{Learn, Create, Innovate};

Challenges

Mini challenge 2

MCR2
Manchester **Robotics**

Mini challenge 2

This challenge is intended for the student to review the concepts introduced in this week. The challenge is divided into three parts:

Part 1:

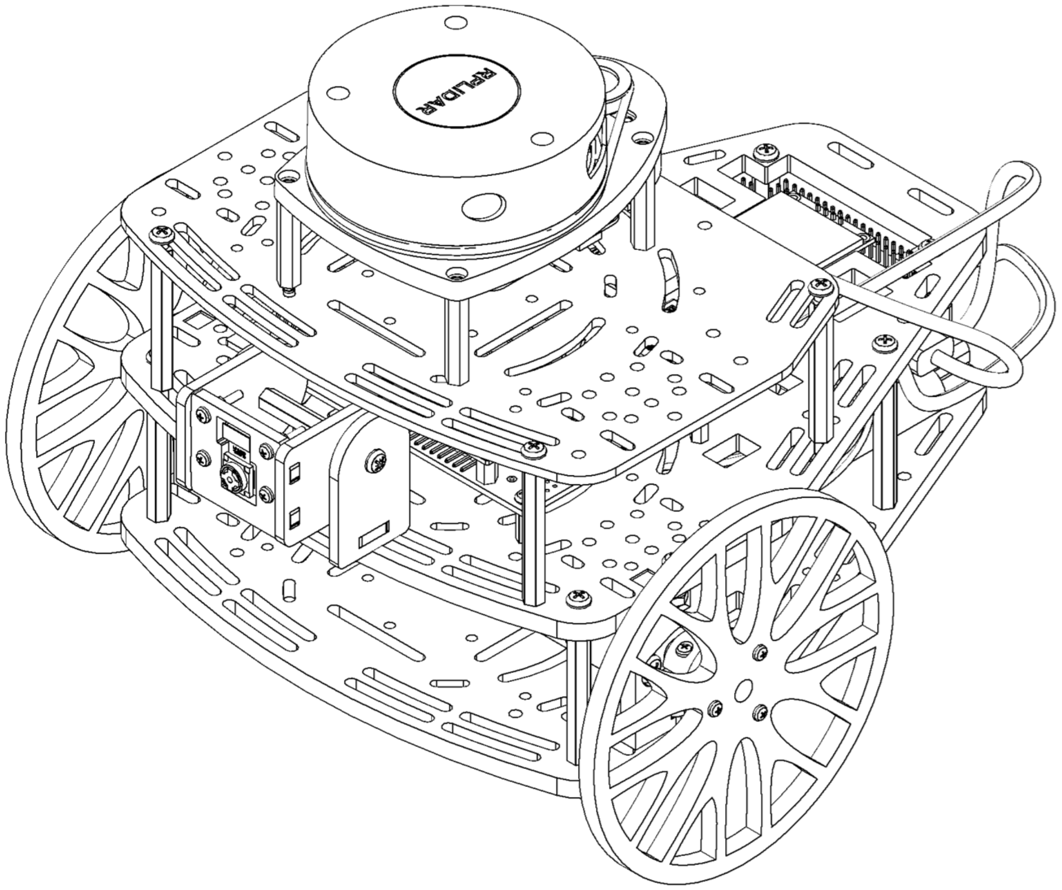
- Development of a kinematic simulator for the Puzzlebot robotic platform using the kinematic model of a nonholonomic robot.

Part 2:

- Develop a dead reckoning localisation node for the Puzzlebot. The results of the simulation must be visualised in RVIZ.
- The visualisation must be a 3D robot on RVIZ.

