ATOSTEK

ROSCon 2022: Native Rust

Contents

ATOSTEK

ATOSTEK

In business for 23 years
~120 employees
Turnover ~ 9 M€
Owned by personnel

Digitalization Engineering

Machines and Automation
Healthcare and Social Services
Public Sector IT

ATOSTEK

Why Rust is important for robot programming

Rust is so far the best candidate for replacing C and C++ in systems programming.

Produces low-overhead bare-metal code like C++
– with safety guarantees of a high-level language:

- Memory safety
- Data race safety*
- Reasonably easy to learn for C++ programmers
- Algebraic type system

No historical baggage of C/C++:

- Undefined or implementation-defined behaviour
- Large bag of pitfalls to learn

ATOSTEK

* Yes, really! The compiler can statically detect data races!
(But deadlocks are still on the programmer's responsibility.)

How does Rust achieve that?

- Ownership-based memory handling
  - Handles a mix of stack and heap allocations
    - Stack allocation beats even the fastest malloc library
  - Statically checked lifetimes → zero overhead
  - Also checks safe memory sharing between threads

- unsafe constructs
  - Clearly separates dangerous constructs, e.g. raw pointer handling, from safe code.
  - unsafe is needed for very low-level data structure implementations and FFI, but that is a very small fraction of code lines.

- Compiler is built on LLVM
  - Very advanced back-ends for x86(-64), ARM, and others

