Programming mobile robots with ROS2 and the RCLAda Ada Client Library

Alejandro R. Mosteo

2021-jun-07

Ada-Europe International Conference on Reliable Software Technologies

CONTENTS

- About
- ROS2 introduction
- ROS2 build process
- Ada package creation
- RCLAda big picture
- RCLAda API highlights
- Robotics specifics

First of all…

Install ROS2 + Webots

(https://github.com/ada-ros/tutorial-aeic21/blob/master/Exercises.md#0-setup-of-the-working-environment)

or download the Virtual Machine

(https://github.com/ada-ros/tutorial-aeic21/releases)

or update your workspace

```
git pull --recurse-submodules && dev/make.sh
```

⚠️ Local networks ⚠️

```
$ export ROS_LOCALHOST_ONLY=1
$ export ROS_DOMAIN_ID=$(($RANDOM % 102))
```

Ada & me

- BSc in Computer Engineering (1999)
  - Introduction to programming
  - Data structures and algorithms
  - Concurrent programming
  - Real-time systems
- Ada-Spain / Ada-Europe (2006)
  - Back from industry
- PhD in multi-robot coordination (2010)
  - SANCTA robotics library
  - player-ada
- Alire project (2017-)
  - Ada package manager
- RCLAda (2018-)
  - ROS2 client library

Research group

Robotics, Perception and Real-Time group (RoPeRT)

Engineering Research Institute of Aragon (I3A)

Defense University Center (CUD)

RoPeRT

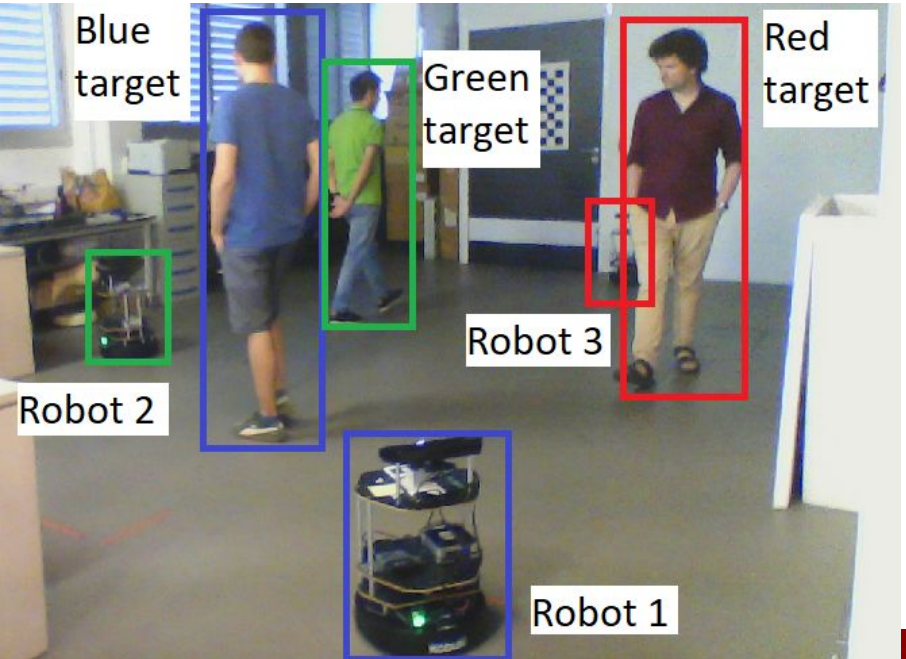
Optimal distributed coordination

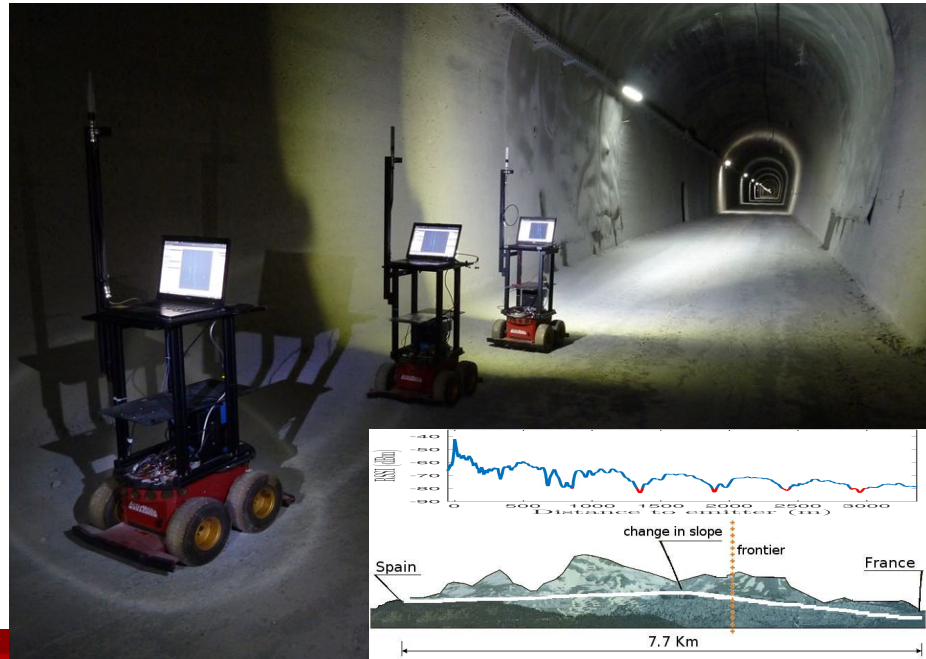
Real-time multi-hop communication

Underground drone reconnaissance

ROBOTICS
UNIVERSIDAD DE ZARAGOZA

http://robots.unizar.es/





ROBOTICS
UNIVERSIDAD DE ZARAGOZA

http://robots.unizar.es/

Acknowledgements

- María T. Lorente
  - Beta testing
- Danilo Tardioli
  - Simulator models

- AdaCore
  - Research visit
- ROS-industrial
  - FTP 2020

AdaCore
Build software that matters.

ROSin

In a nutshell

What do we want: to “move” robots

How we will do it: with ROS2

Supported languages: C++, Python

Unsupported favorite language: Ada

Robotics needs

Robotics

Academia ⟷ Industry

Fast Prototyping

Long runs

Correctness

Certifiability

The Seattle Times

FAA orders Boeing 787 safety fix: Reboot power once in a while

FAA orders Boeing 787 safety fix: Reboot power once in a while. Originally published December 1, 2016 at 11:05 am Updated December 2, 2016 at 12:35 am.

Dec 1, 2016

High-integrity subsets

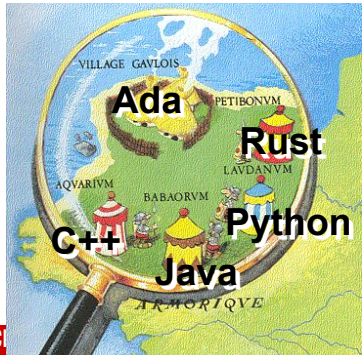
Clamp down on general-purpose (C/C++)

Language for High-Integrity Systems

Design objective (Ada)

Ada history

- 1975: Working group DoD / UK MoD
  - STRAWMAN first discussions
- 1978: STEELMAN requirements
  - **Embedded, reliability, maintainability, efficiency requirements**
  - No suitable existing candidate
- 1979: Green proposal by Jean Ichbiah of Honeywell Bull
  - Renamed to Ada
- 1983: standard ANSI /MIL-STD-1815A (Ada 83)
- 1991-1997: DoD mandate years
  - From 450 to 37 languages by 1998
- Today: niche in many critical industries
  - **Aerospace, railways, automotive, …**



Evolution

- Ada Rapporteur Group
  - Receives suggestions, requests, comments
  - Prioritizes “not” doable in current Ada
- Ada Reference Manual (ARM)
  - AARM: Annotated ARM for experts, compiler writers
  - All are ISO standards
- **Ada Conformity Assessment Test Suite (ACATS)**

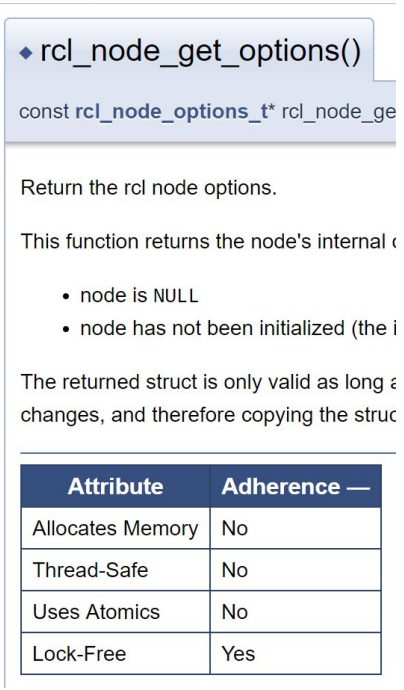
Source: https://www.adacore.com/about-ada

Feature	Ada 83	Ada 95	Ada 2005	Ada 2012	Ada 202X
Packages	✓	✓	✓	✓	✓
Generics	✓	✓	✓	✓	✓
Derived ADTs	✓	✓	✓	✓	✓
Object orientation (tagged types)		✓	✓	✓	✓
Multiple inheritance (abstract interfaces)			✓	✓	✓
Design by Contract				✓	✓
Numeric types (fixed, floating, decimal, custom)	✓	✓	✓	✓	✓
Tasks	✓	✓	✓	✓	✓
Monitors		✓	✓	✓	✓
Real-time systems annex		✓	✓	✓	✓
Ravenscar profile			✓	✓	✓
Multiprocessor affinities, Multiprocessor Ravenscar				✓	✓
Parallel constructs (blocks, loops)					✓

Why ROS2 + Ada

ROS2

- More focus on
  - Embedded, real-time, industrial use
- Traditional strong points of Ada
  - Portable & precise memory layouts, at bit level
  - Annex C: systems programming
    - Interrupts, atomics, volatiles
  - Annex D: (hard) real time
    - Priorities, schedulers, monotonic clock, tasking profiles
  - Portability (ISO/IEC 8652:2012)
  - Certification
    - DO-178B/C (aero), EN 50128 (railway), ECSS-E-ST-40C/ECSS-Q-ST-80C (space), IEC 61508 (industrial automation), ISO 26262 (automotive)



◆ rcl_node_get_options()

const **rcl_node_options_t*** rcl_node_ge

Return the rcl node options.

This function returns the node's internal

- node is `NULL`
- node has not been initialized (the i

The returned struct is only valid as long a changes, and therefore copying the struc

Attribute	Adherence —
Allocates Memory	No
Thread-Safe	No
Uses Atomics	No
Lock-Free	Yes

ROS2 introduction

What is ROS

- Robot Operating System
  - But not really

ROS

Open Source Robotics Foundation

*"The Robot Operating System (ROS) is a set of **software libraries** and **tools** that help you build robot applications. From **drivers** to state-of-the-art **algorithms**, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all **open source**."*

Main OSRF project (10 years now)

ROS2 Main Parts

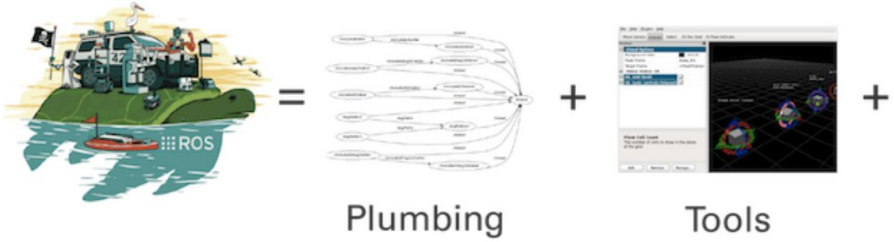
ROS

- Collection of Debian/Ubuntu packages ready to use
  - Sensor/platform/actuator drivers
  - High-level algorithms
- Build system (∈ “tools”)
  - Heterogeneous language environment
  - Nodes isolated as processes/threads
- Intercommunication facilities (“plumbing”)
  - Message publishing
  - Remote Procedure Calls
  - Actions

Colcon

CMake
Cross-platform Make

python

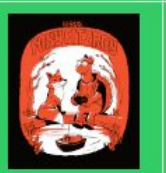


Ties to Ubuntu

ROS (1)

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys **(Recommended)**	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019

ROS2

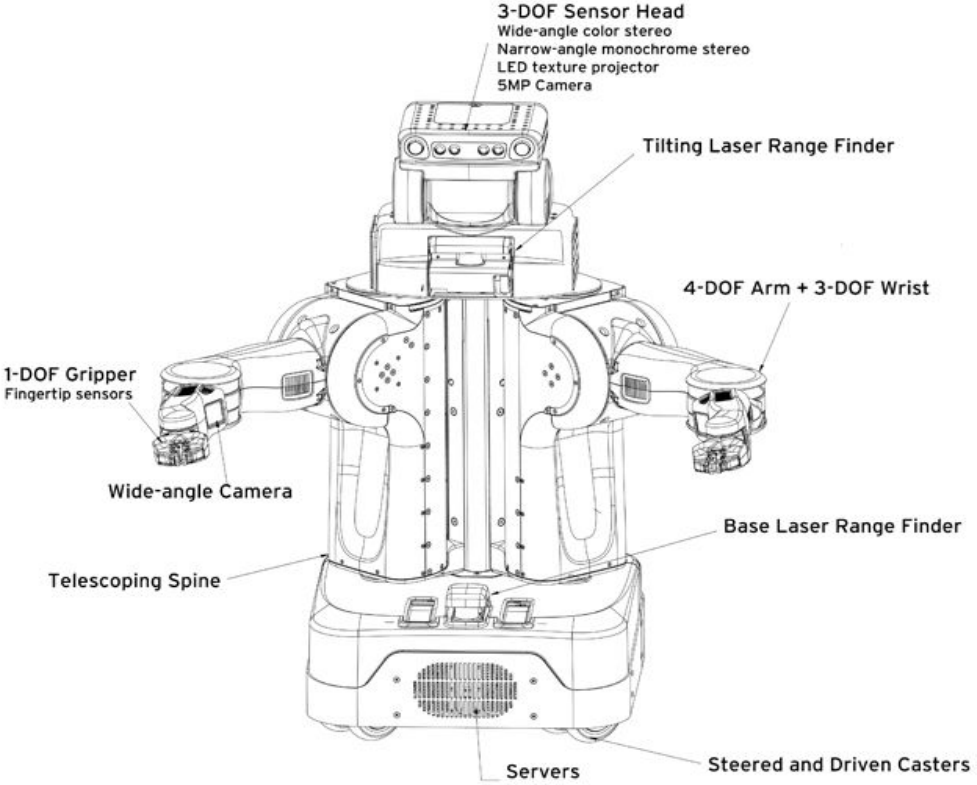
Distro	Release date	Logo	EOL date
Galactic Geochelone	May 23rd, 2021		November 2022
Foxy Fitzroy	June 5th, 2020		May 2023
Eloquent Elusor	November 22nd, 2019		November 2020

ROS (+ Ada) history

- (2000) Player/Stage project
  - Player-Ada
- (2008) ROS (Willow Garage)
  - No Ada in ROS
  - Ada4CMake
- (2015) ROS2 (Open Source Robotics Foundation)
  - RCLAda

Brian Gerkey

PR2: the kick-off robot



https://www.youtube.com/watch?v=c3Cq0sy4TBs

Academia & Research

- ROS support widespread
  - Expected in research/academia contexts
  - Either by 1st or 3rd parties
  - https://robots.ros.org/

Summit XL 4WD Autonomous Robot

Product Code : RB-Rtk-04

1 Review(s)

USD $15,000.00

Svenzva Robotics Revel 6 DoF Robotic Arm

Product Code : RB-Svz-01

USD $6,495.00

ROSbot 2.0 w/ LIDAR & RGBD Robotic Platform

Product Code : RB-Rco-07

USD $1,899.00

TeraRanger Duo ToF Rangerfinder with Sonar Sensor

Product Code : RB-Ter-07

Excl. Tax: €185.00

Incl. Tax: €223.85

2.3 3D Sensors (range finders & RGB-D cameras)

- Argos3D P100 ToF camera
- Basler ToF ES camera
- DUO3D™ stereo camera
- Ensenso stereo cameras
- Forecast 3D Laser with SICK LIDAR
- SceneScan and SP1 by Nerian Vision Technologies
- OpenNI driver for Kinect and PrimeSense 3D sensors
- Trifo Ironsides
- PMD Camcube 3.0
- IFM O3M250 ToF camera
- Intel® RealSense™ F200/VF0800
- Intel® RealSense™
- Roboception rc_visard stereo camera
- Terabee 3D ToF camera
- Orbbec Astra
- SICK MRS1xxx lasers
- SICK MRS6xxx lasers
- SICK LD-MRS laser (identical to IBEO LUX) or csiro-asl/sick_ldmrs
- Sentis ToF M100 camera
- Mesa Imaging SwissRanger devices (3000/4000/4500)
- Velodyne HDL-64E 3D LIDAR
- Livox 3D LiDAR

2.2 2D range finders

- NaviRadar
- HLS-LFCD LDS
- Hokuyo Scanning range finder
- PepperI+Fuchs R2000 laser
- Leuze rotoScan laser rangefinder driver (ROD-4, RS4)
- RPLIDAR 360 laser scanner Driver(python)
- RPLIDAR A1/2 laser(c++)
- SICK LMS1xx lasers or LMS1xx
- SICK LMS2xx lasers or sicktoolbox_wrapper
- SICK S3000 laser
- SICK S300 Professional
- SICK TiMxxx lasers or sick_tim
- SICK Safety Scanners (microScan3)
- TeraRanger Multiflex
- TeraRanger Hub & Tower
- TeraRanger Hub Evo & Tower Evo
- TeraRanger Evo 64px ToF range finder
- Neato XV-11 Laser Driver

In mobile robots

Why ROS


GAZEBO


pointcloudlibrary


colcon


ROS


ROS
industrial

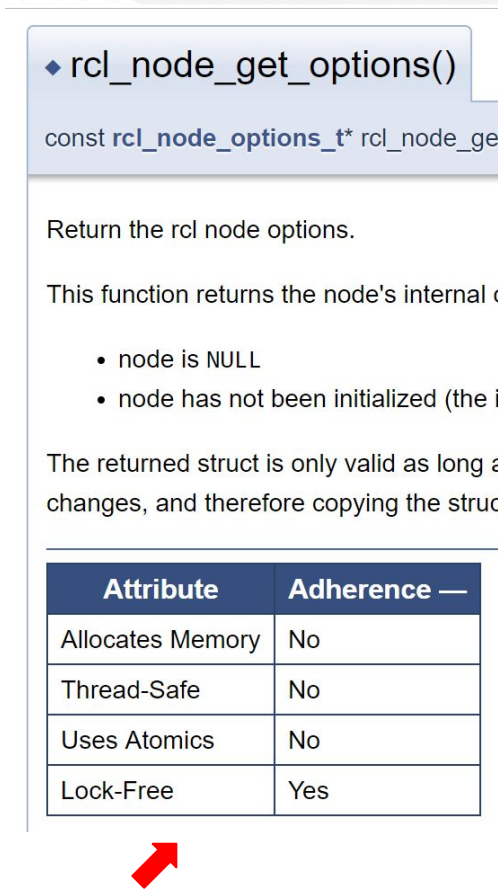

MoveIt!


OpenCV


ROSin

ROS2 vs ROS

- Improve shortcomings for
  - Embedded (microcontrollers)
    - ROS has been mostly a linux affair
  - Real-time
    - Coming from firm/soft real-time
  - Actual readiness for industrial settings
    - Long lived processes vs short experiments
    - Standard DDS for data transport
      - Swappable implementation
- Traditional strongholds of Ada



◆ rcl_node_get_options()

const **rcl_node_options_t*** rcl_node_ge

Return the rcl node options.

This function returns the node's internal

- node is `NULL`
- node has not been initialized (the i

The returned struct is only valid as long a changes, and therefore copying the struc

Attribute	Adherence —
Allocates Memory	No
Thread-Safe	No
Uses Atomics	No
Lock-Free	Yes

ROS2 Working Groups

- Working Groups on
  - Real-time
  - Safety-critical

ROS 2 and Real-time

Next Generation ROS

Dejan_Pangercic 9d

In one of the previous ROS 2 TSC meeting it was suggested that we form a Working Group in which we will try to analyse the current state of ROS 2 and make it real-time.

To this date we have the following articles about real-time in ROS 2:

1. Original article by Jackie: https://design.ros2.org/articles/realtime_background.html 11
2. ROS 2 ported on some RTOS (https://www.esol.com/embedded/ros.html 9, http://blackberry.qnx.com/en/articles/what-adas-market-needs-now 2)
3. Apex.AI article about porting ROS 1 applications to ROS 2 applications: https://www.apex.ai/blog/porting-algorithms-from-ros-1-to-ros-2 12
4. Bosch proposing how to make Callback-group-level Executor real-time https://vimeo.com/292707644 7

Since real-time is not something that can start and stop within the ROS 2 “borders”, we would like to propose to analyse an entire stack, from the hardware platform to the applications written with ROS 2.

