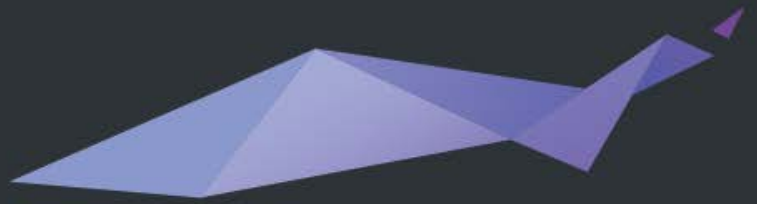


Zephyr Project Overview

Anas Nashif

Agenda

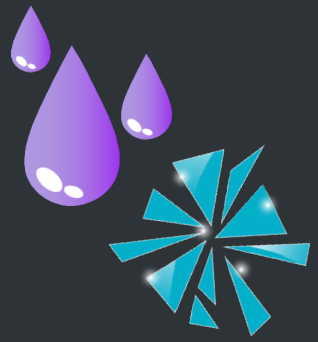
- Zephyr Project Overview
- Architecture Overview
- The Zephyr Kernel
- Getting Involved

What is Zephyr?

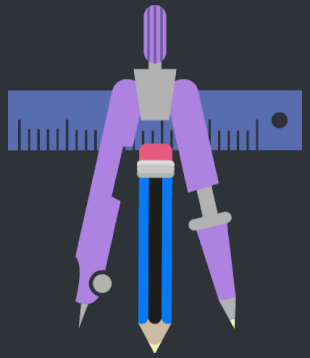
- Microcontroller operating system
- Very small memory footprint (will run in 8k)
- Open Source under Apache* 2.0 license, hosted by Linux* Foundation
- Supports multiple architectures

Small OS & RTOS Market Analysis

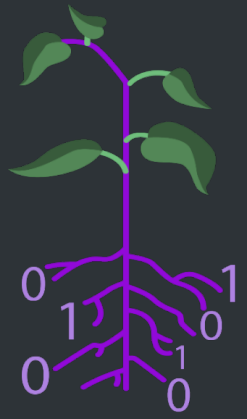
Zephyr™



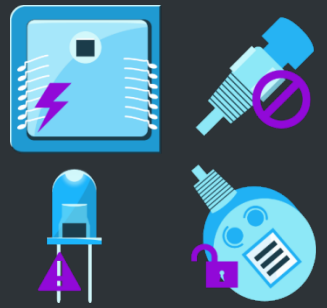
Saturated RTOS/ Fragmentation



Roll Your Own/No OS



Adoption growth in IoT development



Compromised Devices

Opportunity to build a leading IoT OS

Why Zephyr Project?