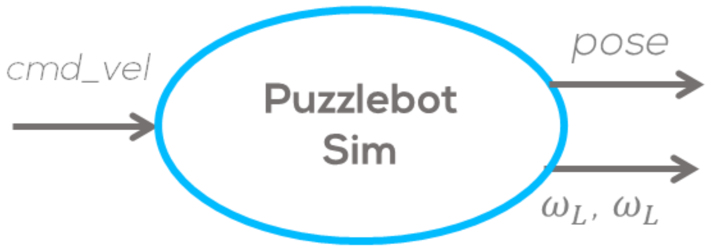
Part 3:

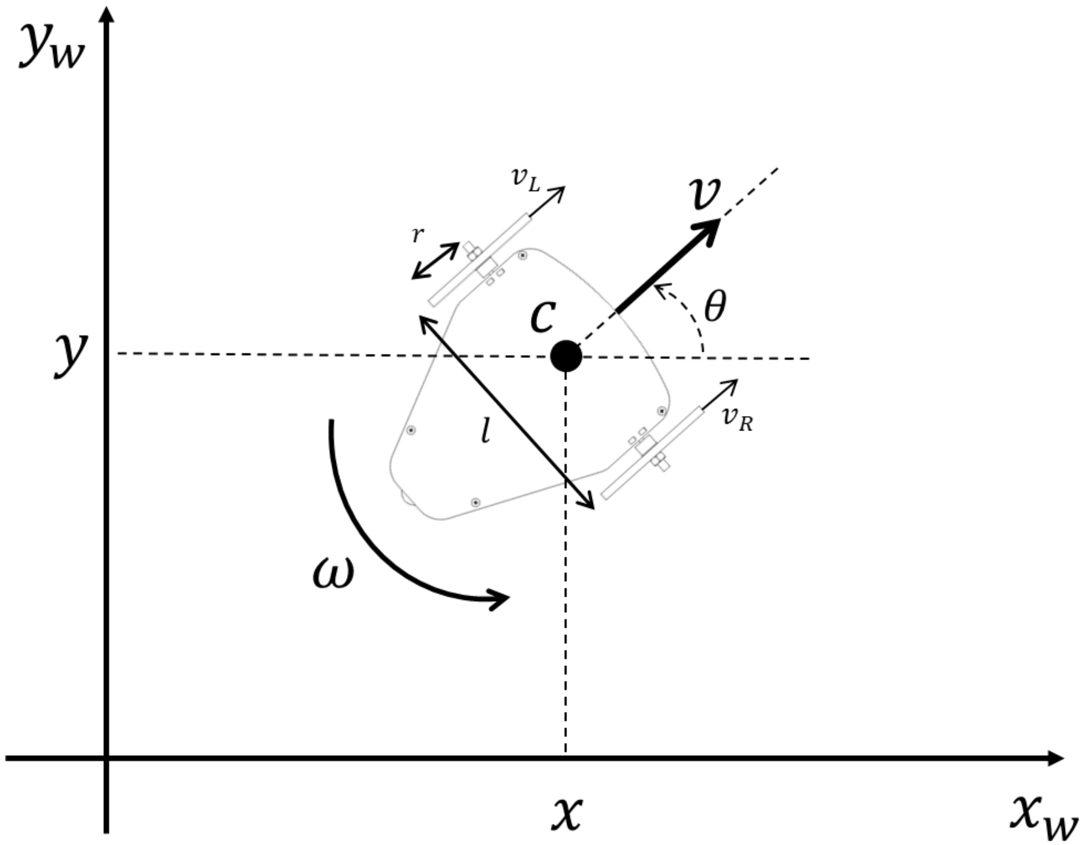
- Develop a node for controlling the position of the Puzzlebot (point stabilisation).



Mini challenge 2: Part 1

- This part of the activity consists of creating a node that simulates a dynamical system.
- Simulate a nonholonomic robot (e.g., Puzzlebot) using ROS.



