Challenges

Mini challenge 2

{Learn, Create, Innovate};

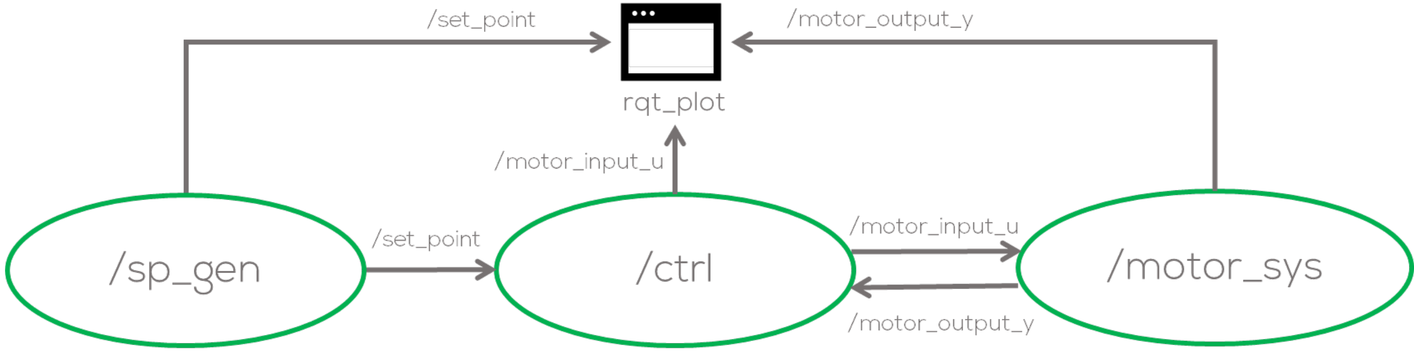
Manchester **Robotics**

Mini Challenge 2

Introduction

This mini-challenge is intended for the student to review the concepts introduced in the previous sessions.

- The activity involves creating a controller for ROS's simulated DC motor (/motor_sys) Used during Activity 3.
- The "/motor_sys" node, and a "/sp_gen" simple program structure (not mandatory) are provided by MCR2.
- The controller can be "P", "PI" or "PID" controller (other controllers can be accepted upon agreement with the professor.).



Mini Challenge 2

Instructions

- Download the package “motor_control” from the mini-challenge folder. You can use the package developed during the class just be CAREFUL with the Topics of the sp_node.
- Save and compile the file

```
$ cd ~/ros2_ws
$ colcon build
$ source install/setup.bash
```

- Launch the node

```
$ ros2 launch motor_control motor_launch.py
```

- Open the rqt_graph and rqt_plot

```
$ ros2 run rqt_graph rqt_graph
```

```
$ ros2 run rqt_plot rqt_plot
```

- Publish a message to test that everything is working

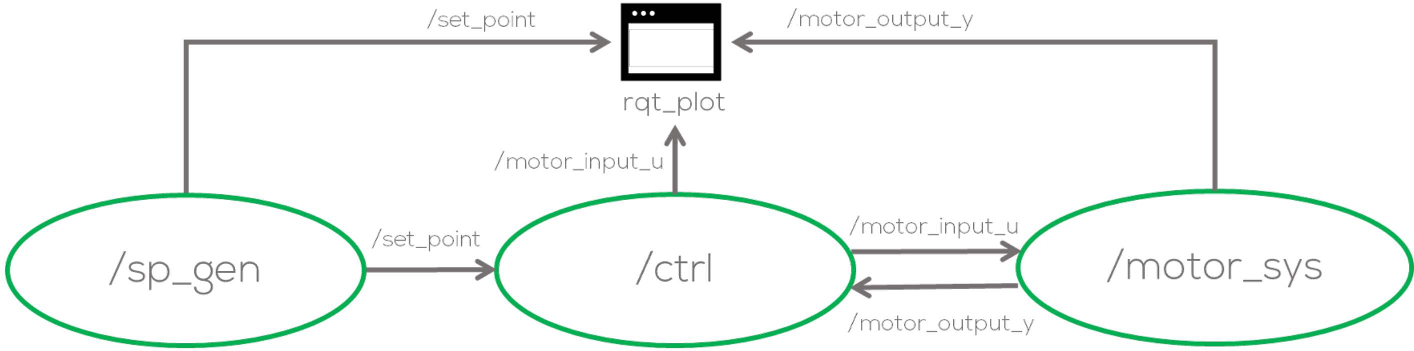
```
$ ros2 topic pub /motor_input_u
std_msgs/msg/Float32 "data: 5.0"
```