Zephyr™

Strategic Investment

Best-of-Breed RTOS

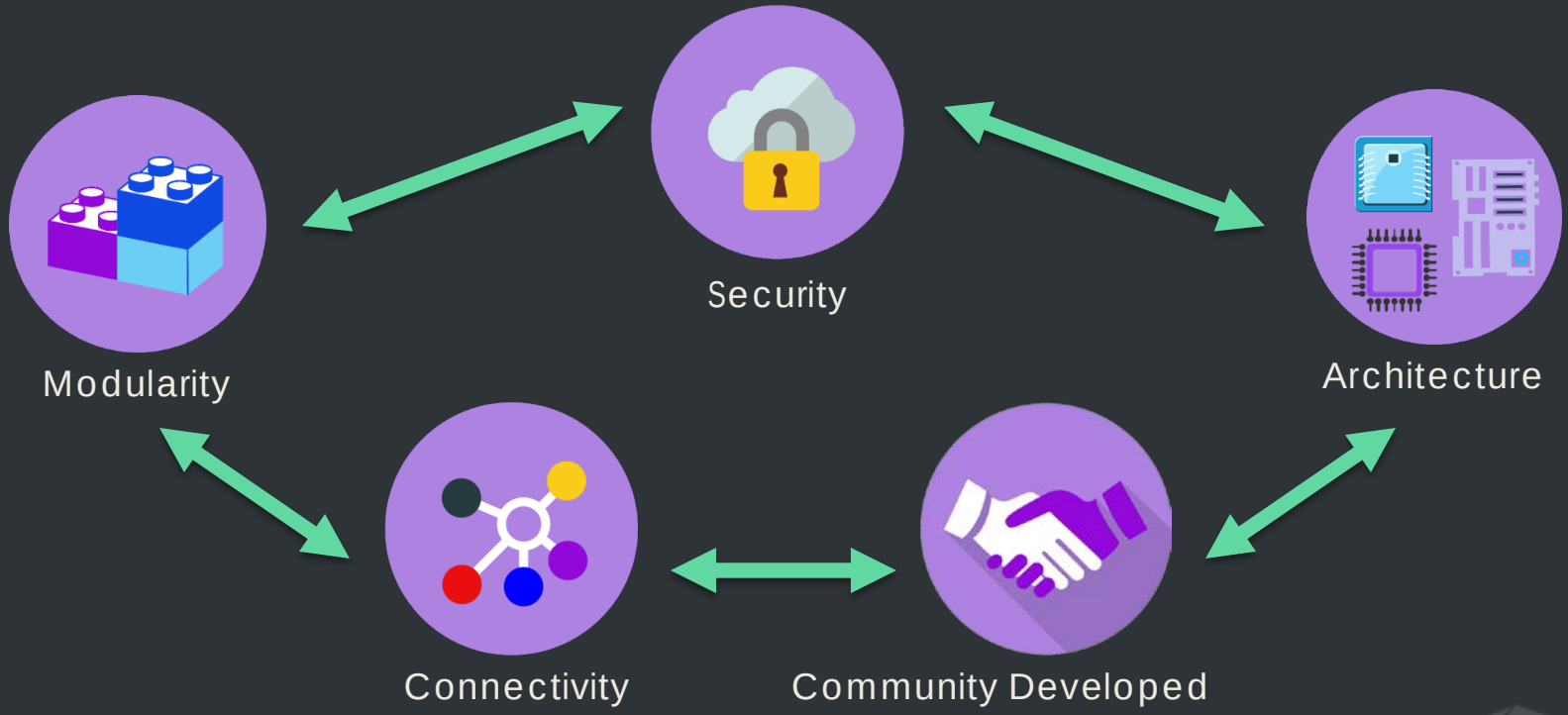
True Open Source

Permissively Licensed

Established Code Base

Secure

Zephyr Project Features



Agenda

- Zephyr Project Overview
- Architecture Overview
- The Zephyr Kernel
- Getting Involved

Top 10 IoT Technologies for 2017/2018 (Gartner)

- IoT Security
- IoT Analytics
- IoT Device Management
- Low-Power, Short-Range Networks
- Low-Power, Wide-Area Networks
- IoT Processors
- IoT Operating Systems
- Event Stream Processing
- IoT Platform
- IoT Standards and Ecosystem

Source: http://www.gartner.com/newsroom/id/3221818

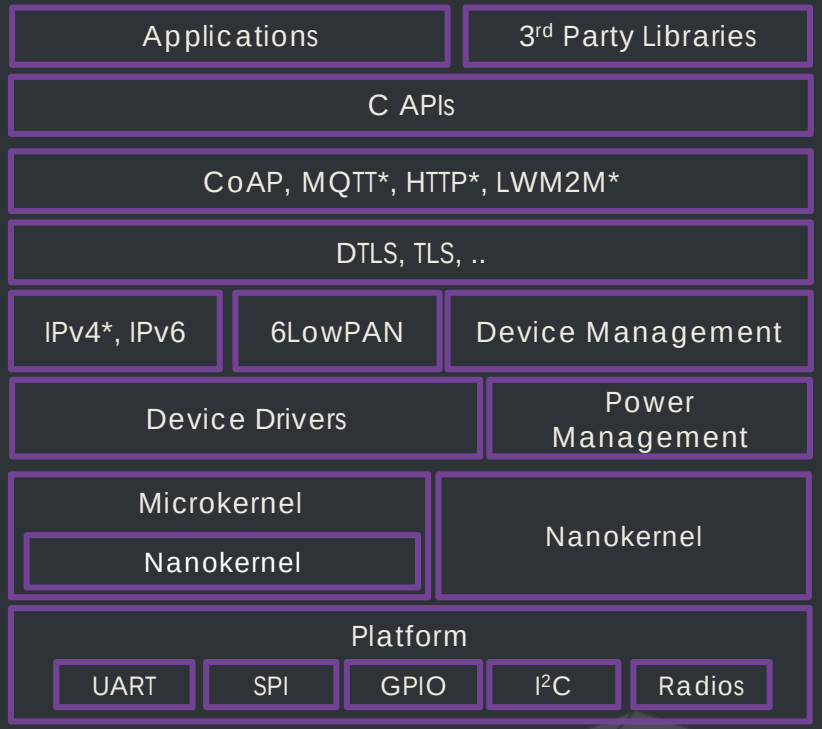
Top 10 IoT Technologies for 2017/2018 (Gartner)

