MICRO-ROS

ROS-INDUSTRIAL SPRING WORKSHOP

MAY 7TH 2018

DR.-ING. INGO LÜTKEBOHLE, BOSCH CORPORATE RESEARCH

Partially supported by EU grant 780785

“In the future it should be possible to implement the ROS protocol directly on the devices embedded system”

ROS2 Design Wiki “Stories”

Micro-ROS

Robots are networks of devices

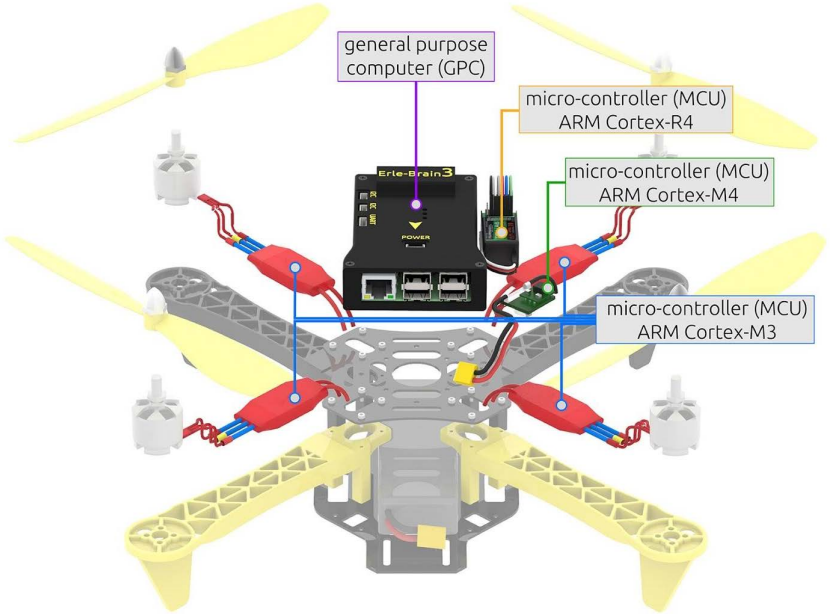


Image source: Erle Robotics, taken from OFERA proposal.

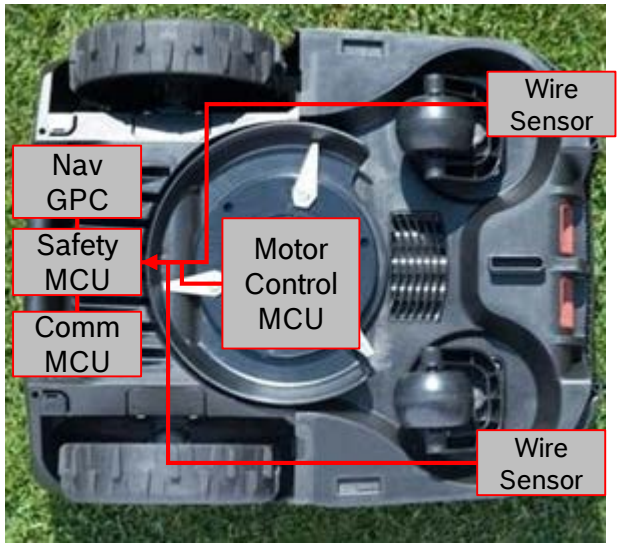
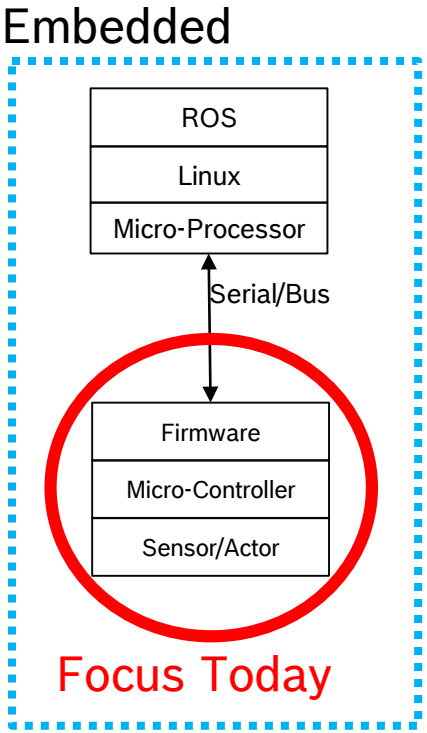