- If using the template, the nodes should appear disconnected. If using the package form the class activity the nodes will be connected via the the “motor_input_u” topic. CHANGE IT!! To control it to the controller.

Mini Challenge 2

Instructions

- In this challenge the student must generate a new node called “controller”
- The student must make all the changes necessary to the package to generate the feedback structure.
- The node must subscribe to the “set_point” node and publish to the “motor_node”.
  - If using your nodes from the class activity, modify the “set_point” node to change the topic from “motor_input_u” to “set_point”
  - If using the template, this is already done for you.



Mini Challenge 2

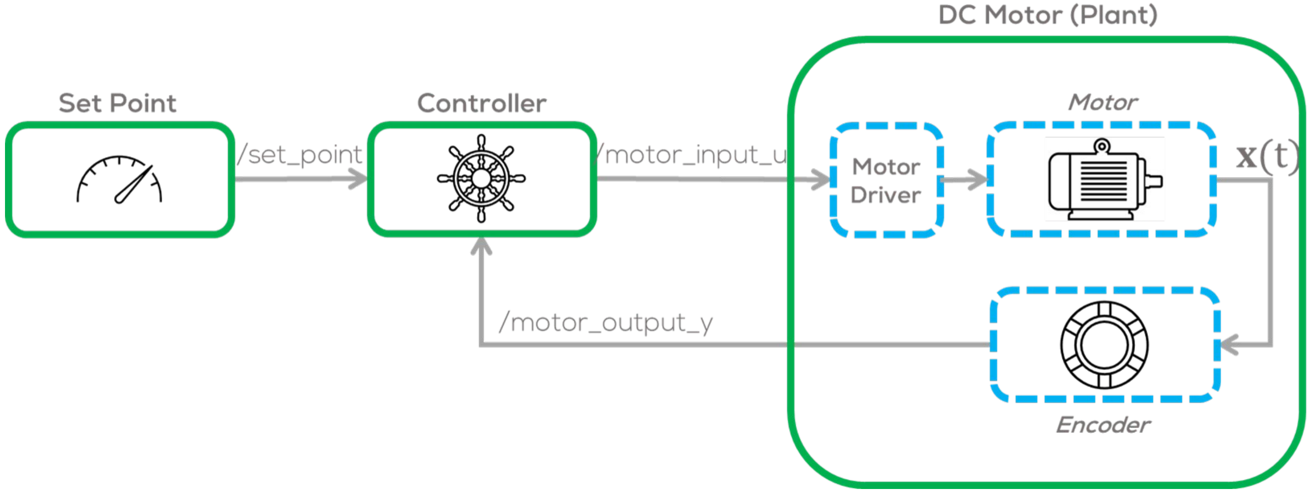
motor_sys node

The “process node” is a node made by MCR2, that simulates a first order system of the form

(1)

Approximating a simple DC Motor with a motor driver and an encoder.

