ROS2 EMBEDDED

ROS-INDUSTRIAL CONFERENCE EUROPE DECEMBER 12TH 2018

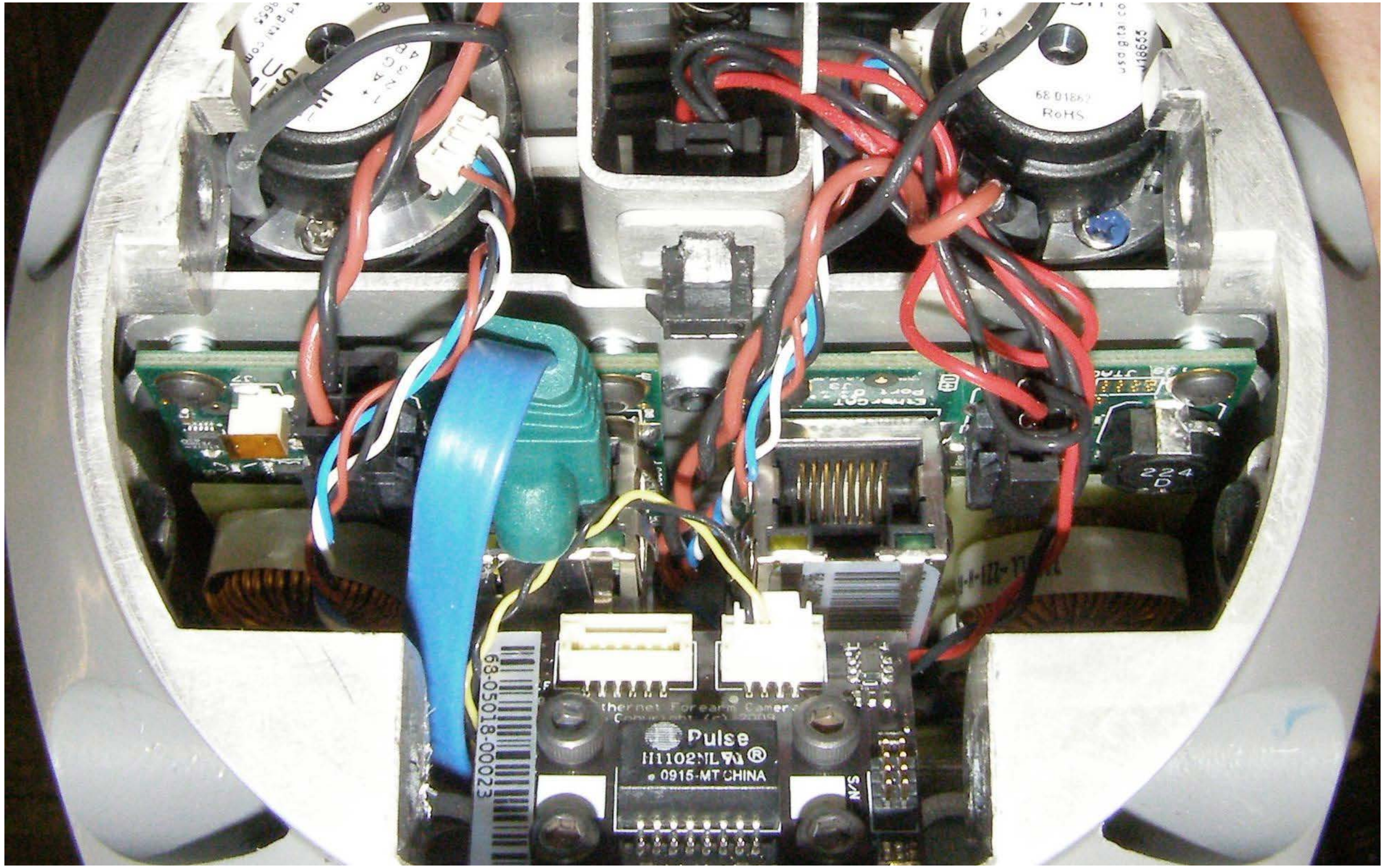
DR.-ING. INGO LÜTKEBOHLE, BOSCH CORPORATE RESEARCH