ATOSTEK

ATOSTEK

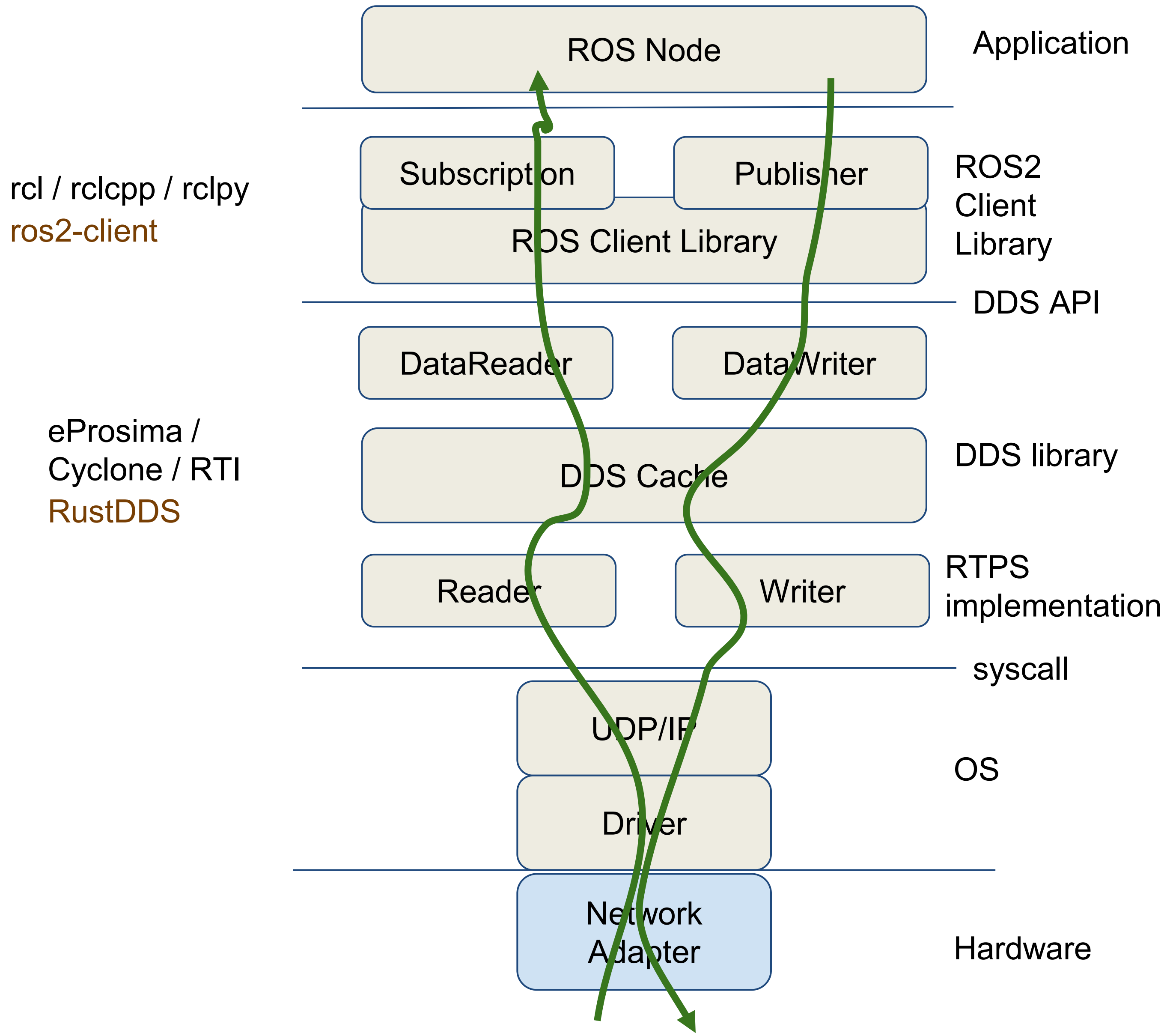
Programming ROS2 with Rust

ROS2 client in Rust: Alternatives

Name	Author	URL	Notes
ros2-rust	Multiple contributors	https://github.com/ros2-rust/ros2_rust	Official Rust binding to rcl
ros2-client / RustDDS	Atostek Oy / Juhana Helovuo and others	https://github.com/jhelovuo/ros2-client	Native Rust implementation of ROS2 client library - and DDS!
r2r	Martin Dahl and others	https://github.com/sequenceplanner/r2r	Binding to rcl. Rust API uses async functions.
rclrust	Yuma Hiramatsu	https://github.com/rclrust/rclrust	Uses Rust macros(!) to translate IDL to Rust types.
rus2	Marshal SHI	https://github.com/marshalshi/rus2	Inactive since Sep 2020
rosrust	Adnan Ademovic and others	https://github.com/adnanademovic	ROS 1 Inactive since Aug 2020

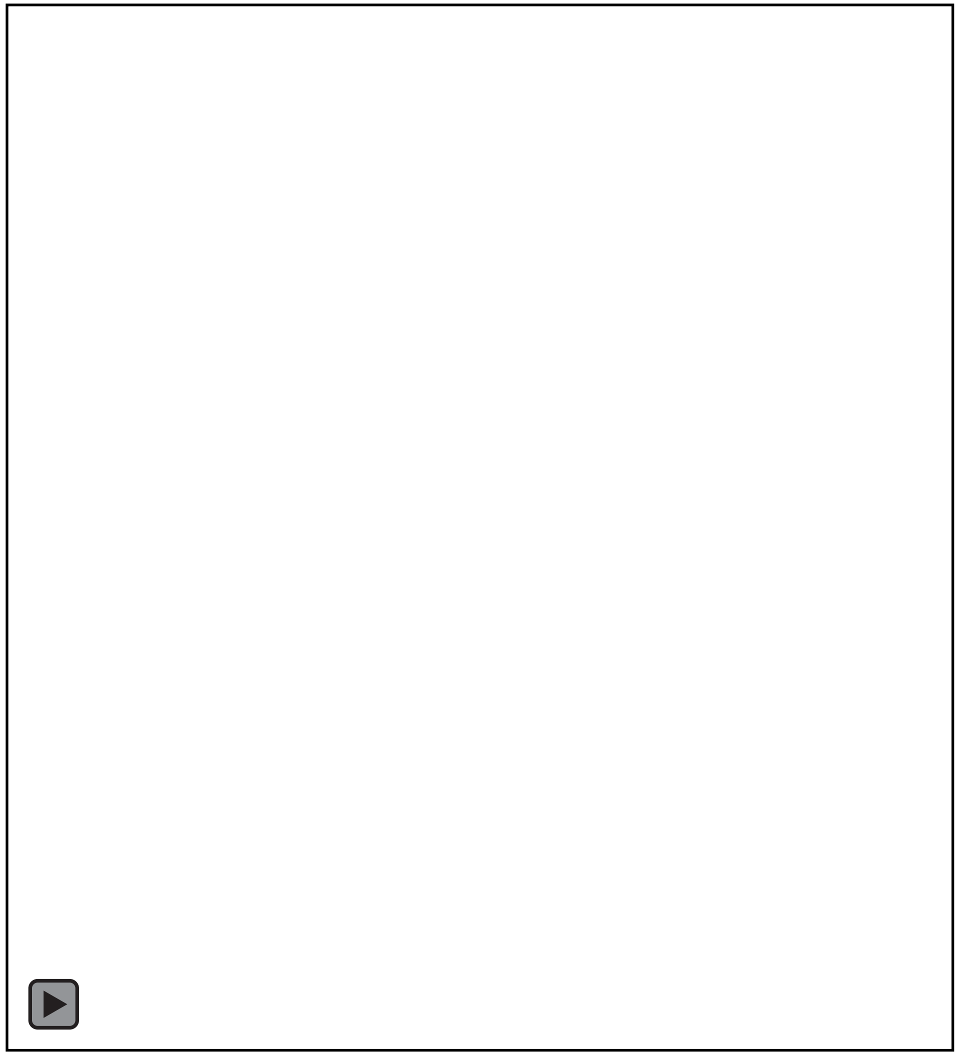
RustDDS

- Native Rust implementation of DDS API and RTPS network protocol from scratch
- Apache 2.0 -licensed open source: https://github.com/jhelovuo/RustDDS
- Features
  - Discovery (peer autodetection)
  - Non-blocking I/O
  - “Zero-copy” receive path
    - Single-copy transmit path
  - Serialize/deserialize directly to Rust objects
  - Reliable and Best Effort QoS
  - History QoS
  - Fragmentation (large objects)

