Tricycle Controller with ros2_control

ROS Meetup Munich Feb 2023

Presented by:

Johannes Plapp - CTO

Tony Najjar - Senior Robotics Software Engineer

0 | Outline

1. **Pixel Robotics**
    a. Who are we?
    b. Our concept
    c. Robot steering geometries
2. **Introduction to ros2_control**
    a. What is ros2_control?
    b. Why we migrated to ros2_control?
    c. Our ros2_control architecture
3. **Tricycle controller**
    a. Input/Output
    b. Core Logic
4. **Future Plans**
5. **Q&A**

1| Pixel Robotics

Who are we?

Founded in 2020

Munich startup

International team

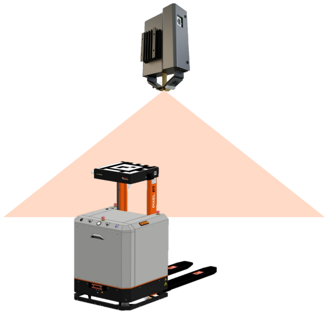
In operation at 3 customers

Series production in Munich

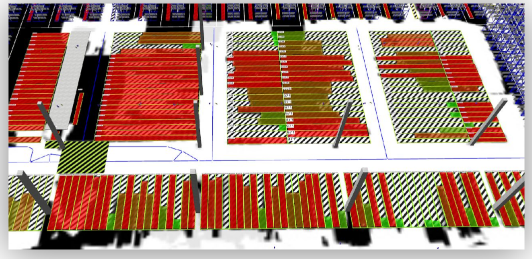