Partially supported
by EU grant
780785

BOSCH

ROS 2 Embedded

Q: What’s this? Hint: It’s from a very important ROS robot

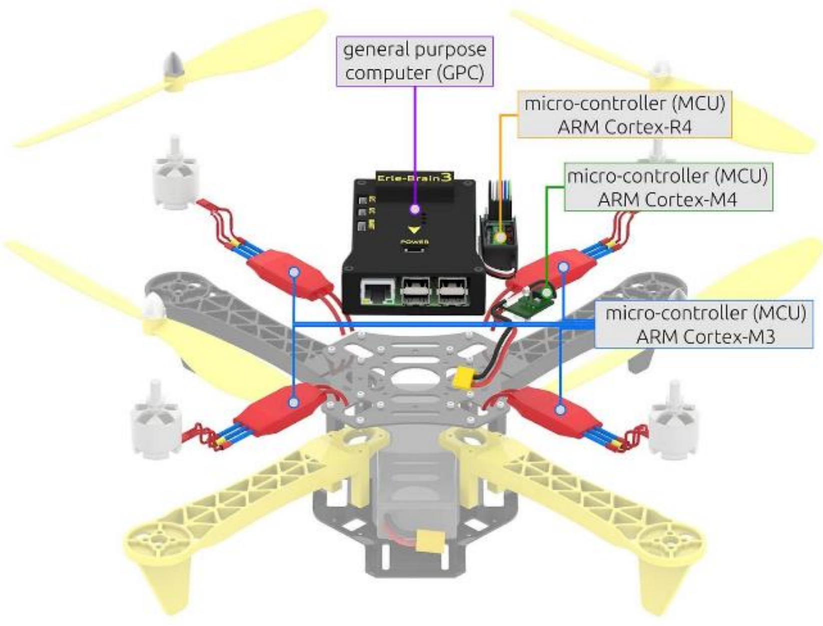


“In the future it should be possible to implement the ROS protocol directly on the devices embedded system”

ROS2 Design Wiki “Stories”

ROS 2 Embedded

Robots are networks of devices



general purpose computer (GPC)
micro-controller (MCU) ARM Cortex-R4
micro-controller (MCU) ARM Cortex-M4
micro-controller (MCU) ARM Cortex-M3
Erle-Brain3

Image source: Erle Robotics, taken from OFERA proposal.

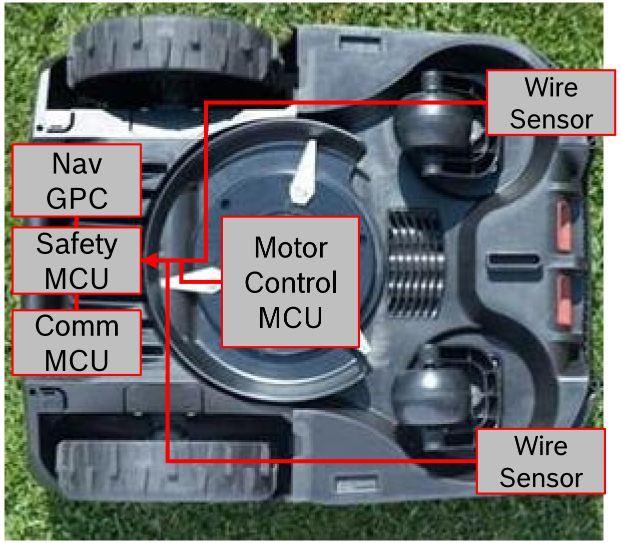
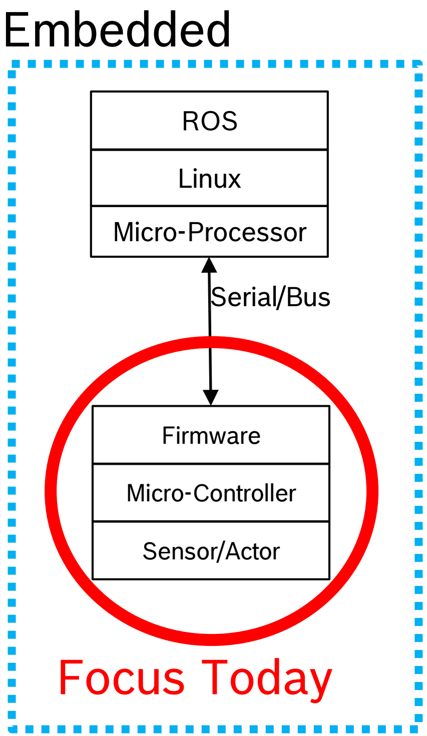