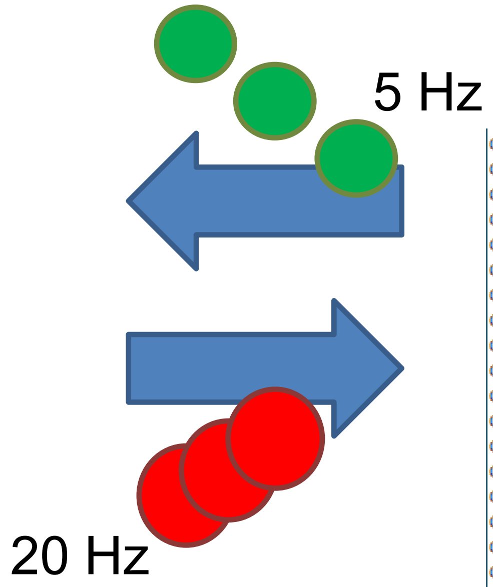


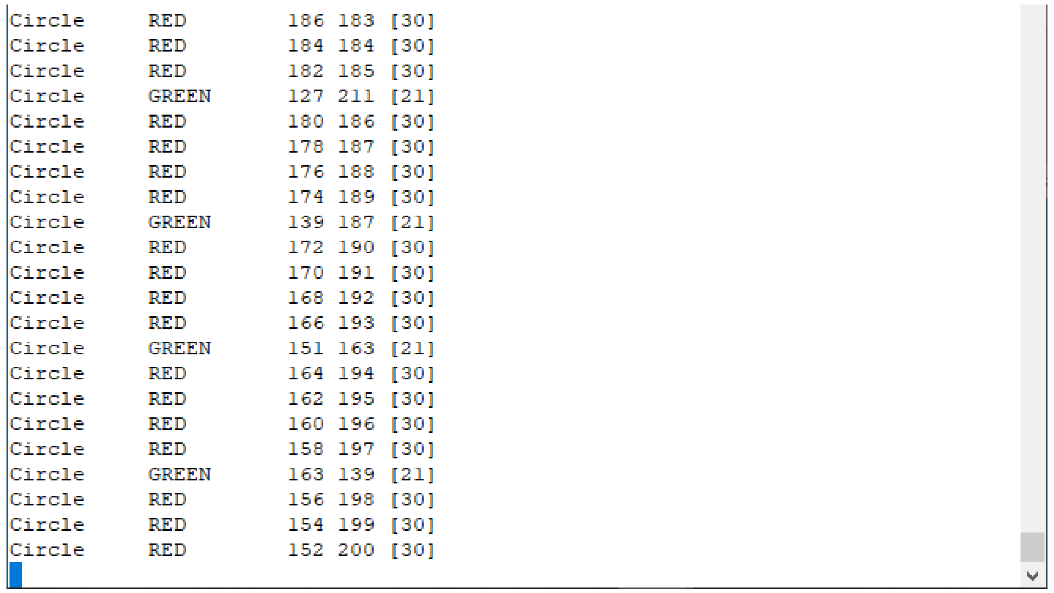
ATOSTEK

RustDDS



```
$ cargo run --example=shapes_demo
-- -P -t Circle -r -c GREEN -S
```





```
Circle     RED        186 183 [30]
Circle     RED        184 184 [30]
Circle     RED        182 185 [30]
Circle     GREEN      127 211 [21]
Circle     RED        180 186 [30]
Circle     RED        178 187 [30]
Circle     RED        176 188 [30]
Circle     RED        174 189 [30]
Circle     GREEN      139 187 [21]
Circle     RED        172 190 [30]
Circle     RED        170 191 [30]
Circle     RED        168 192 [30]
Circle     RED        166 193 [30]
Circle     GREEN      151 163 [21]
Circle     RED        164 194 [30]
Circle     RED        162 195 [30]
Circle     RED        160 196 [30]
Circle     RED        158 197 [30]
Circle     GREEN      163 139 [21]
Circle     RED        156 198 [30]
Circle     RED        154 199 [30]
Circle     RED        152 200 [30]
```

ATOSTEK

Code: Using RustDDS

```
#[derive(Serialize, Deserialize, Clone)]
struct Shape {
  color: String,
  x: i32,
  y: i32,
  shapesize: i32,
}
impl Keyed for Shape {
  type K = String;
  fn key(&self) -> String {
    self.color.clone()
  }
}

…

let domain_participant = DomainParticipant::new(domain_id)
  .unwrap_or_else(|e|
    panic!("DomainParticipant construction failed: {:?}", e));
```