Image source: Bosch PowerTools GmbH, All rights reserved

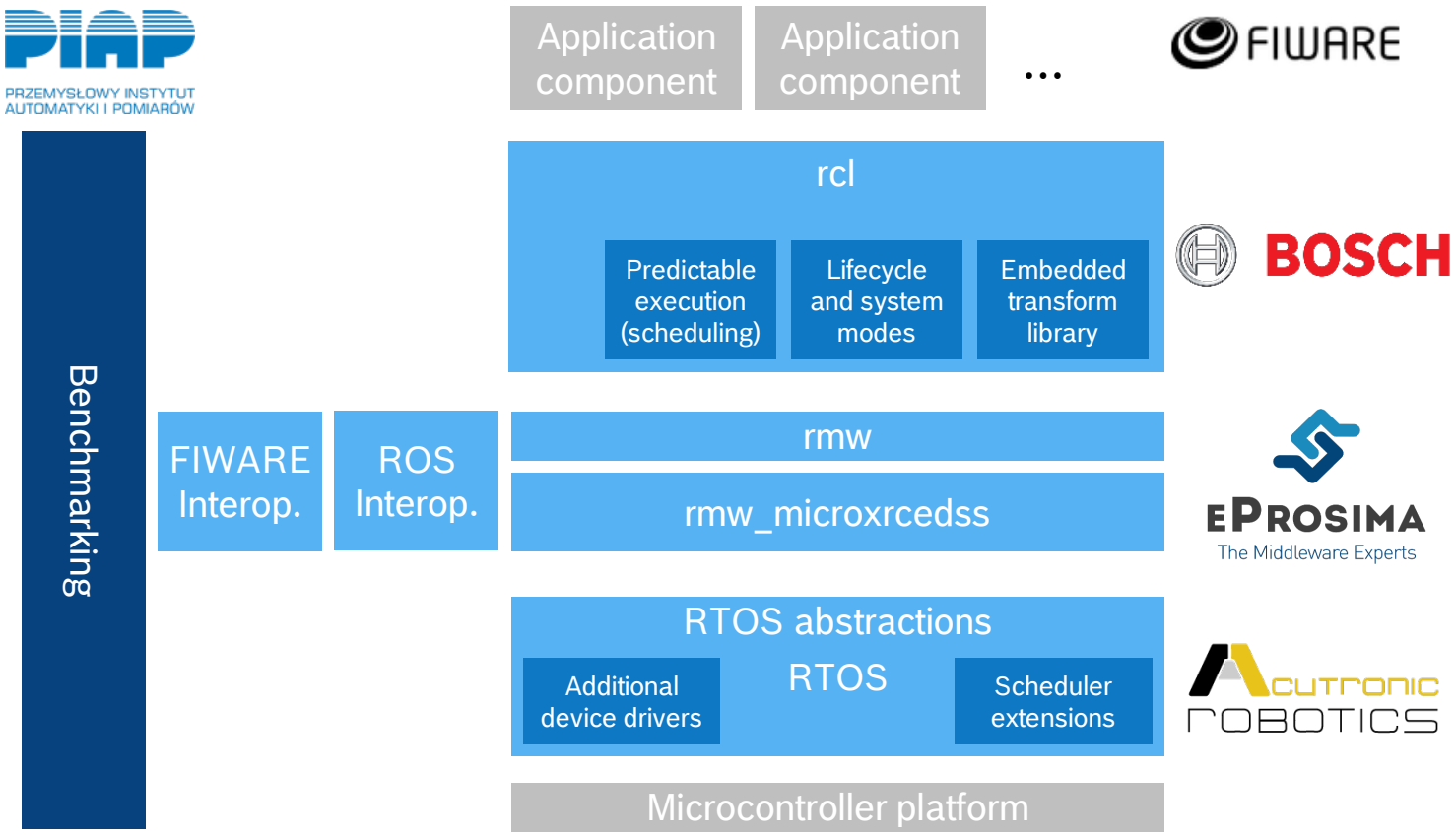


Micro-ROS
Open Framework for Embedded Robot Applications (OFERA)

OFERA will extend ROS2 to allow its use in MCUs

https://ofera.eu/

The OFERA project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780785



Micro-ROS
Situation

- ROS+Linux is a powerful combo
  - Excellent libraries for perception, planning, networking, etc
  - Unified developer eco-system: One kernel, most devices
  - It's what we all have on our desks
- But…
  - Issue 1: Hardware access
  - Issue 2: Hard, low-latency RT
  - Issue 3: Power saving
  - Issue 4: Safety

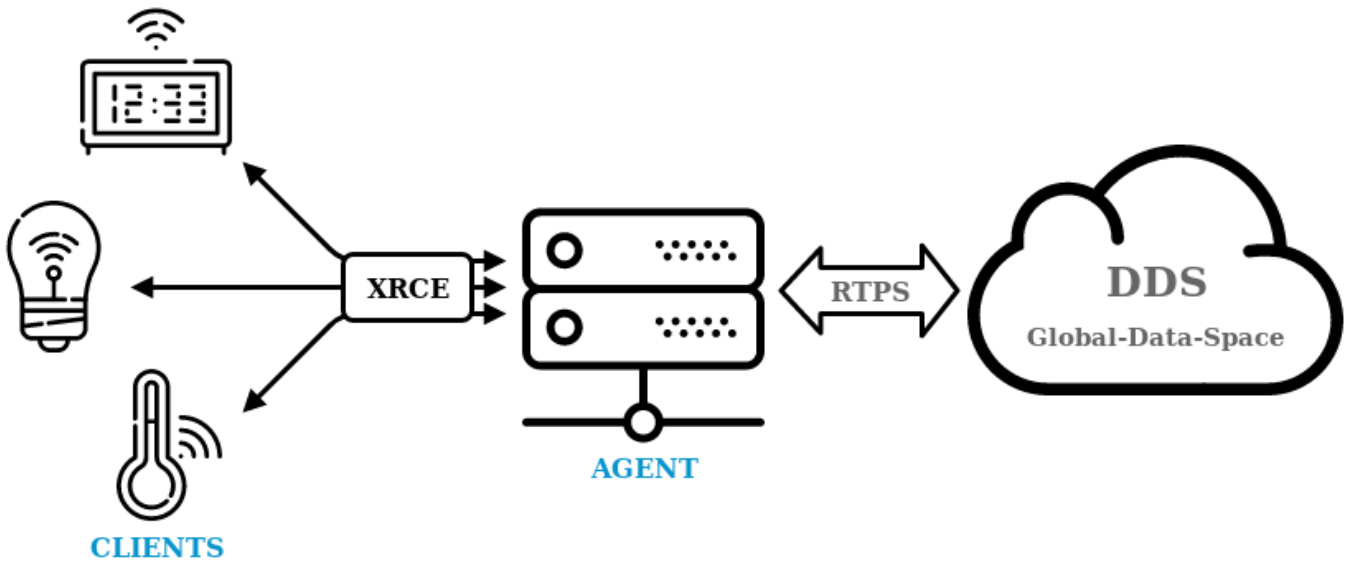
Micro-ROS
Real-Time Operating Systems (RTOSs)

- RTOSs are optimized for real-time performance
- In OFERA, we're using NuttX as the default
  - POSIX-style API makes porting easy
- Other interesting choices include RIOT, FreeRTOS, Zephyr, etc

- RTOS diversity is an issue
- Hardware diversity is an even bigger issue

- Something unifying would go a long way…

Micro-ROS
New DDS-XRCE Standard



- DDS-XRCE for e**X**tremely **R**esource **C**onstrained **E**nvironments
  … brings DDS on MCUs
- Client-server approach
  - Power-saving
  - Disconnected use

