Task 1.a :
Nodes and topic list when launching uuv_gazebo/rexrov_default.launch are: -

No.	Node name	Description
1	acceleration_control	This node takes robot acceleration and forces to convert it to acceleration data
2	ground_truth_to_tf_rexrov	Generate tf from robot_pose
3	joy_uuv_velocity_teleop	Take joystick command and convert it to velocity data
4	Joystick	Read joystick key and convert it to velocity data
5	robot_state_publisher	Publish the state of a robot to tf
6	thruster_allocator	Allows a client node to command the thrusters.
7	urdf_spawner	Run a python script to send a service that's called gazebo_ros to spawn robot URDF.
8	velocity_control	Used velocity and odom data to generate accelaration data of robot.
9	rviz	Visualize what the robot is seeing and doing

No	Topic Name	Description
1	/rexrov/thrusters/input	Type: uuv_gazebo_ros_plugins_msgs/FloatStamped Publishers: /rexrov/thruster_allocator Subscribers: /gazebo
2	/rexrov/thruster_manager/input_stamped	Type: geometry_msgs/WrenchStamped Publishers: None Subscribers: /rexrov/thruster_allocator
3	/rexrov/thruster_manager/input	Type: geometry_msgs/Wrench Publishers: /rexrov/acceleration_control Subscribers: /rexrov/thruster_allocator
4	/rexrov/teleop_twist_keyboard/cmd_vel	Type: geometry_msgs/Twist Publishers: /teleop Subscribers: /rexrov/velocity_control
5	/rexrov/rexrov/camera/camera_image	Type: sensor_msgs/Image Publishers: /gazebo Subscribers: /rviz
6	/rexrov/joy	Type: sensor_msgs/Joy Publishers: /rexrov/joystick Subscribers: /rexrov/joy_uuv_velocity_teleop
7	/rexrov/joy/set_feedback	Type: sensor_msgs/JoyFeedbackArray Publishers: None Subscribers: /rexrov/joystick
8	/tf_static	Type: tf2_msgs/TFMessage Publishers: /rexrov/robot_state_publisher /world_ned_frame_publisher Subscribers: /rexrov/thruster_allocator /rviz
9	/rexrov/cmd_accel	Type: geometry_msgs/Accel Publishers: /myrov/velocity_control

		Subscribers: /myrov/acceleration_control
10	/rexrov/cmd_force	Type: geometry_msgs/Accel Publishers: None Subscribers: /myrov/acceleration_control
11	/rexrov/current_velocity_marker	Type: visualization_msgs/Marker Publishers: /gazebo Subscribers: /rviz
12	/rexrov/ground_truth_to_tf_rexrov/euler	Type: geometry_msgs/Vector3Stamped Publishers: /rexrov/ground_truth_to_tf_rexrov Subscribers: None
13	/rexrov/ground_truth_to_tf_rexrov/pose	Type: geometry_msgs/PoseStamped Publishers: /rexrov/ground_truth_to_tf_rexrov Subscribers: None
14	/rexrov/joint_states	Type: sensor_msgs/JointState Publishers: /gazebo Subscribers: /myrov/robot_state_publisher
15	/rexrov/velocity_control/parameter_descriptions	Type: dynamic_reconfigure/ConfigDescription Publishers: /rexrov/velocity_control Subscribers: None
16	/rexrov/velocity_control/parameter_updates	Type: dynamic_reconfigure/Config Publishers: /rexrov/velocity_control Subscribers: None
17	/tf	Type: tf2_msgs/TFMessage Publishers: /rexrov/ground_truth_to_tf_rexrov /rexrov/robot_state_publisher /gazebo Subscribers: /rexrov/thruster_allocator /rviz /gazebo

