Micro-ROS

The people

The original Micro-ROS Team

Your presenter today

Ingo Lütkebohle
Senior Expert SW Platforms
Bosch Corporate Research
ingo.luetkebohle@de.bosch.com
Micro-ROS

It’s all about robots: A small selection from the partners

Mara robot arm by Acutronic Robotics

EOD robot by PIAP

Indego lawn-mower by Bosch
Micro-ROS

ROS: Robot Operating System

„Linux for Robotics“

- **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
Micro-ROS
Architectural Principles of ROS (2)

- Basic entity: Nodes that exchange messages
  - Default protocol is TCP, UDP also possible
  - Nodelets for in-process communication
- 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
Micro-ROS
Selected Core Features

http://www.ros.org/core-components/

The Core Components images by the OSRF are licensed under CC BY 3.0
Current Status of ROS 2 - Hands-on Feature Overview

Historic Milestones

2007  Pre-cursors of ROS created at Stanford University
2008  Willow Garage, Inc takes over ROS development and starts internship program for PhD students
2010  ROS 1.0 released and eleven PR2 robots donated to universities/institutions (including Bosch)
2013  Maintenance moves to Open Source Robotics Foundation (OSRF)
      ROS Industrial Consortium founded
2014  Willow Garage shuts down, but has seven spin-offs
      ROS Industrial Consortium Europe (RIC-EU) founded
2015  OSRF starts design of ROS 2
2016  ROS Kinetic, the 10th release, is launched
2017  ROS 2 V1.0 (named Ardent Apalone) released


Colleagues with PR2 robot at the Bosch RTC in Palo Alto
Current Status of ROS 2 - Hands-on Feature Releases – and How to Get ROS

- Annual release
- Long-term releases together with Ubuntu LTS
- Debian packages at [http://packages.ros.org/ROS/ubuntu](http://packages.ros.org/ROS/ubuntu)

Installation
- Full installation: `sudo apt install ros-[distro]-desktop-full`
- Single packages: `sudo apt install ros-[distro]-[packagename]`

<table>
<thead>
<tr>
<th>Distro</th>
<th>Release</th>
<th>Poster</th>
<th>EOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Melodic Morenia</td>
<td>May 2018</td>
<td><a href="#">Poster</a></td>
<td>May 2023</td>
</tr>
<tr>
<td>Lunar Loggerhead</td>
<td>23 May 2017</td>
<td><a href="#">Poster</a></td>
<td>May 2019</td>
</tr>
<tr>
<td>Kinetic Kame</td>
<td>23 May 2016</td>
<td><a href="#">Poster</a></td>
<td>May 2021</td>
</tr>
<tr>
<td>Jade Turtle</td>
<td>23 May 2015</td>
<td><a href="#">Poster</a></td>
<td>May 2017</td>
</tr>
<tr>
<td>Indigo Igloo</td>
<td>22 Jul 2014</td>
<td><a href="#">Poster</a></td>
<td>April 2019</td>
</tr>
</tbody>
</table>

The ROS Posters by the OSRF are licensed under CC BY 3.0
“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.

Embedded

- ROS
- Linux
- Micro-Processor
- Serial/Bus

Micro-ROS Focus

- Firmware
- Micro-Controller
- Sensor/Actor
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

- Robot application development happens on Linux and Windows

- 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...
  1. Peripheral access...Hard, low-latency RT... Power saving ... Safety
  2. Open Source Firmware! → composable modules
Micro-ROS
NuttX

- OFERA has chosen NuttX as the default RTOS
  - Primary reason: POSIX-style API makes porting easy
  - Secondary reason: Linux-like in many respects

- Contenders
  - Prior work has also used the RIOT OS
  - Zephyr is interesting for safety-rated applications
  - FreeRTOS has a large userbase
Micro-ROS
New DDS-XRCE Standard

- DDS-XRCE for eXtremely Resource Constrained Environments
  ... brings DDS on MCUs
- Client-server approach
  - Power-saving
  - Disconnected use

Open-source at github.com/eProsima/Micro-XRCE-DDS
## Micro-ROS
### Side-by-Side Comparison

