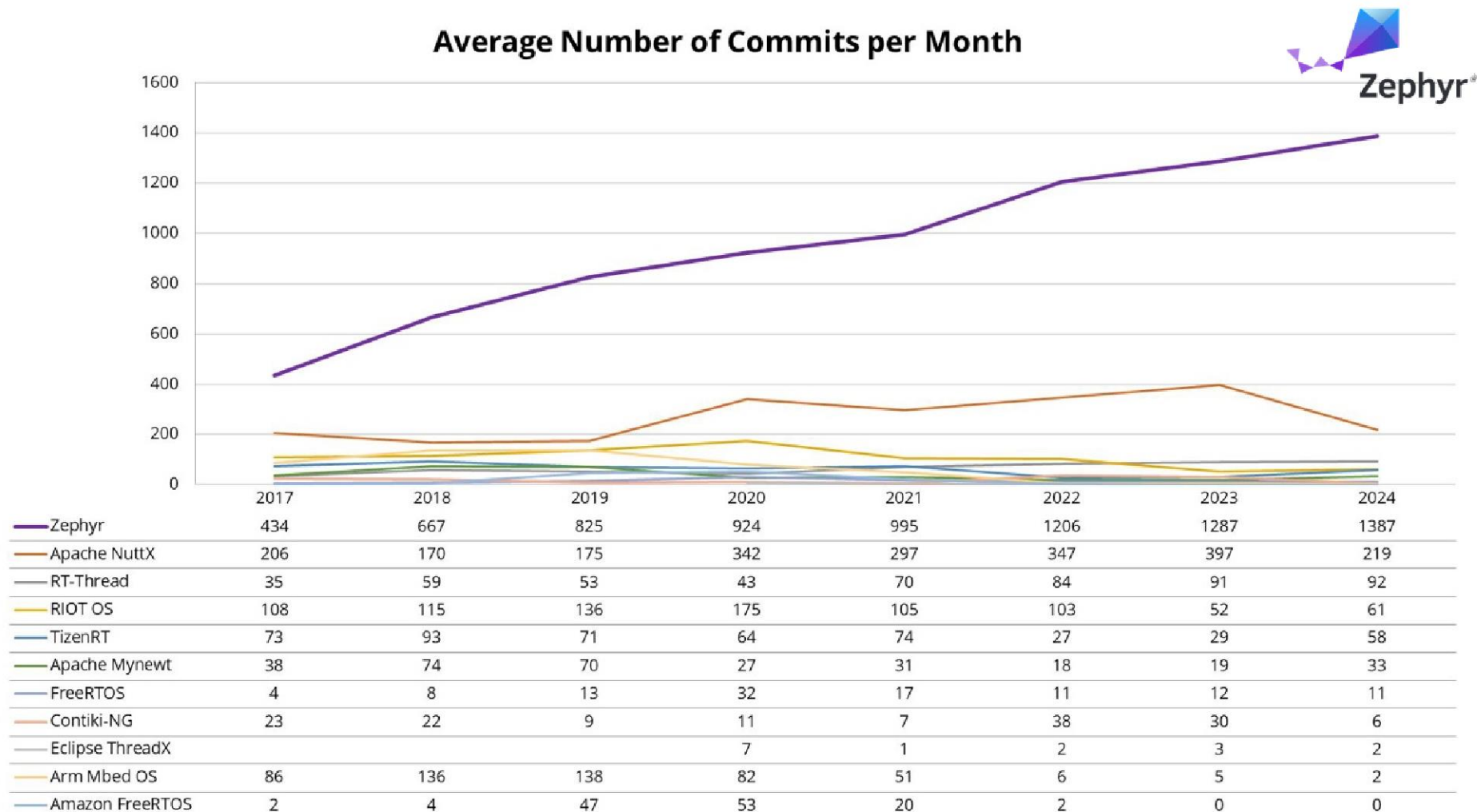
Zephyr's Repository Activity



© 2024 The Zephyr Project — Content made available under CC BY-SA 4.0.



Zephyr's Repository Activity



© 2024 The Zephyr Project — Content made available under CC BY-SA 4.0.



Zephyr Trends

- **RISC-V and AI/ML:**

- Increasing adoption for AI and ML workloads on low-power devices, with RISC-V support expanding.
- Zephyr's ML frameworks and TensorFlow Lite Micro integration address these needs.

