EE324: Experiment 2
PID Line Follower Robot

Akhilesh Chauhan 21d070010
Uvesh Mohammad 21d070084
Parth Arora 21d070047
Batch-16

October 10, 2023

Contents

1 Overview of the experiment

1.1 Aim

To design a Line Follower Robot using PID controller.

1.2 Objective

Bot should complete the path within 30 seconds.

1.3 Track

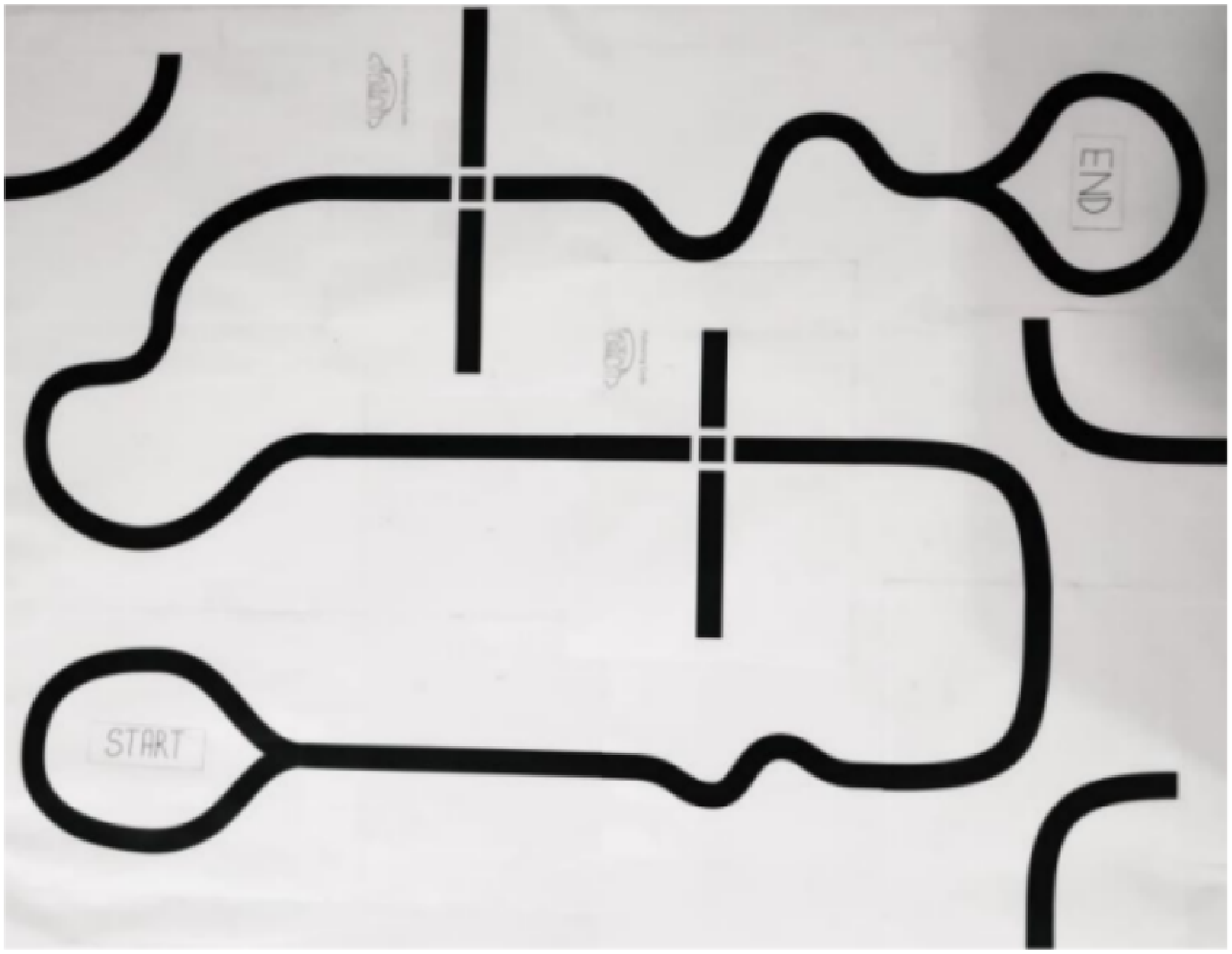

Figure 1: Design of the track to be traced

2 Design

2.1 PID controller

The use of a PID controller can be justified by the following :

- **Proportional Controller:**It is responsible for driving the output according to the present error. The present error is the difference between the current position and the set point which gets multiplies by a proportional gain $\boldsymbol{k_p}$. Though, it drives the system towards the required output,there always exists some steady state error upon using a proportional controller alone.It basically gives an idea of how sensitive you want the controller to be.

