# micro-ROS – putting the open-source Robot Operating System onto microcontrollers

Dr. Ralph Lange
Bosch Research

BOSCH

ROS

The “nine dots” ROS logo is a trademark of Open Source Robotics Foundation.



Ralph Lange (CR/AD CE-RSE) | 2022-07-01



BOSCH

# *“70% of service robots use ROS in some form”*

*Fraunhofer IPA in 2020 for the World Robotics Report 2020 by the IFR (International Federation of Robotics)*


Ralph Lange (CR/AD CE-RSE) | 2022-07-01

# Goals of this lecture

- What is ROS? What is micro-ROS?
- What made ROS such a successful OSS project?
- Why do large industrial companies use ROS and contribute to it?
- What are my learnings from micro-ROS on starting a successful OSS project?

# Agenda

- Short introduction to Bosch and myself
- Introduction to ROS: Technics • History • Governance • Contributing • Success factors
- ROS@Bosch: Types of use • Our contributions • Why we contribute
- micro-ROS: Technics • Story of the project • Learnings for a successful OSS project

# Introducing Bosch

**Bosch is a supplier of technology and services in all relevant sectors for robotics and AI**

**Consumer** **Industry** **Automotive**


**Bosch Group:** 402,600 associates, 78.7 bn€ sales, 6.1 bn€ R&D*

- World's largest automotive parts supplier
- Largest manufacturer of home appliances in Europe
- World's largest supplier of MEMS sensors

**Bosch Research:** 1,800 highly specialized employees

- 11 research facilities around the globe, headquarters in the Stuttgart area
- Meet our robotics researchers: https://youtu.be/7Fm5MUaaxEY

* Business figures 2021

# Introducing Myself

**Dr. Ralph Lange**
ralph.lange@de.bosch.com
github.com/ralph-lange

- Chief expert for robotics systems and software engineering at Bosch Research
- Following and contributing to development of ROS 2
- Supporting Bosch business units on use of ROS
- Principal investigator in EU project OFERA (2018-2021), which launched micro-ROS

# Introduction to ROS

# ROS Equation

ROS = plumbing + tools + capabilities + community

**plumbing**

- process management
- communication
- device drivers
- data models
- language-independence

**tools**

- visualization
- simulation
- data recording
- monitoring

**capabilities**

- control
- perception
- planning
- manipulation

**community**

- shared development
- robot models
- documentation
- exchange
- market





Ralph Lange (CR/AD CE-RSE) | 2022-07-01

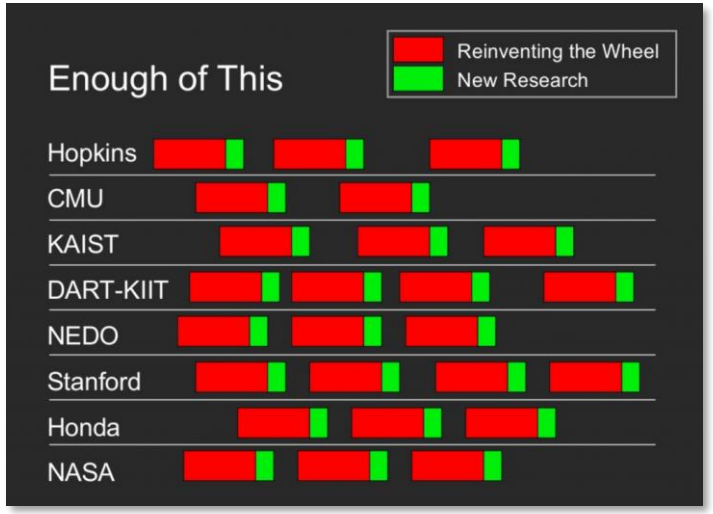


BOSCH


speech-processing
elastic-band
PDDL
obstacle-detection
context
vision
kinematics
DWA
Ackermann
calibration
UAV
transform
meshes
simulation
image-processing
IK
planning
navigation
control
RRT
teleoperation
voxel
costmap
optimization
geometry
machine-learning
answer-set-programming
multi-robot
segmentation
motion-prediction
GNSS
task-trees
locomotion
octomap
3d-reconstruction
behavior-tree
interaction
quadcopters
logic
pointcloud
grasping
dynamical-systems
collision-detection
markers
diagnostics
camera
timestamping
bundle-adjustment
state-machine
radar
SLAM
A*
TEB
CNN
filtering
tracking
modeling

# History of ROS

| | |
|---|---|
| 2007 | Inception by Keenan Wyrobek and Eric Berger at Stanford University |
| 2008 | Willow Garage, founded by Scott Hassan, takes over and starts highly popular internship program for PhD students |
| 2010 | ROS 1.0 release; eleven PR2 robots donated to universities/institutions (including Bosch) |
| 2013 | Maintenance moved to Open Source Robotics Foundation (OSRF) – now “Open Robotics” |
| | ROS Industrial Consortium founded |
| 2014 | Willow Garage shuts down, but has seven spin-offs |
| | ROS Industrial Consortium Europe (RIC-EU) founded |
| 2015 | Open Robotics starts design of ROS 2 |
| 2018 | Foundation of Technical Steering Committee (TSC) |
| 2019 | First ROS 2 LTS release: Dashing |
| 2020 | Last ROS 1 release: Noetic |
| 2022 | Third ROS 2 LTS release: Humble |




Slide from pitch deck by Keenan Wyrobek and Eric Berger

https://spectrum.ieee.org/automaton/robotics/robotics-software/the-origin-story-of-ros-the-linux-of-robotics

https://www.redhat.com/en/about/videos/how-start-robot-revolution-part-1-breaking-wheel

# Governance of ROS

Apex.AI
amazon
Microsoft
SAMSUNG
ubuntu
TOYOTA RESEARCH INSTITUTE
BOSCH Invented for life
intel
iRobot
WNDRVR