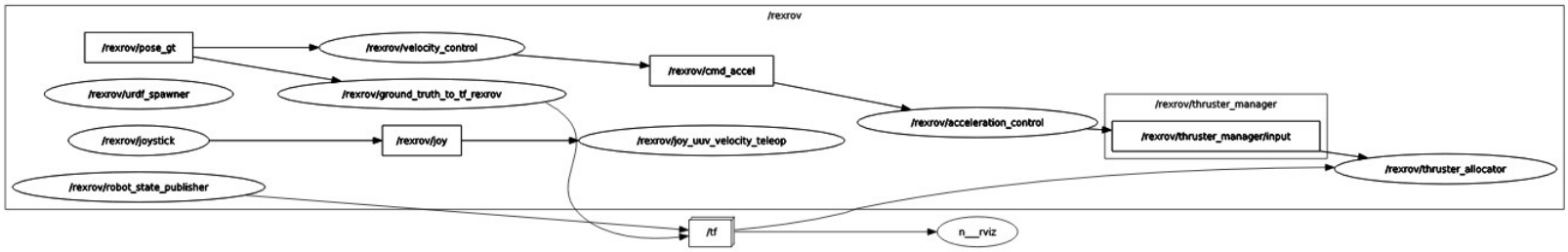
Messages

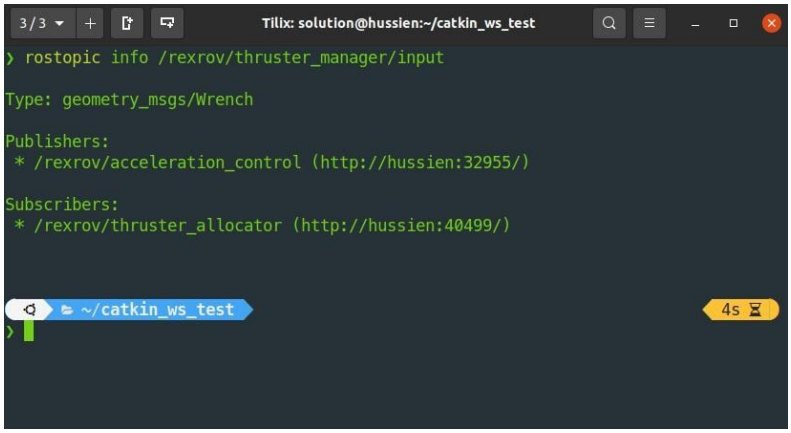
No.	Message type	Description
1	geometry_msgs/Twist	Velocity message in free space broken into its linear and angular parts
2	geometry_msgs/Wrench	Force and torque message
3	sensor_msgs/FluidPressure	This message is appropriate for measuring the pressure inside of a fluid (air, water, etc).
4	nav_msgs/Odometry	This represents an estimate of a position and velocity in free space.
5	sensor_msgs/MagneticField	Measurement of the Magnetic Field vector at a specific location.
6	sensor_msgs/Joy	Reports the state of a joysticks axes and buttons.
7	sensor_msgs/Imu	This is a message to hold data from an IMU (Inertial Measurement Unit)
8	sensor_msgs/NavSatFix	Navigation Satellite fix for any Global Navigation Satellite System
9	sensor_msgs/Range	Single range reading from an active ranger
10	uuv_sensor_ros_plugins_msgs/DVL	This is a message to hold data from a DVL sensor (Doppler Velocity Log).
11	geometry_msgs/Accel	This expresses acceleration in free space broken into its linear and angular parts.

Services

No.	Service name	Description
1	GetThrusterCurve	Get thruster index, for the case the vehicle has different models.
2	ThrusterManagerInfo	Return some information about thruster manager.
3	SetThrusterManagerConfig	Set the configuration for thruster manager.
4	GetThrusterManagerConfig	Take then configuration of thruster manager.
5	SetThrusterState	Set the state of thrusters for vehicle.
6	SetThrusterEfficiency	Set the thruster output efficiency for vehicle.
7	GetThrusterState	Know the state of thrusters in vehicle.