- IoT Security
- IoT Analytics
- IoT Device Management
- Low-Power, Short-Range Networks
- Low-Power, Wide-Area Networks
- IoT Processors
- IoT Operating Systems
- Event Stream Processing
- IoT Platform
- IoT Standards and Ecosystem

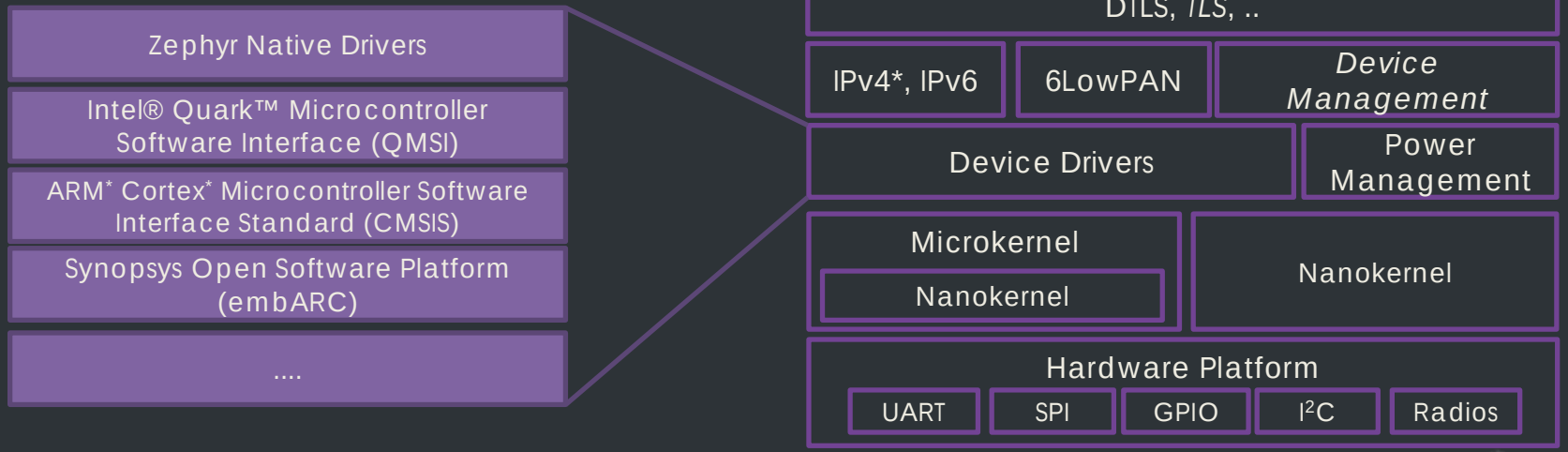
Source: http://www.gartner.com/newsroom/id/3221818

Zephyr Overview

- Provide an OS that runs best on MCUs for wearable and IoT devices, where the cost of the silicon is minimal
- Highly Configurable, Highly Modular
- Kernel mode only
- Two Modes:
  - Nanokernel: Limited functionality targeting small footprint (below 10k)
  - Microkernel (superset of nanokernel): with additional functionality and features
- No user-space and no dynamic runtimes
- Memory and Resources are typically statically allocated
- Cross architecture (IA32, ARM*, ARC, others under discussion)