+ 3 community members

**Technical Steering Committee**

&

open robotics

**Functional Packages**

**ROS Core Stack**

**Working Groups**

**Community**

incl. ROS industrial

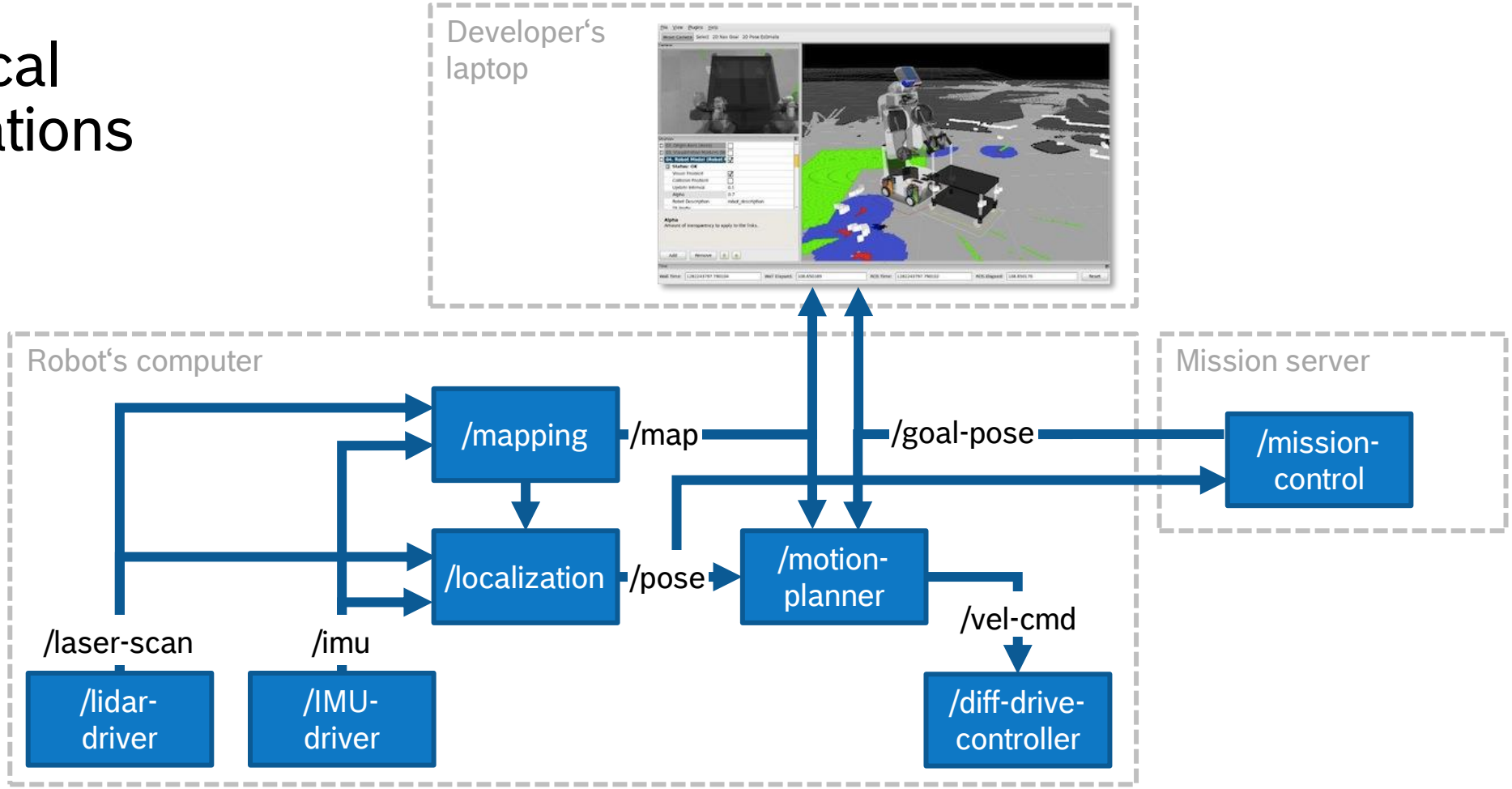
All company logos are the property of their respective owners.

Photo from ROSCon 2018 used with permission from Open Robotics.