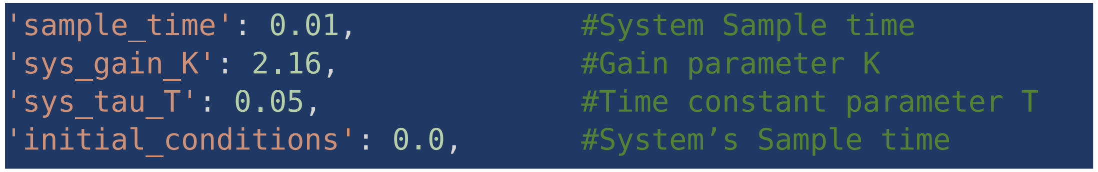
- The input and output of the node are simple messages Float32.
- <u>The student must use these messages to communicate with the system.</u>



Mini Challenge 2

motor_sys node

- The system parameters can be set from a launch file “motor_launch”
- The node’s parameters are based on Eq. 1.
- It is suggested that for this exercise the parameters remain unchanged.



```
'sample_time': 0.01,               #System Sample time
'sys_gain_K': 2.16,                #Gain parameter K
'sys_tau_T': 0.05,                 #Time constant parameter T
'initial_conditions': 0.0,         #System’s Sample time
```

Hints

- It is encouraged to analyse the system before using it with the controller.
- Sending different input signals to verify its behaviour.
- The system can be tested by using ROS command line tools as follows.

```
$ ros2 topic pub /motor_input_u
std_msgs/msg/Float32 "data: 5.0"
```

Mini Challenge 2

Controller Node

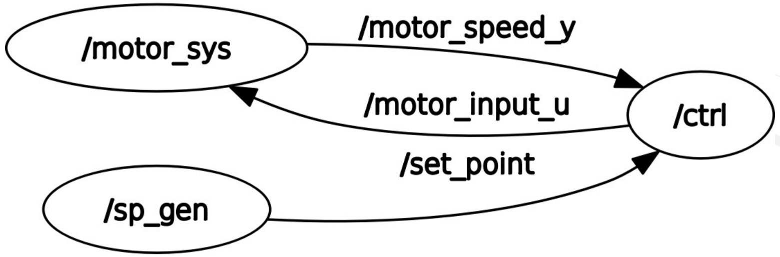
1. Make a node called “/ctrl” to generate a control input to the “/motor_sys” node.
2. The node must publish in the “/motor_input_u” topic and subscribe to the “/motor_output_y” and “/set_point” topics.
3. The control node parameters must be set in the launch file or as a config file (YAML).
4. The user must be able to tune the parameter at runtime, and if a parameter is incorrect, the node must let the user know.
5. The sampling time and rate can be the same as the “/system” node.
6. **It is strictly forbidden to use any other python library, other than NumPy. The controller must be made without using any predefined online controllers.**