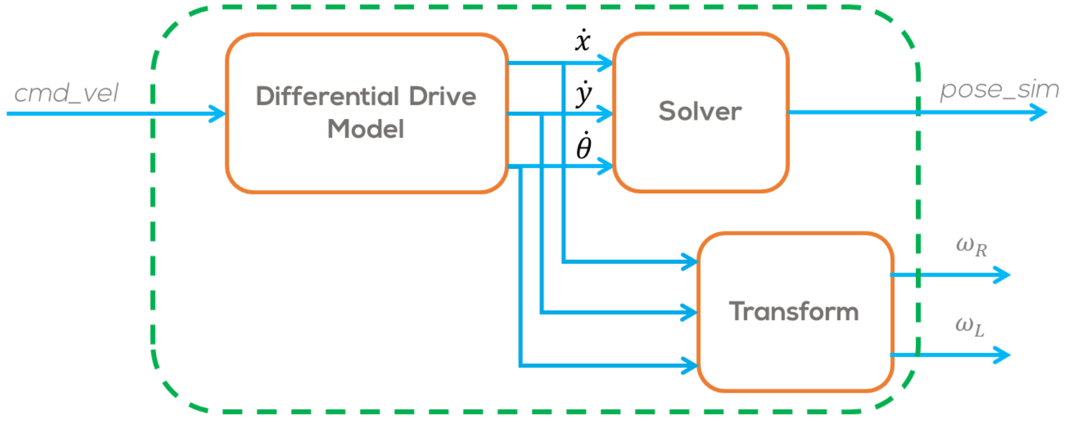


Mini challenge 2: Part 1

- The robot kinematical model is given by:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases}$$

- The name of the package for the simulated node must be *"puzzlebot_sim"*.
- For the robot's pose, use the "PoseStamped" message.
  - The pose topic must be named *"pose_sim"*
- For the input to the robot use *"Twist"* message
  - The topic for commanding the robot must be named *"cmd_vel"*



/puzzlebot_kinematic_model
(Node)

Mini challenge 2: Part 1

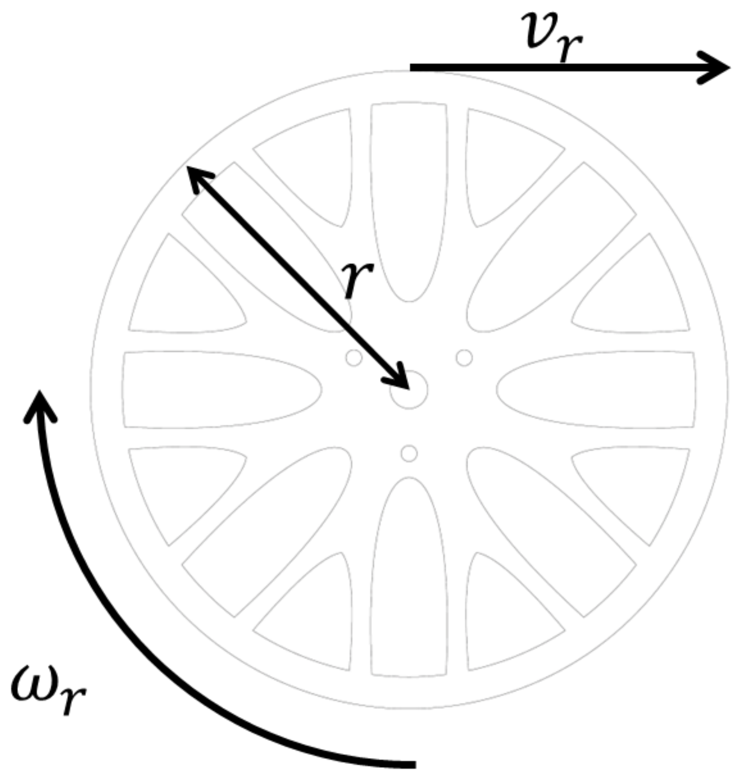
- The wheel's speed must also be published using a *"Float 32" std_msg.*
- The topics for each wheel must be *"wr"* and *"wl",* for the left and right wheels respectively.