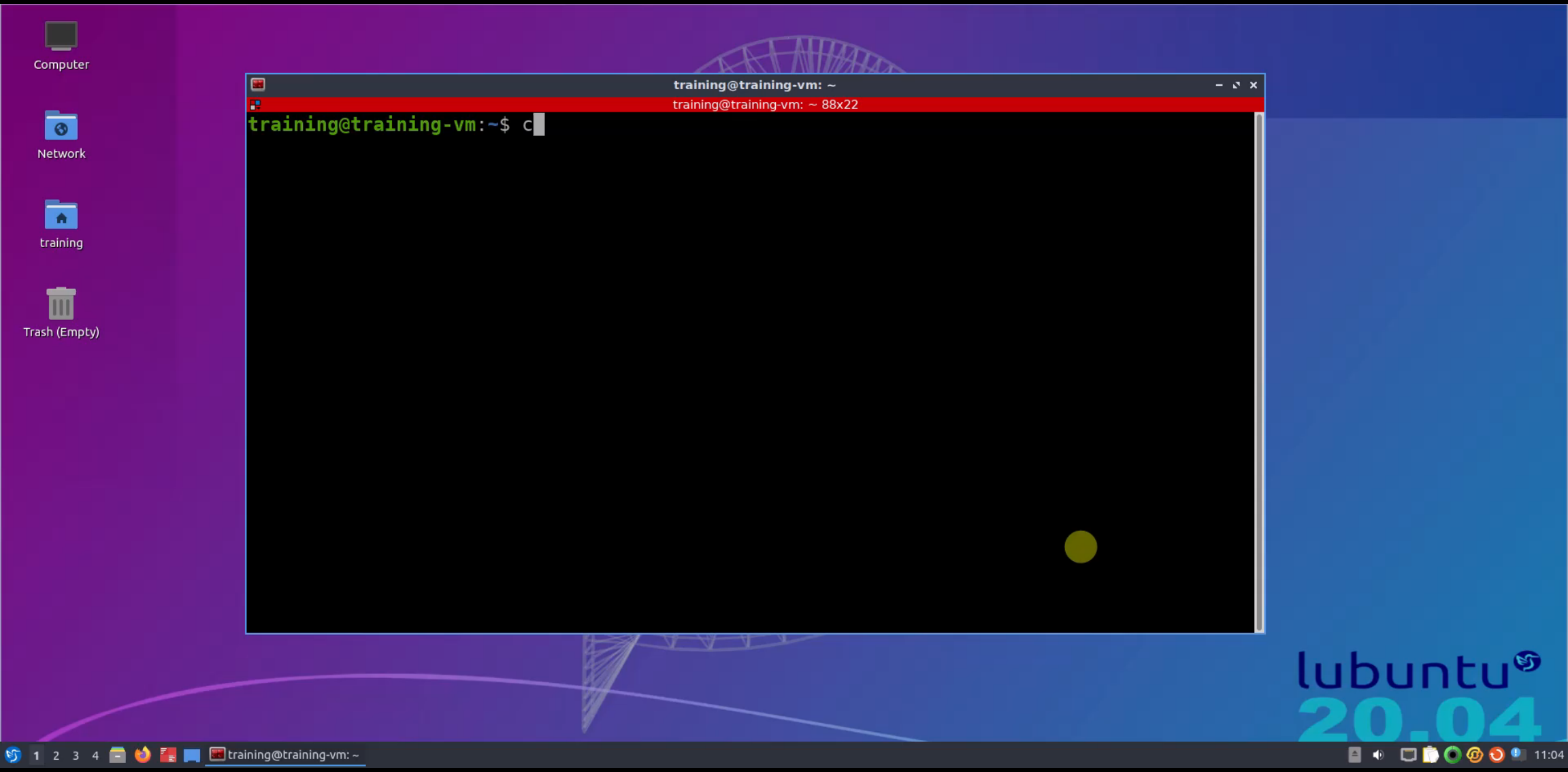

Ralph Lange (CR/AD CE-RSE) | 2022-07-01


BOSCH

# Technical Foundations




- Components: **nodes** with data- or time-triggered **callbacks**
- Communication: **topics**, **services**, **actions**


Computer
Network
training
Trash (Empty)
training@training-vm: ~
training@training-vm: ~ 88x22
training@training-vm:~$ c
lubuntu
20.04
1 2 3 4
training@training-vm: ~
11:04

Ubuntu (Debian) — ROS 2 Docu

https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html

ROS 2 Documentation: Humble

Search docs


- Installation
  - Ubuntu (Debian)
  - Windows (binary)
  - RHEL (RPM)
  - Alternatives
  - Maintain source checkout
  - Testing with pre-release binaries
  - DDS implementations
- Distributions
- Tutorials
- How-to Guides
- Concepts
- Contact
- The ROS 2 Project
- Related Projects
- Glossary
- Citations

Other Versions v: humble


Desktop Install (Recommended): ROS, RViz, demos, tutorials.

```
sudo apt install ros-humble-desktop
```

ROS-Base Install (Bare Bones): Communication libraries, message packages, command line tools. No GUI tools.

```
sudo apt install ros-humble-ros-base
```

## Environment setup

### Sourcing the setup script

Set up your environment by sourcing the following file.

```
source /opt/ros/humble/setup.bash
```

## Try some examples

### Talker-listener

If you installed `ros-humble-desktop` above you can try some examples.

In one terminal, source the setup file and then run a C++ `talker`:

```
source /opt/ros/humble/setup.bash
ros2 run demo_nodes_cpp talker
```

In another terminal source the setup file and then run a Python `listener`:

```
source /opt/ros/humble/setup.bash
ros2 run demo_nodes_py listener
```

**docs.ros.org → Humble → Installation**

Executors — ROS 2 Documenta

https://docs.ros.org/en/humble/Concepts/About-Executors.html

Suchen

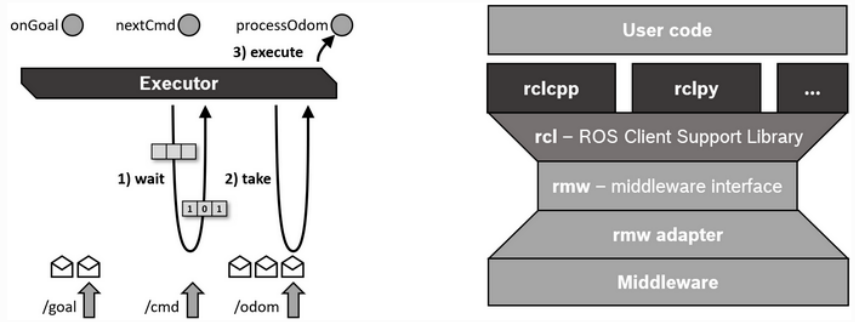
Search docs


Installation
Distributions
Tutorials
How-to Guides
Concepts
The ROS_DOMAIN_ID
About different ROS 2 DDS/RTPS vendors
About logging and logger configuration
About Quality of Service settings
About ROS 2 client libraries
About ROS 2 interfaces
About parameters in ROS 2
Executors
About topic statistics
Introspection with command line tools
Overview and usage of RQt
About Composition
On the mixing of ament and catkin (catment)
Other Versions v: humble


The call to `spin(node)` basically expands to an instatiation and invokation of the Single-Threaded Executor, which is the simplest Executor:

```
rclcpp::executors::SingleThreadedExecutor executor;
executor.add_node(node);
executor.spin();
```

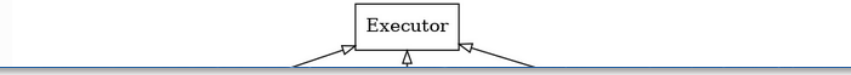
By invoking `spin()` of the Executor instance, the current thread starts querying the rcl and middleware layers for incoming messages and other events and calls the corresponding callback functions until the node shuts down. In order not to counteract the QoS settings of the middleware, an incoming message is not stored in a queue on the Client Library layer but kept in the middleware until it is taken for processing by a callback function. (This is a crucial difference to ROS 1.) A *wait set* is used to inform the Executor about available messages on the middleware layer, with one binary flag per queue.