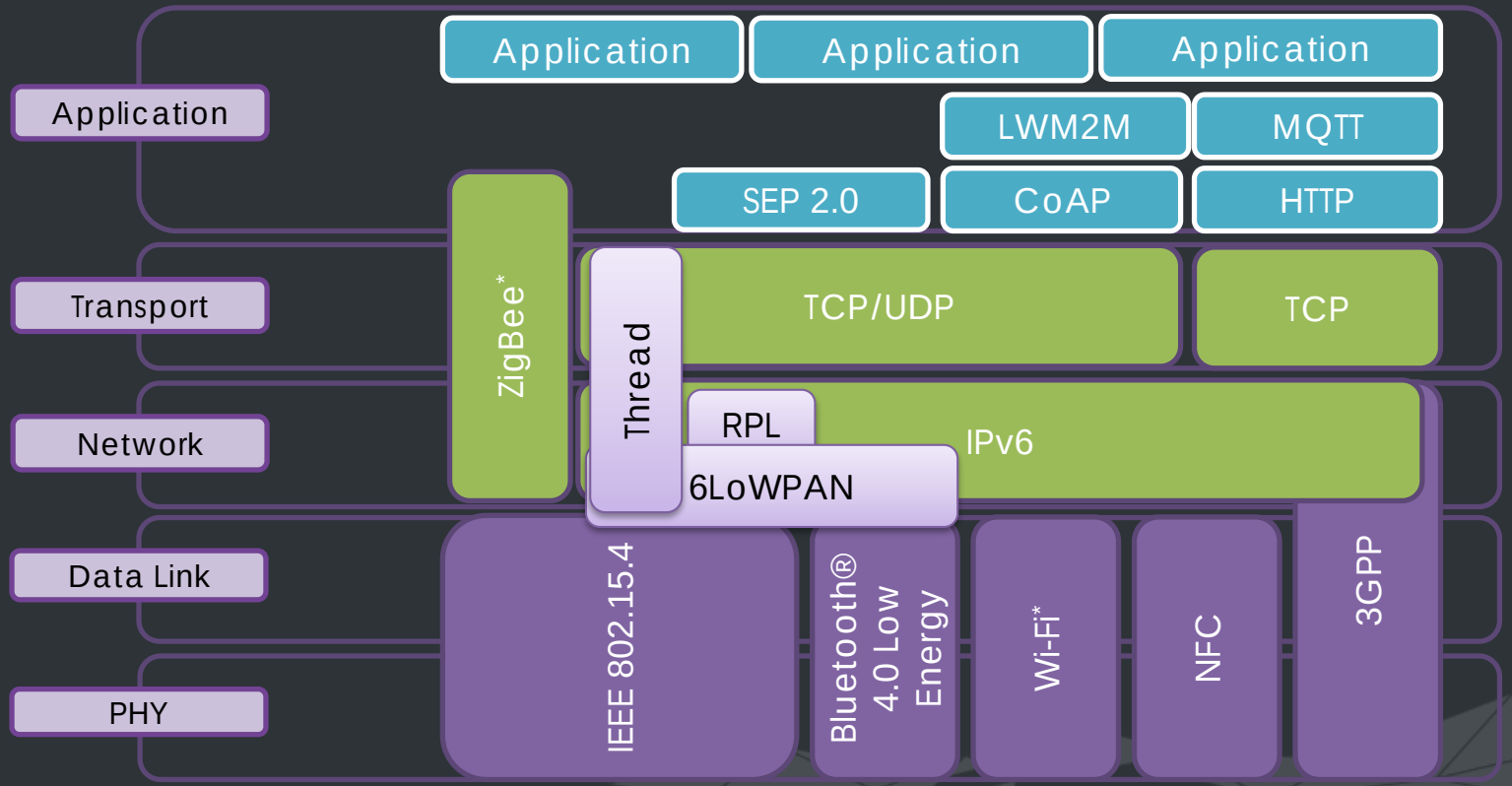


Supporting Driver Frameworks & HALs

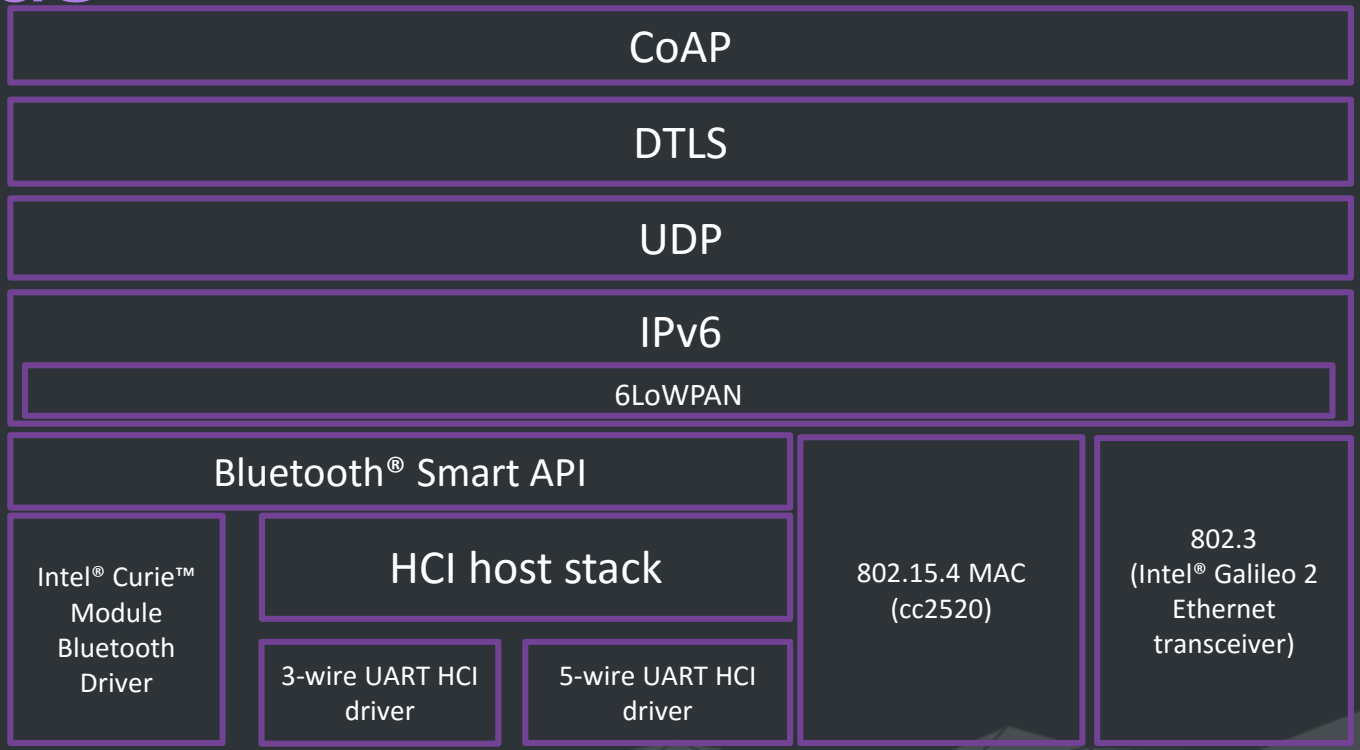


* planned

IOT Technologies: Connectivity

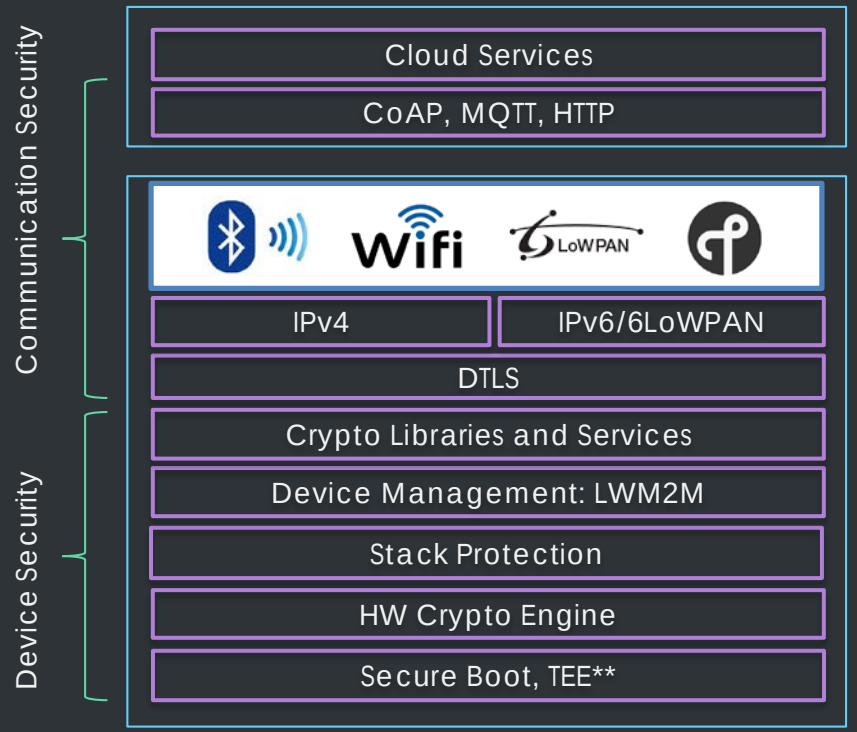


Zephyr Connectivity: Current Status



Integrated Security

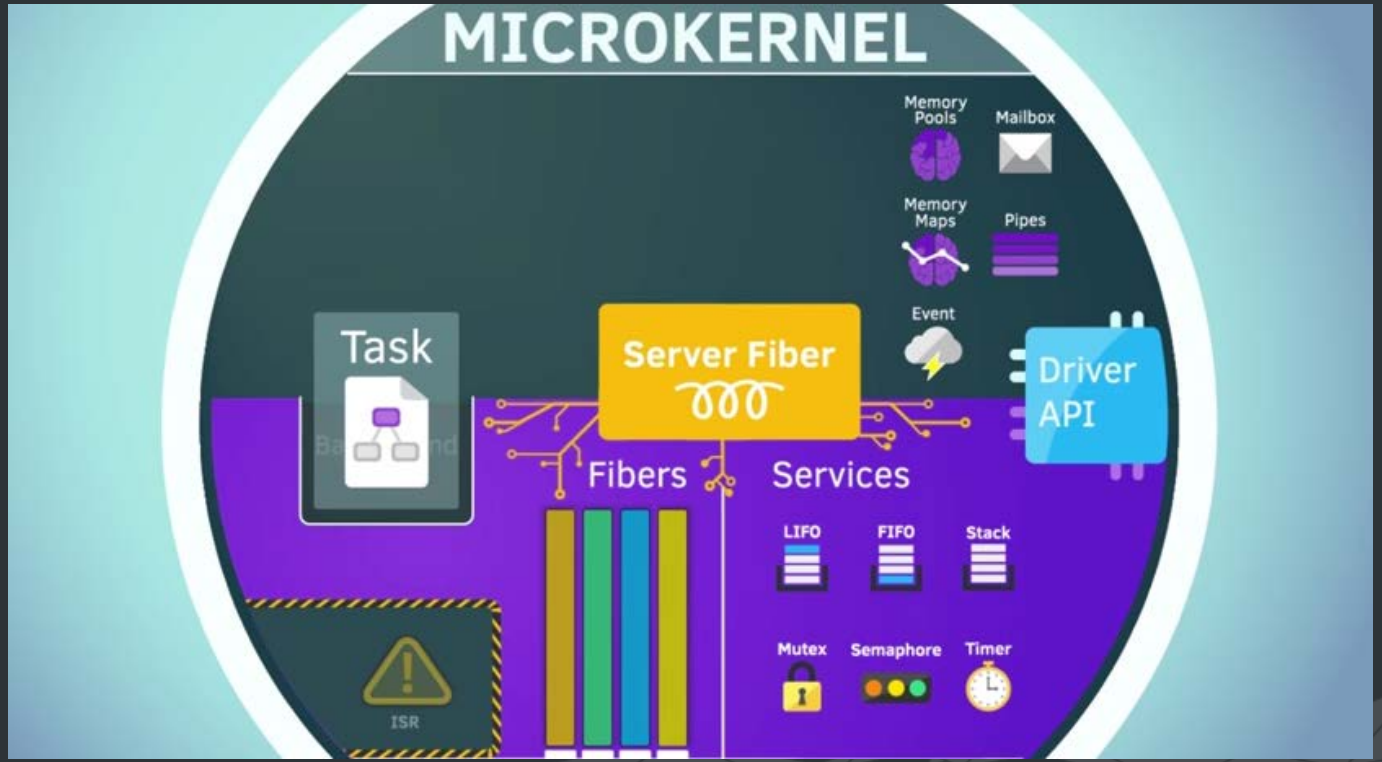
- Standardized building block and robust communication stacks
- Cryptographic library based on TinyCrypt2
- Static and single binary applications, Single address space, No loadable modules
- Planned security features:
  - Device Management and Updates
  - APIs to support vendor specific Crypto implementations (software/hardware)
  - Secure Key Storage



Agenda

- Zephyr Project Overview
- Architecture Overview
- The Zephyr Kernel
- Getting Involved

Zephyr Technical Overview (Video)



Zephyr Kernel – Key Features