1 | Pixel Robotics

Our Concept

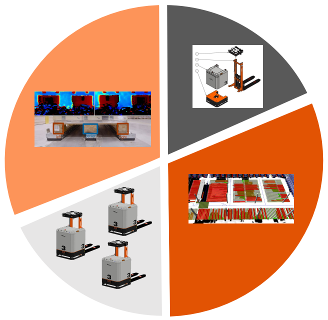
Navigation with
AI cameras

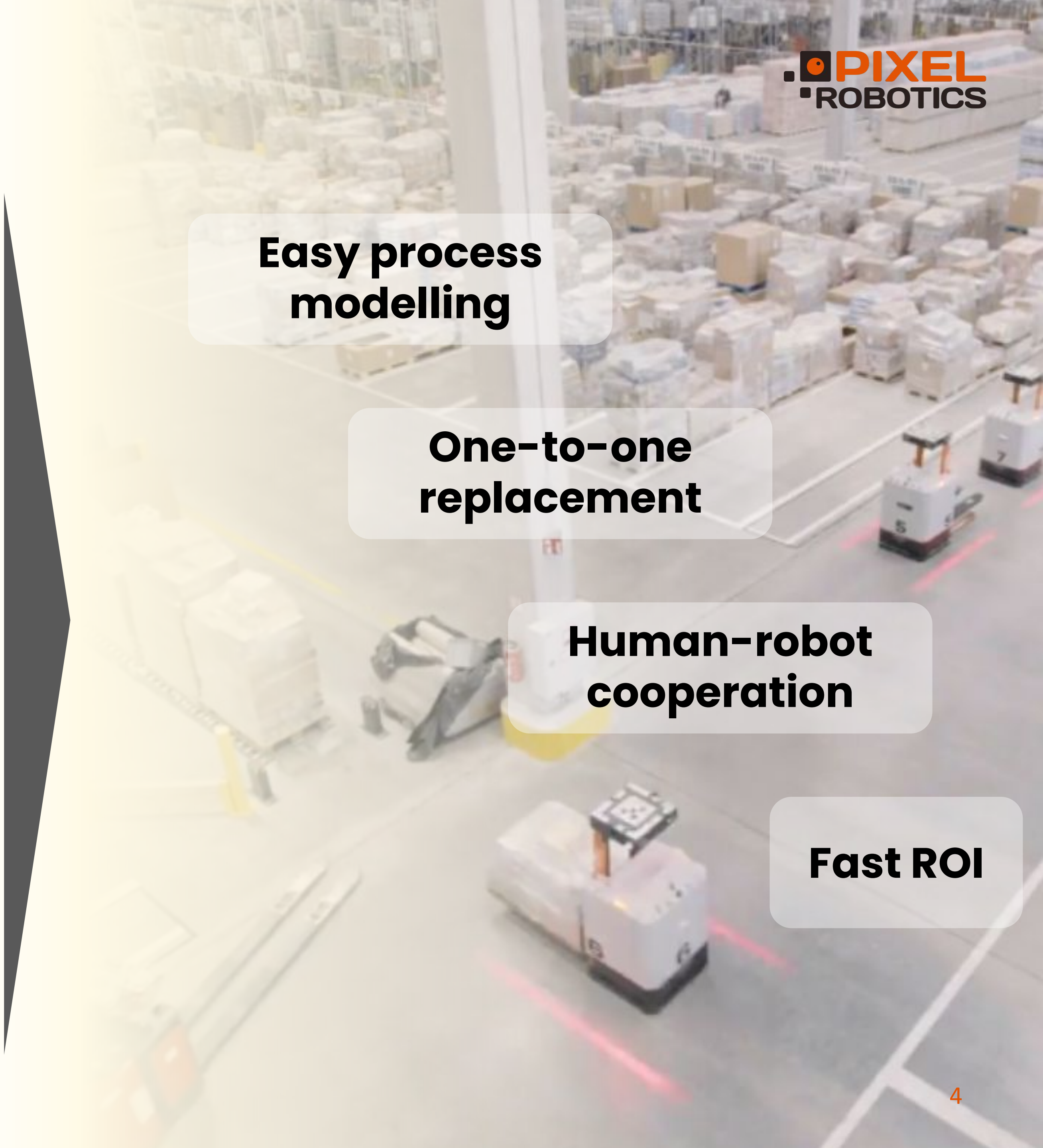


Digital Twin
Integration



Scalable
system approach





Easy process modelling

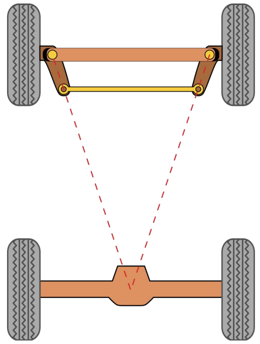
One-to-one replacement

Human-robot cooperation

Fast ROI

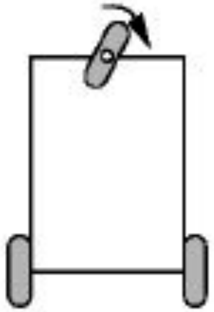
PIXEL ROBOTICS

Robot steering geometries



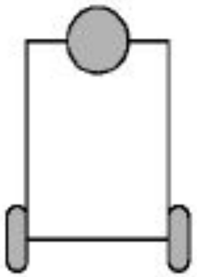
Ackermann

Commonly used for cars



Tricycle

Similar to ackermann, but only a single front wheel



Differential drive

Commonly used for small robots

Can turn on the spot

Hard to control if robot heavy and driven wheels not centrally placed

PIXEL ROBOTICS

What is ros2_control?

