EE324 : Controls Lab
Experiment - 2 : PID Line Follower

Batch 4

Ankith R	200070006
Anmol Saraf	200070007
Anubhav Bhatla	200070008

October 21, 2022

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 for this purpose.

2 Objectives

To trace the track given under 30 seconds.

3 Theory

We are required to implement a PID velocity controller to set the velocity for the left and right motors based on our requirement. The control signal output of a PID controller is given as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{1}$$

where $e(t) = r - l$ is the error between the right and left IR sensor values, and K_p, K_i, K_d are the respective PID constants.

4 Implementation

- For our implementation, we define the error as $l - r$ i.e., the value read from the left sensor minus the value from the right IR sensor value. This error is then fed into the control algorithm, i.e., PID controller with $K_p = 0.7$, $K_d = 2.5$, $K_i = 0$. We decided to not add an integral term as there is no use of minimizing the steady state error in this case. This control is capped to $\pm MAX$, which is 200 in our case.

- In order to control the movement of the line follower bot, we used velocity control, for which we defined the `velocity` function which takes in two unsigned char inputs. The first input sets the speed of the left motor and the second inputs sets the speed for the right motor.

- For the case when the center input (c) is high (> 40), we go straight irrespective of the other two readings. This is done in order to avoid turning at the + junctions and the bot goes straight.

- For the case when our control value is in the range [-thres1, thres1], the control value is not very high which means that the error must not have been very high and thus there is no need to turn, so we keep going straight with MAX speed.

- When our control value is $>$ thres1, error must have been high in the positive direction, i.e., $r \gg l$, and we need to turn right. Thus we use the `velocity` function to assign speed to the left wheel equal to the control value and zero speed to the right wheel. We also specify the bot that it needs to move in the right direction, i.e., left wheel moves forward and right wheel moves backwards.

- Alternatively, when our control value is $<$ thres1, error must have been high in the negative direction, i.e., $l \gg r$, and we need to turn left. Thus we use the `velocity` function to assign speed to the right wheel equal to the control value and zero speed to the left wheel. We also specify the bot that it needs to move in the left direction, i.e., left wheel moves backwards and right wheel moves forward.

- As a default case, we go forward with MAX speeds for both wheels.

5 Code

The code for the PID controller (added on top of the given sample code) for the Sparks V robot to trace the track is as follows:

```
void velocity (unsigned char left_motor, unsigned char right_motor){
        OCR1AL = left_motor;
        OCR1BL = right_motor;
}

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

        lcd_set_4bit();
        lcd_init();

        double error, prev_error = 0;
        double error_sum = 0;
        double control;

        double Kp = 0.7;
        double Kd = 2.5;
        double Ki = 0;

        double thres1 = 30;

        unsigned char leftmotor, rightmotor;
        unsigned char MAX = 200;

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

                error = r-l;
                error_sum += error;

                control = Kp * error + Kd * (error - prev_error) +
                Ki * (error_sum);

                if (control > MAX){
                        control = MAX;
                }
                else if (control < -MAX){
```

```
                control = -MAX;
            }

            prev_error = error;

            if (c > 40){
                velocity(MAX, MAX);
                forward();
            }
            else if (control < thres1 && control > -thres1){
                velocity(MAX, MAX);
                forward();
            }
            else if (control > thres1){
                velocity(control, 0);
                right();
            }
            else if (control < -thres1){
                velocity(0, -control);
                left();
            }
            else{
                velocity(MAX, MAX);
                forward();
            }
        }
}
```

6 Challenges Faced

- Initially, we had no dependence on the center IR readings and all operation was controlled by the right and left sensor values. But this approach failed for the + junctions as the bot was turning instead of going straight ahead. In order to fix this we added an initial threshold on the center sensor value, asking the bot to go forward at MAX speed when the center sensor value exceeds this threshold.

- We also had trouble in tuning the sensor values on our bot. The first bot that we were testing our final code had a very small IR reading for the center sensor compared to the other two sensors, because of which the bot was turning at the + junctions. Replacing the bot with one

that had very similar and high values for all three IR sensors fixed this problem and the bot was able to go straight at these junctions.

- One major problem we were facing was that our bot was not able to turn left and was only turning right even though the code suggested a left turn. After a lot of debugging and helpful suggestions from our TA, we realised that we had declared the `error` variable as an `unsigned char` type because of which it could never become negative, which was the condition for a left turn. Changing the type to `double` fixed the problem and the bot was now able to make both left and right turns.

- We were initially assuming that only setting the velocity using the defined `velocity` function would be enough to make the appropriate turn, but we later realised that we also need to set the direction of movement for the two wheels which is done by calling the `left()` and the `right()` functions as per our requirement.

7 Results and Conclusion

Given below are values of the PID constants we used:

K_p	0.7
K_i	2.5
K_d	0

Using our implementation, we were able to trace the given track in around **22 seconds**.
To conclude, we were able to successfully design PID control for the line follower bot which satisfies the required design specification of completion under 30 seconds.