<table>
<thead>
<tr>
<th></th>
<th>ROS2</th>
<th>Micro-ROS</th>
<th>XRCE++</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Hardware</strong></td>
<td>X86, ARM Cortex-A, ...</td>
<td>ARM Cortex-M, ....</td>
<td></td>
</tr>
<tr>
<td><strong>Resources</strong></td>
<td>&gt;512MB RAM, &gt;8G Disk</td>
<td>~100K RAM, ~1MB Flash</td>
<td>~10k RAM</td>
</tr>
<tr>
<td><strong>Communications</strong></td>
<td>GBit/s: Ethernet, 802.11 WiFi</td>
<td>Serial, WPAN – 250k to 1MBit/s</td>
<td></td>
</tr>
<tr>
<td><strong>Operating System</strong></td>
<td>Linux, Windows, MacOS</td>
<td>RTOS (NuttX by default)</td>
<td>any</td>
</tr>
<tr>
<td><strong>Middleware</strong></td>
<td>DDS variant (by default)</td>
<td>XRCE-DDS (by default)</td>
<td>XRCE-DDS</td>
</tr>
<tr>
<td><strong>Middleware Abstraction</strong></td>
<td>RMW</td>
<td>RMW</td>
<td>-</td>
</tr>
<tr>
<td><strong>Client Support Library</strong></td>
<td>RCL</td>
<td>RCL</td>
<td>-</td>
</tr>
<tr>
<td><strong>Execution Layer</strong></td>
<td>RCLCPP / RCLPY / ...</td>
<td>RCL + RCLCPP</td>
<td>Micro-ROS</td>
</tr>
<tr>
<td><strong>Executors</strong></td>
<td>Generic</td>
<td>Micro-ROS custom</td>
<td>Micro-ROS</td>
</tr>
</tbody>
</table>
CURRENT WORK
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/](https://microros.github.io/system_modes/)
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/
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

<table>
<thead>
<tr>
<th>Sense</th>
<th>Model</th>
<th>Plan</th>
<th>Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laser</td>
<td>Sensor Fusion</td>
<td>Path Planning</td>
<td>Velocity control</td>
</tr>
<tr>
<td>IMU</td>
<td>Mapping</td>
<td>Obstacle Avoidance</td>
<td>Report progress</td>
</tr>
<tr>
<td>Wheel Encoders</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sense
• Laser
• IMU
• Wheel Encoders

Model
• Sensor Fusion
• Mapping

Plan
• Path Planning
• Obstacle Avoidance

Act
• Velocity control
• Report progress

Laser Driver
- Read Scan
- Scan Covariance
- Diagnostics

Obstacle Avoidance
- Update Map
- Compute Velocity

Drive Base
- Read Encoders
- Speed Control
Micro-ROS
NuttX Feedback: My personal journey

- I had no NuttX, or any RTOS, experience before micro-ROS
- NuttX was chosen by a partner active in the drone community
- At first I was indifferent
- Then I became annoyed with the configuration process
- ...and more annoyed
- And started to doubt whether we would ever be able to make micro-ROS user-friendly using NuttX
- ...but the alternatives were even worse
- Then I got more experience and started to like some aspects
- To the point where I started recommending it
- But I'm still unsure whether we can put this into a product
Micro-ROS
NuttX Feedback: Why it was chosen

- Good track record in the drone community (PX4)
- Developer ecosystem
  - POSIX API
  - TCP/IP stack
  - Fairly complete C library
  - C++ support
  - Promises ability to work on multiple OS’s
- Good support for STM32-based boards

- „Feels like tiny Linux“
Micro-ROS
...but the configuration...

- NuttX could do a lot of things
  - If you would only find them!
    - Thank god for „/“!
  - And if you’ve got the right board

- but this whole „upper half“/„lower half“ stuff...

→ and then it fails to compile...
Micro-ROS

Hardware support

- Many boards are supported
  - But not fully!
  - Reduces trust
- This is a general problem with RTOS’s
  - Maybe only Open Source ones?

- How about model-driven code generation?
Micro-ROS

POSIX or not POSIX? Abstraction or not?

- Sometimes, the POSIX API is inferior
- Example: timers
  - `clock_gettime` only supports real-time clock
  - Pre-scalers are not configurable
    → low resolution
  - Dedicated timer API much more capable – but not POSIX
- File-system mapping of devices feels awkward at times
  - Direct reading and writing of registers is one of the attractions of microcontrollers (for me)
  - C++ template mechanisms could make this safe
Micro-ROS

The stumbling block: Safety

- We need safety for many applications
  - Currently using proprietary RTOS’s
- Only one Open Source RTOS in this space:
  - 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

Micro-ROS
On foundations...

A SMALL, SCALABLE OPEN SOURCE RTOS FOR IOT EMBEDDED DEVICES

The Zephyr™ Project is a scalable real-time operating system (RTOS) supporting multiple hardware architectures, optimized for resource constrained devices, and built with safety and security in mind.

Find out more
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
THANK YOU!

QUESTIONS?

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