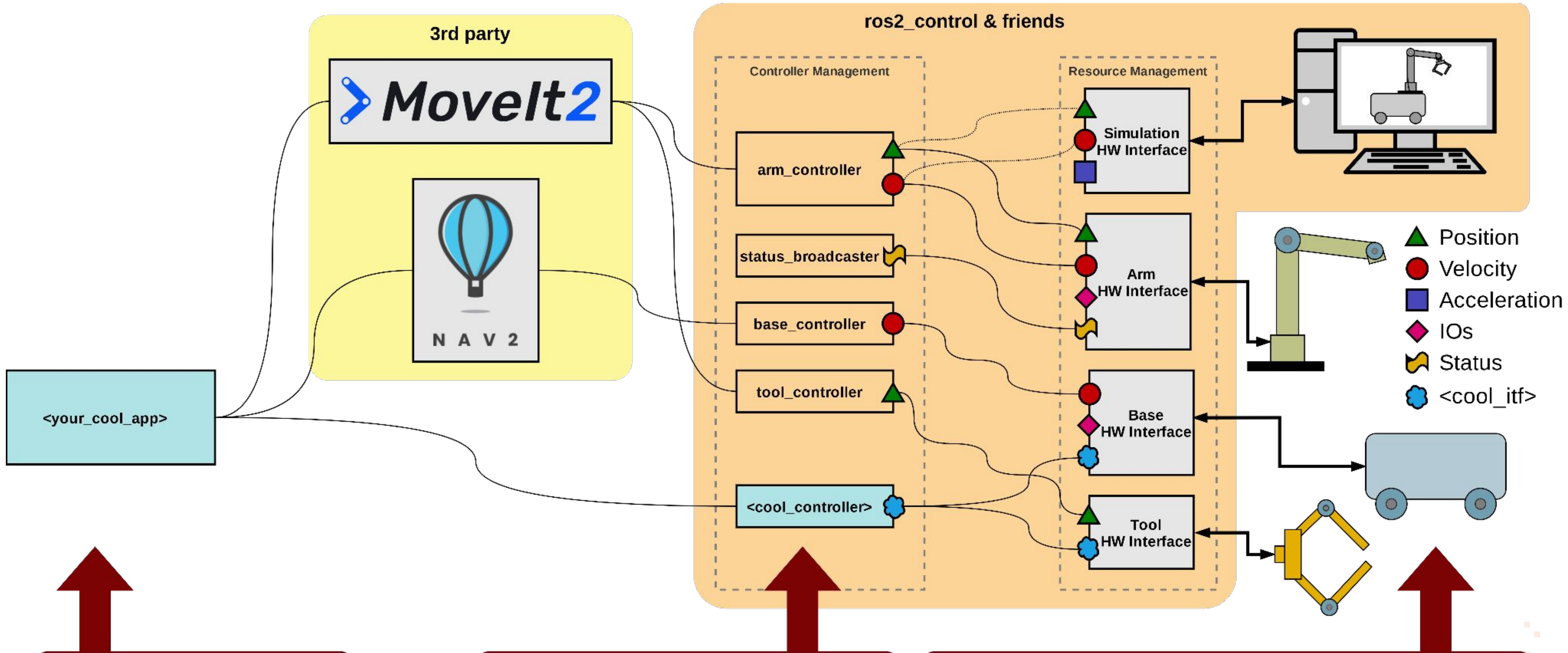
ros2_control is a framework for the control of robots using ROS2.

Its main goals are:

- Offer a “home” for the controllers and hardware interfaces to manage them (loading/unloading, execution loop, resource access management…)
- Abstract hardware and low-level control for other frameworks like MoveIt2 and Nav2
- Decouple controllers from hardware interfaces for modularity and reusability

2 | Introduction to ros2_control

What is ros2_control?