Rqt output





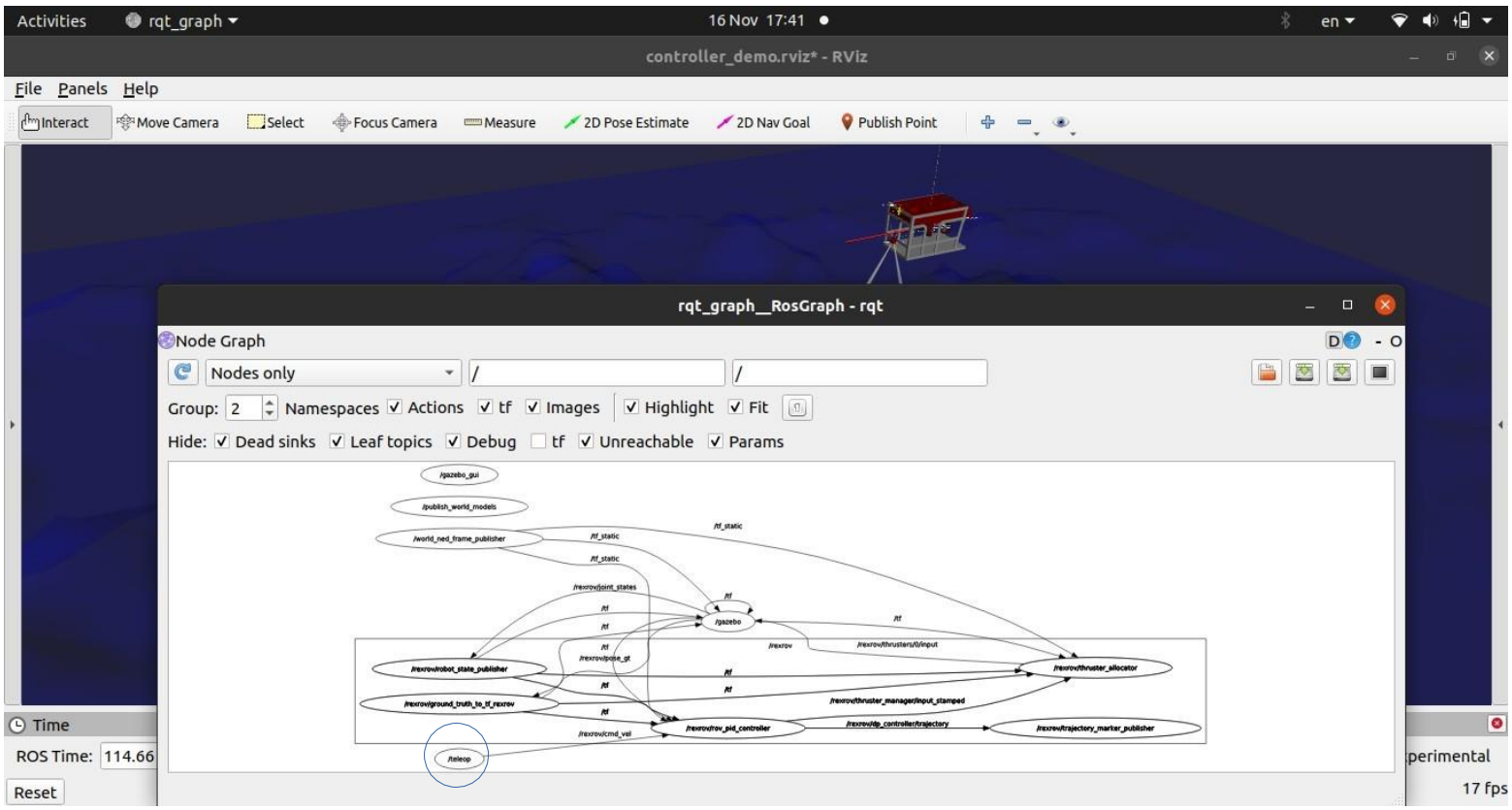
Task 1.b :