Image source: Bosch PowerTools GmbH, All rights reserved



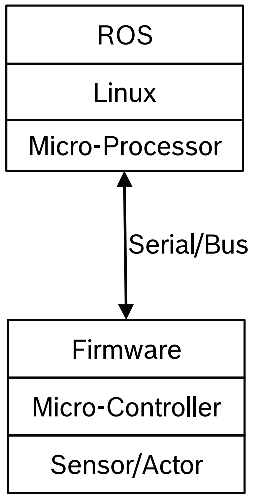
ROS 2 Embedded
Situation

- 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

ROS 2 Embedded

Issue 1: Hardware access

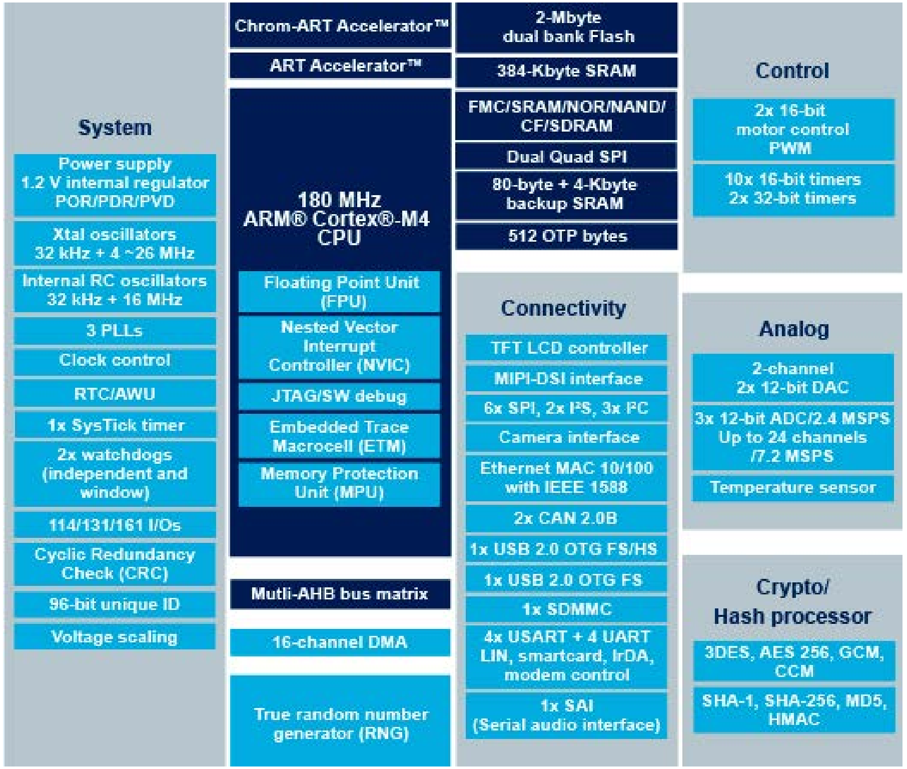
- You’re always talking to some piece of firmware over a comm link
- It usually doesn’t do exactly what you want
- There’s latency
- Driver implementation…
  - A multitude of serial protocols
  - Almost as bad for field buses
  - Lots of important things (timing…) are not in the data-sheets
  - State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



→ We need to get into the firmware

ROS 2 Embedded
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



System
Power supply 1.2 V internal regulator POR/PDR/PVD
Xtal oscillators 32 kHz + 4 ~26 MHz
Internal RC oscillators 32 kHz + 16 MHz
3 PLLs
Clock control
RTC/AWU
1x SysTick timer
2x watchdogs (independent and window)
114/131/161 I/Os
Cyclic Redundancy Check (CRC)
96-bit unique ID
Voltage scaling