Maximize resources spent on actual **client application**

- **Leverage** existing controllers
- **Implement** custom ones, extend existing

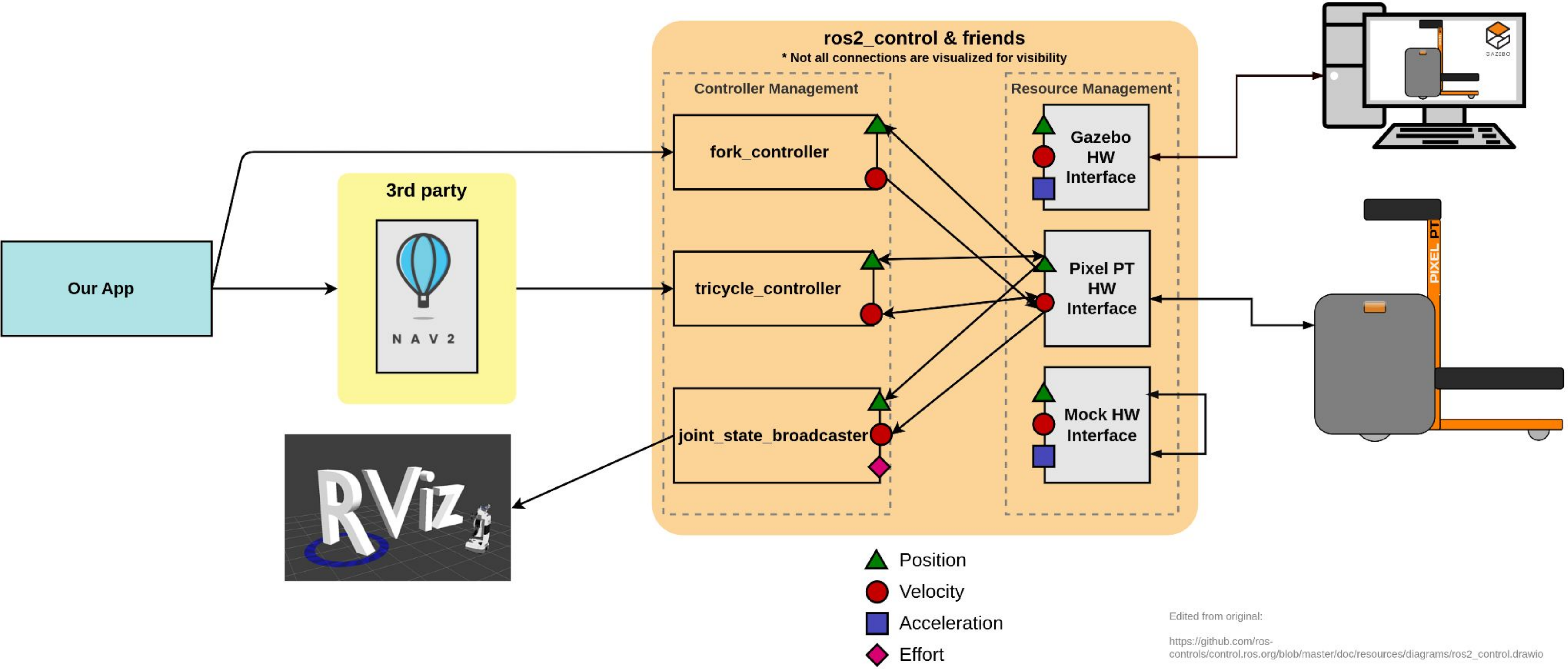
- **Leverage** simulation backend
- **Real robot** backend - extend existing ones or create your own

2 | Introduction to ros2_control

Why we migrated to ros2_control?

- Opportunity to migrate control stack to C++ for better performance (initial prototype was in Python)
- Free code! There are already public controllers and hardware interfaces to take advantage of:
  - Gazebo/Webots Hardware interfaces
  - joint_state_broadcaster
  - Mobile base controllers
  - joint_trajectory_controller
  - general-purpose PID controller (coming soon)
- Use the same controller for all hardware interfaces
- General lifecycle management and resource access control
- All the other goodies we might want to use some day (chaining controllers, emergency stop handling, variable rate controllers, transmissions, custom interface types, etc…)

