Simulated Controls System Node

Your group has been tasked with the implementation of a ROS Node using the uuv simulator and rexrov2. The node has to subscribe to three topics that are publishing unfiltered data for the controls system division. The group must filter this said data with the goal of only delivering what the division needs. An architecture of the project is provided below:

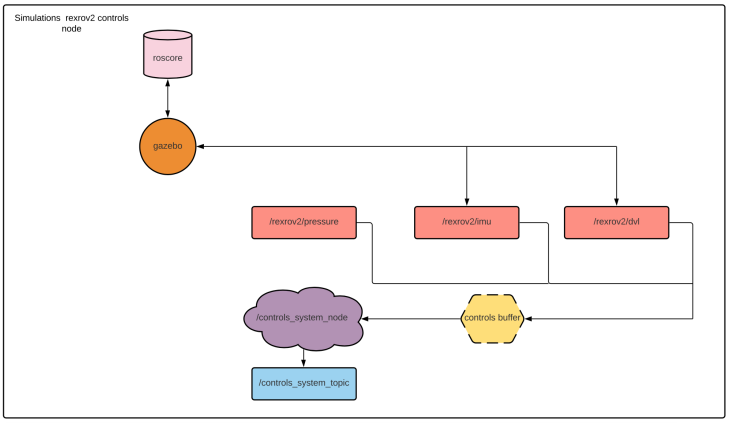

Figure 1: Architecture of The Simulated Control System Node

UUV Simulator & Rexrov2

As mentioned above, the uuv that will be used is the rexrov2. This device is part of the uuv simulator repository (which is also needed). With this, the team will have the tools to spawn a functional uuv within a gazebo environment.

To clone the uuv simulator in your catkin workspace:

```
git clone https://github.com/uuvsimulator/uuv_simulator
```

To clone the rexrov2 in your catkin workspace:

```
git clone https://github.com/uuvsimulator/rexrov2
```

To launch a underwater world with rexrov2:

```
roslaunch uuv_gazebo_worlds ocean_wave.launch
roslaunch rexrov2_description upload_rexrov2.launch
```

ROS Topics & Target Data

A ROS Topic is used to represent the different components a device can have. They are the ones that publish the unfiltered message structures. These are the following topics that the controls system division needs:

- IMU → `rexrov2/imu`
  - Target message → `angular_velocity`
  - Data type of message → `sensor_msgs/Imu`

```
Ros IMU info: Target: angular_velocity
std_msgs/Header header
uint32 seq
time stamp
string frame_id
geometry_msgs/Quaternion orientation
float64 x
float64 y
float64 z
float64 w
float64[9] orientation_covariance
geometry_msgs/Vector3 angular_velocity
float64 x
float64 y
float64 z
float64[9] angular_velocity_covariance
geometry_msgs/Vector3 linear_acceleration
float64 x
float64 y
float64 z
float64[9] linear_acceleration_covariance
```

Figure 2: Structure of IMU Topic

- DVL → `rexrov2/dvl`
  - Target message: `velocity`
  - Data type of message → `uuv_sensor_ros_plugins_msgs/DVL`

```
33  Ros DVL info: Target: velocity
 1  std_msgs/Header header
 2  uint32 seq
 3  time stamp
 4  string frame_id
 5  geometry_msgs/Vector3 velocity
 6  float64 x
 7  float64 y
 8  float64 z
 9  float64[9] velocity_covariance
10  float64 altitude
11  uuv_sensor_ros_plugins_msgs/DVLBeam[] beams
12  float64 range
13  float64 range_covariance
14  float64 velocity
15  float64 velocity_covariance
16  geometry_msgs/PoseStamped pose
17  std_msgs/Header header
18  uint32 seq
19  time stamp
20  string frame_id
21  geometry_msgs/Pose pose
22  geometry_msgs/Point position
23  float64 x
24  float64 y
25  float64 z
26  geometry_msgs/Quaternion orientation
27  float64 x
28  float64 y
29  float64 z
30  float64 w
```

Figure 3: Structure of DVL Topic

- Pressure Sensor → `rexrov2/dvl`
  - Target message → `fluid_pressure`
  - Data type → `sensor_msgs/FluidPressure`

```
65  Ros Pressure info: Target: fluid_pressure
 1  std_msgs/Header header
 2  uint32 seq
 3  time stamp
 4  string frame_id
 5  float64 fluid_pressure
 6  float64 variance
 7
```

Figure 4: Structure of Pressure Sensor Topic

Custom Message & Control Buffer

A custom message must be created to structure all the incoming target messages with the goal of creating a buffer and publishing into a new topic. The buffer facilitates the control division job as it puts all the data in the same place.

The message must have the following data types:

- `geometry_msgs/Vector3` (for imu - angular velocity)
- `geometry_msgs/Vector3` (for dvl - velocity)
- `float64` (for pressure sensor)

Filtered Topic

The team must create a topic that is publishing the created custom Controls System Message. This will give easy access to the values that the other division needs.

Code Structure & Documentation

The team should be able to apply OOP concepts that encapsulate all the publisher and subscribers strategies inside of a python class. With this said, the ROSPY API is needed to complete this task. PEP 8 guidelines should also be followed to document the final script that is developed. Finally, the team must show their progress via reports.