PID Controller

Overview

This project implements a PID controler in C++ and tested on Simulator provided by Udacity. The communication between the simulator and the application is done using WebSocket using the uWebSockets. This project utilizes the starter code given by Udacity. Output of the simulator is available as a video.

Thanks to https://medium.com/@cacheop/pid-control-for-self-driving-1128b42ab2dd. It helped me to write an iterative code to test multiple PID scenarios. Running each of these scenarios by editing the source file, compiling and testing was a huge task. This helped me to run all at one shot

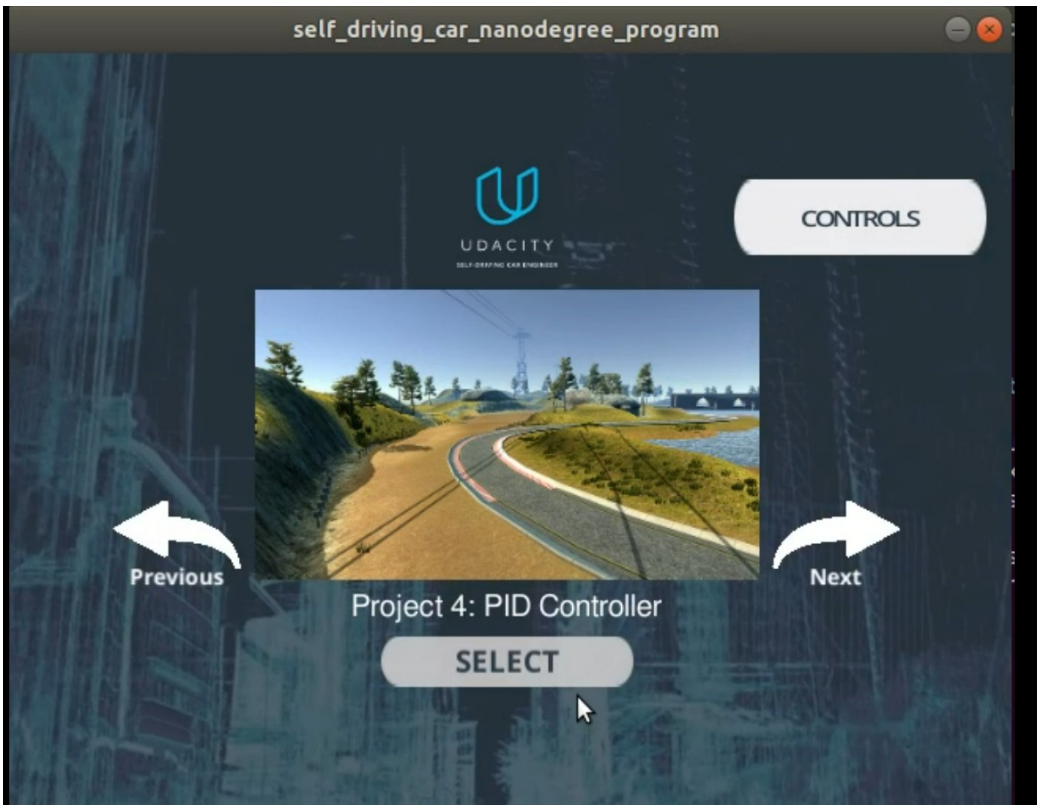
Environment:

