Assignment 2

Due: 11:59PM 09/15

Using Encoder Data

Calculate the Pulses per inch of travel given the following information, show your work:

1. A 40 tick single channel encoder attached to a 6 in wheel
2. A 500 tick single channel encoder attached to a motor with 5:1 gearbox and a 1.5 in wheel.
3. A 250 tick quadrature encoder on a 3 inch wheel.
4. A Quadrature encoder with a 1000 tick disk attached to a motor followed by a 32.2:1 Gearbox attached to a 2.5 in Diameter Shaft.
5. During a routine car repair, you replace the tires on your car. The original tires are 22 inches in diameter, but the store only has 25 inch tires; you purchase and install them anyways. While driving on the highway with your cruise control set to the speed limit you get pulled over by the police. The police officer writes you a ticket which you can trace back to the previous car repair. What is the ticket for, and why?

System Design

Pick one of the following robots to perform a brief analysis of the system design. Describe some problem(s) it is trying to solve, and how it attempts to address them. Describe a problem the robot does not address that you either think it should, or is well suited to solve. Include some idea of how and why it might solve it, and some ideas about how the robot could be tested to validate its new performance.

1. Atrias
2. Big Dog
3. Asimo
4. Roomba
5. ABB IRB 6640

First steps in Programming

For this part, you will be working on the demo code which was presented in class, and is available through the Scholar site. You will need to make several modifications and extensions to this code. Remember, in order to submit programming assignments, you must have a GitHub account, and send your username to jpz@vt.edu along with your name to get a repository set up for you. This repository is where code will be submitted - whatever version of code is listed on GitHub's servers by the submission deadline in the master branch will be graded. Please note you will need to make commits and push them to the repository - making local commits isn't sufficient.

1. One big issue that commonly occurs in programming deals with making sure that inputs are valid. For the two harmonic series functions, insert additional code which checks the input argument for several troublesome states:
    a. Protect against the input being 0 or negative. If this occurs, the function should return 0
    b. Protect against the input being a floating point value (e.g. not an integer.) If given a floating point value, safely convert the value to an integer and continue execution
2. The user input interface lacks a few important features, which you should add to the program.
    a. Make sure that the user chooses a valid function to use. If the user enters a number which isn't 1 or 2, print out an error message chastising the user.
    b. Make the user input sequence loop, so that it keeps asking the user to select a method and number of iterations repeatedly. Make sure to introduce some value which the user can input to end the loop and quit the program, and update the printed instructions to include this command.

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}.$$

3. You can approximate π through the use of the infinite summation below. Write a function which returns the estimated value of π after some number of iterations. Add this command to the user interface.
4. Modify the demo file to work as a Python module. This means that if python is executed in the correct location, someone could execute "import first_demo" and be able to access the functions defined inside, without executing the body of code below. This requires more searching online and reading than coding, but is useful for testing code and understanding how Python works.

Quadrature Decoding Task

For this part, you are going to write a Python module which can be given a sequence of updates to the two lines of a quadrature encoder, and estimates a position and velocity based on the results. The basic framework is available from Scholar. Implement the update function with the following assumptions:

1. a_state and b_state are boolean values indicating if that channel is high (true) or low (false). Note that these values may be repeated between calls - you might see update(true, true, 1.) and then update(true, true, 2.0).

2. time is some floating point value which represents when the update occurred. This value will represent some number of seconds since execution.

After each call to update(), the position and velocity values should contain the latest estimate of each quantity. It is fine to do a naive estimate of the velocity - that is to say, you can consider the velocity estimate to only cover the time from the previous call to update() to the current call.

In order to test your work, create a ROS node which subscribes to a topic containing the values given to update(), and publishes a message of type JointState with the position and velocity set. Subscribe to the incoming topic as “/encoder_in”, and publish on topic “/joint_out”.