Our ros2_control architecture



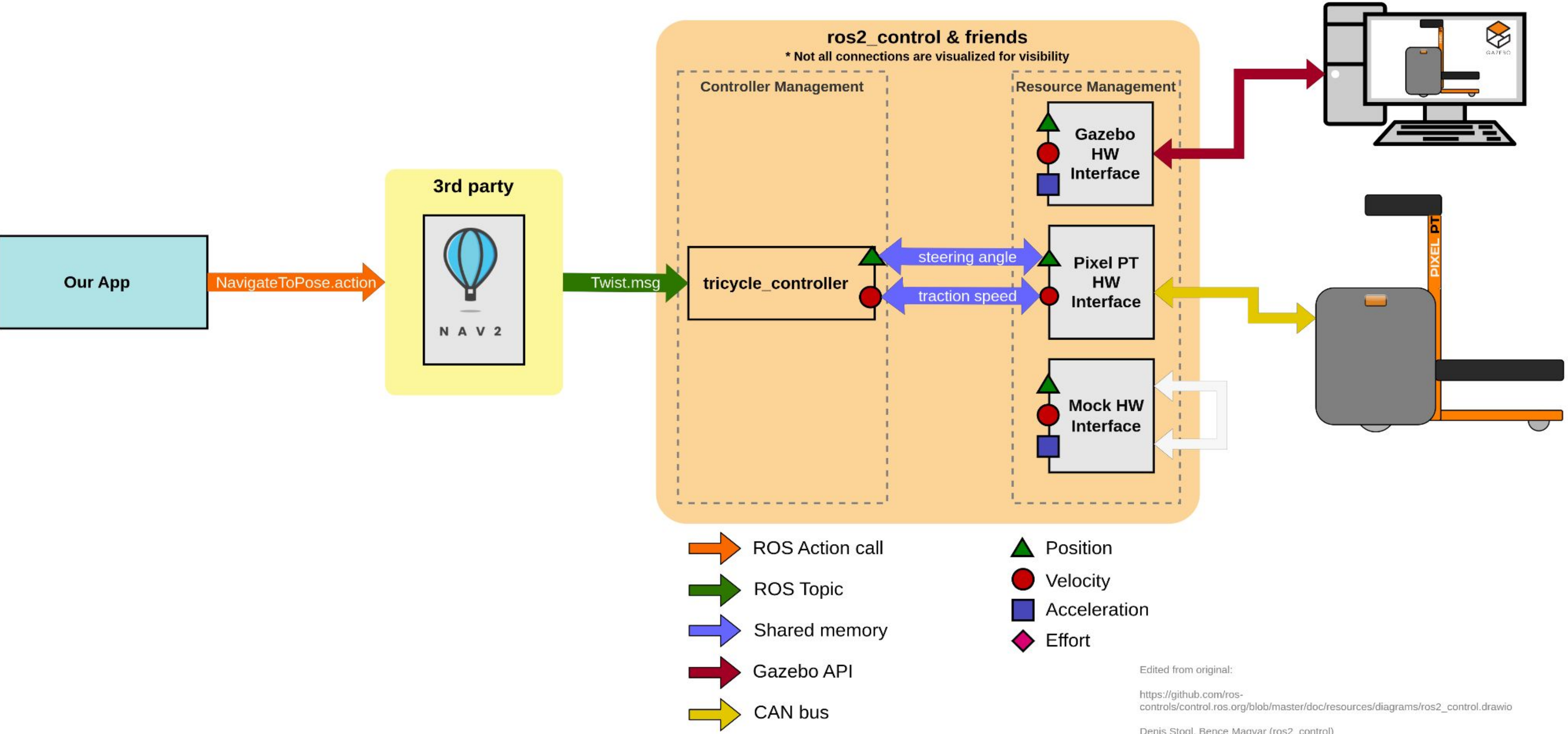
Edited from original:

https://github.com/ros-controls/control.ros.org/blob/master/doc/resources/diagrams/ros2_control.drawio

Denis Stogl, Bence Magyar (ros2_control)