Safety-critical WG

Next Generation ROS

gbiggs 4d

Some time ago I was asked to lead a working group looking at the use of ROS 2 in safety-critical systems. These are systems that may potentially cause harm to people or the environment, and I think that most of us agree that a large number of robot applications fall into this category.

The working group will look at topics including:

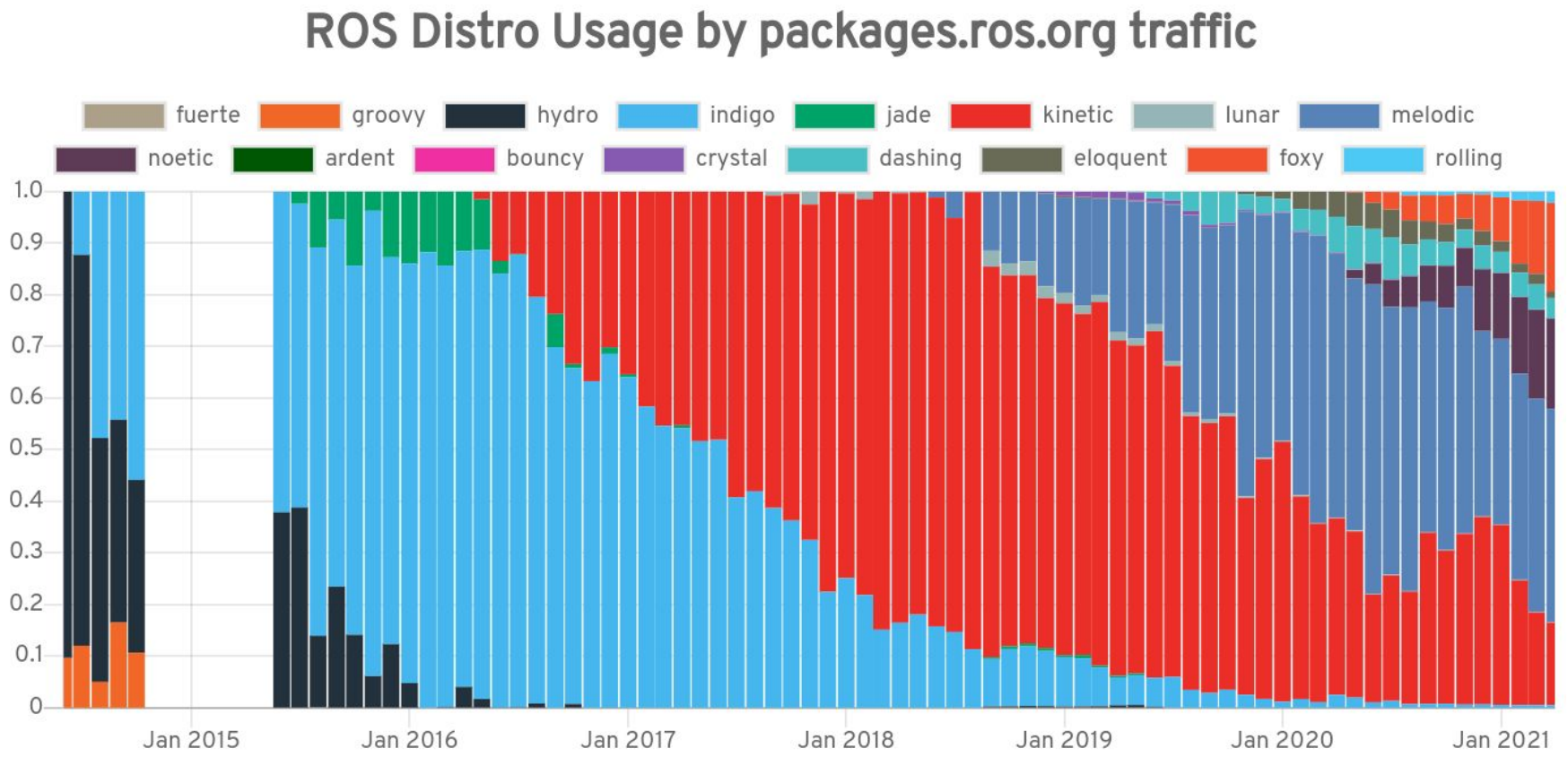
- Documenting how to use ROS 2 in a safety-critical application
- Use of tools to support the above
- Additional processes, tools and methods needed for building a safety-critical robot that are not currently covered by something in ROS but could be
- How to make the client libraries usable in a safety-critical system, and work on safety-focused client libraries (for example, a SPARK client library)
- Cross-over issues with the QA and real-time working groups for infrastructure, tooling and methods
- Cross-over issues with the navigation and manipulation working groups for sample applications
- Anything else safety-related someone brings along

ROS2 Technical Steering Committee

	ADLINK Technology: Joe Speed
amazon	Amazon: Aaron Blasdel
Apex.AI	Apex.AI: Lyle Johnson
BOSCH Invented for life	Bosch: Karsten Knese
ubuntu	Canonical: Sid Faber
eProsima The Middleware Experts	eProsima: Jaime Martin Losa
DEVCOM GROUND VEHICLE SYSTEMS CENTER	GVSC: Jerry Towler (SwRI)
intel	Intel: Harold Yang
iRobot	iRobot: Ori Taka
LG Electronics	LG Electronics: Lokesh Kumar Goel
Microsoft	Microsoft: Lou Amadio
open robotics	Open Robotics: Chris Lalancette
PICKNIK	PickNik: Dave Coleman
ROBOTIS	ROBOTIS: Will Son
ROVER ROBOTICS	Rover Robotics: Nick Fragale
SAMSUNG	Samsung: Steven Macenski
SONY	Sony: Tomoya Fujita
TOYOTA RESEARCH INSTITUTE	Toyota Research Institute: Ian McMahon
WNDRVR	Wind River: Andrei Kholodnyi

https://docs.ros.org/en/foxy/Governance.html

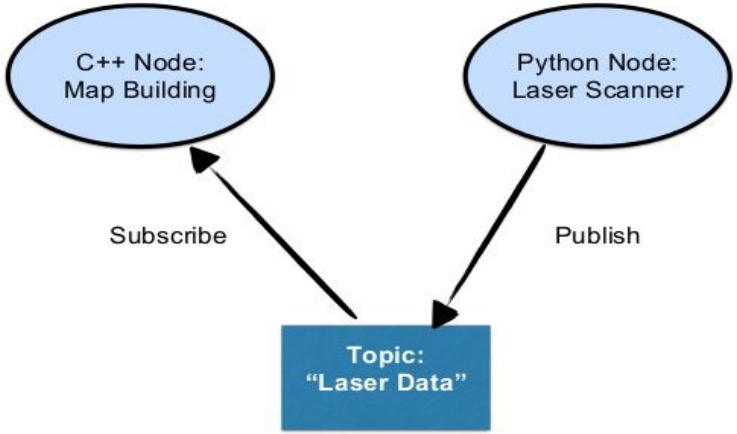
ROS version adoption

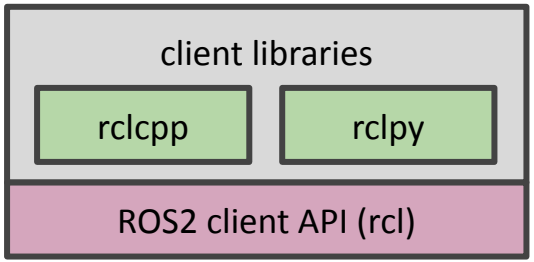


https://metrics.ros.org/packages_rosdistro.html

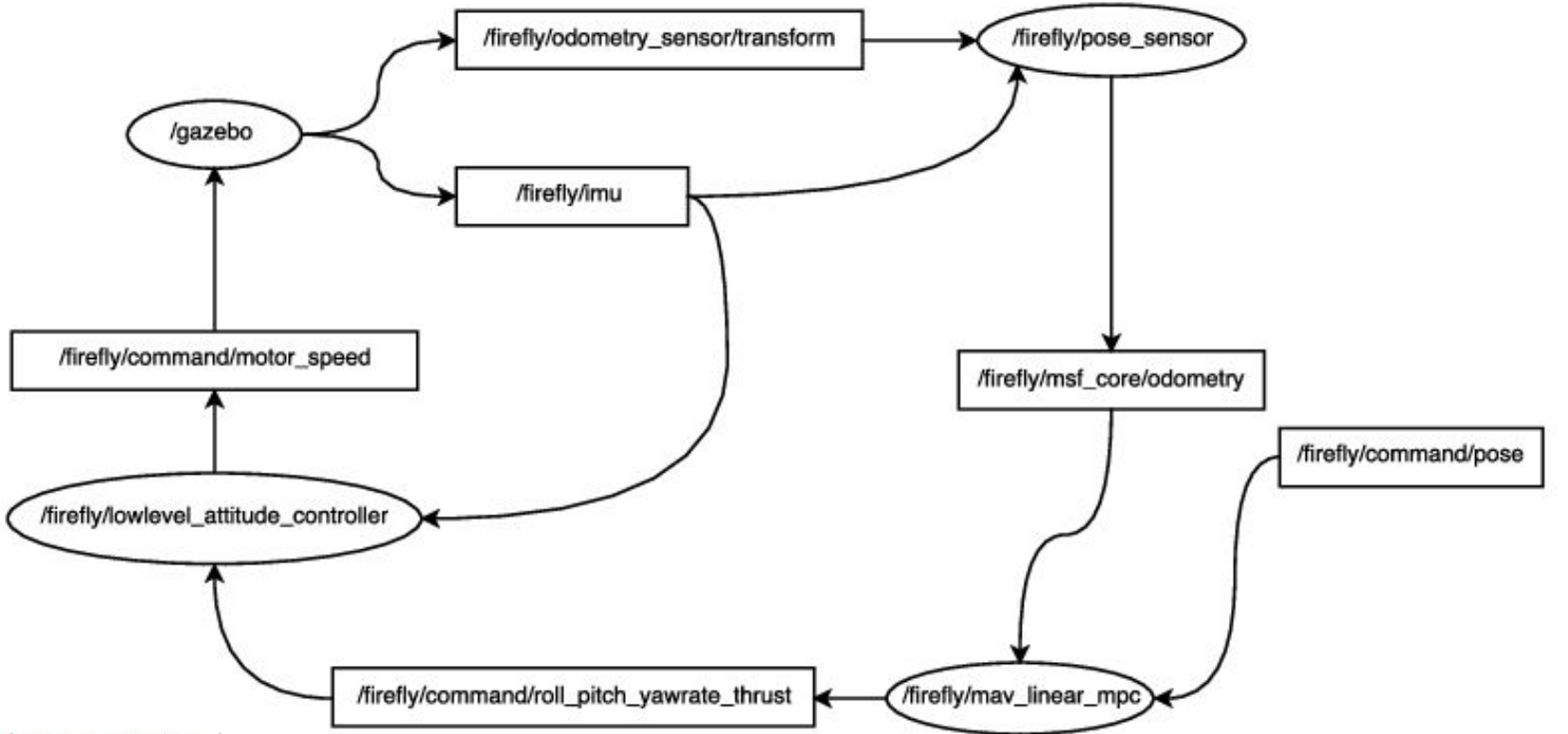
Down to business

- ROS2 “program”
  - Set of nodes (processes)
    - Found in ROS packages
  - Interconnected (DDS) by
    - Topics
      - Publish, Subscribe
    - Services
      - Request + Response
  - Supported languages
    - C++, Python (Client APIs)
    - C (low-level API)





Nodes + Topics



Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System

May 2017 · Studies in Computational Intelligence

DOI: 10.1007/978-3-319-54927-9_1

In book: Robot Operating System (ROS) The Complete Reference, Volume 2 · Publisher: Springer
Editors: Anis Koubaa

Mina Samir Kamel · Thomas Stastny · Kostas Alexis · Roland Siegwart

Task 0.5: play with TurtleSim

https://docs.ros.org/en/foxy/Tutorials/Turtlesim/Introducing-Turtlesim.html

Part 1: ROS2 build process

Workflow (developer)

1. `$ ros2 pkg create`

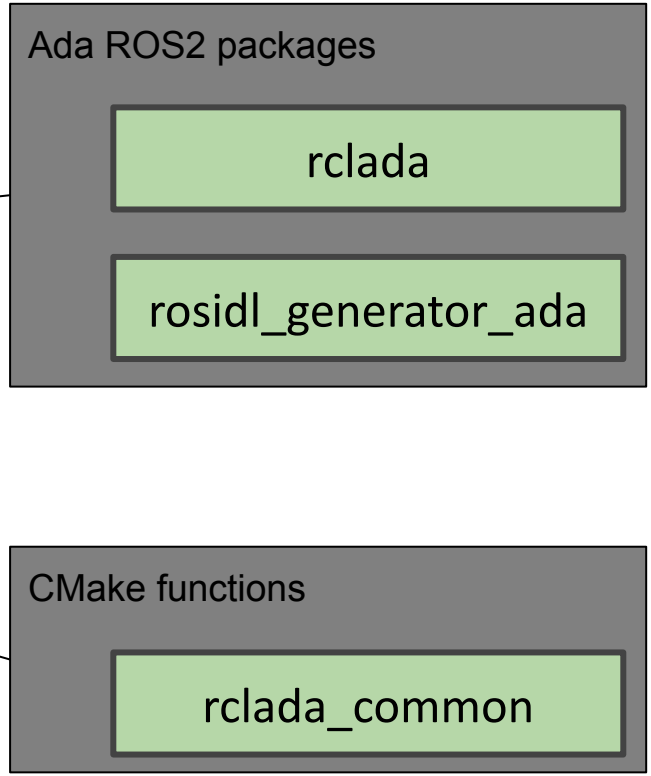
   write code

2. `$ colcon build`

   compile code

3. `$ ros2 run`

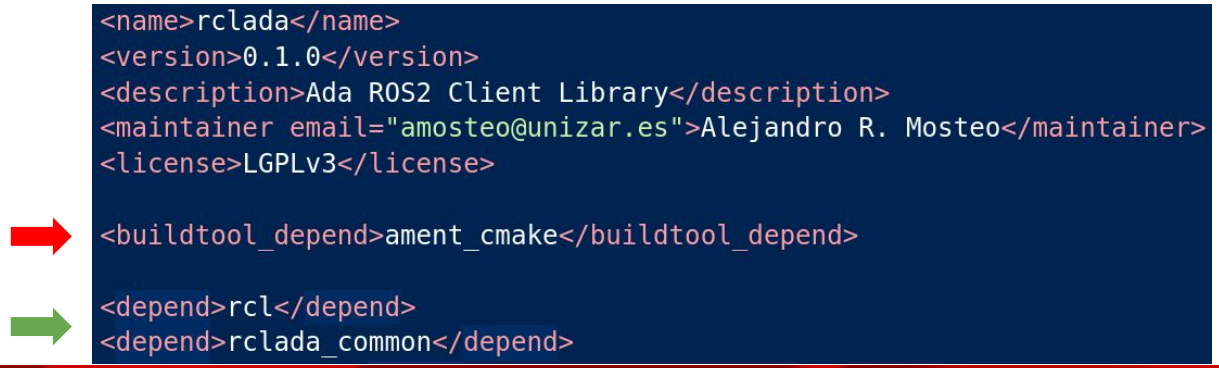
   execute code



What is colcon?

- Collective Construction

  "colcon is a command line tool to improve the workflow of building, testing and using multiple software packages. It automates the process, handles the ordering and sets up the environment to use the packages."
- Meta-build tool
  - Relies on single package.xml file
  - Recurses from launch folder looking for packages
- Example package.xml contents:

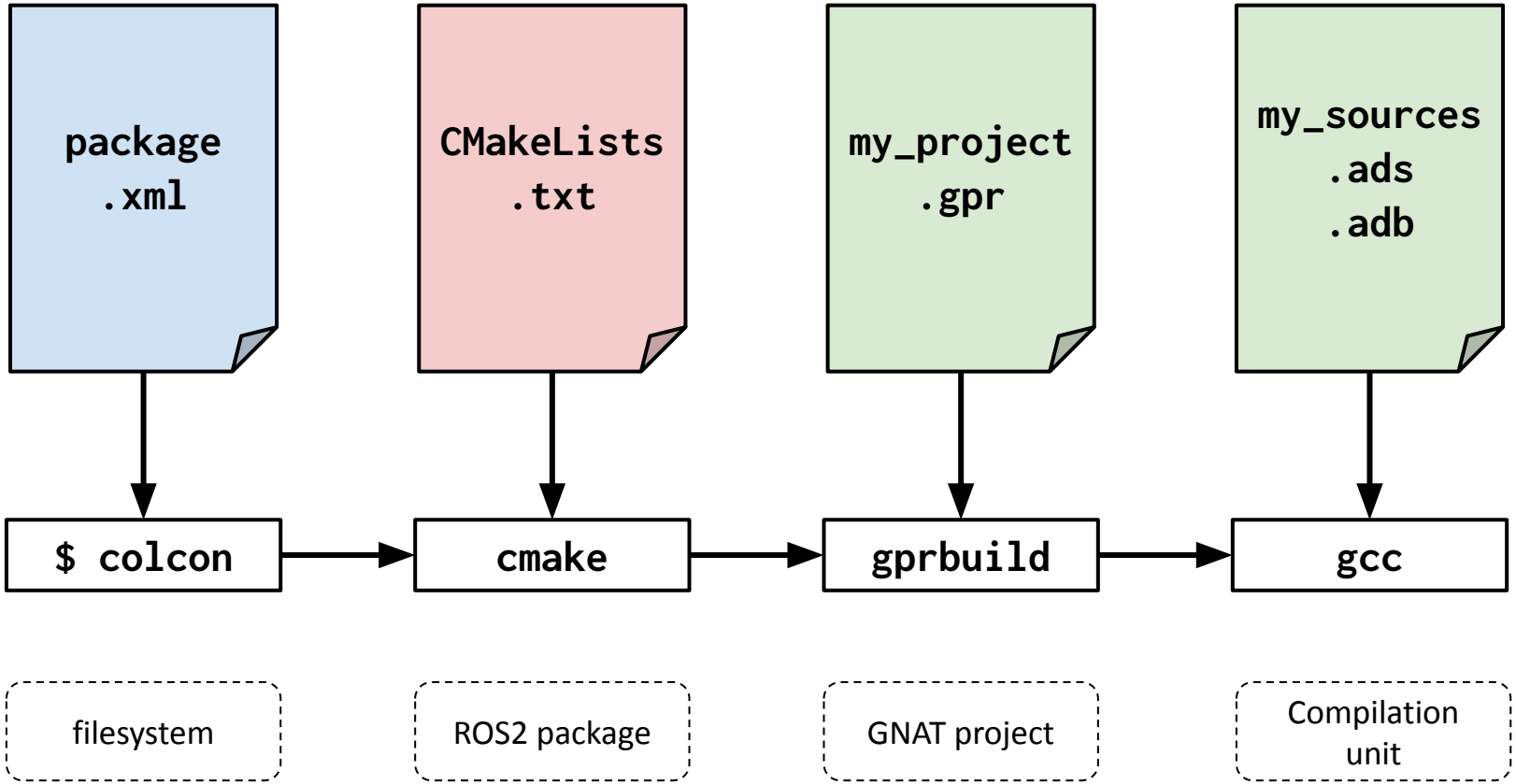


```
<name>rclada</name>
<version>0.1.0</version>
<description>Ada ROS2 Client Library</description>
<maintainer email="amosteo@unizar.es">Alejandro R. Mosteo</maintainer>
<license>LGPLv3</license>

<buildtool_depend>ament_cmake</buildtool_depend>

<depend>rcl</depend>
<depend>rclada_common</depend>
```

COLCON GRAPH

```
$ colcon graph

rclada_common                +***...****
rosidl_generator_ada          +*....***
rclada                         +*******
rclada_client_skeleton          +
rclada_examples                  +
rclada_fosdem20                   +
rclada_tf2                         +* *
tutorial_common                     +**
tutorial_exercises                   +
tutorial_solutions                    +
```

Files and interactions



Filesystem

```
ada4ros2                                  # Workspace
├── build                                     # Out-of-tree build location
│   ├── rclada
│   ├── rclada_common
│   ├── ...
│   └── tutorial_exercises
├── install                                   # Install location (akin to /opt/ros/foxy)
│   ├── setup.bash                              # Environment activation script
│   └── rclada
│       ├── include
│       ├── lib
│       └── share
├── log
└── src                                       # ROS2 packages
    ├── rclada
    │   ├── package.xml                         # ROS2 manifest file (colcon)
    │   └── gpr_rcl                               # A GNAT project
    │       ├── rcl.gpr                           # Project file
    │       ├── src                                 # Project sources
    │       └── gen                                 # g++ -fdump-ada-spec
    ├── rclada_common
    ├── rosidl_generator_ada
    └── tutorial
        ├── common                            # Nested packages (colcon searches recursively)
        └── exercises
```