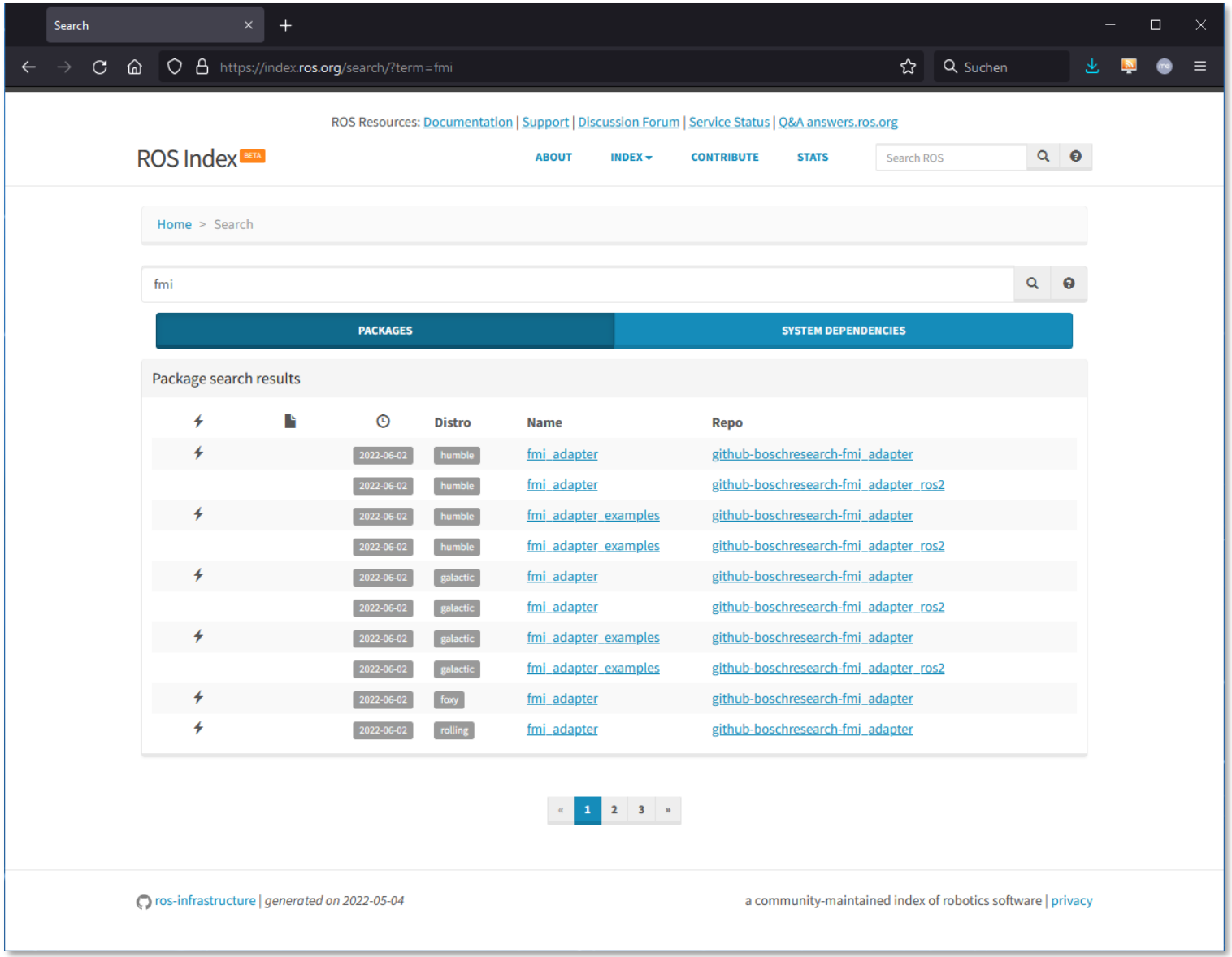

The Single-Threaded Executor is also used by the container process for components, i.e. in all cases where nodes are created and executed without an explicit main function.

## Types of Executors

Currently, rclcpp provides three Executor types, derived from a shared parent class:




**docs.ros.org → Humble → Tutorials, How-to-Guides, Concepts, ...**




**index.ros.org (here: search for '*fmi*')**

ROS Index

https://index.ros.org/p/fmi_adapter/github-boschresearch-fmi_adapter_ros2/#humble

| | |
|---|---|
| Last Updated | 2022-06-02 |
| Dev Status | DEVELOPED |
| CI status | No Continuous Integration |
| Released | UNRELEASED |
| Tags | *No category tags.* |
| Contributing | Help Wanted (0) |
| | Good First Issues (0) |
| | Pull Requests to Review (0) |

fmi_adapter/README.md

General information about this repository, including legal information, build instructions and known issues/limitations, are given in README.md in the repository root.

# The fmi_adapter package

fmi_adapter is a small ROS 2 package for wrapping *functional mockup units (FMUs)* for co-simulation of physical models into ROS nodes. FMUs are defined in the FMI standard. Currently, this package supports co-simulation FMUs according to the FMI 2.0 standard only.

FMUs can be created with a variety of modeling and simulation tools. Examples are Dymola, MATLAB/Simulink, OpenModelica, SimulationX, and Wolfram System Modeler.

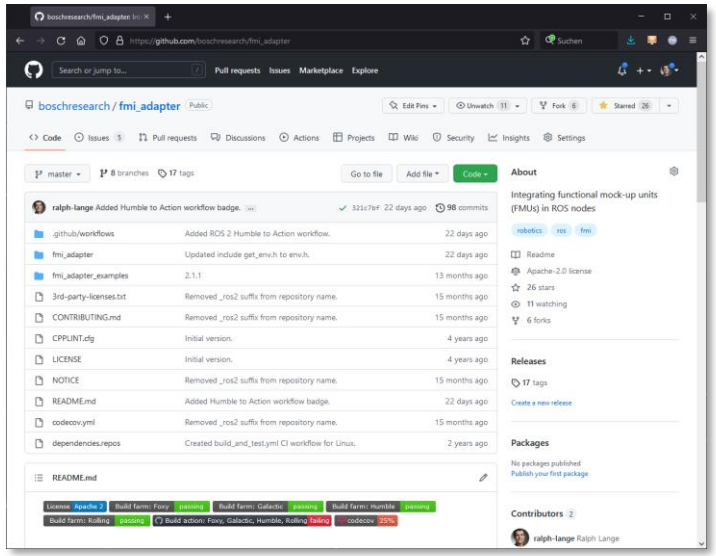
Technically, a co-simulation FMU is a zip file (with suffix .fmu) containing a physical model and the corresponding solver as a shared library together with an XML file describing the inputs, outputs and parameters of the model and details of the solver configuration. An addition, the zip file may contain the source code of the model and solver in the C programming language.

## fmi_adapter_node

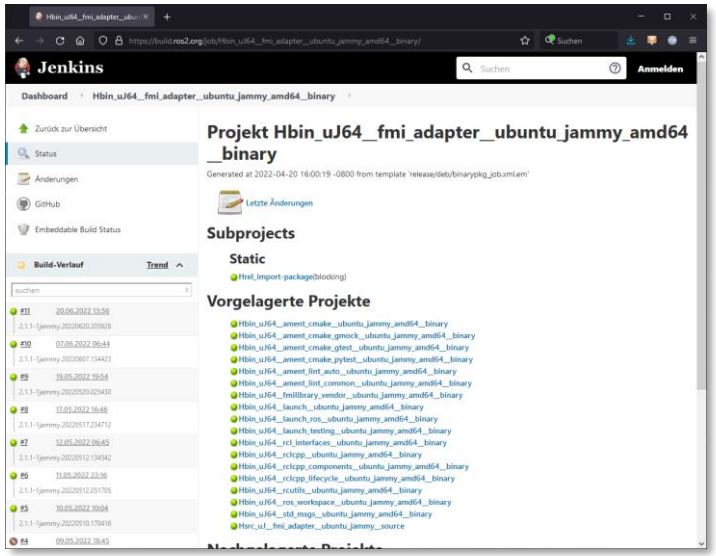
fmi_adapter provides a ROS node *fmi_adapter_node* (class FMIAdapterNode derived from LifecycleNode), which takes an FMU and creates subscribers and publishers for the input and output variables of the FMU, respectively. Then, it runs the FMU's solver with a user-definable update period. This approach is illustrated in the following diagram.