ATOSTEK

Code: Using RustDDS

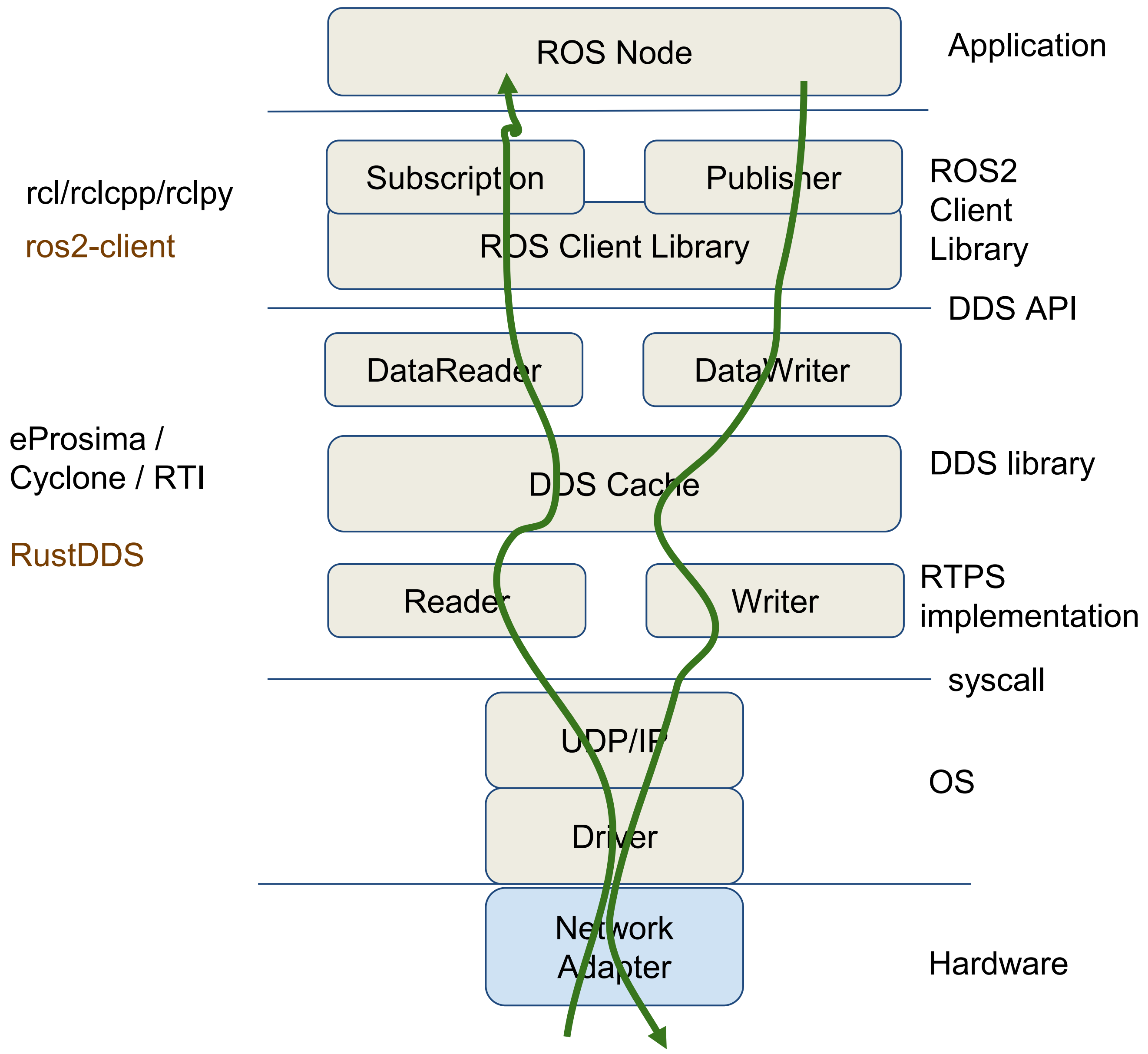
```
poll.poll(&mut events, Some(loop_delay)).unwrap();
for event in &events {
  match event.token() {
   // ...
   READER_READY => {
     // ...
     match reader.take_next_sample() {
         Ok(Some(s)) =>
           match s.into_value() {
             Ok(sample) => println!( "{:10.10} {:10.10} {:3.3} {:3.3} [{}]",
                   topic.name(), sample.color, sample.x, sample.y, sample.shapesize, ),
             Err(key) => println!("Disposed key {:?}", key),
           },
         Ok(None) => break, // no more data
         Err(e) => println!("DataReader error {:?}", e),
     }
   }
```

ATOSTEK

Code has been edited for presentability. Please see sources at GitHub for full version.

ros2-client

- ros2 - client is a Rust crate that implements something similar to rcl and (parts of) rclcpp/rclpy.
  - Topics
  - Services
- Runs on top of RustDDS.
- Does not yet have an event loop. Nodes must use .poll() from the Metal I/O library to implement event loop.