I'm assuming you heard a lot about it yesterday

Open-source at github.com/eProsima/Micro-XRCE-DDS

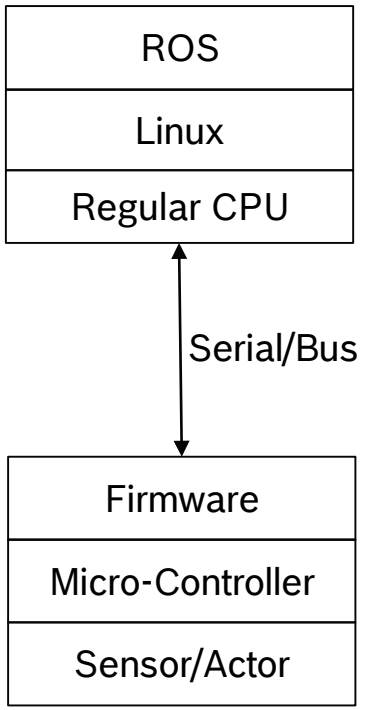
Micro-ROS
Side-by-Side Comparison

	ROS2	Micro-ROS
Hardware	X86, ARM Cortex-A, …	ARM Cortex-M, ….
Resources	>512MB RAM, >8G Disk	~100K RAM, ~1MB Flash
Communications	GBit/s, Ethernet, 802.11 WiFi	Serial, WPAN – 250k to 1MBit/s
Operating System	Linux, Windows, MacOS	RTOS (NuttX by default)
Middleware	DDS variant (by default)	XRCE-DDS (by default)
Middleware Abstraction	RMW	RMW
Client Support Library	RCL	RCL
Execution Layer	RCLCPP / RCLPY / …	**RCL** + RCLCPP
Executors	Generic	Custom

Micro-ROS
Issue 1: Hardware access

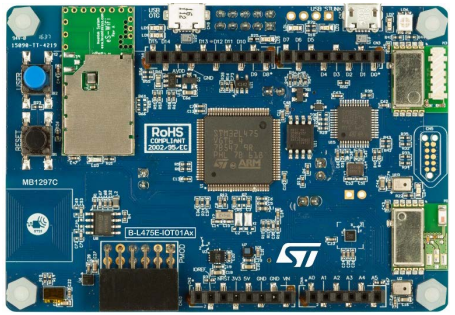
- Why not use Industrial PCs?
  - You're always talking to some piece of firmware over a comm link
  - It usually doesn't do exactly what you want
  - There's latency
- Driver implementation…
  - A multitude of serial protocols
  - Almost as bad for field buses
  - Often, important things (timing…) are not in the data-sheets
  - State management for external devices is a mess

→ We need to get into the firmware

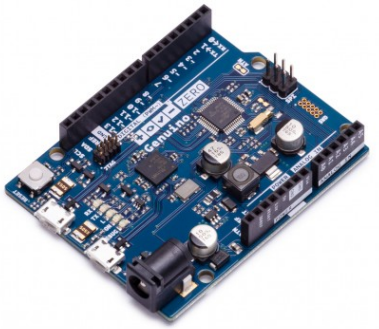


Micro-ROS
Hacker-friendly single-board computers

- Cortex-A class
  - E.g., Raspberry Pi
  - 512MB RAM, SD card storage
  - Reasonable set of I/O pins
  - WIFI, BT, Ethernet, USB
  - Linux capable



- Cortex-M4 class
  - E.g., STM32 IoT Discovery
  - ~128kB RAM, 1MB flash
  - Arduino-I/Os, PMOD
  - Low-power networks
  - Built-in sensors + I2C, SPI, etc



- Arduino-class
  - 8/32 bit MCU
  - 4-16kB RAM
  - Bare-metal
  - Arduino I/Os
  - Huge shield ecosystem

Micro-ROS
Micro-Controllers: Hardware Access

- Micro-Controller, n: Chip that contains a processor *and peripherals*
  - analog/digital converters (ADC)
  - Quadrature decoders (QED)
  - PWM generators
  - Digital IOs (GPIO)
  - ...
- Buses with register support
  - CAN, UART, SPI, I²C,...
  - Register mapping for read/write
- Much higher diversity and rate of evolution than general purpose CPUs

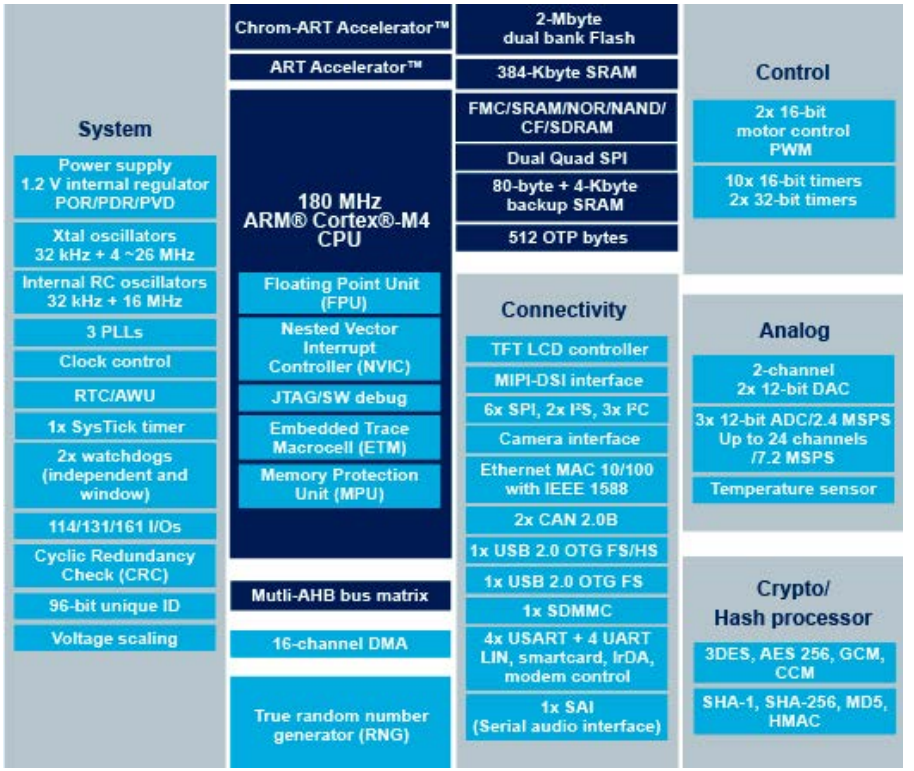
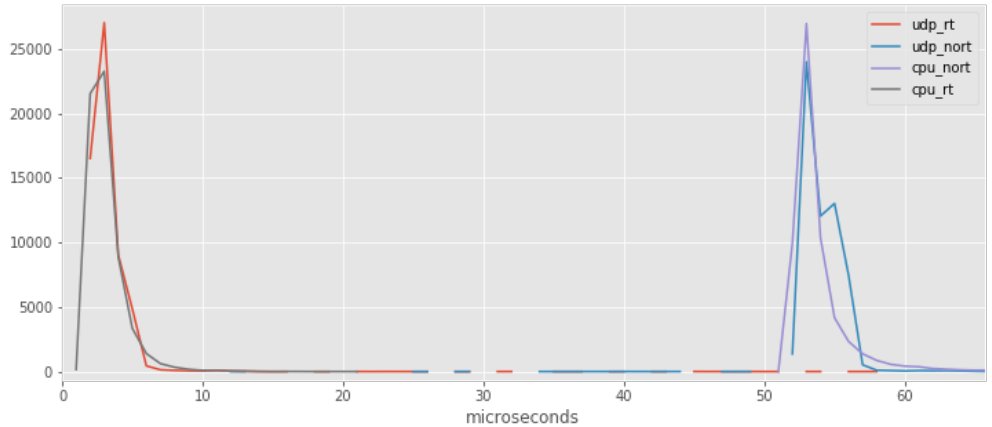