**index.ros.org (here: view of fmi_adapter package)**

# Release Your Own Package




```
      version: ros2
    status: developed
  fmi_adapter:
    doc:
      type: git
      url: https://github.com/boschresearch/fmi_adapter.git
      version: humble
    release:
      packages:
      - fmi_adapter
      - fmi_adapter_examples
      tags:
        release: release/humble/{package}/{version}
      url: https://github.com/ros2-gbp/fmi_adapter-release.git
      version: 2.1.1-1
    source:
      type: git
      url: https://github.com/boschresearch/fmi_adapter.git
      version: humble
    status: maintained
  fmilibrary_vendor:
    release:
```




1. **Publish code at GitHub, GitLab or similar**
2. **Bloom release tool opens PR at github.com/ros/rosdistro**
3. **Package is built on Jenkins at build.ros2.org**

# Message Types

```
geometry_msgs/PoseStamped
  std_msgs/Header header
  geometry_msgs/Pose pose
```

expands to →

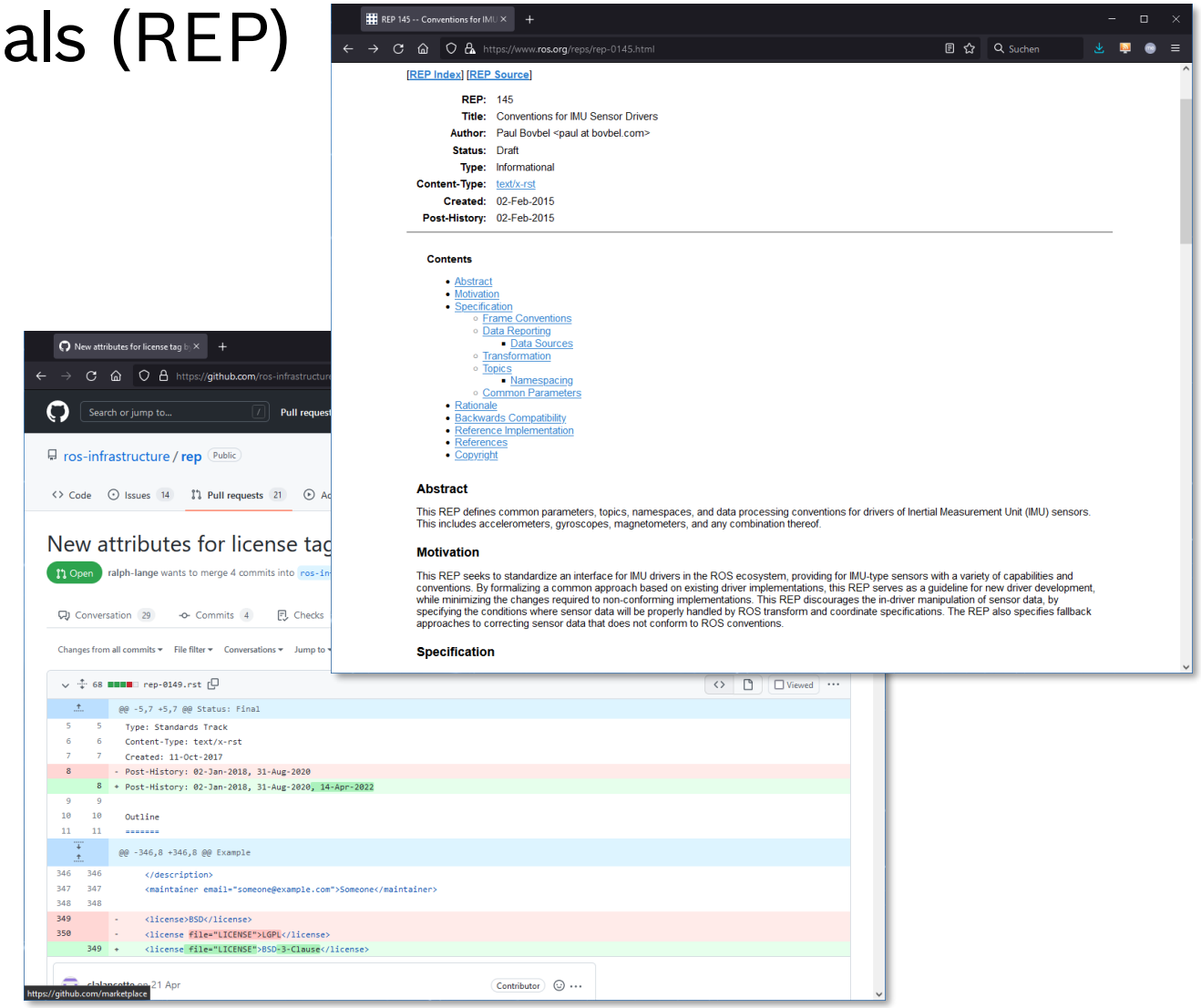
```
geometry_msgs/PoseStamped
  std_msgs/Header header
    time stamp
    string frame_id
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
```

**Dozens of built-in message types – form a de-facto standard:**

- **sensor_msgs:** BatteryState, CameraInfo, Image, Imu, PointCloud, LaserScan, Temperature, ...
- **geometry_msgs:** Accel, Inertia, Point, Polygon, Pose, Quaternion, Transform, ...
- **nav_msgs:** MoveBaseAction, Path, Odometry, GridCells, OccupancyGrid, ...