$$Remember:$$

$$v = \frac{v_R + v_L}{2} = r\frac{\omega_R + \omega_L}{2}$$

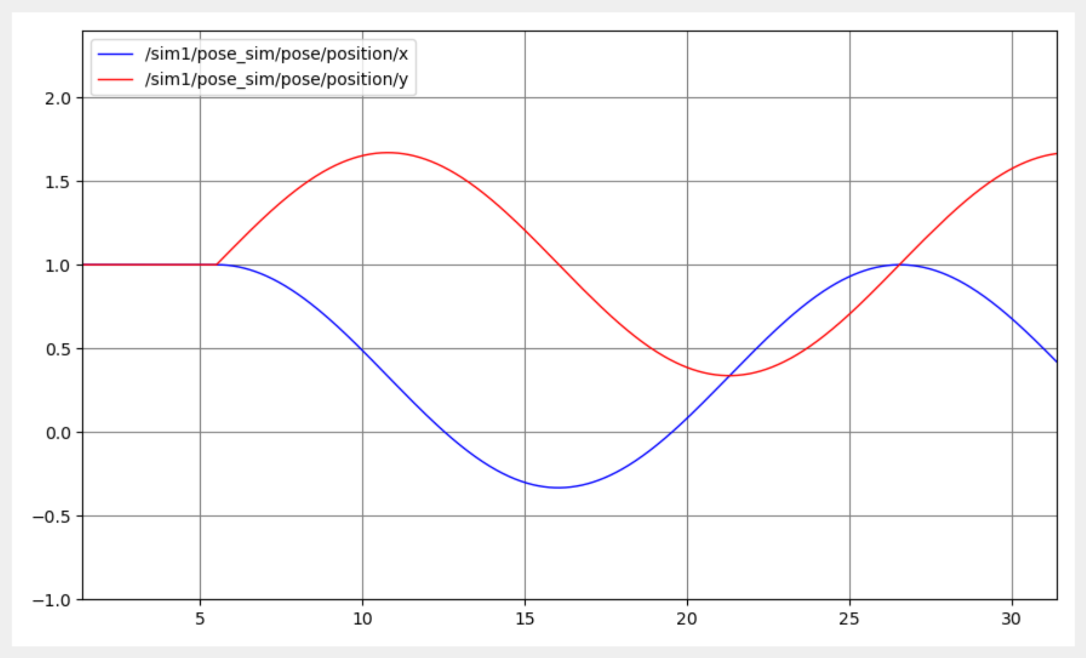
$$\omega = \frac{v_R - v_L}{l} = r\frac{\omega_R - \omega_L}{l}$$

- Puzzlebot parameters:
  - **Radius of the wheel: 5 cm**
  - **Wheelbase: 19 cm**



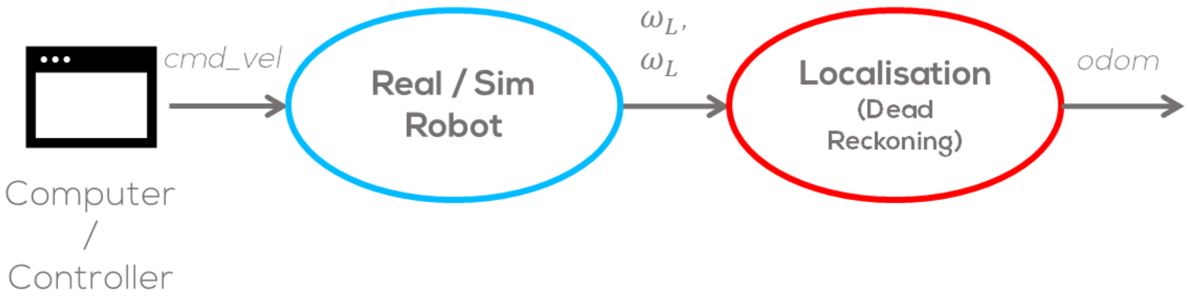
Mini challenge 2: Part 1

- Test your simulation by inputing some commands into the "cmd_vel" topic.
  - Test the linear speed and position
  - Test angular speed and angle.
- Use the "rqt_plot" to verify your results.
- Plot the position and wheel speeds.
- Verify if the results are correct.
- You can use the "teleop_twist_keyboard" node to test your results