Image source: STMicro website, https://www.st.com/en/microcontrollers/stm32f479bi.html

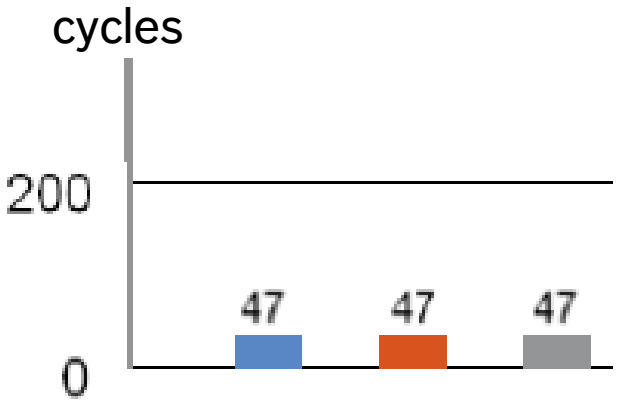
Micro-ROS
Issue 2: Status of RT on Linux

- Linux scheduler has an RT class
  - On a high-end PC, it gets you down to ~5µs task activation time
  - But kernel processes can stall it
  - Outliers up to tens of milliseconds
- Linux PREEMPT-RT Patch solves this
  - Can be difficult to integrate with other patches (e.g., BSP and proprietary drivers)
  - This is after more than a decade of work

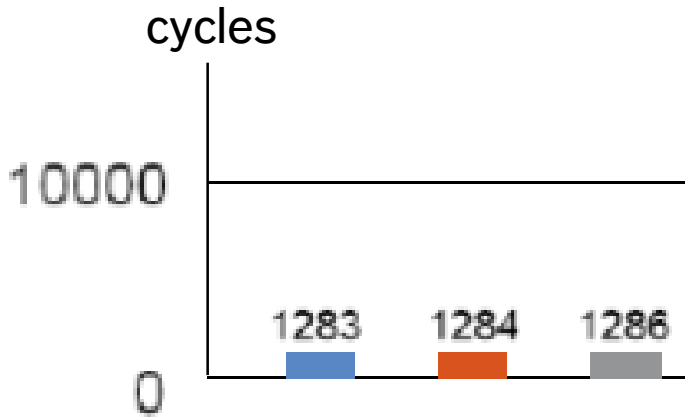


Micro-ROS

Example: Context Switch Time RTOS vs. Linux



Zephyr



Linux

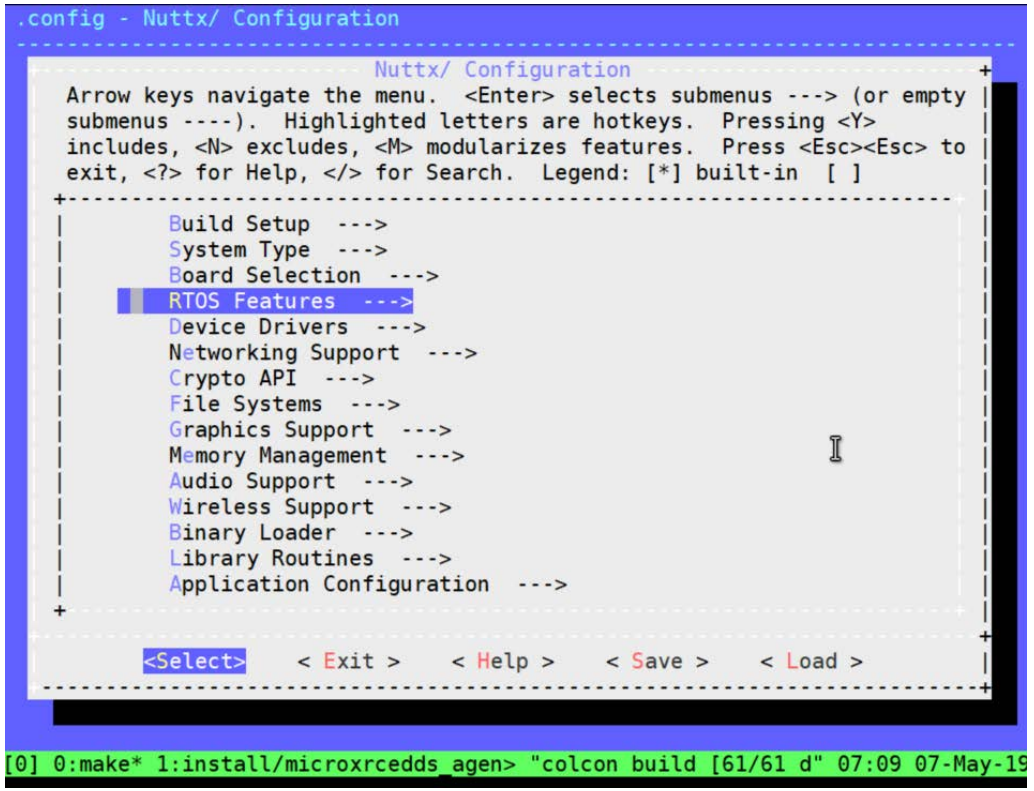
Source: “PERFORMANCE ANALYSIS USING NXP’S I.MX RT1050 CROSSOVER PROCESSOR AND THE ZEPHYR™ OS”, MAUREEN HELM, LEOTESCU FLORIN, MARIUS CRISTIAN VLAD, NXP, 2018.
https://www.nxp.com/docs/en/training-reference-material/BENCHMARK-ZEPHYR-OS-PDF.pdf

Micro-ROS
NuttX, the „Tiny Linux“

- RTOS with POSIX API
  - Released first in 2007 by Gregory Nutt
- „Batteries included“
  - Shell
  - TCP/IP
    - Incl. DHCP, NTP
  - 6LowPAN
  - C++ standard libs: libcxx or uclibc++
  - Many device drivers and BSPs
    - Focused mainly on ARM, ARC, Atmel
  - much more…
- http://nuttx.org/

RTOS	POSIX?	libstdc++?
Zephyr	Threads, Time, IPC	No
ARM Mbed	No	No
FreeRTOS	Threads, Time	No
RIOT	Partial	??