Chrom-ART Accelerator™
ART Accelerator™
180 MHz ARM® Cortex®-M4 CPU
Floating Point Unit (FPU)
Nested Vector Interrupt Controller (NVIC)
JTAG/SW debug
Embedded Trace Macrocell (ETM)
Memory Protection Unit (MPU)
Mutli-AHB bus matrix
16-channel DMA
True random number generator (RNG)

2-Mbyte dual bank Flash
384-Kbyte SRAM
FMC/SRAM/NOR/NAND/ CF/SDRAM
Dual Quad SPI
80-byte + 4-Kbyte backup SRAM
512 OTP bytes

Connectivity
TFT LCD controller
MIPI-DSI interface
6x SPI, 2x I²S, 3x I²C
Camera interface
Ethernet MAC 10/100 with IEEE 1588
2x CAN 2.0B
1x USB 2.0 OTG FS/HS
1x USB 2.0 OTG FS
1x SDMMC
4x USART + 4 UART LIN, smartcard, IrDA, modem control
1x SAI (Serial audio interface)

Control
2x 16-bit motor control PWM
10x 16-bit timers 2x 32-bit timers

Analog
2-channel 2x 12-bit DAC
3x 12-bit ADC/2.4 MSPS Up to 24 channels /7.2 MSPS
Temperature sensor

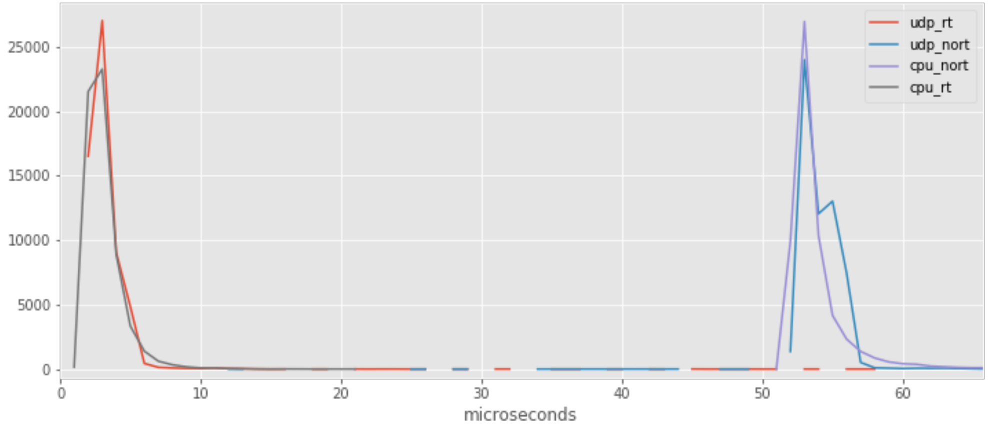
Crypto/ Hash processor
3DES, AES 256, GCM, CCM
SHA-1, SHA-256, MD5, HMAC

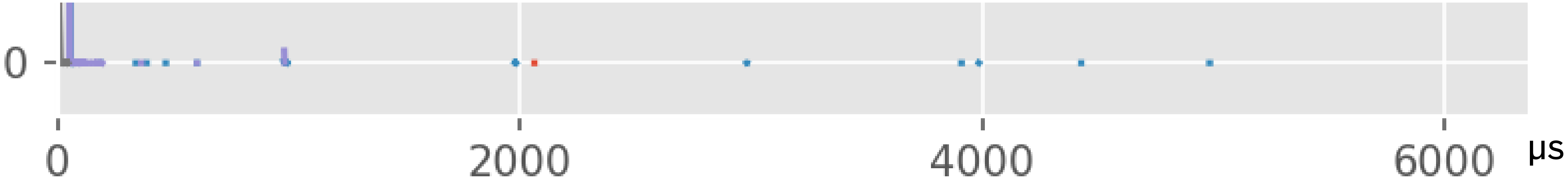
Image source: STMicro website, https://www.st.com/en/microcontrollers/stm32f479bi.html