No.	Commands name	Function
1	rosnode list	list active nodes
2	rosnode info	print information about node
3	rostopic info	print information about active topics
4	rostopic list	list active topic
5	rosrun rqt_gui rqt_gui	provides the main to start an instance of the ROS integrated graphical user interface provided by qt_gui
6	rqt_graph	rqt_graph provides a GUI for visualizing the ROS computation graph
7	rosservice list	list active services
8	rossrv info	print information about service in ros
9	rqt_plot	provides a GUI plugin visualizing numeric values in a 2D plot

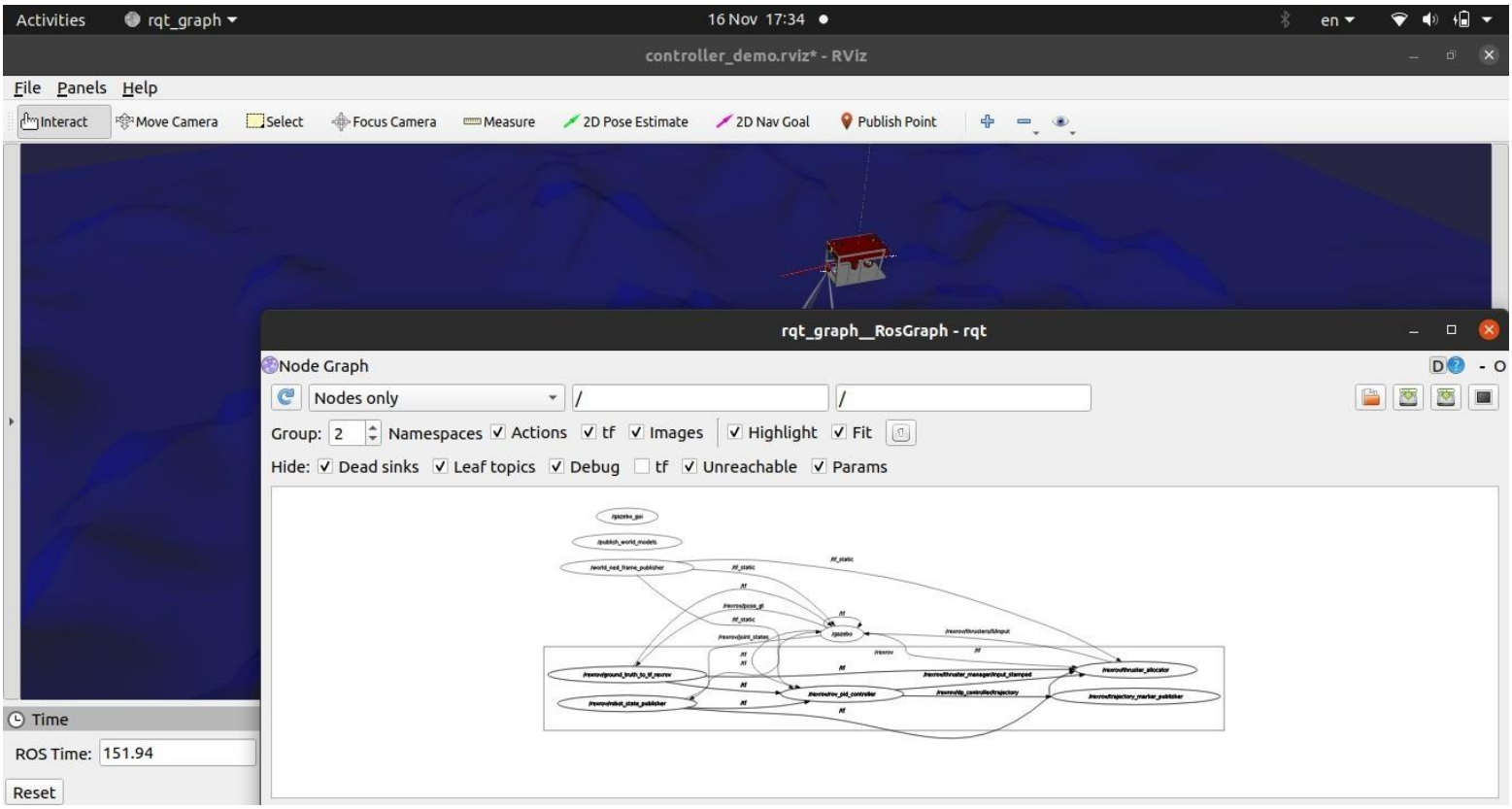
Task 2.a:
The command that's launching the file is: - *roslaunch uuv_gazebo rexrov_default_teloep.launch*

```
<launch>
   <!-- this will launch Gazebo with a rexrov model with its thruster_manager and control  -->
   <include file="$(find uuv_gazebo)/launch/rexrov_demos/rexrov_default.launch"></include>

   <!-- this will launch Gazebo with a empty underwater world model -->
   <include file="$(find uuv_gazebo_worlds)/launch/empty_underwater_world.launch"></include>

   <!-- This will launch teleop_twist_keyboard node that's used to control robot using keyboard-->
   <node pkg="teleop_twist_keyboard" type="teleop_twist_keyboard.py" name="teleop">
      <remap from="cmd_vel" to="/rexrov/teleop_twist_keyboard/cmd_vel"/>
   </node>
</launch>
```