# ROS Enhancement Proposals (REP)

- ROS-specific standards developed by the community
- Discussed publicly in PRs at github.com/ros-infrastructure/rep



# Conclusions 1: What made ROS so successful?

- Many developers – Willow Garage and Open Robotics actively built a huge community
  - PhD internship program and PR2 program, Industrial TSC, Community TSC members
  - Broad scope: hobbyists to professional developers; ROS is standard in many robotics courses at universities
- Low barriers for contributions
  - Strict quality processes for contributions to core packages, but no barriers for own new packages
- Documentation and tutorials
  - Comprehensive Wiki with tons of tutorials and example, in particular for ROS 1
- Permissive licenses and trademark rules
  - Eases use in industry and formation of a lively startup scene around ROS
- The human factor
  - National and global developer conferences, meet-ups, industrial consortia, …

# ROS@Bosch

# Classes of Use of ROS

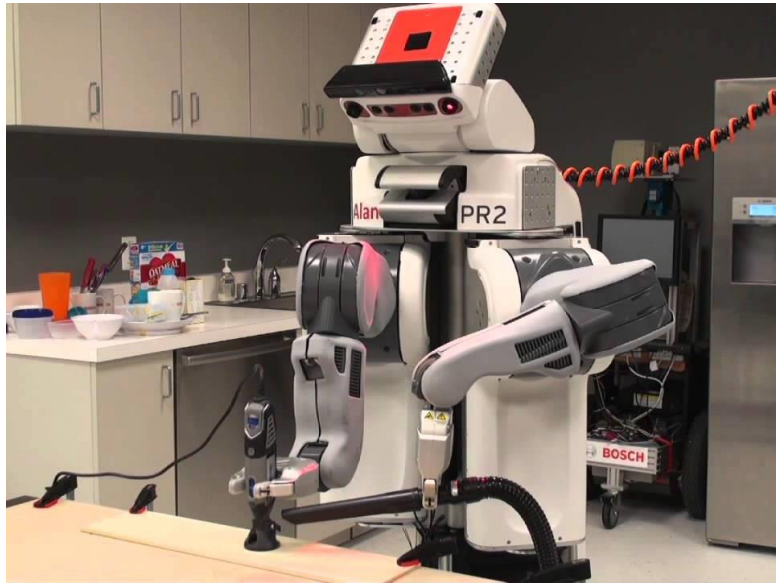
Research and advance development

Development tool and environment

Middleware and framework in product software

# ROS in Research and Advance Development

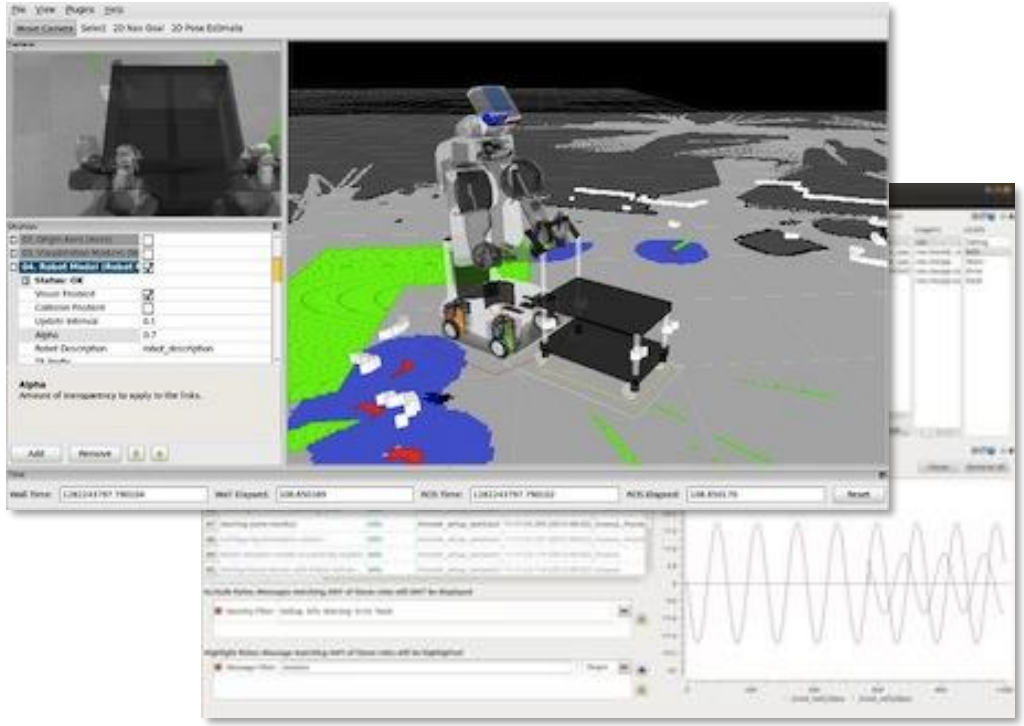
- Bosch has long history with ROS
  - … as the only industrial participant in PR2 beta program
- First use in service robotics research
- Active use in automated driving research




**ROS software ecosystem and tools accelerate prototyping significantly**

# ROS as Development Tool and Environment

- ROS comes with several tools for data recording and visualization
  - Open-source and commercial solutions for large-scale data storage and management
- De-facto standardized message types ensure long-term compatibility
- ROS used in various projects for system testing
  - … including control of test setup and measurement system

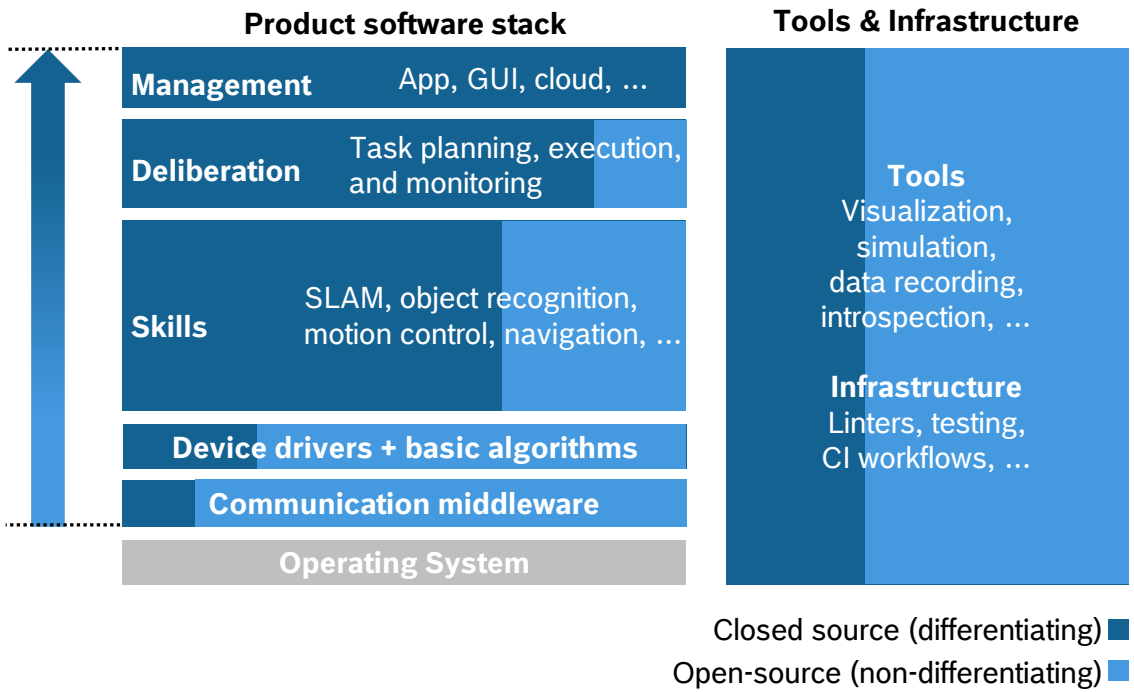




**ROS tools facilitate system testing and data analysis**

# ROS as Middleware and Framework in Product Software

- ROS 2 has paved way for professional use
  - Proven middleware standard DDS
  - Quality assurance, cf. REP-2004
  - Application-level stacks for robot navigation and manipulation available
- Great technical foundation for internal software reuse
- Of course, there are also shady sides
  - Transition to ROS 2 is slow, industry goes ahead
  - Open issues in execution management
  - Pure open-source ROS stack is not safety certified




