ENPM663: Building a Manufacturing Robotic Software System

RWA 1

v1.0

Lecturer: Z. Kootbally

School: University of Maryland

Semester/Year: Spring/2025

Table of Contents

Changelog

- **v1.0**: Original version.

Conventions

Conventions

bad practice	👎
best practice	👍
code syntax	</>
example	✈
exercise	✎
file	📄
folder	📂
guideline	🚀
note	✍
question	?
task	☰
terminology	📜
warning	🔥
web link	link
package	🧰

ROS node	n
ROS topic	t
ROS message	m
ROS parameter	p
ROS frame	f

- C++ class attributes have a trailing underscore, e.g., `std::string camera_name_;`
- Python class attributes have a leading underscore, e.g., `self._camera_name`

Guidelines

Guidelines

This assignment must be completed **individually**. Ensure compliance with all specified guidelines, as failure to follow any part of these guidelines will lead to a grade of **zero** for the assignment.

- Do not reuse a package obtained from peers.
- Keep your work confidential and refrain from sharing it with peers.
  - If you use github, you have to make your repository private.
- While discussing assignment challenges is encouraged, refrain from exchanging solutions with peers.
- Do not use code generated by AI tools.
  - The potential risks outweigh the benefits.

Objectives & Skills Practiced

Objectives & Skills Practiced

- **ROS 2 Node Implementation** – Writing Python scripts or C++ code for ROS 2 nodes.
- **Publisher and Subscriber Implementation** – Writing publisher and subscriber components.
- **Topic Communication** – Ensuring proper topic naming and message flow between nodes.
- **Timer-Based Publishing** – Using timers to publish messages at a fixed rate.
- **Launch File Development** – Writing Python-based ROS 2 launch files.
- **Software Deployment** – Configuring ***CMakeLists.txt*** and ***package.xml*** for installation.
- **Software Deployment** – Use ROS 2 CLI for node execution, introspection, and debugging.

Students may complete the assignment using Python, C++, or a combination of both. If you choose to only use Python for the assignment, you still need to create C++ ROS packages.

Packages

Packages

Each student must create **three** separate ROS C++ packages, following the naming format:

- ***rwa1_exercise#***

Where **#** represents the exercise number (1 to 3).

One suggestion is to create each package in a folder (similar to the package structure used in lecture 3).

```
enpm663_ws
└─ src
   └─ rwa1_<last name>
      ├─ rwa1_exercise1
      ├─ rwa1_exercise2
      └─ rwa1_exercise3
```

Deliverable

Deliverable

Compress ***rwa1_<last name>*** into **a single file** named ***rwa1_<last name>.<extension>***[1], where ***<last name>*** is the student's last name, and submit it on Canvas by the due date.

Late submissions will incur a penalty according to the guidelines specified in the syllabus, with exceptions for any valid reason.

- Valid reasons include a doctor's note, proof of travel, or a note from a professor/MAGE.

Students with special circumstances may submit their assignments late without incurring any penalties. However, it is required that these students inform me in advance of their intention to submit their work past the deadline.

[1] ***<extension>*** can be ***zip***, ***tar***, ***tar.gz***, etc

Exercises

Exercises

- **Exercise #1**: Implementing bi-directional communication between two nodes.
  - Package: *rwa1_exercise1*
- **Exercise #2**: Launching multiple nodes with conditional execution.
  - Package: *rwa1_exercise2*
- **Exercise #3**: Grouping and managing nodes with remapping.
  - Package: *rwa1_exercise3*

Exercise #1

Develop two nodes where each node publishes data to a topic and subscribes to the topic from the other node. Both nodes publish at a rate of 1 Hz.

- Implement the first node n number_publisher
  - Publishes an integer from 1 to 20 to the topic t numbers, restarting at 1.
  - Subscribes to the topic t letters and prints received messages.
- Implement the second node n alphabet_publisher
  - Publishes letters from ***a*** to ***z*** to the topic t letters, restarting at ***a***.
  - Subscribes to the t numbers topic and prints received messages.
- Example output:

```
number_publisher receives: b
alphabet_publisher receives: 3
```

- Create the launch file ***exercise1.launch.py*** to start both nodes simultaneously.
- Run the launch file and verify:
  - n number_publisher publishes numbers cyclically.
  - n alphabet_publisher publishes letters cyclically.
  - Each node properly subscribes and prints received messages.