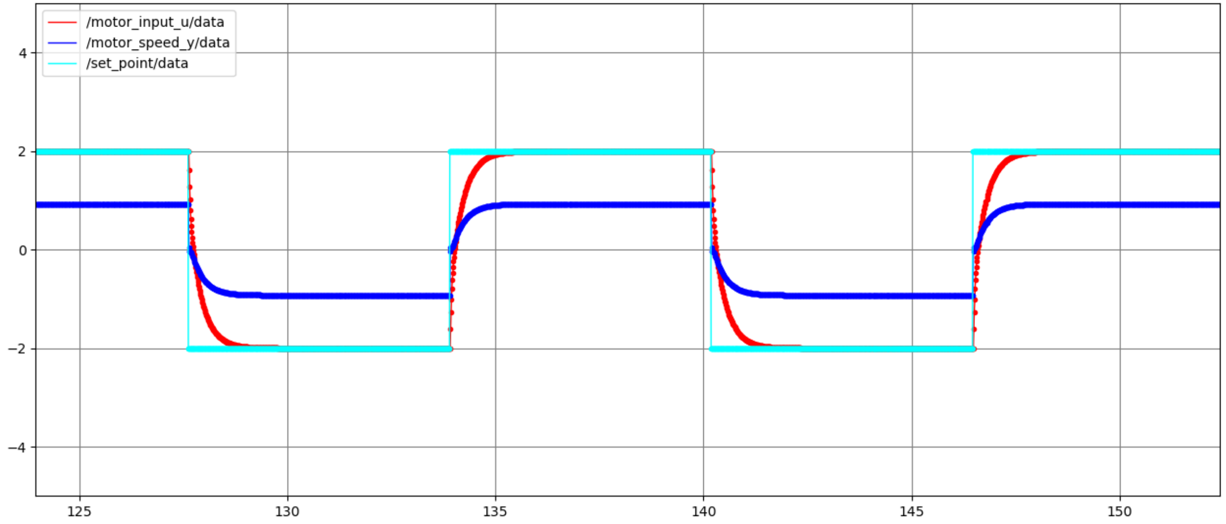
Mini Challenge 2

Launch File

- Make or modify a Launch file that opens and runs the three nodes continuously.
- The parameters of the three nodes must be set in the launch file.
- Use the rqt_plot and rqt_graphs to visualise the results.
- Use the rqt_reconfigure to modify the parameters at runtime.

Expected Results





Mini Challenge 2

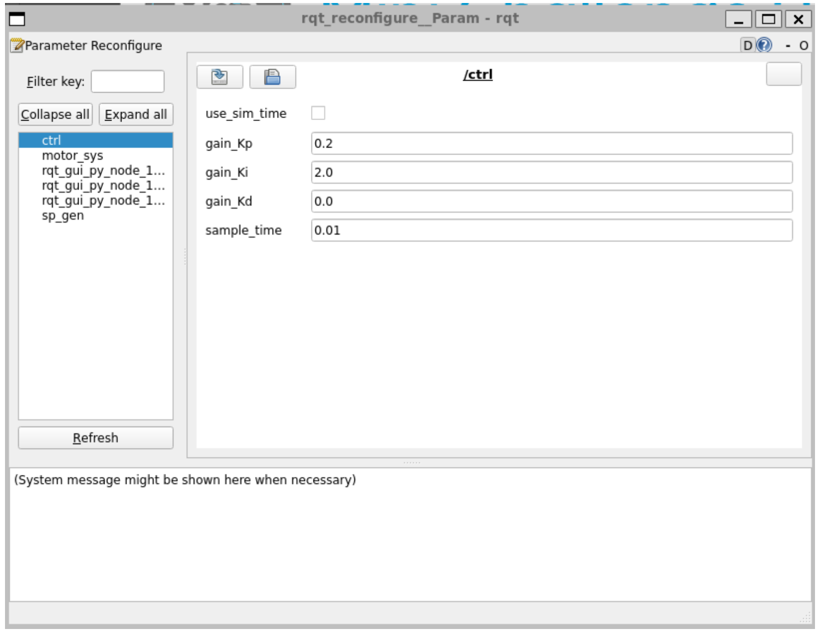
Hints

Discrete PID controller:

where are the controller output and error at time step , such that time where is the sampling time. are the proportional, integral and derivative gains, respectively. More information here.

- Use rqt_reconfigure to modify the parameters at runtime and tune the PID controller.

```
$ ros2 run rqt_reconfigure rqt_reconfigure
```