ATOSTEK

Code: Using ros2-client

```
let mut node = create_node();
let topic_qos = create_qos();
let chatter_topic = node
  .create_topic( "/chatter",
    String::from("std_msgs::msg::dds_::String_"),
    &topic_qos, )
  .unwrap();
let mut chatter_subscription = node
   .create_subscription::<String>(&chatter_topic, None)
   .unwrap();

// ... initialize polling here ...
```

```
loop {
  poll.poll(&mut events, None)
    .unwrap();
  for event in events.iter() {
    match event.token() {
      Token(1) => match chatter_subscription.take() {
        Ok(Some((message, _messafe_info))) => {
          let l = message.len();
          println!("message len={} : {:?}", l, &message[..min(l,50)]);
        }
        Ok(None) => println!("No message?!"),
        Err(e) => {
          println!(">>> error with response handling, e: {:?}", e)
        }
      },
      _ => println!(">>> Unknown poll token {:?}", event.token()),
    } // match
  } // for
} // loop
```

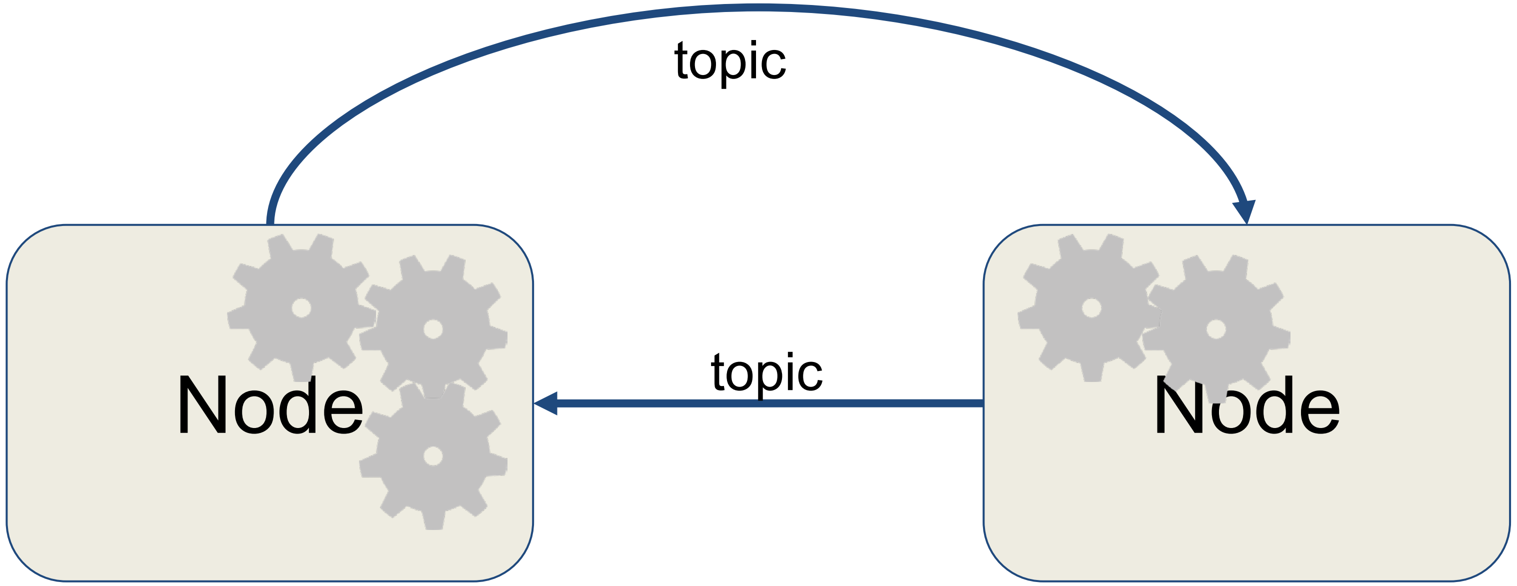
Code has been edited for presentability. Please see sources at GitHub for full version.