ENVIRONMENT SETUP

- AFTER a successful `colcon build`
  - Found in ./install/setup.bash
  - Build incrementally in case of problems

```
--packages-select
--packages-up-to
--event-handlers console_direct+
--event-handlers console_cohesion+
--executor sequential
```

- **setup.bash** recursively loads environment layers
  - ROS2 ← RCLAda ← Ada projects

- **local_setup.bash** loads only one layer

Build scripts

- Custom scripts
  - in `ada4ros2/dev`
  - `make.sh [package]`
    - Build from scratch package and all dependencies
  - `update.sh [package]`
    - `colcon build` package and all dependencies
  - `make-package.sh [package]`
    - Build from scratch package only
    - Dependencies must be already built and environment loaded
  - `build-package.sh [package […]]`
    - `colcon build` packages only
    - Dependencies must be already built and environment loaded

Environment example

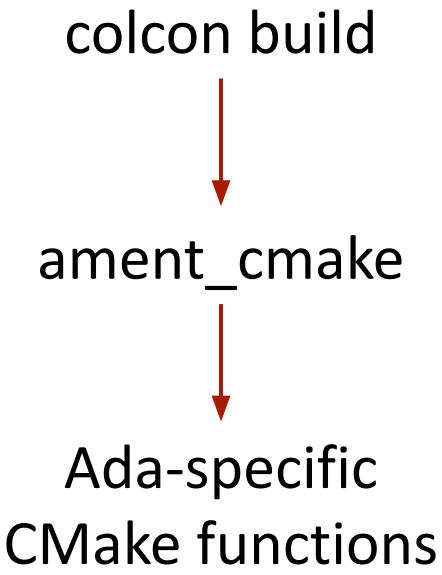
AMENT_PREFIX_PATH=/home/user/prog/ada4ros2/install/webots_ros2_p3at:/home/user/prog/ada4ros2/install/tutorial_solutions:/home/user/prog/ada4ros2/install/tutorial_exercises:/home/user/prog/ada4ros2/install/tutorial_common:/home/user/prog/ada4ros2/install/rclada_tf2:/home/user/prog/ada4ros2/install/rclada_fosdem20:/home/user/prog/ada4ros2/install/rclada_examples:/home/user/prog/ada4ros2/install/rclada_client_skeleton:/home/user/prog/ada4ros2/install/rclada:/home/user/prog/ada4ros2/install/rosidl_generator_ada:/home/user/prog/ada4ros2/install/rclada_common

CMAKE_PREFIX_PATH=/home/user/prog/ada4ros2/install/tutorial_solutions:/home/user/prog/ada4ros2/install/tutorial_exercises:/home/user/prog/ada4ros2/install/tutorial_common:/home/user/prog/ada4ros2/install/rclada_tf2:/home/user/prog/ada4ros2/install/rclada_fosdem20:/home/user/prog/ada4ros2/install/rclada_examples:/home/user/prog/ada4ros2/install/rclada_client_skeleton:/home/user/prog/ada4ros2/install/rclada:/home/user/prog/ada4ros2/install/rosidl_generator_ada:/home/user/prog/ada4ros2/install/rclada_common

COLCON_PREFIX_PATH=/home/user/prog/ada4ros2/install

Task 1: create an Ada ROS2 package

Structure



ament_cmake

- ament_cmake is the CMake build method for ROS2
- It *is* plain CMake
  - Plus a few useful extra functions
  - ament packages defined by the ROS2 project
  - Sharing of resources (ament index)
- Ada ROS2 packages are built using ament_cmake
  - Plus Ada-specific functions
    - These are CMake functions however
    - A bit of CMake knowledge is useful in case of problems

build cycle

- The sequence is:
  - colcon invokes cmake on our CMakeLists.txt
    - with ament_cmake functions preloaded
  - find_package(rclada_common)
    - Regular CMake import, makes Ada CMake functions available
  - After package configuration:
    - colcon invokes make
    - which in turn invokes gprbuild
  - After successful build:
    - source setup.bash

Support for ament_cmake

rclada_common

- **ada_begin_package**()
- **ada_end_package**()

  Needed to propagate Ada information through ROS2 packages

- **ada_add_executables**(TARGET SRCDIR BINDIR EXECUTABLES...)

  Declares an Ada executable to be built and exported (tab completion)

- **ada_add_library**(TARGET SRCDIR GPRFILE)

  Declares an Ada library project to be built and exported to other Ada packages

- **ada_import_interfaces**(PKG_NAME...)

  Generates bindings to the typesupport handle functions

  Generates Ada types for messages

- **ada_generate_binding**(TARGET SRCDIR GPRFILE INCLUDE HEADERS...)

  Invokes the binding generator in the context of an Ada project

Support for ament_cmake

rclada_common

```
ada_add_executables(
    TARGET          # CMake target name
    SRCDIR          # ABSOLUTE path to sources
                    # Use ${PROJECT_SOURCE_DIR}
    BINDIR          # Relative path from GPR file to binaries folder
                    #   for Object_Dir use “<BINDIR>”; -- in GPR
    EXECUTABLES…    # Simple name of built executables
                    # Blank- or newline-separated
)
```

```
// CMakeLists.txt
cmake_minimum_required(VERSION 3.5)

project(my_ada_ros2_project VERSION 0.1.0)

find_package(rclada_common REQUIRED)

ada_begin_package()

find_package(rclada REQUIRED)
find_package(rosidl_generator_ada REQUIRED)

ada_import_interfaces(PKG_NAME)

ada_add_executables(
       my_ada_project         # CMake target
       ${PROJECT_SOURCE_DIR} # Path to *.gpr
       bin                    # Path to binaries
       my_ada_main)           # Binaries (nodes)

ada_end_package()
```

Standard CMake project declaration

Import Ada-specific CMake functions

Import Ada environment

Import RCLAda GPR projects

- RCLAda: Nodes, Topics, etc
- ROSIDL_Ada: Messages

Import message definitions

Declare our Ada binary/library project

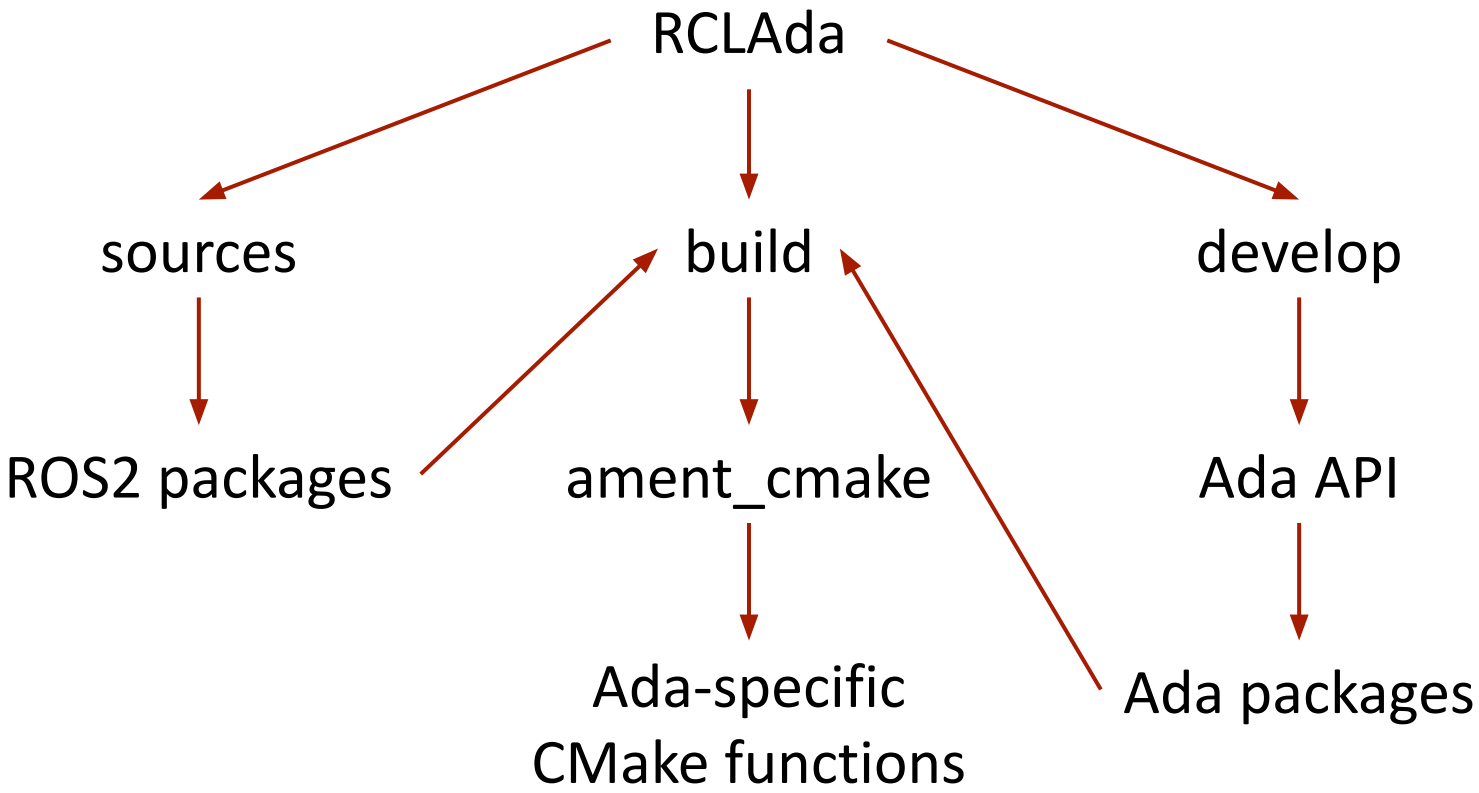
Export our additions to downstream

TASK 1 hints

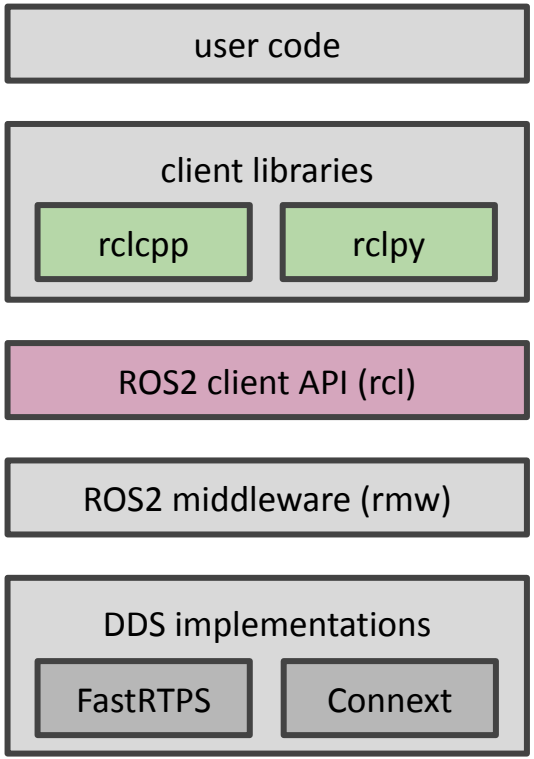
1. Source ROS2 env:
   - `$ source /opt/ros/foxy/setup.bash`
2. Inside `ada4ros2:`
   - `$ colcon build`
3. Source the newly built environment
   - `$ source install/local_setup.bash`
4. Create the new package
   - `$ pkg create … # Check exercise directions`
   - Check that it builds with no modifications
5. Create the GNAT project
   - Verify that it builds standalone in GNATstudio/GPS
6. Modify your CMakeLists.txt to embed the GNAT project
   - `$ colcon build --event-handlers console_direct+`

Part 2: RCLAda big picture

STRUCTURE

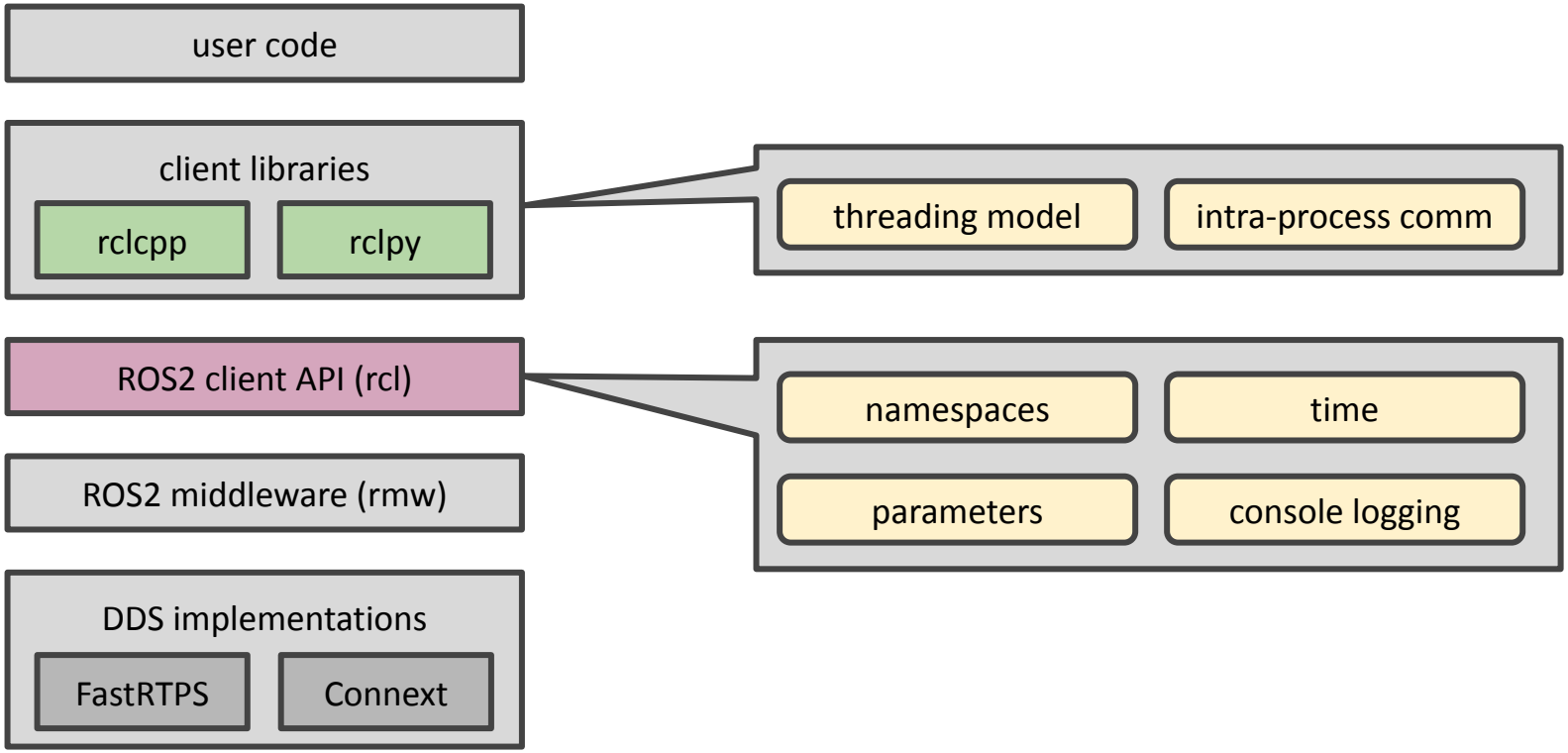

RCLAda
sources
build
develop
ROS2 packages
ament_cmake
Ada API
Ada-specific
CMake functions
Ada packages