Task 2.b:
Trajectory trace using teleop node



Trajectory using rosbag records



ros commands used:

$ rosbag record teleop_twist_keyboard/cmd_vel
$ rosbag info file_name
$ rosbag play file_name
$ rqt_garph

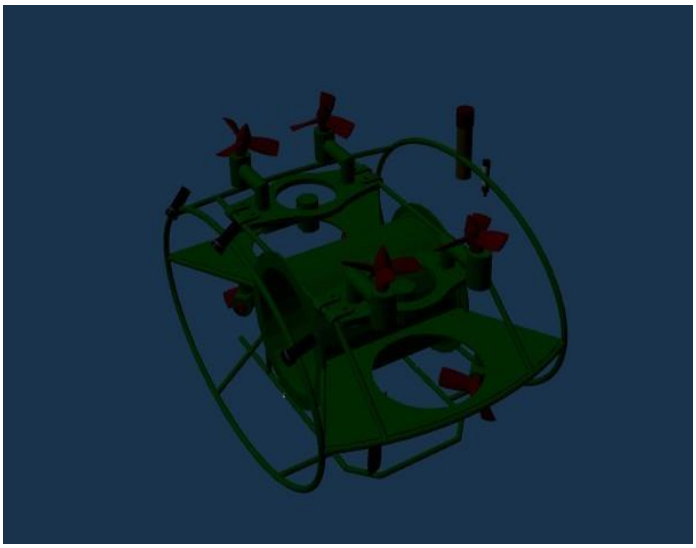
Task 3.a:

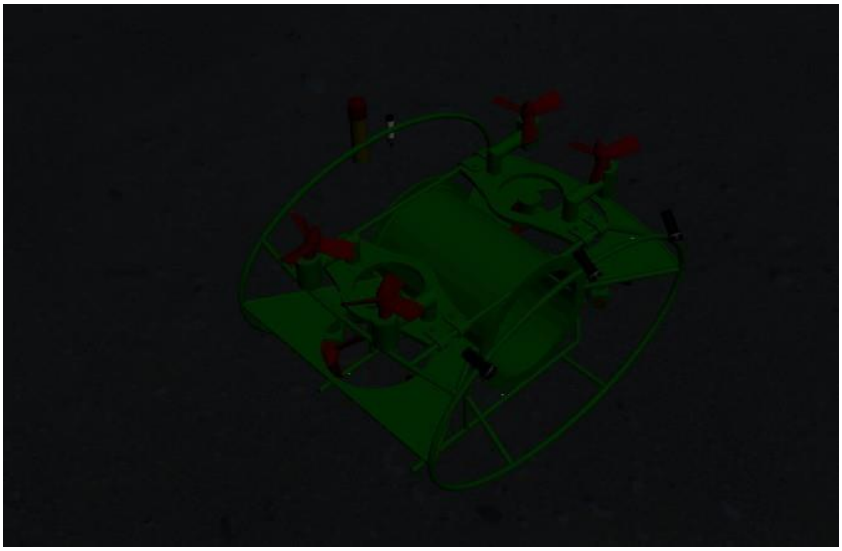
uuv_control_cascaded_pid/teleop_wernch_keyboard.py is a node that publishes forces and torques to topic /rexrov/thruster_manager/input

Task 3.b:
The command that's launching the file is: *roslaunch uuv_gazebo rexrov_wrench_control.launch*

```
<launch>
  <arg name="namespace" default="rexrov"/>
  <arg name="x" default="0"/>
  <arg name="y" default="0"/>
  <arg name="z" default="-70"/>
  <arg name="yaw" default="0.0"/>
  <arg name="joy_id" default="0"/>
  <arg name="axis_yaw" default="0"/>
  <arg name="axis_x" default="4"/>
  <arg name="axis_y" default="3"/>
  <arg name="axis_z" default="1"/>
  <arg name="launch_rviz" default="1"/>
  <include file="$(find uuv_descriptions)/launch/upload_rexrov.launch">
    <arg name="mode" value="default"/>
    <arg name="namespace" value="$(arg namespace)"/>
    <arg name="x" value="$(arg x)"/>
    <arg name="y" value="$(arg y)"/>
    <arg name="z" value="$(arg z)"/>
    <arg name="yaw" value="$(arg yaw)"/>
  </include>
  <!-- this will launch Gazebo with a empty underwater world model -->
  <include file="$(find uuv_gazebo_worlds)/launch/empty_underwater_world.launch">
  </include>
  <rosparam file="$(find uuv_control_cascaded_pid)/config/rexrov/inertial.yaml" command="load"/>
  <rosparam file="$(find uuv_control_cascaded_pid)/config/rexrov/vel_pid_control.yaml" command="load"/>
  <include file="$(find uuv_thruster_manager)/launch/thruster_manager.launch">
    <arg name="uuv_name" value="$(arg namespace)" />
    <arg name="model_name" value="rexrov" />
  </include>
  <!-- This will launch teleop_wrench_keyboard node that's used to control robot using keyboard by force and torques --
>
  <node pkg="uuv_control_cascaded_pid" type="teleop_wernch_keyboard.py" name="teleop_wernch_keyboard"
output="screen">
    <remap from="thruster_manager/input" to="/rexrov/thruster_manager/input"/>
  </node>
</launch>
```

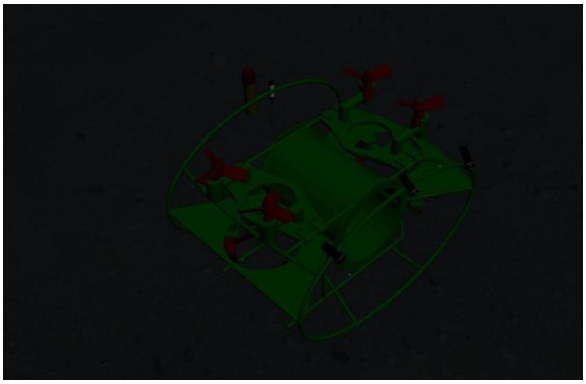
Task 4.a:
My Own ROV is called **myrov**