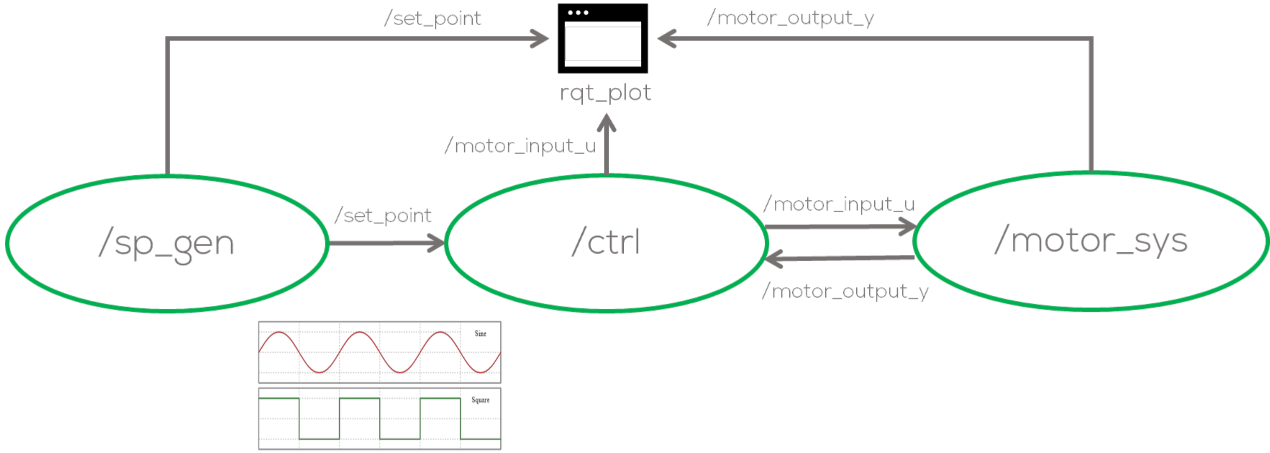
use_sim_time □
gain_Kp 0.2
gain_Ki 2.0
gain_Kd 0.0
sample_time 0.01

(System message might be shown here when necessary)

Mini Challenge 2 (Extension)

Instructions

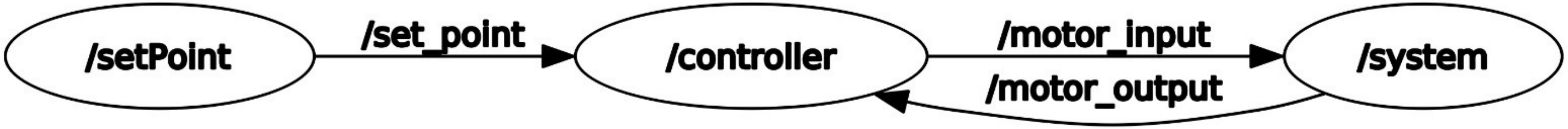
- Use and modify the “/sp_gen” node to generate different set point signals.
  - The set point generator can be a sinusoidal signal, square signal, etc.
  - The user must be able to set the type of signal at runtime must set the type of signals.
  - Set a parameter in charge of this change.
  - As before, It is forbidden to use any libraries, except from NumPy, for this exercise.
- Make the necessary plots to analyse the system in rqt_plot