ROS2 client support

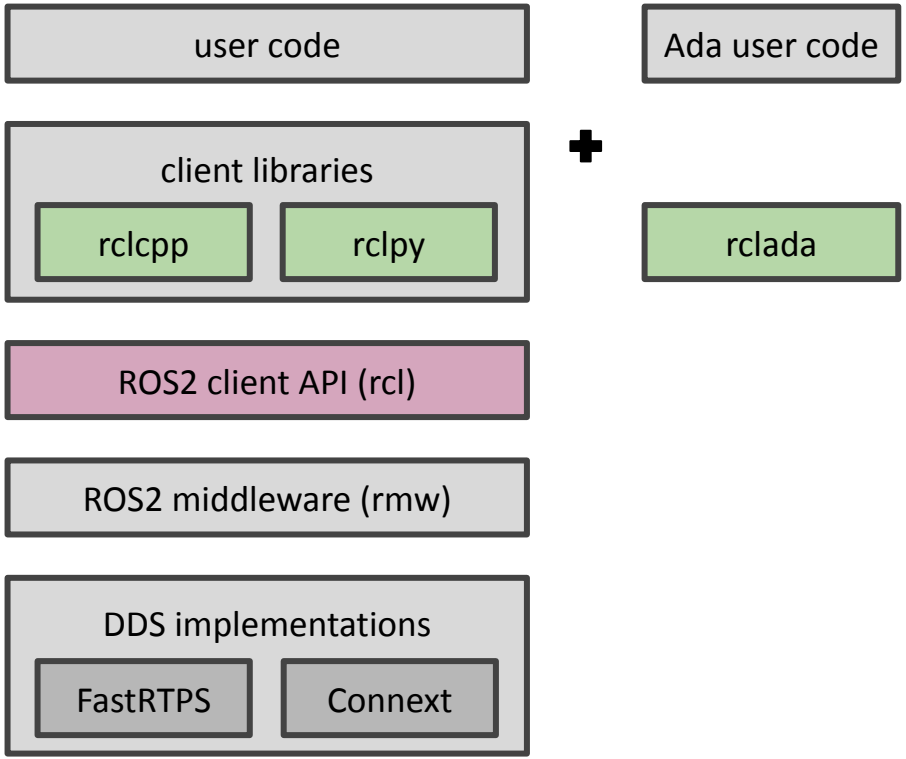


Original diagram by Deanna Hood, William Woodall: https://goo.gl/oCHR7H

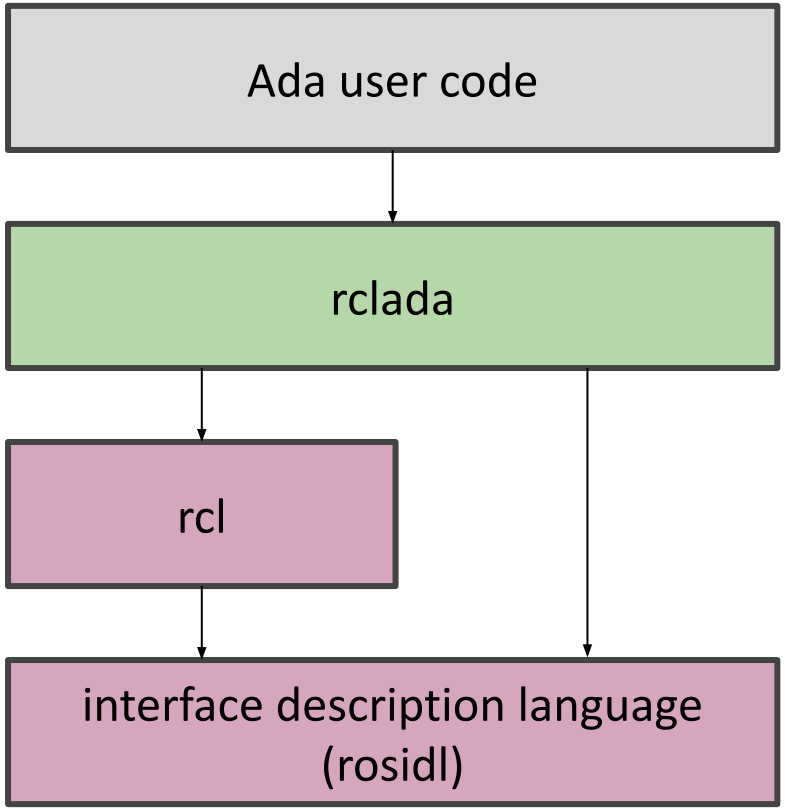
ROS2 client support



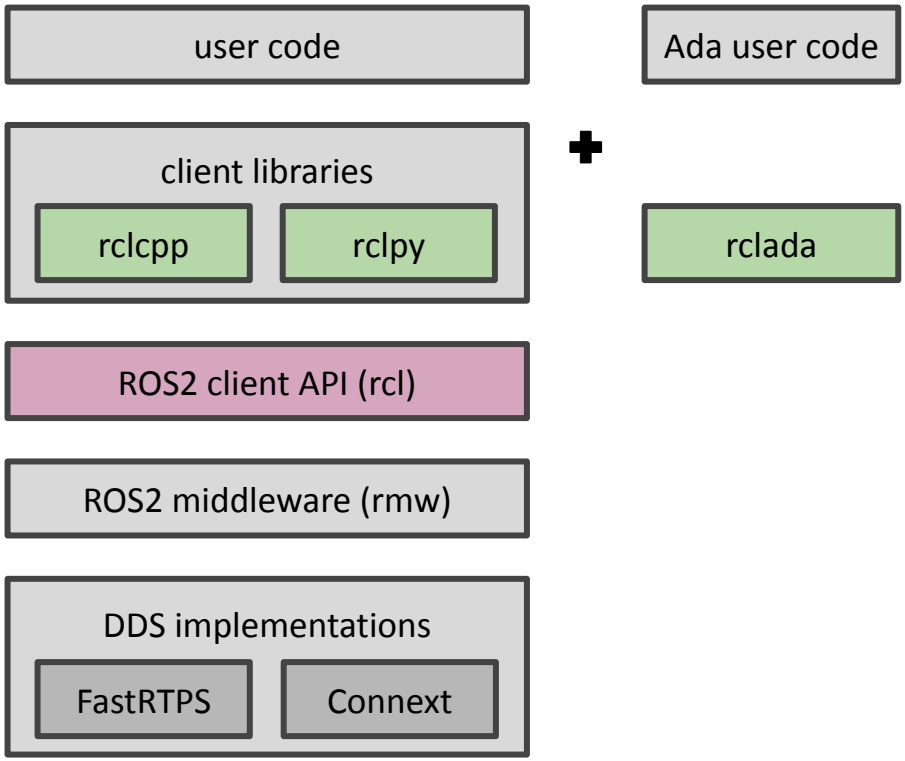
ROS2 client support

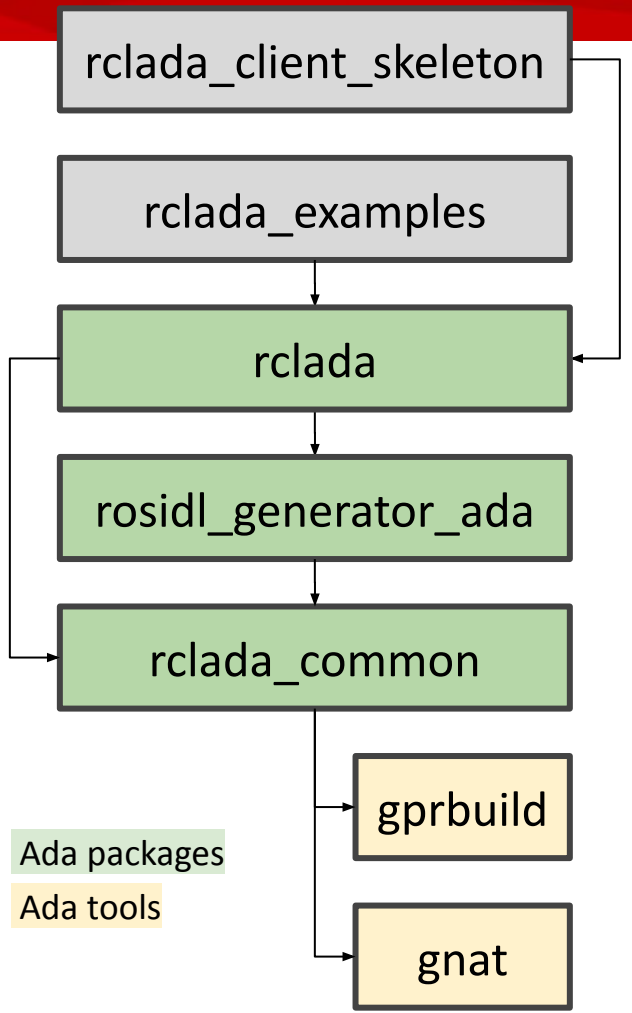


ROS2 client support

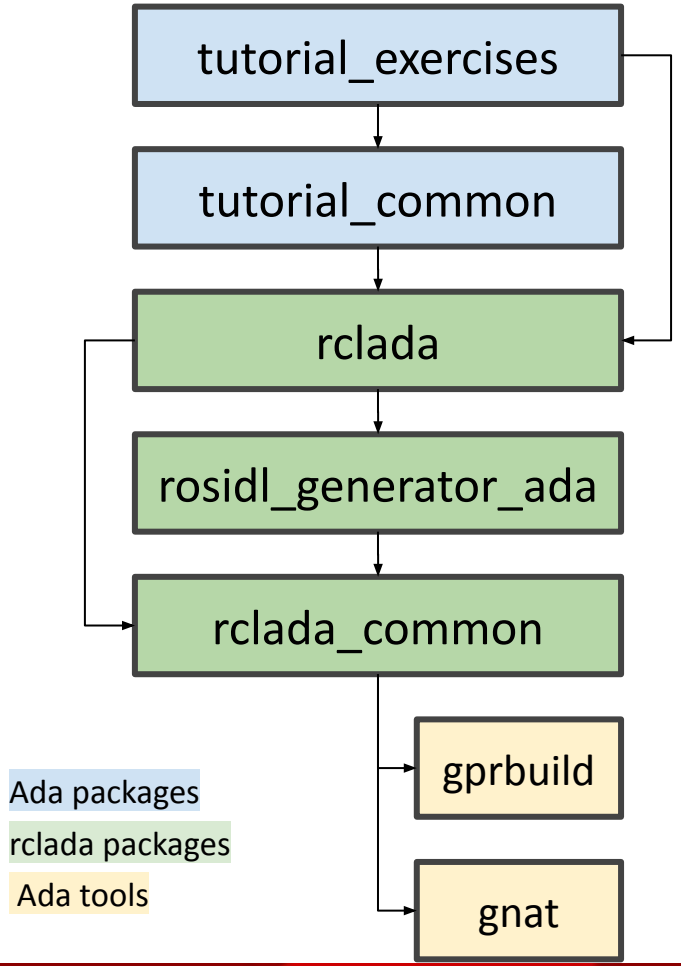


ROS2 client support





Ada ROS2 packages



Project with tutorial exercises

A few reusable types/messages/algorithms

Client API (nodes, topics, services, …) & self-tests

Message support

CMake functions

GNAT build system

Ada compiler (gnat-gcc) and tools

Writing Ada bindings

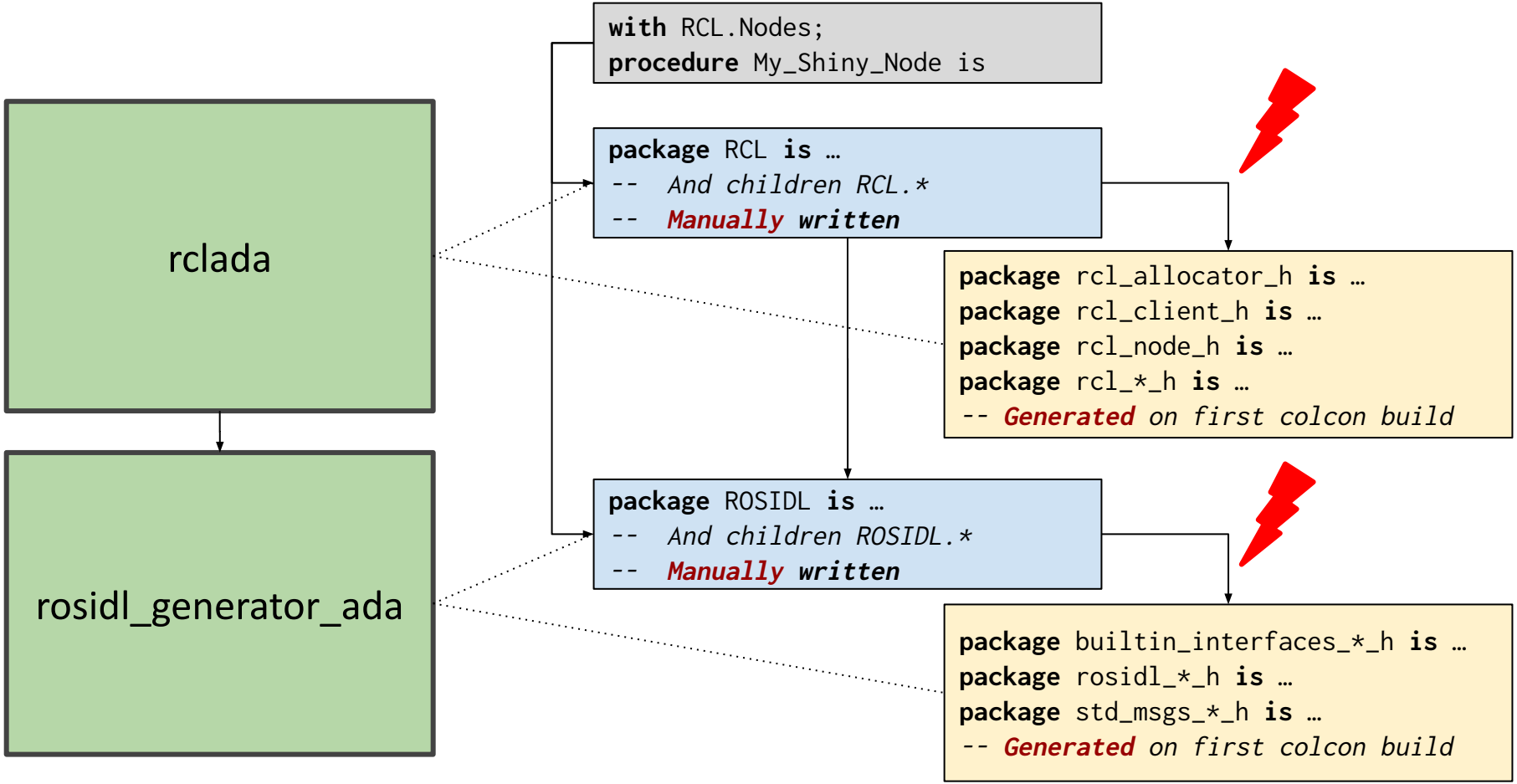
- Writing bindings:
  - Manual writing
    - 🙂 No need to be exhaustive
    - 🙂 High quality (thick binding)
    - 😕 More effort
    - 😕 May become de-sync'd
  - Automated generation
    - 🙂 "Less" work
    - 🙂 Completeness
    - 🙂 Assured consistency
    - 😕 Lower quality (thin binding)
    - 😕 Might not compile
- Ada/GNAT support:
  - Annex B: interface to other languages
    - C/C++, Fortran, Cobol
  - **`gcc -fdump-ada-spec file.h`**

```
/* C prototype */
int initialize(options_t *opts,
               char *argv[]);
```

```
-- Ada automatic binding
function Initialize
   (opts : access Options_T;
    argv : System.Address)
    return Interfaces.C.int
    with Import, Convention => C;
```

```
-- Ada manual binding
type Arg_Array is
  array (Natural range <>) of aliased
  Interfaces.C.Strings.Chars_Ptr
  with Convention => C;

function Initialize
   (opts : in out Options_T;
    argv :        Arg_Array)
    return Interfaces.C.int
    with Import, Convention => C;
```

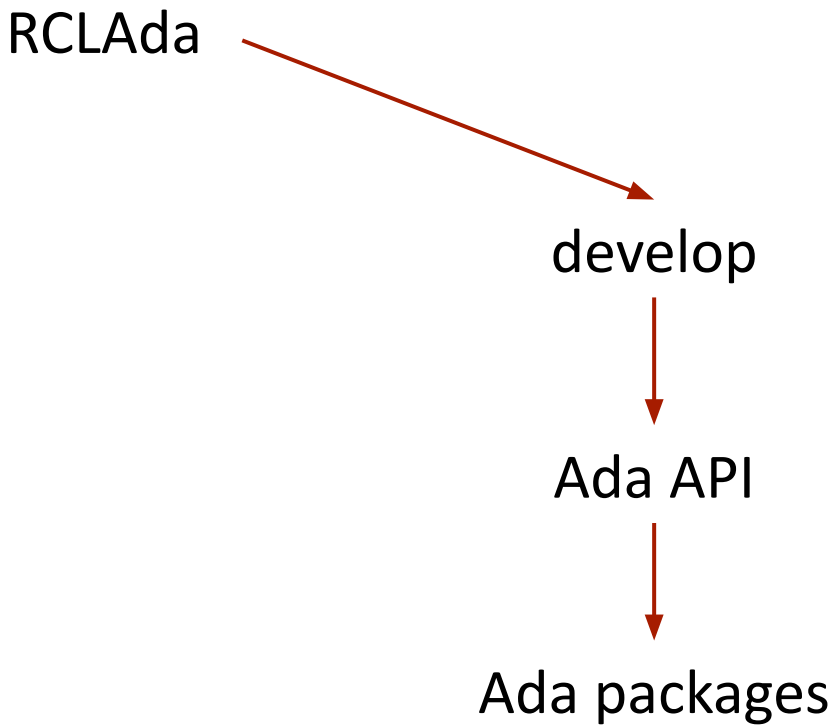
RCLAda: leverage *colcon* for best of both worlds



RCLAda API highlights

https://ada-ros.github.io/ada4ros2/

Structure



RCLAda completeness

Main features:

- `RCL.Nodes` : Complete
- `RCL.Publishers` : Complete
- `RCL.Subscriptions` : Complete
- `RCL.Clients` : Complete
- `RCL.Services` : Complete
- `RCL.Actions` : Partial (thin binding only ATM)
- `RCL.Parameters` : Pending

Support:

- `RCL.Allocators` : Complete
- `RCL.Calendar` : Complete
- `RCL.Executors` : Complete
- `RCL.Graph` : Complete
- `RCL.Options` : Partial (only QoS predefined profiles)
- `RCL.Timers` : Complete
- `RCL.Wait` : Complete

Dynamic access (through introspection):

- Typesupport: Complete
- Simple types: Complete
- Nested types: Complete
- Array types: Complete
- Matrix types: Complete

Static access (through generated types):

- Typesupport: Complete
- Simple types: Complete
- Nested types: Complete
- Array types: Complete
- Matrix types: Deprecated by ROS2

RCLAda API

- rcl.gpr
  - RCL.*

rclada

- rosidl.gpr
  - ROSIDL.*

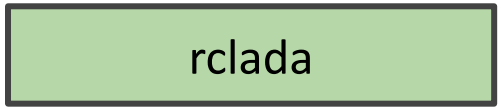
rosidl_generator_ada

- ros_tf2.gpr
  - RCL.TF2.*

rclada_tf2

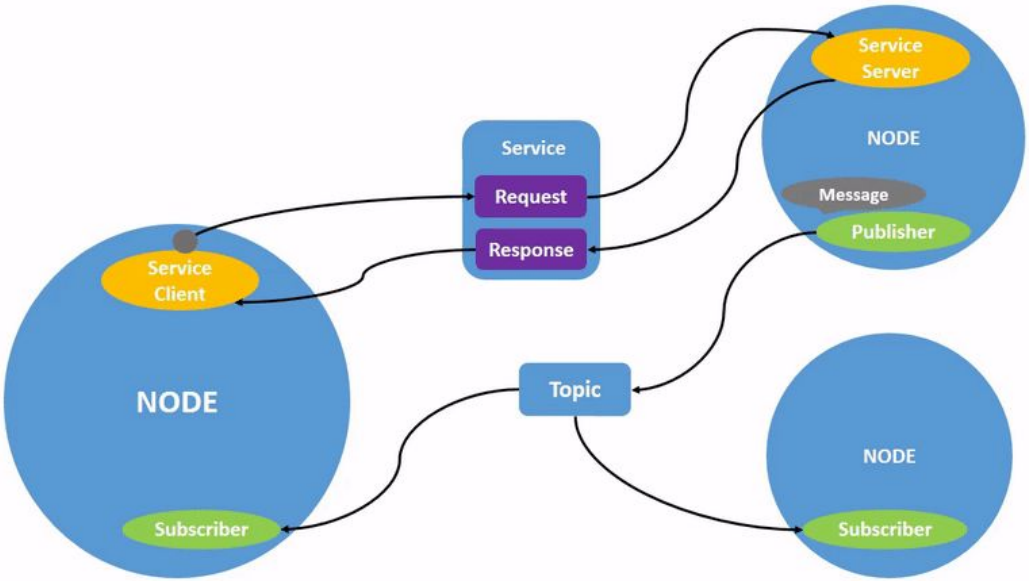
RCLAda API: RCL

- rcl.gpr
  - RCL.Nodes
    - Declare and initialize Node
  - RCL.Logging
    - Emit messages in ROS2 logging format
    - Ada.Text_IO works just the same
  - RCL.Publishers
    - Publish to a topic
    - Requires a Node
  - RCL.Subscribers
    - Listen to a topic



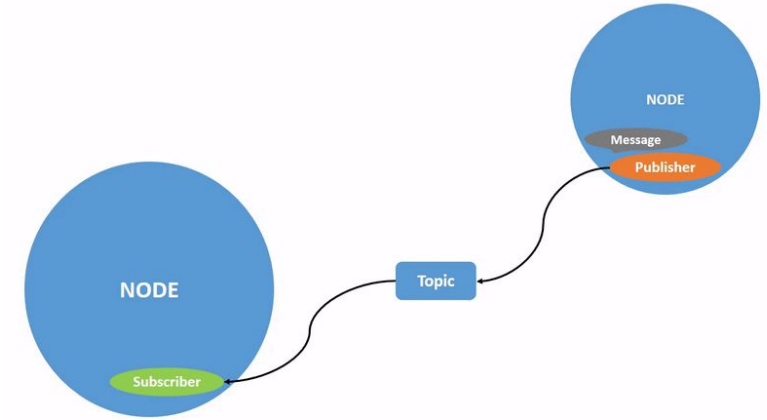
Communications

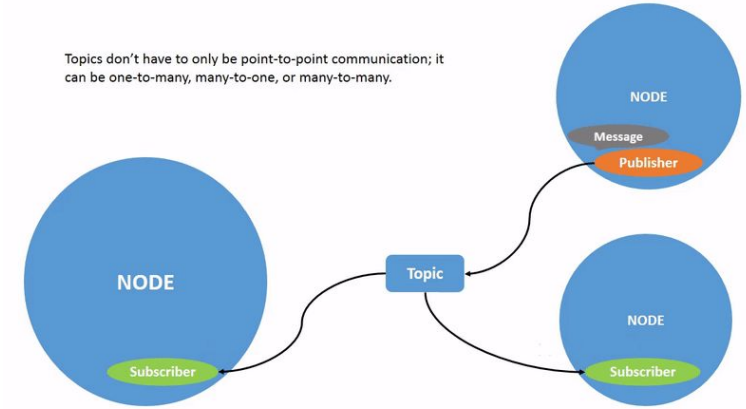
- Topics
  - Message
- Services
  - Request
  - Response
- Actions
  - Goal
  - Feedback
  - Result



TOPICS

- Topics
  - Message
    - ROSIDL.Dynamic.Message
    - **ROSIDL.Static.***
  - Publisher
    - **RCL.Nodes.[Typed_]Publish**
    - RCL.Publishers
  - Subscriber
    - **RCL.Nodes.[Typed_]Subscribe**
    - RCL.Subscribers

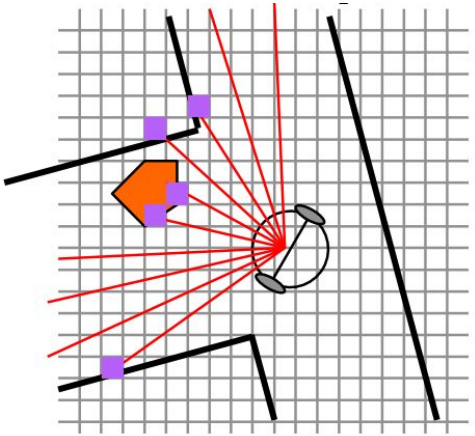




https://docs.ros.org/en/foxy/Tutorials/Topics/Understanding-ROS2-Topics.html

ROS2 IDL message definitions

```
# LaserScan.msg: Single scan from a planar laser range-finder

std_msgs/Header header  # timestamp in header is the acquisition time-axis

float32 angle_min       # start angle of the scan [rad]
float32 angle_max       # end angle of the scan [rad]
float32 angle_increment # angular distance between measurements [rad]

float32 time_increment  # time between measurements [seconds]

float32 scan_time       # time between scans [seconds]

float32 range_min       # minimum range value [m]
float32 range_max       # maximum range value [m]

float32[] ranges        # range data [m]
float32[] intensities   # intensity data [device-specific units].  If your
                        # device does not provide intensities, please leave
                        # the array empty.
```



ROS2 basic types (ROSIDL.Types / RCL.Types)

```
type Bool is new C.Extensions.bool;

type Byte is new Natural range 0 .. 2**8 - 1
  with Convention => C,
       Size       => 8;

type Char is new C.Char;

type Float32 is new C.C_Float;
type Float64 is new C.Double;

type Int8  is new C.Signed_Char;
type Uint8 is new C.Unsigned_Char;

type Int16  is new C.Short;
type Uint16 is new C.Unsigned_Short;

type Int32  is new C.Int;
type Uint32 is new C.Unsigned;

type Int64  is new C.Long;
type Uint64 is new C.Unsigned_Long;
```

```
type ROS_String is private;
--  Matches a ROS2 C type

function Capacity (Str : ROS_String)
                   return Natural;

function Get_String (Str : ROS_String)
                     return String;

procedure Set_String
            (Str  : in out ROS_String;
             Text : String);
```