The Flexbot Framework

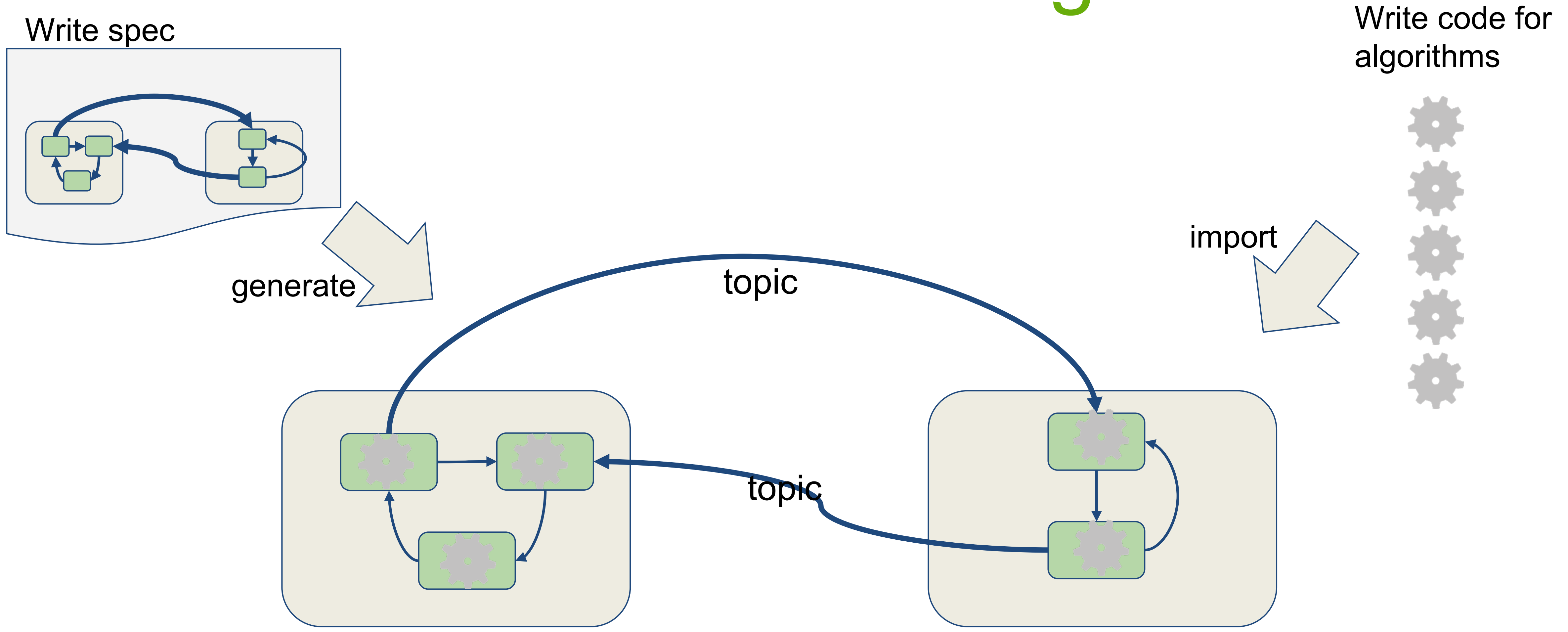
- Robot programming framework developed at Atostek since 2019.
- Not (yet) open source
- Main idea is to construct software from nodes and communication channels just like in ROS.
- We implemented ROS2-compatibility in 2020-2021 → Flexbot can be now seen as an extension to ROS2.
  - Supports programming nodes in Rust.
  - Whole software is described by a machine-readable data flow specification.
  - Boilerplate code is generated from the specification
  - Closely coupled nodes can simplify inter-node communication
  - → Enables fine-grained data flow programming with tens or hundreds of nodes.
  - → Simple individual nodes → Improved reusability