Micro-ROS
NuttX Demo



Micro-ROS
Issue 3: Power-saving

- Power use is important in many embedded applications
  - Battery-powered sensors
  - Unmanned aerial vehicles
  - Standby operation

- Linux SBC use 1-2 orders of magnitude more power

(Sources: http://www.pidramble.com/wiki/benchmarks/power-consumption, https://learn.adafruit.com/embedded-linux-board-comparison/power-usage, OFERA measurements)

Device	Idle	Operational
Rpi A	~150mA	~180mA
Rpi 3	~350mA	500-800mA
STM32L1	~3mA	~10mA
STM32F4	~10mA	~100mA

Micro-ROS
Issue 4: Safety

- Being worked on since (at least) 2011
  - SIL2Linux
  - Project P
  - …
- SIL2Linux
  - Target: Safety Integrity Level 2
    - Strips much of Linux, most notably many drivers
    - Going on for years, not clear what the outcome is
    - But do watch https://elisa.tech/ !
  - The highest SIL level is 4…
- And then there's the question of appropriate compute hardware

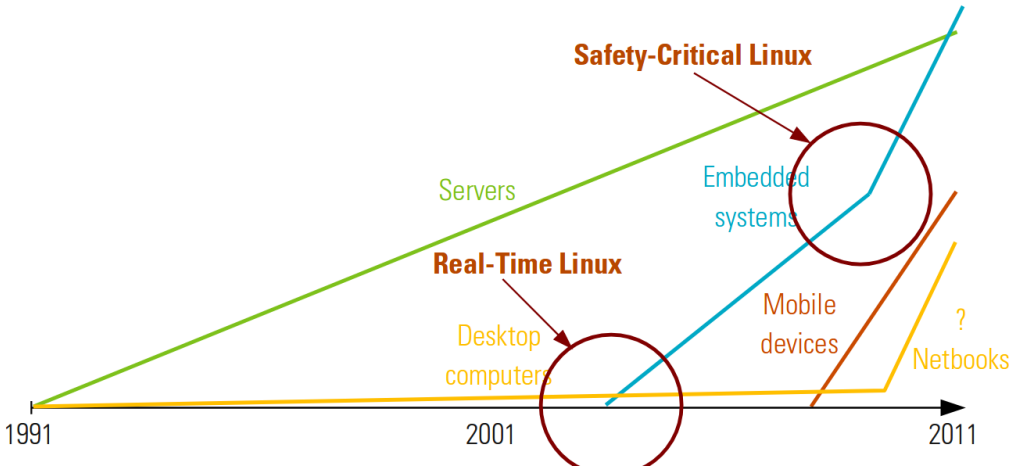
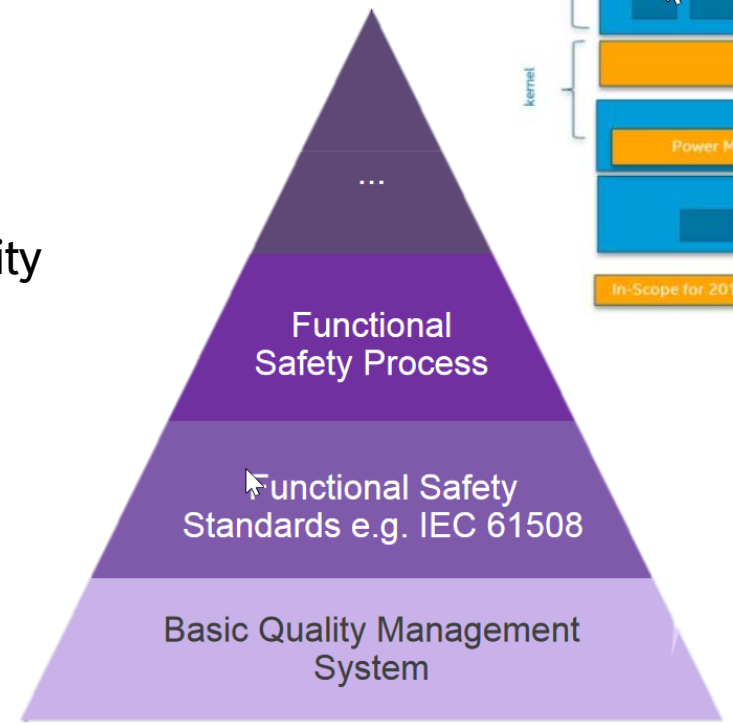


Image source: Carsten Emde, OSADL. Embedded World Presentation March 3rd 2011

Micro-ROS
Open Source and Safety

- Zephyr RTOS (a Linux Foundation project) attempts Safety Certification in 2019
- Subset of whole OS
  - Orange boxes: In scope for 2019
  - Notably no drivers!
- Based on existing work on security