Multi-threading services, including both priority-based, non-preemptive fibers and priority-based, preemptive tasks (with optional round robin time-slicing).

Interrupt services, including both compile-time and run-time registration of interrupt handlers, which can be written in C or assembly language.

Inter-thread synchronization services, including binary semaphores, counting semaphores, and mutex semaphores.

Inter-thread data passing services, including basic message queues, enhanced message queues, and byte streams.

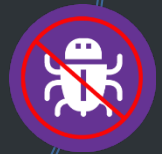
Memory allocation services, including dynamic allocation and freeing of fixed-size or variable-size memory blocks.

Zephyr Kernel – Key Features

Power management services, including tick-less idle and an advanced idling infrastructure.

Highly configurable, allowing an application to incorporate only the capabilities it needs, and to specify their quantity and size.

Zephyr requires all system resources to be defined at compile-time to reduce code size and increase performance.



Provides minimal run-time error checking to reduce code size and increase performance. An optional error checking infrastructure is provided that can assist in debugging during application development

Library based RTOS ("kernel-less")

Library-Based RTOS (“kernel-less”)

- One single executable which is executed in one single address space
- No loader is required to dynamically load applications at run-time
  - Minimizing the operating system code
  - System calls are implemented as function calls
  - No context switches are required when calling an operating system call
- Lack of security through hardware memory separation
  - Application and operating system calls are implemented as thread in the same address space
  - Bugs in one part of the system can affect the whole system

