Node: sensor_simulator_node

Header

Team: G4

Created: 18.11.2025

name	date	Short description
Burhan Topaloglu	18.11.2025	created by Burhan Topaloglu (based on assignment specification)
Burhan Topaloglu	25.11.2025	improved integration with other nodes changed message type to ImuSim and published accelerations instead of velocities, new
Melissa van Leeuwen	28-11-2025	added code to declare interval parameters so they can be loaded from YAML

Simulates IMU linear and angular accelerations using time-based polynomial intervals. Publishes ImuSim messages.

Node description

This node simulates IMU acceleration readings for:

linear acceleration: x, y, z

angular acceleration: yaw (z-axis)

Each axis's acceleration is computed as the derivative of a time-dependent polynomial (constant → 0 derivative, linear → constant acceleration, quadratic → varying acceleration).

Intervals are loaded from ROS2 parameters, supporting:

constant

linear

quadratic (via 3-point Lagrange interpolation)

Overlapping intervals are summed.

When all intervals have ended, simulation time loops.

Publishes g425_assign4_interfaces_pkg::msg::ImuSim.

Node sub-objects and functions (communication objects):

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

Wall_timer: main simulation timer Name: implicit timer created via create_wall_timer
bind function: on_timer()
Periodically computes accelerations from intervals and publishes `ImuSim` message.

publisher : imu_sim_acceleration (default) Short description: Publishes simulated IMU accelerations in ImuSim format.
Publisher function: pub_->publish(msg) Publishes computed accelerations each cycle.

Node actions, messages and services:

- Node actions, messages, and services
- Messages:
- g425_assign4_interfaces_pkg::msg::ImuSim
- Fields:
- x → linear acceleration (x-axis)
- y → linear acceleration (y-axis)
- z → linear acceleration (z-axis)
- yaw_z → angular acceleration about z-axis
- stamp

Custom Node functions :

Custom function :load_intervals()
Short description:
Reads parameter sets under intervals.* and forms acceleration intervals.
Validates time ranges and polynomial types.

Custom function: eval_derivative(const Interval&, double t)

Short description:
Computes acceleration as the derivative of the velocity polynomial.
Handles constant, linear, and quadratic polynomials.

Custom function: on_timer()

Short description:
Periodically evaluates current time, loops time if needed, computes accelerations for each axis, and publishes message.

Node implementation (main):

Short description of differences to a standard implementation:
Interval-based acceleration generation.
Computes derivatives of polynomial segments (acceleration instead of velocity).
Supports multiple axes and summing overlapping segments.
Declares up to 10 interval parameter groups dynamically.
Includes time-looping of simulation.
Uses custom message type ImuSim.

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;

}
```

Node-specific implementation:
Same structure, but instantiates SensorSimulator, which includes:
Parameter-based interval loading
Timer-driven publishing
Polynomial derivative evaluation

Node dependencies

g425_assign4_interfaces_pkg
Provides the ImuSim message type used for simulated accelerations.
Standard libraries used:
<rclcpp/rclcpp.hpp> → ROS2 node, logging, timers, parameters
<chrono> → for timer period
<functional>, <vector>, <string>, <set> → container and parsing utilities
<stdexcept> → error handling
Non-standard dependencies:
None.
(No external libraries like OpenCV or Eigen are used.)