3 | Tricycle Controller

Input/Output

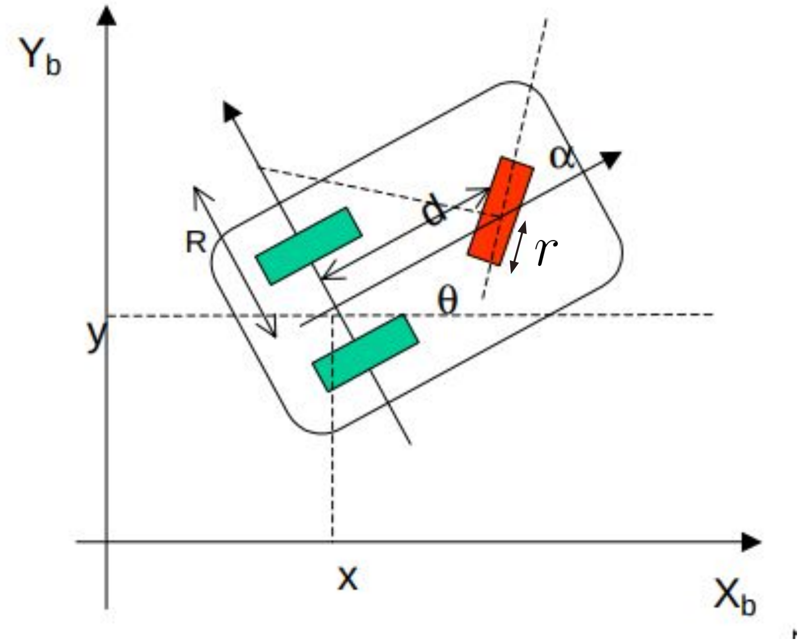


Edited from original:

https://github.com/ros-controls/control.ros.org/blob/master/doc/resources/diagrams/ros2_control.drawio

Denis Stogl, Bence Magyar (ros2_control)

Core Logic



Joints command generation

$$steering\_angle(t) = \arctan\left(\frac{angular.z(t) \cdot d}{linear.x(t)}\right)$$