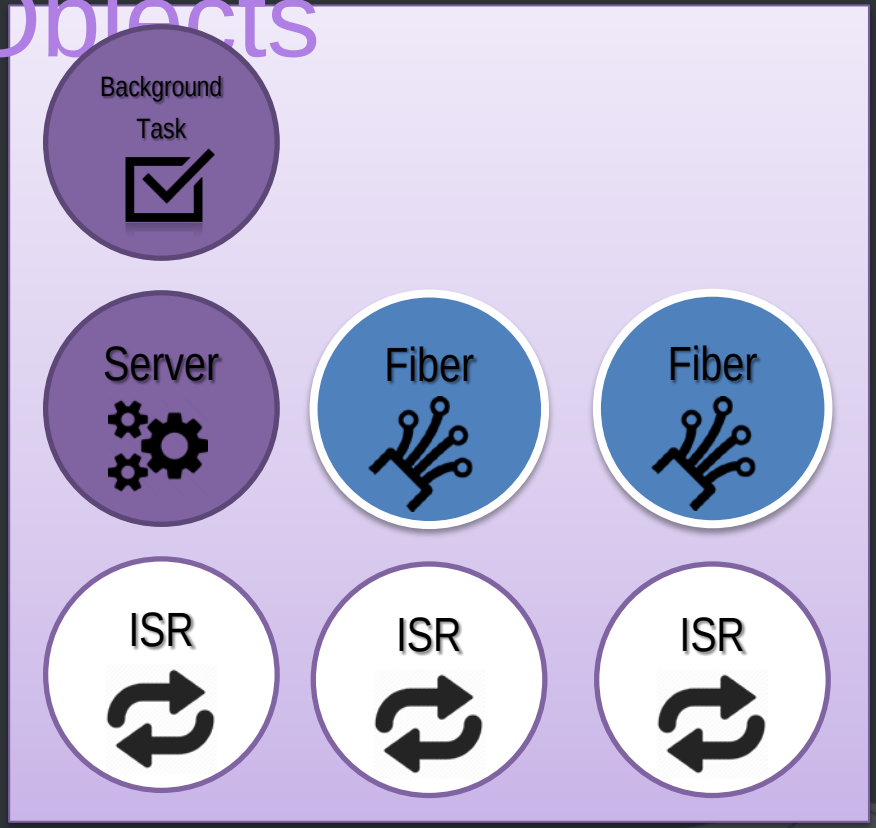
Often more efficient and less time consuming as a full context switch with address space changes.

On small microcontrollers on which only one application is executed this disadvantage is acceptable.

Zephyr Nanokernel Overview

- A high-performance, multi-threaded execution environment with a basic set of kernel features
- Ideal for systems with sparse memory (the kernel itself requires as little as 2 KB!) or only simple multi-threading requirements (such as a set of interrupt handlers and a single background task)
- Examples of such systems include:
  - embedded sensor hubs
  - environmental sensors
  - simple LED wearables
  - store inventory tags

