Project 4 (100 Points)

The goal of this project is for you to produce a node that keeps your Duckiebot in its lane. Note that this node already exists in the Duckietown code repo. Feel free to look at it for inspiration but make sure that the actual node you turn in is your own. You do NOT need to perform any of the state checking done in the actual node, you only need to accept the lane position message and publish the command message as described below.

1. Run the lane controller demo on your Duckiebot. Look at the inputs and outputs of the lane_controller_node.
    a. You should be somewhat familiar with the output but may not be familiar with the exact topic. Note that this time, it goes to the switch node, not straight to kinematics. This is because lane following waits for a signal from keyboard control before it starts. You will want to do the same.
    b. Pay attention to the input topic lane_pose. This contains the output from the sensing pipeline and tells you where you are in the lane. Pay special attention to the d and phi parameters in that message. Experiment a bit with your robot on a lane. The d parameter tells you where you are in the lane and the phi parameter tells you your orientation. Ideally, they should both be 0.

2. Think about what variables you need to control here. Does the speed affect the robot's position in the lane? How about the angular velocity?

3. Create a PID controller method in your node.

4. Create a lane controller that uses the PID controller (not all terms are necessarily needed) to keep your robot in the lane. Note that two PID controllers (one for each parameter described in 1b) added together tends to work best.

5. Tune your controller(s) until they work well on your Duckietown mats.

6. Some notes on writing your code:
    a. Use whichever method of syncing and revising code worked for you in Project 3
    b. Do NOT include your Duckiebot's name in the topics (or anywhere else) in your code. Instead, use a namespace in the launch file as required in Project 3. This allows your code to be run on any Duckiebot.
    c. You will need to include several other Duckietown nodes in your launch file. Take a look at how the Duckietown lane following demo launch file works or look at the example launch file at the end of these instructions.
    d. Use your knowledge from the class material to write a PID controller method. You can have that method as part of the node, but you are free to separate it into its own module.

e. This is a real robot with real hardware, so there will be errors in sensing. You may want to threshold (limit) the maximum/minimum errors given to your controller and/or the maximum control signal you will send. Be patient and work out what your robot is telling you versus what it should be doing.

7. To prove it is your code running, and not the lane following demo, please do the following:
    a. Have your code output a unique message to loginfo or similar (logwarn/logerr will show up in amber/red on the screen and may be better for visibility) every time your main callback is run. Something like "<your name> LANE FOLLOWING CODE" is preferred to show that it is your code.
    b. Record the screen on your computer (preferably using a screen capture program) while your code is running with the following programs open:
        i. A terminal showing that no lane controller nodes are running at the beginning (run "rosnode list" and show the output) and then showing roslaunch as it starts your controller.
        ii. rqt_image_view showing the Duckiebot's camera view
        iii. rqt_console (if your custom message is not shown in the terminal)
        iv. NOTE: to run both rqt_console and rqt_image_view at the same time, use & at the end of the command to run in the background, like:
            ```
            $ rqt_image_view &
            $ rqt_console &
            ```
            You may need to press enter again to get back to the command prompt
    c. Also record a top-down video of your Duckiebot performing lane following for at least three laps of your mats. Make sure this recording overlaps with the recording in 6b. You do NOT need to sync the videos or any other serious video editing, just make sure it is from the same run.

Turn in to blackboard:

- The URL of your git repo and the git tag representing the lab
- A link to the videos you recorded
- A short description of what you choose to control (input and output signals) and why. What final PID gain values did you use?
- A description of any problems you had completing this project

Upload to dropbox a video of your best top-down video of your lane following robot.

Rubric:

- Algorithm: 30 points
- Implementation: 40 points
    - Reasonable tuning parameters: 10 points
    - ROS package setup: 5 points
    - ROS node implementation: 15 points
    - PID class integration: 10 points
- Accuracy (scaled to your peers): 10 points
- Demo video: 15 points (submitted to Dropbox)
- Answers to submission questions: 5 points

Launch file template:

```
<launch>

    <arg name="veh" default="$(env VEHICLE_NAME)"/>
    <arg name="ai_trafo_mode" default="cb" doc="'cb' for colo balance only;
'both' for color balance and linear trafo"/>
    <arg name="ai_interval" default="5" doc="interval with which the linear trafo
gets updated. color balance is performed every second."/>
    <arg name="verbose" default="false"/>

    <!-- start Duckietown nodes -->
    <arg name="demo_name" value="lane_following"/>
    <!-- start basic args -->
    <include file="$(find duckietown_demos)/launch/master.launch">
            <!-- Basic arguments -->
                <arg name="veh" value="$(arg veh)"/>
       <arg name="demo_name" value="$(arg demo_name)"/>
                <arg name="param_file_name" value="default" />
                <arg name="visualization" value="true" />

            <!-- Finite state machine -->
            <arg name="fsm" value="true"/>
            <arg name="/fsm/logic_gate" value="false"/>

                <!-- Camera and anti intagram -->
                <arg name="/camera/raw" value="false" />
                <arg name="anti_instagram" value="true" />

            <!-- Lane Following stack -->
            <arg name="lane_following" value="true"/>
            <arg name="/lane_following/line_detection" value="true"/>
            <arg name="line_detector_param_file_name" value="default" />
            <arg name="/lane_following/ground_projection" value="true"/>
            <arg name="/lane_following/lane_filter" value="true"/>
            <arg name="/lane_following/lane_controller" value="false"/>

    </include>
    <group ns="$(env VEHICLE_NAME)">

        YOUR NODE/PARAMS/ETC GO HERE

     </group>

</launch>
```