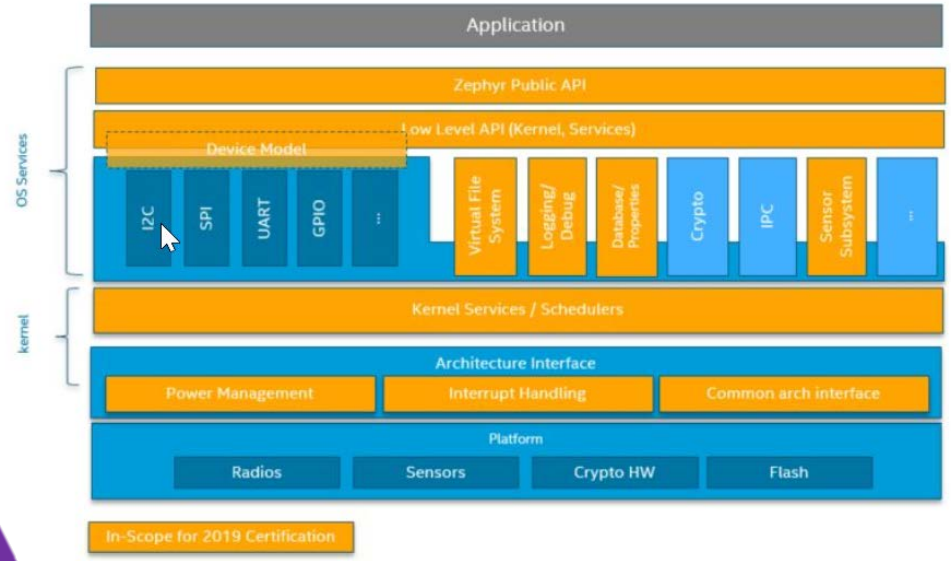
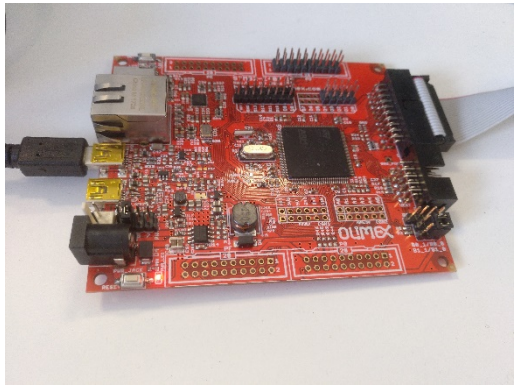


Image source: https://events.linuxfoundation.org/wp-content/uploads/2018/07/OSLS-2019_-Zephyr-Project-.pdf

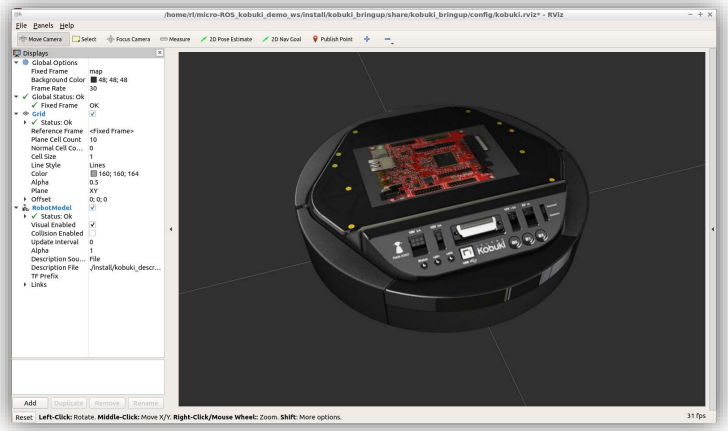
CURRENT STATUS

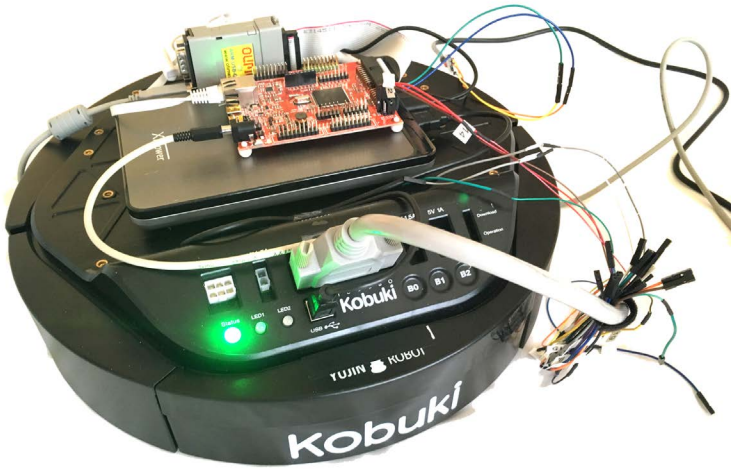
Micro-ROS
Target Devices

- Device Classes
  - Low-end: MCUs starting at 32kB RAM with low-power consumption
    - E.g., STM32L1
  - Typical: Cortex-M4 devices with ~100kB RAM
    - E.g., STM32F4
- Going below 32kB would likely require a different architectural approach and is not currently in scope
- OFERA has two references boards with full OS support provided by partner Acutronic Link Robotics
  - STM32L1-DISCOVERY
  - OLIMEX STM32E407



Micro-ROS
Turtlebot 2 Demo





ROS 2 (Crystal) running
- Visualization
- Keyboard control
- odometry to TF
- DDS <-> DDS-XRCE agent

DDS-XRCE over UDP

micro-ROS running
- thin_kobuki_driver
- DDS-XRCE client

at less than 100 KB RAM

Preliminary version at github.com/microROS/micro-ROS_kobuki_demo

Micro-ROS
Turtlebot 2 Demo

- Based on „thin kobuki“ driver
- Converted to use rcl API
  - rclcpp wasn‘t ready at the time
- Porting issues?
  - A few issues with C++ initialization