**ROS increases development efficiency and fosters SW reuse**


Ralph Lange (CR/AD CE-RSE) | 2022-07-01

# Contributions by Bosch to ROS

Bosch participates in PR2 beta program

Founding member of ROS-I Europe

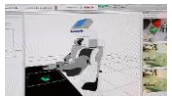

Bosch sponsors development of ROS 2

Founding member of ROS TSC

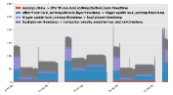
EU project micro-ROS

EU ITP MROS

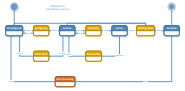
**2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021**


Tools and basic algorithms from PR2 beta program

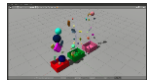


Tracepoints



Node lifecycle

rosbag tool



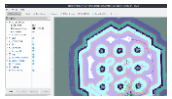
pcg_gazebo

Client library for C

ros2_control

System Modes

UUV simulator



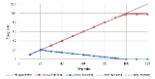
rviz2

fmi FUNCTIONAL MOCK-UP INTERFACE

fmi_adapter

iceoryx

Middleware adapter



Refined Executor

**+ dozens of talks, tutorials, and exhibitions**

**+ regular sponsoring of developer conferences**

## Give-and-take basis • Helped to shape the core architecture



Ralph Lange (CR/AD CE-RSE) | 2022-07-01



BOSCH

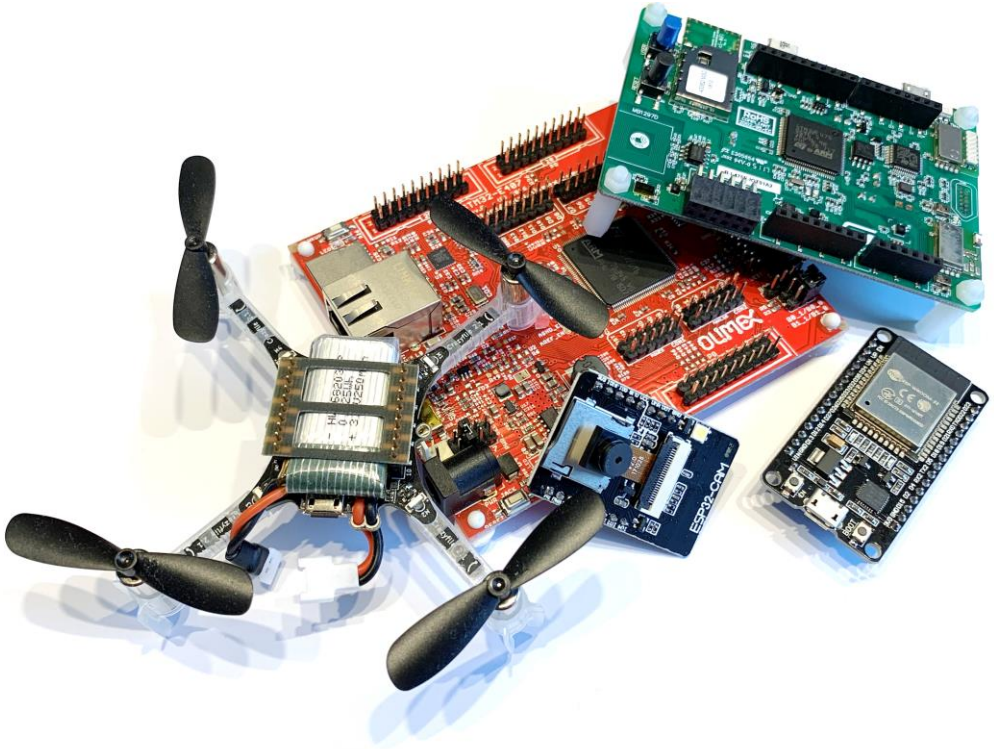
# Conclusions 2: Why using and contributing to ROS from industry?

- Fast proofs of concepts and minimum viable products
  - ROS provides solutions for almost all aspects of robotics – allows to focus on innovation
- Community support
  - Tons of documentation, videos, Discourse, ROS Answers, Working Groups, etc.
- Quality and speed
  - ROS stack is used – and thus tested – by tens of thousands developers.
- Customer interactions and partner collaborations
  - Many of our customers and partners use ROS themselves and ask for ROS compatibility
  - Access to engineering services and consulting
- Access to talents and easier teach-in
  - Most students in robotics bring ROS know-how

**Overview article for benefits of open-source software:**
**https://blog.bosch-si.com/bosch-iot-suite/the-benefits-of-open-source/**