Nanokernel Scheduling and Objects



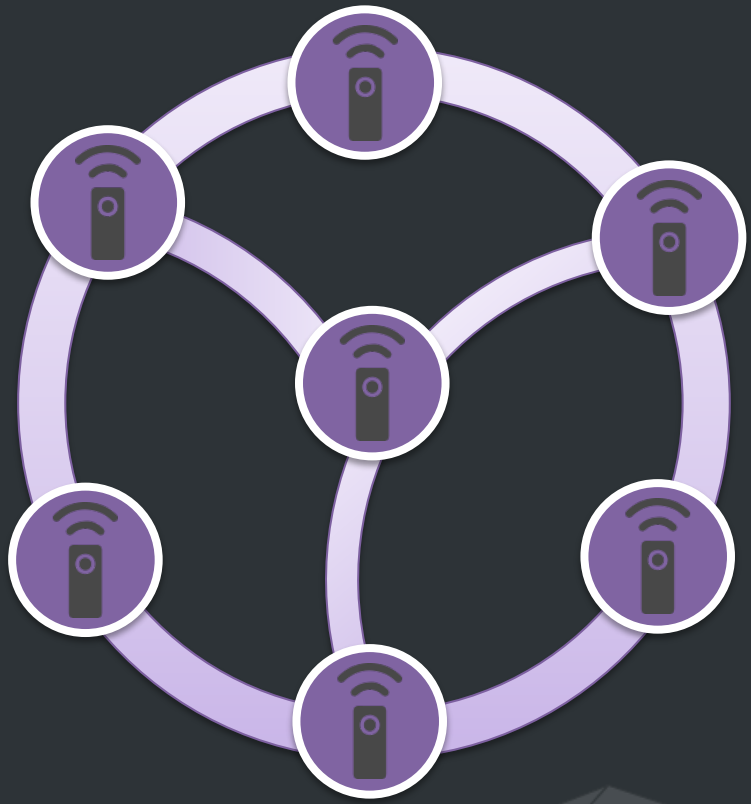
Kernel Services

Nanokernel Scheduling and Objects

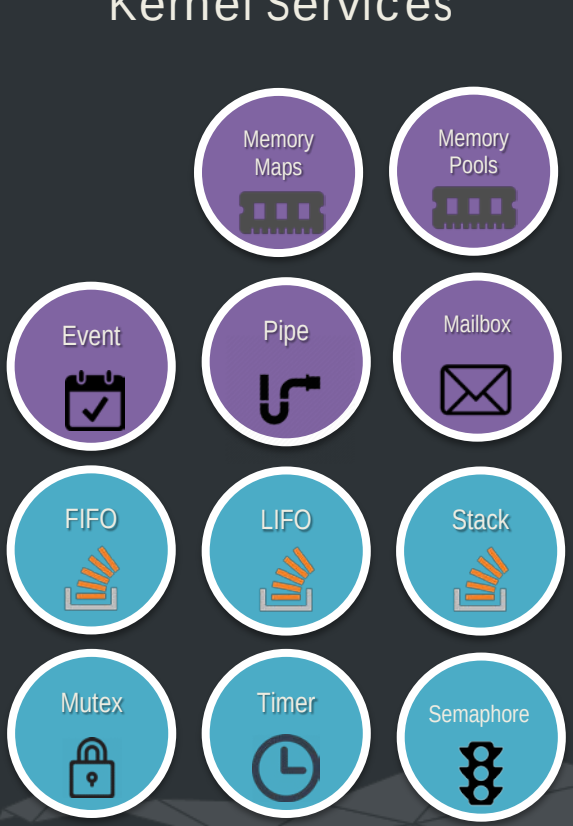
- Cooperatively scheduled
- Run until they yield or call a blocking API
  - Marked as not runnable
  - Next highest priority fiber is then run
- Typically used for device drivers and performance-critical work

Zephyr Microkernel Overview

- Supplements the capabilities of the nanokernel to provide a richer set of kernel features
- Suitable for systems with
  - heftier memory (50 to 900 KB)
  - multiple communication devices (like Wi-Fi and Bluetooth® Low Energy)
  - and multiple data processing tasks
- Examples of such systems include:
  - Fitness wearables
  - Smart watches
  - IoT wireless gateways



Microkernel Scheduling and Objects



Microkernel Scheduling and Objects

- A task is scheduled when no fibers are runnable
- Tasks are preemptible
- Highest priority task runs first
- Round-robin time-slicing between tasks of equal priority
- Used for data processing

Agenda

- Zephyr Project Overview
- Architecture Overview
- The Zephyr Kernel
- Getting Involved

Best-of-Breed Tools

- Kbuild - The build system of the Linux kernel
- Kconfig - The configuration system of the Linux kernel
- Rich and Powerful SDK developed specifically for Zephyr and powered by the Yocto project:
  - 5 different cross-compilers for all supported architectures and platforms
  - Support for baremetal c library, based on newlib
  - Host tools needed for debugging and downloading images (flashing) into target platforms
- Builds natively on Linux*, MacOS* and Microsoft* Windows
- Building on all various operating systems using Docker containers

Supported Platforms

Zephyr™

Arduino* 101

Intel® Quark™ D2000 CRB

2nd Generation Intel® Galileo

FRDM-K64F

Arduino Due

ARC EM Starter Kit

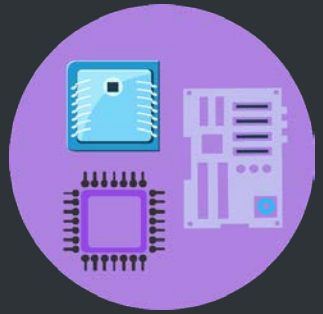
ST Nucleo F103RB

► More platforms and boards to follow ◄

Summary and Next Steps

- Established code-base with **strong** community and industry support
- Focused and committed to support industry standards
- Innovative and forward looking by design
- Open to all
- A true open-source project, ready for **your ideas**, feedback and contributions

Participate!



Impact architecture

Direction

Marketing / Advocacy

Decision making

Examine the code and join!

Visit Us!
Booth #108
Zephyr™ Project Demos
from our member
companies

Zephyr™

www.zephyrproject.org

Other Zephyr™ Project Sessions @ ELC/OpenIoT		
Zephyr™ Project Security	Monday, April 4, 2016	Harbor Ballroom E
Zephyr™ Project Overview	Tuesday, April 5, 2016	
Power Management in Zephyr™ RTOS	Wednesday, April 6, 2016	