Node: DatabasePublisher

Header

Team: Groep 4 RMB

Created: 02.12.2025 by Rik van Velzen

name	date	Short description
DatabasePublisher	25.11.2025	This node subscribes to both Mecanum wheel (position and velocity) data and IMU (acceleration, velocity and position) data, processes the incoming messages, and stores them in a database.

Node description

The DatabasePublisher node receives data from two subsystems:

- Mecanum data – wheel velocities and calculated positions.
- IMU data – acceleration, calculated velocities, and calculated positions.

It converts the incoming ROS2 messages into internal database structures defined in OdometryDatabase.hpp and stores them.

The node logs every received message to the console and allows topic names to be configured via ROS2 parameters for flexible integration with other nodes.

The node has only multiple subscribers and a database writer.

Node sub-objects and functions (communication objects):

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

subscriber : mecanum_sub_pos_ Receives mecanum-computed position data (x, y, yaw_z).
Callback function: publish_mecanum_pos Stores positions into database using addPositionmecanum().

subscriber : mecanum_sub_velocity_ Receives simulated mecanum angular velocity data (wfl, wfr, wrl, wrr).
Callback function: publish_mecanum_pos Stores velocities into database using addvelocitymecanum().

subscriber : imu_sim_sub_pos_ Receives imu-computed position data (x, y, yaw_z).
Callback function: publish_imu_sim_pos Stores positions into database using addPositionImuSim().

subscriber : imu_sim_sub_velocity_ Receives imu-computed velocity data (x, y, yaw_z).
Callback function: publish_imu_sim_velocity Stores velocities into database using addvelocityImuSim().

subscriber : imu_sim_sub_acceleration_ Receives imu-simulated acceleration data (x, y, yaw_z).
Callback function: publish_imu_sim_acceleration Stores accelerations into database using addaccelerationImuSim().

Node actions, messages and services:

Messages:

- Subscribed Message Type: g425_assign4_interfaces_pkg::msg::PositionData
- Subscribed Message Type: g425_assign4_interfaces_pkg::msg::Mecanum
- Subscribed Message Type: g425_assign4_interfaces_pkg::msg::ImuSim

Topics:

- mecanum_position: Calculated position data for storage
- mecanum_velocity: simulated mecanum velocity data
- imu_sim_position: Calculated position data for storage
- imu_sim_velocity: Calculated velocity data for storage
- imu_sim_acceleration: simulated IMU data

Custom Node functions :

Custom function: declare_parameters()

Declares ROS2 parameters for all subscribed topic names and loads their values into class variables.

Custom function: publish_mecanum_pos()

Converts PositionData msg into a DBT_Positions structure and stores it in the database.

Custom function: publish_mecanum_velocity()

Converts Mecanum msg into a DBT_Mecanum structure and stores it in the database.

Custom function: publish_imu_sim_pos()

Converts PositionData msg into a DBT_Positions structure and stores it in the database.

Custom function: publish_imu_sim_velocity()

Converts ImuSim msg into a DBT_Measurement structure and stores it in the database.

Custom function: publish_imu_sim_acceleration()

Converts ImuSim msg into a DBT_Measurement structure and stores it in the database.

Node implementation (main):

This node follows a standard single-threaded ROS2 node implementation.

No timers, services, multithreading, or action servers are used.

The only added logic is the exclusion of “main()” and “private:” during unit tests using the macro TESTING_EXCLUDE_MAIN.

Standard implementation:

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

{

  rclcpp::init(argc, argv);
  auto node = std::make_shared<DatabasePublisher>();
  rclcpp::spin(node);
  rclcpp::shutdown();
  return 0;
}
```

Node dependencies :

g425_assign4_interfaces_pkg/msg/mecanum.hpp: Custom message used to receive wheel velocity data to be put into a database.

g425_assign4_interfaces_pkg/msg/position_data.hpp: Custom message used to receive position data to be put into a database.

g425_assign4_interfaces_pkg/msg/imu_sim.hpp: Custom message used to receive acceleration data to be put into a database.

g425_assign4_pkg/OdometryDatabase.hpp: Custom library with different functions used to put data into the database.

rclcpp: Standard ROS2 C++ client library for node, subscription, and publisher functionality.