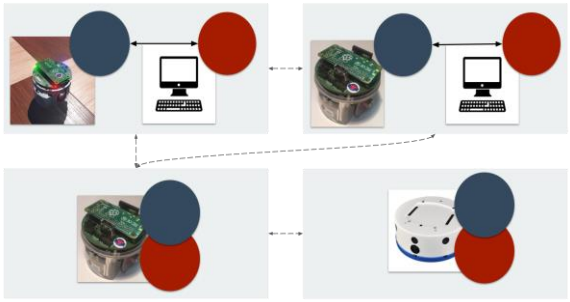
# ROS2 Programming Interface for the E-Puck2 Robot

Darko Lukic

Professor : Alcherio Martinoli
Assistant(s) : Cyrill Baumann

Robotics simulations have been proven to be a powerful tool for developing a robot controller as they are easy to set up, cheap, fast, and convenient to use. However, the final objective is usually to deploy the controller on the real robots or even to run the controller on an arbitrary robot. From the software point of view, solving those problems can be addressed by defining a common API for the physical and simulated robot. ROS is often used as a meta-operating system for this purpose to define the common API.

Therefore, we implemented a ROS2 driver for an e-puck2 physical robot and a generalized ROS2 driver for Webots simulated robots within this project. The ROS2 drivers expose a nearly identical ROS2 interface in both simulation and reality, that allows a controller to interact in the same way with the physical e-puck2 and the simulated robots. Effectively, it allows the controller developers a seamless transition between simulated and physical e-puck2 robots or other simulated robots.



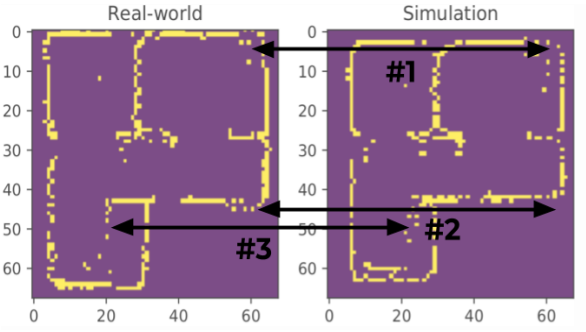
*The desired workflow in robotics. A user can use the same custom controller (red) to control a robot in simulation, in the real world, or an arbitrary robot thanks to the abstraction layer (blue circle).*

The thesis summarizes the implementation details of the mentioned ROS2 drivers. It shows how Webots distance sensors, IMU related devices, LEDs, cameras, motors, and encoders are mapped into the ROS2 interface. It introduces how odometry, velocity control, and coordinate frames are generated from basic Webots devices, such as motors and encoders.

Furthermore, the implementation details and major challenges regarding the ROS2 driver for the e-puck2 physical robot are presented. It provides a solution to various problems originated from a low-performance computer with an Armv6 architecture. Most notably, the tackled challenges also include cross-compilation, offloading image compression to GPU, unit testing in CI (with x86 architecture), and overall performance improvements.

A possibility to automate the creation of ROS2 interfaces for Webots is observed. Therefore, a universal ROS2 driver for Webots is developed. New features are introduced to the Webots core to enable the automatic creation of ROS2 interfaces, including URDF export. On top of Webots, a modular software layer is implemented that performs API conversion from Webots to ROS2.

Moreover, various examples, including navigation and mapping, leveraging the e-puck2's ROS2 interface are built to demonstrate the usability of the interface.




*Maps obtained using an unchanged controller with simulated and physical e-puck2.*

The e-puck2 ROS2 drivers are validated using the same scenarios, showing only minimal difference in the behaviors and results as is illustrated in the figure above. The universal ROS2 driver is tested with multiple robots, including Khepera IV, TurtleBot3 Burger, and TIAGo++.

Thanks to the minimal behavioral difference between simulated and real robots as well as the shared API, researchers can now quickly validate their ROS2 controllers on both the physical and simulated e-puck2 robots, but also experiment with other simulated robots in Webots.