ATOSTEK

ROS2: high-level view



ROS2 and Flexbot together



ATOSTEK

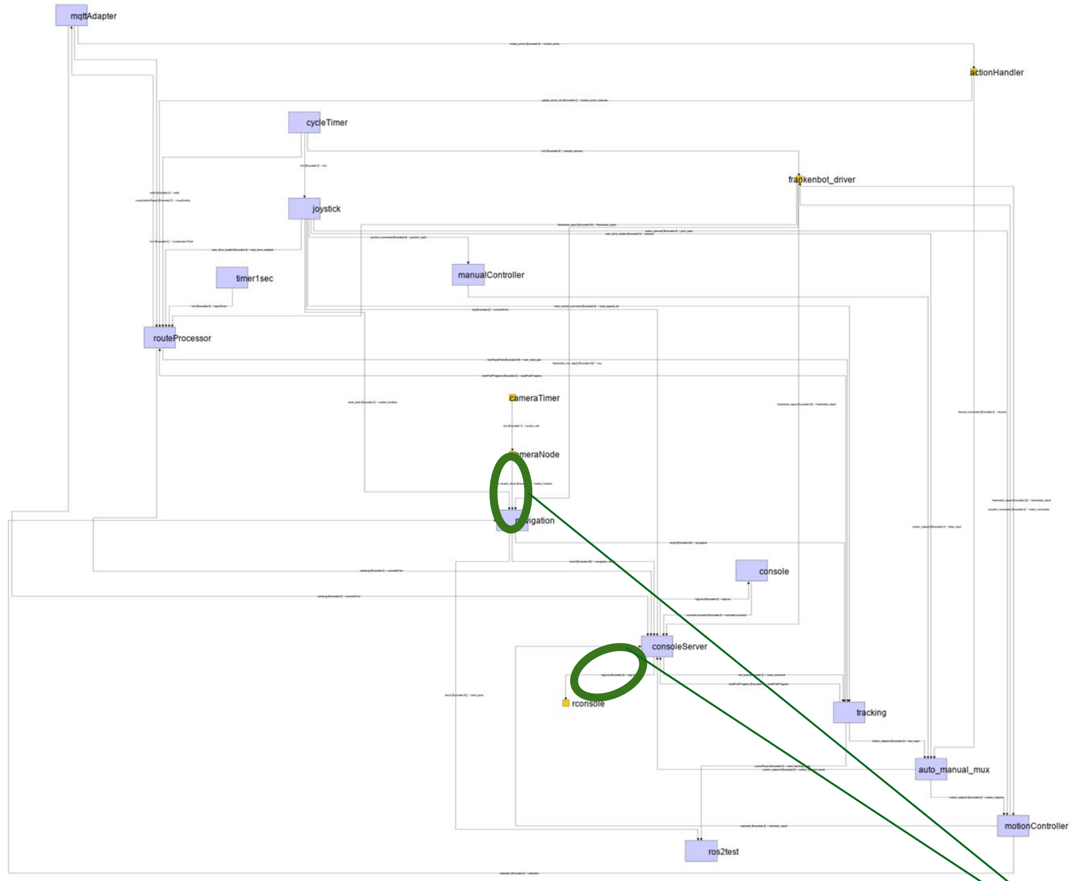
Flexbot Example: Pulu robot path tracking

- Vehicle mechanics & electrics: Pulu Robotics, prototype
- Main controller: Raspberry Pi 4
- Camera navigation: 2 x (RPi3 + Raspberry Camera module)
- Software: ”atosbot” application using Flexbot framework

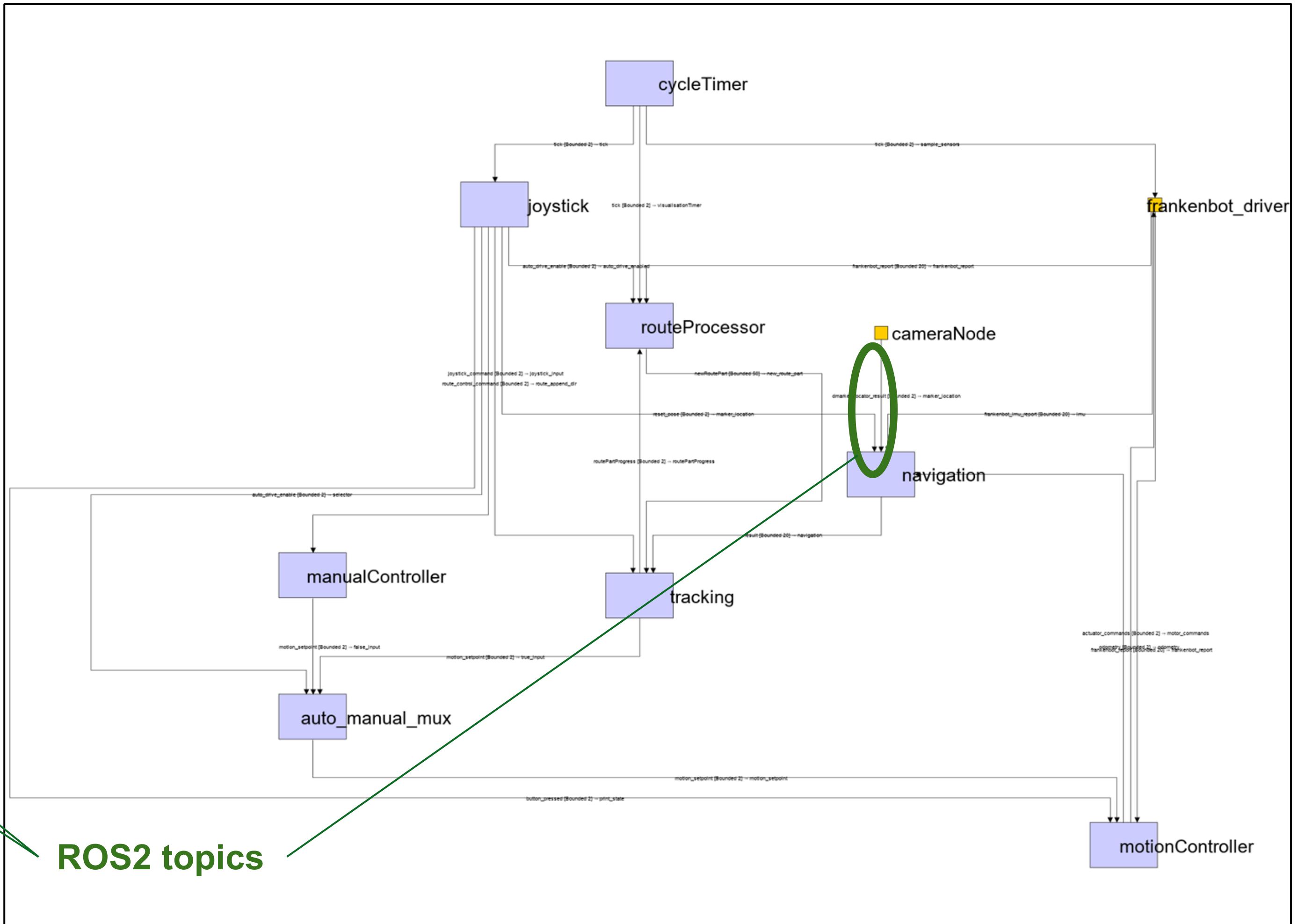
- Rviz2 - ROS2 Foxy, pre-built binary on Ubuntu Linux