ROS 2 Embedded

Issue 2: Status of RT on Linux

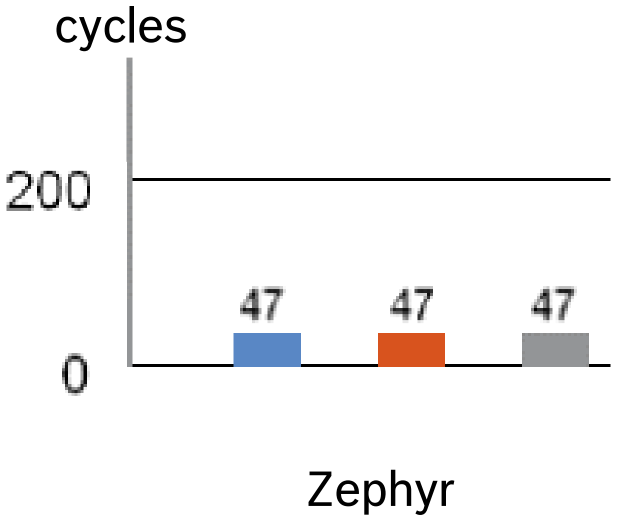
- Linux scheduler has an RT class
  - On a high-end PC, it gets you down to ~5µs task activation time
  - But kernel processes can stall it
  - Outliers up to tens of milliseconds
- Linux PREEMPT-RT Patch solves this
  - But it’s not compatible with many BSPs and proprietary drivers
- This is after more than a decade of work

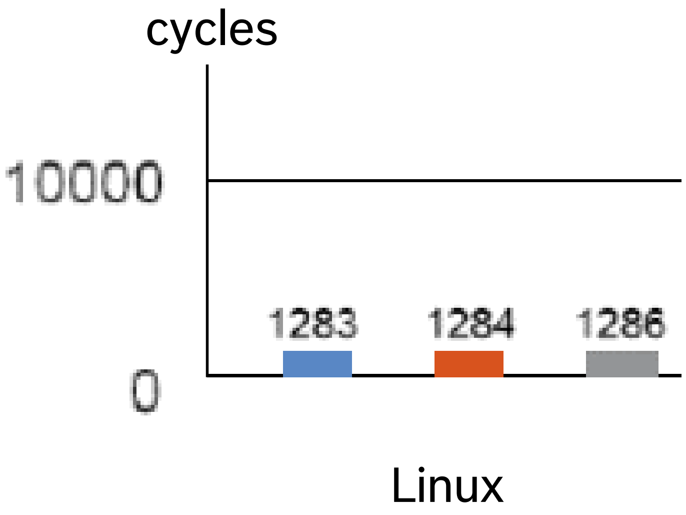




ROS 2 Embedded

Example: Context Switch Time RTOS vs. 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

ROS 2 Embedded
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

ROS 2 Embedded
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
  - The highest SIL level is 4...
- And then there's the question of appropriate compute hardware

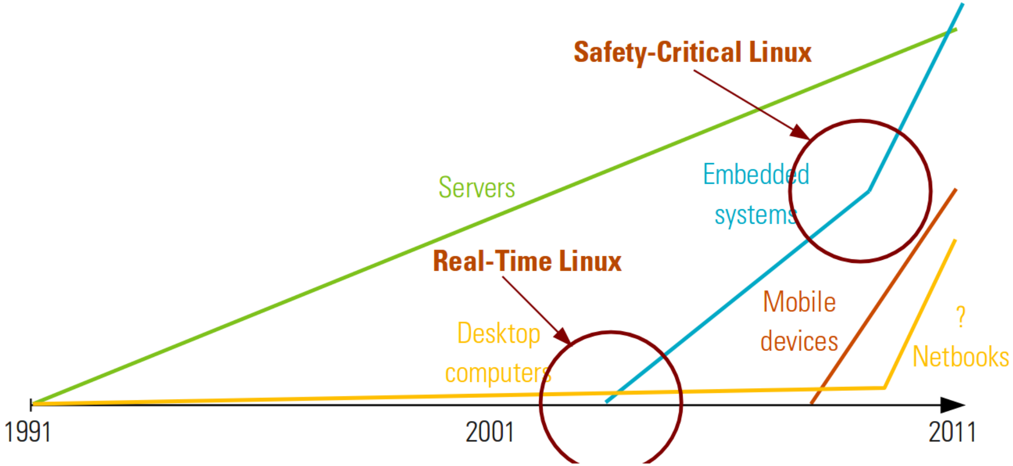


Image source: Carsten Emde, OSADL. Embedded World Presentation March 3rd 2011

ROS 2 Embedded

Introducing... Real-Time Operating Systems (RTOSs)

- RTOSs are optimized for real-time performance
- Since hard RT is a pre-requisite, there are also many safety-oriented RTOSs available
- 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...