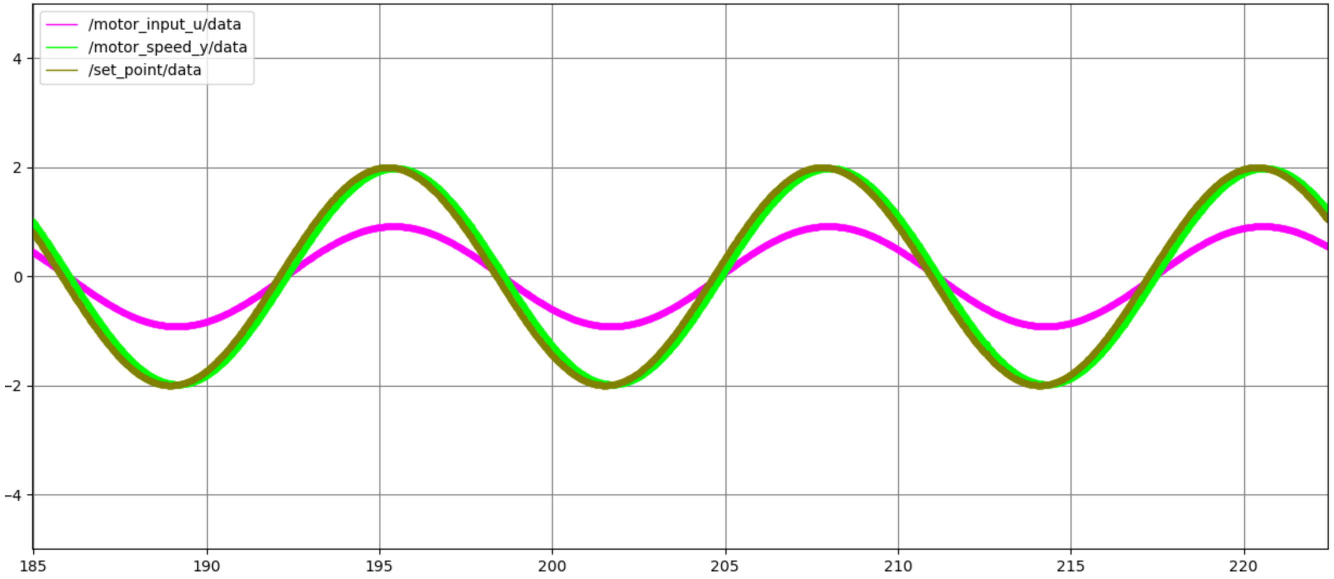


Mini Challenge 2 (Extension)

Expected results

Nodes used for this exercise

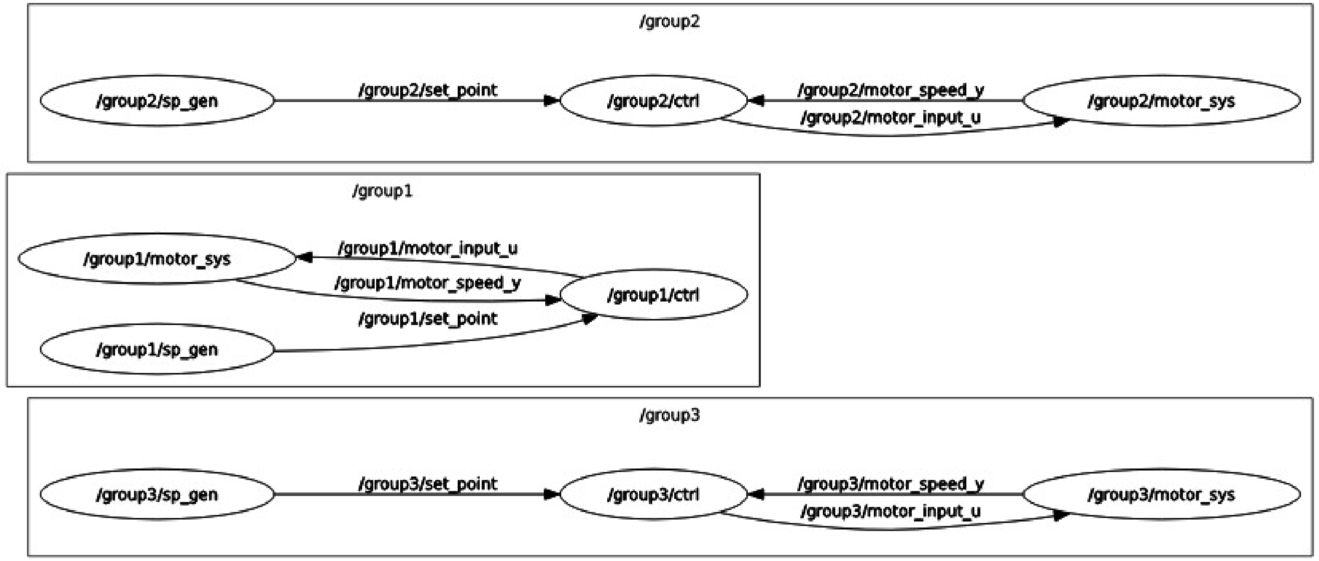




Mini Challenge 2 (Extension)