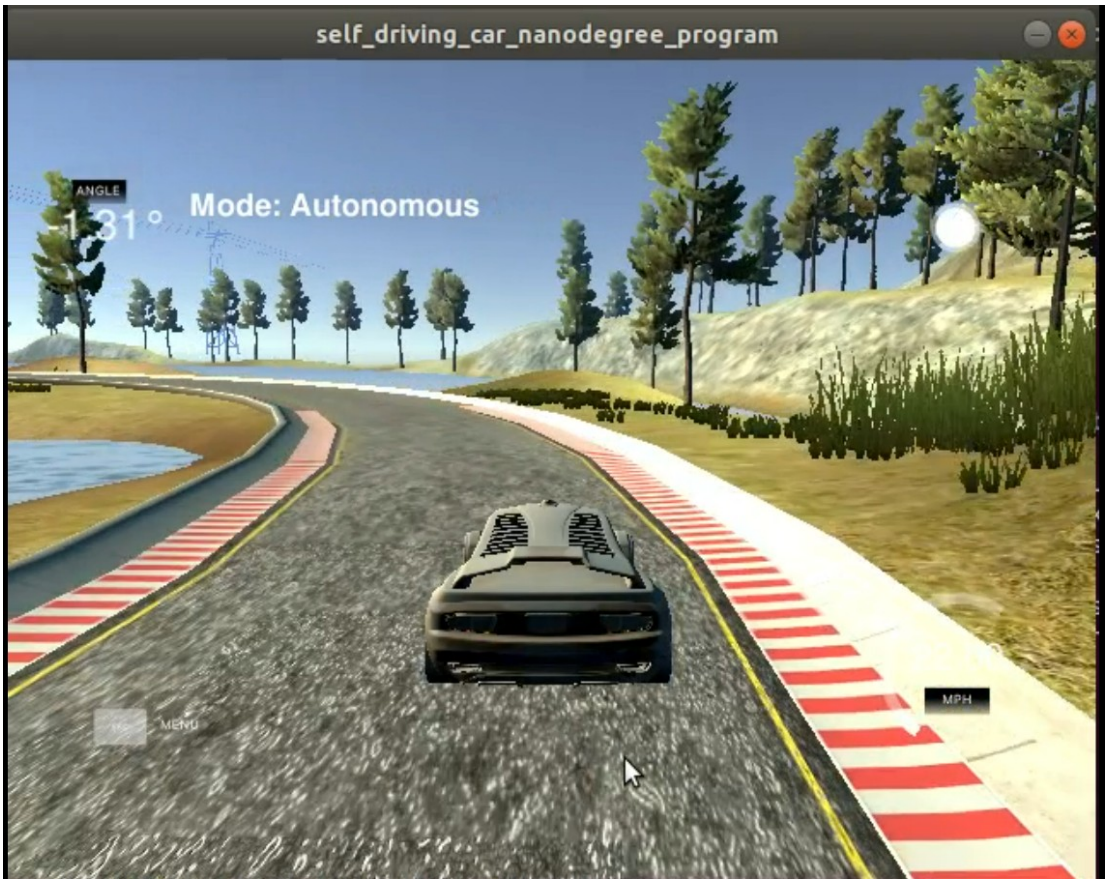
This project was done in Ubuntu 18 LTS and following are the components

- cmake >= 3.5
- make >= 4.1
- gcc/g++ >= 5.4

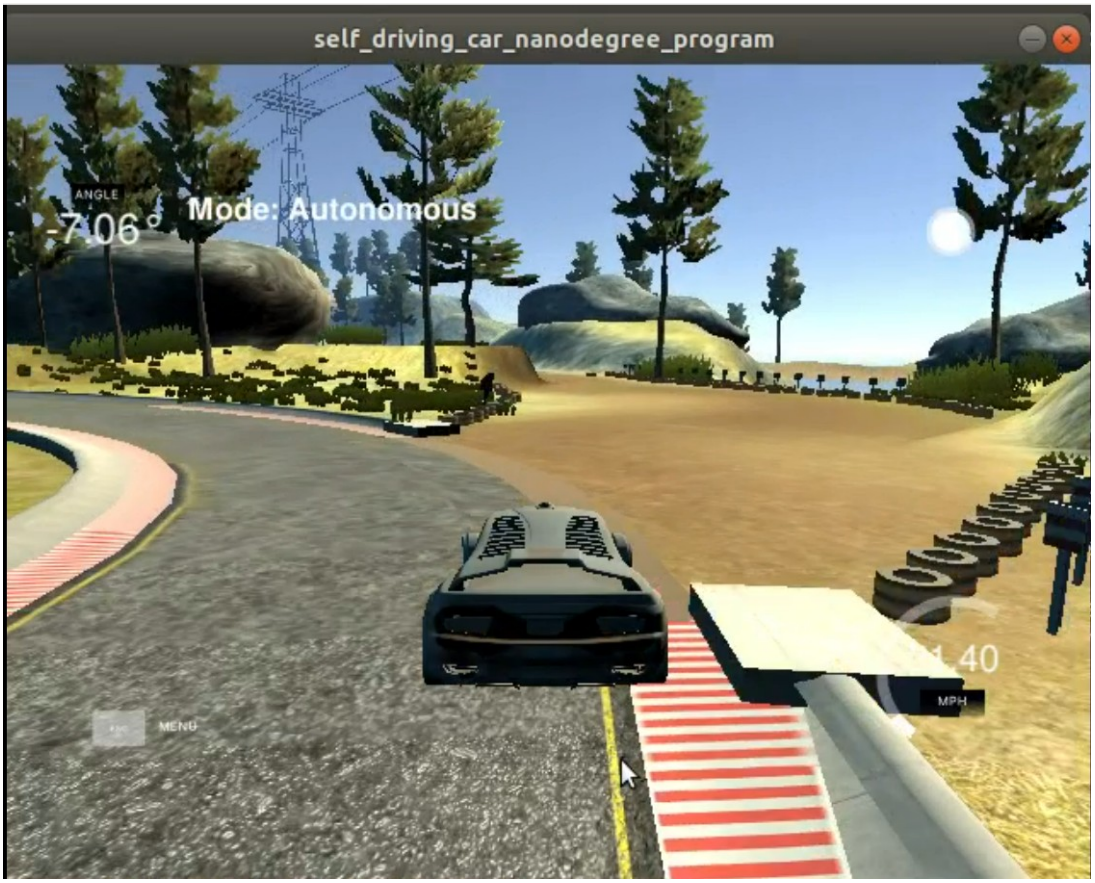
Ouput:

This project is the output from the Simulator . Before the simulator is started:

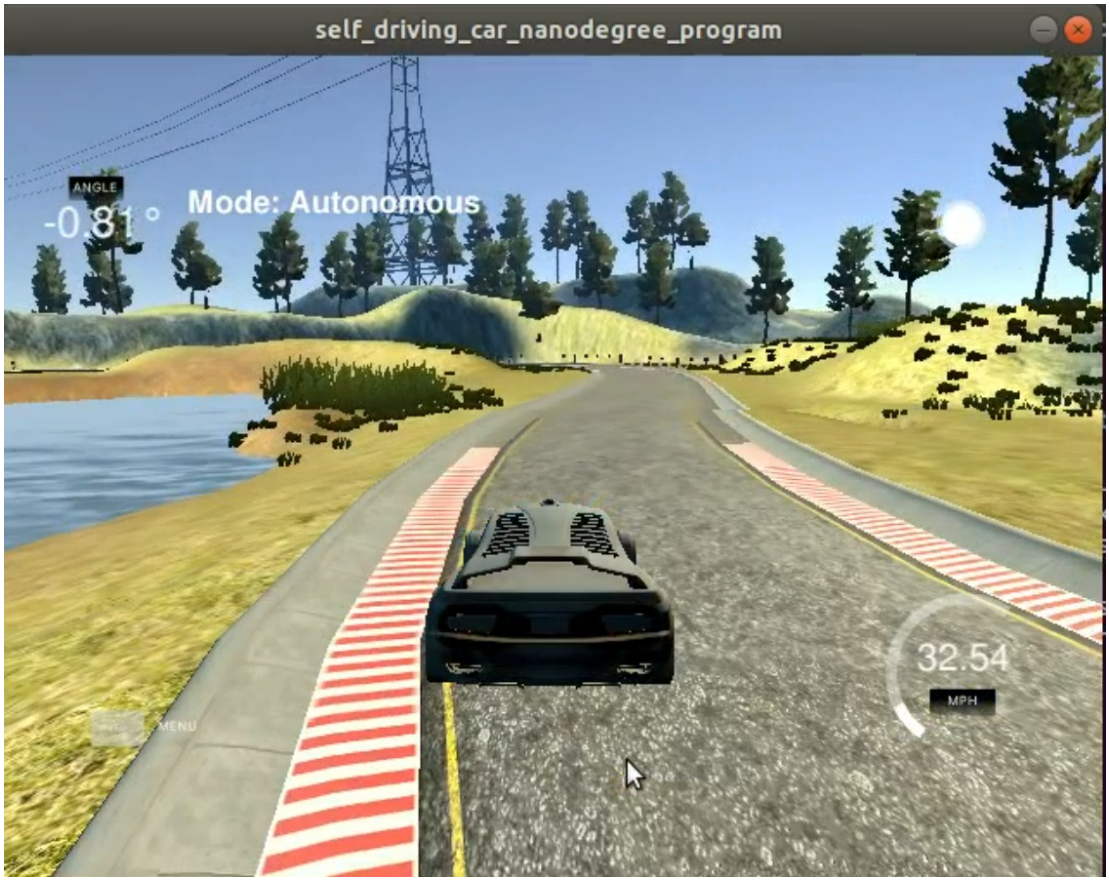




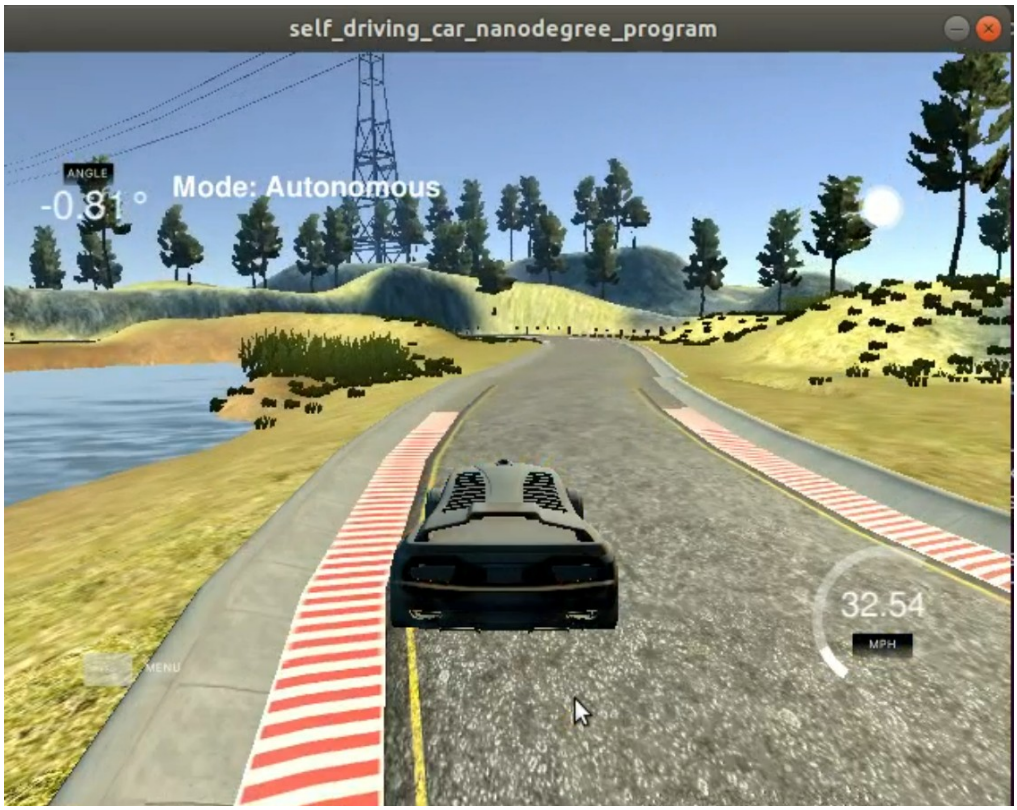
Car on the first bend

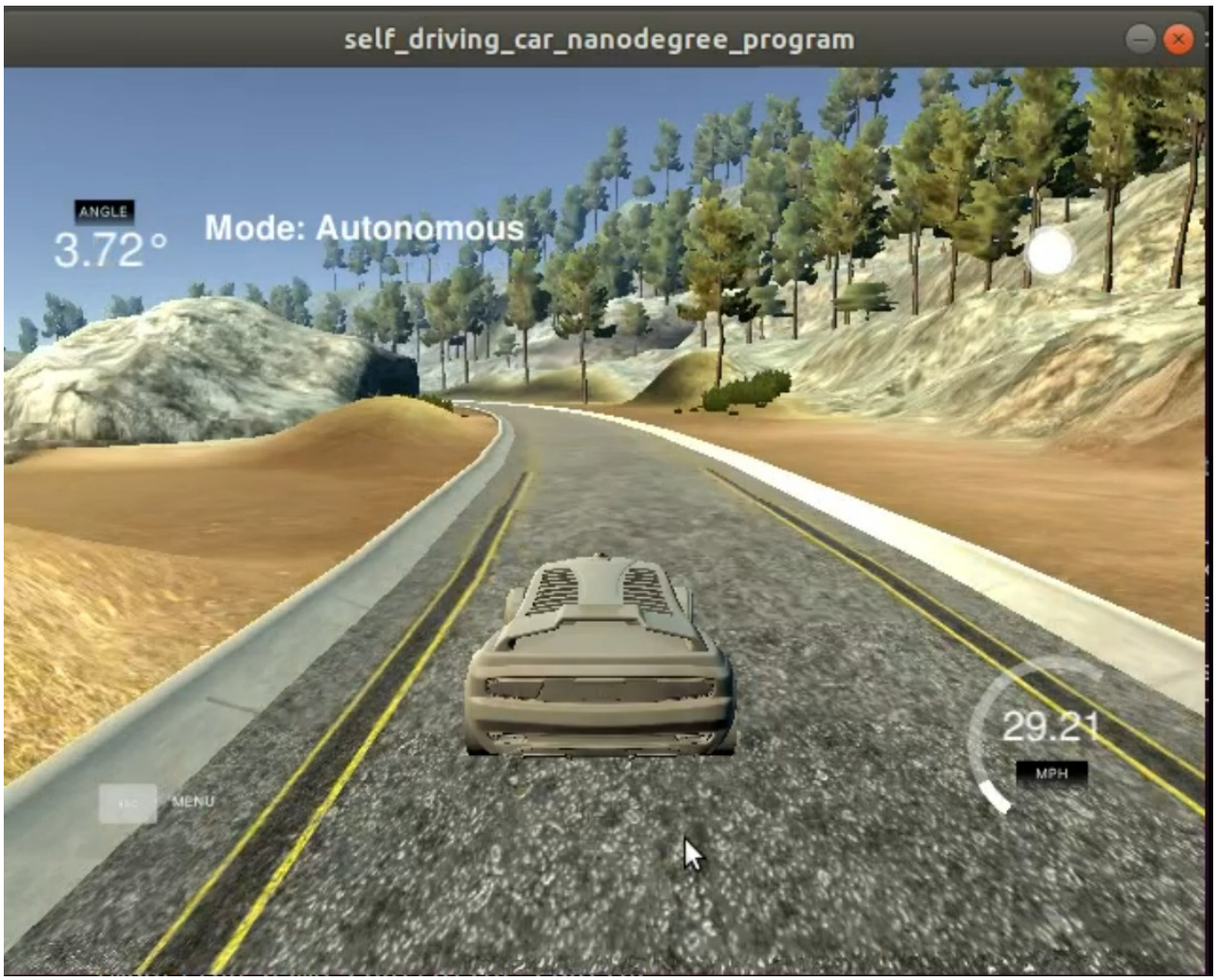

self_driving_car_nanodegree_program
ANGLE
-7.06°
Mode: Autonomous
MPH
MENU

STEEP CURVE



car negotiating on a steep curve





COMPLETING ONE LAP

Algorithm:

The code uses PID Algorithm to control both speed and Steering.

KP - Component – Porportional component – Controls the process with a control component in proportion to the error:

KI – Component – Integral – Sums the error and controls the process against any drift over period of time. Not used in this .

KD- Component – Differential – Calculates the in error over a period and controls the process in proportion to the change in error

Error – CTE, Cross track error is used for steering control and difference between targeted speed and actual speed is used for throttling control

RUBRICS:

This project satisfies all the rubric points

Compiling

Criteria	Meets Specifications
Your code should compile.	Code must compile without errors with `cmake` and `make`.

The code compiles properly without any error.

Criteria	Meets Specifications
The PID procedure follows what was taught in the lessons.	It's encouraged to be creative, particularly around hyperparameter tuning/optimization. However, the base algorithm should follow what's presented in the lessons.