Dynamic Message Usage

rosidl_generator_ada

```
declare
   Support : ROSIDL.Typesupport.Message_Support :=
             ROSIDL.Typesupport.Get_Message_Support
               (Pkg_Name, Msg_Type);
   Msg : ROSIDL.Dynamic.Message := Init (Support);
begin
   Msg ("valid").As_Bool := True;
   Msg ("X").As_Float32  := 1.0;
   --  Individual values

   Msg ("Values").As_Array (42).As_Int8 := 0;
   --  Array indexing

   Msg ("Image").As_Matrix ((100, 50, 1)).As_Int8 := 0;
   --  Matrix indexing
end;
```

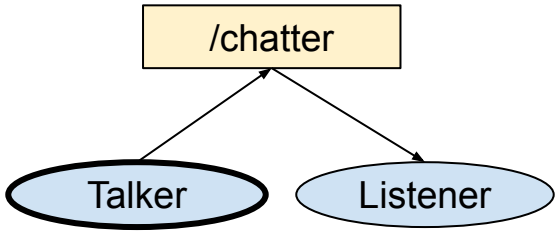
Obtain message type

Reference to fields
- No data copy
- Type-checked

1D vector indexing
- Bounds checked

Matrix indexing
- Tuple of indices
- Dimensions checked

Task 2.1
Publisher, dynamic version



TALKER

```
procedure Sol_Publisher_Dynamic is

   Node : Nodes.Node := Nodes.Init;                                  -- Node in the stack

   Msg_Type : constant ROSIDL.Typesupport.Message_Support :=
                 ROSIDL.Typesupport.Get_Message_Support ("std_msgs",  -- Message type at runtime
                                                         "String");

   Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init (Msg_Type);   -- Actual message allocation

   Pub : Publishers.Publisher := Node.Publish (Msg_Type,             -- Topic binding/creation
                                               "/chatter");

begin
   for Count in Positive'Range loop
      Msg ("data").Set_String ("Hello from the Ada side" & Count'Image);  -- Assign string
      Pub.Publish (Msg);

      Logging.Info ("I just said: " & Msg ("data").Get_String);    -- Standard ROS2 logging

      delay 1.0;                                                   -- Once per second
   end loop;
end Sol_Publisher_Dynamic;
```

Tasks 2.1-2.4 hints

- 2.1+2.4 or 2.2+2.3

- No need to import new interfaces in CMakeLists.txt
  - Because we are using std_msgs/String
  - Already imported by `rclada` ROS2 package
  - It could be re-imported if we were not aware

- CMake needs (if starting from scratch):
  - `find_package(ROS2_PACKAGE_NAME)`
  - `ada_begin_package()`
  - `ada_add_executables(TARGET SRCDIR DSTDIR EXECUTABLES...)`
  - `ada_end_package()`

- Testing
  - ROS2 command-line tools
  - demo_nodes_cpp nodes
  - with each other

ROS2 helper tools

- Command-line
  - **Everything with tab completion**
  - `ros2`
  - `ros2 {node|topic}`
  - `ros2 topic {list|info}`
  - `ros2 topic {echo|pub}`
- GUI
  - `rqt`
  - `rqt_graph`

rosidl_generator_ada: static message generation

```
# LaserScan.msg

std_msgs/Header header

float32 angle_min
float32 angle_max
float32 angle_increment

float32 time_increment

float32 scan_time

float32 range_min
float32 range_max

float32[] ranges
float32[] intensities
```

```
type Message is limited record
   Header          : Std_Msgs.Messages.Header.Message;
   Angle_Min       : Types.Float32;
   Angle_Max       : Types.Float32;
   Angle_Increment : Types.Float32;
   Time_Increment  : Types.Float32;
   Scan_Time       : Types.Float32;
   Range_Min       : Types.Float32;
   Range_Max       : Types.Float32;
   Ranges          : Types.Float32_Sequence;
   Intensities     : Types.Float32_Sequence;
end record
  with Convention ⇒ C;
```

Std_Msgs.Messages.LaserScan

```
type Message is limited record
   Stamp    : Builtin_Interfaces.Messages.Time.Message;
   Frame_Id : aliased Types.ROS_String;
end record
```

Std_Msgs.Messages.Header

Generating Ada types

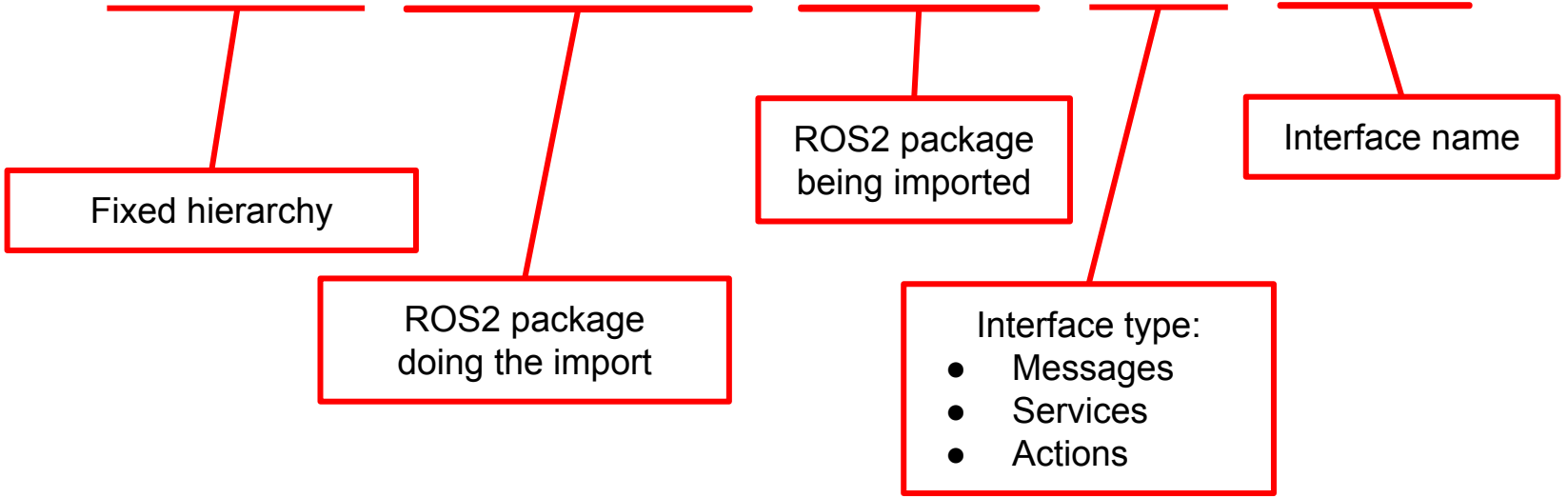
- package.xml
  - `<depend>`**`rclada_common`**`</depend>`
  - `<depend>`**`rosidl_generator_ada`**`</depend>`
  - `<depend>`**`geometry_msgs`**`</depend>`

- CMakeLists.txt
  - **`ada_import_interfaces`**`(geometry_msgs)`

- CMake triggers the Ada generator in rosidl_generator_ada:
  - `ros2_interfaces_<pkg name>.gpr`
  - `ROSIDL.Static.Pkg_Name.Geometry_Msgs.*;`

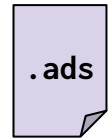
Raw C messages / Managed Ada messages

```
package ROSIDL.Static.Tutorial_Solutions.Geometry_Msgs.Messages.Posestamped is
```



Generated Ada types corresponding to C structs

```
package ROSIDL.Static.Tutorial_Solutions.Geometry_Msgs.Messages.Posestamped is

   type Message is limited record
      Header : Std_Msgs.Messages.Header.Message;
      Pose   : Geometry_Msgs.Messages.Pose.Message;
   end record
     with Convention ⇒ C;

   package Handling is new
     ROSIDL.Static.Message
       (Pkg       ⇒ "geometry_msgs",
        Name      ⇒ "PoseStamped",
        Part      ⇒ ROSIDL.Message,
        C_Message ⇒ Message);
```



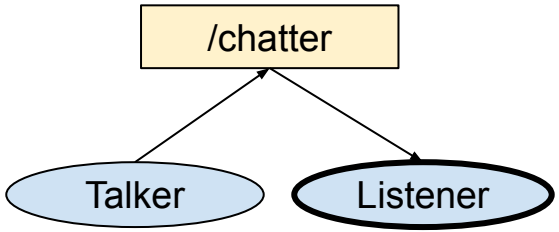
“Handling” package

```
package ROSIDL.Static.Tutorial_Solutions.Geometry_Msgs.Messages.Posestamped.Handling is

   Support : constant Typesupport.Message_Support;
   --  Support to create messages of this type


   type Message is new Wrapped_Message with private;
   --  Use this type for RAII managed memory

   function Data (This : Message) return C_Message_Access;
   --  Access the underlying fields in the raw C C_Message type

   function Dynamic (This : Message) return ROSIDL.Dynamic.Message;
   --  Access the same message via introspection
```

package RCL.Logging is

```
procedure Set_Level (Severity : Levels;
                     Name     : String := "");
--  Sets the minimum level that will be output.

procedure Info (Message  : String;
                Locate   : Boolean      := False;
                Location : Log_Location := Logging.Location;
                Name     : String       := "");

procedure Warn (Message  : String;
                Locate   : Boolean      := False;
                Location : Log_Location := Logging.Location;
                Name     : String       := "");
```

Task 2.4
Subscriber, statically typed version



LISTENER

```
with RCL.Logging;
with RCL.Nodes;

with ROSIDL.Static.Rclada.Std_Msgs.Messages.String;

use RCL;
use ROSIDL.Static.Rclada;

procedure Sol_Subscriber_Static is

   Node : Nodes.Node'Class := Nodes.Init;

   --------------
   -- Callback --
   --------------

   procedure Callback (Node : in out Nodes.Node'Class;
                       Msg  :        Std_Msgs.Messages.String.Message;
                       Info :        ROSIDL.Message_Info)
   is
      pragma Unreferenced (Node, Info);
   begin
      Logging.Info ("Radio chatter: " & Types.Get_String (Msg.Data));
   end Callback;

   ---------------
   -- Subscribe --
   ---------------

   procedure Subscribe is new Nodes.Typed_Subscribe
     (Handling ⇒ Std_Msgs.Messages.String.Handling,
      Callback ⇒ Callback);

begin
   Subscribe (Node, "/chatter");
   Node.Spin (During ⇒ Forever);
end Sol_Subscriber_Static;
```

-- Import of static message types

-- Callback uses actual message type
-- ROSIDL.Static.Tutorial_Solutions.Std_Msgs…

-- Access message data as regular record.

-- Create the instance with the Handling package

-- Spin the node to wait for events

Tasks 2.1-2.4 hints

- 2.1+2.4 or 2.2+2.3

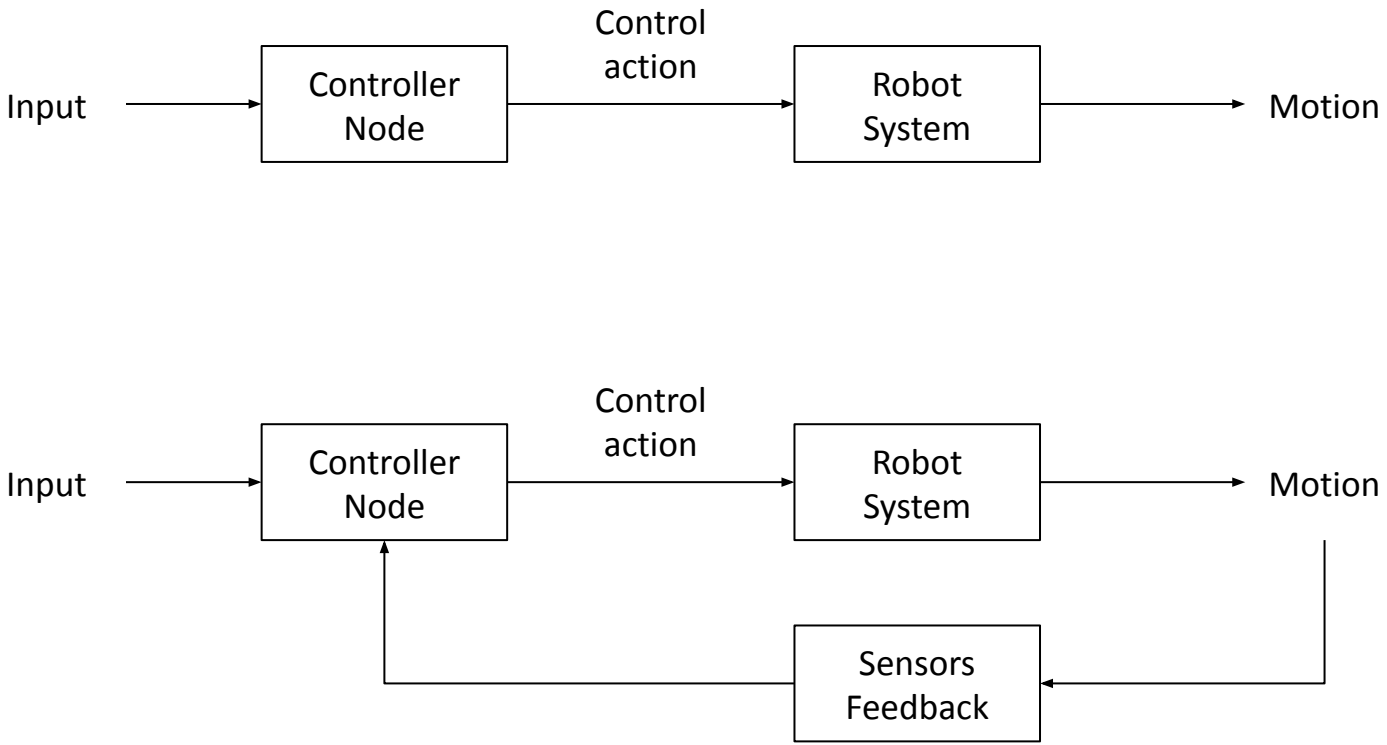
- No need to import new interfaces in CMakeLists.txt
  - Because we are using std_msgs/String
  - Already imported by `rclada` ROS2 package
  - It could be re-imported if we were not aware

- CMake needs (if starting from scratch):
  - `find_package(ROS2_PACKAGE_NAME)`
  - `ada_begin_package()`
  - `ada_add_executables(TARGET SRCDIR DSTDIR EXECUTABLES...)`
  - `ada_end_package()`

- Testing
  - ROS2 command-line tools
  - demo_nodes_cpp nodes
  - with each other

Tasks 2.5-2.8
Controlling simulated robots in open-loop

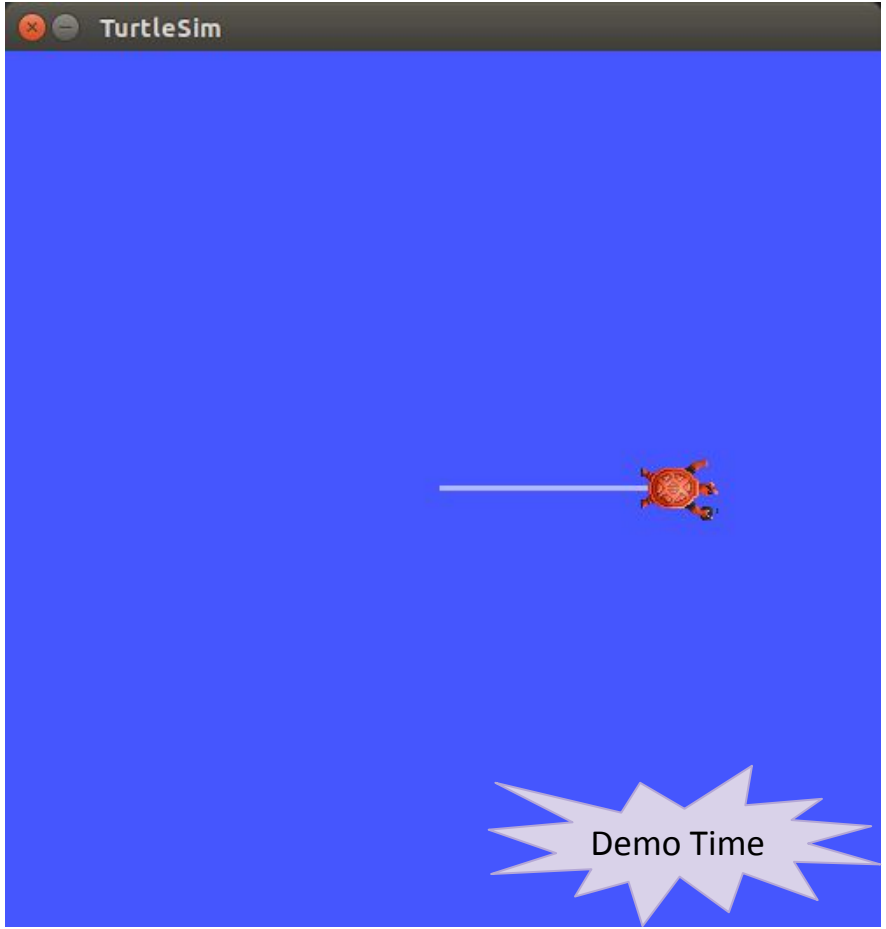
OPEN-LOOP vs CLOSED-LOOP



TurtleSim

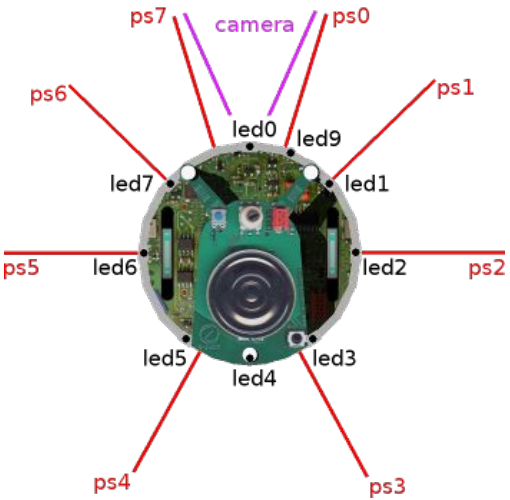
- LOGO homage
- ROS2 tutorials
- Task 2.5: launch and inspect
- Task 2.6: fixed sequence
- Task 2.7: keyboard remote

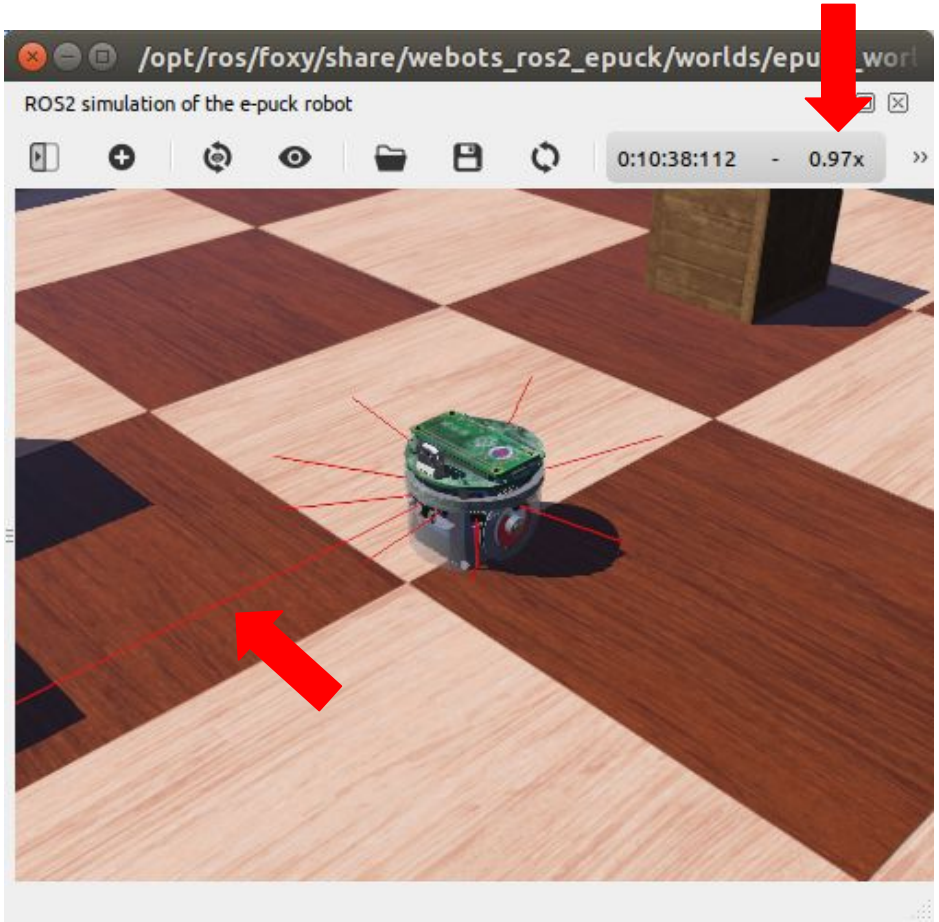
```
$ ros2 run turtlesim turtlesim_node
$ ros2 topic info -v /turtle1/cmd_vel
$ ros2 topic pub /turtle1/cmd_vel …
$ ros2 topic echo /turtle1/pose
$ ros2 run turtlesim turtle_teleop_key
```



