better_launch: A better replacement for the ROS2 launch system

Nikolas Dahn [1]

1 German Research Center for Artificial Intelligence, Germany

DOI: 10.21105/joss.08958

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @eramakanth
- @lwilson

Submitted: 01 August 2025
Published: 24 February 2026

Summary

Today, most robots use middleware in order to manage and connect the various pieces of software they need for fulfilling their tasks. Over the last years, the most popular middleware - the Robot Operating System (ROS) - has received a complete overhaul and been released as ROS2. However, the reception so far has been mixed. While the code base has seen several important improvements, this reinvention has also come with severe downgrades in terms of user friendliness and usability. One of the biggest and most exposed offenders is the launch system, responsible for orchestrating the startup and execution of the robot's software components. The software presented in this paper - *better_launch* - is a complete replacement for the ROS2 launch system and solves several technical and usability issues. It is lightweight, intuitive, convenient, fully documented, and comes with many examples.

Statement of need

On one hand, being able to write launch files in Python is a nice feature that does away with many of the limitations of the old ROS1 xml launch files. On the other hand, the current implementation tries to use Python as a declarative language, forcing users to dance around unintuitive constructs as none of the values and nodes exist yet at the time of execution. Aside from its wordiness and significant challenge to new users, it comes with several anti patterns and technical issues, too:

- execution order of actions is non-deterministic
- dynamic code execution (e.g., passing code as a string for later execution)
- resists linting as many arguments have to be passed as strings
- resists automated code analysis as actions don't have a unified way of exposing internals
- frequent zombie processes
- convoluted and obscure internals

There are, of course, good reasons for the way the launch system has been implemented, at least on a superficial level. According to the design document (Woodall, 2019), the intent is to treat launch files as "a description of what will happen" without executing anything. This is so that tools can "visualize and modify the launch description". The recently released LaunchMap (Mahna & Costa Silva, 2025) is able to do just that. However, given that users may define their own launch actions without a common way of inspecting them, it could be argued that even this use case is not well supported right now. There is also no good argument why the same couldn't be achieved using, e.g., Python's inspect module and/or a non-declarative syntax.

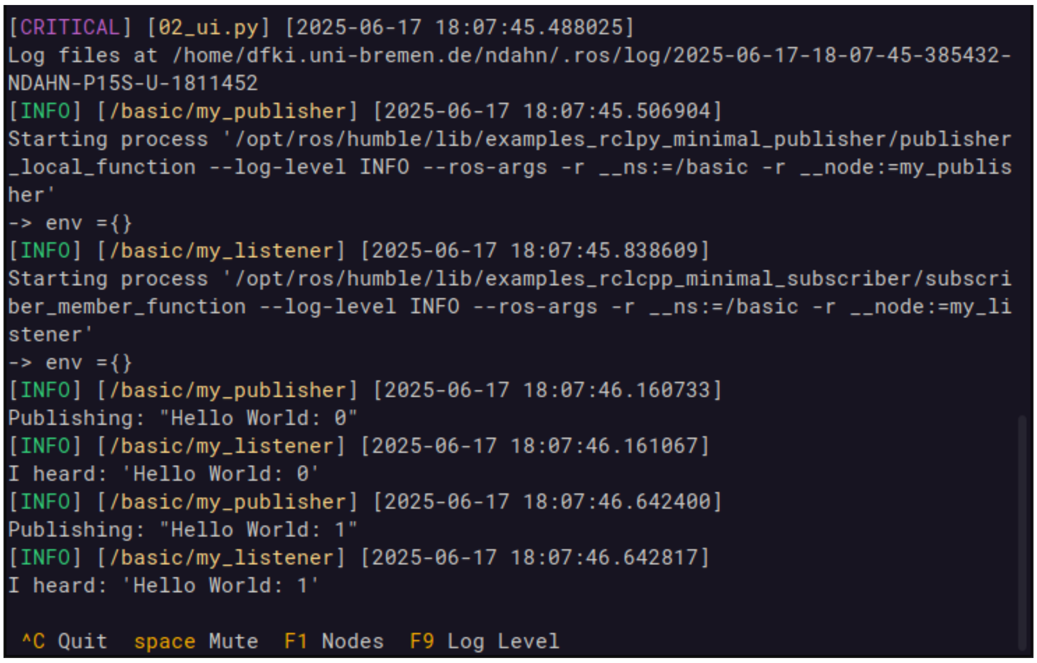
Existing Remedies

The issues in user friendliness have led to the emergence of several packages that simplify writing launch files, e.g., Simple_launch (Kermorgant et al., 2020) and Launch-generator (Sakaguchi, 2023), while other packages like the popular Generate_parameter_library (PickNikRobotics, 2022) are dedicated to particular aspects of the launch process. However, in the end they all face the same issues mentioned above, as they just represent different ways of generating the launch descriptions.

better_launch

In order to actually resolve these issues, I have written *better_launch*, a complete and self-contained replacement for the ROS2 launch system with no dependencies on the existing facilities. Getting started with *better_launch* is easy as it is well-documented and comes with examples for various common use cases. It enables developers to write simple and intuitive launch files using regular Pythonic syntax (see example below). Crucially, it addresses the above issues as follows:

- Any actions taken in the launch files are executed immediately, allowing direct and meaningful interactions with nodes, topics, and services as the launch process develops. This allows, e.g., adjusting one node's parameters based on another node's published topics.
- Arguments to the launch file are "declared" in the form of function arguments that are passed as natural types, enabling common Python syntax like if-else.
- *better_launch* launch files are fully compatible with the ROS2 launch system. This means they can be started through `ros2 launch`, include regular ROS2 launch files, and even get included from regular ROS2 launch files.
- The entire launch logic is contained within a single, fully documented package, which includes examples. Convenience functions exist for various common tasks like starting a `joint_state_publisher` or bridging Gazebo topics.
- *better_launch* comes with a replacement for `ros2 launch` called `bl`, which is faster than its counterpart and provides additional features. For example, `bl` enables auto completion for launch parameters and uses the launch function's docstring to generate `--help` text.
- To improve usability, *better_launch* reformats and colors terminal output by default.
- Unless killed with SIGKILL, *better_launch* will not leave zombie processes behind.
- *better_launch* provides an optional and unobtrusive **terminal UI**, reminiscent of Rosmon (Schwarz & others, 2015), which can be used for stopping and restarting nodes, triggering life cycle transitions, listing a node's subscribed topics, dynamically adjusting the logging level, and more.



```
[CRITICAL] [02_ui.py] [2025-06-17 18:07:45.488025]
Log files at /home/dfki.uni-bremen.de/ndahn/.ros/log/2025-06-17-18-07-45-385432-
NDAHN-P15S-U-1811452
[INFO] [/basic/my_publisher] [2025-06-17 18:07:45.506904]
Starting process '/opt/ros/humble/lib/examples_rclpy_minimal_publisher/publisher
_local_function --log-level INFO --ros-args -r __ns:=/basic -r __node:=my_publis
her'
-> env ={}
[INFO] [/basic/my_listener] [2025-06-17 18:07:45.838609]
Starting process '/opt/ros/humble/lib/examples_rclcpp_minimal_subscriber/subscri
ber_member_function --log-level INFO --ros-args -r __ns:=/basic -r __node:=my_li
stener'
-> env ={}
[INFO] [/basic/my_publisher] [2025-06-17 18:07:46.160733]
Publishing: "Hello World: 0"
[INFO] [/basic/my_listener] [2025-06-17 18:07:46.161067]
I heard: 'Hello World: 0'
[INFO] [/basic/my_publisher] [2025-06-17 18:07:46.642400]
Publishing: "Hello World: 1"
[INFO] [/basic/my_listener] [2025-06-17 18:07:46.642817]
I heard: 'Hello World: 1'

 ^C Quit  space Mute  F1 Nodes  F9 Log Level
```

Figure 1: Screenshot of the TUI

We consider *better_launch* mature enough for general use in research applications, with performance similar or better than `ros2 launch` (benchmarks in repo) (Bloomberg Engineering, 2025; Frederickson, 2025; Rodola, 2025). It is under active development and can be downloaded for free from https://github.com/dfki-ric/better_launch. We hope that *better_launch* will advance the state of ROS2 in a meaningful way.

Example

The ROS2 tutorials provide an example (Open Robotics, 2025) for running a turtlebot simulation. This launch file creates a node, then calls one of its services and updates a parameter. Ironically, due to the asynchronous execution, the parameter update usually fails because the node has not come up yet. *better_launch* does not have this issue, and can in addition express the same launch file with only 28 instead of 73 lines - including documentation!

```python
#!/usr/bin/env python3
from better_launch import BetterLaunch, launch_this

@launch_this
def great_atuin(
    turtlesim_ns: str = "turtlesim1",
    use_provided_red: bool = True,
    new_background_r: int = 200,
):
    """ This docstring will also be used to create text for --help!"""
    bl = BetterLaunch()

    with bl.group(turtlesim_ns):
        turtle_node = bl.node(
            package="turtlesim",
            executable="turtlesim_node",
            name="sim",
            params={"background_r": 120},
        )

    bl.call_service(
        topic=f"/{turtlesim_ns}/spawn",
        service_type="turtlesim/srv/Spawn",
        request_args={"x": 2.0, "y": 2.0, "theta": 0.2},
    )

    if use_provided_red:
        turtle_node.is_ros2_connected(timeout=None)
        turtle_node.set_live_params({"background_r": new_background_r})
```

AI usage disclosure

No generative AI tools were used in the development of this software, the writing of this manuscript, or the preparation of supporting materials.

References

Bloomberg Engineering. (2025). Memray. In *Github repository*. Github. https://github.com/bloomberg/memray

Frederickson, B. (2025). py-spy. In *Github repository*. Github. https://github.com/benfred/py-spy

Kermorgant, O., Wen, H., & Pocedulić, V. (2020). simple_launch. In *GitHub repository*. GitHub. https://github.com/oKermorgant/simple_launch

Mahna, S., & Costa Silva, L. P. da. (2025). LaunchMap – visualize your ROS 2 launch files. In *GitHub repository*. GitHub. https://github.com/Kodo-Robotics/launchmap

Open Robotics. (2025). Using substitutions. In *ROS 2 Tutorials*. Open Robotics. https://docs.ros.org/en/jazzy/Tutorials/Intermediate/Launch/Using-Substitutions.html

PickNikRobotics. (2022). generate_parameter_library. In *GitHub repository*. GitHub. https://github.com/PickNikRobotics/generate_parameter_library

Rodola, G. (2025). psutil. In *Github repository*. Github. https://github.com/giampaolo/psutil

Sakaguchi, T. (2023). launch-generator. In *GitHub repository*. GitHub. https://github.com/Tacha-S/launch_generator

Schwarz, M., & others. (2015). rosmon. In *GitHub repository*. GitHub. https://github.com/xqms/rosmon

Woodall, W. (2019). ROS 2 launch system. In *ROS 2 Design*. Open Source Robotics Foundation. https://design.ros2.org/articles/roslaunch.html