Flexbot Example: Pulu robot path tracking

Full Data flow Graph



Simplified

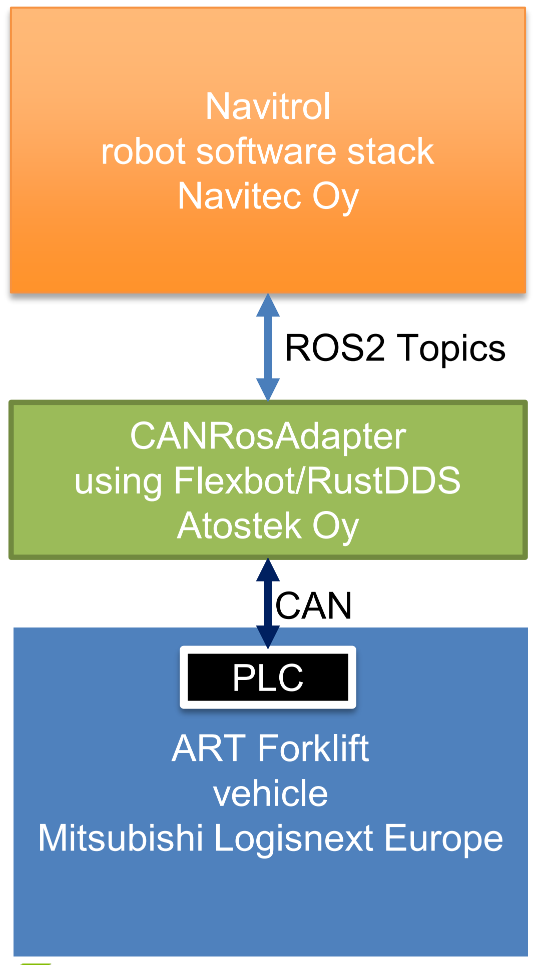

cycleTimer
tick [Bounded 2] → tick
tick [Bounded 2] → sample_sensors
joystick
tick [Bounded 2] → visualisationTimer
frankenbot_driver
auto_drive_enable [Bounded 2] → auto_drive_enabled
frankenbot_report [Bounded 20] → frankenbot_report
routeProcessor
cameraNode
joystick_command [Bounded 2] → joystick_input
route_control_command [Bounded 2] → route_append_dir
newRoutePart [Bounded 50] → new_route_part
dmarker_locator_result [Bounded 2] → marker_location
reset_pose [Bounded 2] → marker_location
frankenbot_imu_report [Bounded 20] → imu
routePartProgress [Bounded 2] → routePartProgress
navigation
auto_drive_enable [Bounded 2] → selector
result [Bounded 20] → navigation
manualController
tracking
actuator_commands [Bounded 2] → motor_commands
odometry [Bounded 2] → odometry
frankenbot_report [Bounded 20] → frankenbot_report
motion_setpoint [Bounded 2] → false_input
motion_setpoint [Bounded 2] → true_input
auto_manual_mux
motion_setpoint [Bounded 2] → motion_setpoint
button_pressed [Bounded 2] → print_state
motionController

ROS2 topics

ATOSTEK

Automated Forklift

Mitsubishi Logisnext Europe
Demo Center, Järvenpää, Finland
Manual teleoperation test,
using software from multiple vendors.



Summary

- Rust is important for robots
  - No-overhead real-time systems programming capability like C/C++
  - Very good memory safety, no garbage collection required
  - Type system more straightforward than C++ classes, but still powerful
- RustDDS
  - Open-source native Rust DDS/RTPS implementation from Atostek
- ros2-client
  - ROS2 topics and services on top of RustDDS
  - Enables ROS2 nodes in native Rust
- Flexbot
  - Framework to produce ROS2-compatible dataflow software
  - Code generation from machine readable specification
    - Local-only data channels use very lightweight communication
    - Enables scaling to large number of nodes.