Webots ePuck

```
$ ros2 launch \
  webots_ros2_epuck \
  robot_launch.py
```





https://github.com/cyberbotics/epuck_ros2

- Task 2.8: keyboard remote
- Light leds

Tasks 2.6-2.8 hints

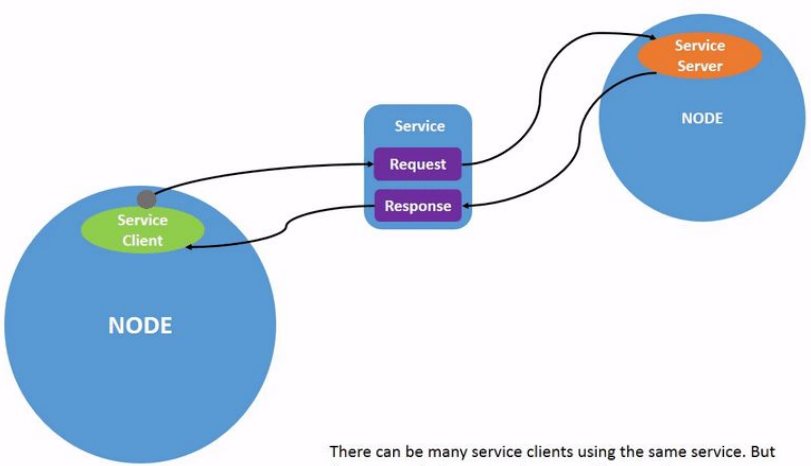
- In package.xml:
  - `<depend>???_msgs</depend>`
- In CMakeLists.txt:
  - `find_package(???_msgs)`
  - `ada_import_interfaces(???_msgs)`
    - Verify build up to this point, and static message types existence under `./build`
  - `ada_add_executables(…)`
    - Verify build up to this point, runnable with `$ ros2 run package_name exec_name`
- In GPR project:
  - `with "ros2_interfaces_[package name].gpr";`
- In Ada sources:
  - `with ROSID.Static.Messages.Package_Name.???_Msgs.Messages.???;`

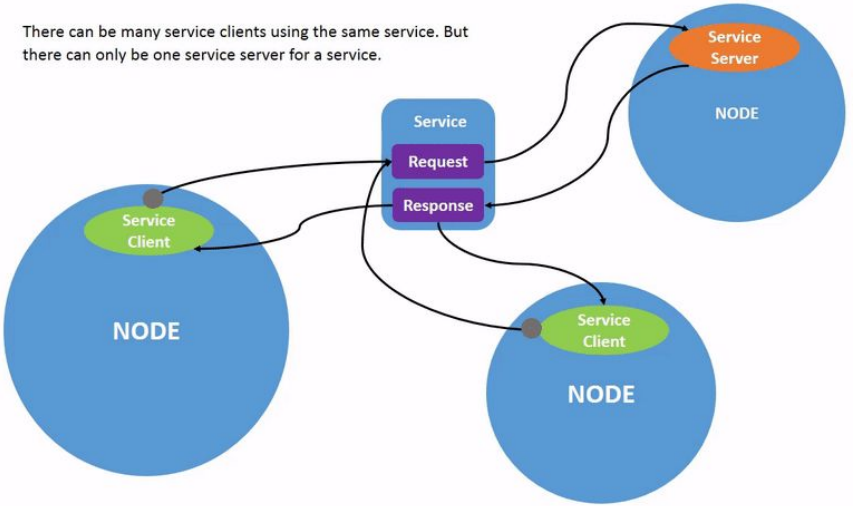
Task 3: client/server communication

SERVICES

- Services
  - Request
  - Response

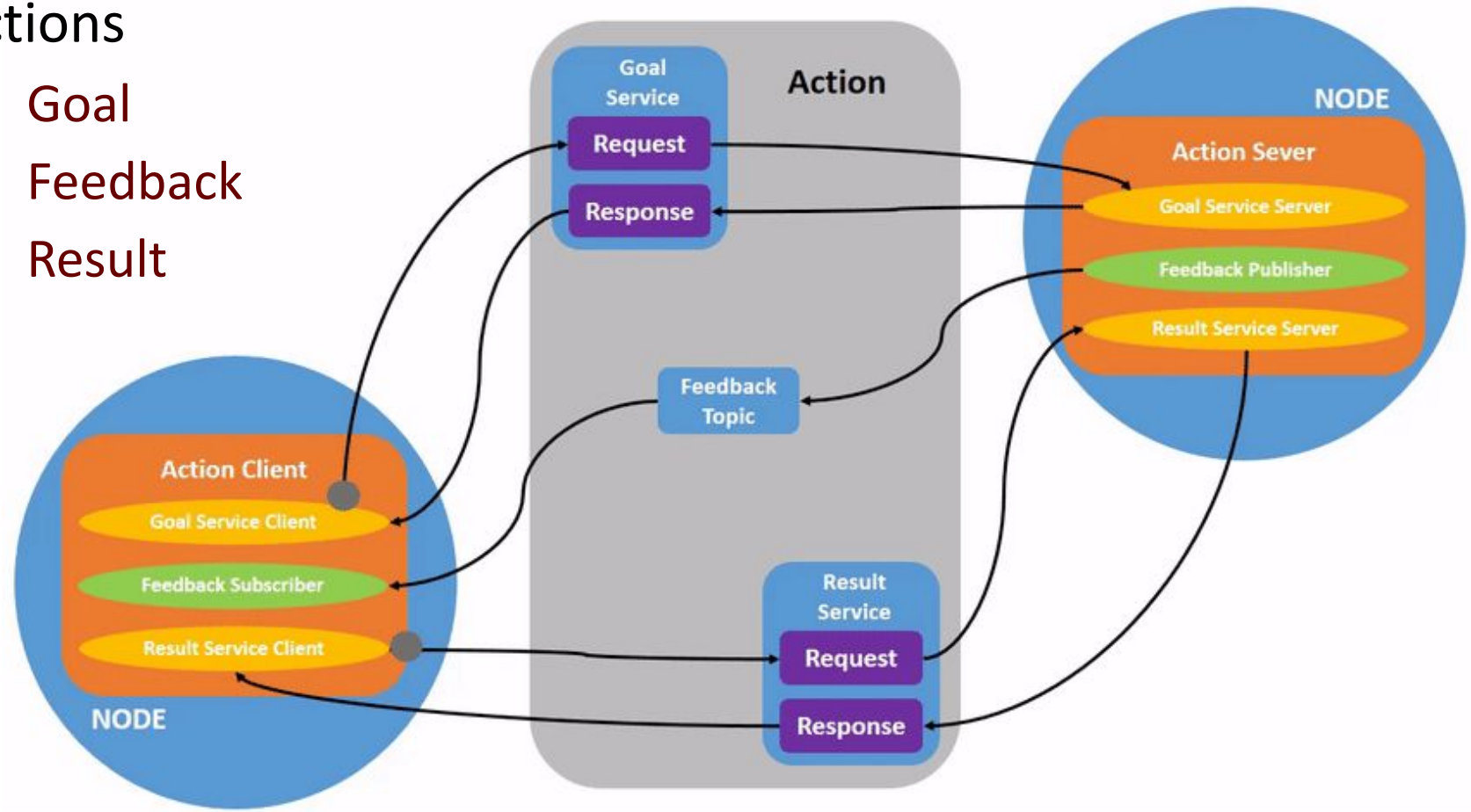
  - RCL.Nodes.[Typed_]Serve
  - RCL.Services





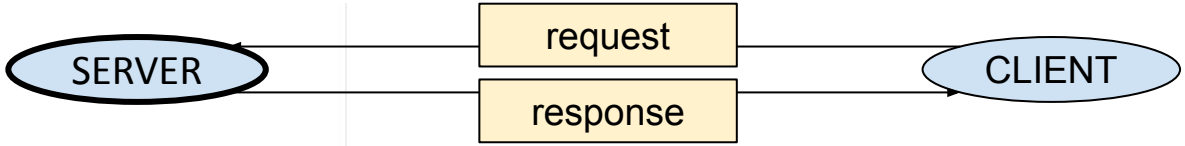
ACTIONS

- Actions
  - Goal
  - Feedback
  - Result



Services: server

```
procedure Sol_Server_Example is

   Node : Nodes.Node'Class := Nodes.Init;

   -----------------------
   -- Answering_Machine --
   -----------------------

   procedure Answering_Machine
     (Node     : in out Nodes.Node'Class;
      Request  :        Std_Srvs.Services.Trigger.Handling.Request_Raw_Message;
      Response : in out Std_Srvs.Services.Trigger.Handling.Response_Raw_Message)
   is
      pragma Unreferenced (Node, Request);
   begin
      Logging.Info ("Incoming call!");
      Response.Success := Types.Bool (True);
      Types.Set_String (Response.Message, "Ada rocks!");
   end Answering_Machine;

   package Answering_Machine_Instance is new Nodes.Typed_Serve
      (Handling ⇒ Std_Srvs.Services.Trigger.Handling,
       Node     ⇒ Node,
       Name     ⇒ "/ada_service",
       Callback ⇒ Answering_Machine);

   pragma Unreferenced (Answering_Machine_Instance);

begin
   Logging.Info ("Ready to answer");
   Node.Spin (During ⇒ Forever);
end Sol_Server_Example;
```

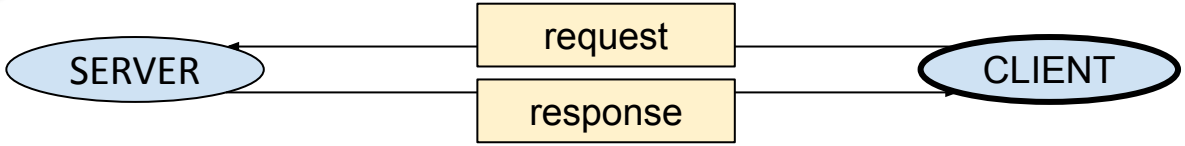


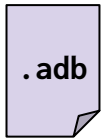
-- Request (filled by the client)
-- Response (to be filled by the service)

-- Instance using service Handling package

Services: client (blocking)

```
procedure Sol_Client_Sync is

   Node : Nodes.Node := Nodes.Init;

   Request : Std_Srvs.Services.Trigger.Handling.Request_Message;

   function Caller is new Nodes.Typed_Client_Call_Func
     (Std_Srvs.Services.Trigger.Handling);                    -- Client with service type

begin
   Logging.Info ("Ada calling...");

   declare
      Response : constant Std_Srvs.Services.Trigger.Handling
                                  .Resp_Handling.Shared_Message :=
                    Caller (Node, "/ada_service",
                            Request,
                            Connect_Timeout => 5.0,
                            Timeout         => 10.0);        -- Timeout > Connect_Timeout
   begin
      Logging.Info ("Gossip is that... "
                    & Types.Get_String (Response.Message));  -- Call could be embedded here
   end;

end Sol_Client_Sync;
```





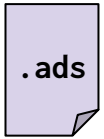
CLIENT CALL TIMEOUTS

Client timeouts guarantee

- Exception on timeout
- Callback not used after timeout
  - So it can be declared locally

Connect timeout vs regular

- Connect: time until service is available
- Regular: time until response is received

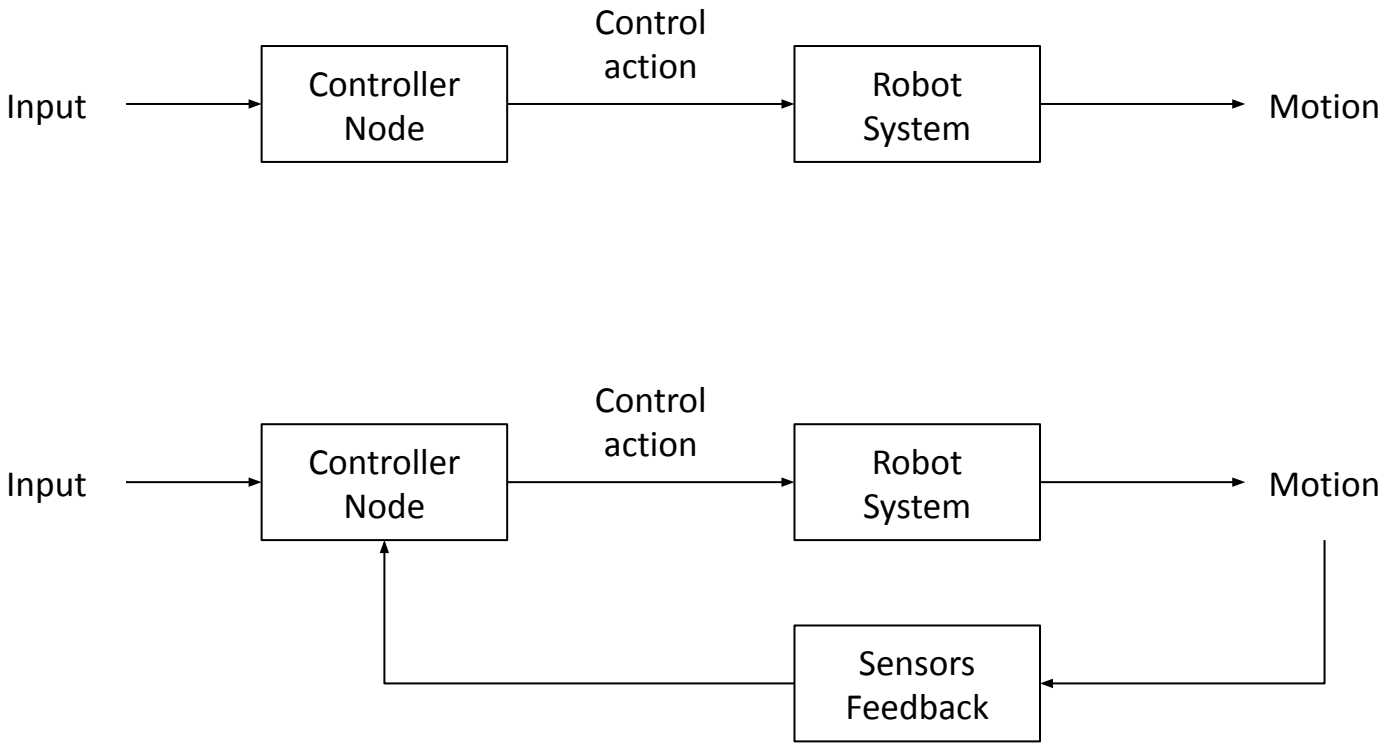


Tasks 3.1-3.4 hints

- In package.xml:
  - `<depend>???_msgs</depend>`
- In CMakeLists.txt:
  - `find_package(???_msgs)`
  - `ada_import_interfaces(???_msgs)`
    - Verify build up to this point, and static message types existence under `./build`
  - `ada_add_executables(…)`
    - Verify build up to this point, runnable with `$ ros2 run package_name exec_name`
- In GPR project:
  - `with “ros2_interfaces_[package name].gpr”;`
- In Ada sources:
  - `with ROSID.Static.Messages.Package_Name.???_Msgs.Services.???;`

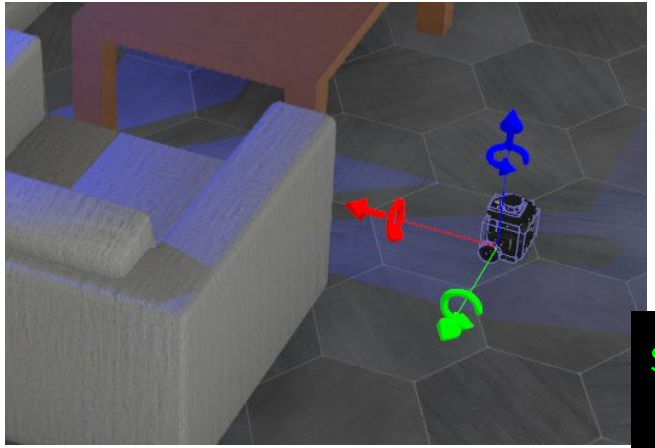
Part 3: Robotics specifics

OPEN-LOOP vs CLOSED-LOOP



Simulated robots

https://cyberbotics.com/doc/guide/turtlebot3-burger

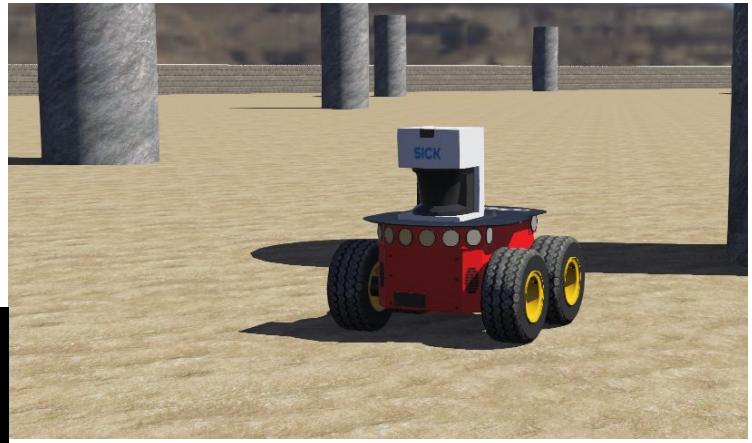


```
$ ros2 launch \
  webots_ros2_NAME \
  robot_launch.py
```

turtlebot

https://cyberbotics.com/doc/guide/pioneer-3at

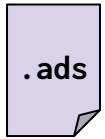
p3at

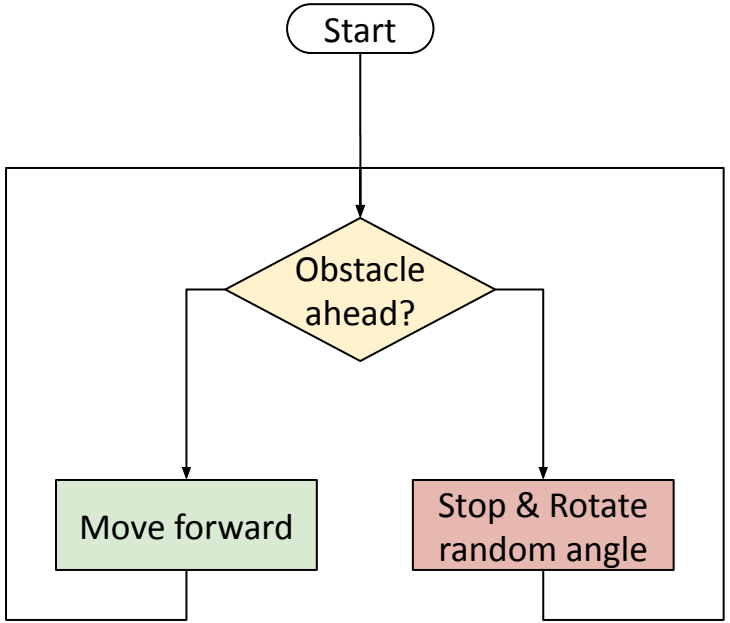


Tasks 4.1-4.2: Wandering Robots

Roomba-like simple movement

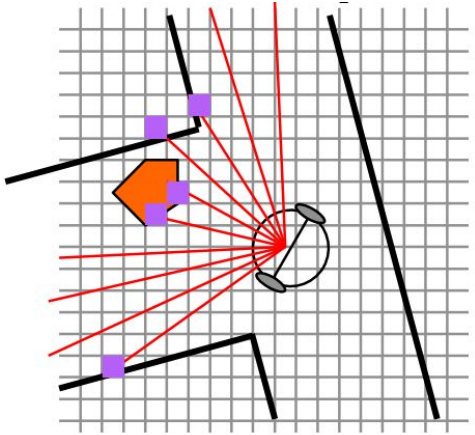
- Inputs
  - /tof
  - /scan
    - QoS in TurtleBot
- Outputs
  - /cmd_vel