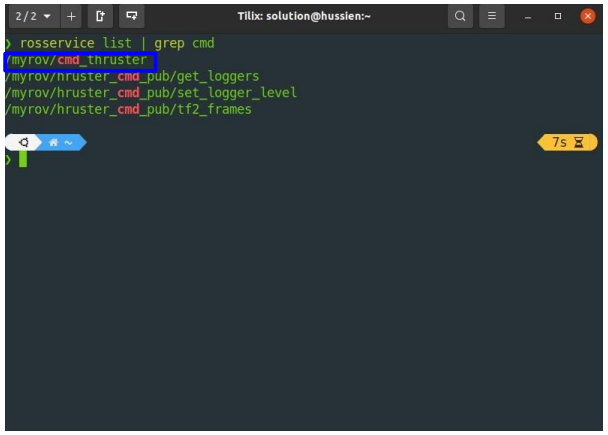
Task 4.b:

Myrov robot was equipped with 8 thrusters as shown in Figure below. Four horizontal thrusters T4, T5, T6, and T7, were responsible for the motions along the horizontal plane. Meanwhile, four vertical thrusters T0, T1, T2 and T3 were responsible for the motions along the vertical plane.



Task 4.c:
myrov_control/thruster_cmd_pub.py is server node that publishes thrust commands by converting force and torque data and the conversion is done by **CmdThruster** service.

myrov_control/wernch_thruster_conv.py is a node that is subscribed to thruster_manager/input topic and call **CmdThruster** service as shown below to convert data to Thrust commands.



Task 4.d:
The command that's launching the file is: *roslaunch uuv_gazebo myrov_wernch_conv_srv.launch*

```
<launch>
  <arg name="model_name" default="myrov" />
  <arg name="uuv_name" default="$(arg model_name)"/>
  <arg name="base_link" default="base_link" />
  <arg name="timeout" default="-1" />
  <arg name="reset_tam" default="false"/>
  <arg name="output_dir" default="$(find uuv_thruster_manager)/config/$(arg model_name)"/>
  <arg name="config_file" default="$(find uuv_thruster_manager)/config/$(arg model_name)/thruster_manager.yaml"/>
  <arg name="tam_file" default="$(find uuv_thruster_manager)/config/$(arg model_name)/TAM.yaml"/>
  <arg name="namespace" default="myrov"/>
  <arg name="x" default="0"/>
  <arg name="y" default="0"/>
  <arg name="z" default="-70"/>
  <arg name="yaw" default="0.0"/>
  <arg name="joy_id" default="0"/>
  <arg name="axis_yaw" default="0"/>
  <arg name="axis_x" default="4"/>
  <arg name="axis_y" default="3"/>
  <arg name="axis_z" default="1"/>
  <arg name="launch_rviz" default="1"/>
  <rosparam file="$(find uuv_control_cascaded_pid)/config/$(arg model_name)/inertial.yaml" command="load"/>
  <rosparam file="$(find uuv_control_cascaded_pid)/config/$(arg model_name)/vel_pid_control.yaml"
command="load"/>

  <include file="$(find myrov_description)/launch/upload_myrov.launch">
    <arg name="mode" value="default"/>
    <arg name="namespace" value="$(arg namespace)"/>
    <arg name="x" value="$(arg x)"/>
    <arg name="y" value="$(arg y)"/>
    <arg name="z" value="$(arg z)"/>
    <arg name="yaw" value="$(arg yaw)"/>
  </include>

   <!-- this will launch Gazebo with a empty underwater world model -->
    <include file="$(find uuv_gazebo_worlds)/launch/empty_underwater_world.launch">
    </include>
    <!-- this will launch thruster wrnech publuiser nodes -->
    <include file="$(find myrov_control)/launch/start_thruster_service.launch">
    </include>

<!-- This will launch teleop_wrench_keyboard node that's used to control robot using keyboard by force and torques -->
    <node pkg="uuv_control_cascaded_pid" type="teleop_wernch_keyboard.py" name="teleop_wernch_keyboard"
output="screen">
    </node>
</launch>
```