# micro-ROS

# Microcontrollers in Robotics




- Hardware access
- Hard, low-latency real-time
- Power saving
- Safety

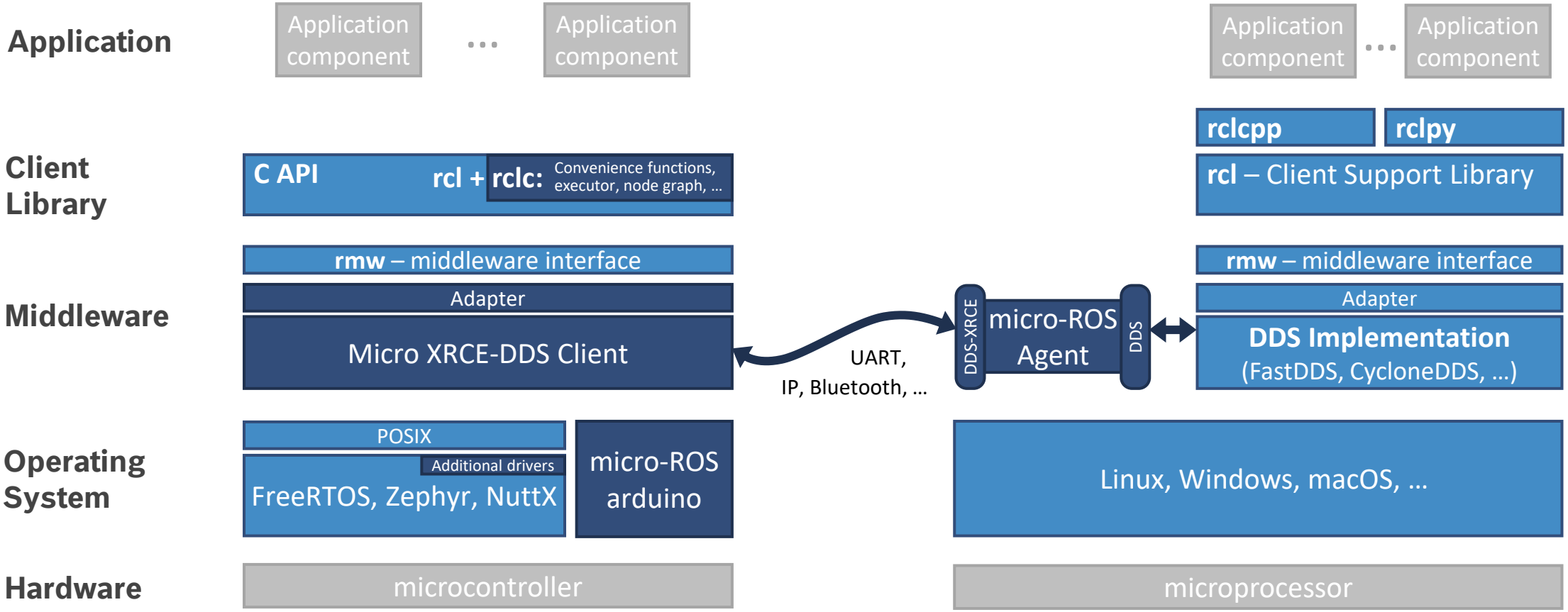
# “micro-ROS puts ROS 2 onto microcontrollers”

To bridge the gap between microcontrollers and larger processors, the EU-funded project OFERA (2018-2021) launched


- Seamless integration of µCs with ROS 2
- Ease portability of ROS 2 code to µCs
- Ensure long-term maintenance of micro-ROS stack

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

# micro-ROS Architecture





Ralph Lange (CR/AD CE-RSE) | 2022-07-01

micro-ROS | ROS 2 for microco

https://micro.ros.org

micro-ROS Overview Concepts Tutorials API Blog

Search via Lunr.js

»micro-ROS
puts ROS 2 onto microcontrollers«

# µROS

## Mission

Bridging the gap between resource-constrained microcontrollers and larger processors in robotic applications that are based on the Robot Operating Systems.


## Why Microcontrollers?

Microcontrollers are used in almost every robotic product. Typical reasons are:

- Hardware access
- Hard, low-latency real-time
- Power saving

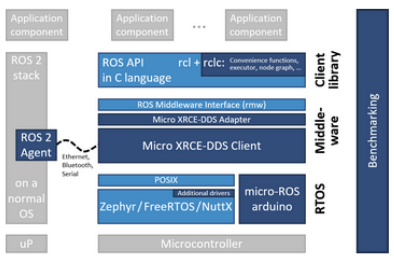
Another important reason is safety, but note that micro-ROS is not developed according to any safety standard.

## Key Features

✓ Microcontroller-optimized client API supporting all major ROS concepts
✓ Seamless integration with ROS 2
✓ Extremely resource-constrained but flexible middleware
✓ Multi-RTOS support with generic build system
✓ Permissive license
✓ Vibrant community and ecosystem
✓ Long-term maintainability and interoperability

## Architecture

The architecture of the micro-ROS stack follows the ROS 2 architecture. Dark blue components are developed specifically for micro-ROS. Light blue components are taken from the standard ROS 2 stack.




## Getting Started

Our tutorials and demos give you a quick start with micro-ROS. The basic tutorials can even be completed without a microcontroller.


## Source Code

Source code can be found at github.com/micro-ROS. It comes under the permissive license Apache 2.0 just as the standard ROS 2 stack.

The primary repository is micro_ros_setup, which provides command line scripts for creating your first micro-ROS application.

Developed an new feature or found a bug? We answer both pull requests and tickets.

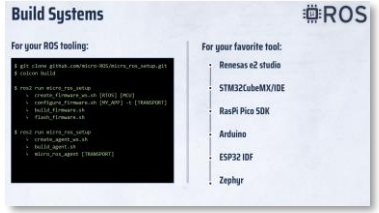
## News

## Questions

micro.ros.org

# Development of the micro-ROS project






**2017**

- Analysis of fundamental requirements and needs
- High-level concept
- Successful application for EU project

**2018**

- First prototypes with one specific µC
- Generic SW architecture
- Little open-source only
- Foundation of ROS Embedded WG

**2019**

- Open-source demo with Turtlebot
- Easy-to-use build tooling for ROS developers
- micro-ROS website
- Demos at fairs and developer conferences
- micro-ROS Slack channel

**2020**

- Focus on tutorials and open-source demos
- Support for more and more µC families
- Growing the community by the Embedded WG
- Talk, presentations, videos, tutorials, …

**2021**

- Build tooling for typical embedded developers
- Active involvement non-OFERA developers in maintenance tasks
- Engineering services by eProsima

**2022**

- Successful final review of EU project OFERA

# Conclusions 3: My learnings on a successful new OSS project

- Connect to something existing
  - Contribute to an existing OSS project rather starting from scratch
- Make entry barrier for users a low as possible
  - High-quality README, documentation, tutorials, etc.
  - Easy-to-use tooling; rely on standard tools if possible
  - First impression counts!
- Communicate frequently and respond quickly
  - Answer Slack, GitHub issues, Stackoverflow questions timely and accurately
  - Go the extra mile for requests and new requirements
- Larger team is better
  - Involve fellow students, friends – and active users
  - But be prepared to give up some control!
  - Meet frequently with your community (video, in-person)
- Marketing also applies to open-source
  - Twitter, trade fairs, developer conferences, …
- Choose the license carefully
  - Important trade-off between copyleft and permissive

# Looking forward to your questions!

Dr. Ralph Lange
Bosch Research

ralph.lange@de.bosch.com | bosch.com/research | github.com/ralph-lange

BOSCH