$$speed(t) = \frac{linear.x(t)}{r \cdot \cos(steering\_angle(t))}$$

Odometry

$$linear.x(t) = speed(t) \cdot r \cdot \cos(steering\_angle_{read}(t))$$

$$linear.y(t) = 0$$

$$angular.z(t) = \frac{speed(t) \cdot r}{d} \sin(steering\_angle_{read}(t))$$

see
http://users.isr.ist.utl.pt/~mir/cadeiras/robmovel/Kinematics.pdf

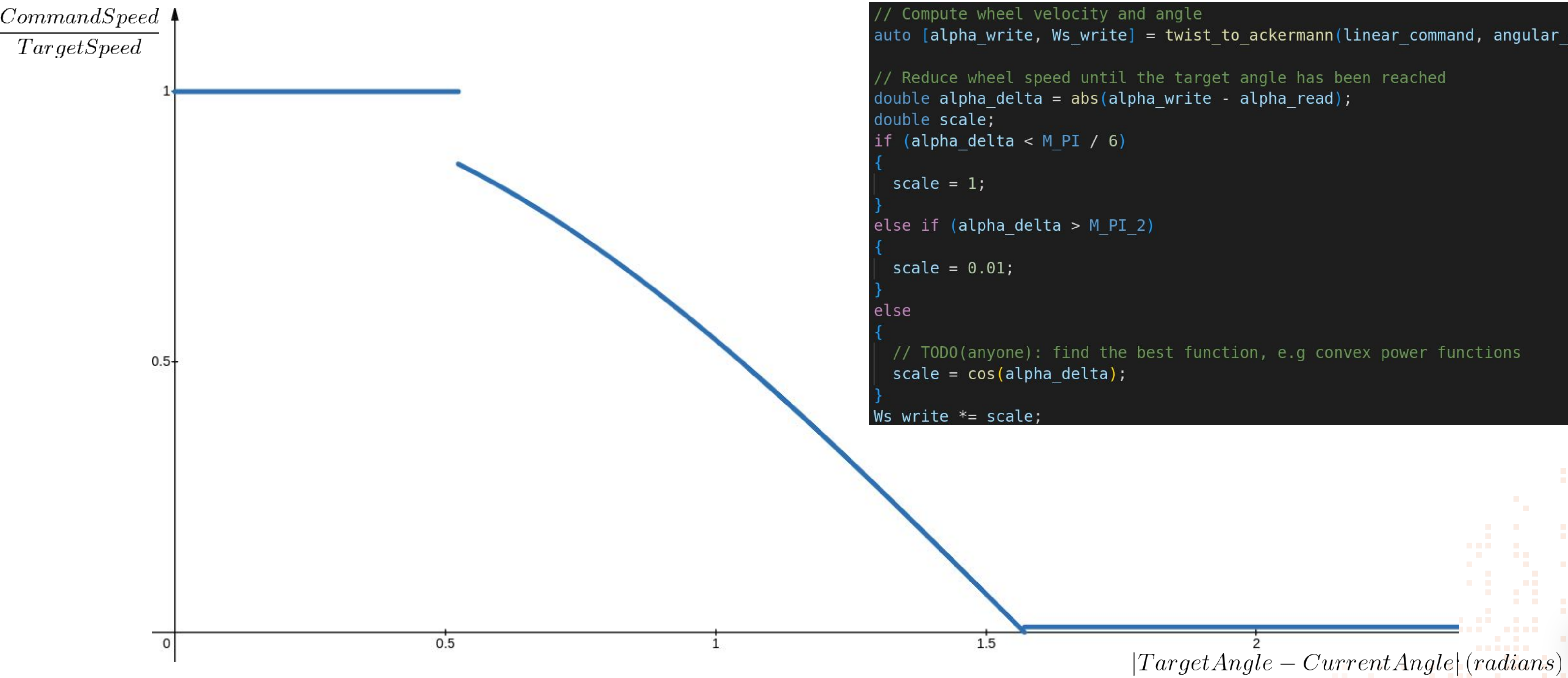
3 | Tricycle Controller

Extra Features

- Option to publish **odom=>base_link** directly as *tf2_msgs/TFMessage* or *nav_msgs/Odometry* (for further fusion)
- Timeout and stop if no Twist is received for a configurable amount of time
- Rate limiter - Limits velocity, acceleration, jerk on wheel commands

Extra Features

Scaling the command speed in function of the steering angle difference, in other words: don’t roll too much before the steering wheel is at the correct angle