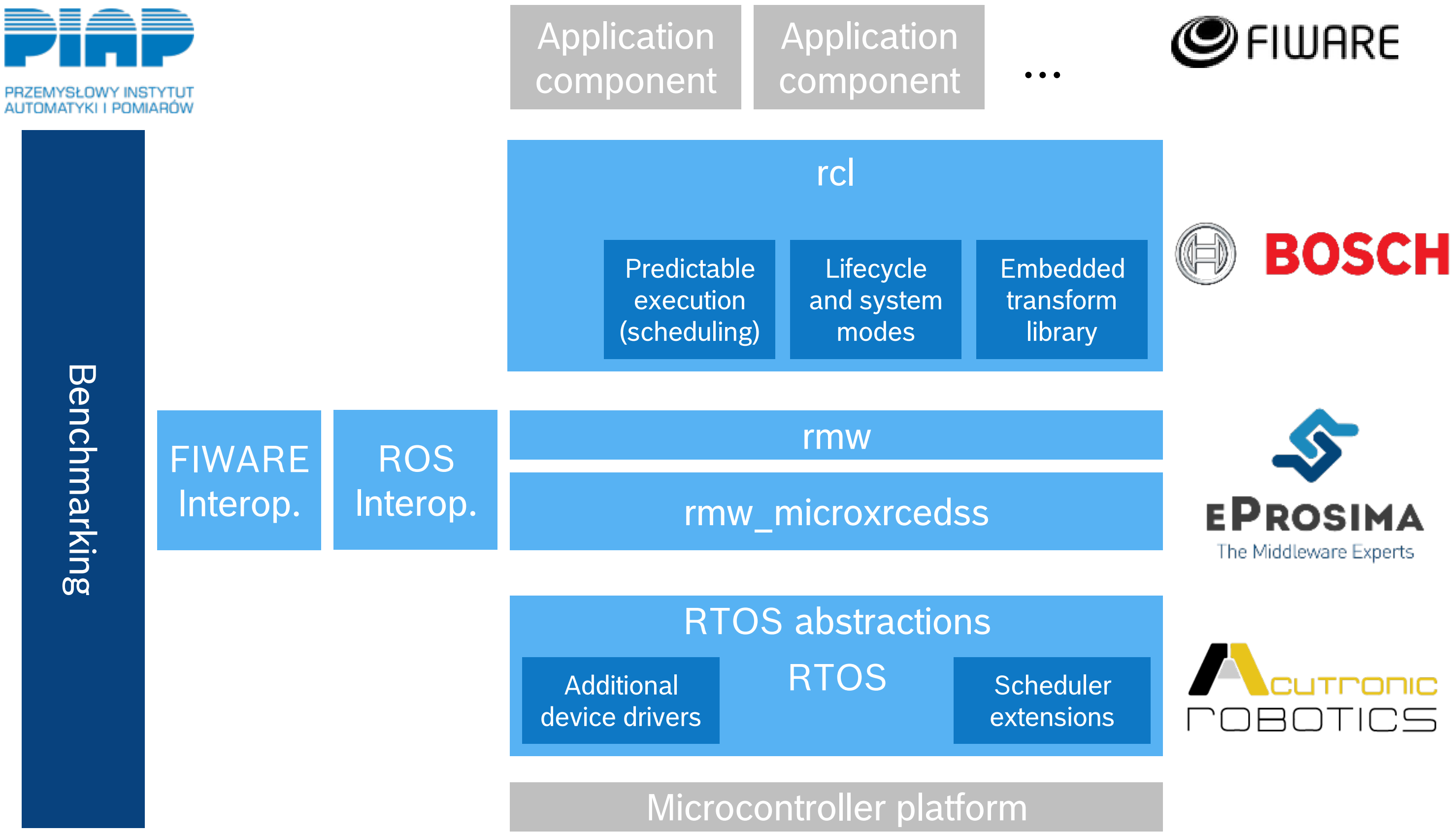
ROS 2 Embedded

Open Framework for Embedded Robot Applications (OFERA)