PID Algorithm is used in file PID CPP. Update Error function calculates the individual components and total error calculates the final control value. Apart from these, i have created the below customize functions:

1. Manual override – Implemented in Pid.cpp from line no 71 to 92. This checks for the CTE. If cte > 1.75 that is car wheers off towards the end of road, the algorithm manually pushes the car into middle of the road. OVR_ARRAY contains the steering value. To avoid sudden disturbance to PID algorithm, the function executes three times and ensures that car turns smoothly. Once the car comes to center, the PID override is switched off

2. Still if car goes beyond the drivable track, then steering is turned sharp to bring car back into track. This is implemented in main.cpp from line no 119 to 122.

Reflection

Criteria	Meets Specifications
Describe the effect each of the P, I, D components had in your implementation.	Student describes the effect of the P, I, D component of the PID algorithm in their implementation. Is it what you expected?o.

P – Component corrects the error. Had to tune this as very low P made the car to wheer off the road in turns and was not sensitive to any errors and very high P made the car to oscillate

I – Component – corrects the drift. didn't use as there is no drift

D – Component. Reduce the oscillation that occurs because of p component.

Describe how the final hyperparameters were chosen.	Student discusses how they chose the final hyperparameters (P, I, D coefficients). This could be have been done through manual tuning, twiddle, SGD, or something else, or a combination!

I started wtih a random P and tried to increase/decrease the P till car oscillates continuously without going off track. Then increased d value to ensure taht oscillations stops. Had to do multiple iterations and arrived at the below nos. Used twiddle to optimize the nos. (Twiddle is not part of the code)

```
/********working controls***************/
  // steering_pid.Init(0.0395, 0, 0.2); - video 1
  // steering_pid.Init(0.038, 0, 0.2);
  // steering_pid.Init(0.042, 0, 0.2);
  // steering_pid.Init(0.044, 0, 0.2)
  // steering_pid.Init(0.048, 0, 0.2);
  // steering_pid.Init(0.040, 0, 0.2);
  // steering_pid.Init(0.0399, 0, 0.2); - video 2

/*****************************************/
```

Simulation

Criteria	Meets Specifications
The vehicle must successfully drive a lap around the track.	No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle).

Car remains in the road. Sometimes, it goes off road but majority of the time it reamins and does not wheer off the road. Attached video with the output.