Node: wheel_velocity_simulator_node

Header

Team: G4

Created: 21.11.2025

name	date	Short description
Burhan Topaloglu	21.11.2025	Created
Burhan Topaloglu	25.11.2025	Improved integration with other nodes using GPT

Simulates wheel encoder angular velocities for mecanum wheels based on parameter-defined time intervals and polynomial motion models. Publishes *g425_assign4_interfaces_pkg::msg::Mecanum* messages.

Node description

This node simulates angular velocities of four mecanum robot wheels (front-left, front-right, rear-left, rear-right).
Each wheel's velocity is computed as a piecewise polynomial function over time, defined through ROS2 parameters.
After reaching the end of the defined interval sequence, the simulation loops automatically.

The published message contains angular velocities in rad/s for each wheel and a timestamp.

Node sub-objects and functions (communication objects):

(timer, publisher, subscriber, service server, action server, service client, action client

Wall_timer: main simulation timer Called at a user-defined frequency (rate_hz), evaluates wheel velocities and publishes a Mecanum message.
bind function: on_timer()

publisher : mecanum_sim / <topic> Publishes mecanum wheel angular velocities based on interpolated interval values. Default topic: **mecanum_velocity**
Publisher function: pub_->publish(msg) Short description: Publishes the computed mecanum message.

Node actions, messages and services:

Custom Messages used:
g425_assign4_interfaces_pkg::msg::Mecanum
wfl, wfr, wrl, wrr -> wheel angular velocities
stamp -> message timestamp

Custom Node functions :

Custom function: load_intervals() Parses the parameter tree under intervals.* and constructs a list of motion intervals for each wheel. Computes looping time window and validates interval consistency. Custom function: eval(const Interval&, double t) Evaluates an interval at time t based on polynomial type: constant linear quadratic (3-point Lagrange interpolation) Custom function: on_timer() Triggered periodically. Computes simulation time (with looping) and wheel velocities by summing all interval contributions. Publishes output.

Node implementation (main):

Short description of differences to a standard implementation:

Uses multiple custom parameters to construct time-based wheel motion intervals.

Implements polynomial interpolation and looping time logic.

Timer-based continuous simulation.

Dynamically scans and loads interval sets (up to 10).

More complex internal state (intervals, loop timing, polynomial models).

Standard implementation:

```
int main(int argc, char **argv)

{

   rclcpp::init(argc, argv);

   auto node = std::make_shared<MyCustomNode>();

   rclcpp::spin(node);

   rclcpp::shutdown();

   return 0;

}
```

Implemented main:

Same structure as standard, but loads custom node class WheelVelocitySimulator.

Node dependencies :

g425_assign4_interfaces_pkg
Provides the Mecanum message type, which holds wheel angular velocities.

cmath
Used for fmod() to wrap time for looping.

chrono
Used for timer frequency setup.

Standard ROS2 deps:
rclcpp – Node base class, parameters, timers, publishers

std::chrono – timing
std::vector, std::set, std::string – storing intervals + helper structures
No additional non-standard libraries (e.g., OpenCV is NOT used).