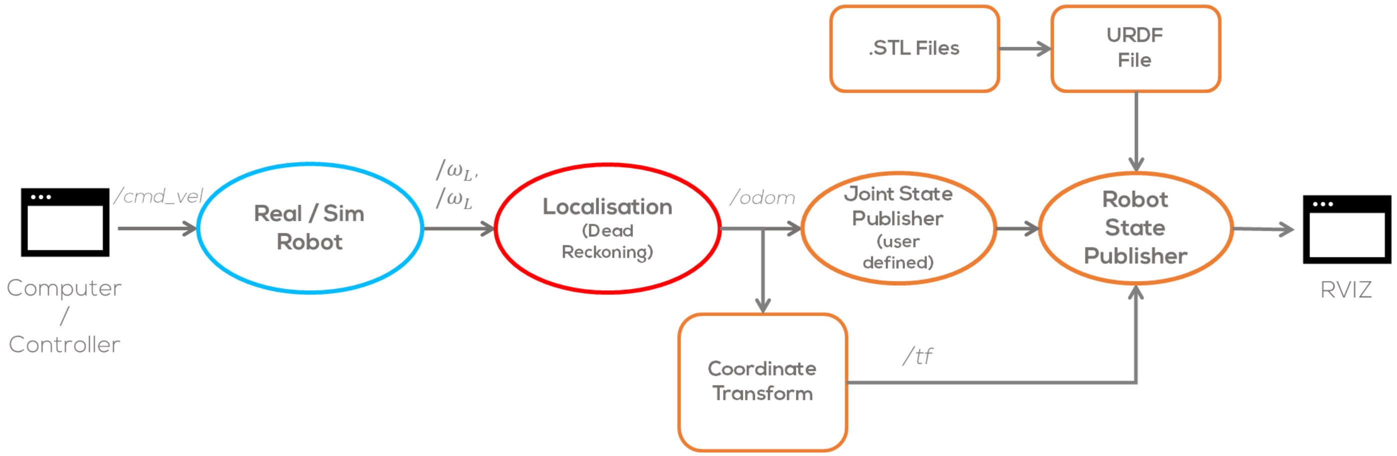
Mini challenge 2: Part 2

- Make a simple node called *"localisation"* that subscribes to the *"wr"* and *"wl"* topics of the simulated robot.
- Use this information to create an Odometry Message and publish the message in an *odom* topic.
- **For this activity, it is not required to fill in the covariance part of the odometry message.**
- Verify that the results are correct using the "*rqt_plot*" or the *"rqt_multiplot"*



Mini challenge 2: Part 2

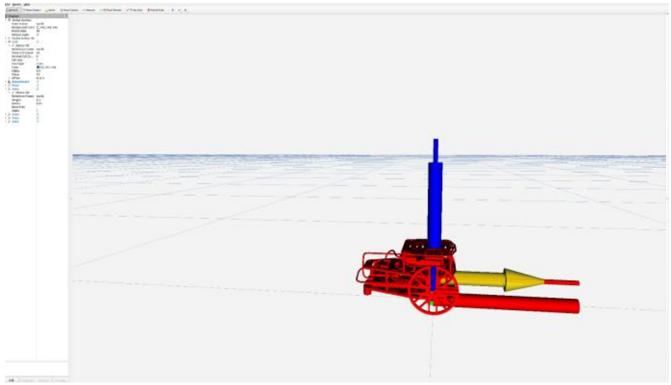
- Modify the custom joint publisher for the Puzzlebot of the previous challenge to read the odometry message and publish the joint information, to the simulated robot.
- Establish the inertial frame to be called *"odom" (the frame can be set up in the launch file)*
- Make a transform between the *"odom"* frame and the *"base_link*" or *"base_footprint"* frame. The transformation can be set up in the Joint State publisher node or a separate node.
- Use the teleoperation node or a command to test your simulation.