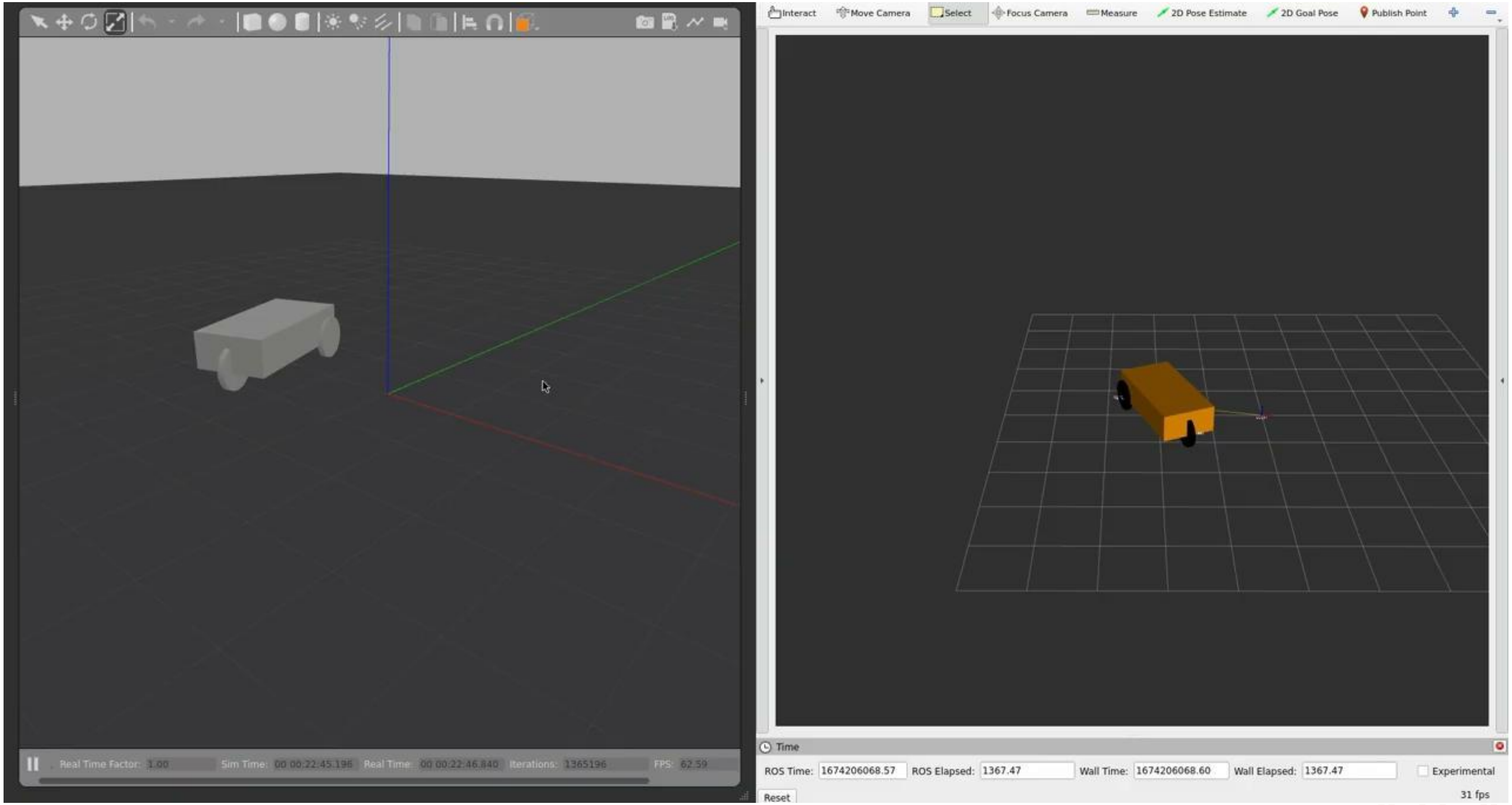

```cpp
// Compute wheel velocity and angle
auto [alpha_write, Ws_write] = twist_to_ackermann(linear_command, angular_command);

// Reduce wheel speed until the target angle has been reached
double alpha_delta = abs(alpha_write - alpha_read);
double scale;
if (alpha_delta < M_PI / 6)
{
  scale = 1;
}
else if (alpha_delta > M_PI_2)
{
  scale = 0.01;
}
else
{
  // TODO(anyone): find the best function, e.g convex power functions
  scale = cos(alpha_delta);
}
Ws_write *= scale;
```

4 | Future plans

The controller has been quite stable for us. No bugs have been reported since the release in September 2022 but we are looking forward to more community testing and feedback

There are plans to create a base *SteeringController* that steering controllers would inherit from (ackermann, tricycle, bicycle). For more details see: https://github.com/ros-controls/ros2_controllers/pull/484

5 | Q&A



https://github.com/ros-controls/gazebo_ros2_control

Contact

Johannes Plapp
CTO
johannes.plapp@logivations.com

Pixel Robotics GmbH
Riesstraße 18
80992 München

Tony Najjar
Senior Robotics Software Engineer
tony.najjar@logivations.com

Pixel Robotics GmbH
Riesstraße 18
80992 München

www.pixel-robotics.eu

Open positions