Assignment 1
Hello ROS
Due Date: Sept 18

In this assignment, you will write a ROS package that allows you to drive Husky with your keyboard. You will need to find the desired twists for the robot based on the keyboard inputs.

1 Preparation

Do not start this assignment before you try and understand the tutorials.

Download the Starting Packages

- Download husky packagese
  `git clone git@gitlab.cs.mcgill.ca:applied-robotics/robots/husky.git`
- Download Keyboard reader
  `git clone git@gitlab.cs.mcgill.ca:applied-robotics/examples/keyboard_reader.git`

Setting up your gitlab account

Note: You only need to do this once for the first assignment.

1. Follow the instructions here to setup your repository. Make sure you grant the teaching staff the right to access your repository and fill out the Google form to inform us of your account name. Otherwise, we will not be able to mark your assignments. If we cannot mark your assignments, we will give you a 0.
2. Clone this new repository `robotic-coursework-f2023` to your machine. We will use the same repository throughout the semester.

Setting up a catkin workspace

1. Create a catkin workspace following the instructions in the tutorial.
2. Add `husky`, `keyboard_reader`, and `robotic-coursework-f2023` into your catkin workspace using symbolic links.
3. Build your packages by `catkin build [package_name]`. You should be able to launch `husky`. Do not continue unless this step is successful.

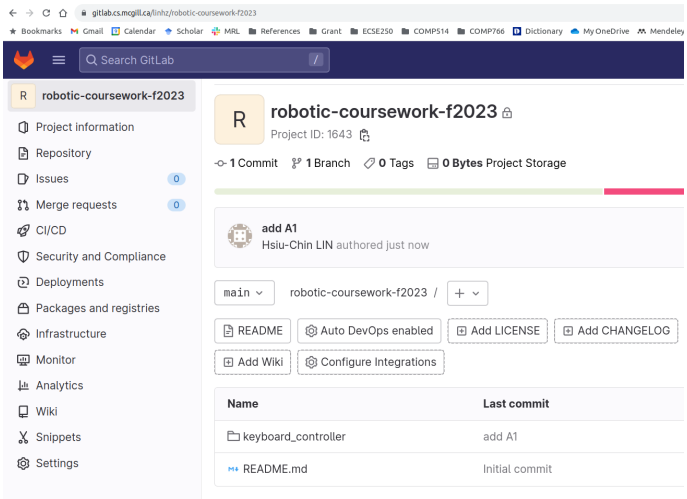
2 Specifications

Your main task for this assignment is to create a node that can read the input from the keyboard and publish a twist for the robot.

1. In `robotic-coursework-f2023`, create a new ROS package called it `keyboard_controller`. You can copy the tutorial examples on ROS topics and modify them accordingly.
2. The package `keyboard_reader` checks the keyboard inputs and publishes the value to the topic `teleop/cmd`. Your node should subscribe to this topic to retrieve the value.
3. Determine the twist based on the value `teleop/cmd`. We will only use three keys:

- `i`: move forward. Set the linear velocity on x-axis to 0.5
- `u`: turn left. Set the linear velocity on x-axis to 0.5 and angular velocity on z-axis to 0.5
- `o`: turn right. Set the linear velocity on x-axis to 0.5 and angular velocity on z-axis to –0.5
- Otherwise, the twist should be a zero vector

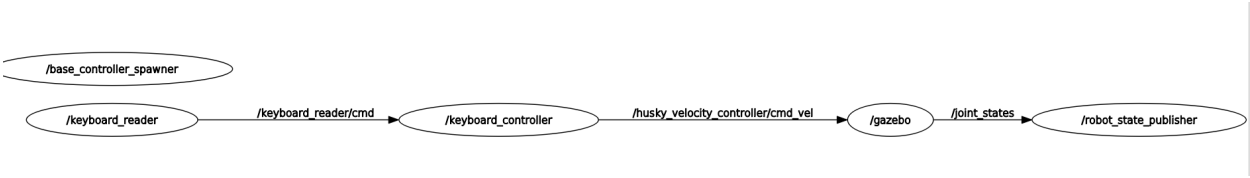
4. The Husky robot has a velocity control mode that allows you to input the twist. The robot is reading the velocity from `/husky_velocity_controller/cmd_vel`.
5. Write a launch file called `a1.launch`. This launch file should start the node you created in `keyboard_controller`
6. To run your program, you need to have 3 terminals, each of them running one of the following commands:
   ```
   > roslaunch husky_gazebo husky_empty_world.launch
   > rosrun keyboard_reader keyboard_reader
   > roslaunch keyboard_controller a1.launch
   ```
7. Submit your work by pushing your implementations to your gitlab repository. After you commit, please open your repository and double-check that you have submitted it correctly. If you open `https://gitlab.cs.mcgill.ca/[your CS ID]/robotic-coursework-f2023`, you should only see `keyboard_controller` and an optional README.md. Do not push `husky`, `keyboard_reader`, or your catkin workspace. You should open `keyboard_controller` to verify that your code is indeed there.



3 Useful ROS/Ubuntu Comments

- What topics does a node publish?
  `> rosnode info [node name]`
- What is the data type of a topic?
  `> rostopic type [topic name]`
- When ROS or Gazebo does not close properly. A quick way to close them all is to run
  `> rosnode kill -a ; rosnode cleaup ; killall -9 gzclient ; killall -9 gzserver`

- To check if the nodes communicate correctly, run `> rqt_graph` in a separate terminal. You should see the following figure. If the connections are broken, it is unlikely that your robot will move.



4 Evaluation

We will test your implementations by running
`> roslaunch keyboard_controller a1.launch`
We will not look into your repository to find out how you name your package and your launch file. It is your obligation to ensure the package and file names are correct

T1: [1 pt] for packages correctly set up and run
T2: [1 pt] for correct twists moving left
T3: [1 pt] for correct twists moving right
T4: [1 pt] for correct twists moving forward
T5: [1 pt] for correct twists without pressing any keys

Experimental Scoreboard

Historically speaking, students have a lot of issues with assignment submission. In most cases, the ROS package works on the student's side but does not work on the grader's side. Typical issues are (1) you name your packages differently from what we specified or (2) you hard-coded a folder/file name that only exists on your machine.

This year, we are trying to set up a scoreboard that can provide you with some feedback before the assignment deadlines. **The scoreboard is completely experimental, and we do not guarantee it will work well throughout the semester**. If you open the scoreboard, you can expect something similar to the following:

	T1	T2	T3
Of Course I Still Love You	-	-	-
A Shortfall of Gravitas	-	-	-
Just Read the Instructions	P	F	F

T1, T2, T3 are the test names, 'P' stands for 'Pass', 'F' stands for "Fail", and '-' means your packages are not available.