- **Integral Controller:**The integral factor integrates the error term until it reaches zero;thereby reducing the steady state error to 0.However, the response speed is very slow. The output is obtained by integrating the error and multiplying it by a constant $\boldsymbol{k_i}$ also called as the integral gain.The controller pretty much adds up the errors from each of the previous loops from the beginning.

- **Differential Controller:**It observes the rate at which the system variables are changing and generates the output in correspondance to the rate of change. The output is obtained by multiplying the error's rate of change with a constant $\boldsymbol{k_d}$ also called the derivative gain.It tries to anticipate the relative correction at a future instance.

2.2 Control Algorithm

- For the accuracy of the Line Follower Robot, we used PID controller.

- We know, a PID controller's output is a funtion of error $e(t)$ which is difference between **final expected value** and **current value of the variable**.

- The Output of the PID Controller can be written as :

$$\boxed{\boldsymbol{control = k_p * e(t) + k_i * \int (e(t)dt) + k_d * \frac{de(t)}{dt}}}$$

- We have used name control as PID controller's output.

- The PID output is then used to determine the direction in which bot moves.

- We simply calculated the output of the PID Controller and used that to determine the speed of the left and right motors to take proper turn.

- We have set two thresholds : Hard threshold and Soft threshold. If the left error is positive then the bot moves left and else if the right error is positive the bot moves right as shown in the code. If the error is greater that hard threshold then it will move left, else if it is greater than soft threshold, it will move softleft.Similar will happen for right.

2.3 Programming

Code for main function is given below:

```
   //Main Function
int main(void)
{
 init_devices();

 lcd_set_4bit();
 lcd_init();

  double kp = 2.0;
  double kd = 0.01;
  double ki = 0.0005;

  double pre_err;
double err ;
double int_err=0;
double err_der;
double control;
int count = 0;

while(1)
{
l=ADC_Conversion(3);
c=ADC_Conversion(4);
r=ADC_Conversion(5);

lcd_print(1, 1, l, 3);
lcd_print(1, 5, c, 3);
lcd_print(1,9 , r, 3);

//_delay_ms(5);

err = r-l;
err_der = err-pre_err;
int_err += err;
control = kp*err+kd*err_der + ki*int_err;
pre_err = err;
if(c>=9 && r<=7 && l<=7){
forward();
_delay_ms(100);
}

//edit
if(c>7 && l>7 && r>7){
forward();
```

```
_delay_ms(100);
}
//edit over

int k;
if(control>255){
k = 255;
}
else if(control<-255){
k = 255;
}
else{
k = abs(control);
}
//edit
if(c>7 && l>7 && r>7){
forward();
_delay_ms(1000);
}
//edit over

if(control > 3){
if(control<80*kp){
right();
}
else if(control<180*kp && control>80*kp){
velocity(k, k);
soft_right();
}
else{
velocity(k, k);
right();
}
}

else if(control< -3){
if(control>=-80*kp){
velocity(k, k);
left();
}
else if(control<-80*kp && control>-180*kp){
velocity(k, k);
soft_left();
}
else{
velocity(k, k);
left();
}
}
else{
velocity(255, 255);
forward();
int_err  = 0;
}
if(l>=8 && r<=8 && c<7 && count == 0){
left();
count = 1;
}
```

3 Experiment

3.1 Observation and Results

- This system has the following gain constants:

$$\boldsymbol{k_p = 2.0}$$
$$\boldsymbol{k_d = 0.01}$$
$$\boldsymbol{k_i = 0.0005}$$

- We kept varying the values of k_p , k_i and k_d until the bot completed the track within 30 seconds.
- Our bot completed the track in **22 seconds.**
- We noticed that the speed is mainly dependent on k_p.
- We noticed that the smoothness is mainly dependent on k_d.
- We kept the values of k_d and k_i small as compaered to k_p.

3.2 Challenges Faced and their Solutions

- Sometime the bot was faulty, so even after applying simple and couple of if loops it didn't work properly.
- One day, the bot was not properly measuring the values by IR sensors.
- Speed of the bots was also influenced by the state of charge of the battery.
- On some of the bots, the connector wire to one of the motors was loose, so it took time to figure out why one of the wheels was not rotating.