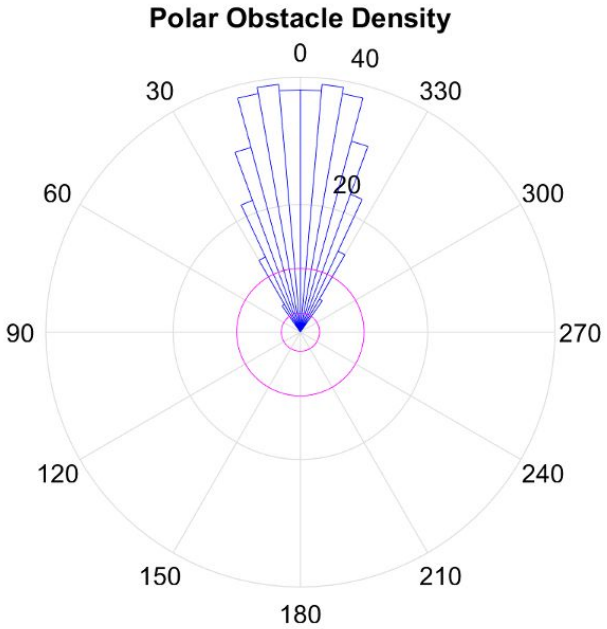


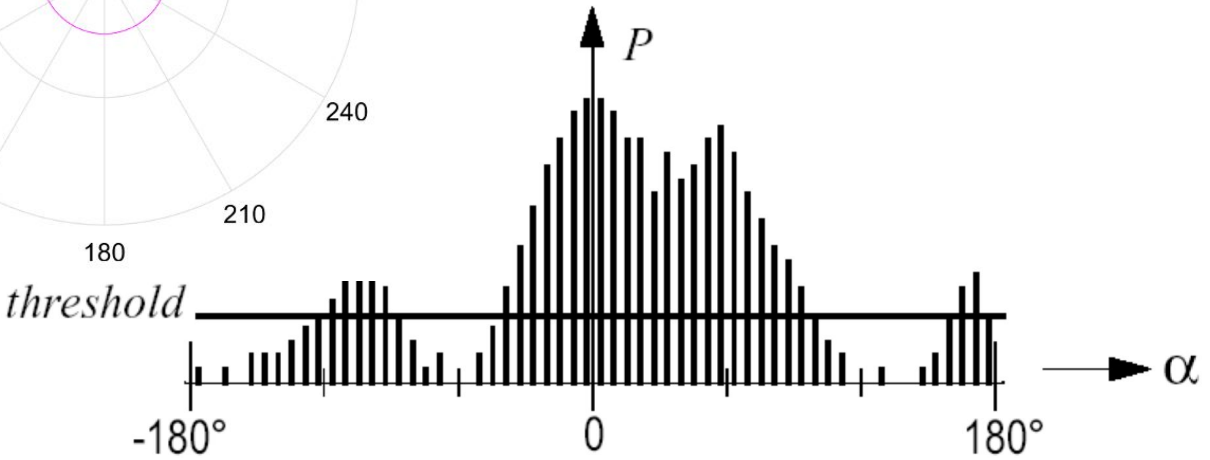
Tasks 4.3: VFH Navigation

VECTOR FIELD HISTOGRAM



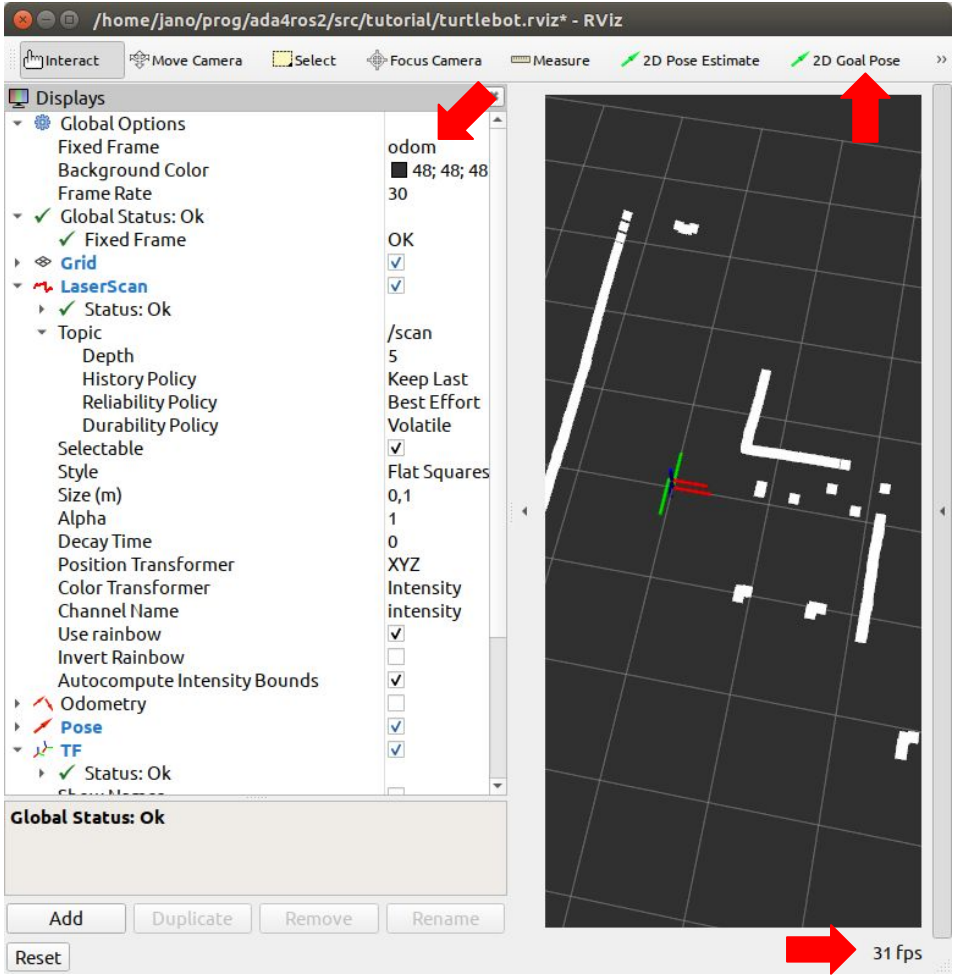
Adapted from © R. Siegwart, I. Nourbakhsh.
Source: https://es.mathworks.com/help/nav/ug/vector-field-histograms.html
Source: Lynn E. Parker,
http://web.eecs.utk.edu/~leparker/Courses/CS594-fall08/Lectures/Oct-23-ObstAvoid-II+Architectures.pdf





RVIZ2

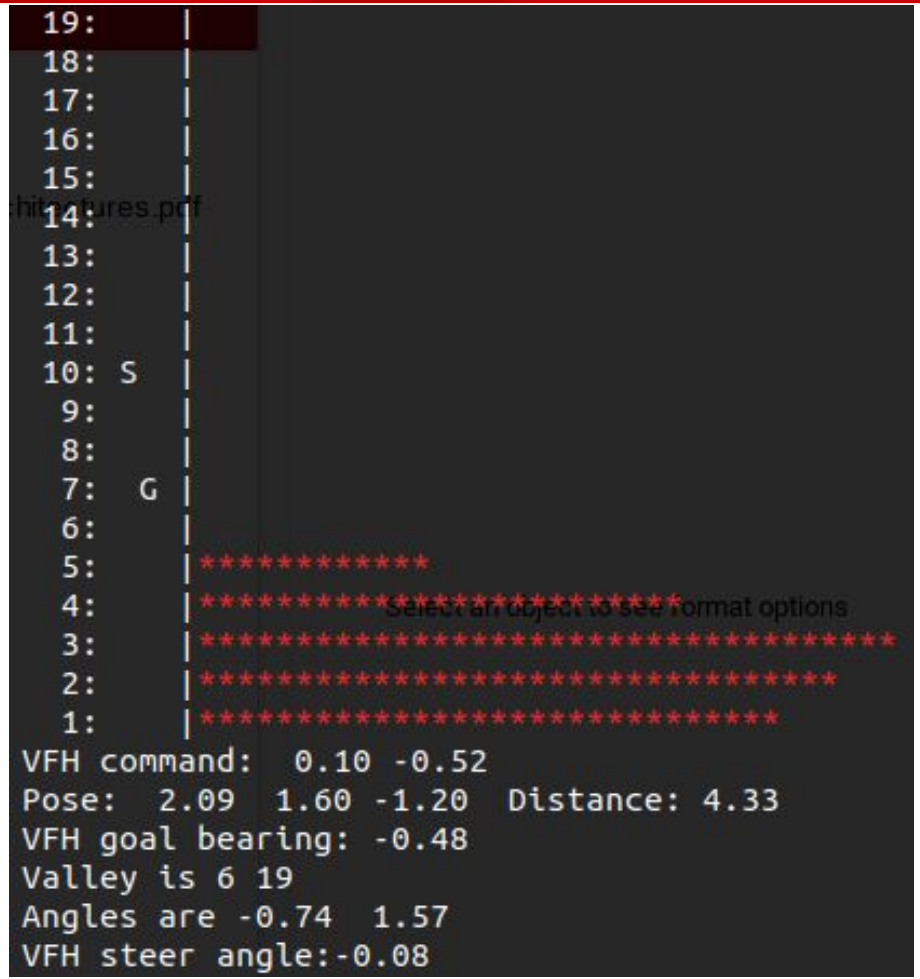
- Configurable visualization tool
  - `$ rviz2`
  - ./src/tutorial/turtlebot.rviz
- Needs a “global” frame
  - Usually provided by a map
  - /odom for this tutorial
- Can publish points
  - /goal_pose
- Check FPS if slow
  - `$ export LIBGL_ALWAYS_SOFTWARE=1`



VFH local navigation

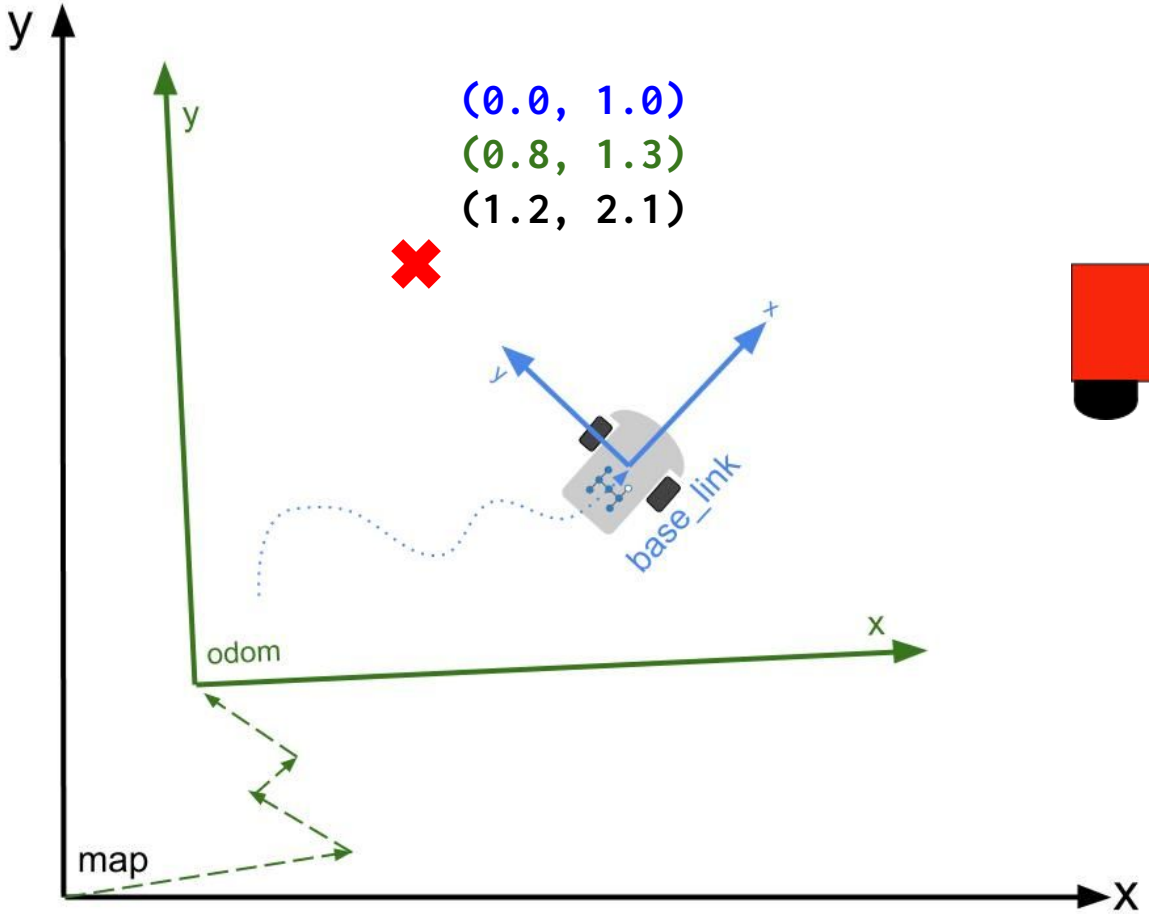
- Inputs
  - /goal_pose
  - /scan
  - /odom
- Output
  - /cmd_vel
- VFH.Steer
  - In package tutorial_common
  - Compute a velocity command

```
function Steer
  (Odom,
   Goal   : Pose2D;
   Scan   : Sensor_Msgs.Messages.Laserscan.Message;
   Params : Parameters := (others ⇒ <>))
   return Velocity2D;
```

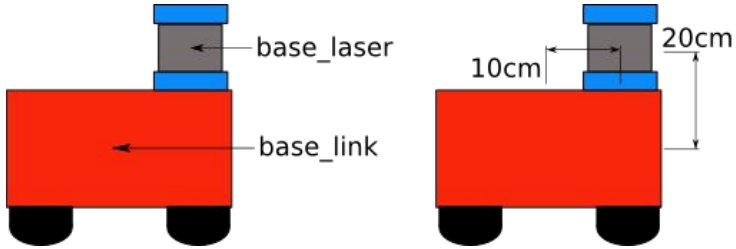


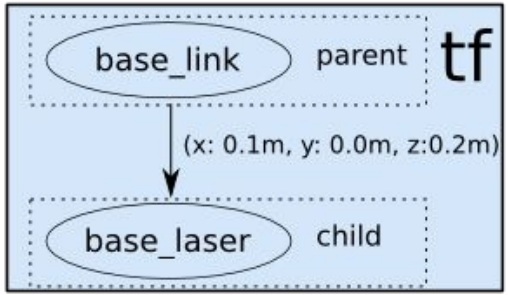
Tasks 4.4: VFH + TF correction

REFERENCE FRAMES



https://blog.hadabot.com/ros2-navigation-tf2-tutorial-using-turtlesim.html



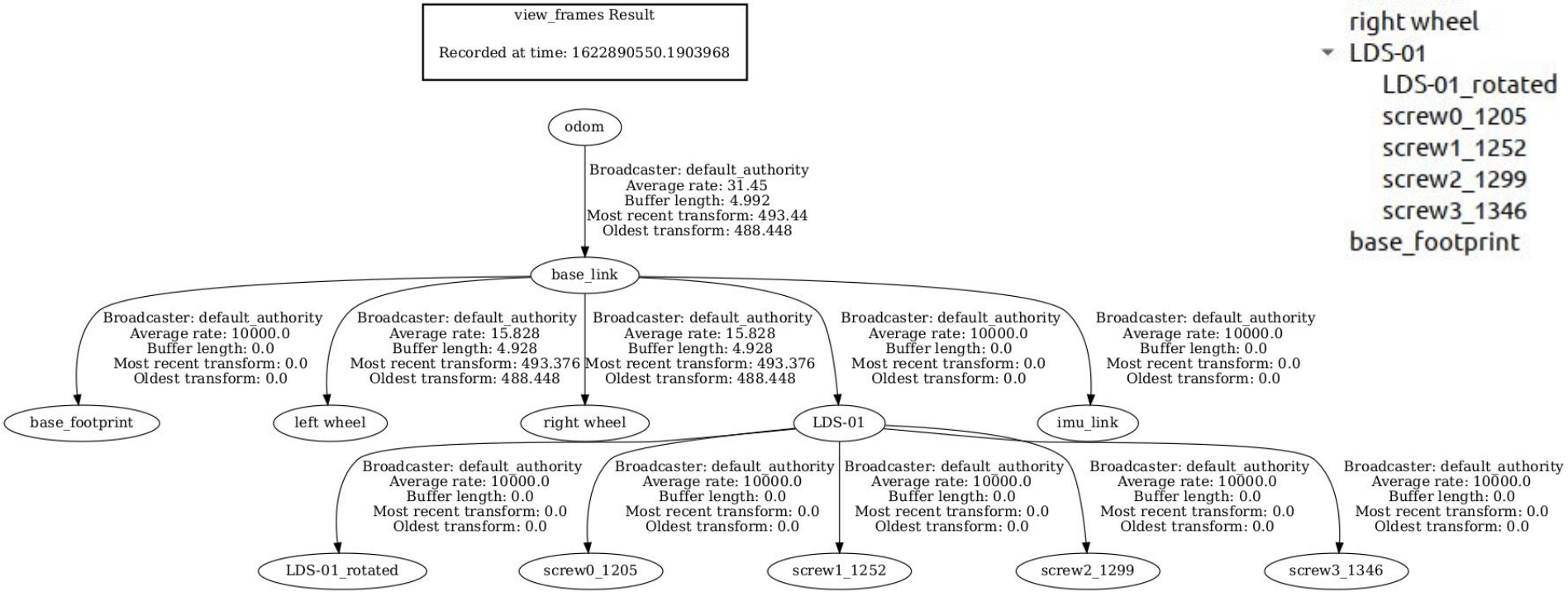


https://navigation.ros.org/setup_guides/transformation/setup_transforms.html

Transformation tree

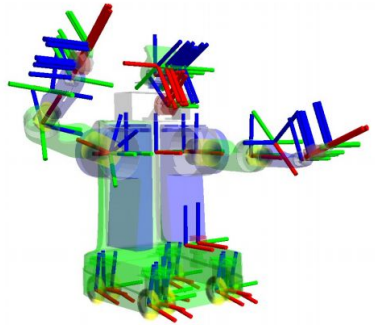
```
$ ros2 run tf2_tools view_frames.py
$ xdg-open frames.pdf
```

- Tree
  - odom
    - base_link
      - imu_link
      - left wheel
      - right wheel
      - LDS-01
        - LDS-01_rotated
        - screw0_1205
        - screw1_1252
        - screw2_1299
        - screw3_1346
      - base_footprint



RCL.TF2 package

```
function Can_Transform (From, Into : String) return Boolean;

function Transform (Point : Point3D;
                    From,
                    Into  : String)
                    return Point3D;
--  May raise if either the transformation is unknown or else the node is
--  shutting down.

procedure Publish_Transform
  (From, Into  : String;
   Translation : TF2.Translation;
   Rotation    : TF2.Euler;
   Static      : Boolean := False);
--  A static transform is published via static broadcast, and is not
--  expected to change (fixed parts of robots).
```



RCLAda trivia

Everything on the stack: Ada indefinite types

```
declare
   type Int_Array is array (Positive range <>)
                          of Integer;

   Arr   : Int_Array (1 .. 100);
   Hello : constant String := "Hello";

   Other_Arr : Int_Array (1 .. Get_Elsewhere);
begin
   -- Variable stack use so measure it or limit it!
end;

declare
   type Unconstrained (Length : Natural) is record
      Name : String (1 .. Length);
   end record;

   U1 : constant Unconstrained := Get_Unconstrained;
   U2 :          Unconstrained (10);
begin
```

Indefinite type *(unknown size at compile time)*
- **but definite values! *(known size at runtime)***

Constrained by declaration with static size

Constrained by initialization with static size

Constrained by declaration with unknown size at compile time

RCLAda does not use heap allocations

Constrained by initialization with unknown size

Constrained by declaration with static size

ROS2 allocators ⇔ Ada storage pools

- Ada defines Storage_Pool type for different:
  - memory areas (typical in some small boards)
  - allocation policies (including user-defined)
- ROS2 allocators mapped into Ada storage pools
  - Idiomatic use in Ada
  - Reuse of existing pools (e.g., GNAT.Debug_Pools)

```
$ rclada_test_allocators 1
Total allocated bytes:              2335
Total logically deallocated bytes:  2335
Total physically deallocated bytes: 0
Current Water Mark:                 0
High Water Mark:                    415

$ rclada_test_allocators 4
Total allocated bytes:              8095
Total logically deallocated bytes:  8095
Total physically deallocated bytes: 0
Current Water Mark:                 0
High Water Mark:                    415
```

```
type Int_Ptr is access Integer -- named pointer type
  with Storage_Pool => Deterministic_Pool;

type Node_Access is access all RCL.Nodes.Node'Class
  with Storage_Size => 0;       -- No heap allocations

pragma No_Allocators;
pragma No_Implicit_Heap_Allocations;
pragma No_Standard_Allocators_After_Elaboration;
pragma No_Standard_Storage_Pools;
--  See Restrictions in GNAT manual for many more
```

