EE324: Experiment No.2
Line Follower

Batch No:3
Mudit Goyal 200070045
Mudavath Vishnuvardhan-200070044
Narne Avinash Chowdary-200070047

October 12, 2022

1 Overview of the experiment

1.1 Aim of the experiment

To design and implement a PID controller for the Spark V robot to make it follow a continuous track using the IR sensors provided on the robot.

1.2 Objective

To follow the given track and complete it within 30 seconds

2 Control Algorithm

2.1 PID Control Application

Output of PID controller is calculated in time domain as follows:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

The variable (e) represents the tracking error, the difference between the desired angle(r) and the current angle(y). This error signal (e) is fed to the PID controller, and the controller computes both the derivative and the integral

of this error signal with respect to time. The control signal (u) is fed to the pwm signal and is equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error.
$\mathrm{K}_p = proportionalgain$
$K_i = Integralgain$
$K_d = differentialgain$

2.2 Characteristics of P,I,D

- P-controller as term suggests follow the error and provides stability with good speed of response.This causes for a given level of error the closed-loop system to react more quickly, but can also increase the overshoot. Secondary effect of K_p is of changing the steady-state error.
- I-controller is mainly used to reduce the steady state error. By decreasing the integral gain Ki, the speed of the response is increased or in other words it can make the system more sluggish (and oscillatory) since when the error signal changes sign, it take a while for the integrator to unwind.
- D-controller increases the speed of the response because it anticipates the future behavior of the error. The more rapidly D-controller responds to the changes in the process variable, if the derivative term is large (K_d).

Response	Rise Time	Over Shoot	Settling Time	Steady state error
K_P	Decrease	Increase	-	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_D	-	Decrease	Decrease	-

Table 1: PID

3 Implementation

In this experiment, we got ourselves familiarized with Microchip Studio IDE and used it write a C++ code to complete our task. The three IR sensors

attached to the bot were first tuned to give the same value under similiar environment. The difference between the values from the lest and right sensors were used as the input to the PID. The output of the PID was used as the input to the velocity of the two motors. Using the relative velocity of the two motors we traced the track in almost 27 seconds.

3.1 Code

This code contains just the main function and the functions that we defined to accomplish the given task. Rest of the code was same as the sample code that was provided to us.

```
void vel(float val,float threshold, int offset, float pid)
{
  unsigned char velocity=pid;

  if(val>0)
  {
    if(val<threshold+offset){
      OCR1BL = 0.10*velocity;
      OCR1AL = velocity;
      soft_right();
    }

    else{
      OCR1BL = -velocity;
      OCR1AL = velocity;
      right();
    }
  }
  else
  {
    if(val>-(threshold+offset)){
      OCR1BL = abs(velocity);
      OCR1AL = 0.1*abs(velocity);
      soft_left();
    }
    else{
      OCR1BL = abs(velocity);
      OCR1AL = -abs(velocity);
      left();
    }
  }
}
```

```
33
34 int main()
35 {
36   init_devices();
37
38   lcd_set_4bit();
39   lcd_init();
40   unsigned char ls   =0;
41   unsigned char rs  =0;
42   unsigned char cs =0;
43   unsigned char ols =0;
44   unsigned char ors =0;
45   float th =0;
46   float Kp =10;
47   int Kd =135;
48   int rl =0;
49   int orl =0;
50   int po =128;
51   int so =22;
52
53   while(1)
54   {
55
56           ls = ADC_Conversion(3);
57     cs = ADC_Conversion(4);
58     rs = ADC_Conversion(5);
59
60     rl   = rs - ls;
61
62
63     float pid = Kp*rl+Kd*(rl - orl);
64                 if(pid>255){pid=255;}
65     if(rl>0)
66     th = (po - pid);
67     else
68     th = -(po + pid);
69     if(rl>-th && rl<th)
70     {
71       forward();
72     }
73     else if(rl<-th || rl>thresh)
74       vel(rl,th,so,pid);
75     orl=rl;
76   }
77 }
```

3.2 Explanation

- One PID controller has been used, whose input is (right reading - left reading).
- The function vel is where our velocity of the motors is controlled.
- Two different offsets have been defined.
- The first offset (po) decides the forward motion of the bot. The reasoning is that if the difference between right and left readings is in the range of (pid-po) then there is no need for the left and right motion and the bot should move forward. But if the difference is larger, then accordingly the bot will move left or right.
- The second offset is used to decide wether the motion should be smooth or hard. If the difference is large thus the turn radius should be small or in other words hard left/right and vice-versa.
- OCR1BL is to control the velocity of left motor while OCR1AL controls the right motor.

4 Challenges faced and their Solutions

- The first problem we encountered was that the readings of the sensors were not same. So, we had to use the potentiometers to bring all the values of the sensor same. But the range of the sensor values were very small and different, which made it almost impossible to use and a different bot had to be used.
- The next problem we faced was the values of the two offsets we had to use and how to use the PID to actually control the motion of the bot. So, from trial and error we approximated the values of the offsets so that it doesn't go out of bounds and also complete it in the given time.
- As we were not using the centre sensor's data, we got stuck with the box region but a little fine tuning of the Kp, Kd values and it worked perfectly fine.

- First our bot was moving very slow and shakily so we made Ki = 0 as it was interfering when many turns were introduced and we had to test for vaious combinations of Kp, Kd to get the right value.

5 Conclusion

- This experiment could be completed without using the centre sensor.
- Using two PIDs and hence using the centre sensor value makes the system more robust.
- The experiment could also be completed without the use of any PID and could simply be done by hard coding it.