Instructions

- Make a new Launch called “challenge_launch.py” file to generate three motor control groups.
- Use namespaces to verify that the data is sent to the appropriate nodes.
- Namespaces should be “group1”, “group2” and “group3”

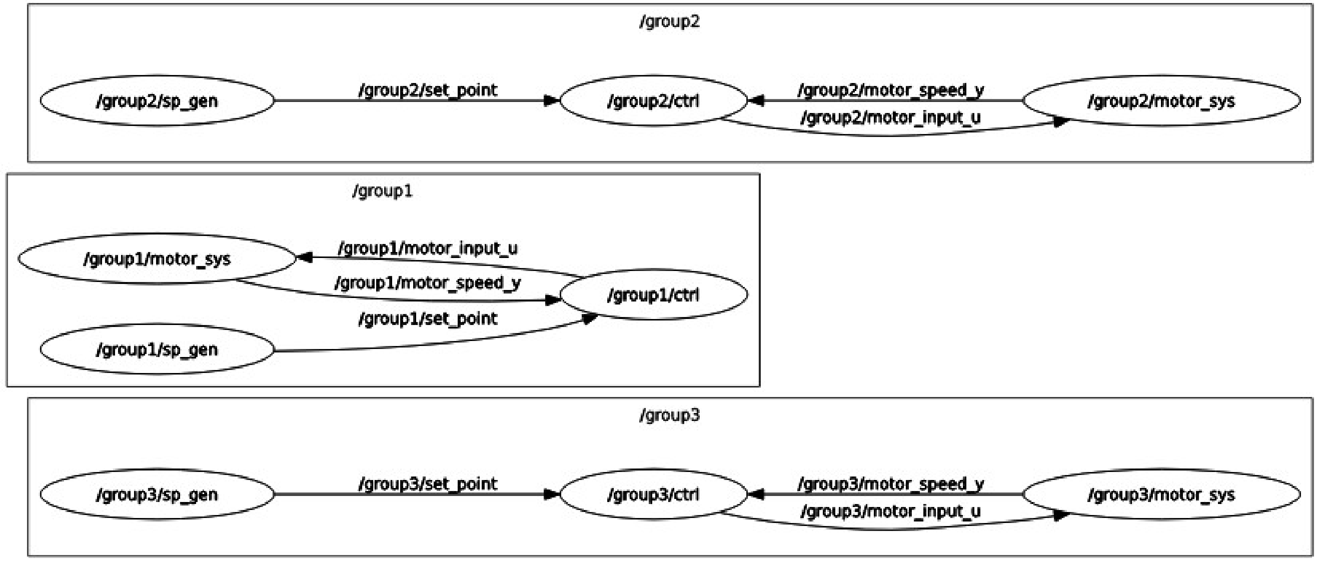


Mini Challenge 2 (Extra)

Instructions

- Use services to start and stop the “motor_sys” and the “ctrl” nodes.
- You can activate the nodes however you want, i.e., in the constructor of the “sp_gen” or using the keyboard so that the user presses a key or writes a word to start activating all the nodes.

This part of the exercise is not mandatory but will help you practice the concept of services.



Rules

- This is challenge **not** a class. The students are encouraged to research, improve tune explain their algorithms by themselves.
- MCR2(Manchester Robotics) Reserves the right to answer a question if it is determined that the questions contains partially or totally an answer.
- The students are welcomed to ask only about the theoretical aspect of the classed.
- No remote control or any other form of human interaction with the simulator or ROS is allowed (except at the start when launching the files).
- It is **forbidden** to use any other internet libraires with the exception of standard libraires or NumPy.
- If in doubt about libraires please ask any teaching assistant.
- Improvements to the algorithms are encouraged and may be used as long as the students provide the reasons and a detailed explanation on the improvements.
- All the students must be respectful towards each other and abide by the previously defined rules.
- Manchester Robotics reserves the right to provide any form of grading. Grading and grading methodology are done by the professor in charge of the unit.