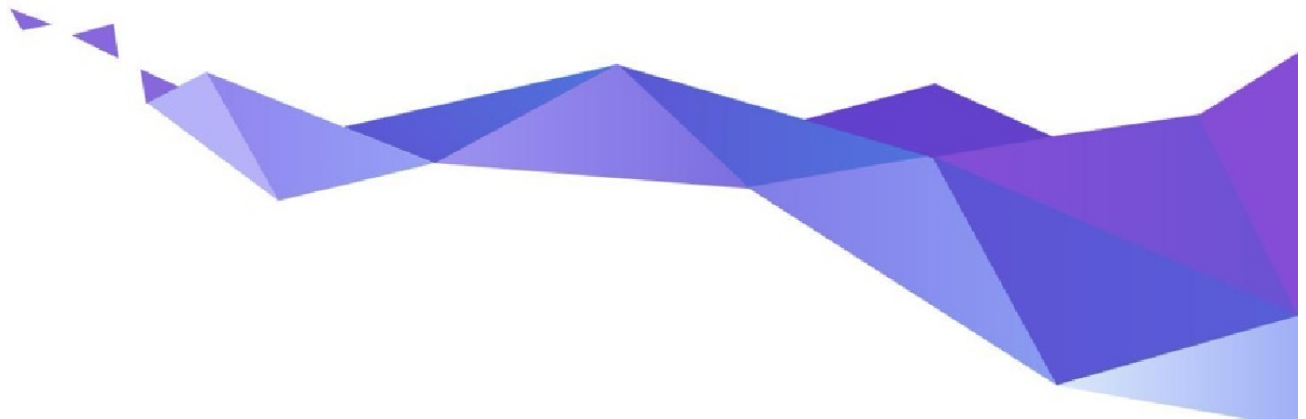
- **IoT and Edge Computing:**

- Seamless cloud integration for IoT platforms like Azure IoT, AWS IoT, and ThingsBoard.
- Efficient power management makes Zephyr ideal for edge computing.

- **Safety-Critical Applications:**

- Zephyr is working toward certifications like **ISO 26262** (automotive) and **IEC 61508** (industrial automation).

Getting started & links



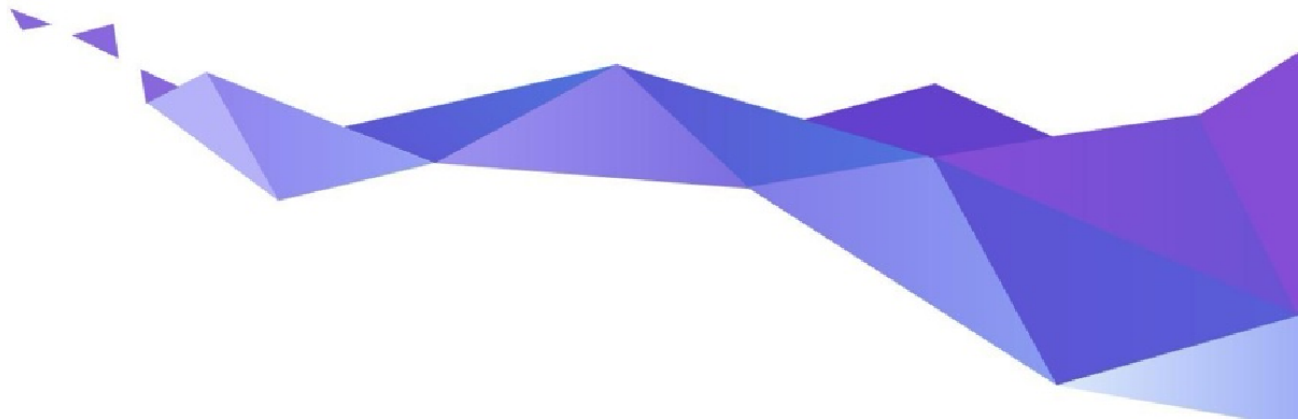
Getting started

- Check out the official [Getting Started Guide](#)
- Dig into the hundreds of [code samples](#)
- Check the catalog of 100s of available [Devicetree bindings](#)
- No driver for your HW?
 - Chances are a similar driver already exists and writing one is not as hard or daunting as you would think!

Resources

- **Zephyr Project Website:** <https://zephyrproject.org>
- **Help:** [Asking for Help Tips](#)
- **Documentation:** <http://docs.zephyrproject.org> ([Getting Started Guide](#))
- **Source Code:** <https://github.com/zephyrproject-rtos/zephyr> is the main repository; <https://elixir.bootlin.com/zephyr/latest/source> contains a searchable index
- **Releases:** <https://github.com/zephyrproject-rtos/zephyr/releases>
- **Samples and example code:** see [Sample and Demo Code Examples](#)
- **Wiki:** [Zephyr GitHub wiki](#)

Zephyr on RVfpga



Zephyr on Rvfpga: intro

- RVfpga is based on Veerwolf SoC
- VeeRwolf is a FuseSoC-based reference platform for the CHIPS Alliance VeeR family of RISC-V cores.
- VeeRwolf can be used with Zephyr Project RTOS
 - Currently it only support v2.7.4 (old LTS maintenance release with fixes)
- It can be used either in ***verilator*** or ***Nexys A7 FPGA*** board

Zephyr on Rvfpga: key steps to build an app

- Install Fusesoc & VeeRwolf
- Install Zephyr
- Add West manifest ([west.yml](#))
- Do `west update`
 - This also installs the [VeeRwolf module](#) which adds support for the board
- Compile as usual

The lab documents include all the necessary details and provide references to the appropriate documentation for further clarification.

Zephyr on Rvfpga: labs

- **Zephyr on VeeRwolf:** This lab illustrates step-by-step how to install all the tools and build some examples.
- **Zephyr Blinky Sample on VeeRwolf:** This lab provides an in-depth analysis of the blinky sample and proposes exercises based on it.
- **VeeRwolf Zephyr Module:** This lab examines the module that implements support for VeeRwolf and suggests exercises related to devicetree overlays.