Exercise #2

Create a launch file that starts a publisher and a subscriber but allows the publisher to be conditionally enabled based on a launch argument.

- Implement the first node n counter_publisher
  - Publishes messages at a rate of 2 Hz (2 messages per second) to the topic t counter. Each message follows the format ***<N>: exercise 2*** where ***<N>*** is a number that starts at 0 and increases by 1 with each new message.
- Implement the second node n counter_subscriber
  - Subscribes to the t counter topic and prints received messages with the format: ***<node name> receives: <data>***
    - Example output:

```
counter_subscriber receives: '1: exercise 2'
counter_subscriber receives: '2: exercise 2'
```

- Create a launch file ***exercise2.launch.py*** that:
  - Starts both nodes.
  - Accepts a launch argument that determines whether the publisher should run.
  - Uses conditional execution to enable or disable the publisher.

Exercise #3

Launch multiple nodes as a group while remapping topics based on a launch argument.

- Implement the first node n camera_publisher
  - Publishes simulated camera data at a rate of 2 Hz (2 messages per second) to the topic t camera. Published messages are of type m sensor_msgs/msg/Image. Reuse code from `publish_camera_data()` from ***camera_demo.cpp***
- Implement the second node n camera_subscriber
  - Subscribes to the t camera topic and prints received messages with the format: ***<node name> receives camera data***
    - Example output:

```
camera_subscriber receives camera data
```

- Create the launch file ***exercise3.launch.py*** that:
  - Groups both nodes together. See ***demo5.launch.py***
  - Accepts a launch argument that is used to remap the topic t camera. See ***demo8.launch.py***
  - Remaps the t camera topic using the launch argument.
  - Ensures that n camera_subscriber listens to the remapped topic.

Comments & Documentation

Comments

Include comments in your program so that the user (me) understands what is happening.

Documentation

Docstring documentation (Python) or Doxygen documentation (C++) is required. Ensure you document classes, methods, and functions.

Grading Rubric ▶ Exercise #1

Exercise #1 (10 points)

- **Core functionality (6 points)**:
  - Implements two nodes, each with both a publisher and a subscriber (**2 points**).
  - Publishes numbers (***1*** to ***20***, restarting at ***1***) on t numbers (**1 point**).
  - Publishes letters (***a*** to ***z***, restarting at ***a***) on t letters (**1 point**).
  - Subscribers correctly receive and print messages in the format: ***<node name> receives: <data>*** (**2 points**).
- **Launch File Implementation (2 points)**:
  - Creates ***exercise1.launch.py*** and correctly starts both nodes (**2 points**).
- **Package Configuration (2 points)**:
  - Updates ***package.xml*** with package description and maintainer (**1 point**).
  - Updates ***CMakeLists.txt*** to properly install executables and launch files (**1 point**).

Grading Rubric ▶ Exercise #2

Exercise #2 (10 points)

- **Core functionality (6 points)**:
  - Implements a publisher that publishes messages at 2 Hz (**2 points**).
  - Implements a subscriber that listens and prints received messages (**2 points**).
  - Messages follow the format: ***<N>: exercise 2*** and ***<node name> receives: <data>*** (**2 points**).
- **Launch File Implementation (2 points)**:
  - Implements conditional execution to enable or disable the publisher using a launch argument (**2 points**).
- **Package Configuration (2 points)**:
  - Updates ***package.xml*** with package description and maintainer (**1 point**).
  - Updates ***CMakeLists.txt*** to properly install executables and launch files (**1 point**).

Exercise #3 (10 points)

- **Core functionality (6 points)**:
  - Implements a camera publisher that publishes simulated image data (2 points).
  - Implements a camera subscriber that listens for data and prints messages (2 points).
  - Messages follow the format: ***<node name> receives camera data*** (2 points).
- **Launch File Implementation (2 points)**:
  - Implements topic remapping using a launch argument (2 points).
- **Package Configuration (2 points)**:
  - Updates ***package.xml*** with package description and maintainer (1 point).
  - Updates ***CMakeLists.txt*** to properly install executables and launch files (1 point).

Deductions

- Late submission: **As per syllabus guidelines**.
- Use of AI-generated code: Assignment grade of **zero**.
- Code sharing/plagiarism: Assignment grade of **zero**.
- Non-private GitHub repository: Assignment grade of **zero**.