UPCOMING WORK

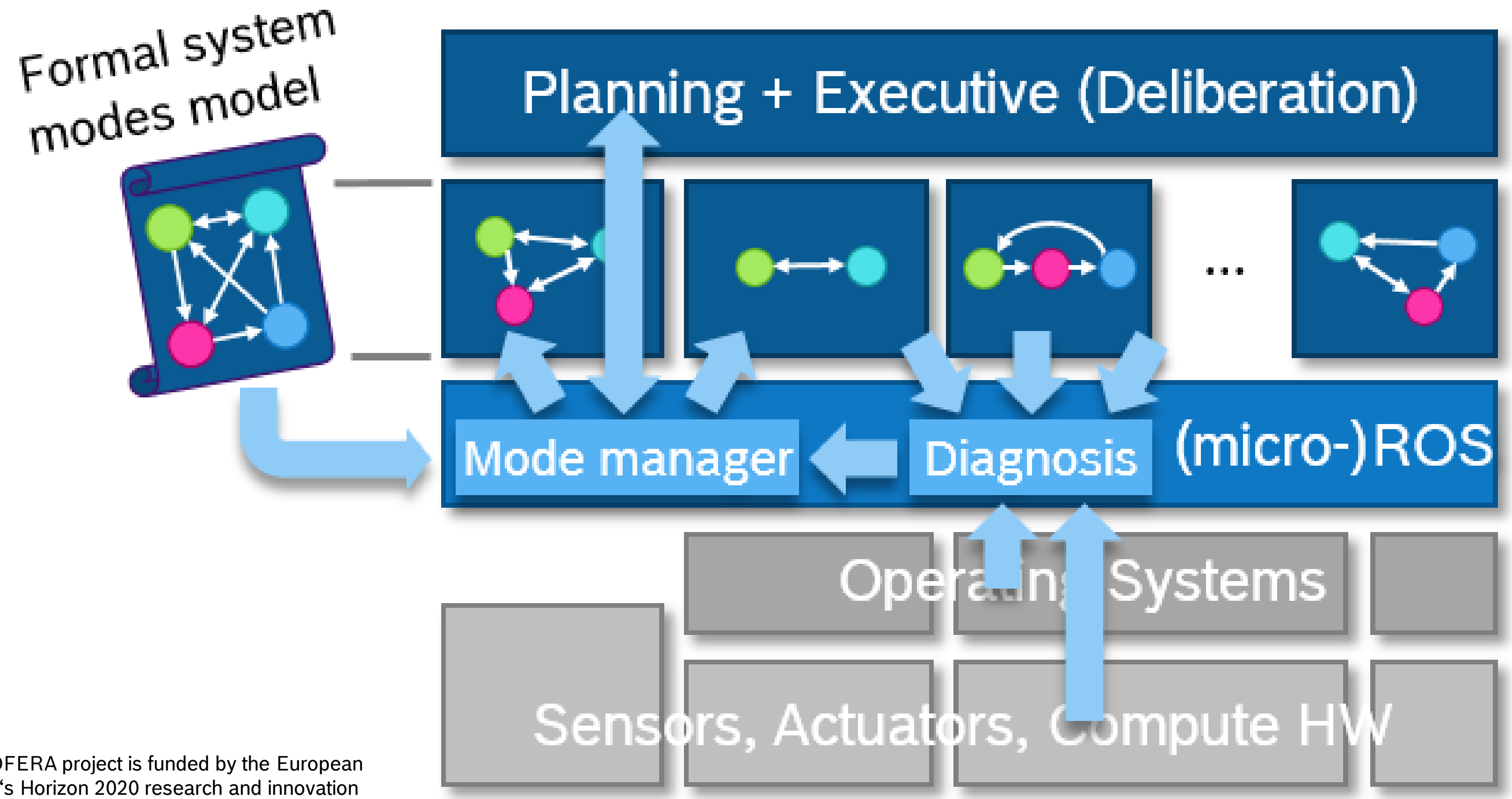
BOSCH

Micro-ROS
Recap: Composable firmware

- Nowadays, firmware provided by vendor
  - Unforeseen features? Bad luck…
- Vision: Add new features to existing firmware
  - ROS2 way: Just add nodes
- Challenge: Interference
  - Need to make sure existing stuff still works!

- Micro-ROS Approach:
  - System Modes
  - Domain-specific scheduling, towards providing guarantees

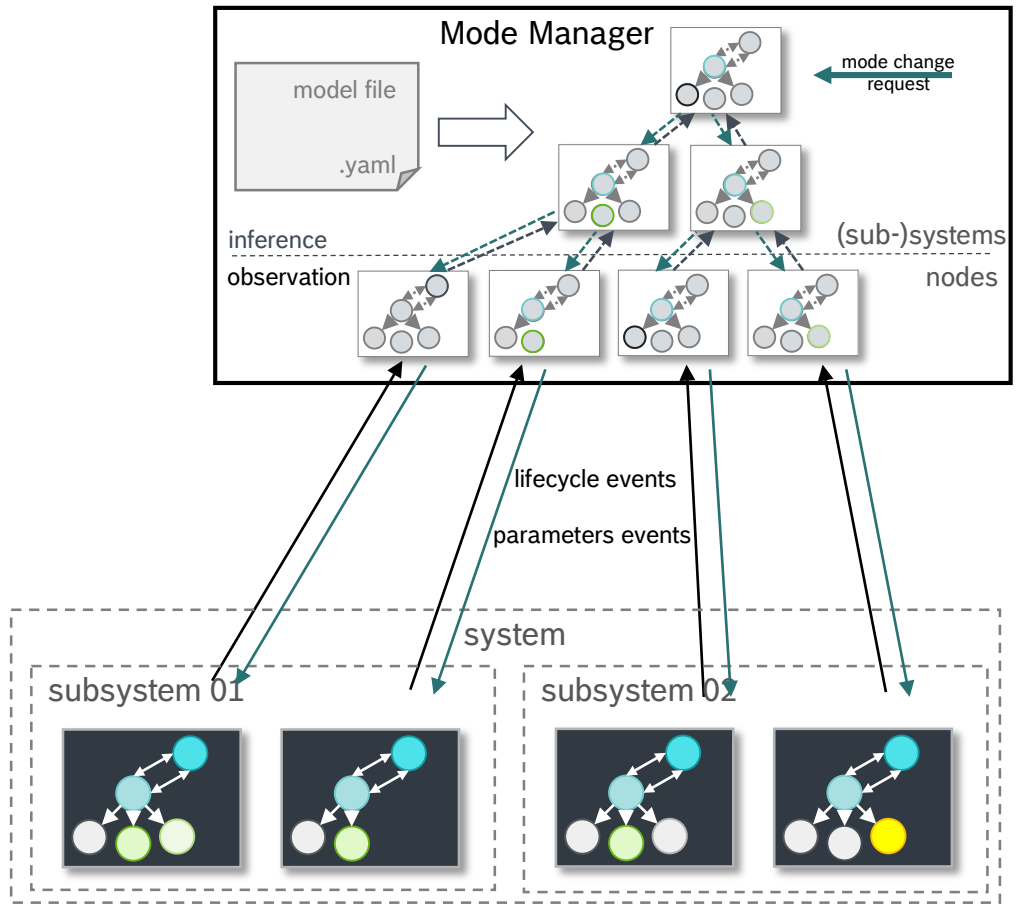
Micro-ROS
Towards explicit architecture



The OFERA project is funded by the European Union‘s Horizon 2020 research and innovation programme under grant agreement No 780785