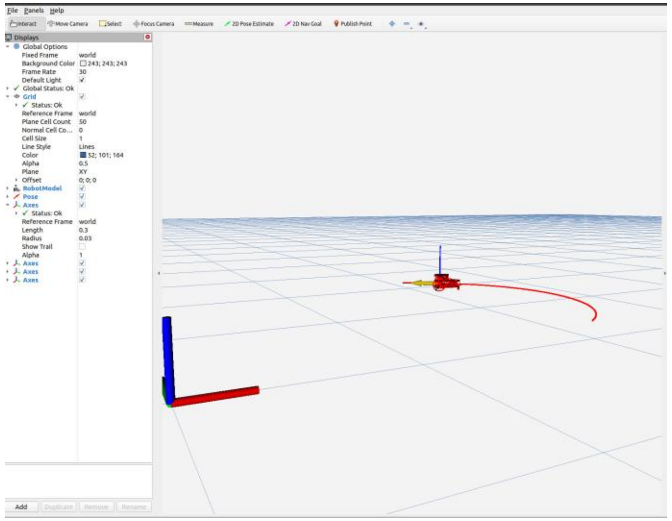


Mini challenge 2: Part 2

- The student is allowed to use *"tf"* coordinate transforms or *"URDF "* files for the simulation, or a combination of both.
- The student must define the coordinate frames and transformations to be used.
- The students must define the required launch files for this activity.
- The simulation must be tested under different conditions, i.e., different speeds.
- The students must define a correct sampling time for the simulation .
- The students must solve the differential equations using numerical methods.
- The usage of any library is strictly **forbidden**.

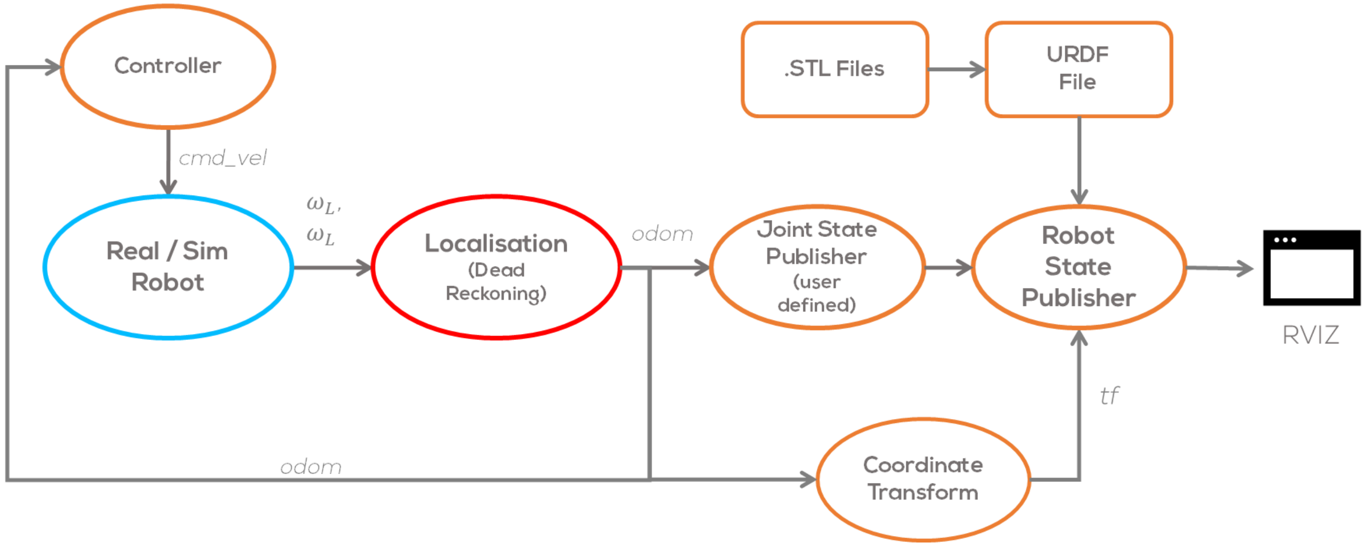
Expected results:





