# PID controller

## Overview:

In this project, simulator has a car on a track and it sends cross track error over a websocket and we need to calculate the steering value using the PID controller. The base project code is given by Udacity and we need to implement figuring out the optimal co-efficients (tau) either by manual tuning or using twiddle algorithm and calculate the steering angle so that the car drives within the track.

## Build steps:

In order to build and compile the code, from within the main repository directory:

- `mkdir build && cd build` to create and enter the build directory
- `cmake .. && make` to compile your project
- `./path_planning` to run the  code

Click on the "Simulator" button in the bottom of the Udacity workspace, which will open a new virtual desktop. You should see a "Simulator" icon on the virtual desktop. Double-click the "Simulator" icon in that desktop to start the simulator.

**Important:** You need to open a terminal before attempting to run the simulator and keep the GPU mode on.

## Rubric points:

### Compilation:

Code compiles correctly.

### Implementation:

### The PID procedure follows what was taught in the lessons.

PID controller is implemented in PID.cpp, where Init() function initializes the member co-efficients, proportional, differential and integral errors and also max and min errors. Function Update() calculates the proportional, differential and integral errors based on cte received. Total error is calculated in TotalError() function.

## Reflection:

### Describe the effect each of the P, I, D components had in your implementation.

The proportional portion of the controller tries to steer the car towards the center line (against the cross-track error). If used along, the car overshoots the central line very easily and go out of the road very quickly. An example video where this component is used alone is uploaded under path: [src/only-proportional.mp4]

The integral portion tries to eliminate a possible bias on the controlled system that could prevent the error to be eliminated. If used along, it makes the car to go in circles. In the case of the simulator, no bias is present. An example video where this component is used alone is uploaded under path: [src/only-integral.mp4]

The differential portion helps to counteract the proportional trend to overshoot the center line by smoothing the approach to it. An example video where this component is used alone is uploaded under path: [/src/only-differential.mp4]

### Describe how the final hyperparameters were chosen.

The parameters were chosen manually by try and error. First, make sure the car can drive straight with zero value for all parameters. Then add the proportional and the car start going on the road but it starts overshooting out of it. Then add the differential to try to overcome the overshooting. The integral part only moved the car out of the road; so, it stayed as zero. After the car drove the track without going out of it, the parameters increased to minimize the average cross-track error on a single track lap. The final parameters where [P: 1.5, I: 0.0, D: 2.5], which makes it a PD controller.

## Simulation

The vehicle must successfully drive a lap around the track.

A short video of final parameters set is uploaded in the project repo.[ src/final_params_enabled.mp4]