1. Theoretical background

- Official tutorial
- Oficial documentation
- Is a system for executing multiple nodes in one step with an specific **configuration**
- ROS2 launch system help the user to **describe the configuration (describe its dynamics)** of their system and then execute it as described. The **configuration** of the system includes:
  - What programs to run
  - Where to run them
  - What arguments to pass them
  - ROS specific conventions which make it easy to reuse components throughout the system by giving them each different configurations
- Launch files written in Python can start and stop different nodes as well as trigger and act on various events.
- The package providing this framework is **launch_ros**, which uses the non-ROS-specific launch framework underneath.

2. Launch file

- Naming:
  - It is a python file, so its extension is **.py**
  - It is recommended to add the suffix **.launch.py**
  - **Must be executable**
- Directory: Inside a folder called **launch**

2.1 Setup

- Construct the launch file
- Modify the package.xml file to add the ros2launch dependency
- If it is a C++ package:
  - Modify the CMakeLists file to install the launch directory at the install/.../share folder.
- If it is a Python package:
  - Modify the setup.py file to install the launch file(s) install/.../share folder.
- Build the package

2.1 Basic structure

- Import the relevant modules and classes
- Create a generate_launch_description function
  - That must be the exact name
  - Everyting else is going to be inside
- Instanciate a LaunchDescription object
- Define and add the nodes to be launched to the LaunchDescription object
- Optionally, use other actions, substitutions and event handlers if needed
- Return the LaunchDescription object

2.3 Basic elements

- LaunchDescrition class:
  - Main object return by the file
  - Contents the configuration description, i.e. the launch dynamics

```
from launch import LaunchDescription
```

- Node class:

```
from launch_ros.actions import Node
```

  - Define a node to be executed
  - Attributes:
    - package:
    - namespace: To avoid name collisions
    - executable: As defined in the **setup.py entry-point** or in the **CMakeList.txt**
    - name: To **rename** the node. Used for start 2 nodes from the same executable.
    - remappings:
      - **List of tuples** to change the names of topics, services, action-services.
      - Every tuple is:

```
remappings = [(old_name1, new_name1),
              (...),
              ...]
```

    - parameters: **List of dictionaries**

```
parameters = [{<parameter>: <new_value>},
              {...},
              ...]
```

    - arguments: As it where executed fom the CLI
    - output: Where to show output messages. Could be: 'screen'
  - Add a node to the description to be launched

```
description = LaunchDescription()
...
node = Node(...)
description.add_action(<node>, ...)
```

2.3 Actions

- Every action **must be added to the description**

- Launch argument:
  - An argument that can be modified from the CLI at launch-time.
  - Creates an instance defined by a **name** , **default_value** and **description**

```
from launch.actions import DeclareLaunchArgument
...
launch_argument = DeclareLaunchArgument(<'name'>,
default_value = <'value'>, description =
<'description'>)
...
# launch_argument must be added to the description
description = LaunchDescription()
...
description.add_action(<launch_argument>, ...)
```

2.4 Substitutions

- Launch configuration:
  - Similar to a **variable** that can be modified from the outside of the program (CLI) at launch-time as an argument or at the run

time

- Creates an instance defined by a **name** and a **variable value**
- It can be binded to a **Launch argument action** by **equating the names** to make it configurable at launch-time

```
        from launch.substitutions import
LaunchConfiguration
        ...

        launch_configuration =
LaunchConfiguration('name', default = 'value')
```

2.5 Event handlers

Alternative information sources

Robotics Back-End
Automatic Addison 1
Automatic Addison 2