Mini challenge 2: Part 3

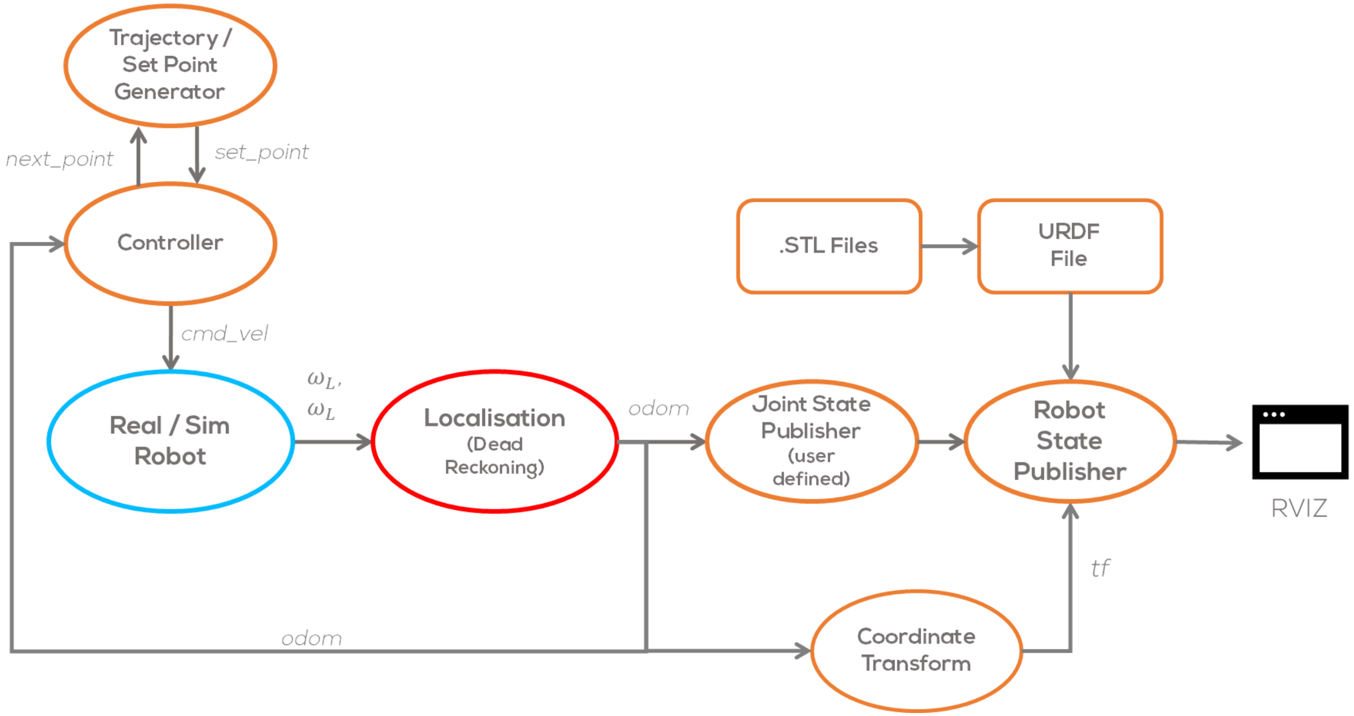
- Make a simple node called *"control"* that subscribes to the *"odom"* topic of the simulated robot.
- Set Points can be stablished as parameters.
- Use this information to send commands to the robot.
- Verify that the results are correct using *"rviz"*
- The user is encouraged to set a trajectory (square, pentagon, etc.) to test the controller algorithm.



Mini challenge 2: Extra

Set Point Generator:

- Make a simple node to pass the different set points to the controller.
- Set a flag to be published from the controller node once the robot has reached the goal point.
- In this node the user can define different trajectories (square, pentagon, etc.) to test the controller algorithm.



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.