OFERA will extend ROS2 to allow its use in MCUs

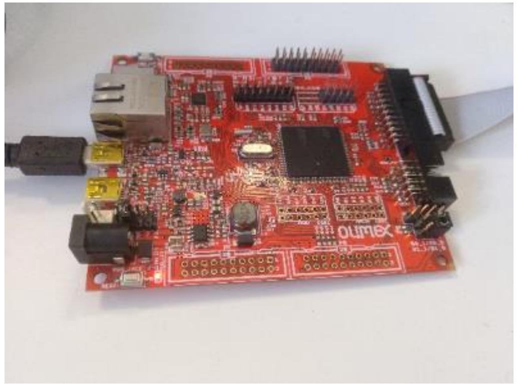
https://ofera.eu/

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



ROS 2 Embedded
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



ROS 2 Embedded
Middleware: DDS-XRCE

ePROSIMA
The Middleware Experts

- DDS is ROS 2’s default middleware
- Issues
  - DDS implementations larger than typical MCU memory
  - DDS assumes participants are always connected →problematic for battery powered devices
- DDS for eXtremely Resource Constrained Devices: DDS-XRCE
  - New OMG standard
  - Client-server approach: “Agent” keeps state for client
- Serialization format same as DDS’s
- OFERA work carried out by partner eProsima
  - rmw_microxrcedds now available https://github.com/microROS/rmw-microxrcedds

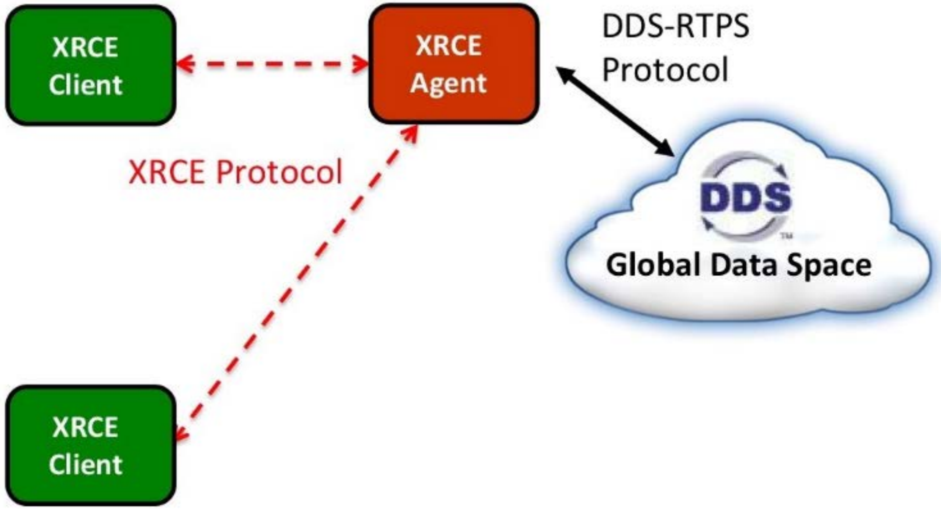


Image source: https://www.omg.org/spec/DDS-XRCE/1.0/Beta1 “XRCE Deployments”

ROS 2 Embedded
Client Library

- Current Approach
  - Use standard rmw, rcl
  - Provide specialized support for TF, scheduling, system modes
  - Provide support for time synchronization
- A rosserial-like approach has been discussed, but is *not* pursued at this time

- Approach
  - Prototype implementation provided by OFERA
  - Formation of ROS 2 Embedded Interest Group
  - ROS 2 Design PR at https://github.com/ros2/design/pull/197

ROS 2 Embedded
Further information

- microROS organization at GitHub
  - https://microros.github.io/
  - https://github.com/microROS/
- 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

OFERA → microROS

- This project is not primarily about developing new stuff
- We want to enable the community to move into into deep embedded in a sustainable way

THANK YOU