PID Controller Breakout Session 2:

1. Using the PID\_v1.h library, create a PID controller scheme to keep the car driving straight (angular velocity in z-axis = 0) while driving forward. In this case, do not add your scaling factor to the raw\_motor\_control() parameters. Your raw\_motor\_control() function at the end of loop() should look something **LIKE** the following:
   1. raw\_motor\_control(base\_motor\_speed + output, base\_motor\_speed - output);
   2. Base\_motor\_speed = ~150
2. Write down your controllers hyper-parameters:
   1. Setpoint = ?
   2. Output limits = [?, ?]
   3. Sample time = ?
3. Choose a Kp gain that will cause the system to be UNSTABLE. Write down the gains
   1. Kp = ?
   2. Ki = 0
   3. Kd = 0
4. Decrease your Kp gain until you get a system that is RELATIVELY STABLE. In this case, watch your robot and make a judgement call on if it is driving straight. (Turning about 2-3 degrees/second is okay.) Write down the gains.
   1. Kp = ?
   2. Ki = 0
   3. Kd = 0
5. Like before, we have a semi-stable system with only a Kp gain. Let’s slowly adjust our Kd value until we have an optimal response. Optimal response means very little “shaking” and again a relatively straight movement path. Take a video of the robot driving straight after tuning your gains.
   1. Kp = ?
   2. Ki = 0
   3. Kd = ?
   4. Any “shaking”?
   5. Any turning bias?
      1. Estimate how many deg/sec your robot is turning (pay attention to sign)
6. Try playing around with the Ki value to see if you can make the response of the system to become MORE optimal. If it doesn’t seem to help, keep Ki = 0.
   1. Kp = ?
   2. Ki = ?
   3. Kd = ?
7. Now we are going to purposely add a bias to one of the motors to represent a “disturbance” in the system. Add a variable called motor\_bias and add it to one of the motors in the raw\_motor\_control() function (pick a value ~25). Run the robot on the floor and analyze if your controller is mitigating the “forced disturbance” applied.
   1. Write your analysis here:
   2. Pick a larger value for bias and analyze:
   3. Finally pick a very large value for bias (>=150). Does the controller mitigate this large disturbance still? If so make the value larger until the controller is incapable of mitigating the disturbance. Why do you think it is unable to mitigate the “forced disturbance”? Think of 2 ways we could adjust controller hyper-parameters, or change hardware to fix this problem.
   4. Remote the bias from raw\_motor\_control()
8. Change the setpoint for the controller to 6 various values, choose negative and positive values between |1| -> |360|. For each, use a stopwatch to estimate the angular velocity of the robot in the z-axis.
   1. Num1:
      1. Estimated ang\_vel:
   2. Num2:
      1. Estimated ang\_vel:
   3. Num3:
      1. Estimated ang\_vel:
   4. Num4:
      1. Estimated ang\_vel:
   5. Num5:
      1. Estimated ang\_vel:
   6. Num6:
      1. Estimated ang\_vel:
   7. At what point does the robot seem unable to achieve the setpoint?
      1. :
9. Create an instance of the “State” class and call it controller\_state. Initialize the linear and rotation state by using the setLinearState() and setRotationState() functions to (linear: 150, rotation: 0). Try incorporating the State.getLinearState() in your raw\_motor\_control() function to control the direction the robot will drive in. Flip the sign of the linear state and make sure the robot moves backwards.
10. Now, in your loop() function, update the setpoint up the drive\_straight PID controller to be the rotation state of the controller\_state State instance. Change this rotation\_state value to a couple positive and negative values and make sure the controller is still capable of controlling the z-axis angular velocity while also changing the “linear movement” of the robot with the edits from step 9.
11. Lastly, make a timing loop that will switch the linear direction the robot drives in every 2 seconds (set the rotation state to 45 deg/sec). Note that when the robot is driving backwards, to follow the same path it took driving forwards, we will need to flip the sign of the rotation state as well, this way the robot drives along the same path segment continuously. Take a video of the robot doing this. Note: the robot should be driving back and forth along a small arc of a circle.