Micro-ROS
System Modes

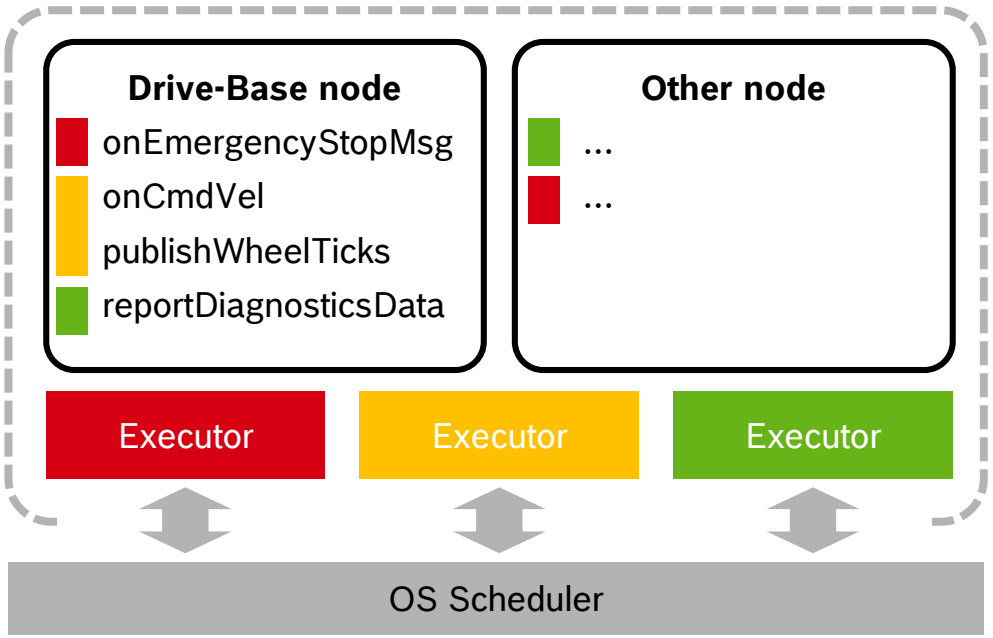
- Introduces **(sub-)systems** hierarchy to ROS 2
- Abstraction for hierarchical configuration, called **system modes**
- **Mode manager** manages consistent, system-wide configuration
- See **microros.github.io/system_modes/**



The OFERA project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780785

Micro-ROS
Predictable Execution

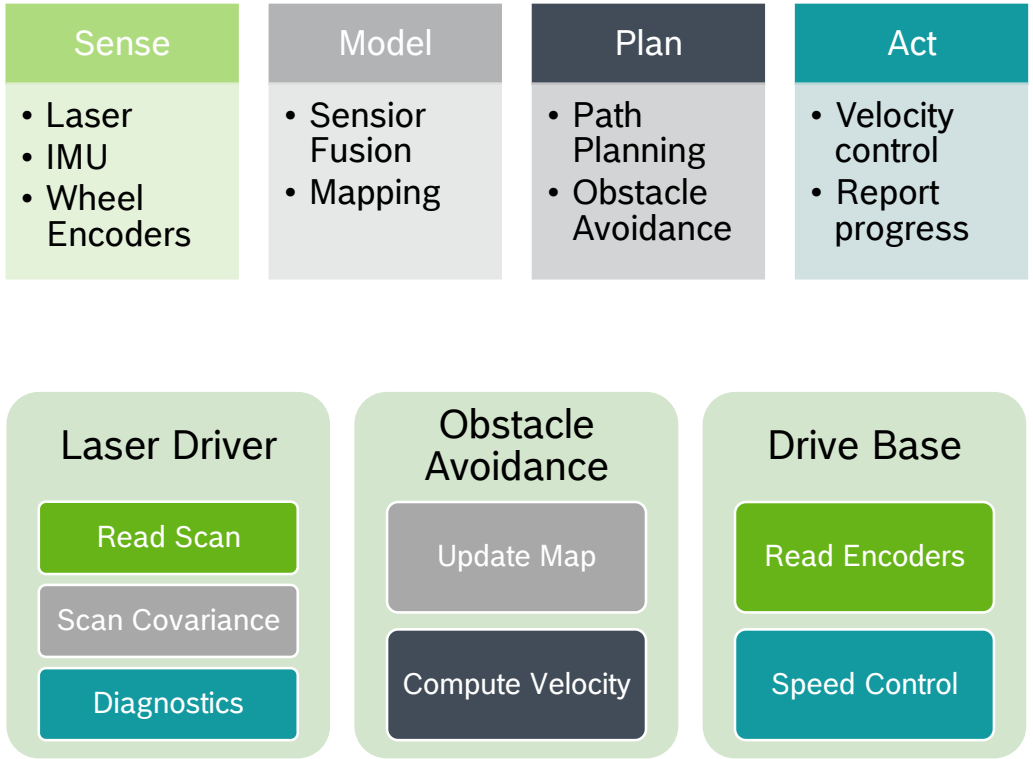
- First approach enables **multiple executors** per operating system process
- Executors can be configured individually using standard scheduling mechanisms
- Open-sourced prototype for ROS 2
- See **microros.github.io/real-time_executor/**



The OFERA project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780785

Micro-ROS
Domain-specific scheduling

- Current real-time schedulers typically use priorities
  - Not composable!
  - Not domain-appropriate
- Micro-ROS Approach: Domain-specific schedulers
  - E.g., stage-based approach with „Sense-Plan-Act“
  - Or more stages…
  - Assign callbacks to stage using callback groups
  - Derive within-group order from communication links
- Provide „budgets“ by group



BOSCH

Micro-ROS
Building micro-ROS…

- Challenge: RTOS defines toolchain and system headers
  - What‘s the build order?
- Currently: ROS2 workspace built as part of NuttX build
  - Pro: Handles toolchain (cross-compiling, etc.)
  - Con: Time consuming
  - Con: Workspace is not aware of being a micro-ROS target
- Current work: Build everything using colcon
  - NuttX vendor package to configure and build NuttX
  - Colcon configuration to configure toolchain

Micro-ROS
Building an ecosystem

- Does this mean that every ROS developer can now start using MCUs?
- Well…

ROS 2 Embedded
Further information

- microROS organization at GitHub
  - https://micro-ros.github.io/
  - https://github.com/micro-ROS/
- OFERA website: https://ofera.eu/
- ROS 2 Embedded Design Page
  - Currently at https://github.com/ros2/design/pull/197
  - After merge: http://design.ros2.org/articles/embedded.html

THANK YOU

BOSCH