Allocator details

```
typedef struct rcutils_allocator_t
{
  void * (*allocate)(size_t size,
                     void * state);

  void (* deallocate)(void * pointer,
                      void * state);

  void * (*reallocate)(void * pointer,
                       size_t size,
                       void * state);

  void * (*zero_allocate)(size_t number_of_elements,
                          size_t size_of_element,
                          void * state);
                          void * state;
} rcutils_allocator_t;
```

```
package System.Storage_Pools is

   type Root_Storage_Pool is tagged private;

   procedure Allocate
     (Pool                     : in out Root_Storage_Pool;
      Storage_Address          :    out Address;
      Size_In_Storage_Elements : in     Storage_Count;
      Alignment                : in     Storage_Count)
   is abstract;

   procedure Deallocate
     (Pool                     : in out Root_Storage_Pool;
      Storage_Address          : in     Address;
      Size_In_Storage_Elements : in     Storage_Count;
      Alignment                : in     Storage_Count)
   is abstract;
```

```
Pool  : aliased GNAT.Debug_Pools.Debug_Pool;                    -- Ada pool, compiler provided
Alloc : aliased RCL.Allocators.Allocator (Pool'Access);         -- ROS2 allocator, wrapping Ada pool
Node  :         RCL.Node := Node.Init
                            (Options => (Allocator => Alloc'Access)); -- Set node allocator
```

Concurrent executors

```
package RCL.Executors.Concurrent is

   type Runner_Pool is array (Positive range <>) of Runner;
   -- Runner task type declaration omitted

   type Executor (Max_Nodes  : Count_Type :=
                     Default_Nodes_Per_Executor;
                  Queue_Size : Count_Type :=
                     Count_Type (System.Multiprocessors.Number_Of_CPUs) * 32;
                  Threads    : Positive :=
                     Positive (System.Multiprocessors.Number_Of_CPUs);
                  Priority   : System.Priority :=
                     System.Max_Priority) is
   new Executors.Executor (Max_Nodes) with
      record
         Pool  : Runner_Pool  (1 .. Threads);
         Queue : Queues.Queue (Capacity => Queue_Size,
                               Ceiling  => Priority);
         Started : Boolean := False;
      end record;

end RCL.Executors.Concurrent;
```

Parent abstract type in RCL.Executors

Task pool type

Executor type with discriminants

- **# of nodes supported**
- **Queue size**
- **Threads in the pool**
- **Priority**

System.* defined in ARM

OO derivation syntax

Members constrained by discriminants

- Standard Ada bounded queues
- All Ada bounded containers are stack based

See `rclada_test_multicore.adb`

- One producer
- Pooled consumers

rclada vs rclcpp

```
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

C++

```cpp
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

C++

```
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

C++

```
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

C++

```
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

C++

```cpp
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

C++

```cpp
class Talker : public rclcpp::Node
{
public:
 explicit Talker : Node("talker")
 {
   msg_ = std::make_shared<std_msgs::msg::String>();

   auto publish_message = [this]() -> void
     {
       msg_->data =
         "Hello World: " + std::to_string(count_++);
       pub_->publish(msg_);
     };

   pub_   = this->create_publisher
              <std_msgs::msg::String>(topic_name);
   timer_ = this->create_wall_timer(1s, publish_message);
 }
private:
  size_t count_ = 1;
  std::shared_ptr<std_msgs::msg::String> msg_;
  rclcpp::Publisher<std_msgs::msg::String>::SharedPtr pub_;
  rclcpp::TimerBase::SharedPtr timer_;
};

int main(int argc, char * argv[])
{
 auto topic = std::string("chatter");
 auto node = std::make_shared<Talker>(topic);
 rclcpp::spin(node);
}
```

```
procedure Talker is
  Support : constant ROSIDL.Typesupport.Message_Support :=
    ROSIDL.Typesupport.Get ("std_msgs", "String");

  Node : Nodes.Node             := Nodes.Init
                                     (Utils.Command_Name);
  Pub : Publishers.Publisher    := Node.Publish
                                     (Support, "/chatter");
  Msg : ROSIDL.Dynamic.Message := ROSIDL.Dynamic.Init
                                     (Support);

  Counter : Positive := 1;

  procedure Callback (Node    : in out Nodes.Node'Class;
                      Timer   : in out Timers.Timer;
                      Elapsed :        Duration)
  is
     Txt : constant String := "Hello World:" & Counter'Img;
  begin
     Msg ("data").Set_String (Txt);
     Pub.Publish (Msg);
     Counter := Counter + 1;
  end Callback;

begin
  Node.Timer_Add (1.0, Callback'Access);
  Node.Spin;
end Talker;
```

CONCLUSION

- CMake functions for build integration with system tools
  - Ada nodes have same standing as other nodes
- Ada API for pure Ada node writing
  - Ada nodes can interact with other language nodes
  - Plus standard interoperability with C/C++ sources
- RCLAda distinguishing features (vs rclcpp, rclpy, others)
  - No heap allocations
    - Guaranteed by language restrictions & libraries
  - Relies on automatic low-level binding
    - Early detection of mismatches on ROS2 API changes
  - Language ingrained in safety/HRT culture
    - All dynamic message data accesses are type and bounds checked (at runtime)
    - Static messages are size-checked during elaboration

THANKS FOR YOUR INTEREST

https://github.com/ada-ros/ada4ros2/

amosteo@unizar.es

@mosteobotic

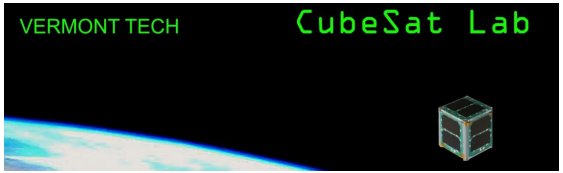
Centro Universitario de la Defensa Zaragoza
cud.unizar.es

Academia General Militar · Ctra. Huesca s/n · 50090 Zaragoza · 976 739 500

SPARK success stories

CubeSat from Vermont Tech

http://www.cubesatlab.org



- Three years of flight time (2013-2016)
- Others: 1x4month, 2x<1week, 8xUnheard of

IRONSIDES DNS server

`https://ironsides.martincarlisle.com/`

Proven free of (among others):

- Buffer overflows
- Integer overflows
- Information leaks
- Race conditions

SND navigation algorithm

`https://github.com/riveras/spark-navigation`

- IROS 2014 paper on errors in robotic navigation algorithms
- SND reimplemented in SPARK
- Proven without runtime errors

Tokeneer ID station project (NSA)

`https://www.adacore.com/tokeneer`

Not only SPARK but full development methodology

- Formal language (Z) for specification
- ~10KLOC
- 4 defects since delivery

Spark initial suggested adoption levels

- Stone level
  - Valid SPARK
- Bronze level
  - Initialization and correct data flow [target for most code]
- Silver level
  - Absence of run-time errors (AoRTE) [target for critical software]
- Gold level
  - Proof of key integrity properties [target for critical code sections]
- Platinum level
  - Full functional proof of requirements

AdaCore, Thales. "Implementation Guidance for the Adoption of SPARK." (2018): 39-52.

Ada numeric types

```
type Robot_ID is new Natural;                -- Type compatibility is by name
type Task_ID  is new Natural;                --
type Distance is range 0 .. 1_000_000_000; -- Explicit bounds

type Coordinate is range -180.0 .. 180.0;   -- Floating point with range

type Probability is digits 5                 -- Floating point with
                    range 0.0 .. 1.0;        -- minimum guaranteed precision

type Laser_Readings is delta 10.0 / 2**8     -- Binary fixed point
                       range 0.0 .. 10.0;

type Euros is delta 0.001 digits 12;         -- Decimal fixed point

type Weekdays is (Monday, Tuesday, Wednesday, Thursday, Friday);
type Escaped_Robot_Counter is array (Weekdays) of Natural
      with Default_Component_Value => 0;   -- Arbitrarily indexed arrays
```

Ada packages

Specification (*.ads)

```
package RCL.Logging is

   procedure Initialize;

   -- public methods

private

   -- protected methods
   -- 1-pass compiler needed info

end RCL.Logging;
```

Implementation (*.adb)

```
package body RCL.Logging is

   procedure Initialize is
   begin
      -- do whatever has to be done
   end Initialize;

   -- private methods

end RCL.Logging;
```

Main procedure (*.adb)

```
with RCL.Logging;

procedure RCL.Talker is
begin
   Logging.Initialize;
end RCL.Talker;
```

Hello, Ada

- Comparable in purpose to C++
  - **Emphasis in**
    - **Maintainability**
    - **Correctness**
    - **Early error detection**
- Structured
  - **Separate specifications**
- Strongly, statically typed
  - Named types (even pointers)
- Imperative (Pascal-like)
  - Object oriented, optionally
- **High-level concurrency**
  - **Tasks, Rendezvous, Monitors**
- **Design-by-contract**
  - **Pre-, post-conditions**
  - **Type predicates, type/loop invariants**

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Hello is
begin
  Put_Line ("Hello, ROSin!");
end Hello;
```

```
type Speeds  is range 0.0 .. 999.9;
type Lengths is new Float;

spd : Speeds  := 0.0;
len : Lengths := spd; -- Bzzzt
```

```
procedure Inc (X : in out Integer)
  with Pre => X < Integer'Last;

type Prime is new Positive
   with Predicate =>
     (for all D in 2 .. Prime / 2 =>
         Prime mod D /= 0);
```

SPARK highlights

- SPARK/Ada
  - Subset of Ada
    - **Same compiler** plus tools for verification/proofs
  - Contracts supported by Ada syntax
    - Enforceable at run-time by the compiler

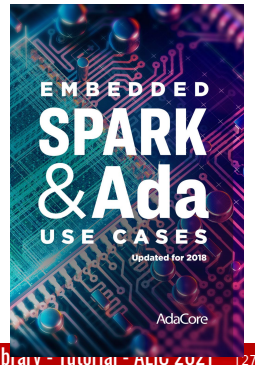
```
procedure Increment (X : in out Integer)
  with Global  => null,
       Depends => (X => X),
       Pre     => X < Integer'Last,
       Post    => X = X'Old + 1;
```

- Can prove:
  - Absence of runtime errors (exceptions)
    - Run-time checks can be safely disabled
  - Properties of the program
    - Guided by the programmer
    - Can be adopted incrementally

```
type Prime is new Positive
   with Dynamic_Predicate =>
     (for all D in 2 .. Prime / 2 =>
         Prime mod D /= 0);
```

- Similar in some respects to Frama-C/Misra-C
  - But Ada has fewer undefined behaviors
  - The SPARK subset grows with each version

Johannes Kanig. “A comparison of SPARK with MISRA C and FRAMA-C”. 2018.



SPARK success stories

ElectronicDesign

MARKETS > AUTOMOTIVE

Ada and RISC-V Secure Nvidia's Future

Nvidia is using RISC-V for its security processor, and programming is handled via Ada/SPARK.

William G. Wong

JAN 23, 2020

https://www.youtube.com/watch?v=2YoPoNx3L5E

Latest News

- Recent developments
  - Dashing, Foxy migration
  - Continuous integration
  - Ada Package Manager integration
- TBD:
  - Actions (ETA early 2021)

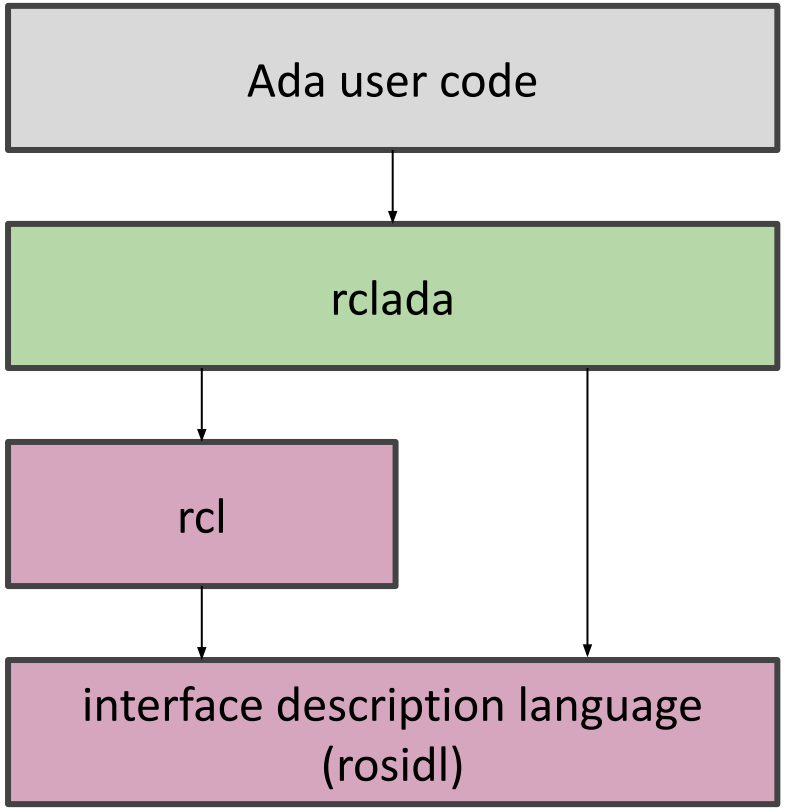
```
$ git branch -a
* foxy
  remotes/origin/HEAD -> origin/foxy
  remotes/origin/bouncy
  remotes/origin/dashing
  remotes/origin/foxy
```

ada4ros2

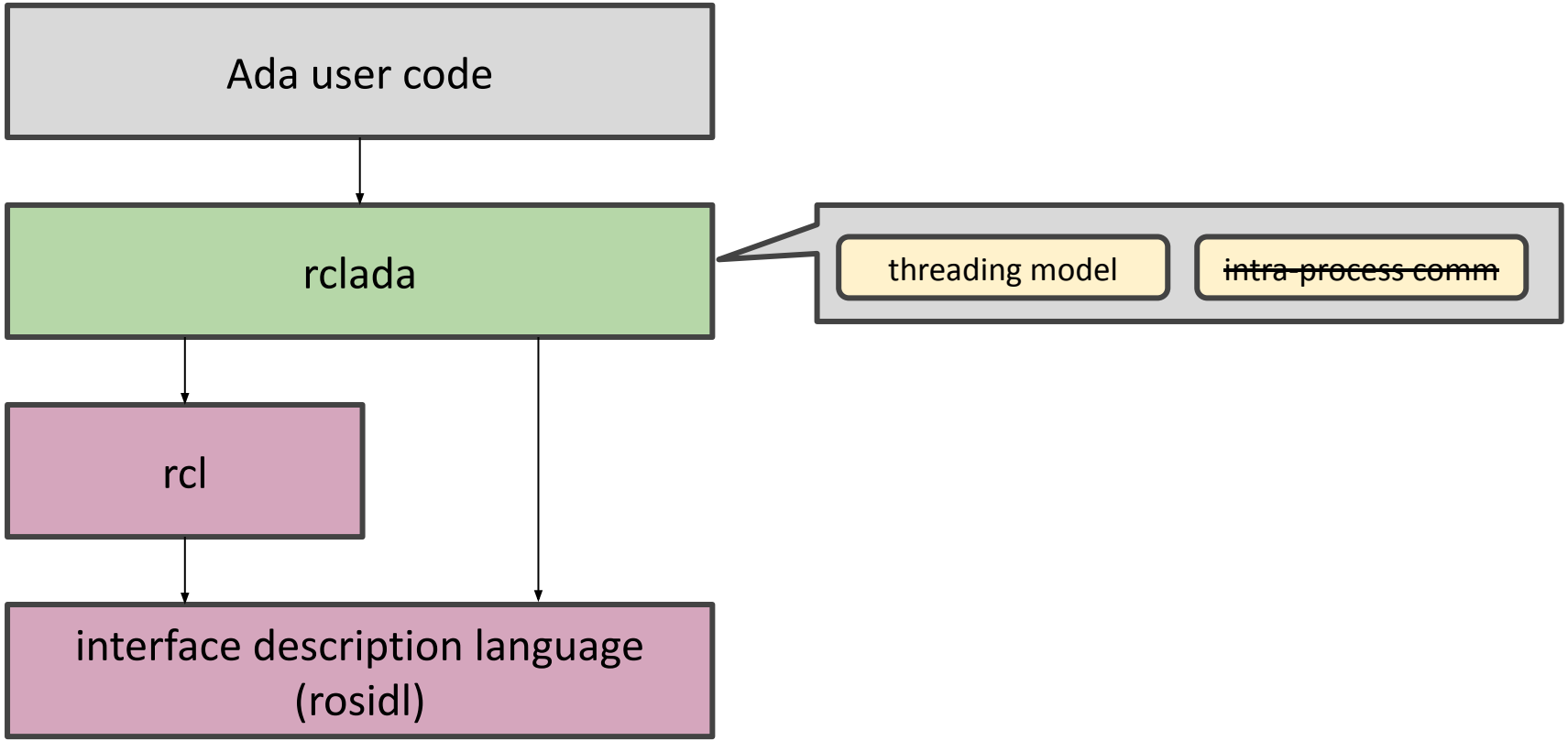
Main repository of the RCLAda project.

```
$ alr list rclada
rclada           ROS2 Ada Client Library
rclada_examples  ROS2 Ada Client Library - Examples
```

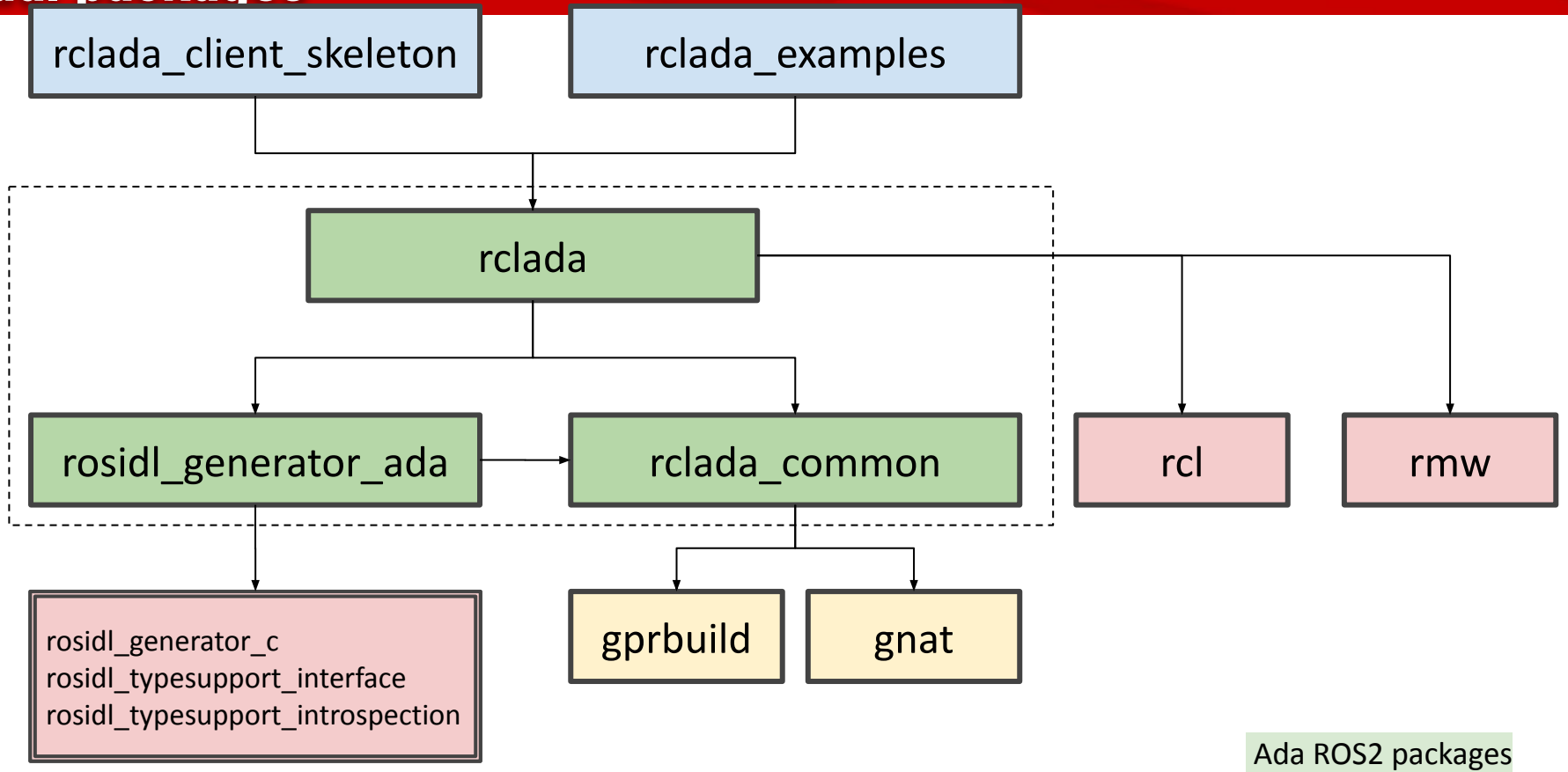
ROS2 architecture II



ROS2 architecture II



Actual packages



Actual packages