Bringing the Next Generation Robot Operating System on Deeply Embedded Autonomous Platforms

Ralph Lange
Robert Bosch GmbH
Corporate Sector Research and Advance Engineering
Renningen, Germany
ralph.lange@de.bosch.com
Bringing ROS 2 on Microcontrollers

Ralph Lange
Robert Bosch GmbH
Corporate Sector Research and Advance Engineering
Renningen, Germany
ralph.lange@de.bosch.com
Bringing ROS 2 on Microcontrollers

What is the Robot Operating System? (ros.org)
Bringing ROS 2 on Microcontrollers

The ROS Equation

Plumbing
- Process management
- Communication
- Device drivers
- Data models
- Language-independence

Tools
- Visualization
- Simulation
- Data recording
- Monitoring

Capabilities
- Control
- Perception
- Planning
- Manipulation

Ecosystem
- Shared development
- Robot models
- Documentation
- Exchange
- Market
sensor and actuator drivers
Gazebo simulator
MoveIt
nav stack
task execution
motion planning
real-time control
SLAM
gazebo simulator
real-time control
sensor and actuator drivers
Architectural Principles

- Basic entity: *Nodes* (components) which exchange messages
- Can be distributed across machines
- Standard communication patterns
  - *Topics*: Publish-Subscribe (1→n, uni-directional, async)
  - *Services*: Request-Response (n→1, bi-directional, sync+async)
  - *Actions*: Advanced Request-Response (1→1, multi-state)
- Nodes comprised of *callables* (functions), which are *data- or time-triggered*
  - Implemented in C++, Python, ...
  - Run-to-completion
- Further core functionalities: Parameters, coordinate transformation, diagnostics, ...
Bringing ROS 2 on Microcontrollers

ROS goes for series development

Key features of ROS 2

- Data Distribution Service (DDS) as middleware
- Run multiple nodes in one process
- Node lifecycle
- Deterministic launch
- ROS core functionality implemented in C
- Real-time-ready core algorithms
- Support of Windows and MacOS

- Backed by Technical Steering Committee with Intel, Amazon, Microsoft, Bosch, Arm, Apex.AI, Toyota Research, TARDEC, LG Electronics, eProsima, Acutronic Robotics, ...

Application component

Application component

... 

rclcpp

rclpy

ROS Client Support Library (rcl)
Core concepts like nodes, publisher subscription, parameters, ...

ROS Middleware Interface (rmw)
DDS Middleware
Publish/Subscribe + Request/Response
Rich QoS mechanisms

Linux / Windows / MacOS

Microprocessor
Bringing ROS 2 on Microcontrollers

ROS is great for development on microprocessors ...

... but robots are networks of computing devices!
Why microcontrollers?

**Hardware access**
- GPIOs, PWM generators, ...
- Buses such as CAN, UART, SPI I²C

**Hard, low-latency real-time**
- Context switching in less than 100 cycles

**Power saving**
- Linux single-board computer requires 10 to 100x more power than an MCU

**Safety**
- Number of safety-certified RTOS available
Our mission: Bridge the gap between MCUs and larger processors by

- Seamless integration of MCUs with ROS 2
- Ease portability of ROS 2 code to MCUs
- Ensure long-term maintenance of micro-ROS stack
Bringing ROS 2 on Microcontrollers

New DDS-XRCE Standard

- Data Distribution Service
- OMG standard since 2004
- DDS-XRCE for eXtremely Resource Constrained Environments
  ... brings DDS on MCUs

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

DDS-XRCE standard is advanced by OFERA partner eProsima

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

© Robert Bosch GmbH 2016. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Bringing ROS 2 on Microcontrollers

Architecture

Two types of APIs:

1. Plain C API based on rcl for embedded developers
2. C++ API implemented against rclcpp interfaces for typical ROS developer
Bringing ROS 2 on Microcontrollers
micro-ROS Client Library Features: System Modes

- **Task Handling**: Orchestration of the actual task, the straight-forward, error-free flow
- **Contingency Handling**: Handling of task-specific contingencies, e.g., expectable retries and failure attempts, obstacles, low battery
- **System Error Handling**: Handling of exceptions, e.g., sensor/actuator failures
Bringing ROS 2 on Microcontrollers

**micro-ROS Client Library Features: System Modes**

The OFERA project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 780785
Bringing ROS 2 on Microcontrollers

micro-ROS Client Library Features: System Modes

- Introduces (sub-)systems hierarchy to ROS 2

- Abstraction for hierarchical configuration, called system modes

- Mode manager manages consistent, system-wide configuration

- See micro-ros.github.io/system_modes/
Bringing ROS 2 on Microcontrollers

micro-ROS Client Library Features: 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 micro-ros.github.io/real-time_executor/

OS Scheduler

Drive-Base node
- onEmergencyStopMsg
- onCmdVel
- publishWheelTicks
- reportDiagnosticsData

Other node
- ...
- ...

The OFERA project is funded by the European Union’s Horizon 2020 research and innovation programme under grant agreement No 780785
Bringing ROS 2 on Microcontrollers

Community Use-Case: Kobuki with STM32 F4 Cortex-M4

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

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

Preliminary version at github.com/micro-ROS/micro-ROS_kobuki_demo
Bringing ROS 2 on Microcontrollers

Further information

https://micro-ros.github.io/

+ Join discussion in ROS 2 Embedded SIG at discourse.ros.org/c/embedded